



## CAN 300 PRO

### CAN Communication Module for S7-300 CANopen® Master, CAN Layer 2 or SAE J1939

700-600-CAN12

## Manual

Edition 9 / 02.03.2015 with HW30 & FW1.32 and higher

Manual order number: 900-600-CAN12/en

All rights are reserved, including those of translation, reprinting, and reproduction of this manual, or parts thereof. No part of this manual may be reproduced, processed, copied, or transmitted in any way whatsoever (photocopy, microfilm, or other method) without the express written permission of Systeme Helmholtz GmbH, not even for use as training material, or using electronic systems. All rights reserved in the case of a patent grant or registration of a utility model or design.

Copyright © 2014 by

**Systeme Helmholtz GmbH**

Hannberger Weg 2, 91091 Grossenseebach, Germany

**Note:**

We have checked the content of this manual for conformity with the hardware and software described. Nevertheless, because deviations cannot be ruled out, we cannot accept any liability for complete conformity. The information in this manual is regularly updated. When using purchased products, please heed the latest version of the manual, which can be viewed in the Internet at [www.helmholz.com](http://www.helmholz.com), from where it can also be downloaded.

Our customers are important to us. We are always glad to receive suggestions for improvement and ideas.

**Revision history of this document:**

<b>Edition</b>	<b>Date</b>	<b>Revision</b>
1	11.03.2009	1st version
2	13.03.2009	Minor corrections
3	02.07.2009	CANopen <sup>®</sup> master Power LED behavior; correction of call parameters of the FCs CANopen <sup>®</sup> Tools of CANParam v4.10 added
4	15.04.2010	Adjustments for CANParam 4.20 and firmware 1.20 Extended temperature range stated (up to -25°C)
5	30.06.2010	Notice to alarm-OBs
6	05.08.2011	Changed behavior of BF-LED
7	22.12.2011	Error Codes completed, new FB generation for all handling blocks completed and further small corrections
8	02.07.2012	Handling block overview and the usage of them
9	24.02.2015	New handling block added, Windows 7 adaptations

# Contents

<b>1</b>	<b>Safety Information</b>	<b>7</b>
1.1	General	7
1.2	Restriction of access	8
1.3	Information for the user	8
1.4	Use as intended	8
1.5	Avoiding use not as intended!	8
<b>2</b>	<b>Installation and Mounting</b>	<b>9</b>
2.1	Vertical and horizontal mounting	9
2.2	Minimum clearance	10
2.3	Mounting of the module on the DIN rail	10
<b>3</b>	<b>System Overview</b>	<b>12</b>
3.1	CAN bus	12
3.2	CAN cabling	12
3.3	Application and function description	13
3.4	Connections	13
3.5	LED displays	14
3.6	DIP switch	15
3.7	Project memory card MMC	15
3.8	Items supplied	15
3.9	Accessories	15
<b>4</b>	<b>Configuration in the PLC</b>	<b>16</b>
<b>5</b>	<b>Process image in the PLC</b>	<b>18</b>
5.1	Byte 0: Module status	18
5.2	Byte 1: Error status (EFLG) of the CAN controller	19
5.3	Byte 2: FIFO status bits	19
5.4	Byte 3/4: CAN controller Tx/Rx error counter	19
5.5	Byte 5: CANopen <sup>®</sup> master status	20
5.6	Byte 6: Assignment SDO requests (CANopen <sup>®</sup> master)	20
5.7	Byte 7: Nodes in Operational (CANopen <sup>®</sup> master)	20
5.8	Byte 8: Active node ID	20
<b>6</b>	<b>Configuration of the module</b>	<b>21</b>
6.1	Overview	21
6.2	Installation of the USB interface driver	21

6.3	Creating a layer 2 project	26
6.3.1	Setting the CAN bus baud rate	27
6.3.2	Setting the protocol type	27
6.3.3	PLC transfer mode	27
6.3.4	Mask filter	28
6.3.5	Bit filter	28
6.3.6	Event scripts	29
6.3.7	Timer	30
6.3.8	Synchro window	31
6.4	Creation of a CANopen® project	32
6.4.1	Settings of the master	32
6.4.2	Creating a slave	34
6.4.3	Setting TPDOs	35
6.4.4	Setting RPDOs	37
6.4.5	PLC I/O buffer principle	38
6.4.6	Adding SDOs	40
6.5	Uploading	41
6.6	Downloading	41
6.7	Diagnostics/debugging	41
6.7.1	Layer 2 debug display	41
6.7.2	CANopen® Debug display	43
6.8	CANopen® Tools	45
6.8.1	Scan slaves	45
6.8.2	SDO transmission	46
6.8.3	Slave mapping	47
6.9	Slave PDO distribution in the PLC I/O buffer	48
<b>7</b>	<b>Programming in the PLC</b>	<b>49</b>
7.1	Overview	49
7.2	The right handling blocks	49
7.3	Layer 2 handling blocks	50
7.3.1	General	50
7.3.2	FB 65 CANSEND	51
7.3.3	FB 66 CANRCV	52
7.3.4	FB 67 CANCTRL	53
7.3.5	FB 68 CANSEND Block Mode	54
7.3.6	FB 69 CANRCV Block Mode	55
7.3.7	FB 64 CANSendTimer	56

7.3.8	Parameter STAT	57
7.4	CANopen®	58
7.4.1	General	58
7.4.2	Objects	58
7.4.3	Functions	59
7.4.4	Network management	60
7.5	Start-up behavior of the CANopen® master	62
7.6	Operating conditions for CANopen® slave devices	63
7.7	Tips on start-up / troubleshooting	64
7.8	CANopen® data handling modules	65
7.8.1	FB 20 CANopen® IO Read	66
7.8.2	FB 21 CANopen® IO Write	67
7.8.3	FB 22 CANopen® Service	68
7.8.4	FB 23 CANopen® Netzmanagement	70
7.8.5	FB 24 CANopen® SDO	71
7.8.6	FB 27 CANopen® SDO Segmented	73
7.8.7	FB 25 CANopen® L2 Receive	75
7.8.8	FB 26 CANopen® PDO resend	76
7.8.9	FB 28 CANopen® SYNC trigger	77
7.8.10	Using of alarm-OBs	77
7.9	SDO abort codes	78
7.10	SAE J1939 communication	79
7.10.1	General	79
7.10.2	FB 70 CANSEND_SAE_J1939	80
7.10.3	FB 71 CANRCV_SAE_J1939	81
<b>8</b>	<b>Return parameter RETVAL</b>	<b>82</b>
<b>9</b>	<b>Upgrading from CAN 300 to CAN 300 PRO</b>	<b>83</b>
9.1	Differences between CAN 300 and CAN 300 PRO	83
9.2	Layer 2	83
9.3	SAE J1939	83
9.4	CANopen®	84
<b>10</b>	<b>Appendix</b>	<b>85</b>
10.1	Technical data	85
10.2	Pin assignment	86
10.3	Further documentation	86

# 1 Safety Information

Please observe the safety information given for your own and other people's safety. The safety information indicates possible hazards and provides information about how you can avoid hazardous situations.

The following symbols are used in this manual.



*Caution, indicates hazards and sources of error*



*Gives information*



*Hazard, general or specific*



*Danger of electric shock*

## 1.1 General

The CAN 300 PRO module is only used as part of a complete system.



*The operator of a machine system is responsible for observing all safety and accident prevention regulations applicable to the application in question.*



*During configuration, safety and accident prevention rules specific to the application must be observed.*



*Emergency OFF facilities according to EN 60204 / IEC 204 must remain active in all modes of the machine system. The system must not enter an undefined restart.*



*Faults occurring in the machine system that can cause damage to property or injury to persons must be prevented by additional external equipment. Such equipment must also ensure entry into a safe state in the event of a fault. Such equipment includes electromechanical safety buttons, mechanical interlocks, etc. (see EN 954-1, risk assessment).*



*Never execute or initiate safety-related functions using the operator terminal.*



*Only authorized persons  
must have access to the  
modules!*

## **1.2 Restriction of access**

The modules are open equipment and must only be installed in electrical equipment rooms, cabinets, or housings. Access to the electrical equipment rooms, barriers, or housings must only be possible using a tool or key and only permitted to personnel having received instruction or authorization. See also Section 2.

## **1.3 Information for the user**

This manual is addressed to anyone wishing to configure or install the CAN 300 PRO module.

It is intended for use as a programming manual and reference work by the configuring engineer. It provides the installing technician with all the necessary data.

The CAN 300 PRO module is exclusively for use in a S7-300 programmable controller from Siemens. For that reason, the configuring engineer, user, and installing technician must observe the standards, safety and accident prevention rules applicable in the particular application. The operator of the automation system is responsible for observing these rules.

## **1.4 Use as intended**

The CAN 300 module must only be used as a communication system as described in the manual.

## **1.5 Avoiding use not as intended!**

Safety-related functions must not be controlled using the CAN 300 PRO module alone.



## 2 Installation and Mounting

The CAN 300 PRO module must be installed according to VDE 0100 IEC 364. Because it is an “OPEN type” module, you must install it in a (switching) cabinet. Ambient temperature: -25 °C – 60 °C.



*Before you start installation work, all system components must be disconnected from their power source.*



*Danger of electric shock!*



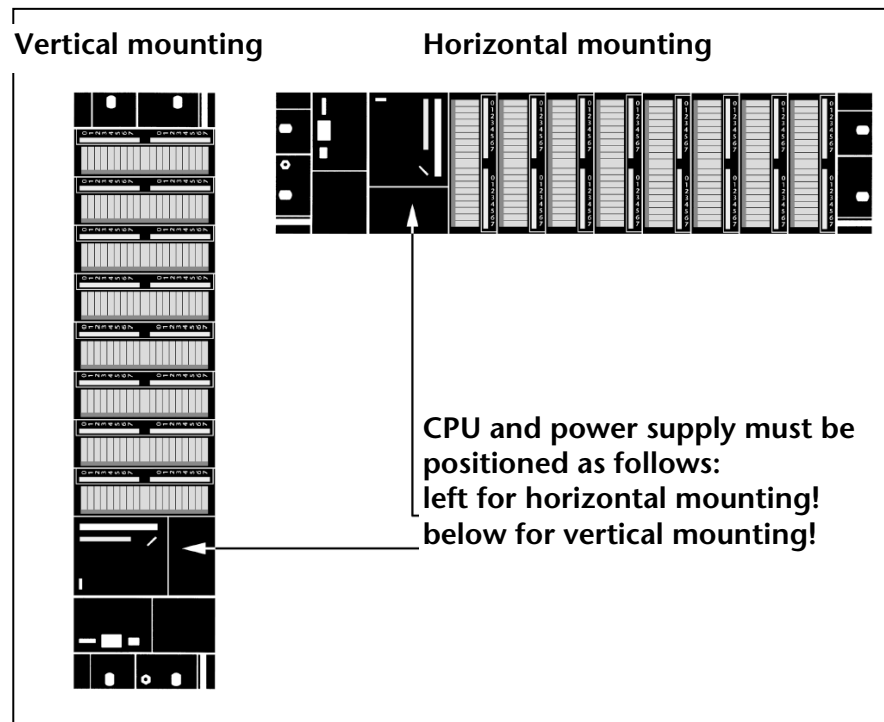
*During installation, application-specific safety and accident prevention rules must be observed.*

### 2.1 Vertical and horizontal mounting

The modules can be mounted either vertically or horizontally.

*Permissible ambient temperature:*

- for vertical mounting: from -25 to 40 °C
- for horizontal mounting: from -25 to 60 °C



## 2.2 Minimum clearance

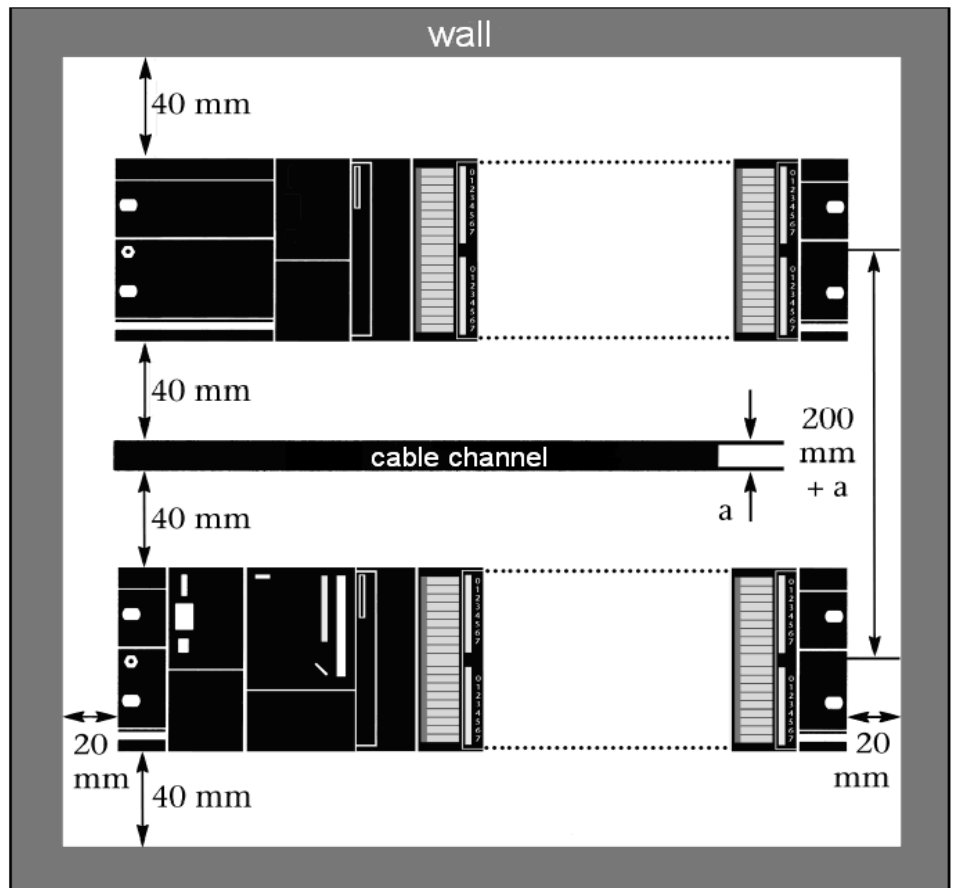
Minimum clearances must be observed because

- it ensures cooling of the CAN 300 PRO modules
- it provides space to insert and remove modules
- it provides space to route cables
- it increases the mounting height of the module rack to 185 mm, although the minimum spacing of 40 mm must still be observed

The following diagram shows the minimum spacing between the module racks and between these and any adjacent cabinet walls, equipment, cable ducts, etc. for S7-300s mounted in several module racks.



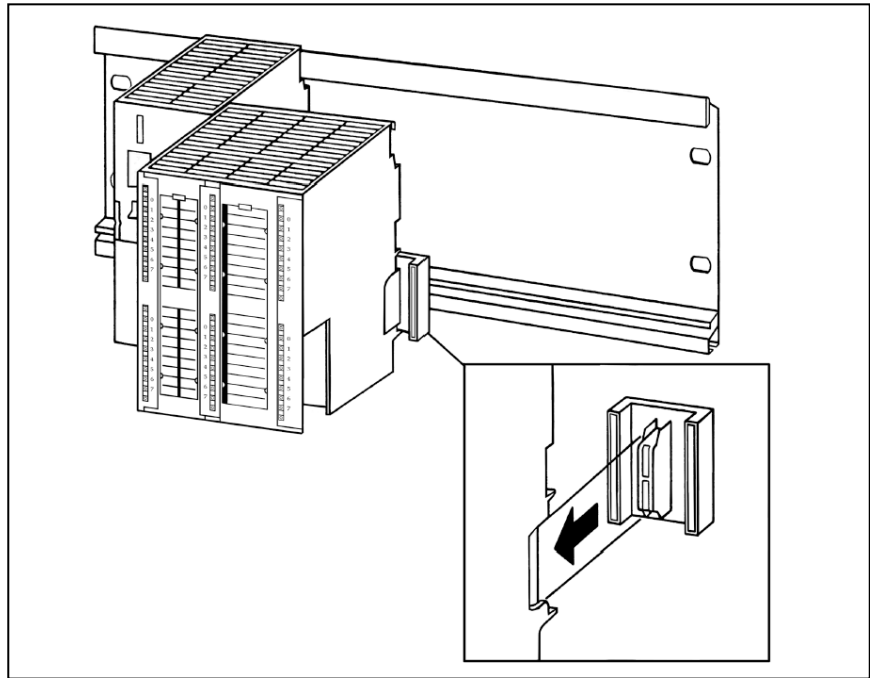
*Non-observance of the minimum distances can destroy the module at high ambient temperatures!*



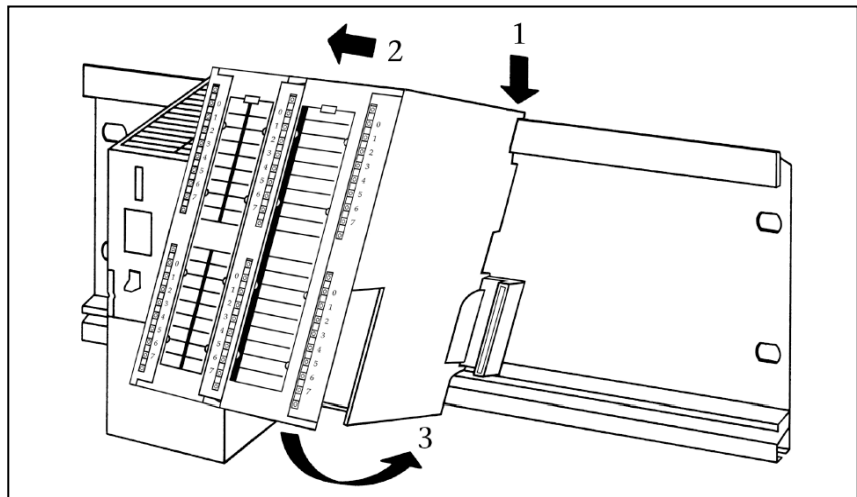
## 2.3 Mounting of the module on the DIN rail

A bus connector is included with each signal module but not with the PLC. When connecting the bus connector, always start with the PLC.

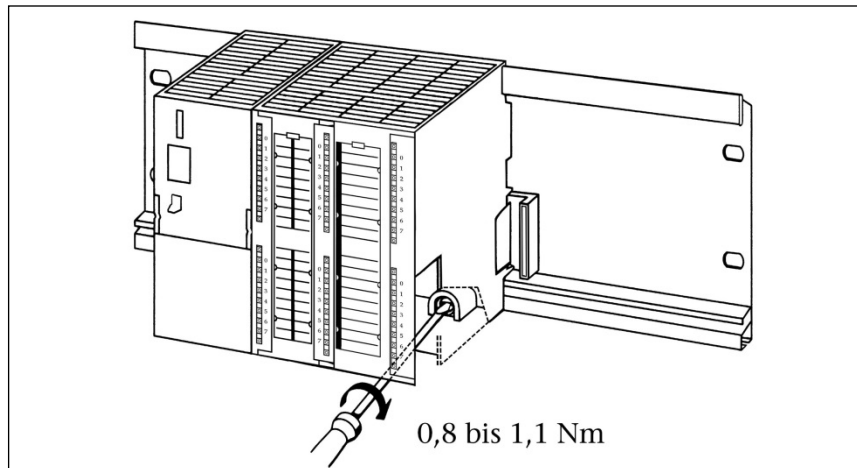
Take the bus connector off the last module and insert it into the PLC. Do not plug a bus connector into the last module of the tier.



Hook on the modules (1), slide them up to the left module (2), and click them downward (3).



Screw the modules on with a torque of 0.8 to 1.1 Nm.





## 3 System Overview

### 3.1 CAN bus

The CAN bus (Control Area Network) is an international and open field bus standard for applications in building, production, and process automation.

The comprehensive error detection measures make the CAN bus currently the most reliable bus system with a residual error probability of less than  $4.7 \times 10^{-11}$ .

### 3.2 CAN cabling

A CAN bus cable requires at least 3 conductors: CAN High, CAN Low, and CAN Ground. Only a bus topology is permitted. At both ends of the CAN bus cable, a terminating resistor of 120 ohms must be connected between CAN High and CAN Low. The CAN 300 PRO module does not have an integrated terminating resistor.



*No terminating resistor is integrated into the CAN 300 PRO module.*

The maximum cable lengths primarily depend on the baud rate used.

Bit rate	Bus length	Bit time
1 Mbps	30 m	1 µsec.
800 kbit/s	50 m	1.25 µsec.
500 kbit/s	100 m	2 µsec.
250 kbit/s	250 m	4 µsec.
125 kbit/s	500 m	8 µsec.
20 kbit/s	2500 m	50 µsec.
10 kbit/s	5000 m	100 µsec.

The cable lengths stated are for guidance only. The maximum cable length also depends on the number of connected stations and the type of cable.

More precise information is available in the document "CANopen® Recommendation DR 303-1" that is provided on the CD of the software package.

Check for correct cabling in the debug dialog box of the CANParam (see also Section 6.7).

**CAN**open



*For initial start-up as the CANopen® master, please also use the document "Start-Up Guide CANopen® for CAN 300 PRO."*

### 3.3 Application and function description

The CAN 300 PRO module from System Helmholtz GmbH allows you to connect any CAN stations to the programmable controller. The module is plugged into the backplane bus of the programmable controller. It can be used both in the central controller and in the expansion rack (e.g. with the IM360, 361, 365). Use of the CAN 300 PRO is also possible in an ET200M (with IM153), but this drastically reduces the performance.

The CAN 300 PRO module must be parameterized as a communication module in the Hardware Configurator and takes up 16 bytes in the analog process image. Data is exchanged with the PLC via the backplane bus.

Data handling blocks that enable simple handling of CAN communication are contained in the separately available software package. Data handling blocks are available both for simple layer 2 communication, SAE J1939 communication, for CANopen® master communication, and for CANopen® slave communication.

The scope of supply of the software package also includes the Windows parameterization tool "CANParam V4" for setting the CAN bus communication parameters and for creating CANopen® master projects.

The CAN 300 PRO module supports both CAN 2.0A (11 bits) and CAN 2.0B (29 bits) frames as a high-speed node according to ISO 11898-2 with a freely selectable baud rate of 10Kbps to 1Mbps.

The CAN 300 PRO module contains the network management functions "Power on," "Stop>>Run," and "Run>>Stop." Behind each of the three functions, it is possible to use a simple macro language to configure a CAN bus response with up to 100 frames per script that is executed automatically by the module when the event occurs.

In a multi-level acceptance mask it is possible to prefilter the frame IDs relevant to the programmable controller. Only those CAN frames are accepted that are required, which off-loads the cycle of the programmable controller.

16 freely settable timers are available in the CAN 300 PRO module. Each timer can trigger a freely programmable CAN frame. That way, it is easy to implement the synchronous protocols in common use in drive and servo systems using the CAN 300 PRO module.

It is also possible to have the data sent via the CAN bus only in a time window. The data to be transmitted are transferred non-cyclically by the programmable controller and transmitted from the CAN 300 PRO module after the parameterized time has elapsed.

### 3.4 Connections

The CAN 300 PRO module features a 9-way SubD connector behind the hinged front cover for the CAN bus and a USB connector for configuration and diagnostics.



A 24V power supply is not applied to the CAN bus connector.

### Pin assignment:

Pin	SUBD connector CAN
1	-
2	CAN Low
3	CAN GND
4	-
5	-
6	-
7	CAN High
8	-
9	-

### 3.5 LED displays

The LEDs on the front of the module inform you about its operating state.

#### LED "SF" (orange):

System error: shows a project with an error or a memory card that has an error or that is too small.

#### LED "Bus Error" (red):

This LED indicates a CAN error. A CAN error has occurred if the CAN status "EFLG" (Section 5.2) is not "OK," or a CAN FIFO overflow has occurred. You can obtain further information in debug mode of the CANParam software (see also Section 6.7).

#### LED "RX" (green):

CAN bus reception active: Indicates correct reception of a CAN frame.

#### LED "TX" (orange):

CAN bus transmission active: Indicates correct transmission of a CAN frame.

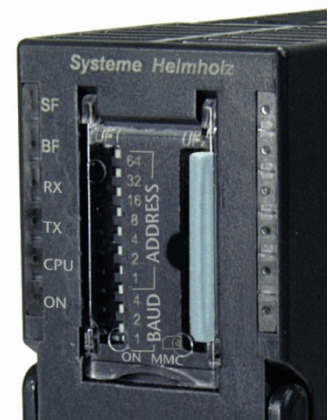
#### LED "CPU" (orange):

Data transmission to the PLC active: Indicates transmission of a frame or command on the backplane bus (between the S7-CPU and the module).

#### LED "ON" (green):

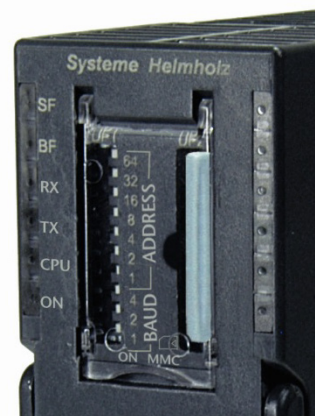
indicates that the module is correctly supplied with power and that the operating system is running.

In CANopen® master mode, a flashing LED indicates that the master is not in "operational" mode.



### 3.6 DIP switch

The 10-fold DIP switch on the front of the housing is for setting the CAN baud rate and for defining the node address (bit filter) to use the module as a CANopen® slave.



Address	$2^6$	+ 64
	$2^5$	+ 32
	$2^4$	+ 16
	$2^3$	+ 8
	$2^2$	+ 4
	$2^1$	+ 2
	$2^0$	+ 1
Baud	$2^2$	+ 4
	$2^1$	+ 2
	$2^0$	+ 1

Baud rates:

0	1	2	3	4	5	6	7
10k	50k	100k	125k	250k	500k	800k	1M



*MMCs which are used in the CAN 300 module cannot be used as standard Siemens MMC*



*If a faulty or too small MMC is plugged into the CAN 300 PRO module, the SF-LED lights up.*

### 3.7 Project memory card MMC

The CAN 300 PRO modules store the project in an internal memory (256 kBytes). As an option, the project can be stored on an MMC. With the MMC, the valid project can be transferred onto the new module when the module is replaced.

If an MMC is in the slot of the module, this is also copied to the MMC when the project is transferred to the module.

The MMC must have a memory capacity of 256 kBytes.

Micro Memory Card, 256 kBytes 700-953-8LH30

### 3.8 Items supplied

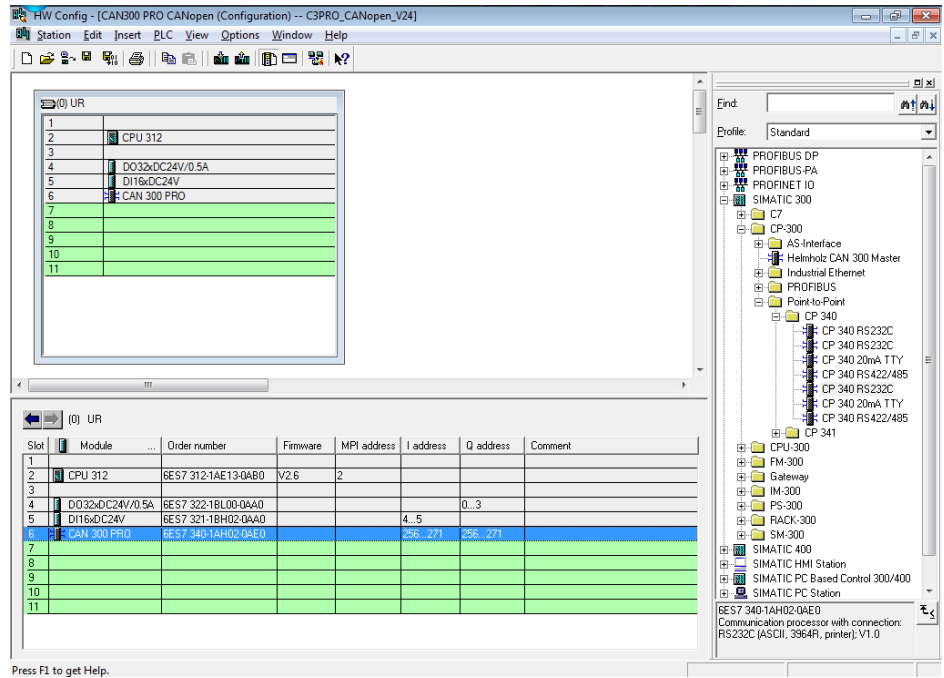
CAN 300 PRO module, bus connector, USB cable

### 3.9 Accessories

CAN CD with parameterization software "CANParam," "Layer 2," "CANopen®," and "SAE J1939" data handling blocks	800-600-1AA11
Manual, German/English	900-600-CAN01
CAN bus plug connector	700-690-0BA12
CAN bus plug connector with cable connector	700-690-0BB12
CAN bus plug connector with axial cable outlet	700-690-0CA11

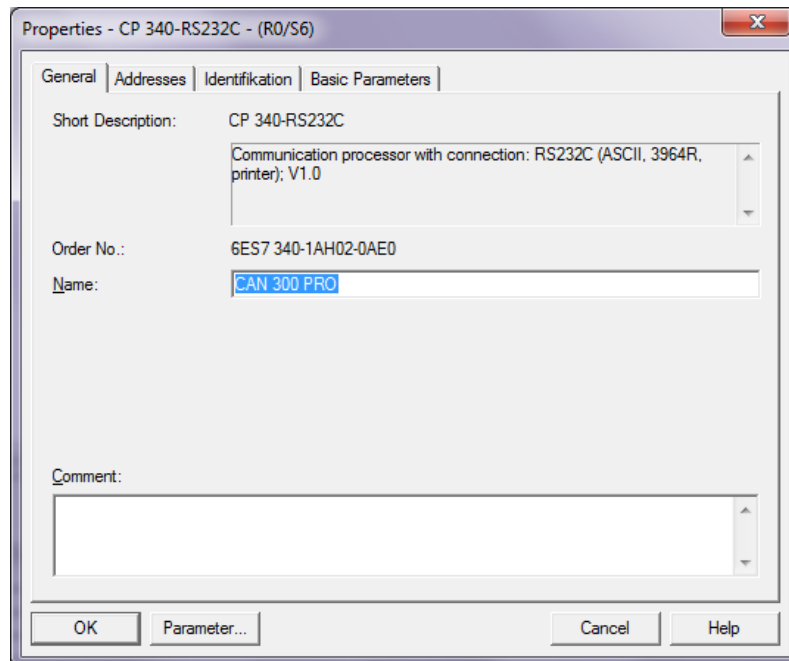
## 4 Configuration in the PLC

The CAN 300 PRO module is configured as the CP 340 communication module in the programming software of the PLC.



*When the CAN 300 PRO module is used in an ET200M system, noticeably poorer performance must be expected.*

The module can be used wherever a CP module is allowed, i.e. also in the expansion unit after an interface module.

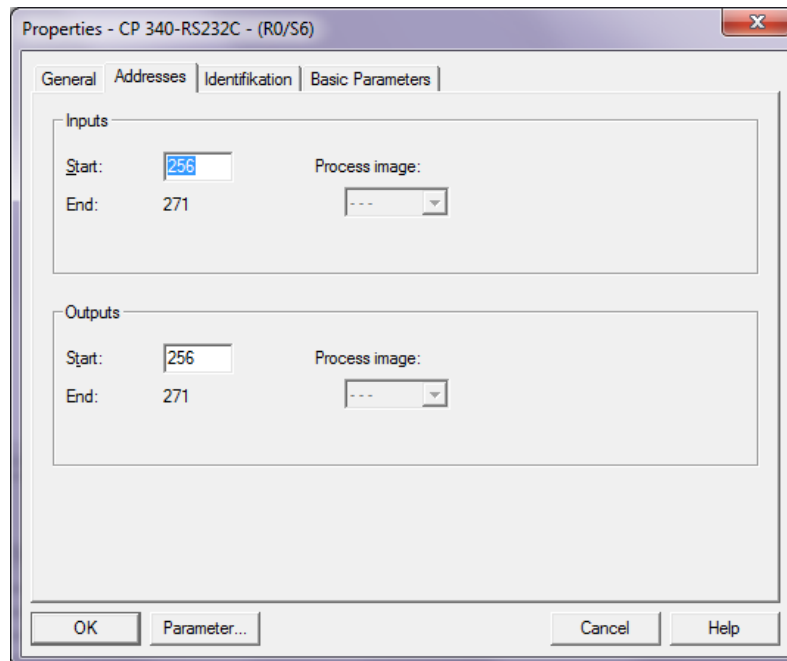






The I/O addresses should **not** be in the cyclic process image!

In parameterization of the module, only the range of I/O addresses is relevant. All other settings have no effect on the module.



Only the input image is used in the data handling blocks; the output image has no relevance to the function.

The input image can only be accessed with the I/O direct access commands: L PIB, L PIW.

In the case of the CPU 318, the I/O addresses must be outside the cyclic process image.

## 5 Process image in the PLC

The CAN 300 PRO module occupies 16 bytes in the input and output process image. The content of the output process image is not used.

The content of the input process image can be used for information purposes by the user in the application.

Byte	Meaning
0	Module status generally, CAN group error
1	CAN controller status (register of the CAN controller)
2	FIFO status bits (send & receive)
3	CAN controller: TX error counter
4	CAN controller: RX error counter
5	CANopen <sup>®</sup> : Masterstatus
6	CANopen <sup>®</sup> : Assignment of the SDO request mailboxes
7	CANopen <sup>®</sup> : Number of nodes in operational
8	Node ID on use of the bit filter or of the master
9	<i>Reserved</i>
10	<i>Reserved</i>
11...15	<i>used internally</i>

The input image can only be accessed with the I/O direct access commands: L PIB, L PIW

Bytes 5, 6, and 7 are only assigned data that can be evaluated in the CANopen<sup>®</sup> master mode.

### 5.1 Byte 0: Module status

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CAN controller group error	Module is CAN 300 PRO						Module parameterized and running

Bit 0: The CAN 300 PRO module has processed the configuration and is ready for operation.

Bit 6: This bit is always 1 in order to detect the CAN 300 PRO.

Bit 7: Group error bit for errors on the CAN controller, more precise information about the cause of error can be found in byte 1.

## 5.2 Byte 1: Error status (EFLG) of the CAN controller

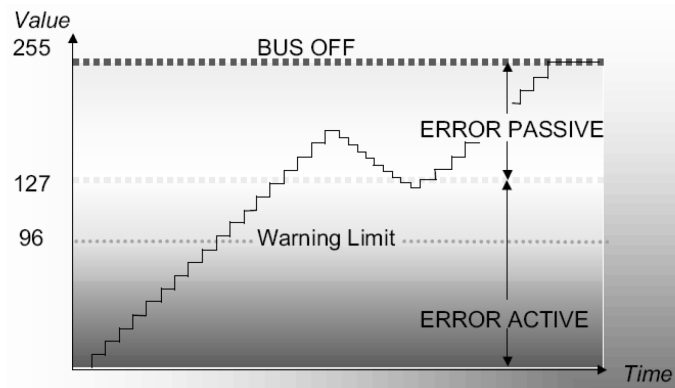
	RX1OVR	RX0OVR	TXBO	TXEP	RXEP	TXWAR	RXWAR	EWARN
	bit 7							bit 0
bit 7	<b>RX1OVR:</b> Receive Buffer 1 Overflow Flag - Set when a valid message is received for RXB1 and CANINTF.RX1IF = 1 - Must be reset by MCU							
bit 6	<b>RX0OVR:</b> Receive Buffer 0 Overflow Flag - Set when a valid message is received for RXB0 and CANINTF.RX0IF = 1 - Must be reset by MCU							
bit 5	<b>TXBO:</b> Bus-Off Error Flag - Bit set when TEC reaches 255 - Reset after a successful bus recovery sequence							
bit 4	<b>TXEP:</b> Transmit Error-Passive Flag - Set when TEC is equal to or greater than 128 - Reset when TEC is less than 128							
bit 3	<b>RXEP:</b> Receive Error-Passive Flag - Set when REC is equal to or greater than 128 - Reset when REC is less than 128							
bit 2	<b>TXWAR:</b> Transmit Error Warning Flag - Set when TEC is equal to or greater than 96 - Reset when TEC is less than 96							
bit 1	<b>RXWAR:</b> Receive Error Warning Flag - Set when REC is equal to or greater than 96 - Reset when REC is less than 96							
bit 0	<b>EWARN:</b> Error Warning Flag - Set when TEC or REC is equal to or greater than 96 (TXWAR or RXWAR = 1) - Reset when both REC and TEC are less than 96							

## 5.3 Byte 2: FIFO status bits

Bit 7	Bit 6	Bit 5	Bit 4
Send-FIFO (high) half full	Send-FIFO (high or low) overflow	Send-FIFO (low) half full	Send-FIFOs (high & low) completely empty
Bit 3	Bit 2	Bit 1	Bit 0
Receive-FIFO (high) half full	Receive-FIFO (high or low) overflow	Receive-FIFO (low) half full	Receive-FIFOs (high & low) completely empty

## 5.4 Byte 3/4: CAN controller Tx/Rx error counter

The error counter is incremented on every CAN frame transmitted or received with an error. If a CAN frame has been correctly transmitted, the error counter is decremented again. If the counter is greater than 96, the CAN controller goes into “warning” mode (see 5.2). If the error counter exceeds 127, the CAN controller goes into “error passive.”



### **5.5 Byte 5: CANopen® master status**

This byte indicates the current state of the CANopen® master state machine.

Values up to 20 indicate that the master is still starting up or is in the initialization phase of the CANopen® slaves. Values greater than 21 indicate that the master is in cyclic operation.

### **5.6 Byte 6: Assignment SDO requests (CANopen® master)**

The CAN 300 PRO module can process up to 8 SDO requests at the same time in CANopen® master mode. 4 request channels are used internally for this purpose. This byte provides information about the number of currently running SDO requests, both internal requests and those started by the PLC.

*Note:* The CANopen® master data handling software can currently only manage one request at a time. This byte is intended for later more complex application.

### **5.7 Byte 7: Nodes in Operational (CANopen® master)**

This byte indicates the number of slaves already parameterized by the master and started.

### **5.8 Byte 8: Active node ID**

This byte indicates in layer 2 mode, the active node ID of the bit filter (see Section 6.3.4) or, in CANopen® mode, the node ID of the master.

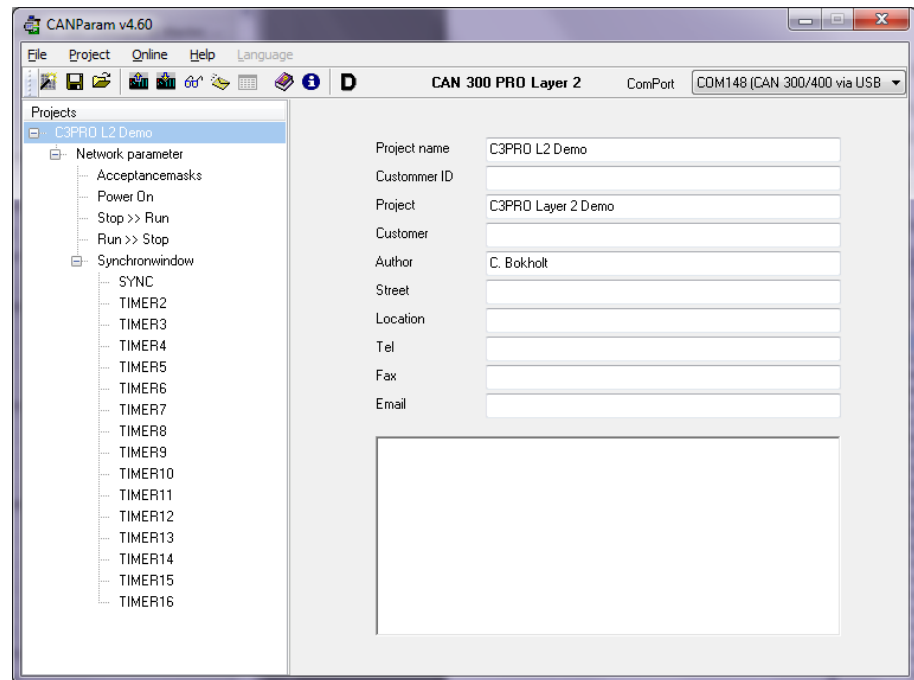
If no bit filter is used in layer 2, the value 0xFF will be in this byte.

## 6 Configuration of the module

### 6.1 Overview

The CAN 300 PRO module is configured on the PC with the “CANParam V4” software. This software is supplied together with the data handling blocks or can be downloaded from [www.helmholz.com](http://www.helmholz.com).

The software is executable on any Windows 2000/XP/7 computer.



The configuration of a module can be stored in a project file on the PC.

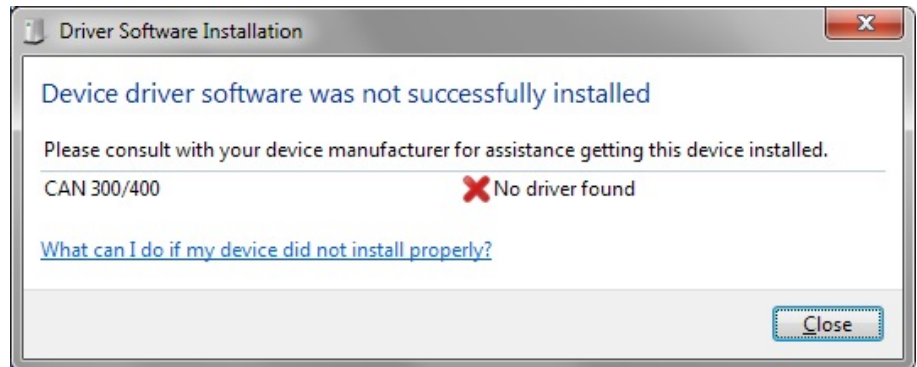
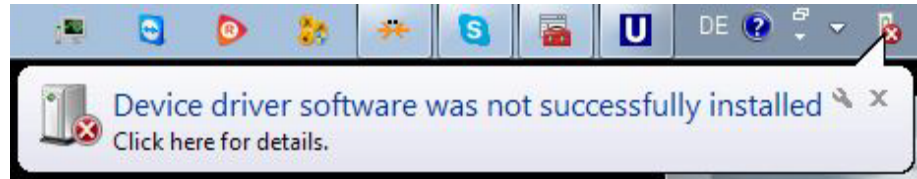
You can use a normal commercial type USB cable to link the PC to the CAN 300 PRO module.

### 6.2 Installation of the USB interface driver

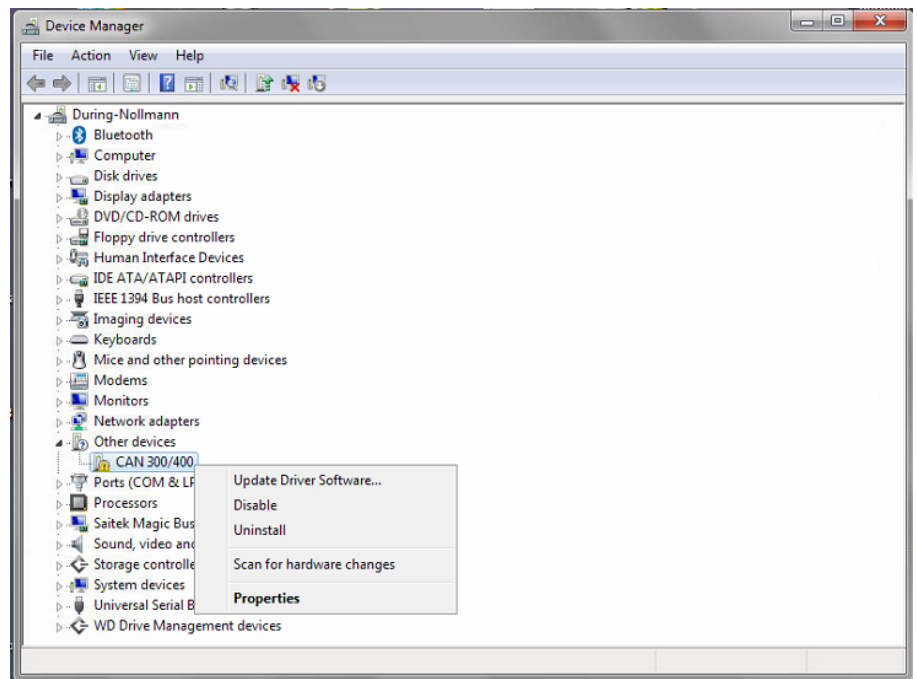
If this is the first time a CAN 300 PRO module is being connected to the PC, the operating system will try to install a suitable USB driver. This driver provides the interface between the USB interface and the operating system (Windows).

This initialization can take some time and is as follows:

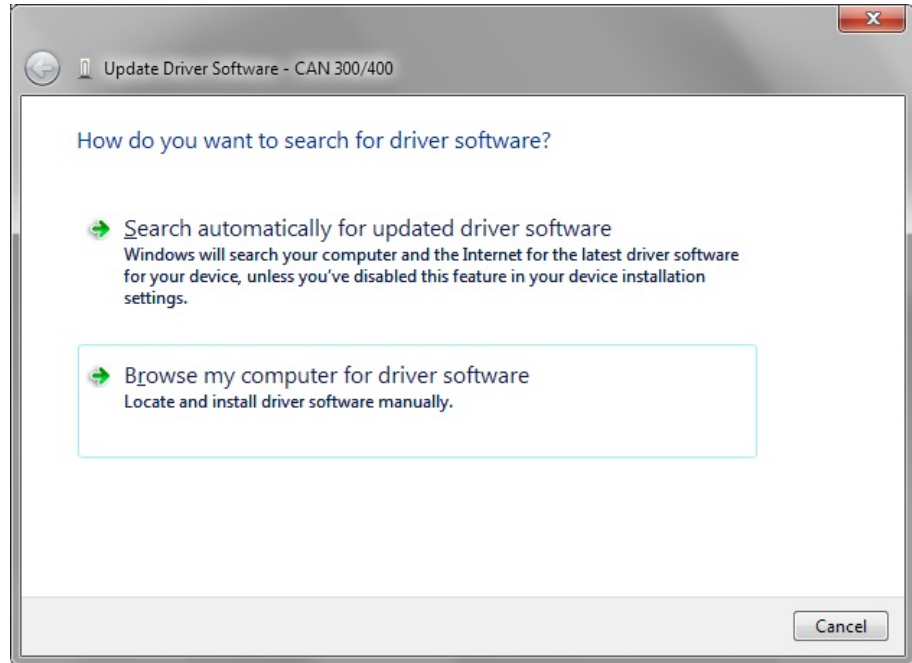
- The operating system starts an installation wizard that performs an automatic installation. In case it ends with the message “Device driver software was not successfully installed” you have to perform the following steps:



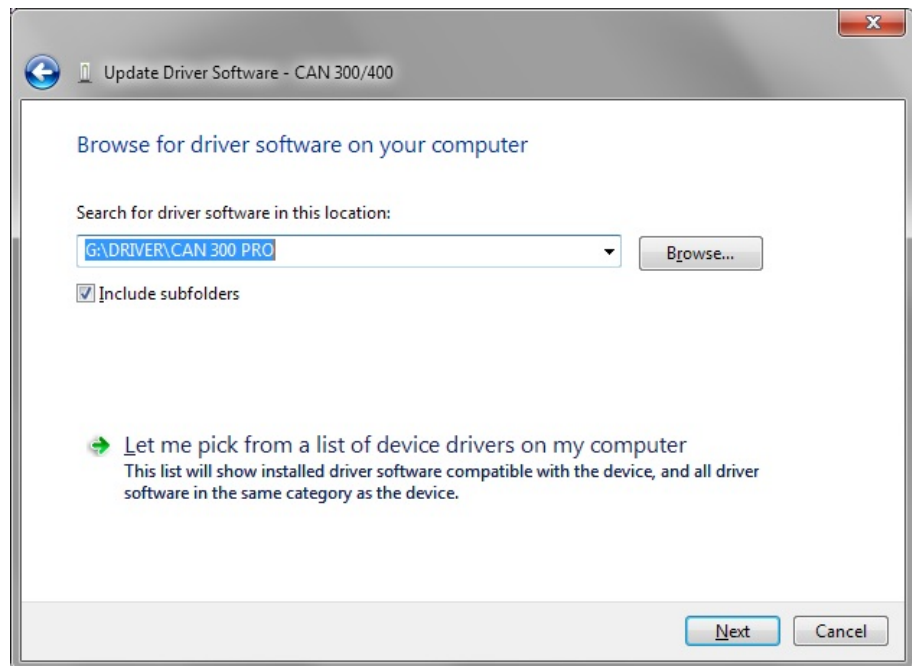
- Open the device manager and right click on the entry “CAN300/400”. Select “Update Driver Software...”.



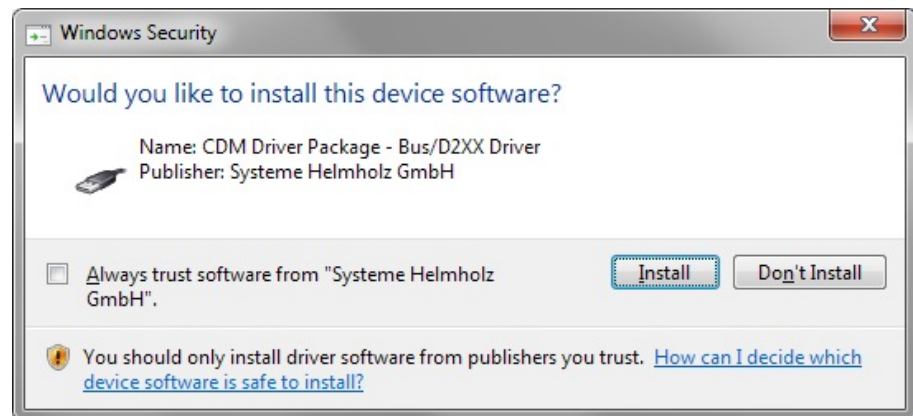
- Select “Browse my computer for driver software”.



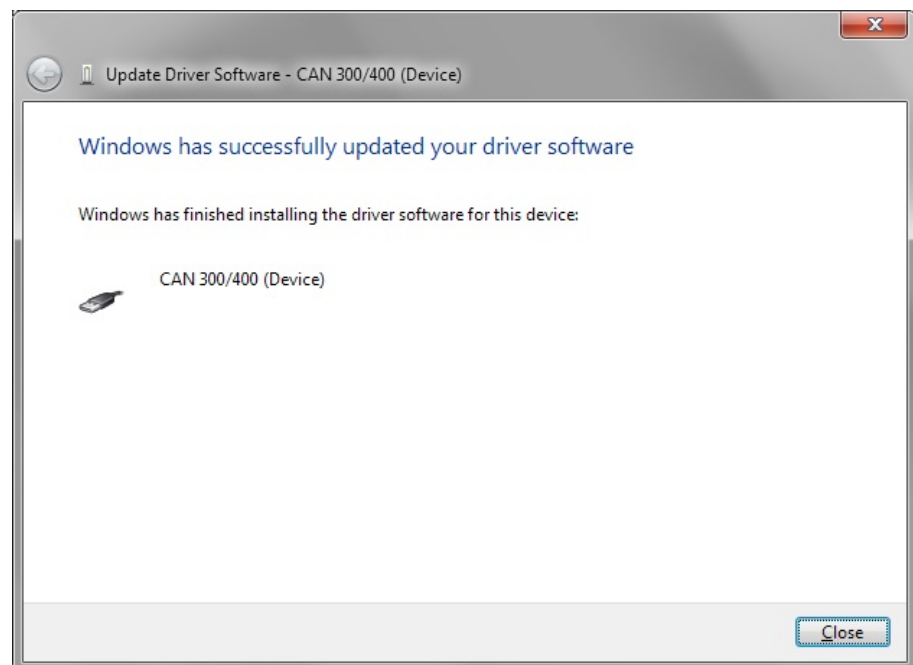
- Specify the location of the driver, generally it should be the CAN CD and click “Next”.



- If the driver is found a Security request is displayed. Click “Install”.

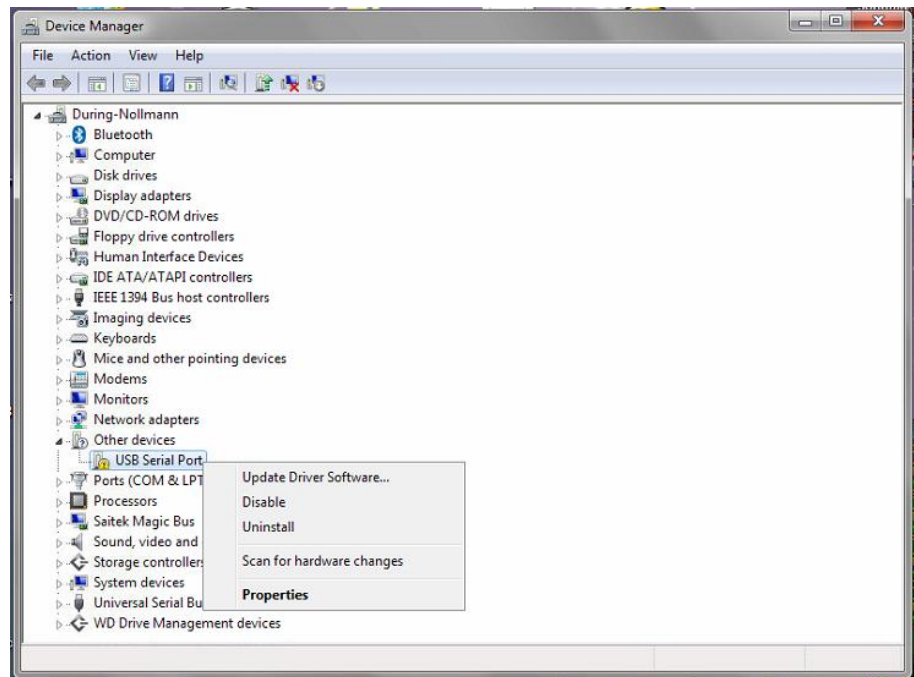


- After successful installation click the “Close” button.

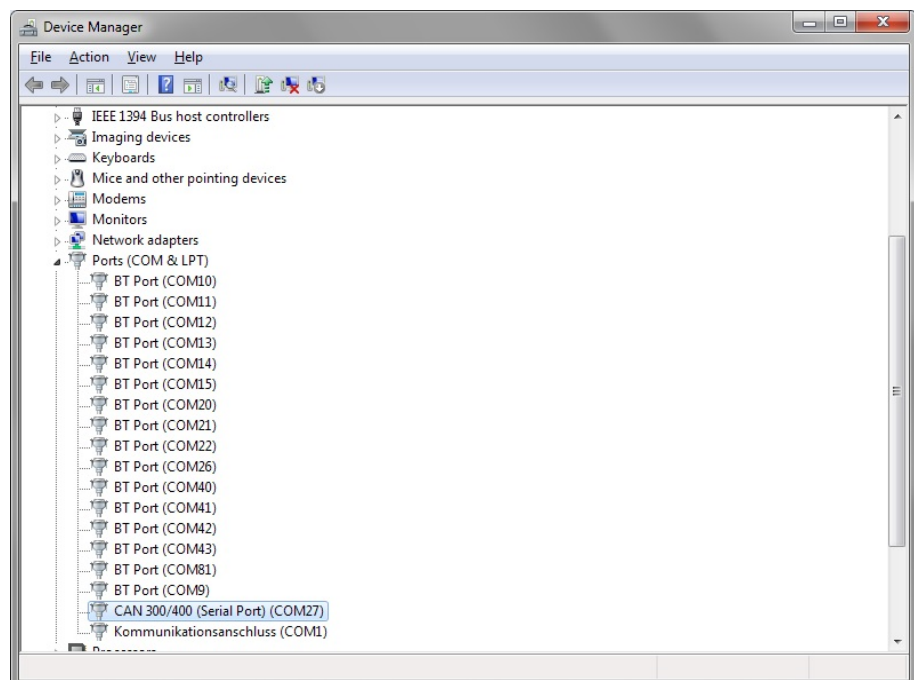




- The operating system starts the installation wizard a second time to install the virtual COM port driver. The installation routine is identical to the procedure described above.

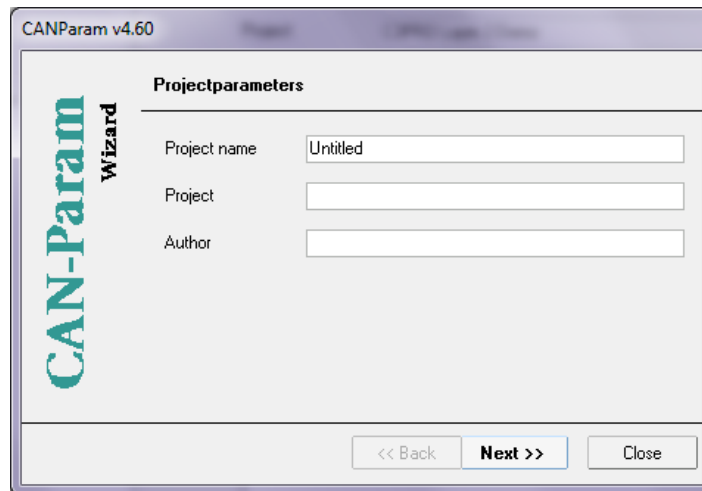


- After successful installation a new COM port is added in the device manager. This COM port must be used in the CANParam software to be able to communicate with the CAN 300 PRO module.

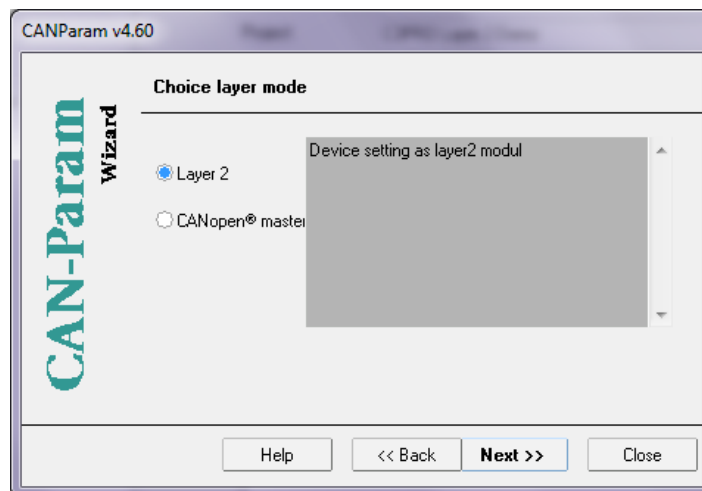
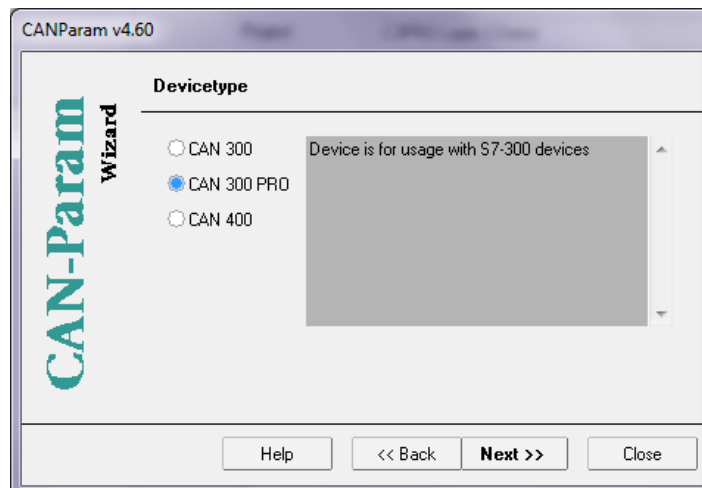


### 6.3 Creating a layer 2 project

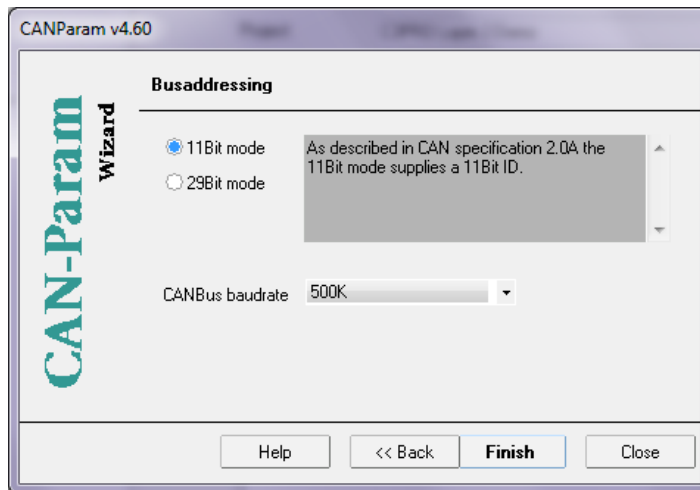
A new project can be created via the “Project / Create project / New project” menu item or with the project wizard.



The project wizard guides you through the most important settings to obtain a new and complete project.

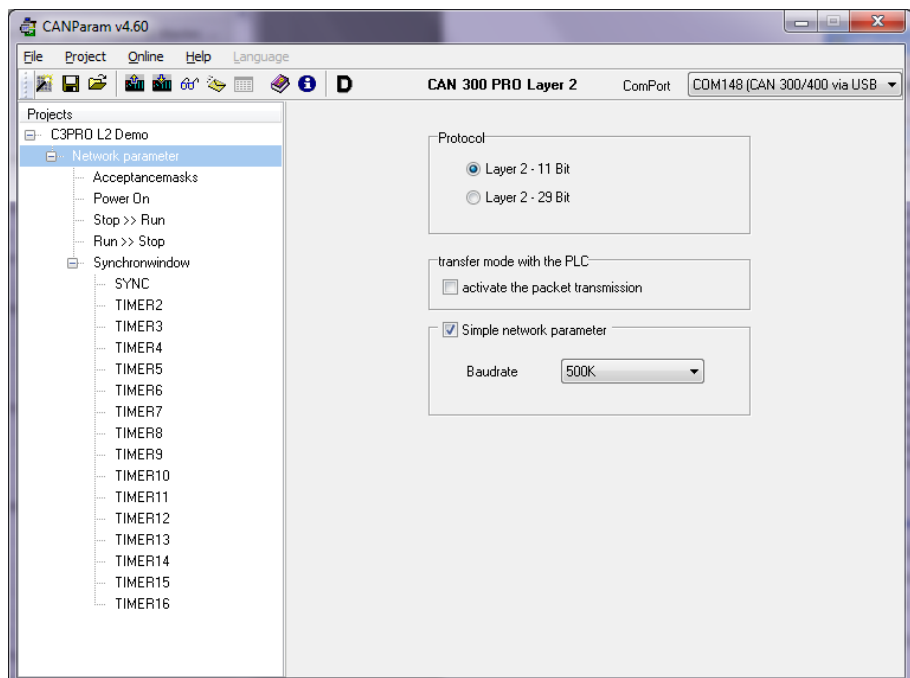


*On creating CANopen master projects, please read Section 6.4 .*



### 6.3.1 Setting the CAN bus baud rate

The CAN baud rate can be selected in the range of 10kbps to 1Mbps.



As an alternative, the baud rate can be set using the DIP switch.



*Mixed operation with 11-bit and 29-bit identifiers is not possible!*



*Depending of the transfer mode setting, the right layer 2 communication handling blocks are to use.*

### 6.3.2 Setting the protocol type

The CAN 300 PRO module supports both the protocol format CAN 2.0A (11 bits) and CAN 2.0 B (29 bits).

For use of the SAE J1939 data handling blocks, a CAN 2.0B (29 bits) must always be selected.

### 6.3.3 PLC transfer mode

For using the layer 2 packet oriented communication select the option “activate the packet transmission” and use the “CANSEND Block Mode”/“CANRCV Block Mode” handling blocks.

### 6.3.4 Mask filter

16 mask filters (acceptance masks) are available in the CAN 300 PRO module. Using these masks you can permit or block various frame IDs from being received.

		Begin	End
<input checked="" type="checkbox"/> Mask 1	<input checked="" type="checkbox"/> Highprior	0x000	0x7FF
<input type="checkbox"/> Mask 2	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 3	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 4	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 5	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 6	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 7	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 8	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 9	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 10	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 11	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 12	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 13	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 14	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 15	<input type="checkbox"/> Highprior	0x000	0x000
<input type="checkbox"/> Mask 16	<input type="checkbox"/> Highprior	0x000	0x000

**I**  
The default setting of the acceptance mask (0h to 7FFh) is to allow receipt of all frames.

With the “high priority” option, it is possible to deal with CAN frames with priority. Frames that are received with the IDs set there will be passed to the S7 as the next frame bypassing the normal receive buffer.

### 6.3.5 Bit filter

As an alternative to the acceptance masks, the CAN frames received can also be filtered according to a node ID.

The node ID is used, for example, in CANopen® networks to identify CANopen® slaves.

Node Address

Use from DIP

If the CAN 300 PRO is to be used as a slave station, filtering for all CAN frames for this station can be defined via the node ID setting. The node ID is stored in the lower 7 bits of the CAN ID.

In addition to the CAN frames with the defined node ID, the following frames with the node ID 0 are let through:  
COB-ID 0x0, 0x80, 0x100, 0x180, 0x200, ...

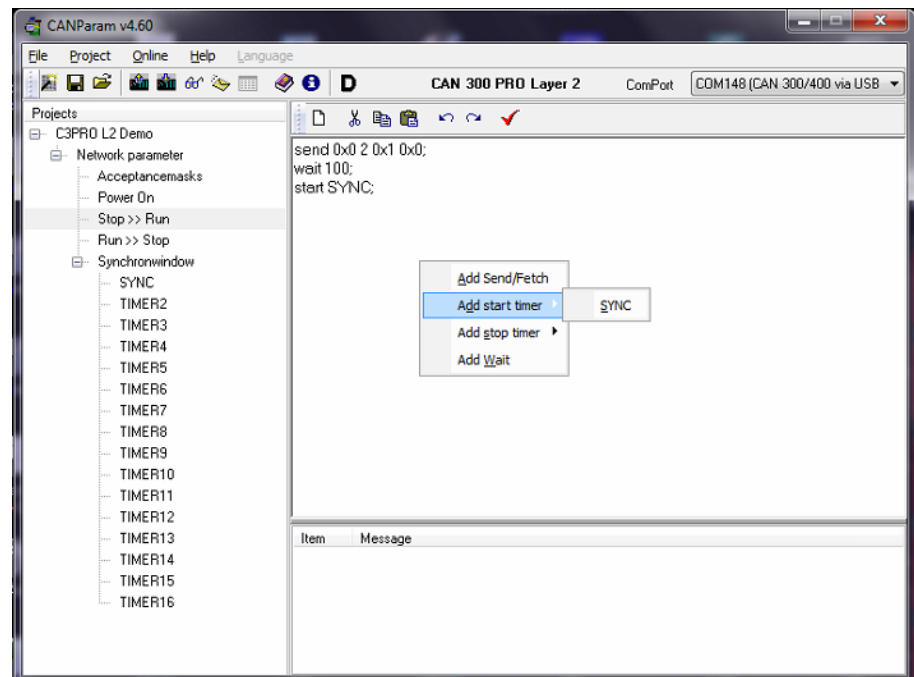
The node ID can either be defined permanently in the project, or set on the module via the DIP switch.

### 6.3.6 Event scripts

The CAN 300 PRO module can transmit freely programmable CAN frames (layer 2) for the PLC events “Power ON,” “Stop >> Run,” and “Run >> Stop,” and start and stop timers.

The following commands are available:

<b>Send</b>	Transmit frame (Structure ID, length, data byte 1, data byte 2, etc.)
<b>Fetch</b>	Transmit frame with RTR bit 1
<b>Start</b>	Start Timer X
<b>Stop</b>	Stop Timer X
<b>Wait</b>	Wait X (1...65535) ms
<b>//</b>	Comment line



100 lines per script can be created with the CAN 300 PRO module.

### 6.3.7 Timer

16 timers are available for time-dependent events in the CAN 300 PRO module. Each timer can transmit any CAN frame.

The screenshot displays two configuration panels: 'Timer' and 'Action'.

**Timer Panel:**

- Alias: SYNC
- Highprior
- To PLC
- Repetition: 500 msec
- Phase: 0 msec

**Action Panel:**

- ID: 0x080
- Fetch
- RTR Length: 0
- Data fields 1 through 8, each containing 0x00

An alias can be assigned to each timer. This name can then be used in the scripts of the PLC events.

The time *repeat* states the repeat interval for the timer, the *phase* the starting point within the interval.

For the timer *period*, times from 1 msec. to 65535 sec. can be set in steps of 1 msec. For the *phase* times from 0 msec to 1 msec before the period duration can be set.

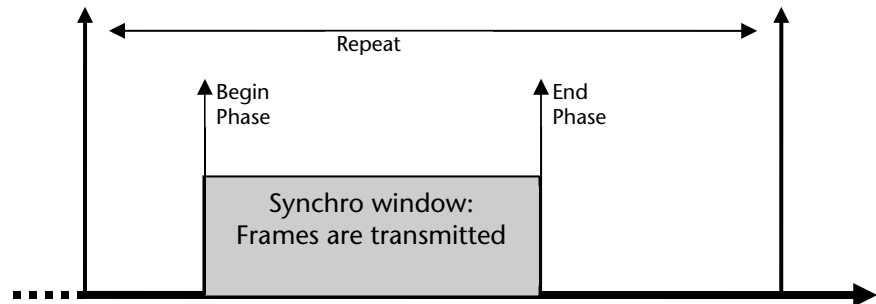
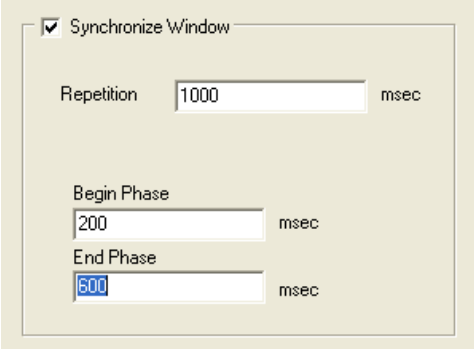
If the option “to PLC” is selected, the frame is transmitted simultaneously on the CAN bus and to the PLC. With this option, the PLC can be synchronized on a CAN frame.

### 6.3.8 Synchro window

If you are using the synchronous timer (setting “synchronous queue”), the frames transmitted asynchronously by the FB 60 “CANSEND” are transmitted within a time window. “Repeat” indicates the repeat rate, “Begin phase” & “End phase” defines the transmit window within the repeat time.

The frames to be transmitted are only transmitted within the time window between “Begin Phase” & “End Phase.”

This makes time on the bus outside the synchronous window for communication by other stations.



*To use the synchronous window, both timers 1 & 2 must be started directly one after the other in a script!*

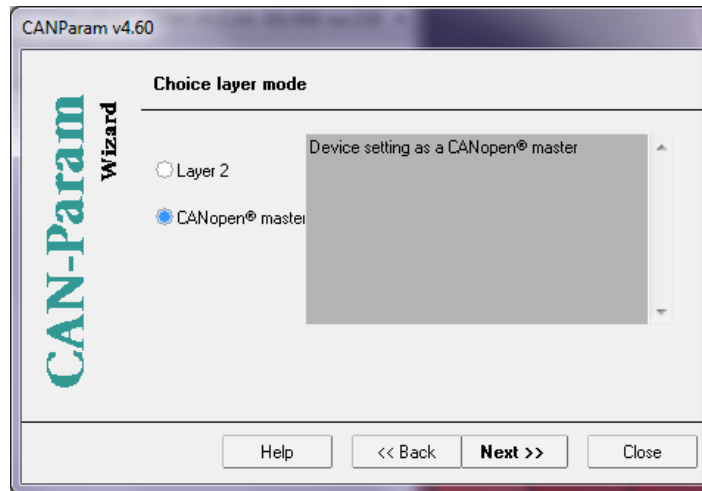
Timer 1 “SYNCBEGIN” and Timer 2 “SYNCEND” are used internally, if the synchronous window is used. However, this must be started by the user, e.g. in a script. For the synchronous window to function correctly, these two timers must be started in succession.

## 6.4 Creation of a CANopen® project

For applications with CANopen® stations, the CAN 300 PRO module can be parameterized as a CANopen® master.

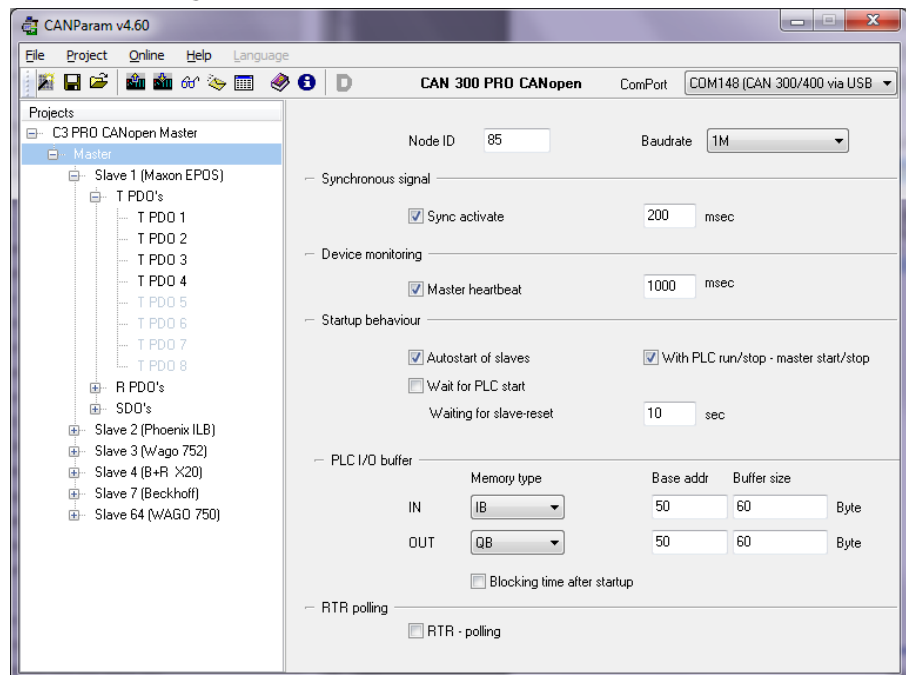


*For initial start-up as the CANopen master, please also use the document “Start-Up Guide CANopen for CAN 300 PRO.”*



The CANopen® master in the CAN 300 PRO module works independently of the PLC.

### 6.4.1 Settings of the master



**Node ID:** The master requires a node ID for broadcasting the master heartbeat. The node ID can be defined in the range 0...127.

**Baud rate:** Definition of the CAN bus baud rate (10Kbaud .. 1Mbaud)

**Synchronous signal:** The master can broadcast a SYNC frame (ID: 0x80) in fixed time period.

**Device monitoring “master heartbeat”:** The master broadcasts a heartbeat frame in the parameterized timebase (ID: 700 + node ID of the master).



**Startup behavior “Autostart of the slave”:** Slaves that fail during operation and then recover are automatically re-initialized and included in cyclic operation again.

**Startup behavior “With PLC Run/Stop – Master Start/Stop”:** The master starts and stops if the PLC is started or stopped.

**Startup behavior “Wait for PLC start”:** On restart the master initializes the slaves and waits for the release command from the PLC (Sec. 7.8.4) before it goes into cyclic operation.

**Startup behavior “Wait for slave reset”:** The master transmits an NMT reset to all slaves. Some slaves require a fairly long time to complete the reset and register on the master again (bootup). The maximum waiting time can be defined here.

*Note:* Some CANopen® devices require up to 10 seconds or longer before they register on the bus after a reset. The slave wait time should not be set too short on initial start-up.

**PLC I/O buffer “IN”:** Buffers for receive PDOs

**PLC I/O buffer “OUT”:** Buffers for PDOs to be transmitted

The size of the buffer must match the size specified in the PLC to ensure correct data transmission.

The memory type and basic address stated are for information only and should match the programming in the PLC to make assignment of the PDOs in the PLC easier.

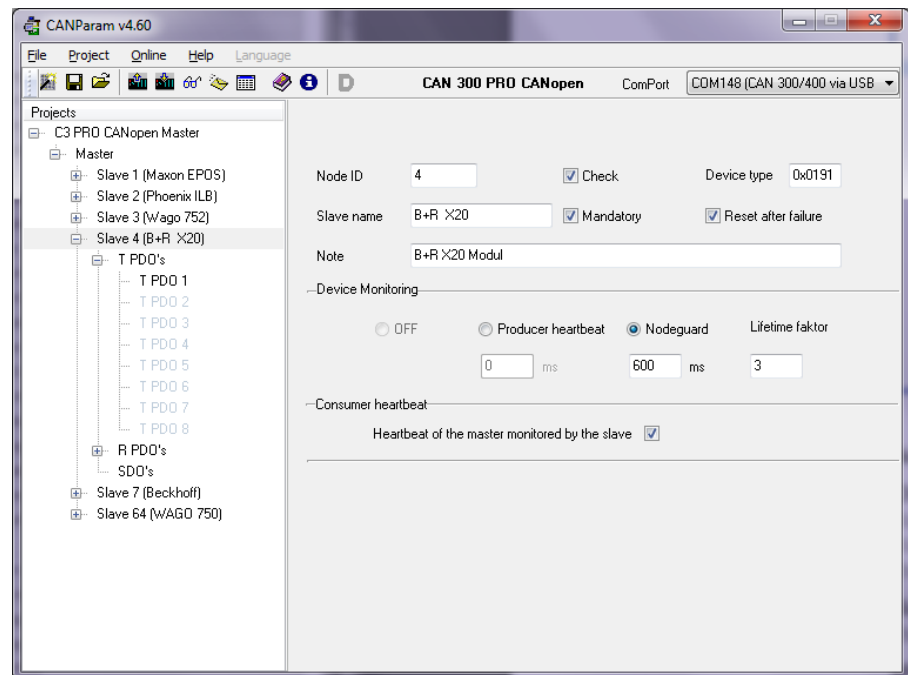
Section 6.4.5 contains a precise explanation of the PLC I/O buffer.

**PLC I/O buffer “Blocking time after start-up”:** Definition of the waiting time after start-up of the master until the first data are transmitted to the PLC.

**RTR Polling:** Definition of the timebase in which PDOs are to be queried via RTR frames.

## 6.4.2 Creating a slave

With operation of the right mouse button in the project tree, it is possible to create a new slave. Slaves can be defined manually or generated from a EDS file.



**Node ID:** Node ID of the slave

**Check device type:** With this option, the master checks during start-up whether the slave had the specified device type on the stated node number. For this purpose, the SDO 1000 is read and the lower 2 bytes are used as a comparison.

**Slave name:** Name of the slave, freely selectable, is displayed in the project tree.

**Mandatory device:** The slave must exist so that the master can enter cyclic operation.

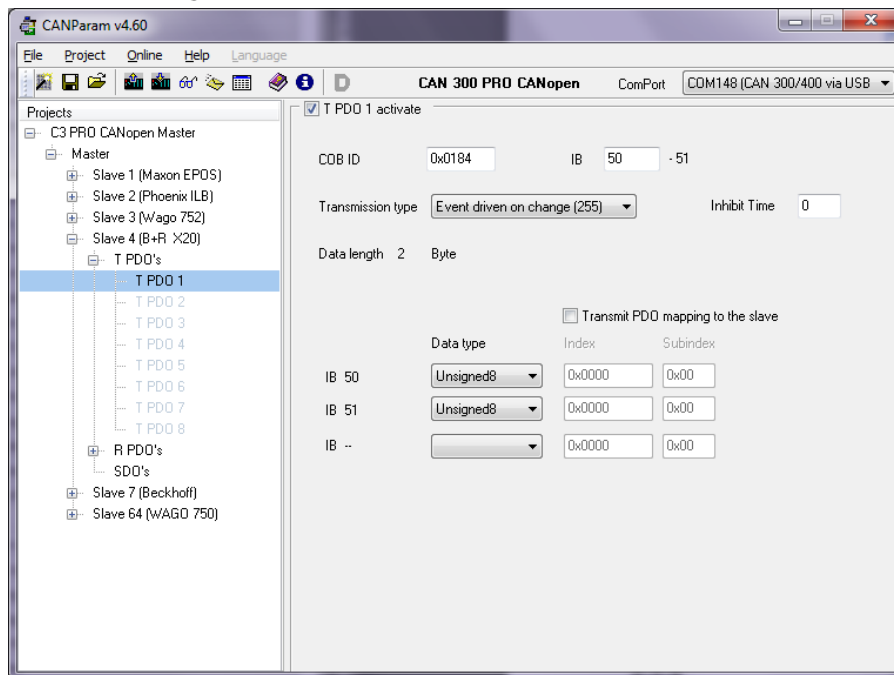
**Comment:** Further information on the slave

**Device monitoring “producer heartbeat”:** Monitoring of the slave by the master through the producer heartbeat signal of the slave. The setting is written into the SDO 1017 on start-up.

**Device monitoring “Nodeguarding”:** Monitoring of the slave by the master by nodeguarding. The settings are written into the SDOs 100C and 100D of the slave on start-up.

**Consumer heartbeat:** The slave monitors the heartbeat of the master. As the monitoring time, the master heartbeat time times 1.5 is used. The object 1016/1 is written during start-up.

## 6.4.3 Setting TPDOs



**COB-ID:** Indication of the associated CAN-ID, depending on the node ID and the PDO number. The COB-ID should only be changed in special application cases.

**Address in the IO buffer:** Definition of the start address of the data of the PDO

**Transmission type:** Setting of the transmission mechanism. The following options are possible:

SYNC acyclic (0): Transmission after the next sync frame, if there is a change.

SYNC (1..240): Transmission after every nth sync frame

RTR-only synchronous (252): Transmission on request (RTR) after the next sync telegram. In the master, the RTR polling must be activated and a sync frame transmitted on the bus.

RTR-only asynchronous (253): Transmit immediately on request (RTR). The RTR polling must be activated in the master.

Event-driven manufacturer (254): Transmission after change

Event-driven standard (255): Transmission after change

The setting is automatically written into the object 0x1800 (TPDO 1), 0x1801 (TPDO 2), 0x1802 (TPDO 3), and 0x1803 (TPDO 4).

**Inhibit Time:** Minimum time interval for two PDO frames to be transmitted consecutively using transmission type “Event driven ...” (254 / 255).

**Data length:** Number of bytes used in the PLC input buffer and by the PDO frame



*Data overlaps due to an incorrectly set offset can corrupt data!*



*If the data types are set incorrectly, data may be corrupted!*

**Data type:** The data type allocation of the PDO must be defined in this list. A PDO frame consists of up to 8 bytes.

The data types now have to be assigned from the left/first byte to the last byte of the PDOs used.

Example:

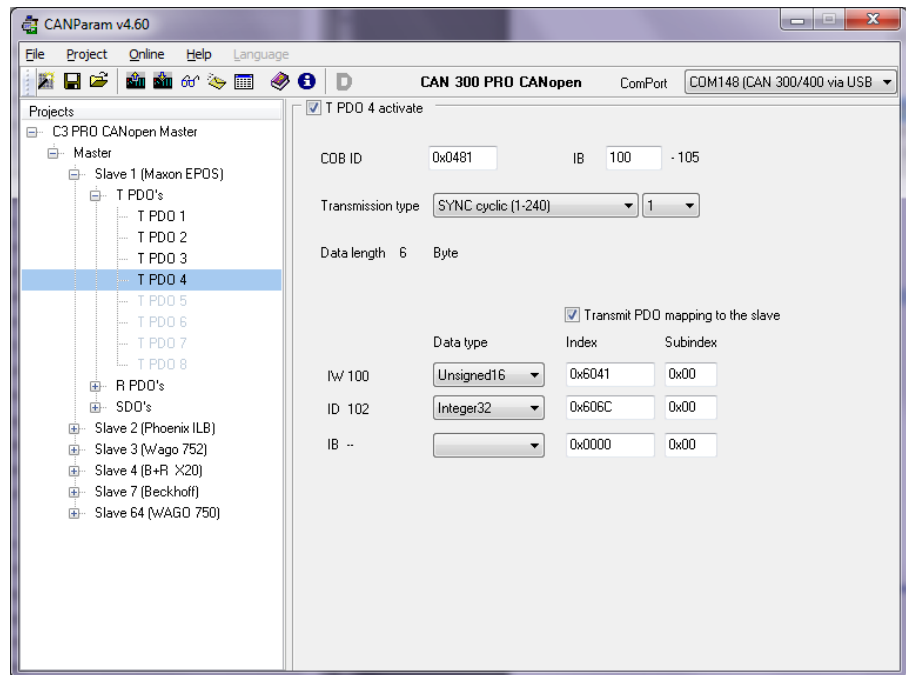
The TPDO4 has 6 used bytes; the first two bytes are interpreted as unsigned16 and bytes 3-6 as integer32.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Unsigned16		Integer32				Unused	Unused

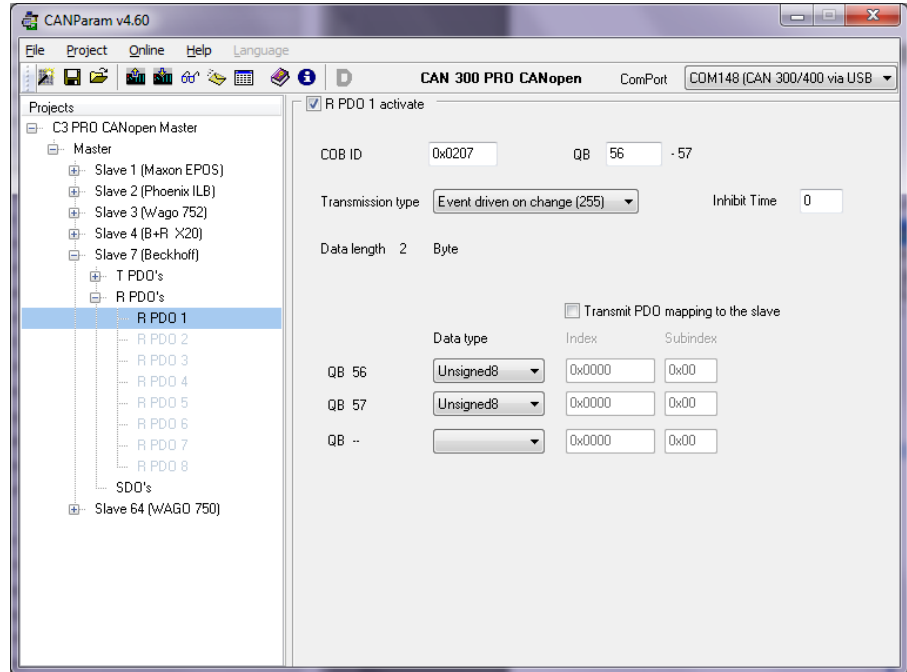
The CAN 300 PRO will condition the values internally with this information and enter them correctly in the IO buffer for the PLC.

The following data types are available: Integer8, integer16, integer32, unsigned8, unsigned16, unsigned32.

**Transmit PDO mapping to the slave:** The SDO mapping of the PDOs can be specified directly in addition to the list of data types. If the option is activated, this mapping will be written into the objects 0x1A00ff. during start-up.



## 6.4.4 Setting RPDOs



*Data overlaps due to an incorrectly set offset can corrupt data!*

**COB-ID:** Indication of the associated CAN-ID, depending on the node ID and the PDO number. The COB-ID should only be changed in special application cases.

**Address in the IO buffer:** Definition of the start address of the data of the PDO

**Transmission type:** Setting of the transmission mechanism. The following options are possible:

Sync trigger by PLC (0): RPDO is transmitted to the slave if a SYNC signal arrives and the data have changed

Sync (1..240): Acceptance after every nth sync frame

Event-driven on PLC cycle (254): After each IO write call, the master transmits the data on the bus; the slave accepts the data immediately

Event-driven on change (255): Master transmits after each change of the value; the slave immediately accepts the data

The setting is automatically written into the object 0x1400 (RPDO 1), 0x1401 (RPDO 2), 0x1402 (RPDO 3), and 0x1403 (RPDO 4).

**Inhibit Time:** The minimum time interval for two PDO frames to be transmitted consecutively.

**Data length:** Number of bytes used in the PLC output buffer and by the PDO frame.



*If the data types are set incorrectly, data may be corrupted!*

**Data type:** The data type allocation of the PDO must be defined in this list. A PDO frame consists of up to 8 data bytes. The data types now have to be assigned from the left/first byte to the last used byte of the PDOs.

Example:

The RPDO1 has 3 used bytes; the first two bytes are interpreted as unsigned16 and byte 3 as unsigned8.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Unsigned16		Unsigned8	Unused	Unused	Unused	Unused	Unused

The following data types are available: Integer8, integer16, integer32, unsigned8, unsigned16, unsigned32.

**Transmit PDO mapping to the slave:** The SDO mapping of the PDOs can be specified directly in addition to the list of data types. If the option is activated, this mapping will be written into the objects 0x1600ff. during start-up.

#### 6.4.5 PLC I/O buffer principle

The PDOs of the slaves defined in the project are entered in data buffers that are exchanged with the PLC. There is one data buffer for input of PDOs (TPDOs of the slave) and one for transmitting PDOs (RPDOs of the slave).

The size of the buffer can be defined in the master setting dialog box. Data of the PDOs can be placed at any positions in these buffers.

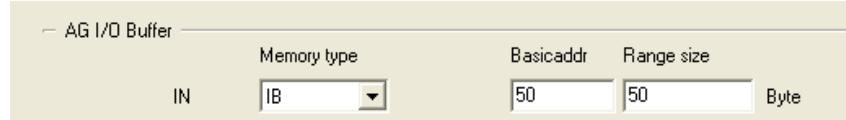
These PLC I/O buffers are loaded from the data handling software in to the PLC, or from the PLC. FB 20 "IO Read" (see Sec. 7.8.1) and FB 21 "IO Write" (see Sec. 7.8.2) are used for this.

In the PLC, the PLC I/O buffer can be copied to any memory area. The PLC IN buffer, for example, can be copied directly into the input process image (IB) and the PLC OUT buffer from the output process image (OB). This permits direct processing of the CANopen® PDOs in the process image of the PLC.

If the memory of the process image is not large enough, the marker memory or data blocks can also be used. In this way, assignment of the PDO data in this memory is simple due to the selected offset.

Example:

Size of the PLC IN buffer: 50 bytes



The ANY pointer information on FB 30 “IO Read” with “I 50.0 BYTE 50” causes the PLC IN buffer to be copied into the process image memory from IB50 to IB99:

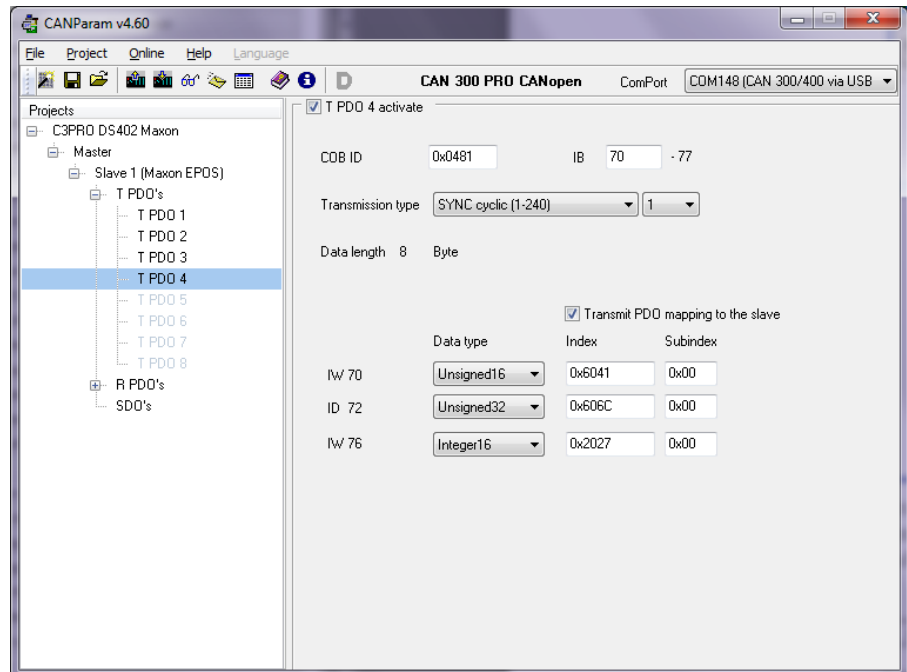
```

CALL FB 20 , DB20
Base :=256
Dest :=P#E 50.0 BYTE 50
STAT :=MW20
Err :=M22.6
RetVal :=MW24
NewData:=M22.0
    
```

Example:

The TPDO4 of Node 1 is placed at IB 70 and contains 6 bytes, assigned as Unsigned16 and Unsigned32.

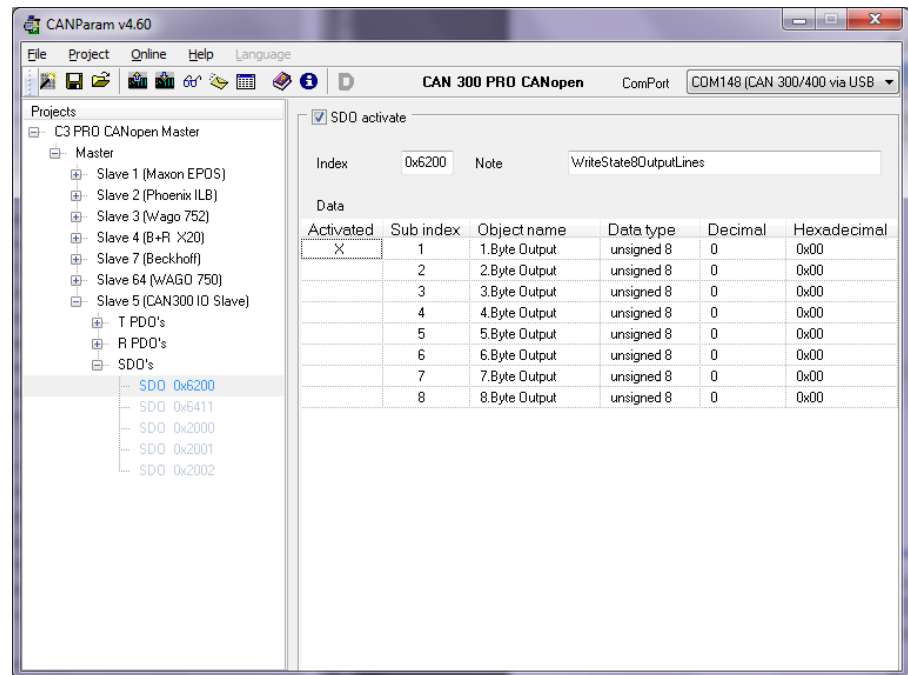
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Unsigned16		Unsigned32				Unused	Unused



In the PLC, the value of SDO 6041 in IW 70 can now be processed and the value SDO 606C is in ID 72. The values are automatically adapted to the data format of the PLC.

Address in the PLC	Assignment
...	...
IW 70	Node 1: TPDO4 (Byte 0-1) Unsigned16
ID 72	Node 1: TPDO4 (Byte 2-5) Unsigned32
IB 76	Unused
...	...
IB 99	Unused

### 6.4.6 Adding SDOs



For each slave, a list of SDOs can be stored. The values of these SDOs are written to the slave after start-up of the master and detection of the slave.

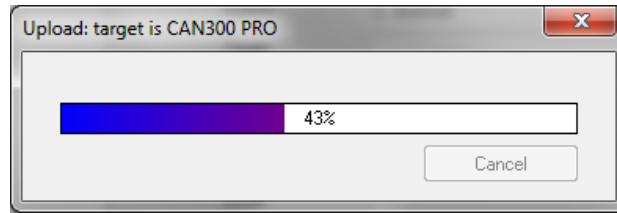
If a slave fails during cyclical operation of the master and be detected on the bus again later, the SDOs are also written (only with the master option: "Autostart of the slaves").

An error while writing the SDOs does not result in cancellation of slave initialization. The last error to occur can be viewed in the CANopen® Debug Screen (see Sec. 6.7.2).



## 6.5 Uploading

The project currently being worked on can be imported into the CAN 300 PRO module (“upload”).



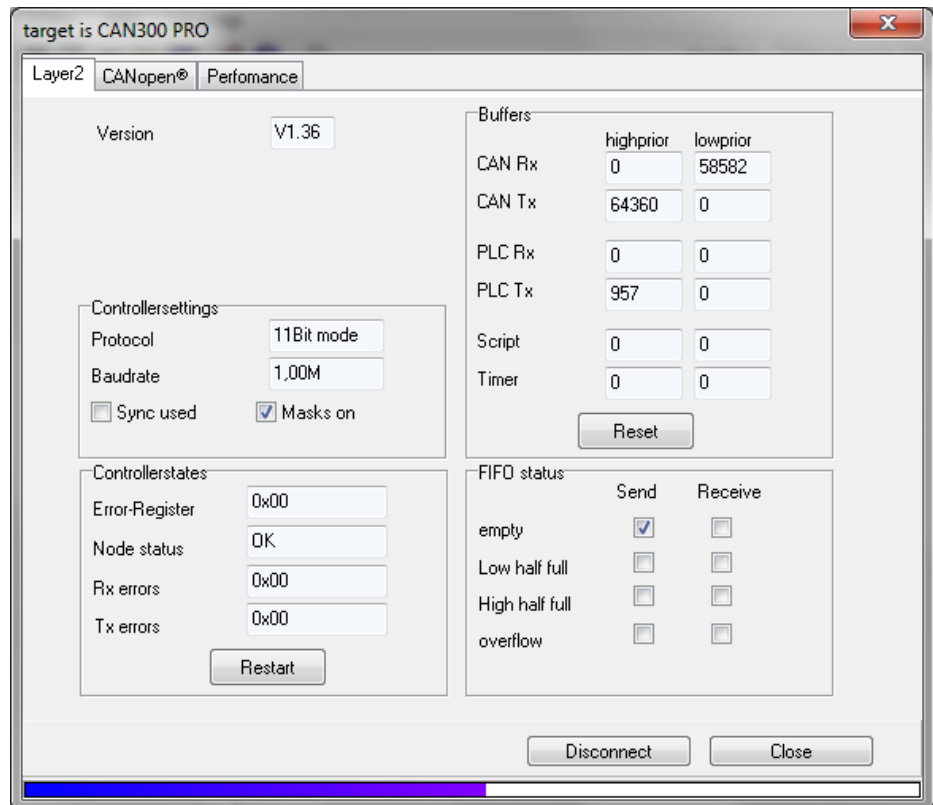
## 6.6 Downloading

Project on the module can be loaded into the CANParam software with this function also the processing.

## 6.7 Diagnostics/debugging

To simplify debugging, you can query the status of the CAN 300 PRO module with menu item “Debug.” Debug mode requires a USB link with the module.

### 6.7.1 Layer 2 debug display



The “Connect” button activates monitoring mode. If you press the button again, the link will be disconnected again.

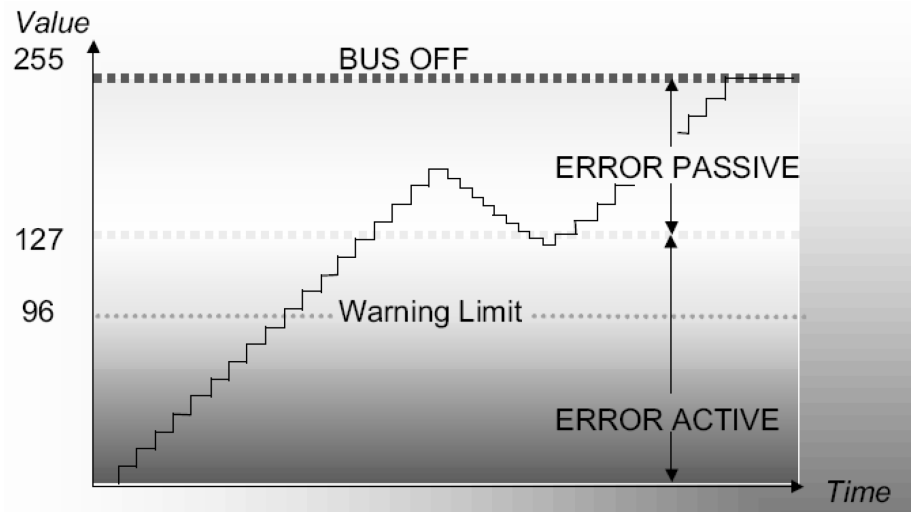
The debug dialog box provides the following information:

- Version**           Version number of the operating system
- Protocol**          Configured CAN protocol (11bit/29bit)
- Baud rate**        Active CAN baud rate
- Controller status** Content of the CAN status register:
- Error register**   Content of the CAN error register EFLG (Sec. 5.2)
- Node status**      Content of the CAN status register (see above):  
"OK," "Warning," "Passive," "Bus Off"
- Rx error counter** Error counter CAN reception
- Tx error counter** Error counter CAN transmission

!  
*Node status should always be "OK" to ensure fault-free CAN data transmission.*

!  
*The error counters must be "0"; otherwise data transmission on the CAN bus is faulty.*

**Note:** The transmit and receive error counters are incremented by the can controller if transmission or receipt of a frame has failed. As soon as a frame has been correctly sent or received, the corresponding counter is decremented again. These counters should always be 0 if the CAN bus is functioning correctly!



!  
*The information about the buffers and FIFOs are only relevant in layer 2. In CANopen master mode, the firmware performs control of the buffers.*

**Buffer**            Display of the number of received and transmitted CAN frames in the buffers on the CAN bus, to the PLC and internally

**Note:** The CAN 300 PRO module has receive and transmit buffers of 400 frames (low priority) and 20 frames (high priority). The counters show how many frames have been processed.

There should never be a big difference between the Rx and Tx counter pairs. However, if this does occur, the CAN frames are not being fetched from the PLC fast enough or are being transmitted to the PLC too fast.

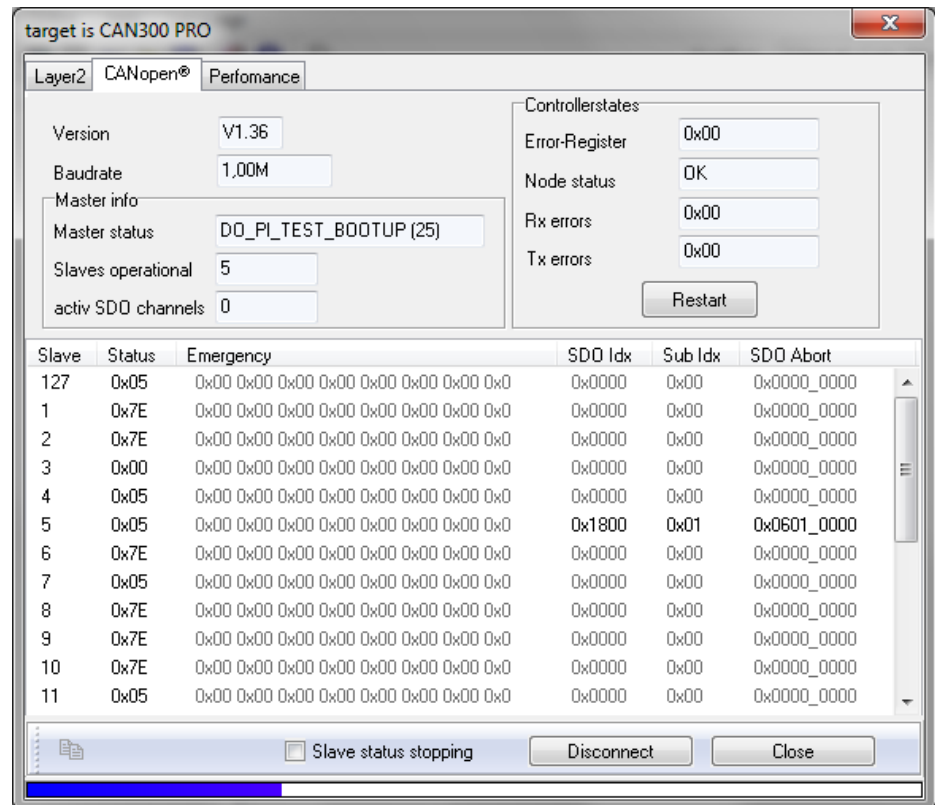
If, in the case of a full FIFO, further frames are received or transmitted, the error bit FIFO overflow is set and the oldest frame is deleted from the FIFO.

**FIFO status** Display of the filling level of the FIFOs

In PLC, the FIFO status can be evaluated via the peripheral byte 2 (Section 5.3). The FIFOs can be deleted with the data handling block FC 67 CANCTRL (see also Sec. 7.3.4). An overflow error that has occurred must also be reset for the FC 67 CANCTRL (see also Sec. 7.3.4).

### 6.7.2 CANopen® Debug display

If the CAN 300 PRO module is operated in CANopen® master operation, the status of the master and the livelist can be displayed on the second diagnostics page.



The CANopen® Debug display provides the following information:

**Version** Version number of the operating system

**Baud rate** Active CAN baud rate

**Controller status** Content of the CAN status register:

**Error register** Content of the CAN error register EFLG (Sec. 5.2)

**Node status** Content of the CAN status register (see above):  
"OK," "Warning," "Passive," "Bus Off"

**Rx error counter** Error counter CAN reception

**Tx error counter** Error counter CAN transmission

**!**  
*Node status should always be "OK" to ensure fault-free CAN data transmission.*

### Master information display:

<b>Master status</b>	Current status of the CANopen® master. <20 = upload of the master, initialization of the slave 21,22 = normal operation >22 = start-up of failed slaves in normal operation
<b>Slaves operational</b>	Number of slaves in operational
<b>Active SDO channels</b>	Number of assigned SDO communication channels

### Slave status list:

<b>Slave status</b>	0x00 = unknown 0x04 = stop 0x05 = operational 0x6A = slave is detected, but not yet initialized (bootup) 0x7E = failure of the slave 0x7F = preoperational
<b>Emergency</b>	Data of the last emergency frame
<b>SDO Idx, Sub Idx, SDO Abort</b>	Last received SDO abort code

Updating of the slave status list can be halted to copy the list into the clipboard. The text can then be copied, for example, into an e-mail or documentation.

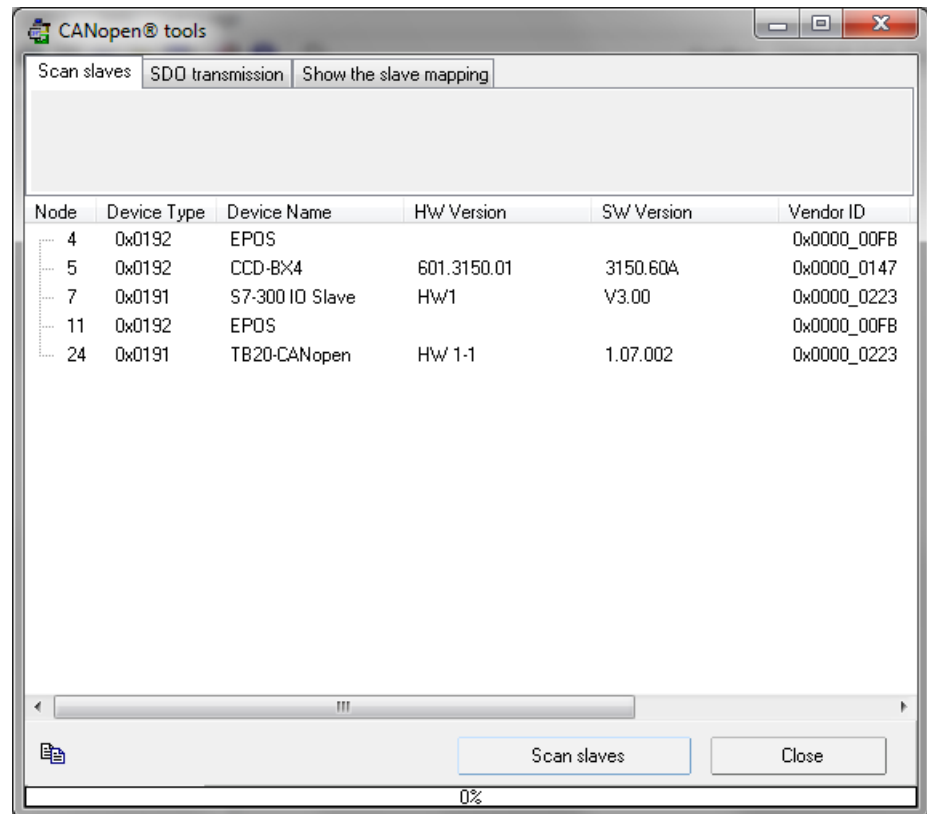
## 6.8 CANopen® Tools

To make troubleshooting easier, it is possible query or parameterize any CANopen® slave using SDO communication via the menu item “CANopen® Tools.” CANopen® Tools requires a USB link with the module. The module must be parameterized for CANopen® Tools in CANopen® master mode.

### 6.8.1 Scan slaves

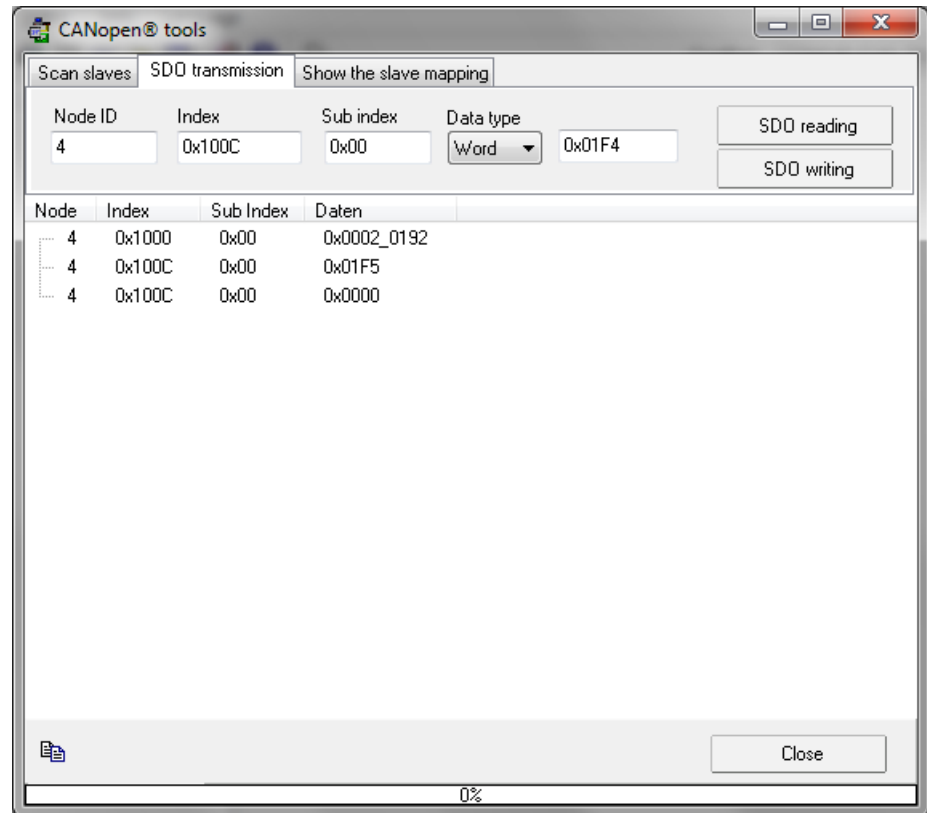
With “Scan slave,” it is possible to search for CANopen® slaves using the CAN300 PRO module. The slaves must respond to the query of the object 0x1000. Otherwise, they will not be recognized.

**!**  
*The slaves must have the object 0x1000.*



## 6.8.2 SDO transmission

SDO transmission can be used to read out or write the SDO objects of a slave. For this purpose, the corresponding property of the object must be available on the slave addressed.



**Node ID:** The CAN ID of the slave to be addressed,

**Index:** Index of the object on the slave to be read or written.

**Sub Index:** The subindex of the object on the slave to be read or written.

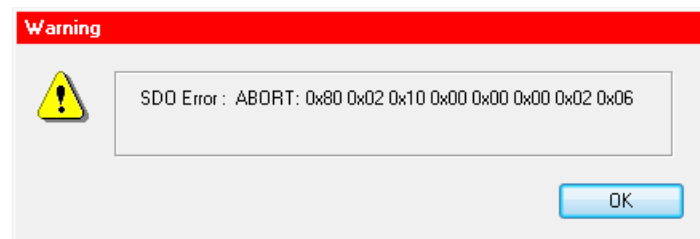
**Data type:** Is the data size (byte, word, double) that is defined for this object in the slave.

**Write data:** Is the value to be written into the object. If data are to be read out of the object, "write data" will simply be deactivated.

**Transmit SDO:** With this button, the SDO is transmitted to the module and therefore to the slave. If the slave transmits an abort as its response, this will be displayed as an error message with the abort code of the slave.



*If the information is incorrect, an error message is output with the abort code of the slave!*

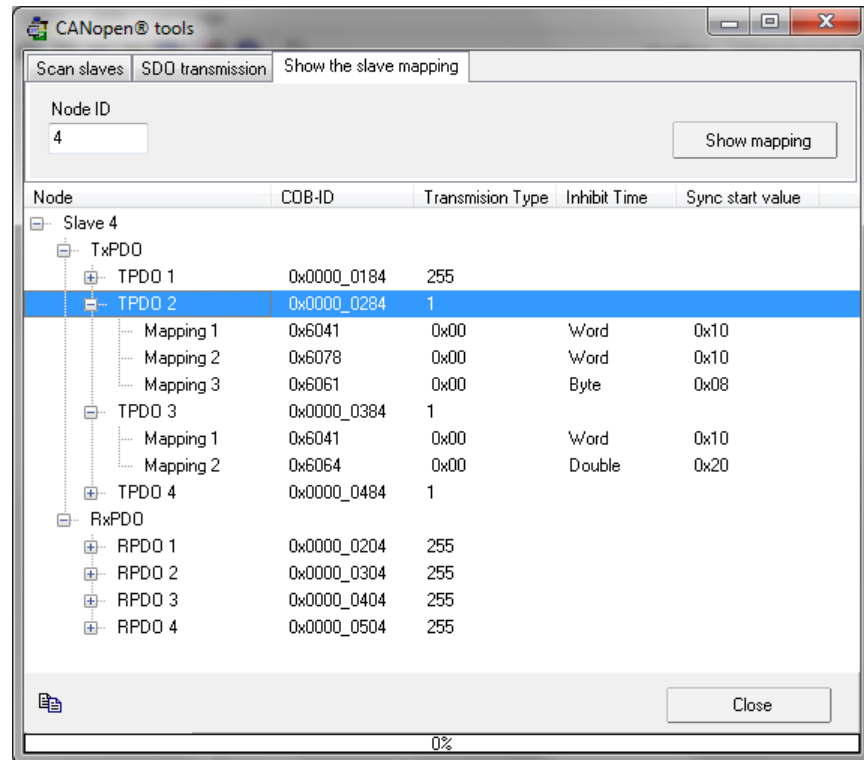


### 6.8.3 Slave mapping

With “Show the slave mapping,” the mapped PDOs of the addressed slave are queried and displayed.

With this function it is possible to read out the default mapping if a unknown device to be able to set the PDOs in the CANopen® project.

It is also possible to check after start-up of the master whether the required mapping has been set.



**Node ID:** Is the slave ID from which the PDO mapping data is to be fetched.

**Show mapping:** If the button is clicked, the module starts reading out the mapping from the slave and if at least one mapped PDO is present, it will be shown in the window.

## 6.9 Slave PDO distribution in the PLC I/O buffer

To simplify troubleshooting, it is possible to show the current distribution of the active PDOs of the slaves in the PLC I/O buffer using the graphical overview.

With the menu item “Slave PDO distribution,” a tabular image of the assignment of the I/O buffer with the PDOs of the slaves and their overlaps is created and displayed.

The screenshot shows a window titled "PDO allocation in the io range" with two main sections: TPDOs (IB 50-110) and RPDOs (QB 50-110). Each section contains a table with columns for IN/OUT, Node, Device name, RPDO, and four bit positions (0, 1, 2, 3, 4). The status of each bit is indicated by a shaded cell.

TPDOs IB 50 - 110						RPDOs QB 50 - 110											
IN	Node	Device name	TPDO	0	1	2	3	4	OUT	Node	Device name	RPDO	0	1	2	3	4
IB 50, IB 51	4	B+R X20	1						QB 50, QB 51	4	B+R X20	1					
IB 52 - 55									QB 52 - 55								
IB 56, IB 57	7	Beckhoff	1						QB 56, QB 57	7	Beckhoff	1					
IB 58 - 59									QB 58 - 59								
IB 60, IB 61	3	Wago 752	1						QB 60, QB 61	3	Wago 752	1					
IB 62	64	WAGO 750	1						QB 62	64	WAGO 750	1					
IB 63 - 70									QB 63 - 70								
IB 71 - 73									QB 71 - 73								
IB 74, IB 75	2	Phoenix ILB	1						QB 74, QB 75	2	Phoenix ILB	1					
IB 76 - 79									QB 76 - 79								
Iw 80	1	Maxon EPOS	1						Qw 80	1	Maxon EPOS	1					
Iw 82, IB 84	1	Maxon EPOS	2						Qw 82, QB 84	1	Maxon EPOS	2					
IB 85 - 89									QB 85 - 89								
Iw 90, ID 92	1	Maxon EPOS	3						Qw 90, QD 92	1	Maxon EPOS	3					
IB 96 - 99									QB 96 - 99								
Iw 100, ID 102	1	Maxon EPOS	4						Qw 100, QD 102	1	Maxon EPOS	4					
IB 106 - 109									QB 106 - 109								

To update the display after changes, close this window and re-open it.



## 7 Programming in the PLC

### 7.1 Overview

The programming of the CAN 300 PRO module is performed in the PLC via the data handling blocks contained in the software package, which have to be purchased separately but only once.

Data handling blocks are available for layer 2 communication (no interpretation of the CAN frames), for the SAE J1939 protocol, and for CANopen® master operation.

Data handling blocks for use of the CAN 300 PRO as a CANopen® slave or as a DeviceNet-Slave are available on request.

The choice of handling blocks and the configuration of the CAN 300 PRO module must match.

For Layer 2 and SAE J1939 data handling blocks, a Layer 2 Project must be imported into the CAN 300 PRO module (see Sec. 6.3).

For use of the CANopen® master data handling blocks, the CAN 300 PRO can be configured as a CANopen® master (see Sec. 6.4).

CANopen handling blocks version 2.0 and higher need the system handling blocks SFB 52 and 53. Call the support for PLCs which does not have these system handling blocks.



*The data handling blocks and configuration of the CAN 300 PRO must match!*

### 7.2 The right handling blocks

It is important to choose the appropriate handling blocks. The examples are categorized into project type and the engineering tool in use. Only the internal functions of the handling blocks are different, but the interface is identical.

Application Engineering tool	Rack or expansion rack	Distributed I/Os
Step 7 up to V5.5	C3PRO_CANopen_Vxx C3PRO_Layer2_Vxx C3PRO_SAE_J1939_Vxx C3PRO_IO_Slave_Vxx	C3PRO_CANopen_DP_Vxx C3PRO_Layer2_DP_Vxx C3PRO_SAE_J1939_DP_Vxx C3PRO_IO_Slave_DP_Vxx
Step 7 V11 „TIA Portal“	C3PRO_CANopen_TIA_Vxx C3PRO_Layer2_TIA_Vxx C3PRO_SAE_J1939_TIA_Vxx C3PRO_IO_Slave_TIA_Vxx	C3PRO_CANopen_DP_TIA_Vxx C3PRO_Layer2_DP_TIA_Vxx C3PRO_SAE_J1939_DP_TIA_Vxx C3PRO_IO_Slave_DP_TIA_Vxx

## 7.3 Layer 2 handling blocks

### 7.3.1 General



*If layer 2 data handling blocks are used, the CAN 300 PRO module must also be parameterized with a layer 2 project.*

With the layer 2 data handling blocks it is possible to process any CAN protocol. The CAN frames are received without being interpreted, provided to the PLC, and any frames can be transmitted.

The following FBs are available in layer 2:

<b>FB 65</b>	<b>CANSEND</b>	Transmission of a CAN frame
<b>FB 66</b>	<b>CANRCV</b>	Receiving a CAN frame
<b>FB 67</b>	<b>CANCTRL</b>	Module management functions

Initialization of the module in the start-up OBs is not necessary. The module starts automatically if the PLC is switched to RUN and stops if the PLC goes into the STOP state.

Here is an example of a call:

```
CALL FB 66 , DB66      CANRCV
Base   :=256
IDHI   :=MW60
IDLO   :=MW62
RTRLEN:=MB64
DW0    :=MW70
DW1    :=MW72
DW2    :=MW74
DW3    :=MW76
STAT   :=MW78
Error  :=M6.1
RetVal:=MW82
Recd   :=M6.0

AN     M      67.0
JC     send
...

...

SET
=      M      10.0           // Senden aktivieren
...

CALL FB 65, DB65      CANSEND
Base   :=256
IDHI   :=W#16#0
IDLO   :=MW80
RTRLEN:=MB64
DW0    :=MW4
DW1    :=MW72
DW2    :=MW74
DW3    :=MW76
STAT   :=MW78
Error  :=M10.1
RetVal:=MW82
Snd    :=M10.0

...

BE
```

### 7.3.2 FB 65 CANSEND

The CANSEND function block (FB 65) transfers a CAN frame to the module from which it is transmitted immediately.

Parameter	Direction	Type	Example
Base	IN	INT	256
IDHI	IN	WORD	W#16#0
IDLO	IN	WORD	MW 80
RTRLEN	IN	BYTE	MB 64
DW0	IN	WORD	MW 4
DW1	IN	WORD	MW 72
DW2	IN	WORD	MW 74
DW3	IN	WORD	MW 76
STAT	OUT	WORD	MW 78
Error	OUT	BOOL	M10.1
RETVAL	OUT	INT	MW 82
Snd	IN/OUT	BOOL	M 10.0

As the passed parameters, the base address of the module must be passed as an integer number (*Base*), a status byte (*STAT*), and a bit for transmit enable (*Snd*).

The word *RTRLEN* contains the number of data bytes (0..8) in the lower 4 bits (bit 0 to bit 3). Bit 6 is the RTR bit of the CAN frame. Setting bit 7 transmits the frame as a high-priority message.

The bit *Snd* is always reset after the block has been executed, if the frame to be transmitted has been transferred to the module. If the transmit buffer in the module is full, older frames that have not been transmitted yet are deleted.

Parameter *STAT* contains the status of the CAN 300 PRO module (see Sec. 7.3.8). The parameter is always assigned value, even if the *Snd* bit is not set.

If Timer 0 has been set as the synchronous timer, the data is only ever transmitted in a defined synchronous time window.

Assignment of parameters *ID-HI* and *ID-LO*:



*The FB65 "CANSEND" must not be called in OB 1 (cycle) and OB 35 (time OBs) simultaneously or mixed!*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	COB-ID 11 Bit													
0	0	0	COB-ID 29Bit																															
<b>ID-HI</b>														<b>ID-LO</b>																				

### 7.3.3 FB 66 CANRCV

The CANRCV function block (FB 66) transfers a CAN frame from the module into the PLC, if a frame has been received and let through by the acceptance filter.

Parameter	Direction	Type	Example
Base	IN	INT	256
IDHI	OUT	WORD	MW60
IDLO	OUT	WORD	MW62
RTRLEN	OUT	BYTE	MB64
DW0	OUT	WORD	MW70
DW1	OUT	WORD	MW72
DW2	OUT	WORD	MW74
DW3	OUT	WORD	MW76
STAT	OUT	WORD	MW 78
Error	OUT	BOOL	M 6.1
RETVAL	OUT	INT	MW 82
Recd	INOUT	BOOL	M 6.0

As the passed parameter, the base address of the module must be passed as an integer number (Base).

The elements of the frame are passed as data words (IDHI, IDLO, RTRLEN, DW0...3).

The word RTRLEN contains the number of data bytes (0...8) in the lower 4 bits (bit 0 to bit 3). Bit 6 is the RTR bit of the CAN frame. If bit 7 is set, the message went via high-priority FIFOs.

If the function block has read a frame from the CAN 300 PRO module, bit Recd is set.

Parameter STAT contains the status of the CAN 300 PRO module (see Sec. 7.3.8). The parameter is always assigned a value even if no frame has been received.

Assignment of parameters ID-HI and ID-LO:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	COB-ID 11 Bit										
0	0	0	COB-ID 29Bit																												
ID-HI														ID-LO																	

### 7.3.4 FB 67 CANCTRL

The function block CANCTRL (FB 67) provides special functions of the CAN 300 PRO module.

Parameter	Direction	Type	Example
Base	IN	INT	256
Func	IN	INT	MW90
Param	IN	INT	MW92
STAT	OUT	WORD	MW 78
RETVAL	OUT	INT	MW 84
Error	OUT	BOOL	M 8.1

As the passed parameters, the base address of the module must be passed as an integer number (*Base*), a function code (*Func*), and, if necessary, a parameter (*Param*).

The following functions are available:

- 1 = Start a timer (timer number 1-16 in *Param*)
- 2 = Stop a timer (timer number 1-16 in *Param*)
- 5 = Trigger CAN controller reset and re-init;  
Caution: CAN frame may be lost
- 6 = Clear all FIFOs on the module
- 7 = Reset FIFO error bits

Parameter *STAT* contains the status of the CAN 300 PRO module (see Sec. 7.3.8). The parameter is always assigned a value even if no frame has been received.

### 7.3.5 FB 68 CANSEND Block Mode

The CANSEND Block Mode function block (FB 68) transfers up to 17 CAN frames to the module. The module will send the frames immediately.

Parameter	Richtung	Typ	Beispiel
Base	IN	INT	256
IDHI	IN	WORD	W#16#0
IDLO	IN	WORD	MW 80
RTRLEN	IN	BYTE	MB 64
DW0	IN	WORD	MW 4
DW1	IN	WORD	MW 72
DW2	IN	WORD	MW 74
DW3	IN	WORD	MW 76
Res_Snd	IN	BOOL	M10.2
Message_Count_to_Snd	IN	BYTE	B#16#11
STAT	OUT	WORD	MW 78
Error	OUT	BOOL	M10.1
RETVAL	OUT	INT	MW 82
Snd	IN/OUT	BOOL	M 10.0

As the passed parameters, the base address of the module must be passed as an integer number (*Base*), a status byte (*STAT*) and a bit for enabling the sending (*Snd*).

The byte *Message\_Count\_to\_Snd* contains the number of the CAN-frames to be sent. The frames are stored in a common data block.

The bit *Res\_Snd* will initialize the handling block and all stored CAN-frames until this command will be discarded.

The word *RTRLEN* contains the number of data bytes (0...8) in the lower 4 bits (bit 0 to bit 3). bit 6 is the RTR bit of the CAN frame. Setting bit 7 transmits the frame as a high-priority message.

The bit *Snd* is always reset after the block has been executed, if the frame to be transmitted has been transferred to the module. If the transmit buffer in the module is full, older frames that have not been transmitted yet are deleted.

Parameter *STAT* contains the status of the CAN 300 PRO module (see Sec. 7.3.8). The parameter is always assigned value, even if the *Snd* bit is not set.

If Timer 0 has been set as the synchronous timer, the data is only ever transmitted in a defined synchronous time window.

Assignment of parameters *ID-HI* and *ID-LO*:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	COB-ID 11 Bit										
0	0	0	COB-ID 29Bit																												
ID-HI													ID-LO																		



*The block mode is to activate in the CANParam project and it is not allowed to use this mode with the standard-handling blocks CANSEND/CANRCV (FB65/66)!*



*The FB68 "CANSEND block mode" must not be called in OB 1 (cycle) and OB 35 (time OBs) simultaneously or mixed!*

### 7.3.6 FB 69 CANRCV Block Mode

The CANRCV Block Mode function block (FB 69) transfers up to 17 CAN frames from the module into the PLC, if at least a frame has been received and let through by the acceptance filter.

Parameter	Richtung	Typ	Beispiel
Base	IN	INT	256
IDHI	OUT	WORD	MW60
IDLO	OUT	WORD	MW62
RTRLEN	OUT	BYTE	MB64
DW0	OUT	WORD	MW70
DW1	OUT	WORD	MW72
DW2	OUT	WORD	MW74
DW3	OUT	WORD	MW76
Message_Count	OUT	BYTE	MB66
STAT	OUT	WORD	MW 78
Error	OUT	BOOL	M 6.1
RETVAL	OUT	INT	MW 82
Recd	INOUT	BOOL	M 6.0



The block mode is to activate in the CANParam project and it is not allowed to use this mode with the standard-handling blocks CANSEND/CANRCV (FB65/66)!

As the passed parameter, the base address of the module must be passed as an integer number (Base).

The elements of the frame are passed as data words (IDHI, IDLO, RTRLEN, DW0...3).

The word RTRLEN contains the number of data bytes (0...8) in the lower 4 bits (bit 0 to bit 3). Bit 6 is the RTR bit of the CAN frame. If bit 7 is set, the message went via high-priority FIFOs.

If the function block has read at least one frame from the CAN 300 PRO module, bit Recd is set.



The FB69 "CANRCV block mode" must not be called in OB 1 (cycle) and OB 35 (time OBs) simultaneously or mixed!

The byte Message\_Count contains the number of CAN-frames are stored in the handling block. The next CAN-frame(s) will be transmitted (if available) from the CAN 300 PRO module if all stored CAN-frames has been taken out from the internal frame storage.

Parameter STAT contains the status of the CAN 300 PRO module (see Sec. 7.3.8). The parameter is always assigned a value even if no frame has been received.

Assignment of parameters ID-HI and ID-LO:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	COB-ID 11 Bit										
0	0	0	COB-ID 29Bit																												
ID-HI														ID-LO																	

### 7.3.7 FB 64 CANSendTimer

The CANSendTimer function block (FB 64) transmits a CAN-frame to one of the programmed CAN-timers. The data is transferred to the corresponding timer and will be sent with the next timer event.

Parameter	Richtung	Typ	Beispiel
Base	IN	INT	256
Timer_No	IN	INT	MW94
IDHI	IN	WORD	MW96
IDLO	IN	WORD	MW 98
RTRLEN	IN	BYTE	MB 86
DW0	IN	WORD	MW 116
DW1	IN	WORD	MW 118
DW2	IN	WORD	MW 120
DW3	IN	WORD	MW 122
STAT	OUT	WORD	MW 124
Error	OUT	BOOL	M87.1
RETVAL	OUT	INT	MW 126
Snd	IN/OUT	BOOL	M 87.0

As the passed parameters, the base address of the module must be passed as an integer number (*Base*), a status byte (*STAT*) and a bit to release the storing /sending (*Snd*).



*The FB64 „CANSendTimer“ must not be called in OB 1 (cycle) and OB 35 (time OBs) simultaneously or mixed!*

The word *RTRLEN* contains the number of data bytes (0..8) in the lower 4 bits (bit 0 to bit 3). Bit 6 is the RTR bit of the CAN frame. Setting bit 7 transmits the frame as a high-priority message.

The bit *Snd* is always reset after the block has been executed and the frame has been transferred to the module. If the transmit buffer in the module is full, older frames that have not been transmitted yet are deleted.

Parameter *STAT* contains the status of the CAN 300 PRO module (see Sec. 7.3.8). The parameter is always assigned value, even if the *Snd* bit is not set.



*The CANID has to be the same like the corresponding projected CAN ID in this timer number!*

The CAN ID contains *ID-HI* and *ID-LO* and it has to be the same like the CAN ID in the projected timer. Otherwise it can be sent frames with in consist data occurs.

If Timer 0 has been set as the synchronous timer, the data is only ever transmitted in a defined synchronous time window.

Assignment of parameters *ID-HI* and *ID-LO*:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	COB-ID 11 Bit										
0	0	0	COB-ID 29Bit																												
ID-HI											ID-LO																				



### 7.3.8 Parameter STAT

The STAT parameter has the same meaning in all data handling blocks and indicates the status of the module:

Bit 15	Bit 14	Bit 13	Bit 12
CAN controller group error	Module is CAN 300 PRO (always 1)		
Bit 11	Bit 10	Bit 9	Bit 8
			Module running, read-in of the parameters completed

Bit 7	Bit 6	Bit 5	Bit 4
Send-FIFO (high) half full	Send-FIFO (high or low) overflow	Send-FIFO (low) half full	Receive-FIFOs (high & low) completely empty
Bit 3	Bit 2	Bit 1	Bit 0
Receive-FIFO (high) half full	Receive-FIFO (high or low) overflow	Receive-FIFO (low) half full	Receive-FIFOs (high & low) completely empty

The STAT parameter corresponds to the I/O input bytes 0 and 2.



CiA = CAN in Auto-  
mation e.V.,  
www.can-cia.org

## 7.4 CANopen®

### 7.4.1 General

The CANopen® protocol is a layer 7 protocol (application layer) based on the CAN bus (ISO 11898). Layer 1 and 2 (physical layer and data link layer) of the CAN bus are not affected.

The CANopen® communication profiles for the various applications are managed by the CIA (CAN in Automation e.V.).

The services elements provided by the application layer permit implementation of an application distributed over the network. These service elements are described in “CAN Application Layer (CAL) for Industrial Applications.”

The 11 bit identifier and the 8 data bytes of a CAN layer 2 frame have a fixed meaning. Every device in a CANopen® network has a fixed node ID (module number, 1-127).

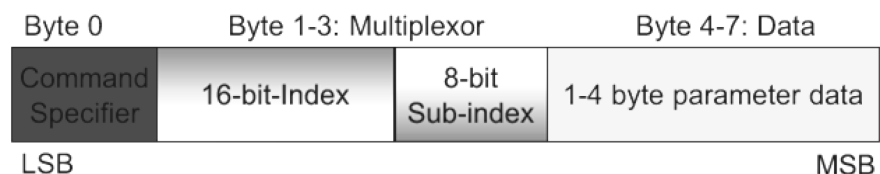
### 7.4.2 Objects

Data exchange with a CANopen® slave is performed either using permanently defined service data objects (SDO) or using freely configurable process data objects (PDO).

Each CANopen® slave has a fixed list of SDOs that are addressed by an object number (16 bits) and an index (8 bits).

*Example:* Object 0x1000/ Index 0 = Device Type, 32Bit Unsigned SDOs with a width of 8/16/32 bits can be read and written with a CANopen® frame. SDOs that are longer are transmitted in more than one frame.

SDOs can be processed as soon as a CANopen® slave is ready for operation. For the SDOs, only the COB ID functions “SDO request” or “SDO response” are available. The object number, access mode, and type are stored in the first 4 bytes of the CAN frame. The last 4 bytes of the CAN frame then contain the value for the SDO.





Each CANopen slave should have a directory containing the objects it supports.

PDOs contain the “working values” of a CANopen® slave for cyclic process operation. Each CANopen® slave can manage several PDOs (normally up to 4 for transmission and 4 for receiving).

Each of the existing PDOs has its own COB-ID. It is possible to map any information of the CANopen® slave to the 8 data bytes of the frame for reading and writing. The values from the object directory (SDOs) are always mapped.

The PDOs are automatically mapped from most CANopen® slaves on startup. The assignment can be changed using certain SDOs.

### 7.4.3 Functions

The CANopen® functions are divided into the following basic groups:

- Reading and writing SDO
- Reading and writing PDO
- Network management
- Emergency messages

The function code is stored in the upper 4 bits of the identifier. Together with the node ID this makes up the COB identifier.

COB identifier (COB-ID):

10	9	8	7	6	5	4	3	2	1	0
Function				Node ID						



It is possible to change some COB-IDs to other values using special service data objects (SDOs).

This is NOT supported by the CANopen data handling!

Broadcast functions:

Function	Function code (binary)	Resulting COB-ID
NMT	0000	0h
SYNC	0001	80h
TIME STAMP	0010	100h


Node functions:

Function	Function code (binary)	Resulting COB-ID
EMERGENCY	0001	81h – FFh
PDO1 (tx)	0011	181h – 1FFh
PDO1 (rx)	0100	201h – 27Fh
PDO2 (tx)	0101	281h – 2FFh
PDO2 (rx)	0110	301h – 37Fh
PDO3 (tx)	0111	381h – 3FFh
PDO3 (rx)	1000	401h – 47Fh
PDO4 (tx)	1001	481h – 4FFh
PDO4 (rx)	1010	501h – 57Fh
SDO (tx)	1011	581h – 5FFh
SDO (rx)	1100	601h – 67Fh
NMT Error Control	1110	701h – 77Fh



"Tx" = is transmitted by the slave  
"Rx" = is transmitted by the slave

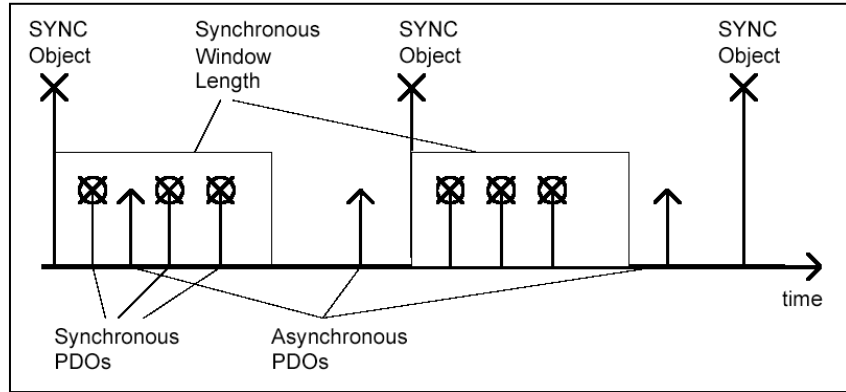
## 7.4.4 Network management


 The SYNC frame can be implemented using a timer with the CAN 300 module.

### SYNC:

The SYNC frame is a cyclic “broadcast” frame and sets the basic bus clock. To ensure isosynchronism, the SYNC frame has a high priority.

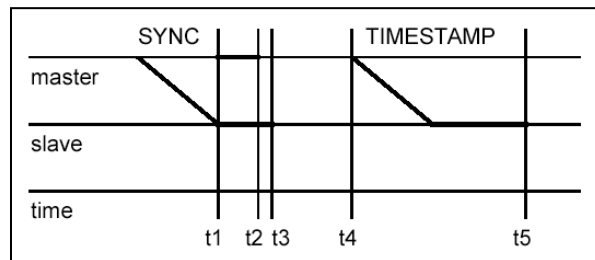
[COB-ID: 80h]



 The time stamp frame can be implemented using a timer with the CAN 300 module.

### Time Stamp:

The time stamp frame is a cyclic “broadcast” frame and provides the system time. The time stamp frame is usually transmitted directly after a SYNC frame and then provides the system time of the SYNC frame.



To ensure a precise transmission, the time stamp frame has a high priority.

[COB-ID: 100h]

### Nodeguarding:

With the nodeguarding function, the master monitors the CANopen® slave modules by transmitting frames cyclically to each slave. Each CANopen® slave must respond to the nodeguarding frame with a status frame.

The control can detect failure of a CANopen® slave using nodeguarding.

[COB-ID: 700h + Node-ID + 1Byte data with status of the slave]

### *Lifeguarding:*

In lifeguarding, each CANopen<sup>®</sup> slave continuously monitors whether the master is performing nodeguarding once it has been started within certain time limits. If the nodeguarding frame of the master fails, the distributed I/O module can detect that using lifeguarding and, for example, put all outputs into the safe state.

Nodeguarding and lifeguarding should always run together.

### *Heartbeat:*

Heartbeat monitoring is equivalent to nodeguarding although no request frames are generated by CANopen<sup>®</sup> master. The heartbeat frame is transmitted automatically by the node and can be evaluated in the master. [COB-ID: 700h + Node-ID + 1Byte data with status of the slave]

### *Emergency message:*

If a fault occurs on a CANopen<sup>®</sup> slave, for example, the lifeguarding timer elapses, it transmits an emergency message on the bus.


[COB-ID: 80h + Node-ID]

All stations can perform an emergency stop on receiving an emergency frame, for example.

### *BootUp message:*

CANopen<sup>®</sup> slaves generate a boot-up message after switch-on that the master can recognize to initialize this new station.

[COB-ID: 700h + node ID + 1 byte data: 00h]

  
*Some CANopen slave modules generate special emergency messages on switch-on or switch-off.*

## 7.5 Start-up behavior of the CANopen® master

The master features automatic start-up behavior to initialize the configured slaves correctly and get them running.

1. An NMT reset command is sent to all slaves and the boot-up messages is awaited. The waiting time is settable.
2. All slaves are put in pre-operational by NMT command (without checking).
3. An NMT guard request is transmitted to those slaves to query the current status of the slaves.
4. The SDO 0x1000 (device type) is read from each slave and compared with the slave settings.

*If the mandatory devices are no available at this point or do not have the right device type, the procedure will start at step 1 again.*

5. The configured initialization SDOs are transmitted to each slave. Abort codes do not result in cancellation

*If the master option "Wait for PLC start" is selected, the NMT command 20 is awaited at this point.*

6. NMT start is transmitted to all slaves
7. An NMT guard request is transmitted to those slaves to query the current status of the slaves
8. Cyclic operation is started. The IO input image is transmitted to the PLC (possibly after the blocking time).

If a slave fails in cyclic operation and the master option "Autostart of the slaves" is active, this slave is automatically re-initialized and started.

If the failed slave is a mandatory device, a master restart is performed and the procedure restarts at step 1.

## 7.6 Operating conditions for CANopen® slave devices

To be able to use a CANopen® slave on the CAN 300 PRO in CANopen® master operation, the slave must meet the following conditions:

1. It must be possible to read out the SDO 0x1000.
2. The slave should perform a restart after the NMT reset command.
3. After the slave has been reset, it should transmit a boot-up message (0x700 + node ID; data: 0x00)
4. If monitoring is activated by the master, the slave should support either nodeguarding or producer heartbeat.

Please use the device manual to check whether the conditions stated above have been met or speak to your device manufacturer.

If a device to be used does not meet these conditions, some functional restrictions may result, or the device cannot be used with the CAN 300 PRO via CANopen®.

Alternatively, such a device can be used in Layer 2 mode of the CAN 300 PRO. Please contact us about this ...

## 7.7 Tips on start-up / troubleshooting

To facilitate start-up of a CANopen® network, the following tips must be taken into account:



*For initial start-up as the CANopen master, please also use the document “Start-Up Guide CANopen for CAN 300 PRO.”*

- If the LED BF “CAN bus error” is still lit or blinking, check the physical structure of the CAN bus (terminating resistor, baud rate, etc.). Use layer 2 debug display (Sec. 6.7.1).
- If the SF “system errors” LED lights up, there is a defective MMC in the module, or the imported CAN project has an internal error.
- Check which “device monitoring” method (heartbeat or nodeguarding) is supported by the slave in question. Do not use “Consumer Heartbeat” until everything else has been started up.
- On all devices, do not yet activate the option “mandatory device” so that the master will definitely start up and show all the stations found in the slave list of the CANopen® debug display (Sec. 6.7.2). These stations can be defined as a mandatory device if necessary.
- Initially set “Wait after reset” to a long time (e.g. 10 seconds) if the master displays all slaves as “operational” for a shorter time, this time can be reduced.
- If the slave device fulfills all requirements from Section 7.6?
- If a slave is displayed as operational after the master has started up, but no data is being received, check the PDO mapping and the assignment of the PDOs in the PLC I/O buffer of the PLC (see Sec. 6.8.3)
- Check whether an error has occurred during SDO initialization of the slaves: CANopen® debug display/SDO abort code display

You will find detailed instructions for initial start-up of a CANopen® network in the document “Start-Up Guide for CANopen® with CAN 300 PRO” on the CD or on the Web site of Systeme Helmholtz.

If these tips do not help, contact our support per phone or per e-mail ([support@helmholtz.de](mailto:support@helmholtz.de)).

If information about their slave device is also at hand (manual), please also send this information.





*If the CANopen master data handling blocks are used, the CAN 300 PRO module must also be parameterized with a CANopen master project.*

## 7.8 CANopen® data handling modules

The data handling blocks for CANopen® communication provide all the necessary functions to replace the process image, to process SDOs, and to perform network management.

These data handling blocks can only be used if the CAN 300 PRO module has also been parameterized in CANopen® master mode, see Section 6.4 “Creation of a CANopen® project”



*The CANopen data handling blocks should not be called up together with layer 2 data handling blocks!*

Block	Name	Function	Section
FB 20	CANopen® IO Read	Read the I/O data from the module	7.8.1
FB 21	CANopen® IO Write	Transmit the I/O data to the module	7.8.2
FB 22	CANopen® Service	Fetch livelist, receive emergency frames	7.8.3
FB 23	CANopen® NMT	Network management for masters and slaves	7.8.4
FB 24	CANopen® SDO	SDO orders up to 4 bytes	0
FB 27	CANopen® SDO segmented	SDO orders of any length	7.8.6
FB 25	CANopen® L2 Receive	Receive layer 2 frames	7.8.7
FB 26	CANopen® PDO resend	Transmit a PDO again with the same data	7.8.8
FB 28	SYNC Trigger	Transmit PDOs with SYNC	7.8.9

Blocks that are no longer required can be removed from the project to save storage space. The blocks do not call each other.

The SDO-FBs 24 and 27 must never be activated simultaneously.



If the CANopen handling blocks are used, the CAN 300 PRO module have to be projected with a CANopen master project!

### 7.8.1 FB 20 CANopen® IO Read

With this FB, the current data of the PLC IN buffer are read into the PLC. The FB should be called only once at the beginning of the OB cycle.

Parameter	Direction	Type	Example
Base	IN	INT	256
Dest	IN	ANY	P#E 50.0 BYTE 100
STAT	OUT	WORD	MW 20
Err	OUT	BOOL	M 22.6
RetVal	OUT	INT	MW 24
NewData	OUT	BOOL	M 22.0

- Base Address of the CAN 300 PRO module
- Dest ANY pointer to the target area for the input buffer in the PLC. The data are copied from the module into this area up to the maximum number of bytes specified. The size of the target area should match the size specified in the CANopen® master project.
- STAT Status of the module, see Section 7.3.8
- Err Error bit is 0, on successful implementation
- RetVal Error number, see Section 8
- NewData Bit is 1 if new data are successfully fetched from the CAN 300 PRO module.

*Example of call:*

```
CALL FB 20 , DB20
Base :=256
Dest :=P#E 50.0 BYTE 60
STAT :=MW20
Err :=M22.6
RetVal :=MW24
NewData:=M22.0
```

*Examples of the ANY pointer:*

P#E 50.0 BYTE 100 copies the input data into the I/O input image of the PLC from IB 50 to max. IB 149 (100 bytes). For this purpose, the maximum memory size of the I/O image of the selected PLC must be taken into consideration.

P#M 100.0 BYTE 50 copies the data into the marker memory area from MB 100 to max. MB 149 (50 bytes). For this purpose, the maximum memory size of the marker memory of the selected PLC must be taken into consideration.

P#DB 2.DBX 0.0 BYTE 300 copies the data into the data block 2 from DBB 2 to max. DBB 299 (300 bytes).

Section 6.4.5 contains a precise explanation of the PLC I/O buffer.

### 7.8.2 FB 21 CANopen® IO Write

With this FB, data are written out of the PLC into the PLC OUT buffer of the module. The FB should be called only once at the end of the OB cycle.

Parameter	Direction	Type	Example
Base	IN	INT	256
Dest	IN	ANY	P#DB3.DBX 0.0 BYTE 200
STAT	OUT	WORD	MW 20
Err	OUT	BOOL	M 22.7
RetVal	OUT	INT	MW 24

Base	Address of the CAN 300 PRO module
Source	ANY pointer to the target area for the output buffer in the PLC. The data are from the PLC into the module, no more than the specified number of bytes.
STAT	Status of the module, see Section 7.3.8
Err	Error bit is 0, on successful implementation
RetVal	Error number, see Section 8

*Example of call:*

```
CALL FB 21 , DB21
Base :=256
Source:=P#DB3.DBX0.0 BYTE 200
STAT :=MW20
Err :=M22.7
RetVal :=MW24
```

*Examples of the ANY pointer:*

P#A 50.0 BYTE 100 copies the output data into the I/O output image of the PLC from QB 50 to max. QB 149 (100 bytes). For this purpose, the maximum memory size of the I/O image of the selected PLC must be taken into consideration.

P#M 100.0 BYTE 50 Copies the data into the marker memory area from MB 100 to max. MB 149 (50 bytes). For this purpose, the maximum memory size of the marker memory of the selected PLC must be taken into consideration.

P#DB 3.DBX 0.0 BYTE 300 copies the data into the data block 3 from DBB 2 to max. DBB 299 (300 bytes).

Section 6.4.5 contains a precise explanation of the PLC I/O buffer.

### 7.8.3 FB 22 CANopen® Service

With this FB, emergency messages can be received and the livelist updated. The FB should be called cyclically in the PLC program.

Parameter	Direction	Type	Example
Base	IN	INT	256
Emergency	IN	ANY	P#M 92.0 BYTE 8
Livelist	IN	ANY	P#DB30.DBX0.0 BYTE 8
Node	OUT	INT	MW 90
STAT	OUT	WORD	MW 20
Err	OUT	BOOL	M 22.7
RetVal	OUT	INT	MW 24
New_Emergency	INOUT	BOOL	M 30.0
Req_Livelist	INOUT	BOOL	M 30.1
Busy_Livelist	INOUT	BOOL	M 30.2
New_Livelist	INOUT	BOOL	M 30.3

- Base Address of the CAN 300 PRO module
- Emergency ANY pointer to 8 bytes memory in which the data of the emergency frame are stored
- Livelist ANY pointer to the memory in which the data of the livelist are stored
- Node Node number for the emergency message
- STAT Status of the module, see Section 7.3.8
- Err Error bit is 0, on successful implementation
- RetVal Error number, see Section 8
- New\_Emergency If a new emergency message was received from the module, this bit will be set. The bit is only set for one PLC cycle
- Req\_Livelist Request bit for a livelist, is reset to 0 immediately after calling.
- Busy\_Livelist Flag indicating that the livelist is being awaited
- New\_Livelist Flag indicating whether the requested livelist has been received.

*Example of call:*

```
CALL FB 22 , DB22
Base      :=256
Emergency :=P#M 92.0 BYTE 8
Livelist  :=P#DB30.DBX0.0 BYTE 127
Node      :=MW90
STAT      :=MW20
Err       :=M30.7
RetVal    :=MW26
New_Emergency:=M30.0
Req_Livelist :=M30.1
Busy_Livelist:=M30.2
New_Livelist :=M30.3
```

Structure of the emergency message:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Emergency Error Code		Error Register (1001h)	Manufacturer-specific Error				

For the livelist, an ANY pointer must be transferred to a memory area that is at least as large as the highest projected node number.

The module provides the status of each node at the corresponding point in the memory relative according to the node number.

The following node states are indicated:

- 0x00 = unknown/not defined
- 0x04 = stop
- 0x05 = operational
- 0x6A = slave is detected, but not yet initialized (boot-up)
- 0x7F = preoperational
- 0x7E = failure of the slave

M	30.1	"Req_Livelist"	BOOL	false
M	30.2	"New_Livelist"	BOOL	true
DB30.DBB	1	"Livelist".Nodes[1]	HEX	B#16#00
DB30.DBB	2	"Livelist".Nodes[2]	HEX	B#16#05
DB30.DBB	3	"Livelist".Nodes[3]	HEX	B#16#7E
DB30.DBB	4	"Livelist".Nodes[4]	HEX	B#16#05
DB30.DBB	7	"Livelist".Nodes[7]	HEX	B#16#05

The example indicates the following states:

- Node 1: 0x00 = not defined
- Node 2: 0x05 = operational
- Node 3: 0x7E = failed
- Node 4: 0x05 = operational
- Node 7: 0x05 = operational

Failure detection of a slave by the master can only work if the device monitoring has been activated on the slave.

### 7.8.4 FB 23 CANopen® Netzmanagement

With this FB, network management functions can be executed both for individual slaves and for the master.

Parameter	Direction	Type	Example
Base	IN	INT	256
Node	IN	INT	1
Func	IN	INT	2
STAT	OUT	WORD	MW 20
Err	OUT	BOOL	M 22.7
RetVal	OUT	INT	MW 24

Base	Address of the CAN 300 PRO module
Node	Node number (1...127)
Func	Function code: 0 = Resume Node Control 1 = Start Node 2 = Stop Node 3 = Disconnect Node 4 = Node Enter Preoperational-State 5 = Reset Node 6 = Reset Communication of Node 12 = FIFO clear 13 = Clear Error 14 = CAN -Controller Reset 20 = CANopen® Master Start 21 = CANopen® Master Stop 22 = CANopen® Master Reset (complete restart of the master) 24 = CANopen® Master Run (enable of the master if the option "Wait for PLC start" has been selected)
STAT	Status of the module, see Section 7.3.8
Err	Error bit is 0 on successful implementation
RetVal	Error number, see Section 8

If one of the NMT commands (start, stop, preoperational, reset, disconnect) is used, this slave will be taken out of the master automatic. To have the slave taken over by the master again, the function code 0 is transmitted for the corresponding slave.

*Example of call:*

```
CALL FB 23 , DB23
Base :=256
Node :=1
Func :=2
STAT :=MW20
Err :=M126.7
RetVal:=MW124
```



*FB 24 must not be activated simultaneously with FB 27!*

### 7.8.5 FB 24 CANopen® SDO

With this FB, SDOs up to 4 bytes can be transmitted. If SDOs are more than 4 bytes are transmitted, FB 27 (see Section 7.8.6) must be used.

Parameter	Direction	Type	Example
Base	IN	INT	256
Node	IN	INT	MW 41
SDO_Request	IN	BYTE	MB 43
SDO_Index	IN	WORD	MW 44
SDO_Subindex	IN	BYTE	MB 46
SDO_Response	OUT	BYTE	MB 48
SDO_Abortcode	OUT	DWORD	MD 50
STAT	OUT	WORD	MW 20
RetVal	OUT	INT	MW 54
SDO_Data	INOUT	DWORD	MD 150
SDO_DataLen	INOUT	BYTE	MB 49
Activate	INOUT	BOOL	M 40.0
Busy	INOUT	BOOL	M 40.1
Err	INOUT	BOOL	M 40.7
Done	INOUT	BOOL	M 40.2

Base	Address of the CAN 300 PRO module
Node	Node number (1...127)
SDO_Request	SDO function: 0 = Reading the SDOs 1 = Writing the SDOs
SDO_Index	SDO index
SDO_SubIndex	SDO subindex
SDO_Response	Response according to execution of the SDO request: 0x40 = positive read response 0x80 = negative read response (→ abort code) 0x41 = positive write confirmation 0x81 = negative write confirmation (→ abort code)
SDO_Abortcode	Abort code on incorrect execution of the SDO request
STAT	Status of the module, see Section 7.3.8
RetVal	Error number, see Section 8
SDO_Data	Data for the SDO as a DWORD (right justified)
SDO_DataLen	Size of the SDO (1, 2, 4 bytes)
Activate	Bit for starting the request, is reset by the FB
Busy	Flag “request running”
Err	Error bit is 0 on successful implementation
Done	Flag “request complete” (with/without error)

To start a new SDO request, the “activate” bit must be set on calling the FB. The bit is reset and the “busy” bit is set as the flag for the current request. The FB must call up each PLC cycle until the “done” bit indicates that the request has been completed.

If it was not possible to complete the request successfully, this can be indicated by the bit "Err." Otherwise the return value SDO\_Response can be evaluated to check the success of the request.

*Example of call:*

```
SDO: CALL FB 24 , DB24
      Base      :=256
      Node      :=MW41
      SDO_Request :=MB43
      SDO_Index  :=MW44
      SDO_Subindex :=MB46
      SDO_Response :=MB48
      SDO_Abortcode:=MD50
      STAT      :=MW20
      RetVal    :=MW54
      SDO_Data  :=MD56
      SDO_DataLen :=MB49
      Activate  :=M40.0
      Busy      :=M40.1
      Err       :=M40.7
      Done      :=M40.2

A     M     40.1
JC    NEXT
A     M     40.2
A     M     40.7
JC    ERR
```





*FB 27 must not be activated simultaneously with FB 24!*

### 7.8.6 FB 27 CANopen® SDO Segmented

With this FB, SDOs more than 4 bytes in length can be transmitted. SDOs up to 230 bytes can be transmitted

Parameter	Direction	Type	Example
Base	IN	INT	256
Node	IN	INT	MW 41
SDO_Request	IN	BYTE	MB 43
SDO_Index	IN	WORD	MW 44
SDO_Subindex	IN	BYTE	MB 46
SDO_Data	IN	ANY	P#M 150.0 BYTE 20
SDO_Response	OUT	BYTE	MB 48
SDO_DataLen	OUT	BYTE	MB 49
SDO_Abortcode	OUT	DWORD	MD 50
STAT	OUT	WORD	MW 20
RetVal	OUT	INT	MW 54
Activate	INOUT	BOOL	M 40.0
Busy	INOUT	BOOL	M 40.1
Err	INOUT	BOOL	M 40.7
Done	INOUT	BOOL	M 40.2

Base	Address of the CAN 300 PRO module
Node	Node number (1...127)
SDO_Request	SDO function: 0 = Reading the SDOs 1 = Writing the SDOs
SDO_Index	SDO index
SDO_SubIndex	SDO subindex
SDO_Data	ANY pointer to the data for the SDO; the length of the SDOs to be transmitted at this point is defined. The maximum size is limited to 230 bytes.
SDO_Response	Response according to execution of the SDO request: 0x40 = positive read response 0x80 = negative read response (→ abort code) 0x41 = positive write confirmation 0x81 = negative write confirmation (→ abort code)
SDO_DataLen	Length in bytes of the received SDO data
SDO_Abortcode	Abort code on incorrect execution of the SDO request
STAT	Status of the module, see Section 7.3.8
RetVal	Error number, see Section 8
Activate	Bit for starting the request, is reset by the FB
Busy	Flag “request running”
Err	Error bit is 0 on successful implementation
Done	Flag “request complete” (with/without error)

To start a new SDO request, the “activate” bit must be set on calling the FB. The bit is reset and the “busy” bit is set as the flag for the current request.

The FB must call up each PLC cycle until the “done” bit indicates that the request has been completed.

If it was not possible to complete the request successfully, this can be indicated by the bit “Err.” Otherwise the return value SDO\_Response can be evaluated to check the success of the request.

*Example of call:*

```
SDO: CALL FB 27 , DB27
      Base      :=256
      Node      :=MW41
      SDO_Request :=MB43
      SDO_Index  :=MW44
      SDO_Subindex :=MB46
      SDO_Data   :=P#M 150.0 BYTE 20
      SDO_Response :=MB48
      SDO_DataLen :=MB49
      SDO_Abortcode:=MD50
      STAT      :=MW20
      RetVal    :=MW54
      Activate  :=M40.6
      Busy      :=M40.1
      Err       :=M40.7
      Done      :=M40.2

A      M      40.1
JC     L2RC
A      M      40.2
A      M      40.7
JC     ERR
```

### 7.8.7 FB 25 CANopen® L2 Receive

With this FB, all CAN frames can be received that are not processed by the CANopen® master.

Parameter	Direction	Type	Example
Base	IN	INT	256
L2Frame	IN	ANY	P#M 60.0 BYTE 14
STAT	OUT	WORD	MW 20
Err	OUT	BOOL	M 22.2
RetVal	OUT	INT	MW74
Recd	INOUT	BOOL	MW 22.1

Base	Address of the CAN 300 PRO module
L2Frame	Any pointer for the complete layer 2 frame
STAT	Status of the module, see Section 7.3.8
Err	Error bit is 0 on successful implementation
RetVal	Error number, see Section 8
Recd	This bit is 1 if a frame has been received

#### Allocation of the frame in the pointer L2Frame:

Byte 0:	0x12 (internal ID)
Byte 1:	RTR bit (bit 6), length (bits 0-3)
Byte 2-5:	11-bit identifier (bytes 4+5 are used)
Bytes 6-13:	Data bytes 1-8 of the CAN frame

#### *Example of call:*

```
CALL FB 25 , DB25
Base :=256
L2Frame:=P#M 60.0 BYTE 14
STAT :=MW20
Err :=M22.2
RetVal :=MW74
Recd :=M22.1
```

### 7.8.8 FB 26 CANopen® PDO resend

The CAN 300 PRO uses 4 methods to transmit PDOs to the slave (RPDO of the slave):

1. Transmission on change of the value (event-driven [255])
2. Transmission on every transfer to the module (event-driven on PLC cycle [254])
3. Transmission to SYNC frame
4. Transmission by SYNC trigger by the PLC (see also Sec. 7.8.9)

In addition to these methods, PDOs can be retransmitted to the slave in a targeted way and immediately using this FB. The module transmits a PDO frame with the currently available data.

Parameter	Direction	Type	Example
Base	IN	INT	256
Node	IN	INT	2
PDO	IN	INT	1
STAT	OUT	WORD	MW 20
Err	OUT	BOOL	M 136.7
RetVal	OUT	INT	M 134

Base	Address of the CAN 300 PRO module
Node	Node number (1...127)
PDO	Relative number of the PDO to be transmitted CAUTION: Only the defined PDOs are counted. Example: PDO1 and PDO4 are defined. PDO1 = 1, PDO4 = 2
STAT	Status of the module, see Section 7.3.8
Err	Error bit is 0 on successful implementation
RetVal	Error number, see Section 8

*Example of call:*

```
CALL FB 26 , DB26
Base :=256
Node :=2
PDO :=1
STAT :=MW20
Err :=M136.7
RetVal:=MW134
```

### 7.8.9 FB 28 CANopen® SYNC trigger

After the FB 28 has been called, the CAN 300 PRO module transmits a SYNC frame. If, in configured CANopen® slaves, RPDOs of type “SYNC triggered (0, 1-240)” are defined, these PDOs will be transmitted before the SYNC frame. FB 28 should always be called after FB 21 “IO-Write.”

Parameter	Direction	Type	Example
Base	IN	INT	256
STAT	OUT	WORD	MW 20
Err	OUT	BOOL	M 136.7
RetVal	OUT	INT	M 134

Base            Address of the CAN 300 PRO module  
 STAT           Status of the module, see Section 7.3.8  
 Err            Error bit is 0 on successful implementation  
 RetVal        Error number, see Section 8

*Example of call:*

```
CALL FB 28 , DB28
Base            :=256
STAT        :=MW20
Err         :=M35.7
RetVal:=MW36
```



*Calling CANopen FBs in alarm-OBs and in the cycle-OB at the same time, alarm-OBs have to be disabled!*

### 7.8.10 Using of alarm-OBs

If the CANopen FBs are called in alarm-OBs (e.g. cyclic in time-of-day-alarm-OBs) then is to regard that the interrupt of the alarm-OBs is disabled if the CANopen FBs in the cyclic is using at the same time.

If the execution of CANopen FB is interrupted of any alarm-OB included by calling CANopen FBs, transmitting problems can occur.

Using the SFC39 “DIS\_IRT” and SFC 40 “EN\_IRT” it is possible to disable/enable the alarm-OBs.

## 7.9 SDO abort codes

Below you will find typical error messages that can be generated by a CANopen® slave.

You will receive these error messages if you perform an SDO transmission (FB 24 or FB27).

Code	Meaning
0503 0000h	"Toggle bit" has not been alternated
0504 0000h	SDO protocol "time out "
0504 0001h	Client/server command designation not valid or unknown
0504 0002h	Unknown block size (block mode only)
0504 0003h	Unknown block number (block mode only)
0504 0004h	CRC error (block mode only)
0504 0005h	Outside the memory
0601 0000h	Access to this object is not supported
0601 0001h	Attempted read access to an object that can only be written
0601 0002h	Attempted write access to an object that can only be read
0602 0000h	Object does not exist in the object directory
0604 0041h	Object cannot be "mapped" to a PDO
0604 0042h	Size and number of "mapped" objects exceeds the possible PDO length
0604 0043h	General parameters -incompatibility
0604 0047h	General incompatibility in the device
0606 0000h	Access violation due to a hardware error
0607 0010h	Data type does not match, length of the service parameter does not fit
0607 0012h	Data type does not match, length of the service parameter too large
0607 0013h	Data type does not match, length of the service parameter too small
0609 0011h	Subindex does not exist
0609 0030h	Out of value range of the parameter (only for write accesses)
0609 0031h	Value of the parameter too large
0609 0032h	Value of the parameter too small
0609 0036h	Maximum value is smaller than the minimum value
0800 0000h	General error
0800 0020h	Data item cannot be transmitted or stored
0800 0021h	Data item cannot be transmitted/stored because of local device control
0800 0022h	Data item cannot be transmitted/stored because of device status
0800 0023 h	Dynamic generation of the object directory not possible or already exists

## 7.10 SAE J1939 communication

### 7.10.1 General



*For SAE\_J1939 handling, the CAN 300 module must be set to "29-bit mode"!*

With the SAE J1939 data handling blocks, it is possible to transmit and receive CAN frames according to SAE J1939. The SAE J1939 protocol always uses CAN identifiers with 29 bits. Please ensure that a relevant project has been imported into the CAN 300 PRO module with the setting "Layer 2 - 29 Bit Mode."

The following FBs are available for SAE J1939:

**FB 70 CANSEND\_SAE\_J1939** Transmission of a telegram

**FB 71 CANRCV\_SAE\_J1939** Reception of a telegram

Initialization of the module in the start-up OBs is not necessary. The module starts automatically if the PLC is switched to RUN and stops if the PLC goes into the STOP state.

Multipacket messages are currently not supported.

### 7.10.2 FB 70 CANSEND\_SAE\_J1939

The CANSEND\_SAE\_J1939 function block (FB 70) transfers a CAN frame to the module from which it is transmitted immediately.

Parameter	Direction	Type	Example
Base	IN	INT	256
Priority	IN	INT	7
DataPageBit	IN	BOOL	FALSE
PDU_Format	IN	INT	123
PDU_Specific	IN	INT	2
Source_Addr	IN	INT	1
B0	IN	BYTE	B#16#0
B1	IN	BYTE	B#16#0
B2	IN	BYTE	B#16#0
B3	IN	BYTE	B#16#0
B4	IN	BYTE	B#16#0
B5	IN	BYTE	B#16#0
B6	IN	BYTE	B#16#0
B7	IN	BYTE	B#16#0
STAT	OUT	WORD	MW 80
Error	OUT	BOOL	
RETVAL	OUT	INT	MW 82
Snd	IN/OUT	BOOL	M 87.0

As the passed parameters, the base address of the module must be passed as an integer number (Base) and a bit for transmit enable (Snd).

The bit Snd is always reset after the block has been executed. The frame to be transmitted is always transferred to the module. If the transmit buffer in the module is full, older frames that have not been transmitted yet are deleted. To prevent that, bit 4 of the STAT byte must always be queried before transmission.

Parameter STAT contains the status of the CAN 300 PRO module (see Sec. 7.3.8). The parameter is always assigned value, even if the Snd bit is not set.

Assignment of the parameters for the 29-bit CAN identifier according to the SAE J1939 protocol:



The FB70 "CANSEND" must not be called in OB 1 (cycle) and OB 35 (time OBs) simultaneously or mixed!

28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Priority	R	P	PGN (Parameter Group Number)													SA (Source Address)												
	R	P	PDU format						PDU-specific																			
	R	P	0...239: PDU1 Format						DA (Destination Address)																			
	R	P	240..255: PDU2 Format						Group Extension																			



### 7.10.3 FB 71 CANRCV\_SAE\_J1939

The function block CANRCV\_SAE\_J1939 (FB 71) transmits a CAN frame from the module to the PLC, if a frame has been received.

Parameter	Direction	Type	Example
Base	IN	INT	256
Priority	OUT	INT	MW60
DataPageBit	OUT	BOOL	MB68.0
PDU_Format	OUT	INT	MW62
PDU_Specific	OUT	INT	MW64
Source_Addr	OUT	INT	MW66
B0	OUT	BYTE	MB70
B1	OUT	BYTE	MB71
B2	OUT	BYTE	MB72
B3	OUT	BYTE	MB73
B4	OUT	BYTE	MB74
B5	OUT	BYTE	MB75
B6	OUT	BYTE	MB76
B7	OUT	BYTE	MB77
STAT	OUT	WORD	MW 78
Error	OUT	BOOL	
RETVAL	OUT	INT	MW 84
Recd	IN/OUT	BOOL	M 79.0

As the passed parameter, the base address of the module must be passed as an integer number (*Base*).

If the function block has read a frame from the CAN 300 PRO module, bit *Recd* is set.

Parameter *STAT* contains the status of the CAN 300 module (see Sec. 7.3.8). The parameter is always assigned a value even if no frame has been received.

Assignment of the parameters for the 29-bit CAN identifier according to the SAE J1939 protocol:

28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Priority</b>	R	P	<b><i>PGN (Parameter Group Number)</i></b>														<b>SA (Source Address)</b>											
	R	P	<b><i>PDU format</i></b>							<b><i>PDU-specific</i></b>																		
	R	P	<b><i>0...239: PDU1 Format</i></b>							<b><i>DA (Destination Address)</i></b>																		
	R	P	<b><i>240..255: PDU2 Format</i></b>							<b><i>Group Extension</i></b>																		

## 8 Return parameter RETVAL

The return parameter RETVAL of the function blocks can contain both function-specific errors or error numbers of the Siemens system function blocks SFC 52, SFC 53, SFC 14 and SFC 20.

### **Error codes of the CAN handling:**

- 80E1h: SDO-FBs: Data len Null is not allowed
- 80E2h: SDO-FBs: Data len greater than 4 is not allowed
- 80F1: Module not ready
- 80F2: Data set assigned
- 80F3: CANopen<sup>®</sup> Master is not „Operational“
- 80F7: CANopen<sup>®</sup> slave still in boot-up
- 80F8: SDO data block pointer too small for SDO data
- 80FA: Abort code for SDO job received.
- 81E1: Node number not permissible (1..127)
- 82E2: Function code not permitted
- 82E3: PDO number with error (1..4)

## 9 Upgrading from CAN 300 to CAN 300 PRO

If projects have so far been performed with the old CAN 300, upgrading to the CAN 300 PRO is always possible because the CAN 300 PRO contains all functions of the CAN 300.

### 9.1 Differences between CAN 300 and CAN 300 PRO

CAN 300 old	CAN 300 PRO
Project memory internal (Flash)	Project memory internal (256K Flash) additional storage of the project on MMC (optional)
128 frames TX and Rx FIFO, 3 frames with high priority	400 frames TX and Rx FIFO, 20 frames with high priority
15 normal and 1 high-priority receive filter (acceptance masks)	16 receive filter, each can be parameterized with high or low priority
Receive filters are only settable as a range via the project	Receive filters can be set via the project and using the DIP switches and as a node address
Baud rate can be set via the project	Baud rate can be set via the project and via the DIP switch
Can only be used in the central controller	Can also be used in expansion racks (ET200M)
3 LEDs for displaying the status	6 LEDs for displaying the status

### 9.2 Layer 2

The project must be created new in the CANParam V4 in accordance with the old CAN 300.

All settings (baud rate, scripts, timer) can be made directly. Only when using a specific bit timing does the setting have to be recalculated.

The data handling software is compatible except for the extension of the RETVAL parameter in all calls.

### 9.3 SAE J1939

The project must be created new in the CANParam V4 in accordance with the old CAN 300.

All settings (baud rate, scripts, timer) can be made directly.

The data handling software is compatible except for the extension of the RETVAL parameter in all calls.

## 9.4 CANopen®

If the CANopen® master data handling software of the old CAN 300 is used, major adaptations to CAN 300 PRO are required. Because of the relocation of the CANopen® master function into the module, a new project must be created in CANParam (see Section 6.4) and the Step 7 software must be revised.

♣Use of the CAN 300 PRO as a CANopen® master has the following advantages:

- Up to 127 slaves can be used
- Lower memory requirement in the PLC program
- No dependence on the cycle time of the PLC
- Lower performance requirement in the PLC

Function	CAN 300 old	CAN 300 PRO
Initialization	Call of the FC 40 in the start-up OB necessary	No call required during start-up
Number of slaves	Up to 64 slaves. Limited by memory size and the speed of the PLC	Up to 127 slaves, number of slaves only has a little influence on the performance and memory allocation in the PLC
Memory assignment	Approx. 11 kByte	Approx. 8 kBytes, unnecessary blocks can be omitted. Standard < 3 kBytes
PDO received	FC49 CANCECLE receives PDO frames one after the other and enters them in the PDO-DBs.	Receive PDOs are entered in the PLC input buffer and loaded into the PLC complete by the FB 20 "CANopen® IO Read" at the start of the PLC cycle.
Transmit PDO	Call of the FC 44 for each PDO	Direct writing into an output buffer (DB, markers, outputs) and complete transmission at the end of the PLC cycle. PDOs are then transmitted automatically.
Read/write SDO	Call of the FC 41 (up to 4 bytes) or FC 42 (more than 4 bytes, own DB necessary); processing by calling the FC 49; only one request can be running at once	Call of the FB 24 for SDOs (1, 2, 4 bytes), FB 27 for SDOs of any length. The module processes the SDOs independently of the PLC cycle.
Network management	Call of the FC 48	Call of the FB 23, NMT functions have the same function codes, additional functions for error checking are contained
Nodeguarding	Call of the FC 47 "Nodeguarding" with FC49 in the cycle; evaluation of the state in the heartbeat DB	Call of the FB 22 "CANopen® Service" with request of the current livelist. The nodeguarding is automatically performed by the master
Emergency receive	Call of the FC 43 "ReceiveAll," if FC 49 indicates receipt of an emergency	On calling the FB 22 "CANopen® Service," emergency frames are passed on directly

## 10 Appendix

### 10.1 Technical data

**Order number** CAN 300 PRO 700-600-CAN12

**Dimensions** 116 x 40 x 125 mm (LxWxH)

**Weight** Approx. 280g

#### CAN interface

**Type:** ISO/DIN 11898-2,  
CAN high speed physical layer

**Transmission rate:** 10 kbps to 1 Mbps

**Protocols:** CAN 2.0A (11 bit)

CAN 2.0B (29 bit)

SAE J1939

CANopen<sup>®</sup> master

CANopen<sup>®</sup> slave

DEVICENET *available soon*

**Connection:** Connector, SUB D 9-way

#### Configuration interface

**Type:** USB 1.1

**Transmission rate:** Fullspeed 12Mbps

**Connector:** USB-B

#### Power supply

**Voltage:** +5V DC via backplane bus

**Current consumption:** 160mA (typ.) / 190mA (max.)

#### Special features

**Quality assurance:** According to ISO 9001:2008

**Maintenance:** Maintenance-free (no battery, rechargeable or non-rechargeable)

## 10.2 Pin assignment

Pin	SUBD connector CAN
1	-
2	CAN Low
3	CAN GND
4	-
5	-
6	-
7	CAN High
8	-
9	-

## 10.3 Further documentation

Internet: [www.can-cia.org](http://www.can-cia.org)

CAN Specification 2.0, Part A & Part B

High Layer Protocol CANopen®

Holger Zeltwanger: "CANopen®," VDE Verlag, ISBN 3-8007-2448-0

## Notes