CAN 400

CAN Communication Module for S7-400
with CAN Layer 2 – or CANopen Handling

700-640-CAN11 / 700-640-CAN21

Manual

Edition 6 - 04.04.2006 / HW1 & FW1.13 and higher

Manual order number: 900-640-CAN21

Systeme Helmholz **GmbH** ● Gewerbegebiet Ost 36 ● D-91085 Weisendorf ● Germany
Phone: +49 9135 7380-0 ● Fax: +49 9135 7380-50 ● E-mail: info@helmholz.de ● Internet: www.helmholz.de

Systeme Helmholz **GmbH**

Gewerbegebiet Ost 36, 91085 Weisendorf, 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 data in this manual have been checked regularly and any necessary corrections will be included in subsequent editions. We always welcome suggestions for improvement.

Contents

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*

ℹ *gives information*

⚠ *hazard, general or specific*

⚡ *danger of* ***electric shock***

1.1 General

The CAN 400 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 estimation).*

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

2 Installation and Mounting

2.1 Foreword

This section describes planning of mechanical assembly, preparation of components for mounting, and final mounting itself.

The S7-400 modules are intended for mounting in module racks of the S7-400 system.

The module racks of the S7-400 system are the basic structure in which the individual modules are mounted. The modules exchange data and signals via the backplane bus of the modules and are supplied with power.

The manufacturers' instructions must be observed in selecting and setting up the module rack of the S7-400 system!

2.2 Restriction of access

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

The S7-400 module must be installed according to VDE 0100/ IEC 364. 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.

2.3 Planning assembly

ℹ *The modules can only be mounted horizontally.*

Permissible ambient temperature:

for horizontal mounting: from 0 to +60 °C

Module racks of the S7-400 modules communicate via the I/O bus (P bus); a module rack with a communication bus (K bus) is not required.

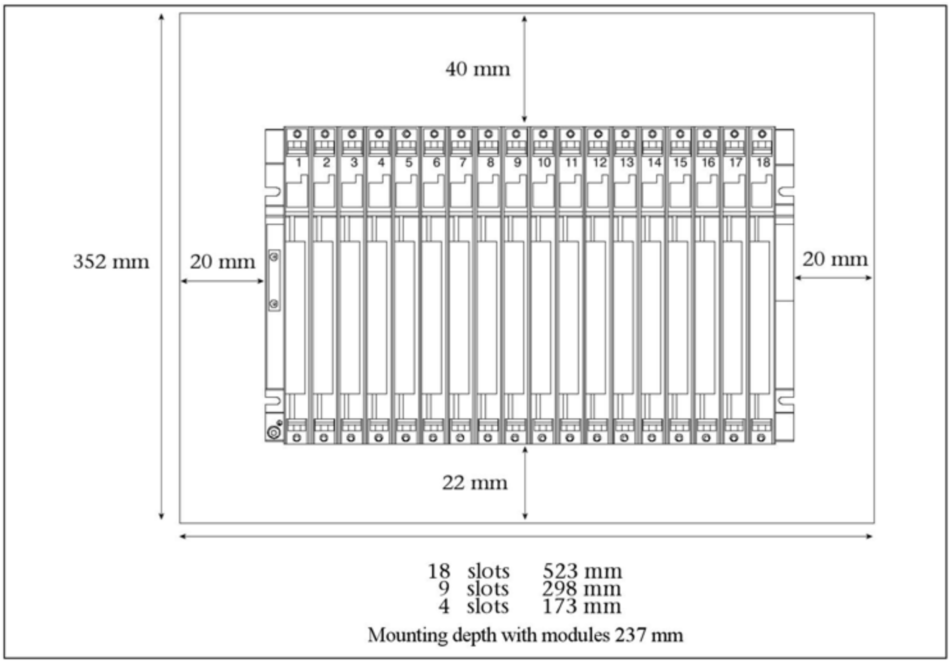


Fig. 2-1: Vertical mounting

2.4 Minimum clearance

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

Minimum clearances must be observed because

it ensures cooling of the S7-400 modules

it provides space to insert and remove modules

it provides space to route cables

Fig. 2-1 shows the minimum spacing to any adjacent cabinet walls, equipment, cable ducts, etc. for S7-400s mounted in several module racks.

To set up an S7-400 with several module racks, additional clearances between the module racks must be observed, or a fan tier or cable duct is mounted, see Fig. 2-2.

The additional clearance is also required if the S7-400 is mounted above devices with similar size or heat dissipation.

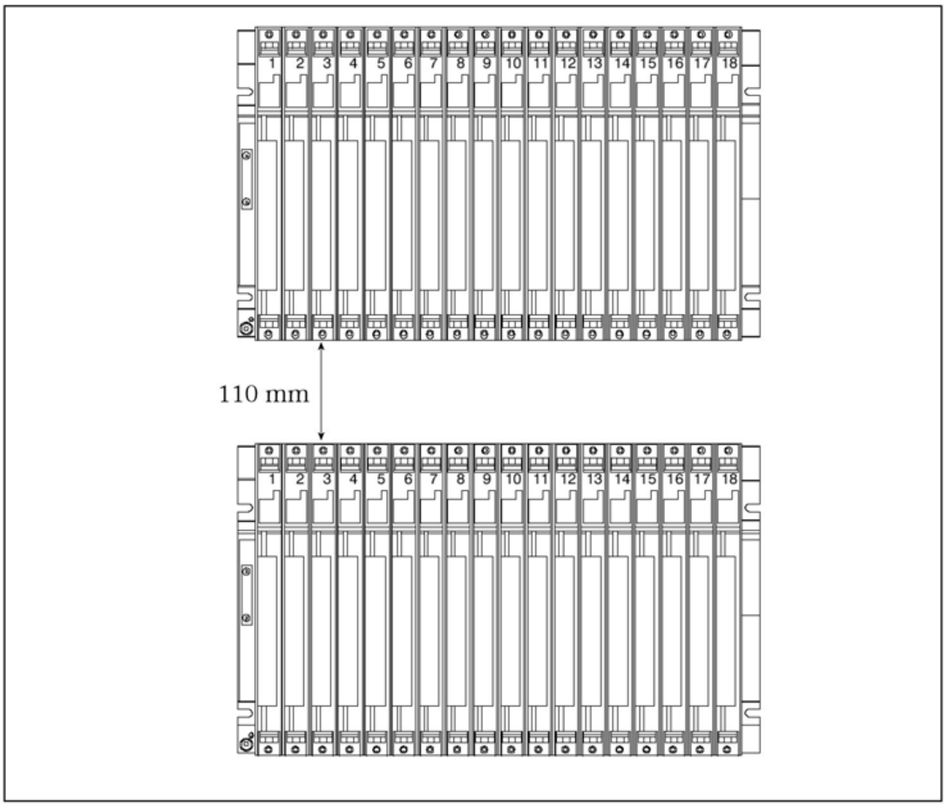


*Fig. 2-2:
Minimum clearances in the case of more than one module rack*

2.5 Connecting a module rack to building ground

A low-impedance connection must be made with building ground. That is achieved with the shortest possible, low-resistance cable with the largest area of contact.

The module rack must be reliably grounded.

For grounding there is a stud bolt in the lower left part of the module rack.

The minimum cross-section of the cable to building ground is 10 mm^2.

If the S7-400 is mounted on a moving framework, a flexible cable to building ground must be used.

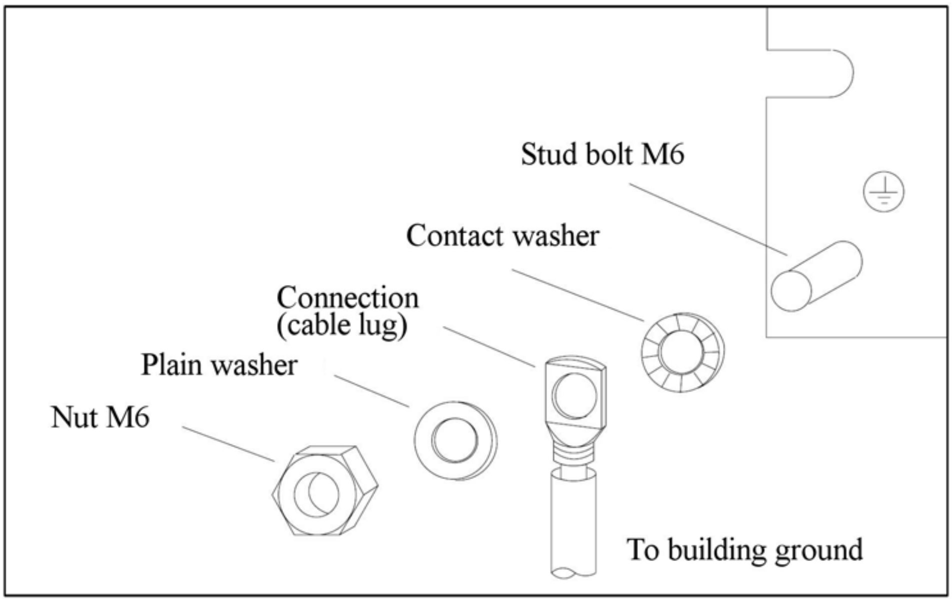


*Fig. 2-3:
Connection of building ground on the module rack*

2.6 Ground connection in a non-isolated set-up

On the module rack, the ground of the 24 V load voltage is connected to the 5 V ground in the non-isolated set-up (reference potential GND, logic ground).

2.6.1 Grounded set-up

For non-isolated modules, the ground is connected to the reference point. The galvanic connection remains installed as shown in the figure. The reference point is galvanically connected to the reference potential GND.

Fig. 2-4 shows the position of the reference point on the module rack.

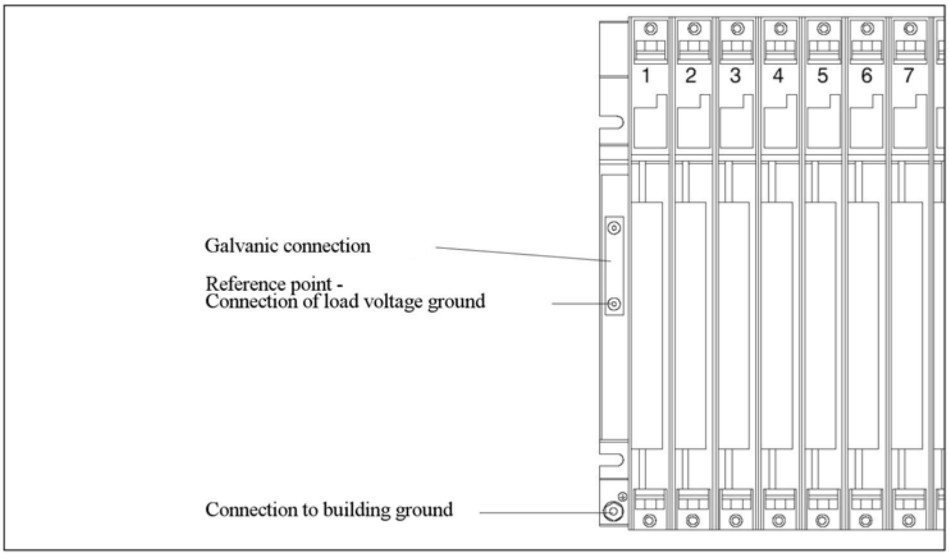

1
2
3
4
5
6
7
Galvanic connection
Reference point -
Connection of load voltage ground
Connection to building ground

Fig. 2-4:
Reference point for ground connection

2.6.2 Ungrounded set-up

The galvanic connection is separated:

disconnect the fixing screws of the galvanic connection on the module rack.

swing the connection downward, use the existing original screw M4 x 8 for the connection at the reference point - the swung down connection functions as a washer, see Fig. 2-5.

⚠ *Do not use screws longer than M4 x 8!*

For connection to the reference point, do not use fillister screws that are longer than shown in Fig. 2-5. Longer screws could make an unwanted connection between the reference point and the rack section behind it and with the connection for building ground. For this reason, the galvanic connection must be mounted as a "washer" on the module rack even in an ungrounded set-up.

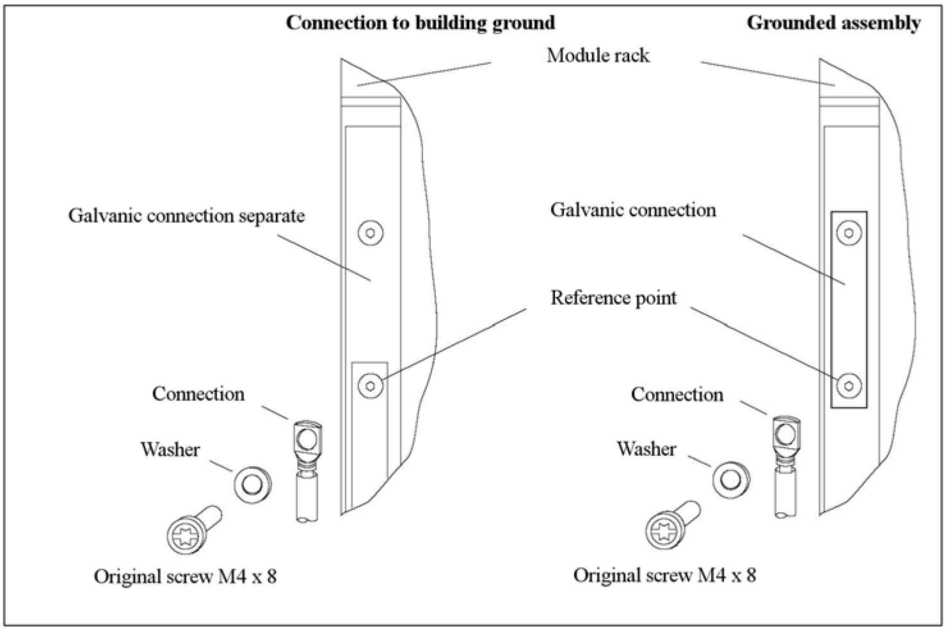


Fig. 2-5:
Ungrounded assembly

2.7 Mounting modules in a module rack

Caution!
If force is used to insert modules into the module rack, the components may be damaged.

Mounting is performed in the following steps:

Remove the blanking plates from the slots where the modules are to be inserted - take hold of the blanking plate at the marked positions and pull it forward to remove it

Unplug the power connector from the power supply module

hook in the module and swing downward
(see Fig. 2-6)

if you feel any resistance when swinging down the module, raise it slightly before resuming insertion

Screw the modules on at the top and bottom with a torque of 0.8 to 1.1 Nm, (see Fig. 2-7)

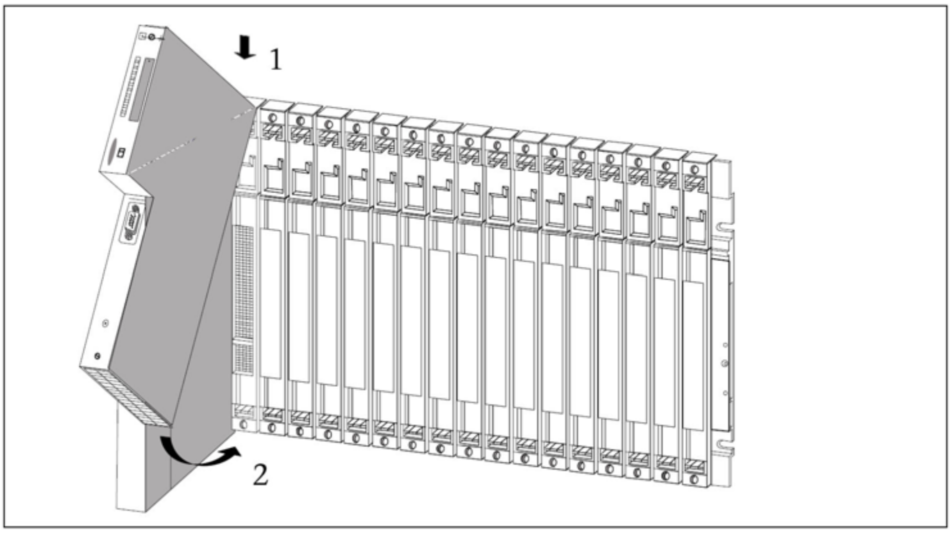


Fig. 2-6:
Inserting modules

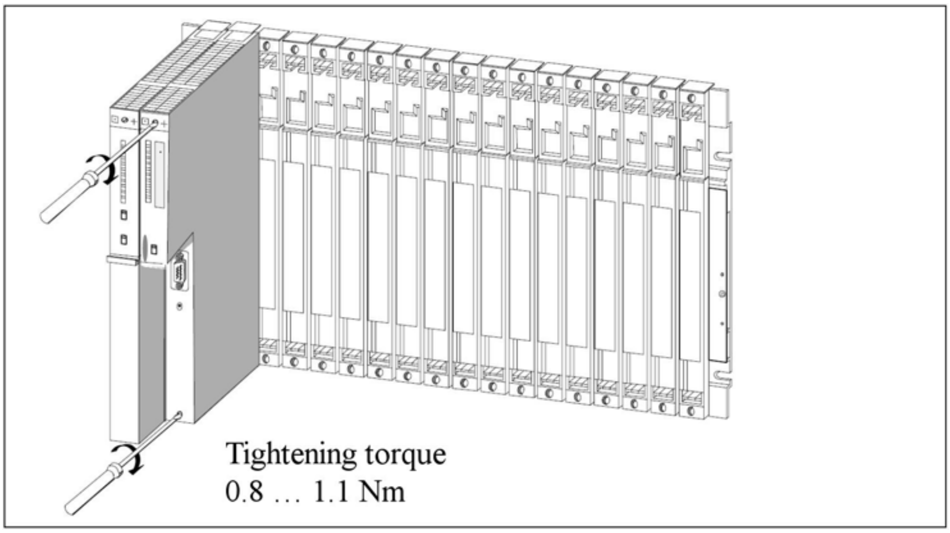


Fig. 2-7:
Screwing the modules tight

3 System Overview

3.1 Application and function description

The CAN 400 module from System Helmholz 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 the expansion unit.

The CAN 400 module must be parameterized as an "FM450-1" counter module in the hardware configurator and takes up 64 bytes in the address space. Data is exchanged with the PLC via the backplane bus.

The data handling blocks that permit simple handling of CAN communication are supplied as source code. Data handling blocks are available both for single layer 2 communication and for CANopen master communication. Handling software is available on request for use of the CAN 400 module as a CANopen Slave or for control of LENZE drives.

The scope of supply also includes a Windows parameterization tool "CANParam V3" for easy setting of the CAN communication parameters.

The CAN 400 module supports both CAN 2.0A (11 bits) and CAN 2.0B (29 bits) frames as Highspeed Node (ISO 11898-2) with a freely selectable baudrate of 10Kbps to 1Mbps. The baudrate can optionally be set via the DIP switch on the front panel.

The CAN 400 module contains the management functions "Power On", "Stop->Run", "Run->Stop", and "Power Off". Behind each of the four functions it is possible to use a simple macro language to configure functions that are automatically performed by the module, if the event occurs.

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

As an alternative, it is possible to filter the CAN telegrams according to a set node ID, i.e. all broadcast frames and frames that are addressed to this node are allowed to pass. The node ID can be stored in the project, be assigned by the PLC or be set via the DIP switches on the front of the housing.

16 freely settable timers are available in the CAN 400 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 400 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 PLC and transmitted from the CAN 400 module after the parameterized time has elapsed.

The CAN 400 module can trigger a process interrupt (OB40) to the S7 when a telegram is received by the CAN bus. That makes it possible to respond to CAN frames irrespective of the cycle of the PLC program.

3.2 Connections

The CAN 400 modules has a (CAN400-1) or two (CAN400-2) 9-pin SubD connectors for the CAN bus.

The USB socket provides the connection to the PC. With the "CANParam V3" Windows program, configuration can be transmitted or a diagnosis of the module performed.

The jack socket is a serial interface. This can be used if a USB connection is not available. An operating system update can also be performed via this interface.

Pin assignment:

There is no 24V power supply on the CAN-connector available.

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

3.3 CAN cabeling

A CAN bus cable requires at least 3 lines: CAN High, CAN Low, and CAN Ground. Only a bus structure is permitted. A 120-ohm terminating resistor between CAN High and CAN Low must be connected to both ends of the CAN bus cables. The CAN 400 module does not contained integrated terminating resistors.

The CAN 400 module does not contain integrated terminating resistors.

Check for correct cabling in the Debug dialog box of the CANParam (see Chapter 5.10)

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

Bitrate	Bus Length	Bit Time
1 Mbps	30 m	1 µsek.
800 Kbps	50 m	1,25 µsek.
500 Kbps	100 m	2 µsek.
250 Kbps	250 m	4 µsek.
125 Kbps	500 m	8 µsek.
20 Kbps	2500 m	50 µsek.
10 Kbps	5000 m	100 µsek.

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

More detailed information is available in document “CANopen Recommendation DR 303-1”.

3.4 LED displays

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

LED Pwr (green): Continuous light indicates that the module is in cyclic operation. If the light is flashing, the module is in update mode. If the "Pwr" LED is off, an internal error has occurred or the module is defective.

LED Param (yellow): A continuous light indicates correct parameterization of the module. The flashing light indicates incorrect parameterization.

LED CAN1/2 Rx (green): CAN frames are received on the CAN bus (channel 1 or 2).

LED CAN1/2 Tx (yellow): CAN frames are transmitted on the CAN bus (channel 1 or 2).

LED CAN1/2 Err (red): A CAN controller or buffer overflow error has occurred.

LED CAN1/2 AG (yellow): Data are exchanged with the PLC (channel 1 or 2)

3.5 Switches

The 10-fold DIP switch on the front of the housing is for setting the CAN baudrate and for defining the node address if the module is used as a CANopen slave.

The switches are counted from bottom to top.

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

Baudrates:

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

3.6 Scope of supply

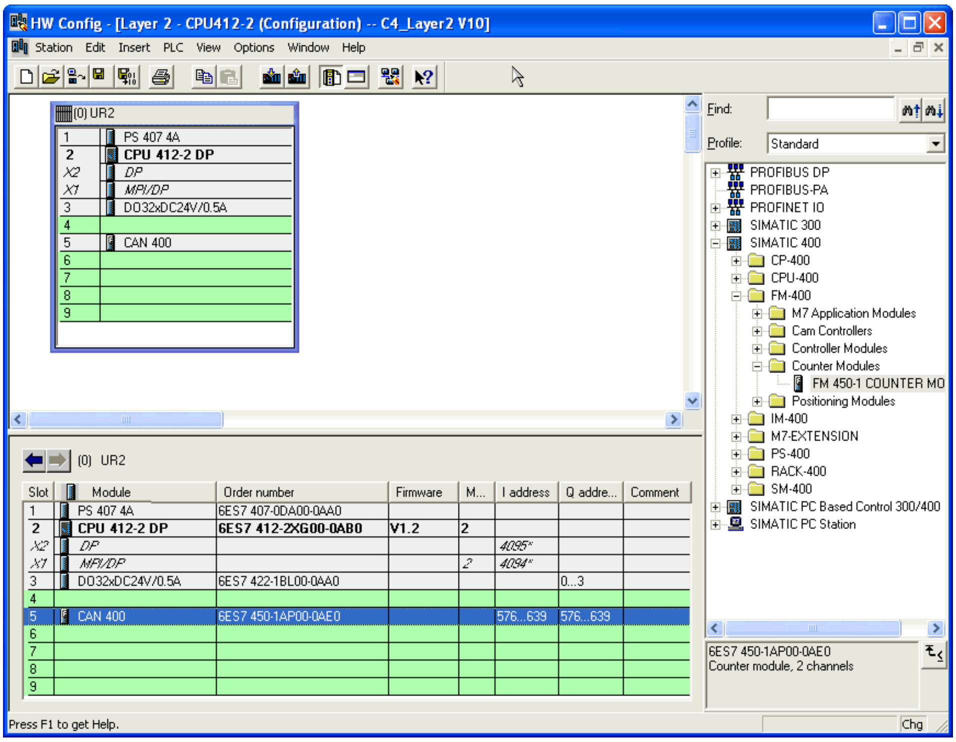
1x module CAN 400-1 or CAN 400-2

3.7 Accessories

CAN CD with parameterization software "CANParam", "Layer 2" and "CANopen" handling blocks	800-600-1AA11
USB cable	700-755-7VK11
Serial/Update cable	700-750-CAN21
Manual, German/English	900-640-CAN21
CAN bus connector	700-690-0BA11
CAN bus connector with cable connector	700-690-0BB11
CAN bus connector axial	700-690-0CA11

4 Configuration in the PLC

The CAN 400 module is configured as a "FM450-1" counter module in the programming software of the PLC. The installation CD contains a Step 7 project that already contains a configured module and the necessary handling blocks.



(0) UR2

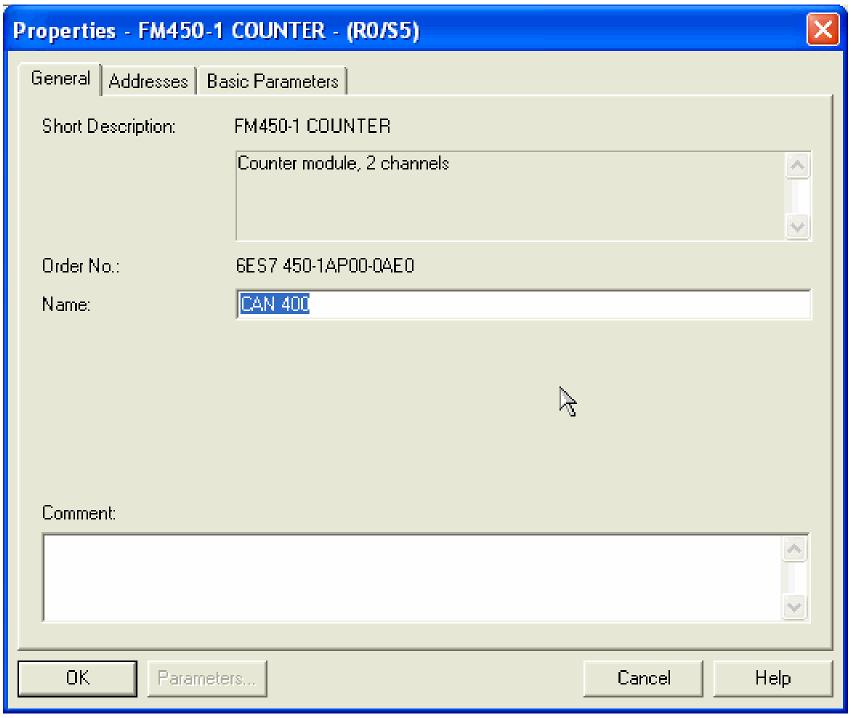
Slot	Module	Order number	Firmware	M...	I address	Q addre...	Comment
1	PS 407 4A	6ES7 407-0DA00-0AA0					
2	**CPU 412-2 DP**	**6ES7 412-2XG00-0AB0**	**V1.2**	**2**			
X2	*DP*				*4095**		
X1	*MPI/DP*			*2*	*4094**		
3	DO32xDC24V/0.5A	6ES7 422-1BL00-0AA0				0...3	
4							
5	CAN 400	6ES7 450-1AP00-0AE0			576...639	576...639	
6							
7							
8							
9							

6ES7 450-1AP00-0AE0
Counter module, 2 channels

Press F1 to get Help.

Only one CAN 400 module can be used in a multiprocessor system.

The module can be used wherever a FM module is allowed, i.e. also in the expansion unit after an interface module. Several CAN 400s can be used in one rack.



Short Description: FM450-1 COUNTER

Counter module, 2 channels

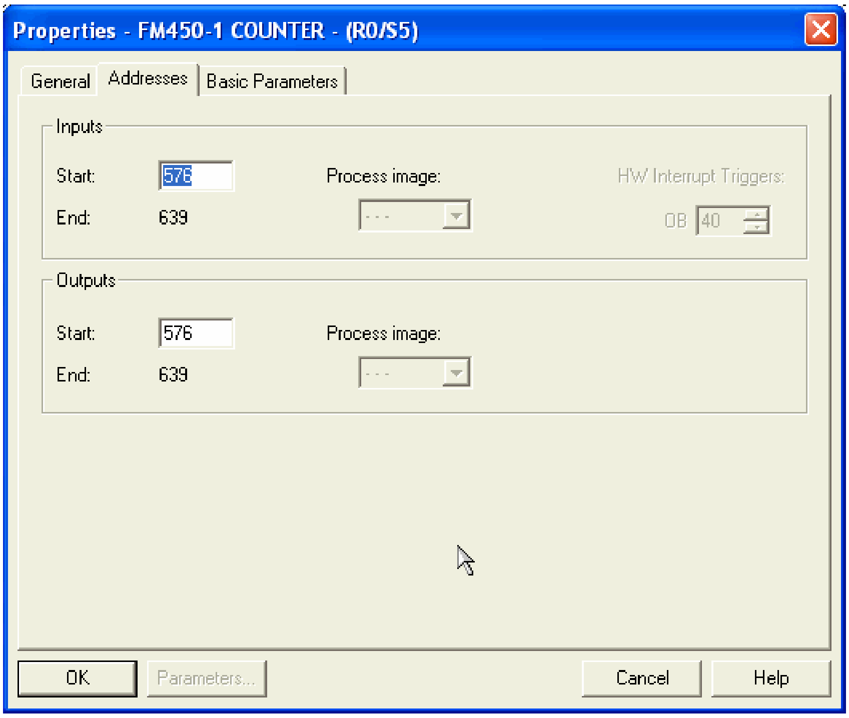
Order No.: 6ES7 450-1AP00-0AE0

Name: CAN 400

Comment:

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

The addresses for the inputs and the outputs must always be the same so that the data handling software can access them correctly.



Properties - FM450-1 COUNTER - (R0/S5)

General | Addresses | Basic Parameters

Inputs

Start: 576

End: 639

Process image: - - -

HW Interrupt Triggers:

OB 40

Outputs

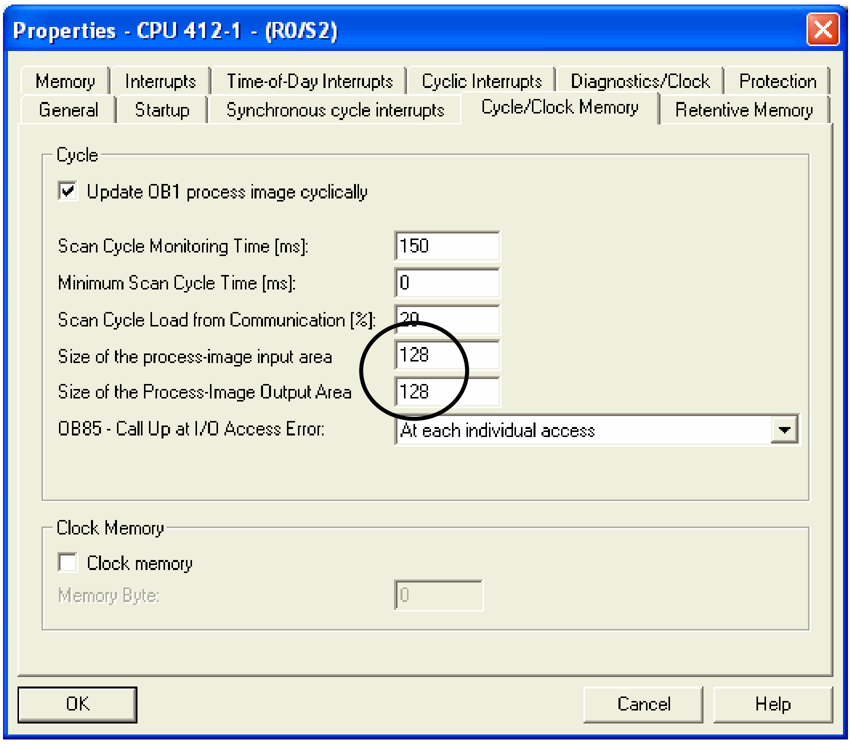
Start: 576

End: 639

Process image: - - -

Do not use the CAN 400 in (cyclic) process image!

The CAN 400 can not be used in (cyclic) process image!



Properties - CPU 412-1 - (R0/S2)

Memory | Interrupts | Time-of-Day Interrupts | Cyclic Interrupts | Diagnostics/Clock | Protection
General | Startup | Synchronous cycle interrupts | Cycle/Clock Memory | Retentive Memory

Cycle

☑ Update OB1 process image cyclically

Scan Cycle Monitoring Time [ms]: 150

Minimum Scan Cycle Time [ms]: 0

Scan Cycle Load from Communication [%]: 20

Size of the process-image input area 128

Size of the Process-Image Output Area 128

OB85 - Call Up at I/O Access Error: At each individual access

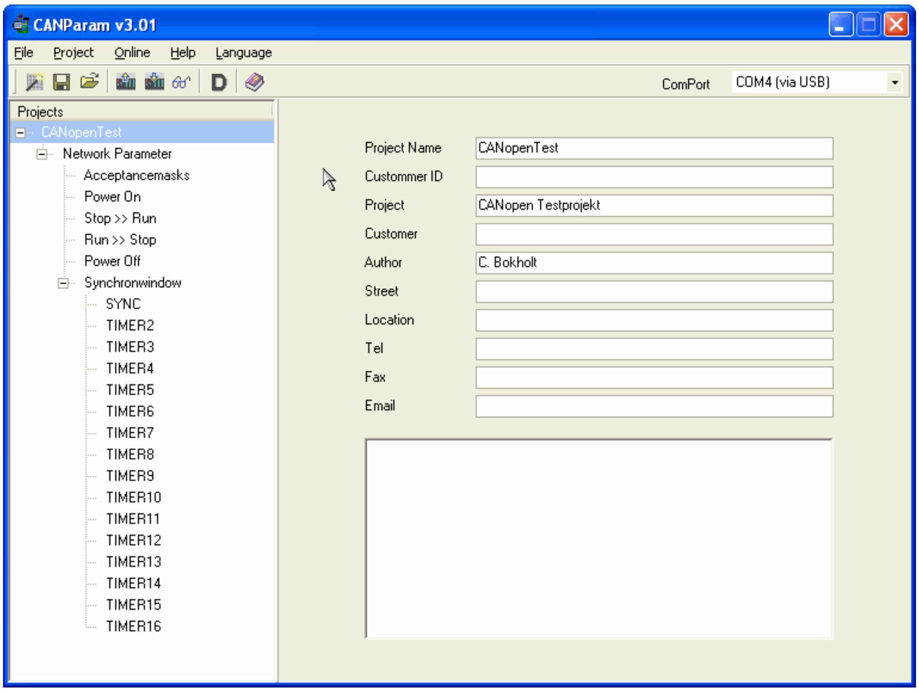
Clock Memory

□ Clock memory

Memory Byte: 0

5 Configuration of the CAN 400 module

The CAN 400 module is configured on the PC with the "CANParam V3" software. This software is supplied together with the handling blocks for the S7 and can run on any Windows 2000/XP computer.



ComPort: COM4 (via USB)

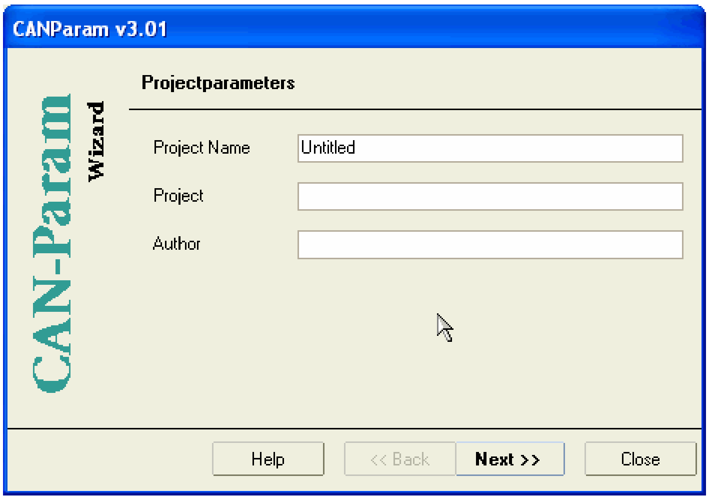
Project Name: CANopenTest
Custommer ID:
Project: CANopen Testprojekt
Customer:
Author: C. Bokholt
Street:
Location:
Tel:
Fax:
Email:

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 400 module. After installation and starting of the CANParam software, you should set the interface top right on the menu bar. The virtual COM interfaces only appear if the USB link has been connected before the software is started.

5.1 Creation of a new project

A new project can be created via the "Project / Create project / Projectwizard" function or with the project wizard.

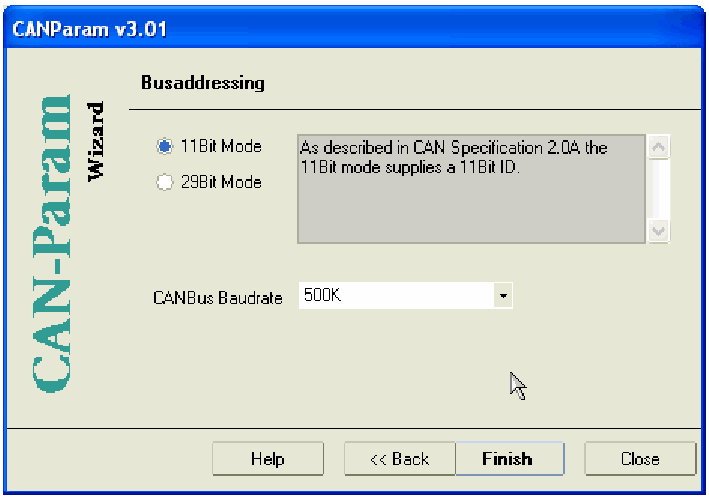


Project Name: Untitled

Project:

Author:

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



● 11Bit Mode

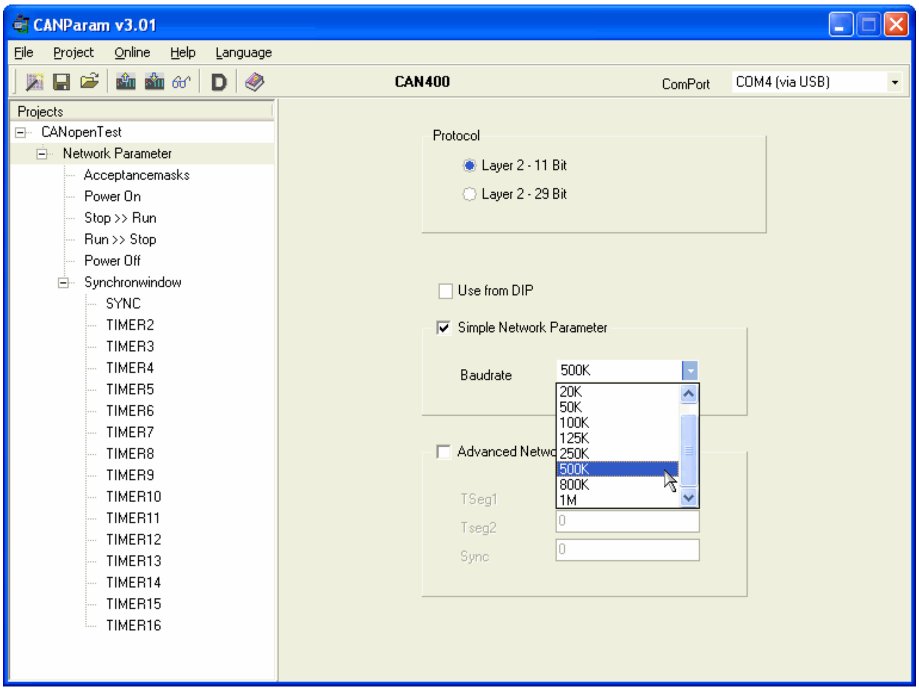
○ 29Bit Mode

As described in CAN Specification 2.0A the 11Bit mode supplies a 11Bit ID.

CANBus Baudrate: 500K

5.2 Setting the CAN bus baudrate

You can select the CAN baudrate in the range from 10kbps to 1Mbps, or define it on the module by setting the DIP switch.



ComPort COM4 (via USB)

Protocol

● Layer 2 - 11 Bit

○ Layer 2 - 29 Bit

□ Use from DIP

☑ Simple Network Parameter

Baudrate 500K

□ Advanced Netwo

TSeg1

Tseg2 0

Sync 0

For special applications you can define the bit time of transmission directly. For a precise description of the bit timing see CAN Specification 2.0 Part B, Chapter 10 onward.

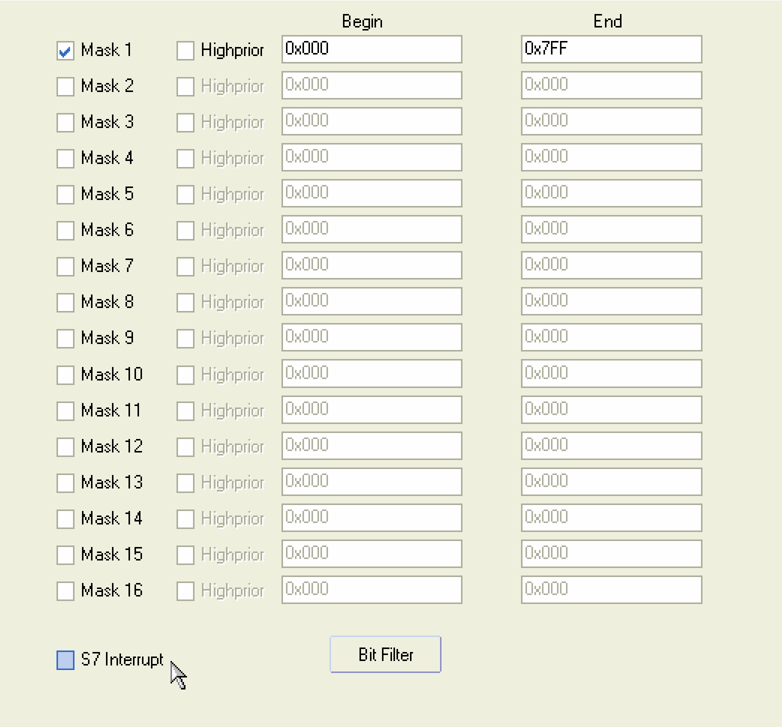
5.3 Setting the transmission mode (protocol)

The CAN 400 module supports both the protocol format CAN 2.0A (11 bits) and CAN 2.0B (29 bits).

For use of the CANopen handling blocks, a CAN 2.0A (11 bits) must always be selected.

5.4 Acceptance masks

16 acceptance masks are provided in the CAN 400 module. Using these masks you can enable or block various frame IDs for receiving.



		Begin	End
☑ Mask 1	□ Highprior	0x000	0x7FF
□ Mask 2	□ Highprior	0x000	0x000
□ Mask 3	□ Highprior	0x000	0x000
□ Mask 4	□ Highprior	0x000	0x000
□ Mask 5	□ Highprior	0x000	0x000
□ Mask 6	□ Highprior	0x000	0x000
□ Mask 7	□ Highprior	0x000	0x000
□ Mask 8	□ Highprior	0x000	0x000
□ Mask 9	□ Highprior	0x000	0x000
□ Mask 10	□ Highprior	0x000	0x000
□ Mask 11	□ Highprior	0x000	0x000
□ Mask 12	□ Highprior	0x000	0x000
□ Mask 13	□ Highprior	0x000	0x000
□ Mask 14	□ Highprior	0x000	0x000
□ Mask 15	□ Highprior	0x000	0x000
□ Mask 16	□ Highprior	0x000	0x000

□ S7 Interrupt

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

With the "highprior" 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 telegram bypassing the normal receive buffer.

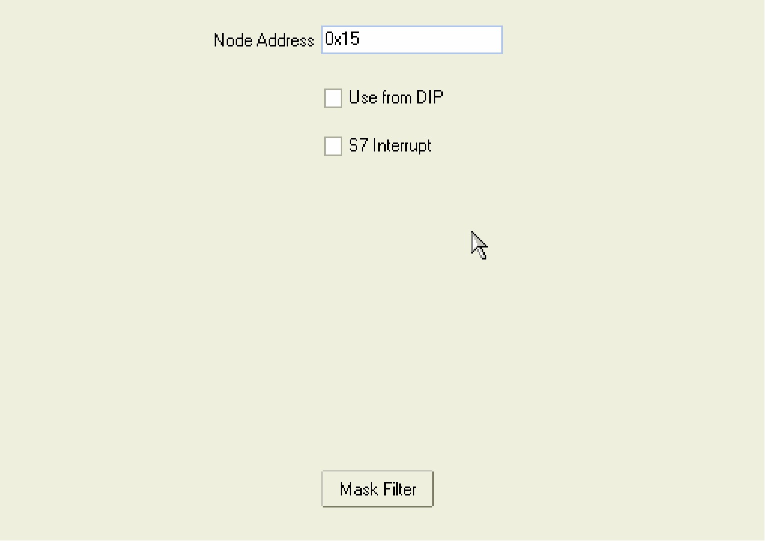
For CAN400-2 it should only 1 channel used with interrupts.

The "S7 Interrupt" option activates triggering of an OB40 call in the S7, if a received telegram is available.

5.5 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 0x15

□ Use from DIP

□ S7 Interrupt

If the CAN 400 is to be used as a CANopen slave, 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, all frames with the node ID 0 (broadcasts) are also allowed to pass with high priority.

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

For CAN400-2 it should only 1 channel used with interrupts.
Do not use Interrupts in multicomputing systems.

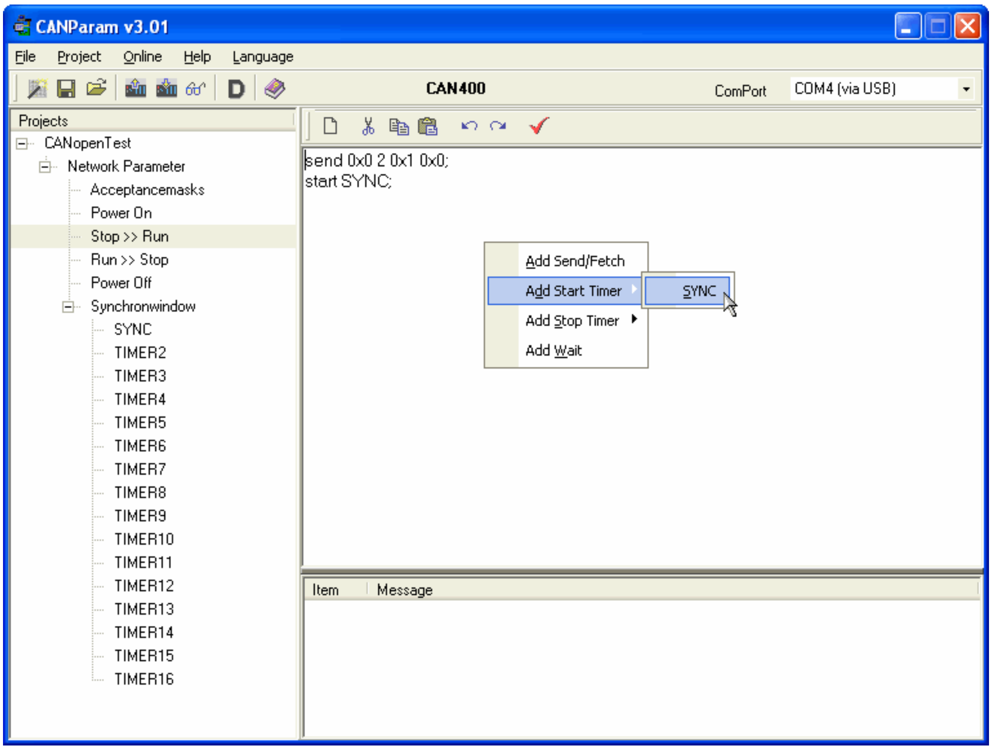
The "S7 Interrupt" option activates triggering of an OB40 call in the S7, if a received telegram is available.

5.6 Scripts

The CAN 400 module can transmit freely programmable CAN frames (layer 2) for the PLC events "Power ON", "Stop -> Run", and "Run -> Stop", and "Power-Off" and start and stop timers.

The following commands are available:

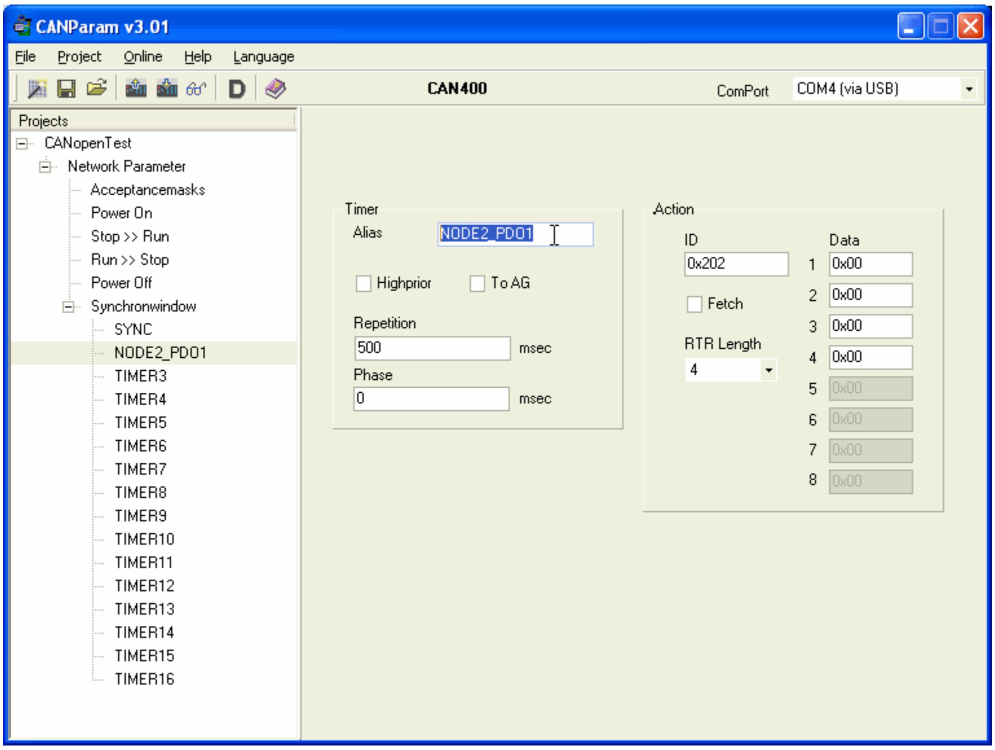
Send	Transmit frame (Structure: ID, length, data byte 1, data byte 2, etc.)
Fetch	Transmit frame with RTR bit 1
Start	Start Timer X
Stop	Stop Timer X
Wait	Wait X ms
//	Comment line



```
send 0x0 2 0x1 0x0;
start SYNC;
```

5.7 Timer

16 timers are available for time-dependent events in the CAN 400 module. Each timer can transmit any CAN frame.



ComPort: COM4 (via USB)

Timer

Alias: NODE2_PDO1

□ Highprior □ To AG

Repetition: 500 msec

Phase: 0 msec

Action

ID: 0x202

□ Fetch

RTR Length: 4

Data	
1	0x00
2	0x00
3	0x00
4	0x00
5	0x00
6	0x00
7	0x00
8	0x00

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

The time *period* 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 msec. can be set in steps of 1 msec. For the *phase* 0msec to 1 msec before the period duration.

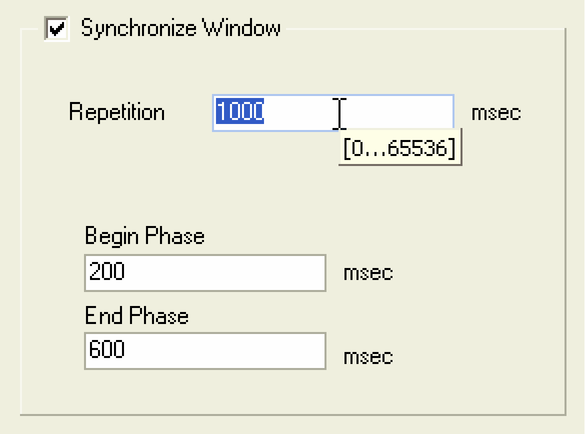
The data of the CAN frame defined for the timer are initialization data and can be overwritten by the S7-CPU in cyclic operation by the FC62 "CANTIMER".

5.8 Synchro window

If you are using the synchronous timer (setting "synchronous queue"), the frames transmitted asynchronously by the FC60 "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.

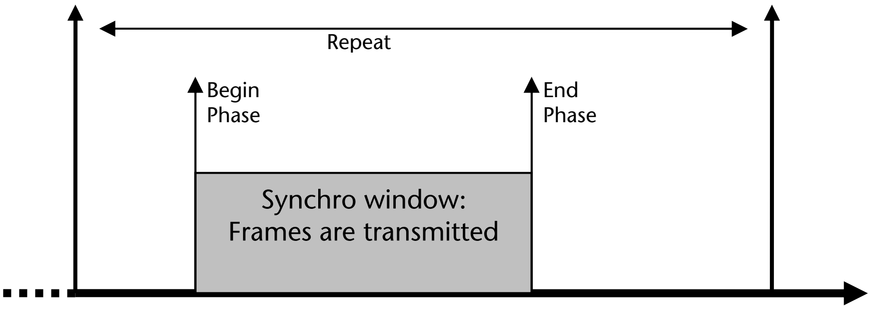


☑ Synchronize Window

Repetition: 1000 msec [0...65536]

Begin Phase: 200 msec

End Phase: 600 msec



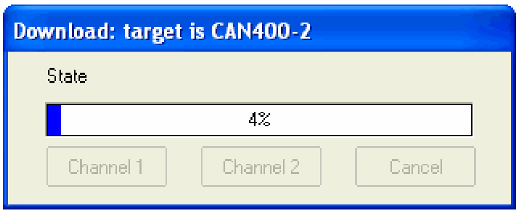
To use the synchronous window, both timers1 & 2 must be started directly one after the other in a script!

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.

The functionality of the other timers is not affected by the synchronous window, i.e. they can also be transmitted outside the synchronous window.

5.9 Download / Upload

The project currently being worked on can be imported into the CAN 400 module again at any time ("Download"). In the case of a CAN 400-2 with 2 channels, the destination channel can be selected in the dialog box.



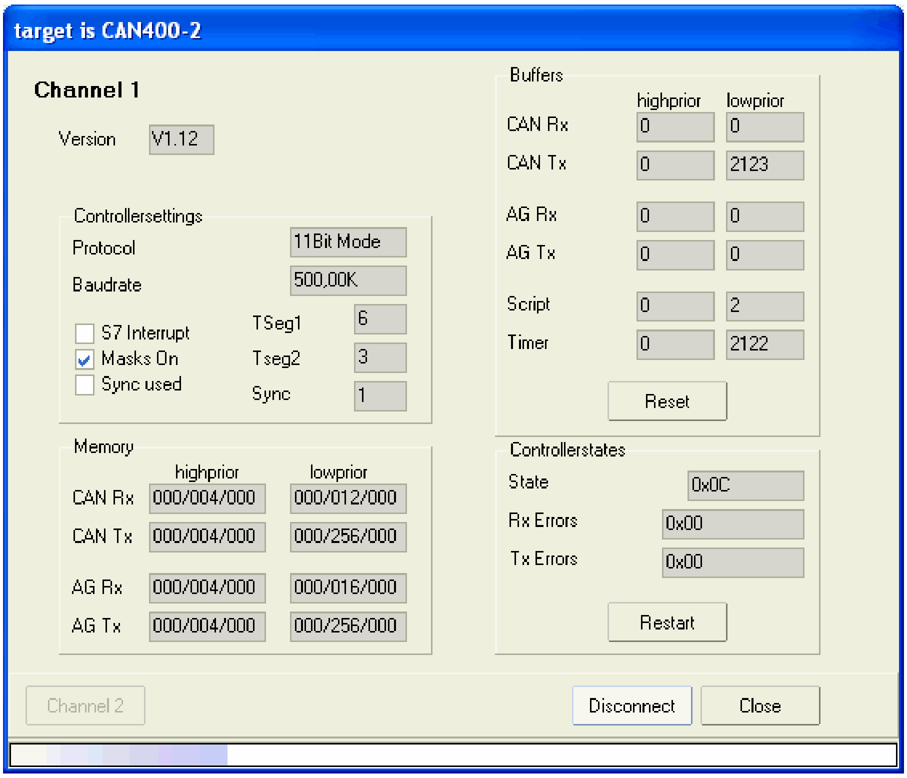
Download: target is CAN400-2

State

4%

5.10 Diagnostics/debugging

To simplify debugging, you can query the status of the CAN 400 module with menu item "Debug". Debug mode requires a serial link with the module.


target is CAN400-2

Channel 1

Version V1.12

Controllersettings

Protocol 11Bit Mode

Baudrate 500,00K

□ S7 Interrupt TSeg1 6

☑ Masks On Tseg2 3

□ Sync used Sync 1

Memory

	highprior	lowprior
CAN Rx	000/004/000	000/012/000
CAN Tx	000/004/000	000/256/000
AG Rx	000/004/000	000/016/000
AG Tx	000/004/000	000/256/000

Buffers

	highprior	lowprior
CAN Rx	0	0
CAN Tx	0	2123
AG Rx	0	0
AG Tx	0	0
Script	0	2
Timer	0	2122

Controllerstates

State 0x0C

Rx Errors 0x00

Tx Errors 0x00

The "Connect" button activates monitoring mode. If you press the button again, the link will be disconnected again. The "Channel x" button switches between channel 1 and 2.

The Restart button executes a restart of the channel. This corresponds to an upload of a project.

The debug dialog provides the following information:

Version: Operating system version

Controller settings: Configuration settings of the CAN controller

Memory: Indication of size and use in the internal FIFO memory in "used/max/lost" format

Buffer: Counter for the telegram buffers

<u>**Note:**</u> The CAN 400 module has a receive buffer and a transmit buffer for 256 telegrams each. The buffer pointers indicate to what extent the buffers are full. For example, if a CAN telegram has been received, “CAN Rx” is incremented. If the telegram has then been passed on to the PLC (fetched by the data handling block), “AG Tx” is incremented (read/write pointer principle). There should never be a big difference between pairs of pointers. If there is, the CAN telegrams are not being fetched fast enough by the PLC, or are being transmitted too fast by the PLC.

"Reset counter" button: resets the buffer counter

!

Error Counters has to „0“, otherwise the CAN communication is malfunctioning.

Controller status: Status information of the CAN controller

Note: The transmit and receive error counters (“Rx Errors” / “Tx Errors”) are incremented by the CAN controller, if transmission and receipt of a telegram has failed. As soon as a telegram has been correctly transmitted or received, the corresponding counter is decremented again. This counter should always be at 0 when the CAN bus is functioning correctly!

6 Programming in the PLC

6.1 Overview

The CAN 400 module is programmed in the PLC using the data handling blocks supplied with the CAN CD.

Data handling blocks are available for pure layer 2 communication and for communication with CANopen stations as the master.

6.2 Layer 2 communication

6.2.1 General

4 FCs are available for layer 2 communication:

FC 60	**CANSEND**	Transmission of a CAN frame
FC 61	**CANRCV**	Receiving a CAN frame
FC 62	**CANTIMER**	Changing the data of a timer
FC 63	**CANCTRL**	Control and command functions

The base address set in the hardware configurator must be passed to each block.

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:

```
SET
=    M 1.1

CALL "CANSEND"           // FC 60
     Base     := 512
     Chan     := 1
     IDHI     := W#16#0
     IDLO     := W#16#202
     RTRLEN   := B#16#8
     DW0      := MW 4
     DW1      := W#16#0
     DW2      := W#16#0
     DW3      := W#16#0
     STAT     := MB 2
     Snd      := M 1.1

CALL "CANRCV"            // FC 61
     Base     := 512
     Chan     := 1
     IDHI     := MW 10
     IDLO     := MW 12
     RTRLEN   := MB 14
     DW0      := MW 16
     DW1      := MW 18
     DW2      := MW 20
     DW3      := MW 22
     STAT     := MB 24
     SOURCE   := MW 25
     Recd     := M 1.0

AN   M 1.0
BEC
...
```

6.2.2 Handling function FC 60 CANSEND

The CANSEND function block (FC 60) transfers a CAN frame to the module from which it is transmitted immediately.

Parameter	Direction	Type	Example
Base	IN	INT	512
Chan	IN	INT	1
IDHI	IN	WORD	W#16#0
IDLO	IN	WORD	W#16#202
RTRLEN	IN	WORD	B#16#8
DW0	IN	WORD	MW 4
DW1	IN	WORD	W#16#0
DW2	IN	WORD	W#16#0
DW3	IN	WORD	W#16#0
STAT	OUT	BYTE	MB 4
Snd	IN/OUT	BIT	M 1.0

As parameters, the base address (`Base`), and the channel (`Chan`) of the module as integers, a status byte (`STAT`), and a bit for transmit enable (`Snd`) must be passed.

The elements of the frame are passed as source 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. All other bits must be set to 0.

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 module is not ready, e.g. if the parameterization of the module contains an error (see `STAT` byte), the `Snd` bit remains set.

⚠ *FC60 „CANSEND“ should not be called in OB 1 and OB 35 in the same programm!*

The status of the CAN 400 module is in the `STAT` byte. The byte is always assigned a value, even if the `Snd` bit is not set.

If the synchronous timer has been set, the data are only ever transmitted in a defined synchronous time window.

6.2.3 Handling function FC 61 CANRCV

The CANRCV function block (FC 61) transfers a CAN frame from the module into the PLC, if a frame has been received and this frame has also been let through by the acceptance filter.

Parameter	Direction	Type	Example
Base	IN	INT	512
Chan	IN	INT	1
IDHI	OUT	WORD	MW 10
IDLO	OUT	WORD	MW 12
RTRLEN	OUT	WORD	MB 14
DW0	OUT	WORD	MW 16
DW1	OUT	WORD	MW 18
DW2	OUT	WORD	MW 20
DW3	OUT	WORD	MW 22
STAT	OUT	BYTE	MB 24
SOURCE	OUT	WORD	MW 25
Rcvd	IN/OUT	BIT	M 1.0

As parameter, the base address (`Base`) and the channel (`Chan`) of the module must be passed as integers.

The elements of the frame are contained in `IDHI, IDLO, RTRLEN, DW0...3`.

The byte `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. Bit 7 indicates receipt of a CAN frame with a 29-bit identifier.

If the function block has read a frame from the CAN 400 module, bit `Recd` is set.

The status of the CAN 400 module is in the `STAT` byte. The byte is always assigned a value even if no frame has been received.

Further information on the type of frame received can be taken from the `SOURCE` word. Bit 7 of `SOURCE` indicates a high-priority receipt. Bits 8-11 indicate whether the received frame came from the CAN bus (bits 8-11 = 2), or was transmitted by a timer in the module (bits 8-11 = 5).

6.2.4 Handling function FC 62 CANTIMER

The function block CANTIMER (FC 62) changes the data values of a timer in the CAN 400 module.

Parameter	Direction	Type	Example
Base	IN	INT	512
Chan	IN	INT	1
TimeNo	IN	INT	1
DW0	IN	WORD	MW 30
DW1	IN	WORD	MW 32
DW2	IN	WORD	MW 34
DW3	IN	WORD	MW 36
ERROR	OUT	BYTE	MB 37
STAT	OUT	BYTE	MB 38
Snd	IN/OUT	BIT	M 1.2

As parameters, the base address (`Base`), and the channel (`Chan`) of the module as integers, the timer number (`TimeNo`), a status byte (`STAT`), and a bit for transmit enable (`Snd`) must be passed.

The new data of the timer are passed in `DW0...3`.

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 module is not ready, e.g. if the parameterization of the module contains an error (see `STAT` byte), the `Snd` bit remains set.

The status of the CAN 400 module is in the `STAT` byte. The byte is always assigned a value even if no frame has been transmitted.

The `ERROR` byte contains the value 0, if no error has occurred, otherwise an error number is passed.

6.2.5 Handling function FC 63 CANCTRL

The function block CANCTRL (FC 63) is used to reset errors or query information.

Parameter	Direction	Type	Example
Base	IN	INT	512
Chan	IN	INT	1
Func	IN	INT	1
IDHI	IN/OUT	WORD	MW 30
IDLO	IN/OUT	WORD	MW 32
DW0	IN/OUT	WORD	MW 34
DW1	IN/OUT	WORD	MW 36
DW2	IN/OUT	WORD	MW 38
DW3	IN/OUT	WORD	MW 40
ERROR	OUT	BYTE	MB 42
STAT	OUT	BYTE	MB 43

As the passed parameter, the base address (`Base`) and the channel (`Chan`) of the module as integers, the function (`Func`) and a status byte (`STAT`) must be passed.

Func: 1 = Error Reset: Overflow errors are reset
2 = Controller Reset: reset CAN controller errors

The status of the CAN 400 module is in the `STAT` byte. The byte is always assigned a value even if no frame has been transmitted.

The `ERROR` byte contains the value 0, if no error has occurred, otherwise an error number is passed.

All other parameter are for future use.

6.2.6 Content of the status byte STAT

In many applications it is necessary to transmit a series of frames to the module in a cycle. The FIFO circulating buffer is 256 frames (lowpriority) long.
If bit 4 of the status byte is set, it is possible to transmit up to 255 frames at once to the module.

The `STAT` status byte indicates the status of the channel:

Bit 0: Module & Channel parameterized and running

Bit 4: FIFO completely empty

Bit 5: FIFO more than half full, overflow imminent, the S7 should read out the FIFO faster, or not transmit any more frames

Bit 6: FIFO overflow

Bit 7: CAN controller group error

The bits of the FIFOs always refer to the direction from which they are viewed. When CANRCV is called, the status of the receive FIFOs is passed on in `STAT`. When CANSEND or CANTIMER is called, the status of the transmit FIFOs is returned.

Bit 6 and Bit 7 must be reset by calling FC 63 CANCTRL.

6.2.7 Interrupts via process alarm OB 40

The CAN 400 can only trigger alarms via interrupt line 1.
For CAN400-2 it should only 1 channel used with interrupts.
Do not use Interrupts in multicomputing systems.

When the "S7 interrupt" option is activated in the CAN project, the CAN 400 modules triggers an interrupt with each CAN frame ready for transmission to the PLC.

Only raised alarms are triggered.

`OB40_MDL_ADDR` indicates the address of the module

`OB40_POINT_ADDR` indicates the channel number. 1/2 for lowprior, 3/4 for highprior telegrams

6.3 CANopen communication

CANopen

ℹ

CIA = CAN in Automation e.V., Am Weichselgarten 26, 91085 Erlangen, Germany

⚠

CANopen always works with CAN 2.0A (11 bits).
This must be taken into account in configuration of the module with CANparam.

6.3.1 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) are not affected by the CAN bus.

The CANopen communication profiles for the various applications are managed by the CIA.

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 message frame have a fixed meaning.

Each device in a CANopen network has a fixed node ID (module number, 1-127).

6.3.2 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 message frame. SDOs that are longer are transmitted in more than one message frame. For very large volumes of data, SDO block transmission is possible.

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 message frame.

The last 4 bytes of the CAN message frame then contain the value for the SDO.

PDOs contain the "working values" of a 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 CANopen slave should have a directory containing the objects it supports.

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 message frame for reading and writing. These can be both existing SDOs and updated values of the slave (e.g. analog value or an input).

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

6.3.3 Functions

The CANopen functions are subdivided into the three basic groups:

- Reading and writing SDO
- Reading and writing PDO
- Netmanagement

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

It is possible to change some COB-IDs to other values using special service data objects (SDOs).
This is NOT supported by the CANopen handlingblock!

COB identifier (COB-ID):

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

Broadcast functions:

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

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

Node functions:

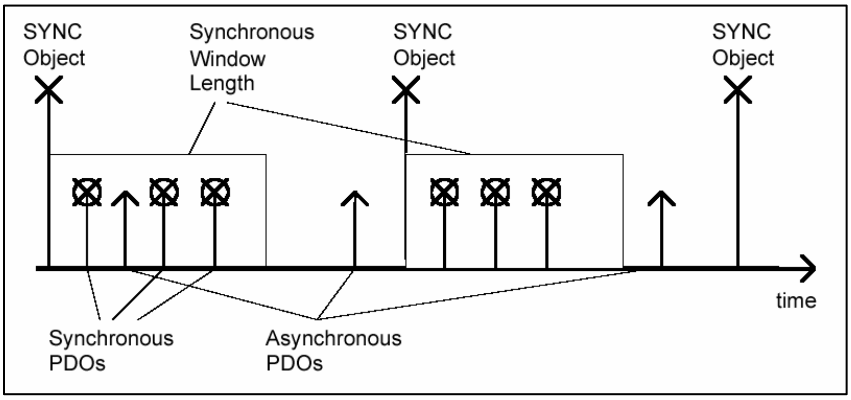
Function	Function code (binary)	Resulting COB-ID
EMERGENCY	0001	81h –FFh
PDO1 (tx)	0011	181h – 1FFh
PDO1 (rx)	0100	201h – 27Fh
PDO2 (tx)	0101	281h – 2FFh
PDO2 (rx)	0110	301h – 37Fh
PDO3 (tx)	0111	381h – 3FFh
PDO3 (rx)	1000	401h – 47Fh
PDO4 (tx)	1001	481h – 4FFh
PDO4 (rx)	1010	501h – 57Fh
SDO (tx)	1011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

6.3.4 Netmanagement

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

SYNC:

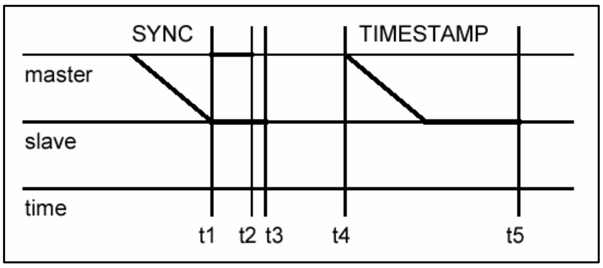
The SYNC message frame is a cyclic "broadcast" frame and sets the basic bus clock. To ensure isosynchronism, the SYNC frame has a high priority.
[COB-ID: 80h]



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]

Lifeguarding:

In Lifeguarding, each CANopen 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.

Heartbeat:

Heartbeat monitoring is equivalent to Nodeguarding although no request frames are generated by CANopen master. The heartbeat frame is transmitted automatically by the node and can be evaluated in the master.

Emergency message:

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

If a fault occurs on a CANopen 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 slaves generate a BootUp message after switch-on that the master can recognize to initialize this new station.
[COB-ID: 700h + node ID + 1 byte data: 00h]

6.3.5 CANopen handling blocks

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

The handling blocks for CANopen communication provide all the necessary functions to process SDOs and PDOs and perform network management.

The following description refers to Version 1.1 of the handling blocks.

The CAN 400 module works with these handling blocks as the *master* in the CANopen network.

Block	Function	User / System	Chapter
FC 40	Initialization (restart)	User	6.3.7
FC 41	Read SDO	User	1.1.1
FC 41	Transmit SDO	User	1.1.1
FC 42	SDO block download	User	6.3.11
FC 42	SDO block upload	User	6.3.11
FC 43	Spontaneous reception (NMT, PDO)	User	1.1.1
FC 44	Transmit PDO	User	1.1.1
FC 45	Request PDO	User	1.1.1
FC 46	CAN service	User	6.3.16
FC 47	Nodeguarding/Heartbeat	User	6.3.17
FC 48	Network management	User	6.3.15
FC 49	Cycle	System	1.1.1
DB-PDO	PDO data received	User	6.3.10
CAN-DB	Management DB	System	6.3.6

6.3.6 CAN-DB

One CAN-DB (length 300 bytes) containing the management information is required for each CAN channel. The CAN-DB is initialized by the FC 40 and used by all other FCs.

In this block, the CAN frames received and transmitted are stored before they are passed on and current jobs are managed.

6.3.7 FC 40 initialization

The FC 40 must be called up during startup of the PLC. The FC 40 initializes the CAN-DB so that all other CANopen data handling blocks can work correctly.

ℹ

FC 40 does not restart the module. It cannot therefore be used to reset the module!

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 40
BaseAddr	IN	INT	512 - ...	512
Chan	IN	INT	1 / 2	1
PDO_DB_1	IN	INT	0 – 2047	51
PDO_DB_2	IN	INT	0 – 2047	0
PDO_DB_3	IN	INT	0 – 2047	0
PDO_DB_4	IN	INT	0 – 2047	0

CanDB internal DB with current CAN data

BaseAddr base address of the module

Chan channel of the module (1 or 2)

PDO_DB_1..4 number of DBs for receiving the PDO 1..4 data of all nodes

6.3.8 PDO-DBs

The data of received PDO frames are automatically copied into DBs by FC 49 "Cycle". For that purpose, it is necessary to specify one DB for each PDO (1-4) during initialization (see 6.2.7).

Each DB contains space for 8 bytes PDO data for all 127 nodes. Each PDO-DB must therefore be at least 1024 bytes long.

		PDO1 DB	**PDO2 DB**	**PDO3 DB**	**PDO4 DB**
	DBB0	not used	not used	not used	not used
	...	not used	not used	not used	not used
	DBB7	not used	not used	not used	not used
Node 1	**DBB8**	1st byte of node 1 / PDO1	1st byte of node 1 / PDO2	1st byte of node 1 / PDO3	1st byte of node 1 / PDO4
	DBB9	2nd byte of node 1 / PDO1	2nd byte of node 1 / PDO2	2nd byte of node 1 / PDO3	2nd byte of node 1 / PDO4
	DBB10	3rd byte of node 1 / PDO1	3rd byte of node 1 / PDO2	3rd byte of node 1 / PDO3	3rd byte of node 1 / PDO4
	DBB11	4th byte of node 1 / PDO1	4th byte of node 1 / PDO2	4th byte of node 1 / PDO3	4th byte of node 1 / PDO4
	DBB12	5th byte of node 1 / PDO1	5th. byte of node 1 / PDO2	5th byte of node 1 / PDO3	5th byte of node 1 / PDO4
	DBB13	8th byte of node 1 / PDO1	6th byte of node 1 / PDO2	6th byte of node 1 / PDO3	6th byte of node 1 / PDO4
	DBB14	7th byte of node 1 / PDO1	7th byte of node 1 / PDO2	7th byte of node 1 / PDO3	7th byte of node 1 / PDO4
	DBB15	8th byte of node 1 / PDO1	8th byte of node 1 / PDO2	8th byte of node 1 / PDO3	8th byte of node 1 / PDO4
Node 2	**DBB16**	1st byte of node 2 / PDO1	1st byte of node 2 / PDO2	1st byte of node 2 / PDO3	1st byte of node 2 / PDO4
	...	...	...	...	...
	DBB23	8th byte of node 2 / PDO1	8th byte of node 2 / PDO2	8th byte of node 2 / PDO3	8th byte of node 2 / PDO4
...	**...**	...	...	...	...

The COB-IDs of the frames affected are permanently assigned:

PDO1	180h + Node-ID
PDO2	280h + Node-ID
PDO3	380h + Node-ID
PDO4	480h + Node-ID

If no DB is specified for a PDO (1-4), the data can be fetched with FC 43 Spontaneous reception (see 6.3.12).

6.3.9 FC 49 cycle

FC 49 must be executed in the cycle of the program. It transmits and receives the frames of the CAN 400 module and assigns the data to the jobs.

The FC 49 also copies the PDO data into the PDO receive DBs (see 6.3.7 FC 40 initialization).

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 40
T	IN	TIMER	T 0 – T 511	T 49
Buffer info	OUT	WORD	MW 0 – MW 1024	MW 140

CanDB internal DB with current CAN data, see FC 40 initialization

T Timers for internal use. If several CAN 400 modules are used in a set-up, a timer is required for each module.

Buffer info Display of the assigned receive buffers, or the current jobs (transmit buffers).

Lower byte for transmit buffer:

Bit 1 = SDO transmit buffer assigned. Do not start new jobs with FC 41 and FC 42.

Upper byte for receive buffer:

Bit 0 = PDO from slave received (COB-IDs 180h-4FFh) => call FC 43 function 0.

Bit 2 = Timestamp received (COB-IDs 100h-17Fh) => call FC 43 function 2

Bit 3 = NMT received (COB-IDs 00h – 7Fh) => call FC 43 function 3

Bit 4 = emergency frame received (COB-IDs 80h-FFh) => call FC 43 function 4

Bit 5 = NMT error frame received (COB-IDs 700h-77Fh) => call FC 43 function 5

Bit 6 = NMT service frame received (COB-IDs 780h-7FFh) => call FC 43 function 1

RcvStat Status of send channel

SndStat Status of receive channel

Bit 0: Module & Channel parameterized and running

Bit 4: FIFO completely empty

Bit 5: FIFO more than half full, overflow imminent, the S7 should read out the FIFO faster, or not transmit any more frames

Bit 6: FIFO overflow

Bit 7: CAN controller group error

FC 49 should be called several times in a cycle for long PLC cycles times. Each call processes no more than one job for transmission and one job for receiving, if a job is present.

If several CAN 400 modules or several channels are used in an assembly, FC 49 must be called with a different CanDB for each channel.

The bits in the Bufferinfo parameter can be used to optimize CAN handling.

6.3.10 FC 41 Reading and writing SDOs

With this FC you can read and write SDOs from a slave with up to 4 data bytes.

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 40
Node	IN	INT	1 – 127	2
Index	IN	WORD	0h – FFFFh	W#16#7300
Subindex	IN	BYTE	0h – FFh	B#16#1
Type	IN	BYTE	2Bh, 23h, 2Fh, 40h	B#16#2B
T	IN	TIMER	T 0-511	T 41
ReturnType	OUT	BYTE	MB 0 – MB 1023	MB 11
Status	OUT	BYTE	MB 0 – MB 1023	MB 10
Error	OUT	WORD	MW 0 – MW 1022	MW 16
Activate	INOUT	BOOL	M 0.0 – M 1023.0	M 1.2
Data	INOUT	DWORD	MD 0 – MD 1020	MD 12
AbortCode	INOUT	DWORD	MD 0 – MD 1020	MD 14

CanDB internal DB with current CAN data, see FC 40 initialization

Node Number of the CAN station

Index Index of the object

Subindex Subindex for the object

Type Size and direction of the object data:
40h = read SDO (8/16/32 bits),
23h = transmit 32 bits SDO,
2Bh = transmit 16 bits SDO,
2Fh = transmit 8 bits SDO

T Timer for timeout, if no response is received.

ReturnType Size of the object data received:
43h = 32 bits, 4Bh = 16 bits, 4Fh = 8 bits

Status Status byte of the job processing:
Bit 0 = job running,
Bit 5 = An abort code exists.
Bit 6 = Error (error number in Error)
Bit 7 = Job complete

Error Error number on error in execution

Activate Activation bit for starting the job; is reset after entry of the job

Data Transfer the data (reading and writing)

AbortCode Error number of the CANopen slave

The FC must by called up cyclically. SDO transmission is only triggered when the activation bit (Activate) is set. The FC resets the bit after acceptance of the job. The current status of job processing can be observed in the Status byte.

The FC enters the required job in the CAN-DB. However, the job is only performed (transmitted on the CAN bus) when FC 49 is called.

FC 42 must be used for transmission of SDOs with more than 4 bytes (see 1.1.1).

Example of call:

```
      AN    M      9.1                  // new SDO-Job ?
      AN    M    111.0                  // SDO-Job running ?
      JC    next

      CALL  FC    41
       CanDB    := DB 40
       Node     := MW 28
       Index    := MW 30
       Subindex := MB 32
       Type     :=B#16#40
       T        := T 41
       ReturnTyp:= MB 33
       Status   := MB 111
       Error    := MW 112
       Activate := M 9.1
       Data     := MD 34
       AbortCode:= MD 94

      AN    M    111.7                 // job finished ?
      JC    next
      A     M    111.6                 // ... with error ?
      JC    next

      L     MD    34                   // ... without error !
      ...                              // Store value...

next: ...
```

6.3.11 FC 42 Downloading/uploading an SDO block/segment

With this FC you can read and write SDOs from a slave with more than 4 data bytes. Transmission is performed with more than one frame on the CAN bus ("SDO block transfer" or "SDO segmented").

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 40
BlockDB	IN	INT	1 – 2047	151
StartByte	IN	INT	1 – 65533	0
Status	OUT	BYTE	MB 0 – MB 1023	MB 10
Error	OUT	WORD	MW 0 – MW1022	MW 12
AbortCode	OUT	DWORD	MD 0 – MD 1020	MD 14
Activate	INOUT	BOOL	M 0.0 – M 1023.7	M 1.2

CanDB internal DB with current CAN data, see FC 40 initialization

BlockDB Number of the CAN station

StartByte Index of the object

Status Status byte of the job processing:
Bit 0 = job running,
Bit 5 = An abort code exists.
Bit 6 = Error (error number in Error)
Bit 7 = Job complete

Error Error number on error in execution

Activate Activation bit for starting the job

AbortCode Error number of the CANopen slave

The information of SDO transmission must be stored in a DB:

Only one job from the DB is executed for each FC 42. It is not possible to concatenate several jobs.

Byte	Type	Example	Purpose
0	BYTE ;	1	Direction: 0= Block Upload, 1= Block Download, 2=Segment Upload, 3=Segment Download
2	WORD	4	Node
4	WORD	1004h	SDO index
6	BYTE	1h	SDO subindex
7	BYTE	32d	Size (number of bytes)
8	ARRAY 1...n		
	BYTE		Data
	ENDARRAY		

The FC must by called up cyclically. SDO transmission is only triggered when the activation bit (Activate) is set. The FC resets the bit after acceptance of the job. The current status of job processing can be observed in the Status byte.

!

Timeout monitoring of the jobs must be performed by the S7 application. The response times of the CANopen slaves can be very different.

The FC enters the required job in the CAN-DB. The job is only executed when FC 49 is called.

If no response comes from the CANopen slave, the current job will be deleted with FC 46.

6.3.12 FC 44 Transmit PDO

This FC transmits a PDO with data to a slave.

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 40
Node	IN	INT	0 – 127	2
PDO	IN	INT	1 – 4	1
Length	IN	INT	1 – 8	4
Data1234	IN	DWORD	0 – FFFFFFFFh	W#16#10203040
Data5678	IN	DWORD	0 – FFFFFFFFh	W#16#00000000
Status	OUT	BYTE	MB 0 – MB 1023	MB 10
Error	OUT	WORD	MW 0 – MW1022	MW 12

CanDB internal DB with current CAN data, see FC 40 initialization

Node Number of the CAN station

PDO Number of the PDO

Length Length of the frame data

Data1234 The first 4 bytes of data (bytewise left -> right)

Data5678 The last 4 bytes of data (bytewise left -> right)

Status Status byte of the job processing:
Bit 6 = Error (error number in)
Bit 7 = Job complete

Error Error number on error in execution

The data bytes are transferred to the PDO frame bytewise from left to right. If only one byte is to be transmitted, for example, it must be in the top 8 bits of parameter Data1234 .

The FC passes on the PDO job immediately to the module, calling the FC 49 is not necessary. That way several PDO jobs can be transmitted in succession in one cycle!

6.3.13 FC 45 Request PDO

This FC request a PDO from a slave. A PDO frame is transmitted with an RTR bit set. The slave should then transmit a PDO with its current data.

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 40
Node	IN	INT	0 – 127	2
PDO	IN	INT	1 – 4	1
Status	OUT	BYTE	MB 0 – MB 1023	MB 10
Error	OUT	WORD	MW 0 – MW1022	MW 12

CanDB internal DB with current CAN data, see FC 40 initialization

Node Number of the CAN station

PDO Number of the PDO

Status Status byte of the job processing:
Bit 6 = Error (error number in Error)
Bit 7 = Job complete

Error Error number on error in execution

The data of the response frame are then found in the PDO-DB , or can be fetched with FC 43 (see 1.1.1).

The FC passes on the PDO job immediately to the module, calling the FC 49 is not necessary. That way several PDO jobs can be transmitted in succession in one cycle!

6.3.14 FC 43 Spontaneous reception

With this FC it is possible to fetch frames that are received from the CAN bus without the associated job.

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB1 – DB 2047	DB 40
Func	IN	INT	1 – 5	4
Status	OUT	BYTE	MB 0 – MB 1023	MB 10
Error	OUT	WORD	MW 0 – MW1022	MW 12
Node	OUT	INT	MW 0 – MW 1022	MW 14
PDO	OUT	INT	MW 0 – MW 1022	MW 16
Length	OUT	WORD	MW 0 – MW 1022	MW 18
Data1234	OUT	DWORD	MD 0 – MD 1020	MD 20
Data5678	OUT	DWORD	MD 0 – MD 1020	MD 24

CanDB internal DB with current CAN data, see FC 40 initialization

Func Query function:
0 = PDO frames from slave without defined PDO-DB (COB-ID 180h-1FFh, 280h-2FFh, 380h-3FFh, 480h-4FFh)
1 = NMT-service frames (COB-ID 780h-7FFh)
2 = TimeStamp frames (COB-ID 100h-17Fh)
3 = NMT frames (COB-ID 00h-7Fh)
4 = Emergency frames (COB-ID 81h-FFh) and SYNC frames (80h)
5 = NMT error frames (COB-ID 700h-77Fh)

Status Status byte of the job processing:
Bit 6 = Error (error number in error)
Bit 7 = Data received

Error Error number on error in execution

Node Number of the CAN station

PDO Number of the PDO (for function 0)

Length Length of the frame data

Data1234 The first 4 bytes of data (bytewise left -> right)

Data5678 The last 4 bytes of data (bytewise left -> right)

If there is a new frame, FC 43 is exited with status bit 7, otherwise an error is output. Parameter Buffer info in the FC 49 cycle call specifies whether a receive buffer is assigned.

For each type of frame (see parameter Func) there is only one receive buffer in the CAN-DB. For that reason, this FC must be called up straight after the FC 49. Of course, that only applies if the frames are important for the S7 application. Otherwise the unimportant frames are ignored or filtered out with the acceptance forms of the module from the very beginning (=> lower cycle time load).

6.3.15 FC 48 Network management

FC 48 can be used to transmit network management frames.

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 40
Node	IN	INT	0 – 127	2
Func	IN	INT	1 – 6	1
Status	OUT	BYTE	MB 0 – MB 1023	MB 10
Error	OUT	WORD	MW 0 – MW1022	MW 12

The scripts should be used for execution of network management functions during startup or stop of the CAN 400 module (see 1.1)!

CanDB internal DB with current CAN data, see FC 40 initialization

Node Number of the CAN station (Node=0 -> all nodes) or timer number (functions 10 & 11)

Func Network management function:
1 = Start Node
2 = Stop Node
3 = Disconnect Node
4 = Enter Preoperational
5 = Reset Node
6 = Reset Communication
10 = Timer Start (timer number in Node)
11 = Timer Stop (timer number in Node)
12 = Reset errors in the module

Status Status byte of the job processing:
Bit 6 = Error (error number in Error)
Bit 7 = Job executed without error

Error Error number on error in execution

The FC passes on the PDO job immediately to the module, calling the FC 49 is not necessary.

6.3.16 FC 46 Service

To delete hanging jobs.

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 40
Func	IN	CHAR	'S', 'C'	'S'
Status	OUT	BYTE	MB 0 – MB 1023	MB 10
Error	OUT	WORD	MW 0 – MW 1020	MW 10

CanDB internal DB with current CAN data, see FC 40 initialization

Func Service function:
S = Delete SDO job
C = Delete all jobs (transmit and receive)

Status Status byte of the job processing:
Bit 6 = Error (error number in Error)
Bit 7 = Job executed without error

Error Error number on error in execution

6.3.17 FC 47 Nodeguarding/Heartbeat

FC 47 can be used for transmitting Nodeguarding frames or receiving Heartbeat frames.

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 40
BeatDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 47
GuardTimer	IN	T	T 0 – T 511	T 47
TMTimer	IN	T	T 0 – T 511	T 48
OnlyReceive	IN	BOOL	TRUE/FALSE	FALSE
Error	OUT	WORD	MW 0 – MW1022	MW 12

CanDB internal DB with current CAN data, see FC 40 initialization

BeatDB DB with list of the slaves that exist (are expected) in the network (see below)

GuardTimer Timer for a delay between to queries

TMTimer Timer for a response timeout of the slave

OnlyReceive False = Nodeguarding frames are transmitted, and a response expected from the slave
True = only Heartbeat frames are received and entered in the BeatDB

Error Error number on error in execution

A DB containing a list of slaves to be monitored (BeatDB) is required to monitor the slaves using Nodeguarding.

Byte	Type	Example	Purpose
0	BYTE	20	SendDelay: 20_{dec} x 10 = 200ms time between two Nodeguarding frames
1	BYTE	50	SendDelay: 50_{dec} x 10 = 500ms Timeout for response frames of the slave
2	BYTE	-	Counter (used internally)
3	BYTE	-	not used
4	BYTE	1	Node number of slave 1
5	BYTE	0	Received status of slave 1
6	BYTE	2	Node number of slave 2
7	BYTE	0	Received status of slave 2
8	BYTE	3	Node number of slave 3
9	BYTE	0	Received status of slave 3
…	…	…	…
X	BYTE	0	End of list
X+1	BYTE	0	

If Heartbeat frames are used, timeout monitoring must be programmed in the application!

If no Nodeguarding frames are transmitted by the master (OnlyReceive = True), the timeouts have no effect.

If a Heartbeat frame is received, the status of this telegram is entered in the BeatDB of the machine node number. The user should then evaluate it and overwrite it with zero. Timeout management is not necessary here.

6.4 Explanation of the example program

The example program supplied with the CANopen handling demonstrates use of the handling blocks in a very simple form. The functions of the handling blocks are triggered by the bits of input byte 8.

A simple CANopen IO slave with 8 outputs and 8 inputs was used as node 2. The inputs have been wired directly with the outputs for this test.

CAN 400 module must be configured on slot 5 with address 576. The handling is initialized in OB100. DB40 is used as the CAN-DB and DB51 as the PDO1 data DB.

The example project "C4 CANopen Test.PAR" (installed with the CANParam) must have been imported into the CAN 400 module. The baudrate of the CAN bus can be selected with the DIP switches.

6.4.1 Example FC 10 (cycle/SDO/PDO/network management)

At the start of FC 10, a cycle block FC 49 (→ Section 1.1.1) is called to fetch received frames via the CAN bus or execute transmit jobs. The buffer info is stored in MW 10 and indicated at QW 0.

In the second network, it is possible to activate Nodeguarding with input bit 8.7 via FC 47 (→ 6.3.17). DB47 contains a list with the node numbers 1+2+3 that are queried cyclically.

In network 3, first of all the first two bytes of the PDO1 are loaded from the POD1-DB from Node 2 and output to QW 2.

With input bit 8.0, it is now possible to trigger cyclic transmission of the PDO1 to Node 2 (FC 44, → 1.1.1). The value is always incremented by 1 and transmitted, if the last value has been returned from the station via the receive PDO1. Remove the comparison from these lines if you want to transmit in each cycle. The data are located in MB12 - MB19 and only MW12 is incremented.

Network 4 contains fetching and transmitting an SDO (FC 41, → 0) via input bits 8.1 and 8.2. The parameters passed have been routed to MW. A variable table (VAT_1) for testing has been stored in the project.

In Network 5, FC 48 (→ 6.3.15) is called for the network management. Please note that in the example project "CANopen Test.PAR", the scripts for start and stop the CPU already contain the CAN frames for "NMT start all nodes" and "NMT stop all nodes".

6.4.2 Example FC 11 (spontaneous receipt)

The FC 11 is called in OB 1 after FC 10. Here "unexpectedly" received frames are fetched from the CAN-DB. This is controlled via the information in MW 10 that contains the buffer info from FC 49.

These functions are only necessary if the application requires the frames. It is not mandatory to fetch the frames.

6.5 Error numbers

Possible error numbers of the `Error` return parameter.

Number	Meaning
1	Node below 1
2	Node above 127
3	PDO below 1, or timer number below 1
4	PDO above 4, or timer number above 16
5	Function not defined
6	No data available
7	Timer number wrong
11	Node below 0
12	There is no EndSegmentMode for UP
15	Initiation still there
16	No response expected
17	Node incorrect
18	Index incorrect
19	Subindex incorrect
22	ComSpec incorrect for DN
23	ComSpec incorrect for DN
24	ComSpec incorrect
25	ComSpec incorrect for DN
26	ComSpec incorrect for SDO write
27	ComSpec incorrect for SDO read
31	BlockSize incorrect for DN
32	DB block too small
33	DB block undefined
35	DB block too small
36	DB block write-protected
80	Toggle bit set incorrectly
90	ComSpec incorrect for UP
91	Expidited incorrect for UP
92	ComSpec incorrect for UP
93	ComSpec incorrect for UP
94	Segment number incorrect for UP
94	Segment number incorrect for DN
99	Timeout for SDO job, no response from the CANopen slave
101	Buffer allocated, busy with a job.
102	Abort code received
105	Function number unknown
140	Module not ready
141	Module cannot transmit (buffer overflow ?)
142	System error
254	System error node scan
255	Function code undefined

6.5.1 Abort codes

Please also observe the error numbers stated directly for the data handling blocks!

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

You will receive these error messages if you have requested SDO transmission (FC 41, FC 42).

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 parameter incompatibility
0604 0047h	General incompatibility in the device
0606 0000h	Access violation due to a hardware error
0607 0010h	Data type does not fit, length of the service parameter does not fit
0607 0012h	Data type does not fit, length of the service parameter too large
0607 0013h	Data type does not fit, 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 Appendix

7.1 Technical data

Order number	CAN 400-1	700-640-CAN11
	CAN 400-2	700-640-CAN21
Dimensions	290 x 210 x 25 mm (LxWxH)	
Weight	Approx. 900g	

CAN interface

Type:	ISO/DIN 11898-2, CAN High Speed physical Layer
Transmission rate:	10 kbps to 1Mbps
Protocol:	CAN 2.0A (11bit) CAN 2.0B (29bit) CANopen master CANopen slave *on request*
Connector:	Connector, SUB D 9-way

Configuration interface USB

Type:	USB 1.1
Transmission rate:	Fullspeed 12Mbps
Connector:	USB-A socket

Configuration interface serial

Type:	RS232
Transmission rate:	115Kbaud
Connector:	3-way jack
Format:	8/N/1

Power supply

Voltage:	+5V DC on backplane bus
Current consumption	CAN400-1: 560mA, 2.8 Watt
Current consumption	CAN400-2: 600mA, 3 Watt

Special features

Quality assurance:	According to ISO 9001:2000
Maintenance:	Maintenance-free (no battery, rechargeable or non-rechargeable)

7.2 Pin assignment

7.2.1 CAN connector

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

7.3 Further documentation

Internet: 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