compatible with you

CAN 300

CAN Communication Module for S7-300
with CANopen- or CAN-Layer 2 handling blocks

700-600-CAN01

700-600-CAN81

Manual

Edition 13 / 22.06.2006 for HW1 & FW2.5 and above

Manual order number: 900-600-CAN01/en

Systeme Helmholz **GmbH** ● Gewerbegebiet Ost 36 ● D-91085 Weisendorf

Phone: +49 9135 7380-0 ● Fax: +49 9135 7380-50 ● E-Mail: info@helmholz.de ● Internet: www.helmholz.de

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

gives information

hazard, general or specific

*danger of **electric shock***

1.1 General

The CAN 300 Master 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.

1.2 Restriction of access

Only authorized persons must have access to the modules!

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 Chapter 2.

1.3 Information for the user

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

2 Installation and Mounting

The CAN 300 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: 0 °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.

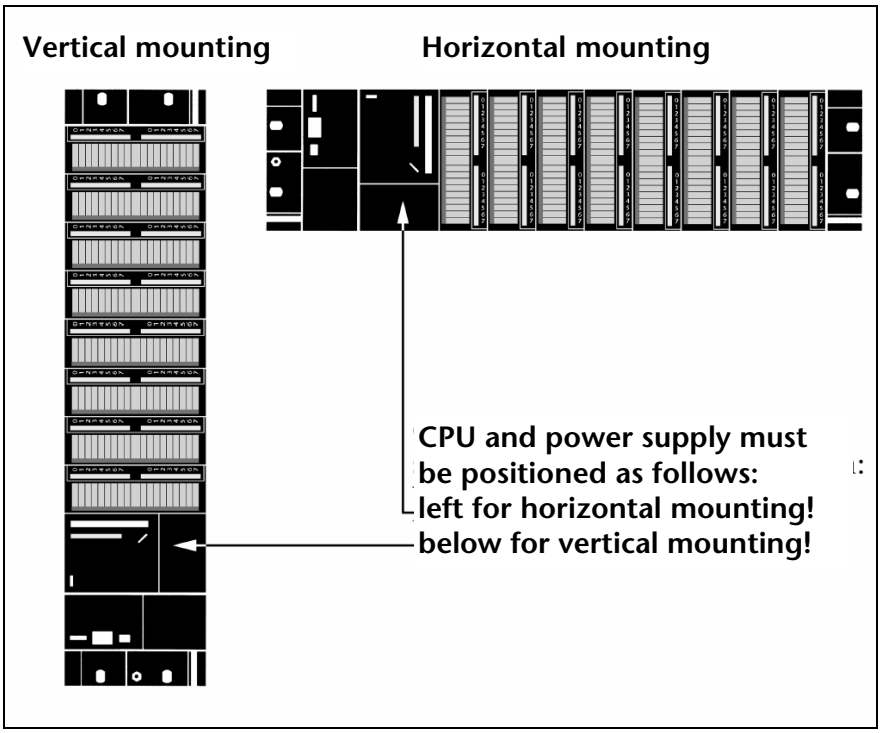
2.1 Vertical and horizontal mounting

The modules can be mounted either vertically or horizontally.

Permissible ambient temperature:

for vertical mounting: from 0 to 40 °C

for horizontal mounting: from 0 to 60 °C



2.2 Minimum clearance

Minimum clearances must be observed because

it ensures cooling of the CAN 300 modules

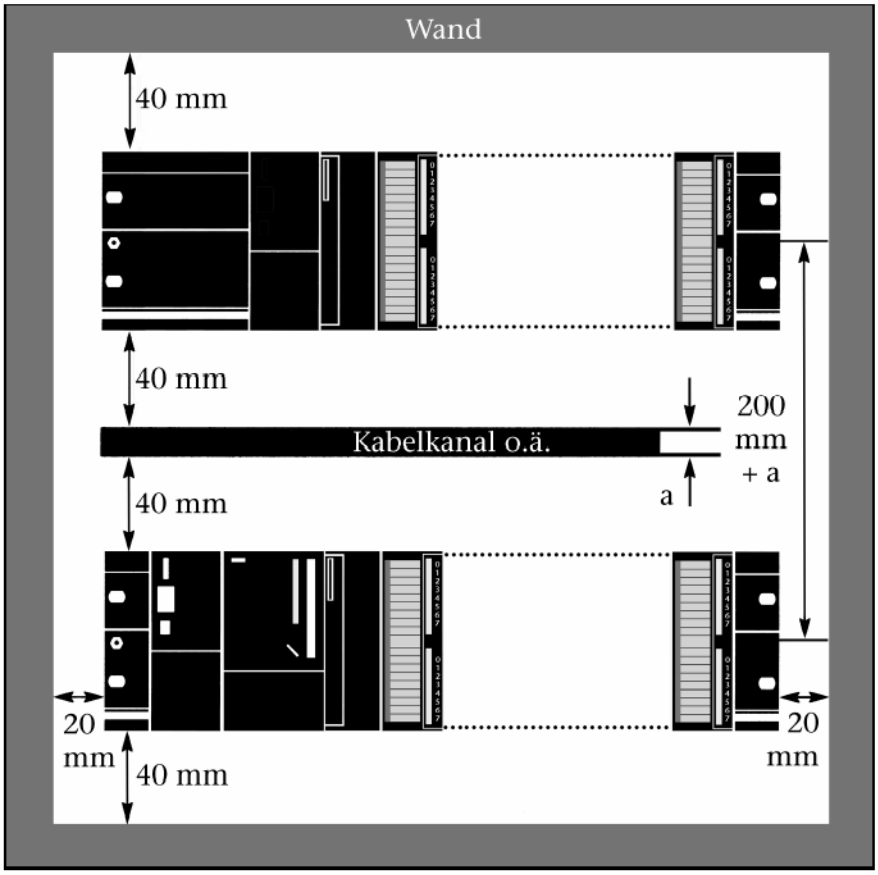
it provides space to insert and remove modules

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.

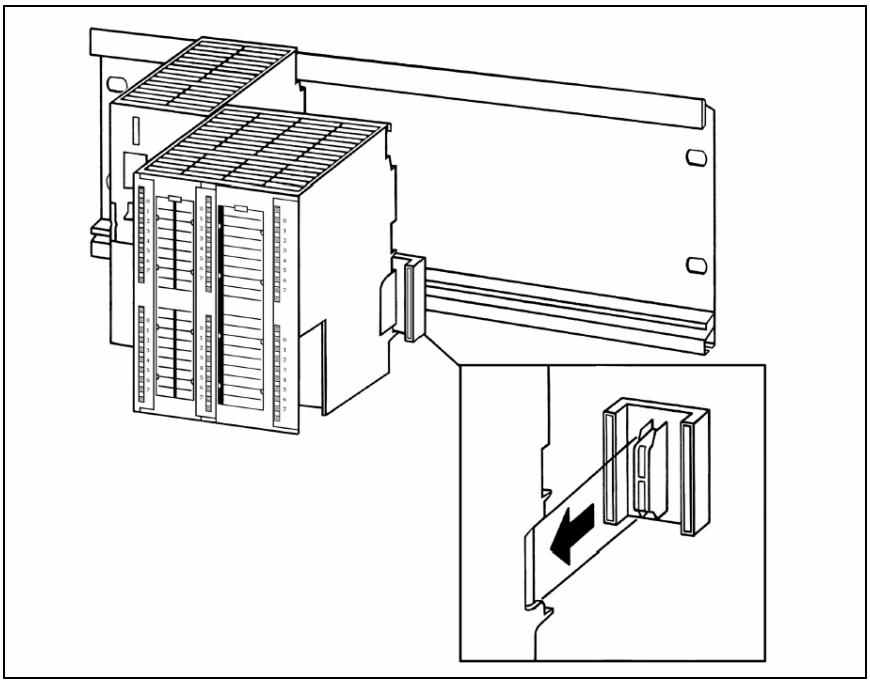
Non-observance of the minimum distances can destroy the module at high ambient temperatures!



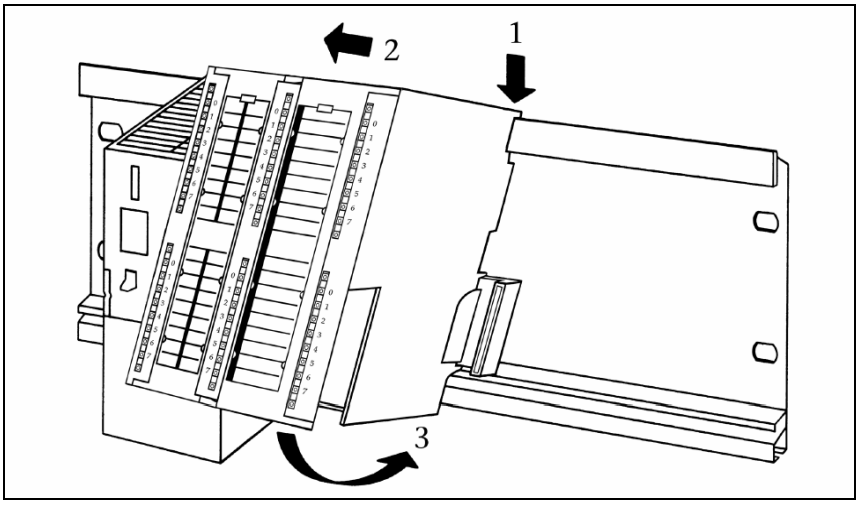
2.3 Mounting of the module on the DIN rail

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

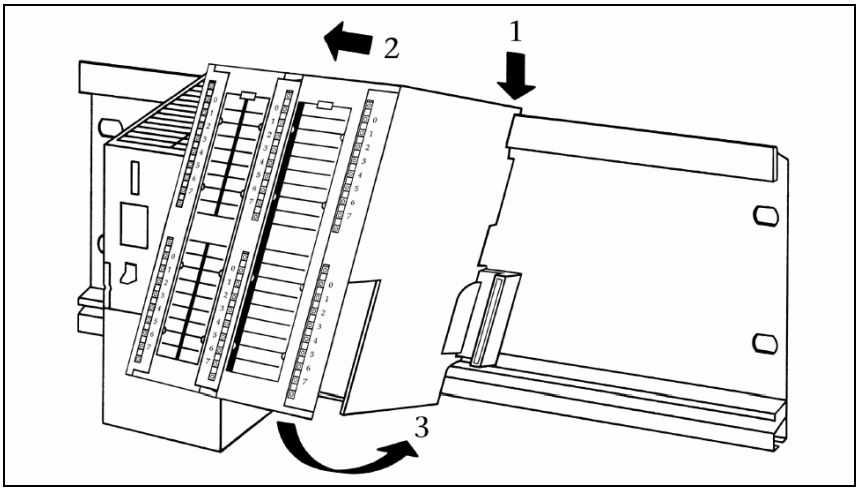
Take the bus connector off the last module and insert it into the CPU. Do not plug a bus connector into the last module of the tier.



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



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



3 System Overview

3.1 Application and function description

The CAN 300 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 300 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.

The handling blocks that permit simple handling of CAN communication are supplied as source code. Handling blocks are available for layer 2 and for CANopen master communication.

Ask also for special CANopen Slave or for LENZE Systembus handling blocks.

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

The CAN 300 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, or freely editable bit timing.

The CAN 300 module contains the 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 CAN bus response with up to 512 frames that is execute automatically by the module when the event occurs.

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

11 freely settable timers are available in the CAN 300 module. Each 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 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 module after the parameterized time has elapsed.

3.2 Connections

The CAN 300 module has two 9-pin SubD connectors behind the front flap.

The upper connector is for the CAN bus, the lower SubD connectors is the RS232 interface with the PC for configuration the module.

Pin assignment:

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

Pin	SUBD connector RS232	SUBD connector CAN
1	-	-
2	Rx	CAN Low
3	Tx	CAN GND
4	-	-
5	GND	-
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 300 module does not contained an integrated terminating resistor.

The CAN 300 module does not contain an integrated terminating resistor.

Check for correct cabling in the Debug dialog box of the CANParam (see Chapter 5.9)

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 RUN (green):

Continuous light indicates that the module is in cyclic operation.
Blinking light indicates that the module is starting up or that the PLC is in the stop state. Communication with the CPU is then not possible. CAN bus frames cannot be transmitted or received then either.

LED SF (red):

A serious error has occurred on the module. Check debug screen with CANParam.

LED CAN (yellow):

CAN bus active: Indicates running communication (transmitting and receiving) via the CAN bus.

CAN-frames that are transmitted via the timer of the module are not indicated by the yellow LED.

3.5 Scope of supply

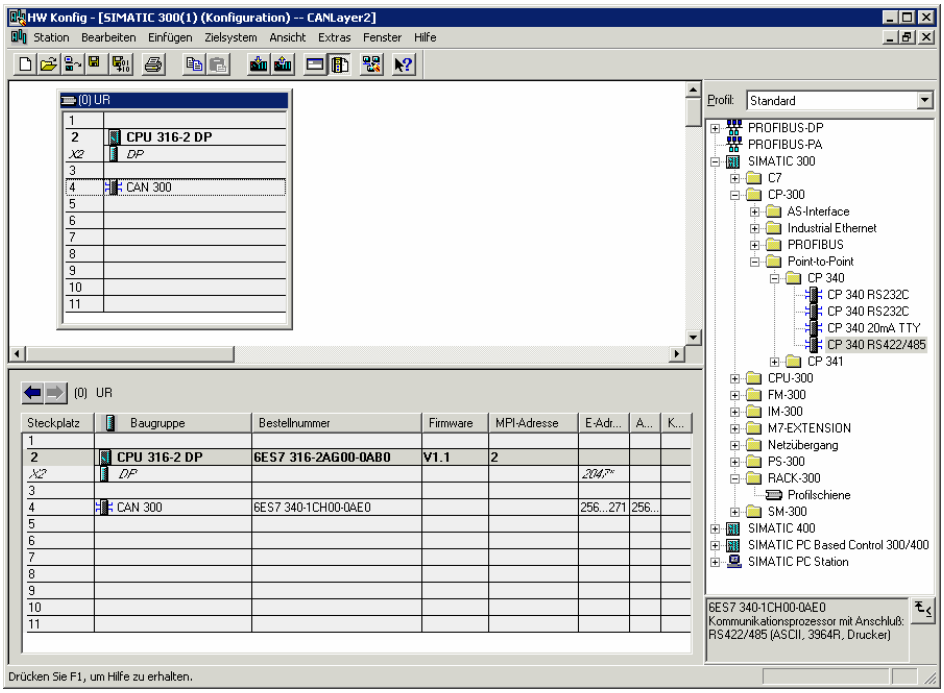
CAN 300 module, Bus connector

3.6 Accessories

CAN CD with Parameterization software “CANParam”, “Layer 2” and “CANopen” handling blocks	800-600-CAN01
CAN CD with parameterization software "CANParam", "Layer 2" and "LENZE systembus" handling blocks	800-600-1LZ11
Manual, german / english	900-600-CAN01
Programming cable PC <-> CAN 300 module	700-610-0VK11
CAN bus connector	700-690-0BA11
CAN bus connector with 2. connector	700-690-0BB11
CAN bus connector axial	700-690-0CA11

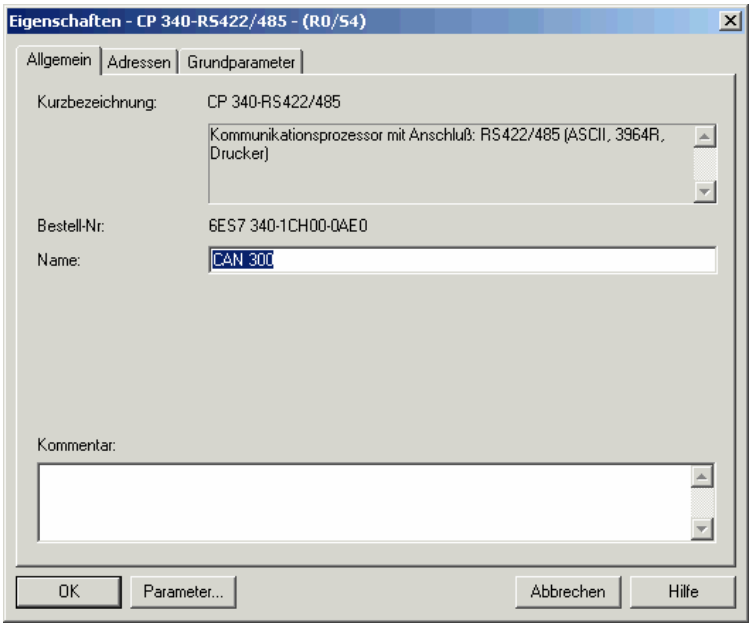
4 Configuration in the PLC

The CAN 300 module is configured as the communication module in the programming software of the PLC. On the installation CD you find a Step 7 project that already contains a configured module and the required data handling blocks.



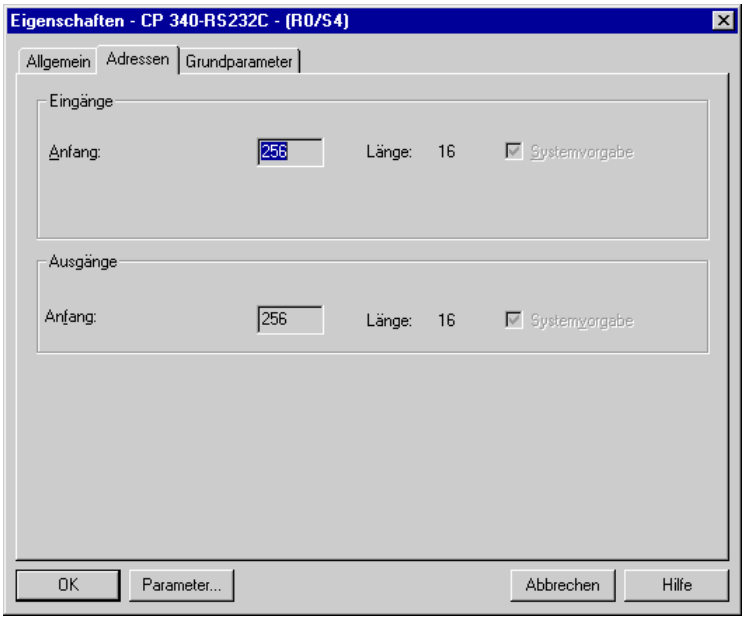
The module can be used wherever a CP module is allowed, i.e. also in the expansion unit after an interface module.

It is not possible to use the CAN300 module in a ET200M !



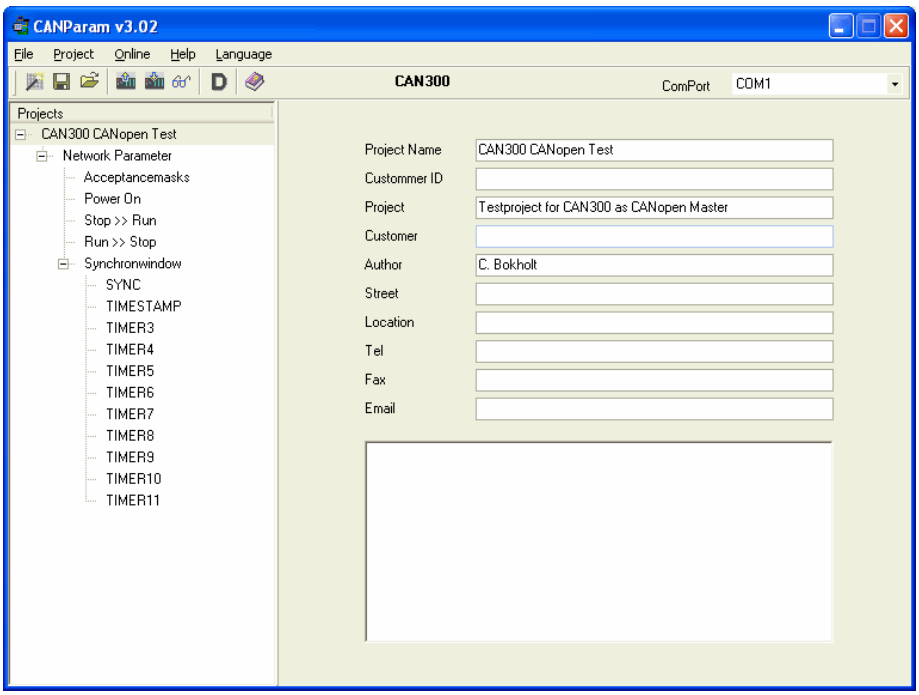
In parameterization of the module, only the range of I/O addresses is relevant. All other settings have no effect on the module.

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



5 Configuration of the CAN 300 module

The CAN 300 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.

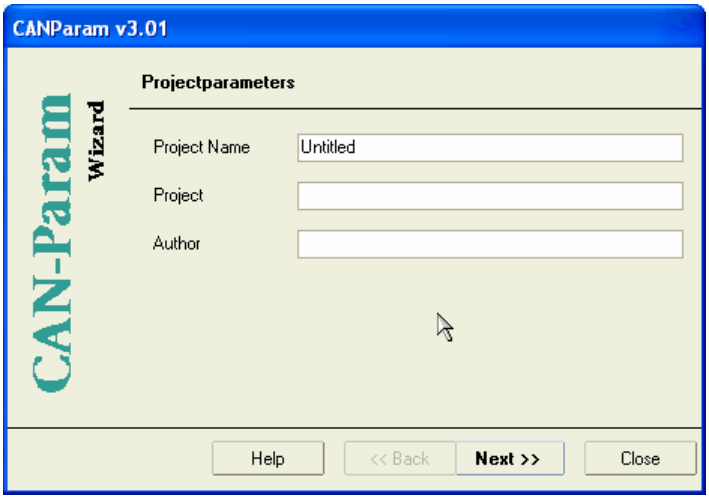


The configuration of a module can be stored in a project file on the PC.

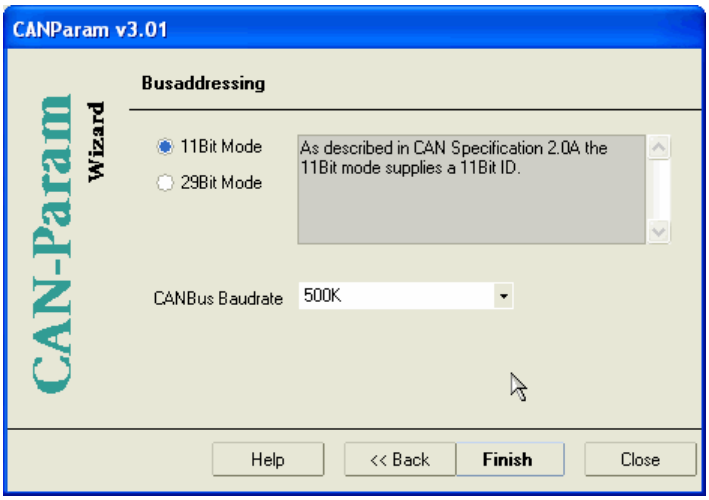
You can use a normal commercial type null modem cable to link the PC to the CAN 300 module (see also 7.3). After installation and starting of the CANParam software, you should set the interface top right on the menu bar.

5.1 Create new project

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

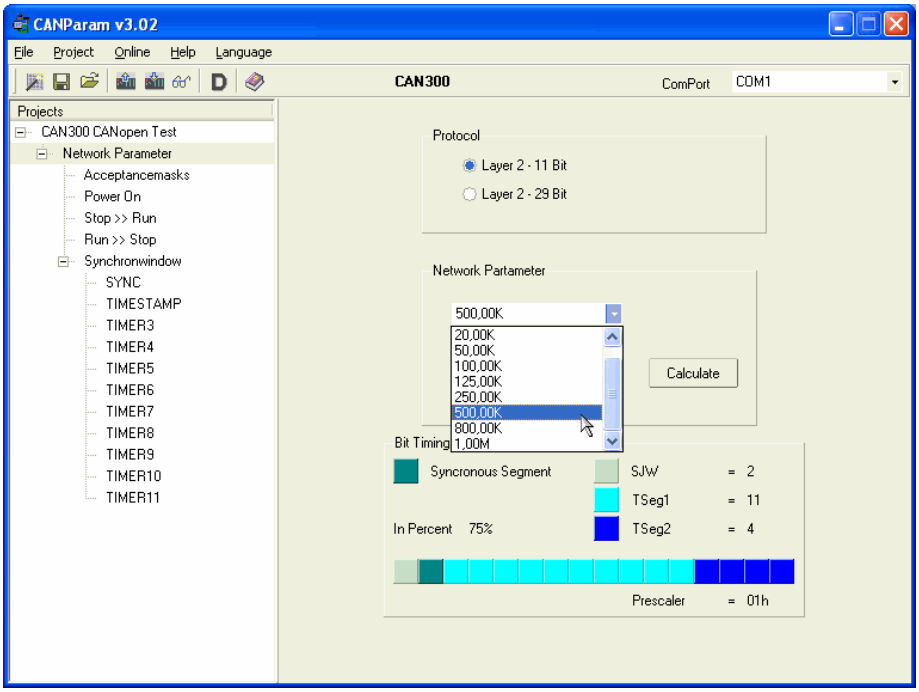


The project wizard guides you through the most important settings to obtain a new and complete project.



5.2 Setting the CAN bus baudrate

You can select the CAN baudrate in the range from 10kbps to 1Mbps in a fixed scale, or enter any baudrate.



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.

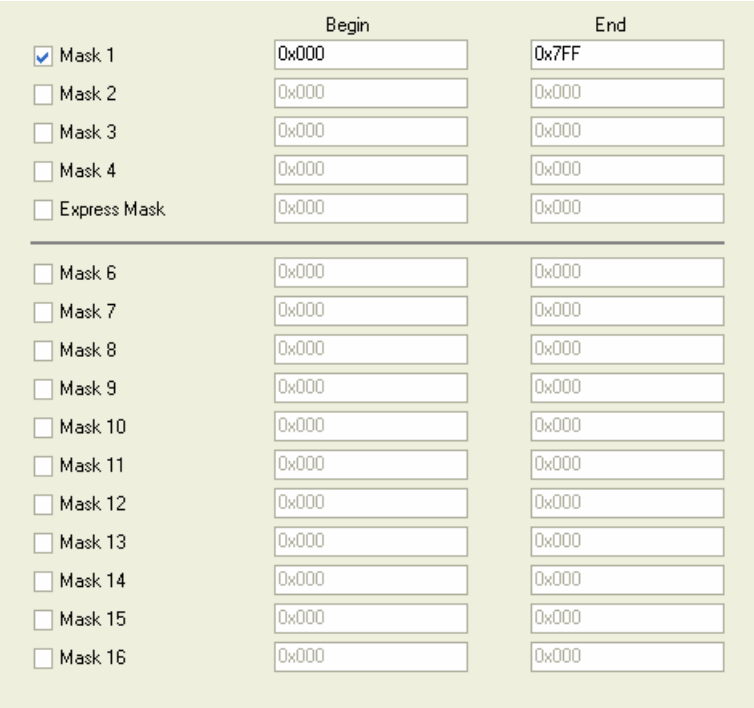
5.3 Setting the transmission mode (protocol)

The CAN 300 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 available in the CAN 300 module. Using these masks you can enable or block various frame IDs for receiving.



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

With the acceptance masks "Express Mask" it is possible to handle high priority CAN frames. Received frames with the accepted IDs of this mask, are routed directly to the S7 CPU without using the receive buffer.

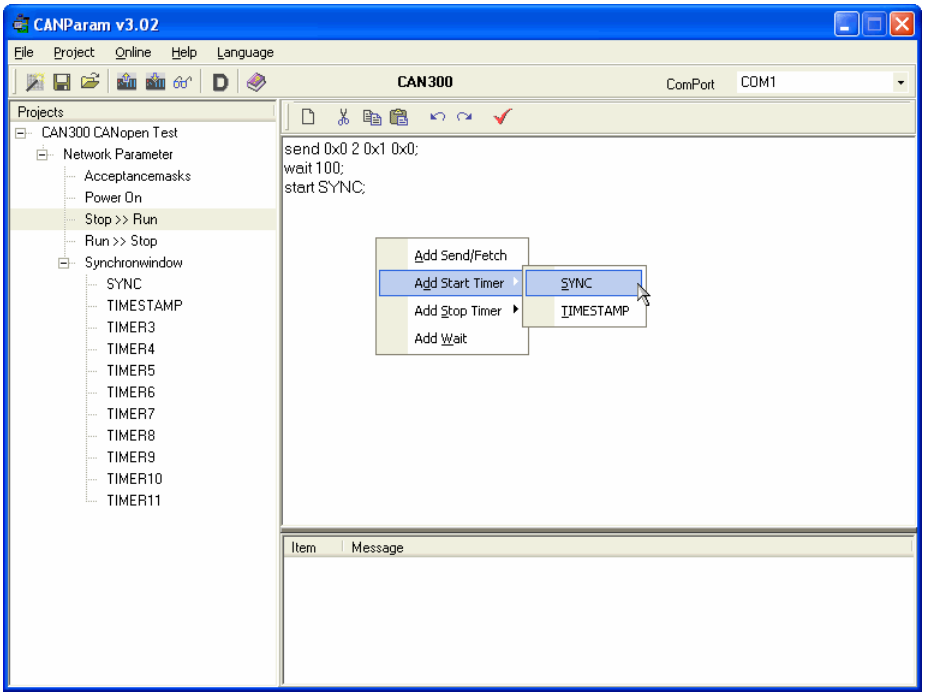
5.5 Network management

The CAN 300 module can transmit freely programmable frames for the PLC events "Power ON", "Stop -> Run", and "Run -> Stop", and start and stop timers.

The following commands are available:

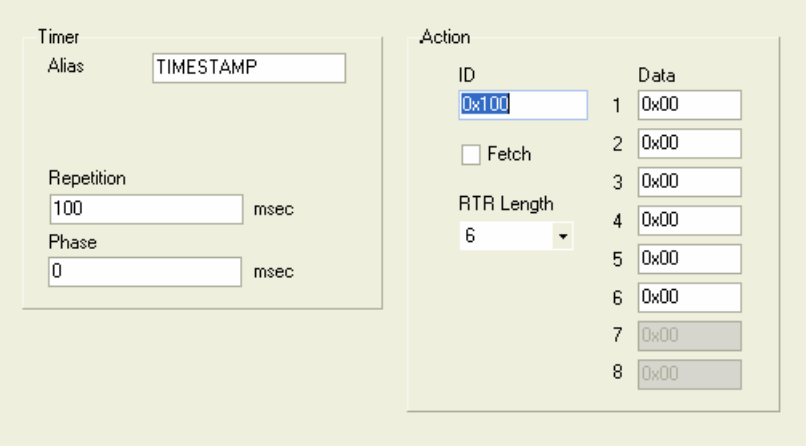
Send	Transmit frame (Structure: ID, length, data byte 1, data byte 2, etc.)
Fetch	Transmit frame with RTR bit 1
Start	Start Timer X
Stop	Stop Timer X
Wait	Wait X ms
//	Comment line

The steps in the scripts are executed in a timebase of 50ms.



5.6 Timer

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



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

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

For the timer *repetition period*, times from 5 msec. to 1 sec. can be set in steps of 5 msec. For the *phase* 0msec to 5 msec before the period duration.

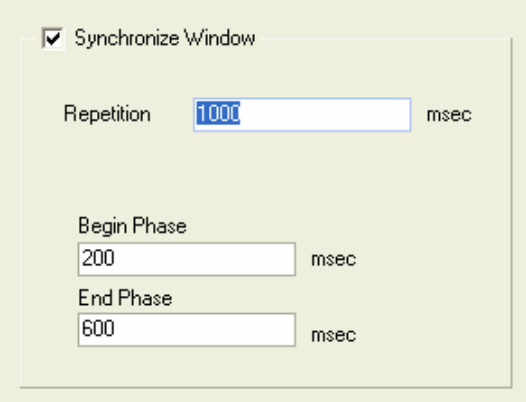
The data of the CAN frame defined for the timer are initialization data and can be overwritten by the S7-CPU in cyclic operation by the FC 63 "CANSYNCSEND".

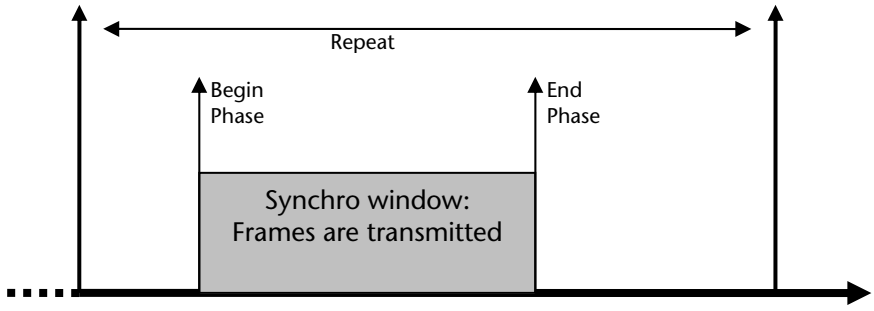
5.7 Synchro window

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



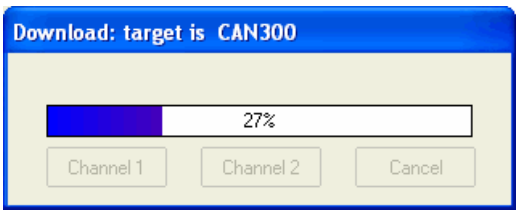


Timer 1 is needed internally, if the synchronous window is used.

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

5.8 Download

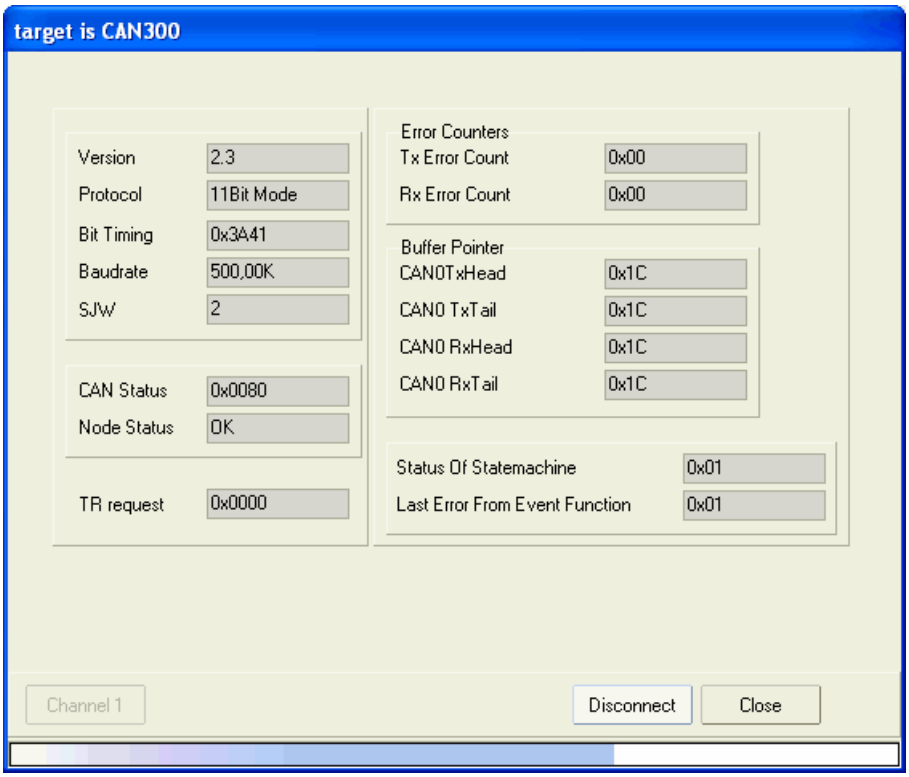
The project currently being worked on can be imported into the CAN 300 module again at any time ("download").



Updload of a project from the CAN 300 module is not possible.

5.9 Diagnostics/debugging

To simplify debugging, you can query the status of the CAN 300 module with menu item "Debug". Debug mode requires a serial link with the module.



You can activate monitoring mode with the "Connect" button. Click the button again to disconnect.

With the "Active ON" button, online monitoring mode is activated. If you press the button again, the link will be disconnected again.

The debug dialog provides the following information:

Version Version number of the CAN 300 operating system

Protocol Configured CAN protocol (11bit/29bit/etc.)

Bit Timing Content of the bit timing register, baudrate and SJW are displayed in plain text

CAN Status Content of the CAN status register:
Bit 15: Transmitting
Bit 14: Receiving
Bit 10: Node state transition occurred
Bit 9+8: Node state (also displayed in plain text)
Bit 7: Enable Tx
Bit 0: Halt/Bus Off State of CAN Controller

Node Status Content of bits 9+8 of CAN status register:
”OK”, “Warning”, “Error passive”, “Bus Off”

!
Node Status has to be on „OK“, for enabling CAN communication.

TR Request Display of the transmit & receive request:

Bit 15:	Script processing
Bits 13 & 14:	Asynchronous transmit buffer
Bit 12:	Lifeguarding
Bit 11 - 1:	Timer 10 – 0
Bit 0:	Receive

Error counters TX: Error counter transmit (active & passive)
RX: Error counter receive (passive)

Note: The transmit and receive error counters 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!

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

Buffer pointers Display of the circulating buffer pointers:

TX Head:	Transmit data from S7
TX Tail:	Data transmitted via CAN
RX Head:	Receive data from CAN
RX Tail:	Receive data transmitted to S7

Note: The CAN 300 module has a receive buffer and a transmit buffer for 128 telegrams each. The buffer pointers indicate to what extent the buffers are full. For example, if a CAN telegram has been received, “Rx Head” is incremented. If the telegram has then been passed on to the PLC (fetched by the data handling block), “Rx Tail” 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.

Status of state machine

Bit 0: Module running, read-in of the parameters completed
Bit 1: Script processing "POWER ON" running
Bit 2: Script processing "STOP->RUN" running
Bit 3: Script processing "RUN->STOP" running
Bit 4: CAN transmit FIFO full, stop sending
Bit 5: CAN receive FIFO more than half full, Overflow immanent, the S7 must read the FIFO faster
Bit 7: PLC in STOP

Last error from Event function

Last error of script processing

6 Programming in the PLC

6.1 Overview

The CAN 300 module is programmed in the PLC using the handling blocks supplied.

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

3 FCs are available for layer 2 communication:

FC 60	**CANSEND**	Transmission of a CAN frame (asynchronous)
FC 61	**CANRCV**	Receiving a CAN frame
FC 63	**CANSYNCSEND**	Transmission of a CAN frame (synchronous)

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:

```
      CALL  FC    61
       Base  :=256
       IDHI  :=MW60
       IDLO  :=MW62
       RTRLEN:=MW64
       DW0   :=MW70
       DW1   :=MW72
       DW2   :=MW74
       DW3   :=MW76
       STAT  :=MB66
       Recd  :=M67.0

      AN    M     67.0
      JC    send
      ...

send: SET
      =     M     87.0

      CALL  FC    60
       Base  :=256
       IDHI  :=MW80
       IDLO  :=MW82
       RTRLEN:=MW84
       DW0   :=MW90
       DW1   :=MW92
       DW2   :=MW94
       DW3   :=MW96
       STAT  :=MB86
       Snd   :=M87.0
```

6.2.2 CANSEND handling function

The CANSEND function block (FC60) transfers a CAN frame to the module from which it is transmitted immediately.

Parameter	Direction	Type	Example
Base	IN	INT	256
IDHI	IN	WORD	DBW20
IDLO	IN	WORD	DBW22
RTRLEN	IN	WORD	DBW24
DW0	IN	WORD	DBW26
DW1	IN	WORD	DBW28
DW2	IN	WORD	DBW30
DW3	IN	WORD	DBW32
STAT	OUT	BYTE	MB 3
Snd	IN/OUT	BIT	M 1.0

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 frame is located at any point in a data block that must be open before FC60 is called. 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 4 is the RTR bit of the CAN frame.

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.

The status of the CAN 300 module is in the `STAT` byte. The byte is always assigned value, even if the `Snd` bit is not set. It is advisable to call up block CANRCV before the CANSEND block so that the up-to-date status information is available.

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

6.2.3 CANRCV handling function

The CANRCV function block (FC61) transfers a CAN frame from the module to a data block 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	256
IDHI	OUT	WORD	DBW20
IDLO	OUT	WORD	DBW22
RTRLEN	OUT	WORD	DBW24
DW0	OUT	WORD	DBW26
DW1	OUT	WORD	DBW28
DW2	OUT	WORD	DBW30
DW3	OUT	WORD	DBW32
STAT	OUT	BYTE	MB 3
Rcvd	IN/OUT	BIT	M 1.0

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 target 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 4 is the RTR bit of the CAN frame.

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

The status of the CAN 300 module is in the `STAT` byte. The byte is always assigned a value even if no frame has been received. It is advisable to call up block CANRCV before the CANSEND block so that the up-to-date status information is available.

6.2.4 CANSYNSEND handling function

The function block CANSYNCSEND (FC63) transfers a CAN frame to the module. The CAN 300 module uses the ID to search for the timer suitable for the frame and only transmits the frame on the next timer event.

Parameter	Direction	Type	Example
Base	IN	INT	256
IDHI	IN	WORD	DBW0
IDLO	IN	WORD	DBW2
RTRLEN	IN	WORD	DBW4
DW0	IN	WORD	DBW6
DW1	IN	WORD	DBW8
DW2	IN	WORD	DBW10
DW3	IN	WORD	DBW12
STAT	OUT	BYTE	MB 2
Snd	IN/OUT	BIT	M 1.0

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

The bit `Snd` is always reset after the block has been executed. The frame to be transmitted is always transferred to the module.

The status of the CAN 300 module is in the `STAT` byte. The byte is always assigned a value even if no frame has been received.

6.2.5 Content of the status byte STAT

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

Bit 0: Module running, read-in of the parameters completed.

Bit 1: Script processing "POWER ON" running.

Bit 2: Script processing "STOP->RUN" running.

Bit 3: Script processing "RUN->STOP" running.

Bit 4: CAN send FIFO more than half full, overflow imminent, S7 should not transfer any more frames to the module.

Bit 5: CAN receive FIFO more than half full, overflow imminent, the S7 should read out the FIFO faster.

Bit 6: CAN send FIFO full. If further frames are transferred to the module, older, not transmitted frames are deleted.

Bit 7: Module in RUN/Reset (hardware status)

In many applications it is necessary to transmit a series of frames to the module in a cycle. The FIFO circulating buffer is 128 frames long. If bit 4 of the status byte is set, it is possible to transmit another 64 frames at once to the module.

6.3 CANopen communication

6.3.1 General

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

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".

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

The 11 bit identifier and the 8 data bytes of a CAN layer 2 message frame have a fixed meaning.

Each devices in a CANopen network has a fixed node ID (module number, 0-127).

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 and 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 object dictionary 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 slaves (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

Networkmanagement

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 handling blocks!

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

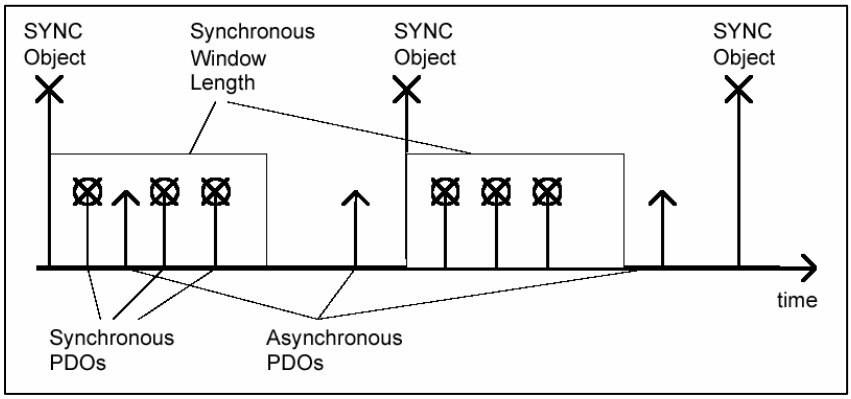
„Tx“ = Slave is sending
„Rx“ = Slave is receiving

6.3.4 Netmanagement

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

SYNC:

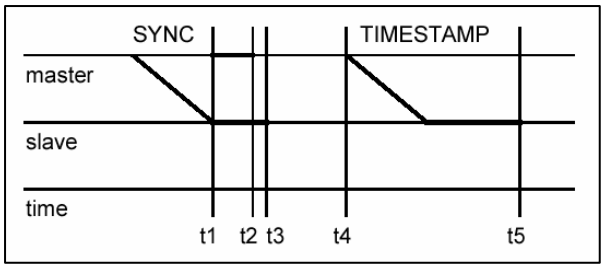
The SYNC message frame is a cyclic "broadcast" frame and sets the basic bus clock. To ensure isosynchronism, the SYNC frame has a high priority. [COB-ID: 80h]



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 CAN 300 Master monitors the CANOpen slave modules by transmitting message frames cyclically to each slave. Each CANopen slave must respond to the Nodeguarding message 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 response frames are generated by CANopen slave. The Heartbeat frame can only work usefully if Lifeguarding is active on the CANopen slave.

Some CANopen slave modules generate special messages after power-on or power-off

Emergency message:

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 must perform an emergency stop on receiving an emergency frame.

BootUp message:

CANopen slaves generate a boot-up message when they are switched on, which the master can detect for the purpose of initializing this new node.
[COB-ID: 700h + node ID + 1 byte data: 00h]

6.3.5 Handling blocks

The CANopen data 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.

This manual decribes the handling blocks of version 2.2.

The CAN 300 module works with these handling blocks as a *master* in the CANopen network.

Block	Function	User / System	Chapter
FC 40	Initialization (restart)	User	1.1.1
FC 41	Read SDO	User	1.1.1
FC 41	Transmit SDO	User	1.1.1
FC 42	SDO block download	User	1.1.1
FC 42	SDO block upload	User	1.1.1
FC 43	Spontaneous receive (NMT, PDO, etc.)	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	1.1.1
FC 49	Cycle	System	1.1.1
DB-PDO	PDO data received	User	1.1.1
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 300 module. 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.

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

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 40
BaseAddr	IN	INT	256 – 512	256
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

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 47 "Cycle". For that purpose, it is necessary to specify one DB for each PDO (1-4) during initialization (see 1.1.1).

Each DB contains space for 8 bytes PDO data for all 63 nodes. Each PDO-DB must therefore be at least 512 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**	1. Byte of Node 1 / PDO1	1. Byte of Node 1 / PDO2	1. Byte of Node 1 / PDO3	1. Byte of Node 1 / PDO4
	DBB9	2. Byte of Node 1 / PDO1	2. Byte of Node 1 / PDO2	2. Byte of Node 1 / PDO3	2. Byte of Node 1 / PDO4
	DBB10	3. Byte of Node 1 / PDO1	3. Byte of Node 1 / PDO2	3. Byte of Node 1 / PDO3	3. Byte of Node 1 / PDO4
	DBB11	4. Byte of Node 1 / PDO1	4. Byte of Node 1 / PDO2	4. Byte of Node 1 / PDO3	4. Byte of Node 1 / PDO4
	DBB12	5. Byte of Node 1 / PDO1	5. Byte of Node 1 / PDO2	5. Byte of Node 1 / PDO3	5. Byte of Node 1 / PDO4
	DBB13	6. Byte of Node 1 / PDO1	6. Byte of Node 1 / PDO2	6. Byte of Node 1 / PDO3	6. Byte of Node 1 / PDO4
	DBB14	7. Byte of Node 1 / PDO1	7. Byte of Node 1 / PDO2	7. Byte of Node 1 / PDO3	7. Byte of Node 1 / PDO4
	DBB15	8. Byte of Node 1 / PDO1	8. Byte of Node 1 / PDO2	8. Byte of Node 1 / PDO3	8. Byte of Node 1 / PDO4
Node 2	**DBB16**	1. Byte of Node 2 / PDO1	1. Byte of Node 2 / PDO2	1. Byte of Node 2 / PDO3	1. Byte of Node 2 / PDO4
	...	...	...	...	...
	DBB23	8. Byte of Node 2 / PDO1	8. Byte of Node 2 / PDO2	8. Byte of Node 2 / PDO3	8. Byte of Node 2 / PDO4
Node 63	**DBB504**	1. Byte of Node 63 / PDO1	1. Byte of Node 63 / PDO2	1. Byte of Node 63 / PDO3	1. Byte of Node 63 / PDO4
	...	...	...	...	...
	DBB511	8. Byte of Node 63 / PDO1	8. Byte of Node 63 / PDO2	8. Byte of Node 63 / PDO3	8. Byte of Node 63 / PDO4

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

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

The PDO data can also be fetched by FC 43 Spontaneous receive (see 6.3.14).

6.3.9 FC 49 cycle

FC 49 should be executed in the cycle of the program. It transmits and receives the frames and assigns the data to the jobs. It 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 140
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 — Timer for internal use. If more than one CAN300 module is used in a system, one timer per module is required.

Buffer info — Display of the assigned receive buffers, or the current jobs (transmit buffers). This information can be evaluated before new jobs are started.

Lower byte for transmit buffer:

Bit 1 = SDO transmit buffer assigned. Do not start a new job with FC41 or FC 42.

Upper byte for receive buffer:

Bit 0 = PDO frames received (COB-IDs 200h-57Fh) => Call up FC 43 function 0.

Bit 2 = Timestamp frames received (COB-IDs 100h) => Call up FC 43 function 2.

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

Bit 4 = emergency (COB-IDs 81h-FFh) or SYNC (COB-ID 80h) frame received => Call up FC 43 function 4.

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

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

The FC 49 should be called more than one time in longer SPS cycles. Each call will send or receive one telegram.

If more than one CAN300 module is used in a system, the FC 49 can be called up with another CanDB for each module.

The bits in Bufferinfo can be used to optimize the calls of the handling blocks.

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 140
Node	IN	INT	1 – 63	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	Node-ID 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 there is no answer from CANopen slave
ReturnType	Write SDO: Size of the object data received (43h = 32 bits, 4Bh = 16 bits, 4Fh = 8 bits) Read SDO: 60h = OK
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
Data	Transfer the data (reading and writing)
AbortCode	CANopen error number from 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 and the job is only executed when the FC 49 is called up.

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

Example:

```
      UN    M      9.1                  // new job ?
      UN    M    111.0                  // job runnng ?
      SPB   next

      CALL  FC    41
       CanDB    := DB 40
       Node     := MW 28
       Index    := MW 30
       Subindex := MB 32
       Typ      := 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

      UN    M    111.7                  // Ready ?
      SPB   next
      U     M    111.6                  // Error ?
      SPB   err

      L     MD    34                    // Get SDO-Value !

      // use here SDO Value depending on ReturnTyp (size)

next: ...
```

6.3.11 FC 42 Download and upload SDO block

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").

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 140
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
Activate	INOUT	BOOL	M 0.0 – M 1023.7	M 1.2
AbortCode	INOUT	DWORD	MD 0 – MD 1020	MD 14

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	CANopen error number from CANopen slave

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

Byte	Type	Example	Purpose
0	BYTE	1	Direction: 0 = Block upload, 1 = Block download, 2= Segmented upload, 3=Segmented 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		Net data
	ENDARRAY		

Only one job from the DB is executed for each FC 42. It is possible to concatenate several jobs.

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 and the job is only executed when the FC 49 is called up.

If no response comes from the CANopen slave, the current job can 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 140
Node	IN	INT	0 – 63	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
SYNC	IN	INT	1, 3	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
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)
Sync	PDO-Telegramm will be sent asynchronous (Sync = 1) or using one of the configured timers of the CAN 300 module synchronous (Sync = 3).
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 44 transferes the CAN-Telegram directly to the CAN 300 module, calling the FC 49 is not necessary. The FC 44 can be called several times 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 140
Node	IN	INT	0 – 63	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) 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 6.3.14).

The FC 45 transferes the CAN-Telegram directly to the CAN 300 module, calling the FC 49 is not necessary. The FC 45 can be called several times in one cycle.

6.3.14 FC 43 Spontaneous receive

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

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB1 – DB 2047	DB 140
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 = TPDO frames from Slave with no defined PDO-DB (COB-ID 180h-1FFh, 280h-2FFh, 380h-4FFh, 480h-4FFh)
1 = NMT service frames (COB-ID 780h-7FFh)
2 = Timestamp frames (COB-ID 100h)
3 = NMT frames (COB-ID 00h-7Fh)
4 = Emergency (COB-ID 81h-FFh) or SYNC (80h) frames
5 = NMT error frames (COB-ID 700h-77Fh)

Status — Status byte of the job processing:
Bit 6 = Error (error number in)
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. You can find out whether a receive buffer is allocated with the buffer info parameter when calling up the FC 49 cycle.

For each type of frame (see parameter Func) there is only one receive buffer in the CAN-DB. For that reason, this FC should be called up straight after the FC 49. Of course, that only applies if the frames are important for the S7 application. Other wise 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 140
Node	IN	INT	0 – 63	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 of stop of the CAN 300 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 Timernumber (for 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 (Timernumber in Node)
11 = Timer Stop (Timernumber in Node)

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

Error — Error number on error in execution

The FC 48 transferes the NMT-Telegram directly to the CAN 300 module, calling the FC 49 is not necessary.

6.3.16 FC 46 Service

For deletion of suspended jobs and fetching of abort codes.

Parameter	Direction	Type	Range	Example
CanDB	IN	BLOCK-DB	DB 1 – DB 2047	DB 140
Func	IN	CHAR	'P', 'S', 'p', 's', 'C'	'S'
Status	OUT	BYTE	MB 0 – MB 1023	MB 10
Error	OUT	WORD	MW 0 – MW 1023	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 for execution with error

6.3.17 FC 47 Nodeguarding/Heartbeat

FC 47 can be used to transmit nodeguarding frames or receive 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 available (expected) slaves in the network (see below)

GuardTimer — Timer for delay between two requests

TMTimer — Timer for response timeout of slave

OnlyReceive — False = nodeguarding frames are being transmitted and a response is expected from the slave
True = only heartbeat frames are received from the slaves and entered in the BeatDB

Error — Error number for incorrect execution

A DB that contains a list of the slaves to be monitored is required for monitoring slaves with nodeguarding (BeatDB):

Byte	Type	Example	Purpose
0	BYTE	14h	SendDelay: 14h=20d x 10 = 200ms time between two nodeguarding frames
1	BYTE	32h	TimeOut: 32h=50d x 10 = 500ms timeout for response frames from slave
2	BYTE	-	Counter (used internally)
3	BYTE	-	reserved
4	BYTE	1	Node number of 1st slave
5	BYTE	0	Received status of 1st slave
6	BYTE	2	Node number of 2nd slave
7	BYTE	0	Received status of 2nd slave
8	BYTE	3	Node number of 3rd slave
9	BYTE	0	Received status of 3rd slave
…	…	…	…
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 telegrams are to be transmitted by the master (OnlyReceive = True) , the timeouts have no effect.

Whenever a heartbeat frame is received, the status of this frame is entered in the BeatDB next to the relevant node number. The user should then evaluate this status and overwrite it with zero. Separate timeout management is not necessary.

6.4 Explanation of the example program

The example program supplied with CANopen handling shows how handling blocks are used in a very simple way. The functionalities of the handling blocks are triggered by the bits in input byte 8.

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

The CAN 300 module must be configured on address 256. Handling is initialized in OB100, where DB40 is used as the CAN-DB and DB51 is used as the PDO1 data DB.

The example project "C3 CANopen Master.PAR" (installed with CANParam) should be adapted to the test device (for example, to the baudrate of the CAN bus) and then imported into the CAN300 module.

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

When FC 10 starts, cycle block FC 49 (→ Chapter 1.1.1) is called in order to fetch frames received by the CAN bus or to process send requests. The buffer info is stored in MW 10 and displayed on QW 0.

In network 2, nodeguarding can be activated via FC 47 (→ 6.3.17) with input bit 8.7. DB47 contains a list with node numbers 1+2+3, which are scanned cyclically.

In network 3, first of all the first two bytes of PD01 from POD1-DB by node 2 and output on QW 2.

Cyclic transmission of PDO1 to node 2 can now be triggered with input bit 8.0 (FC 44, → 1.1.1). The value is always incremented by 1 and transmitted when the last value is returned by the node via receive PDO1. Remove the compare query in these lines if you want to transmit every cycle. The data are in MB12 - MB19, of which only MW12 is incremented.

Network 4 comprises the fetching and receiving of a SDO (FC 41, 6.3.10) via input bits 8.1 and 8.2. The transfer parameters are in the MW. A variables table (VAT_1) for testing is provided in the project.

In network 5, FC 48 (→ 6.3.15) is called up for network management. Make sure that in the example project "DEMO.PR", the CAN frames for "NMT start all nodes" and "NMT stop all nodes" are in the scripts for starting and stopping the CPU.

6.4.2 Example FC 11 (spontaneous receive)

FC 11 is called up in OB 1 after FC 10, here, "unexpected" frames that have been received are fetched from the CAN-DB buffer. This function is controlled by the information in MW 10, which contains the buffer info from FC 49.

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

6.4.3 Example FC 12 (reading a SDO list)

Reading of a list of SDOs is organized in FC 12. The list of SDOs to be read is to be found in DB 12 and might have to be adapted to the devices used on the CANopen slave device(s) used.

In FC 12 the FC 41 (→ 6.3.10) is started by input bit 8.4 for every bit in DB 12 one after the other and the result (value or error number) also placed in DB 12. DBW 0 must be reset before the list in DB 12 can be processed again.

6.4.4 Example FC 13 (writing an SDO list)

The writing of a list of SDOs is organized in FC 13. The list of SDOs to be written are stored together with their values in DB 13 and may have to be adapted to the CANopen slave device(s) used.

In FC 13 the FC41 (→ 6.3.10) is started by input bit 8.5, for every SDO in DB 13 one after the other and the result (error number or OK status) also placed in DB 13. DBW 0 must be reset before the list in DB 13 can be executed again.

6.4.5 Example FC 3 (SDO Block/Segmented Download)

FC 3 contains example calls for FC 42 (→ 6.3.11) SDO Block/Segmented Download. DB 3 contains a sample list and may have to be adapted to the CANopen slave device if SDO Download is actually needed.

6.5 Error numbers

Possible error numbers of the return parameter Error.

Number	Meaning
1	Node below 1
2	Node above 63
3	PDO below 1, or timer number below 0
4	PDO above 4, or timer number above 11
6	No data available
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 on SDO-job, no answer from CANopen slave
101	Buffer allocated, busy with a job.
102	Abort code received
105	Function number unknown
140	CAN 300 module not ready
141	CAN 300 module not ready for communication (buffer overfow)
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 "was 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 service parameter does not match
0607 0012h	Data type does not match, length of service parameter too large
0607 0013h	Data type does not match, length of 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 300	700-600-CAN01
	CAN 300 (DNV)	700-600-CAN81
Dimensions	116 x 40 x 125 mm (LxWxH)	
Weight	Approx. 280g	

CAN interface

Type:	ISO/DIN 11898, 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* DEVICENET *available soon*
Connection:	Connector, SUB D 9-way

Configuration interface

Type:	RS232, serial asynchronous
Transmission rate:	9.6 kbps
Format:	8/N/1
Connection:	Connector, SUB D 9-way

Power supply

Voltage:	+5V DC via backplane bus
Current consumption:	160mA (typ.) / 190mA (max.)

Special features

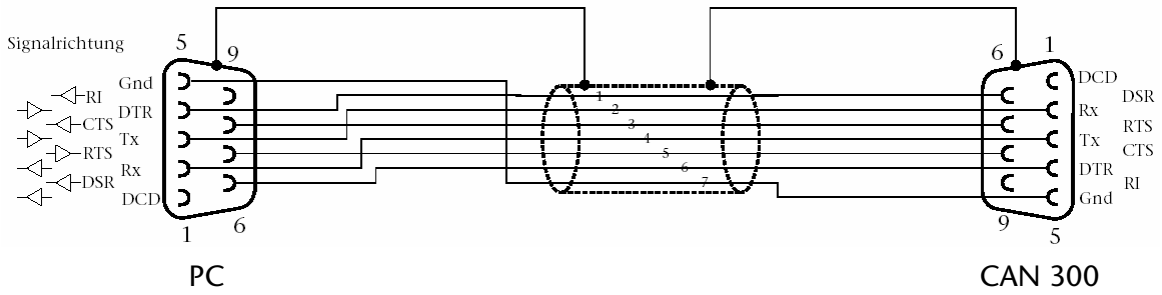
Quality assurance:	According to ISO 9001:2000
Maintenance:	Maintenance-free (no battery, rechargeable or non-rechargeable)

7.2 Pin assignment

Pin	SUB-D connector RS232	SUB-D connector CAN
1	-	-
2	Rx	CAN Low
3	Tx	CAN GND
4	-	-
5	GND	-
6	-	-
7	-	CAN High
8	-	-
9	-	-

7.3 Connecting cable

RS232 parameterization (700-610-0VK11) / Nullmodem:



7.4 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