CAN 300

CAN communication module for S7-300
with LENZE-Systembus handlingblocks

Manual

Edition 2 / 30.09.2005 for HW2 & FW2.3

Order number: 900-600-1LZ11

Systeme Helmholz GmbH ● Gewerbegebiet Ost 36 ● D-91085 Weisendorf
Fon: +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 are checked regularly and any necessary corrections will be included in subsequent editions. We always welcome suggestions for improvement.

Contents

1	**Overview**	**6**
1.1	General	6
1.2	Handling for LENZE Systembus	6
1.3	LENZE system bus	7
1.3.1	Parameter channels	7
1.3.2	Process data channels	8
1.3.3	Structure of the process data	8
1.3.4	Cyclic process data	9
1.3.5	Event-controlled process data	10
2	**Configuring in the PLC**	**11**
3	**Configuration of the CAN 300 module**	**13**
3.1	Create new project	14
3.2	Setting the CAN bus baudrate	15
3.3	Setting the transmission mode (protocol)	15
3.4	Acceptance masks	16
3.5	Network management	17
3.6	Timer	18
3.7	Upload / download	18
3.8	Diagnostics/debugging	19
4	**Programming in the PLC**	**21**
4.1	Overview	21
4.1.1	FC50 LCANINIT	22
4.1.2	Process data DBs	23
4.1.3	FC59 LCANCYCL	24
4.1.4	FC51 LCANPARA	25
4.1.5	FC52 LCANPDO	27
4.1.6	FC54 LCANLAY2	28
4.1.7	FC58 LCANNMT	29
4.1.8	Error numbers	29
4.2	Example project	30
4.2.1	FC1 Demo	30
4.2.2	FC2 Loading a parameter list	31
4.2.3	FC3 Transmitting a parameter list	31

5 Appendix 32
5.1 CAN identifiers 32
5.2 Technical data 33
5.3 Pin assignment 34
5.4 Connecting cable 34
5.5 Further documentation 34

1 Overview

1.1 General

These instructions are a supplement to the manual of the CAN 300 module of System Helmholz GmbH.

The software (handling blocks) described in this manual is not part of the scope of supply of the CAN 300 module and must be ordered separately.

1.2 Handling for LENZE Systembus

The CAN300 module is for use in Siemens programmable controllers from the S7-300 range. The programmable controller can be connected to the CAN bus using the CAN 300 module.

The CAN300 module receives and transmits the CAN frame without further interpretation. By using handling blocks the CAN frames can be interpreted or initiated.

Handling blocks for simple "layer 2" transmission are included in the scope of supply of the CAN300 module as well as a CANopen Master for implementation. These handling blocks are available for general use.

In addition to these, special handling blocks for implementing the LENZE system bus are available.

This guide describes how to use handling modules to connect LENZE controllers.

1.3 LENZE system bus

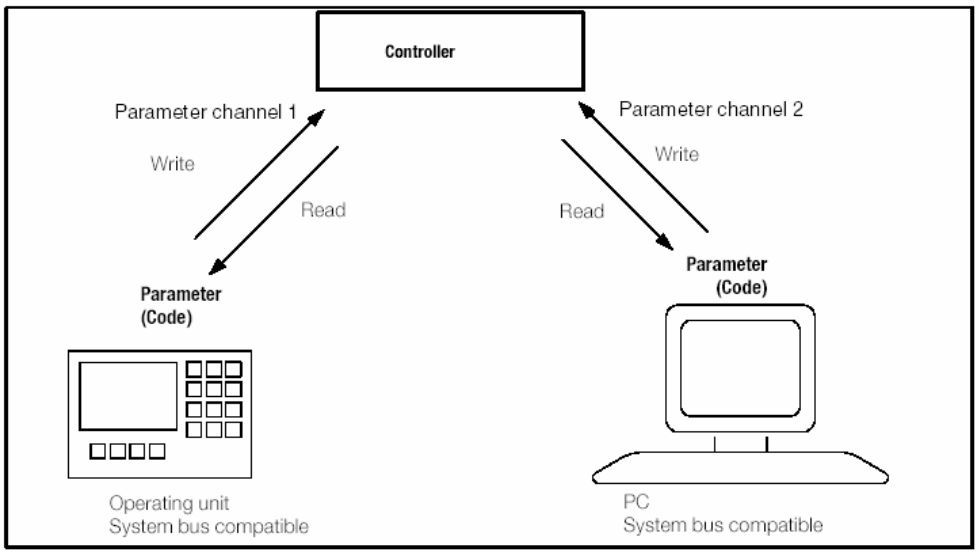
(extract from LENZE system bus documentation)

1.3.1 Parameter channels

Parameters are values that are stored in codes in Lenze drive controllers. Parameters are modified, for example, for once-only system settings or when changing materials in a machine.

The two parameter channels in the function module system bus (CAN) allow you to connect two different devices for parameterization, for example, connection of a PC and an operator panel at the same time.

Parameters are transferred with low priority.



The parameters are identified by their code number (for example "C0350" -> parameter 350). In addition to a code number there might also be a subcode if different values can be set in a parameter.

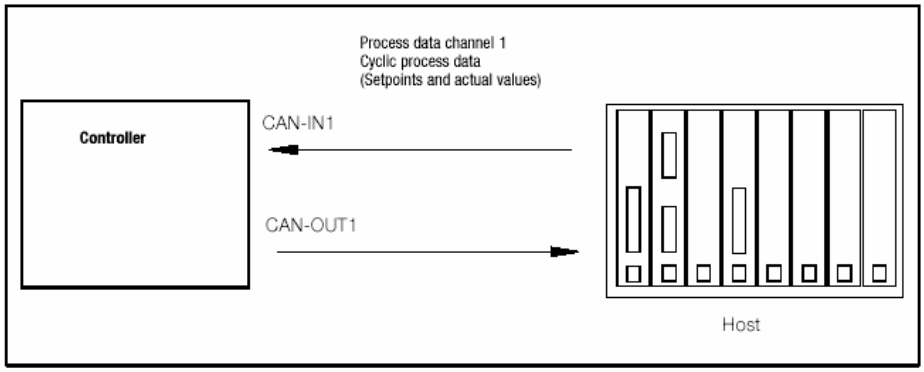
Extract from a code table:

C0010	Minimum output frequency	0.00	0.00 → **14.5 Hz**	{0.02 Hz}	480.00	• C0010 is not effective with bipolar setpoint selection (-10 V ... + 10 V) • C0010 has no effect on AIN2 → **Speed setting range 1 : 6 for Lenze geared motors:** Setting absolutely required for operation with Lenze geared motors.	🕮 7-14
C0011	Maximum output frequency	50.00	7.50 → **87 Hz**	{0.02 Hz}	480.00		
C0012	Acceleration time main setpoint	5.00	0.00	{0.02 s}	1300.00	Reference: frequency change 0 Hz ... C0011 • Additional setpoint ⇨ C0220 • Acceleration times to be activated via digital signals ⇨ C0101	🕮 7-16
C0013	Deceleration time main setpoint	5.00	0.00	{0.02 s}	1300.00	Reference: frequency change C0011 ... 0 Hz • Additional setpoint ⇨ C0221 • Deceleration times to be activated via digital signals ⇨ C0103	

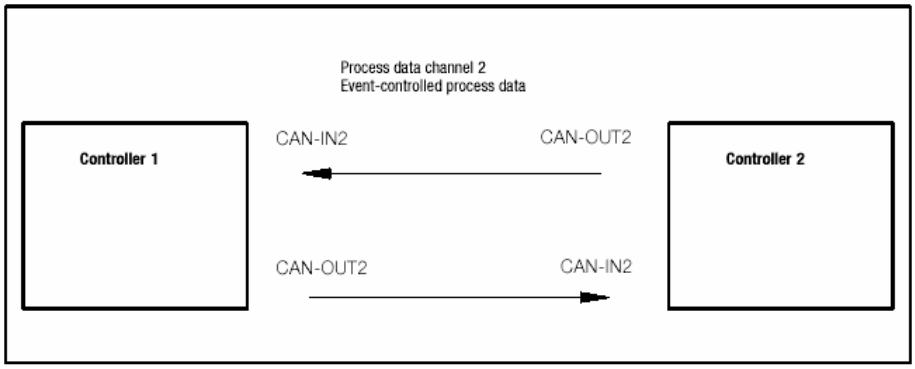
1.3.2 Process data channels

Process data (for example, setpoints and actual values) are transmitted and processed with high priority and at high speed. The system bus (CAN) functional module provides the following.

A cyclic, synchronized process data channel (CAN1) for communication with an I&C system (process data objects CAN-IN1 and CAN-OUT1):



An event-controlled process data channel (CAN2) for communication between drive controllers (process data objects CAN-IN2 and CAN-OUT2)



Distributed input and output terminals and higher-level I&C systems can also use CAN2.

1.3.3 Structure of the process data

Two process data objects for input information (CAN-IN1, CAN-IN2) and two process data objects for output information (CAN-OUT1, CAN-OUT2) are available for fast data exchange between the drive controllers themselves or with a higher-level I&C system.

This allows transmission of simple binary signals such as the states of digital input terminals or data in 16 bit format, such as analog signals, for example.

Cyclic, synchronized process data (process data channel CAN1):

- One process data object for input signals (CAN-IN1) and one process data object for output signals (CAN-OUT1) each with 8 bytes of useful data are available for fast cyclic data transmission.
- These data are intended for communication with the higher-level I&C system, such as the PLC, for example.
- CAN1 can also be used under event control (setting with C0360).

Event-controlled process data (process data channel CAN2):

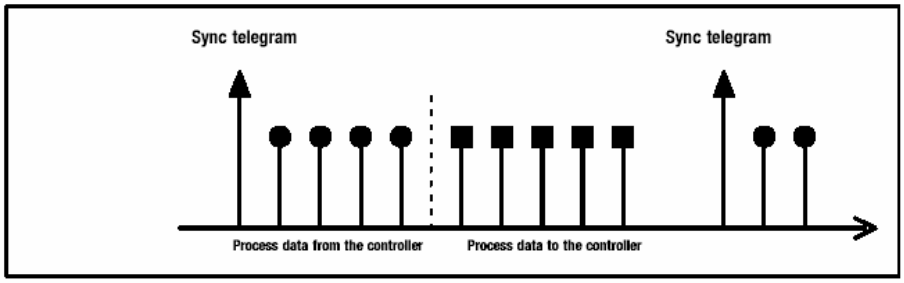
- A process data object for input signals (CAN-IN2) and a process data object for output signals (CAN-OUT2) each with 8 bytes of useful data are available for event-controlled data transmission.
- The output data are always transferred when a value changes in the useful data.
- This process data channel is particularly suitable for data exchange between drive controllers and distributed terminal expansion. However, it can also be used by an I&C system.

1.3.4 Cyclic process data

Cyclic process data can only be read by the drive controller and drive controllers only accept process data with a Sync frame.

The Sync frame is the trigger point for data transfer in the drive controller and initiates the transmission procedure of the drive controller. The Sync frame must be generated by the I&C system before cyclic process data processing can start.

Synchronization of cyclic process data:



After a Sync frame the cyclic process data are transmitted by the drive controllers. That is followed by data transfer to the drive controllers which are transferred from the individual drive controllers with the next Sync frame.

All remaining frames, such as parameters or event-controlled process data, for example, are transferred asynchronously after successful transmission from the drive controllers.

Structure of the process data frames in the cyclic process data channel:

Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

	User data assignment			
	Byte	**Word assignment (16 bit)**	**Bit assignment**	**Internal signal assignment via**
Cyclic process data telegram to drive CAN-IN1	1	CAN-IN1.W1 (LOW byte)	CAN-IN1.B0 ... CAN-IN1.B15	C0410 (digital) C0412 (analog)
	2	CAN-IN1.W1 (HIGH byte)		
	3	CAN-IN1.W2 (LOW byte)	CAN-IN1.B16 ... CAN-IN1.B31	C0410 (digital) C0412 (analog)
	4	CAN-IN1.W2 (HIGH byte)		
	5	CAN-IN1.W3 (LOW byte)		C0412
	6	CAN-IN1.W3 (HIGH byte)		
	7	CAN-IN1.W4 (LOW byte)		C0412
	8	CAN-IN1.W4 (HIGH byte)		
				Configuration via
Cyclic process data telegram from drive CAN–OUT1	1	CAN-OUT1.W1 (LOW byte)	CAN-OUT1.B0 ... CAN-OUT1.B15	C0417 (digital) C0421/3 (analog)
	2	CAN-OUT1.W1 (HIGH byte)		
	3	CAN-OUT1.W2 (LOW byte)		C0421/4
	4	CAN-OUT1.W2 (HIGH byte)		
	5	CAN-OUT1.W3 (LOW byte)		C0421/5
	6	CAN-OUT1.W3 (HIGH byte)		
	7	CAN-OUT1.W4 (LOW byte)		C0421/6
	8	CAN-OUT1.W4 (HIGH byte)		

1.3.5 Event-controlled process data

8 bytes are available for each data object.

The output data are transferred whenever a value changes within the 8 bytes of useful data or at the cycle time set under C0356/2 for CAN-OUT2 or under C0356/3 for CAN-OUT1.

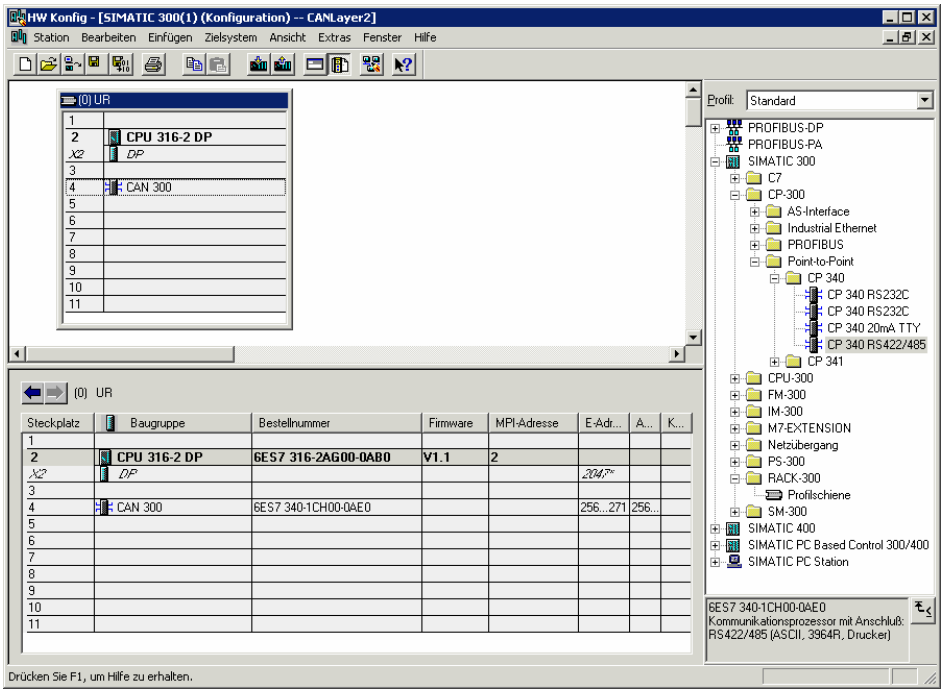
Structure of the process data frames in the event-controlled process data channel:

Identifier	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

	User data assignment			
	Byte	**Word assignment (16 bit)**	**Bit assignment**	**Internal signal assignment via**
Process data telegram to drive CAN-IN2 (accepts system bus device immediately)	1	CAN-IN2.W1 (LOW byte)	CAN-IN2.B0 ... CAN-IN2.B15	C0410 (digital) C0412 (analog)
	2	CAN-IN2.W1 (HIGH byte)		
	3	CAN-IN2.W2 (LOW byte)	CAN-IN2.B16 ... CAN-IN2.B31	C0410 (digital) C0412 (analog)
	4	CAN-IN2.W2 (HIGH byte)		
	5	CAN-IN2.W3 (LOW byte)		C0412
	6	CAN-IN2.W3 (HIGH byte)		
	7	CAN-IN2.W4 (LOW byte)		C0412
	8	CAN-IN2.W4 (HIGH byte)		
				Configuration via
Event-controlled process data telegram from drive CAN-OUT2	1	CAN-OUT2.W1 (LOW byte)	CAN-OUT2.B0 ... CAN-OUT2.B15	C0418 (digital) C0421/7 (analog)
	2	CAN-OUT2.W1 (HIGH byte)		
	3	CAN-OUT2.W2 (LOW byte)		C0421/8
	4	CAN-OUT2.W2 (HIGH byte)		
	5	CAN-OUT2.W3 (LOW byte)		C0421/9
	6	CAN-OUT2.W3 (HIGH byte)		
	7	CAN-OUT2.W4 (LOW byte)		C0421/10
	8	CAN-OUT2.W4 (HIGH byte)		

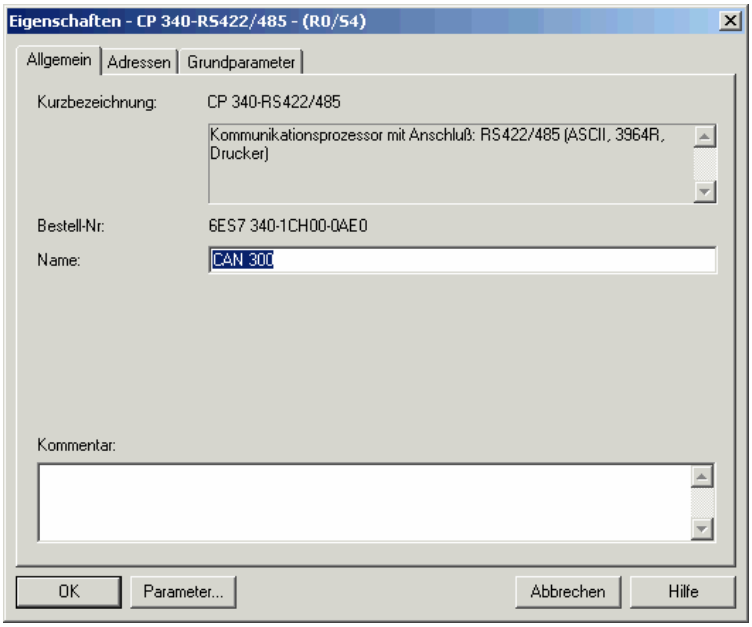
2 Configuring in the PLC

The CAN 300 module is configured as communications module "CP340" in the programming software of the PLC.



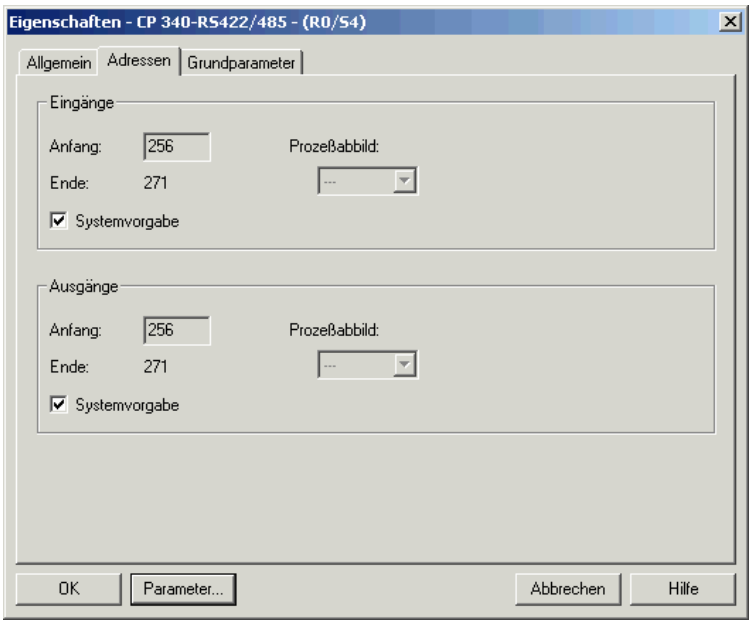
The module can be used wherever it would also be possible to use a CP module, including the the expansion rack after an interface.

The CAN300 module cannot be used in ET200M systems.



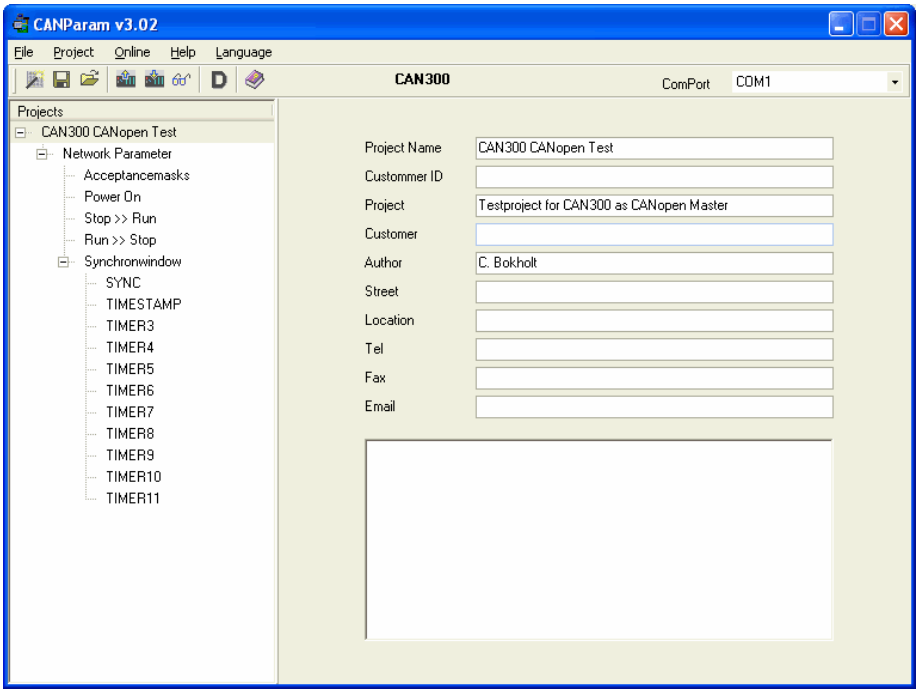
Only the I/O address area is relevant to parameterization of the module. The other settings have no effect on the module.

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



3 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.

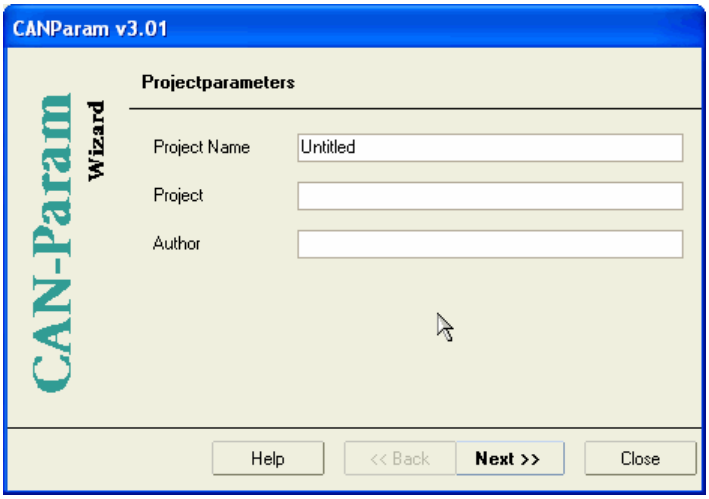


The configuration of a module can be stored in a project file on the PC.

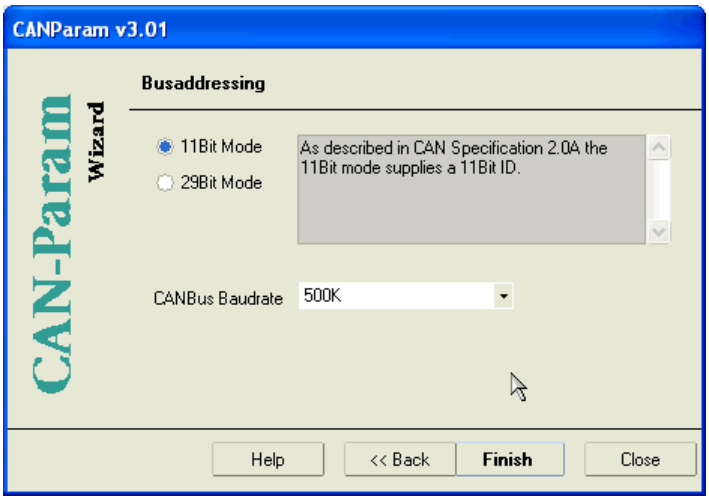
You can use a normal commercial type null modem cable to link the PC to the CAN 300 module (see also 5.4). After installation and starting of the CANParam software, you should set the interface top right on the menu bar.

3.1 Create new project

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

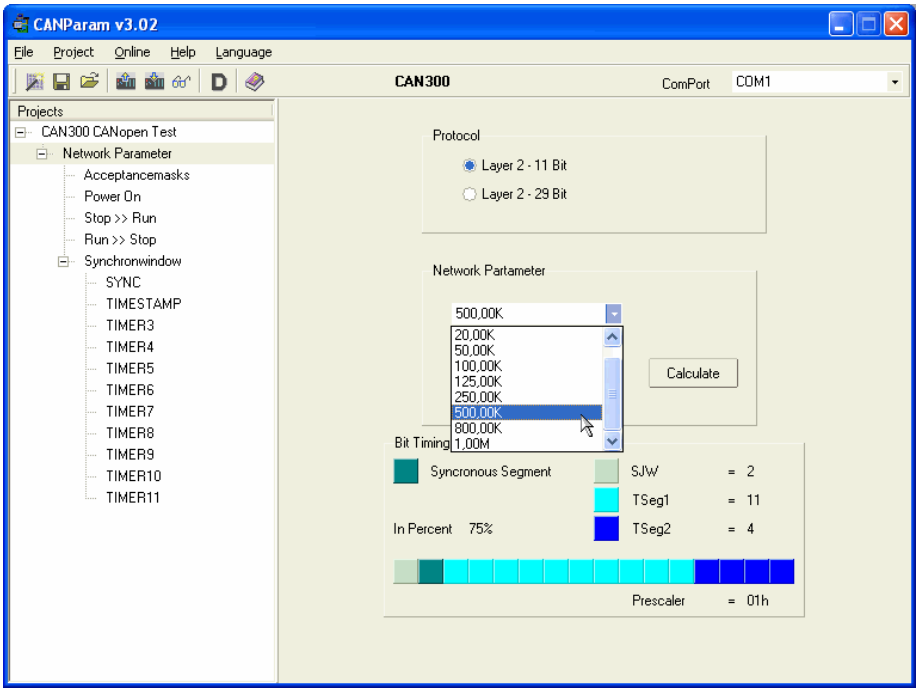


The project wizard guides you through the most important settings to obtain a new and complete project.



3.2 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.

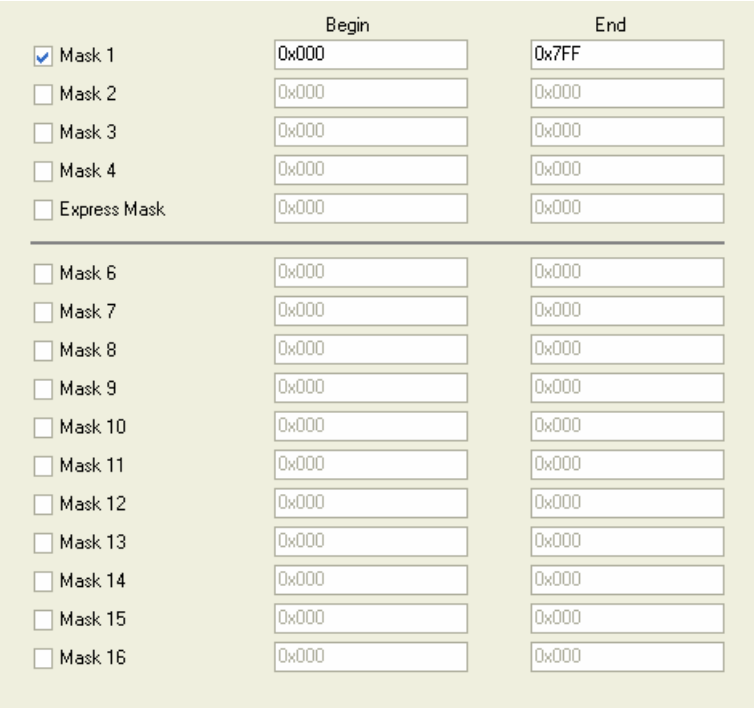
3.3 Setting the transmission mode (protocol)

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

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

3.4 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.

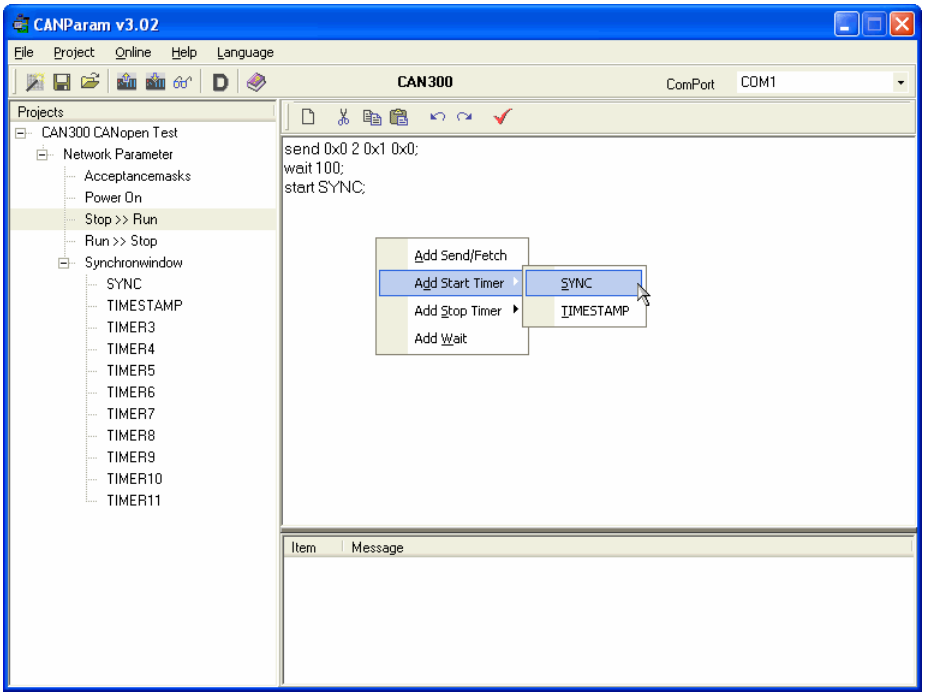
3.5 Network management

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:

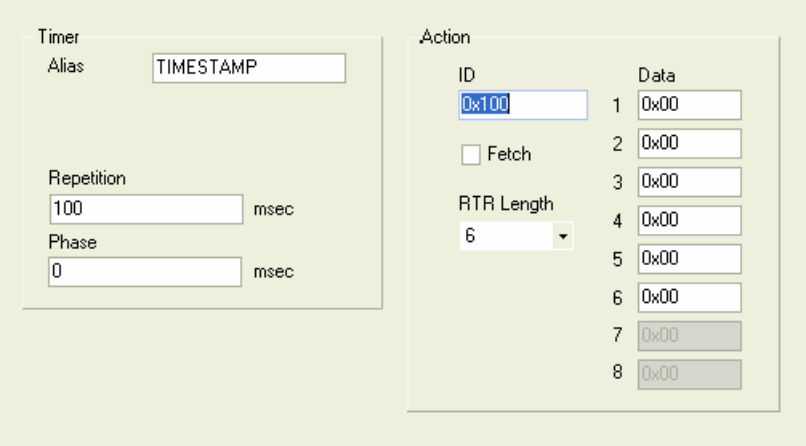
Send	Transmit frame (Structure: ID, length, data byte 1, data byte 2, etc.)
Fetch	Transmit frame with RTR bit 1
Start	Start Timer X
Stop	Stop Timer X
Wait	Wait X ms
//	Comment line

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



3.6 Timer

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



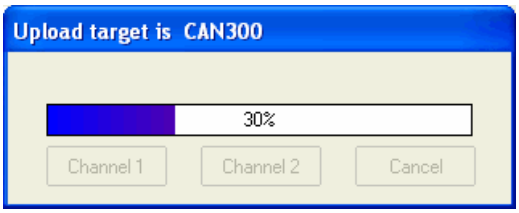
An alias can be assigned to each timer. This name can the be used in the scripts of the PLC events.

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 15 msec. to 1 sec. can be set in steps of 5 msec. For the *phase* 0msec to 5 msec before the period duration.

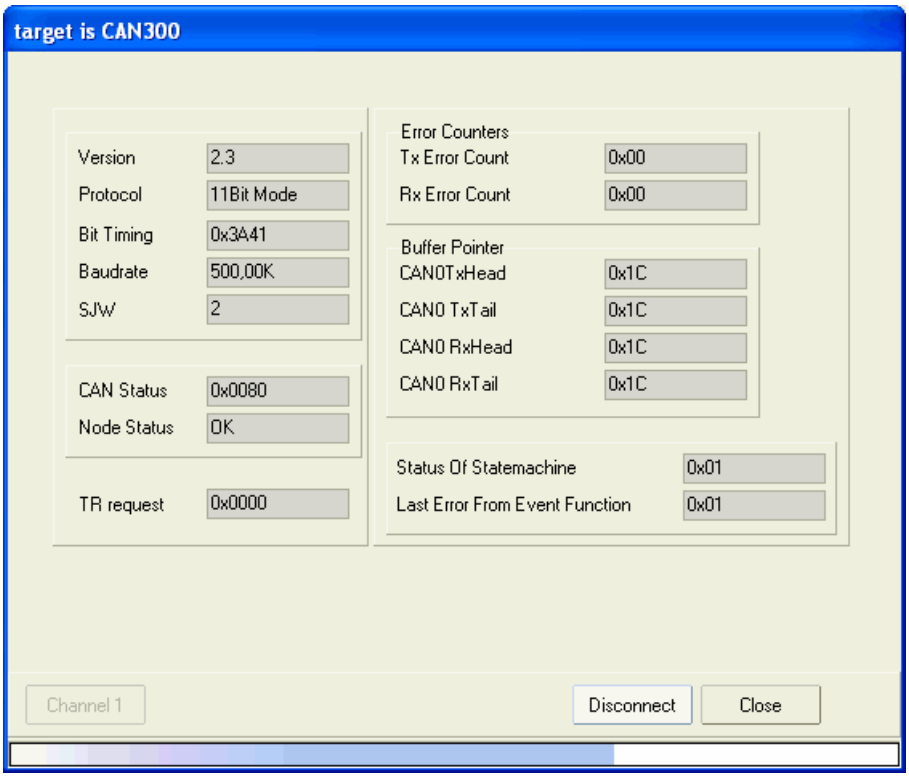
3.7 Upload / download

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



3.8 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.

!

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

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)

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

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
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

4 Programming in the PLC

4.1 Overview

The CAN 300 module is programmed in the PLC using the following handling blocks:

FC50	**LCANINIT**	Initialization of CAN-DB
FC51	**LCANPARA**	Read and transmit parameter data
FC52	**LCANPDO**	Transmit process data
FC54	**LCANLAY2**	Transmit Layer 2 frames
FC58	**LCANNMT**	Network management functions
FC59	**LCANCYCL**	Cyclic communication

LENZE system bus handling blocks cannot be used at the same time as other CAN300 handling blocks (mixed operation)!

Handling requires a permanent management data block (CanDB). This DB is initialized when FC50 "LCANINIT" is called up and is used by all other handling blocks. In the example project below this is DB50.

4.1.1 FC50 LCANINIT

The FC50 does not trigger a cold restart of the module. It can therefore not be used to reset the module!

Function block "LCANINIT" (FC50) initializes Lenze system bus communication. The FC should be called up during initialization of OB100 and must have been called up before the other handling blocks are used.

Parameter	Direction	Type	Example
CanDB	IN	BLOCK-DB	DB50
BaseAddr	IN	INT	256
CAN_OUT1_SYNC	IN	INT	51
CAN_OUT1_TIME	IN	INT	52
CAN_OUT2	IN	INT	53

CanDB Internal DB with current CAN data

BaseAddr Basic address of module

CAN_OUT1_SYNC Number of DB for receiving the OUT1_SYNC process data of all nodes

CAN_OUT1_TIME Number of DB for receiving the OUT1_TIME process data of all nodes

CAN_OUT2 Number of the DB for receiving the OUT2 process data of all nodes

4.1.2 Process data DBs

The data of received process data frames are automatically copied by FC59 "LCANCYLC" into DBs. For this, a DB must be specified during initialization for every process data type (OUT1-Sync, OUT1-Time, Out2).

Each DB has space for 8 bytes of process data values for 63 nodes. This means that each PDO-DB must be at least 512 bytes long.

		Datablock Out1-Sync	Datablock Out1-Time	Datablock Out2
	DBB0	not used	not used	not used
	...	...	...	...
	DBB7	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
	DBB9	2. Byte of Node 1 / PDO1	2. Byte of Node 1 / PDO2	2. Byte of Node 1 / PDO3
	DBB10	3. Byte of Node 1 / PDO1	3. Byte of Node 1 / PDO2	3. Byte of Node 1 / PDO3
	DBB11	4. Byte of Node 1 / PDO1	4. Byte of Node 1 / PDO2	4. Byte of Node 1 / PDO3
	DBB12	5. Byte of Node 1 / PDO1	5. Byte of Node 1 / PDO2	5. Byte of Node 1 / PDO3
	DBB13	6. Byte of Node 1 / PDO1	6. Byte of Node 1 / PDO2	6. Byte of Node 1 / PDO3
	DBB14	7. Byte of Node 1 / PDO1	7. Byte of Node 1 / PDO2	7. Byte of Node 1 / PDO3
	DBB15	8. Byte of Node 1 / PDO1	8. Byte of Node 1 / PDO2	8. Byte of Node 1 / PDO3
Node 2	DBB16	1. Byte of Node 2 / PDO1	1. Byte of Node 2 / PDO2	1. Byte of Node 2 / PDO3
	...	...	...	...
	DBB23	8. Byte of Node 2 / PDO1	8. Byte of Node 2 / PDO2	8. Byte of Node 2 / PDO3
Node 63	DBB504	1. Byte of Node 63 / PDO1	1. Byte of Node 63 / PDO2	1. Byte of Node 63 / PDO3
	...	...	...	...
	DBB511	8. Byte of Node 63 / PDO1	8. Byte of Node 63 / PDO2	8. Byte of Node 63 / PDO3

4.1.3 FC59 LCANCYCL

Function block "LCANCYLC" (FC59) must be called up in the cycle of the PLC in order to process received CAN frames or execute requests to be transmitted.

Parameter	Direction	Type	Example
CanDB	IN	BLOCK-DB	DB50
T	IN	TIMER	T10
Identifier	OUT	WORD	MW12
HData	OUT	DWORD	MD20
LDATA	OUT	DWORD	MD24
Bufferinfo	OUT	BYTE	MB10

CanDB — Internal DB with current CAN data

T — Timer for timeout monitoring of requests

Identifier — COB-ID of a received and unprocessed CAN frame

HData — Data bytes 1-4 of a received and unprocessed CAN frame

LData — Data bytes 5-8 of a received and unprocessed CAN frame

Bufferinfo — Display of current requests or received CAN frame.

- Bit 0 = Parameter transmission active (transmission buffer busy)
- Bit 4 = CAN frame received
- Bit 5 = CAN frame receive buffer overflow

Bit 4 of Bufferinfo should be reset when the received CAN frame is accepted. If this bit is not reset, bit 5 is initially set the next time a frame is received.

4.1.4 FC51 LCANPARA

Parameters in the controller can be read and written with function block "LCANPARA" (FC51).

Parameter	Direction	Type	Example
CanDB	IN	BLOCK-DB	DB50
Node	IN	INT	1
Channel	IN	BYTE	B#16#1
Codenummer	IN	WORD	MW60
SubCode	IN	BYTE	MB62
Direction	IN	CHAR	‚R‘
SendData	IN	DWORD	MD70
Status	OUT	BYTE	MB63
Error	OUT	WORD	MW65
ReceiveData	OUT	DWORD	MD74
Activate	INOUT	BIT	M69.0

CanDB	Internal DB with current CAN data
Node	Node number of controller
Channel	Channel number (1, 2)
Codenumber	Parameter code (e.g. "C0412" -> 412)
SubCode	SubCode of parameter (e.g. 3)
Direction	'R' = Read (data in ReceiveData) 'W' = Write (data in SendData)
SendData	Data for Write Parameter
Status	Status of request Bit 0 = Request running, Bit 6 = Completed with errors (error no. in Error) Bit 7 = Request completed
Error	Error number
ReceiveData	Data received from Receive Parameter
Activate	Activate request

The FC must be called cyclically. SDO transmission is not triggered until the activation bit (Activate) is set. The FC resets the bit as soon as it has received the request. The current status of request processing can be monitored in the Status byte.

FC 51 enters the request in the CAN-DB. But the request is not executed (sent along the CAN bus) until FC 59 is called.

Example of FC51 call:

```
      AN    M     18.0                  // new Job ?
      AN    M     63.0                  // not running ?
      JC    end

      CALL  FC    51
       CanDB      :=DB50
       Node       :=1
       Channel    :=B#16#1
       Codenumber :=MW60
       SubCode    :=MB62
       Direction  :='R'
       SendData   :=MD70
       Status     :=MB63
       Error      :=MW65
       ReceiveData:=MD74
       Activate   :=M18.0

      AN    M     63.7                  // Request complete ?
      JC    end
      A     M     63.6                  // No errors
      JC    end
      L     MD    74
      T     MD    80                    // Accept value

end:  ...
```

4.1.5 FC52 LCANPDO

Process data can be read into the controller or requested by the controller with function block "LCANPDO" (FC52).

Parameter	Direction	Type	Example
CanDB	IN	BLOCK-DB	DB50
Node	IN	INT	1
Channel	IN	BYTE	B#16#01
HData	IN	DWORD	MD40
LData	IN	DWORD	MD44
Error	OUT	WORD	MW50

CanDB	Internal DB with up-to-date CAN data
Node	Node number of the controller
Channel	Channel number: 01 = Transmit CAN-IN1 Sync PDO 11 = Transmit CAN-IN1 Time PDO 02 = Transmit CAN-IN2 PDO 51 = Request CAN-OUT1 Sync PDO 61 = Request CAN-OUT1 Time PDO 52 = Request CAN-OUT2 PDO
HData	Higher 4 bytes of PDO frame during transmission
LData	Lower 4 byte of PDO frame during transmission
Error	Error number

The process data frame is immediately transferred by the FC to the CAN300 module and thus transmitted. It is therefore not necessary to call FC59 "LCANCYCL", several process data frames can be sent in succession.

If process data have been requested (channel = 51,61, or 52), then FC 59 "LCANCYCL" will receive the frames and, if necessary, enter the process data in the PDO-DBs.

4.1.6 FC54 LCANLAY2

With function block "LCANLAY2" (FC54) you can send any CAN frame to supply other CAN nodes with data, if necessary.

Parameter	Direction	Type	Example
CanDB	IN	BLOCK-DB	DB50
Identifier	IN	WORD	W#16#0289
Count	IN	BYTE	B#16#8
Sync	IN	BYTE	B#16#1
HData	IN	DWORD	MD110
LData	IN	DWORD	MD114
Error	OUT	WORD	MW118

CanDB	Internal DB with up-to-date CAN data
Identifier	COB ID of CAN frame
Count	Number of bytes to be sent
Sync	1 = Synchronous transmission (transmit immediately), 3 = Synchronous transmission (the data of the timer belonging to the "identifier" are updated)
Error	Error number

The CAN frame is transferred immediately by FC54 to the CAN300 module and thus transmitted. It is therefore not necessary to call FC59 "LCANCYCL", several CAN frames can be transmitted in succession.

Response frames from the called CAN nodes can be received via FC59 "LCANCYLC".

4.1.7 FC58 LCANNMT

Network management functions can be executed with "LCANNMT" (FC58).

Parameter	Direction	Type	Example
CanDB	IN	BLOCK-DB	DB50
Node	IN	INT	1
Func	IN	INT	1
Error	OUT	WORD	MW32

CanDB — Internal DB with up-to-date CAN data

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

Func — Network management function:
1 = Start node (code: 0x01)
2 = Stop node (code:0x80)
5 = Reset node (code:0x81)
6 = Reset communication (code: 0x82)
8 = Transmit SYNC frame
10 = Timer start (timer number in Node)
11 = Timer stop (timer number in Node)

Error — Error number on error in execution

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

The frame is transferred immediately by FC58 to the CAN300 module and thus transmitted. It is therefore not necessary to call FC59 "LCANCYCL".

4.1.8 Error numbers

Possible error numbers of the return parameter Error:

Number	Meaning
1	Node below 1
2	Node above 63
3	Channel value not correct
4	Direction wrong ('R' or 'W' ?)
101	Buffer assigned, busy with a request.
102	Abort code received
105	Function number unknown
140	Module not ready
141	Module cannot transmit (buffer overflow ?)
142	System error
198	System error
199	Timeout in parameter request

4.2 Example project

The example project includes a possible use for handling modules. Only one controller on node address 1 is addressed (C0350 = 1). An "8200 vector E82EV551" was used as the controller.

The process data channel of the controller must be set as follows:

CAN-IN1.W1.Bit 0 = DCTRL1-QSP	(C0410/4 = 30000)
CAN-IN1.W1.Bit 1 = DCTRL1-CINH	(C0410/10 = 310000)
CAN-IN1.W1.Bit 2 = DCTRL1-CW/CCW	(C0410/3 = 320000)
CAN-IN1.Word 2 = NSET1-N1	(C0412/1 = 210000)
CAN-OUT1.W2 = MCTRL1-NOUT+SLIP	(C0421/4 = 0)
CAN-OUT1.W3 = MCTRL1-MOUT	(C0421/5 = 10000)
CAN-OUT1.W4 = PCTRL1-ACT	(C0421/6 = 80000)

The process data channel is updated via the Sync frame that is transmitted from the CAN300 module by timer 0 every 500ms.

4.2.1 FC1 Demo

FC1 contains example calls of FC52 "LCANPDO" to activate the motor, of FC51 "LCANPARA" to fetch and transmit a parameter, and of FC58 "LCANNMT" to transmit network management frames.

The various functions of FC1 are triggered by the bits of input byte 8.

I 8.0 = Sends the process data frame to CAN-IN2 to activate the controller. Setpoint awaited in MW112.

I 8.1 = Clockwise/counterclockwise rotation of controller

I 8.2 = Quickstop on/off

I 8.6 = Fetch a parameter. Parameter number in MW60, SubCode in MB62, data returned to MD74

I 8.7 = Send a parameter. Parameter number in MW60, SubCode in MB62, data returned to MD70

I 8.3 = Send a network management frame to all nodes. Function code in MW30

4.2.2 FC2 Loading a parameter list

A list of parameters is read out of the controller with FC2. The parameters are stored in DB2 as a linked list. The values are then also stored in the DB.

Input bit 8.4 triggers read-out of the parameters. For that, DBW0 of DB2 must additionally be set to zero. The node number is stored in DBW2.

4.2.3 FC3 Transmitting a parameter list

A list of parameters is written to the controller with FC3. The parameters and their values are stored in DB3 as a linked list.

Input bit 8.5 initiates writing of the parameters. For that, DBW0 of DB3 must be additionally set to zero. The node number is stored in DBW2.

DB3 contains a list of parameters that prepare the controller for CAN communication.

5 Appendix

5.1 CAN identifiers

The identifier used by the Lenze system bus for the CAN frames (COB-IDs) are permanently assigned. They cannot be altered.

Message		Identifier for C0353/x = 0 (System bus address source is C0350)	Identifier for C0353/x = 1 (System bus address source is C0354/x)
Network management		0	
Sync telegram		128	
Parameter channel 1 to drive		1536 + address in C0350	
Parameter channel 2 to drive		1600 + address in C0350	
Parameter channel 1 from drive		1408 + address in C0350	
Parameter channel 2 from drive		1472 + address in C0350	
Process data channel to drive (CAN-IN1)	Sync-controlled (C0360 = 1)	512 + address in C0350	384 + address in C0354/1
	Time-controlled (C0360 = 0)	768 + address in C0350	384 + address in C0354/5
Process-data channel from drive (CAN-OUT1)	Sync-controlled (C0360 = 1)	384 + address in C0350	384 + address in C0354/2
	Time-controlled (C0360 = 0)	769 + address in C0350	384 + address in C0354/6
Process data channel from drive (CAN-IN2)		640 + address in C0350	384 + address in C0354/3
Process data channel from drive (CAN-OUT2)		641 + address in C0350	384 + address in C0354/4

These identifiers are not supported...

5.2 Technical data

Order number	CAN 300	700-600-CAN01
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

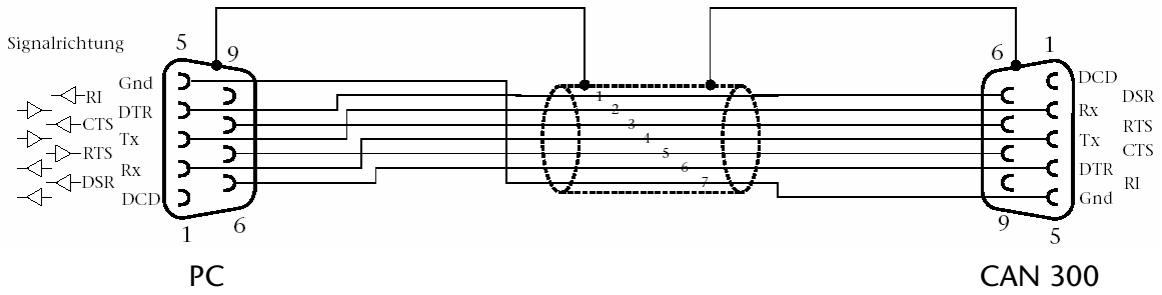
Quality assurance:	According to ISO 9001:2000
Maintenance:	Maintenance-free (no battery, rechargeable or non-rechargeable)

5.3 Pin assignment

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	-	-

5.4 Connecting cable

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



5.5 Further documentation

Internet: www.can-cia.org

"CAN 300 - Communications Module for CAN-Bus" manual

CAN Specification 2.0, Part A & Part B

Lenze instruction manual "Frequency converter series 8200 vector", Chapter 9 "Function Module System Bus"

Lenze "CAN Communications Manual", Chapter 5 & 10

Notes