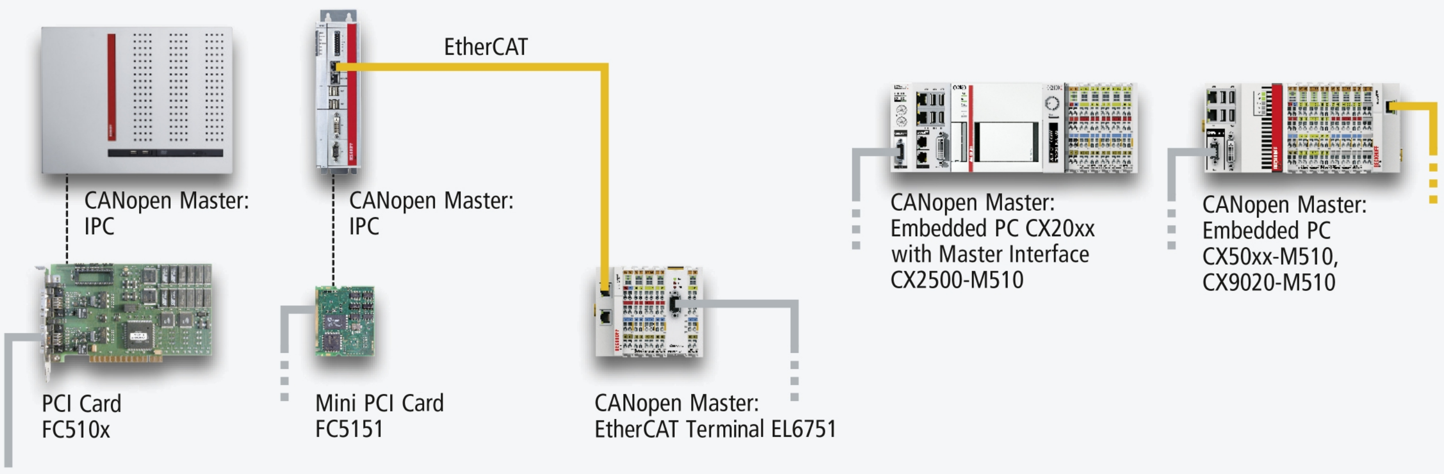
BECKHOFF New Automation Technology

Documentation | EN

CAN Interface

Interface for CANopen Master from Beckhoff



2022-08-31 | Version: 1.5

Table of contents

1 Foreword

1.1 Notes on the documentation

Intended audience

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning these components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without prior announcement.

No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents: EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702 with corresponding applications or registrations in various other countries.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Copyright

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

Description of instructions

In this documentation the following instructions are used.
These instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow this safety instruction directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow this safety instruction endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow this safety instruction can lead to injuries to persons.

NOTE

Damage to environment/equipment or data loss

Failure to follow this instruction can lead to environmental damage, equipment damage or data loss.

Tip or pointer

This symbol indicates information that contributes to better understanding.

1.3 Documentation Issue Status

Version	Comment
1.5	• Chapter *CAN FD access with FC532x and CX-M530* update
1.4	• Chapter *CAN FD access with FC532x and CX-M530* added
1.3	• Introduction updated • Safety instructions adapted to IEC 82079-1.
1.2	• Migration and structural adaptation
1.1	• Revision
1.0	• First release

2 Introduction

Almost all CANopen masters from Beckhoff offer the so-called CAN interface. The CAN interface is a Layer-2 implementation of the CAN interface. It enables any desired CAN telegrams to be received and transmitted. The higher-level protocol is not important here, i.e. all CAN-based protocols can be used; however the protocol part must then be implemented in the PLC.

The CAN interface consists of a buffer that is processed cyclically. The buffer can contain 11 to 32 data telegrams.

The transmit buffer (Tx) contains the data to be transmitted and the receive buffer (Rx) the data that have been received. 11-bit or 29-bit messages can be received or transmitted, depending on the CAN master. The buffer is processed with the cycle time of the task. With a buffer size of 10, therefore, a maximum of 10 CAN telegrams can be transmitted or received per task cycle.

11-bit identifier, also known as "Base Frame Format" (CAN 2.0A)

29-bit identifier, also known as "Extended Frame Format" (CAN 2.0B)

CAN interface – supported functions

	CAN2.0A 11-bit ID	CAN2.0B 29-bit ID	CAN FD	Fast CAN Queue1	Optimized CAN Queue1	Transaction Number [▸ 12]	Time Stamp2
EL6751 Legacy Mapping	✓	✓	-	-	✓	✓3	-
EL6751 MDP Mapping	✓	✓	-	✓	✓	✓	-
CCAT	(✓)4	✓	-	-	-	✓ from FW 1.17	✓ from FW 1.17
CX1500-M510	✓	✓	-	-	-	-	-
FC5151 FC510x	✓	✓ from FW 2.14	-	-	-	-	-
FC532x CX-M5305	✓	✓	✓	-	-	✓	✓

1) not in 29-bit mode, not with Transaction Number [▸ 12]
2) only in 29-bit mode and with Transaction Number [▸ 12]
3) only in 29-bit mode
4) covered by the 29-bit ID option
5) in preparation

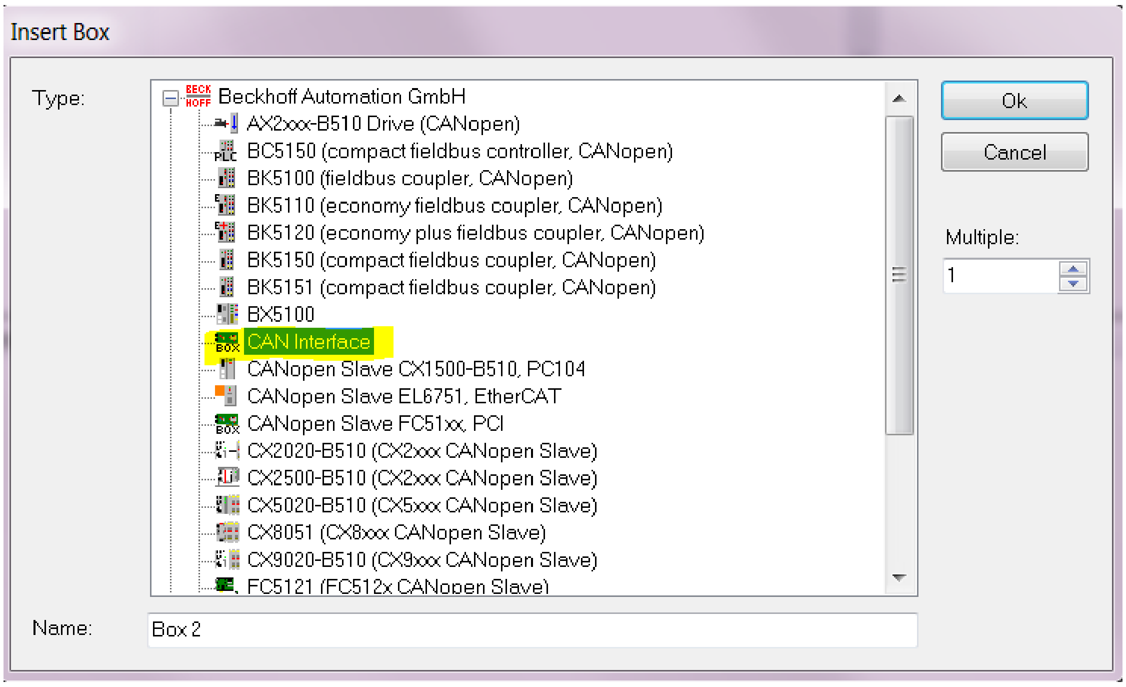
CCAT

What is CCAT? Which Beckhoff products is it backed by?

The CCAT interface is the Beckhoff company's current CAN implementation and is used by the Beckhoff PCI-Express cards and the onboard interfaces of the Beckhoff Embedded PCs. These are, for example, the following products and only available for the CANopen Master: FC512x, C20xx-M510, CX8x50, CX9x20-M510, CX51xx-M510

3 Integration in TwinCAT

If you have created a CANopen master in TwinCAT, you can select the CAN interface instead of CANopen slaves.



Insert Box

Type:

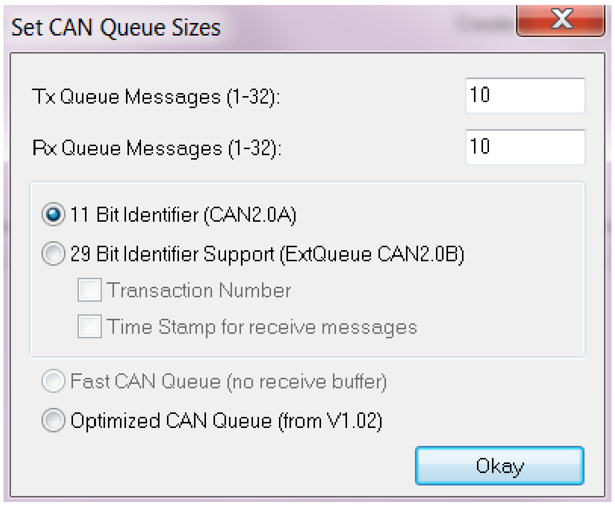
Beckhoff Automation GmbH
- AX2xxx-B510 Drive (CANopen)
- BC5150 (compact fieldbus controller, CANopen)
- BK5100 (fieldbus coupler, CANopen)
- BK5110 (economy fieldbus coupler, CANopen)
- BK5120 (economy plus fieldbus coupler, CANopen)
- BK5150 (compact fieldbus coupler, CANopen)
- BK5151 (compact fieldbus coupler, CANopen)
- BX5100
- CAN Interface
- CANopen Slave CX1500-B510, PC104
- CANopen Slave EL6751, EtherCAT
- CANopen Slave FC51xx, PCI
- CX2020-B510 (CX2xxx CANopen Slave)
- CX2500-B510 (CX2xxx CANopen Slave)
- CX5020-B510 (CX5xxx CANopen Slave)
- CX8051 (CX8xxx CANopen Slave)
- CX9020-B510 (CX9xxx CANopen Slave)
- FC5121 (FC512x CANopen Slave)

Name: Box 2

Multiple: 1

Fig. 1: Selecting the CAN interface

You will then be requested to select the appropriate interface that you wish to use, or the size of the buffer. If you change the interface again later on, please note that it is usually the case that, depending on the mode, the interface is set up again and links are thus deleted. Also, select only modes that your hardware actually supports (see Table *CAN Interface – supported functions*).



Set CAN Queue Sizes

Tx Queue Messages (1-32): 10

Rx Queue Messages (1-32): 10

[✓ 11 Bit Identifier (CAN2.0A)]

29 Bit Identifier Support (ExtQueue CAN2.0B)

Transaction Number

Time Stamp for receive messages

Fast CAN Queue (no receive buffer)

Optimized CAN Queue (from V1.02)

Fig. 2: Selection of the frame format

4 Buffer size in relation to the cycle time

You will have to estimate how big the buffer has to be or how fast the task time must be. The following table will help you to define this in advance.

The CCAT CAN master has a memory for 512 messages, the EL6751 for 150 messages (RxMessages). The data will be lost if they are not fetched quickly enough from the memory. No indication is given, therefore the worst case should or must be estimated, or the variable NoOfRxMessages should as far as possible be smaller than the maximum buffer value. If this is always or in almost every cycle at the maximum value, then this indicates that more data are being received than can be recorded per cycle. Remedy: Shorten the task cycle time or enlarge the buffer of the CAN queue.

Example

A CAN telegram with 11-bit identifier and 8 bytes of user data needs about 260 µs at 500 Kbit/s. If one assumes a 100% bus load in the worst case, it would be maximally 3 telegrams with 1 ms. This means that a buffer of maximally 4 would be adequate in this case. If a task time of 5 ms is used instead of 1 ms, the buffer should be at least 20 (5000 µs / 260 µs). It must be remembered here that in this study only the data in one direction are considered and that the CAN data always have 8 bytes. Since a 100% bus load is not usually assumed, one can also evaluate the variable NoOfRxMessages and see whether it lies in most cases below the maximum number of buffers created. If NoOfRxMessages is often at the maximum value, the task time should be shortened or the buffer enlarged.

Worst case

However, the CAN interface is designed such that, as a rule, the data can always be fetched faster than they run into the buffer.

Example

1 MBaud data length 0 means 50 µs per CAN message. With a 1 ms task time this would be
1000 µ / 50 µs = 20
This means that even in this extreme worst case a buffer of 20 would be adequate to receive all CAN telegrams.

Table for the telegram runtimes with 11-bit ID [ms][1)]

Bit rate [kbit/s]	Data length in bytes								
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
50	1.09	1.28	1.47	1.66	1.86	2.05	2.24	2.34	2.62
125	0.44	0.51	0.59	0.67	0.74	0.82	0.90	0.97	1.05
250	0.22	0.26	0.29	0.33	0.37	0.41	0.45	0.49	0.52
500	0.11	0.13	0.15	0.17	0.19	0.21	0.22	0.24	0.26
1000	0.05	0.06	0.07	0.08	0.09	0.10	0.11	0.12	0.13

[1)] Data from CIA

5 Functionalities

5.1 CAN queues

5.1.1 Fast CAN Queue (unbuffered)

Received messages are no longer buffered on the EL6751. The RxQueue should therefore be large enough for all messages theoretically receivable within an EtherCAT cycle to fit inside.

The received messages no longer have to be confirmed. The EL6751 increments the RxCounter when new messages are received.

In the transmission direction too, only the data dependent on the changed TxCounter and the NoOfTxMessages are copied, so that the number of parallel messages in the queue actually have no further influence on the runtime (only the NoOfTxMessages).

Since the EL6751 operates in 3-buffer mode (so that it always has a buffer in which it can copy the CAN messages received), it may be the case that the **unused** messages contain incorrect or old data.

The object directory can be read on the *CoE-Online* tab. If the index 0x1C32:08 has been/is set to 1, the local cycle time of the EL6751 is measured and stored in index 0x1C32:05 (maximum value). You can thus see whether the EL6751 will be finished within the EtherCAT cycle.

The Fast CAN Queue may not contain further CANopen or CAN-Layer-2 nodes.

5.1.2 Optimized CAN Queue (buffered)

The received messages of the EL6751 are buffered. The EL6751 operates in 1-buffer mode.

The following applies to both functions, Fast CAN Queue and Optimized CAN Queue:

Advantages

- Higher processing speed
- The Fast CAN Queue does without all attachments and is thus predestined for the fastest processing/ reaction of the data from the bus.

Disadvantages

- Both modes support only 11-bit identifiers.
- No filters may be used.

5.2 Transaction Number

On the basis of the *transactionNumber* one can determine the CAN message in a CAN queue up to which transmission took place in the last CAN cycle. With the individual TxMessages[n], any *transactionNumber* can be entered (e.g. a sequential number). At the end of a CAN cycle the *transactionNumber* of the last-sent TX message is written in Inputs.RxQueue.TransactionNumber.

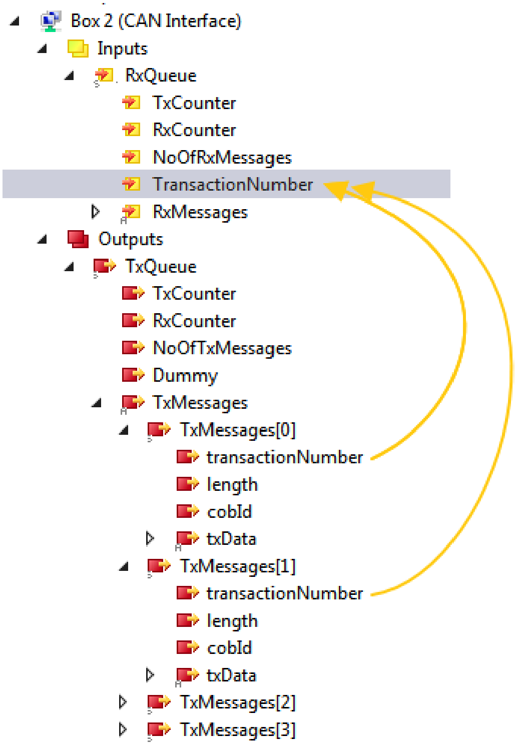


Fig. 3: Transaction Number

5.3 Time Stamp

CCAT-based CAN controllers (e.g. FC512x, -M510) return the time at which the CAN frame arrived with a reception time stamp (64-bit integer value in nanoseconds).

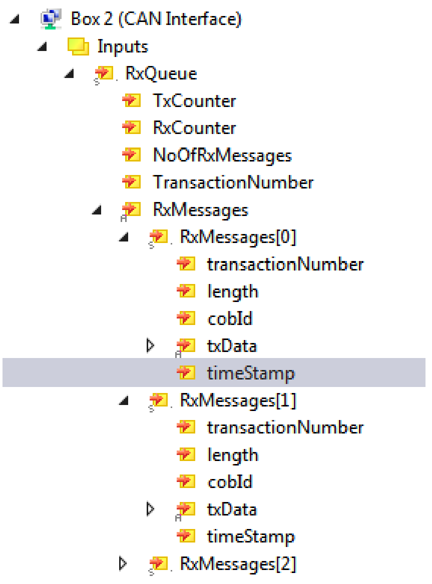


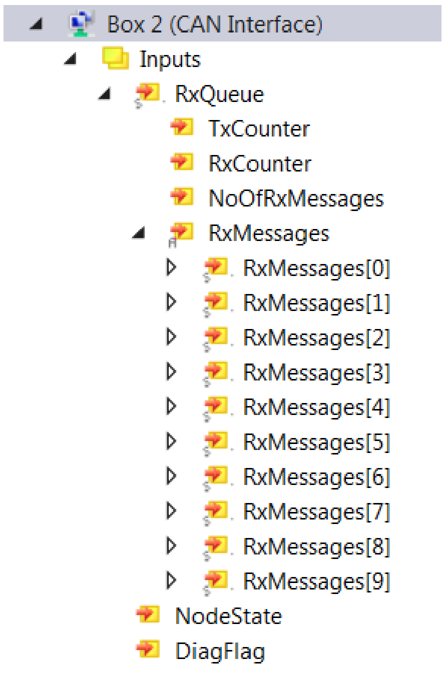
Fig. 4: Time Stamp

6 Structure of the CAN interface

The CAN interface looks different depending on the options selected. An 11-bit identifier interface is structured differently to a 29-bit identifier interface. In addition, the interface may contain the Transaction Number as well as the Time Stamp. When using structures the target system must be considered and which alignment it supports. Corresponding attributes are usable under TwinCAT 3 {attribute 'pack_mode':= '0'}.

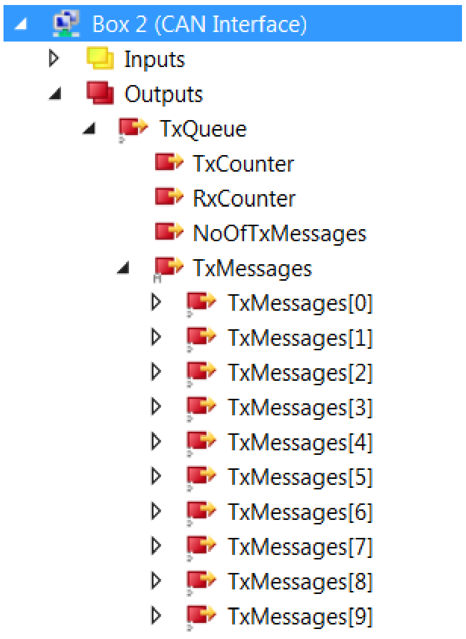
The interface consists of the communication with the interface and up to 32 CAN messages. The inputs can only be read if the outputs are written.

The interface is addressed as follows:



```
◢ Box 2 (CAN Interface)
  ◢ Inputs
    ◢ RxQueue
        TxCounter
        RxCounter
        NoOfRxMessages
      ◢ RxMessages
        ▷ RxMessages[0]
        ▷ RxMessages[1]
        ▷ RxMessages[2]
        ▷ RxMessages[3]
        ▷ RxMessages[4]
        ▷ RxMessages[5]
        ▷ RxMessages[6]
        ▷ RxMessages[7]
        ▷ RxMessages[8]
        ▷ RxMessages[9]
      NodeState
      DiagFlag
```

Fig. 5: CAN interface - inputs



```
◢ Box 2 (CAN Interface)
  ▷ Inputs
  ◢ Outputs
    ◢ TxQueue
        TxCounter
        RxCounter
        NoOfTxMessages
      ◢ TxMessages
        ▷ TxMessages[0]
        ▷ TxMessages[1]
        ▷ TxMessages[2]
        ▷ TxMessages[3]
        ▷ TxMessages[4]
        ▷ TxMessages[5]
        ▷ TxMessages[6]
        ▷ TxMessages[7]
        ▷ TxMessages[8]
        ▷ TxMessages[9]
```

Fig. 6: CAN interface - outputs

Outputs.TxCounter is set to +1 if data are to be transmitted. NoOfTxMessages also indicates how many messages are to be transmitted from the buffer. The RxCounter indicates whether there are new data in the buffer. NoOfRxMessages indicates how many new data are in the buffer.

If you have fetched the data, then set Outputs.RxCounter:=Inputs.RxCounter. The CAN interface then knows that it can fill the buffer again. All data must always be read out, because the CAN interface fills all message structures again when necessary.

Sample code for transmission

```
if Outputs.TxCounter = Inputs.TxCounter then
      for i=0 to NumberOfMessagesToSend do
               Outputs.TxMessage[i] = MessageToSend[i];
      End_for
      Outputs.NoOfTxMessages = NumberOfMessagesToSend;
      Outputs.TxCounter := Outputs.TxCounter + 1;
end_if
```

Sample code for reading

```
if Outputs.RxCounter <> Inputs.RxCounter then
       for I := 0 to (Inputs.NoOfRxMessages-1) do
            MessageReceived[i] := Inputs.RxMessage [i];
       End_for
       Outputs.RxCounter := Inputs.RxCounter;
end_if
```

Message structure when using the 11-bit identifier

The message structure when using the 11-bit identifier consists of the COB ID [2 bytes] and the 8 bytes of data. The COB ID has the following structure:

- Bit 0-3: Length of the data (0…8)
- Bit 4: RTR
- Bit 5-15: 11-bit identifier

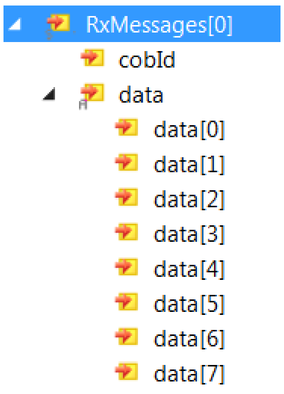


Fig. 7: Message structure when using the 11-bit identifier

Since COB ID, length and RTR bit are coded in one word in the 11-bit identifier, the following example is helpful for decoding the data from the word. Select a structure here in which to store the decoded data.

```
IF RXCounter_Out <> RXCounter_In THEN
   FOR I := 0 TO (NoOfTxMessages-1) DO
       stCANInterfaceMessageValue[i].Lengh:=WORD_TO_BYTE(stCANInterfaceMessage[i].CobID) AND 16#0F;
       stCANInterfaceMessageValue[i].RTR:=stCANInterfaceMessage[i].CobID.4;
       stCANInterfaceMessageValue[i].CobID :=ROR(stCANInterfaceMessage[i].CobID,5) AND 16#07FF;
       stCANInterfaceMessageValue[i].Data := stCANInterfaceMessage[i].Data;
       CASE stCANInterfaceMessageValue[i].CobID OF
            16#318: COB318:=COB318+1;
            16#718: COB718:=COB718+1;
            16#1CD: COB1CD:=COB1CD+1;
            memcpy(ADR(TempValue),ADR(stCANInterfaceMessage[i].Data[6]),2);
            16#1ED: COB1ED:=COB1ED+1;
       ELSE
            COBALLOther:=COBAllOther+1;
       END_CASE
   End_for
   RXCounter_Out:=RXCounter_In;
END_IF
```

Message structure when using the 29-bit identifier

The message structure when using the 29-bit identifier consists of the length [2 bytes] of the COB ID [4 bytes] and the 8 bytes of data.

Lenght: Length of the data (0…8)

The COB ID has the following structure:

- Bit 0-28: 29-bit identifier
- Bit 30: RTR
- Bit 31:
  - 0: 11-bit identifier,
  - 1: 29-bit identifier

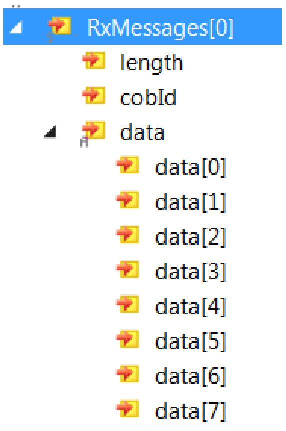
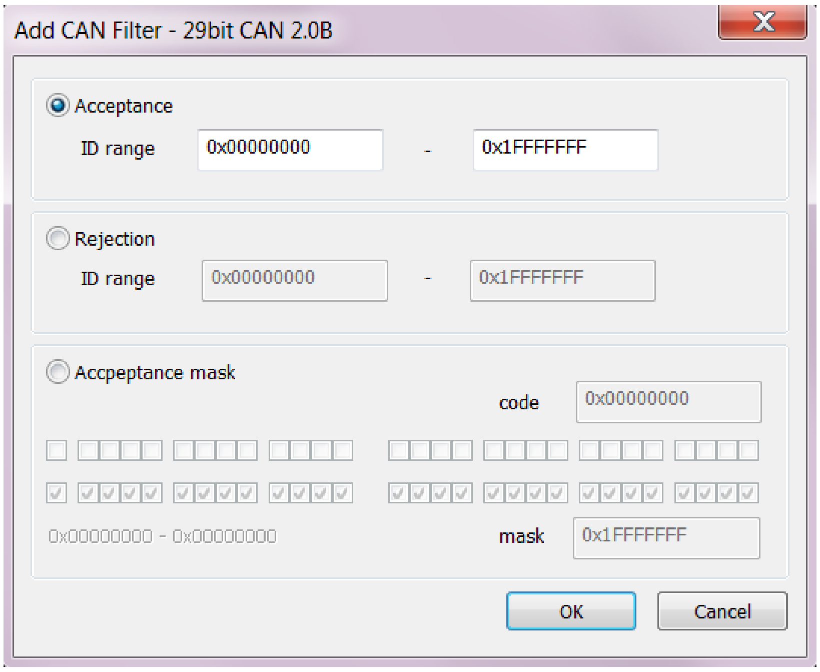


Fig. 8: Message structure when using the 29-bit identifier

7 Use of a filter

If you don't wish to receive all the telegrams in the CAN interface, there is an option to set filters. This reduces the number of CAN telegrams in the CAN interface and thus permits only those telegrams that are actually required.


Add CAN Filter - 29bit CAN 2.0B

[✓ Acceptance]
ID range: 0x00000000 - 0x1FFFFFFF

Rejection
ID range: 0x00000000 - 0x1FFFFFFF

Accpeptance mask
code: 0x00000000
0x00000000 - 0x00000000
mask: 0x1FFFFFFF

Fig. 9: CAN filters

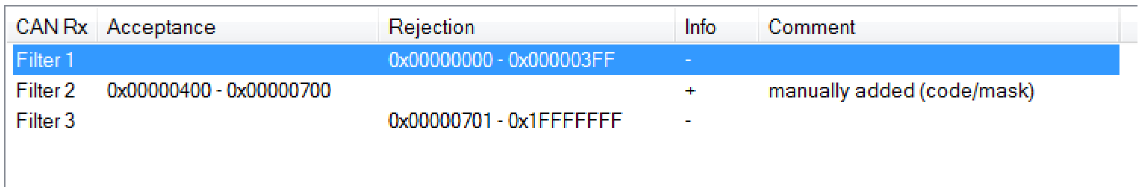
Acceptance:
The identifiers that are to be forwarded to the CAN interface are entered here.

Rejection:
The identifiers that are not to be forwarded to the CAN interface are entered here.

Acceptance mask:
Here you can specify at bit level which identifiers are to be forwarded to the CAN interface.

Example on the basis of the 29-bit identifier

In the example, all the telegrams from identifier 0x0400 … 0x0700 are transmitted into the CAN interface. This is displayed with a "+" next to Info.
"+" means that the filter lets the data through to the CAN interface (Acceptance)
"-" means that the filter does not let the data through to the CAN interface (Rejection)



CAN Rx	Acceptance	Rejection	Info	Comment
Filter 1		0x00000000 - 0x000003FF	-	
Filter 2	0x00000400 - 0x00000700		+	manually added (code/mask)
Filter 3		0x00000701 - 0x1FFFFFFF	-	

Fig. 10: Example on the basis of the 29-bit identifier

8 CAN FD access with FC532x and CX-M530

Sending and receiving FD messages

This chapter describes the CAN FD function of the CAN FD interface.

8.1 CAN FD interface

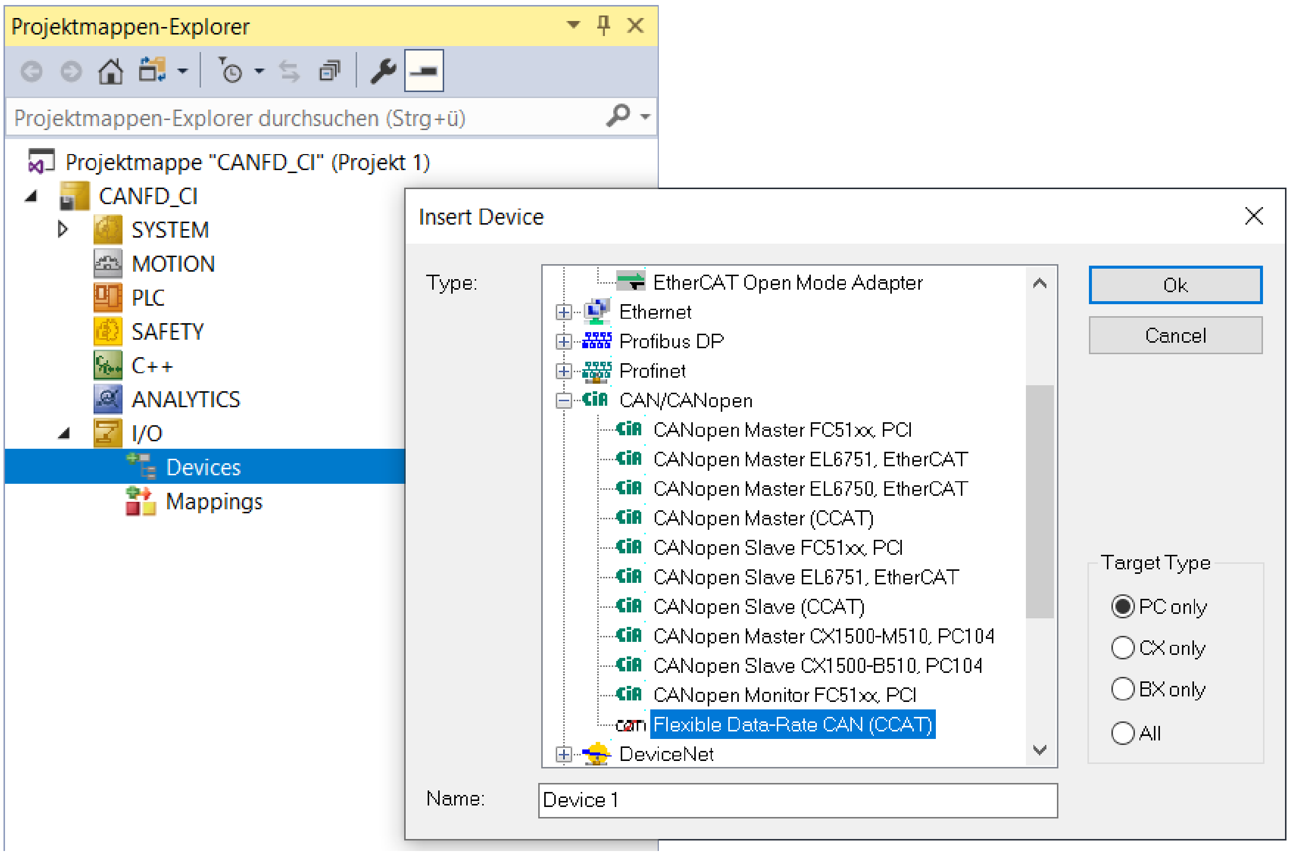
The CAN interface for the FC532x, CX2500-M530 and the option interface for CX-M530 CAN support access to the CAN FD functionality.

The operation of this CAN interface as well as the functions *Transactions-Number* and *Timestamp* correspond to that of the known interface (see chapter Structure of the CAN interface [▶ 13]).

New are the message data type of the RX and TX queue and the baud rate setting.

8.2 CAN FD Device and baud rate setting

The CAN FD function can be accessed via the device *Flexible Data-Rate CAN (CCAT)* .


Projektmappen-Explorer
Projektmappen-Explorer durchsuchen (Strg+ü)
Projektmappe "CANFD_CI" (Projekt 1)
CANFD_CI
SYSTEM
MOTION
PLC
SAFETY
C++
ANALYTICS
I/O
Devices
Mappings
Insert Device
EtherCAT Open Mode Adapter
Ethernet
Profibus DP
Profinet
CAN/CANopen
CANopen Master FC51xx, PCI
CANopen Master EL6751, EtherCAT
CANopen Master EL6750, EtherCAT
CANopen Master (CCAT)
CANopen Slave FC51xx, PCI
CANopen Slave EL6751, EtherCAT
CANopen Slave (CCAT)
CANopen Master CX1500-M510, PC104
CANopen Slave CX1500-B510, PC104
CANopen Monitor FC51xx, PCI
DeviceNet
Ok
Cancel
CX only
BX only
All

Type: Flexible Data-Rate CAN (CCAT)

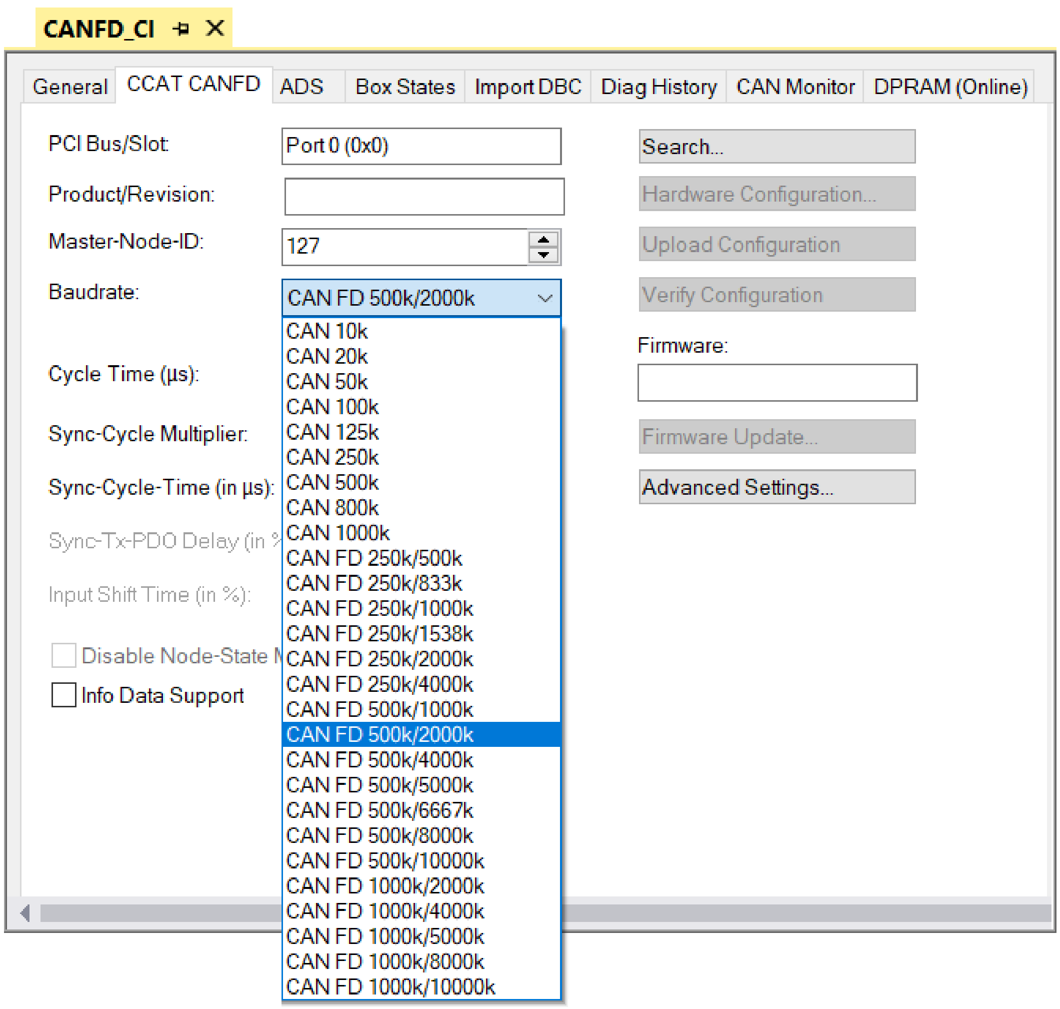
Name: Device 1

Target Type: PC only

Fig. 11: Flexible Data Rate CAN (CCAT)

The baud rates for CAN FD can be set differently for the arbitration phase and the data transmission phase. Likewise, it is still possible to set the same baud rate for both phases as with classic CAN.

The following baud rates are possible:



PCI Bus/Slot: Port 0 (0x0)
Product/Revision:
Master-Node-ID: 127
Baudrate: CAN FD 500k/2000k
Cycle Time (µs):
Sync-Cycle Multiplier:
Sync-Cycle-Time (in µs):
Sync-Tx-PDO Delay (in %
Input Shift Time (in %):
Firmware:

CAN 10k
CAN 20k
CAN 50k
CAN 100k
CAN 125k
CAN 250k
CAN 500k
CAN 800k
CAN 1000k
CAN FD 250k/500k
CAN FD 250k/833k
CAN FD 250k/1000k
CAN FD 250k/1538k
CAN FD 250k/2000k
CAN FD 250k/4000k
CAN FD 500k/1000k
CAN FD 500k/2000k
CAN FD 500k/4000k
CAN FD 500k/5000k
CAN FD 500k/6667k
CAN FD 500k/8000k
CAN FD 500k/10000k
CAN FD 1000k/2000k
CAN FD 1000k/4000k
CAN FD 1000k/5000k
CAN FD 1000k/8000k
CAN FD 1000k/10000k

Fig. 12: Possible baud rates

The first number indicates the baud rate for the arbitration phase and the second for the data phase.

Note on baud rate 10 Mbit/s

The baud rate 10 Mbit/s is currently only possible under very ideal conditions and does not correspond to a practical setting at the moment. A currently common setting for CAN FD is 500k/2000k.

8.3 CAN FD Message data structures

The following new data structures have been introduced for CAN FD support at the CAN interface.

```
TYPE CANFDTSRXQUEUE :
      STRUCT
            dataLenght : BYTE;
            EDL : BIT;
            BSR : BIT;
            ESI : BIT;
            cobId : UDINT;
            rxData : CANFDMESSAGE;
            timeStamp : ULINT;
      END_STRUCT
END_TYPE
```

```
TYPE CANFDTXQUEUE :
      STRUCT
            transactionNumber : UINT;
            dataLenght : BYTE;
            EDL : BIT;
            BSR : BIT;
            ESI : BIT;
            cobId : UDINT;
            txData : CANFDMESSAGE;
      END_STRUCT
END_TYPE
```

```
TYPE CANFDMESSAGE :
      ARRAY [0..63] OF USINT;
END_TYPE
```

These structures are available for the IOs in the System Manager as well as in the PLC.

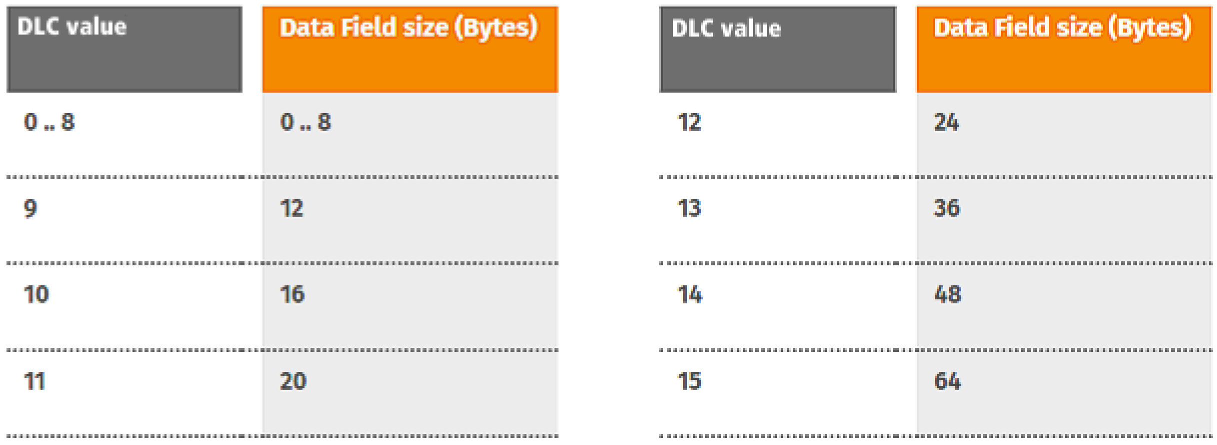
8.3.1 Data length

For the data length values up to 64 bytes are possible for Can FD frames. Since these values are transmitted in the classic CAN arbitration header, the following values are possible:

0 ... 8, 12, 16, 20, 24, 36, 48 and 64

If the lengths do not correspond to these values during transmission, they are adjusted to the next higher value by the device.

For a CAN FD frame the values 0 to 15 are valid for the DLC field. The value of the DLC field determines the number of bytes in the data field and is interpreted in the following way:



DLC value	Data Field size (Bytes)	DLC value	Data Field size (Bytes)
0 .. 8	0 .. 8	12	24
9	12	13	36
10	16	14	48
11	20	15	64

Fig. 13: DLC value vs. Data field size

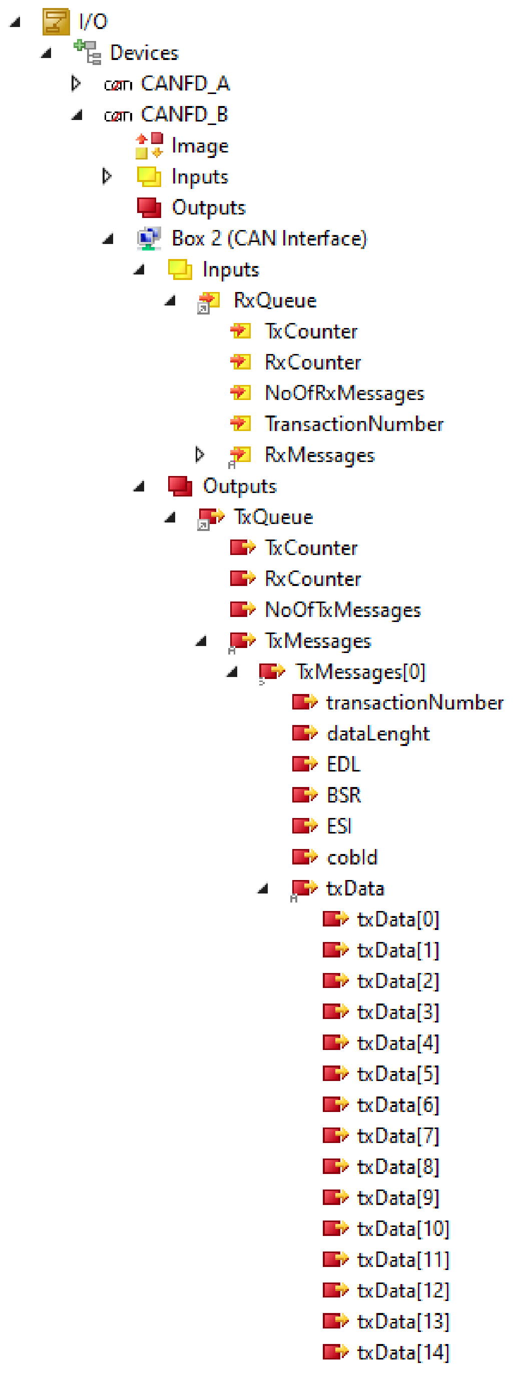
The CAN interface does this conversion automatically, i.e. in the CAN interface you only specify the actual number of bytes (values from 0..64 bytes are allowed).

The conversion to the DLC values is then performed by the CAN interface in the background with the next larger DLC value.

Example: if you enter the data length 32 in the CAN interface, 36 bytes are sent with the DLC value 13.

8.3.2 CAN FD bits

The new bits in the data structure *Queue* compared to the classic CAN interface have the following meanings.



```
I/O
  Devices
    CANFD_A
    CANFD_B
      Image
      Inputs
      Outputs
      Box 2 (CAN Interface)
        Inputs
          RxQueue
            TxCounter
            RxCounter
            NoOfRxMessages
            TransactionNumber
            RxMessages
        Outputs
          TxQueue
            TxCounter
            RxCounter
            NoOfTxMessages
            TxMessages
              TxMessages[0]
                transactionNumber
                dataLenght
                EDL
                BSR
                ESI
                cobId
                txData
                  txData[0]
                  txData[1]
                  txData[2]
                  txData[3]
                  txData[4]
                  txData[5]
                  txData[6]
                  txData[7]
                  txData[8]
                  txData[9]
                  txData[10]
                  txData[11]
                  txData[12]
                  txData[13]
                  txData[14]
```

Fig. 14: Data structure *Queue*

EDL
The **EDL** (Enhanced Data Length) bit is used to control whether an FD or a classic frame is to be sent or what kind of frame was received.

BSR
The **BSR** (Bit Rate Switched) bit specifies whether in the data phase should be switched to the higher baud rate or how the frame was received.

ESI
The bit **ESI** (Error State Indicator) indicates whether there was (Rx) or is (Tx) an error with the frame.

8.3.3 InfoData

The following data can be read from the info data of the FC532x / CX-M530:

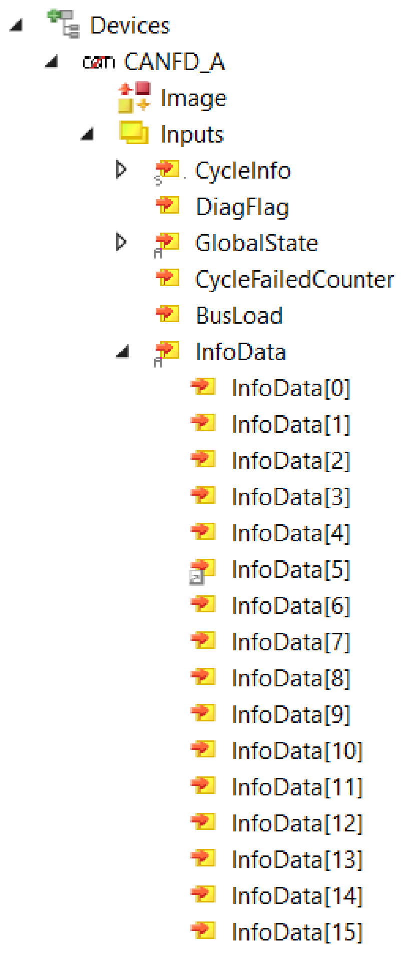


Fig. 15: InfoData

InfoData[0] – CAN Status
Bit0 – BusOff (0 no error - 1 bus off error)
Bit1 – Error Passive
Bit2 – Node Active
Bit3 – Warning Limit
Bit4 – Overload
Bit6 – Bus Idle;

InfoData[1] – Arbitration phase baudrate
Byte0 = Jump Width
Byte1 = Time A
Byte2 = Time B
Byte3 = Pre Scaler:

InfoData[2] – Data phase baudrate
Byte0 = Jump Width
Byte1 = Time A
Byte2 = Time B
Byte3 = Pre Scaler:

InfoData[4] – Send Fifo (High-Prio) UsedWords;

InfoData[5] – Send Fifo (Low-Prio) UsedWords;

InfoData[6] – Receive Fifo Counter;

InfoData[7] – AckError Counter;

InfoData[8] – BitError Counter;

InfoData[9] – CrcError Counter;

InfoData[10] – FormError Counter;

9 Appendix

9.1 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on her internet pages: https://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963 157
Fax:	+49 5246 963 9157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963 460
Fax:	+49 5246 963 479
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963 0
Fax:	+49 5246 963 198
e-mail:	info@beckhoff.com
web:	https://www.beckhoff.com

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com