COMBIVERT **Generation 6**

PROGRAMMING MANUAL | Fieldbus systems **– V 3.2**

Translation of the original manual
Document 20430097 EN 00

1 Preface

The hardware and software described are developments of KEB Automation KG. The enclosed documents correspond to conditions valid at printing. Misprint, mistakes and technical changes reserved.

1.1 Signal words and symbols

Certain operations can cause hazards during the installation, operation or thereafter. There are safety informations in the documentation in front of these operations. Security signs are located on the device or on the machine. A warning contains signal words which are explained in the following table:

⚠ DANGER	➢ Dangerous situation, which will cause death or serious injury in case of non-observance of this safety instruction.
⚠ WARNING	➢ Dangerous situation, which may cause death or serious injury in case of non-observance of this safety instruction.
⚠ CAUTION	➢ Dangerous situation, which may cause minor injury in case of non-observance of this safety instruction.
NOTICE	➢ Situation, which can cause damage to property in case of non-observance.

RESTRICTION

Is used when certain conditions must meet the validity of statements or the result is limited to a certain validity range.

i	➢ Is used if the result is better, more economical or more trouble-free if the instructions are followed.

1.2 More symbols

► This arrow starts an action step.

• / - Enumerations are marked with dots or indents.

=> Cross reference to another chapter or another page.

	Note to further documentation. Document search on www.keb.de	

1.3 Laws and guidelines

KEB Automation KG confirms with the CE mark and the EU declaration of conformity, that our device complies with the essential safety requirements.

The EU declaration of conformity can be downloaded on demand via our website. Further information is provided in the chapter "Certification".

1.4 Warranty

The warranty and liability on design, material or workmanship defects for the acquired device is given in the general conditions of sale.

	Here you will find our general sales conditions. AGB	

Further agreements or specifications require a written confirmation.

1.5 Support

Through multiple applications not every imaginable case has been taken into account. If you require further information or if problems occur which are not treated detailed in the documentation, you can request the necessary information via the local KEB Automation KG agency.

The use of our devices in the target products is beyond of our control and therefore exclusively the responsibility of the customer.

The information contained in the technical documentation, as well as any user-specific advice in spoken and written and through tests, are made to best of our knowledge and information about the intended use. However, they are considered for information only without responsibility and changes are expressly reserved, in particular due to technical changes. This also applies to any violation of industrial property rights of a third-party. Selection of our products in view of their suitability for the intended use must be done generally by the user.

Tests can only be carried out by the customer within the scope of the intended end use of the product (application). They must be repeated, even if only parts of hardware, software or the device adjustment are modified.

1.6 Copyright

The customer may progamming manual for use as well as further documents or parts from it for internal purposes. Copyrights are with KEB Automation KG and remain valid in its entirety.

This KEB product or parts thereof may contain foreign software, incl. free and/or open source software. If applicable, the license terms of this software are contained in the instructions for use. The instructions for use are already available, can be downloaded from the KEB website or can be requested from the respective KEB contact person.

Other wordmarks and/or logos are trademarks (™) or registered trademarks (®) of their respective owners.

Content

Figures

Tables

2 Basic Safety Instructions

The COMBIVERT is constructed and built according to the state of the art and the recognized safety regulations. Nevertheless, their use may create dangers to life and limb of the user or third parties or damage to the machine and other material property.

The following safety instructions have been created by the manufacturer for the area of electric drive technology. They can be supplemented by local, country or application-specific safety instructions. This list is not exhaustive. Non-observance of the safety instructions by the customer, user or other third party leads to the loss of all claims against the manufacturer.

NOTICE

Hazards and risks through ignorance!

- ➢ Read all parts of the instructions for use!
- ➢ Observe the safety and warning instructions!
- ➢ If anything is unclear, please contact KEB!

2.1 Target group

This manual is intended exclusively for electrical personnel. Electrical personnel for the purpose of this manual must have the following qualifications:

- Knowledge and understanding of the safety instructions.
- Skills for installation and assembly.
- Start-up and operation of the product.
- Understanding about the function in the used machine.
- Recognition of dangers and risks of electrical drive technology.
- Knowledge about DIN IEC 60364-5-54.
- Knowledge about national accident prevention regulations (e.g. *DGUV regulation 3*) .
- When operating with the COMBIVIS operating software, basic knowledge of the Windows operating system is required.

2.2 Validity of this manual

This programming manual is part of the instructions for use.

The programming manual

describes the parameterization

- of the Multi-Real-Time Ethernet module at F6A and S6A
- of the fieldbus interface on F6P and S6P
- the Ethernet TCP/IP interface on F6P and S6P
- contains only supplementary safety instructions.
- is only valid in connection with the installation manual and programming manual of the respective device.

2.3 Electrical connection

⚠ DANGER

Electrical voltage at terminals and in the device !
Danger to life due to electric shock !

- ➢ For any work on the device switch off the supply voltage and secure it against switching on.
- ➢ Connect or disconnect all pluggable terminals in a deenergised state only.
- ➢ Wait until all drives have stopped in order that no regenerative energy can be generated.
- ➢ Await capacitor discharge time (5 minutes) if necessary, measure DC voltage at the terminals.
- ➢ Never bridge upstream protective devices (also not for test purposes).
- ➢ For operation, install all necessary covers and protective devices.

For a trouble-free and safe operation, please pay attention to the following instructions:

- The electrical installation shall be carried out in accordance with the relevant requirements.
- Cable cross-sections and fuses must be dimensioned according to the design of the machine manufacturer. Specified minimum / maximum values may not be exceeded.
- With existing or newly wired circuits the person installing the units or machines must ensure the EN requirements are met.
- For drive converters without safe isolation from the supply circuit (in accordance with *EN 61800-5-1*), all control cables must be included in further protective measures (e.g. double insulated or shielded, earthed and insulated).
- When using components without isolated inputs/outputs, it is necessary that equipotential bonding exists between the components to be connected (e.g. by the equipotential line). Disregard can cause destruction of the components by equalizing currents.

2.4 Start-up and operation

The drive controller must not be started until it is determined that the installation complies with the machine directive; Account is to be taken of *EN 60204-1*.

⚠ WARNING

Software protection and programming!
Hazards caused by unintentional behavior of the drive!

- ➢ Check especially during initial start-up or replacement of the drive controller if the parameterization is compatible to the application.
- ➢ Securing a unit solely with software-supported functions is not sufficient. It is imperative to install external protective measures (e.g. limit switch) that are independent of the drive controller.
- ➢ Secure motors against automatic restart.

3 Product Description

3.1 Functional overview

The topics covered in these instructions are listed below:

- Operation of the fieldbus interface
- Details of the EtherCAT interface
- Details of the CANopen interface
- Details of the PROFINET interface
- Details of the POWERLINK interface
- Details of the EtherNet/IP™ interface
- Modbus interface details
- Details of the Ethernet TCP/IP interface
- Details of parallel fieldbus operation

3.2 Used terms and abbreviations

Term	Description
Adapter	EtherNet/IP™ slave during process data communication
AS	Automation Studio (B&R development environment for POWERLINK)
ASCII code	Standard for 7-bit character encoding
Assemblies	Assemblies of supported communication channels for EtherNet/IP™
Baud rate	Data transmission rate of a serial interface in characters / s
Bit rate	Data transmission rate of a serial interface in bit/ s
CAN / CANopen	Fieldbus system
CiA	CAN in Automation Association
CiA DS301	CAL based communication profile
CiA DS402	CAN device profile for drives
CIP	Common Industrial Protocol
Client	Modbus/TCP name for the (fieldbus) master
CN	Controlled Node (POWERINK Slave)
COMBIVERT	KEB drive controller
COMBIVIS	KEB start-up and parameterization software
Controller	PROFINET master
Device	PROFINET slave
DIN	German Institute for Standardization
Recipient (consumer)	EtherNet/IP™ master during SDO communication
EMC	Electromagnetic compatibility
EN	European standard
EoE	Ethernet over EtherCAT
ESI file	EtherCAT Slave Information file
EtherCAT	Real-time Ethernet bus system; EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany. It is marked by the following logo:

Term	Description
	EtherCAT®
Ethernet	Real-time bus system - defines protocols, connectors, cable types
EtherNet/IP™	Real-time Ethernet bus system of the company Rockwell Automation
Explicit Messaging	Exchange of data via request/response-oriented functions
Fieldbus	(Real-time capable) communication protocol at field level
FSoE	Functional safety via EtherCAT
GSDML	PROFINET device description file
HSP5	Fast, serial protocol
IEC	International standard
IEEE	Institute of Electrical and Electronics Engineers
Implicit Messaging	Exchange of data via an I/O connection
(Fieldbus) master	Fieldbus node responsible for controlling the processes
MN	Managing Node (POWERLINK Master)
Modbus	Communication protocol based on client/server architecture
Mod	Modbus protocol in which the data is transmitted via TCP/IP
Modb	Modbus protocol in which data is transmitted in binary form
Mo	Modbus protocol in which data is transmitted via ASCII code
Modulation	Means in drive technology that the power semiconductors are controlled.
MRTE	Multi-Realtime-Ethernet
MTTF	Mean service life to failure
NMT	Network Management Telegram
ODVA	Open DeviceNet Vendors Association
PD-In	Process input data (from perspective of the control)
PD-Out	Process output data (from perspective of the control)
PDO	Process data object
POWERLINK	Real-time Ethernet bus system of the company B&R
PROFINET	Industrial Ethernet standard of the PROFIBUS user organization e.V.
PROFIsafe	Functional safety via PROFINET or PROFIBUS
Process data mapping	Process data mapping (mapping of the configured process data)
Cross-communication	CANopen communication between fieldbus slaves
Request	Request (in connection with SDO communication)
Response	Response (in connection with SDO communication)
RTR	Remote transmission request (CANopen-specific frame that can trigger the sending/receiving of process data)
Scanner	EtherNet/IP™ master during process data communication
SDO	Service Data Object
Transmitter (producer)	EtherNet/IP™ Slave during SDO communication
Server	Modbus/TCP name for the (fieldbus) slave
(Fieldbus) slave	Fieldbus device that processes the individual subtasks
SYNC / SYNC-Interrupt	Signal in the fieldbus communication for synchronization of the partici-pants
TCP/IP	Group of network protocols for communication via IP addresses
VARAN	Real-time Ethernet bus system of the company SIGMATEK
XDD	POWERLINK device description file

Table 3-1: Used terms and abbreviations

3.3 Trademarks

CANopen® is registered trademark of CAN in AUTOMATION - International Users and Manufacturers Group e.V.

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

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

EtherNET/IP is a trademark of ODVA, Inc.

ETHERNET POWERLINK is a registered trademark of Bernecker + Rainer Industrie Elektronik Ges.m.b.H.

PROFINET® is a registered trademark of PROFIBUS user organisation e.V.

PROFIsafe® is a registered trademark of Siemens Aktiengesellschaft

VARAN-Bus is a registered trademark of Sigmatek GmbH &Co. KG.

Modbus/TCP is an open de facto standard for industrial communication

4 KEB operating tools

4.1 COMBIVIS studio 6

COMBIVIS studio 6, the KEB software tool based on CODESYS IEC 61131, enables wizard-guided component selection, fieldbus configuration, drive parameterization and IEC 61131-3 project generation.

4.2 COMBIVIS 6

License-free operating and analysis program for KEB COMBIVERT.

4.3 KEB Ftp Application

This tool runs on Windows XP or higher with the .NET Framework. It uses a COM port or the Windows IP stack to transfer files to and from KEB devices.

5 Instructions for the start-up of the fieldbus interfaces

The following chapter contains a short, general manual for the first start-up of the KEB fieldbus interfaces as well as a description and explanation of the KEB process data communication. More detailed information about the individual fieldbus systems can be found in the following chapters.

NOTICE

Devices of control type K do not support software version 3.x!

- For the devices of Generation 6 **Compact**, development was discontinued with software version 2.10. The functional status of this version is described for the fieldbus systems in the Programming Manual | Fieldbus Systems V 3.0. **This document is not to be used to describe the devices of control type K.**
- In case of necessary extensions / changes of the software for the devices of control type K, please contact your KEB sales partner for consultation.

5.1 Explanation of terms

5.1.1 KEB process data communication

To describe the KEB process data communication, the following chapters use a combination of terms from the CANopen DS301 nomenclature and the KEB nomenclature.

The KEB pr parameters (communication profile objects) are named according to the DS301 nomenclature. The process data communication of all fieldbus systems is mapped via these parameters. The following terms are used:

- Process data object (PDO), single object which can be read or written via the process data communication.
- Receive PDO (RPDO), PDO received from the inverter
- Transmit PDO (TPDO), PDO transmitted by the inverter
- Mapping, the mapping of the PDOs to the process data communication
- Communication parameters, CANopen specific parameters for setting the process data communication (=> chapter 6.8 CANopen process data objects (PDO))

In addition, KEB-specific nomenclature is also used in this document for the description of process data communication. This includes the following terms:

- Process input data (PD-In), PDO received from the control (TPDO)
- Process output data (PD-Out), PDO sent by the control (RPDO)

5.1.2 Baud rate and bit rate

The speed of data transmission via serial interfaces is referred to as the baud rate for KEB inverters. A baud here means the transfer of a drawing.
For the protocols supported by the serial interfaces of the KEB inverters one bit is required for each transmitted character. Baud rate and bit rate are therefore identical for the serial interfaces of the KEB inverters.
For this reason, baud rates are also specified in bit/s in this document.

5.2 Identify KEB communication interfaces

The F6 / S6 devices offer the following communication interfaces:

- One serial diagnostic interface for the connection with diagnostic tools (e.g. COMBIVIS) or an operator.
- One CANopen fieldbus interface
- One real-time Ethernet interface (2 Ethernet Ports)

A second serial interface is available on the devices of control type P in control board variant 3 instead of the second Ethernet port.

Depending on the control board version, various fieldbus systems are available on the real-time Ethernet interface:

- Control type A: EtherCAT, PROFINET, POWERLINK, EtherNet/IP, Modbus TCP
- Control type P: EtherCAT, PROFINET, Modbus TCP, Ethernet TCP/IP

The fieldbus systems Modbus ASCII and Modbus RTU are available on the second serial interface of control type P instead of EtherCAT and PROFINET.

The diagnostic interface is always available parallel to the fieldbus interface.
It can be changed between CAN interface and real-time Ethernet interface. If EtherCAT or PROFINET is active on the real-time Ethernet interface, the CAN interface can be operated in parallel to the real-time Ethernet interface.

NOTICE

- ➢ The diagnostic interface is not intended for permanent operation of the device.
- ➢ Parallel access (read or write) to the object dictionary via the diagnostic interface and the acyclic fieldbus channel can lead to errors in the acyclic fieldbus channel.
- ➢ In the device variant "F6 PRO" (device part numbers xxF6Pxx-xxxx), the CAN interface is not available for all hardware revisions. Notes on the support of the interface can be found in the associated installation manual in chapter "Fieldbus interfaces".

5.2.1 Serial diagnostic interface

The diagnostic interface is used to connect the F6 / S6 device with diagnostic tools, such as COMBIVIS. KEB DIN66019-II is used as protocol. This is an asynchronous serial protocol with the following key data:

1 start bit, 7 data bits, 1 parity bit (even), 1 stop bit.

These settings are preset and cannot be configured at the drive. A description of the protocol is available on the KEB website.

Info:

- The diagnostic interface supports point-to-point connections via RS232 or RS485 (full duplex). Networks via RS485 (half duplex) are not supported.
- The configuration of the diagnostic interface on the devices of control type P differs from the configuration on the devices of control type A and K.
- The term "node address" or "node ID" frequently used in this chapter refers to the identification number used by the DIN66019 protocol to address a KEB device.
- The DIN66019 node ID (fb13) and the node ID (fb64, fb100 - fb102), which can be set via the rotary coding switches and is used to identify the KEB device via the fieldbus interface, are not identical.

Configuration of the diagnostic interface on devices of control type P

The following objects can be used to configure the diagnostic interface on the devices of control type P:

Index	SubIdx	Id-Text	Name	Value	Function
0x2B0D	0	fb13	drive node ID	0-238	Node address of the inverter object directory (Default = 1)
0x2B0E	0	fb14	DIN66019 baud rate		Baud rate (Default = 38400 bit/s)
0x2B0F	1	fb15	application node ID	0-255	Node address of the application object directory (Default = 2)
0x2B0F	2	fb15	debugger node ID	0-255	Node address of the debugger object directory (Default = 255)
0x2B10	0	fb16	fieldbus node injection	0-255	Node address of the object dictionary where the Ethernet communication of the fieldbus interface is sent (Default = fb13)
0x200A	0	de10	operator cfg data		Structure with configuration data for F6 / S6 operators or for the application software and for connection to the file system

In the KEB DIN66019-II protocol, each participant is identified by its own node address. Each participant has a separate object directory.

A device of control type P can contain several object directories in different parts of the device. Therefore, a device of control type P also has several DIN66019 devices, which are identified by their own node address.

The object dictionary of the inverter is addressed via the node address and baud rate set in parameters fb13 and fb14.

Communication with other object directories on the device can be set via the subparameters under fb15 . The additional object directories are application- and/or customer-specific.

Telegrams that arrive at the device via the fieldbus interface are forwarded to the object dictionary of the inverter by default and after each reset.
If it is desired to address a different object directory with DIN66019 telegrams that are sent via the Ethernet channel of the fieldbus interface, the corresponding node address can be set via the parameter fb16 fieldbus node injection.
The DIN66019 telegrams that are transmitted via the Ethernet channel of the fieldbus interface are forwarded to the object dictionary, which is addressed via the node address in fb16.

Info:

- To enable automatic detection of replacement devices, for example, the default value of fb13 should not be used in productive systems.
- fb16 is a volatile parameter and is set to the value of fb13 with every reset. This means that the object directory of the inverter is automatically addressed after each reset.
- Information on the baud rate and the operator cfg data can be found in the Programming Manual | Control Application / Compact / Pro

Configuration of the diagnostic interface on devices of control type A

The following objects can be used to configure the diagnostic interface on the devices of control type A and K:

Index	SubIdx	Id-Text	Name	Value	Function
0x2B0D	0	fb13	DIN66019 node id	0-238	Node address of the inverter object directory (Default = 1)
0x2B0E	0	fb14	DIN66019 baud rate		Baud rate (Default = 38400 bit/s)
0x200A	0	de10	operator cfg data		Structure with configuration data for F6 / S6 operators or for the application software and for connection to the file system

In the KEB DIN66019-II protocol, each participant is identified by its own node address. Each participant has a separate object directory.
There is only one object directory on the devices of control type A and K. Therefore there is one DIN66019-II participant and one variable node address per device.

Info:

- To enable automatic detection of replacement devices, for example, the default value of fb13 should not be used in productive systems.
- Information on the baud rate and the operator cfg data can be found in the Programming Manual | Control Application / Compact / Pro

5.2.2 Second serial interface

A second serial interface is available on the devices of control type P in control board variant 3, which can be used by means of to communicate using the protocols Modbus ASCII or Modbus RTU.
If these protocols shall not be used, the second serial interface can also be used as additional diagnostic interface.
The baud rate of the second serial interface is set via parameter fb116. Baud rates between 1200 baud and 500 kbaud can be set. The default baud rate is set to 38400 bit/s. The baud rate corresponds to 1 bit/s.

Index	Id-Text	Name	Function
0x2B74	fb116	Baud rate of the 2nd serial interface	Setting of the baud rate

NOTICE
Only the baud rates 4800 baud, 19200 baud and 38400 baud were tested for the Modbus field bus system.
Other baud rates are technically possible but are not recommended for Modbus.
Sollte Modbus mit den nicht empfohlenen Baudraten betrieben werden kann es dazu kommen, dass zum Beispiel die Modbus-RTU Pausenzeiten nicht eingehalten werden.

5.2.3 Real-time Ethernet interface

Two Ethernet ports (X4C-RJ45 sockets) for Ethernet communication are available on all devices of control type A and the standard devices of control type P. An Ethernet port is available on control board variant 3 of control type P.

Communication via the active real-time Ethernet fieldbus system (fb68) or Ethernet TCP/IP (control type P) is possible via the existing Ethernet ports

Multi-Realtime-Ethernet module on the real-time Ethernet interface

On the devices of control type A, the fieldbus systems of the real-time Ethernet interface are realised with a multi-real-time Ethernet module (MRTE module). The functions of the control board are independent of this MRTE module.

Parameter fb80 is used to display information about the Multi-Realtime Ethernet module in the COMBIVIS interface.

Parameter fb80 is an invisible parameter and is not displayed in the COMBIVIS interface with the default settings. To display the invisible parameters in COMBIVIS the KEB parameterisation must be changed under Tools -> Options.

Index	Id-Text	Name	SubIdx	Name-SubIdx	Function
0x2B50	fb80	MRTE module	0		Number of sub-indices
			1	MRTE module support	Support of the MRTE module (MRTE module required / MRTE module not required)
			2	MRTE module state	State of the MRTE module (ready / not found)
			3	MRTE module version	Version of the MRTE module (NetX50 / NetX51 / not found)

Cards of control type A can also be operated without the MRTE module if no Ethernet based fieldbus system is used.

In order to avoid an error message on a device without MRTE module it is necessary to set parameter fb80 Sub-Index 1 MRTE module support from "MRTE module required“ to "MRTE module not required“.

Ethernet TCP/IP on the real-time Ethernet interface

On the devices of control type P it is possible to use the real-time Ethernet interface as standard TCP/IP interface. This is only possible if no fieldbus system is connected.

The real-time Ethernet interface can be changed to TCP/IP interface via parameter fb71. EtherCAT or CANopen must be set as active fieldbus system in fb68.

It is not necessary to restart the device to switch between EtherCAT and Ethernet TCP/IP.

Settings in fb71 only have a function if EtherCAT or CANopen is selected as active fieldbus system in fb68.

fb71	fieldbus options				**0x2B47**
Bit	Function	Val-ue	Plaintext	Notice	
0...1	Mode	0	automatic EtherCAT / Ethernet	The device switches automatically between EtherCAT and Ethernet TCP/IP.	
		1	EtherCAT	No Ethernet only EtherCAT	
		2	Ethernet	No EtherCAT only Ethernet TCP/IP	
		3	reserved	Not yet used	

5.3 Adjust fieldbus parameter

To set the fieldbus parameters, it is necessary either to establish a connection between the device and the KEB COMBIVIS operating tool or to program the device via the fieldbus control.

A COMBIVIS connection can be established via the diagnostic interface, EoE, the Profinet Ethernet channel or a KEB control with a suitable gateway program.

When using the fieldbus control, pay attention to the fieldbus-specific feedback for parameter requests.

When establishing a connection via the real-time Ethernet interface, make sure that EtherCAT (standard) or CANopen (control board variant 3 of control type P) is set by default.

See also:

- Chapter 7.5 Ethernet over EtherCAT (EoE)
- Chapter 9.3.3 Additional diagnostic channel via standard Ethernet communication
- Chapter 15.3 Fieldbus specific feedback for parameter requets

5.3.1 Selection of the fieldbus system via fb68

When selecting the fieldbus system, please note that

- the new fieldbus system only becomes active after a restart
- cards of control type A require time to switch between two Ethernet-based fieldbus systems after the device is restarted.
- the error code "ERROR fieldbus type changed" in ru01 indicates that the parameter fb68 has changed since the last restart.
- a change of the active fieldbus system will reset the process data mapping.

fb68	fieldbus selection	0x2B44
Value	Name	Notice
0	EtherCAT	Fieldbus communication occurs via EtherCAT (Default) Not for type P variant with additional RS485 interface (type 3)
1	CANopen	Fieldbus communication occurs via CANopen (default x6P type 3)
2	VARAN	Fieldbus communication occurs via VARAN Requires VARAN hardware version of control type K
3	PROFINET	Fieldbus communication via PROFINET Requires control type A or P (not type 3)
4	POWERLINK	Fieldbus communication via POWERLINK Requires control type A
5	Reserved	-
6	EtherNet/IP	Fieldbus communication occurs via EtherNet/IP Requires control type A
7	Mod	Fieldbus communication occurs via Modbus TCP Requires special software for control type A or any variant of control type P
8	Modb	Fieldbus communication occurs via Modbus ASCII Requires variant of type P with additional RS485 interface (type 3)
9	Mo	Fieldbus communication occurs via Modbus RTU Requires variant of type P with additional RS485 interface (type 3)
32	EtherCAT + cross communication	Fieldbus communication occurs via EtherCAT and Fieldbus communication occurs via CANopen
35	PROFINET + cross communication	Fieldbus communication occurs via PROFINET and Fieldbus communication occurs via CANopen

The active fieldbus system and the application software module (control type P)

Application software modules are developed by regional subsidiaries and KEB Service to realise the requirements of specific customers.
An application software module is loaded from the internal memory during start-up of the device and serves as interface between a device of control type P and the used communication interface.

If fb68 is changed, fb81.1 is set to the appropriate default value on a restart. Resetting to default values is **not** carried out if there is no default string in fb81.1. (Start default strings with X6P_FC_RD_)

If a specific fieldbus system is supported by a customer-specific application software module must be clarified with the relevant contact person.

Independent of the loaded application software module, the required fieldbus system must be selected in fb68.

Parameter fb81: Application software module can be used to check the name of the loaded / to be loaded application software module.

fb81	Application software module			**0x2B51**
SubIdx	Type	Access	Name	Notice
0	Uint8	RO		Number of subindices
1	String	RW	Target application software	Name of the application software that will be loaded at the next restart
2	String	RO	Active application software	Name of the currently active application software or "Default code" in error case

The name of the application software module also contains the software version. The names of the default application software modules are structured as follows:

X6P_FC_RD_[MAIN][FB][TAG]_[DATE]_[TIME]

Scheme	Digit	Meaning
[MAIN]	AABB	AA = main version and BB = sub version
[FB]	CC	00 = EtherCAT, CAN, Modbus / 01 = PROFINET
[DAY]	DD	Day of the respective version
[DATE]	JJJJMMDD	JJJJ = year, MM = month and DD = Date of creation
[TIME]	HHMM	HH = hour and = minute of the creation time

NOTICE

Changes to parameter fb81 can cause that the device is no longer accessible via the fieldbus interface!

- Changes to parameter fb81 should only be made in consultation with your KEB sales partner.
- With a "Factory Reset" via co08 and co09, the parameter fb81.1 is also reset to its default value. This can lead to the wrong application software module being loaded after a "factory reset".
- In error case, the string "Default Code" is displayed in fb81.2
- In error case, you can still communicate with the device via the diagnostic interface
- If you are interested in a customer-specific application software module, please contact your KEB sales partner.

5.3.2 Selection of the process data size via fb60

After setting the required fieldbus system via fb68 fieldbus selection the required maximum process data size can be set via parameter fb60 process data size selection.

Parameter fb60 process data size selection influences the length and the maximum number of process data objects of the first process data mapping. Additionally, fb60 influences the minimum adjustable fieldbus cycle time.

The possible process data variables differ depending on the selected control type (A, K or P). Currently, different process data sizes are only supported for the feldbus systems EtherCAT (ECAT) and PROFINET (PNET) and Modbus (MB)..

If the active fieldbus system is changed via fb68 fieldbus selection, fb60 process data size selection is reset to the default value.

NOTICE
For an optimal time behaviour of the software it is recommended to choose the process data size selected by fb60 as small as possible. If the KEB device is particularly high utilised by other functions (encoder, operating mode, CAN-Cross, etc.), a runtime error (ru01: 121 "Error! Runtime") can occur if the process data size is large. With Modbus as an active field bus system, runtime errors can occur at maximum process data size even with medium utilisation of the KEB device.

If a process data size is selected via fb60 process data size selection, this influences the configuration of the active fieldbus system, which is visible via fb67.

A change of the process data size becomes active only after a restart of the software. The error code "ERROR fieldbus type changed" indicates in ru01 whether the parameter fb60 has changed since the last restart.

fb60 can also be changed if an active mapping is already set. However, the size of the mapping (number of PDOs and length) must be permissible for the newly selected process data size.

By writing on fb60, the minimum SYNC interval set via is22 Basic Tp can be influenced. The value of is22 is not changed by writing on fb60.

SYNC intervals adjustable on KEB devices correspond to the minimum SYNC interval adjustable via is22 or a full multiple thereof.
A minimum value of the SYNC interval is preset by fb60, which cannot be fallen below independently of is22.
The adjustable cycle time still depends on is22.

The minimum cycle time of 250 µs for EtherCAT can be selected via the plain text, but is currently not functional. The minimum cycle time of EtherCAT can be found in the EtherCAT device description file (ESI).

The minimum value of the SYNC interval is determined for PROFINET by the device description file (GSDML) and is independent of fb60.

Further information on is22 and the minimum SYNC interval can be found in chapter 5.7 Synchronisation.

fb60	process data size selection			**0x2B3C**
Bit	Function	Value	Plaintext	Notice
0…7	mode	0	8 objects, 32 byte, 400µs (ECAT) / 1ms (PNET) min cycle time	Default, active for all fieldbuses (1ms restriction for PROFINET)
		1	8 objects, 32 byte, 250µs (ECAT) min cycle time	Control type P, **currently not supported**

		2	-	Reserved for 125µs
		3	16 objects, 32 byte, 500µs (ECAT) / 1ms (PNET) min cycle time	Only EtherCAT, PROFINET & Modbus
		4	32 objects, 64 byte, 1ms (ECAT / PNET) min cycle time	Only EtherCAT, PROFINET & Modbus

5.3.3 Checking the configuration of the active fieldbus system

After setting the required process data size via fb60 process data size selection, the configuration of the selected fielbus system can be checked via parameter fb67 fieldbus configuration.

fb67 is only a display parameter and cannot be written by the user.

Bit 0 indicates whether a synchronous mode is supported for the selected fieldbus system settings.

If Bit 1 is set the KEB default process data mapping is loaded after each restart in the first mapping object. (aa16 to aa23 or aa16 and aa17 for CANopen)

Bit 2 is not yet supported.

If Bit 3 is set, the set mapping is deleted after a restart. The effect of bit 1 overwrites the effect of bit 3**.** If both bits are set, the device starts with the KEB default process data mapping.

Bit 4 indicates whether process data and acyclic data are processed in the same interrupt or in several interrupts in succession. The separate processing of process data and acyclic data avoids internal runtime problems.

Bit 5 indicates whether all process data are processed in the same interrupt or several interrupts in succession. If this bit is active the time required by the KEB device to process the process data completely increases significantly.The separate processing of the process data avoids internal runtime problems.

fb67	fieldbus configuration			**0x2B43**
Bit	Function	Value	Plaintext	Notice
0	Synchronous mode	0	No SyncMode support	No synchronous mode possible
		1	SyncMode support	Synchronous mode possible
1	Enable exemplary PD mapping at startup	0	No Exemplary PDMapp	No change to the mapping
		2	Exemplary PdMapp	A sample mapping is loaded during start-up
2	Enable application specific alarm messages	0	No ApplAlarm	Standard alarm messages
		4	ApplAlarm	Not yet supported
3	Disable non-volatile mapping parameter	0	Non volatile Mapp	Mapping is retained on reset
		8	Volatile Mapp	Mapping is deleted by reset
4	PDO / SDO multiplexing	0	No PDO / SDO multiplexing	PDOs, SDOs in the same Mid IRQ
		16	PDO / SDO multiplexing	One Mid IRQ PDOs, in the next SDOs
5	PDO multiplexing	0	No PDO multiplexing	All PDOs in the same Mid IRQ
		32	Multiplexing per 16 PDOs	16 PDOs per Mid IRQ

5.3.4 Selection of the node address of the fieldbus system

In order to communicate with a KEB device via the fieldbus interface, it is necessary to assign an identification number to the KEB device. This identification number is named as node address or node ID.
The node ID of the fieldbus systems and the DIN66019 node ID are **different** identification numbers.

The CAN node ID

For communication via CANopen, each KEB device has a unique CAN address, the CAN node ID.

After delivery, KEB devices have the CAN node address "1".
If several KEB devices shall be networked via CANopen different CAN addresses must first be assigned.

Index	Id-Text	Name	Function
0x2B40	fb64	CAN node ID	Setting the CANopen node address value

In addition to the CAN node ID, it is also necessary to set the CAN baud rate to establish communication via CANopen. This is described in chapter 6.10 CAN bit timing and baud rate.

The node ID specification via the rotary coding switches

For devices of control type A or P, the node address can be set via the rotary coding switches at the device or via parameter fb101.

The associated effective node address value (fb102) is only updated when the device is restarted. Without a restart, changes to the rotary coding switch or fb101 have no effect.

The effective node address value (fb102) is accepted in fb64 CAN node ID when the device is restarted, if it corresponds to a value valid for CAN (1 - 127) and CANopen is the active fieldbus system.

The effective node address value (fb102) affects the base IP configuration used by PROFINET, POWERLINK, EtherNet/IP, Modbus / TCP and the Ethernet channel.

The effective node address value (fb102) affects the storage of the PROFINET device name. fb101 is furthermore coupled to the writing of the PROFINET device name. (See PROFINET device name)

Index	Id-Text	Name	Function
0x2B64	fb100	node ID switch value	Indicates the read node address value of the rotary coding switches.
0x2B65	fb101	adjusted node ID value	Parameter for the alternative setting of the node address.
0x2B66	fb102	effective node ID	Active node address value in the device. Corresponds to fb100 as long as fb101 = 0. Otherwise equivalent to fb101. Updated when the device is restarted.

5.4 Restart the device

Many communication parameters (parameters of the fb and pr groups) are only activated after a restart.

To carry out a restart, either parameter co09 "reset ctrl" can be set or the device is restarted via the 24V supply voltage.

Restarting the software interrupts communication to the device for a short period.

Index	Id-Text	Name	Function
0x2509	co09	reset ctrl	Trigger a software restart by writing the value 1. (Writing of value 0 has no effect)

The device can be reset to factory setting by restarting the software. This requires additional settings in parameter co08 .

Resetting to the factory settings is not considered in these instructions. Resetting to factory settings does not affect communication parameters.

NOTICE
After a restart, the new selected Ethernet-based fieldbus stack is copied on the devices of control type A. While copying is in progress the fieldbus STATUS LED (NET ST) and the control STATUS Led (DEV ST) will flash yellow.The selected fieldbus system can only be used after copying has been completed.

5.5 Check the fieldbus status

To determine whether the fieldbus system is in the required state, the parameters presented in this chapter can be checked. Such a check is recommended after the active fieldbus system has been changed and when fieldbus communication is started up for the first time.

5.5.1 Checking of MAC addresses and IP configuration

MAC addresses and IP configurations are required for the Ethernet based fieldbus systems PROFINET, POWERLINK and EtherNet/IP as well as the EtherCAT functionality EoE. These are indicated via the parameters fb103 to fb106 as well as fb108 and fb109.

Depending on the used control type and the active fieldbus system the function and support of these parameters differs significantly.

Control type	Fieldbus system	fb103: FB MAC Address (Base)	fb104: PNet MAC Address (Port 1)	fb105: PNet MAC Address (Port 2) / MAC Address (EoE Channel)	fb106: MAC Address (EthChannel)	fb108: Ethernet over fieldbus IP configuration	fb109: Basic IP configuration
S6A / F6A	EtherCAT	-	-	-	x	x	-
	PROFINET	x	x	x	x	x	x
	POWERLINK	x	-	-	-	-	x
	EtherNet/IP	x	-	-	x	x	x
	Mod	x	-	-	x	x	x
S6P / F6P	EtherCAT	-	-	x	-	x	-
	Ethernet TCP/IP	-	-	-	x	-	x
	PROFINET	-	-	-	x	x	x
	Mod	-	-	-	x	-	x

Table 5-1: Differences in the support of MAC / IP parameters

Checking the MAC addresses

The MAC addresses fb103 – fb105 on the devices of control type A are set to a fixed value in the production process by a protected mechanism.

The MAC address fb106: MAC Address (EthChannel) on the devices of control type A is a virtual MAC address which displays the EoE MAC address, the MAC address of the PROFINET Ethernet channel and the MAC address of the EtherNet/IP Ethernet channel.

The virtual MAC address is set on the devices of control type A to a default value intended for PROFINET. To the value of fb103: FB MAC Address (Base) + 3. An EoE capable EtherCAT master can overwrite this value. The value written by the EtherCAT master is only stored on the device until the next restart.

On the devices of control type P, the virtual EoE MAC address is displayed via parameter fb105: MAC Address (EoE Channel) . The EoE MAC address behaves as on the devices of control type A and K.

On the devices of the control type P a MAC address is displayed in fb106, which is set to a fixed value in the production process by a protected mechanism. Depending on the active fieldbus system this MAC address is used for PROFINET or Ethernet TCP / IP.

PROFINET requires additional MAC addresses for port 1 and 2. These are generated from the MAC address displayed in fb106. The U/L bit of the manufacturer identification (OUI) is used for this. Setting this bit turns the universal MAC address into a MAC address that is only valid locally. The following table gives an overview of the MAC addresses generated for PROFINET.

	OUI

	Byte0								Byte1	Byte2
Basic MAC	0	0	0	0	0	0	0	0	08h	FAh
Port 1 MAC	0	0	0	0	0	0	1	0	08h	FAh
Port 2 MAC	0	0	0	0	0	1	1	0	08h	FAh
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	U/L	I/G		

Table 5-2: Generation of the additional PROFINET MAC addresses on the P cards

Further information on the use of MAC addresses as well as the virtual MAC address can be found in the corresponding chapters of the respective fieldbus systems.

Index	Id-Text	Name	Function
0x2B67	fb103	Fb MAC Address (Base)	MAC address for Ethernet-based fieldbus systems (A)
0x2B68	fb104	PNet MAC Address (Port1)	PROFINET MAC Address for Port1
0x2B69	fb105	PNet MAC Address (Port2) / MAC Address (EoE Channel)	PROFINET MAC Address for Port2 (A) / volatile EoE MAC address (P)
0x2B6A	fb106	MAC Address (EthChannel)	MAC address for all Ethernet channels (EoE (A), PROFINET Ethernet channel (A), Ethernet TCP / IP, PROFINET and Modbus TCP (P))

Checking the IP configuration for the fieldbus Ethernet channel

The IP configuration for the fieldbus Ethernet channel is displayed via parameter fb108 Ethernet over fieldbus IP configuration.

Index	Id-Text	Name	SubIdx	Name-SubIdx	Function
0x2B6C	fb108	Ethernet over fieldbus IP configuration	0		Number of sub-indices
			1	IP address	IP address
			2	subnet mask	Subnet mask
			3	gateway address	Gateway address

The default value of fb108 depends on the active fieldbus system. The following default values apply to fb108:

Name	PROFINET	Other fieldbus systems
IP address	192.168.0.200	192.168.1.100
subnet mask	255.255.255.0	255.255.255.0
gateway address	0.0.0.0	0.0.0.0

ATTENTION
Make sure that the selected IP addresses in fb108 and fb109 are **not identical** . Otherwise there may be problems with the fieldbus communication. This also applies if fb108 has no function for the active fieldbus system. (see below)

Various functions of fb108
The function of fb108 depends on the active fieldbus system. For some fieldbus systems, fb108 is influenced by the effective node address(fb102) and/or the active base IP configuration(fb109). The function and influences on fb108 are described below.

Ethernet over EtherCAT (EoE)
If EoE is active, parameter fb108 is set by the EtherCAT master. It is recommended **not** to modify fb108 via COMBIVIS. Changes to fb108 via COMBIVIS do not affect the active EoE IP configuration. They only become active after a restart. Der EtherCAT Master kann Änderungen durch COMBIVIS überschrieben.

PROFINET Ethernet channel
If PROFINET is active on the devices of control type A, parameter fb108 displays the IP configuration for the parallel Ethernet channel. The default value of the IP configuration

for the parallel Ethernet channel corresponds to the default value of the PROFINET basic IP configuration (fb109) increased by 100. (fb109: 192.168.0.100 leads to fb108: 192.168.0.200)

If PROFINET is active, parameter fb108 can change its value when the device is restarted. This depends on the effective node address value (fb102).

- If the effective node address value is 254, fb108 is reset to the default value.
- If the effective node address value is 241 to 243, fb108 is set to 0.
- For other values of the effective node address, fb108 is not changed. The value set by COMBIVIS or the PROIFNET host before the restart remains kept.

If PROFINET is active, parameter fb108 can change its value by writing to fb109. This depends on the value in fb108.

- If fb108 corresponds to the default value (192.168.0.200; 255.255.255.0; 0), then fb108 is set to fb109 when writing to fb109, with an offset of 100 on the IP address.
- If fb108 does not correspond to the default value, fb108 is not affected by writing to fb109.

Example:
fb108 corresponds to the default value. (192.168.0.200; 255.255.255.0; 0)
fb109 is written to (192.168.100.5; 255.255.252.0; 0).
fb108 is automatically set to (192.168.100.105; 255.255.252.0; 0).
With a new write access to fb109, fb108 does not change, because fb108 does not correspond to the default value.

When accessing via the COMBIVIS parameters, fb108 is not updated until the gateway address of fb109 is written.

On the devices of control type P PROFINET does not require a second IP configuration to receive parallel Ethernet telegrams. Therefore, the IP configuration shown in fb109 and fb108 is identical.
The PROFINET IP configuration is saved on the file system on the devices of control type P. The PROFINET IP configuration is therefore not affected by resetting to factory settings. (PnetDevRemData.bin)

EtherNet/IP Ethernet diagnostic channel
If EtherNet/IP is active, parameter fb108 displays the IP configuration for the Ethernet diagnostic channel. The value of the IP configuration for the EtherNet/IP diagnostic channel corresponds to the value of the EtherNet/IP basic IP configuration (fb109) increased by a subnet. (e.g.: fb109: 192.168.0.100 leads to fb108: 192.168.1.100)

Modbus TCP diagnostic channel on the devices of control type A
If Modbus/TCP TCP is active, parameter fb108 indicates the IP configuration of the Ethernet diagnostic channel. It can only be changed via COMBIVIS. This only applies to devices of control type A.

CANopen, VARAN, POWERLINK and Modbus on devices of control type P
Parameter fb108 has no function

Checking the basic IP configuration

The basic IP configuration for the Ethernet-based fieldbus systems PROFINET, POWERLINK, EtherNet/IP and Modbus/TCP is displayed on the devices of control type A via parameter fb109.

On the devices of the control type P the IP configuration for PROFINET, Modbus TCP and the Ethernet channel is displayed via parameter fb109. The Ethernet channel automatically adopts a new IP configuration

Index	Id-Text	Name	SubIdx	Name-SubIdx	Function
0x2B6D	fb109	Basic IP configuration	0		Number of sub-indices
			1	IP address	IP address
			2	subnet mask	Subnet mask
			3	gateway address	Gateway address

The default value of fb109 depends on the active fieldbus system. The following default values apply to fb109:

Name	PROFINET, POWERLINK, EtherNet/IP, ModbusTCP, basic Ethernet	EtherCAT, CANopen, VARAN
IP address	192.168.0.100	0.0.0.0
subnet mask	255.255.255.0	0.0.0.0
gateway address	0.0.0.0	0.0.0.0

The value of the basic IP configuration is changed on the devices of control type A and P depending on the effective node address value (fb102) after restarting the device. How the IP configuration is affected depends on the fieldbus system selected in fb68 fieldbus selection.

The node address values 0, 240 and 255 for a POWERLINK device are invalid values. Connection with the device via POWERLINK is then **not** possible.

Modbus/TCP only displays the current IP configuration in fb109 if the IP configuration "static" is selected. Parameter fb114 is only available for Modbus TCP on the devices of control type A

Fieldbus system	IP configuration method (fb113 / fb114)	Effective node address value (fb102)	IP configuration	Changeable via COMBIVIS
PROFINET		0 – 240, 244 – 253, 255	Is not affected	Yes, active after reset
		241 - 243	0, 0, 0	Yes, active after reset
		254	192.168.0.100 255.255.255.0 0	No
POWERLINK		1 – 239, 241 - 254	192.168.100.Node address value 255.255.255.0 192.168.100.254	Only the gateway address, active after reset
Ethernet/IP, Modbs TCP (A)	Bootp or DHCP(default)	0	Is not affected	Value is specified by the fieldbus system
	Static	1 - 255	192.168.0.Node address value 255.255.255.0 0	No

5.5.2 Checking the fieldbus STATUS LED (NET ST)

The status of the fieldbus can be determined with the STATUS LED (NET ST). It is a two-colour LED with the basic colours green and red. If no error has occurred, the LED should display the flashing pattern "Operational".

In the case of parallel fieldbus communication, the light pattern of the Ethernet-based fieldbus system is always displayed.

A detailed description of the status LEDs can be found in the Programming Manual | Control Application / Compact / Pro.

Fieldbus system	Flashing	Color	State
EtherCAT	Continuous flashing	Green	Pre-Operational
	Flickering	Green	Boot
	Single lightning	Green	Safe Operational
	On	Green	OPERATIONAL
	Continuous flashing	Red	Error in the configuration
	Flickering	Red	Error during boot
	Single lightning	Red	Error during status change
	Double lightning	Red	Watchdog error
CANopen	Continuous flashing	Green	Pre-Operational
	Single lightning	Green	Stopped
	On	Green	Operational
	Flickering	Green	Error in the initialization
	Continuous flashing, double flash	Green, red	Error in Pre-Operational
	Single flash, double flash	Green, red	Error: CANopen stopped
	Single flash, on	Red, green	Error in Operational
VARAN	Continuous flashing	Red	Initialization
	On	Green	Operational
PROFINET	Continuous flashing	Green	PROFINET LED flashing mode
POWERLINK	Single lightning	Green	Pre-Operational 1
	Double lightning	Green	Pre-Operational 2
	Triple lightning	Green	Ready for Operational
	On	Green	Operational
	Continuous flashing	Green	Stopped
	Flickering	Green	Ethernet mode
	On	Red	Error
EtherNet/IP	Continuous flashing	Green	No connection
	Continuous flashing	Red	Timeout
	On	Green	connected
	On	Red	Double IP
	Continuous flashing	Green, red	Self test
Mod	Continuous flashing	Green	Waiting for connection
	On	Green	connected
All fieldbus systems	Off	-	Fieldbus system not active

Flashing	Description
Off	Sensor is constantly off
On	Sensor is constantly on
Continuous flashing	Cyclic change with 200ms on/200ms off
Single lightning	Cyclic change with 200ms on/1000ms off
Double light-ning	Cyclic change with 200ms on/200ms off/200ms on/1000ms off
Triple lightning	Cyclic change with 200ms on/200ms off/200ms on/200ms off/200ms on/1000ms off
Flickering	Cyclic change with 50ms on/50ms off

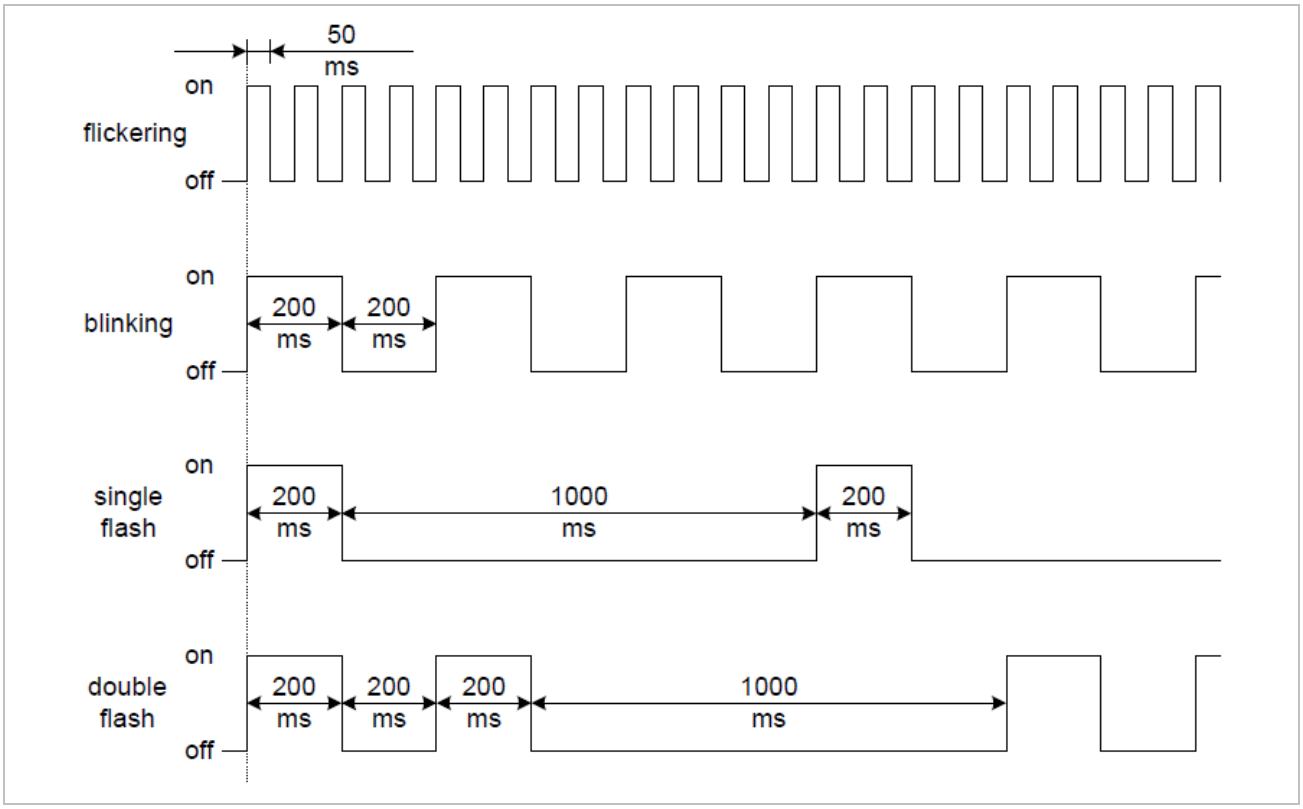


Figure 1: Timing of the LED flashing pattern

5.5.3 Checking the fieldbus status and possible errors

If it is not possible to connect to the device via the selected fieldbus system, there may be various reasons for this. In such a case, it is recommended to check the fieldbus status (fb90), the fieldbus error code (fb91) as well as the exception status (ru01) and the associated error code (st01) using the diagnostic interface.

Please note that the Modbus fieldbus system is not listed under fb90, fb91 because this fieldbus system does not support a communication status machine. It should also be noted that in the case of parallel fieldbus communication, status and error codes of all active fieldbus systems are displayed in their respective subindexes.

The fieldbus status and the fieldbus error code of all inactive fieldbus systems corresponds to "0xFFFF: Fieldbus inactive". If no error has occurred, the fieldbus error code typically corresponds to "0x0000". The fieldbus error codes (fb91) and the general error codes (st01) are independent of each other. Information on fieldbus-specific error codes can be found in the documentation of the respective fieldbus system.

This document only considers the ru01 errors that can occur when there are errors in the fieldbus module. A detailed description of the parameter ru01 and a list of all error codes can be found in the Programming Manual | Control Application / Compact / Pro.

A list of possible fieldbus statuses and fieldbus error codes as well as possible solutions for some KEB-specific fieldbus error codes can be found in the appendix.

Parameters for checking the fieldbus errors

Index	Id-Text	Name	Function
0x2C01	ru01	exception state	Display of an occurred error as plaintext
0x2101	st01	error code	Display of an occurred error as error code
0x2B5A	fb90	fieldbus state	Array with the active status of the available fieldbus systems
0x2B5B	fb91	fieldbus error code	Array with the active error codes of the available fieldbus systems

Fieldbus system dependent errors in ru01 and st01

ru01	Error text	Description	st01
24	ERROR fieldbus memory	Incorrect drive software configuration	8: Fault
58	ERROR fieldbus watchdog	Timeout of the fieldbus watchdog	8: Fault
124	General Fieldbus Error	General fieldbus error, see fb91 for detailed description	8: Fault
125	ERROR fieldbus type changed	Fieldbus system changed, restart required	8: Fault

fb90 and fb91

Index	SubIdx	Id-Text	Name	Function
0x2B5A	0	fb90	fieldbus state (number)	Number of available fieldbus systems
0x2B5A	1	fb90	EtherCAT fieldbus state	Status display of the EtherCAT fieldbus system (register AL status)
0x2B5A	2	fb90	CANopen fieldbus state	Status display of the CANopen fieldbus system (Register CO_RunState)
0x2B5A	3	fb90	VARAN fieldbus state	Status display of the VARAN fieldbus system (only control type K)
0x2B5A	3	fb90	PROFINET fieldbus state	Status display of the PROFINET state machine (only control type A and P)

Index	SubIdx	Id-Text	Name	Function
0x2B5A	4	fb90	POWERLINK fieldbus state	Status display of the POWERLINK status machine (control type A only)
0x2B5A	5	fb90	EtherNet/IP fieldbus state	Status display of the EtherNet/IP state machine (only control type A)
0x2B5A	6 (x6A) / 4 (x6P)	fb90	ModbusTCP fieldbus state	Status display of the ModbusTCP communication status
0x2B5B	0	fb91	fieldbus error code (number)	Number of available fieldbus systems
0x2B5B	1	fb91	EtherCAT fieldbus error code	Error code of the EtherCAT fieldbus system (Register AL Status Code)
0x2B5B	2	fb91	CANopen fieldbus error code	Error-Code of the CANopen fieldbus system (Register EmergencyErrField)
0x2B5B	3	fb91	VARAN fieldbus error code	Error code of the VARAN fieldbus system (only control type K)
0x2B5B	3	fb91	PROFINET fieldbus error code	Error code of the PROFINET fieldbus system (only control type A and P)
0x2B5B	4	fb91	POWERLINK fieldbus error code	Error code of the POWERLINK fieldbus system (only control type A)
0x2B5B	5	fb91	Ethernet/IP fieldbus error code	Error code of the EtherNet/IP fieldbus system (only control type A)

5.6 Checking the process data mapping

5.6.1 Basic structure of process data mapping

KEB devices support up to four process data mappings for process output data and process input data. The mapping of the PDOs is based on the CANopen DS301 standard.

The available number of process data mappings, the length of these mappings and the maximum number of PDOs per mapping is depending on the used fieldbus system.

Further information can be found in the fieldbus-specific chapters of this manual.

CANopen

CANopen uses four process data mappings. Each mapping supports up to 8 process data objects. The maximum length of a CANopen process data mapping is 8 bytes.

EtherCAT

By default, EtherCAT supports up to 8 EtherCAT and 8 FSoE PDOs in one mapping. A maximum of 64 bytes are available for EtherCAT. For FSoE, the number of bytes is specified by the mapping to be selected. The PDOs are divided KEB internally to the first and second process data mapping. The EtherCAT objects are always mapped to the first process data mapping. The FSoE objects are always mapped to the second process data mapping.

The size of the EtherCAT process data mapping is variable. Both the length and the maximum number of PDOs can be changed via parameter fb60: process data size selection. Depending on the set size of the EtherCAT process data mapping the minimum possible EtherCAT cycle time changes.

The FSoE process data mapping is not influenced by fb60.

PROFINET

The PROFINET mapping objects are mapped to the first process data mapping. The size of the PROFINET process data mapping is variable. Both the length and the maximum number of PDOs can be changed via parameter fb60: process data size selection.

PROFIsafe process data are handled similarly to FSoE process data via the second process data mapping.

VARAN, POWERLINK, EtherNet/IP and Modbus

VARAN, POWERLINK, EtherNet/IP use the first process data mapping. Up to 8 PDOs are supported. The maximum length of a process data mapping is 32 bytes.

Modbus

Modbus uses the first process data mapping. The size of the PROFINET process data mapping is variable. Both the length and the maximum number of PDOs can be changed via parameter fb60: process data size selection.

Parallel fieldbus communication

Control type A and P devices with software version 3.1 and higher support parallel operation of CANopen and EtherCAT or PROFINET.
In this case, only 2 process data mappings with up to 8 bytes and 8 objects are available for CANopen.
The process data size of EtherCAT or PROFINET is not restricted by parallel fieldbus operation.

5.6.2 Mapping of the process data

The mapping of the process data for all fieldbuses occurs CANopen-conform via the mapping objects 0x1600 - 0x1603 (process output data) and 0x1A00 - 0x1A03 (process input data).

A KEB mapping object can contain up to 8 parameters and up to 32 bytes by default. A KEB mapping object can contain a maximum of 32 parameters and 64 bytes. To enable smaller SYNC intervals, a KEB mapping object can also contain only up to 8 objects and 16 bytes.

Mapping parameters **cannot** be changed as long as the mapping is active. The mapping is active as long as the subindex 0 of the respective mapping object corresponds to a value unequal zero. To set the process data mapping it is recommended to use the COMBIVIS Fieldbus Wizard.

Fieldbus-specific process data mappings

EtherCAT and PROFINET support one mapping object each for unsafe process output data (0x1600) and unsafe process input data (0x1A00)) as well as one mapping object each for safe (FSoE / PROFIsafe)) process output data (0x1601) and safe process input data (0x1A01).

CANopen supports all KEB mapping objects. However, a CANopen mapping object can only contain up to 8 bytes.

VARAN, POWERLINK and EtherNet/IP each support only one mapping object for process output data (0x1600) and one mapping object for process input data (0x1A00).

The different Modbus variants use their own process data addressing. Further information on this can be found in the chapter Modbus interface.

The KEB default process data mapping

Depending on the control type and the active fieldbus system, there are different KEB default process data mappings. These are listed in the appendix.

Changes of the process data mapping when changing the fieldbus system

In order to avoid invalid process data settings, the mapping of the process data is deleted when the fieldbus system is changed and the process data are deactivated. The following special cases must be observed:

- When changing to EtherCAT on the devices of control type A, user parameters: aa16 – aa23 are loaded into the first mapping object and activated.
- For PROFINET, the device must be restarted again after the required process data has been set.
- For the different Modbus implementations, the number of registers to be set depends on the set mapping.

Index	SubIdx	Id-Text	Name	Function
0x1600	0	-	1st receive PDO mapping (process output data)	Number of the mapped objects
	1…8			Mapping of the process data object
0x1601	0	-	2nd receive PDO mapping (CANopen/safe PD)	Number of the mapped objects
	1…8			Mapping of the process data object
0x1602	0	-	3rd receive PDO mapping (only CANopen)	Number of the mapped objects
	1…8			Mapping of the process data object
0x1603	0	-	4th receive PDO mapping	Number of the mapped objects

	1…8		(only CANopen)	Mapping of the process data object

Index	SubIdx	Id-Text	Name	Function
0x1A00	0	-	1st transmit PDO mapping (process input data)	Number of mapped objects
	1…8			Mapping of the process data object
0x1A01	0	-	2nd transmit PDO mapping (CANopen/safe PD)	Number of mapped objects
	1…8			Mapping of the process data object
0x1A02	0	-	3rd transmit PDO mapping (only CANopen)	Number of the mapped objects
	1…8			Mapping of the process data object
0x1A03	0	-	4th transmit PDO mapping (only CANopen)	Number of the mapped objects
	1…8			Mapping of the process data object

The description of the mapped objects occurs in the following format:

Bit	Meaning
0…7	Object length in bits (8, 16 or 32)
8…15	Subindex of the mapped object
16…31	Index of the mapped object

5.6.3 Parameters for the process data mapping of different fieldbus systems

Depending on the active fieldbus system there are additional parameters that are required for process data mapping. These are described in the respective chapters of the fieldbus systems.

5.6.4 Feldbus Wizard

The COMBIVIS 6 fieldbus wizard provides a graphical representation of the process data mapping. It is possible to set the process data mapping via the fieldbus wizard, but the specifications of the active fieldbus system must be observed.

The following boundary conditions are automatically considered by the fieldbus wizard:

- Data type (the complete value of the mapping parameter is calculated depending on the data type).
- Properties (depending on the object properties (RO,RW, mapping allowed), the mapping is allowed or disabled)

The wizard displays how many PDOs are available, how many parameters can be used per PDO and how many bytes are available per PDO.

It is expected that from version 6.7.0 it will also be possible to set the required process data size via the wizard.

The wizard can also be used to load examples of process data mapping.

Depending on the used fieldbus system, the device description files corresponding to the fieldbus system can also be exported. The following files can be created by the wizard:

- ESI files for EtherCAT
- EDS files for CANopen
- XDD files for POWERLINK

No description files can be exported for VARAN, PROFINET, EtherNet/IP and different Modbus variants.

The fieldbus wizard cannot be used for parallel fieldbus communication (CANopen + EtherCAT / PROFINET).

Further information on the fieldbus wizard can be found in the COMBIVIS manual.

5.7 Synchronisation

To ensure that the process data is processed every fieldbus cycle, KEB devices can be synchronised to the fieldbus cycle using an internal PLL. It is essential that the fieldbus system provides a SYNC signal. Synchronisation to the fieldbus cycle occurs automatically as soon as a SYNC signal is available. How this SYNC signal is realised depends on the fieldbus system.

The CANopen fieldbus system must be considered separately here, since this fieldbus system can support synchronous and asynchronous process data **simultaneously**.

The synchronous/asynchronous operating mode is supported depending on the active fieldbus system and the selected control type. The following table gives an overview in which combinations which mode is available.

Control type	Fieldbus system	Synchronous mode	Asynchronous mode
S6A / F6A	EtherCAT	X	X
	CANopen	~	~
	PROFINET	X	X
	POWERLINK	X	-
	EtherNet/IP	-	X
	Mod	-	X
S6P / F6P	EtherCAT	X	X
	CANopen	~	~
	PROFINET	-	X
	Modbus	-	X

Table 5-3: Support of the synchronous / asynchronous mode

5.7.1 Check synchronisation

Checking via the statusword

In case that a fieldbus system provides an interrupt for synchronisation, the KEB device will automatically synchronise to this interrupt.
Depending on the accuracy of the interrupt and the selected synchronisation interval, additional settings may be necessary.

With the synchron bits in st00: statusword it is possible to detect whether the KEB device was able to synchronize to the specified synchronization interval.
If the bit is set, the synchronisation was successful.
If the bit is not set, the KEB device could not synchronise to the specified sync interrupt.

If the synchronous bit in st00: statusword is set, the application is in synchronous mode and the process data are processed synchronously.statusword is set, the application is in synchronous mode and the process data is processed synchronously.

st00	statusword	**0x2100**
Bit	Name	Notice
8	synchron	1 = Drive control synchronised to the fieldbus

Preset and measured synchronisation interval

KEB devices differentiate between an internal synchronisation interval (fb10: sync interval) and an external synchronisation interval (fb19: measurend sync interval).

fb10 is used as a template for the internal cycle of the KEB device. As soon as fb10 is set to a value unequal to 0, the internal synchronisation PLL is activated and synchronous operation is active.

fb10 is set for EtherCAT, PROFINET and POWERLINK once and automatically after power-on to the specified synchronisation interval of the fieldbus master.

With CANopen and VARAN fb10 is set to fb19 once after power-on.

The measured distance between the last two received synchronisation signals is displayed in fb19.

EtherNet/IP, Modbus and PROFINET on the devices of control type P do not support synchronous application. In these cases,fb10 and fb19 have no function.

Index	Id-Text	Name	Function
0x2B0A	fb10	sync interval	Activation of the synchronous operating mode Specification for the cycle time (400µs - 16000µs)
0x2B13	fb19	measured sync interval	Measured cycle time of the active fieldbus system

The differentiation between internal and external synchronisation interval is necessary since the external sync signal not always trigger exactly at the same interval.
In order to compensate this jitter of the external sync signal, the internal synchronisation interval is used as basis for the cyclical program sequence of the KEB device.

The synchronisation PLL attempts to synchronise the internal and external synchronisation interval by way that there is no more than the time interval defined in fb11: set sync interval between the internal and external sync signal.
If this is the case, the synchronous bit of st00: statusword is set.

If the PLL cannot synchronise to the external signal, the synchron bit in st00 st00 is not set.

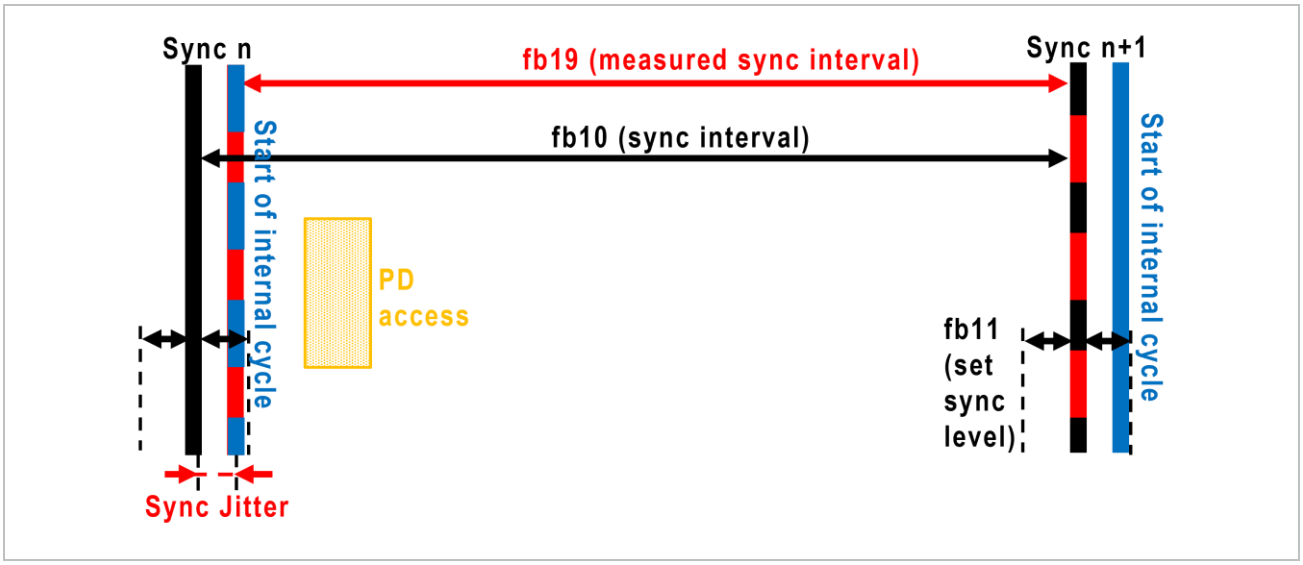


Figure 2: Relation between internal and external SYNC

Special features of different fieldbus systems

- With EtherCAT, the synchronisation interval can also be read out via the Sync Manager parameters.
- With EtherCAT, unsupported SYNC interval values are acknowledged with an error code. This is displayed in parameter fb91 with the error code ECAT_AL_STATUS_CODE_INVALID_SYNC_CYCLE_TIME.
- Depending on the active fieldbus system, there may also be a minimum sync interval that does not correspond to the value in is22. This is described in the chapter of the respective fieldbus system in this manual.
- Additionally to the display in fb10, the internal synchronisation interval is displayed in the pr parameter group under parameter interpolation time period. The synchronization interval is divided into a value (interpolation time value) and an exponent (interpolation time index). Writing to interpolaton time period affects the value in fb10 and vice versa

5.7.2 Synchronisation PLL

Each inverter has a PLL that automatically synchronises with the fieldbus master. As soon as the sync interval is set by the master, all controller algorithms are also synchronised to the sync signal via the PLL.

As soon as the sync interval of the master is set after a power-on under fb10, all controller algorithms are also synchronised to this PLL. Sollte es der Synchronisations-PLL nicht gelingt sich auf das externe Signal zu synchronisieren kann dies verschiedene Gründe haben:

KEB devices cannot support any SYNC intervals.

The supported SYNC intervals depend on the MidIrq grid set in is22: Basic Tp. Only whole multiples of this value (up to a limit of 16ms) are supported. If the external sync interval does not correspond to a multiple of the MidIrq grid, the KEB device will not synchronise to the external interval.

Index	Id-Text	Name	Function
0x3516	is22	Basic Tp	Selection of the switching frequency group

The parameters for setting the synchronisation PLL are not set to the jitter of the external sync signal

Parameter fb11: set sync level lässt sich die Toleranz der Synchronisations-PLL für das externen Sync-Signal einstellen.
The higher this value, the greater the jitter of the external sync signal can be. However, the time difference between the external and internal synchronisation interval also increases.

Parameter fb12: KP sync PLL lässt sich der KP-Wert der PLL beeinflussen.
If the value of fb12 is increased, the internal PLL can adapt more quickly to the jitter of the external sync signal. Thereby, this also increases the volatility of the internal synchronisation interval. (not recommended)

Index	Id-Text	Name	Function
0x2B0B	fb11	set sync level	Tolerance value between measured and set SYNC interval
0x2B0C	fb12	KP sync PLL	KP value of the controller of the SYNC-PLL

Recording of the external sync signal jitter for PLL error diagnosis

If there are problems with the PLL, it is recommended to make an offline recording of fb19 with the COMBIVIS Scope.

When analysing such a recording, the following should be noted:

- The mean value of fb19 should match with the value of fb10. The following applies:
  - For x6A fb10 +/- 0.24%. (Example: 4000us +/- 9.6us)
  - Für x6P fb10 +/- 1.6% (Example: 4000us +/-64us)

- Adjustments to fb11 are best suited to solve problems with the synchronisation. However, fb11 should not become higher than the difference between the minimum and maximum value of fb19 in the recording.
- The gain of the PLL can be influenced with fb12 KP sync PLL and thus the speed reaction of SYNC changes. You can also make an offline recording with aa85, st00 and fb10. The offline measurement is started by setting a value in fb10, so you can see the transient behaviour of the PLL. At the point where fb10 is preset, aa85 will

change to a maximum value until the SYNC bit is set in st00. This behaviour can now be changed with fb12 until the required process is achieved.

- If the required behaviour still does not occur after the above-mentioned adjustments, we recommend contacting KEB Service. An offline recording is often very helpful for the first error diagnosis by the service centre.
- The changed values of fb11 and fb12 only have an effect when fb10 is written again.

A recording with the COMBIVIS Scope could look like this:

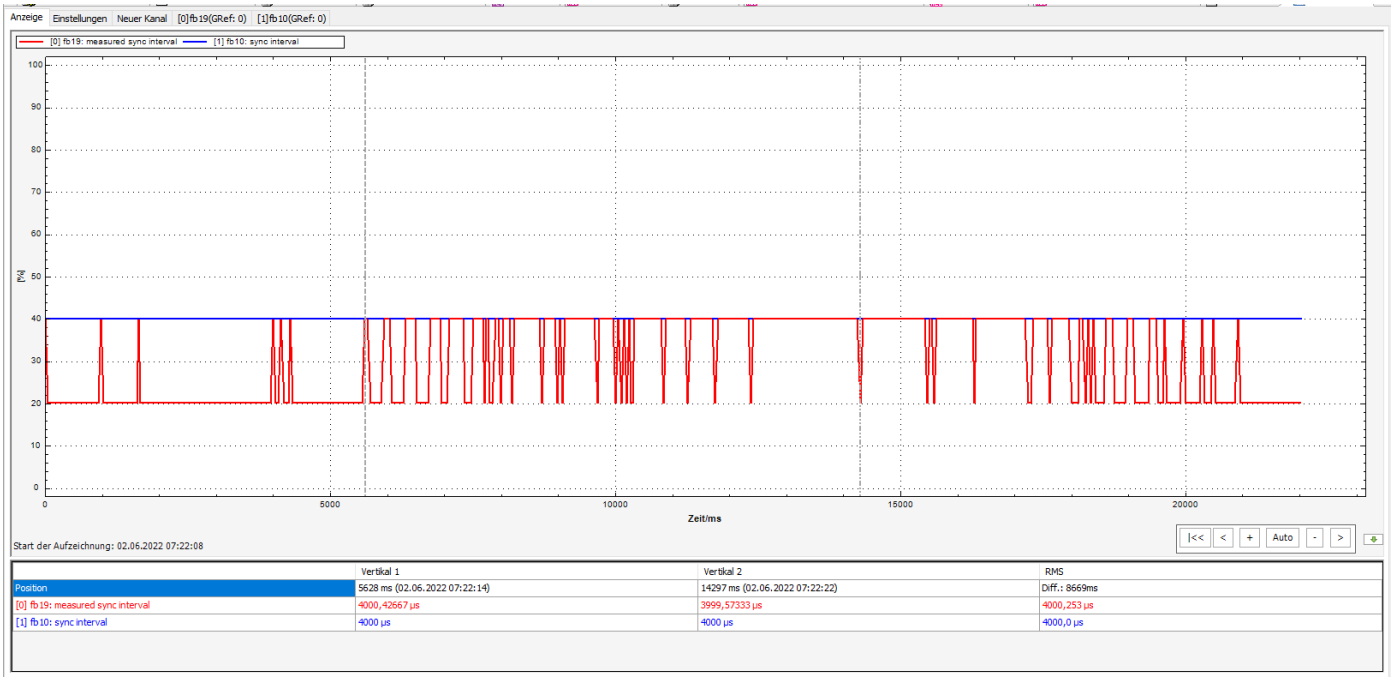


Figure 3: COMBIVIS Scope recording of fb19

5.7.3 Synchronous transfer of process data

As soon as the synchron bit of the st00: statusword is set, the KEB device is in synchronous mode and process data is transferred synchronously.

As soon as this is the case, the processing of the process data is time-dependent on the SYNC interrupt of the active field bus system.

During a synchronous fieldbus cycle, the process data are read out once from the device / written to the bus, updated in the object directory and activated in the controller.

The synchronisation intervals in which the KEB device has not received any process data are counted in fb31. fb31 is reset with each restart of the device.

Index	Id-Text	Name	Function
0x2B1F	fb31	no PDO data per sync cnt	Number of synchronization intervals wherein no process data was received

The minimum time between the external sync signal and the start/end of process data access by the KEB device is displayed in fb37 and fb38.

Index	Id-Text	Name	Function
0x2B25	fb37	process data access min. sync delay	The minimum interval between the SYNC interrupt and the start of process data access.
0x2B26	fb38	process data access max. sync delay	The maximum distance between the SYNC interrupt and the end of the process data access.

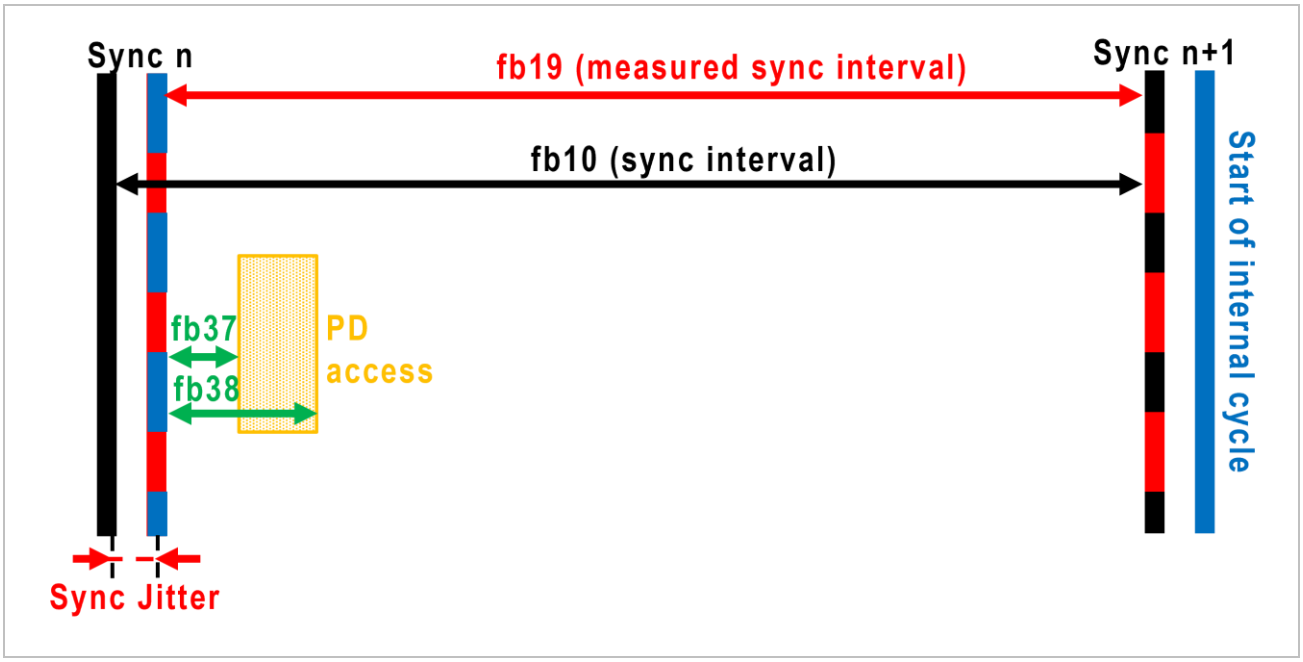


Figure 4: Process data access of the device

NOTICE

The correct time to access the process data is essential for the functionality in synchronous operation!

- The external sync signal should be received before the process data are accessed, even with maximum jitter.
  fb37: process data access min. sync delay can be used for checking.
- Access to the process data by the fieldbus master and KEB device must not overlap, otherwise incorrect values may occur in the process data objects. Access of the KEB devices to the process data is fixed, so this must be ensured by settings in the fieldbus master.

NOTICE

The KEB internal interpolation of process data values only works if the process data are received once per SYNC cycle!

- x process data per SYNC cycle or receiving process data only every xth SYNC cycle is not supported by KEB devices.
  If process data are received in this way, the interpolation cannot be carried out correctly and there will be jumps in the value of the process data.
  This malfunction is not visible at fieldbus level.

- During the initialisation phase of most fieldbus systems, synchronisation interrupts are sent, although process data communication has not yet started. After initialization it is common that the value of fb31 is higher than 0. It is significant for synchronous communication that fb31 does not increase during operation.
- The fieldbus systems Modbus, Ethernet/IP and PROFINET on the devices of control type P cannot be operated synchronously and therefore cannot accept their process data synchronously.
- The field bus system CANopen can transmit synchronous and asynchronous process data simultaneously.
- The time window for accessing the PD telegram is displayed graphically in the COMBIVIS fieldbus wizard for EtherCAT on the devices of control type P.

5.7.4 Asynchronous transfer of process data

In asynchronous operation, the process data are processed as fast as possible. The processing time of the process data is then only dependent on the set Mid-Irq grid (is22) and the cycle time set in the fieldbus master.

In asynchronous operation, it is not necessary to pay attention to the time of the process data transfer or the internal synchronisation PLL.

Asynchronous operation is supported by all fieldbus systems except POWERLINK and VARAN.

5.7.5 Special case CANopen: Synchronous and asynchronous process data simultaneously

CANopen does not differentiate between synchronous or asynchronous operation, but between synchronous or asynchronous process data.

KEB devices support up to 4 CANopen process data mappings per direction.
Each of these mappings can contain either synchronous or asynchronous process data, depending on the subparameter transmission type of parameters 1st – 4th RPDO communication parameter and 1st – 4th TPDO communication parameter in the pr group.

Due to this variance, there is not just one time for accessing the process data with CANopen. Parameters fb37 and fb38 have **no** function if CANopen is the active fieldbus system.

Instead of fb37 and fb38, with CANopen it must be ensured that the SYNC telegram is not delayed by other telegrams on the bus. To avoid this, the CAN system integrator should ensure that a sufficiently high CAN baud rate (fb66) and sufficiently long cycle time are set for the utilisation of the system.

KEB devices do not provide any parameters that can be used to check the utilisation of the CAN bus.

5.8 Fieldbus watchdog

With the fieldbus watchdog function, the drive can switch to a defined state in case of process data communication loss.

The fieldbus watchdog is activated with the first received process data telegram.
If the fieldbus watchdog is active, a process data telegram must be received within the watchdog time specified in pn21 otherwise the selected event in parameter pn22 is triggered and the corresponding bit in the warning status (ru02) is set.
Apart from the special cases described below, the telegrams received must be process data telegrams. **The fieldbus watchdog currently only monitors the receipt of process data.**
The sources for the fieldbus watchdog that can be set in pn22 have no effect on the watchdog behaviour, with the exception of one special case. (s. special cases)
The fieldbus watchdog is not available in jog mode.If the watchdog time specified in pn21 is 0, the watchdog is triggered immediately after receiving the first process data telegram.

Special cases:
- On the devices of control type A and CANopen, the watchdog reacts to all received telegrams.
- On the devices of control type P and modeTCP, ModbusASCII or ModbusRTU, various sources that the watchdog monitors can be defined via pn22. (see chapter 5.8.1)

Index	Id-Text	Name	Function
0x2A15	pn21	fieldbus watchdog time	Time in milliseconds until the watchdog is triggered
0x2A16	pn22	E.fb watchdog stop mode	Error response
0x2C02	ru02	Warning bits	Display of warnings bit-coded
0x2A1C	pn28	Warning mask	Mask for warning bit in the status word

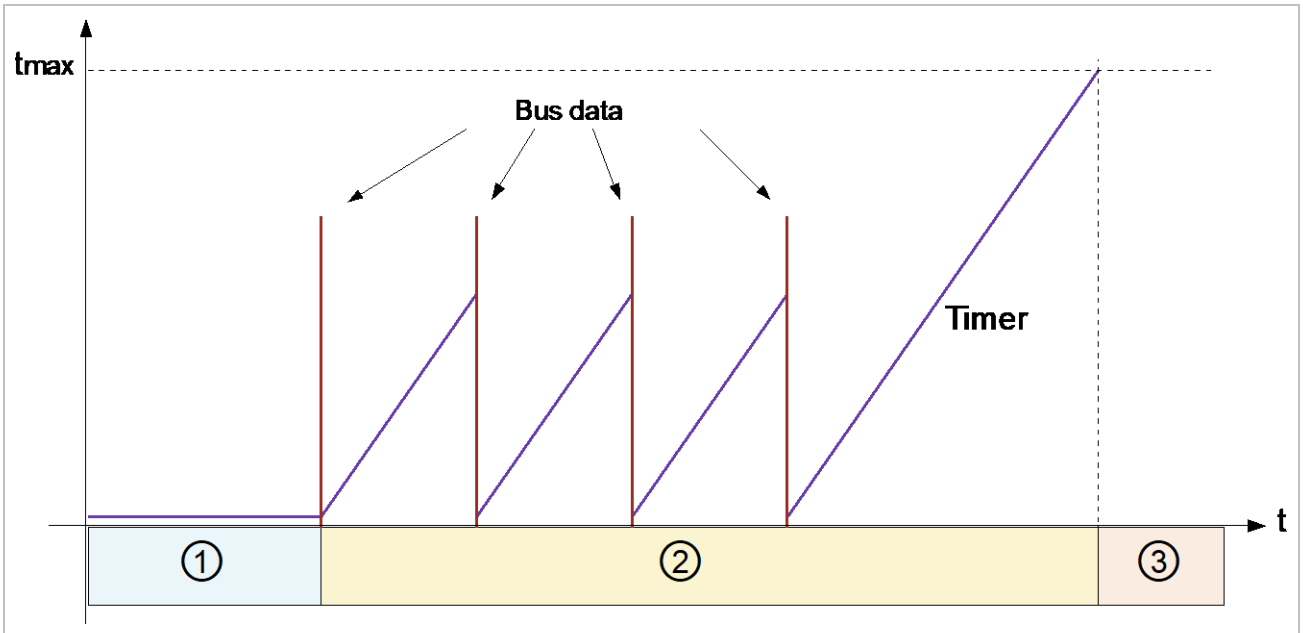


Figure 5: Fieldbus watchdog

tmax	Triggering level
①	Waiting for activation
②	Communication
③	Error

5.8.1 Supplement to the standardisation of the fieldbus watchdog

The behaviour of the fieldbus watchdog can be set via parameter pn22. Various functions of the watchdog can be defined via the different bits of the pn22.
Bits 0 to 3 determine the error behaviour of the watchdog. This is identical to the settings for other pn parameters.
Bits 5 and 6 determine the sources or events to which the watchdog reacts. Bits 5 and 6 are currently only supported by the Modbus field bus on control type P devices. These bits have no function for all other fieldbus systems.

Index	Id-Text	Name	Bit	Function
0x2A16	pn22	E.fb watch-dog stop mode	0-3	Error response of the watchdog Details can be found in the "Errors and warnings" section of the programming manual | Application / Compact / Pro control unit.
			5-6	Sources to which the watchdog reacts (Modbus only): 0: Receiving PDOs 32: Receiving and transmitting PDOs 64: Receiving and transmitting PDOs 96: Receiving and transmitting PDOs and SDOs

5.8.2 Influence of the fieldbus status change to the fieldbus watchdog

The fieldbus status (fb90) can be changed during operation in some fieldbus systems. This usually also stops the process data communication. To prevent the fieldbus watch-dog from being triggered incorrectly in such a case, it is dependent on the fieldbus status.

Active fieldbus system	State transition	Watchdog
EtherCAT	To Operational	Watchdog can be activated
	From OP to other status without error code	Watchdog deactivated
POWERLINK	To Operational	Watchdog can be activated
	From OP to PreOp2 / Stopped	Watchdog deactivated
Modbus	-	Watchdog can always be activated
Others	To Operational / communication is running	Watchdog can be activated

5.8.3 Manual triggering of the fieldbus watchdog

It is possible to trigger the fieldbus watchdog manually on devices of control type P. To do this, value -1 must be written in parameter pn21. The watchdog error is reset by writing value -2.

Writing -1 or -2 does not affect the adjusted watchdog time. After resetting the watchdog, it can be reactivated by receiving a process data telegram.

Manually setting/resetting of the fieldbus watchdog is independent of the status of the de-vice and does not affect any other existing errors.

This function is **not** available if the manufacturer-specific CiA402 operation mode Jog Mode and / or PROFINET are active.

Attention This also influences the values for manually triggering or resetting the watch-dog. With the current standardisation (1/4), the value for triggering the watchdog corre-sponds to -0.25, the value for resetting the watchdog corresponds to -0.5.

pn21	fieldbus watchdog time		0x2A15
Value	Standardised value	Name	Notice
0 - 16000	0.00 – 4000.00ms		Time until the watchdog is triggered
-1	-0,25	Trigger watchdog manually	Manual triggering of the watchdog
-2	-0,5	Reset watchdog manually	Manual reset of the watchdog

6 CANopen interface

6.1 Basics of the CAN BUS

Here we like to introduce the system of the CAN (Controller-Area-Network) BUS and also explain some terms that are frequently used in the following.

The CAN is a multi master system, i.e. each user has access to the BUS and can send telegrams.

To ensure that no invalid states arise when multiple participants access at the same time, the telegrams on the CAN bus are prioritised according to the lowest telegram number (identifier). The participant with the lowest telegram number can send his telegram, while all other participants (who are willing to send) change into receive status and initially cancel sending their telegram. As soon as a telegram has been sent, this process is repeated.

CAN telegrams can contain a maximum of 8 bytes of user data.

The term logical CAN master used in the following, refers to the CAN participant, who is responsible for the control of the entire CAN system. Even if, there are only masters for CAN, in most applications there will often be a participant who takes over control of the system.
KEB devices are to be seen as command receivers (logical slave). When using parallel fieldbus communication, KEB devices can take the function of the logical CAN master.

KEB devices realise the minimum capability device defined by the CiA DS301 standard.

Info:
On KEB F6/S6 devices, the CANopen specification is the basis for all bus functionalities. The object dictionary as well as the mapping functionality are CANopen conform. The basis for the implementation of other bus systems is always CANopen.

6.2 Identifier according to CiA DS301

According to CiA DS301, the sync identifier of the device and information on the SYNC is specified in parameter cob-ID sync message at address 0x1005.

KEB devices support an 11-bit identifier. This is constant and cannot be changed. With the exception of parallel fieldbus communication, KEB devices cannot generate a SYNC.

cob-ID sync message	Var		0x1005
Bit	Value	Function	
0…10	128	Standard SYNC Identifier (non-adjustable)	
11…28	0	0	
29	0	supports only value 0: 11-Bit CAN-ID	
30	0	Device does not generate a SYNC message (for value 1 see chapter 14)	
31	0	Reserved	

Additionally to the SYNC identifier, other broadcast and communication identifiers are defined in the CiA DS301 standard. The identifiers supported by KEB devices are shown in the following tables:

Identifier	Broadcast object
0 (000h)	Network Management Telegram (NMT)
128 (080h)	Synchronisation telegram (SYNC)

from identifier	to identifier	Communication object
129 (081h)	255 (0FFh)	Emergency telegram (EMCY)
385 (181h)	511 (1FFh)	Transmit/send process data object 1 (TPDO1)
513 (201h)	639 (27Fh)	Receive process data object 1 (RPDO1)
641 (281h)	767 (2FFh)	TPDO2
769 (301h)	895 (37Fh)	RPDO2
897 (381h)	1023 (3FFh)	TPDO3
1025 (401h)	1151 (47Fh)	RPDO3
1153 (481h)	1279 (4FFh)	TPDO4
1281 (501h)	1407 (57Fh)	RPDO4
1409 (581h)	1535 (5FFh)	Service Data Object (SDO (tx))
1537 (601h)	1663 (67Fh)	Service Data Object request (SDO (rx))
1793 (701h)	1919 (77Fh)	NMT Error Control / Boot-Up / Node-Guarding / Heartbeat

6.3 CANopen Node ID

Wie bereits in vorherigen Kapiteln beschrieben, definiert CAN verschiedene Identifier, um die verschiedenen CAN-Telegramtypen zu identifizieren. Um neben dem Telegramtyp auch den Empfänger oder Sender eines Telegrams zu identifizieren wird jedem CANopen Gerät ein netzwerkweit eindeutiger Identifier (Node ID) zugewiesen.

Telegrams with broadcast identifiers are given the CAN node ID as an argument. For telegrams with communication identifiers, the CAN node ID is linked to the identifier via addition. No CAN node ID is transferred to SYNC telegrams, because they are intended for all participants in the network.

***Example**: TPDO1 from CAN device with CAN Node ID 15h = 180h + 15h = 195h*

On KEB devices, the CAN node ID can take values between 1 and 127.

The CAN node ID can be preset via the rotary coding switches of the device, via parameter fb101 adjusted node ID value or parameter fb64 CAN node ID.

If parameter fb64 CAN node ID is changed, the new value is saved non-volatile and becomes active immediately. Changes to parameter fb101 or the rotary coding switch are only activated after restarting the software.

Index	Id-Text	Name	Function
0x2B40	fb64	CAN node ID	Setting the CANopen node address value

6.4 CANopen Service Data Object (Request/Response Identifier)

In CAN, the term Service Data Object (SDO) means the mechanism of requests and responses. This is also known as confirmed service.

The logical CAN master can request (read) or change (write) the value of a parameter via the SDO telegrams. In the communication profile a write-service is referred to as domain download and a read service as domain upload.

To clearly identify SDOs, the previously listed request and response identifiers are used.

Over the request-identifier any CAN node can request the reading or writing of a parameter value. The response identifier is reserved for the corresponding response of the KEB device.

Process data object	Name of the CAN identifier	Value of the CAN identifier
SDO (receive)	Request-Identifier	600h (1536) + CAN node ID of the receiver (fb64)
SDO (transmit)	Response-Identifier	580h (1408) + CAN node ID of the transmitter (fb64)

The KEB-CAN interface only supports the short form of the SDOs. Only one telegram is exchanged for the request and another for the response.

With an SDO - read access, the length of the addressed parameter and the parameter value are returned. Every readable parameter in the KEB device can be requested via read access.

With an SDO - write access always 4 bytes are transferred as parameter value. For parameters with smaller length the remaining bytes are set to 0. In these cases, pay attention to the position of the parameter value within the telegram. Each writeable parameter in the KEB device can be changed within its permitted limits via write access.

The addressing of the parameter is done via unsigned 16 bit index and unsigned 8 bit subindex. The CAN index is identical to the KEB parameter address.

The subindex serves as additional addressing for complex parameters. The following applies:

Subindex	Type	Access to
0	Variable	Parameter value
	Field/structure	Subindex 0 (number)
1…n	Variable	not possible
	Field/structure	Subindex 1…n; Multiple selection not possible

Info:
The SDO channel is continuous operated independent of the process data transfer. However, the processing time of the requests depends on the utilisation of the device

6.4.1 SDO telegram structure

The structure of the SDO requests and responses is described bit by bit in the following. The following applies to write requests from the master and read responses from the device:

- If the bit (s) is 1, then the bits (nn) have a meaning.
- If the bit (s) is 0, then the bits (nn) have a meaning.
- If the bits (nn) are 0, then there are 4 bytes of data in the telegram.
- If the bits (nn) are 1, then there are 3 bytes of data in the telegram.
- If the bits (nn) are 2, then there are 2 bytes of data in the telegram.
- If the bits (nn) are 3, then there are 1 bytes of data in the telegram.

Initiate Domain Download Request (write request of the master)

Bit	7…0	7…0	15…8	7…0	7…0	15…8	23…16	31…24
	0010nn1s	Index		Sub-Index	Data			
Byte	0	LB 1	HB 2	3	LSB 4	5	6	MSB 7

Initiate domain upload request (read request of the master)

Bit	7…0	7…0	15…8	7…0	7…0	15…8	23…16	31…24
	01000000	Index		Sub-Index	reserved			
Byte	0	LB 1	HB 2	3	LSB 4	5	6	MSB 7

Initiate Domain Download Response (Write confirmation from KEB device)

Bit	7…0	7…0	15…8	7…0	7…0	15…8	23…16	31…24
	01100000	Index		Sub-Index	reserved			
Byte	0	LB 1	HB 2	3	LSB 4	5	6	MSB 7

Initiate Domain Upload Response (Read confirmation from KEB device)

Bit	7…0	7…0	15…8	7…0	7…0	15…8	23…16	31…24
	0100nn1s	Index		Sub-Index	Data			
Byte	0	LB 1	HB 2	3	LSB 4	5	6	MSB 7

6.4.2 SDO error response

Abort Domain Transfer (Error response from the KEB device)

Bit	7…0	7…0	15…8	7…0	7…0	15…8	7…0	7…0
	10000000	Index		Sub-Index	Additional code		Error code	Error class
Byte	0	LB 1	HB 2	3	LB 4	HB 5	6	7

Error class	Error code	Additional code	Meaning	
00	00	0000h	OK, no error	
05	04	0001h	Service not supported	
06	01	0000h	Invalid operation	
06	01	0002h	Attempt to write to a read-only parameter	
06	01	0010h	Invalid password	
06	02	0000h	Invalid address	
06	09	0011h	Subindex does not exist	
06	09	0012h	Invalid language identifier	
06	09	0030h	Invalid value for this parameter	
08	00	0020h	Data cannot be transferred / saved	
08	00	0022h	Device busy	

6.5 CANopen process data objects (PDO)

In CANopen, the term process data object (PDO) means the mechanism of unaddressed and unacknowledged communication.

At PDOs, a differentiation is made between the two data directions Out (master to slave) and In (slave to master).

The data exchanged via PDOs corresponds to the read / write value of one or more KEB parameters. The parameters belonging to this is determined via the process data image.

Whether a KEB parameter can be exchanged via PDOs is defined in the parameter property "Available for process data". The parameter properties can be checked via the property editor in COMBIVIS.

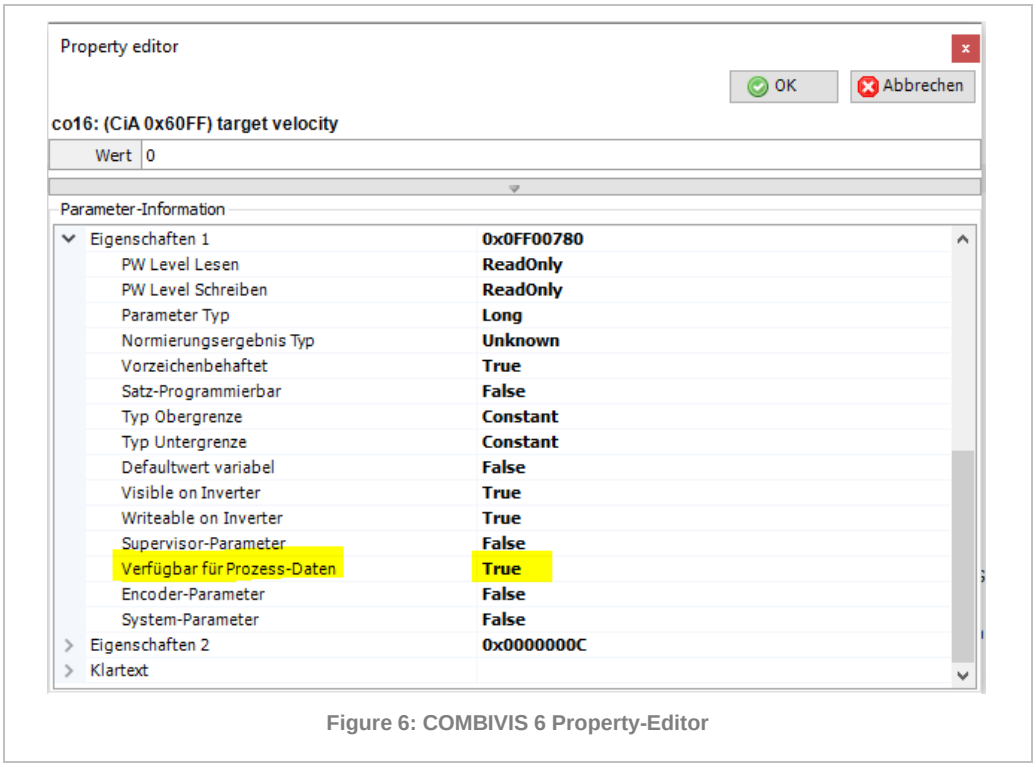


Figure 6: COMBIVIS 6 Property-Editor

6.5.1 Process data objects and process data addressing

4 process data objects and 4 process data mappings are available for each data direction on the devices of control type A and P. Up to 8 KEB parameters with a total of 8 bytes can be defined per mapping.

Address	Parameter	Address	Parameter
0x1400	1st RPDO communication parameter	0x1800	1st TPDO communication parameter
0x1401	2nd RPDO communication parameter	0x1801	2nd TPDO communication parameter
0x1402	3rd RPDO communication parameter	0x1802	3rd TPDO communication parameter
0x1403	4th RPDO communication parameter	0x1803	4th TPDO communication parameter
0x1600	1st receive PDO mapping	0x1A00	1st transmit PDO mapping
0x1601	2nd receive PDO mapping	0x1A01	2nd transmit PDO mapping
0x1602	3rd receive PDO mapping	0x1A02	3rd transmit PDO mapping
0x1603	4th receive PDO mapping	0x1A03	4th transmit PDO mapping

The various process data objects are addressed via the CAN identifiers.
Both the CAN node ID of the out identifier and the CAN node ID of the in identifier correspond to the CAN node ID of the slave.

Process data object	Name of the CAN identifier	Value of the CAN identifier
PDO1(receive)	Out-Identifier1	200h + CAN Node ID (fb64)
PDO1(transmit)	In-Identifier1	180h + CAN Node ID (fb64)
PDO2(receive)	Out-Identifier2	300h + CAN Node ID (fb64)
PDO2(transmit)	In-Identifier2	280h + CAN Node ID (fb64)
PDO3(receive)	Out-Identifier3	400h + CAN Node ID (fb64)
PDO3 (transmit)	In-Identifier3	380h + CAN Node ID (fb64)
PDO4(receive)	Out-Identifier4	500h + CAN Node ID (fb64)
PDO4(transmit)	In-Identifier4	480h + CAN Node ID (fb64)

6.5.2 Figure of the process data to be received (receive PDO)

The identification and receipt behaviour of the process data to be received (also PD-Out or RPDO) is defined via the RPDO communication parameters of the respective PDO.

A change of the communication parameters is immediately activated and saved non-volatile.

Index	Type	Id-Text	Name		
140xh	RECORD		RPDO communication parameter		
SubIdx	Type	Name			
1	UINT32	cob-ID	Bit	Value	Function
			0..10		11-bit CAN-ID, identifier of the PDO, not writable Corresponds to Out-Identifier 1-4 (writable and variable at parallel field bus communication)
			11..28		29-Bit CAN-ID (is not supported)
			29	0	Length of the PDO identifier (11-Bit)
			30	x	reserved
			31	0	Processing of process output data activated.
				1	Processing of process output data switched off.
2	UINT8	transmission type		0.. 240	When a SYNC command is received, the current process output data are taken over.
				254	RPDOs are taken over when changes are made (asynchro-nous, manufacturer-specific)
				255	(asynchronous, profile-specific)

NOTICE

Bei Tranmission Type 2 – 240 werden die Prozessdaten nicht kor-rekt interpoliert!

6.5.3 Mapping of the process data to be sent (transmit PDO)

The identification and transmission behaviour of the process data to be sent (also PD-In or TPDO) is defined via the TPDO communication parameters of the respective PDO.

A change of the communication parameters is immediately activated and saved non-volatile.

Index	Type	Id-Text	Name		
180xh	RECORD		TPDO communication parameter		
SubIdx	Type	Name			
1	UINT32	cob-ID	Bit	Value	Function
			0..10		11-bit CAN-ID, identifier of the PDO, not writable Corresponds to In-Identifier 1-4 (writable and variable at parallel fieldbus communication)
			11.. 28		29-Bit CAN-ID (not supported)
			29	0	Length of the PDO identifier (11-Bit)
			30	0	Support of RTR functionality (transmission type 252 or 253)
				1	Remote frame is not answered
			31	0	Processing of process output data activated.
				1	Processing of process output data switched off.
2	UINT8	transmission type		0	(Synchronous, acyclic) Transmission of the TPDO after the next SYNC, after a value change.
				1.. 240	(Synchronous, cyclic) Transmission of the TPDO after every xth SYNC. x corresponds to the transmission type.
				252, 253	Transmission of the TPDO after an RTR. Update after every SYNC / every value change
				254, 255	Transmission of the TPDO after value change or expiry of the event time, transmission only after expiry of the inhibit time
3	UINT16	inhibit time		0.. 65535	The minimum time interval between two CAN telegrams on this identifier. Value in 100µs steps. Currently only adjustable in millisecond increments.
5	UINT16	event time		0.. 65535	After the time has elapsed, TPDOs with transmission type 245 or 255 are sent. 0 deactivates this function. Value in multiples of 1ms.

6.5.4 Figure of the mapping parameters

The figure of the mapping parameters is described in the manual for the start-up of the fieldbus interface in subchapter Mapping of the process data.

6.5.5 Time behaviour of the PDOs

In contrast to SDOs, PDOs are always processed in the same time regardless of the utilisation of the device. The minimum possible time wherein PDOs can be processed is described as minimum cycle time and can be defined via parameter is22.

By default, the minimum asynchronous cycle time is 500µs. The cycle time changes depending on the MidIRQ cycle set in is22. KEB devices only support cycle times that are whole multiples of the MidIRQ cycle set in is22.

It should be noted that a shorter cycle time significantly increases the bus traffic.

Further information on is22 can be found in the Programming manual | Control Application / Compact / Pro and in section 5.7 Synchronisation in this manual.

6.6 CANopen Bootup-Sequence and state machine

KEB F6/S6 CANopen devices automatically change to Pre-Operational State after the initialisation phase and send a boot-up message.

The boot-up message is a telegram with the identifier = 700h + CAN node ID (fb64), data length 1 and value 0.

Communication via SDO read and write access is already possible in the Pre-Operational State. However, process data communication is still inactive. To change the state, the CANopen master must trigger a state transition via the corresponding NMT service. NMT service always have the identifier 0.

State transition	NMT Service	Byte 0	Byte 1
3, 6	Start Remote Node	01h	CAN Node ID (fb64)
5, 8	Stop Remote Node	02h	CAN Node ID (fb64)
4, 7	Enter Pre-Operational State	80h	CAN Node ID (fb64)
9, 10, 11	Reset Node	81h	CAN Node ID (fb64)
12, 13, 14	Reset Communication	82h	CAN Node ID (fb64)

The KEB F6/S6 CANopen devices have the state transitions shown in the following figure:

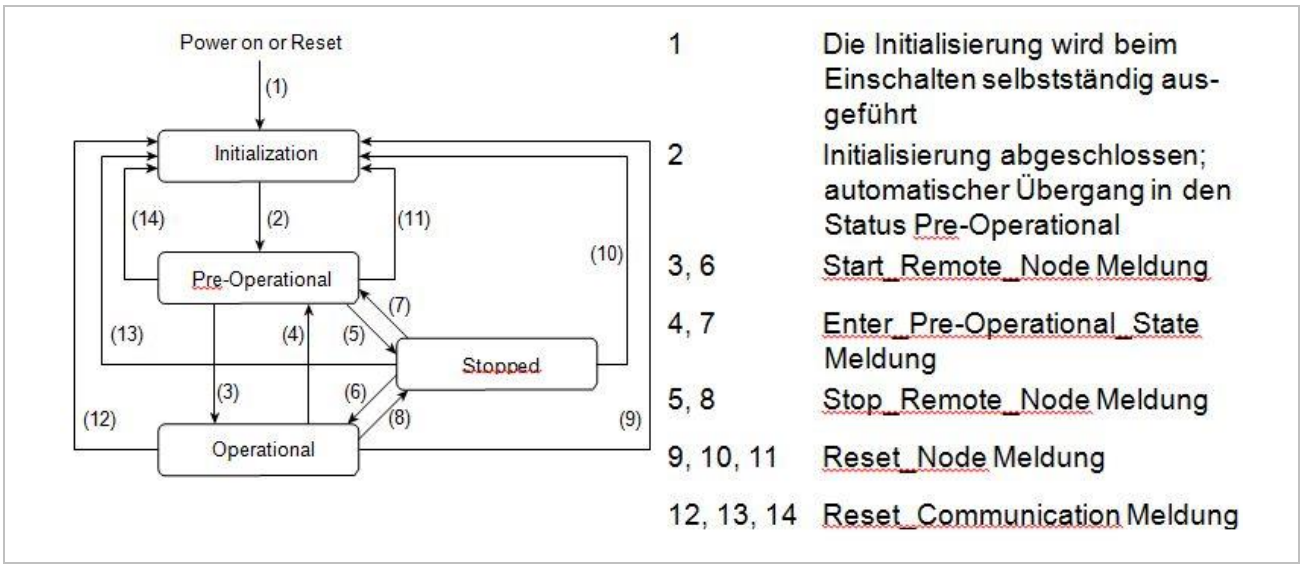


Figure 7: CANopen Bootup-Sequence

6.7 CANopen monitoring functions

NOTICE

Note on version 3.1!

➢ In version 3.1, the scan time of the monitoring functions on the devices of control type A has deteriorated compared to version 3.0. Even if lower values are set in Heartbeat or Nodeguarding Time, the device only checks every 2 milliseconds if a corresponding telegram has been received. The sending of heartbeat telegrams is now also limited to a minimum of 2 milliseconds on devices of control type A. The time behaviour on devices of control type P has not changed compared to version 3.0.

6.7.1 Node guarding

Node guarding is a network management functionality that can be used to monitor the status and functionality of a CAN node. With node guarding, the CAN master sends node guarding requests to a slave at regular intervals (guard time), which the slave then responds to with a node guarding response.The node guarding identifier for request and response is identical to the identifier of the boot-up message.

The response from the monitored CAN node contains the current state of the node and a toggle bit that alternates between the values 0 and 1 with each sent response.

The value of the node state is defined as follows:

Value of the node state	Meaning
1	DISCONNECTED
2	CONNECTING
3	PREPARING
4	PREPARED (STOPPED)
5	OPERATIONAL
127	PRE_OPERATIONAL

If the monitored CAN node does not receive another node guarding request within a specified time (life time) after receiving the first node guarding request, an error is triggered.

The life time is calculated by multiplying the guard time (0x100C) with a life time factor (0x100D). If this is 0, node guarding is deactivated.

The error behaviour of the CAN node can be set via parameter 0x1029 error behaviour.

Index	Name	Bit	Function
0x100C	guard time		Time between 2 node guarding requests (in ms)
0x100D	life time factor		Factor for calculating the life time
0x1029	error behaviour	0	Change into NMT state "Pre-Operational“ (only state "Operational“)
		1	no change of the NMT state

		2	Change to the NMT state "Stopped“

NOTICE
It is not possible to activate the node guarding and the heartbeat protocol simultaneously on one device.

6.7.2 Heartbeat

The heartbeat protocol provides monitoring of the CAN bus without knowledge of the heartbeat producer via the connected participants. When using the heartbeat protocol, the heartbeat producer sends heartbeat telegrams cyclically.

The heartbeat identifier is identical to the node guarding identifier and the identifier of the boot-up message.

The sending of the heartbeat telegrams starts with the state transition to the state Pre-Operational or by setting the producer heartbeat time(0x1017) to a value unequal to 0.

producer heartbeat time	**0x1017**
Value	Function
0	Deactivates the producer heartbeat
1…65536	Time between sending two heartbeat telegrams in ms

The heartbeat telegrams are received by heartbeat consumers (receivers). After receiving the first heartbeat telegram, time monitoring starts in the heartbeat consumer.

If the receiver does not receive another heartbeat telegram within the consumer heartbeat time (0x1016), an error response is executed.

consumer heartbeat time		**0x1016**
Subindex 0 (Byte)		
Value	Function	
10	Number of sub-indices	
Subindex 1...10 (Long)		
Bit	Function	Notice
0...15	heartbeat time [ms]	Time (in ms) wherein a new heartbeat telegram must be received. Value 0 deactivates the heartbeat monitoring.
16...23	node-ID	CAN node-ID of the HB producer. Value 0 deactivates the HB monitoring.
24...31	reserved	reserved

The error behaviour of the CAN node can be set via parameter 0x1029 error behaviour and pn23 E.fb heartbeat stop mode.

Index	**Name**	**Bit**	**Function**
0x1029	error behaviour	0	Change into NMT state "Pre-Operational“ (only state "Operational“)
		1	no change of the NMT state
		2	Change into NMT state "Stopped“
0x2A17	E.fb heartbeat stop mode		Error reaction for CANopen Heartbeat function

6.8 CANopen emergency object

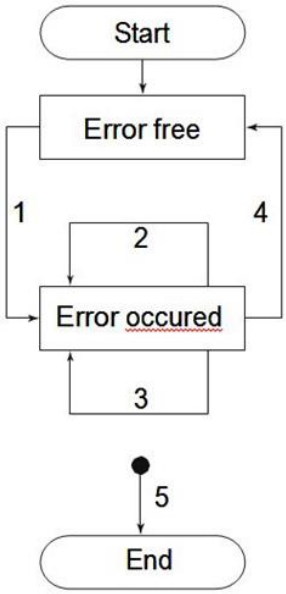
If an internal error occurs in the CANopen device, an emergency (EMCY) object is generated and sent. Emergency objects are only sent once for each error that occurs.

The emergency object identifier is generated via 80h (128) + CAN node-ID (fb64).
The emergency object identifier can be checked via parameter 0x1014.

EMCY cob-ID		**0x1014**
Bit	Value	Function
0...10	80h + CAN Node ID	11-bit CAN ID
11...28	0	29-bit CAN ID
29	0	Length of the EMCY identifier (11-Bit)
30	0	reserved
31	0	EMCY exists / is valid

The transition from an error state back to the normal operating state is also signalled by sending an emergency object.

The error state machine looks like this:



0	If no error is detected, the device changes after initialisation into state "Error free". No error message will be sent.
1	An internal error has been recognised. The device changes into error state. An EMCY message with the associated error code and register is sent.
2	One, but not all error causes have been corrected. An emergency message with error code 0000h (error reset) is sent together with the remaining errors.
3	A new internal error has been recognised. The device remains in error state and sends another EMCY message with the new error code and register.
4	All errors are corrected. The device changes into state "Error free" and sends an emergency message with the error code 0000h (error reset).
5	Restarting or switching off the device

In an emergency message, the error code listed in bytes 0 and 1 usually corresponds to parameter st01 error code.
The error register listed in byte 2 corresponds to parameter ru01 exception state. Bytes 3 to 7 contain a history with the last occurred error register values. The history is filled starting with byte 3.

Byte 0 - 1	**Byte 2**	**Byte 3 - 7**

Error code	Error register	Manufacturer-specific error field
Corresponds (mostly) to st01	Corresponds to ru01	Error register history

The differences between st01 and the CANopen error code and the meaning of the different CANopen error codes are described in the following tables:

ru01	st01	CAN error code
8: reset E. overload	1000h	FF01h
11: reset E overheat pmod.	4210h	FF02h
13: reset E. overheat intern	4110h	FF03h
15: reset E. motorprotection	1000h	FF04h
17: reset ERROR drive overheat	4310h	FF05h
58: ERROR fieldbus watchdog	1000h	8100h

CAN error code	Explanation
0000h	No error
1000h	General error
2200h	Overcurrent power unit (ERROR overcurrent PU)
2300h	Overcurrent analog (ERROR overcurrent analog)
3210h	Overvoltage
3220h	Undervoltage
4210h	Overtemperature power semiconductors (heat sink)
4110h	Overtemperature interior
4310h	Temperature sensor in the motor (e.g. PTC or KTY) has triggered
8100h	General communication error
...	KEB specific error (reset required)
FF01h	Reset overload error
FF02h	Reset power unit overtemperature
FF03h	Reset internal overtemperature
FF04h	Reset error motor protection
FF05h	Reset error motor overtemperature

6.9 CANopen error history

An error history is realized for CANopen via parameter 0x1003 Pre-defined error field.

The parameter is an array with five subindices listing the last five errors that occurred. The newest error is stored in subindex 1. Previous errors are transferred to the next higher subindices.

The number of errors stored in the array is displayed in subindex 0. By writing the value 0 to subindex 0, all error codes stored in the array are deleted. The error status of the inverter is **not** reset by way.

The error codes displayed in subindex 1 to 5 consist of the 16-bit CANopen error code(st01) in the lower two bytes and the corresponding KEB error index(ru01) in the upper two bytes.

Index	Name	SubIdx	Function
0x1003	Pre-defined error field	0	Number of errors contained in this array
	Standard error field [1]	1	First error field, contains the last error that occurred
	Standard error field [2]	2	Second error field, contains the last but one error
	Standard error field [3]	3	Third error field, contains the third last error
	Standard error field [4]	4	Fourth error field, contains the fourth last error
	Standard error field [5]	5	Fifth error field, contains the fifth last error

6.10 CAN bit timing and baud rate

The KEB devices adhere to the specifications of the CiA standard with regard to the set bit timing. The nominal bit timing is as follows:

Area for each segment: Bit time = 8 T_q to 25 T_q

Bit time		
SYNC (1 T_q)	TSEG1 (4…16 T_q)	TSEG2 (2…8 T_q)

Bit rate	Timequantum (t_q)	TSEG1	TSEG2	SJW
20 Kbit/s	4,975 µs	6 T_q	3 T_q	1 T_q
25 Kbit/s	3.95 µs	6 T_q	3 T_q	1 T_q
50 Kbit/s	1.975 µs	6 T_q	3 T_q	1 T_q
100 Kbit/s	995 ns	6 T_q	3 T_q	1 T_q
125 Kbit/s	750 ns	6 T_q	3 T_q	1 T_q
250 Kbit/s	350 ns	6 T_q	3 T_q	1 T_q
500 Kbit/s	150 ns	6 T_q	3 T_q	1 T_q
1000 Kbit/s	50 ns	6 T_q	3 T_q	1 T_q

Info: With SYNC, only the edges from recessive to dominant are used for synchronisation.

For the basic configuration of CANopen, the transmission speed must be selected by parameter fb66.

What kind of transmission rates are possible depends on the line length, the sum of the deceleration times and the bit-timing and must be cleared up for each individual case.

fb66	CAN baud rate	0x2B42
Value	Name	Meaning
1	-	20kBit
2	-	25kBit
3	-	50kBit
4	-	100kBit
5	-	125kBit
6	-	250kBit
7	-	500kBit
8	-	1MBit

6.11 CANopen diagnosis

General diagnosis parameters for fieldbuses and the flashing pattern of the fieldbus LED can be used for the diagnosis of the CANopen interface. It should be observed that parameter fb69 is not available on devices of control type P.

Index	SubIdx	Id-Text	Name	Function
0x2B1F	0	fb31	no PDO data per sync cnt	SYNC intervals without new process data
0x2B5A	2	fb90	CANopen fieldbus state	Value of the register CO_RunState
0x2B5B	2	fb91	CANopen fieldbus error code	Value of the register EmergencyErrField
0 x 2B45	0	fb69	lost messages	Increased when received data is too long for the receive data buffer. (number of lost data in bits).

6.11.1 CANopen manufacturer name

Parameter 0x1008 Manufacturer device name displays the name of the device assigned by KEB as string. The name can not be changed.

"x6P" is displayed in the parameter on the devices of the control type P, "F6A" or "S6A" is displayed on the devices of the control type A depending on the device class.

Index	Id-Text	Name	Function
0x1008		Manufacturer device name	Display of the device name assigned by KEB

7 EtherCAT interface

For the successful start-up of a KEB-x6 EtherCAT slave the master has two different possibilities to determine the device description according to the EtherCAT specification.

On the one hand there is a ESI file (EtherCAT slave information file). This is an XML-based device description file according to the EtherCAT standard. This file can be created amongst others with the COMBIVIS process data wizard for KEB x6 slaves.

Furthermore, the device description can be read from the contents of the EtherCAT EEPROM, the so-called SII (Slave Information Interface). This is made possible by a special protocol which uses special registers in the EtherCAT slave. It is even possible to change the EtherCAT EEPROM (SII) from the EtherCAT master by way.

Most EtherCAT masters use the ESI files for the configuration. However, in order to be able to create an assignment between ESI files and the slaves active on the BUS during so-called 'scanning' of the EtherCAT bus, for example, EtherCAT masters also read the values for VendorId and ProductCode from the slave via the SII protocol. Then the EtherCAT master can read all further configurations from the assigned ESI file.

KEB devices of different control types can not be operated with the same ESI file, because the mailbox sizes of the individual control types differ, for example. MBoxIn and MBoxOut are always identical.

ESI mailbox details	Control type A	Control type P
MinSize	128	96
MaxSize	128	1024
DefaultSize	128	1024

Table 7-1: Differences between the mailbox sizes of the control types

7.1 Hardware and software version

Parameters 0x1009 Hardware Version and 0x100A Software Version are used for the versioning of the KEB devices. They are placed under the profile objects in the pr parameter group.

Parameter 0x1009 Hardware Version is always set to value 1.

The main version of the currently used software is displayed in parameter 0x100A Software Version. This corresponds to the upper half of parameter de16 : ctrl software version.

Index	Name	Function
0x1009	Hardware Version	No function, always 1
0x100A	Software Version	Version Software

7.2 Sync Manager

The pr parameter sync manager com type specifies the communication type of the used SyncManager. Each subindex represents a SyncManager.

A distinction is made between the following communication types:

- 1: Mailbox receive (master to slave)
- 2: Mailbox send (slave to master)
- 3: Processdata output (master to slave)
- 4: Processdata input (slave to master)

Index	Name	SubIdx	Function	Value
0x1C00	sync manager com type	0	Number of Sync Manager channels	4
		1	Communication type of SyncManager 0	1
		2	Communication type of SyncManager 1	2
		3	Communication type of SyncManager 2	3
		4	Communication type of SyncManager 3	4

Parameters sync manager 2 PDO assign and sync manager 3 PDO assign refer via indices to the mapping objects to which SyncManager 2 or SyncManager 3 refers.

Index	Name	SubIdx	Value	Function
0x1C12	sync manager 2 PDO assign	0	1…2	Number of available receive PDOs
		1	0x1600	Index of parameter 1st receive PDO mapping
		2	0x1601	Index of parameter 2nd receive PDO mapping
0x1C13	sync manager 3 PDO assign	0	1…2	Number of available transmit PDOs
		1	0x1A00	Index of parameter 1st transmit PDO mapping
		2	0x1A01	Index of parameter 2nd transmit PDO mapping

7.2.1 Sync Manager Parameters

The sync manager parameters specify information about the available and current state of the synchronization. There is one sync manager parameter object for the process output data (0x1C32 Output sync manager para) and one object for the process input data (0x1C33 Input sync manager para).

The sync manager parameters cannot be written via COMBIVIS. They only serve as display parameters for the current synchronization status.

The sync manager parameters cannot be used as process data. The only exceptions are the subobjects 0x1C32:20 Sync Error and 0x1C33:20 Sync Error.

KEB devices support the following sub-objects of the Sync Manager parameters:

Index	Name	Subindex	Name	Function
0x1C32 / 0x1C33	Output sync manager para/Input sync manager para	1	Sync mode	Current synchronization mode
		2	Cycle Time	Current cycle time
		4	Sync modes supported	Supported synchronization modes
		5	MinCycleTime	Minimum supported cycle time (is22)
		6	Calc and Copy Time	Time measurement PdOut / time reserve Pd-In
		11	SM-Event Missed	Error counter: missing SYNC events
		12	Cycle Time Too Small	Error counter: cycle time too small
		32	Sync Error	Error during synchronization

The subobject [1] Sync mode is automatically set to Sync 0 synchronous if [2] Cycle Time is set to a valid value unequal 0. Restarting or writing an invalid cycle time resets [1] Sync mode to FreeRun.

The subobject [2] Cycle Time is automatically set by the EtherCAT master. Valid values for [2] Cycle Time are integer multiples of [5] Minimum Cycle Time. The value of [2] Cycle Time is used in synchronous mode for the synchronisation of the KEB device.

The subobject [4] Sync modes supported indicates all supported synchronization modes. The modes FreeRun, and DC Sync 0 are supported for KEB devices.

Subobject [5] Minimum Cycle is updated when parameter is22 Basic Tp is written.

The subobject [6] Calc and Copy Time of the output sync manager parameters display the time difference between the SYCN event and the earliest possible time when the process output data (master -> slave) become active in the KEB slave.

The subobject [6] Calc and Copy Time of the input sync manager parameters display the time reserve between the locking of the process input data (slave -> master) and the minimum cycle time.

The subobject [11] SM-Event Missed is an error counter. It is increased if an expected SYNC event does not receive the KEB slave.

The subobject [12] Cycle Time Too Small is an error counter. It is increased if a cycle time specified by the master is lower than [5] Minimum Cycle Time.

The subobject [32] Sync Error indicates if a synchronisation error has occurred. A synchronisation error occurs if the SYNC event fails for two cycles.

The subobjects [6] Calc and Copy Time, [11] SM-Event Missed and [32] Sync Error are only used in synchronous mode.

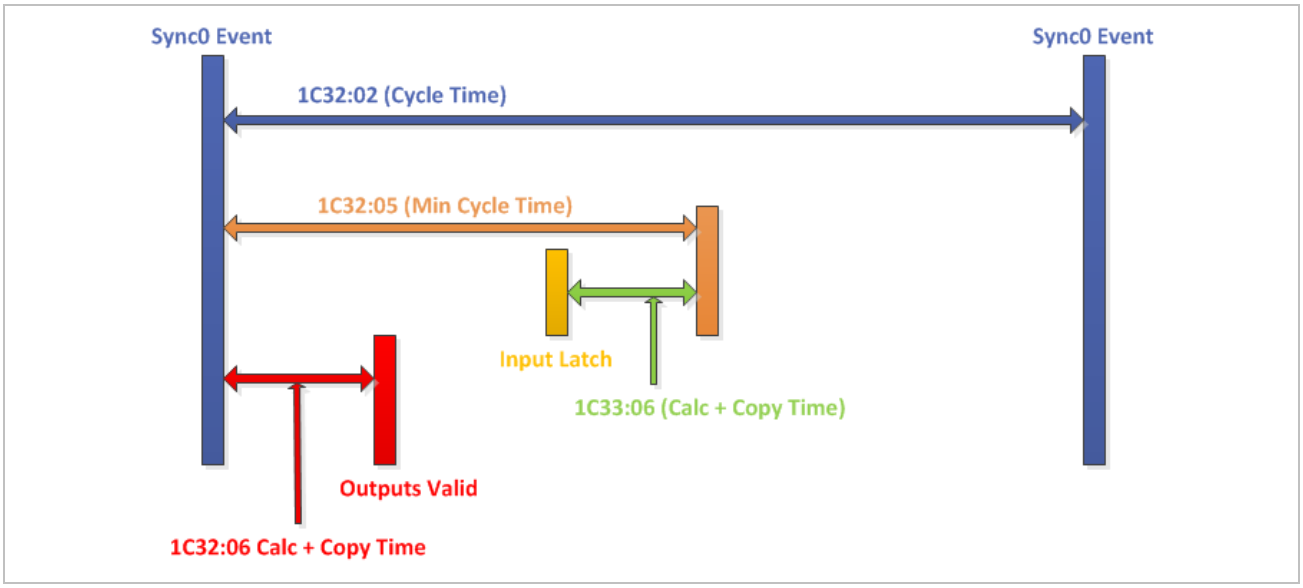


Figure 8: Sync Manager Parameter Calc and Copy Time

NOTICE

Using different Sync 0 and Sync Unit cycles is not correctly supported by KEB devices!

➢ With such a setting, either x process data per SYNC cycle or process data only every xth SYNC cycle are received. The KEB internal interpolation of the data can then not be carried out correctly.

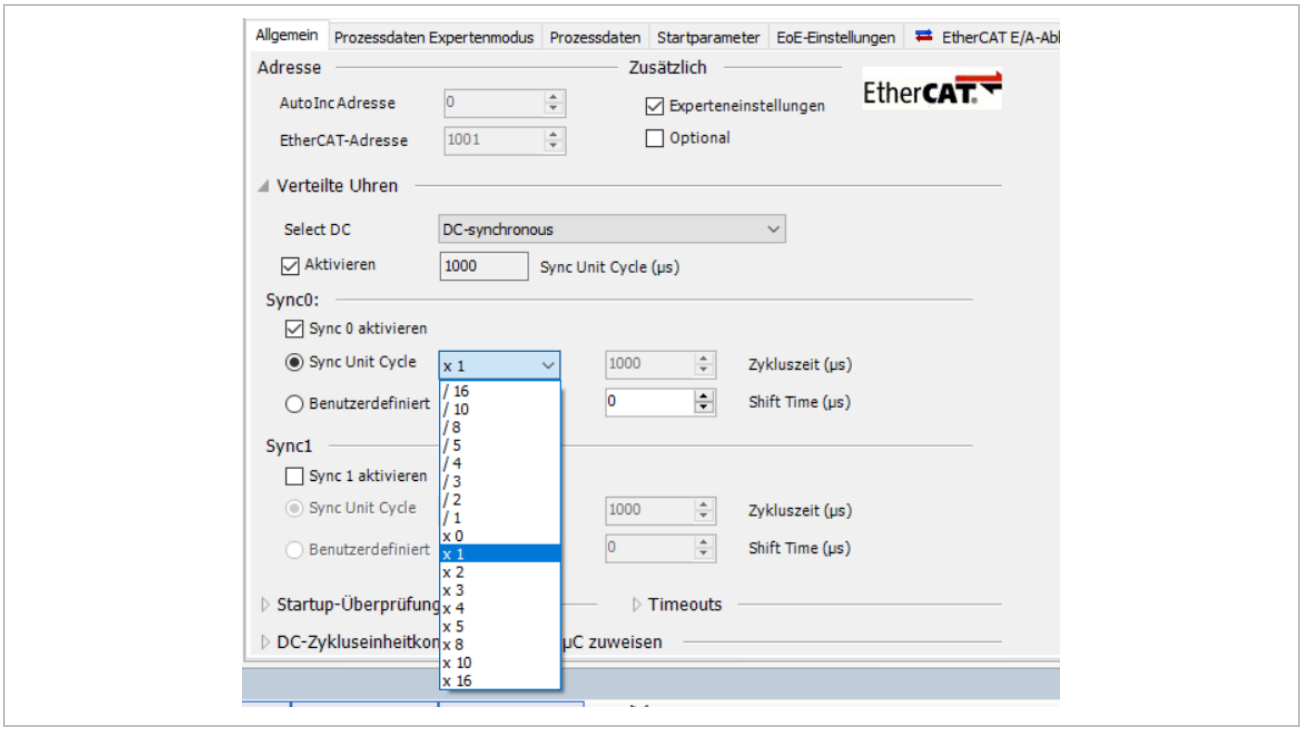


Figure 9: View for different Sync 0 and Sync Unit cycles in COMBIVIS

7.3 EtherCAT diagnosis and timing (control type P)

In order to facilitate the diagnosis of communication faults on the EtherCAT bus, the inverters of generation 6 offer a range of error counters and measured values for control type P.

Index	Id-Text	Name	SubIdx	Function
0x2B5A	fb90	fieldbus state (number)	0	
		EtherCAT fieldbus state	1	Value of register AL status (register address 130h)
0x2B5B	fb91	fieldbus error code (number)	0	
		EtherCAT fieldbus error code	1	Value of register AL status code (register address 134h)

7.3.1 Diagnostic cells of the EtherCAT core (hardware)

The following objects of the EtherCAT core are available:

Index	Id-Text	Name	Function
0x2B14	fb20	ETC invalid frame count P0	Counts the invalid RX frames at port 0
0x2B15	fb21	ETC RX error count P0	Counts the RX errors at port 0
0x2B16	fb22	ETC invalid frame count P1	Counts the invalid RX frames at port 1
0x2B17	fb23	ETC RX error count P1	Counts the RX errors at port 1
0x2B18	fb24	ETC forwarded RX error count P0	Counts the faulty transferred frames at port 0
0x2B19	fb25	ETC forwarded RX error count P1	Counts the faulty transferred frames at port 1
0x2B1A	fb26	ETC processing unit error count	Error counter of the processing unit

These objects are direct images of the error counters integrated in the hardware. They serve to support troubleshooting and must not be observed during normal operation of the EtherCAT fieldbus system.

7.3.2 Time measurement EtherCAT Frame <=> Sync pulse

In applications where drive control and motion control are operated synchronously, faults caused by non-synchronous data processing in the drive control or the control application are difficult to diagnose.

These problems are often caused by the fact that the processing of the EtherCAT frame and the access of the drive control to the data will be overlap.

By using the following two objects it is possible to check the temporal position of the frame processing relative to the sync pulse. This check can also be carried out via the fieldbus wizard.

Index	Id-Text	Name	Function
0x2B1B	fb27	ETC min. sync delay	Minimum time between EtherCAT frame and Sync pulse [us]
0x2B1C	fb28	ETC max. sync delay	Maximum time between EtherCAT frame and Sync pulse [us]

7.3.3 Application error counter

The following error counters are available on the application level:

Index	Id-Text	Name	Function
0x2B1D	fb29	ETC no frame per sync cnt	Number of synchronization intervals wherein no EtherCAT frame was received
0x2B1E	fb30	ETC multiple frames per sync cnt	Number of synchronization intervals wherein several EtherCAT frames were received
0x2B1F	fb31	no PDO data per sync cnt	Number of synchronization intervals wherein no new process data was received

7.3.4 EtherCAT diagnosis assistant

The temporal position of the frame processing relative to the sync pulse can also be diagnosed graphically with COMBIVIS. However, the graphic diagnosis requires that not more than one telegram is sent per sync pulse or that the device has already reached the synchronous state.

For safe synchronous operation, the time position of the frame processing must not overlap the range of the incoming process data telegrams. A shift of the process data telegrams can be achieved by setting a "User Time Shift" from the master or from the slave.

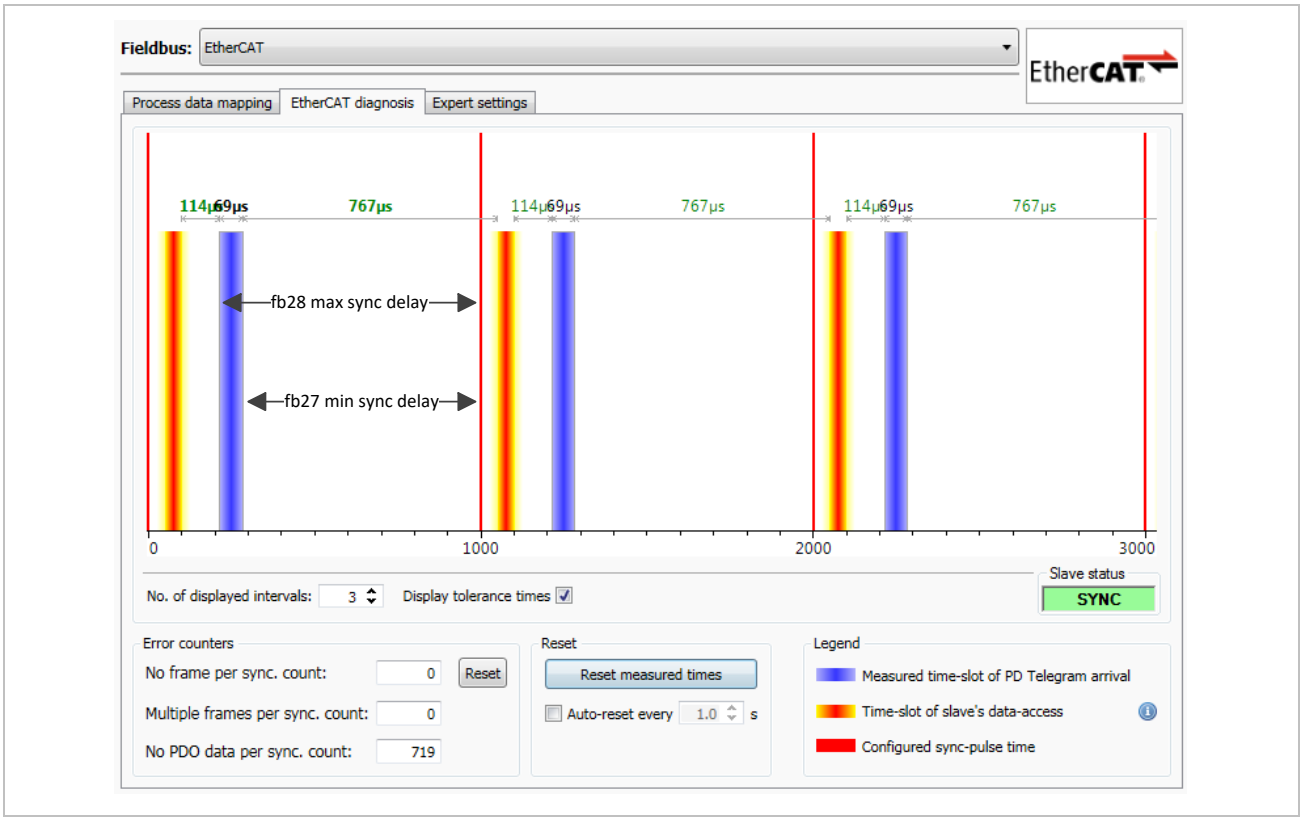


Figure 10: EtherCAT diagnosis assistant

7.4 EtherCAT diagnosis (control type A)

No additional parameters for the EtherCAT diagnosis are available for control type A. The general diagnostic parameters and the fieldbus LEDs should be used here. Only the following parameters can be used for diagnosis:

Index	Id-Text	Name	SubIdx	Function
0x2B1F	fb31	no PDO data per sync cnt	0	Number of synchronization intervals wherein no new process data was received
0x2B5A	fb90	fieldbus state (number)	0	
		EtherCAT fieldbus state	1	Value of register AL status (register address 130h)
0x2B5B	fb91	fieldbus error code (number)	0	
		EtherCAT fieldbus error code	1	Value of register AL status code (register address 134h)

7.5 Ethernet over EtherCAT (EoE)

EoE enables Ethernet frames to be fed into an EtherCAT network via an EoE-capable master. This allows participants of such a network to send and receive Ethernet frames.

From software version 2.4 KEB devices of control type A support the function Ethernet over EtherCAT (EoE). KEB devices of control type P also support EoE.

7.5.1 Start-up of Ethernet over EtherCAT

The function EoE is activated by selecting EtherCAT as active fieldbus system via parameter fb68 fieldbus selection.

The default value for parameter fb68 is EtherCAT. A restart is required if the active fieldbus system is changed during runtime.

An EoE-capable master is required for EoE. The master serves as interface between the Ethernet network and the EtherCAT network.

EoE uses a digital MAC address which is specified by the master. This MAC address can be read out via COMBIVIS via parameter fb106 MAC Address (EthChannel) (A) or fb105 MAC Address (EoE Channel) (P) after it has been specified by the EtherCAT master.

NOTICE

- On the devices of control type A parameter fb106 MAC Address (EthChannel) is also used for the PROFINET Ethernet channel. Therefore, fb106 is written with a default value when the device is started. This default value has **no** influence on the functions of the EoE channel.

NOTICE

- On the devices of control type P, the virtual MAC address is displayed via parameter fb105 MAC Address (EoE Channel). The Ethernet MAC address is always displayed in fb106.

The IP configuration is also preset by the master. Parameter fb108 Ethernet over fieldbus IP configuration can be used to read out the IP configuration set by the master.

The preset parameters by the master are only stored volatile.

- With the setting of the IP configuration it must be observed that the transmitter of the Ethernet telegrams (e.g. PC with COMBIVIS) and the KEB inverter are **not** in the same subnet.
- Several configuration steps are required to configure a COMBIVIS PC and a KEB control for a functional EoE network. These are described in a FAQ. The FAQ can be found under the name "SetupEoEcommunication" on the KEB website.

7.5.2 COMBIVIS functions via Ethernet over EtherCAT

By EoE it is possible to operate COMBIVIS functionalities (parameterization, KEB FTP, etc.) via the fieldbus interface.

When accessing the file system via EoE (KEB FTP via EoE), it is recommended to use the COMBIVIS device memory wizard.

Functional limitations of Ethernet over EtherCAT

COMBIVIS functionalities which are operated via the fieldbus interface are not as performant as COMBIVIS functionalities which are operated via a serial interface.

Depending on the load of the device, the number of active connections and the number of nodes between EoE capable master and COMBIVIS user, the performance may vary.

If the COMBIVS user waits too long for a response due to poor performance, COMBIVIS timeout monitoring is activated and the communication is terminated with an error message.

With the help of the following steps it is possible to improve the performance of the EoE channel in order to avoid interruption of the communication:

- A direct connection between the EoE capable master and the COMBIVIS user should be implemented.
- Not more than one COMBIVIS instance should be active at the same time.
- No files should be downloaded or uploaded via KEB FTP when the device is in active operation (active modulation, active encoder, ...).
- No other COMBIVIS functionality should be used when downloading or uploading files via KEB FTP.

If the steps described above are not feasible / not sufficient, it is also possible to increase the timeout time in COMBIVIS to avoid an interruption of the communication.

The timeout time can be increased under Tools – Options – KEB Parameterisation – Communication. Further information about timeout monitoring can be found in the COMBIVIS manual.

7.6 EtherCAT Hot-Connect

KEB devices of control types A and P support the function EtherCAT Hot-Connect via the station alias.

EtherCAT Hot-Connect via a DIP switch is not supported.

EtherCAT Fast Hot Connect is not supported.

EtherCAT Hot Connect is primarily a functionality of the EtherCAT master. For information about functionalities and limitations of EtherCAT Hot-Connect please contact the manufacturer of your EtherCAT master.

8 VARAN interface

VARAN is only available on devices of control type K, which are not considered in this manual.

Information on the KEB implementation of VARAN can be found in the Programming Manual | Fieldbus Systems V 3.0.

9 PROFINET interface

9.1 PROFINET certificate

The KEB COMBIVERT S6A device has been successfully certified for PROFINET Conformance Class C.

The devices of the control type P are successfully certified for PROFINET Conformance Class B.

The certificates can be found on the KEB homepage.

9.2 PROFINET device description (GSDML)

A device description file for KEB COMBIVERT S6A F6A, S6P and F6P according to GSDML standard is available in the download area of the KEB homepage.

9.3 PROFINET functions

In the following only the essential information about PROFINET functionalities are listed, which are required for the operation of a KEB x6 device with PROFINET. A basic understanding of how the PROFINET bus system works is required for this chapter.

Information about the PROFINET bus system can be obtained from the PROFIBUS User Organization e.V. (PNO).

9.3.1 Cyclic data exchange (process data communication)

The KEB COMBIVERT devices of control type A and P can receive and process up to 64-byte process output data per cycle from the controller. The devices can also provide up to 64 bytes of process input data per cycle to the controller.

On the devices of control type A, both the non-synchronous operating mode (PROFINTE IO-RT) and the synchronous operating mode (PROFINET IO_IRT) are supported.

The synchronous operating mode (PROFINET IO_IRT) is not supported on the devices of the control type P.

The minimum cycle time is 1ms.

9.3.2 Acyclic data exchange (parameter - channel) to PROFIdrive

KEB COMBIVERT devices of control type A and P support the coding of the acyclic services according to PROFIdrive profile also called Base Mode Parameter Access. However, this only applies to the transport of the data, not to its content or coding. This means, the PROFINET interface connection does not support parameters according to the PROFIdrive profile, but only the transport mechanism.

The acyclic parameter request according to Profidrive provides a mechanism via a list of up to 39 parameters can be written or read by means of two PROFINET record accesses. The basic procedure at PROFINET level is as follows:

1. Write-record-request of the master (index = B02Eh, net data = para-request)
2. Write-response of the slave, no net data
3. Read-record-request of the master (Index = B02Eh, no net data)
4. Read-response d of the slave, net data = parameter response

A detailed description of a PROFINET parameter request / a PROFINET parameter response can be found in the annex.

For the revision the network byte order (MSByte first) is valid.

Error codes that are returned for acyclic communication consist of a 32-bit value composed of four components: ErrorCode, ErrorDecode, ErrorCode1, ErrorCode2. Currently, the application of the x6-PROFINET device does not generate its own error codes. All values not equal to 0 are always generated by the used PROFINET stack. The meaning of these error codes can be found in the PROFINET specification.

9.3.3 Additional diagnostic channel via standard Ethernet communication

All Ethernet communication which is not assigned to the PROFINET communication is transferred to the standard Ethernet channel of the KEB COMBIVERT. Thus it is possible

- to read/write parameters (KEB COMBIVIS via Ethernet)
- to download or upload files from the internal file system (KEB FTP via Ethernet)

The exact specifications of the two options are shown in the following table:

Identification	Ethernet communication layer	Parameter	Protocol on the user data
KEB COMBIVIS via Ethernet	IpV4/UDP	UDP-Port = 8000	KEB DIN66019II
KEB FTP via Ethernet	IpV4/UDP	UDP Port = 8002	KEB FTP

➢ The same restrictions apply to the standard Ethernet channel for PROFINET as to the Ethernet over EtherCAT channel. (=> Chapter 7.5 Ethernet over EtherCAT (EoE)).

9.4 PROFINET diagnosis

See also the instructions for commissioning the field bus interface.

9.4.1 PROFINET MAC addresses

PROFINET has a special position with regard to the demand for MAC addresses among the Ethernet-based fieldbuses.

For PROFINET operation, a PROFINET device with 2 ports requires three MAC addresses. One for the interface(IF), one for Port1 and another for Port2. If you also want to use the standard Ethernet channel via the PROFINET ports, an additional MAC address is required for KEB-x6. On the devices of the control type A an additional IP configuration is required.

PROFINET MAC addresses on devices of control type A

The MAC addresses fb103 to fb105 are assigned to each device of control type A in the production process by a protected mechanism. The MAC address fb106 is reserved in the production process for each KEB device. This is a virtual MAC address which is set to the value of fb103 + 3 when the device is started.

Index	Id-Text	Name	Function
0x2B67	fb103	MAC Address (Base)	The basis MAC address. Required for all Ethernet based fieldbus systems. (except EtherCAT)
0x2B68	fb104	MAC Address (Port1)	Only required for PROFINET. Own MAC address for Port1
0x2B69	fb105	MAC Address (Port2)	Only required for PROFINET. Own MAC address for Port2
0x2B6A	fb106	MAC Address (EthChannel)	Virtual MAC address for the Ethernet channel. Required for EoE and the parallel Ethernet channel (PROFINET).

PROFINET MAC addresses on the devices of control type P

On the devices of the control type P fb106 is reserved in the production process for each KEB device. The MAC addresses for port 1, port 2 are generated from this MAC address.

The MAC address for the Ethernet channel on the P cards corresponds to the MAC address in fb106.

Index	Id-Text	Name	Function
0x2B6A	fb106	MAC Address (EthChannel)	The basis MAC address. Required for the Ethernet channel and the PROFINET channel.

9.4.2 PROFINET device name

The PROFINET device name is used to simplify the identification of the device for the user.

The KEB default value for the PROFINET device name is kebx6-*NodeId*. *x6* stands for the device type (S6 or F6). *NodeId* corresponds to the value of the effective node address value (fb102).

If the node address value is smaller than 240, the device name is reset to the default value described above when the device is restarted.

If the effective node address value is 241, 242 or 254, the device name is completely deleted when the device is restarted.

For all other values the PROFINET device name is not changed. The device name can be changed via a PROFINET controller (e.g. the TIA portal) or via COMBIVIS.

When setting the device name by a PROFINET controller, the preset node address value (fb101) is automatically set to 240 to avoid resetting the device name.

When setting the device name by a PROFINET controller, the *remanent* attribute must be observed. Only if the *remanent* attribute is set to TRUE, the device name is also stored non-volatile.

The following rules apply for setting the PROFINET device name:

- Name parts are separated by dots
- The total length is between 1 and 240 characters
- A name part has a total length of 1 to 63 characters
- Name parts consist exclusively of lowercase letters, numbers and the hyphen
- Neither the first nor the last character of a name part is a hyphen

➢ The PROFINET device name is updated in the COMBIVIS view only after a restart of the device. Internally, however, the new name is active.

➢ If PROFINET is not the active fieldbus system, the PROFINET device name is also not initialized. The displayed default value is kebx6-1.

Index	Id-Text	Name	Function
0x2B6B	fb107	PROFINET NameOfStation	Array for the display of the PROFINET device name

10 POWERLINK interface

10.1 Basics of the POWERLINK BUS

POWERLINK is an open real-time Ethernet protocol.

The master is referred to as Managing Node (MN) in a POWERLINK network. A slave is called Controlled Node (CN).

POWERLINK networks operate according to the so-called "polling" approach. The MN controls access to the network. CNs only send on request of the MN.

A POWERLINK cycle is divided into an isochronous phase and an asynchronous phase. Process data (PDOs) are transmitted cyclically in the isochronous phase. The asynchronous phase allows the exchange of non-cyclic data (SDOs).

The request of the MN to a CN is referred to in the isochronous phase as Poll Request (PReq) and in the asynchronous phase as Start of Asynchronous (SoA). The response of a CN is called Poll Response (PRes) in the isochronous phase and Asynchronous Send (ASnd) in the asynchronous phase.

The following characteristics apply to KEB CNs:

- A network can contain up to 253 CNs.
- The minimum cycle time is 400µs.
- The maximum process data size is 32 bytes per cycle for process input and process output data.
- A maximum of 8 bytes of data can be transmitted asynchronously per cycle and client.
- The process data image is dynamic and is stored **volatile**.
- Hot plugging is supported.
- Multiplexing, direct cross-communication and Poll Response chaining are **not** supported.
- The KEB CN cannot be used for the POWERLINK IP routing functionality.Standard Ethernet communication parallel to POWERLINK is not supported.

Further information about the POWERLINK bus system can be found on the Ethernet POWERLINK Standardization Group website.

10.2 POWERLINK functions

The following functions are POWERLINK specific functions which are only available on devices of control type A.

If a bus system other than POWERLINK is active, these functions are not available and the parameters described in this chapter have no effect.

10.2.1 Variable process data offset

The offset of the individual process data objects can be freely selected in a POWERLINK network. The process data offset is specified in bits.

KEB CNs only support whole multiples of bytes as values for the process data offset. Other values are rejected.

The mapping must be deactivated to change the process data offset.

By default, the process data offset of a parameter corresponds to the sum of the length of all preceding parameters. When downloading a complete list, the process data offset must be adjusted subsequently if it does not correspond to this default.

NOTICE
Freely selected offsets can lead to gaps and/or overlaps between the individual process data objects.

Index	Id-Text	Name	Function
0x2B6F	fb111	POWERLINK RPDO offset	Array with 8 elements, enables freely selectable process data offset for receive PDO mapping
0x2B70	fb112	POWERLINK TPDO offset	Array with 8 elements, enables freely selectable process data offset for transmit PDO mapping

10.2.2 Mapping version

The mapping version can be used to uniquely identify different process data mappings. Each mapping is assigned to a mapping version. CNs reject incoming process data with a wrong mapping version. The mapping version is updated by activating the process data mapping.

Info:
The POWERLINK objects 0x1400 and 0x1800 can not be read out via the diagnostic interface. The pr parameters RPDO/TPDO communication parameter which can be read out via the diagnostic interface are CANopen parameters.

Index	SubIdx	Name	Function
0x1400	0	PDO_RxCommParam	
	1	Node ID (Source of Data)	No function (cross-traffic is not supported)
	2	Mapping version	Version number, incoming process data with a different version number are discarded

Index	SubIdx	Name	Function
0x1800	0	PDO_TxCommParam	
	1	Node ID (Source of Data)	Not used by CNs (always 0)
	2	Mapping version	Version number, is sent with process data

10.3 POWERLINK diagnosis

The general diagnosis parameters and the fieldbus LED are available for POWERLINK diagnosis.

A more detailed description of the parameters mentioned here, as well as the flashing pattern of the fieldbus LED, can be found in the chapter Instructions for the start-up of the fieldbus interfaces.

Index	Id-Text	Name	SubIdx	Function
0x2B1F	fb31	no PDO data per sync cnt	0	Number of synchronization intervals where-in no process data was received
0x2B5A	fb90	fieldbus state (number)	0	Number of supported bus systems
		POWERLINK fieldbus state	4	Value of the register NmtState (EPSG DS301 state machine)
0x2B5B	fb91	fieldbus error code (number)	0	Number of supported bus systems
		POWERLINK fieldbus error code	4	Value of the register EplErrorCode (EPSG DS301 error code)

POWERLINK MAC address

The MAC address supported by POWERLINK is displayed in parameter fb103. The MAC addresses displayed in fb104 to fb106 are not used.

Index	Id-Text	Name	Function
0x2B67	fb103	MAC Address (Base)	Basis MAC address.

POWERLINK IP configuration

The POWERLINK specification prescribes a fixed IP configuration. The IP address corresponds to 192.168.100.Node ID, the subnet mask corresponds to 255.255.255.0. The default value of the gateway address corresponds to 192.168.100.254.

The value of the NodeID, also called node address value in this document, can be read out via parameter fb102. The values 0, 240 and 255 are invalid for a POWERLINK CN.

The IP configuration is updated at each restart depending on the node address. The node address is displayed in parameter fb102.

The IP configuration can not be changed by the user, with the exception of the gateway address. Values that do not correspond to the configuration specified by fb102 are rejected. The user can specify any gateway address, which is stored non-volatile.

10.4 Configuration of the KEB POWERLINK CN

A tool for configuring the POWERLINK master is required to configure a POWERLINK CN. (e.g.: the B&R Tool Automation Studio (AS))

A device description file (XDD) is required to import a KEB CN into AS. The XDD of the KEB CN can be generated via the COMBIVIS fieldbus wizard.

Info:
In contrast to other device description files (e.g. EtherCAT ESI file) the XDD file generated by COMBIVIS does not contain the preset mapping. The mapping must be set in the POWERLINK Master configuration tool after the XDD file has been imported.

Not all parameters of the KEB CN are displayed in the AS interface. Parameters that are not writable (AccessType = ro) and cannot be used as process data (PDOMapping = no) are not displayed in the AS interface.

A complete overview of all parameters supported by the CN is provided by the device description file or the interface of the KEB configuration tool COMBIVIS.

11 EtherNet/IP™ interface

EtherNet/IP™ is a communication protocol that is primarily used in industrial applications. It enables the exchange of information between devices. EtherNet/IP™ is based on standardized Ethernet and TCP/IP technology. It is used to transport communication packets based on CIP (Common Industrial Protocol). The EtherNet/IP™ protocol and the CIP are managed by the ODVA (Open DeviceNet Vendors Association).

11.1 Data transmission

EtherNet/IP™ uses a producer/consumer model to exchange time-critical information. The transmitter (producer) provides information for the recipient (consumer). This procedure enables the producer to provide the data to several consumers without a need to transmit the data multiple times.
EtherNet/IP™ uses the standard of the IEEE 802.3 specification. In order to achieve the best possible deterministic behavior, it is recommended to use commercial switch technology with 100 Mbps bandwidth and full duplex cables.

11.1.1 Explicit messaging (parameterizing channel)

The "explicit messaging" enables the access of the master (scanner) to read or write to each parameter in the KEB device. In addition, special management functions can be executed, which are referred to as services.

Service code	Service name	Service description
0x01	Get_Attributes_All	Read all attributes of an object. (not possible for all classes)
0x0E	Get_Attribute_Single	Read one attribute of an object.
0x10	Set_Attribute_Single	Write one attribute of an object.
0x05	Reset	Reset an object. (only objects of class 0x01)

With each message, the user must define the type of service as well as the **class**, **instance**, and the **attribute** of the object. All KEB objects are compliant with the CiA DS301 object specification. Each parameter is defined by an index and a sub-index. A list of the general state messages that can be contained in a response message is described in the annex.

Class
The class determines the type of the object to be accessed. In this case, the value 0x64 or 100 decimal describes the parameter objects.

Instance
The instance represents the index and the address of the parameter in the inverter. Further information about the parameter index can be found in the application description.

Attribute
The attribute corresponds to the sub-index of the parameter with an offset of 0x64 or 100 decimal.

Example READ (get_attribute_single)
The system time ru53: system time is read by the inverter. The parameter has the index 0x2C35. The sub-index is 0x00.
Service: 0x0E (Get_Attribute_Single), class: 0x64, instance: 0x2C35, attribute: 0x64

Example WRITE (set_attribute_single)
Parameter vl05 velocity max is described with the value 1000. Index is 0x2305. The sub-index corresponds to 0x00.
Service: 0x10 (Set_Attribute_Single), class: 0x64, instance: 0x2305, attribute: 0x64, data: 0x03E8 or 1000 decimal

11.1.2 Implicit Messaging (process data communication)

"Implicit Messaging" enables a dedicated communication between producers and consumers. Application-specific data can be exchanged via this communication path. This is also called I/O connection.

In the context of Ethernet/IP™, PD-Out refers to the data that are sent from the control (scanner) to the inverter (adapter) and PD-In refers to the data that are sent from the inverter (adapter) to the control (scanner).

The inverter supports different combinations (assemblies) of the process data for the communication. The combinations 100 and 101 are special, manufacturer-specific combinations of KEB for controlling the inverter based on the CiA402 application profile. The combinations 20 and 70 are used for the "basic speed control", which is based on the CIP provided by ODVA.
The free setting of process data via the pr parameters is not supported by Ethernet/IP. Only the prefabricated assemblies 20, 70, 100 and 101 are supported.

The selection of the active "Assembly", as well as the selection of the IP configuration method, is made via parameter fb113 EtherNet/IP Configuration.

fb113	EtherNet/IP Configuration			**0x2B71**
Bit	Function	Value	Plaintext	Description
0...1	IP configuration method	0	Static	IP configuration is static
		1	Bootp	IP configuration is set via Bootp
		2	DHCP	IP configuration is set via DHCP
3	Assembly Instances	0	Basic Speed Control 20/70	CIP Basic Speed Control
		4	KEB Control 100/101	KEB specific

KEB Control 100/101
Parameters co00 controlword and st00 statusword are required for the KEB specific assemblies 100 and 101. A description of the parameter is in the Programming manual | Application / Compact / Pro.

Basic Speed Control 20/70
The process data assembly for the "CIP Basic Speed Control" is displayed below. This is a simple configuration of the application to set start and stop commands and the reference speed.

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
20	0						Fault Reset		Run Fwd
	1								
	2	Speed Reference (Low Byte)							
	3	Speed Reference (High Byte)							
70	0						Running1		Faulted
	1								
	2	Speed Actual (Low Byte)							
	3	Speed Actual (High Byte)							

11.1.3 Ethernet/IP IP configuration method

The IP configuration method for Ethernet/IP can be selected viathe parameter fb113.
A total of three different methods are available:

1. Static: The IP configuration is specified statically by the user via the parameter fb109.
2. Bootp: The IP configuration must be assigned to the device via the Bootp protocol. The IP configuration must be reassigned when the device is restarted.
3. DHCP: The IP configuration must be assigned to the device via the Bootp protocol. The IP configuration must be reassigned when the device is restarted.

NOTICE If a node address not equal to 0 is set in the device (fb100 - fb102), the IP configuration **cannot** be specified statically.

fb113	EtherNet/IP Configuration			**0x2B71**
Bit	Function	Value	Plaintext	Description
0...1	IP configuration method	0	Static	IP configuration is static
		1	Bootp	IP configuration is set via Bootp
		2	DHCP	IP configuration is set via DHCP

11.2 EtherNet/IP™ objects

11.2.1 Identity Object (class code: 0x01)

The Identity Object contains parameters that allow identification and provide general information about the device. These parameters can be used for electronic coding, general status displays and for determining the devices on the network.

Class attribute					
Class	Instance	Attribute	Name	Type	Access
1	0	1	Revision	UINT	RO
1	0	2	Max Instance	UINT	RO
1	0	6	Max. Class Attribute	UINT	RO
1	0	7	Max. Instance Attribute	UINT	RO

Instance attributes					
Class	Instance	Attribute	Name	Type	Access
1	1	1	Vendor ID	UINT	RO
1	1	2	Device Type	UINT	RO
1	1	3	Product Code	UINT	RO
1	1	4	Revision	STRUCT of	RO
			Major	USINT	RO
			Minor	USINT	RO
1	1	5	State	WORD	RO
1	1	6	Serial Number	UDINT	RO
1	1	7	Product Name	SHORT STRING	RO
1	1	8	State	USINT	RO
1	1	9	Configuration	UINT	RO

11.2.2 Assembly object (class code: 0x04)

The assembly object contains parameters which connect the attributes of several objects. This allows data to be sent or received from or to each of these objects via a single connection.

Assembly objects can be used to connect input or output data. KEB process output data (PD-Out) are sent from a controlling scanner, while process input data (PD-In) are output from the drive converter to the control (scanner) or network.

Class attribute					
Class	Instance	Attribute	Name	Type	Access
4	0	1	Revision	UINT	RO

Instance attributes					
Class	Instance	Attribute	Name	Type	Access
4	100	4	Size in Bytes	UINT	RO
4	100	3	KEB Data	ARRAY of BYTE	RW
4	100	4	Size in Bytes	UINT	RO

Instance attributes					
4	20	3	Basic Speed Control Data	ARRAY of BYTE	RW
4	20	4	Size in Bytes	UINT	RO
4	70	3	Basic Speed Control Data	ARRAY of BYTE	RW
4	70	4	Size in Bytes	UINT	RO

11.2.3 TCP/IP Interface Object (class code: 0xF5)

The TCP/IP interface object provides a mechanism to configure the network interface of the device. The physical connection contains parameters such as the IP address or the network mask.

Class attributes					
Class	Instance	Attribute	Name	Type	Access
F5	0	1	Revision	UINT	RO
F5	0	2	Max Instance	UINT	RO

Instance attributes					
Class	Instance	Attribute	Name	Type	Access
F5	1	1	State	DWORD	RO
F5	1	2	Configuration Capability	DWORD	RO
F5	1	3	Configuration Control	DWORD	RW
F5	1	4	Physical Link	STRUCT of	
			Path Size	UINT	RO
			Path	EPATH	RO
F5	1	5	Interface Configuration	STRUCT of	
			IP Address	UDINT	RW
			Network Mask	UDINT	RW
			Gateway Address	UDINT	RW
			Name Server	UDINT	RW
			Name Server 2	UDINT	RW
			Domain Name	STRING	RW
F5	1	6	Host Name	STRING	RW

11.2.4 Ethernet link object (class code: 0xF6)

The Ethernet link object contains connection-specific status information for the IEEE 802.3 interface. The interface supports two instances for the two connected ports.

Class attributes					
Class	Instance	Attribute	Name	Type	Access
F6	0	1	Revision	UINT	RO
F6	0	2	Max Instance	UINT	RO
F6	0	3	Number of Instances	UINT	RO

Instance attributes					
Class	Instance	Attrib-ute	Name	Type	Ac-cess
F6	1,2	1	Interface Speed	UDINT	RO
F6	1,2	2	Interface Flags	DWORD	RO
F6	1,2	3	Physical Address	ARRAY of USINT	RO
F6	1,2	4	Interface Counters	STRUCT of	
			IN Octets	UDINT	RO
			IN Ucast Packets	UDINT	RO
			IN Non-Ucast Packets	UDINT	RO
			IN Discards	UDINT	RO
			IN Errors	UDINT	RO
			IN Unknown Protocols	UDINT	RO
			OUT Octets	UDINT	RO
			OUT Ucast Packets	UDINT	RO
			OUT Non-Ucast Packets	UDINT	RO
			OUT Discards	UDINT	RO
			OUT Errors	UDINT	RO
F6	1,2	5	Media Counters	STRUCT of	
			Alignment Errors	UDINT	RO
			FCS Errors	UDINT	RO
			Single Collisions	UDINT	RO
			Multiple Collisions	UDINT	RO
			SQE Test Errors	UDINT	RO
			Deferred Transmissions	UDINT	RO
			Late Collisions	UDINT	RO
			Excessive Collisions	UDINT	RO
			MAC Transmit Errors	UDINT	RO
			Carrier Sense Errors	UDINT	RO
			Frame Too Long	UDINT	RO
			MAC Receive Errors	UDINT	RO
F6	1,2	6	Interface Control	STRUCT of	
			Control Bits	WORD	RO
			Forced Interface Speed	UINT	RO
F6	1,2	7	Interface Type	USINT	RO
F6	1,2	8	Interface State	USINT	RO
F6	1,2	9	Admin State	USINT	RW
F6	1,2	10	Interface Label	SHORT STRING	RW
F6	1,2	11	Interface Capability	STRUCT of	
			Capability Bits	WORD	RO
			Speed/Duplex Options	STRUCT of	
			Array Count	USINT	RO
			Speed/Duplex Array	ARRAY of STRUCT of	

Instance attributes					
			Interface Speed	UINT	RO
			Interface Duplex Mode	USINT	RO

11.2.5 KEB inverter object (class code: 0x64)

The KEB inverter object enables access to the manufacturer-specific parameters of the inverter. This class enables the writing and reading of objects. Services Get_Attribute_Single and Set_Attribute_Single can be used to access these parameters.

There are no class attributes for this class.

A list with all index values of the inverter parameters can be found in the Programming Manual | Control Application / Compact / Pro.

Instance Attribute					
Class	Instance	Attribute	Name	Type	Access
0x64	Index	Sub Index	Parameter Name	Type	RO/RW

12 Modbus interface

The Modbus protocol is a communication protocol based on the client/server architecture. It is part of the IEC 61158 standard. KEB-Gerät mit Modbus Unterstützung werden im Folgenden synonym als Server bezeichnet. Applications that access the KEB device via Modbus are called clients.

Each Modbus message is divided into user data and a prefixed identifier (Function Code) that defines the function of the message.

Modbus on devices of control type A

KEB devices of control type A only support the Modbus variant Modbus TCP in a **special version**. For further information, please contact your sales partner.

Class 0 implementation is supported. Clients can read parameters from the server and write them to the server. The functions 3: Read Multiple Registers and 16: Write Multiple Registers are supported. All requested Modbus TCP transactions are carried out with 16-bit registers.

As long as Modbus TCP is the active fieldbus system, the current state of the Modbus TCP connection is displayed via the fieldbus LED of the device.

A connection is established for transactions between client and server. If the client does not send a request to the server for more than 30 seconds, the established connection is rejected.

NOTICE
Up to version 3.0, the "Least Significant Byte First" byte order was assumed for Modbus on control type A devices. This does not correspond to the Modbus specification and was changed to "Most Significant Byte First" with version 3.1.

Modbus on devices of control type P

KEB devices of control type P support the Modbus variants Modbus TCP, Modbus ASCII and Modbus RTU. A second serial interface is required for Modbus ASCII and Modbus RTU, which is only available in control board variant 3. The standard version of control type P only supports Modbus TCP.

Functions 3 and 4: Read Multiple Register, 6: Write Single Register, 16: Write Multiple Register and the KEB-specific functions 100 and 101 are available. Function 100 / 101 is a 32-bit read / write access that uses the KEB-specific addressing via index and subindex. Functions 3 and 4 are implemented identically.

Transaction according to the client/server model

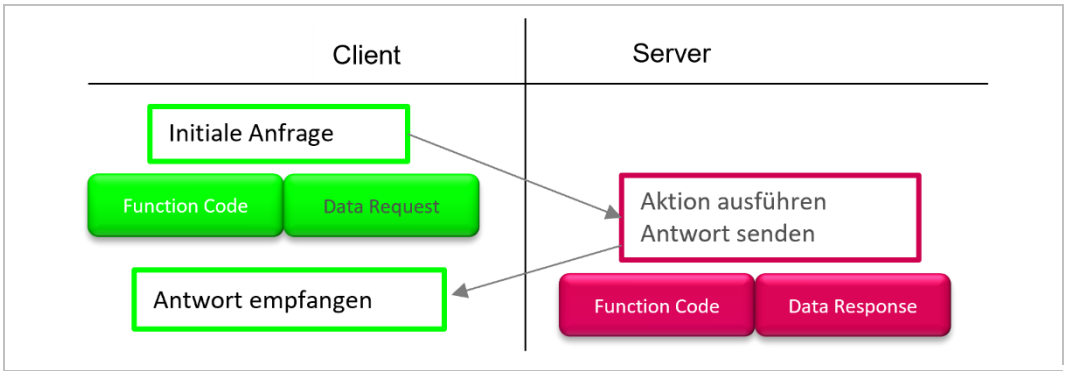


Figure 11: Client / Server Model

12.1 Modbus Error handling (exception handling)

In case of a faulty transaction, the server sends an error message as response to the client. An error message is always 2 bytes long. The most significant bit of the message is set to 1.

Each error message consists of an identifier(Exception Function Code) and an error code. The identifier of the error message corresponds to the identifier of the request with the most significant bit set to 1.

Description	Size	Data
Exception Function Code	1 byte	Function Code + 0x80
Error code	1 byte	Error code (see below)

Modbus error codes		
Code	Name	Description
0x01	ILLEGAL FUNCTION	The received function code is not supported.
0x02	ILLEGAL DATA ADDRESS	Address or data length / number of registers invalid
0x03	ILLEGAL DATA VALUE	The received value is not permissible for the parameter.
0x04	SLAVE DEVICE FAILURE	A critical error occurred on the server while processing the request. The device cannot correct the error on its own.
0x06	DEVICE BUSY	Server is already processing a request (P only)
0x0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Device address invalid (P only, Modbus TCP only)
0x14	READ ONLY	Write access to read-only parameters (P only)
0x18	INVALID PASSWORD	Access to password-protected parameters (P only)
0x1B	INVALID SUBINDEX	Selected parameter subindex does not exist (P only)
0x1E	INVALID OPERATION	Parameter access currently not possible (P only)

Error handling according to the client/server model:

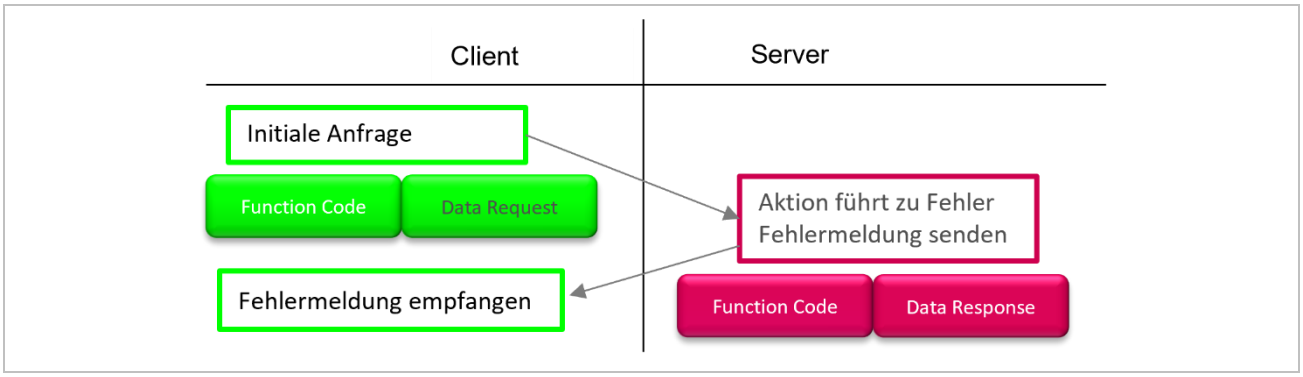


Figure 12: Exception handling

12.2 Modbus address range

The Modbus server is divided into two address ranges. One for process data communication and one for parameter access to individual parameters.

In order to use process data communication, the parameters must be set via the CANopen-compliant mapping objects 0x1600 - 0x1603 (process output data) and 0x1A00 - 0x1A03 (process input data). The restrictions on the length of the process data mapping apply. All data of the different process data mappings are addressed in one access. The number of registers available for mapping is determined by the length of the mapping. One register is available for each 16 bit process data mapping.

The parameter channel allows read or write access to individual parameters. The parameter channel has a maximum size of 2 registers on KEB devices. Modbus registers are 16 bit long by default.

Info
When accessing different data types via the parameter channel, it is important to pay attention to the different length of these parameters. KEB parameters can be 1, 2 or 4 bytes long. To avoid data loss, the right size for setting the registers must be considered. This applies in particular to the access of parameter types "signed" and "unsigned".

Control type A

The process data range is from 0x0000 to 0x1FFF, the parameter channel from 0x2000 to 0xFFFF.

The process data mapping is a continuous memory area that starts at address 0. Only the first process data mapping is used. All data of the process data mapping are addressed in one access.

Control type P

The process data range is from 0x0000 to 0x0FFF, the parameter channel from 0x1000 to 0xFFFF.

The process data mapping is not a continuous memory area. The memory area starts at address 0h with the first process data mapping. The following mappings are each shifted by 100 hex:

Mapping	Start address
First process data mapping	0x0000
Second process data mapping	0x0100
Third process data mapping	0x0200
Fourth process data mapping	0x0300

Table 12-1: Start address of the Modbus mappings on the devices of control type P

12.3 Supported Modbus functions

12.3.1 Function 3 and 4: Read Multiple Registers

This function is used to read registers. The start address of the first register and the number of registers are sent as a request.

Access to several parameters via the parameter channel is not supported. The number of registers must match the data type of the responded parameter.

Parameter fb115 Modbus SubIndex is used to enable access to KEB subindex parameters.

Only function 3 is supported on control type A devices.

Description	Size	Data
Request read process data		
Function Code	1 byte	0x03 / 0x04
Address	2 bytes	Start address of the mapping (0x0000)
Register number	2 bytes	Length of the mapping (in 16-bit registers)
Response read process data		
Function Code	1 byte	0x03 / 0x04
Size of the register values in bytes	1 byte	2 * register number
Register values	2 bytes * number of registers	Read out values
Request read parameter channel		
Function Code	1 byte	0x03 / 0x04
Address	2 Byte	Parameter index
Register number	2 Byte	0x0001 (8, 16-bit) or 0x0002 (32-bit)
Response read parameter channel		
Function Code	1 byte	0x03 / 0x04
Size of the register values in bytes	1 byte	2 * register number
Register values	2 bytes * number of registers	Read out value

12.3.2 Function 6: Write Single Register

This function is used to write a single register. Parameter fb115 Modbus SubIndex is used to enable access to KEB subindex parameters.

This function is only used on devices of control type P.

Description	Size	Data
Request write parameter channel		
Function Code	1 byte	0x06
Address	2 bytes	Parameter index
Register value	2 bytes	Value to be written
Request write parameter channel		
Function Code	1 byte	0x06
Address	2 bytes	As in the inquiry
Register values	2 bytes	As in the inquiry

12.3.3 Function 16: Write Multiple Registers

This function is used to write registers. The start address of the first register, the number of registers, the number of bytes (2 * number of registers) and the values to be written are sent as a request.

Access to several parameters via the parameter channel is not supported. The number of registers must match the data type of the responded parameter.

Parameter fb115 Modbus SubIndex is used to enable access to KEB subindex parameters.

Description	Size	Data
Request write process data		
Function Code	1 byte	0x10
Address	2 bytes	Start address of the mapping (0x0000)
Register number	2 bytes	Length of the mapping (in 16-bit registers)
Size of the values in byte	1 byte	2 * register number
Values to be written	2 bytes * Register number	Values to be written
Response write process data		
Function Code	1 byte	0x10
Address	2 Byte	As in the inquiry
Register number	2 Byte	As in the inquiry
Request write parameter channel		
Function Code	1 byte	0x10
Address	2 Byte	Parameter index
Register number	2 Byte	0x0001 (8, 16-bit) or 0x0002 (32-bit)
Size of the values in byte	1 byte	2 * register number
Value to be written	2 byte * register number	Value to be written
Request write parameter channel		
Function Code	1 byte	0x10
Address	2 Byte	As in the inquiry
Register number	2 Byte	As in the inquiry

12.3.4 Function 100 / 101

These KEB-specific functions are used to enable 32-bit read / write access to a single parameter. The addressing corresponds to the KEB-specific addressing with index and subindex.

This function is only used on devices of control type P.

Description	Size	Data
Request read parameter channel		
Function Code	1 byte	0x64 (Function 100)
Index	2 bytes	>= 0x1000, parameter index
Subindex	1 byte	Parameter subindex
Response read parameter channel		
Function Code	1 byte	0x64 (Function 100)
Index	2 Byte	As in the inquiry
Subindex	1 byte	As in the inquiry
Parameter value	4 Byte	Read parameter value
Request write parameter channel		
Function Code	1 byte	0x65 (Function 101)
Index	2 byte	>= 0x1000, parameter index
Subindex	1 byte	Parameter subindex
Parameter value	4 byte	Parameter value to be written
Request write parameter channel		
Function Code	1 byte	0x65 (Function 101)
Index	2 Byte	As in the inquiry
Subindex	1 byte	As in the inquiry

12.4 Modbus specific parameters

12.4.1 Configure IP addressing of the device (fb114)

To reach the KEB device via TCP/IP, the IP address, subnet mask and gateway must be configured.

On the devices of control type A, the default IP configuration is variable and can be defined via parameter fb114. Similar to Ethernet/IP, the IP configuration can be specified statically, via Bootp or via DHCP.

fb114	ModbusTCP Configuration			**0x2B72**
Bit	Function	Value	Text	Description
0...1	IP configuration method	0	Static	IP configuration is static
		1	Bootp	IP configuration via bootp
		2	DHCP	IP configuration via DHCP

12.4.2 Addressing the subindex via the parameter channel (fb115)

The Modbus parameter channel allows access to individual parameters of the server. However, the addressing specified by Modbus does not support subindices.

To be able to access also the subindexes of parameters, KEB devices provide the parameter fb115. If required, the subindex of the parameter to be addressed can be entered here.

If a parameter is addressed that does not support the subindex set in fb115, an error message is returned.

The value entered in fb115 is deleted by a restart of the device.

On the devices of control type A, the parameter fb115 can be parameterised both via Modbus and via the COMBIVIS interface.

On devices of control type P, write access to parameter fb115 is only possible via Modbus.

Index	SubIndex	Id-Text	Name	Function
0x2B73	0	fb115	Modbus SubIndex	Subindex for access via the parameter channel

12.4.3 Baud rate of the 2nd serial interface (fb116)

See chapter 5.2.2 Second serial interface.

13 Ethernet TCP/IP interface

13.1 General information about Ethernet TCP/IP

Ethernet TCP/IP is only available on the devices of control type P and is activated via parameter fb71: fieldbus options.

Ethernet TCP/IP is only active as long as EtherCAT or CANopen has been selected via fb68: fieldbus selection.

Ethernet TCP/IP uses the parameter fb106: MAC Address (EthChannel) als MAC-Adresse and the parameter fb109: basic IP configuration as IP Configuration.

Establishing a connection between a KEB device and the KEB operating tool COMBIVIS via Ethernet TCP/IP is possible by entering the IP device address (fb109) in COMBIVIS if the KEB device and the COMBIVIS computer are in the same network.

All parameters of the KEB object directory can be accessed via Ethernet TCP/IP depending on the selected password level. Access to the file system is also possible via Ethernet TCP/IP.

NOTICE

Caution against attacks from the Internet!

- In the KEB x6P devices of version 3.1, **no** further security measures have been implemented to protect the communication. (e.g. no end-to-end encryption, no TLS, etc.)
- The Ethernet TCP/IP connection should only be used in local networks.
- It is the responsibility of the end user to take sufficient protective measures to protect data traffic with KEB devices.

13.2 Special functions of KEB-TCP/IP devices

Device identification via the Ethernet TCP/IP interface

On the devices of the control type P parameter fb110 is available for localization of the device via Ethernet. The parameter contains two sub-indices.
Sub-index 1 contains a string that describes the device type and cannot be changed.
Sub-index 2 contains a string to identify the device, e.g. by the name / location of the device. It can be changed.
The parameter is not used by any fieldbus system.

Index	Id-Text	Name	SubIdx	Name-SubIdx	Access	Function
0x2B6E	fb110	Scan names	0		READ ONLY	Number of sub-indices
			1	DeviceType [1]	READ ONLY	Device type
			2	Location [2]	Read-Write	Place / name of the device

Flashing pattern of the Ethernet ports during Ethernet TCP/IP operation

To indicate that the real-time Ethernet interface is used as Ethernet TCP/IP interface, the orange LEDs of the Ethernet ports flash alternately. If there is only one Ethernet port on the device, the orange LED flashes.

With an active fieldbus system, the green LEDs of the Ethernet ports light up when there is communication with the fieldbus system on the respective port.

If there is no communication with the field bus system and Ethernet TCP/IP is not active, the LEDs of the Ethernet ports do not light up.

14 Parallel fieldbus communication (CANopen cross communication)

From version 3.1, KEB devices of control type A and P offer the functionality of exchanging process data with each other via the CANopen field bus system, while at the same time exchanging process data with an EtherCAT, CANopen or PROFINET control system.This functionality is hereinafter referred to as cross communication.

With the help of this functionality it is possible to transfer any values via CANopen process data from one KEB device to another KEB device / several other KEB devices. This allows better synchronisation times between KEB devices or functions such as torque sharing.

Setting the CANopen cross communication or setting up parallel fieldbus communication requires the user to have sufficient knowledge of the used fieldbus architectures and of the implementations of the respective fieldbus systems on KEB devices. This chapter assumes this knowledge.

14.1 CANopen cross communication options

Parameter fb73 can cross communication options is used to handle and parameterise CANopen cross communication. This parameter can be used to activate or deactivate the various functionalities of CANopen cross communication bit by bit.

The functionalities listed here are explained in more detail in the following chapters.

fb73	CAN cross communication options	**0x2B49**
Bit	Name	Notice
0	cross communication activation	Activates the cross communication.
1	variable TPDO1 11-bit COB-ID	Activates the variable assignment of the respective CAN COB-ID. The restrictions of the CAN address range and the 11-bit COB-ID remain unchanged. The self-selected COB-ID is assigned via the CAN communication parameters in the pr parameter group. If the respective flag is deleted, the associated COB ID is reset to its default value.
…		
4	variable TPDO4 11-bit COB-ID	
5	variable RPDO1 11-bit COB-ID	
…		
8	variable RPDO4 11-bit COB-ID	
9	cross communication sync producer	Activates the production of the SYNC telegram. Depending on parameters 0x1005 and 0x1006.
10	automatic boot up to operational	Device changes into the Operational state after initialisation even without a Start-Up command
11	send st00: statusword with heartbeat	The device sends its own status word with every heartbeat telegram
12	dominant sync-interrupt	The device synchronises to the main or cross communication sync-interrupt
13	cross communication function	Master: Device monitors other participants Slave: Device is monitored via HB telegrams

14.2 Activating CANopen cross communication

The various cross communication functionalities, which can be set via the various flags of fb73, are deactivated in the normal state.

It should not be possible for cross communication functionalities to be activated unintentionally, e.g. by incorrectly setting the CANopen pr parameters or by setting a flag in fb73.

This purpose is served by bit 0 of parameter fb73 CAN cross communication options.

fb73	CAN cross communication options	**0x2B49**
Bit	Name	Notice
0	cross communication activation	Activates the cross communication.

As long as fb73 Bit 0 is set to the value 0, **no** functionality of the cross communication is active.

No SYNC telegrams are generated, the CANopen device does not automatically reach Operational, the heartbeat telegrams never contain st00: statusword and the SYNC interrupt of the Ethernet-based fieldbus system is always used.

The single PDO watchdog can also be activated if bit 0 of fb73 is set to value 0, as this is not an explicit CANopen cross communication functionality.

By setting fb73 bit 0 to value 1, the CANopen driver on the KEB device is initialised with the new values of the cross communication. All cross communication functions set up to this point become active.

It is also possible to activate the cross communication function after bit 0 of fb73 has been set to value 1.

14.3 Taking-over of new COB-IDs

fb73	CAN cross communication options	0x2B49
Bit	Name	Notice
1	variable TPDO1 11-bit COB-ID	Activates the variable assignment of the respective CAN COB-ID. The restrictions of the CAN address range and the 11-bit COB-ID remain unchanged. The self-selected COB-ID is assigned via the CAN communication parameters in the pr parameter group. If the respective flag is deleted, the associated COB ID is reset to its default value.
…		
4	variable TPDO4 11-bit COB-ID	
5	variable RPDO1 11-bit COB-ID	
…		
8	variable RPDO4 11-bit COB-ID	

CANopen cross communication makes it possible to variably set the previously fixed COB IDs (COB ID = base address + CAN node ID (fb64)).

Since CANopen process data are received depending on its COB-ID, the variable setting of the process data enables any communication connections between the different CANopen devices in a network.

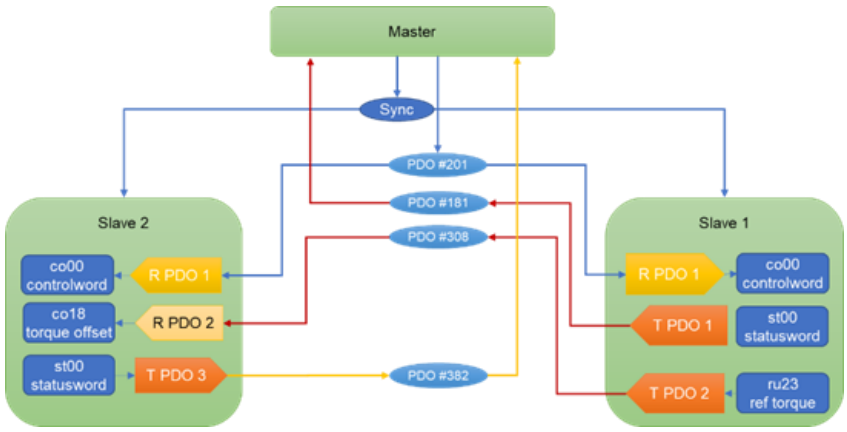


Figure 13: Example of CANopen cross communication

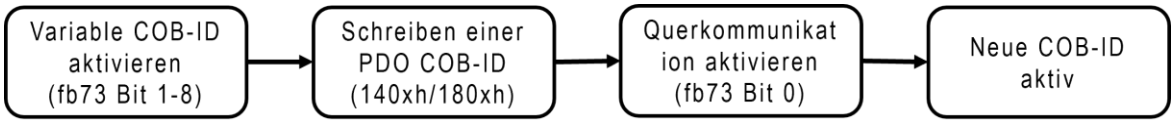


Figure 14: Setting a new COB-ID

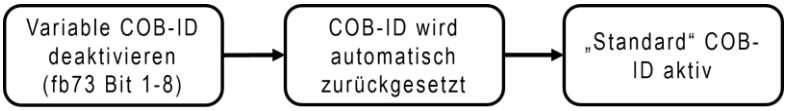


Figure 15: Restoring the "standard" COB IDs

For more information on CANopen process data communication, see chapter 6.5 CANopen process data objects (PDO).

14.4 Synchronisation via CANopen cross communication

fb73	CAN cross communication options	0x2B49
Bit	Name	Notice
9	cross communication sync producer	Activates the production of the SYNC telegram. Depending on parameters 0x1005 and 0x1006.

CANopen cross communication enables KEB devices to produce a CANopen SYNC telegram to which other CAN participants can synchronise. There should never be more than one KEB device producing a SYNC in a CAN network. In addition, KEB devices should not produce a SYNC in a network in which a CANopen master is already organising the network traffic.

The generation of a SYNC telegram is activated via bit 9 of parameter fb73 and objects 0x1005 and 0x1006 in the pr parameter group.

0x1005 is constant except for bit30. Changes that do not affect bit30 are ignored.

Parameter 0x1006 is used to set the cycle time wherein CANopen SYNC telegrams are generated. The time in 0x1006 is specified in µs, but can only be the whole multiples of the MidIrq grid set in is22 : Basic Tp. When writing to 0x1006 the written value is rounded to the next multiple of the MidIrq grid (see fb10).

The minimum possible cycle time is also depending on is22. The maximum cycle time results from the value limitation of the data type. If the value is 0, no SYNC telegrams are generated. If CANopen is not the active field bus system and CAN cross communication is not active, 0x1006 is reset to value 0 on a restart.

Bit30 of parameter 0x1005 and bit 9 of parameter fb73 are always identical. When writing to fb73 bit 9, bit30 in 0x1005 is always affected and vice versa. If bit 9 in parameter fb73 or bit30 of parameter 0x1005 is set to 0, the cycle time in 0x1006 is also set to 0.

Note
It is not recommended to synchronise a KEB device to its own SYNC telegram. A KEB device that generates a SYNC telegram should use the SYNC of a higher-level fieldbus system (EtherCAT / PROFINET) for synchronisation (if possible) or be operated in asynchronous mode.

pr	cob-ID sync message	0x1005
Bit	Value	Notice
0…10	128	Standard SYNC Identifier (non-adjustable)
11…28	0	-
29	0	11-bit CAN ID supported
30	0	does not generate SYNC telegrams
	1	Generates SYNC telegrams
31	x	-

pr	communication cycle period	0x1006
Value range	Unit	Notice
500 – 4.294.967.295	µs	Time between 2 CANopen SYNC commands in µs

14.5 Automatic ramp-up of CANopen cross communication

fb73	CAN cross communication options	0x2B49

Bit	Name	Notice
10	automatic boot up to operational	Device changes into the Operational state after initialisation even without a Start-Up command

In order for KEB devices to reach the CANopen Operational status, a start node NMT of a CANopen master is required. This NMT is sent when the CANopen master receives a boot-up telegram from a CANopen slave.

Cross communication makes it possible to establish CANopen communication between different KEB devices even without a CANopen master.
In this case, it is possible to set the KEB device into the Operational state even without a CANopen master by setting bit 10 in fb73 and restarting the device with co09.

14.6 Monitoring of CANopen cross communication

fb73	CAN cross communication options	**0x2B49**
Bit	Name	Notice
11	send st00: statusword with heartbeat	The device sends its own status word with every heartbeat telegram
13	cross communication function	Master: Device monitors other participants Slave: Device is monitored via HB telegrams

The status of KEB communication participants can be monitored as part of cross communication. For this purpose, a participant in the communication network is defined as cross communication master by bit 13 in fb73. The status of the other communication participants can then be monitored via this device.

In order to monitor a communication participant, it must produce CANopen heartbeat telegrams. For additional monitoring of participants, bit 11 in fb73 can be used to activate that a communication participant also transmits its status word (st00) with the heartbeat telegram.

The devices to be monitored must be offer to the cross communication master via an entry in the array 0x1016. The CAN-ID (fb64) and the associated monitoring time (0x1017) of the device to be monitored must be entered in the array. (See chapter 6.7.2 Heartbeat)

The status, CAN node ID and the status word of the monitored devices are displayed in the cross communication master via the array fb74. The number of the monitored devices is displayed in fb74[0] .

fb74	CAN cross communication monitoring		**0x2B4A**
SubIdx	Byte	Name	Notice
0		number of monitored slaves	Number of monitored devices
1…10	0	monitored state	State of the monitored device
	1	node ID	CAN node ID of the monitored device
	2…3	statusword	Received statusword of the monitored device

➢ Transmitting the status word in the heartbeat telegram increases this by 3 bytes, which increases the bus traffic and can result in the heartbeat telegram no longer being recognised by other CAN masters. It is therefore recommended to use this option only with a KEB cross communication master.

14.7 Parallel fieldbus communication

CANopen cross communication makes it possible to establish CANopen process data communication in parallel with an Ethernet-based fieldbus. The first two process data mappings are used by the Ethernet-based fieldbus system and process data mappings three and four by the CANopen cross communication.

Faster cycle times can be reached by the direct exchange of process data and the bus load on the Ethernet-based fieldbus system can be reduced. In addition, cross communication enables the structure of multi-level communication systems wherein the fieldbus slave of a system acts as the fieldbus master of another system.

Parallel fieldbus communication is only supported by the Ethernet-based fieldbus systems EtherCAT and PROFINET.

It is possible that CANopen and the Ethernet-based fieldbus system use different synchronisation cycles. However, a KEB device can only synchronise to one synchronisation cycle.

The fieldbus LED (NET ST) always indicates the status of the Ethernet-based fieldbus system during parallel fieldbus communication.

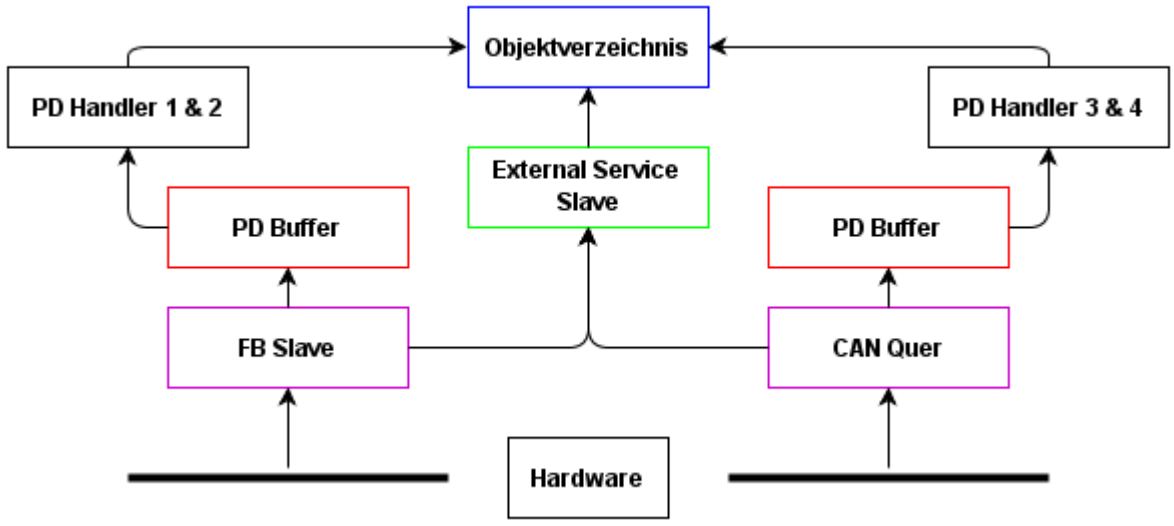


Figure 16: Concept of parallel fieldbus communication

The restriction to two PDOs reduces the number of available CAN cob-IDs for CAN communication via PDOs. This is not a problem for sending PDOs, because the CANopen specification allows several participants to receive the same PDO. However, when receiving PDOs, the number of cob-IDs is limitated, since PDOs would be overwritten if several participants send to the same PDO.

14.7.1 Parameterisation of parallel fieldbus communication

To activate parallel fieldbus communication, one of the corresponding modes must be selected in parameter fb68.

fb68	fieldbus selection	0x2B44
Value	Name	Notice
32	EtherCAT + cross communication	Activates EtherCAT and CAN cross communication
35	PROFINET + cross communication	Activates PROFINET and CAN cross communication

In addition, all CANopen cross communication options must be set as described in the previous chapters. (e.g.: setting the COB IDs or monitoring)

- If parallel fieldbus communication is active via fb68 , the first or second process data mapping **cannot** be activated for cross communication in parameter fb73 . A corresponding write access is rejected by the KEB device.
- The same applies if the first or second process data mapping for cross communication is activated in fb73, then parallel fieldbus communication **cannot** be activated in fb68.

NOTICE
The fieldbus systems operated in parallel cannot both be operated synchronously. If both fieldbus systems are set to synchronous operation, the KEB device only synchronises to one of the two sync telegrams received.
This can lead to problems on the field bus system, to which the KEB device then does not synchronise. It is therefore strongly recommended that KEB devices that are operated as CAN Cross slaves and that are connected to both the CAN Cross and the Ethernet fieldbus are only operated synchronously by one of the two fieldbus systems.
If possible, a KEB device that is operated as a CAN cross master should not synchronise to its own sync telegram.

For CAN cross devices, a dominant SYNC must always be defined to which the KEB device synchronises. This is done via bit 12 in parameter fb73.

fb73	CAN cross communication options		0x2B49
Bit	Name	Value	Notice
12	dominant sync interrupt	0	Default, SYNC interrupt of the Ethernet-based fieldbus system is used for synchronisation
		1	SYNC interrupt of the cross communication field bus system is used for synchronisation

Only the sync interval of the dominant fieldbus system is displayed in parameter fb19 measured sync interval. If the second fieldbus system is also operated synchronously, the sync interval can only be checked via the fieldbus-specific pr parameters.

Parameter 0x1006 communication cycle period. is used for CANopen.
Parameter 0x1C32.2 Cycle Time is used for EtherCAT.
There is no parameter for Profinet.

pr	Communication cycle period	0x1006

Value range	Unit	Notice
500 - 4.294.967.295	µs	Time between 2 CANopen SYNC telegrams in µs

pr	Cycle Time	0x1C32.2

Value range	Unit	Notice
500.000 – 16.000.000	ns	Time between 2 received EtherCAT SYNC commands in ns

14.8 Cross communication watchdog (Single PDO WD)

CANopen cross communication allows independent communication with a fieldbus master / fieldbus slave for each process data mapping.

The fieldbus watchdog (see chapter 5.8 Fieldbus watchdog) is reset when any PDO is received. When using cross communication, communication breakdowns could remain undetected.

Example: An EtherCAT slave (PDO 1 & 2) is also a CAN cross communication master for two CAN slaves (PDO 3 & 4). Communication to the EtherCAT master is stopped, no new process data are received on PDO 1 & 2. The fieldbus watchdog is not triggered in this case, since process data are still receiving on PDO 3 & 4.

The cross communication watchdog can be used to monitor communication via several fieldbus systems.

The cross communication watchdog recognises when process data communication has failed on one of the four process data objects and triggers the watchdog error after an adjustable time.

Both the fieldbus watchdog and the cross communication watchdog can trigger the watchdog error (ERROR fieldbus watchdog) and the watchdog warning bit (64: watchdog).

The cross-communication watchdog can be configured via parameter pn24.

Index	Id-Text	SubIdx	Name	Function
0x2A18	pn24	0	cross communication watchdog	Structure of the cross communication watchdog
		1	cross communication watchdog stop mode	Error response (see pn22)
		2	watchdog time 1st PDO	see pn21, only responds to PD of the 1st PDO
		3	watchdog time 2nd PDO	see pn21, only responds to PD of the 2nd PDO
		4	watchdog time 3rd PDO	see pn21, only responds to PD of the 3rd PDO
		5	watchdog time 4th PDO	see pn21, only responds to PD of the 4th PDO

14.9 Example configuration of parallel fieldbus communication

The steps required to configure parallel fieldbus communication are described below using an example. **Only** the configuration of parallel fieldbus communication is considered. The configuration of the respective fieldbus systems is not part of this example.
In this example, we assume a machine with three KEB inverters. One of these inverters is to be configured. The inverter is connected to an EtherCAT controller via Ethernet and receives the parameter co00 from it synchronously every millisecond via process data mapping 1: controlword and the parameter co16: target velocity. In addition, the inverter is connected to the two other KEB inverters in a CANopen network.
The inverter is to be configured so that it acts as the CANopen master in the CANopen network and forwards the controlword (co00) received from the EtherCAT master to the other two inverters synchronously in a 2 ms cycle. In addition, the inverter should forward the received target velocity (co16) to one of the two other inverters when the value changes.
In this example, it is assumed that the communication between the KEB device and the EtherCAT master is already configured.

Step 1: Changing fb68
As the inverter is to use EtherCAT and CAN cross-communication, the parameter fb68 must first be set to the value 32: EtherCAT + CAN Cross can be switched.It is assumed at this point that fb73 is set to the value 0. If this is not the case, the changeover of fb68 may fail (interlocking mapping 1 & 2).

Step 2: Setting fb73
After the requirements described above, the "variable TPDO3 11-bit cob-ID" and the "variable TPDO4 11-bit cob-ID" are activated in fb73. In addition, the flags "cross communication sync producer" and "automatic boot up to operational" must be set and the "cross communication function" must be set to "cc master". Bit 0 "cross communication activation" must also be set to activate the parallel fieldbus communication functions.

Step 3: Restart
After changing fb68 and fb73, the device must be restarted. In addition, the device must be restarted so that the device enters the CANopen Operational state. This is done either via co09 or by switching the power supply off and on again.

Step 4: Setting the CANopen SYNC to be produced
As synchronous CANopen process data is to be transmitted, the device must produce a CANopen SYNC. This is done by entering the desired cycle time of 2ms in pr parameter 0x1006. After this step, the SYNC produced by the device should already be visible on the CANopen network.

Step 5: Setting the process data
In the "3rd transmit PDO mapping" (0x1A02), the parameter "co00: controlword" must be set and in the "4th transmit PDO Mapping"(0x1A03) "co16: target velocity" can be set.
In addition, the "cob-ID" and the "transmission type" of the "3rd TPDO communication parameter" (0x1802) and the "4th TPDO communication parameter" (0x1803) must be adjusted. In this example, the process data should be sent with each SYNC, so the "transmission type" must be set to "1" for both process data. The "cob-ID" is largely variable, but care must be taken to ensure that the "PDO valid" flag is set and that the cob-IDs of the two transmit PDOs are not identical.
After setting and activating, the process data sent should already be visible on the CANopen network.

Step 6: Setting the CANopen slave inverters
The other two inverters in this example act as CANopen slaves. CANopen slaves can also send process data, but in this example we are only configuring the slaves as recipients of process data.
As both slaves are not connected to the EtherCAT master, the fb68 must be set to the value 1: CANopen can be provided.

The flags "automatic boot up to operational" and "cross communication activation" must be set in fb73. In addition, the "variable RPDO3 11-bit cob-ID" and the "variable RPDO4 11-bit cob-ID" must be activated.
The "receive PDO mappings" (0x1602, 0x1603) must be set to receive the "co00: controlword" and "co16: target velocity" can be set.
The "cob-ID" and the "transmission type" of the "RPDO communication parameter" (0x1402, 0x1403) must be adjusted. The "transmission type" must be set to 1. **The "cob-ID" of the slave RPDOs must be set to the value of the cob-ID of the associated master TPDO.**
After the corresponding settings have been made, the devices must also be restarted.

Optional: Monitoring
Monitoring of the KEB devices connected via CANopen is possible via the CANopen heartbeat function. For additional information, the send st00. flag can be set in the monitored devices in fb73: statusword with heartbeat can be activated.

.

15 Annex

15.1 Mapping parameters and Sync Manager parameters version 3.2

Abbreviations	RO	Read Only		
	nPD	not available for Process Data Communication		
	CAN	CANopen Type	V	Variable
			ST	Structure
			A	Array
	HW	Control board version	A	S6A / F6A
			P	S6P / F6P

Index	SubIdx	CAN	Type	Name	Upper limit	Lower limit	Mult.	Div	Unit	nPD	RO	HW
1009h	0	V	STRING	Hardware Version	1	1	1	1	---	X	X	A, P
100Ah	0	V	STRING	Software Version	4294967295	0	1	1	---	X	X	A, P
1400h	0	ST	UINT8	1st RPDO communication parameter	2	2	1	1	---	X	X	A, P
	1		UINT32	cob-ID	4294967295	1	1	1	---	X	---	
	2		UINT8	transmission type	255	0	1	1	---	X	---	
1401h	0	ST	UINT8	2nd RPDO communication parameter	2	2	1	1	---	X	X	A, P
	1		UINT32	cob-ID	4294967295	1	1	1	---	X	---	
	2		UINT8	transmission type	255	0	1	1	---	X	---	
1402h	0	ST	UINT8	3rd RPDO communication parameter	2	2	1	1	---	X	X	A,K,P
	1		UINT32	cob-ID	4294967295	1	1	1	---	X	---	
	2		UINT8	transmission type	255	0	1	1	---	X	---	
1403h	0	ST	UINT8	4th RPDO communication parameter	2	2	1	1	---	X	X	A, P
	1		UINT32	cob-ID	4294967295	1	1	1	---	X	---	
	2		UINT8	transmission type	255	0	1	1	---	X	---	
1600h	0	ST	UINT8	1st receive PDO mapping	32	0	1	1	---	X	---	A, P
	1…32		UINT32		4294967295	0	1	1	---	X	---	
1601h	0	ST	UINT8	2nd receive PDO mapping	8	0	1	1	---	X	---	A, P
	1…8		UINT32		4294967295	0	1	1	---	X	---	
1602h	0	ST	UINT8	3rd receive PDO mapping	8	0	1	1	---	X	---	A, P
	1…8		UINT32		4294967295	0	1	1	---	X	---	
1603h	0	ST	UINT8	4th receive PDO mapping	8	0	1	1	---	X	---	A, P
	1…8		UINT32		4294967295	0	1	1	---	X	---	
1800h	0	ST	UINT8	1st TPDO communication parameter	5	5	1	1	---	X	X	A, P
	1		UINT32	cob-ID	4294967295	1	1	1	---	X	---	
	2		UINT8	transmission type	255	0	1	1	---	X	---	
	3		UINT16	inhibit time	6553	0	1	1	ms	X	---	

	4		UINT8	reserved	0	0	1	1	---	X	X	
1800h	5	ST	UINT16	event time	65535	0	1	1	---	X	---	A, P
Index	**SubIdx**	**CAN**	**Type**	**Name**	**Upper limit**	**Lower limit**	**Mult.**	**Div**	**Unit**	**nPD**	**RO**	**HW**
1801h	0	ST	UINT8	2nd TPDO communication parameter	5	5	1	1	---	X	X	A, P
	1		UINT32	cob-ID	4294967295	1	1	1	---	X	---	
	2		UINT8	transmission type	255	0	1	1	---	X	---	
	3		UINT16	inhibit time	6553	0	1	1	ms	X	---	
	4		UINT8	reserved	0	0	1	1	---	X	X	
	5		UINT16	event time	65535	0	1	1	---	X	---	
1802h	0	ST	UINT8	3rd TPDO communication parameter	5	5	1	1	---	X	X	A, P
	1		UINT32	cob-ID	4294967295	1	1	1	---	X	---	
	2		UINT8	transmission type	255	0	1	1	---	X	---	
	3		UINT16	inhibit time	6553	0	1	1	ms	X	---	
	4		UINT8	reserved	0	0	1	1	---	X	X	
	5		UINT16	event time	65535	0	1	1	---	X	---	
1803h	0	ST	UINT8	4th TPDO communication parameter	5	5	1	1	---	X	X	A, P
	1		UINT32	cob-ID	4294967295	1	1	1	---	X	---	
	2		UINT8	transmission type	255	0	1	1	---	X	---	
	3		UINT16	inhibit time	6553	0	1	1	ms	X	---	
	4		UINT8	reserved	0	0	1	1	---	X	X	
	5		UINT16	event time	65535	0	1	1	---	X	---	
1A00h	0	ST	UINT8	1st transmit PDO mapping	32	0	1	1	---	X	---	A, P
	1…32		UINT32		4294967295	0	1	1	---	X	---	
1A01h	0	ST	UINT8	2nd transmit PDO mapping	8	0	1	1	---	X	---	A, P
	1…8		UINT32		4294967295	0	1	1	---	X	---	
1A02h	0	ST	UINT8	3rd transmit PDO mapping	8	0	1	1	---	X	---	A, P
	1…8		UINT32		4294967295	0	1	1	---	X	---	
1A03h	0	ST	UINT8	4th transmit PDO mapping	8	0	1	1	---	X	---	A, P
	1…8		UINT32		4294967295	0	1	1	---	X	---	
1C00h	0	A	UINT8	Sync Manager Communication Type	4	4	1	1	---	X	X	A, P
	1…4				4	0						
1C12h	0	A	UINT16	sync manager 2 PDO assign	2	2	1	1	---	X	---	A, P
	1…2				1601h	1600h						
1C13h	0	A	UINT16	sync manager 3 PDO assign	2	2	1	1	---	X	---	A, P
	1…2				1A01h	1A00h						

Index	SubIdx	CAN	Type	Name	Upper limit	Lower limit	Mult.	Div	Unit	nPD	RO	HW
1C32h	0	ST	UINT8	Output sync manager para	32	32	1	1	---	X	X	A, P
	1		UINT16	Sync mode	3	0	1	1	---	X	X	
	2		UINT32	Cycle Time	16000000	0	1	1000	us	X	X	
	4		UINT16	Sync modes supported	65535	0	1	1	---	X	X	
	5		UINT32	Minimum Cycle Time	16000000	0	1	1000	us	X	X	
	6		UINT32	Calc and Copy Time	4294967295	0	1	1000	us	X	X	
	11		UINT16	SM-Event Missed	65535	0	1	1	---	X	X	
	12		UINT16	Cycle Time Too Small	65535	0	1	1	---	X	X	
	32		UINT8	Sync Error	1	0	1	1	---	---	X	
1C33h	0	ST	UINT8	Input sync manager para	32	32	1	1	---	X	X	A, P
	1		UINT16	Sync mode	3	0	1	1	---	X	X	
	2		UINT32	Cycle Time	16000000	0	1	1000	us	X	X	
	4		UINT16	Sync modes supported	65535	0	1	1	---	X	X	
	5		UINT32	Minimum Cycle Time	16000000	0	1	1000	us	X	X	
	6		UINT32	Calc and Copy Time	4294967295	0	1	1000	us	X	X	
	11		UINT16	SM-Event Missed	65535	0	1	1	---	X	X	
	12		UINT16	Cycle Time Too Small	65535	0	1	1	---	X	X	
	32		UINT8	Sync Error	1	0	1	1	---	---	X	

15.2 Fieldbus parameters version 3.2

Abbreviations	RO	Read Only		
	nPD	not available for Process Data Communication		
	CAN	CANopen Type	V	Variable
			ST	Structure
			A	Array
	HW	Control board version	A	S6A / F6A
			P	S6P / F6P

For structures, the name of the structure is entered in the line of subindex 0 (number).
HW indicates the control types of the devices which support the specified parameter.

Index	SubIdx	CAN	Type	Name	Upper limit	Lower limit	Mult.	Div	Unit	nPD	RO	HW
2B0Ah	0	V	UINT16	sync interval	16000	0	1	1	µs	X	---	A, P
2B0Bh	0	V	UINT16	Set sync level	1000	0	1	10	µs	X	---	A, P
2B0Ch	0	V	UINT16	KP sync PLL	256	0	1	1	---	X	---	A, P
2B0Dh	0	V	UINT8	DIN66019 node id	238	1	1	1	---	X	---	A
2B0Dh	0	V	UINT8	drive node ID	238	1	1	1	---	X	---	P
2B0Eh	0	V	UINT8	DIN66019 baud rate	12	0	1	1	---	X	---	A, P
2B0Fh	0	ST	UINT8	node IDs	2	2	1	1	---	X	X	P
	1		UINT8	application node ID	255	0	1	1	---	X	---	P
	2		UINT8	debugger node ID	255	0	1	1	---	X	---	P
2B10h	0	V	UINT8	Fieldbus node injection	255	1	1	1	---	X	---	P
2B13h	0	V	UINT16	measured sync interval	65535	0	2	3	µs	---	X	A
2B13h	0	V	UINT16	measured sync interval	65535	0	64	75	µs	---	X	P
2B14h	0	V	UINT8	ETC invalid frame count P0	255	0	1	1	---	X	X	P
2B15h	0	V	UINT8	ETC RX error count P0	255	0	1	1	---	X	X	P
2B16h	0	V	UINT8	ETC invalid frame count P1	255	0	1	1	---	X	X	P
2B17h	0	V	UINT8	ETC RX error count P1	255	0	1	1	---	X	X	P
2B18h	0	V	UINT8	ETC for. RX error count P0	255	0	1	1	---	X	X	P
2B19h	0	V	UINT8	ETC for. RX error count P1	255	0	1	1	---	X	X	P
2B1Ah	0	V	UINT8	ETC processing unit error count	255	0	1	1	---	X	X	P
2B1Bh	0	V	UINT16	ETC min. sync delay	32000	0	8	25	µs	X	---	P
2B1Ch	0	V	UINT16	ETC max. sync delay	32000	0	8	25	µs	X	---	P
2B1Dh	0	V	UINT16	ETC no frame per sync cnt	32000	0	1	1	---	X	---	P
2B1Eh	0	V	UINT16	ETC mult. frames per sync cnt	32000	0	1	1	---	X	---	P
2B1Fh	0	V	UINT16	no PDO data per sync cnt	32000	0	1	1	---	X	---	A, P
2B20h	0	V	UINT8	LED 'DEV ST' blink status	1	0	1	1	---	---	---	A, P
2B25h	0	V	INT16	ETC PD access min. sync delay	3200	0	8	25	µs	X	---	P
2B26h	0	V	INT16	ETC PD access max. sync delay	3200	0	8	25	µs	X	---	P
2B3Ch	0	V	UINT8	process data size selection	4	0	1	1	---	X	---	A, P
2B40h	0	V	UINT8	CAN node ID	127	1	1	1	---	X	---	A, P
Index	**SubIdx**	**CAN**	**Type**	**Name**	**Upper limit**	**Lower limit**	**Mult.**	**Div**	**Unit**	**nPD**	**RO**	**HW**

2B42h	0	V	UINT8	CAN baud rate	8	1	1	1	---	X	---	A, P
2B43h	0	V	UINT16	fieldbus configuration	15	0	1	1	---	X	---	A, P
2B44h	0	V	UINT8	fieldbus selection	6	0	1	1	---	X	---	A
2B44h	0	V	UINT8	fieldbus selection	1	0	1	1	---	X	---	P
2B45h	0	V	UINT16	CAN lost messages	65535	0	1	1	---	---	X	A
2B46h	0	ST	UINT8	CAN options	7	7	1	1	---	X	X	A
	1		UINT32	CAN option code	4294967295	0	1	1	---	X	---	A
	2		UINT16	Tx PDO base ID	65535	0	1	1	---	X	---	A
	3		UINT16	Rx PDO base ID	65535	0	1	1	---	X	---	A
	4		UINT16	Tx PDO1 cycle time	10000	10	1	1	---	X	---	A
	5		UINT16	Tx PDO2 cylce time	10000	10	1	1	---	X	---	A
	6		UINT16	Tx PDO3 cycle time	10000	10	1	1	---	X	---	A
	7		UINT16	Tx PDO4 cycle time	10000	10	1	1	---	X	---	A
2B47h	0	V	UINT32	fieldbus options	2	1	1	1	---	---	---	P
2B48h	0	V	UINT32	change cnt	4294967295	0	1	1	---	---	---	P
2B50h	0	ST	UINT8	MRTE module	3	3	1	1	---	X	X	A
	1		UINT8	MRTE module support	1	0	1	1	---	X	---	A
	2		UINT8	MRTE module state	255	0	1	1	---	X	X	A
	3		UINT16	MRTE module version	255	0	1	1	---	X	X	A
2B5Ah	0	ST	UINT8	fieldbus state	5	5	1	1	---	X	X	A, P
	1		UINT16	EtherCAT fieldbus state	65535	0	1	1	---	X	X	A, P
	2		UINT16	CANopen fieldbus state	65535	0	1	1	---	X	X	A, P
	3		UINT16	PROFINET fieldbus state	65535	0	1	1	---	X	X	A
	4		UINT16	POWERLINK fieldbus state	65535	0	1	1	---	X	X	A
	5		UINT16	EtherNet/IP fieldbus state	65535	0	1	1	---	X	X	A
	6/4		UINT16	ModbusTCP fieldbus state	65535	0	1	1	---	X	X	A, P
2B5Bh	0	ST	UINT8	fieldbus error code	5	5	1	1	---	X	X	A, P
	1		UINT16	EtherCAT fieldbus error code	65535	0	1	1	---	X	X	A, P
	2		UINT16	CANopen fieldbus error code	65535	0	1	1	---	X	X	A, P
	3		UINT32	PROFINET fieldbus error code	9568255	0	1	1	---	X	X	A
	4		UINT16	POWERLINK fieldbus error code	65535	0	1	1	---	X	X	A
	5		UINT16	EtherNet/IP fieldbus error code	65535	0	1	1	---	X	X	A
2B64h	0	V	UINT8	node ID switch value	255	0	1	1	---	X	X	A
2B65h	0	V	UINT8	adjusted node ID value	255	0	1	1	---	X	---	A
2B66h	0	V	UINT8	effective node ID	255	0	1	1	---	X	X	A
2B66h	0	V	UINT8	effective node ID	255	0	1	1	---	X	---	P
2B67h	0	V	UINT32	FB MAC Address (Base)	4211081215	0	1	1	---	X	---	A
2B68h	0	V	UINT32	PNET MAC Address (Port1)	4211081215	0	1	1	---	X	---	A

Index	SubIdx	CAN	Type	Name	Upper limit	Lower limit	Mult.	Div	Unit	nPD	RO	HW
2B69h	0	V	UINT32	PNET MAC Address (Port2)	4211081215	0	1	1	---	X	---	A
2B69h	0	V	UINT32	MAC Address (EoE Channel)	4211081215	0	1	1	---	X	---	P
2B6Ah	0	V	UINT32	MAC Address (EthChannel)	4211081215	0	1	1	---	X	X	P
2B6Ah	0	V	UINT32	MAC Address (EthChannel)	4211081215	0	1	1	---	X	---	A
2B6Bh	0	A	UINT8	PROFINET NameOfStation	240	240	1	1	---	X	X	A
	1 - 240				32	122						
2B6Bh	0	V	STRING	PROFINET NameOfStation	240	0	1	1	---	X	X	P
2B6Ch	0	ST	UINT8	Ethernet over fieldbus IP configuration	3	3	1	1	---	X	X	A, P
	1		UINT32	IP address	4294967295	0	1	1	---	X	---	A, P
	2		UINT32	subnet mask	4294967295	0	1	1	---	X	---	A, P
	3		UINT32	gateway address	4294967295	0	1	1	---	X	---	A, P
2B6Dh	0	ST	UINT8	basic IP configuration	3	3	1	1	---	X	X	A, P
	1		UINT32	IP address	4294967295	0	1	1	---	X	---	A, P
	2		UINT32	subnet mask	4294967295	0	1	1	---	X	---	A, P
	3		UINT32	gateway address	4294967295	0	1	1	---	X	---	A, P
2B6Eh	0	A	UINT8	Scan names	2	2	1	1	---	X	---	P
	1 - 2				32	0						
2B6Fh	0	A	UINT8	POWERLINK RPDO offset	8	8	1	1	---	X	---	A
	1 - 8				255	0						
2B70h	0	A	UINT8	POWERLINK TPDO offset	8	8	1	1	---	X	---	A
	1 - 8				255	0						
2B71h	0	V	UINT8	EtherNet/IP Configuration	6	0	1	1	---	X	---	A
2B72h	0	V	UINT8	ModbusTCP Configuration	2	0	1	1	---	X	---	A
2B73h	0	V	UINT8	ModbusTCP Subindex	255	0	1	1	---	X	---	A

15.3 Fieldbus-specific requests for parameters

Feedback to parameter requests contain a return value.

For requests via the diagnostic interface, this is a KEB specific value. The possible values are described in the Programming Manual | Control Application / Compact / Pro.

For SDO requests via the fieldbus interface the KEB specific return value is converted to a fieldbus specific value.

An exception is the fieldbus system VARAN, via which the return values are not transmitted and the fieldbus system Modbus TCP, whose return values are described in chapter 12.3 Function 3 and 4: Read Multiple Registers

This function is used to read registers. The start address of the first register and the number of registers are sent as a request.

Access to several parameters via the parameter channel is not supported. The number of registers must match the data type of the responded parameter.

Parameter fb115 Modbus SubIndex is used to enable access to KEB subindex parameters.

Only function 3 is supported on control type A devices.

Description	Size	Data
Request read process data		
Function Code	1 byte	0x03 / 0x04
Address	2 bytes	Start address of the mapping (0x0000)
Register number	2 bytes	Length of the mapping (in 16-bit registers)
Response read process data		
Function Code	1 byte	0x03 / 0x04
Size of the register values in bytes	1 byte	2 * register number
Register values	2 bytes * number of registers	Read out values
Request read parameter channel		
Function Code	1 byte	0x03 / 0x04
Address	2 Byte	Parameter index
Register number	2 Byte	0x0001 (8, 16-bit) or 0x0002 (32-bit)
Response read parameter channel		
Function Code	1 byte	0x03 / 0x04
Size of the register values in bytes	1 byte	2 * register number
Register values	2 bytes * number of registers	Read out value

15.3.1 Function 6: Write Single Register

This function is used to write a single register. Parameter fb115 Modbus SubIndex is used to enable access to KEB subindex parameters.

This function is only used on devices of control type P.

Description	Size	Data
Request write parameter channel		
Function Code	1 byte	0x06
Address	2 bytes	Parameter index
Register value	2 bytes	Value to be written
Request write parameter channel		
Function Code	1 byte	0x06
Address	2 bytes	As in the inquiry
Register values	2 bytes	As in the inquiry

15.3.2 Function 16: Write Multiple Registers

This function is used to write registers. The start address of the first register, the number of registers, the number of bytes (2 * number of registers) and the values to be written are sent as a request.

Access to several parameters via the parameter channel is not supported. The number of registers must match the data type of the responded parameter.

Parameter fb115 Modbus SubIndex is used to enable access to KEB subindex parameters.

Description	Size	Data		
Request write process data				
Function Code	1 byte	0x10		
Address	2 bytes	Start address of the mapping (0x0000)		
Register number	2 bytes	Length of the mapping (in 16-bit registers)		
Size of the values in byte			1 byte	2 * register number
Values to be written	2 bytes * Register number	Values to be written		
Response write process data				
Function Code	1 byte	0x10		
Address	2 Byte	As in the inquiry		
Register number	2 Byte	As in the inquiry		
Request write parameter channel				
Function Code	1 byte	0x10		
Address	2 Byte	Parameter index		
Register number	2 Byte	0x0001 (8, 16-bit) or 0x0002 (32-bit)		
Size of the values in byte	1 byte	2 * register number		
Value to be written	2 byte * register number	Value to be written		
Request write parameter channel				
Function Code	1 byte	0x10		
Address	2 Byte	As in the inquiry		
Register number	2 Byte	As in the inquiry		

15.3.3 Function 100 / 101

15.3.4 These KEB-specific functions are used to enable 32-bit read / write access to a single parameter. The addressing corresponds to the KEB-specific addressing with index and subindex.

This function is only used on devices of control type P.

Description	Size	Data
Request read parameter channel		
Function Code	1 byte	0x64 (Function 100)
Index	2 bytes	>= 0x1000, parameter index
Subindex	1 byte	Parameter subindex
Response read parameter channel		
Function Code	1 byte	0x64 (Function 100)
Index	2 Byte	As in the inquiry
Subindex	1 byte	As in the inquiry
Parameter value	4 Byte	Read parameter value
Request write parameter channel		
Function Code	1 byte	0x65 (Function 101)
Index	2 byte	>= 0x1000, parameter index
Subindex	1 byte	Parameter subindex
Parameter value	4 byte	Parameter value to be written
Request write parameter channel		
Function Code	1 byte	0x65 (Function 101)
Index	2 Byte	As in the inquiry
Subindex	1 byte	As in the inquiry

.

- There is no 1 to 1 conversion between KEB return values and fieldbus specific return values. Therefore, for some fieldbus return values it is not possible to infer to the KEB return value.
- e.g.: The EtherCAT return value 08000021h is assigned to the KEB return values 4, 6, 8 and 14.

The following tables describe this conversion:

KEB return	EtherCAT	CANopen	PROFINET	POWERLINK	EtherNet/IP
0: OK	00000000h No error	00000000h No error	FFFFh No error	00000000h No error	0000h Success
1: Device not ready	05040000h SDO protocol timeout	08000020h Device not ready	11h Device not ready	05040000h SDO protocol timeout	0002h Resources not available
2: Address or password invalid	06020000h Object does not exist	06010010h Password invalid	00h Invalid parameter number	06010000h Object access not supported	000Ch Object status conflict
3: Data invalid	06090030h Value range exceeded	06090030h Data invalid	17h Data invalid	06090030h Value range exceeded	0020h Invalid parameter
4: Parameter write protected	08000021h No transfer due to local settings	06010002h Parameter write protected	01h Parameter write protected	06010002h Parameter write protected	000Eh Non-modifiable attribute
5: BCC error	06040047h Internal incompatibility	06010000h Operation not possible	65h Manufacturer-specific 0	06060000h No access due to hardware error	001Fh Manufacturer-specific
6: Device busy	08000021h No transfer due to local settings	08000022h Device busy	14h Value inadmissible	08000022h Device busy	0002h Resources not available
7: Service not available	06040047h Internal incompatibility	05040001h Service not available	66h Manufacturer-specific 1	06040047h Internal incompatibility	001Fh Manufacturer-specific
8: Password invalid	08000021h No transfer due to local settings	06010010h Password invalid	67h Manufacturer-specific 2	08000021h No transfer (local settings)	000Ch Object status conflict
9: Telegram frame error	06060000h No access due to hardware error	08000020h No data transfer	68h Manufacturer-specific 3	06060000h No access due to hardware error	001Fh Manufacturer-specific
10: Transmission error	06060000h No access due to hardware error	06010000h Operation not possible	69h Manufacturer-specific 4	06060000h No access due to hardware error	001Fh Manufacturer-specific
11: Subindex invalid	06090011h Subindex invalid	06090011h Subindex invalid	03h Subindex invalid	06090011h Subindex invalid	0009h Invalid attribute data
12: Language invalid	06040047h Internal incompatibility	06090012h Language invalid	6Ah Manufacturer-specific 5	06040047h Internal incompatibility	001Fh Manufacturer-specific
13: Address invalid	06020000h Object does not exist	06020000h Invalid address	00h Invalid parameter number	06020000h Object does not exist	0016h Object does not exist
14: Operation not possible	08000021h No transfer due	06010000h Operation not	6Bh Manufacturer-	06040043h Parameter incom-	001Fh Manufacturer-

KEB return	EtherCAT	CANopen	PROFINET	POWERLINK	EtherNet/IP
	to local settings	possible	specific 6	patibility	specific

Table 15-1: Conversion of the KEB return values

15.4 Values for fb90: fieldbus state

Fieldbus system	Value	Plaintext
EtherCAT	0x0001	ECAT_ALSTATUS_INIT
	0x0002	ECAT_ALSTATUS_PREOP
	0x0003	ECAT_ALSTATUS_BOOTSTRAP
	0x0004	ECAT_ALSTATUS_SAFEOP
	0x0008	ECAT_ALSTATUS_OPERATIONAL
	0x0010	ECAT_ALSTATUS_ERROR
CANopen	0x0000	CAN301_STATE_INITIALISATION
	0x0004	CAN301_STATE_STOPPED
	0x0005	CAN301_STATE_OPERATIONAL
	0x007F	CAN301_STATE_PREOPERATIONAL
VARAN	0x0000	VARAN_STATE_INIT
	0x0001	VARAN_STATE_RUNNING
PROFINET	0x0000	PROFINET_STATE_OFF
	0x0001	PROFINET_STATE_INIT_DONE
	0x0002	PROFINET_STATE_COMM_READY
	0x0004	PROFINET_STATE_COMM_RUN
	0x0008	PROFINET_STATE_SYNC_ACTIVE
POWERLINK	0x0000	EPSG_DS301_NMT_GS_OFF
	0x0019	EPSG_DS301_NMT_GS_INITIALISING
	0x001C	EPSG_DS301_NMT_CS_NOT_ACTIVE
	0x001D	EPSG_DS301_NMT_CS_PREOP
	0x001E	EPSG_DS301_NMT_CS_BASIC_ETH
	0x0029	EPSG_DS301_NMT_GS_RESET_APPL
	0x0039	EPSG_DS301_NMT_GS_RESET_COMM
	0x004D	EPSG_DS301_NMT_CS_STOPPED
	0x005D	EPSG_DS301_NMT_CS_PREOP2
	0x006D	EPSG_DS301_NMT_CS_READYTOOP
	0x0079	EPSG_DS301_NMT_GS_RESET_CONFIG
	0x00FD	EPSG_DS301_NMT_CS_OPERATIONAL
EtherNet/IP	0x0100	ENIPSLV_STATE_SETMACADDR
	0x0200	ENIPSLV_STATE_SETCONFIG
	0x0300	ENIPSLV_STATE_REGAP
	0x0400	ENIPSLV_STATE_CHANNELINIT
	0x0A00	ENIPSLV_STATE_RUN_IDLE
ModbusTCP	0x0000	MBTCPSLV_COMSTATE_OFF
	0x0001	MBTCPSLV_COMSTATE_CONNECTING
	0x0002	MBTCPSLV_COMSTATE_CONNECTED

15.5 Values for fb91: fieldbus error code

Fieldbus system	Value	Plaintext
Error codes for all fieldbus systems	0xFE70	INIT_ERR_PD
	0xFED4	INIT_ERR_HSP5_NR_PD
	0xFF9C	INIT_ERR_HSP5_PD
	0xFFCE	INIT_ERR_SHM
	0xFFE2	INIT_ERR_NETX_BASE
	0xFFEC	INIT_ERR_I2C_MUTEX
	0xFFF5	INIT_ERR_FB_SLV_OR_DRV
	0xFFF6	INIT_ERR_I2C
	0xFFF7	INIT_ERR_LED_TMR
	0xFFF8	INIT_ERR_LED_DRV
	0xFFF9	INIT_ERR_BASE_DRV_NOT_ONLINE
	0xFFFA	INIT_ERR_MSG_QUEUE
	0xFFFB	INIT_ERR_FB_TYPE_INVALID
	0xFFFF	Fieldbus inactive
EtherCAT	0x0000	ECAT_AL_STATUS_CODE_NO_ERROR
	0x0001	ECAT_AL_STATUS_CODE_UNSPECIFIED_ERROR
	0x0011	ECAT_AL_STATUS_CODE_INVALID_REQUESTED_STATE_CHANGE
	0x0012	ECAT_AL_STATUS_CODE_UNKNOWN_REQUESTED_STATE
	0x0013	ECAT_AL_STATUS_CODE_BOOTSTRAP_NOT_SUPPORTED
	0x0014	ECAT_AL_STATUS_CODE_NO_VALID_FIRMWARE
	0x0015	ECAT_AL_STATUS_CODE_INVALID_MAILBOX_CONFIGURATION_BOOTSTRAP
	0x0016	ECAT_AL_STATUS_CODE_INVALID_MAILBOX_CONFIGURATION_PREOP
	0x0017	ECAT_AL_STATUS_CODE_INVALID_SYNC_MANAGER_CONFIGURATION
	0x0018	ECAT_AL_STATUS_CODE_NO_VALID_INPUTS_AVAILABLE
	0x0019	ECAT_AL_STATUS_CODE_NO_VALID_OUTPUTS
	0x001A	ECAT_AL_STATUS_CODE_SYNCHRONIZATION_ERROR
	0x001B	ECAT_AL_STATUS_CODE_SYNC_MANAGER_WATCHDOG
	0x001C	ECAT_AL_STATUS_CODE_SYNCTYPES_NOT_COMPATIBLE
	0x001D	ECAT_AL_STATUS_CODE_INVALID_OUTPUT_CONFIGURATION
	0x001E	ECAT_AL_STATUS_CODE_INVALID_INPUT_CONFIGURATION
	0x001F	ECAT_AL_STATUS_CODE_INVALID_WD_CONFIGURATION
	0x0020	ECAT_AL_STATUS_CODE_SLAVE_NEEDS_COLD_START
	0x0021	ECAT_AL_STATUS_CODE_SLAVE_NEEDS_INIT
	0x0022	ECAT_AL_STATUS_CODE_SLAVE_NEEDS_PREOP
	0x0023	ECAT_AL_STATUS_CODE_SLAVE_NEEDS_SAFEOP
	0x0024	ECAT_AL_STATUS_CODE_INVALID_INPUT_MAPPING
	0x0025	ECAT_AL_STATUS_CODE_INVALID_OUTPUT_MAPPING
	0x0026	ECAT_AL_STATUS_CODE_INCONSISTENT_SETTINGS
	0x002C	ECAT_AL_STATUS_CODE_FATAL_SYNC_ERROR
	0x0030	ECAT_AL_STATUS_CODE_DC_INVALID_SYNC_CONFIG
	0x0031	ECAT_AL_STATUS_CODE_DC_INVALID_LATCH_CONFIG
	0x0032	ECAT_AL_STATUS_CODE_DC_PLL_SYNC_ERROR
	0x0033	ECAT_AL_STATUS_CODE_DC_SYNC_IO_ERROR
	0x0034	ECAT_AL_STATUS_CODE_DC_SYNC_MISSED
	0x0035	ECAT_AL_STATUS_CODE_INVALID_SYNC_CYCLE_TIME
	0x0036	ECAT_AL_STATUS_CODE_DC_SYNC0_CYCLE_TIME
	0x0037	ECAT_AL_STATUS_CODE_DC_SYNC1_CYCLE_TIME
	0xFE0A	ECAT_SLAVE_FLAGS_INIT_ERROR
	0xFE0B	ECAT_NO_STACK_INIT_ERROR

Fieldbus system	Value	Plaintext
	0xFE0C	ECAT_SSI_INIT_ERROR
	0xFFD8	INIT_ERR_EOE
	0xFFFC	ECAT_EOE_INIT_ERROR
	0xFFFD	ECAT_MBX_INIT_ERROR
	0xFFFE	ECAT_NODEID_INIT_ERROR
CANopen	0x0000	CAN301_ERROR_NO_ERROR
	0x0010	CAN301_ERROR_GENERIC_ERROR
	0x0020	CAN301_ERROR_CURRENT
	0x0030	CAN301_ERROR_VOLTAGE
	0x0040	CAN301_ERROR_TEMPERATURE
	0x0050	CAN301_ERROR_DEVICE_HARDWARE
	0x0060	CAN301_ERROR_DEVICE_SOFTWARE
	0x0070	CAN301_ERROR_ADDITIONAL_MODULES
	0x0080	CAN301_ERROR_MONITORING
	0x0090	CAN301_ERROR_EXTERNAL
	0x00F0	CAN301_ERROR_ADDITIONAL_FUNCTIONS
	0x00FF	CAN301_ERROR_DEVICE_SPECIFIC
	0xFE0A	CAN_SLAVE_FLAGS_INIT_ERROR
	0xFE0B	CAN_NO_STACK_INIT_ERROR
	0xFFCC	CAN_ERROR_HB_TIMEOUT
	0xFFCD	CAN_ERROR_NG_TIMEOUT
	0xFFFE	INIT_ERR_CAN_DRV
VARAN	0x0000	VARAN_NO_ERROR
	0xFFFE	VARAN_FPGA_INIT_ERROR
PROFINET	0x0000	PROFINET_NO_ERROR
	0x0001	PROFINET_ERROR_INIT
	0xFD12	PROFINET_PD_CFG_FLAGS_ERROR
	0xFD44	PROFINET_PD_CFG_ERROR
	0xFE0A	PROFINET_SLAVE_FLAGS_INIT_ERROR
	0xFE0B	PROFINET_NO_STACK_INIT_ERROR
	0xFE0C	PROFINET_MSG_QUEUE_INIT_ERROR
	0xFF38	PROFINET_MAC_ADDR_ERROR
	0xFFFC	PROFINET_MAC_ADDR_INV
	0xFFFE	PROFINET_CLIENT_INV
POWERLINK	0x0000	EPSG_DS301_ERR_NO_ERR
	0xFC0E	EPL_EPSG_MAP_INIT_ERROR
	0xFD44	EPL_PD_SIZE_ERROR
	0xFE0A	EPL_SLAVE_FLAGS_INIT_ERROR
	0xFE0B	EPL_NO_STACK_INIT_ERROR
	0xFE0C	EPL_MSG_QUEUE_INIT_ERROR
EtherNet/IP	0xFE0A	ENIP_SLAVE_FLAGS_INIT_ERROR
	0xFE0B	ENIP_NO_STACK_INIT_ERROR
	0xFE0C	ENIP_MSG_INIT_ERROR
	0xFF38	ENIP_MAC_ADDR_INIT_ERROR
	0xFFFE	ENIP_CLIENT_INV

15.6 Solutions for KEB specific fieldbus error codes

The following table describes KEB specific fieldbus errors. The listed solutions should be carried out in consultation with a KEB service employee. If other error codes occur in fb91, please check whether they are error codes of the respective fieldbus system. If this is not the case, please contact KEB Support.

Error code	Description	Solution
INIT_ERR_PD	PD mapping error, PD mapping is deactivated.	Check and reactivate the PD mapping, then perform a software reset.
INIT_ERR_INV_NODE_ID	Node address value is not supported by the selected fieldbus system.	Change node address value or active fieldbus system and perform a software reset.
INIT_ERR_FB_TYPE_INVALID	Selected fieldbus system is not supported in the current configuration.	Change fieldbus system or check connection to MRTE module (fb80). After changes software reset.
INIT_ERR_I2C	No access to I2C, detection of the MRTE module and reading of the node address is not possible.	Check the physical connection between the MRTE module and the control board. Perform a software reset.
INIT_ERR_BASE_DRV_NOT_ONLINE	Driver of the MRTE module does not receive a response from the MRTE module.	Check the connection between the MRTE module and the control board. (fb80)
INIT_ERR_NETX_BASE	Error in the initialization of the MRTE module driver.	Check the connection between the MRTE module and the control board. (fb80)
INIT_ERR_INV_NODE_ID	Error during configuration due to incorrect node address	Set node address value by fb100 or fb101 to a valid value for POWERLINK CN
XXX_NO_STACK_INIT_ERR	MRTE module file for selected fieldbus system is not in the file system	Check files in the file system via KEB FTP. Check MRTE module version via fb80.
XXX_MAC_ADDR_ERROR	Missing MAC addresses.	Add MAC addresses (fb103 – fb105).
XXX_SLAVE_FLAGS_INIT_ERR	Error in communication between software and MRTE module.	Check software version, contents of the file system and NetX version. Follow the start-up manual (chapter 5)
ECAT_DMA_ERROR	Error in the communication with the EtherCAT Core	Hardware error. Please contact your KEB support.
ECAT_DCSYNCMISSEDERROR	No process data received between two SYNCs	Checking the EtherCAT master configuration
ECAT_INVALID_SYNC_CYCLE_TIME	Specified cycle time is not supported by the slave	Checkis22 and fb19 (cycle time must be integer multiple of MidIRQ cycle)
CAN301_ERROR_GENERIC_ERROR	Special case: Is displayed when CAN is the active fieldbus system and an error has occurred in the drive.	Check ru01 and ru02.

15.7 The KEB default process data mapping

The KEB default process data mapping differs depending on the control type of the used device and the active fieldbus system.

The KEB default process data mapping is an empty, inactive process data mapping for devices of the control type P.

On the devices of the control type A for EtherCAT, PROFINET, EtherNet/IP and Modbus TCP the KEB default process data mapping contains the user parameters aa16 to aa23. All 8 user parameters are active.

The default mapping is not activated if CANopen is the active fieldbus system on the devices of control type A.

The KEB default process data mapping is an empty, inactive process data mapping if POWERLINK is the active fieldbus system on the devices of control type A.

Index	Subidx	Value	Function
0x1600	0	8	Number of mapped process output objects = 8
0x1600	1	0x29100020	Index = 0x2910, Subindex = 0x00, Length = 0x20 (32bit)
0x1600	2	0x29110020	Index = 0x2911, Subindex = 0x00, Length = 0x20 (32bit)
0x1600	3	0x29120020	Index = 0x2912, Subindex = 0x00, Length = 0x20 (32bit)
0x1600	4	0x29130020	Index = 0x2913, Subindex = 0x00, Length = 0x20 (32bit)
0x1600	5	0x29140020	Index = 0x2914, Subindex = 0x00, Length = 0x20 (32bit)
0x1600	6	0x29150020	Index = 0x2915, Subindex = 0x00, Length = 0x20 (32bit)
0x1600	7	0x29160020	Index = 0x2916, Subindex = 0x00, Length = 0x20 (32bit)
0x1600	8	0x29170020	Index = 0x2917, Subindex = 0x00, Length = 0x20 (32bit)
0x1A00	0	8	Number of mapped process output objects = 8
0x1A00	1	0x29100020	Index = 0x2910, Subindex = 0x00, Length = 0x20 (32bit)
0x1A00	2	0x29110020	Index = 0x2911, Subindex = 0x00, Length = 0x20 (32bit)
0x1A00	3	0x29120020	Index = 0x2912, Subindex = 0x00, Length = 0x20 (32bit)
0x1A00	4	0x29130020	Index = 0x2913, Subindex = 0x00, Length = 0x20 (32bit)
0x1A00	5	0x29140020	Index = 0x2914, Subindex = 0x00, Length = 0x20 (32bit)
0x1A00	6	0x29150020	Index = 0x2915, Subindex = 0x00, Length = 0x20 (32bit)
0x1A00	7	0x29160020	Index = 0x2916, Subindex = 0x00, Length = 0x20 (32bit)
0x1A00	8	0x29170020	Index = 0x2917, Subindex = 0x00, Length = 0x20 (32bit)

15.8 PROFINET Parameter Request

The Request Reference (0x01-0xFF) of the parameter request is a sequential number to differentiate between the various requests.

The parameter values (Values) are only contained in a request with request ID = 2.

The Axis-No. always corresponds to value 0, since PROFINET represents a 1:1 gateway for the x6 devices.

Only a maximum of 39 parameters can be read / written via a request. Therefore, a minimum of 1 and a maximum of 39 can be specified for the No. of parameters .

The No. of elements as well as the No. of values always correspond to value 1.

The required object for writing / reading is addressed via Parameter number (PNU) and Subindex. The addressing corresponds to the addressing in the KEB object directory. The Parameter number (PNU) corresponds to the KEB parameter address and Subindex to the KEB subindex.

When reading all KEB parameters on the x6 the format = 4 is returned. Accordingly, the PROFINET master can also specify the format = 4 when writing all parameters.

Parameter-Request			
Block definition	Byte n	Byte n+1	n
Request Header	Request Reference	Request ID	0
	Axis-No.	No. of parameters = n	2
1st parameter address	Attribute	No. of elements	4
	Parameter Number (PNU)		6
	Subindex		8
:	:		:
n th parameter address	Attribute	No. of elements	4 + 6 x (n-1)
	Paramater number (PNU)		
	Subindex		
1st parameter value(s)	Format	No. of values	4 + 6 x n
	Values		
:	:		:
n th parameter value(s)			

Request IDs	
Value	Meaning
0x01	Request parameter (read)

0x02	Change parameter (write)

Attribute	
Value	Meaning
0x10	Value
0x20	Description (not supported here)
0x30	Text (not supported here)

Format	
Value	Meaning
1	Boolean
2	Integer8
3	Integer16
4	Integer32
5	Unsigned8
6	Unsigned16
7	Unsigned32
0x41	Byte
0x42	Word
0x43	DWord
0x44	Error

15.9 PROFINET Parameter Response

If an error occurred during read access to a parameter, the Response ID = 0x81 is set. Furthermore, the format = 0x44 (Error) is set for this parameter value and the error value (Value) is set as a 16-bit value.

The parameter values are only contained in the response if Response ID = 1 or 0x81 or 0x82. Error-free accesses are listed in a negative write response (Response-ID = 0x82) with Format = 0x40 (Zero) and No of Values = 0.

Parameter response:			
Block definition	Byte n	Byte n+1	n
Response Header	Request Reference mirrored	Response ID	0
	Axis-No. mirrored	No. of parameters = n	2
1st parameter value(s)	Format	No. of values	4 + 6 x n
	Values		
:	:		:
n th parameter value(s)			

Response ID	
Value	Meaning
0x01	Request parameter ok (read)
0x02	Change parameter ok (write)
0x81	Request parameter with error (read)
0x82	Change parameter with error (write)

Error value (Value)	
Value	Meaning
0x11	Timeout when accessing the parameter or drive controller busy
0x14	Drive controller busy
0x00	Invalid parameter address or password
0x17	Data invalid
0x65	Error in internal communication (BCC error)
0x66	Error in internal communication (invalid service or invalid operation)
0x67	Invalid password

0x68	Error in internal communication (invalid telegram)
0x69	Error in internal communication (parity error)
0x03	Invalid parameter set (subindex)
0x6B	Error in internal communication (invalid operation)

15.10 Ethernet/IP™ Status Codes

The following table lists the general status messages which can be contained in a response message. The "extended code field" is available for the future description of general status messages.

Status codes	Name of the status	Description
0x00, 0	Success	The service was successfully executed by the object.
0x01, 1	Connection failure	The connection to the device has failed.
0x02, 2	Resource unavailable	The required resources for the requested service are not available.
0x03, 3	Invalid parameter value	The parameter value of the request is not supported.
0x04, 4	Path segment error	An error occurred during the evaluation of the communication path.
0x05, 5	Path destination unknown	The communication path references an unknown object class, instance or unknown attribute.
0x06, 6	Partial transfer	Only a part of the data could be transferred.
0x07, 7	Connection lost	The connection to the device has been lost.
0x08, 8	Service not supported	The requested service is not supported by the selected object.
0x09, 9	Invalid attribute value	The selected attribute is not supported by the object.
0x0A, 10	Attribute list error	An attribute in the Get_Attribute_List or Set_Attribute_List has a non-zero status.
0x0B, 11	Already in requested state	The object is already in the requested state.
0x0C, 12	Object state conflict	The object can not execute the requested service in the current state.
0x0D, 13	Object already exists	The requested creation of the object has failed, since the object already exists.
0x0E, 14	Attribute not settable	The selected attribute cannot be changed.
0x0F, 15	Privelage violation	Authorization check failed
0x10, 16	Device state conflict	The current state of the device prohibits the requested service.
0x11, 17	Reply data too large	The response memory is too small for the data to be sent.
0x12, 18	Fragmentation of primitive	The selected service attempts to perform an operation that fragments a primitive data type.
0x13, 19	Not enough data	The data provided by the service are not enough.
0x14, 20	Attribute not supported	The selected attribute is not supported.
0x15, 21	Too much data	The service has provided too much data.
0x16, 22	Object does not exist	The selected object does not exist.
0x17, 23	Service fragmentation	Service fragmentation is not active.
0x18, 24	No stored attribute data	The attribute of the selected object was not stored.
0x19, 25	Store operation failure	The attribute of the selected object could not be stored.
0x1A, 26	Routing failure, request packet too large	The selected service request packet was too large to be sent over the network. The routing service has aborted the transmission.
0x1B, 27	Routing failure, response packet too large	The selected service response packet was too large to be sent over the network. The routing service has aborted the transmission.
0x1C, 28	Missing attribute list entry data	The selected service does not have a suitable attribute to perform the selected behaviour.
0x1D, 29	Invalid attribute value list	The selected service returns the list of invalid attributes with

Status codes	Name of the status	Description
		associated status information.
0x1E, 30	Embedded service error	An embedded service has triggered an error.
0x1F, 31	Vendor specific error	A vendor-specific error has occurred. This error occurs if no other general error code can be assigned to the behavior.
0x20, 32	Invalid parameter	A parameter belonging to the request is invalid. The parameter does not meet the requirements of the specification or application.
0x21, 33	Write-once already written	An attempt was made to write-once already written.
0x22, 34	Invalid reply	An invalid reply was received. The received service code is non-compliant with the sent service code or the reply is less than the minimum expected size.
0x23, 35	Buffer overflow	The received message is larger than the corresponding memory. The message has been rejected.
0x24, 36	Invalid message format	The format of the received message is not supported.
0x25, 37	Key failure in path	The key segment contained as the first segment in the path does not correspond to the target module.
0x26, 38	Path size invalid	The size of the received routing path is too small or too much routing data is included.
0x27, 39	Unexpected attribute	An attempt has been made to set an attribute that cannot be set at this time.
0x28, 40	Invalid Member ID	The received Member ID does not exist.
0x29, 41	Member not settable	A request to modify a non-modifiable member was received.
0x2A-0xCF, 42-207	Reserved	Reserved for future extensions.
0xD0-0xFF, 208-255	Reserved	Reserved for object class-specific error codes.

15.11 Revision history

Version	Chapter	Change	Date
00		First version created	07.02.2024
	13.2	Adaptation fb110	20.02.2024
	5.3, 5.6	Modbus process data extension	20.02.2024
	14.7.1	Note on synchronisation added	22.02.2024
	5.5.2, 5.5.3, 15.2, 15.4	Modbus state and LED flashing pattern added	13.03.2024
	14.9	Example of CAN-Cross configuration added	18.04.2024
	11.1.2, 11.1.3	Additional notes on fb113 added	03.05.2024
	5.2.2	Correction of the baud rates in fb116	05.06.2024
	7.1	Correction of the description of software version 0x1009	18.06.2024
	3.2, 5.1.2	Explanation of baud and bit rate added	18.06.2024
	5.3.2	Modbus process data size warning added	18.06.2024
	5.2.2	Modbus process data size warning added	09.07.2024
	5.7.1	Supplement borders fb10	11.07.2024
	6.5.3	Addition of step width restriction inhibit time	11.07.2024
	5.3.1	Addition of default shop fb81.1	16.07.2024
	10	POWERLINK IP configuration	30.07.2024
	5.3.2	Clarification of minimum cycle time EtherCAT x6P	13.08.2024
	5.5.1	Addition of default values for the IP configuration	15.08.2024
	5.5.1	Supplement special case x6P PROFINET IP configuration	15.10.2024
	5.8	Revision of the field watchdog	18.10.2024
	6.4.1	Execution of the bit meaning for CANopen SDO telegrams	21.11.2024
	5.5.1	Added warning for same IP configuration	26.11.2024
	5.8	Extended description of the pn22	12.12.2024
01			

Benelux | KEB Automation Benelux
Dreef 4 – box 4 1703 Dilbeek Belgien
Tel: +32 2 447 8580
E-Mail: info.benelux@keb.de Internet: www.keb.de

Brazil | KEB SOUTH AMERICA - Regional Manager
Rua Dr. Omar Pacheco Souza Riberio, 70
CEP 13569-430 Portal do Sol, São Carlos Brasilien Tel: +55 16 31161294 E-Mail: roberto.arias@keb.de

China | KEB Power Transmission Technology Co. Ltd.
No. 435 QianPu Road Chedun Town Songjiang District
201611 Shanghai P.R. China
Tel: +86 21 37746688 Fax: +86 21 37746600
E-Mail: info@keb.cn Internet: www.keb.cn

Germany | **Geared Motors**
KEB Antriebstechnik GmbH
Wildbacher Straße 5 08289 Schneeberg Germany
Telefon +49 3772 67-0 Telefax +49 3772 67-281
Internet: www.keb-drive.de E-Mail: info@keb-drive.de

France | Société Française KEB SASU
Z.I. de la Croix St. Nicolas 14, rue Gustave Eiffel
94510 La Queue en Brie France
Tel: +33 149620101 Fax: +33 145767495
E-Mail: info@keb.fr Internet: www.keb.fr

United Kingdom | KEB (UK) Ltd.
5 Morris Close Park Farm Indusrial Estate
Wellingborough, Northants, NN8 6 XF United Kingdom
Tel: +44 1933 402220 Fax: +44 1933 400724
E-Mail: info@keb.co.uk Internet: www.keb.co.uk

Italia | KEB Italia S.r.l. Unipersonale
Via Newton, 2 20019 Settimo Milanese (Milano) Italia
Tel: +39 02 3353531 Fax: +39 02 33500790
E-Mail: info@keb.it Internet: www.keb.it

Japan | KEB Japan Ltd.
15 - 16, 2 - Chome, Takanawa Minato-ku
Tokyo 108 - 0074 Japan
Tel: +81 33 445-8515 Fax: +81 33 445-8215
E-Mail: info@keb.jp Internet: www.keb.jp

Austria | KEB Automation GmbH
Ritzstraße 8 4614 Marchtrenk Austria
Tel: +43 7243 53586-0 Fax: +43 7243 53586-21
E-Mail: info@keb.at Internet: www.keb.at

Poland | KEB Automation KG
Tel: +48 60407727
E-Mail: roman.trinczek@keb.de Internet: www.keb.de

Switzerland | KEB Automation KG
Witzbergstraße 24 8330 Pfäffikon/ZH Switzerland
Tel: +41 43 2886060 Fax: +41 43 2886088
E-Mail: info@keb.ch Internet: www.keb.ch

Spain | KEB Automation KG
c / Mitjer, Nave 8 - Pol. Ind. LA MASIA
08798 Sant Cugat Sesgarrigues (Barcelona) Spain
Tel: +34 93 8970268 Fax: +34 93 8992035
E-Mail: vb.espana@keb.de

Republic of Korea | KEB Automation KG
Room 1709, 415 Missy 2000 725 Su Seo Dong
Gangnam Gu 135- 757 Seoul Republic of Korea
Tel: +82 2 6253 6771 Fax: +82 2 6253 6770
E-Mail: vb.korea@keb.de

Czech Republic | KEB Automation GmbH
Videnska 188/119b 61900 Brno Czech Republic
Tel: +420 544 212 008
E-Mail: info@keb.cz Internet: www.keb.cz

United States | KEB America, Inc
5100 Valley Industrial Blvd. South Shakopee, MN 55379 United States
Tel: +1 952 2241400 Fax: +1 952 2241499
E-Mail: info@kebamerica.com Internet: www.kebamerica.com

WEITERE KEB PARTNER WELTWEIT:

…www.keb.de/de/contact/contact-worldwide

Automation **mit Drive**

www.keb.de

KEB Automation KG Südstsraße 38 32683 Barntrup Tel. +49 5263 401-0 E-Mail: info@keb.de