# BETRIEBSANLEITUNG
# INSTRUCTION MANUAL


CAN in AUTOMATION

**KEB COMBICOM**

**CANopen-ANSCHALTUNG**
**CANopen-INTERFACE**

00.58.00A-K101

KEB
ANTRIEBSTECHNIK

09/99

Das in dieser Betriebsanleitung verwendete Pictogramm entspicht folgender Bedeutung:

**Achtung,
Unbedingt
beachten**

In dieser Betriebsanleitung befindet sich auf Seite D - 42 ein Literaturverzeichnis, in dem Nachschlagewerke aufgeführt sind, die bestimmte Normen und Aussagen in dieser Anleitung erläutern. An den entsprechenden Textstellen, befinden sich mit eckigen Klammern [ ] gekennzeichnete Ziffern.



The pictographs used in this manual mean:

**Attention,
observe at
all costs**

On Page GB - 42 in this Manual you can find a literature list which contains referance books. Standards and statements are described in this reference books. At the corresponding text you can find digits in square brackets [ ].

# Table of Contents

# 1 General

This manual as well as the hardware and software are developments of the Karl E. Brinkmann GmbH. Errors and omissions excepted! Karl E. Brinkmann GmbH prepared the documents, software and hardware to the best of their knowledge, however, no guarantee is given that the specifications will bring the user the efficiency aimed at. The Karl E. Brinkmann GmbH reserves the right to change the specifications without obligation. 

# 2 Ordering Information

| | |
|---|---|
| This Instruction Manual | : 00.58.00A-K101 |
| F4-CAN-Operator | : 00.F4.010-5009<br>(can be inserted in all KEB-F4-devices) |
| F4-CAN-Submounted Card | : 00.58.035-0009<br>(can be used as of housing size G) |

# 3 F4-CAN-Interface

KEB-Antriebstechnik develops, produces and sells static frequency inverters worldwide in the industrial power range. Inverter type **F4** can be optionally equipped with a **CAN**-interface (**C**ontroller-**A**rea-**N**etwork). This is an intelligent interface which controls the access via CAN to the parameters in the inverter.

To program the KEB F4 inverter with CAN you need the respective manual for the inverter control [1] in addition to this manual.

## 3.1 The Hardware

F4-CAN-interface is available in 2 versions:

1. Operator with CAN-interface. This is inserted in the inverter housing and fits in all KEB-F4 inverters.
2. Submounted card. This can be inserted in all KEB F4 inverters as of housing size G.

The difference between the 2 CAN-interfaces is the connection between CAN-interface and the actual inverter-control. With the Operator the CAN-interface uses the internal serial interface of the inverter to relay the data from CAN-Bus to the control. The F4-CAN submounted card communicates with the inverter control via a parallel connection through dual-ported-memory.

### 3.1.1 The Interface of KEB-F4-CAN-Interface

The CAN-Bus has one D-SUB-9 pole male connector and a D-SUB-9 pole female connector (in accordance with DIN41652 Part 1) for the submounted card.
The F4-CAN-Operator only has one D-SUB-9 pole male connector (in accordance with DIN41652 Part 1).
For more information about the assignment of the CAN-connector see [3] in the Literature index.

| Pin | Signal | Description |
|---|---|---|
| 1 | - | Reserved |
| **2** | **CAN_L** | **CAN_L bus line (dominant low)** |
| 3 | CAN_GND | Not connected here |
| 4 | - | Reserved |
| 5 | (CAN_SHLD) | Not connected here |
| 6 | (GND) | Not connected here |
| **7** | **CAN_H** | **CAN_H bus line (dominant high)** |
| 8 | - | Reserved |
| 9 | (CAN_V+) | Not connected here |

**Transmission level of CAN :** in accordance with ISO/DIS 11898, ISO-High Speed.
**Transmission speed of CAN :** Adjustable via CAN (10, 20, 50, 100, 125, 250, 500 Kbit/s) .
**Isolation :** Safe isolation in accordance with VDE0160.
**Bus connection :** 124 W , must be done externally e.g. in the connector (between Pins 2 and 7).

# 4 Fundamentals of CAN-BUS

The **CAN**(**C**ontroller-**A**rea-**N**etwork)-BUS system and the terms often used for this system are described in this chapter.
CAN is a **Multi-Master-System**. This means every user can access BUS and send telegrams. In order to prevent problems when two users access BUS at the same time, the CAN-BUS has an arbitration phase which determines who may continue to send his telegram. When there is a conflict in accessing BUS the user with the lowest telegram number (identifier) has priority. This user then can completely send his telegram without repeating the first part. All other users go into receiving status and stop sending their telegram. The available telegram numbers in the CAN version 2.0A are limited to 2032 identifiers (0...2031).
CAN-telegrams can have a maximum of 8 byte user data.
When speaking of the **logical CAN-Master**, the CAN-user is meant who manages the control of the entire CAN-System. Even though there are only Masters in CAN it usually is the case that one or more users have the control. In connection with this the KEB inverter is seen as the command receiver (logical slave).

# 5 Function

The CAN protocol is uniformly standardized up to layer 2. The complete processing of the protocol is done by the CAN-controller. The **C**AN **i**n **A**utomation Association **(CiA)** adopted a standard for the higher protocol level called **C**AN **A**pplication **L**ayer **(CAL)**. Based on this standard the **'CAL-based Communication Profile'** (CiA,DS301) was published in September 1995. This standard is the basis for all **CANopen**-device-profiles. In the **'CAL-based Communication Profile'** certain functions of the CAL-Standard are selected. The communication profile defines the **Minimum Capability Device.** This is the minimum required functionality that a CANopen-node must provide. The KEB CAN-Interface realizes such a Minimum Capability Device.

An important function in every CAN network is the distribution of telegram numbers (identifiers). The numbers are limited to 2032 in CAN V2.0 A. A special process for this is defined in the CAL-standard that realizes this identifier distribution dynamically through its own protocol (DBT = Distributor). This relatively extensive procedure for allocating identifiers is not required and not integrated in the KEB-CAN-Interface. Thus a simple process is specified in the communication profile for agreement of the identifier allocation. This process is supported by the KEB-CAN-Interface and is as follows:
Every inverter has a CAN-address (the **Module_Id**). This results from the parameter **inverter address**, which every KEB-inverter supports (see manual of the KEB-inverter used):

| Module_Id = Inverter Address(ud.06) + 1 |
|---|

After delivery all KEB-inverters have the inverter address = 1. If several KEB inverters should be networked via CAN, they must have different inverter addresses. This is done via CAN-Bus. Observe that only one KEB-inverter may be connected to CAN-Bus. Only after all inverters have different addresses they all can cooperate on a CAN-network.
Another possibility is to enter the inverter address via keyboard. To adjust the inverter address the CAN-Operator must be replaced by an Interface-Operator (switch off device), in order to enter the inverter address.

Every inverter is assigned 4 identifiers. The 4 identifiers are assigned to 3 communication services (or functions).
Via one identifier the logical CAN-BUS-Master requests the reading (upload) or writing (download) of any parameter in the KEB-inverter (Request-Identifier)**.**
Another identifier is reserved for the respective answer (confirmation) from the inverter
(Response-Identifier)**.** This mechanism for request and response is called **confirmed service**. The communication profile calls this function **Service-Data-Object (SDO)** and the two parts are SDO(rx) und SDO(tx).

| |
|---|
| SDO(rx) = Request-Identifier = 1536+Module_Id = 1537+inverter addr.(ud.06) |
| SDO(tx) = Response-Identifier = 1408+Module_Id = 1409+inverter addr.(ud.06) |

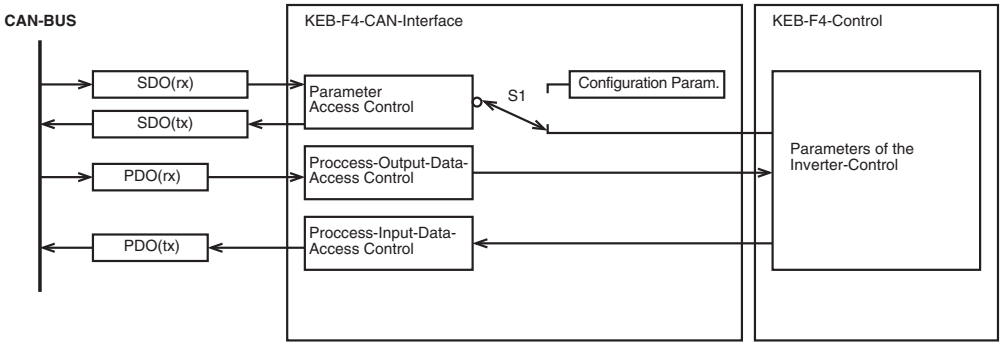
Example:
Inverter address = 30 ==> Read/Write request via Identifier = 1567
==> Read/Write confirmation via Identifier = 1439

The 3rd identifier enters data in the inverter which is not addressed nor confirmed. This identifier is called **OUT-Identifier**, because it goes from the Master to the inverter.
The inverter uses the 4th identifier to provide new data unaddressed and unconfirmed to the logical CAN-Master **(IN-Identifier)**.

These functions are characterized by the communication profile as **Process-Data-Object (PDO)**. Both identifiers are called PDO(rx) and PDO(tx) (as seen from the device). The distribution of the identifier for the PDO can be changed and inquired by the confirmed service (SDO). The standard setting is as follows:

| |
|---|
| PDO(rx) = Out-Identifier = 512 + Module_Id = 513 + inverter address |
| PDO(tx) = IN-Identifier = 384 + Module_Id = 385 + inverter address |

The CAN-Interface controls the data flow from CAN (SDO(rx) and PDO(rx)) to the inverter control and also from the inverter to CAN-BUS (SDO(tx) and PDO(tx)):




The diagram above shows the function of CAN-Interface. The position of switch S1 is determined solely by the parameter-address (16 Bit Index plus 8 Bit Subindex) in the CAN-SDO(rx)-telegram. In a certain index range the so-called configuration data of CAN-interface is found. These parameters determine the behaviour of the CAN-interface and thus are realized in this. Access to parameters in the index range 2000(hex) to 5EFF(hex) are transferred as read/write services to the inverter control.

## 5.1 Process-Data Mapping

The type of determination of the destination for that data in the PDO(rx)-telegram and the source for the data in PDO(tx)-telegram complies with the regulations of the CANopen-Communication Profile (see [13] in literature index). For every data direction a complex object (parameter) defines the PDO-mapping. Another object determines the communication definition (PDO communication parameter) for each data direction. See parameter description from:

- **1st receive PDO Mapping**
- **1st transmit PDO Mapping**
- **1st receive PDO Parameter**
- **1st transmit PDO Parameter**

in this manual

## 5.2 Extended Initialization Phase with CAN-F4-Operator

After Power-On the CAN-Operator carries out an additional initialization phase in which the following is done:

1. Adjust the transmission speed between Operator and inverter control to 9600 Bit/s.
2. Determine the current adjusted inverter address in the inverter-control.
3. Step by step increase the transmission speed between CAN-Operator and the inverter control to the maximum when AUTO_ANSIBAUD = ON. Otherwise the transmission speed is set onto the value in AUTO_ANSIBAUD.

The total amount of time this initialization phase takes is about 6 seconds (with unchanged inverter addresses) and a maximum of 12 seconds (e.g. when changing the inverter address from 0 to 239).

## 5.3 Automatic Adaption of the Process Data Assignment to the Inverter Type (from SW V2.5)

The standard process data assignment can not be used for every inverter type, because not all inverter types support the DRIVECOM-profile parameters. From software version 2.5 the initialization is extended by one step, so the user has the possibility to operate exclusively with the PDO-communication via CAN. The CANOPEN-interface checks if the stored process data assignment is qualified for the connected KEB inverter. If not the CANOPEN-interface searchs for a qualified process data assignment. The sequence is described as follows:

1. First of all the inverter type must be defined. There are three different inverter types :
   – Servo (S4) and F4-F-frequency inverter,
   – F4-C-frequency inverter and
   – F4-S-frequency inverter.

2. All parameters of the stored process data assignment (for PDO(tx) and PDO(rx)) are read by the frequency inverter. This assignment will be invalid if the inverter doesn't recognize one of these parameters. If all parameters could be read, the CAN-interface accepts this process data assignment and the check is finished.

3. If the stored process data assignment is not qualified for this inverter type, one of the three possible standard process data assignments will be selected and accepted for the respective inverter type. But this process data assignment will **not be stored.**

The following standard-process data assignments with 2 parameters for the PDO(rx) and the PDO(tx) are defined:

| Inverter Type | PDO(rx) CAN ⇒ KEB | | PDO(tx) KEB ⇒ CAN | |
|---|---|---|---|---|
| | 1. Parameter | 2.Parameter | 1.Parameter | 2.Parameter |
| F4-C | Pr.06 | Pr.08 | Pr.07 | Pr.09 |
| S4 od. F4-F | SP.03 | SP.01 | ru.00 | ru.01 |
| F4-S | oP.03 | oP.01 | ru.00 | ru.03 |

The automatic adaption of the PD-assignment has the consequence that a CANOPEN-Interface takes part in the CAN-communication of different KEB-inverter types with different PD-assignment. This is very important for the SW of the controlling CAN-Master, because this CAN-Master must know all process data. Of course it is possible to read out the active process data mapping via SDO-communication. These parameters are called **1st receive PDO Mapping** and **1st transmit PDO Mapping**. The active PD-assignment can be permanently stored by writing of value 255d on parameter **Save_PD_Map_And_Para**.
With reading the same parameter you get the answer if the actual adjustments are correspond or not to the stored adjustments (value = 255d).
To make a clear statement which PD-assignment is active without reading of this via SDO-communication, the user must know the inverter type which operates with the CANOPEN-Interface and also the stored PD-assignment of the CANOPEN-Interface . Attention: If the PD-assignment is not changed or stored by the user via SDO-communication the correspond PD-assignment is active (see table) for each inverter type.

Pay attention to the following two examples :

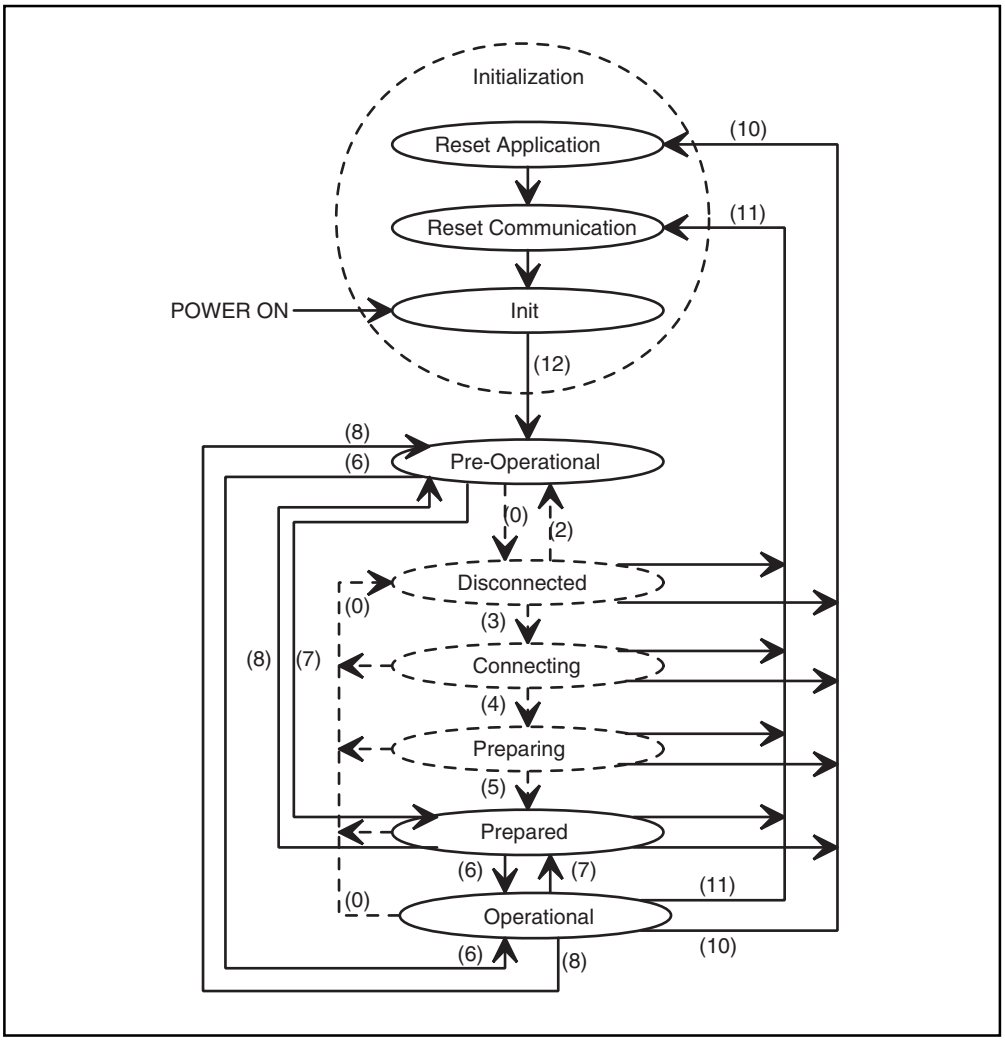
**Example 1: F4-C ⇒ S4/F4-F:**
The standard-PD-assignment for a F4-C is stored in the CAN-Interface (delivery state). This CAN-Interface shall operate with a KEB-Servo (S4) or KEB-F4-F:
After initialization the CAN-Interface starts with the Standard **S4/F4-F**-PD-assignment but cannot store these.

**Example 2: F4-S ⇒ F4-C:**
The standard-PD-assignment for a F4-S is stored in the CAN-Interface. This CAN-Interface shall operate with a KEB-F4-C:
After initialization the CAN-Interface starts with the Standard **F4-S**-PD-assignment (not as expected with the standard F4-C-assignment). The reason herefore is that the F4-C supports all parameters of the standard-F4-S-assignment and these parameters are also valid for this inverter.

## 5.4 CANopen Bootup-Sequence

The CANopen Communication Profile (see literature index [13]) defines a start phase that every CAN-slave must go through until the actual communication is activated.
The KEB-CAN-Interface automatically goes into **Pre-Operational** status after the initialization phase. In this status the communication is activated via the SDO(rx) and SDO(tx) with the services Domain Download (write parameter) and Domain Upload (read parameter). The process-data communication is inactive in this status. This is released by the NMT-Command Start_Remote_Node() (see below). After this command the KEB-CANopen-Slave switches its operation state to **Operational.** In this state the communication is completely activated. Certain CAN-nodes are addressed by the NMT-protocol by the **Node-Id.** This results as with the Module-Id from the value of the parameter inverter address :

| Node-Id = Module-Id = Inverter Address (ud.06) + 1 |
|---|




The KEB-CANopen-Interface realizes the following transitions (characterized by a straight line):

**6: Start_Remote_Node()**

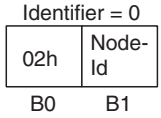
CAN-telegram:

| Identifier = 0 | |
|---|---|
| 01h | Node-Id |
| B0 | B1 |

with Node_Id = 0 (all NMT-Slaves are addressed) or
with Node_Id = Inverter-Address + 1 (only 1 inverter is addressed)

**7: Stop_Remote_Node()**

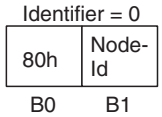
CAN-telegram:




with Node_Id = 0 (all NMT-Slaves are addressed) or
with Node_Id = Inverter-Address + 1 (only 1 inverter is addressed)

**8: Enter_Pre-Operational_State()**

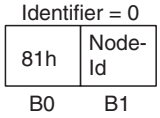
CAN-telegram:




with Node_Id = 0 (all NMT-Slaves are addressed) or
with Node_Id = Inverter-Address + 1 (only 1 inverter is addressed)

**10: Reset_Node():** When this function is carried out a Software-Reset is done in the KEB-CAN-Interface.

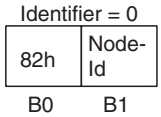
CAN-telegram:




with Node_Id = 0 (all NMT-Slaves are addressed) or
with Node_Id = Inverter-Address + 1 (only 1 inverter is addressed)

**11: Reset_Communication():** Function as with Reset_Node().

CAN-telegram:




with Node_Id = 0 (all NMT-Slaves are addressed) or
with Node_Id = Inverter-Address + 1 (only 1 inverter is addressed)

**12: Enter Pre-Operational automatically():** s.o.

## 5.5 Bootup-Message

From SW-version 2.2 the KEB-CAN-node gives a bootup-message, when after POWER ON the initialization period has finnished. This is a telegram on identifier = 128 + Node-Id with data length = 0.

## 5.6 Node-Guarding

CAL (**C**AN **A**pplication **L**ayer) defines a protocol trough which one can-node can require the actual node-state of a certain node. This is part of the network management functionality(NMT) of a can-node and is called Node-Guarding. This is supported by the KEB-can-node. The node-guarding-request is done by sending a remote frame with the node-guarding-identifier. The response is returned as data-telegram with 1 byte of data on the same identifier. This byte containes the actual node-state of that node plus one bit (MSBit) which is altered from one to the next message. Each node has its own node-guarding-identifier. For minimum capability devices this identifier directly results from the node-id as follows:

Node-Guarding-Identifier = 1792 + Node-Id = 1793 + Inverter Addr.(ud.06)

| Value of node-state | Meaning |
|---|---|
| 1 | DISCONNECTED |
| 2 | CONNECTING |
| 3 | PREPARING |
| 4 | PREPARED |
| 5 | OPERATIONAL |
| 127d | PRE_OPERATIONAL |

## 5.7 Life-Guarding

Life-Guarding describes the monitoring function of the Node-Guarding. In this case each NMT-slave-node monitors if the node was addressed from the NMT-master to the Node-Guarding-Identifier during an adjustable time. From software version 2.2 Life-Guarding is realized in the KEB-CAN-interface. The monitoring time is adjusted with two CANOPEN-parameters (guard_time (Index=100Ch) and life_time_factor (Index=100Dh)). The **guard_time** is preset in milliseconds. The monitoring time results from multiplication of **life_time_factor** with guard_time.

**Example: guard_time = 1000d, life_time_factor = 5 ==>**
**monitoring time = 5s.**

Life-Guarding is active if the product of guard_time and life_time_factor is unequal zero. Life-Guarding is not active in the standard adjustment, because guard_time = life_time_factor = 0. The function which is triggered when the monitoring time was exceeded without a receipt of a node-guarding-telegram can be adjusted here via parameter **Func_Life_Guard_Tout**. This function is realized as single writing access to the inverter control. With parameter Func_Life_Guard_Tout the index(parameter address+2000h) and the parameter value are defined, which are written when life-guarding to the inverter control is active. For the single writing cycle the KEB-node is switched into **PRE_OPERATIONAL** state if the monitoring time was exceeded. SDO-communication is activated here, but not PDO-communication. With the NMT-command Start_Remot_Node the PDO-communication can be activated again.
For a detailed description see chapter configuration parameter of this manual.

## 5.8 CAN-Communication Monitoring

Additionally to the Life-Guarding a general CAN-communication-monitoring can be activated from SW-version 2.3 When this function is activated via parameter **CAN_Tout_Time** the KEB-CAN-interface checks if in a preset time cycle telegrams are received via CAN. If the time was exceeded without a receipt of a:
SYNC-telegram ,
data telegram on OUT-identifier,
data telegram on Request-identifier,
data telegram on identifier 2026d (NMT-service specification) or a
data telegram on identifier 0 (NMT-Start/Stop-commands)
the KEB-CAN-node fullfills the same function as operation with **Life-Guarding**.
See chapter configuration parameter of this manual for coding parameter CAN_Tout_Time.

## 5.9 Communication Monitoring Between CAN-Interface and Control

When the CAN-Interface does not receive an answer 5 consecutive times from the inverter control, then the complete software is re-initialized by the CAN-interface. During this time **NO COMMUNICATION IS POSSIBLE VIA CAN**.

# 6 Coding of the Data in the 4 Types of CAN Telegrams

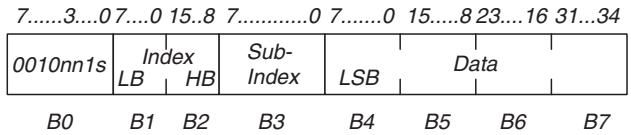
## 6.1 SDO(rx)-Telegram

Using this telegram the logical CAN-Master can inquire (read) or change (write) the value of one parameter. In the communication profile a write-service is called a **Domain Download** and a read-service a **Domain Upload**. The KEB-CAN-Interface only supports the expedited form of these services, so that only one telegram is exchanged for the service request and another for the service confirmation between logical CAN-Master and the KEB-CAN-Interface.
The parameter is addressed by the unsigned 16-Bit-Index and 8-Bit-Subindex. The parameters of the inverter control lie in the index range 2000(hex) to 5EFF(hex). The CAN-Index results from the parameter address (see parameter description of the inverter control used) by addition with the Offset 2000(hex):

| CAN-Index = KEB-Parameter-Adress + 2000(hex). |
|---|

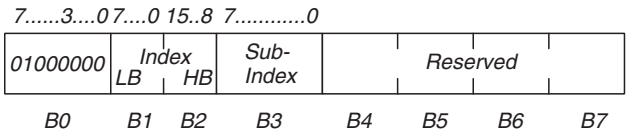
The subindex is used as an additional addressing for complex parameters. For example the CANopen-Object **1st receive PDO Parameter.inhibit-time is** adressed via Index = 1400(hex) and Subindex = 3.

### 6.1.1 Initiate Domain Download Request (Write Request of the Master)




**nn**: only valid, when s==1: contains the number of bytes of the datafield which does not contain data.
**s:** When = 1, then contains nn the number of bytes in the data field which do not contain data.
Otherwise no display of the data length in nn.
**Index:** 16-Bit (unsigned) addressing of the parameter (see above).
**Subindex:** 8-Bit (unsigned) sub-address for complex parameters
**Data:** data to be transmtted. The LSbyte is transmitted first.

### 6.1.2 Initiate Domain Upload Request (Read Request of the Master)

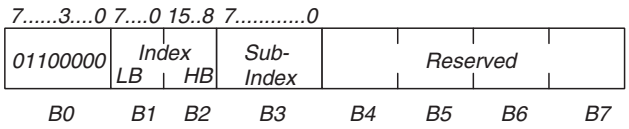



**Index:** 16-Bit (unsigned) addressing of the parameter (see above).
**Subindex:** 8-Bit (unsigned) sub-addressing for complex parameters.

## 6.2 SDO(tx)-Telegramm

### 6.2.1 Initiate Domain Download Response (Write Confirmation from Inverter)

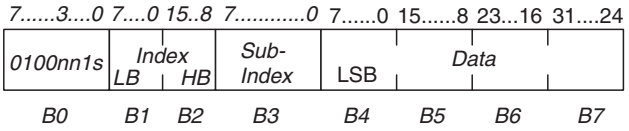
This answer is sent from the KEB-CAN-Interface when the requested write service could be completed without an error.




**Index:** see above
**Subindex:** see above

### 6.2.2 Initiate Domain Upload Response (Read Confirmation from Inverter)

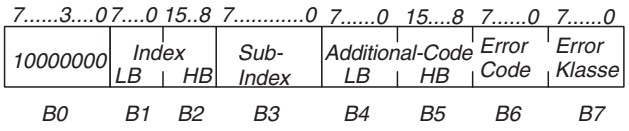
This answer is sent by the KEB-CAN-Interface when the requested read service could be completed without an error.




**nn:** see above
**s:** see above
**Index:** see above
**Subindex:** see above
**Data:** see above

### 6.2.3 Abort Domain Transfer (Error Answer from the Inverter)

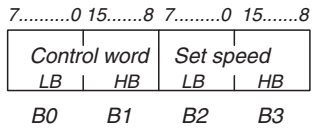
The KEB-CAN-Interface sends this answer when the requested read/ write services cannot be completed. In this case an error description is supplied.



| Error Class | Error Code | Additio-nal Code | Significance |
|---|---|---|---|
| 6 | 4 | 0 | Invalid parameter address (Index). |
| 8 | 0 | 30h | Invalid parameter value. |
| 6 | 1 | 0 | Invalid write access to Read-Only parameter. |
| 6 | 6 | 0 | Timeout between CAN-Interface and inverter control appears. |
| 8 | 0 | 22h | Inverter busy or password invalid. Due to the operating condition the inverter control cannot complete the service at this point. |
| 6 | 7 | 0 | Invalid type. The inverter control does not support the desired type. |
| 6 | 9 | 0 | Invalid Subindex. |
| 8 | 0 | 42h | Invalid PD-mapping, number of entries is larger than 8. |
| 8 | 0 | 41h | Invalid PD-mapping, object cannot be mapped on the process data. |
| 8 | 0 | 47h | Invalid PD-mapping, object length from the inverter is not supported or the set selection via subindex is not supported by the inverter. |
| 8 | 0 | 43h | Invalid PD-mapping, invalid object length. |

## 6.3 PDO(rx)-Tele-gramm

Using this telegram the logical CAN-Master transfers the new process-output data to the inverter. In the standard setting the KEB-CAN interface waits for a telegram with ≥ 4 byte data with the following contents:




The length and assignment of the PDO(rx)-telegram can be changed by different configuration parameters. This change can only be done with the SDO(tx)-telegram (see above). The following configuration parameters have an influence on the structure of the process-output data.

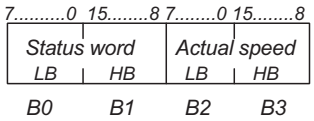
**- PD_Motorola**
**- 1st receive PDO Mapping**
**- 1st receive PDO Parameter**

## 6.4 PDO(tx)-Telegramm

Using this telegram the KEB-CAN-Interface provides the (logical) CAN-Master with the process-input data. The length, assignment and control of this telegram is influenced by the following configuration parameters:

**- PD_Motorola**
**- 1st transmit PDO Mapping**
**- 1st transmit PDO Parameter**
**- Auto_Event_Ack**

The standard setting causes the following telegram structure:



# 7 Configuration Parameters

These parameters determine the configuration of the KEB-F4-CAN interface and thus are realized in the F4-CAN and not in the inverter control.

| | |
|---|---|
| **Name:** | **CAN_Baud** |
| **Index:** | **5FFFh** |
| **Object-Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | Index for CAN-transmission speed |
| **Coding:** | **0 = 10 Kbit/s.**<br>1 = 20 Kbit/s.<br>2 = 50 Kbit/s.<br>3 = 100 Kbit/s.<br>4 = 125 Kbit/s.<br>5 = 250 Kbit/s.<br>6 = 500 Kbit/s*.<br>7 = 1000 Kbit/s*.<br>8 = 800 Kbit/s*.<br>9 = 25 Kbit/s. |
| **Standard Setting:** | **1 = 20 Kbit/s.** |
| **Note:** | A value changed is immediately active but not automatically stored non-volatiley. The Bit-Timing adheres to the specifications of the Working Group Physical-Layer of CiA (see Literature index [3]). See annex for Bit-Timing.<br>Which transmission speeds can be used depends on the line length, the sum of the deceleration times and the Bit-timing. |

| | |
|---|---|
| **Name:** | **SAVE_CAN_Baud** |
| **Index:** | **5FFEh** |
| **Object-Typ:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | Used for non-volatile storage of adjusted CAN- transmission speed. |
| **Coding:** | FFh = non-volatile storage of CAN_Baud.<br>0 = no storage.<br>In case of reading the value 0 is always returned. |

* see ‘chapter 9.2.1 Wichtiger Warnhinweis’

| | |
|---|---|
| **Name:** | **Dynamic_SDO_Len** |
| **Index:** | **5FFDh** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | Determines whether the data length should be processed in Byte 0 of the write/read request or not. |
| **Coding:** | FFh = ON: Processing of the data length in the SDO(rx)- and SDO(tx)-telegram.<br>0 = OFF. The data length for write requests is assumed to be 2 Bytes. The data length for read confirmations is fixed at 4 bytes. |
| **Standard Setting:** | 0 (OFF). |
| **Note:** | The value of this parameter only has an effect when accessing parameters in the Index-range 2000h to 5EFFh. When writing on the configuration parameter the length specification in the CAN-telegram doesn't matter. A value changed is immediately active and non-volatiley stored. |

| | |
|---|---|
| **Name:** | **Auto_PDO_DBT** |
| **Index:** | **5FF8h** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | Determines whether the telegram numbers (identifier) for the PDO(rx) and PDO(tx) - telegram are automatically assigned dependent on the **Module_Id** or if the identifier is preset by the objects **1st receive PDO Parameter** or **1st transmit PDO Parameter.** |
| **Coding:** | FFh (ON): Identifier for PDO(rx) = 512 + Module_Id<br>Identifier for PDO(tx) = 384 + Module_Id<br><br>0 (OFF): Identifier for PDO(rx) = cob_id-entry of the object 1st receive PDO parameter.<br>Identifier for PDO(tx) = cob_id-entry of the object 1st trans mit PDO parameter. |
| **Standard Setting:** | FFh (ON). |
| **Note:** | Every time the identifier for PDO(rx) or PDO(tx) is changed the value of the parameter Auto_PDO_DBT automatically changes to zero. A value changed is immediately active and non-volatiley stored.<br>Module_Id = Inverter address + 1. |

| | |
|---|---|
| **Name:** | **Auto_Ansi_Baud (only available with the operator)** |
| **Index:** | **5FF7h** |
| **Subindex:** | **0** |
| **Object Type:** | **Simple Variable (Var)** |
| **Data length:** | **n** Byte |
| **Significance:** | Determines whether the transmission speed of KEB-DIN66019 is automatically set to the highest possible value or a predefined one. |
| **Coding:** | FFh (ON). The communication between CAN-Operator and the inverter control via KEB DIN66019 is done with the highest possible transmission speed.<br>0 : 1200 Bit/s<br>1 : 2400 Bit/s<br>2 : 4800 Bit/s<br>3 : 9600 Bit/s (Standard transmission speed for all KEB-inverters)<br>4 : 19200 Bit/s (not supported by every inverter)<br>5 : 38400 Bit/s (not supported by every inverter) |
| **Standard Setting:** | FFh (ON) |
| **Note:** | The value of this parameter does not have an effect on the inverters, in which the KEB-DIN66019 transmission speed is not adjustable (e.g. F4-S V1.0).<br>A changed value is non-volatiley stored and first active after being switched on again. |

| | |
|---|---|
| **Name:** | **Ansi_Tout (only available with the Operator)** |
| **Index:** | **5FF6h** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | Provides the timeout-time for KEB-DIN66019 protocol. |
| **Coding:** | n * 2 ms (Example: 10 = 20 ms) |
| **Standard Setting:** | **Dependent on the KEB-DIN66019-tranmission speed:**<br>**At 1200 Bit/s : 164 = 328 ms**<br>**At 2400 Bit/s : 88 = 176 ms**<br>**At 4800 Bit/s : 50 = 100 ms**<br>**At 9600 Bit/s : 30 = 60 ms**<br>**At 19200 Bit/s : 20 = 40 ms**<br>**At 38400 Bit/s : 15 = 30 ms** |
| **Note:** | This parameter can only be read (Upload). |

| | |
|---|---|
| **Name:** | **Ansi_Baud** |
| **Index:** | **5FF5h** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | Provides the index with the actual adjusted transmission speed between Operator and inverter control. |
| **Coding:** | 0 : 1200 Bit/s<br>1 : 2400 Bit/s<br>2 : 4800 Bit/s<br>3 : 9600 Bit/s<br>4 : 19200 Bit/s<br>5 : 38400 Bit/s |
| **Standard Setting:** | **See Auto_Ansi_Baud.** |
| **Note:** | This parameter can only be read (Upload). |

| | |
|---|---|
| **Name:** | **Auto_Event_Ack** |
| **Index:** | **5FF4h** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | Determines whether new process-input data is automatically sent to CAN-Bus (ON) or if the CAN-Master must request actual process-input data via Remote-Frame on the PDO(tx)-identifier. |
| **Coding:** | FFh (ON)<br>0 (OFF) |
| **Standard Setting:** | **FFh (ON)** |
| **Note:** | The adjustment of this parameter is only relevant if the parameter 1st transmit PDO Parameter, transmission-type has the value 254d/FEh. A value changed is immediately active and non-volatiley stored. |

| | |
|---|---|
| **Name:** | **Save_PD_Map_And_Para** |
| **Index:** | **5FF3h** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | Used to non-volatiley store parameters:<br>- 1st receive PDO Parameter.<br>- 1st receive PDO Mapping.<br>- 1st transmit PDO Parameter.<br>- 1st transmit PDO Mapping. |

| | |
|---|---|
| **Coding at Writing:** | FFh (ON) : Parameter above must be stored non-volatiley.<br>otherwise (OFF) : no storing requested . |
| **Coding at Reading:** | FFh (ON) : All parameter values above are correspond to the non-volatiley stored values.<br>otherwise: min. one of the parameter value above does not correspond to the non-volatiley stored value. |
| **Standard Setting:** | **FFh** |

| | |
|---|---|
| **Name:** | **PD_Motorola** |
| **Index:** | **5FF1h** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | For coding each process-data object you can choose between Intel format (PD_Motorola = Intel, first LSByte) and Motorola format (PD_Motorola = Motorola, first MSByte). |
| **Coding:** | FFh (Motorola)<br>**0 (Intel)** |
| **Standard Setting:** | 0 (Intel) |
| **Note:** | A changed value is immediately active and non-volatiley stored. |

| | |
|---|---|
| **Name:** | **SW_Version** |
| **Index:** | **5FEEh** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | Shows the actual SW version of the CANopen-interface. |
| **Coding:** | 0.1 (Example: 22d = version 2.2) |
| **Standard Setting:** | **dependent on SW-update** |
| **Note:** | This parameter is read-only and available from SW version 2.2. |

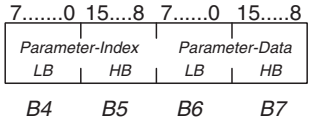
| | |
|---|---|
| **Name:** | **SW_Date** |
| **Index:** | **5FEDh** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length**: | **2** Byte |
| **Significance:** | Shows the actual SW date of the CANopen-interface. |
| **Coding:** | The last digit shows the year -1990. The next two higher digits show the month. The highest digit(s) shows the day.<br>Example : 24117d ==> 24.11.1997. |
| **Standard Setting:** | **dependent on SW-date** |
| **Note:** | This parameter is read-only and available from SW version 2.2. |

**Name:** **CAN_Baud2**
**Index:** **5FECh**
**Object Type:** **Simple Variable (Var)**
**Subindex:** **0**
**Data length:** **1** Byte
**Significance:** Index for CAN-transmission speed alternatively to CAN_Baud (see obove).
**Coding:**
**0 = 10 Kbit/s.**
1 = 20 Kbit/s.
2 = 50 Kbit/s.
3 = 100 Kbit/s.
4 = 125 Kbit/s.
5 = 250 Kbit/s.
6 = 500 Kbit/s*.
7 = 1000 Kbit/s*.
8 = 800 Kbit/s*.
9 = 25 Kbit/s.
**Standard Setting:** **1 = 20 Kbit/s.**
**Note:** Contrary to parameter CAN_Baud a changed value is immediately stored non-volatiley but only active after a Reset Node-command or Power ON. This parameter is available from SW version 2.2.

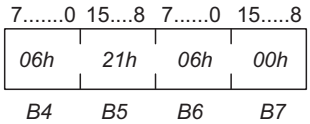
**Name:** **Activate_Life_Guarding**
**Index:** **5FEAh**
**Object Type:** **Simple Variable (Var)**
**Subindex:** **0**
**Data length:** **1** Byte
**Significance:** Switching ON/OFF Life-Guarding.
**Coding:** **FFh:** Life-Guarding active.
**otherwise :** Life-Guarding not active.
**Standard Setting:** **0**
**Note:** When Life-Guarding is active the values of parameters **Guard_Time** and **Life_Time_Factor** are checked for (valid values). If the product of Guard_Time and Life_Time_Factor is higher than 120000d, switching on of Life-Guarding is refused with the error message “invalid parameter value”. Life-Time is limited to 120 s. In this case the non-volatily stored values of both parameter are restored. This parameter is available from SW version 2.2.

* see ‘chapter 9.2.1 Wichtiger Warnhinweis’

**Name:** **Func_Life_Guard_Tout**
**Index:** **5FE9h**
**Object Type:** **Simple Variable (Var)**
**Subindex:** **0**
**Data length:** **4** Byte
**Significance:** Determines the write process which is executed one time when the Life-Guarding operates to the frequency inverter control.
**Coding:** The first word determines the writing parameter-index (parameter Address+2000h). The second word shows the corresponding value. The coding in the CAN-Data telegram is displayed as follows:




**Standard Setting:** Parameter-Index = 2106h, Parameter-Value = 6:




**Note:** A changed value is immediately active and non-volatiley stored. This parameter is available from SW version 2.2.

**Name:** **CAN_Tout_Time (new from SW V2.3)**
**Index:** **5FE8h**
**Object Type:** **Simple Variable (Var)**
**Subindex:** **0**
**Data length:** **2** Byte
**Significance:** Shows the Timeout or the CAN-monitoring. The CAN-Timeout-monitoring is activated for values other than zero. The monitoring becomes active when the first telegram is received at this node (see below). After that the Timeout counter is reset by the following telegrams:
- Remote-Frame on IN-Identifier.
- Remote-Frame on Node-Guarding Identifier.
- SYNC-Telegram.
- Data telegram on OUT-Identifier.
- Data telegram on Request-Identifier.
- Data telegrams on Identifier 2026d (NMT-service).
- Data telegrams on Identifier 0 (NMT-Start/Stop-commands).

**Coding:** 0 = deactivated
otherwise n * 2 ms (Example: 250d = 500 ms).
**Standard Setting:** **0 (=deactivated).**
**Note:** In case of monitoring a CAN-Timeout the same action is executed like operation with Life-Guarding (see above).

| | |
|---|---|
| **Name:** | **WR_PDOUT_If_Chg (new from SW V2.3)** |
| **Index:** | **5FE7h** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | This parameter determines the conditions for received PDO(rx)-data to the FU-control. Optimization of the communication between CAN-Interface and inverter-control is the sense of this parameter i.e only the needed communication for the process and the process data mapping to the inverter control shall take place. |
| **Coding:** | 0: All received PDO(rx)-data are transfered to the inverter, it doesn't matter if they have been changed or not.<br>FFh: If new PDO(rx)-data are received and only one value has changed, all values are transfered to the inverter control. This value can be important if with PDO(rx)-data e.g. 32-bit-consistency parameter like the set position are preset. The difference between value FFh and values other than 0 support only the SW from version V2.6.<br>otherwise: Only the changed PDO(rx)-data are transfered to the inverter control. |
| **Standard Setting:** | **0** |
| **Note:** | A changed value is immediately active and non-volatiley stored. |

| | |
|---|---|
| **Name:** | **PDIN_Cycle_Time (new from SW V2.3)** |
| **Index:** | **5FE6h** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **2** Byte |
| **Significance:** | Shows the cycle time in which the process input data in OPERATIONAL state are read from the FU-control. |
| **Coding:** | n * 2ms. |
| **Standard Setting:** | **15 (= 30ms)** |
| **Note:** | A changed value is immediately active and non-volatiley stored. |

| | |
|---|---|
| **Name:** | **device type** (in accordance with CANopen [13]) |
| **Index:** | **1000h** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **4** Byte |
| **Significance:** | Describes the unit type in accordance with the CANopen communication profile. |
| **Coding:** | None specified at this point. |
| **Standard Setting:** | **0** |
| **Note:** | This parameter is constant and can only be read. |

| | |
|---|---|
| **Name:** | **error register** (in accordance with CANopen [13]) |
| **Index:** | **1001h** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | Shows the error status of CANopen-devices. |
| **Coding:** | Bit0 : =1 ==> error has occured.<br>Bit7 : =1 ==> error occurred while writing the process-output data or while reading the process-input data. |
| **Standard Setting:** | **0** |
| **Note:** | This parameter can only be read. Every read out of this parameter returns the value to zero. |

| | |
|---|---|
| **Name:** | **guard time** (in accordance with CANopen [13]) |
| **Index:** | **100Ch** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **2** Byte |
| **Significance:** | Shows the time (together with the life time factor) when Life-Guarding is operated, if no Node-Guarding-Request was received during this time. |
| **Coding:** | 1 ms |
| **Standard Setting:** | **1000d** |
| **Note:** | When the value is changed Life-Guarding is switched off (Activate_Life_Guarding = 0). Then it must be explicit switched on again. (see above).<br>This parameter is available from SW version 2.2. |

| | |
|---|---|
| **Name:** | **life time factor** (in accordance with CANopen [13]) |
| **Index:** | **100Dh** |
| **Object Type:** | **Simple Variable (Var)** |
| **Subindex:** | **0** |
| **Data length:** | **1** Byte |
| **Significance:** | Shows the time (together with guard time) when Life-Guarding is operated, if no Node-Guarding-Request was received during this time. |
| **Coding:** | n * guard time |
| **Standard Setting:** | **5d (==> Life-Time = 5 * 1000 ms = 5 s)** |
| **Note:** | When the value is changed Life-Guarding is switched off. (Activate_Life_Guarding = 0). Then it must be explicitly switched on again (see above).<br>This parameter is available from SW version 2.2. |

**Name:** **1st receive PDO Parameter**
**Index:** **1400h**
**Object Type:** **Structured Variable (Record)**
**Subindex:** **0 (Number of entries in this object)**
**Data length:** **1** Byte
**Significance:** Provides the number of entries which can be accessed in this object.
**Coding:** 1
**Standard Setting:** **2**
**Note:** The value of this parameter can only be read.

**Subindex:** **1 (COB-ID used by PDO)**
**Data length:** **4** Bytes
**Significance:** Specifies, on which identifier the PDO(rx) is send for the transfer of the process-output data. Control information for this PDO is found in the upper Bits.
**Coding:**
Bit31(MSB) = 0 ==> The processing of the process-output data is activated.
Bit31(MSB) = 1 ==> The process-output data are notprocessed.
Bit30 = 0 ==> The Remote Frame on the respective Identifier is answered.
Bit30 = 1 ==> The Remote Frame is not answered.
Bit29 = 0 ==> 11-bit Identifier (CAN V2.0A)
Bit29 = 1 ==> 29-bit Identifier (CAN V2.0B), not possible here.
Bit28...Bit0: Identifier (Bit0 = LSB), here for Bit28 to Bit11 value “1” invalid.
**Standard Setting:** **00000202h**
**Note:** A changed value is immediately active but not automatically stored non-volatiley(see also Save_PD_Map_And_Para). When the process-output data processing is switched on (Bit31 changed from “1” to “0”) the process-data mapping (Index 1600h) is checked for validity. If invalid mappings are entered there, then the process-output data (see Abort Domain Transfer) cannot be switched on.
**Subindex:** **2 (transmission type) (geändert ab SW V2.3)**
**Data length:** **1** Byte
**Significance:** Determines when and how this object was sent on the CAN-Bus.
**Coding:**
0 (synchronous acyclic), possible from SW version 2.3.
1-253d (synchronous cyclic), not possible here.
254d (asynchronous, specified by manufacturer).
255d (asynchronous, profile specific), not possible here.
**Standard Setting:** **254d**
**Note:** A changed value is immediately active but not automatically stored non-volatiley (see also Save_PD_Map_And_Para). Please read the chapter “Changing the transmission-type of the PDOs”.

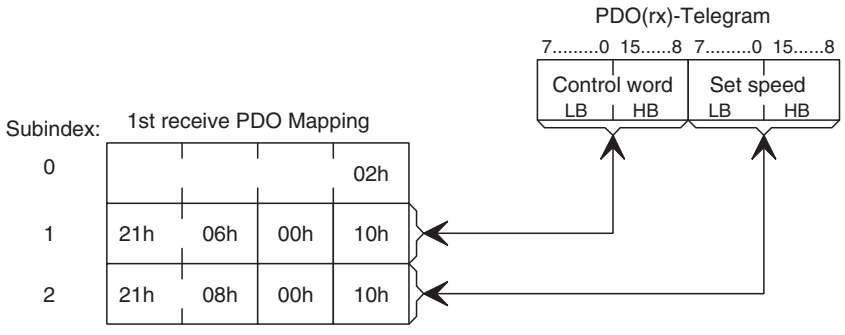
**Name:** **1st receive PDO Mapping**
**Index:** **1600h**
**Object Type:** **Structured Variable (Record)**
**Subindex:** **0 (Number of mapped objects in this PDO)**
**Data length:** **4** Bytes
**Significance:** Provides the number of entries, which can be accessed in this object.
**Coding:** 1 (Maximum valid value range 1.....8).
**Standard Setting:** **2**

**Note:** When a value is changed in this parameter the process-output data processing (Bit31 from Index 1400h, Subindex=1 is set at “1”) is automatically switched off.
**Subindex:** **1 to a maximum of 8 (object mapping)**
**Data length:** **4** Bytes
**Significance:** Determines the object mapping. The Index, Subindex and the **object length in Bits** are specified.
**Coding:**

| *Index* | | *Sub-Index* | *Object Length* |
|---|---|---|---|
| *HB* | *LB* | | |
| *B3* | *B2* | *B1* | *B0* |

**Standard Setting:** See below.
**Note:** When a value is changed in this parameter the process-output data processing (Bit31 of Index 1400h, Subindex=1 and is set at “1” ) is automatically switched off.

The diagram below shows the relationship between the process-output data mapping and the respective PDO(rx) telegram structure in the standard assignment.




Example for coding of data on CAN-BUS:
The first mapped parameter in the receive-PDO should be changed to the parameter with index=4103h and subindex=0. In this case the complete 8 data bytes of the Initiate Domain Download Request have to be set as follows:

| 23h | 00h | 16h | 01h | 10h | 00h | 03h | 41h |
|---|---|---|---|---|---|---|---|
| B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 |

| | |
|---|---|
| **Name:** | **1st transmit PDO Parameter** |
| **Index:** | **1800h** |
| **Object Type:** | **Structured Variable (Record)** |
| **Subindex:** | **0 (Number of entries in this object)** |
| **Data length:** | **1** Byte |
| **Significance:** | Provides the number of entries which can be accessed in this object. |
| **Coding:** | 1 |
| **Standard Setting:** | 3 |
| **Note:** | The value of this parameter can only be read. |

| | |
|---|---|
| **Subindex:** | **1 (COB-ID used by PDO)** |
| **Data length:** | **4** Bytes |
| **Significance:** | Specifies which identifier the PDO(tx) is sent for the transfer of the process-intput data. Control information for this PDO is found in the upper Bits. |
| **Coding:** | Bit31(MSB) = 0 ==> The processing of the process-input data is activated.<br>Bit31(MSB) = 1 ==> The process-input data is not processed.<br>Bit30 = 0 ==> Remote Frame is answered on the respective identifier.<br>Bit30 = 1 ==> Remote Frame is not answered.<br>Bit29 = 0 ==> 11-bit Identifier (CAN V2.0A)<br>Bit29 = 1 ==> 29-bit Identifier (CAN V2.0B), not possible here.<br>Bit28...Bit0: Identifier (Bit0 = LSB), here for Bit28 to Bit11 value “1” invalid. |
| **Standard Setting:** | **00000182h** |
| **Note:** | When a value is changed it is immediately active but not automatically stored non-volatiley (see also Save_PD_Map_And_Para). When the process-input data processing is switched on (Bit31 changed from “1” to “0”) the process-data mapping (Index 1A00h) is checked for validity. If invalid mappings are entered there then the process-input data (see Abort Domain Transfer) cannot be switched on. |

| | |
|---|---|
| **Subindex:** | **2 (transmission type) (changed from SW-V2.3)** |
| **Data length:** | **1** Byte |
| **Significance:** | Determines when and how this object can sent to the CAN-Bus. |
| **Coding:** | 0 (synchronous acyclic), available from SW version 2.3.<br>1-253 (synchronous cyclic), not possible here.<br>254 (asynchronous, specified by manufacturer).<br>255 (asynchronous, profile specific), not possible here. |

**Standard Setting:** **254**
**Note:** A changed value is immediately active but not automatically stored non-volatiley. (see also Save_PD_Map_And_Para). Please read the chapter “Changing t0he transmission-type of the PDOs”.

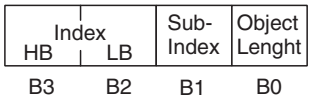
**Subindex:** **3 (inhibit time)**
**Data length:** **2** Bytes
**Significance:** Specifies the minimum time between 2 consecutive telegrams on this identifier. For SW versions < 2.3 also the cycle time is determined in which the process input data in an OPERATIONAL state are read from the FU-control. From SW version 2.3 the inhibit-time for the PDO(tx)-telegram and the cycle time of the process input data are seperate parameters (see PDIN_Cycle_Time).
**Coding:** 100 ms
**Standard Setting:** 300 (= 30 ms).
**Note:** The realized resolution is 2 ms. The adjusted value has an inaccuracy of ± 2 ms.

**Name:** **1st transmit PDO Mapping**
**Index:** **1A00h**
**Object Type:** **Structured Variable (Record)**

**Subindex:** **0 (number of mapped objects in this PDO)**
**Data length:** **4** Bytes
**Significance:** Specifies the number of entries that can be accessed in this object.
**Coding:** 1 (Maximum valid value range 1.....8).
**Standard Setting:** **2**
**Note:** When the value of this parameter is changed, the process-input data processing (Bit31 from Index 1800h,Subindex=1 is set at “1”) is automatically switched off.

**Subindex:** **1 to maximum 8 (Object Mapping)**
**Data length:** **4** Bytes
**Significance:** Specifies the object mapping. The index, subindex and the **length of the object in Bits** is specified.

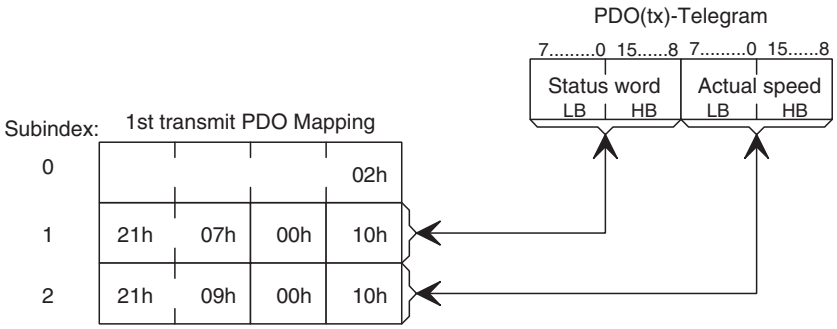
**Coding:**



**Standard Setting:** See above

**Note:** When the value of this parameter is changed, the process-input data processing (Bit31 from Index 1800h,Subindex=1 is set at “1”) is automatically switched off.

The diagram below shows the relation between the process-input data mapping and the respective PDO(tx) telegram structure in the standard assignment.



# 8 Changing the transmission-type of the PDOs

The transmission-type of parameter **1st receive PDO Parameter** and **1st transmit PDO Parameter** is changeable from SW version 2.3. Valid values are:
- Asynchronous manufacturer specified (Value = 254 = Standard) and
- Synchronous acyclic (Value = 0).

This chapter describes the difference between both adjustments.

## 8.1 Asynchronous Manufacturer Specified (Value = 254d/FEh)

If this value of the transmission-type is adjusted in parameter **1st receive PDO Parameter**, the process output data in OPERATIONAL state are transfered to inverter control when a valid PDO(rx)-telegram was received. A valid PDO(rx) telegram is a telegram on the corresponding identifier with a data length of ≥ that length which results from the PDO(rx)-Mapping. Standardly that means all telegrams on the OUT-identifier with a data length of ≥ 4 Byte are accepted.
In OPERATIONAL state the process input data are read cyclicly by the FU-control. If in parameter **1st transmit PDO Parameter** the value 254d is adjusted a PDO(tx)-telegram is sent to the CAN, when the process input data have been changed and the parameter **Auto_Event_Ack** is set to **ON**.

## 8.2 Acyclic Synchronous (Value =0)

If this value of the transmission-type is adjusted in parameter **1st receive PDO Parameter**, the process output data in OPERATIONAL state are transfered to inverter control after a SYNC-telegram was received .
Condition: first a valid PDO(rx)-telegram was received.
For parameter **1st transmit PDO Parameter** the value transmission-type = 0 means that in OPERATIONAL state a PDO(tx)-telegram is sent to the CAN after a SYNC-telegram was received.

# 9 Annex

## 9.1 Coding of the Four Basic DRIVECOM-Parameters

The KEB inverters contain some parameters which are coded in accordance with the profile for power transmission of the DRIVECOM User Group Association. A description of the profile parameters used is found in the manual for the KEB-F4 inverter control. The four basic parameters are listed below. Standardly, the process-data mapping is done with these parameters.
See also points [1] and [2] in the literature index.

**Parameter Name:** **DRIVECOM Control Word**
**Parameter Address:** 0106h
**Significance:** Control the operating conditions of the inverter.

- **Not all KEB-F4 inverters support this parameter. In this case the process-data mapping of the CAN-interface should be adapted respectively.**
- **In order to control the inverter with the DRIVECOM-control word you must configurate one or more parameters in the inverter control. Refer to the manual for the control.**

**Coding: Bit 0:** 1=On , 0=off
Bit 1 : 1=do not isolate voltage
Bit 2 : 1=no quick stop
Bit 3 : 1=release operation, 0=block operation
Bit 4 : 1=do not block speed ramp
Bit 5 : 1= do not stop speed ramp
Bit 6 : 1=do not block set value
Bit 7 : (0->1) = reset malfunction
Bit 8 : currently no function.
Bit 9 : currently no function.
Bit 10 : currently no function.
Bit 11 : currently no function.
Bit 12 : currently no function.
Bit 13 : currently no function.
Bit 14 : currently no function.
Bit 15 : currently no function.

**Parameter Name:** **DRIVECOM Status Word**
**Parameter Address:** **0107h**
**Significance:** Specifies the current status of the inverter.

- Not all KEB-F4 inverters support this parameter. In this case the process-data mapping of the CAN-interface should be adapted respectively. Refer to the manual for the inverter control.

**Coding:**

Bit 0 : 1=Ready to start
Bit 1 : 1=on
Bit 2 : 1=release operation
Bit 3 : 1=malfunction
Bit 4 : 1=no voltage released
Bit 5 : 1=no quick stop
Bit 6 : 1=closing lock-out
Bit 7 : 1=warning
Bit 8 : currently no function
Bit 9 : 1=no local operation active
Bit 10 : 1=set value reached
Bit 11 : 1=limitation active
Bit 12 : currently no function
Bit 13 : currently no function
Bit 14 : currently no function
Bit 15 : currently no function

**Parameter Name:** **DRIVECOM Set Speed**
**Parameter Address:** **0108h**
**Significance:** Specifies the set speed of the inverter.
**Coding:** 1 $min^{-1}$

**Parameter Name:** **DRIVECOM Actual Speed**
**Parameter Address:** **0109h**
**Significance:** Specifies the set speed of the inverter.
**Coding:** 1 $min^{-1}$

## 9.2 CAN-Bit-Timing

The KEB-CAN interface adheres to the adjusted Bit-timings defined by the CIA standards (see [3] in index literature):
The nominal bit timing is as follows:


The following is valid for all adjustable baudrates:

- $t_q$ : Basis time unit. All segments of the Bit-Timing result in a multiple of this time unit.
- SYNC: = 0 ==> Only the edges which go from recessive to dominant are used for synchronization.
- SJW: = 0 ==> Synchronization jump width = 1 * $t_q$
- TSEG2: = 1 ==> $t_{SEG2}$ = 2 * $t_q$

| Baudrate | Time-Quantum ($t_q$) | Sample-Point | TSEG1 |
|---|---|---|---|
| 10 Kbit/s | 6.25 µs | 14 * $t_q$ = 87.5 µs | 12 ==> $t_{SEG1}$ = 13 * $t_q$ |
| 20 Kbit/s | 3.125 µs | 14 * $t_q$ = 43.75 µs | 12 ==> $t_{SEG1}$ = 13 * $t_q$ |
| 25 Kbit/s | 2,5 µs | 14 * $t_q$ = 35,0 µs | 12 ==> $t_{SEG1}$ = 13 * $t_q$ |
| 50 Kbit/s | 1.25 µs | 14 * $t_q$ = 17.5 µs | 12 ==> $t_{SEG1}$ = 13 * $t_q$ |
| 100 Kbit/s | 625 ns | 14 * $t_q$ = 8.75 µs | 12 ==> $t_{SEG1}$ = 13 * $t_q$ |
| 125 Kbit/s | 500 ns | 14 * $t_q$ = 7.0 µs | 12 ==> $t_{SEG1}$ = 13 * $t_q$ |
| 250 Kbit/s | 250 ns | 14 * $t_q$ = 3.5 µs | 12 ==> $t_{SEG1}$ = 13 * $t_q$ |
| 500 Kbit/s | 125 ns | 14 * $t_q$ = 1.75 µs | 12 ==> $t_{SEG1}$ = 13 * $t_q$ |
| 800 Kbit/s | 125 ns | 8 * $t_q$ = 1,0 µs | 6 ==> $t_{SEG1}$ = 7 * $t_q$ |
| 1000 Kbit/s | 125 ns | 6 * $t_q$ = 750 ns | 4 ==> $t_{SEG1}$ = 5 * $t_q$ |

**The transmission speeds marked grey in the table must be seen very critical in relation to the possible line length.**

### 9.2.1 Important Warning Information

The KEB-CAN-Module has a potential isolated CAN-Interface. The line length or the possible transmission speed is reduced due to additional delay elements (optocoupler) in the signal line. The possible line length or transmission speed is dependent on the delay times of all users in the CAN-network. It is up to the user which bitrate is possible for the line length. Necessary information for the KEB-CAN-Interface is described as follows:

| | |
|---|---|
| Sum of the input/output delay times of the CAN-controller | : ≤ 62 ns. |
| Transmission-delay time of the CAN-driver | : ≤ 80 ns. |
| Receiving-delay time of the CAN-driver | : ≤ 70 ns. |
| Transmission-delay time of the used optocoupler | : ≤ 100 ns. |
| Receiving-delay time of the used optocoupler | : ≤ 100 ns. |

**In any case the smallest CAN-transmission speed should be used which is required by the process.**

## 9.3 Literature Index


[1]: Instruction Manual KEB Combivert F4 with Application Manual.

[2]: Profile Antriebstechnik (21) from 16.12.1991.
Publisher: DRIVECOM User Group Association, Postfach 1102, 32825 Blomberg

[3]: Document of Specifications of Working Group Physical-Layer the CAN in Automation (CiA) User Group : CiA/DS 102-1.
Publisher: CiA International Users and Manufacturers Group Association, Am Weichselgarten 26, D-91058 Erlangen.
Documents of Specification of Working Group Higher-Layer-Protocols of the CiA (Publisher see above)

[4]: CiA/WG2/DS201 : CAN in the OSI Reference Model.

[5]: CiA/WG2/DS202-1 : CMS Service Specification.

[6]: CiA/WG2/DS202-2 : CMS Protocol Specification.

[7]: CiA/WG2/DS202-3 : CMS Encoding Rules.

[8]: CiA/WG2/DS203-1 : NMT Service Specification.

[9]: CiA/WG2/DS203-2 : NMT Protocol Specification.

[10]: CiA/WG2/DS204-1 : DBT Service Specification.

[11]: CiA/WG2/DS204-2 : DBT Protocol Specification.

[12]: CiA/WG2/DS207 : Application Layer Naming Conventions.

[13]: CiA/DS301 : CAL-based Communication Profile v. 22.9.1995

[14]: CiA/RFC for CiA/DS301 V3.0 : "Request for Comments" zu [13] v. 1.8.1997

## 9.4 Table of configuration-parameters according to CANopen

| Index | Name | Objekt-Type | Subindex | Data Lenght in Byte | Data Type | Acc. |
|---|---|---|---|---|---|---|
| 1000h | device type | VAR | 0 | 4 | UNSIGNED32 | ro |
| 1001h | error register | VAR | 0 | 1 | UNSIGNED8 | ro |
| 100Ch | guard time | VAR | 0 | 2 | UNSIGNED16 | rw |
| 100Dh | life time factor | VAR | 0 | 1 | UNSIGNED8 | rw |
| 1400h | 1st receive PDO Parameter | RECORD | | | PDOCommPar | |
| 1400h | 1st receive PDO Parameter, number of entries | | 0 | 1 | UNSIGNED8 | ro |
| 1400h | 1st receive PDO Parameter, COB-ID used by PDO | | 1 | 4 | UNSIGNED32 | rw |
| 1400h | 1st receive PDO Parameter, transmission type | | 2 | 1 | UNSIGNED8 | ro |
| 1600h | 1st receive PDO Mapping | ARRAY | | | UNSIGNED32 | |
| 1600h | 1st receive PDO Mapping, number of mapped application objects in PDO | | 0 | 4 | UNSIGNED32 | rw |
| 1600h | 1st receive PDO Mapping, PDO mapping for the nth application object to be mapped | | 1 to maximum 8 | 4 | UNSIGNED32 | rw |
| 1800h | 1st transmit PDO Parameter | RECORD | | | PDOCommPar | |
| 1800h | 1st transmit PDO Parameter, number of entries | | 0 | 1 | UNSIGNED8 | ro |
| 1800h | 1st transmit PDO Parameter, COB-ID used by PDO | | 1 | 4 | UNSIGNED32 | rw |
| 1800h | 1st transmit PDO Parameter, transmission type | | 2 | 1 | UNSIGNED8 | ro |
| 1800h | 1st transmit PDO Parameter, inhibit time | | 3 | 2 | UNSIGNED16 | rw |

| Index | Name | Objekt-Type | Subindex | Data Lenght in Byte | Data Type | Acc. |
|---|---|---|---|---|---|---|
| 1A00h | 1st transmit PDO Mapping | ARRAY | | | UNSIGNED32 | |
| 1A00h | 1st transmit PDO Mapping, number of mapped application objects in PDO | | 0 | 4 | UNSIGNED32 | rw |
| 1A00h | 1st transmit PDO Mapping, PDO mapping for the nth application object to be mapped | | 1 to max. 8 | 4 | UNSIGNED32 | rw |
| 5FFFh | CAN-Baud | VAR | 0 | 1 | UNSIGNED8 | rw |
| 5FFEh | SAVE_CAN_BAUD | VAR | 0 | 1 | UNSIGNED8 | rw |
| 5FFDh | Dynamic_SDO_Len | VAR | 0 | 1 | UNSIGNED8 | rw |
| 5FF8h | Auto_PDO_DBT | VAR | 0 | 1 | UNSIGNED8 | rw |
| 5FF7h | Auto_Ansi_Baud | VAR | 0 | 1 | UNSIGNED8 | rw |
| 5FF6h | Ansi-Tout | VAR | 0 | 1 | UNSIGNED8 | ro |
| 5FF5h | Ansi_Baud | VAR | 0 | 1 | UNSIGNED8 | ro |
| 5FF4h | Auto_Event_Ack | VAR | 0 | 1 | UNSIGNED8 | rw |
| 5FF3h | Save_PD_Map_And_Para | VAR | 0 | 1 | UNSIGNED8 | rw |
| 5FF1h | PD_Motorola | VAR | 0 | 1 | UNSIGNED8 | rw |
| 5FEEh | SW_Version | VAR | 0 | 1 | UNSIGNED8 | ro |
| 5FEDh | SW_Date | VAR | 0 | 2 | UNSIGNED16 | ro |
| 5FECh | CAN_Baud2 | VAR | 0 | 1 | UNSIGNED8 | rw |
| 5FEAh | Activate_Life_Guarding | VAR | 0 | 1 | UNSIGNED8 | rw |
| 5FE9h | Func_Life_Guard_Tout | VAR | 0 | 4 | UNSIGNED32 | rw |
| 5FE8h | CAN_Tout_Time | VAR | 0 | 2 | UNSIGNED16 | rw |
| 5FE7h | WR_PDOUT_If_Chg | VAR | 0 | 1 | UNSIGNED8 | rw |
| 5FE6h | PDIN_Cycle_Time | VAR | 0 | 2 | UNSIGNED16 | rw |

**KEB-Antriebstechnik Ges.m.b.H.**
**Ritzstraße 8 • A - 4614 Marchtrenk**
**Tel.: 0043 / 7243 / 53586 - 0 • FAX: 0043 / 7243 / 53586-21**

**KEBCO Inc.**
**1335 Mendota Heights Road**
**USA - Mendota Heights, MN 55120**
**Tel.: 001 / 651 / 4546162 • FAX: 001 / 651 / 4546198**

**KEB (UK) Ltd.**
**6 Chieftain Buisiness Park, Morris Close**
**Park Farm, Wellingborough, GB - Northants, NN8 6 XF**
**Tel.: 0044 / 1933 / 402220 • FAX: 0044 / 1933 / 400724**

**KEB - YAMAKYU Ltd.**
**711 Fukudayama, Fukuda**
**J - Shinjo City, Yamagata (996-0053)**
**Tel.: 0081 / 233 / 29 / 2800 • FAX: 0081 / 233 / 29 / 2802**

**KEB Italia S.r.l.**
**Via Newton, 2 • I - 20019 SETTIMO MILANESE (Milano)**
**Tel.: 0039 / 02 / 33500782 • FAX: 0039 / 02 / 33500790**

**Société Francaise KEB**
**Z.I. de la Croix St Nicolas • 14, rue Gustave Eiffel**
**F - 94510 LA QUEUE EN BRIE**
**Tél.: 0033 / 1 / 49620101 • FAX: 0033 / 1 / 45767495**

**Karl E. Brinkmann GmbH**
**Försterweg 36 - 38 • D - 32683 Barntrup**
**Telefon 0 52 63 / 4 01 - 0 • Telefax 4 01 - 116**
**Internet: www.keb.de • E-mail: info@keb.de**