HEIDENHAIN

EIB 74x

User's Manual for Application Development

English (en)
12/2019

Contents

1

Fundamentals

1.1 Documentation

The documentation for EIB 741, EIB 742 and EIB 749, referred to in the following as EIB 74x, comprises the following documents:

- Operating Instructions
  - Documents required for commissioning, as well as technical specifications.
- User's Manual for Application Development
  - Description of the features of the EIB 74x
  - Description of the installation and function calls of the driver software.

DHCP

The EIB 74x can work with static IP addresses or with dynamic IP addresses that are obtained from a DHCP server. By default, DHCP is deactivated and the EIB 74x uses static IP addresses. This address can be set by the user, in order to conform to the requirements of a specific network.

If DHCP is activated, then after it has booted, the EIB 74x tries to get an IP address from a DHCP server. This address is used until the duration of validity shown under “Lease” expires. If necessary, the EIB 74x can renew the “lease” by itself. If no DHCP server can be found that provides an address before the timeout is reached, the EIB 74x will use the default IP address. If DHCP is selected, but no DHCP server is available, then booting takes longer.

The DHCP client requests an IP address, the subnet mask, and the default gateway. In addition, the host name of the EIB 74x is transmitted to the DHCP server. If the DHCP server is connected to a DNS server, then the host name can be used instead of the IP address for communication with the EIB 74x.

The default host name is distinct for each EIB 74x, and contains the unit name and a unique serial number. Example of a host name:
EIB741-SN123456
The unit name is "EIB741" and the serial number is "SN123456". The serial number is printed on the ID label on the rear of the EIB 74x. The host name can be changed by a software command.

To change the network settings, you can e.g. use the program called "networksettings" supplied on CD (path: **...\windows\tools \networksettings\networksettings.exe**).

In order to avoid interference of other network participants (not required for the application), HEIDENHAIN recommends setting up a separate network in order to connect the EIB 74x.

1.2 Target group and personnel qualification

This document must be read and observed by every person who is involved in application development.

Application development requires adequate qualification. The application developer must have obtained sufficient information from the documentation supplied with the product and with the connected peripherals.

The application developer has the required technical training, knowledge and experience and knows the applicable regulations, and is thus capable of performing the assigned work regarding the application concerned and of pro-actively identifying and avoiding potential risks.

1.3 Firmware version

This document describes Firmware version: 633281-14.

1.4 Change history

Changes from the previous versions are listed in the change history. The document on change history is on the CD in the subdirectory EIB_74x/doc. Please read this document, in particular the notes on new, changed or obsolete function calls.

2

Available functions

2.1 General description of function

The EIB 74x includes evaluation electronics for precise position measurement, especially for inspection stations and multipoint inspection apparatuses as well as for mobile data acquisition, e.g. during machine calibration.

The EIB 74x is ideal for applications requiring high-resolution encoder signals and fast measured-value acquisition. In addition, Ethernet transmission enables you to use switches or hubs for connecting more than one EIB 74x.

Up to four HEIDENHAIN encoders, either with sinusoidal incremental signals (1 V_{pp}) or with EnDat interfaces (EnDat 2.1 and EnDat 2.2), can be connected to the EIB 74x.

The EIB 74x subdivides the periods of the incremental signals 4096-fold for measured-value generation. The deviations within one signal period are automatically reduced by adjustment of the sinusoidal incremental signals (signal compensation).

The integrated measured-value memory enables the EIB 74x to save up to 250,000 measured values per axis in Recording operating mode. Internal or external triggers can be used for axis-specific storage of the measured values.

A standard Ethernet interface using TCP/IP or UDP communication is available for data output. This permits direct connection to a PC, laptop or industrial PC. The method of measured-value transmission can be set via the operating mode. Driver software for Windows, Linux, and LabVIEW is included in the items supplied, in order to process the measured values on the PC. The driver software enables customers to easily program their own applications. It also contains sample programs demonstrating the performance range of the EIB 74x.

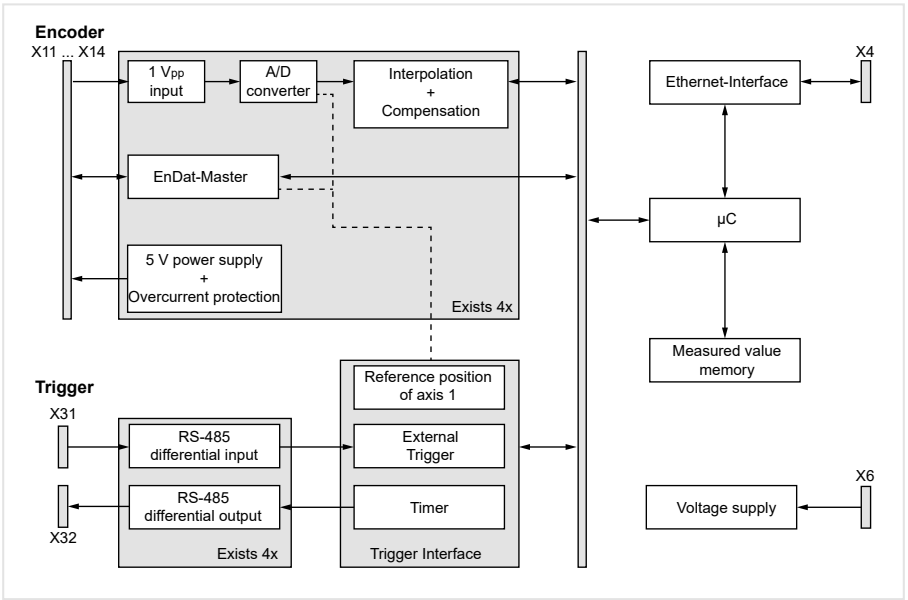

Figure 1: Basic Circuit Diagram

A maximum of four HEIDENHAIN encoders with the following (freely programmable) interfaces can be connected to the EIB 74x:

- Incremental signals 1 V_{pp}
- EnDat 2.1
- EnDat 2.2
- Incremental signals 11 μA_{PP} (upon request)

The power supply to the encoders is provided by the EIB 74x and is protected by a resettable overload cutout.

For specifications, see the Operating instructions.

2.2 Configuration of the encoder inputs

After power-up, the power supply to the encoders is activated. The other parameters for operating the encoder input must be configured by initialization:

- Interface type
- Bandwidth for the 1 V_{pp} input signals
- Signal compensation
- Processing the reference marks
- Processing the homing/limit signals

These settings can be changed via software.

The interface for the encoder input can be operated in incremental or EnDat mode. In EnDat mode, the incremental block can also be operated if, in addition to EnDat, the encoder also supports the 1 V_{pp} interface.

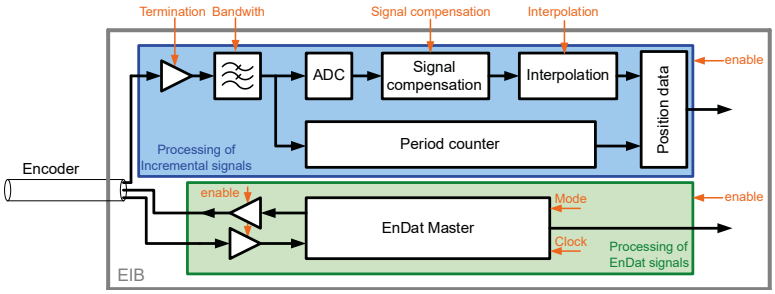

Figure 2: Encoder input

2.2.1 Processing incremental signals

The EIB 74x subdivides the periods of the incremental signals 4096-fold (12 bits) for generating the position value. The period counter has a width of 32 bits. The counter value is increased or decreased by the value "1" with each signal period of the connected encoder.

The deviations within one signal period are automatically reduced by adjustment of the sinusoidal incremental signals (signal compensation). Compensation of the incremental encoder signals and the terminating resistor can be activated or deactivated by software.

The 44 bit position value at the time of the trigger event is formed from the interpolation value (12 bits) and the value of the period counter (32 bits). The position value is saved in a 48 bit register (see table). The period counter is mapped in the two's complement notation; bits 43 to 47 represent the algebraic sign.

Depending on the encoder type (linear or rotational), the higher-level customer software application can use this value to calculate the angle or length, respectively. The overflow of the period counter will occur at `0x07FF FFFF FFFF` (positive maximum) →`0xF800 0000 0000` (negative maximum) position in two's complement notation

This overflow does not affect the functionality of the period counter or the interpolator. The overflow, however, must be handled by the higher-level customer software application.

Register contents, position value for incremental signals

Bit no.	Width (bits)	Contents
0 ... 11	12	Interpolation value
12 ... 43	32	Period counter (bit 43 = algebraic sign)
44 ... 47	4	Value identical to bit 43

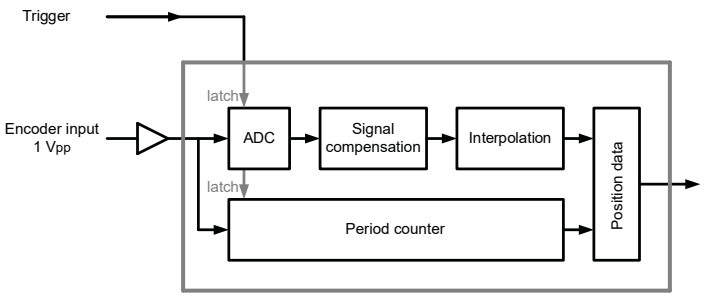

Figure 3: Block diagram of the encoder input

Interpolation value

At the time of the trigger event, the incremental signals are scanned and used to calculate a 12-bit wide interpolation value (not for the EnDat interface). The correlation between interpolation value and incremental signals is derived as follows:

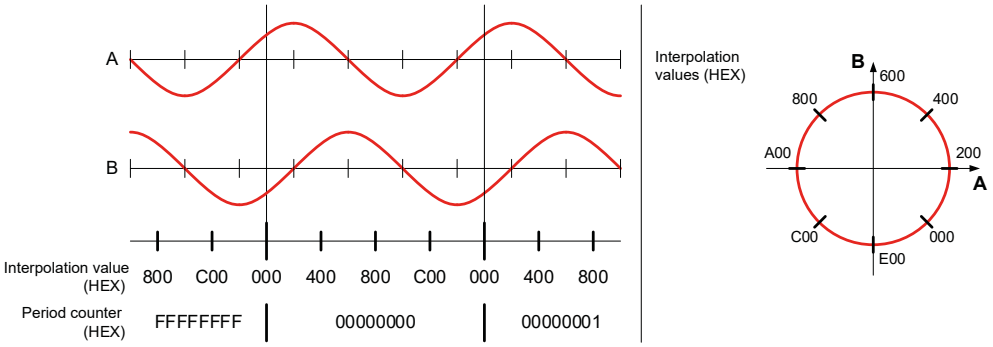

Figure 4: Interpolation value for incremental signals

Setting options:

- Terminating resistors for the incremental signals
  The terminating resistor 120 Ω for the 1 V_{pp} incremental signals can be activated or deactivated by software (for all channels at the same time; default: resistors are activated)
- Bandwidth settings of the incremental signals
  The bandwidth of the encoder's incremental signals can be toggled by software. Set the high bandwidth (500 kHz) as the default. The low bandwidth setting (33 kHz) should only be selected for specific applications.
- Signal compensation
  Compensation of the incremental encoder signals can be activated or deactivated by software.

Analog values of the 1 V_{pp} incremental signals A and B

The transmitted values correspond to the values of the AD converter at the time of the trigger event.

Register contents, AD converter for incremental signals

Bit no.	Width (bits)	Contents
0 ... 11	12	12-bit AD converter value
12 ... 15	4	Reserved

Table of AD converter values for incremental signals

Value (hex)	Incremental signal value
`000`	Negative maximum
`800`	Zero
`FFF`	Positive maximum

2.2.2 Dealing with reference marks

With incremental encoders, the reference mark(s) is/are used to create an absolute reference for the incremental signals.

For encoders with a single reference mark, this mark has a unique reference to a specific signal period. This signal period can be used as a reference to generate absolute position values. Traversing the reference mark does not affect the period counter or the interpolation value. The period counter value valid at the time of traversing is simply saved in a register for the reference position. This value can be used in the customer software application to calculate absolute position values.

The figure below shows the general process of determining a reference position. The displayed values are only given as an example. To improve clarity, only a section of the position-value register is shown.

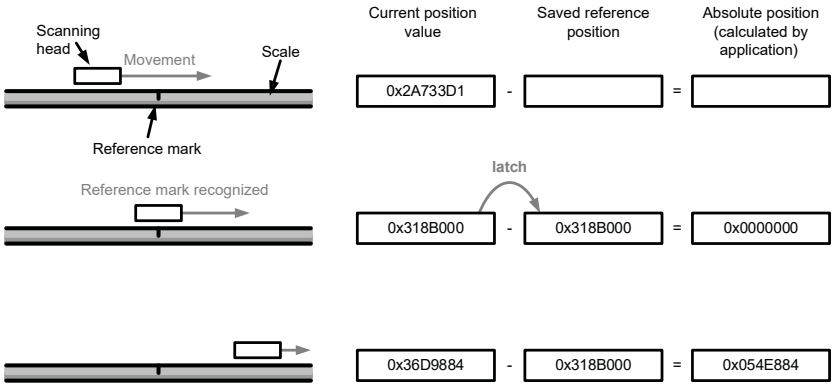

Figure 5: Determining a reference position

Register contents, reference position

Bit no.	Width (bits)	Contents
0 ... 11	12	Always 0
12 ... 43	32	Reference position (period counter value at the time the reference mark was detected; bit 43 = algebraic sign)
44 ... 47	4	Value identical to bit 43

Automatic saving of the reference position must be activated by software. After this command, the EIB 74x waits for the next reference mark and then saves the reference position. You must activate this feature again if renewed saving is desired.

Normally, the reference position register is transmitted together with the position register and the status word in a shared position data packet after the next trigger event. During this process, the EIB 74x transmits two reference positions and, where applicable, the coded reference value:

- Encoders with only one reference mark typically only use reference position 1.
- For encoders with distance-coded reference marks, both register values or the coded reference value will be used, depending on the selected evaluation method.

Distance-coded reference marks

For distance-coded encoders, the reference for generating absolute position values from the counter values is obtained through the distance of two traversed (adjacent) reference marks.

For this purpose, the period counter value is saved twice, once each time a reference mark is traversed. The coded reference value is generated from the distance of the (adjacent) reference marks, and thus, the reference for generating absolute position values is also created.

When the absolute position value is calculated by the customer software application, this value is treated exactly in the same way as a saved reference position value in the case of encoders with only one reference mark (see drawing). The coded reference value thus corresponds to the offset between the absolute position value and the generated (incremental) position value.

There are various procedures for generating the coded reference value:

- Method 1 (recommended method):
  The axis is initialized as an incremental system with distance-coded reference marks. During this process, further type-dependent information about the measuring system is transferred to the EIB 74x. Once the reference positions have been saved successfully, the EIB 74x uses this information to automatically calculate the coded reference value. The save process is initiated by a software command (for two reference marks). After the second reference mark has been traversed, the EIB 74x automatically calculates the coded reference value and transfers it to the customer software application.
- Method 2 (especially for low traversing speed applications):
  The axis is initialized as an incremental system with a single reference mark. The customer software application sends the appropriate software command for saving the reference position (one single reference mark). Each time the reference position has been saved successfully, the save process is activated again. This procedure must be repeated until two different reference positions have been measured. The customer software application can then calculate the coded reference value and hence the absolute position from these two values. It must be guaranteed that the customer software application can complete this procedure quickly enough, otherwise reference marks could be lost, resulting in an incorrect calculation of the absolute position.
- Method 3:
  The axis is initialized as an incremental system with a single reference mark. The customer software application sends the appropriate software command for saving two reference positions. After storing the two reference positions successfully, (both reference position registers will be used), the customer software application calculates the coded reference value and hence the absolute position from these two values.

2.2.3 Monitoring the reference marks

The reference marks of an encoder can be monitored automatically. For this purpose, the reference position is saved and checked continuously. This results in the reference-position output being updated with each reference mark, and therefore can lead to a change in the reference position. The check varies slightly depending on the encoder, as described below. If an error occurs, a bit is set in the status word for the position.

Encoders with one reference mark

For linear encoders with a single reference mark, the position value at the reference mark must always be identical. The reference position is saved continuously and compared to the old value.

For rotational encoders with a single reference mark, the position value may change if the encoder is moved one revolution in the same direction. Two successively saved reference positions must be the same or may differ by the number of signal periods per revolution. The transmitted data packet always contains the current reference position.

Encoders with distance-coded reference marks

For encoders with distance-coded reference marks, the coded reference position will be recalculated continuously. The calculation always uses two adjacent reference positions as illustrated in the figure below. The transmitted data packet always contains the currently calculated reference position.

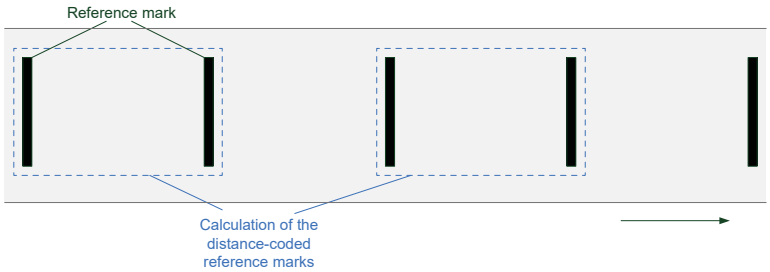

Figure 6: Continuous calculation of the reference positions using distance-coded reference marks

For linear encoders, the calculated reference position must always be identical. For rotational encoders, the calculated reference position may vary by the number of signal periods per revolution. If the same reference mark is traversed twice before and after a change in direction, the distance-coded reference mark cannot be calculated. No check takes place in this case. This must be taken into account in case of minor movements about the reference mark.

No reference run can be performed while monitoring of the reference marks is active, because this could lead to an incorrect error message in the monitoring process. The following order is recommended:

- Configure the axes
- Perform the reference run
- Activate monitoring of the reference marks

2.2.4 Processing EnDat signals

HEIDENHAIN absolute encoders are available with EnDat 2.1 or EnDat 2.2 interface. In addition to the EnDat signals, 1 V_{pp} signals will also be transmitted, especially with EnDat 2.1 encoders. The EIB 74x is able to process the signals from all EnDat encoders with EnDat 2.1 or EnDat 2.2 interface both serially and also with 1 V_{pp} incremental signals.

The EnDat master will be set individually during the initialization of the axis:

- EnDat 2.1 or EnDat 2.2 communication can be set.
- The clock frequency for EnDat communication can be set.
- Delay compensation (EnDat 2.2) can be activated or deactivated.
- The recovery time I can be set if the encoder supports it.
- Monitoring of the calculation time can be set.

Notes on EnDat 01:

- If EnDat position polls and 1 V_{pp} incremental signals are used at the same time, only commands in EnDat 2.1 mode can be sent to the encoder (the axis must be configured for EnDat 01).
- The EnDat position can be imported only by a software command. This means that the EnDat position and the incremental position must be imported once (using a special command). Then, the incremental position can be cyclically transferred.

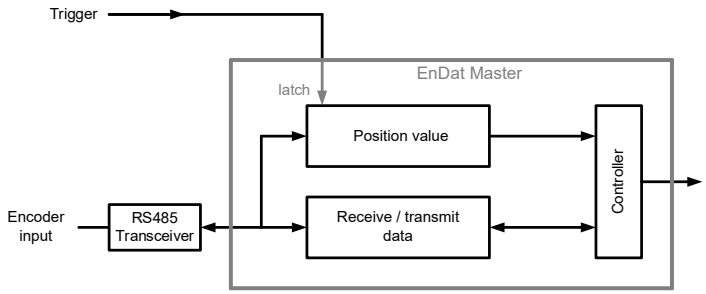

Figure 7: EnDat master block diagram

Position value register

The position register maps the position transmitted via the EnDat interface at the time of the trigger event. The position register for the EnDat position has a width of 48 bits. The number of bits used for the position value depends on the connected EnDat encoder; the uppermost, unused bits must be masked out. See the encoder specifications for more detailed information.

Register contents, position value

Bit no.	Width (bits)	Contents
0 ... 47	48	EnDat position value

Clock frequency

The EnDat clock frequency can be set by a software command. The clock frequency can be set at specific intervals between 100 kHz and 6.66 MHz. The maximum permitted frequency depends both on the length of the cable connecting the encoder and the EIB 74x and on whether a delay compensation has been activated or not.

EnDat master clock frequency

Clock frequency parameter	Clock frequency	Comment
100000	100 kHz	
300000	300 kHz	Default for EnDat 2.1
500000	500 kHz	
1000000	1 MHz	
2000000	2 MHz	Default for EnDat 2.2
4000000	4 MHz	
5000000	5 MHz	
6666666	6.66 MHz	

Delay compensation

Delay compensation for EnDat transmissions can be activated/deactivated when configuring the axis. Delay compensation is not approved by HEIDENHAIN for EnDat 2.1 encoders (except for EnDat encoders with the EnDat21 ordering designation). Delay compensation is approved by HEIDENHAIN for EnDat 2.2 encoders. This results in the following dependency of the maximum permitted EnDat clock frequency:

EnDat clock frequency depending on the cable length

Clock frequency	Cable length in meters (using HEIDENHAIN cables)	
	Without delay compensation	With delay compensation
100 kHz	150	100
300 kHz	150	100
500 kHz	100	100
1 MHz	55	100
2 MHz	10	100
4 MHz	-	50
5 MHz	-	40
6.66 MHz	-	25

Recovery time l

For EnDat 2.2 encoders (ordering designations EnDat02 and EnDat22), you can set the recovery time l. Here, the options are: “long” (10 µs < tm < 30 µs) and "short" (1.25 µs < tm < 3.75 µs). For EnDat 2.1 encoders, the long option is always used for the recovery time l.

Remarks:

- The default setting is "long."
- Select the “short” setting in order to obtain shorter cycle times for EnDat transmissions.
- If you select “short”, make sure to set the EnDat clock frequency to > 1 MHz at the same time.

Calculation time

The calculation time indicates the time for position formation in the encoder and therefore affects the duration of the position poll. In order to monitor the communication, a timeout is generated if the position poll exceeds a certain duration. This is displayed in the status word as an error.

If the calculation time was set too short, this error message can appear even though the encoder transmitted the data correctly. Conversely, an excessively long calculation time can cause an undesired delay in the error message. Particularly in the case of high trigger rates, the error message may be delayed by multiple samples.

The calculation time can be set individually for the connected encoder. Here, the options are: "long" (< 1 ms) and "short" (< 15 µs).

Remarks:

- The default setting is "long."
- If you select “short”, make sure to set the EnDat clock frequency to > 1 MHz at the same time.

EnDat 2.2—additional data

The EnDat 2.2 additional data can be transmitted in various ways in the Soft Real-Time, Streaming, and Recording operating modes.

- No additional data
  A position poll is started with each trigger event. Additional data is not transmitted.
- Fixed additional data
  Besides the position value, with each trigger event, one piece of fixed information is sent as additional datum 1 and additional datum 2. This has to be set before activation of the corresponding operating mode. The setting can only be changed in Polling operating mode. It is also possible to transmit only additional datum 1 or additional datum 2.
- Variable additional data
  The additional data is switched cyclically. The EIB 74x features a ring buffer with 10 entries for setting the additional data, which is processed cyclically. The position value and the additional data 1 and 2 are transmitted with each trigger event. In addition, the EnDat 2.2 transmission supplement is transmitted, over which a new additional datum is selected based on the data in the ring buffer. The additional data 1 and 2 can be mixed in the ring buffer. Only one of the two additional data can be switched over per position poll.

Processing of additional incremental signals with EnDat

If EnDat encoders use the incremental signals for position formation, it is possible to save the EnDat and the incremental position at the same time in order to generate an absolute reference. To do this, a special command is sent to the EIB 74x via the customer software application, which then generates an internal trigger signal. This trigger signal initiates simultaneous position determination via the EnDat interface and via the incremental signals. Both positions are transmitted to the customer software application as return values.

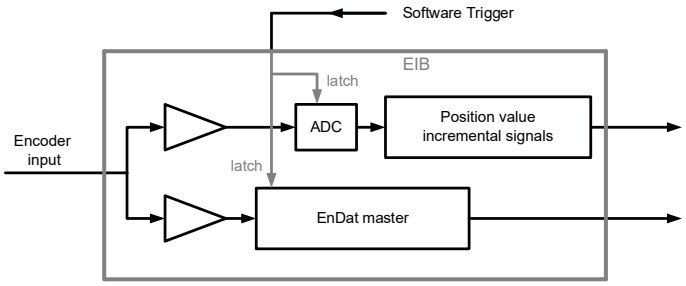

Figure 8: Block diagram for the processing of additional incremental signals with EnDat encoders

Remarks:

- The interpolation datums for the incremental signals and the EnDat position are different and must also be taken into account by the customer software application.
- Furthermore, any differing resolution between EnDat and incremental position must also be taken into account:
  Incremental signals: interpolation datum see "Processing incremental signals", Page 14
  EnDat position: interpolation datum, see graphic

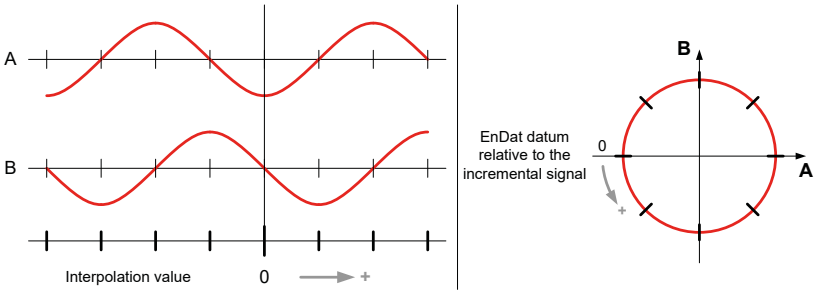

Figure 9: Interpolation value for EnDat with incremental signals

2.2.5 Auxiliary axis

The auxiliary axis is coupled to axis 1 and can be used for encoders with 1 V_{pp} interface. The signals of axis 1 are interpolated and forwarded to a position counter. The interpolation factor can be set at specific intervals. In addition, you can select the edge evaluation (1-fold, 2-fold, or 4-fold).

The maximum permissible input frequency of the encoder signals for the interpolator depends on the interpolation factor and is shown in the table below. In order not to limit the input frequency unnecessarily, set the edge evaluation to 4-fold and, in turn, select a lower interpolation factor. Thus, a 5-fold interpolation factor with 4-fold edge evaluation results in the same increment as a 20-fold interpolation factor with 1-fold edge evaluation, but the maximum permissible input frequency is higher.

Signal period for a linear encoder:

$$\frac{Increment}{\mu m} = \frac{\frac{Signalperiod\ of the\ encoder}{\mu m}}{Interpolationfactor \cdot Edgeevaluation}$$

Signal period for a rotational encoder:

$$\frac{Increment}{1^\circ} = \frac{\frac{360^\circ}{Linecount\ of the\ encoder}}{Interpolationfactor \cdot Edgeevaluation}$$

Maximum input frequency of the auxiliary axis

Interpolation factor	Max. input frequency in kHz[1)]
1-fold	500
2-fold	500
4-fold	500
5-fold	500
10-fold	400
20-fold	200
25-fold	160
50-fold	80
100-fold	40

1) starting with part number xxx-02: Maximum input frequency 70 kHz for homing

Besides the position value, a timestamp and a reference position are available to the auxiliary axis. The status word for the auxiliary axis contains status and error messages. Both the position value and the reference position are 32-bit values. The number of counting steps per signal period of the encoder depends on the interpolation factor for the auxiliary axis. The position value is represented as a two's complement number. Accordingly, an overflow will occur at position `0x7FFF FFFF` (positive maximum) →`0x8000 0000` (negative maximum). If an overflow occurs, it must be handled by the higher-level customer software application.

Register contents, position value for the auxiliary axis

Bit no.	Width (bits)	Contents
0 ... 31	32	Position value of the auxiliary axis (bit 31 = algebraic sign)

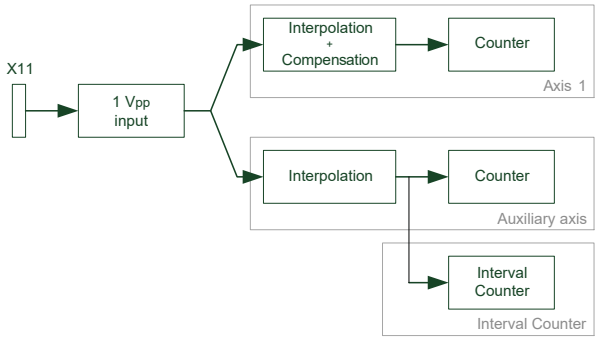

Figure 10: Block diagram of the auxiliary axis and the interval counter

2.3 Processing trigger events

Position value determination inside the EIB 74x is initiated via a so-called trigger event. The EIB 74x supports the following trigger sources:

- 4 external trigger inputs
- Internal periodic trigger source, timer-controlled
- Software command
- Reference pulse of the encoders
- Position trigger (interval counter)

The trigger source must be set by software command for each axis; only one trigger source per axis can be active at a time. However, it is possible to activate different trigger sources for different axes.

Also, one trigger source must be defined as the master trigger source that defines the time of data transmission. For all axes triggered by the master trigger source, a new position is transmitted in each data packet. For all other axes, a valid position is transmitted only if a trigger event also occurred for the respective axis. Otherwise, the position value is marked as invalid.

Not all modes of operation support the features of the trigger interface.

Further information: "Operating modes", Page 34

2.3.1 Trigger inputs and outputs

Four trigger inputs and outputs are supported. For specifications on the trigger input, see the Operating Instructions.

Trigger inputs

Trigger inputs are used to synchronize the position poll with external events. For details on restricting the trigger rate see "Maximum trigger rate", Page 29.

The 120 Ω terminating resistor can be activated or deactivated by configuration.

Trigger outputs

The trigger outputs forward trigger events, e.g to other EIB 74x units. This permits the generation of a trigger chain, which synchronizes multiple EIB 74x units with an external trigger event. The various EIB 74x units must be configured separately by software commands. The position data is sent via the respective Ethernet connection. To generate a trigger chain, the following connection between the EIB 74x units must be used:

- Trigger out + → Trigger in +
- Trigger out – → trigger In –
- GND to GND

A pulse at the trigger output is 2 µs long and is generated synchronously with the system clock of the EIB 74x. The trigger event corresponds to the rising edge of the pulse. If a signal is forwarded from the trigger input to the output, it is affected by jitter due to the synchronization with the system clock. In order to avoid this jitter, it is possible to switch the signal at the trigger input directly to the corresponding trigger output.

The trigger signals can be fed through from the input to the output separately for each channel. Input 1 can be connected to output 1, and so on. The trigger input and the corresponding output are shown in the figure below.

The differential signals "Trigger out +" and Trigger out –" can be swapped in order to change the polarity of the signal at the trigger output. For single-ended signals, correspondingly use the Trigger out – output (see Operating Instructions).

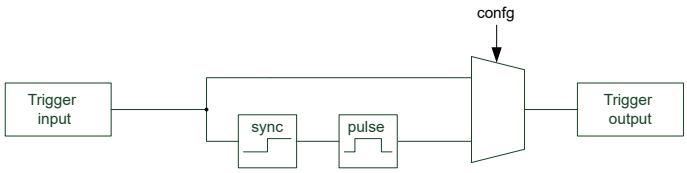

Figure 11: Block diagram for forwarding trigger events

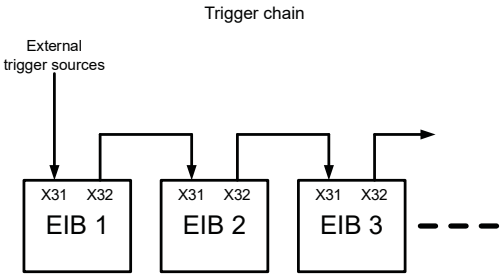

Figure 12: Trigger chain with multiple EIB 74x units

Configuring the trigger inputs and outputs as logical inputs and outputs

The trigger inputs/outputs can also be used as logical inputs/outputs. Trigger inputs and outputs are set by default. The ports can be individually configured as logical inputs and outputs or as trigger inputs and outputs by a software command. It is not possible to configure them as triggers and logical inputs/outputs at the same time.

2.3.2 Logical inputs and outputs

Logical inputs

Each trigger input can be individually converted to a logical input. The level of the corresponding input can be read out by a software command. The 120 Ω terminating resistor can also be activated or deactivated in this mode by configuration.

Logical outputs

Each trigger output can be individually converted to a logical output. The output level can also be read back. The outputs can be individually activated or deactivated, irrespective of the configuration.

The graphic below shows the trigger input and output switching options at a glance. Only one channel is shown.

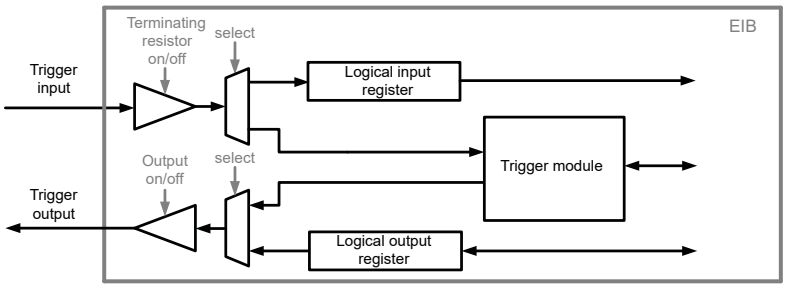

Figure 13: Block diagram of trigger inputs and outputs

2.3.3 Trigger module

The trigger module enables you to select and control the trigger sources. Furthermore, it generates internal trigger signals. You can delay external trigger signals; the delay time can be set for each input separately.

The reference pulse of an encoder with 1 V_{pp} interface can be used as a trigger source for the associated axis. In addition, the reference pulse of axis 1 is gated with the signals A and B of axis 1 by logic AND, and can be used as a trigger signal for any axes. The active edge of the reference signal can be set in each case. If the reference pulse is selected as the trigger source for more than one axis, each axis will be triggered with its own reference pulse. If the reference pulse is also selected as the master trigger source in this case, a reference pulse must occur on each axis before transmitting the data packet. Furthermore, there are four freely assignable channels for software triggers.

The interval counter generates trigger signals depending on the position of the encoder on axis 1. One signal period of the encoder can be divided into multiple counting steps through an adjustable interpolator. The triggering occurs either at a certain position or at equidistant intervals.

The pulse counter is not a separate trigger source; it allows you to limit the number of trigger pulses of other sources. A selectable trigger source can supply pulses that are disabled until the gate is opened by the start signal. All trigger pulses are then counted, and the gate is closed again after a selectable number of pulses. Besides, it is possible to reload the counter while the gate is open. The number of trigger pulses can be increased in this way.

Switch matrix

Trigger source	Trigger output	Axis	Auxiliary axis	Pulse counter trigger	Pulse counter start
Trigger input	x	x	x	x	x
Reference pulse	-	x	x	x	x
Reference pulse	x	x	x	x	x
Interval counter	x	x	x	x	x
Pulse counter	x	x	x	-	-
Software trigger	x	x	x	-	x
Timer	x	x	x	x	-

The switch matrix makes it possible to connect the trigger sources individually to the sinks, such as trigger outputs or axes. However, it is not possible to connect all sources to all sinks. The table above provides an overview of the possible combinations. Only one trigger source is permissible per sink. For the pulse counter, there is a trigger signal whose trigger pulses are controlled via the internal gate. The start signal opens the gate for the trigger pulses.

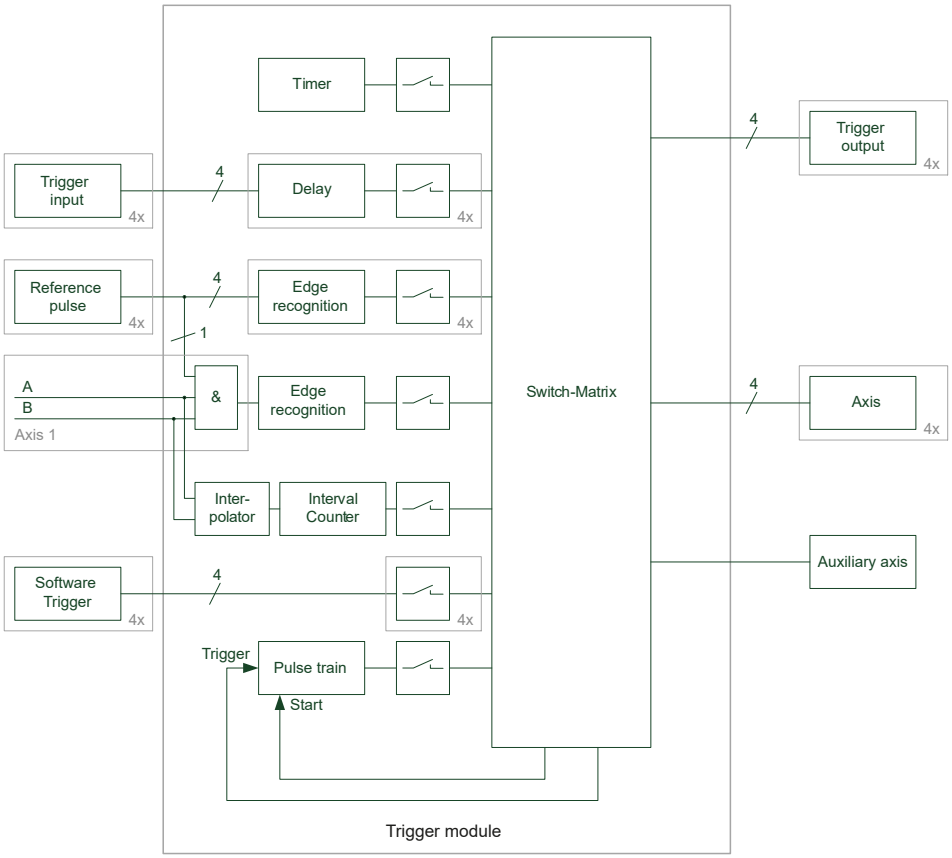

Figure 14: Block diagram of the trigger module

2.3.4 Interval counter

The interval counter permits position-dependent triggering in connection with an incremental encoder on axis 1. For this purpose, the encoder signal can be interpolated (see "Auxiliary axis", Page 23).

The triggering occurs at a certain position, or equidistant trigger pulses with an adjustable position interval are generated. Output of the trigger pulses starts after an adjustable starting position has been traversed, and is then continued at the position interval in both counting directions. The position interval ΔX must be indicated in counting steps (for the calculation of the increment see "Auxiliary axis", Page 23).

A hysteresis prevents multiple triggering, especially in case of a high interpolation factor selected for the encoder signal. After a trigger pulse has been generated at a position, the value of the position counter must change by +H or –H before a new trigger pulse is generated at the same position.

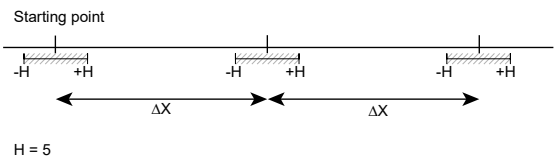

Figure 15: Hysteresis of the interval counter

2.3.5 Maximum trigger rate

The maximum trigger rate of the EIB 74x depends on the selected mode of operation (except for the Polling operating mode):

- Soft Real-Time operating mode: max. 10 kHz
- Recording operating mode: max. 50 kHz
- Streaming operating mode: max. 50 kHz

In Streaming operating mode, the data rate is additionally limited to 1200000 byte/s. The data rate is derived from the size of a data packet and the trigger rate.

$$\frac{Datapacketsize}{Byte} \cdot \frac{Trigger\ rate}{Hz} \leq 1\ 200\ 000$$

It must be guaranteed that the data rate is not limited by the host where the data will be further processed.

A certain interval, which the EIB 74x requires for the position calculation, must be respected between two trigger events. If this interval is not respected, i.e. the trigger rate is too high, then the trigger events cannot be accepted by the EIB 74x and get lost (lost triggers). This is detected by the EIB 74x and displayed in the status word of the position data packet by the "lost trigger" bit. This bit remains set to"1" until actively reset by the customer software application using a clear command.

The above values apply if incremental signals are used. When using the EnDat interface, make sure to take the EnDat transmission time into account.

If the maximum trigger rate is massively exceeded (e.g. by wrong parameterization or if too many events are signaled at the external trigger input), the result may be that the EIB 74x no longer responds to external commands. To make it responsive again, a hardware reset is required.

2.3.6 Counter for accepted trigger events

In addition to monitoring for lost triggers, the EIB 74x has, for further error detection, a counter that is incremented by each incoming and accepted trigger event from the master trigger source. A trigger event is accepted if the above-mentioned interval is maintained. Trigger events resulting in lost triggers are not counted.

The counter value is transmitted in the position data packet and can be monitored for continuity. This makes it possible to detect lost position data packets.

2.4 Timestamp

The "timestamp" function is also used to monitor the data flow. The timestamp counter is a free-running timer with a freely programmable time interval. Each trigger event that results in a position value determination also causes the current timer value to be saved to the timestamp register. If the timestamp function is active, the contents of this register will be transmitted together with the position data packet. This enables the customer software application to check whether the latch time of each individual position value corresponds to the expected value. In applications that do not have a periodical trigger, this register can be used to transmit the time of the trigger event.

The time interval of the timestamp counter is a multiple of the EIB 74x's internal system clock. Before the timestamp can be used, the time interval must be set by a software command. For this purpose, first read out the "clock ticks per µs" value and then set the required time interval depending on this value. This is necessary to keep the software compatibility independent of various settings for the system clock.

To calculate the value of a period (e.g. for the `period` parameter of the `EIB7SetTimestampPeriod`) function call correctly, pass the following values to the function:
`period` = interval in µs * clock ticks per µs
To read out the value of "clock ticks per µs", you can e.g. use the `EIB7GetTimerTriggerTicks` or `EIB7GetTimestampTicks` function.

2.5 Status word

The status word must be interpreted depending on the poll type:

- Incremental position data
- EnDat position data
- Polling of EnDat additional data

The status word is transmitted separately for each encoder channel and does not depend on the operating mode set.

Register contents, status word

Bit no.	Incremental position	EnDat position	EnDat additional datum	Auxiliary axis
0	1 = Valid position	1 = Valid position		1 = Valid position
1	1 = Signal amplitude error	1 = CRC error	1 = CRC error	1 = Signal amplitude error
2	Reserved	Reserved	Reserved	Reserved
3	1 = Frequency exceeded	Reserved	Reserved	1 = Frequency exceeded
4	1 = Encoder power supply error	1 = Encoder power supply error	Reserved	1 = Encoder power supply error
5	1 = Fan error	1 = Fan error	Contents I0	1 = Fan error
6	Reserved	Reserved	Contents I1	Reserved
7	1= Lost trigger	1= Lost trigger	Contents I2	1= Lost trigger
8	1 = Reference position 1 saved	1 = EnDat error message 1	Contents I3	1 = Reference position saved
9	1 = Reference position 2 saved	1 = EnDat error message 2	Contents I4	Reserved
10	1 = Coded reference value for distance-coded reference marks is valid	Reserved	EnDat busy bit	Reserved
11	1 = Error during the calculation of the coded reference value for distance-coded reference marks. Error during reference mark monitoring	Reserved	EnDat RM Bit	Reserved
12	1 = Homing signal is active	Reserved	EnDat WRN bit	Reserved
13	1 = Limit signal is active	Reserved	Reserved	Reserved
14	Reserved	Reserved	Reserved	Reserved
15	Reserved	Reserved	Reserved	Reserved

Meaning of the error bits

Name	Meaning
Valid position	1 → no error occurred This bit indicates whether the transmitted position is valid or not
Valid additional datum	1 → EnDat additional datum was received. Otherwise, no additional datum was selected or it was not received
Signal amplitude error	1 → Signal amplitude of the 1 V_{pp} incremental signals was too low (once or multiple times after this error message was cleared for the last time)
Frequency exceeded	1 → Excessive Input signal frequency was detected (once or multiple times after this error message was cleared for the last time)
CRC error	1 → CRC error during EnDat data transmission
Encoder power supply error	1 → The encoder power supply was switched off (overload cutout responded)
Fan error	1 → The EIB 74x fan is faulty
Lost trigger	Information see "Maximum trigger rate", Page 29
Reference position 1 saved	1 → Reference position 1 was saved (since the last corresponding software command)
Reference position 2 saved	1 → Reference position 2 was saved (since the last corresponding software command)
Coded reference value for distance-coded reference marks is valid	1 → coded reference value for distance-coded reference marks was calculated successfully (since the last corresponding software command)
Error during the calculation of the coded reference value for distance-coded reference marks	1 → Error during the calculation of the coded reference value; must be reset explicitly. An error was detected during automatic monitoring of the reference marks. The error must be cleared explicitly.
Homing signal	1 → Homing signal (L1) is active at the time of the position poll
Limit signal	1 → Limit signal (L2) is active at the time of the position poll
EnDat error message 1	1 → Error message 1 active
EnDat error message 2	1 → Error message 2 active
EnDat busy bit	1 → Busy bit has been set
EnDat RM Bit	1 → RM (reference mark) bit has been set
EnDat WRN bit	1 → WRN (warning) bit has been set
Contents I0 ... I4	These five bits define the contents of the received additional datum. The customer software application requires this information to interpret the data.

The error bits are not reset automatically, but must be reset explicitly using a software command in the customer software application. If you co not reset an error, it will be retransmitted with each new position data packet.

For incremental encoders, an error in the position data packet indicates that the position is no longer valid and has lost its reference to the reference mark or other encoder channels.

The occurrence of one error can initiate others. If an encoder power supply error is reported, other errors will be reported as well. Any error in the power supply must

therefore be reset first. Wait until the power supply is stable again (waiting time approx. 1.5 seconds), reset the other errors.

- Lost Trigger
  This bit indicates that at least one trigger event could not be processed correctly due to a too short time interval between two trigger events. This bit can also be set if malfunctions superimpose the trigger line or EMC influences have a negative effect on the transmission. The `Lost Trigger` bit does not mean that the position values are wrong, it indicates merely that the trigger events could not be processed correctly. The reset must also be done actively by a software command.
- Reference position saved
  The two `Reference position 1 (2) saved` bits indicate that a valid reference mark was detected and saved. This means that the corresponding reference position in the position data packet is valid.
- Coded reference value for distance-coded reference marks is valid
  This bit is reset by sending the corresponding software command for saving reference positions. After the coded reference value has been calculated successfully, this bit is set to active. This means that the “coded reference value for distance-coded reference marks” value transmitted in the position data packet can be used for calculating the absolute position.
- Error in the reference position for distance-coded reference marks
  This bit is set if an error has occurred while the coded reference value for distance-coded reference marks was being calculated. One possible reasonis that during the reference run, a change of direction has occurred, causing the same reference mark to be detected twice. You need to reset this error actively because it will not be reset automatically when the software command for saving reference positions is resent. Furthermore, this bit is set if automatic monitoring of the reference marks has been activated and an error has occurred. This is also true for encoders that are not distance-coded. The error must be cleared explicitly in this case, too.
- Homing/limit signals
  The homing/limit signal indicates whether the corresponding signal is active, provided it is supported by the encoder. The signal state is not saved and does therefore not need to be cleared.
- Fan error
  This bit indicates whether the EIB 74x is working correctly or not. The error bit does not influence the position data. This error bit is not saved and does therefore not need to be cleared. The bit is set as long as fan operation is faulty.

For more information, refer to the Operating Instructions. If fan monitoring is not supported, this bit is always set to "0".

2.6 Ethernet interface

The Ethernet interface (LAN) is used for the configuration of the EIB 74x and for transmitting the position data packets. TCP commands are used for configuration whilst UDP packets are used for transmitting the data in Soft Real-Time operating mode. The settings of the PC firewall must be selected correspondingly.

The network settings of the EIB 74x can be changed by means of the software commands. The IP address can either be permanently set or can be obtained dynamically from a DHCP server.

For more information, refer to the Operating Instructions or see "DHCP", Page 8.

2.7 Operating modes

The EIB 74x supports the following operating modes:

- Polling
- Soft Real-Time
- Streaming
- Recording

2.7.1 Configuration of data packets

You need to configure a data packet for the Soft Real-Time, Streaming, and Recording operating modes. With each trigger event, certain data will be transferred or recorded, depending on this configuration. This makes it possible to limit the quantity of data to the elements actually required. This reduces the required transmission capacity and the memory space needed in Recording operating mode.

A data packet is divided into several regions. Each region contains the data for a certain axis of the EIB 74x or global information. The global information must always be contained in the first region of the data packet. Then one or more regions can follow for the axes. Axes can be omitted here; however the axes must be contained in ascending order in the data packet. For example, "GlobalInfo-Axis1-Axis3-Axis4" would be a valid data packet, but "GlobalInfo-Axis1-Axis4-Axis3" would be invalid. If an auxiliary axis is used, its configuration must be included in the last region of the data packet.

Different data elements can be contained within each region. All possibilities are listed in the table below. The length indicates the number of bytes for the data element. The sum of all elements from all regions is the size of the data packet. However, the length of a data packet must always be a multiple of 4 bytes. If a certain configuration does not meet this requirement, the packet is padded at the end with a corresponding number of fill bytes.

Packet configuration

Region	Data element	Description	Length in bytes
Global	Trigger counter	Counter for trigger events	2
Axis	Status word	Status and error messages	2
	Position value	Current position value of the encoder	6
	Timestamp	Timestamp for position value	4
	Reference position	Position value of reference marks	12
	Coded reference position for distance-coded reference marks	Calculated reference position for distance-coded reference marks	6
	Amplitude value of incremental signal	Byte 0 … 1: signal A Byte 2 ... 3: signal B	4
	EnDat additional datum 1	Byte 0 … 1: status word Byte 2 ... 3: additional datum	4
	EnDat additional datum 2	Byte 0 … 1: status word Byte 2 ... 3: additional datum	4
Auxiliary axis	Status word	Status and error messages	2
	Position value	Current position value of the encoder	4
	Timestamp	Timestamp for position value	4
	Reference position	Position value of reference marks	4

Example

The following example illustrates the configuration of a data packet for two axes. In addition, a region has been added for the global information.

- Global information: Trigger counter
- Axis 1: Incremental interface (1 V_{pp}), one reference mark
- Axis 2: Incremental interface (1 V_{pp}), one reference mark

Packet configuration

Region	Data element	Length in bytes
Global	Trigger counter	2
Axis1	Status word	2
	Position value	6
	Timestamp	4
	Reference position	12
Axis2	Status word	2
	Position value	6
	Timestamp	4
	Reference position	12
	Fill bytes	2

This results in a total data packet length of 52 bytes.

Default configuration

After the device is switched on, the EIB 74x loads a default configuration for the data packet. This configuration comprises the global information and one region each for all four axes. The following table shows the structure of the data packet.

Packet configuration

Region	Data element	Length in bytes
Global	Trigger counter	2
Axis1	Status word	2
	Position value	6
	Timestamp	4
	Reference position 1	6
	Reference position 2	6
	Coded reference value for distance-coded reference marks	6
	Amplitude value of incremental signal	4
Axis2	Status word	2
	Position value	6
	Timestamp	4
	Reference position 1	6
	Reference position 2	6
	Coded reference value for distance-coded reference marks	6
	Amplitude value of incremental signal	4
Axis3	Status word	2
	Position value	6
	Timestamp	4
	Reference position 1	6
	Reference position 2	6
	Coded reference value for distance-coded reference marks	6
	Amplitude value of incremental signal	4
Axis4	Status word	2
	Position value	6
	Timestamp	4
	Reference position 1	6
	Reference position 2	6
	Coded reference value for distance-coded reference marks	6
	Amplitude value of incremental signal	4

2.7.2 Polling operating mode

This operating mode is activated by default after the initialization of the EIB 74x. The position data is determined in the EIB 74x as soon as a corresponding command is received. The EIB 74x transmits the data within the response packet to the customer application.

The diagram below illustrates the sequence of a position poll. A command is sent to the EIB 74x from a customer software application on the PC. The EIB 74x generates the position data and returns it in a TCP packet. The data is transmitted to the application.

Processing of trigger events:

- Since the software influences the time at which the position value is formed, this time can thus not be determined exactly.
- Only software triggers are used for triggering.

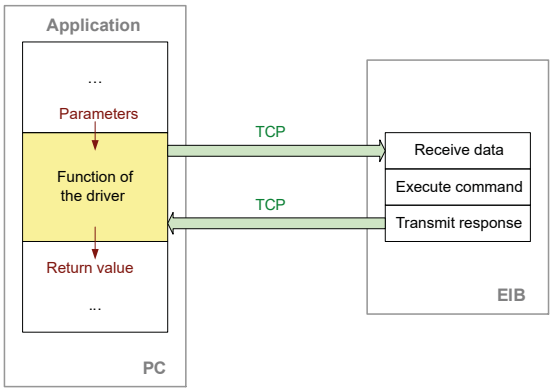

Figure 16: Position poll procedure in Polling operating mode

The data packets in Polling operating mode depend on the selected function, see "Driver software", Page 45.

2.7.3 Soft Real-Time operating mode

The position data is transported with UDP packets from the EIB 74x to the PC. This occurs in parallel to the TCP communication via the standard Ethernet interface. The position data is generated when the EIB 74x receives a trigger signal. With each trigger event, a data packet is sent to the PC automatically. There, the packets can be read from a FIFO.

For Soft Real-Time operating mode, the EIB 74x must be configured following the steps listed below:

- Initialization of the EIB 74x
- Initialization and configuration of the axes
- Configuration of the data packet
- Configuration of the trigger logic
- Selection of the Soft Real-Time operating mode
- Activation of the trigger source

The diagram below illustrates the communication process schematically. The customer software application has to configure the EIB 74x. The data is then transmitted to the FIFO independently. From here, the application can read out the data within a program loop.

In parallel to the position poll, the status of the EIB 74x can be polled or error messages can be cleared.

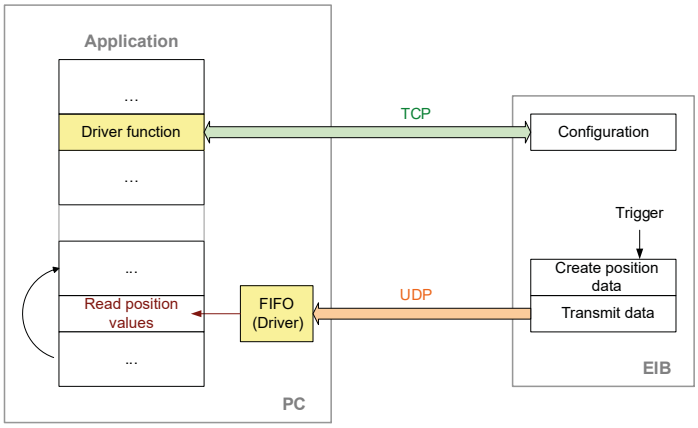

Figure 17: Position poll procedure in Soft Real-Time operating mode

As soon as the position values from the FIFO are read on the PC, this is confirmed to the EIB. If further data is available in the EIB 74x, it will be transmitted.

When the application is closed, the above-mentioned initialization steps must be taken in the reverse order. The trigger source must be deactivated first. The operating mode can then be changed or the connection to the EIB 74x can be closed.

Processing of trigger events:

- External trigger inputs are supported
- Internal trigger sources are supported
- Software triggers are supported

In order to interpret the data correctly when evaluating them, you need to take the data packet structure into account.

If a "Frequency exceeded" or "Lost Trigger" error occurs, it is recommended that the measurement be repeated. To clear the errors, the "Polling" operating mode should be selected temporarily.

2.7.4 Streaming operating mode

The position data is buffered by the EIB 74x and transported to the PC. This occurs in parallel to the TCP communication via the standard Ethernet interface. The position data is generated when the EIB 74x receives a trigger signal. A data packet is generated with each trigger event. Depending on the trigger rate and data volume, multiple data packets are grouped and transmitted to the PC. There, the packets can be read from a FIFO.

For Streaming operating mode, the EIB 74x must be configured following the steps listed below:

- Initialization of the EIB 74x
- Initialization and configuration of the axes
- Configuration of the data packet
- Configuration of the trigger logic
- Selection of the Streaming operating mode
- Activation of the trigger source

The diagram below illustrates the communication process schematically. The customer software application has to configure the EIB 74x. The data is then transmitted to the FIFO independently. From here, the application can read out the data within a program loop.

In parallel to the position poll, the status of the EIB 74x can be polled or error messages can be cleared. In particular, the status of the FIFO in the EIB 74x can be polled in order to detect overflow early.

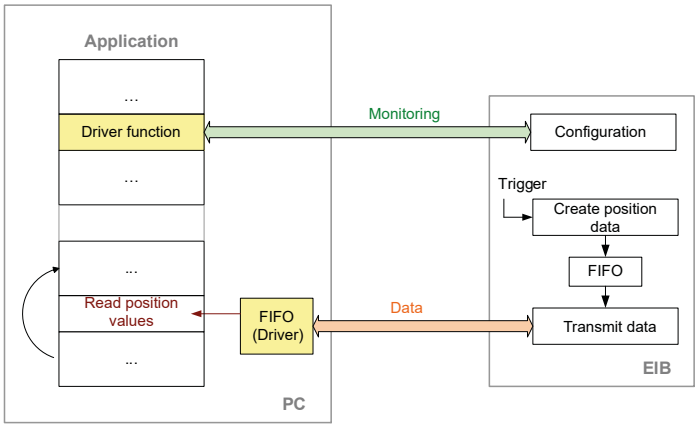

Figure 18: Position poll procedure in Streaming operating mode

As soon as the position values from the FIFO are read on the PC, this is confirmed to the EIB. As further data is available in the EIB 74x, it will be transmitted.

When the application is closed, the above-mentioned initialization steps must be taken in the reverse order. The trigger source must be deactivated first. The operating mode can then be changed or the connection to the EIB 74x can be closed.

Processing of trigger events:

- External trigger inputs are supported
- Internal trigger sources are supported
- Software triggers are supported

In order to interpret the data correctly when evaluating them, you need to take the data packet structure into account.

If a "Frequency exceeded" or "Lost Trigger" error occurs, it is recommended that the measurement be repeated. To clear the errors, the "Polling" operating mode should be selected temporarily.

2.7.5 Recording operating mode

The position data is saved in the memory of the EIB 74x. A data packet is generated and saved with each trigger event. After the recording phase ends, the data can be transmitted.

The Recording operating mode supports two submodes. In Single Shot mode, data recording is automatically ended as soon as the memory is full. In Rolling mode, the data is saved in a ring buffer. When the memory is full, the oldest entry is overwritten. When Recording operating mode is terminated, the last n samples can be read from the memory.

The recording memory size depends on the data packet size and can be read out (see "Recording – reading the memory size", Page 82).

For Recording operating mode, the EIB 74x must be configured following the steps listed below:

- Initialization of the EIB 74x
- Initialization and configuration of the axes
- Configuration of the data packet
- Configuration of the trigger logic
- Selection of the Recording operating mode
- Activation of the trigger source

After the recording phase is concluded, the following steps have to be taken:

- Deactivation of the trigger source
- Selection of the Polling operating mode
- Start data transmission

The diagram below illustrates the communication process schematically. The customer software application has to configure the EIB 74x. In the first phase (recording), the data will be saved in the EIB 74x. In the second phase (data transmission), the data will be transmitted to the host and saved in a FIFO. From here, the application can read out the data within a program loop.

During the recording, the status of the EIB 74x can be interrogated or error messages can be cleared. In particular, the status of the memory in the EIB 74x can be polled.

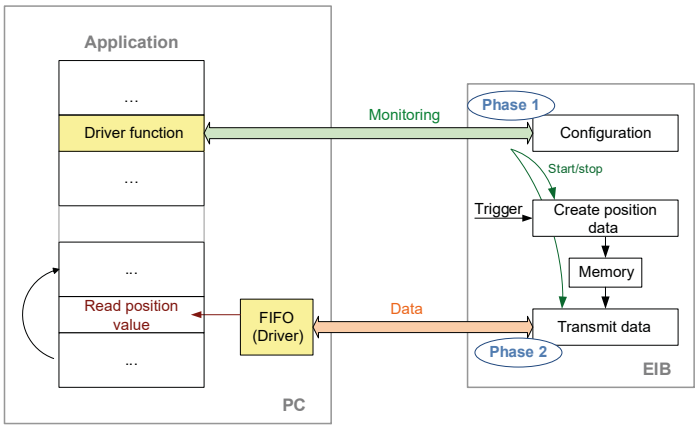

Figure 19: Position poll procedure in Recording operating mode

Processing of trigger events:

- External trigger inputs are supported
- Internal trigger sources are supported
- Software triggers are supported

In order to interpret the data correctly when evaluating them, you need to take the data packet structure into account.

2.8 Firmware update

Users can update the EIB 74x firmware with a TFTP client. However, only special update files from HEIDENHAIN may be installed on the EIB 74x.

2.8.1 Updating the firmware

The example below assumes that the firmware update will be performed for a computer running a Windows operating system. The EIB 74x must be connected to the computer via Ethernet. In this example, the file name for the update is **update_633281-14.flash**. This file is saved under **C:\temp\EIB**.

- Open the Windows command line window
- Save the update file under **C:\temp\EIB\update_633281-14.flash**
- Start the TFTP file transfer **> tftp -i 192.168.1.2**
  put C:\temp\EIB\update_633281-14.flash tmp\update.flash
  - The **-i** option activates binary file transfer
  - IP address: **192.168.1.2** or customer-specific setting
  - **Put** command: Transfer from the host to the EIB 74x
  - The source file in this example is **C:\temp\EIB\update_633281-14.flash**
  - The target file is always **tmp\update.flash**

If the file has been transferred successfully, the TFTP client displays a corresponding message in the command line. The status LED of the EIB 74x is switched off. After the internal data transfer to the flash memory, the status LED will be switched back on. This process can take up to 60 seconds. While the status LED is off, the power supply must not be switched off, nor may commands be sent to the EIB 74x via the Ethernet interface.

When the status LED is active again, make sure to use the relevant software command to poll whether the update has been completed successfully or not. The status of the update process can be polled until the EIB 74x is booted again.

The EIB 74x will boot the new firmware version after the next reset. If an error occurs during the firmware update, the corresponding settings, as indicated in the "Device Resets" chapter of the Operating Instructions, will be used for booting.

If you perform the update on a Linux system, please note that the use of "/" (slash) and "\" (backslash) is different. The EIB 74x requires the backslash, as indicated above. Otherwise, while the update file will be transferred to the EIB 74x, it will not be installed on the EIB 74x.

Example for a Linux system:
user@pc> tftp -v 192.168.1.2 -m binary -c
put update_633281-14.flash tmp\\update.flash

2.8.2 Activating the TFTP client

The instructions below describe how to activate the TFTP client in Microsoft Windows 10. Follow the same procedure to activate the TFTP client in Microsoft Windows 7.

- Select the following in succession in Microsoft Windows:
  - **Start**
  - **Control Panel**
  - **Programs**
  - **Programs and Features**

> The **Programs and Features** dialog opens

- Select the **Turn Windows features on or off** dialog

> The **Turn Windows features on or off** dialog opens

or

- Press the Windows key + "R"

> The **Run** dialog opens

- Enter "appwiz.cpl" via the keyboard
- Confirm your entry with **OK**

> The **Programs and Features** dialog opens

- Select the **Turn Windows features on or off** dialog

> The **Turn Windows features on or off** dialog opens

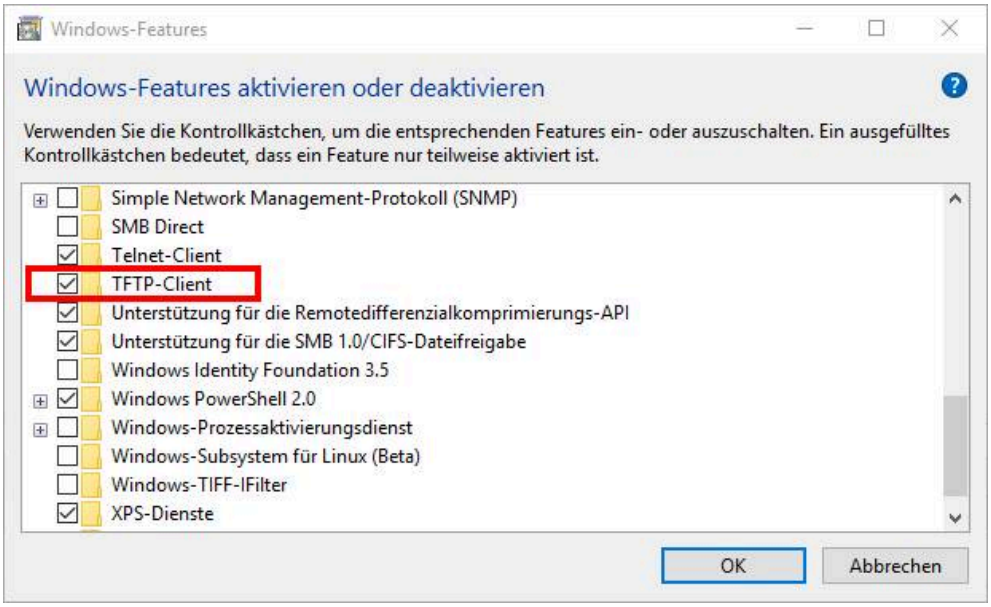

Figure 20: **Turn Windows features on or off** dialog

- Enable the **TFTP Client** by checking the box
- Confirm with **OK**

> The **TFTP client** is available as a command in the command line without restarting the system.

To be able to make optimal use of the TFTP client, some firewall adjustments may be required.

For more information, please refer to Microsoft Support.

2.9 Reset

Please refer to the Operating Instructions

3

Driver software

3.1 General information

Functions are provided for accessing the EIB 74x from a software application. This group of functions is supplied as a DLL file for Windows systems and as SO library for Linux. The following operating systems are supported:

- Windows
- Linux/Unix

If not stated otherwise, both the 32-bit and the 64-bit versions are supported if they are available for the Windows version you use.

Note the system requirements in the ReadMe file included in the installation package. If third parties require any of the function packages, then these should be installed at a later date as needed.

In addition to the libraries, a header file that enables the functions to be integrated into C/C++ programs is also supplied. To create a program, you need to include the library in the project.

".vi"s (virtual instruments) are provided for LabView. They are based on the Windows DLL. The designation, functionality, and input/output parameters of the each ".vi" are in line with the respective function calls, which are documented below. Particularly for complex data types, a ".vi" may require a LabView-specific customization, along with the DLL call. The corresponding customizations are apparent when you open the ".vi".

Support for Visual Basic and C# upon request.

3.2 Installation instructions

The stated directories and files refer to the driver CD for the EIB 74x.

3.2.1 Installation under Microsoft Windows

Before an application can load the DLL, the **eib7.dll** file must be copied from the **EIB_74x\windows\bin** directory to the Windows system directory (e.g. **C:\Windows\system32**).

For 64-bit operating systems, copy the **eib7_64.dll** file from the **EIB_74x\windows\bin64** directory to the Windows system directory (e.g. **C:\Windows\system32**) and then rename it to **eib7.dll**. To ensure compatibility with 32-bit applications, additionally copy the **eib7.dll** file to the **SysWOW64** system directory in the Windows folder (e.g. **C:\Windows**).

Alternatively, the path for the DLL can be declared in the system. The DLL interface is defined via the following two files: **eib7.lib** in **EIB_74x\windows\lib** and **eib7.h** in **EIB_74x\windows\include**. These files must be included in the software project of your development environment (for C/C++ environments). Copy the **eib7.lib** file to the library directory of the development environment or enter its path.

3.2.2 Installation under Linux

For an application to load the SO library for 32-bit operating systems, copy the **libeib7.so** file from the **EIB_74x/linux/lib** directory of the CD to the **usr/local/lib** directory.

For 64-bit operating systems, use the **libeib7_64.so** file from the **EIB_74x/linux/lib64** directory and rename it to **libeib7.so**. The library interface is defined via the **eib7.h** file from the **EIB_74x/linux/include** directory. Copy this file to **usr/local/include** and then include it in the software project in the development environment.

The stated directories are in line with the "Filesystem Hierarchy Standard" for Linux operating systems. The **libeib7.so**" library was compiled for i386 systems under kernel 2.6.

3.3 Overview

3.3.1 Establishing communication

The data packet for the Soft Real-Time, Streaming, and Recording operating modes must be configured before you activate one of the modes.

The configuration data is saved in an array of the `EIB7_DataPacketSection` type. One element of the array must be configured for each region. This can be done using the `EIB7AddDataPacketSection()` function (see "Creating a data packet", Page 56).

Then, you can load this configuration into the EIB 74x using the `EIB7ConfigDataPacket()` function (see "Configuring a data packet", Page 58)

3.3.2 Polling operating mode

The axis functions can be used to access the encoders. For this purpose, you must first configure the axis with `EIB7InitAxis()`. You can then read out the position values or acknowledge error messages.

There is no need to select a trigger source. Triggering is done implicitly with the `EIB7GetPosition()` function call.

3.3.3 Soft Real-Time operating mode

First, initialize the axes with `EIB7InitAxis()` and then configure the data packet and the trigger logic. Then, the Soft Real-Time operating mode can be activated. In Soft Real-Time operating mode, only the error messages from the status word for the position values can be reset.

After you have switched to Soft Real-Time operating mode, the trigger source can be activated. The customer software application on the host must continuously read out the position data from the receive buffer to prevent an overflow. This can be done using the `EIB7ReadFIFOData()` (see "Reading data from the FIFO", Page 83) or `EIB7ReadFIFODataRaw()` (see "Reading and converting data from the FIFO", Page 87) function. Each of these functions reads one or several entries from the FIFO.

Each entry contains a data packet from the EIB 74x. The entry size can be determined beforehand using the `EIB7SizeOfFIFOEntry()` (see "Reading the size of a FIFO element", Page 84) and `EIB7SizeOfFIFOEntryRaw()` (see "Reading the size of a FIFO element after conversion", Page 88) functions.

You can access the individual components of a FIFO entry with the `EIB7GetDataFieldPtr()` or `EIB7GetDataFieldPtrRaw()` function.

It is also possible, by using the callback mechanism, to register a function that will be called as soon as new data is available in the FIFO (see "Activating the callback mechanism", Page 91).

3.3.4 Streaming operating mode

First, initialize the axes with `EIB7InitAxis()` and then configure the data packet and the trigger logic. Then, the Streaming operating mode can be activated. In Streaming operating mode, only the error messages from the status word for the position values can be reset and the buffer status can be read out (see "Checking the streaming status", Page 83).

After you have switched to Streaming operating mode, the trigger source can be activated. The customer software application on the host must continuously read out the position data from the receive buffer to prevent an overflow. This can be done using the `EIB7ReadFIFOData()` (see "Reading and converting data from the FIFO", Page 87) or `EIB7ReadFIFODataRaw()` (see "Reading data from the FIFO", Page 83) functions. Each of these functions reads one or several entries from the FIFO.

Each entry contains a data packet from the EIB 74x. The entry size can be determined beforehand using the `EIB7SizeOfFIFOEntry()` (see "Reading the size of a FIFO element", Page 84) and `EIB7SizeOfFIFOEntryRaw()` (see "Reading the size of a FIFO element after conversion", Page 88) functions.

You can access the individual components of a FIFO entry with the `EIB7GetDataFieldPtr()` or `EIB7GetDataFieldPtrRaw()` function.

It is also possible, by using the callback mechanism, to register a function that will be called as soon as new data is available in the FIFO (see "Activating the callback mechanism", Page 91).

3.3.5 Recording operating mode

First, initialize the axes with `EIB7InitAxis()` and then configure the data packet and the trigger logic. Once the Recording operating mode is active, the data will be saved in the EIB 74x with each trigger event. You can select the trigger source in the Recording mode of operation. The status of the buffer can be read out "Recording – verifying the status".

Once the recording is finished, the data can be transmitted to the host. This is done in Polling mode of operation. The transfer can be started with the `EIB7TransferRecordingData()` function.

The customer software application on the host can read out the position data from the receive buffer with the `EIB7ReadFIFOData()` (see "Reading and converting data from the FIFO", Page 87) or `EIB7ReadFIFODataRaw()` (see "Reading data from the FIFO", Page 83) function. Each of these functions reads one or several entries from the FIFO.

Each entry contains a data packet from the EIB 74x. The entry size can be determined beforehand using the `EIB7SizeOfFIFOEntry()` (see "Reading the size of a FIFO element", Page 84) and `EIB7SizeOfFIFOEntryRaw()` (see "Reading the size of a FIFO element after conversion", Page 88) functions.

You can access the individual components of a FIFO entry with the `EIB7GetDataFieldPtr()` or `EIB7GetDataFieldPtrRaw()` function.

3.3.6 Data types and data packets

Simple data types

- `EIB7_HANDLE`: Handle for EIB 74x
- `EIB7_AXIS`: Handle for an axis on the EIB 74x
- `EIB7_IO`: Handle for an input or output port on the EIB 74x
- `EIB7_ERR`: Error message
- `ENCODER_POSITION`: Position value (64-bit integer)

For the EIB 74x, the following data formats are assumed for integer in C/C++:

- short: 16-bit integer
- long: 32-bit integer
- long long: 64-bit integer

These data formats must be taken into account, especially for 64-bit operating systems.

EnDat additional datum

`struct ENDAT_ADDINFO`

Component	Description
`Status`	Status word for the additional datum
`info`	Data of the additional datum

Information for TCP connection

`struct EIB7_CONN_INFO`

Component	Description
`id`	Identification number for this connection
`local ip`	Local IP address for this connection
`local port`	Local port number for this connection
`remote ip`	IP address of the EIB 74x for this connection
`remote port`	Port number of the EIB 74x for this connection

Configuration for data packet

`struct EIB7_DataPacketSection`

Component	Description
`region`	Global information, or axis of the EIB 74x
`items`	Data elements within the region

3.3.7 Parameters and return values

All functions supply a return value of the `EIB7_ERR` type. This labels a function call as successful or reports an error that occurred during execution.

The input values for the functions are passed as variables (transfer by value). For return values, a pointer to a variable is passed that contains the result after the function has been executed successfully (transfer by reference).

The device, axis, and IO functions can provide the following error messages as standard return values:

Standard return values

`EIB7_NoError`	Function call was successful
`EIB7_CMDErr`	Function call canceled with an error message
`EIB7_InvalidHandle`	The handle to EIB 74x or to the axis on the EIB 74x is invalid
`EIB7_FuncNotSupp`	Function not supported by the EIB 74x
`EIB7_InvalidResponse`	Error during data transmission
`EIB7_AccNotAllowed`	Function cannot be executed because the EIB 74x does not allow access
`EIB7_ConnReset`	Connection closed by the EIB 74x
`EIB7_ConnTimeout`	Timeout during the data transmission to the EIB 74x
`EIB7_ReceiveError`	Error while receiving the data
`EIB7_SendError`	Error while sending the data
`EIB7_OutOfMemory`	The system cannot allocate sufficient memory
`EIB7_PortDirInv`	Signal direction of the port is incorrect (only IO function)

In addition, other return values may be returned. These are indicated separately for each function.

3.4 Auxiliary functions

3.4.1 Determining the IP address

The host name of the EIB 74x or its IP address (as a C string) is converted into an IP address in "host byte order". The name must be passed as a C string. Examples are "192.168.1.2" or "EIB74x-SN1234567".

Function

```
EIB7_ERR EIB7GetHostIP          ( const char*          hostname,
                                  unsigned long*       ip
                                )
```

Parameters

`Host name`	Pointer to a C string containing the IP address or the host name of the EIB 74x.
`ip`	*[return value]* Pointer to a variable to which the IP address of the EIB 74x is saved

Return value

The return value supplies a status for the function call. Possible values are listed below:

`EIB7_NoError`	Function call was successful
`EIB7_HostNotFound`	IP address could not be determined

3.4.2 Changing the position data format

The data format of a position value is converted from a 64-bit integer format to double format. The function can be used only for incremental encoders. The converted value has the unit "1 signal period." The period counter value corresponds to the part of the result before the decimal point; the decimal places are formed from the interpolation value.

Function

```
EIB7_ERR EIB7IncrPosToDouble    ( ENCODER_POSITION    src,
                                  double*             dest
                                )
```

Parameters

`src`	Position value of an incremental encoder
`dest`	*[return value]* Pointer to a variable to which the converted position is saved

Return value

The return value supplies a status for the function call. Possible values are listed below:

`EIB7_NoError`	Function call was successful
`EIB7_ParamInvalid`	Passed position value invalid

3.5 Device functions

The device functions always refer to the entire EIB 74x. There is no distinction between the individual axes. In some functions, the parameters affect all axes.

Besides the standard return values ("Parameters and return values"), all device functions may also return further values. These are indicated separately for each function.

3.5.1 Opening a connection to the EIB 74x

A TCP connection is established with the EIB 74x. This does not result in any changes in the settings in the EIB 74x. If it is not possible to establish the connection, an error message will be returned. The driver must be compatible with the EIB 74x firmware to function correctly. This is verified once the connection is established. If necessary, the EIB 74x firmware version can be read out using this function. To do this, the `ident` parameter must be used to pass the address of a memory area to which the version number is written as a C string.

Function

```
EIB7_ERR EIB7Open        ( unsigned long            ip,
                           EIB7_HANDLE*             eib,
                           long                     timeout,
                           char*                    ident,
                           unsigned long            len
                         )
```

Parameters

`ip`	IP address in "host byte order"
`eib`	*[return value]* Handle for the EIB 74x if the function was completed successfully
`timeout`	Timeout for the following commands in milliseconds (not valid for `EIB7Open()`)
`ident`	*[return value]* Pointer to the target memory to which the firmware version of the EIB 74x is saved as a C string. The size of this memory must be at least 9 bytes. If this parameter is a NULL pointer, the firmware version of the EIB 74x will not be read out
`len`	Size of the target memory in bytes (0 if `ident` = NULL)

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantInitWinSock`	Unable to initialize the operating system socket layer (only on Windows)
`EIB7_CantOpenSocket`	System resources for the connection not available
`EIB7_OutOfMemory`	Not enough memory available
`EIB7_IFVersionInv`	EIB 74x firmware is incompatible with the driver
`EIB7_CantConnect`	The connection cannot be established (the EIB 74x is either switched off or cannot be reached)

3.5.2 Closing the connection to the EIB 74x

The connection to the EIB 74x is being closed. The EIB handle may not used any longer. Likewise, all handles to axes generated from this EIB handle are invalid. If a special operating mode of the EIB 74x has been activated via this handle, Polling operating mode will be activated when the connection is closed. All other settings in the EIB 74x are retained.

Function

```
EIB7_ERR EIB7Close          ( EIB7_HANDLE            eib
                            )
```

Parameters

`eib`	EIB handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.3 Polling the connection status

The status of the connection to the EIB 74x is polled. This makes it possible to determine whether a connection has already been closed or a communication error has occurred. This function will not transfer any data to the EIB 74x. The status refers to the previous commands.

Function

```
EIB7_ERR EIB7GetConnectionStatus   ( EIB7_HANDLE         eib,
                                     EIB7_CONN_STATUS*   status
                                   )
```

Parameters

`eib`	EIB handle
`Status`	*[return value]* Pointer to the target variable for the status

`Status`	Description
`EIB7_CS_Connected`	Connection to the EIB 74x established
`EIB7_CS_Closed`	No connection to the EIB 74x
`EIB7_CS_Timeout`	Timeout during data transfer
`EIB7_CS_ConnectionReset`	The connection has been closed by the EIB 74x
`EIB7_CS_TransmissionError`	Transmission error occurred

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.4 Setting up the timeout

The timeout for the TCP connection to the EIB 74x is reset. This value is applicable to all subsequent function calls. The timeout must be at least 100 ms. Lower values are automatically increased to 100.

Function

```
EIB7_ERR EIB7SetTimeout     ( EIB7_HANDLE            eib,
                              long                   timeout
                            )
```

Parameters

`eib`	EIB handle
`timeout`	Timeout in milliseconds (≥ 100)

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_IllegalParameter`	Timeout value cannot be set

3.5.5 Reading out the number of axes

The number of axes with D-sub input in the EIB 74x is read out.

Function

```
EIB7_ERR EIB7GetNumOfAxes   ( EIB7_HANDLE            eib,
                              unsigned long*         dsub,
                              unsigned long*         res1,
                              unsigned long*         res2,
                              unsigned long*         res3
                            )
```

Parameters

`eib`	EIB handle
`dsub`	*[return value]* Pointer to the target variable for the number of axes with D-sub input
`res1`	*[Return value]* Reserved
`res2`	*[Return value]* Reserved
`res3`	*[Return value]* Reserved

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.6 Requesting handle for axis

The handles for access to the axes of the EIB 74x are generated. They are each saved in an array whose size must also be transferred as a parameter. The return value contains the number of valid handles. The function returns one handle for each axis of the EIB 74x, but only as many as can be accommodated by the array (`size` parameter). The handles are stored in the array in ascending order, starting with axis 1.

Function

```
EIB7_ERR EIB7GetAxis        ( EIB7_HANDLE           eib,
                              EIB7_AXIS*            set,
                              unsigned long         size,
                              unsigned long*        len
                            )
```

Parameters

`eib`	EIB handle
`set`	*[return value]* Pointer to the first array element
`size`	Maximum number of entries in the array
`len`	*[return value]* Number of valid entries in the array

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.7 Requesting IO port handles

Handles are generated for access to the IO ports of the EIB 74x. The handles for the inputs and the outputs are each saved in an array whose size must also be transferred as a parameter. The number of valid handles in the array is indicated in the `ilen` or `olen` parameter, respectively. The function returns one handle for each IO port of the EIB 74x, but only as many as can be accommodated by the array (`isize` and `osize` parameters).

Function

```
EIB7_ERR EIB7GetIO          ( EIB7_HANDLE            eib,
                              EIB7_IO*               iset,
                              unsigned long          isize,
                              unsigned long*         ilen,
                              EIB7_IO*               oset,
                              unsigned long          osize,
                              unsigned long*         olen
                            )
```

Parameters

`eib`	EIB handle
`iset`	*[return value]* Pointer to the first element of the array with the input handles
`isize`	Maximum number of entries in the `iset` array
`ilen`	*[return value]* Number of valid entries in the `iset` array
`oset`	*[return value]* Pointer to the first element of the array with the output handles
`osize`	Maximum number of entries in the `oset` array
`olen`	*[return value]* Number of valid entries in the `oset` array

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.8 Creating a data packet

You can use this function to write the configuration for the data packet. For each function call, one element from the array will be initialized for the configuration data. The index indicates the element, the first element having the index 0. Each element consists of a region and the data elements. The region specifies the axis or the global information. Different data elements can be added for each region. All data elements for a region must be OR gated and be passed as the `items` parameter. The data elements are specified as `EIB7_DataPacketItem`.

Function

```
EIB7_ERR EIB7AddDataPacketSection  ( EIB7_DataPacketSection*   packet,
                                     unsigned long             index,
                                     EIB7_DataRegion           region,
                                     unsigned long             items
                                   )
```

Parameters

`packet` Pointer to the array for the configuration data

`index` Index of the array element

`region` Region of the data packet

`region`	Description
`EIB7_DR_Global`	Global information
`EIB7_DR_Encoder1`	Data for axis 1
`EIB7_DR_Encoder2`	Data for axis 2
`EIB7_DR_Encoder3`	Data for axis 3
`EIB7_DR_Encoder4`	Data for axis 4
`EIB7_DR_AUX`	Data for auxiliary axis

`items` Data elements within the region (OR-operation with multiple elements is possible)

`items`	Description
`EIB7_PDF_TriggerCounter`	Trigger counter (only in `EIB7_DR_Global`)
`EIB7_PDF_StatusWord`	Status word for position
`EIB7_PDF_PositionData`	Position value
`EIB7_PDF_Timestamp`	Timestamp for position
`EIB7_PDF_Analog`	ADC values for signals A and B
`EIB7_PDF_ReferencePos`	Reference positions 1 and 2
`EIB7_PDF_DistCodedRef`	Coded reference value
`EIB7_PDF_EnDat_AI1`	EnDat 2.2 additional datum 1
`EIB7_PDF_EnDat_AI2`	EnDat 2.2 additional datum 2

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.5.9 Configuring a data packet

The data packet for the Soft Real-Time, Streaming, and Recording operating modes can be configured. The configuration is possible only in the Polling mode of operation. The configuration will be adopted as soon as another operating mode (except for Polling) is activated.

Function

```
EIB7_ERR EIB7ConfigDataPacket      ( EIB7_HANDLE              eib,
                                     EIB7_DataPacketSection*  packet,
                                     unsigned long            size
                                   )
```

Parameters

`eib`	EIB handle
`packet`	Pointer to the array with the configuration data for the data packet
`size`	Number of entries in the `packet` array

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid`	Invalid parameter
`EIB7_PacketTooLong`	The configuration data describe an excessively long data packet
`EIB7_InvalidPacket`	The configuration data describe an invalid data packet

3.5.10 Selecting the operating mode

The operating mode for the EIB 74x can be set. The Polling, Soft Real-Time, Streaming, and Recording operating modes are supported. In the Recording operating mode, you can choose between Single Shot and Rolling operation.

Function

```
EIB7_ERR EIB7SelectMode     ( EIB7_HANDLE            eib,
                              EIB7_OPERATING_MODE    mode
                            )
```

Parameters

`eib` EIB handle

`mode` Operating mode

`mode`	Description
`EIB7_OM_Polling`	Polling operating mode
`EIB7_OM_SoftRealtime`	Soft Real-Time operating mode
`EIB7_OM_Streaming`	Streaming operating mode
`EIB7_OM_RecordingSingle`	Recording operating mode – Single Shot
`EIB7_OM_RecordingRoll`	Recording operating mode – Rolling

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantOpenSocket`	Internal error (socket error)
`EIB7_CantStartThread`	Internal error (thread error)
`EIB7_InvalidOpMode`	Selected operating mode not supported
`EIB7_OpModeActive`	Selected operating mode already active
`EIB7_OpModeBlocked`	Selected operating mode cannot be activated
`EIB7_InvalidIPAddr`	Internal error (IP address error)

3.5.11 Saving network parameters

The parameters for the Ethernet interface of the EIB 74x can be set. This means the EIB 74x can be adapted to the network. The settings will only take effect after the next boot process. If the DHCP client is active, the EIB 74x tries to obtain an IP address from the DHCP server. If the server does not respond within the specified timeout, the configured IP address will be used.

Function

```
EIB7_ERR EIB7SetNetwork     ( EIB7_HANDLE           eib,
                              unsigned long         ip,
                              unsigned long         netmask,
                              unsigned long         gateway,
                              EIB7_MODE             dhcp,
                              unsigned long         timeout
                            )
```

Parameters

`eib` EIB handle

`ip` IP address of the EIB 74x in "host byte order"

`netmask` Network mask for the network in "host byte order"

`gateway` IP address of the standard gateway in "host byte order"

`dhcp` Flag for the DHCP client in the EIB 74x

`dhcp`	Description
`EIB7_MD_Disable`	Deactivate DHCP client
`EIB7_MD_Enable`	Activate DHCP client

`timeout` Timeout for the DHCP client in seconds

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantSaveCustNW` Network settings could not be saved

`EIB7_CantSaveDHCP` DHCP timeout could not be saved

`EIB7_DHCPTimeoutInv` DHCP timeout is invalid

`EIB7_ParamInvalid` Parameters are not a valid network configuration

3.5.12 Reading out the network parameters

The parameters for the Ethernet interface can be read out. The output always contains the user-defined settings, even if standard settings were used for booting.

Function

```
EIB7_ERR EIB7GetNetwork     ( EIB7_HANDLE           eib,
                              unsigned long*        ip,
                              unsigned long*        netmask,
                              unsigned long*        gateway,
                              EIB7_MODE*            dhcp
                            )
```

Parameters

`eib`	EIB handle
`ip`	*[return value]* Pointer to the variable for the IP address in "host byte order"
`netmask`	*[return value]* Pointer to the variable for the network mask in "host byte order"
`gateway`	*[return value]* Pointer to the variable for the IP address of the standard gateway in "host byte order"
`dhcp`	*[return value]* Pointer to the variable for DHCP client flag

`dhcp`	Description
`EIB7_MD_Disable`	DHCP client deactivated
`EIB7_MD_Enable`	DHCP client activated

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_NoCustNetwork`	No customer-specific (user-defined) settings present

3.5.13 Saving the host name

The host name of the EIB 74x is saved. The name must be passed as a C string, which can have a maximum length of 32 characters including the null byte. If it is any longer, the rest will be cut off (truncated). If a string with the length null or a NULL pointer is passed, the EIB 74x sets the host name to the factory default setting.

Function

```
EIB7_ERR EIB7SetHostname    ( EIB7_HANDLE           eib,
                              const char*           hostname
                            )
```

Parameters

`eib`	EIB handle
`Host name`	Pointer to the new host name

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_HostnameTooLong`	Host name too long
`EIB7_HostnameInvalid`	Host name invalid
`EIB7_CantSaveHostn`	Host name cannot be saved
`EIB7_CantRestDefHn`	Default host name cannot be loaded

3.5.14 Reading out the host name

The host name of the EIB 74x is read out and saved in the target memory as a C string. The string has a maximum length of 32 characters (including a null byte). If the target memory is too small to take the entire string, only the first part will be copied.

Function

```
EIB7_ERR EIB7GetHostname    ( EIB7_HANDLE           eib,
                              char*                 hostname,
                              unsigned long         len
                            )
```

Parameters

`eib`	EIB handle
`Host name`	*[return value]* Pointer to the target memory for the host name
`len`	Size of the target memory in bytes

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantRdHostname`	Host name cannot be read

3.5.15 Reading out the serial number

The serial number of the EIB 74x is returned as a C string. The string is written to the target memory. If the target string is too short for the serial number, an error message will be displayed. The serial number may have a maximum length of 24 characters (including a null byte).

Function

```
EIB7_ERR EIB7GetSerialNumber    ( EIB7_HANDLE         eib,
                                  char*               serial,
                                  unsigned long       len
                                )
```

Parameters

`eib`	EIB handle
`serial`	*[return value]* Pointer to the target memory for the serial number
`len`	Size of the target memory in bytes

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantRdHostname`	Serial number cannot be read
`EIB7_BufferTooSmall`	Target memory is too small

3.5.16 Reading out the device ID

The device ID of the EIB 74x is returned as a C string. The string is written to the target memory. If the target string is too short for the ID, an error message will be displayed. The ID may have a maximum length of 16 characters (including a null byte).

Function

```
EIB7_ERR EIB7GetIdentNumber    ( EIB7_HANDLE          eib,
                                 char*                ident,
                                 unsigned long        len
                               )
```

Parameters

`eib`	EIB handle
`ident`	*[return value]* Pointer to the target memory for the device ID
`len`	Size of the target memory in bytes

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantRdIdent`	The device ID cannot be read
`EIB7_BufferTooSmall`	Target memory is too small

3.5.17 Reading out the MAC address

The MAC address of the EIB 74x is generated. The address is output in binary format. The target memory must be at least 6 bytes. The first 6 bytes are always used. The least significant byte of the MAC address is copied to the first byte of the target memory.

Example for `00:A0:CD:85:00:01`:

Offset	Memory contents
0	`0x01`
1	`0x00`
1	`0x85`
1	`0xCD`
1	`0xA0`
1	`0x00`

Function

```
EIB7_ERR EIB7GetMAC            ( EIB7_HANDLE         eib,
                                 unsigned char*      mac
                               )
```

Parameters

`eib`	EIB handle
`mac`	*[return value]* Pointer to the target memory for the MAC address

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.18 Reading out the firmware version number

The version number of the EIB 74x firmware is read out. The `select` parameter determines the firmware for which the version number is returned as a C string. For the string, including the null byte, the target memory must be at least 9 bytes. If the target memory is too small to take the entire string, only the first part is copied.

Function

```
EIB7_ERR EIB7GetVersion     ( EIB7_HANDLE           eib,
                              char*                 ident,
                              unsigned long         len,
                              EIB7_FIRMWARE         select
                            )
```

Parameters

`eib`	EIB handle
`ident`	*[return value]* Pointer to the target memory for the firmware version number
`len`	Size of the target memory in bytes
`select`	Selects the firmware whose version number will be read out

`select`	Description
`EIB7_FW_CurrentlyBooted`	Firmware currently loaded
`EIB7_FW_Factory`	Firmware of the factory default settings
`EIB7_FW_User`	Firmware of the last update

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.19 Reading out the boot mode

The boot mode in which the EIB 74x was started during the last boot process is read out.

Function

```
EIB7_ERR EIB7GetBootMode    ( EIB7_HANDLE           eib,
                              EIB7_BOOT_MODE*       mode
                            )
```

Parameters

`eib` EIB handle

`mode` *[return value]* Pointer to the variable for the boot mode

`mode`	Description
`EIB7_BM_User`	Firmware of the last update with user's network settings
`EIB7_BM_FactoryUser`	Firmware on delivery (factory default settings) with user network settings
`EIB7_BM_FactoryDefault`	Firmware on delivery (factory default settings) with standard network settings

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.20 Reading out the update status

The status can be read out to verify whether an update was completed successfully. After having read the status, this function call resets the status back to default (`EIB7_US_NoUpdate`). The status information is cleared with each boot process.

Function

```
EIB7_ERR EIB7UpdateState    ( EIB7_HANDLE           eib,
                              EIB7_UPDATE_STATUS*   status
                            )
```

Parameters

`eib` EIB handle

`Status` *[return value]* Pointer to the variable for the update status

`select`	Description
`EIB7_US_NoUpdate`	No update loaded
`EIB7_US_UpdateFailed`	The update failed
`EIB7_US_UpdateSuccessful`	Update performed successfully
`EIB7_US_VersionIncompatible`	The firmware is not compatible with the EIB 74x hardware

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.21 Reading the number of open connections

The number of currently open connections to the EIB 74x is generated. This also includes semi-open connections that the remote station has already closed, but that are still open on the EIB 74x.

Function

```
EIB7_ERR EIB7GetNumberOfOpenConnections    ( EIB7_HANDLE       eib,
                                             unsigned long*    cnt
                                           )
```

Parameters

`eib`	EIB handle
`cnt`	*[return value]* Pointer to the variable for the number of open connections

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.22 Reading out the connection data

The connection data of all currently open connections to the EIB 74x can be read out. An entry in the array will be assigned to each connection, but the maximum number of entries is restricted by the `size` parameter. The number of valid elements in the array is returned by the `cnt` parameter. For more information on the contents of the connection data, see "Data types and data packets", Page 49.

Function

```
EIB7_ERR EIB7ConnectionInfo    ( EIB7_HANDLE         eib,
                                 EIB7_CONN_INFO*     info,
                                 unsigned long       size,
                                 unsigned long*      cnt
                               )
```

Parameters

`eib`	EIB handle
`info`	*[return value]* Pointer to the first element in the array for the connection data
`size`	Size of the `info` array
`cnt`	*[return value]* Pointer to the variable for the number of valid elements in the array

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.23 Terminating the connection

An open connection to the EIB 74x can be terminated. It is not possible to terminate the connection used for the function call. This function should mainly be used to terminate semi-open connections that could not be closed properly e.g. due to an error at the host. The ID is contained in the `EIB7_CONN_INFO` connection data (see "Reading out the connection data", Page 66).

Function

```
EIB7_ERR EIB7TerminateConnection   ( EIB7_HANDLE      eib,
                                     unsigned long    id
                                   )
```

Parameters

`eib`	EIB handle
`id`	ID of the connection to be terminated

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantTermConn`	Connection cannot be terminated
`EIB7_CantTermSelf`	Connection cannot terminate itself
`EIB7_ParamInvalid`	The parameter is not a valid index for a connection

3.5.24 Reading the timestamp ticks

The timestamp counter is supplied from a clock-pulse source. The timestamp ticks indicate how many clock pulses per microsecond are generated by the clock-pulse source.

Function

```
EIB7_ERR EIB7GetTimestampTicks     ( EIB7_HANDLE      eib,
                                     unsigned long*   ticks
                                   )
```

Parameters

`eib`	EIB handle
`ticks`	*[return value]* Pointer to the variable for the number of clock pulses per microsecond

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.25 Setting the timer stamp period duration

The period duration of the freely running timestamp counter can be set. To do this, indicate the length of the timestamp period in timestamp ticks. This value must be a natural number greater than zero.

> To calculate the value of a period (e.g. for the `period` parameter of the `EIB7SetTimestampPeriod`) function call correctly, pass the following values to the function:
> `period` = interval in µs * clock ticks per µs
> To read out the value of "clock ticks per µs", you can e.g. use the `EIB7GetTimerTriggerTicks` or `EIB7GetTimestampTicks` function.

Function

```
EIB7_ERR EIB7SetTimestampPeriod    ( EIB7_HANDLE      eib,
                                     unsigned long    period
                                   )
```

Parameters

`eib`	EIB handle
`period`	Ticks per timestamp period (> 0), for further details, see "Timestamp", Page 30

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid`	Invalid timestamp period

3.5.26 Resetting the timestamp counter

The timestamp counter is set to zero and continues counting from this value.

Function

```
EIB7_ERR EIB7ResetTimestamp       ( EIB7_HANDLE      eib
                                  )
```

Parameters

`eib`	EIB handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.27 Timer trigger – reading the time unit

The timer trigger is supplied from a clock-pulse source. The timer-trigger ticks indicate how many clock pulses per microsecond are generated by the clock-pulse source.

Function

```
EIB7_ERR EIB7GetTimerTriggerTicks  ( EIB7_HANDLE      eib,
                                     unsigned long*   ticks
                                   )
```

Parameters

`eib`	EIB handle
`ticks`	*[return value]* Pointer to the variable for the number of clock pulses per microsecond

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.28 Timer trigger – setting the period duration

The period duration of the timer trigger can be set. To do this, the number of timer-trigger ticks making up a period must be stated. This value must be a natural number greater than zero. If the timer trigger is activated, it will initiate a trigger event after each period.

Further information: "Maximum trigger rate", Page 29

To calculate the value of a period (e.g. for the `period` parameter of the `EIB7SetTimestampPeriod`) function call correctly, pass the following values to the function:
`period` = interval in µs * clock ticks per µs
To read out the value of "clock ticks per µs", you can e.g. use the `EIB7GetTimerTriggerTicks` or `EIB7GetTimestampTicks` function.

Function

```
EIB7_ERR EIB7SetTimerTriggerPeriod    ( EIB7_HANDLE         eib,
                                        unsigned long       period
                                      )
```

Parameters

`eib`	EIB handle
`period`	Ticks per timer trigger period (> 0), for further details, see "Timestamp", Page 30

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid`	The timer trigger period is invalid

3.5.29 Reading the time unit for the delay time at the trigger inputs

The trigger input delay ticks indicate how many clock pulses per microsecond are generated by the clock-pulse source for the delay time of the signals at the trigger input. The delay time of the trigger signals can be set to a multiple of the internal clock-pulse period.

Function

```
EIB7_ERR EIB7GetTriggerDelayTicks  ( EIB7_HANDLE      eib,
                                     unsigned long*   ticks
                                   )
```

Parameters

`eib`	EIB handle
`ticks`	*[return value]* Pointer to the target variable for the number of clock pulses per microsecond

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.30 Clearing the trigger counter

The trigger counter is set to zero.

Function

```
EIB7_ERR EIB7ResetTriggerCounter   ( EIB7_HANDLE      eib
                                   )
```

Parameters

`eib`	EIB handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.31 Software trigger

The software trigger generates a trigger event and causes the EIB 74x to send data to the remote station. The `source` parameter is used to select one of the software trigger channels. This function cannot be performed in Polling mode of operation.

Function

```
EIB7_ERR EIB7SoftwareTrigger    ( EIB7_HANDLE         eib,
                                  unsigned long       source
                                )
```

Parameters

`eib` EIB handle

`source` Software trigger channel

source	Description
EIB7_ST_SWtrigger1	Software trigger channel 1
EIB7_ST_SWtrigger2	Software trigger channel 2
EIB7_ST_SWtrigger3	Software trigger channel 3
EIB7_ST_SWtrigger4	Software trigger channel 4

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.5.32 Selecting the master trigger source

The master trigger signal can be selected from different sources. This function must be performed for the axes after configuring the trigger matrix and is permitted only in the Polling mode of operation.

Function

```
EIB7_ERR EIB7MasterTriggerSource  ( EIB7_HANDLE           eib,
                                    EIB7_AxisTriggerSrc   src
                                  )
```

Parameters

`eib` EIB handle

`src` Trigger source

src	Description
`EIB7_AT_TrgInput1`	Trigger input, channel 1
`EIB7_AT_TrgInput2`	Trigger input, channel 2
`EIB7_AT_TrgInput3`	Trigger input, channel 3
`EIB7_AT_TrgInput4`	Trigger input, channel 4
`EIB7_AT_TrgSW1`	Software trigger channel 1
`EIB7_AT_TrgSW2`	Software trigger channel 2
`EIB7_AT_TrgSW3`	Software trigger channel 3
`EIB7_AT_TrgSW4`	Software trigger channel 4
`EIB7_AT_TrgRI`	Reference pulse of the corresponding axis
`EIB7_AT_TrgRImaskedCH1`	Linked reference pulse from axis 1 (A&B&RI)
`EIB7_AT_TrgIC`	Interval counter
`EIB7_AT_TrgPuls`	Trigger pulse counter
`EIB7_AT_TrgTimer`	Timer trigger

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.5.33 Activating trigger sources

The trigger sources of the EIB 74x can be activated or deactivated individually or together. In the `src` parameter, several trigger sources can be selected through an OR operation of the corresponding constants. The period duration for the timer trigger must be configured before activation. If you program the activation of multiple trigger sources by one single function call, they will be activated simultaneously.

Function

```
EIB7_ERR EIB7GlobalTrigger-Enable    ( EIB7_HANDLE         eib,
                                       EIB7_MODE           mode,
                                       unsigned long       src
                                     )
```

Parameters

`eib` EIB handle

`mode` Activate or deactivate the trigger sources

mode	Description
EIB7_MD_Enable	Activate the trigger source
EIB7_MD_Disable	Deactivate the trigger source

`src` Trigger source

src	Description
EIB7_TS_TrgInput1	Trigger input, channel 1
EIB7_TS_TrgInput2	Trigger input, channel 2
EIB7_TS_TrgInput3	Trigger input, channel 3
EIB7_TS_TrgInput4	Trigger input, channel 4
EIB7_TS_TrgRI1	Reference pulse of axis 1
EIB7_TS_TrgRI2	Reference pulse of axis 2
EIB7_TS_TrgRI3	Reference pulse of axis 3
EIB7_TS_TrgRI4	Reference pulse of axis 4
EIB7_TS_TrgRImaskedCH1	Linked reference pulse from axis 1 (A&B&RI)
EIB7_TS_TrgIC	Interval counter
EIB7_TS_TrgPuls	Trigger pulse counter
EIB7_TS_TrgTimer	Timer trigger
EIB7_TS_All	All trigger sources

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.5.34 Configuring the pulse counter

A trigger signal and a start signal can be selected for the pulse counter. The start signal enables the pulse counter. The counter will then be decremented with each pulse of the trigger signal, until a value of zero is reached. Then, all further trigger pulses will be disabled. If the function is re-run before the counter has reached zero, the counter will be reset to the initial value.

Function

```
EIB7_ERR EIB7ConfigPulsCounter   ( EIB7_HANDLE              eib,
                                   EIB7_PulsCounterStart    start,
                                   EIB7_PulsCounterTrigger  trigger,
                                   unsigned long            count
                                 )
```

Parameters

`eib` EIB handle

`start` Start signal for the pulse counter

`start`	Description
`EIB7_PS_TrgInput1`	Trigger input, channel 1
`EIB7_PS_TrgInput2`	Trigger input, channel 2
`EIB7_PS_TrgInput3`	Trigger input, channel 3
`EIB7_PS_TrgInput4`	Trigger input, channel 4
`EIB7_PS_TrgRI1`	Reference pulse of axis 1
`EIB7_PS_TrgRI2`	Reference pulse of axis 2
`EIB7_PS_TrgRI3`	Reference pulse of axis 3
`EIB7_PS_TrgRI4`	Reference pulse of axis 4
`EIB7_PS_TrgSW1`	Software trigger channel 1
`EIB7_PS_TrgSW2`	Software trigger channel 2
`EIB7_PS_TrgSW3`	Software trigger channel 3
`EIB7_PS_TrgSW4`	Software trigger channel 4
`EIB7_PS_TrgRImaskedCH1`	Linked reference pulse from axis 1 (A&B&RI)
`EIB7_PS_TrgIC`	Interval counter

`trigger` Trigger signal for the pulse counter

`trigger`	Description
`EIB7_PT_TrgInput1`	Trigger input, channel 1
`EIB7_PT_TrgInput2`	Trigger input, channel 2
`EIB7_PT_TrgInput3`	Trigger input, channel 3
`EIB7_PT_TrgInput4`	Trigger input, channel 4
`EIB7_PT_TrgRI1`	Reference pulse of axis 1
`EIB7_PT_TrgRI2`	Reference pulse of axis 2
`EIB7_PT_TrgRI3`	Reference pulse of axis 3
`EIB7_PT_TrgRI4`	Reference pulse of axis 4
`EIB7_PT_TrgRImaskedCH1`	Linked reference pulse from axis 1 (A&B&RI)
`EIB7_PT_TrgIC`	Interval counter
`EIB7_PT_TrgTimer`	Timer trigger

`count` Start value for pulse counter (`0x00000` … `0xFFFFF`)

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.5.35 Setting the interpolation factor for the interval counter

The interpolation factor for the interval counter is adjustable and determines the number of counting steps per signal period. This setting has an effect on both the interval counter and the auxiliary axis. The number of counting steps per signal period of the connected encoder is determined from the interpolation factor multiplied by the edge evaluation.

Function

```
EIB7_ERR EIB7SetIntervalCounterInterpolation ( EIB7_HANDLE              eib,
                                               EIB7_IntervalCounterIPF  ipf,
                                               EIB7_IntervalCounterEdge edge
                                             )
```

Parameters

`eib` EIB handle

`ipf` Interpolation factor

ipf	Description
EIB7_ICF_1x	1-fold interpolation
EIB7_ICF_2x	2-fold interpolation
EIB7_ICF_4x	4-fold interpolation
EIB7_ICF_5x	5-fold interpolation
EIB7_ICF_10x	10-fold interpolation
EIB7_ICF_20x	20-fold interpolation
EIB7_ICF_25x	25-fold interpolation
EIB7_ICF_50x	50-fold interpolation
EIB7_ICF_100x	100-fold interpolation

`edge` Edge evaluation

edge	Description
EIB7_ICE_1x	1-fold edge evaluation
EIB7_ICE_2x	2-fold edge evaluation
EIB7_ICE_4x	4-fold edge evaluation

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.5.36 Configuring the interval counter

This function is used to configure interval-counter triggering. The interval counter provides two modes. In the first mode, a trigger pulse is generated only at a fixed position. This position is adjustable. The second mode permits triggering at fixed intervals. The first trigger pulse is generated at the starting position. Then, a trigger pulse is generated at fixed intervals that can be adjusted using the `interval` parameter. As an alternative, the current position can also be used as starting position. Before the interval counter trigger can be reconfigured, the trigger function must be deactivated with the `EIB7_ICM_Disable` mode.

Function

```
EIB7_ERR EIB7SetIntervalCounterTrigger ( EIB7_HANDLE                eib,
                                         EIB7_IntervalCounterMode   mode,
                                         EIB7_IntervalCounterStart  start,
                                         unsigned long              startpos,
                                         unsigned long              interval
                                       )
```

Parameters

`eib` EIB handle

`mode` Trigger mode

`mode`	Description
`EIB7_ICM_Disable`	No triggering
`EIB7_ICM_Single`	Triggering only at a fixed position
`EIB7_ICM_Periodic`	Periodic triggering at fixed intervals

`start` Start of triggering

`start`	Description
`EIB7_ICS_Current`	Triggering starts at the current position
`EIB7_ICS_StartPos`	Triggering starts at the starting position

`startpos` Position value for the first trigger pulse

`interval` Interval between two trigger pulses in counting steps (6)

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.5.37 Setting the terminating resistors

The terminating resistors for the incremental signals of the encoder inputs can be deactivated. The setting will always apply to all 1 V_{pp} inputs of the EIB 74x. The resistors are activated after every boot process of the EIB 74x.

Function

```
EIB7_ERR EIB7EnableIncrementalTermination   ( EIB7_HANDLE   eib,
                                              EIB7_MODE     mode
                                            )
```

Parameters

`eib` EIB handle

`mode` Activate or deactivate the terminating resistors

`mode`	Description
`EIB7_MD_Disable`	Deactivate terminating resistors
`EIB7_MD_Enable`	Activate terminating resistors

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantChIncInpTrm` The mode cannot be changed

3.5.38 Reset

The EIB 74x performs a reset and reboots. This function has the same effect as pressing the reset button. The standard boot mode will be used (firmware of the last update with user's network settings). The connection to the EIB 74x is closed automatically, such as with `EIB7Close()`.

Function

```
EIB7_ERR EIB7Reset              ( EIB7_HANDLE         eib
                                )
```

Parameters

`eib` EIB handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.39 Identifying the EIB 74x

The LAN LED on the front panel of the EIB 74x can be set to flash mode. If several devices are arranged side by side, an EIB 74x with a certain IP address is easily located. The LED flashes until the mode is terminated using the function.

Function

```
EIB7_ERR EIB7Identify          ( EIB7_HANDLE         eib,
                                 EIB7_MODE           mode
                               )
```

Parameters

`eib` EIB handle

`mode` Activate or deactivate LED flashing

`mode`	Description
`EIB7_MD_Disable`	Deactivate flash mode
`EIB7_MD_Enable`	Activate flash mode

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_IllegalParameter` The LED status cannot be changed (parameter is invalid)

3.5.40 Recording – transmitting the data

The transmission of data from the EIB 74x's internal recording buffer can be activated or deactivated. When the data transmission is activated, it is possible to select only a certain range of the recorded data for transmission. The first byte to be transmitted is given through the offset and the length parameter specifies the number of bytes.

To transmit all data, the parameters should be set as follows: `offset` = 0 and `length` = `0xFFFFFFFF`.

Function

```
EIB7_ERR EIB7TransferRecordingData      ( EIB7_HANDLE      eib,
                                          EIB7_MODE        mode,
                                          unsigned long    offset,
                                          unsigned long    length
                                        )
```

Parameters

`eib` EIB handle

`mode` Activate or deactivate the terminating resistors

`mode`	Description
`EIB7_MD_Disable`	Stop data transmission
`EIB7_MD_Enable`	Start data transmission

`offset` Offset for the first byte to be transmitted

`length` Number of bytes that are to be transmitted (`0xFFFFFFFF` = until the end of recording)

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantOpenSocket`	Internal error (socket error)
`EIB7_CantStartThread`	Internal error (thread error)
`EIB7_OpModeBlocked`	The EIB 74x is not in Polling operating mode
`EIB7_RecDataReadErr`	The data cannot be read from the EIB 74x
`EIB7_ParamInvalid`	Invalid parameter

3.5.41 Recording – verifying the status

The status in the Recording operating mode can be read out. In addition, the current contents in the buffer memory can be determined. This is also possible during recording in the Recording operating mode. The progress of data transmission from the EIB 74x's buffer to the host can also be read out.

Function

```
EIB7_ERR EIB7GetRecordingStatus      ( EIB7_HANDLE          eib,
                                       unsigned long*       length,
                                       unsigned long*       status,
                                       unsigned long*       progress
                                     )
```

Parameters

`eib` EIB handle

`length` *[return value]* Pointer to the target variable for the number of data packets in the buffer

`Status` *[return value]* Pointer to the target variable for the status

`Status`	Description
`0`	Recording operating mode deactivated
`1`	Recording operating mode activated
`2`	Data is being transmitted
`3`	Waiting for connection setup for data transmission

`progress` *[return value]* Pointer to the target variable for the data transmission progress, progress as a percentage (0 ... 100)

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.42 Recording – reading the memory size

The size of the memory for the recorded data in the EIB 74x can be read out. The size is provided as the number of data packets that can be accommodated by the memory. This number depends on the size of the data packets. The data packet must therefore be configured first.

Function

```
EIB7_ERR EIB7GetRecordingMemSize      ( EIB7_HANDLE          eib,
                                        unsigned long*       size
                                      )
```

Parameters

`eib`	EIB handle
`size`	*[return value]* Pointer to the target variable for the size of the memory (as a number of data packets)

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvalidPacket`	Current configuration for the data packet is invalid

3.5.43 Checking the streaming status

While the Streaming operating mode is active, the status of the buffer in the EIB 74x can be read out. In addition to the size of the buffer in bytes, the amount of currently saved data is indicated in bytes. Also, the maximum amount of data in the buffer since activation of the Streaming operating mode is indicated in bytes.

Function

```
EIB7_ERR EIB7GetStreamingStatus      ( EIB7_HANDLE         eib,
                                       unsigned long*      length,
                                       unsigned long*      max,
                                       unsigned long*      size
                                     )
```

Parameters

`eib`	EIB handle
`length`	*[return value]* Pointer to the target variable for the number of bytes in the buffer
`max`	max *[return value]* Pointer to the target variable for the maximum number of bytes in the buffer since the start of the Streaming operating mode
`size`	*[return value]* Pointer to the target variable for the size of the buffer in bytes

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.44 Reading data from the FIFO

Data packets (in raw data format) are copied from the FIFO to the target memory. The `cnt` parameter indicates the number of entries to be copied from the FIFO. If the FIFO contains fewer data records, the entire contents of the FIFO will be copied. The number of entries actually copied is returned via the `entries` parameter. The function waits until at least one data record has been copied from the FIFO, though no longer than expiration of the timeout. In this case, `entries` returns zero. Entire data packets are always copied from the FIFO. The target memory must be at least big enough to accommodate the indicated number of FIFO entries. The contents of an entry in the FIFO corresponds to the currently configured data packet without the appended "fill bytes." All data words are saved in the "little endian" format.

Function

```
EIB7_ERR EIB7ReadFIFODataRaw         ( EIB7_HANDLE         eib,
                                       void*               data,
                                       unsigned long       cnt,
                                       unsigned long*      entries,
                                       long                timeout
                                     )
```

Parameters

`eib`	EIB handle
`data`	*[return value]* Pointer to the target memory
`cnt`	Number of entries to be read (≥ 0)
`entries`	*[return value]* Number of entries to be copied
`timeout`	Timeout in milliseconds

`timeout`	Description
`0`	Function returns immediately if no data is present
`>0`	Function waits for data for x milliseconds
`-1`	Function waits infinitely

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_FIFOEmpty`	No data in the FIFO
`EIB7_ElementSizeInv`	Internal error
`EIB7_FIFOOverflow`	FIFO overflow since the last function call (loss of data)

3.5.45 Reading the size of a FIFO element

The size of a FIFO element in raw data format is generated. This value corresponds to the size of a FIFO entry as read with the `EIB7ReadFIFODataRaw()` function. A FIFO element contains a data packet whose size depends on the present configuration. The size is given without "fill bytes."

Function

```
EIB7_ERR EIB7SizeOfFIFOEntryRaw      ( EIB7_HANDLE         eib,
                                       unsigned long*      size
                                     )
```

Parameters

`eib`	EIB handle
`size`	*[return value]* Pointer to the variable for the size of a FIFO element in bytes

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.46 Access to the contents of a FIFO element

With this function, you can access individual fields of a FIFO element (in raw data format). A FIFO entry can contain, for example, the trigger counter, position data, the status word and additional data. The contents of the data packet can vary depending on its configuration. When the elements are accessed, this must be remembered in order to interpret the data correctly. This function provides a pointer to the relevant field within the data structure and also the field size in bytes. A general selection is made using the `region` parameter. This makes it possible to select the axis from which the field is obtained. The fine selection can be made using the `type` parameter. This indicates which data field of an axis is to be accessed.

Function

```
EIB7_ERR EIB7GetDataFieldPtrRaw    ( EIB7_HANDLE                 eib,
                                     void*                       data,
                                     EIB7_DataRegion             region,
                                     EIB7_PositionDataField      type,
                                     void**                      field,
                                     unsigned long*              size
                                   )
```

Parameters

`eib` EIB handle

`data` Pointer to the data structure (FIFO element)

`region` Axis of the EIB 74x

`region`	Description
`EIB7_DR_Global`	Global data field for trigger counter
`EIB7_DR_Encoder1`	Data for axis 1
`EIB7_DR_Encoder2`	Data for axis 2
`EIB7_DR_Encoder3`	Data for axis 3
`EIB7_DR_Encoder4`	Data for axis 4
`EIB7_DR_AUX`	Data for auxiliary axis

`type` Data element for an axis

`type`	Description
`EIB7_PDF_TriggerCounter`	Trigger counter (only in `EIB7_DR_Global`)
`EIB7_PDF_StatusWord`	Status word for position
`EIB7_PDF_PositionData`	Position value
`EIB7_PDF_AUXPosition`	Position value for auxiliary axis
`EIB7_PDF_Timestamp`	Timestamp for position
`EIB7_PDF_Analog`	ADC value for signals A and B
`EIB7_PDF_ReferencePos`	Reference positions 1 and 2
`EIB7_PDF_DistCodedRef`	Coded reference value
`EIB7_PDF_EnDat_AI1`	EnDat 2.2 additional datum 1
`EIB7_PDF_EnDat_AI2`	EnDat 2.2 additional datum 2

`field` *[return value]* Pointer to the memory address of the element from the data structure

`size` *[return value]* Size of the element in bytes

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_FieldNotAvail` The indicated field was not found

3.5.47 Reading and converting data from the FIFO

Data packets are copied from the FIFO to the target memory and converted. The `cnt` parameter indicates the number of entries to be copied from the FIFO. If the FIFO contains fewer data records, the entire contents of the FIFO is copied. The number of entries actually copied is returned via the `entries` parameter. The function waits until at least one data record has been copied from the FIFO, though no longer than expiration of the timeout. In this case, `entries` returns zero. Entire data packets are always copied from the FIFO. The target memory must be at least big enough to accommodate the indicated number of FIFO entries. The contents of an entry correspond to the currently configured data packet without the appended "fill bytes." All data words are saved in the standard format for 16-bit or 32-bit integers and the position values are converted to the ENCODER_POSITION format.

Function

```
EIB7_ERR EIB7ReadFIFOData          ( EIB7_HANDLE         eib,
                                     void*               data,
                                     unsigned long       cnt,
                                     unsigned long*      entries,
                                     long                timeout
                                   )
```

Parameters

`eib`	EIB handle
`data`	*[return value]* Pointer to the target memory
`cnt`	Number of entries to be read (≥ 0)
`entries`	*[return value]* Number of entries to be copied
`timeout`	Timeout in milliseconds

`timeout`	Description
`0`	Function returns immediately if no data is present
`>0`	Function waits for data for x milliseconds
`-1`	Function waits infinitely

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_FIFOEmpty`	No data in the FIFO
`EIB7_ElementSizeInv`	Internal error
`EIB7_FIFOOverflow`	FIFO overflow since the last function call (loss of data)

3.5.48 Reading the size of a FIFO element after conversion

This function returns the size of a FIFO element after conversion. This value corresponds to the size of a FIFO entry as read out with the `EIB7ReadFIFOData()` function. A FIFO element contains a data packet whose size depends on the present configuration. The size is given without "fill bytes."

Function

```
EIB7_ERR EIB7SizeOfFIFOEntry          ( EIB7_HANDLE          eib,
                                        unsigned long*       size
                                      )
```

Parameters

`eib`	EIB handle
`size`	*[return value]* Pointer to the variable for the size of a FIFO element in bytes

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.49 Access to the contents of a FIFO element with converted data

With this function, you can access individual fields of a FIFO element that contains converted position data (in the ENCODER_POSITION format). A FIFO entry can contain, for example, the trigger counter, position data, the status word and additional data. The contents of the data packet can vary depending on its configuration. When the elements are accessed, this must be remembered in order to interpret the data correctly. This function provides a pointer to the relevant field within the data structure and also the field size in bytes. A general selection is made using the `region` parameter. This makes it possible to select the axis from which the field is obtained. The fine selection can be made using the `type` parameter. This indicates which data field of an axis is to be accessed.

Function

```
EIB7_ERR EIB7GetDataFieldPtr        ( EIB7_HANDLE                  eib,
                                      void*                        data,
                                      EIB7_DataRegion              region,
                                      EIB7_PositionDataField       type,
                                      void**                       field
                                      unsigned long*               size
                                    )
```

Parameters

`eib`	EIB handle
`data`	Pointer to the data structure (FIFO element)

`region` Axis of the EIB 74x

`region`	Description
`EIB7_DR_Global`	Global data field for trigger counter
`EIB7_DR_Encoder1`	Data for axis 1
`EIB7_DR_Encoder2`	Data for axis 2
`EIB7_DR_Encoder3`	Data for axis 3
`EIB7_DR_Encoder4`	Data for axis 4
`EIB7_DR_AUX`	Data for auxiliary axis

`type` Data element for an axis

`type`	Description
`EIB7_PDF_TriggerCounter`	Trigger counter (only in `EIB7_DR_Global`)
`EIB7_PDF_StatusWord`	Status word for position
`EIB7_PDF_PositionData`	Position value
`EIB7_PDF_AUXPosition`	Position value for auxiliary axis
`EIB7_PDF_Timestamp`	Timestamp for position
`EIB7_PDF_Analog`	ADC value for signals A and B
`EIB7_PDF_ReferencePos`	Reference positions 1 and 2
`EIB7_PDF_DistCodedRef`	Coded reference value
`EIB7_PDF_EnDat_AI1`	EnDat 2.2 additional datum 1
`EIB7_PDF_EnDat_AI2`	EnDat 2.2 additional datum 2

`field` *[return value]* Pointer to the memory address of the element from the data structure

`size` *[return value]* Size of the element in bytes

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_FieldNotAvail` The indicated field was not found

3.5.50 Reading the number of elements in the FIFO

The number of elements currently stored in the FIFO is generated.

Function

```
EIB7_ERR EIB7FIFOEntryCount     ( EIB7_HANDLE         eib,
                                  unsigned long*      cnt
                                )
```

Parameters

`eib`	EIB handle
`cnt`	[return value] Pointer to the variable for the number of FIFO elements

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.51 Clearing the FIFO

The contents of the FIFO are cleared. This command has no effect if Polling operating mode is active.

Function

```
EIB7_ERR EIB7ClearFIFO          ( EIB7_HANDLE         eib
                                )
```

Parameters

`eib`	EIB handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.52 Setting the FIFO size

The size of the FIFO is redefined. All data in the FIFO is cleared. The size can only be set in Polling operating mode. The size of the FIFO must be at least 2000 bytes. If the value is smaller, the value of 2000 bytes is used internally.

Function

```
EIB7_ERR EIB7SetFIFOSize        ( EIB7_HANDLE         eib,
                                  unsigned long       size
                                )
```

Parameters

`eib`	EIB handle
`size`	FIFO size in bytes

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_SoftRTEn`	Soft Real-Time operating mode is activated

3.5.53 Reading the FIFO size

The size of the FIFO in bytes is generated.

Function

```
EIB7_ERR EIB7GetFIFOSize       ( EIB7_HANDLE         eib,
                                 unsigned long*      size
                               )
```

Parameters

`eib`	EIB handle
`size`	*[return value]* Pointer to the variable for the FIFO size in bytes

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.54 Activating the callback mechanism

The callback mechanism is activated or deactivated and the function pointer is saved, where applicable. The callback function is called if there are at least as many elements saved in the FIFO as indicated in the `threshold`. Then, this function will not be called until new data has been written to the FIFO and at least `threshold` elements have been saved in the FIFO.

Function

```
EIB7_ERR EIB7SetDataCallback       ( EIB7_HANDLE              eib,
                                     void*                    data,
                                     EIB7_MODE                activate,
                                     unsigned long            threshold,
                                     EIB7OnDataAvailable      handler
                                   )
```

Parameters

`eib`	EIB handle
`data`	Pointer to user data, this pointer is passed as a parameter to the callback function
`Activate`	Activate or deactivate callback

`Activate`	Description
`EIB7_MD_Disable`	Deactivate the callback mechanism
`EIB7_MD_Enable`	Activate the callback mechanism

`threshold`	Number of elements in the FIFO from which on the callback mechanism will be triggered (> 0)
`handler`	Pointer to the callback function (NULL is permitted if `activate` = 0)

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

Callback function

The callback function is performed by the driver and runs in a separate thread. The user must take care of any necessary synchronization with the main program. The `eib` parameter contains the handle to the EIB 74x that has triggered the callback. `cnt` indicates the number of elements currently stored in the FIFO. The `data` parameter contains the pointer that was specified when registering the callback function.

Prototype

```
typedef void (*EIB7OnDataAvailable)    ( EIB7_HANDLE          eib,
                                         unsigned long        cnt,
                                         void*                data
                                       )
```

Parameters

`eib`	EIB handle
`cnt`	Number of elements in the FIFO
`data`	Pointer to user data

3.5.55 Selecting the trigger source for the auxiliary axis

The trigger signal for the auxiliary axis can be selected from different sources. This setting is only possible in Polling mode of operation.

Function

```
EIB7_ERR EIB7AuxAxisTriggerSource ( EIB7_HANDLE           eib,
                                    EIB7_AxisTriggerSrc   src
                                  )
```

Parameters

`eib` EIB handle

`src` Trigger source

src	Description
EIB7_AT_TrgInput1	Trigger input, channel 1
EIB7_AT_TrgInput2	Trigger input, channel 2
EIB7_AT_TrgInput3	Trigger input, channel 3
EIB7_AT_TrgInput4	Trigger input, channel 4
EIB7_AT_TrgSW1	Software trigger channel 1
EIB7_AT_TrgSW2	Software trigger channel 2
EIB7_AT_TrgSW3	Software trigger channel 3
EIB7_AT_TrgSW4	Software trigger channel 4
EIB7_AT_TrgRI	Reference pulse of the corresponding axis
EIB7_AT_TrgRImaskedCH1	Linked reference pulse from axis 1 (A&B&RI)
EIB7_AT_TrgIC	Interval counter
EIB7_AT_TrgPuls	Trigger pulse counter
EIB7_AT_TrgTimer	Timer trigger

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.5.56 Reading the position of the auxiliary axis

The current position value is read out. A status word informing about potential position errors is also transmitted. The position can only be polled in Polling operating mode. The setting of the interpolation factor and the edge evaluation for the interval counter determine the significance of an LSB (Lowest Significant Bit) in the position value with respect to the signal period of the encoder.

Function

```
EIB7_ERR EIB7AuxGetPosition    ( EIB7_HANDLE           eib,
                                 unsigned short*       status,
                                 ENCODER_POSITION*     pos
                               )
```

Parameters

`eib`	EIB handle
`Status`	*[return value]* Pointer to the target variable for the status word
`pos`	*[return value]* Pointer to the target variable for the position

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantLatchPos`	Position cannot be determined
`EIB7_EncPwrSuppErr`	Encoder power supply error
`EIB7_NotInitialized`	Auxiliary axis not configured

3.5.57 Reading out the data of the auxiliary axis

The current position and certain additional parameters are determined and read out. The function may only be performed in Polling mode of operation. The status word indicates whether the position and the reference position are valid. The position value and the timestamp are saved simultaneously. This requires the internal use of software trigger channel 1. The trigger source for the auxiliary axis is configured accordingly, thereby overwriting the current setting.

Function

```
EIB7_ERR EIB7AuxGetEncoderData ( EIB7_HANDLE              eib,
                                 unsigned short*          status,
                                 ENCODER_POSITION*        pos,
                                 ENCODER_POSITION*        ref,
                                 unsigned long*           timestamp,
                                 unsigned short*          counter
                               )
```

Parameters

`eib`	EIB handle
`Status`	*[return value]* Pointer to the target variable for the status word
`pos`	*[return value]* Pointer to the target variable for the position
`Ref`	*[return value]* Pointer to the target variable for the reference position
`timestamp`	*[return value]* Pointer to the target variable for the timestamp value
`counter`	*[return value]* Pointer to the target variable for the trigger counter

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantLatchPos`	Position cannot be determined
`EIB7_EncPwrSuppErr`	Encoder power supply error
`EIB7_NotInitialized`	Auxiliary axis not configured

3.5.58 Clearing the counter of the auxiliary axis

The position counter of the auxiliary axis is cleared.

Function

```
EIB7_ERR EIB7AuxClearCounter    ( EIB7_HANDLE         eib
                                  )
```

Parameters

`eib`	EIB handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.59 Acknowledging signal errors of the auxiliary axis

The error messages for the auxiliary axis are cleared. This applies to the error for the signal amplitude and the error for exceeding the frequency.

Function

```
EIB7_ERR EIB7AuxClearSignalErrors     ( EIB7_HANDLE   eib
                                        )
```

Parameters

`eib`	EIB handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.60 Acknowledging the trigger errors of the auxiliary axis

The error messages for unrecognized trigger events are cleared from the auxiliary axis trigger logic.

Function

```
EIB7_ERR EIB7AuxClearLostTriggerError ( EIB7_HANDLE   eib
                                        )
```

Parameters

`eib`	EIB handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.61 Clearing the status bit for the reference mark of the auxiliary axis

The "Reference position saved" flag in the status word for the auxiliary axis is reset.

Function

```
EIB7_ERR EIB7AuxClearRefStatus        ( EIB7_HANDLE   eib
                                      )
```

Parameters

`eib`	EIB handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.62 Checking the status of the reference run for the auxiliary axis

The status of the reference run is output. It indicates whether the reference run is still active or whether the reference positions have already been saved.

Function

```
EIB7_ERR EIB7AuxGetRefActive      ( EIB7_HANDLE      eib,
                                    EIB7_MODE*       active
                                  )
```

Parameters

`eib`	EIB handle
`active`	*[return value]* Pointer to the variable for the status of the reference run

`active`	Description
`EIB7_MD_Disable`	Reference run inactive
`EIB7_MD_Enable`	Reference run active

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.63 Starting a reference run for the auxiliary axis

After this command has been called, the reference position is saved when the next reference mark is traversed. The saved value corresponds to the count of the period counter at the respective reference mark. As soon as a reference run has been started, the status of a previously saved reference position is set to "invalid."

Function

```
EIB7_ERR EIB7AuxStartRef        ( EIB7_HANDLE         eib
                                  )
```

Parameters

`eib`	EIB handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.64 Stopping a reference run for the auxiliary axis

A reference run (mode for automatically saving the reference position) is stopped. Reference positions that have already been saved will not be deleted.

Function

```
EIB7_ERR EIB7AuxStopRef         ( EIB7_HANDLE         eib
                                  )
```

Parameters

`eib`	EIB handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.65 Configuring a timestamp for the auxiliary axis

The timestamp can be activated or deactivated for the auxiliary axis. The global setting of the EIB 74x is used for the period duration. The timestamp value for a position poll for the auxiliary axis will be copied if this function has been activated previously.

Function

```
EIB7_ERR EIB7AuxSetTimestamp      ( EIB7_HANDLE      eib,
                                    EIB7_MODE        mode
                                  )
```

Parameters

`eib` EIB handle

`mode` Activate or deactivate timestamp

`mode`	Description
`EIB7_MD_Disable`	Deactivate timestamp
`EIB7_MD_Enable`	Activate timestamp

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.5.66 Setting the trigger edge for the reference pulse of the auxiliary axis

The time for reference-pulse triggering for the auxiliary axis can be set to the rising or falling edge or to both edges of the reference-pulse signal. If both edges are set as trigger events, make sure that the maximum trigger rate is not exceeded.

Function

```
EIB7_ERR EIB7AuxSetRITriggerEdge   ( EIB7_HANDLE            eib,
                                     EIB7_RITriggerEdge     edge
                                   )
```

Parameters

`eib` EIB handle

`edge` Active trigger edge for the reference pulse

`edge`	Description
`EIB7_RI_Rising`	Rising edge of the reference pulse
`EIB7_RI_Falling`	Falling edge of the reference pulse
`EIB7_RI_Both`	Both edges of the reference pulse

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.6 Axis functions

The axis functions always refer to just one axis on the EIB 74x. The other axes are not affected.

Besides the standard return values ("Parameters and return values"), all axis functions may also return further values. These are indicated separately for each function.

3.6.1 Initializing an axis

An axis of the EIB 74x is configured for the connected encoder. The axis number of the EIB 74x is determined via the axis handle. The interface type of the encoder must be selected as a basic option. Certain parameters are required only for incremental interfaces and others only for EnDat interfaces. For an EnDat 2.2 interface, the delay compensation can also be activated using the `iface` parameter. To do this, the `EIB7_IT_EnDat22` and `EIB7_IT_EnDatDelayMeasurement` constants must be gated with OR.

The `EnDatclock`, `recovery`, and `calculation` parameters are only used for encoders with the EnDat interface. The EnDat interface clock pulse can be set. This must be done using pre-defined constants. When an axis is initialized for EnDat operation, an EnDat reset command is sent to the connected encoder. In addition, you can set the recovery time I using the `recovery` parameter for EnDat 2.2 encoders (ordering designations EnDat02 or EnDat22). The "EnDat calculation time" needs to be set using the `calculation` parameter for the encoder.

The `bandwidth` and `comp` parameters are effective only for incremental encoders. Two states (high and low) are available for configuring the bandwidth. The online compensation function can be activated and deactivated. The values for `linecounts` and `increment` are required only in conjunction with distance-coded reference marks.

Calling this function clears the following flags:

- Signal amplitude error
- Frequency exceeded
- Reference position 1 saved
- Reference position 2 saved
- Coded reference value for distance-coded reference marks valid
- Error during the calculation of the coded reference value for distance-coded reference marks
- CRC error

The settings for the terminating resistors for the incremental signals and the value of the period counter will not be affected by this function.

Function

```
EIB7_ERR EIB7InitAxis    ( EIB7_AXIS               axis,
                           unsigned long           iface,
                           EIB7_EncoderType        type,
                           EIB7_RefMarks           refmarks,
                           unsigned long           linecounts,
                           unsigned long           increment,
                           EIB7_Homing             homing,
                           EIB7_Limit              limit,
                           EIB7_Compensation       comp,
                           EIB7_Bandwidth          bandwidth,
                           unsigned long           EnDatclock,
                           EIB7_EnDatRecoveryTime  recovery,
                           EIB7_EnDatCalcTime      calculation
                         )
```

Parameters

`axis` AXIS handle

`iface` Interface type of the encoder

`iface`	Description
`EIB7_IT_Disabled`	Axis deactivated
`EIB7_IT_Incremental`	Encoder with incremental signals (1 V_{pp})
`EIB7_IT_Incremental_11u`	Encoder with incremental signals ((11 µA)
`EIB7_IT_EnDat21`	Encoder with EnDat 2.1 interface
`EIB7_IT_EnDat01`	Encoder with EnDat 2.1 interface and incremental signals (1 V_{pp})
`EIB7_IT_EnDat22`	Encoder with EnDat 2.2 interface
`EIB7_IT_EnDatDelayMeasurement`	Delay compensation for EnDat 2.2

`type` Type of encoder

`type`	Description
`EIB7_EC_Linear`	Linear encoder
`EIB7_EC_Rotary`	Angle encoder / rotational encoder

`Refmarks` Type of reference marks

`Refmarks`	Description
`EIB7_RM_None`	No reference mark
`EIB7_RM_One`	One reference mark (EIB 74x will not calculate a coded reference value)
`EIB7_RM_DistanceCoded`	Distance-coded reference marks (EIB 74x will calculate coded reference value automatically)

`linecounts` Number of signal periods per revolution (only for rotational encoders)

`increment` Nominal increment in signal periods between two fixed reference marks (only for distance-coded reference marks)

`homing` Activate or deactivate homing signal evaluation

`homing`	Description
`EIB7_HS_None`	Homing signal is not evaluated
`EIB7_HS_Available`	Homing signal is evaluated

`limit` Activate or deactivate limit signal evaluation

`limit`	Description
`EIB7_LS_None`	Limit signal is not evaluated
`EIB7_LS_Available`	Limit signal is evaluated

`Compensation` Activate or deactivate online compensation

`Compensation`	Description
`EIB7_CS_None`	Signal compensation deactivated
`EIB7_CS_CompActive`	Signal compensation activated

`Bandwidth` Input bandwidth for incremental signals (high/low)

`Bandwidth`	Description
`EIB7_BW_High`	High input bandwidth for 1 V_{pp} signals
`EIB7_BW_Low`	Low input bandwidth for 1 V_{pp} signals

`EnDatclock` EnDat clock rate

`recovery` Recovery time I for EnDat 2.2

`recovery`	Description
`EIB7_RT_Long`	Long recovery time I according to the EnDat specification
`EIB7_RT_Short`	Short recovery time I according to the EnDat specification

`calculation` Calculation time for EnDat

`calculation`	Description
`EIB7_CT_Long`	Long calculation time
`EIB7_CT_Short`	Short calculation time

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid`	Invalid parameter
`EIB7_InvInterface`	Interface type invalid
`EIB7_InvRefMarkOpt`	Reference mark invalid
`EIB7_InvDistCodeRef`	Parameter for distance-coded reference marks invalid (`linecount`, `increment`)
`EIB7_ConfOptIncons`	Invalid combination of parameters
`EIB7_AccNotAllowed`	Access denied
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)

When the axis for EnDat (EnDat01, EnDat21, EnDat22) is defined, the execution of `EIB7InitAxis` automatically cleared the EnDat error and warning messages (up to firmware version 9). As of firmware version 10, the error and warning messages are no longer cleared. This is due to the error handling for battery-buffered encoders (see also EnDat Application Notes).

3.6.2 Selecting the trigger source for an axis

The trigger signal for the axis can be selected from different sources. This setting is only possible in Polling mode of operation.

Function

```
EIB7_ERR EIB7AxisTriggerSource    ( EIB7_AXIS              axis,
                                    EIB7_AxisTriggerSrc    src
                                  )
```

Parameters

`axis` AXIS handle

`src` Trigger source

src	Description
EIB7_AT_TrgInput1	Trigger input, channel 1
EIB7_AT_TrgInput2	Trigger input, channel 2
EIB7_AT_TrgInput3	Trigger input, channel 3
EIB7_AT_TrgInput4	Trigger input, channel 4
EIB7_AT_TrgSW1	Software trigger channel 1
EIB7_AT_TrgSW2	Software trigger channel 2
EIB7_AT_TrgSW3	Software trigger channel 3
EIB7_AT_TrgSW4	Software trigger channel 4
EIB7_AT_TrgRI	Reference pulse of the corresponding axis
EIB7_AT_TrgRImaskedCH1	Linked reference pulse from axis 1 (A&B&RI)
EIB7_AT_TrgIC	Interval counter
EIB7_AT_TrgPuls	Trigger pulse counter
EIB7_AT_TrgTimer	Timer trigger

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.6.3 Setting the trigger edge for the reference pulse

The time for reference-pulse triggering for the auxiliary axis can be set to the rising or falling edge or to both edges of the reference-pulse signal. If both edges are set as trigger events, make sure that the maximum trigger rate is not exceeded.

Function

```
EIB7_ERR EIB7SetRITriggerEdge     ( EIB7_AXIS              axis,
                                    EIB7_RITriggerEdge     edge
                                  )
```

Parameters

`axis` AXIS handle

`edge` Active trigger edge for the reference pulse

edge	Description
EIB7_RI_Rising	Rising edge of the reference pulse
EIB7_RI_Falling	Falling edge of the reference pulse
EIB7_RI_Both	Both edges of the reference pulse

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.6.4 Clearing the counter

The period counter of an axis is cleared. This function is permitted only if the axis is configured for incremental encoders. Otherwise an error message is generated. Only the period counter will be reset. The interpolation value will not be changed.

Function

```
EIB7_ERR EIB7ClearCounter       ( EIB7_AXIS           axis
                                  )
```

Parameters

`axis` AXIS handle

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface` Interface type invalid

3.6.5 Polling a position

The encoder's current position value is determined and read out. A status word informing about potential position errors is also transmitted. The position can only be polled in Polling operating mode. Depending on whether the axis is configured for incremental or for EnDat encoders, the interpolated value of the incremental signals is returned or an EnDat poll is sent to the encoder. For an EnDat 01 configuration, the interpolated value of the incremental signals is read.

Function

```
EIB7_ERR EIB7GetPosition       ( EIB7_AXIS              axis,
                                 unsigned short*        status,
                                 ENCODER_POSITION*      pos
                                )
```

Parameters

`axis`	AXIS handle
`Status`	*[return value]* Pointer to the target variable for the status word
`pos`	*[return value]* Pointer to the target variable for the position

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantLatchPos`	Position cannot be determined
`EIB7_EncPwrSuppErr`	Encoder power supply error (EnDat encoders only)
`EIB7_NotInitialized`	Axis not configured

3.6.6 Reading out data for a channel

The current position and certain additional parameters are determined and read out. The `refc` parameter is valid only if the axis is configured for encoders with distance-coded reference marks. The function may only be performed in Polling mode of operation. The axis must have been configured for incremental encoders. The position value and the timestamp are saved simultaneously. This requires the internal use of software trigger channel 1. The trigger source for the auxiliary axis is configured accordingly, thereby overwriting the current setting.

Function

```
EIB7_ERR EIB7GetEncoderData    ( EIB7_AXIS                axis,
                                 unsigned short*          status,
                                 ENCODER_POSITION*        pos,
                                 ENCODER_POSITION*        ref1,
                                 ENCODER_POSITION*        ref2,
                                 ENCODER_POSITION*        refc,
                                 unsigned long*           timestamp,
                                 unsigned short*          counter,
                                 unsigned short*          adc00,
                                 unsigned short*          adc90
                                )
```

Parameters

`axis`	AXIS handle
`Status`	*[return value]* Pointer to the variable for the status word
`pos`	*[return value]* Pointer to the variable for the position
`ref1`	*[return value]* Pointer to the variable for reference position 1
`ref2`	*[return value]* Pointer to the variable for reference position 2
`refc`	*[return value]* Pointer to the variable for the coded reference value
`timestamp`	*[return value]* Pointer to the variable for the timestamp value
`counter`	*[return value]* Pointer to the variable for the trigger counter
`adc00`	*[return value]* Pointer to the variable for the ADC value for signal A
`adc90`	*[return value]* Pointer to the variable for the ADC value for signal B

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid`	Invalid parameter (maybe the axis is not configured for incremental encoders)
`EIB7_EncPwrSuppErr`	Encoder power supply error (EnDat encoders only)
`EIB7_NotInitialized`	Axis not configured

3.6.7 Acknowledging power supply errors

The error message for the power supply to the encoder is acknowledged. If no error has occurred for this axis, the function is terminated with an error message. The encoder power supply is switched back on once the error has been acknowledged.

Function

```
EIB7_ERR EIB7ClearPowerSupplyError    ( EIB7_AXIS        axis
                                      )
```

Parameters

`axis`	AXIS handle

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_CantClearEnc`	The error cannot be cleared

3.6.8 Acknowledging a trigger error

The error message for the trigger interface is acknowledged. The trigger error is cleared for all axes of an EIB 74x at the same time, regardless of which AXIS handle is passed.

Function

```
EIB7_ERR EIB7ClearLostTriggerError    ( EIB7_AXIS        axis
                                      )
```

Parameters

`axis`	AXIS handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.6.9 Acknowledging signal errors

The error messages for the signals of the encoder are cleared. For incremental encoders, the error for the signal amplitude and the error for exceeding the frequency are acknowledged. For EnDat encoders, the CRC error for data transmissions and the EnDat error messages are cleared. This does not affect the error memory in the encoder. No EnDat command is sent.

Function

```
EIB7_ERR EIB7ClearEncoderErrors       ( EIB7_AXIS        axis
                                      )
```

Parameters

`axis`	AXIS handle

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.6.10 Clearing EnDat error bits

The EnDat error flags are cleared. This function is permitted only for axes configured for EnDat encoders. An EnDat reset command is sent to the EnDat encoder to clear the error memory. After the reset command, the function waits for 50 ms in accordance with the EnDat specification. The function may only be performed in Polling mode of operation.

Function

```
EIB7_ERR EIB7ClearEnDatErrorMsg       ( EIB7_AXIS        axis
                                      )
```

Parameters

`axis`	AXIS handle

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	Axis not configured for EnDat
`EIB7_EncPwrSuppErr`	Encoder power supply error
`EIB7_NotInitialized`	Axis not configured

3.6.11 Clearing status bits for reference marks

The flags for the reference position in the status word are reset. The following flags are reset:

- "Reference position 1 saved"
- "Reference position 2 saved"

This command is permitted only for axes configured for incremental encoders with reference marks.

Function

```
EIB7_ERR EIB7ClearRefLatched          ( EIB7_AXIS        axis
                                      )
```

Parameters

`axis`	AXIS handle

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for incremental encoders
`EIB7_NotInitialized`	Axis not configured

3.6.12 Clearing status bits for distance-coded reference marks

The flags for the reference position in the status word are reset. The following flags are reset:

- "Reference position 1 saved"
- "Reference position 2 saved"
- "Coded reference value is valid"
- "Error during the calculation of the coded reference value"

This command is permitted only for axes configured for encoders with distance-coded reference marks.

Function

```
EIB7_ERR EIB7ClearRefStatus          ( EIB7_AXIS        axis
                                     )
```

Parameters

`axis`	AXIS handle

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for incremental encoders
`EIB7_NotInitialized`	Axis not configured

3.6.13 Starting the reference run

After this command has been called, the reference position is saved when the next reference mark is traversed. The `ref` parameter can be used to define whether only one or two reference positions will be saved. If two reference positions are activated, one position value is saved with each of the two subsequent reference pulses. The saved values correspond to the count of the period counter at the respective reference mark. This command is permitted only for axes configured for incremental encoders.

If this function is called again before all reference positions from the first call have been saved, the old reference position values become invalid; this is indicated by the flags in the status word. The reference run is restarted.

Function

```
EIB7_ERR EIB7StartRef    ( EIB7_AXIS                axis,
                           EIB7_ReferencePosition    ref
                         )
```

Parameters

`axis` AXIS handle

`Ref` Option for the reference position to be saved

`Ref`	Description
`EIB7_RP_RefPos1`	Save one reference position
`EIB7_RP_RefPos2`	Save two reference positions

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for incremental encoders
`EIB7_NotInitialized`	Axis not configured
`EIB7_ParamInvalid`	Parameter is not a valid option for the reference marks

3.6.14 Stopping the reference run

A reference run (mode for automatically saving the reference position) is stopped. If reference marks have already been traversed, the corresponding position values will be retained. This command is permitted only for axes configured for incremental encoders.

Function

```
EIB7_ERR EIB7StopRef          ( EIB7_AXIS          axis
                              )
```

Parameters

`axis`	AXIS handle

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for incremental encoders
`EIB7_NotInitialized`	Axis not configured
`EIB7_ParamInvalid`	Invalid parameter for axis

3.6.15 Verifying the status of the reference run

The status of the reference run is output. This enables the user to check whether the reference run is still active or all reference positions have already been saved.

Function

```
EIB7_ERR EIB7GetRefActive     ( EIB7_AXIS          axis,
                                EIB7_MODE*         active
                              )
```

Parameters

`axis`	AXIS handle
`active`	*[return value]* Pointer to the variable for the status

`active`	Description
`EIB7_MD_Enable`	Reference run active
`EIB7_MD_Disable`	Reference run complete

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.6.16 Activating the monitoring of the reference marks

Monitoring of the reference mark can be activated for incremental encoders. A tolerance value can also be indicated. This value specifies the maximum permissible deviation from the nominal value of the reference position.

Function

```
EIB7_ERR EIB7SetReferenceCheck    ( EIB7_AXIS        axis,
                                    EIB7_MODE        mode,
                                    unsigned long    limit
                                  )
```

Parameters

`axis` AXIS handle

`mode` Activate or deactivate the check of the reference marks

`mode`	Description
`EIB7_MD_Enable`	Activate the check
`EIB7_MD_Disable`	Deactivate the check

`limit` Maximum deviation between two reference positions in signal periods

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface` The axis is not configured for incremental encoders

`EIB7_ParamInvalid` Invalid parameter

3.6.17 EnDat 2.1 – reading the position

The position of an EnDat encoder is read. This occurs through an EnDat 2.1 command.

The function may only be performed in Polling mode of operation. The axis must have been configured for EnDat01, EnDat21 or EnDat22 encoders. An EnDat 2.1 command is always sent, even if the axis is configured for EnDat 2.2.

Function

```
EIB7_ERR EIB7EnDat21GetPosition   ( EIB7_AXIS           axis,
                                    unsigned short*     status,
                                    ENCODER_POSITION*   pos
                                  )
```

Parameters

`axis` AXIS handle

`Status` *[return value]* Pointer to the variable for the status word

`pos` *[return value]* Pointer to the variable for the position value

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)

3.6.18 EnDat 2.1 – selecting the memory area

The memory area in the EnDat encoder is selected. An EnDat 2.1 command is sent for this purpose.

The function may only be performed in Polling mode of operation. The axis must have been configured for EnDat01, EnDat21 or EnDat22 encoders. An EnDat 2.1 command is always sent, even if the axis is configured for EnDat 2.2.

Function

```
EIB7_ERR EIB7EnDat21SelectMemRange    ( EIB7_AXIS        axis,
                                        unsigned char    mrs
                                      )
```

Parameters

`axis`	AXIS handle
`mrs`	MRS code for the memory area

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)

3.6.19 EnDat 2.1 – sending data

A data word is written to the memory of the EnDat encoder. 16-bit words are always saved. The address indicates the memory location within the active memory block.

The function may only be performed in Polling mode of operation. The axis must have been configured for EnDat01, EnDat21 or EnDat22 encoders. An EnDat 2.1 command is always sent, even if the axis is configured for EnDat 2.2.

Function

```
EIB7_ERR EIB7EnDat21WriteMem      ( EIB7_AXIS          axis,
                                    unsigned char      addr,
                                    unsigned short     data
                                  )
```

Parameters

`axis`	AXIS handle
`addr`	Memory address within the active memory block
`data`	Data word that is written to the memory

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)

3.6.20 EnDat 2.1 – receiving data

A data word is read from the memory of the EnDat encoder. A 16-bit word is always read. The `addr` parameter indicates the memory location within the active memory block from which the data is read.

The function may only be performed in Polling mode of operation. The axis must have been configured for EnDat01, EnDat21 or EnDat22 encoders. An EnDat 2.1 command is always sent, even if the axis is configured for EnDat 2.2.

Function

```
EIB7_ERR EIB7EnDat21ReadMem       ( EIB7_AXIS          axis,
                                    unsigned char      addr,
                                    unsigned short*    data
                                  )
```

Parameters

`axis`	AXIS handle
`addr`	Memory address within the active memory block
`data`	*[return value]* Pointer to the variable for the received data word

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)

3.6.21 EnDat 2.1 – resetting the encoder

The EnDat reset command is sent to the encoder. The encoder is reset and thus cannot be reached for a certain time. Additional information can be found in the data sheet for the encoder.

The function may only be performed in Polling mode of operation. The axis must have been configured for EnDat01, EnDat21 or EnDat22 encoders. An EnDat 2.1 command is always sent, even if the axis is configured for EnDat 2.2.

Function

```
EIB7_ERR EIB7EnDat21ResetEncoder   ( EIB7_AXIS          axis
                                   )
```

Parameters

`axis`	AXIS handle

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)

3.6.22 EnDat 2.1 – reading a test value

A test value is read from the EnDat encoder. The test value is 40 bits long and is returned via two parameters. The contents of the parameters are listed in the table below.

Parameters	Data bit parameter	Data bit test value
`high`	D0 ... D7 D8 ... D31	D32 ... D39 Reserved
`low`	D0 ... D31	D0 ... D31

The function may only be performed in Polling mode of operation. The axis must have been configured for EnDat01, EnDat21 or EnDat22 encoders. An EnDat 2.1 command is always sent, even if the axis is configured for EnDat 2.2.

Function

```
EIB7_ERR EIB7EnDat21ReadTestValue ( EIB7_AXIS           axis,
                                    unsigned long*      high,
                                    unsigned long*      low
                                  )
```

Parameters

`axis`	AXIS handle
`high`	*[return value]* Pointer to the variable for the test value (most significant part)
`low`	*[return value]* Pointer to the variable for the test value (least significant part)

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)

3.6.23 EnDat 2.1 – sending a test command to encoder

A test command is sent to the EnDat encoder. The port address for the test command can be indicated using the `port` parameter. The axis must have been configured for EnDat01, EnDat21 or EnDat22 encoders. An EnDat 2.1 command is always sent, even if the axis is configured for EnDat 2.2.

Function

```
EIB7_ERR EIB7EnDat21WriteTestCommand ( EIB7_AXIS        axis,
                                       unsigned char    port
                                     )
```

Parameters

`axis`	AXIS handle
`port`	Port address for the test command

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)

3.6.24 EnDat 2.2 – reading the position and additional datum

The position of an EnDat22 encoder is read. If activated, the EnDat additional data is also transmitted. Each additional datum consists of a status word and the data word. The status word labels the data as valid or invalid and specifies the contents of the additional datum.

The function may only be performed in Polling mode of operation. The axis must have been configured for EnDat22 encoders.

Function

```
EIB7_ERR EIB7EnDat22GetPosition   ( EIB7_AXIS           axis,
                                    unsigned short*     status,
                                    ENCODER_POSITION*   pos,
                                    ENDAT_ADDINFO*      ai1,
                                    ENDAT_ADDINFO*      ai2
                                  )
```

Parameters

`axis`	AXIS handle
`Status`	*[return value]* Pointer to the variable for the status word
`pos`	*[return value]* Pointer to the variable for the position value
`ai1`	*[return value]* Pointer to the structure for the EnDat additional datum 1
`ai2`	*[return value]* Pointer to the structure for the EnDat additional datum 2

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)
`EIB7_EnDat22NotSupp`	The encoder does not support any EnDat 2.2 commands or the axis is not configured for EnDat 2.2 operation.

3.6.25 EnDat 2.2 – reading the position and additional data and selecting the memory area

The position of an EnDat22 encoder is read. If activated, the EnDat additional data is also transmitted. Each additional datum consists of a status word and the data word. The status word labels the data as valid or invalid and specifies the contents of the additional datum.

The function may only be performed in Polling mode of operation. The axis must have been configured for EnDat22 encoders.

Function

```
EIB7_ERR EIB7EnDat22SelectMemRange( EIB7_AXIS           axis,
                                    unsigned short*     status,
                                    ENCODER_POSITION*   pos,
                                    ENDAT_ADDINFO*      ai1,
                                    ENDAT_ADDINFO*      ai2,
                                    unsigned char       mrs,
                                    unsigned char       block
                                  )
```

Parameters

`axis`	AXIS handle
`Status`	*[return value]* Pointer to the variable for the status word
`pos`	*[return value]* Pointer to the variable for the position value
`ai1`	*[return value]* Pointer to the structure for the EnDat additional datum 1
`ai2`	*[return value]* Pointer to the structure for the EnDat additional datum 2
`mrs`	MRS code for the memory area
`block`	Block address for the "section 2" memory areas

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)
`EIB7_EnDat22NotSupp`	The encoder does not support any EnDat 2.2 commands or the axis is not configured for EnDat 2.2 operation.

3.6.26 EnDat 2.2 – reading the position and additional datum and sending data

The position and additional data of an EnDat22 encoder are transmitted (see "EnDat 2.2 – reading the position and additional datum", Page 117). The 16-bit data word is written to the memory of the encoder. The address (8-bit) indicates the memory location within the active memory block.

The function may only be performed in Polling mode of operation. The axis must have been configured for EnDat22 encoders.

Function

```
EIB7_ERR EIB7EnDat22WriteMem      ( EIB7_AXIS           axis,
                                    unsigned short*     status,
                                    ENCODER_POSITION*   pos,
                                    ENDAT_ADDINFO*      ai1,
                                    ENDAT_ADDINFO*      ai2,
                                    unsigned char       addr,
                                    unsigned short      data
                                  )
```

Parameters

`axis`	AXIS handle
`Status`	*[return value]* Pointer to the variable for the status word
`pos`	*[return value]* Pointer to the variable for the position value
`ai1`	*[return value]* Pointer to the structure for the EnDat additional datum 1
`ai2`	*[return value]* Pointer to the structure for the EnDat additional datum 2
`addr`	Memory address within the active memory block
`data`	Data word that is written to the memory

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)
`EIB7_EnDat22NotSupp`	The encoder does not support any EnDat 2.2 commands or the axis is not configured for EnDat 2.2 operation.

3.6.27 EnDat 2.2 – reading the position and additional datum and sending data

The position and additional data of an EnDat22 encoder are transmitted (see "EnDat 2.2 – reading the position and additional datum", Page 117). The encoder reads a data word from its memory; the address of the memory location within the selected memory area is specified using the `addr` parameter. The data is transmitted via the additional datum and can only be read out when the next EnDat command is issued. For this purpose, the suitable additional datum must be selected (see EnDat specification).

The function may only be performed in Polling mode of operation. The axis must have been configured for EnDat22 encoders.

Function

```
EIB7_ERR EIB7EnDat22ReadMem       ( EIB7_AXIS           axis,
                                    unsigned short*     status,
                                    ENCODER_POSITION*   pos,
                                    ENDAT_ADDINFO*      ai1,
                                    ENDAT_ADDINFO*      ai2,
                                    unsigned char       addr
                                  )
```

Parameters

`axis`	AXIS handle
`Status`	*[return value]* Pointer to the variable for the status word
`pos`	*[return value]* Pointer to the variable for the position value
`ai1`	*[return value]* Pointer to the structure for the EnDat additional datum 1
`ai2`	*[return value]* Pointer to the structure for the EnDat additional datum 2
`addr`	Memory address within the active memory block

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)
`EIB7_EnDat22NotSupp`	The encoder does not support any EnDat 2.2 commands or the axis is not configured for EnDat 2.2 operation.

3.6.28 EnDat 2.2 – reading the position and additional datum and sending the test command

The position and additional data of an EnDat22 encoder are transmitted (see "EnDat 2.2 – reading the position and additional datum", Page 117). The `port` parameter contains the port address for the test command.

The function may only be performed in Polling mode of operation. The axis must have been configured for EnDat22 encoders.

Function

```
EIB7_ERR EIB7EnDat22WriteTestCommand ( EIB7_AXIS           axis,
                                       unsigned short*     status,
                                       ENCODER_POSITION*   pos,
                                       ENDAT_ADDINFO*      ai1,
                                       ENDAT_ADDINFO*      ai2,
                                       unsigned char       port
                                     )
```

Parameters

`axis`	AXIS handle
`Status`	*[return value]* Pointer to the variable for the status word
`pos`	*[return value]* Pointer to the variable for the position value
`ai1`	*[return value]* Pointer to the structure for the EnDat additional datum 1
`ai2`	*[return value]* Pointer to the structure for the EnDat additional datum 2
`port`	Port address for the test command

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)
`EIB7_EnDat22NotSupp`	The encoder does not support any EnDat 2.2 commands or the axis is not configured for EnDat 2.2 operation.

3.6.29 EnDat 2.2 – reading the position and additional datum and sending an error reset

The position and additional data of an EnDat22 encoder are transmitted (see "EnDat 2.2 – reading the position and additional datum", Page 117). The error memory of the EnDat22 encoder is also cleared.

The function may only be performed in Polling mode of operation. The axis must have been configured for EnDat22 encoders.

Function

```
EIB7_ERR EIB7EnDat22ErrorReset    ( EIB7_AXIS           axis,
                                    unsigned short*     status,
                                    ENCODER_POSITION*   pos,
                                    ENDAT_ADDINFO*      ai1,
                                    ENDAT_ADDINFO*      ai2
                                  )
```

Parameters

`axis`	AXIS handle
`Status`	*[return value]* Pointer to the variable for the status word
`pos`	*[return value]* Pointer to the variable for the position value
`ai1`	*[return value]* Pointer to the structure for the EnDat additional datum 1
`ai2`	*[return value]* Pointer to the structure for the EnDat additional datum 2

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)
`EIB7_EnDat22NotSupp`	The encoder does not support any EnDat 2.2 commands or the axis is not configured for EnDat 2.2 operation.

3.6.30 EnDat 2.2 – selecting additional data

The additional data for an EnDat 2.2 encoder can be configured. The configuration must be done in Polling mode of operation. The additional data is transmitted in the Soft Real-Time, Streaming, and Recording operating modes.

The corresponding additional datum is selected in the encoder during a change from the Polling mode of operation to another mode. It is also possible to transmit only additional datum 1 or additional datum 2. To deactivate an additional datum, `EIB7_AI1_Stop` or `EIB7_AI2_Stop` must be passed as a parameter.

Function

```
EIB7_ERR EIB7EnDat22SetAddInfo    ( EIB7_AXIS          axis,
                                    unsigned long      addinfo1,
                                    unsigned long      addinfo2
                                  )
```

Parameters

`axis`	AXIS handle

`addinfo1` Additional datum 1 for EnDat 2.2

addinfo1	Value
EIB7_AI1_NOP	0x00
EIB7_AI1_Diagnostic	0x01
EIB7_AI1_Position2_word1	0x02
EIB7_AI1_Position2_word2	0x03
EIB7_AI1_Position2_word3	0x04
EIB7_AI1_MemoryLSB	0x05
EIB7_AI1_MemoryMSB	0x06
EIB7_AI1_MRS	0x07
EIB7_AI1_TestCommand	0x08
EIB7_AI1_TestValue_word1	0x09
EIB7_AI1_TestValue_word2	0x0A
EIB7_AI1_TestValue_word3	0x0B
EIB7_AI1_Temperature1	0x0C
EIB7_AI1_Temperature2	0x0D
EIB7_AI1_AddSensor	0x0E
EIB7_AI1_Stop	0x0F

`addinfo2` Additional datum 2 for EnDat 2.2

addinfo2	Value
EIB7_AI2_NOP	0x10
EIB7_AI2_Commutation	0x11
EIB7_AI2_Acceleration	0x12
EIB7_AI2_CommAndAccel	0x13
EIB7_AI2_LimitSignal	0x14
EIB7_AI2_LimitAndAccel	0x15
EIB7_AI2_AsyncPos_word1	0x16
EIB7_AI2_AsyncPos_word2	0x17
EIB7_AI2_AsyncPos_word3	0x18
EIB7_AI2_OPSErrorSource	0x19
EIB7_AI2_ReservedA	0x1A
EIB7_AI2_ReservedB	0x1B
EIB7_AI2_ReservedC	0x1C
EIB7_AI2_ReservedD	0x1D
EIB7_AI2_ReservedE	0x1E
EIB7_AI2_Stop	0x1F

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid`	Invalid parameter
`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EnDat22NotSupp`	The encoder does not support any EnDat 2.2 commands or the axis is not configured for EnDat 2.2 operation.

3.6.31 EnDat 2.2 – selecting the sequence for additional data

The additional data for an EnDat 2.2 encoder can be configured. The configuration must be done in Polling mode of operation. The additional data is transmitted in the Soft Real-Time, Streaming, and Recording operating modes.

The sequence of additional data is relayed with each trigger, and after the last entry it begins again with the first entry. The sequence can comprise no more than 10 entries. Additional data 1 and 2 can be selected.

Function

```
EIB7_ERR EIB7EnDat22SetAddInfoCycle  ( EIB7_AXIS        axis,
                                       EIB7_MODE        mode,
                                       unsigned char*   data,
                                       unsigned long    len
                                     )
```

Parameters

`axis` AXIS handle

`mode` Activate or deactivate FIFO

`mode`	Description
`EIB7_MD_Disable`	Deactivate FIFO for additional datum
`EIB7_MD_Enable`	Activate FIFO for additional datum

`data` Pointer to an array with the configuration data. Every byte contains an additional datum 1 or 2

Array element	Value
EIB7_AI1_NOP	0x00
EIB7_AI1_Diagnostic	0x01
EIB7_AI1_Position2_word1	0x02
EIB7_AI1_Position2_word2	0x03
EIB7_AI1_Position2_word3	0x04
EIB7_AI1_MemoryLSB	0x05
EIB7_AI1_MemoryMSB	0x06
EIB7_AI1_MRS	0x07
EIB7_AI1_TestCommand	0x08
EIB7_AI1_TestValue_word1	0x09
EIB7_AI1_TestValue_word2	0x0A
EIB7_AI1_TestValue_word3	0x0B
EIB7_AI1_Temperature1	0x0C
EIB7_AI1_Temperature2	0x0D
EIB7_AI1_AddSensor	0x0E
EIB7_AI1_Stop	0x0F
EIB7_AI2_NOP	0x10
EIB7_AI2_Commutation	0x11
EIB7_AI2_Acceleration	0x12
EIB7_AI2_CommAndAccel	0x13
EIB7_AI2_LimitSignal	0x14
EIB7_AI2_LimitAndAccel	0x15
EIB7_AI2_AsyncPos_word1	0x16
EIB7_AI2_AsyncPos_word2	0x17
EIB7_AI2_AsyncPos_word3	0x18
EIB7_AI2_OPSErrorSource	0x19
EIB7_AI2_ReservedA	0x1A
EIB7_AI2_ReservedB	0x1B
EIB7_AI2_ReservedC	0x1C
EIB7_AI2_ReservedD	0x1D

EIB7_AI2_ReservedE	0x1E
EIB7_AI2_Stop	0x1F

`len` Size of array in bytes (≤ 9)

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid`	Invalid parameter
`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EnDat22NotSupp`	The encoder does not support any EnDat 2.2 commands or the axis is not configured for EnDat 2.2 operation.

3.6.32 Reading absolute and incremental position values simultaneously

The position of an EnDat encoder is read. An EnDat command is sent to the encoder for this purpose. At the same time, the position value is generated from the incremental signals. The two position values are returned together with the status words. The function can only be performed in Polling mode of operation. The axis must be configured for EnDat 01 encoders.

Function

```
EIB7_ERR EIB7ReadEnDatIncrPos     ( EIB7_AXIS           axis,
                                    unsigned short*     statusEnDat,
                                    ENCODER_POSITION*   posEnDat,
                                    unsigned short*     statusIncr,
                                    ENCODER_POSITION*   posIncr
                                  )
```

Parameters

`axis`	AXIS handle
`statusEnDat`	*[return value]* Pointer to the variable for the status word of the EnDat position
`posEnDat`	*[return value]* Pointer to the variable for the EnDat position value
`statusIncr`	*[return value]* Pointer to the variable for the status word of the incremental position
`posIncr`	*[return value]* Pointer to the variable for the incremental position value

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_InvInterface`	The axis is not configured for EnDat encoders
`EIB7_NotInitialized`	Axis not initialized
`EIB7_EncPwrSuppErr`	Encoder power supply error (encoder not ready for operation)
`EIB7_EnDatErrII`	A type II EnDat error occurred
`EIB7_EnDatIfBusy`	EnDat master not ready for operation
`EIB7_EnDatXmitErr`	Data transmission error (encoder might not be connected)
`EIB7_CantLatchPos`	Position cannot be determined

3.6.33 Setting the power supply for encoders

The encoder power supply can be activated or deactivated. The `mode` parameter is used to determine whether the power supply will be switched on or off.

Function

```
EIB7_ERR EIB7SetPowerSupply       ( EIB7_AXIS           axis,
                                    EIB7_MODE           mode
                                  )
```

Parameters

`axis`	AXIS handle
`mode`	Activate or deactivate the power supply

`mode`	Description
`EIB7_MD_Disable`	Switch off power supply
`EIB7_MD_Enable`	Switch on power supply

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.6.34 Reading the power supply status for encoders

The power supply status for the encoder can be read out. The `power` parameter can be used to determine whether the power supply for this axis will be switched on or off. The `err` parameter indicates whether an error has occurred and the power supply has been switched off due to excessive current load.

Function

```
EIB7_ERR EIB7GetPowerSupplyStatus ( EIB7_AXIS           axis,
                                    EIB7_MODE*          power,
                                    EIB7_POWER_FAILURE* err
                                  )
```

Parameters

`axis` AXIS handle

`Power` *[return value]* Pointer to the variable for the status of the power supply

`Power`	Description
`EIB7_MD_Disable`	Power supply switched off
`EIB7_MD_Enable`	Power supply switched on

`Err` *[return value]* Pointer to the variable for the overcurrent error

`Err`	Description
`EIB7_PF_None`	No error
`EIB7_PF_Overcurrent`	Power supply has been deactivated due to overcurrent

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.6.35 Configuring the timestamp

The timestamp can be activated or deactivated for each axis. The period duration is set globally for all axes of an EIB 74x. The timestamp value for a position poll for an axis will be copied if this function has been activated previously.

Function

```
EIB7_ERR EIB7SetTimestamp          ( EIB7_AXIS           axis,
                                     EIB7_MODE           mode
                                   )
```

Parameters

`axis` AXIS handle

`mode` Activate or deactivate timestamp

`mode`	Description
`EIB7_MD_Disable`	Deactivate timestamp
`EIB7_MD_Enable`	Activate timestamp

Return value

The return value supplies a status for the function call. All possible values are listed in the Standard return values table above (see "Table 15: Standard return values", Page 50).

3.7 IO functions

The IO functions always refer only to an individual output or input port on the EIB 74x. The other ports are not affected.

Besides the standard return values ("Parameters and return values"), all IO functions may also return further values. These are indicated separately for each function.

3.7.1 Configuring the input port

The mode for an input port can be configured using this function. The port can be used as a trigger input or a logical input. The terminating resistor of the differential input can also be activated or deactivated. This function is only permitted in conjunction with handles to input ports.

Function

```
EIB7_ERR EIB7InitInput          ( EIB7_IO             io,
                                  EIB7_IOMODE         mode,
                                  EIB7_MODE           termination
                                )
```

Parameters

`io` IO handle

`mode` Trigger input or logical input port

mode	Description
EIB7_IOM_Trigger	Trigger input
EIB7_IOM_Logical	Logical input

`termination` Activate or deactivate the terminating resistor

termination	Description
EIB7_MD_Disable	Deactivate the terminating resistor
EIB7_MD_Enable	Activate the terminating resistor

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.7.2 Configuring the output port

The mode for an output port can be configured using this function. The port can be used as a trigger output or a logical output. Also, the output driver can be deactivated. In this case, the output is in a high-impedance state. This function is only permitted in conjunction with handles to output ports.

Function

```
EIB7_ERR EIB7InitOutput        ( EIB7_IO            io,
                                 EIB7_IOMODE        mode,
                                 EIB7_MODE          enable
                               )
```

Parameters

`io` IO handle

`mode` Trigger output or logical output port

`mode`	Description
`EIB7_IOM_Trigger`	Trigger output
`EIB7_IOM_Logical`	Logical output

`enable` Activate or deactivate output driver

`enable`	Description
`EIB7_MD_Disable`	Deactivate output driver
`EIB7_MD_Enable`	Activate output driver

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.7.3 Selecting the trigger source for a trigger output

The trigger signal for the trigger output can be selected from different sources. This setting is possible only in Polling mode of operation and can only be used for trigger outputs.

Function

```
EIB7_ERR EIB7OutputTriggerSource  ( EIB7_IO               io,
                                    EIB7_OutputTriggerSrc src
                                  )
```

Parameters

`io` IO handle

`src` Trigger source

src	Description
`EIB7_OT_TrgInSync`	Trigger input synchronized
`EIB7_OT_TrgInAsync`	Trigger input not synchronized
`EIB7_OT_TrgSW1`	Software trigger channel 1
`EIB7_OT_TrgSW2`	Software trigger channel 2
`EIB7_OT_TrgSW3`	Software trigger channel 3
`EIB7_OT_TrgSW4`	Software trigger channel 4
`EIB7_OT_TrgRImaskedCH1`	Linked reference pulse from axis 1 (A&B&RI)
`EIB7_OT_TrgIC`	Interval counter
`EIB7_OT_TrgPuls`	Trigger pulse counter
`EIB7_OT_TrgTimer`	Timer trigger

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.7.4 Setting the delay time for the trigger input

The time by which a trigger pulse is to be delayed can be set separately for each trigger input. The delay time must be indicated as an integer multiple of the clock pulse period. The number of clock pulse periods per microsecond can be read out (`EIB7GetTriggerDelayTicks()`). The delay time can be deactivated by setting the respective parameter value to zero. This function is only applicable to trigger inputs.

To calculate the value of a period (e.g. for the `period` parameter of the `EIB7SetTimestampPeriod`) function call correctly, pass the following values to the function:
`period` = interval in µs * clock ticks per µs
To read out the value of "clock ticks per µs", you can e.g. use the `EIB7GetTimerTriggerTicks` or `EIB7GetTimestampTicks` function.

Function

```
EIB7_ERR EIB7SetTriggerInputDelay ( EIB7_IO          io,
                                    unsigned long    dly
                                  )
```

Parameters

`io`	IO handle (only for inputs), for more details see "Timestamp", Page 30
`dly`	Delay time in clock cycles (≤ 256)

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid`	Invalid parameter

3.7.5 Reading out a logical port

The level at a logical input or output is read out (`level` parameter). The operating mode of the port is also determined. If the port is operated as a trigger input or as a trigger output, the value of `level` is invalid. For a logical output, the level that was set is returned.

Function

```
EIB7_ERR EIB7ReadIO            ( EIB7_IO             io,
                                 EIB7_IOMODE*        mode,
                                 unsigned long*      level
                               )
```

Parameters

`io` IO handle

`mode` *[return value]* Pointer to the variable for the mode of operation

`mode`	Description
`EIB7_IOM_Trigger`	Trigger port
`EIB7_IOM_Logical`	Logical port

`level` *[return value]* Pointer to the variable for the logical level of the port

`level`	Description
`EIB7_MD_Disable`	Deactivate output driver
`EIB7_MD_Enable`	Activate output driver

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid`	Invalid parameter

3.7.6 Setting the logical output port

The level of a logical output port is set. The `level` parameter indicates whether the output is set to high or low. This function can only be applied to outputs that have been configured as logical ports. If the port is used as a trigger output, the function generates an error message.

Function

```
EIB7_ERR EIB7WriteIO          ( EIB7_IO             io,
                                unsigned long       level
                              )
```

Parameters

`io` IO handle

`level` Logical level of the output

`Level`	Description
`0`	Low level
`1`	High level

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

`EIB7_TrgNotConf` Output is not a logical port

3.7.7 Reading the configuration data for an input

The configuration data for an input port is read out. The `mode` parameter returns the operating mode of the input. The `termination` parameter returns the status of the terminating resistor. The function may only be used for input ports.

Function

```
EIB7_ERR EIB7GetInputConfig   ( EIB7_IO             io,
                                EIB7_IOMODE*        mode,
                                EIB7_MODE*          termination
                              )
```

Parameters

`io` IO handle

`mode` *[return value]* Pointer to the variable for the mode of operation

`mode`	Description
`EIB7_IOM_Trigger`	Trigger input
`EIB7_IOM_Logical`	Logical input

`termination` *[return value]* Pointer to the variable for the terminating resistor

`termination`	Description
`EIB7_MD_Disable`	Terminating resistor deactivated
`EIB7_MD_Enable`	Terminating resistor activated

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.7.8 Reading the configuration data for an output

The configuration data for an input port is read out. The `mode` parameter returns the operating mode of the output. With the `enable` parameter, the status of the output driver is returned. The function may only be used for output ports.

Function

```
EIB7_ERR EIB7GetOutputConfig    ( EIB7_IO             io,
                                  EIB7_IOMODE*        mode,
                                  EIB7_MODE*          enable
                                )
```

Parameters

`io` IO handle

`mode` *[return value]* Pointer to the variable for the mode of operation

`mode`	Description
`EIB7_IOM_Trigger`	Trigger output
`EIB7_IOM_Logical`	Logical output

`enable` *[return value]* Pointer to the variable for the status of the output driver

`enable`	Description
`EIB7_MD_Disable`	Output driver deactivated
`EIB7_MD_Enable`	Output driver activated

Return value

The return value supplies a status for the function call. In addition to the standard return values (see "Table 15: Standard return values", Page 50), the following error messages can occur:

`EIB7_ParamInvalid` Invalid parameter

3.8 General functions

Besides the standard return values ("Parameters and return values"), all general functions may also return further values. These are indicated separately for each function.

3.8.1 Reading the driver ID number

The product number (ID) of the driver is returned as a C string. The string is saved to the `ident` pointer. The size of the memory for the string must be indicated in bytes via the `len` parameter. If the string, including the final null byte, is longer than the memory area, an error message will be generated. The target memory must be at least 9 bytes.

Function

```
EIB7_ERR EIB7GetDriverID       ( char*              ident,
                                 unsigned long      len
                               )
```

Parameters

`ident`	*[return value]* Target memory for the C string
`len`	Size of the target memory in bytes

Return value

The return value supplies a status for the function call. Possible values are listed below:

`EIB7_NoError`	Function call was successful
`EIB7_OutOfMemory`	The system cannot allocate sufficient memory
`EIB7_BufferTooSmall`	Target memory is too small

3.8.2 Converting an error message into text

An error code is converted into a text message and returned as a C string. A brief description and a descriptive text are defined in the system for all known error codes.

The `mnemonic` parameter returns a brief description of the error messages as text (approx. 30 to 40 characters). The `message` parameter contains a more detailed description of the error messages as text (approx. 100 to 150 characters). If a NULL pointer is passed for one of the `mnemonic` or `message` parameters, the function will not copy the text. If the target memory is too small to take the entire text, only the first part is copied. The string always ends with a null byte.

Function

```
EIB7_ERR EIB7GetErrorInfo   ( EIB7_ERR              code,
                              char*                 mnemonic,
                              unsigned long         mnemlen,
                              char*                 message,
                              unsigned long         msglen
                            )
```

Parameters

`code`	Error code that is converted into text
`mnemonic`	*[return value]* Pointer to the target memory for the brief description
`mnemlen`	Size of the `mnemonic` target memory in bytes
`message`	*[return value]* Pointer to the target memory for the error text
`msglen`	Size of the `message` target memory in bytes

Return value

The return value supplies a status for the function call. Possible values are listed below:

`EIB7_NoError`	Function call was successful
`EIB7_OutOfMemory`	The system cannot allocate sufficient memory
`EIB7_IllegalParameter`	Invalid error code

HEIDENHAIN

DR. JOHANNES HEIDENHAIN GmbH
Dr.-Johannes-Heidenhain-Straße 5
83301 Traunreut, Germany
☏ +49 8669 31-0
FAX +49 8669 32-5061
E-mail: info@heidenhain.de

Technical support FAX +49 8669 32-1000
Measuring systems ☏ +49 8669 31-3104
E-mail: service.ms-support@heidenhain.de
NC support ☏ +49 8669 31-3101
E-mail: service.nc-support@heidenhain.de
NC programming ☏ +49 8669 31-3103
E-mail: service.nc-pgm@heidenhain.de
PLC programming ☏ +49 8669 31-3102
E-mail: service.plc@heidenhain.de
APP programming ☏ +49 8669 31-3106
E-mail: service.app@heidenhain.de

www.heidenhain.de

Original document
752925 · 07 · A · 02 · 12/2019 · PDF