



# HEIDENHAIN



User's Manual

## **EIB 741**

External Interface Box  
for Connecting  
HEIDENHAIN Encoders

November 2008

<b>DOCUMENTATION .....</b>	<b>5</b>
<b>FIRMWARE VERSION .....</b>	<b>5</b>
<b>PART 1: FEATURES .....</b>	<b>6</b>
<b>1 GENERAL DESCRIPTION OF FUNCTIONS .....</b>	<b>6</b>
<b>2 CONFIGURATION OF THE ENCODER INPUTS .....</b>	<b>7</b>
2.1 Processing Incremental Signals .....	7
2.2 Analog Value of the 1 VPP Increment Signals A and B .....	9
2.3 Dealing with Reference Marks .....	10
2.4 Processing EnDat Signals .....	12
<b>3 PROCESSING TRIGGER EVENTS .....</b>	<b>15</b>
3.1 Trigger Inputs and Outputs .....	15
3.2 Logical Inputs and Outputs .....	16
3.3 Maximum Trigger Rate .....	16
3.4 Counter for Accepted Trigger Events .....	16
<b>4 TIMESTAMP .....</b>	<b>17</b>
<b>5 STATUS WORD .....</b>	<b>17</b>
<b>6 ETHERNET INTERFACE .....</b>	<b>20</b>
<b>7 OPERATING MODES .....</b>	<b>20</b>
7.1 "Polling" Mode .....	20
7.2 "Soft Realtime" Mode .....	21
<b>8 FIRMWARE UPDATE .....</b>	<b>23</b>
<b>9 RESET .....</b>	<b>23</b>
<b>PART 2: DRIVER SOFTWARE .....</b>	<b>24</b>
<b>1 GENERAL INFORMATION .....</b>	<b>24</b>
<b>2 INSTALLATION INSTRUCTIONS .....</b>	<b>24</b>
2.1 Windows .....	24
2.2 Linux .....	24
<b>3 OVERVIEW .....</b>	<b>24</b>
3.1 Establishing Communication .....	24
3.2 Polling Mode .....	24
3.3 Soft Realtime Mode .....	24
<b>4 DATA TYPES .....</b>	<b>25</b>
4.1 Simple Data Types .....	25
4.2 EnDat Additional Information .....	25
4.3 Information for TCP Connection .....	25
<b>5 PARAMETERS AND RETURN VALUES .....</b>	<b>25</b>
<b>6 AUXILIARY FUNCTIONS .....</b>	<b>25</b>
6.1 Determining IP Address .....	25
6.2 Changing Format of Position Data .....	26

<b>7</b>	<b>DEVICE FUNCTIONS</b>	<b>27</b>
7.1	Opening Connection to the EIB 741	27
7.2	Closing Connection to the EIB 741	28
7.3	Polling Connection Status	28
7.4	Setting up Timeout	28
7.5	Requesting Handle for Axis	29
7.6	Requesting IO Port Handle	29
7.7	Selecting Operating Mode	30
7.8	Saving Network Parameters	30
7.9	Reading Network Parameters	31
7.10	Saving Hostname	31
7.11	Reading Hostname	32
7.12	Reading Serial Number	32
7.13	Reading the Device ID	32
7.14	Reading MAC Address	33
7.15	Reading Firmware Version Number	33
7.16	Reading Boot Mode	34
7.17	Reading Update Status	34
7.18	Reading Number of Open Connections	34
7.19	Reading Connection Data	35
7.20	Terminating the Connection	35
7.21	Reading the Time Unit for Timestamp	35
7.22	Setting the Timestamp Period	36
7.23	Resetting the Timestamp Counter	36
7.24	Reading the Time Unit for Timer Trigger	36
7.25	Setting the Timer Trigger Period	37
7.26	Activating the Timer Trigger	37
7.27	Clearing the Trigger Counter	37
7.28	Setting the Termination Resistors	38
7.29	Enabling the External Trigger Signal	38
7.30	Software Trigger	39
7.31	Reset	39
7.32	Identifying EIB 741	39
7.33	Reading Data from the FIFO	40
7.34	Reading the Size of a FIFO Element	40
7.35	Access to the Contents of a FIFO Element	41
7.36	Reading and Converting Data from the FIFO	42
7.37	Reading the Size of a FIFO Element (Converted Data)	42
7.38	Access to the Contents of a FIFO Element with Converted Data	43
7.39	Reading the Number of Elements in the FIFO	44
7.40	Clearing the FIFO	44
7.41	Setting the FIFO Size	44
7.42	Reading the FIFO Size	44
7.43	Activating the Callback Mechanism	45

<b>8</b>	<b>AXIS FUNCTIONS</b>	<b>46</b>
8.1	Initializing the Axis	46
8.2	Clearing the Counter	49
8.3	Polling Position	49
8.4	Reading Data for a Channel	50
8.5	Acknowledging the Power Supply Error	50
8.6	Acknowledging the Trigger Error	51
8.7	Acknowledging Signal Errors	51
8.8	Clearing EnDat Error Bits	51
8.9	Clearing Status Bits for Reference Marks	52
8.10	Clearing Status Bits for Distance-Coded Reference Marks	52
8.11	Starting Referencing	52
8.12	Stopping Referencing	53
8.13	Verifying Referencing Status	53
8.14	EnDat 2.1: Reading Position	53
8.15	EnDat 2.1: Selecting Memory Area	54
8.16	EnDat 2.1: Sending Data	54
8.17	EnDat 2.1: Receiving Data	55
8.18	EnDat 2.1: Resetting the Encoder	55
8.19	EnDat 2.1: Reading Test Values	56
8.20	EnDat 2.1: Sending Test Command to Encoder	56
8.21	EnDat 2.2: Reading Position and Additional Information	57
8.22	EnDat 2.2: Reading Position and Additional Information and Selecting Memory Area	57
8.23	EnDat 2.2: Reading Position and Additional Information and Sending Data	58
8.24	EnDat 2.2: Reading Position and Additional Information and Receiving Data	59
8.25	EnDat 2.2: Reading Position and Additional Information and Sending Test Command	60
8.26	EnDat 2.2: Reading Position and Additional Information and Sending Error Reset	61
8.27	Reading Absolute and Incremental Position Values Simultaneously	61
8.28	Setting the Power Supply for Encoders	62
8.29	Reading Power Supply Status for Encoders	62
8.30	Configuring Timestamp	63
<b>9</b>	<b>IO FUNCTIONS</b>	<b>64</b>
9.1	Configuring the Input Port	64
9.2	Configuring the Output Port	65
9.3	Reading the Logical Port	65
9.4	Setting the Logical Output Port	66
9.5	Reading Configuration Data for Input	66
9.6	Reading Configuration Data for Output	67
<b>10</b>	<b>GENERAL FUNCTIONS</b>	<b>68</b>
10.1	Reading the Driver ID Number	68
10.2	Converting Error Message into Text	68

## Documentation

The documentation for the EIB 741 comprises the following documents:

- Installation Instructions:  
Documents required for operation, as well as technical specifications.
- User's Manual:
  - Description of features on the EIB 741.
  - Description of the installation and function calls of the driver software.

## Firmware Version

This document describes Firmware version: 633281-06

This Firmware is subject to the following limitations in comparison with the final version:

- Operating modes "Recording", and "Streaming" are not supported
- Limited features of the trigger interface
  - Interval counter and trigger at reference mark is still not supported
  - Only one external trigger input, which triggers all channels simultaneously, is supported
- EnDat clock frequency can be set only up to 2 MHz
- EnDat position polls are supported only by polling mode

# Part 1: Features

## 1 General Description of Functions

The EIB 741 is an external interface box for precise position measurement. It is ideal for inspection stations and multipoint inspection apparatuses as well as for mobile data acquisition, such as in machine inspection and calibration.

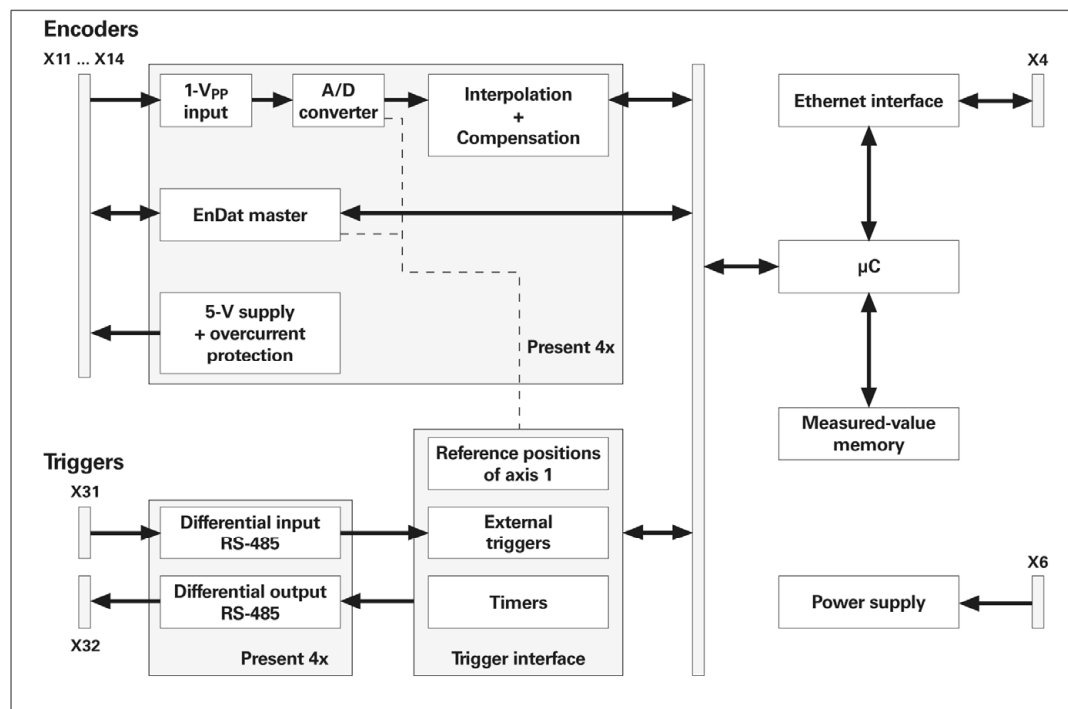
The EIB 741 is ideal for applications requiring high-resolution encoder signals and fast data logging. Ethernet transmission also enables you to use switches or hubs for connecting more than one EIB 741.

A maximum of four HEIDENHAIN encoders, either with sinusoidal incremental signals (1 V<sub>PP</sub>) or with EnDat interfaces (EnDat 2.1 and EnDat 2.2) can be connected to the EIB 741.

The EIB 741 subdivides the periods of the incremental signals 4096-fold for measured-value generation. The deviations within one signal period are automatically reduced by adjusting the sinusoidal incremental signals (signal compensation). The integrated measured-value memory enables the EIB 741 to save up to 250,000 measured values per axis in "Recording mode". Internal or external triggers can be used for the 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. In order to process the measured values on the PC, driver software for Windows, Linux and LabVIEW is included in the items supplied. The driver software facilitates programming of customer applications. It also contains program examples demonstrating the performance range of the EIB 741.

Basic Circuit Diagram



### Encoder Inputs:

The EIB 741 can be connected to up to four HEIDENHAIN encoders with the following interfaces (freely programmable):

- Incremental signals 1 V<sub>PP</sub>
- EnDat 2.1
- EnDat 2.2

The power supply to the encoders is provided by the EIB 741 and is protected by a resettable overload cutout. For technical specifications, see "Installation Instructions".

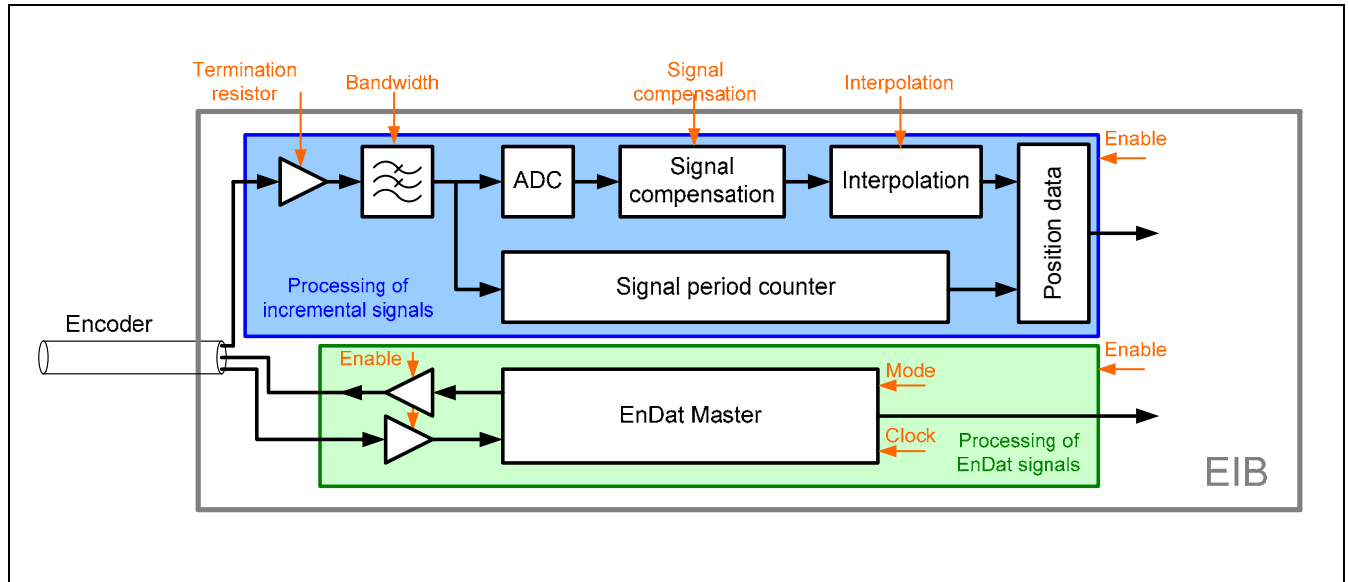
## 2 Configuration of the Encoder Inputs

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<sub>PP</sub> Input signals
- Signal compensation
- Processing the reference marks

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<sub>PP</sub> interface.



### 2.1 Processing Incremental Signals

The EIB 741 subdivides the periods of the incremental signals 4096-fold for position value generation (12-bit). The period counter has a width of 32 bits. Its 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 adjusting the sinusoidal incremental signals (signal compensation). Incremental signal compensation of the encoder and the termination resistor can be activated and deactivated by software.

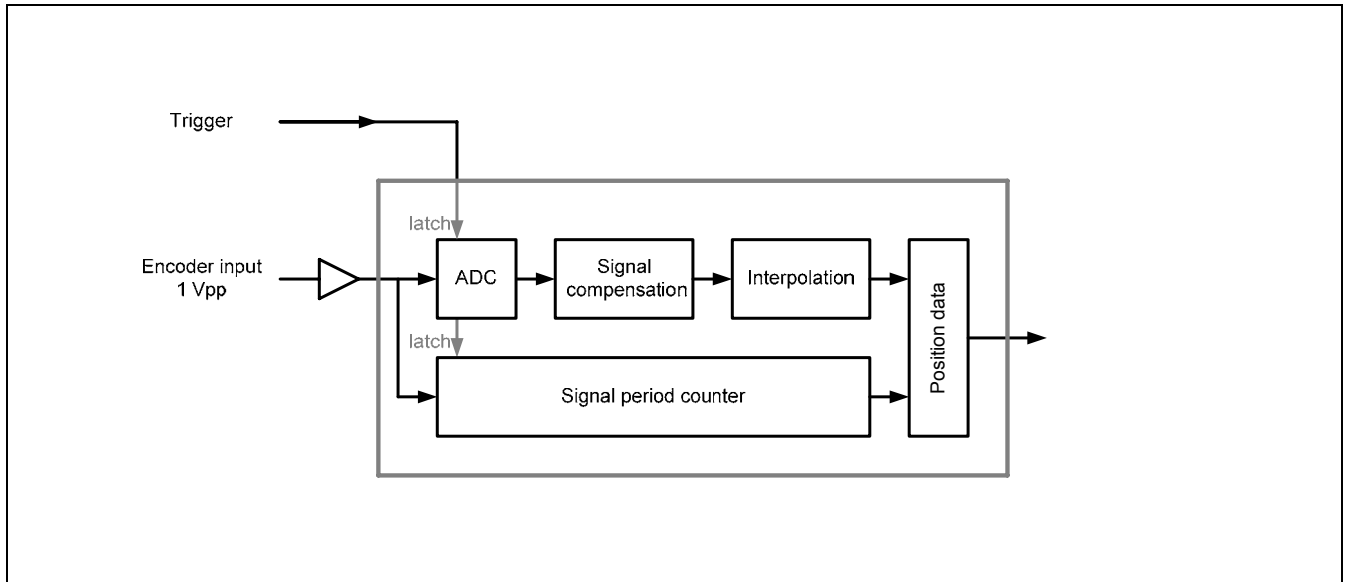
The 44-bit wide position information at the time of the trigger event is formed from the interpolation value (12 bits) and the value of the period counter (32 bits). The position information is saved to a 48-bit wide register (see table). In doing so, the period counter is mapped in the two's complement;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

Depending on the encoder type (linear or rotative), the higher-ranking customer software application can use this value to calculate the angle and length respectively.

The period counter overflows corresponding to the two's complement at position: 0x07FF FFFF FFFF (maximum positive) → 0xF800 0000 0000 (maximum negative). This overflow does not affect the functionality of the period counter or the interpolator. The overflow, however, must be handled by the higher-ranking customer software application.

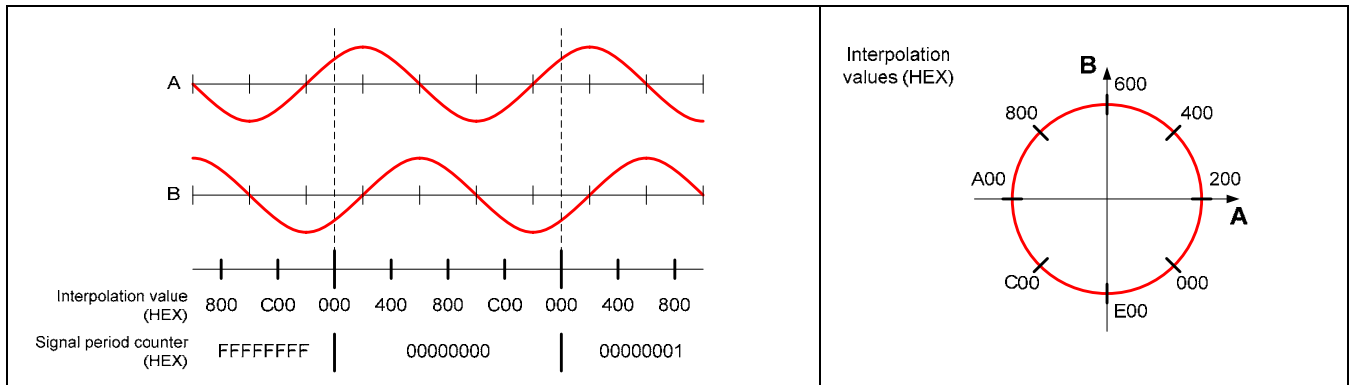
Bit no.	Width (bit)	Contents
0..11	12	Interpolation value
12..43	32	Period counter (bit 43 = sign)
44..47	4	Identical value to bit 43

## Block Diagram



## Interpolation Value

At the time of the trigger event, the incremental signals are sampled and used to calculate a 12-bit wide interpolation value. The correlation between interpolation value and incremental signals is derived as follows:



### Setting Options:

#### Termination Resistor for the Incremental Signals:

The 120 ohm termination resistor for the 1 V<sub>PP</sub> increment signals can be activated or deactivated by software (for all channels simultaneously; default: resistors activated).

#### Bandwidth Setting of the Incremental Signals:

The bandwidth of the encoder's incremental signals can be toggled by software. The high bandwidth (500 kHz) setting should be set as default.

The low bandwidth (33 kHz) setting is only to be selected for special applications.

#### Signal Compensation:

Increment signal compensation of the encoder can be activated or deactivated by software.

## 2.2 Analog Value of the 1 V<sub>PP</sub> Increment Signals A and B

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

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

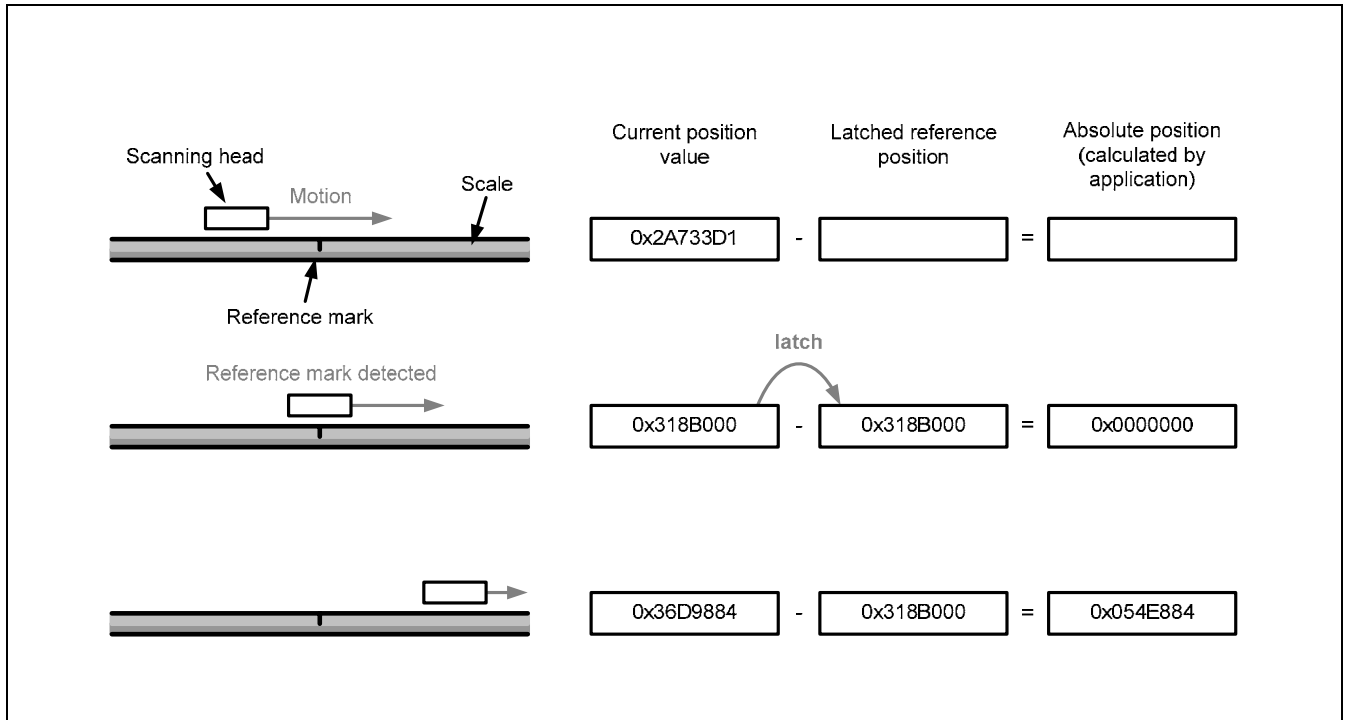
Value (HEX)	Incremental Signal Value
000	Negative maximum
800	Zero
FFF	Positive maximum

### 2.3 Dealing with Reference Marks

In the case of incremental encoders, the reference mark or reference marks is or are used to generate an absolute reference for the incremental signals.

In the case of encoders with one reference mark, this mark has an explicit reference to a specific signal period. This signal period can be used as a reference to generate absolute position values. Crossing the reference mark does not affect the period counter or the interpolation value. The period counter value valid at the time of transfer is simply latched in a register for the reference position. This value can be used in the customer software application to calculate absolute position values.

The diagram below shows the general process for determining a reference position. The displayed values are only an example. For the sake of clarity, only a section of the position register is shown.



Register contents, reference position

Bit no.	Width (bit)	Contents
0..11	12	Always 0
12..43	32	Reference position (value of the period counter when the reference mark is detected; bit 43 = sign)
44..47	4	Identical value to bit 43

Automatic latching of the reference position must be activated by software. After this command, the EIB 741 waits for the next reference mark and then latches the reference position.

This feature must be activated again if further latching 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 741 transmits the two reference positions and, where applicable, the coded reference value:

- In the case of encoders with one reference mark, only reference position 1 is normally used,
- In the case of encoders with distance-coded reference marks, both register values or the coded reference value is used depending on the evaluation method.

**Distance-Coded Reference Marks:**

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

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

When calculating the absolute position value using the customer software application, this value is treated, , in exactly the same way as a latched reference position value from encoders with 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 741. Once the reference position has been latched successfully, the EIB 741 uses this information to automatically calculate the coded reference value. The latching process is started by software command (for two reference marks). After the second reference mark has been crossed, the EIB 741 automatically calculates the coded reference value and transfers it to the customer software application.

**Method 2:** (especially for applications with an extremely low traversing speed)

The axis is initialized as an incremental system with single reference marks. The customer software application sends the appropriate software command for latching the reference position (one reference mark). Each time the reference position is latched successfully, the latch process is activated again. This exercise must be repeated until two different reference positions have been recorded. From these two values the customer software application can then calculate the coded reference value and hence the absolute position. It must be guaranteed that the customer software application can quickly and sufficiently complete this procedure, 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 single reference marks. The customer software application sends the appropriate software command for latching two reference positions. When both reference positions have been saved successfully (both reference position registers are used), the customer software application can then calculate the coded reference value and hence the absolute position.

## 2.4 Processing EnDat Signals

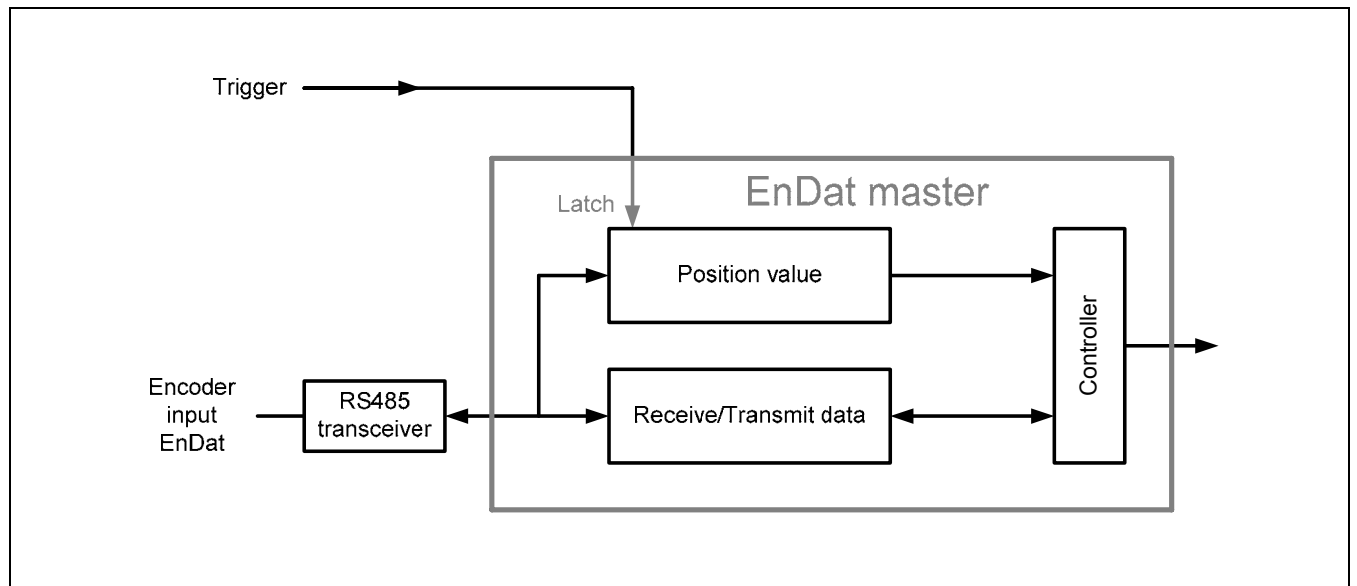
HEIDENHAIN absolute encoders are available with EnDat 2.1 or EnDat 2.2 interface. Especially in the case of EnDat 2.1 encoders, 1 V<sub>PP</sub> incremental signals are transmitted along with the EnDat signals. The EIB 741 is able to process all EnDat encoders with EnDat 2.1 or EnDat 2.2 interfaces both serially and also with 1 V<sub>PP</sub> incremental signals.

The EnDat Master is set-up individually when the axis is initialized:

- EnDat 2.1 or EnDat 2.2 communication can be set-up.
- The clock frequency for the EnDat communication can be set-up.
- Delay compensation (EnDat 2.2) can be activated or deactivated.

Please note:

- If EnDat position polls and 1 V<sub>PP</sub> incremental signals are used at the same time, only EnDat 2.1 mode commands can be send to the encoder (axis must be configured for EnDat 01).
- The EnDat position can only be imported by a software command. Hence, EnDat position and incremental position must be imported once (special command). The incremental position may then be transferred cyclically in soft realtime mode (see operating modes).



### 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 is 48-bit wide. The number of bits used for the position value depends on the connected EnDat encoder; the top, unused bits must be masked. See the encoder specifications for more detailed information.

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

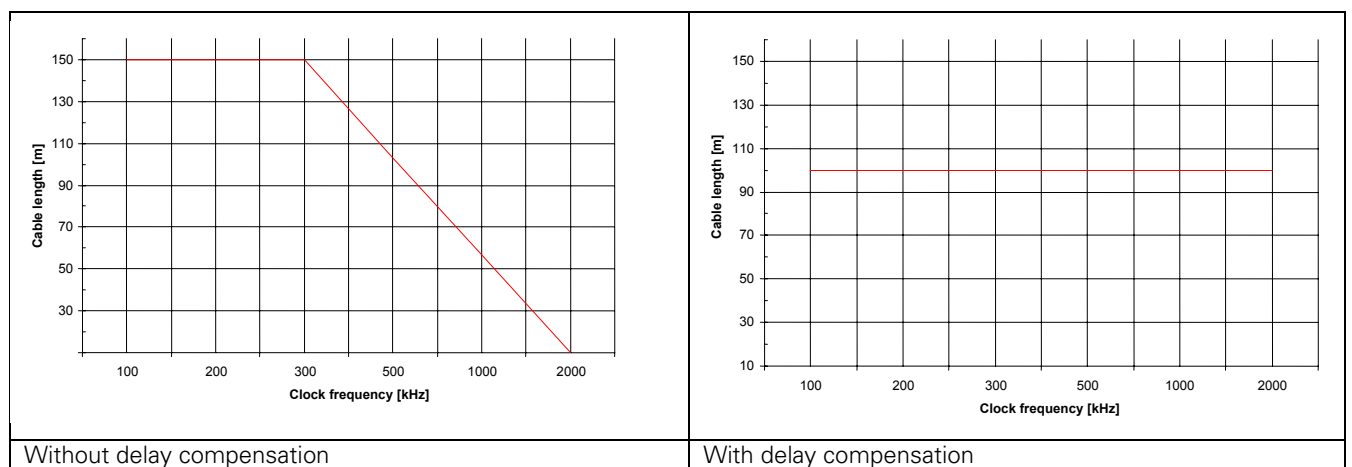
**EnDat Clock Frequency:**

The EnDat clock frequency can be set-up by a software command. The clock frequency can be set at specific intervals between 100 kHz and 2 MHz. The maximum permitted frequency is dependent on both the cable length between encoder and EIB 741 and also on whether a delay compensation is activated or not.

Clock Frequency Parameters	Clock Frequency	Comment
100000	100 kHz	
300000	300 kHz	Default with EnDat 2.1
500000	500 kHz	
1000000	1 MHz	
2000000	2 MHz	Default with EnDat 2.2

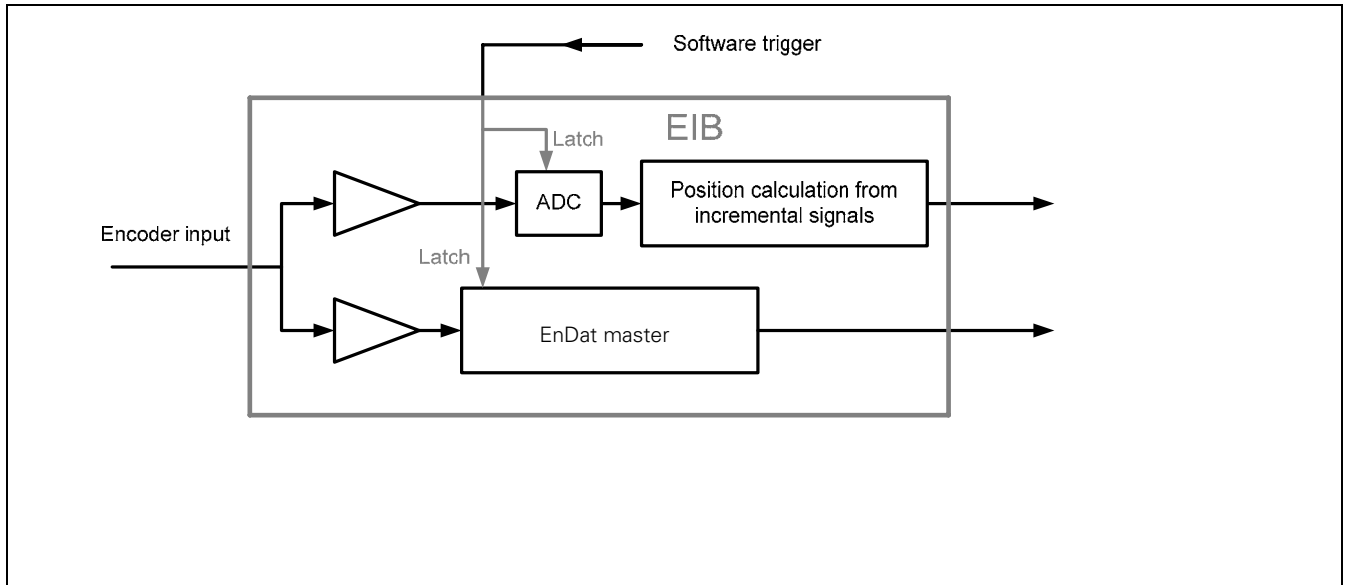
**Delay Compensation:**

Delay compensation for the EnDat transmission can be activated or deactivated during axis configuration. With a few exceptions, delay compensation is not released by HEIDENHAIN for EnDat 2.1 encoders. Delay compensation is released by HEIDENHAIN for EnDat 2.2 encoders. This gives rise to the following dependency on the maximum permitted EnDat clock frequency.



**Processing Additional Incremental Signals with EnDat:**

If, with EnDat encoders, the incremental signals are used for position generation, an absolute reference can be created by latching EnDat and incremental position simultaneously. To do this, a special command is sent via the customer software application to the EIB 741, which then generates an internal trigger signal. This trigger signal initiates simultaneous position capture via the EnDat interface and via the incremental signals. Both positions are transmitted to the customer software application as a return value.



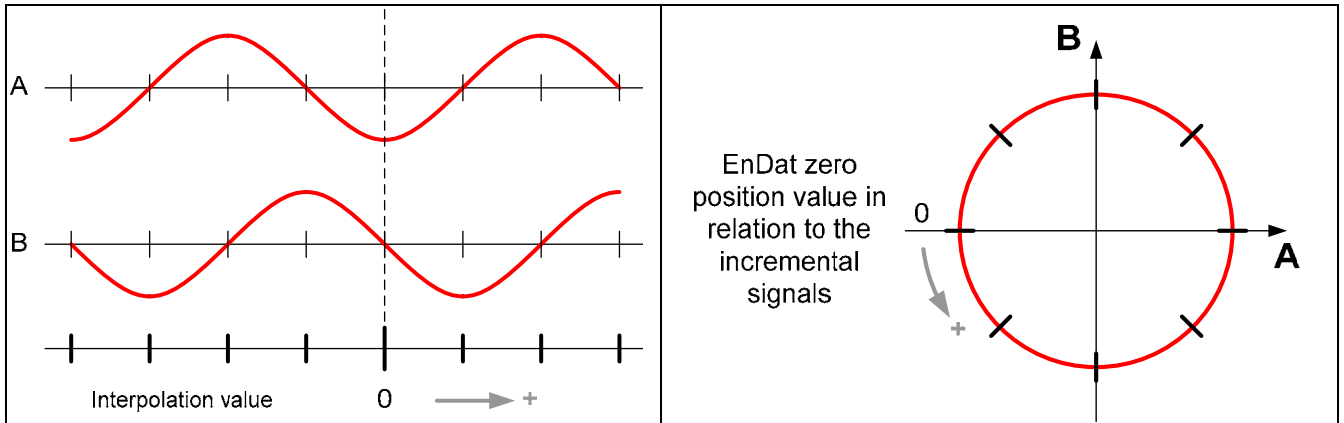
**Please note:**

The interpolation zero points 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: see "Processing incremental signals" section for interpolation zero point

EnDat Position: see graphic for interpolation zero point



### 3 Processing Trigger Events

Position value capturing inside the EIB 741 is initiated via a so-called trigger event. The EIB 741 supports the following trigger sources:

- External trigger input 1
- Internal periodic trigger source, time-controlled
- Software command

Only one of the aforementioned trigger sources can be active at the same time. The trigger source must be set by a software command.

Not all trigger interface options are supported in all operating modes; see "Operating modes" section for details.

#### 3.1 Trigger Inputs and Outputs

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

##### Trigger Inputs:

Trigger inputs synchronize the position polls on external events.

Only trigger input 1 is currently supported. This triggers position value generation simultaneously for all four channels.

The 120 ohm termination resistor can be activated or deactivated by configuration

##### Trigger Outputs:

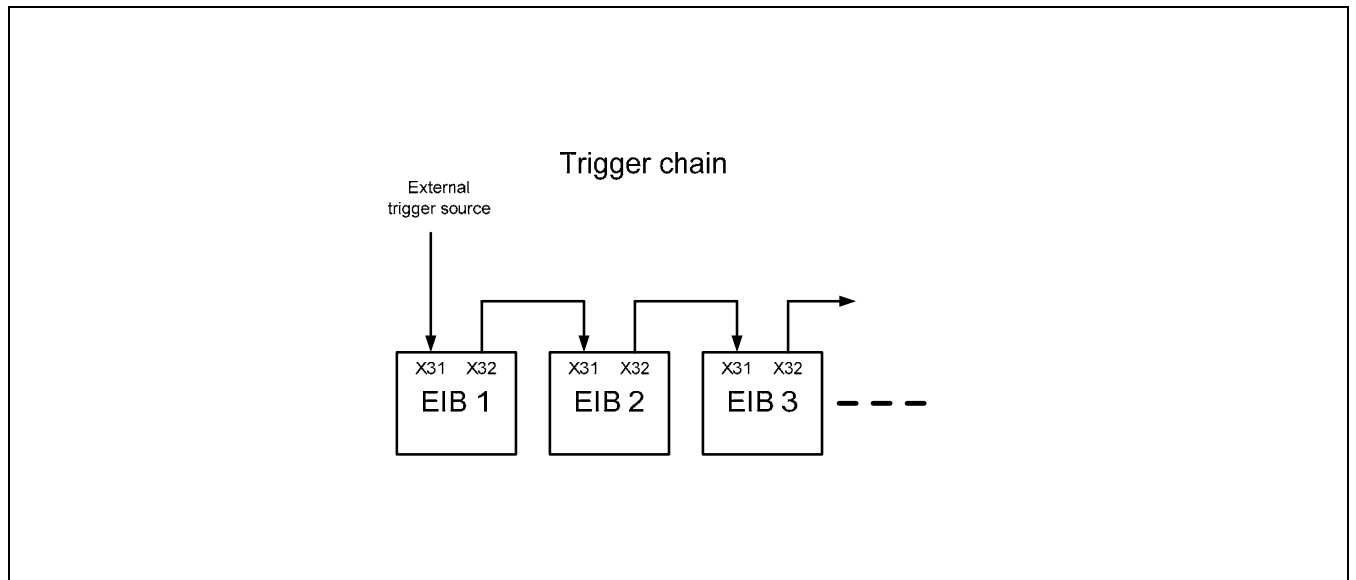
Trigger outputs can be used to forward trigger events, e.g. to other EIB 741 units. This enables the generation of a trigger chain, which synchronizes EIB 741 units with an external trigger event. During this process, the various EIB 741 units must be configured separately by software commands. The position data is sent via the respective Ethernet connection. To generate a trigger chain, the following connection between the EIB 741 units must be used:

- Trigger Out + → Trigger In +
- Trigger Out - → Trigger In -
- GND to GND

Currently, only trigger output 1 is supported.

##### Please note:

Trigger events that have been initiated by a software command or time control cannot be generated on the trigger outputs.



##### Configuring the Trigger Inputs and Outputs as Logical Inputs and Outputs:

The trigger inputs and outputs can also be used as logical inputs and outputs. Trigger inputs and/or outputs are selected by default. The ports can be individually configured as logical inputs and outputs or as trigger inputs and outputs by a software command. They cannot be used as trigger and/or logical inputs or outputs at the same time.

### 3.2 Logical Inputs and Outputs

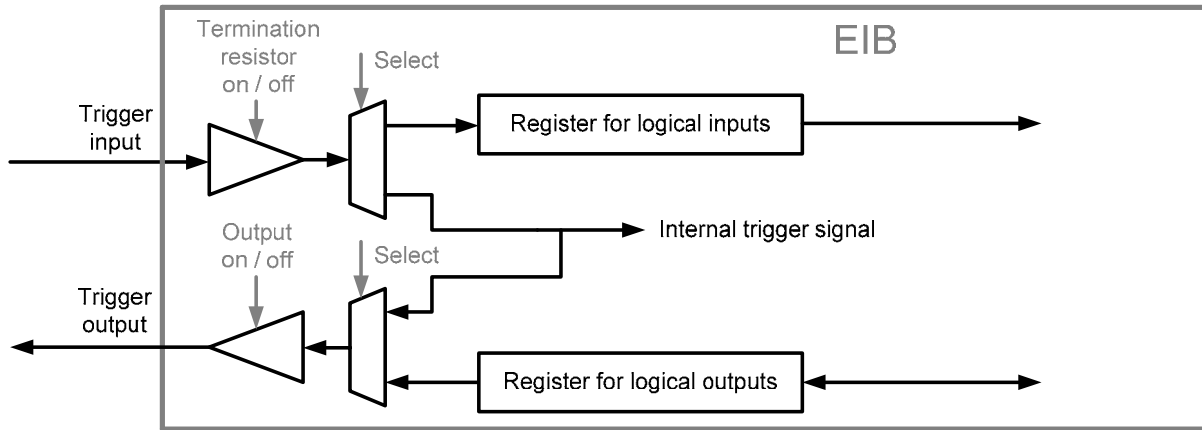
#### Logical Inputs:

Each trigger input can be individually converted to a logical input. The level of the corresponding input can be read by a software command. The 120 ohm termination 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 inputs and/or outputs at a glance. Only channel 1 is shown.



### 3.3 Maximum Trigger Rate

The maximum trigger rate of the EIB 741 depends on the set operating mode ("polling" mode being an exception):

- Soft Realtime Mode: max. 10 kHz

A certain interval, which the EIB 741 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, trigger events cannot be accepted by the EIB 741 and so are lost (lost trigger). This is detected by the EIB 741 and displayed in the status application word of the position data packet by the "lost trigger" bit. This bit is set at "1" until actively reset by the customer software application using a clear command.

### 3.4 Counter for Accepted Trigger Events

In addition to lost trigger monitoring, the EIB 741 has, for further error detection, a counter that is incremented by each incoming and accepted trigger event. A trigger event is accepted if the aforementioned interval is respected. Trigger events that result in lost triggers are not counted. The counter value is transferred in the position data packet and can be monitored for continuity. It is then possible to monitor whether position data packets are lost.

## 4 Timestamp

The "timestamp" function is also used to monitor 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 capture also causes the current time value to be saved to the timestamp register. When the timestamp function is activated, the contents of this register are transmitted 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 the case of applications that do not have a periodic trigger, the time of the trigger event can be transmitted with this register.

### Please note:

The time interval of the counter timestamp is a multiple of the internal system clock in the EIB 741. Before the timestamp function can be used, the time interval has to be set by a software command. The "clock ticks per  $\mu\text{s}$ " value must first be read and the required time interval set in dependence of this value. This is necessary to keep the software compatibility independent of various settings for the system clock.

## 5 Status Word

The status word must be interpreted, irrespective of the poll type:

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

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

Bit no.	Incremental Position	EnDat Position	EnDat Additional Information
0	1 = Valid position	1 = Valid position	1 = Valid additional information
1	1 = Signal amplitude error	1 = CRC error	1 = CRC error
2	Reserved	Reserved	Reserved
3	1 = Frequency exceeded	Reserved	Reserved
4	1 = Encoder power supply error	1 = Encoder power supply error	Reserved
5	1 = Fan error	1 = Fan error	Contents I0
6	Reserved	Reserved	Contents I1
7	1 = Lost Trigger	1 = Lost Trigger	Contents I2
8	1 = Reference position 1 latched	1 = EnDat error message 1	Contents I3
9	1 = Reference position 2 latched	1 = EnDat error message 2	Contents I4
10	1 = Coded reference value for distance-coded reference marks is valid	Reserved	EnDat Busy Bit
11	1 = Error when calculating the coded reference value for distance-coded reference marks	Reserved	EnDat RM Bit
12	Reserved	Reserved	EnDat WRN Bit
13	Reserved	Reserved	Reserved
14	Reserved	Reserved	Reserved
15	Reserved	Reserved	Reserved

## Notes on the Error Bits:

Name	Meaning
Valid position	1 → No error occurred This bit indicates whether the transmitted position is valid or not
Valid additional information	1 → EnDat additional information was received Otherwise, no additional information has been selected or received
Signal amplitude error	1 → Signal amplitude of 1 V <sub>PP</sub> incremental signals is and/or was too low (once or repeatedly since this error message was last cleared)
Frequency exceeded	1 → Excess input signal frequency has been detected (once or repeatedly since this error message was last cleared)
CRC error	1 → CRC error during EnDat data transmission
Encoder power supply error	1 → Power supply to the encoder was switched off automatically. (overcurrent fuse has tripped)
Fan error	1 → The EIB 741 fan has malfunctioned
Lost Trigger	See "Maximum trigger rate" section
Reference position 1 latched	1 → Reference position 1 was latched (since the last corresponding software command)
Reference position 2 latched	1 → Reference position 2 was latched (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 when calculating the coded reference value for distance-coded reference marks	1 → Error with calculating the coded reference value; must be reset explicitly
EnDat error message 1	1 → Error message 1 active
EnDat error message 2	1 → Error message 2 active
EnDat Busy bit	1 → Busy bit is set
EnDat RM bit	1 → RM (reference mark) bit is set
EnDat WRN bit	1 → WRN (warning) bit is set
Contents I0..I4	These five bits define the contents of the received additional information. The customer software application requires this information to interpret the data.

The error bits are not reset automatically. They have to be reset by a software command from a customer software application. If an error is not reset, it will be re-transmitted with every subsequent position data packet.

With incremental encoders, an error in the position data packet indicates that the position is no longer valid and has lost any reference to reference marks or other measuring channels.

The occurrence of one error can initiate others. An error in the power supply to the encoder will bring about other errors. Any error in the power supply must therefore be reset first. Once the power supply is stable (delay of approx. 1.5 seconds), the other errors must be reset.

**Lost Trigger:**

The "lost trigger" bit indicates that at least one trigger event was processed incorrectly due to the period between two trigger events being too short. The "lost trigger" bit can also occur if malfunctions superimpose the trigger line or EMC influences have a negative effect on the transmission. A "lost trigger" does not mean that the position values are false. It merely indicates that trigger events were unable to be processed correctly. The reset must also be done actively by a software command.

**Reference Position Latched**

The two "reference position 1 (2) latched" bits indicate that a valid reference mark has been detected and latched. This means 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 latching reference positions. This bit is set actively when the coded reference value is calculated successfully. This means that the "coded reference value with distance-coded reference marks" in the position data packet can be used to calculate the absolute position.

**Error with 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 is being calculated. One possible reason is that during the referencing phase, a change of direction has occurred causing the same reference mark to be detected twice. The error must be actively reset. It is not reset automatically when the software command for saving reference positions is resent.

**Fan Error:**

This bit indicates whether the fan of the EIB 741 is functioning correctly or not. The error bit does not influence the position data.

**Please note:**

For additional information, see "Installation Instructions".

If the fan monitoring function is not supported, this bit is always set to "0".

## 6 Ethernet Interface

The Ethernet (LAN) interface is used for configuring the EIB 741 and for transferring the position data packets. TCP communication is used for the configuration and UDP communication is used for transferring the position data packets. The EIB 741 network settings can be changed by a software command. Constant values or DHCP can be used to set-up the IP address. For additional details, see "Installation Instructions".

## 7 Operating Modes

The EIB 741 supports the following operating modes:

- Soft Realtime
- Polling

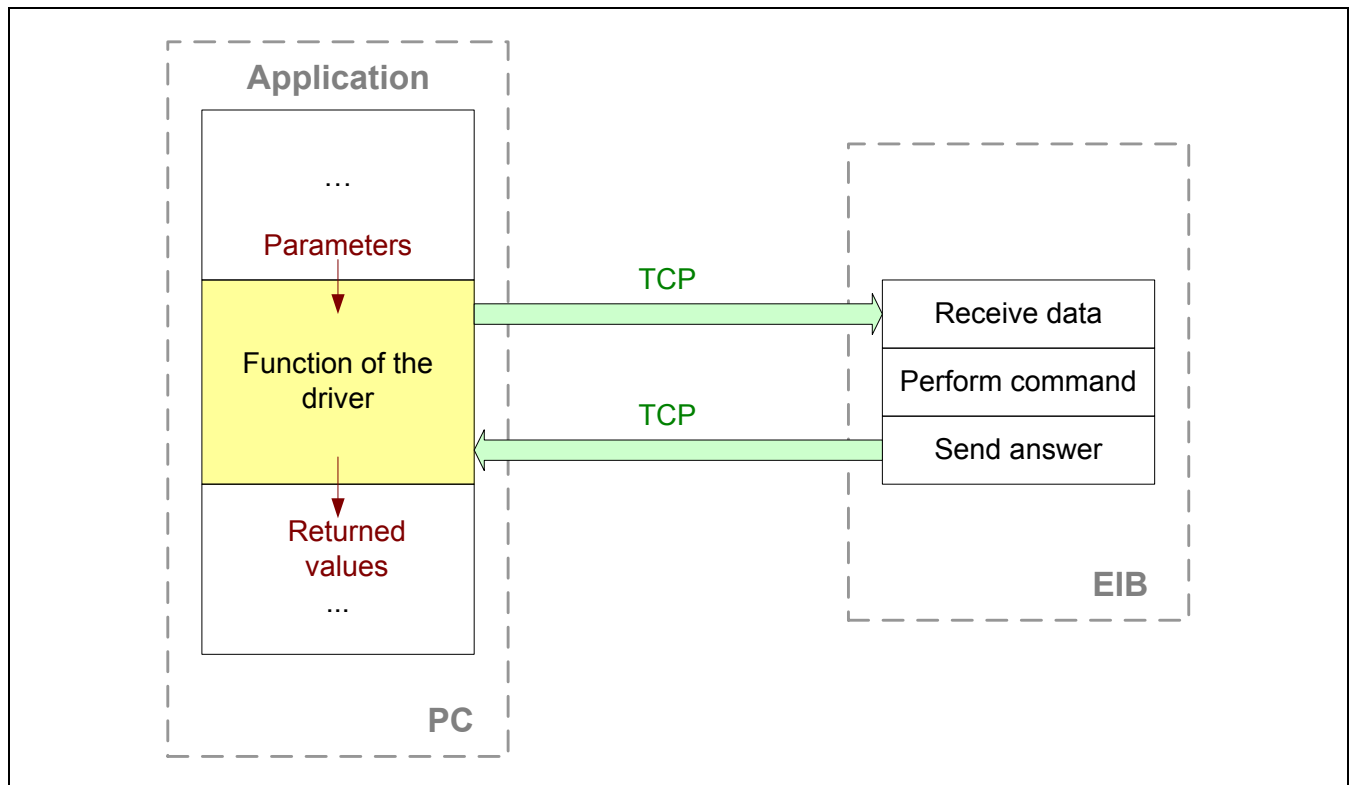
### 7.1 "Polling" Mode

This operating mode is activated by default after the EIB 741 is initialized. The position data is calculated in the EIB 741 as soon as a corresponding command is received. The EIB 741 transmits the data inside the response packet to the customer application.

The diagram below illustrates the sequence of a position poll. A command is sent to the EIB 741 from a customer software application on the PC. The EIB 741 generates the position data and returns it in a TCP packet. The data is transmitted to the application.

Processing trigger events:

- The time of position value generation is influenced by the software and cannot therefore be determined exactly.
- External trigger inputs are not supported
- Internal periodic trigger sources are not supported



### Data Packets in "Polling" Mode:

Dependent upon the selected function; see Function calls section

## 7.2 "Soft Realtime" Mode

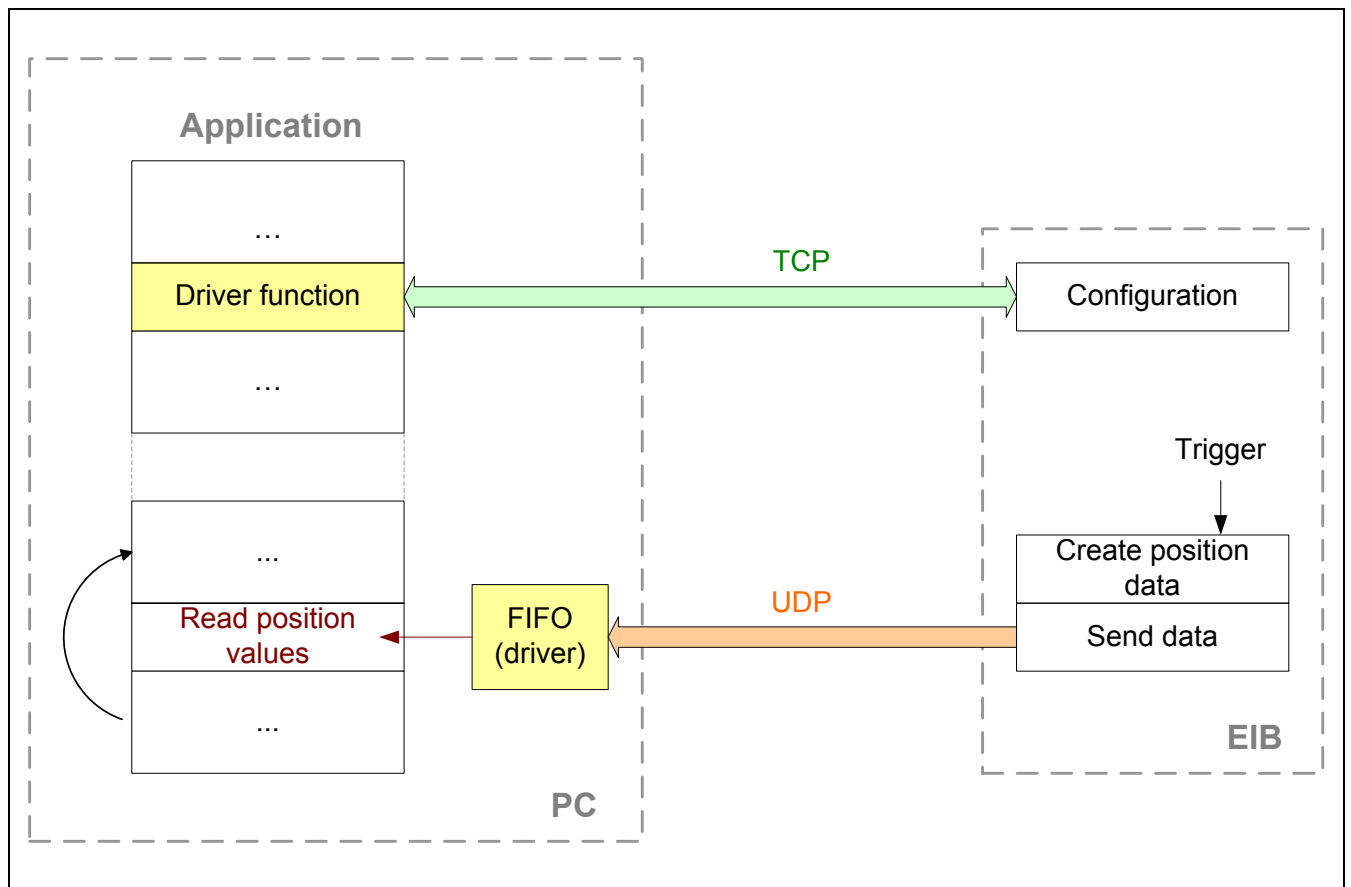
The position data is transported with UDP packets from the EIB 741 to the PC. This occurs parallel to the TCP communication via the standard Ethernet interface. The position data is generated when the EIB 741 receives a trigger signal. With each trigger event, a data packet is sent to the PC automatically. Here, the packets can be read from a FIFO data structure.

For operation in soft realtime mode, the EIB 741 must be configured following the steps listed below.

- Initialization of the EIB 741
- Initialization and configuration of the axes
- Selection of the operating mode (soft realtime)
- Activation of the trigger source

The diagram below illustrates the communication process schematically. The customer software application has to configure the EIB 741. The data is then transmitted to the FIFO data structure independently. From here, the application can read out the data within a program loop.

Parallel to the position poll, the status of the EIB 741 can be called or error messages cleared.



If the Ethernet connection is terminated in "soft realtime" mode, e.g. by unplugging the Ethernet cable, the EIB 741 deactivates the trigger source and sends no further UDP packets. Once the connection has been restored, the EIB 741 must be reconfigured for the operating mode.

When the application is closed, the aforementioned steps must be carried out in the reverse order. The trigger source must be deactivated first. The operating mode can then be changed or the connection to the EIB 741 closed.

Processing trigger events:

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

**Position Data Packets in "Soft Realtime" Mode:**

Position data packet:

<b>Contents</b>	<b>Encoder Channel No.</b>	<b>Data Width (bytes)</b>	
Trigger counter	-	2	
Status word	1	2	
Position value		6	
Timestamp		4	
Position 1 reference		6	
Position 2 reference		6	
Coded reference value with distance-coded reference marks		6	
Amplitude value, incremental signal A		2	
Amplitude value, incremental signal B		2	
Status word		2	2
Position value			6
Timestamp	4		
Position 1 reference	6		
Position 2 reference	6		
Coded reference value with distance-coded reference marks	6		
Amplitude value, incremental signal A	2		
Amplitude value, incremental signal B	2		
Status word	3	2	
Position value		6	
Timestamp		4	
Position 1 reference		6	
Position 2 reference		6	
Coded reference value with distance-coded reference marks		6	
Amplitude value, incremental signal A		2	
Amplitude value, incremental signal B	2		
Status word	4	2	
Position value		6	
Timestamp		4	
Position 1 reference		6	
Position 2 reference		6	
Coded reference value with distance-coded reference marks		6	
Amplitude value, incremental signal A		2	
Amplitude value, incremental signal B		2	

This table shows only the data contents of the UDP packet. All values displayed above are sent in a UDP packet and copied to the FIFO of the driver.

## 8 Firmware Update

The EIB 741 firmware can be updated by the user with a TFTP client. However, only special update files from HEIDENHAIN may be installed on the EIB 741.

The example below is based on a firmware update from a computer with "Windows" as its operating system. The EIB 741 must be connected to the computer via Ethernet. In this example, the file name for the update is "update\_633281-06.flash". This file is saved under "C:\temp\EIB".

- Run the Windows command line
- Save the update file under "C:\temp\EIB\update\_633281-06.flash"
- Start the TFTP file transfer:  
> tftp -i 192.168.1.2 put C:\temp\EIB\update\_633281-06.flash  
tmp\update.flash

Option -i:	activates the "binary file transfer"
IP address:	"192.168.1.2" (default setting) or customer-specific setting
"Put" command:	transfer from the host to the EIB 741
Source file:	in this example "C:\temp\EIB\update_633281-06.flash"
Target file:	always "tmp\update.flash"

If the file is transferred successfully, the TFTP client displays a corresponding message in the command line. The status LED of the EIB 741 is switched off. After the internal data transmission to the flash memory, the status LED is 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 and commands must not be sent to the EIB 741 via the Ethernet interface.

After the status LED is reactivated, the relevant software command must be used to query whether the update was completed successfully. The status of the update process can be polled up until the EIB 741 is booted again.

The EIB 741 boots the new version of the firmware at the next reset.

In case of an error during the firmware update, the corresponding settings shown in the "Resetting the EIB 741" table (see "Installation Instructions") are booted.

## 9 Reset

See "Installation Instructions"

# Part 2: Driver Software

## 1 General Information

Functions are provided for accessing the EIB 741 from a software application. This group of functions is supplied as DLL for Windows systems and as SO library for Linux. The following operating systems are supported:

- Windows 2000, Windows XP, Windows Vista
- Linux/Unix with kernel 2.6, (i386 systems)

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, the library must be incorporated into the project.

## 2 Installation Instructions

The stated directories and files refer to the driver CD for the EIB 741.

### 2.1 Windows

For an application to load the DLL, file "eib7.dll" must be copied from the "EIB\_741\windows\bin" directory to the Windows system directory (e.g. "C:\Windows\system32"). Alternatively, the path for the DLL can be defined in the system. The DLL interface is defined via the two files "eib7.lib" in "EIB\_741\windows\lib" and "eib7.h" in "EIB\_741\windows\include". These must be incorporated into the software project in the development environment (for C/C++ environments). File "eib7.lib" must be copied into the library directory of the development environment or its path entered.

### 2.2 Linux

For an application to load the SO library, the file "libeib7.so" on the CD must be copied from the "EIB\_741/linux/lib" directory to the "usr/local/lib" directory. The library interface is defined via file "eib7.h" in "EIB\_741/linux/include". This must be copied to "usr/local/include" and must be incorporated into the software project in the development environment. The stated directories are based on the "Filesystem Hierarchy Standard" for Linux operating systems. The "libeib7.so" library was compiled for i386 system under kernel 2.6.

## 3 Overview

### 3.1 Establishing Communication

To communicate with the EIB 741, a connection must first be established using the EIB7Open() function. It is sometimes necessary to determine the IP address beforehand using EIB7GetHostIP(). The EIB 741 can then be configured using the device functions.

Access to the axes requires handles, which are generated by the EIB7GetAxis() function. This is also the case for the IO ports, whose handles are generated by the EIB7GetIO() function. The handles can be used for configuration and for polling the status.

At the end of communication, the connection must be closed using the EIB7Close() function.

### 3.2 Polling Mode

The axis functions can be used to access the encoders. First, the axis must be configured via EIB7InitAxis(). The position values can then be read or error messages acknowledged.

It is not necessary to select a trigger source. Triggering occurs implicitly when the EIB7GetPosition() function is executed.

### 3.3 Soft Realtime Mode

The axes must be configured with EIB7InitAxis() before soft realtime mode can be activated. In soft realtime mode, only the error messages from the status word for the position values can be reset.

Once soft realtime mode is activated, a trigger source can be selected. The customer software application on the host must continuously read the position data from the receive buffer to prevent an overflow. This can be done using functions EIB7ReadFIFOData() or EIB7ReadFIFODataRaw() (see 7.33, 7.36). Each of these functions reads one or several entries from the FIFO. Each entry contains data for all axes of the EIB 741. The size of an entry can be determined in advance using functions EIB7GetSizeOfFIFOEntry() and EIB7GetSizeOfFIFOEntryRaw() (see 7.34, 7.37). The individual components of a FIFO entry can be accessed using the functions EIB7GetDataFieldPtr() or EIB7GetDataFieldPtrRaw().

It is also possible, using the callback mechanism, to register a function that is executed as soon as new data is available in the FIFO (see 7.43).

## 4 Data Types

### 4.1 Simple Data Types

EIB7_HANDLE	Handle for an EIB 741
EIB7_AXIS	Handle for an axis on the EIB 741
EIB7_IO	Handle for an input or export port on the EIB 741
EIB7_ERR	Error message
ENCODER_POSITION	Position value (64-bit integer)

### 4.2 EnDat Additional Information

```
struct ENDAT_ADDINFO
```

Component	Description
status	Status word for the additional information
info	Data of the additional information

### 4.3 Information for TCP Connection

```
struct EIB7_CONN_INFO
```

Component	Description
id	Identification number for the connection
local_ip	Local IP address for this connection
local_port	Local port number for this connection
remote_ip	IP address of the EIB 741 for this connection
remote_port	Port number of the EIB 741 for this connection

## 5 Parameters and Return Values

All functions supply a return value of the type EIB7\_ERR. This labels a function call as successful or reports an error that occurred during execution.

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

## 6 Auxiliary Functions

### 6.1 Determining IP Address

The hostname of the EIB 741 or the IP address (as C-string) is converted to an IP address in "Host Byte Order". The name must be transmitted as a C-string. Examples are "192.168.1.2" or "EIB741-SN1234567".

#### Function

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

#### Parameters

Hostname                    Pointer to a C-string containing the IP address or the hostname of the EIB 741.  
ip                            *[return value]* Pointer to a variable to which the IP address of the EIB 741 is saved

#### Return Value

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

EIB7_NoError	Function call successful
EIB7_HostNotFound	IP address was unable to be determined

## 6.2 Changing Format of Position Data

The data format of a position value is converted from 64 Bit Integer data type into Double data type. The function can be used only for incremental measuring systems. The converted value is given in "Signal Periods". The period counter value is mapped onto the integer part of the resulting value and the interpolation value is mapped onto the fractional part.

### Function

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

### Parameter

src	Position value of an incremental measuring system
ip	<i>[return value]</i> Pointer to a variable for the converted position value

### Return Value

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

EIB7_NoError	Function call successful
EIB7_ParamInvalid	Position value is invalid

## 7 Device Functions

The device functions always refer to the entire EIB 741. No distinction between the axes is possible. With some functions, parameters of all axes are influenced.

All device functions are able to deliver the following error messages as a return value. They can also return further values individually. These are listed separately for each function.

### Standard Return Values

EIB7_NoError	Function call successful
EIB7_InvalidHandle	The handle on the EIB 741 is invalid
EIB7_FuncNotSupp	Function is not supported by the EIB 741
EIB7_InvalidResponse	Error during data transmission
EIB7_AccNotAllowed	Function cannot be performed, as the EIB 741 does not permit access
EIB7_ConnReset	Connection terminated by the EIB 741
EIB7_ConnTimeout	Timeout during data transmission to the EIB 741
EIB7_ReceiveError	Error whilst receiving the data
EIB7_SendError	Error whilst sending the data
EIB7_OutOfMemory	The system is unable to allocate sufficient memory

### 7.1 Opening Connection to the EIB 741

A TCP connection is established with the EIB 741. In doing so, no settings are changed in the EIB 741. If the connection cannot be established, then an error message is returned. The driver must be compatible with the EIB 741 firmware to function correctly. This is verified once the connection is established. If necessary, the EIB 741 firmware version can be read using this function. To do this, the "ident" parameter must be used to transmit 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	<i>[return value]</i> Handle for the EIB 741 if the function was closed successfully
timeout	Timeout for subsequent commands in milliseconds (not valid for EIB7Open())
ident	<i>[return value]</i> Pointer to the target memory in which the firmware version of the EIB 741 is saved. This memory must be at least 9 bytes. If this parameter is a NULL pointer, the firmware version of the EIB 741 is not read.
len	Size of the target memory in bytes (0, where ident = NULL)

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_CantInitWinSock	Socket layer of the operating system cannot be initialized (applies only for Windows)
EIB7_CantOpenSocket	System resources for connection not available
EIB7_OutOfMemory	Insufficient memory available
EIB7_IFVersionInv	EIB 741 firmware is incompatible with the driver
EIB7_CantConnect	Connection cannot be established (EIB 741 may have been switched off or is unavailable)

## 7.2 Closing Connection to the EIB 741

The connection to the EIB 741 is closed. The EIB handle must not be used again. Likewise, all handles on the axes generated from this EIB handle are invalid. If a special operating mode of the EIB 741 has been activated via this handle, polling mode will be activated when the connection is closed. All other settings in the EIB 741 are retained.

### Function

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

### Parameters

eib                    EIB handle

### Rückgabewert

The return value delivers a status for the function call. All potential values are listed for the standard return values.

## 7.3 Polling Connection Status

The status of the connection to the EIB 741 is polled. Whether a connection has already been closed or a communication error has occurred can then be determined. This function does not send data to the EIB 741. 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 741 established
EIB7_CS_Closed	No connection to the EIB 741
EIB7_CS_Timeout	Time exceeded during data transmission
EIB7_CS_ConnectionReset	The connection has been closed by the EIB 741
EIB7_CS_TransmissionError	Transmission error occurred

### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

## 7.4 Setting up Timeout

The timeout for the TCP connection to the EIB 741 is reset. This value applies for 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_IllegalParameter    The timeout cannot be set

## 7.5 Requesting Handle for Axis

The handles for access to the axes of the EIB 741 are generated. They are saved in an array, the size of which must also be transferred as a parameter. The number of valid handles is delivered as a return value. For each axis of the EIB 741, the function delivers one handle, the maximum being the number accommodated in 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	<i>[return value]</i> Pointer to the first element of the handle array
size	Maximum number of entries in the array
len	<i>[return value]</i> Number of valid entries in the array

### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

## 7.6 Requesting IO Port Handle

Handles are generated for access to the IO ports of the EIB 741. The handles for the inputs and the outputs are each saved in an array, the size of which must also be transmitted as a parameter. The number of valid handles in the array is generated in "ilen" or "olen". For each IO port of the EIB 741, the function delivers one handle, the maximum being the number accommodated in the array ("isize", "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	<i>[return value]</i> Pointer to the first element of the array with the input handles
isize	Maximum number of entries in the "iset" array
ilen	<i>[return value]</i> Number of valid entries in the "iset" array
oset	<i>[return value]</i> Pointer to the first element of the array with the output handles
osize	Maximum number of entries in the "oset" array
olen	<i>[return value]</i> Number of valid entries in the "oset" array

### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

## 7.7 Selecting Operating Mode

The operating mode for the EIB 741 can be set. Both "Polling" and "Soft Realtime" modes are supported.

### Function

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

### Parameters

eib                            EIB handle  
mode                            Operating mode

Mode	Operating Mode
EIB7_OM_Polling	Polling mode
EIB7_OM_SoftRealtime	Soft Realtime mode

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_CantOpenSocket            Internal error (socket error)  
EIB7\_CantStartThread          Internal error (thread error)  
EIB7\_SoftRTEn                 Soft Realtime mode is already activated  
EIB7\_PortNoInv                Internal error (UDP port error)  
EIB7\_DestIPUnreach            Internal error (IP address error)

## 7.8 Saving Network Parameters

The parameters for the Ethernet interface of the EIB 741 can be set. This means the EIB 741 can be adapted to the network. The settings do not take effect until after the next boot process. If the DHCP client is active, the EIB 741 tries to obtain an IP address from the DHCP server. If the server fails to respond within the set 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 741 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 741

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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_CantSaveCustNW            Network settings cannot be stored  
EIB7\_CantSaveDHCP              DHCP Timeout cannot be saved  
EIB7\_DHCPTimeoutInv            DHCP Timeout invalid  
EIB7\_ParamInvalid               Parameters are not a valid network configuration

## 7.9 Reading Network Parameters

The parameters for the Ethernet interface can be read. The user-defined settings are always displayed, even if standard settings are 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 the flag for the DHCP client

dhcp	Description
EIB7_MD_Disable	DHCP client inactive
EIB7_MD_Enable	DHCP client active

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_NoCustNetwork    No customized settings present

## 7.10 Saving Hostname

The hostname of the EIB 741 is saved. The name must be transmitted as a C-string, which can be a maximum of 32 characters long including the zero byte. If it is any longer, the rest will be cut off. If a string with a length of zero or a NULL pointer is transferred, the EIB 741 sets the hostname to the standard value on delivery.

### Function

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

### Parameters

eib                    EIB handle  
hostname              Pointer to the new hostname

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_HostnameTooLong    Hostname is too long  
EIB7\_HostnameInvalid    Hostname is invalid  
EIB7\_CantSaveHostn      Hostname cannot be saved  
EIB7\_CantRestDefHn      Standard hostname cannot be loaded

### 7.11 Reading Hostname

The hostname of the EIB 741 is read and saved in the target memory as a C-string. The string can be a maximum of 32 characters long (incl. zero byte). If the target memory is not big enough to take the entire string, only the first part is copied.

#### Function

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

#### Parameters

eib	EIB handle
hostname	<i>[return code]</i> Pointer to the target memory for the hostname
len	Size of the target memory in bytes

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_CantRdHostname    Hostname cannot be read

### 7.12 Reading Serial Number

The serial number of the EIB 741 is displayed as a C-string. The string is written to the target memory. If the target string does not provide sufficient space for the serial number, an error will be generated. The serial number can be up to 24 characters long (incl. zero byte).

#### Function

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

#### Parameters

eib	EIB handle
serial	<i>[return value]</i> Pointer to the target memory for the serial number
len	Size of the target memory in bytes

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_CantRdSer                Serial number cannot be read  
EIB7\_BufferTooSmall        Target memory is too small

### 7.13 Reading the Device ID

The device ID of the EIB 741 is generated as a C-string. The string is written to the target memory. If the target string does not provide sufficient space for the number, an error will be generated. The number can be up to 16 characters long (incl. zero byte).

#### Function

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

#### Parameters

eib	EIB handle
ident <i>[return value]</i>	Pointer to the target memory for the device number
len	Size of the target memory in bytes

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_CantRdIdent             Device number cannot be read  
EIB7\_BufferTooSmall        Target memory is too small

### 7.14 Reading MAC Address

The MAC address of the EIB 741 is displayed. The address is returned in binary format. The target memory must be at least 6 bytes. The first six bytes are always used. The lowest-value byte of the MAC address is copied to the first byte of the target memory. For example for "00:A0:CD:85:00:01".

Offset	Memory Contents
0	0x01
1	0x00
2	0x85
3	0xCD
4	0xA0
5	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 delivers a status for the function call. All potential values are listed for the standard return values.

### 7.15 Reading Firmware Version Number

The version number of the EIB 741 firmware is read. The "select" parameter determines the firmware from which the version number is generated as a C-string. For the string, including the zero 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 is read

Select	Description
EIB7_FW_CurrentlyBooted	Firmware currently loaded
EIB7_FW_Factory	Firmware on delivery
EIB7_FW_User	Firmware of the last update

#### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

## 7.16 Reading Boot Mode

The boot mode in which the EIB 741 was started during the last boot process is read.

### Function

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

### Parameters

eib                            EIB handle  
 status                        *[return value]* Pointer to the variable for boot mode

Mode	Description
EIB7_BM_User	Firmware of the last update with user's network settings
EIB7_BM_FactoryUser	Firmware of factory default settings with user's network settings
EIB7_BM_FactoryDefault	Firmware of factory default settings with standard network settings

### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

## 7.17 Reading Update Status

The status can be read to verify whether an update was completed successfully. After the function call the update status value is automatically reset to the default value (EIB7\_US\_NoUpdate). The status information is cleared during 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

Status	Description
EIB7_US_NoUpdate	No update loaded
EIB7_US_UpdateFailed	Update unable to be performed
EIB7_US_UpdateSuccessful	Update performed successfully
EIB7_US_VersionIncompatible	Firmware is incompatible with the EIB 741 hardware

### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

## 7.18 Reading Number of Open Connections

The number of currently open connections to the EIB 741 is returned. This also includes half-open functions that the remote station has already closed but are still open on the EIB 741.

### 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 delivers a status for the function call. All potential values are listed for the standard return values.

### 7.19 Reading Connection Data

The connection data of all currently open connections to the EIB 741 can be read. For each connection, an entry is assigned to the array. The maximum being, however, the number specified by the "size" parameter. The number of valid elements in the array is returned by the "cnt" parameter. The contents of the connection data is listed in the "Data types" section.

#### Function

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

#### Parameters

eib	EIB handle
info	<i>[return value]</i> Pointer to the first element in the array for the connection data
size	Size of the "info" array
cnt	<i>[return value]</i> Pointer to the variable for the number of valid elements in the array

#### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

### 7.20 Terminating the Connection

An open connection to the EIB 741 can be terminated. It is not possible to close the connection used to call the function. The primary use of this function is to close half-open connections that have not been terminated properly due, for example, to an error on the host. The ID can be taken from connection data EIB7\_CONN\_INFO (see "Read connection data").

#### Function

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

#### Parameters

eib	EIB handle
id	ID of the terminated connection

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

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

### 7.21 Reading the Time Unit for Timestamp

The timestamp counter is supplied from a clock source. The timestamp ticks indicate how many cycles per microsecond are generated from the clock source.

#### Function

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

#### Parameters

eib	EIB handle
ticks	<i>[return value]</i> Pointer to the variable for the number of cycles per microsecond

#### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

### 7.22 Setting the Timestamp Period

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

#### Function

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

#### Parameters

eib	EIB handle
period	Ticks per timestamp period (>0)

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_ParamInvalid	Timestamp period invalid
-------------------	--------------------------

### 7.23 Resetting the Timestamp Counter

The timestamp counter is set to zero and counts on from this value.

#### Function

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

#### Parameters

eib	EIB handle
-----	------------

#### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

### 7.24 Reading the Time Unit for Timer Trigger

The timer trigger is supplied from a clock source. The time trigger ticks indicate how many clock cycles per microsecond are generated from the clock source.

#### Function

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

#### Parameters

eib	EIB handle
ticks	<i>[return value]</i> Pointer to the variable for the number of clock cycles per microsecond

#### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

### 7.25 Setting the Timer Trigger Period

The period of the timer trigger can be set. To do this, the number of timer trigger ticks making up a period must be set. 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.

#### Function

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

#### Parameters

eib                                    EIB handle  
period                                Ticks per timer trigger period (>0)

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_ParamInvalid                    Trigger period invalid

### 7.26 Activating the Timer Trigger

The timer trigger can be activated or deactivated. The external trigger and the timer trigger cannot be operated at the same time. If the external trigger signal has already been enabled, the EIB 741 delivers an error message as soon as an attempt is made to activate the timer trigger.

#### Function

```
EIB7_ERR EIB7EnableTimerTrigger ( EIB7_HANDLE eib,
                                   EIB7_MODE   mode
                                   )
```

#### Parameters

eib                                    EIB handle  
mode                                 Activate or deactivate the timer trigger

Mode	Description
EIB7_MD_Disable	Deactivate the timer trigger
EIB7_MD_Enable	Activate the timer trigger

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_ExtTrgEn                        External trigger is already active

### 7.27 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 delivers a status for the function call. All potential values are listed for the standard return values.

## 7.28 Setting the Termination Resistors

The termination resistors for the incremental signals of the encoder inputs can be deactivated. This setting always applies for all 1Vpp inputs on the EIB 741. The resistors are activated after every boot process of the EIB 741.

### Function

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

### Parameters

eib EIB handle  
mode Activate or deactivate termination resistors

Mode	Description
EIB7_MD_Disable	Deactivate termination resistors
EIB7_MD_Enable	Activate termination resistors

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_CantChInclnPrm Mode cannot be changed

## 7.29 Enabling the External Trigger Signal

The external trigger signal can be enabled or disabled. The external trigger and the timer trigger cannot be operated at the same time. If the timer trigger is already active, the EIB 741 delivers an error message as soon as an attempt is made to activate the external trigger.

### Function

```
EIB7_ERR EIB7EnableExternalTrigger ( EIB7_HANDLE eib,
                                      EIB7_MODE mode
                                      )
```

### Parameters

eib EIB handle  
mode Activate or deactivate the external trigger

Mode	Description
EIB7_MD_Disable	Deactivate external trigger
EIB7_MD_Enable	Activate external trigger

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_TimerTrgEn Timer trigger is already active

### 7.30 Software Trigger

The software trigger generates a trigger event and induces the EIB 741 to send data to the remote station. The "source" parameter is reserved for further options and must be set to zero. This function can only be executed in "Soft Realtime" mode. Both the timer trigger and the external trigger must be deactivated.

#### Function

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

#### Parameters

eib                            EIB handle  
 source                        Trigger source (must be 0)

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_TimerTrgEn                Timer trigger is already active  
 EIB7\_ExtTrgEn                 External trigger is already active

### 7.31 Reset

The EIB 741 performs a reset and reboots. This function has the same effect as pressing the reset button. The standard boot mode is used (firmware of the last update with user's network settings). The connection to the EIB 741 is closed automatically, in the same way as EIB7Close().

#### Function

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

#### Parameters

eib                            EIB handle

#### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

### 7.32 Identifying EIB 741

The LAN LED on the front panel of the EIB 741 can be moved to flash mode. If several devices are arranged side by side, an EIB 741 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_IllegalParameter         LED status cannot be changed (parameter is invalid)

### 7.33 Reading Data from the FIFO

Data packets are copied from the FIFO to the target memory (in raw data format). The "cnt" parameter indicates the number of entries to be copied from the FIFO. If the FIFO contains fewer data records, the entire content 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 expiry of the timeout. In this case, zero is returned to "entries". Only entire elements are copied from the FIFO. The target memory must be at least big enough to take the indicated number of FIFO entries. The contents of an entry in the FIFO are defined in Chapter 7.2 on page 21. 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	<i>[return value]</i> Pointer to target memory
cnt	Number of entries to be read (>=0)
entries	<i>[return value]</i> 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 indefinitely

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_FIFOEmpty	No data in the FIFO
EIB7_ElementSizeInv	Internal error
EIB7_FIFOOverflow	FIFO overflow since the function was last called (data lost)

### 7.34 Reading the Size of a FIFO Element

The size of a FIFO element in raw data format is returned in bytes. This value corresponds to the size of a FIFO entry, which is read using the EIB7ReadFIFODataRaw() function.

#### Function

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

#### Parameters

eib	EIB handle
size	<i>[return value]</i> Pointer to the variable for the size of a FIFO element in bytes

#### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

### 7.35 Access to the Contents of a FIFO Element

This function can be used to access individual fields of a FIFO element (in raw data). A FIFO entry contains, for example, the trigger counter, position data, the status word and additional data. These contents are present for all axes of the EIB 741. The data structure of a FIFO element is described in detail in Chapter 7.2 on page 21. 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 selects the axis from which the field is obtained. The fine selection can be made using the "type" parameter. This indicates which data field of the axis will 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 741

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

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_Timestamp	Timestamp for position
EIB7_PDF_ReferencePos1	Reference position 1
EIB7_PDF_ReferencePos2	Reference position 2
EIB7_PDF_DistCodedRef	Coded reference value
EIB7_PDF_AnalogEncA	ADC value for signal A
EIB7_PDF_AnalogEncB	ADC value for signal B

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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_FieldNotAvail                    The indicated field cannot be found

### 7.36 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 content 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 expiry of the timeout. In this case, zero is returned to "entries". Only entire elements are copied from the FIFO. The target memory must be at least big enough to take the indicated number of FIFO entries. The contents of an entry in the FIFO are defined in Chapter 7.2 on page 21. 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	<i>[return value]</i> Pointer to target memory
cnt	Number of entries to be read (>=0)
entries	<i>[return value]</i> 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 indefinitely

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_FIFOEmpty	No data in the FIFO
EIB7_ElementSizeInv	Internal error
EIB7_FIFOOverflow	FIFO overflow since the function was last called (data lost)

### 7.37 Reading the Size of a FIFO Element (Converted Data)

The size of a FIFO element (Converted Data) is returned. This value corresponds to the size of a FIFO entry, which is read using the EIB7ReadFIFOData() function.

#### Function

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

#### Parameters

eib	EIB handle
size	<i>[return value]</i> Pointer to the variable for the size of a FIFO element in bytes

#### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

### 7.38 Access to the Contents of a FIFO Element with Converted Data

This function can be used to access individual fields of a FIFO element with converted position data (ENCODER\_POSITION format). A FIFO entry contains, for example, the trigger counter, position data, the status word and additional data. These contents are present for all axes of the EIB 741. 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 selects the axis from which the field is obtained. The fine selection can be made using the "type" parameter. This indicates which data field of the axis will 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 741

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

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_Timestamp	Timestamp for position
EIB7_PDF_ReferencePos1	Reference position 1
EIB7_PDF_ReferencePos2	Reference position 2
EIB7_PDF_DistCodedRef	Coded reference value
EIB7_PDF_AnalogEncA	ADC value for signal A
EIB7_PDF_AnalogEncB	ADC value for signal B

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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_FieldNotAvail                    The indicated field cannot be found

### 7.39 Reading the Number of Elements in the FIFO

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

#### 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 delivers a status for the function call. All potential values are listed for the standard return values.

### 7.40 Clearing the FIFO

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

#### Function

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

#### Parameters

eib EIB handle

#### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

### 7.41 Setting the FIFO Size

The size of the FIFO is reestablished. All data in the FIFO is cleared. The size can only be set in polling mode. The FIFO size must be at least 2000 bytes. If the value is smaller, the value 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_SoftRTEn Soft realtime mode is activated

### 7.42 Reading the FIFO Size

The size of the FIFO in bytes is returned.

#### 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 delivers a status for the function call. All potential values are listed for the standard return values.

### 7.43 Activating the Callback Mechanism

The callback mechanism is activated or deactivated and the function pointer saved where applicable. The callback function is executed if there are at least as many elements saved in the FIFO as specified in the "threshold" parameter. This function will then only be called again if new data has been written to the FIFO and afterwards at least the "threshold" number of elements are 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 transmitted 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 the callback mechanism is triggered (>0)  
 handler               Pointer to the callback function (NULL is permitted if activate = 0)

#### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

#### Callback Function

The callback function is executed by the driver and runs in a separate thread. The user must attend to any necessary synchronization with the main program. The "eib" parameter contains the handle on the EIB 741 that has triggered the callback. "cnt" indicates the number of elements currently stored in the FIFO. The "data" parameter contains the pointer that was indicated 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

## 8 Axis Functions

The axis functions always refer to just one axis on the EIB 741. None of the other axes are affected.

All axis functions are able to deliver the following error messages as a return value. They can also return further values individually. These are listed separately for each function.

### Standard Return Values

EIB7_NoError	Function call successful
EIB7_InvalidHandle	The handle on the axis of the EIB 741 is invalid
EIB7_FuncNotSupp	Function is not supported by the EIB 741
EIB7_InvalidResponse	Error during data transmission
EIB7_AccNotAllowed	Function cannot be performed, as the EIB 741 does not permit access
EIB7_ConnReset	Connection terminated by the EIB 741
EIB7_ConnTimeout	Timeout during data transmission to the EIB 741
EIB7_ReceiveError	Error whilst receiving the data
EIB7_SendError	Error whilst sending the data
EIB7_OutOfMemory	The system is unable to allocate sufficient memory

### 8.1 Initializing the Axis

An axis of the EIB 741 is configured for the connected encoder. The axis number of the EIB 741 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, constants "EIB7\_IT\_EnDat22" and "EIB7\_IT\_EnDatDelayMeasurement" must be linked with "Or".

The "EnDatclock" parameter is used only for encoders with the EnDat interface. The EnDat interface clock can be set. This must be done using pre-defined constants. On initializing an axis for EnDat mode, an EnDat reset command is sent to the connected encoder. Additionally, the "Recovery time I" is set to  $10 \mu\text{s} < t_m < 30 \mu\text{s}$ . The error messages and warnings are cleared from the encoder's memory.

The "bandwidth" and "comp" parameters act only on incremental encoders. The bandwidth can be configured high and low on the two states. The online compensation function can be activated and deactivated. The "homing" and "limit" parameters are reserved for future options and must be assigned "none". 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 valid for distance-coded reference marks
- Error while calculating the coded reference value of distance-coded reference marks
- CRC error
- EnDat error message 1
- EnDat error message 2

The settings for the termination resistors of the incremental signals as well as the value of the period counter are not influenced by the 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
                               )
    
```

## 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 VPP)
EIB7_IT_EnDat21	Encoder with EnDat 2.1 interface
EIB7_IT_Endat01	Encoder with EnDat 2.1 interface and incremental signals (1 VPP)
EIB7_IT_Endat22	Encoder with EnDat 2.2 interface
EIB7_IT_EnDatDelayMeasurement	Delay compensation for EnDat 2.2

type                    Encoder type

Type	Description
EIB7_EC_Linear	Linear encoder
EIB7_EC_Rotary	Angle encoder / rotary encoder

refmarks                Type of reference marks

Remarks	Description
EIB7_RM_None	No reference mark
EIB7_RM_One	One reference mark (EIB741 does not calculate a coded reference value)
EIB7_RM_DistanceCoded	Distance-coded reference marks (EIB741 calculates coded reference value automatically)

linecounts             Number of signal periods per revolution (for rotative encoders only)  
increment               Nominal distance in signal periods between two fixed reference marks (for distance-coded reference marks only)  
homing                 Must be assigned EIB7\_HS\_None  
limit                   Must be assigned EIB7\_LS\_None  
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 VPP signals
EIB7_BW_Low	Low input bandwidth for 1 VPP signals

<b>EnDatclock</b>	<b>Description</b>
EIB7_CLK_Default	Default EnDat 2.1 / EnDat 2.2 clock
EIB7_CLK_100KHz	EnDat clock 100 kHz
EIB7_CLK_300KHz	EnDat clock 300 kHz (default for EnDat 2.1)
EIB7_CLK_500KHz	EnDat clock 500 kHz
EIB7_CLK_1MHz	EnDat clock 1 MHz
EIB7_CLK_2MHz	EnDat clock 2 MHz (default for EnDat 2.2)
EIB7_CLK_4MHz	EnDat clock 4 MHz
EIB7_CLK_5MHz	EnDat clock 5 MHz
EIB7_CLK_6_66MHz	EnDat clock 6.66 MHz

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_ParamInvalid	Parameter invalid
EIB7_InvInterface	Interface type invalid
EIB7_InvRefMarkOpt	Reference mark invalid
EIB7_InvDistCodeRef	Parameter for distance-coded reference marks invalid (linecount, increment)
EIB7_ConfOptIncons	Parameters cannot be combined in this form
EIB7_AccNotAllowed	Access denied
EIB7_EncPwrSuppErr	Error in the supply voltage for the encoder (encoder not ready)

## 8.2 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 is reset. The interpolation value is not changed.

### Function

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

### Parameters

axis                    AXIS handle

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_InvInterface            Interface type invalid

## 8.3 Polling Position

The encoder's current position value is captured and read. A status word from which potential position errors emerge is also transmitted. The position can only be polled in polling mode. Depending on whether the axis is configured for incremental or for EnDat encoders, the interpolated value of the incremental signals is provided or an EnDat poll is sent to the encoder. With 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 variable for the status word  
pos                     [return value] Pointer to the variable for the position

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_CantLatchPos            Position cannot be determined  
EIB7\_EncPwrSuppErr          Error in the encoder power supply (EnDat encoders only)  
EIB7\_NotInitialized         Axis is not configured

## 8.4 Reading Data for a Channel

The current position and certain additional parameters are captured and read. The "refc" parameter is valid only if the axis is configured for encoders with distance-coded reference marks. The function can only be performed in polling mode. The axis must be configured for incremental encoders.

### 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	<i>[return value]</i> Pointer to the variable for the status word
pos	<i>[return value]</i> Pointer to the variable for the position
ref1	<i>[return value]</i> Pointer to the variable for reference position 1
ref2	<i>[return value]</i> Pointer to the variable for reference position 2
refc	<i>[return value]</i> Pointer to the variable for the coded reference value
timestamp	<i>[return value]</i> Pointer to the variable for the timestamp value
counter	<i>[return value]</i> Pointer to the variable for the trigger counter
adc00	<i>[return value]</i> Pointer to the variable for the ADC value for signal A
adc90	<i>[return value]</i> Pointer to the variable for the ADC value for signal B

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_ParamInvalid	Parameter invalid (axis is possibly not configured for incremental encoders)
EIB7_EncPwrSuppErr	Error in the encoder power supply (EnDat encoders only)
EIB7_NotInitialized	Axis is not configured

## 8.5 Acknowledging the Power Supply Error

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

### Function

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

### Parameters

axis	AXIS handle
------	-------------

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_CantClearEnc	Error cannot be cleared
-------------------	-------------------------



### 8.9 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 latched", "Reference position 2 latched". 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_InvInterface                      Axis is not configured for incremental encoders  
 EIB7\_NotInitialized                      Axis is not configured

### 8.10 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 latched", "Reference position 2 latched", "Coded reference value valid", "Error on calculating 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_InvInterface                      Axis is not configured for incremental encoders  
 EIB7\_NotInitialized                      Axis is not configured

### 8.11 Starting Referencing

After this command is called, the reference position is saved when the next reference pulse is traversed. The "ref" parameter can be used to define whether only one or two reference positions are latched. If two reference positions are activated, one position value is latched with each of the two subsequent reference pulses. The latched values correspond to the period counter value at the time the reference mark is detected. 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 latched, the old reference position values become invalid and this is indicated by the flags in the status word. Referencing 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_InvInterface                      Axis is not configured for incremental encoders  
 EIB7\_NotInitialized                      Axis is not configured  
 EIB7\_ParamInvalid                      Parameter is not a valid option for the reference marks

### 8.12 Stopping Referencing

Referencing (mode for automatically latching the reference position) is stopped. If reference pulses 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_InvInterface            Axis is not configured for incremental encoders  
 EIB7\_NotInitialized        Axis is not configured  
 EIB7\_ParamInvalid           Invalid parameter for axis

### 8.13 Verifying Referencing Status

The status of referencing is returned. This enables the user to check whether referencing is still active or all reference positions have already been latched.

#### Function

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

#### Parameters

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

Mode	Description
EIB7_MD_Disable	Referencing complete
EIB7_MD_Enable	Referencing active

#### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

### 8.14 EnDat 2.1: Reading Position

The position of an EnDat encoder is read. This occurs via an EnDat 2.1 command. This function can only be performed in polling mode. The axis must be configured for EnDat01, EnDat21 or EnDat22 encoders. The position is always polled via an EnDat 2.1 command, 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_InvInterface            Axis is not configured for EnDat encoders  
 EIB7\_NotInitialized        Axis is not initialized  
 EIB7\_EncPwrSuppErr         Error in the encoder power supply (encoder is not operational)  
 EIB7\_EnDatErrII            EnDat error type II occurred  
 EIB7\_EnDatIfBusy           EnDat Master not operational  
 EIB7\_EnDatXmitErr          Error during data transmission (encoder is possibly not connected)

### 8.15 EnDat 2.1: Selecting Memory Area

The memory area in the EnDat encoder is selected. An EnDat 2.1 command is sent for this purpose. This function can only be performed in polling mode. The axis must be configured for EnDat01, EnDat21 or EnDat22 encoders. The memory area is always selected via an EnDat 2.1 command, 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_InvInterface	Axis is not configured for EnDat encoders
EIB7_NotInitialized	Axis is not initialized
EIB7_EncPwrSuppErr	Error in the encoder power supply (encoder is not operational)
EIB7_EnDatErrII	EnDat error type II occurred
EIB7_EnDatIfBusy	EnDat Master not operational
EIB7_EnDatXmitErr	Error during data transmission (encoder is possibly not connected)

### 8.16 EnDat 2.1: Sending Data

A data word is written to the memory of the EnDat encoder. Only 16-bit words are saved. The address indicates the memory cell within the active memory block. This function can only be executed in polling mode. The axis must be 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_InvInterface	Axis is not configured for EnDat encoders
EIB7_NotInitialized	Axis is not initialized
EIB7_EncPwrSuppErr	Error in the encoder power supply (encoder is not operational)
EIB7_EnDatErrII	EnDat error type II occurred
EIB7_EnDatIfBusy	EnDat Master not operational
EIB7_EnDatXmitErr	Error during data transmission (encoder is possibly not connected)

### 8.17 EnDat 2.1: Receiving Data

A data word is read from the memory of the EnDat encoder. Only a 16-bit word is read. The "addr" parameter indicates the memory cell within the active memory block from which the data is read. This function can only be executed in polling mode. The axis must be 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	<i>[return value]</i> Pointer to the variable for the received data word

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_InvInterface	Axis is not configured for EnDat encoders
EIB7_NotInitialized	Axis is not initialized
EIB7_EncPwrSuppErr	Error in the encoder power supply (encoder is not operational)
EIB7_EnDatErrII	EnDat error type II occurred
EIB7_EnDatIfBusy	EnDat Master not operational
EIB7_EnDatXmitErr	Error during data transmission (encoder is possibly not connected)

### 8.18 EnDat 2.1: Resetting the Encoder

The EnDat reset command is sent to the encoder. This function can only be executed in polling mode. The axis must be configured for EnDat01, EnDat21 or EnDat22 encoders. An EnDat 2.1 reset is always sent, even if the axis is configured for EnDat 2.2. The encoder performs a reset and is unavailable for a certain time. Additional information can be found in the datasheet for the encoder.

#### Function

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

#### Parameters

axis	AXIS handle
------	-------------

#### Return Value

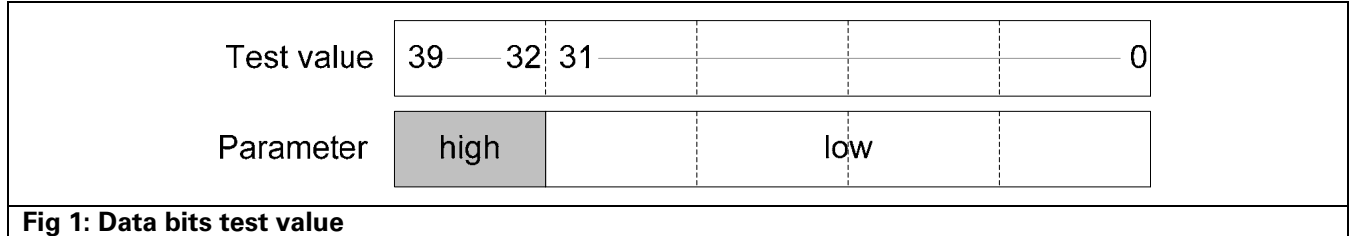
The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_InvInterface	Axis is not configured for EnDat encoders
EIB7_NotInitialized	Axis is not initialized
EIB7_EncPwrSuppErr	Error in the encoder power supply (encoder is not operational)
EIB7_EnDatErrII	EnDat error type II occurred
EIB7_EnDatIfBusy	EnDat Master not operational
EIB7_EnDatXmitErr	Error during data transmission (encoder is possibly not connected)

### 8.19 EnDat 2.1: Reading Test Values

A test value is read from the EnDat encoder. The test value is 40 bits long and is transmitted via two parameters. The content of the parameters are listed in the table below. This function can only be executed in polling mode. The axis must be 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.

Parameters	Data Bits Parameter	Data Bits Test Value
Low	D0..D31	D0..D31
High	D0..D7 D8..D31	D32..D39 Reserved



**Fig 1: Data bits test value**

#### 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 (highest value part)  
low                     [return value] Pointer to the variable for the test value (lowest value part)

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_InvInterface            Axis is not configured for EnDat encoders  
EIB7\_NotInitialized         Axis is not initialized  
EIB7\_EncPwrSuppErr         Error in the encoder power supply (encoder is not operational)  
EIB7\_EnDatErrII             EnDat error type II occurred  
EIB7\_EnDatIfBusy            EnDat Master not operational  
EIB7\_EnDatXmitErr          Error during data transmission (encoder is possibly not connected)

### 8.20 EnDat 2.1: Sending 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 be 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_InvInterface            Axis is not configured for EnDat encoders  
EIB7\_NotInitialized         Axis is not initialized  
EIB7\_EncPwrSuppErr         Error in the encoder power supply (encoder is not operational)  
EIB7\_EnDatErrII             EnDat error type II occurred  
EIB7\_EnDatIfBusy            EnDat Master not operational  
EIB7\_EnDatXmitErr          Error during data transmission (encoder is possibly not connected)

## 8.21 EnDat 2.2: Reading Position and Additional Information

The position of an EnDat22 encoder is read. If activated, the EnDat additional information is also transmitted. Each piece of additional information 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 information. This function can only be executed in polling mode. The axis must be 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	<i>[return value]</i> Pointer to the variable for the status word
pos	<i>[return value]</i> Pointer to the variable for the position value
ai1	<i>[return value]</i> Pointer to the structure for EnDat additional information 1
ai2	<i>[return value]</i> Pointer to the structure for EnDat additional information 2

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_InvInterface	Axis is not configured for EnDat encoders
EIB7_NotInitialized	Axis is not initialized
EIB7_EncPwrSuppErr	Error in the encoder power supply (encoder is not operational)
EIB7_EnDatErrll	EnDat error type II occurred
EIB7_EnDatIfBusy	EnDat Master not operational
EIB7_EnDatXmitErr	Error during data transmission (encoder is possibly not connected)
EIB7_EnDat22NotSupp	The encoder does not support EnDat 2.2 commands or the axis is not configured for EnDat 2.2 mode

## 8.22 EnDat 2.2: Reading Position and Additional Information and Selecting Memory Area

The position and additional information of an EnDat22 encoder are transferred as described in Section 8.21. The encoder selects the memory area, which is specified by the MRS code and the block address. If no block is selected from the "section 2" memory area, the "block" parameter must be set to zero. This function can only be executed in polling mode. The axis must be 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	<i>[return value]</i> Pointer to the variable for the status word
pos	<i>[return value]</i> Pointer to the variable for the position value
ai1	<i>[return value]</i> Pointer to the structure for EnDat additional information 1
ai2	<i>[return value]</i> Pointer to the structure for EnDat additional information 2
mrs	MRS code for the memory area
block	Block address for the "section 2" memory area

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_InvInterface	Axis is not configured for EnDat encoders
EIB7_NotInitialized	Axis is not initialized
EIB7_EncPwrSuppErr	Error in the encoder power supply (encoder is not operational)
EIB7_EnDatErrll	EnDat error type II occurred
EIB7_EnDatIfBusy	EnDat Master not operational
EIB7_EnDatXmitErr	Error during data transmission (encoder is possibly not connected)
EIB7_EnDat22NotSupp	The encoder does not support EnDat 2.2 commands or the axis is not configured for EnDat 2.2 mode

### 8.23 EnDat 2.2: Reading Position and Additional Information and Sending Data

The position and additional information of an EnDat22 encoder are transferred as described in Section 8.21. The 16-bit data word is written to the memory of the encoder. The address (8-bit) indicates the memory cell within the selected memory area. This function can only be executed in polling mode. The axis must be 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	<i>[return value]</i> Pointer to the variable for the status word
pos	<i>[return value]</i> Pointer to the variable for the position value
ai1	<i>[return value]</i> Pointer to the structure for EnDat additional information 1
ai2	<i>[return value]</i> Pointer to the structure for EnDat additional information 2
addr	Memory address within the active memory block
data	Data word that is written to the memory

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_InvInterface	Axis is not configured for EnDat encoders
EIB7_NotInitialized	Axis is not initialized
EIB7_EncPwrSuppErr	Error in the encoder power supply (encoder is not operational)
EIB7_EnDatErrII	EnDat error type II occurred
EIB7_EnDatIfBusy	EnDat Master not operational
EIB7_EnDatXmitErr	Error during data transmission (encoder is possibly not connected)
EIB7_EnDat22NotSupp	The encoder does not support EnDat 2.2 commands or the axis is not configured for EnDat 2.2 mode

## 8.24 EnDat 2.2: Reading Position and Additional Information and Receiving Data

The position and additional information of an EnDat22 encoder are transferred as described in Section 8.21. The encoder reads a data word from its memory. The "addr" parameter specifies the address of the memory cell inside the selected memory area. The data is transmitted via the additional information and cannot be read until the next EnDat command. The appropriate additional information must be selected (see EnDat specification). This function can only be performed in polling mode. The axis must be configured for EnDat 2.2 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	<i>[return value]</i> Pointer to the variable for the status word
pos	<i>[return value]</i> Pointer to the variable for the position value
ai1	<i>[return value]</i> Pointer to the structure for EnDat additional information 1
ai2	<i>[return value]</i> Pointer to the structure for EnDat additional information 2
addr	Memory address within the active memory block

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_InvInterface	Axis is not configured for EnDat encoders
EIB7_NotInitialized	Axis is not initialized
EIB7_EncPwrSuppErr	Error in the encoder power supply (encoder is not operational)
EIB7_EnDatErrII	EnDat error type II occurred
EIB7_EnDatIfBusy	EnDat Master not operational
EIB7_EnDatXmitErr	Error during data transmission (encoder is possibly not connected)
EIB7_EnDat22NotSupp	The encoder does not support EnDat 2.2 commands or the axis is not configured for EnDat 2.2 mode

## 8.25 EnDat 2.2: Reading Position and Additional Information and Sending Test Command

The position and additional information of an EnDat22 encoder are transferred as described in Section 8.21. The "port" parameter contains the port address for the test command. This function can only be executed in polling mode. The axis must be 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	<i>[return value]</i> Pointer to the variable for the status word
pos	<i>[return value]</i> Pointer to the variable for the position value
ai1	<i>[return value]</i> Pointer to the structure for EnDat additional information 1
ai2	<i>[return value]</i> Pointer to the structure for EnDat additional information 2
port	Port address for the test command

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_InvInterface	Axis is not configured for EnDat encoders
EIB7_NotInitialized	Axis is not initialized
EIB7_EncPwrSuppErr	Error in the encoder power supply (encoder is not operational)
EIB7_EnDatErrII	EnDat error type II occurred
EIB7_EnDatIfBusy	EnDat Master not operational
EIB7_EnDatXmitErr	Error during data transmission (encoder is possibly not connected)
EIB7_EnDat22NotSupp	The encoder does not support EnDat 2.2 commands or the axis is not configured for EnDat 2.2 mode

## 8.26 EnDat 2.2: Reading Position and Additional Information and Sending Error Reset

The position and additional information of an EnDat22 encoder are transferred as described in Section 8.21. The error flags of the EnDat22 encoder are also cleared. This function can only be executed in polling mode. The axis must be 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	<i>[return value]</i> Pointer to the variable for the status word
pos	<i>[return value]</i> Pointer to the variable for the position value
ai1	<i>[return value]</i> Pointer to the structure for EnDat additional information 1
ai2	<i>[return value]</i> Pointer to the structure for EnDat additional information 2

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_InvInterface	Axis is not configured for EnDat encoders
EIB7_NotInitialized	Axis is not initialized
EIB7_EncPwrSuppErr	Error in the encoder power supply (encoder is not operational)
EIB7_EnDatErrII	EnDat error type II occurred
EIB7_EnDatIfBusy	EnDat Master not operational
EIB7_EnDatXmitErr	Error during data transmission (encoder is possibly not connected)
EIB7_EnDat22NotSupp	The encoder does not support EnDat 2.2 commands or the axis is not configured for EnDat 2.2 mode

## 8.27 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 with the status words. This function can only be executed in polling mode. The axis must be configured for EnDat01 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	<i>[return value]</i> Pointer to the variable for the status word of the EnDat position
posEnDat	<i>[return value]</i> Pointer to the variable for the EnDat position value
statusIncr	<i>[return value]</i> Pointer to the variable for the status word of the incremental position
posEnDat	<i>[return value]</i> Pointer to the variable for the incremental position value

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_InvInterface	Axis is not configured for EnDat encoders
EIB7_NotInitialized	Axis is not initialized
EIB7_EncPwrSuppErr	Error in the encoder power supply (encoder is not operational)
EIB7_EnDatErrII	EnDat error type II occurred
EIB7_EnDatIfBusy	EnDat Master not operational
EIB7_EnDatXmitErr	Error during data transmission (encoder is possibly not connected)
EIB7_CantLatchPos	Position cannot be determined

### 8.28 Setting the Power Supply for Encoders

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

#### Function

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

#### Parameters

axis                    AXIS handle  
mode                    Activate or deactivate power supply

Mode	Description
EIB7_MD_Disable	Switch off power supply
EIB7_MD_Enable	Switch on power supply

#### Return Value

The return value delivers a status for the function call. All potential values are listed for the standard return values.

### 8.29 Reading Power Supply Status for Encoders

The power supply status for the encoder can be read. The "power" parameter can be used to determine whether the power supply for this axis is 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 power supply status

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 delivers a status for the function call. All potential values are listed for the standard return values.

### 8.30 Configuring Timestamp

The timestamp can be activated or deactivated for each axis. The period is set globally for all axes of a EIB 741. The timestamp value is copied during the position poll for an axis, if this function was activated beforehand.

#### 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 delivers a status for the function call. All potential values are listed for the standard return values.

## 9 IO Functions

The IO functions always refer only to a single output or input port on the EIB 741. None of the other ports are affected.

All IO functions can provide the following error messages as a return value. They can also return further values individually. These are listed separately for each function.

### Standard Return Values

EIB7_NoError	Function call successful
EIB7_InvalidHandle	The handle on the axis of the EIB 741 is invalid
EIB7_FuncNotSupp	Function is not supported by the EIB 741
EIB7_InvalidResponse	Error during data transmission
EIB7_AccNotAllowed	Function cannot be performed, as the EIB 741 does not permit access
EIB7_ConnReset	Connection terminated by the EIB 741
EIB7_ConnTimeout	Timeout during data transmission to the EIB 741
EIB7_ReceiveError	Error whilst receiving the data
EIB7_SendError	Error whilst sending the data
EIB7_OutOfMemory	The system is unable to allocate sufficient memory
EIB7_PortDirInv	Port signal direction invalid

### 9.1 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 termination resistor of the differential input can also be activated or deactivated. This function is only permitted for handles on 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 termination resistor

Termination	Description
EIB7_MD_Disable	Deactivate termination resistor
EIB7_MD_Enable	Activate termination resistor

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_ParamInvalid            Parameter invalid

## 9.2 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. Additionally the output driver can be deactivated. In this case, the output is in a high-impedance state. This function is only permitted for handles on 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_ParamInvalid Parameter invalid

## 9.3 Reading the Logical Port

The level at a logical input or output is read (parameter "level"). The operating mode of the port is also determined. If the port is operated as a trigger input or trigger output, the "level" value is invalid. With a logical output, the set level is read back.

### 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 operating mode

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
0	low level
1	high level

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_ParamInvalid Parameter invalid

## 9.4 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 for logic mode. 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 delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_ParamInvalid            Parameter invalid  
EIB7\_TrgNotConf             Output is not a logical port

## 9.5 Reading Configuration Data for Input

The configuration data for an input port is read. The "mode" parameter delivers the operating mode of the input. In "termination" the state of the termination resistor is returned. The function can 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 operating mode

Mode	Description
EIB7_IOM_Trigger	Trigger Input
EIB7_IOM_Logical	Logical input

termination            *[return value]* Pointer to the variable for the termination resistor

Termination	Description
EIB7_MD_Disable	Termination resistor deactivated
EIB7_MD_Enable	Termination resistor activated

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_ParamInvalid            Parameter invalid

## 9.6 Reading Configuration Data for Output

The configuration data for an output port is read. The "mode" parameter provides the operating mode of the output. In "enable" the state of the output driver is returned. The function can 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 operating mode

Mode	Description
EIB7_IOM_Trigger	Trigger output
EIB7_IOM_Logical	Logical output

enable *[return value]* Pointer to the variable for the output driver status

Enable	Description
EIB7_MD_Disable	Output driver deactivated
EIB7_MD_Enable	Output driver activated

### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7\_ParamInvalid Parameter invalid

## 10 General Functions

All general functions can provide the following error messages as a return value. They can also return further values individually. These are listed separately for each function.

### Standard Return Values

EIB7_NoError	Function call successful
EIB7_OutOfMemory	The system is unable to allocate sufficient memory

### 10.1 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 zero byte, is longer than the memory area, an error message will be generated. The target memory size must be at least 9 bytes.

#### Function

```
EIB7_ERR EIB7GetDriverID      ( char*          ident,  
                               unsigned long      len  
                               )
```

#### Parameters

ident	<i>[return value]</i> Target memory for the C-string
len	Size of the target memory in bytes

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

EIB7_BufferTooSmall	Target memory is too small
---------------------	----------------------------

### 10.2 Converting Error Message into Text

An error code is converted into a text message and returned as a C-string. A descriptive text and a brief description are defined in the system for all known error codes. The "mnemonic" parameter is used to return a brief description of the error message in text format (approx. 30-40 characters). The "message" parameter contains a more detailed description (approx. 100-150 characters). If a NULL pointer is transmitted for either the "mnemonic" or "message" parameters, the function does not copy the corresponding 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 zero 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	<i>[return value]</i> Pointer to the target memory for the brief description
mnemlen	Size of the "mnemonic" target memory in bytes
message	<i>[return value]</i> Pointer to the target memory for the error text
msglen	Size of the "message" target memory in bytes

#### Return Value

The return value delivers a status for the function call. In addition to the standard return values, the following error messages can also occur.

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) 5061

E-mail: [info@heidenhain.de](mailto:info@heidenhain.de)

---

**Technical support** **FAX** +49 (8669) 32-1000

**Measuring systems** ☎ +49 (8669) 31-3104

E-mail: [service.ms-support@heidenhain.de](mailto:service.ms-support@heidenhain.de)

**TNC support** ☎ +49 (8669) 31-3101

E-mail: [service.nc-support@heidenhain.de](mailto:service.nc-support@heidenhain.de)

**NC programming** ☎ +49 (8669) 31-3103

E-mail: [service.nc-pgm@heidenhain.de](mailto:service.nc-pgm@heidenhain.de)

**PLC programming** ☎ +49 (8669) 31-3102

E-mail: [service.plc@heidenhain.de](mailto:service.plc@heidenhain.de)

**Lathe controls** ☎ +49 (8669) 31-3105

E-mail: [service.lathe-support@heidenhain.de](mailto:service.lathe-support@heidenhain.de)

---

**[www.heidenhain.de](http://www.heidenhain.de)**

