



HEIDENHAIN



Benutzer-Handbuch
für die Applikationsentwicklung

EIB 8791

Externe Interface-Box
zum Anschluss von
HEIDENHAIN-Messgeräten

ZIELGRUPPE DES BENUTZER-HANDBUCHES	4
DOKUMENTATION	4
FIRMWAREVERSION	4
CHANGE HISTORY	4
GLOSSAR	4
1 FUNKTIONSBESCHREIBUNG	5
1.1 Allgemein	5
1.2 Konfiguration der Messgeräte-Eingänge	6
1.3 Behandlung von Referenzmarken	6
1.3.1 Messgeräte mit einer Referenzmarke	6
1.3.2 Messgeräte mit abstandscodierten Referenzmarken	7
1.3.3 Ablauf der Referenzierung	8
1.4 Positionswertbildung	9
1.4.1 Online Kompensation	9
1.4.2 Positionswertberechnung	9
1.4.3 Signalformkompensation	10
1.4.4 Korrekturwertaufnahme zur Signalformkompensation	11
1.4.5 Positionswertefilter	13
1.5 Positionsdaten-Ausgabe	13
1.5.1 Positionspakete über TCP/IP Kommandolink	14
1.5.2 Positionsdatenausgabe per UDP	14
1.5.3 Speichern von Positionspaketen auf der EIB 8791 (Recording)	15
1.5.4 Positionsdaten über Position Data Link	15
1.6 Synchronisation und asynchrone Triggerung	16
1.6.1 Asynchroner Betrieb mit Trigger	17
1.6.2 Synchroner Betrieb mit SynCik & PTM	18
1.6.3 „Synchronisation“ mehrere EIB 8791	21
1.7 Digitale und Analoge Hilfeingänge	22
1.7.1 Digital IN	22
1.7.2 Analog IN	22
1.7.3 AUX IO	22
1.8 Events, Selbsttest und Diagnostik	23
1.9 Reset	23
1.10 Shutdown	23
1.11 Firmware Update	23
1.11.1 Update mittels EIB 8791 Application	23
1.11.2 FTP Update	23
1.12 FTP Server der EIB 8791	25
1.12.1 Konfiguration der FTP-Parameter	25
1.12.2 Verzeichnisstruktur des FTP-Servers.	25
1.12.3 Ablage von PDL Daten	25
1.12.4 Ablage von Memorydumpdaten	25
1.12.5 Übertragung von MAP Daten zu einem Slot	26
1.13 DHCP	26
2 KONFIGURATION UND BENUTZUNG DER EIB 8791 MIT DER TREIBERSOFTWARE	27
2.1 Allgemeine Informationen	27
2.2 Inhalt der Treiber CD	28
2.3 Installationsanleitung der Treiber DLL	28
2.3.1 Windows	28
2.3.2 Linux	29
2.4 Ethernet-Verbindung mit der EIB 8791 herstellen	29

2.5	Überblick EIB 8791 Treiber DLL	30
2.5.1	Treiberstruktur	30
2.5.2	Funktionsaufrufe	30
2.5.3	Strings	30
2.5.4	Grundlegende Funktionen	30
2.6	Parameter	33
2.6.1	Nomenklatur	33
2.6.2	Parameter der EIB 8791	34
2.6.3	Lesen und Schreiben von Parametern	34
2.6.4	Konfigurationsdateien	35
2.7	Positionsdaten PD	35
2.7.1	Datenpfad	35
2.7.2	Positionsdatenformat	36
2.7.3	Positionsdatentypen	36
2.8	Standard Anwendungsfälle	38
2.8.1	Reset – Shutdown der EIB 8791	39
2.8.2	Abfragen der Event Queues	39
2.8.3	Laden einer Standardkonfiguration	40
2.8.4	Konfiguration zurücksetzen	41
2.8.5	Polling von Positionsdaten	41
2.8.6	UDP Empfang von Positionsdaten	41
2.9	Eigene Konfiguration einstellen	44
2.9.1	Konfiguration des Systemtakts	44
2.9.2	Konfiguration der Trigger	45
2.9.3	Konfiguration der Messgeräte	48
2.9.4	Positionsdatenausgabe	49
2.9.5	Synchroner Start der Positionsdatenpakete	50
2.10	Weitere Anwendungsfälle	51
2.10.1	Recording von Positionsdaten	51
2.10.2	Referenzfahrt	52
2.10.3	Positionswert Initialisieren	53
2.10.4	Positionsfehler löschen	53
2.10.5	Korrekturwertaufnahme zur Signalformkompensation	53
2.10.6	Netzwerkparameter setzen	56
2.10.7	Software Update prüfen	57
2.10.8	Analog und Digitale Ein-/Ausgänge	57
3 ANHANG		59
3.1	Beispielprogramme in C	59
3.2	Kommandozeilentool	60
3.3	LabVIEW™ EIB 8791 Application	62
3.3.1	Startfenster und verbinden mit der EIB 8791	62
3.3.2	Seite <i>EIB8</i>	63
3.3.3	Seite <i>Application</i>	63
3.3.4	Seite <i>Events</i>	64
3.3.5	Seite <i>Parameter</i>	64
3.3.6	Seite <i>Status</i>	64
3.3.7	Seite <i>Configuration</i>	64
3.3.8	Seite <i>Software Update (FTP)</i>	65
3.3.9	Seite <i>Axis</i>	65
3.3.10	Seite <i>ADC</i>	65
3.3.11	Seite <i>Polling</i>	65
3.3.12	Seite <i>UDP Display</i>	66
3.3.13	Seite <i>UDP Dumping</i>	66
3.3.14	Seite <i>Recording EIB8</i>	66
3.4	LabVIEW™ Beispiele	67

Zielgruppe des Benutzer-Handbuches

Dieses Handbuch muss von jeder Person gelesen und beachtet werden, die mit den folgenden Arbeiten betraut ist:

- Fachpersonal Applikationsentwicklung:
Das Fachpersonal Applikationsentwicklung ist aufgrund seiner fachlichen Ausbildung, Kenntnisse und Erfahrung sowie Kenntnis der einschlägigen Bestimmungen in der Lage, die ihm übertragenen Arbeiten hinsichtlich der jeweiligen Applikation auszuführen und mögliche Gefahren selbstständig zu erkennen und zu vermeiden.

Dokumentation

Die Dokumentation zur EIB 8791 umfasst folgende Unterlagen:

- Betriebsanleitung:
 - Unterlagen, die für die Inbetriebnahme erforderlich sind, sowie Technische Daten.
- Benutzer-Handbuch für die Applikationsentwicklung:
 - Beschreibung des Funktionsumfangs der EIB 8791.
 - Beschreibung der Installation und der Funktionsaufrufe der Treiber-Software.

Anmerkung:

Aufgrund des modularen internen Aufbaus der EIB 8791 wird speziell bei den Treiber-Funktionen und im Zusammenhang mit der Basiskomponente der EIB 8791 an vielen Stellen der Begriff „EIB8“ verwendet.

Firmwareversion

Die vorliegende Dokumentation beschreibt die Firmware-Version:

- 816477-xx
- 1128592-xx

Change History

Änderungen zu vorhergehenden Versionen können aus der Change History entnommen werden.

Das Dokument zur Change History ist auf der CD im Unterverzeichnis EIB_8791/doc zu finden. Bitte lesen Sie dieses Dokument, speziell die Hinweise zu neuen, geänderten und obsoleten Funktionsaufrufen.

Glossar

Begriff	Erklärung
EIB	External Interface Box; Schnittstellen-Elektronik zur präzisen Positionserfassung, speziell für Prüf- bzw. Mehrstellen-Messplätze und zur mobilen Datenerfassung, z.B. bei der Maschinenvermessung
SLOT	Die EIB 8791 stellt 8 Messgeräte-Eingänge zur Verfügung. Die 8 Eingänge sind in 4 Slots mit je 2 Messgeräte-Eingängen gruppiert.
Achse	Bezeichnet eine Messachse der Applikation für die die EIB 8791 verwendet wird. Je Achse wird ein Anschluss für ein Messgerät zur Verfügung gestellt.
Messgerät	Positions-Messgerät von HEIDENHAIN zur Bestimmung von Länge oder Winkel
Trigger	Asynchrones Signal, das eine Positionswertausgabe auslöst
Synchronisation	Zeitliche Synchronisation der Positionswertbildung
PTM	Position Trigger Marker; Signal das einzelne Taktflanken des SynClk markiert
SynClk	Synchronisations Clock; Taktsignal für eine hochgenaue zeitliche Synchronisation des Abtastzeitpunktes der Positionswertbildung
PD	Positionsdaten
PDL	Positionsdatenlink
Online Kompensation	Automatischer, mitlaufender Abgleich von Offset, Phase und Amplitude der Abtastsignale eines Positionsmesssystems
Signalform Kompensation	Stützstellenbasierte, ortsabhängige Kompensation der Auswirkung der Signalformabweichung der Abtastsignale eines Positionsmesssystems auf den Positionswert
Trigger-Ereignis	Flanke eines konfigurierten Triggers oder mit PTM markierte Taktflanke des SynClk
FTP	File Transfer Protokoll
UDP	User Datagram Protokoll
TCP	Transmission Control Protokoll
GPIO	General Purpose Input Output (Allgemein verwendbarer Digitaler Ein- bzw. Ausgang)

1 Funktionsbeschreibung

1.1 Allgemein

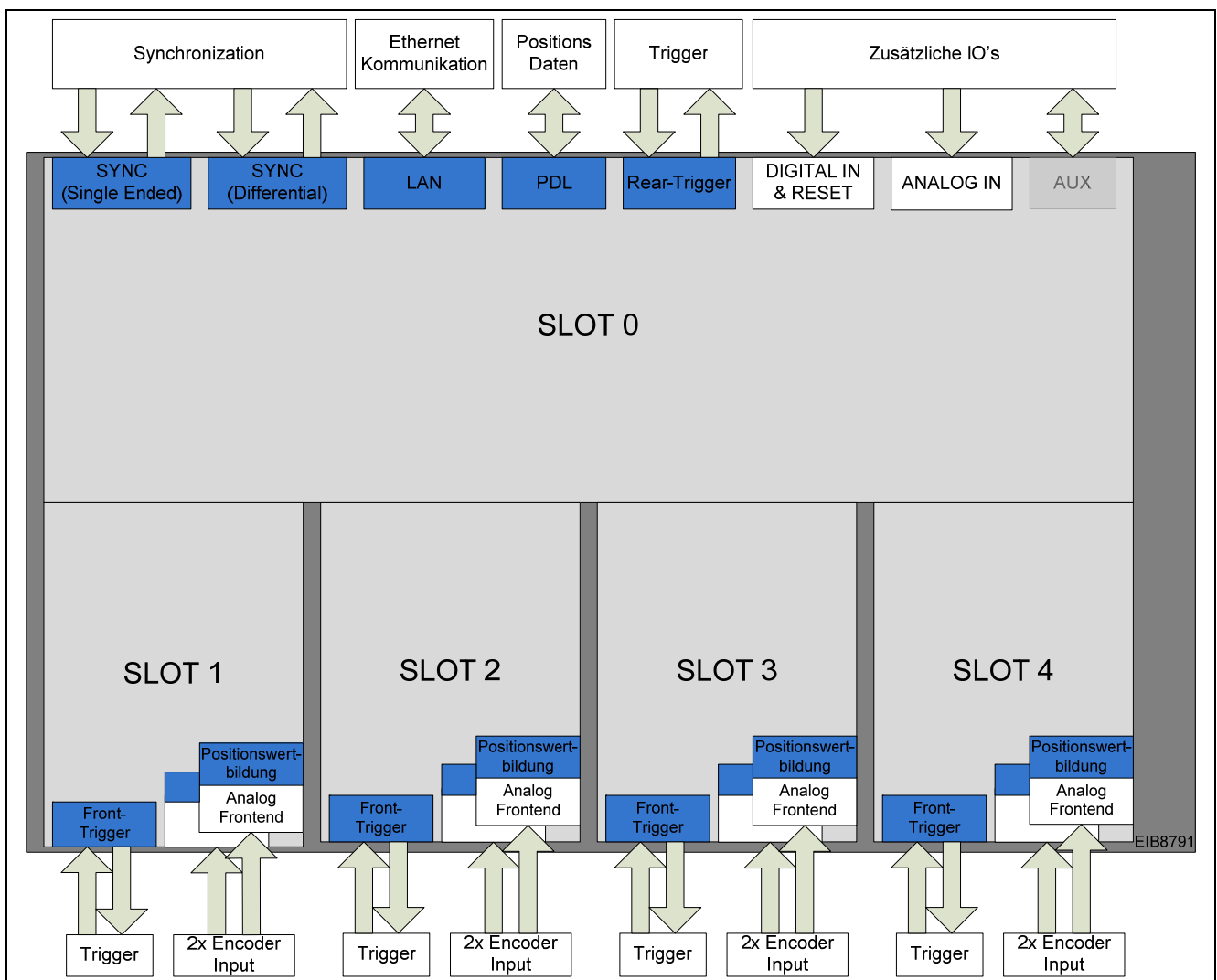
Die EIB 8791 ist eine externe Interface-Box zur präzisen Positionsmessung speziell für Prüfplätze, Mehrstellen-Messplätze, sowie Anlagen und Maschinen mit mehreren Achsen. Die EIB 8791 ist ideal für Anwendungen, die eine hohe Auflösung der Messgerätesignale und eine schnelle Messwerterfassung erfordern. Besonders für hoch-dynamische Maschinen ist die synchrone Positionswertbildung in der EIB 8791 optimiert. Außerdem ermöglicht die Ethernet-Übertragung die Verwendung von Switches bzw. Hubs zur Verschaltung von mehreren EIB 8791. An die EIB 8791 können bis zu acht HEIDENHAIN-Messgeräte mit sinusförmigen Inkrementalsignalen (1 V_{SS}) angeschlossen werden.

Zur Messwertbildung unterteilt die EIB 8791 die Signalperioden der Inkrementalsignale 65 536fach (16 Bit). Der automatische Abgleich der sinusförmigen Inkrementalsignale (Online-Kompensation¹) reduziert die Abweichungen innerhalb einer Signalperiode. Zusätzlich können weitere Signalfehler über eine stützstellenbasierte, ortsabhängige Kompensation (Signalformkompensation²) reduziert werden. Durch die hohe interne Abtastrate von 50 MHz können Positionswerte zusätzlich noch digital gefiltert werden. Es können neben den Positionswerten auch Geschwindigkeiten und/oder Beschleunigungswerte ausgegeben werden.

Durch den integrierten Messwertspeicher ermöglicht die EIB 8791 im Betriebsmodus „Recording“ ein Abspeichern von bis zu 125 000 Messwerten pro Achse. Das Abspeichern der Messwerte erfolgt achsabhängig wahlweise über interne oder externe Trigger. Zur Datenausgabe steht eine Standard-Ethernet-Schnittstelle (Verwendung von TCP- bzw. UDP-Kommunikation) oder ein schneller synchroner PositionDataLink (PDL) zur Verfügung. Damit ist sowohl eine direkte Anbindung an PC, Laptop oder Industrie-PC als auch die Anbindung an dedizierte Steuerungshardware möglich.

Die Art der Messwertübertragung kann über den Betriebsmodus eingestellt werden. Zur Verarbeitung der Messwerte im PC sind im Lieferumfang Treiber-Software für Windows, Linux und LabVIEW enthalten. Die Treiber-Software ermöglicht eine einfache Programmierung von Kundenapplikationen. Zusätzlich demonstrieren Beispielprogramme die Möglichkeiten der EIB 8791.

Prinzipschaltbild:



¹ Automatischer, mitlaufender Abgleich von Offset, Phase und Amplitude der Abtastsignale eines Positionsmesssystems

² Stützstellenbasierte, ortsabhängige Kompensation der Signalformabweichung der Abtastsignale eines Positionsmesssystems auf den Positionswert

1.2 Konfiguration der Messgeräte-Eingänge

An die EIB 8791 können bis zu acht HEIDENHAIN-Messgeräte mit folgenden Schnittstellen angeschlossen werden:

- Inkrementalsignale 1 V_{SS}

Die Spannungsversorgung der Messgeräte erfolgt von der EIB 8791 und ist durch eine rücksetzbare Überstromabschaltung abgesichert.

Technische Daten siehe „Betriebsanleitung“.

Nach dem Power-up ist die Spannungsversorgung der Messgeräte ausgeschaltet. Die Parameter zum Betrieb des Messgerät-Einganges müssen per Initialisierung gesetzt werden. Dabei werden typabhängige Informationen über das Messgerät an eine Achse übergeben:

- Schnittstellentyp
- Verarbeitung der Referenzmarken
- Verarbeitung der Homing/Limit-Signale

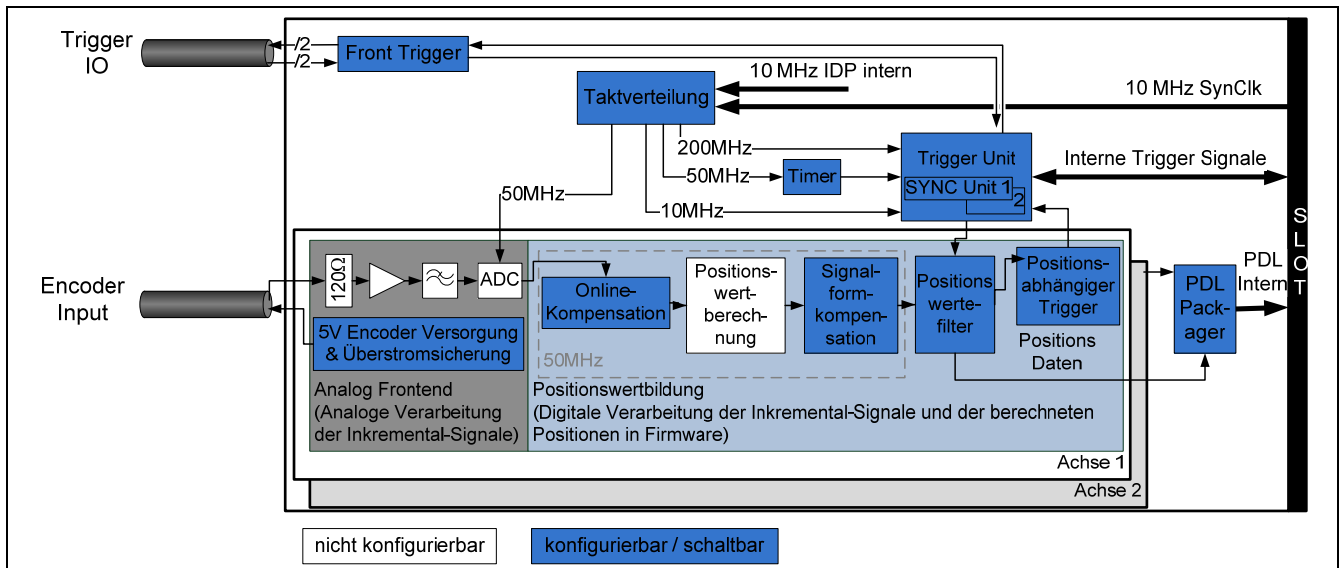
Die Konfiguration einer Achse kann nur bei ausgeschalteter Versorgungsspannung der entsprechenden Achse geändert werden.

Anmerkungen:

Durch Abschalten der Spannungsversorgung des Messgeräts

- geht die Referenzierung der Achse verloren
- werden übertragene Positions-, Geschwindigkeits-, Beschleunigungs-, und Referenzmarken-Pakete als ungültig markiert. (Die genannten Pakettypen haben alle das gleiche Statuswort)
- ist ein Rücksetzen der Konfiguration erforderlich

Blockschaltbild der Messgeräte Anschlüsse und des Signalverarbeitungspfad:



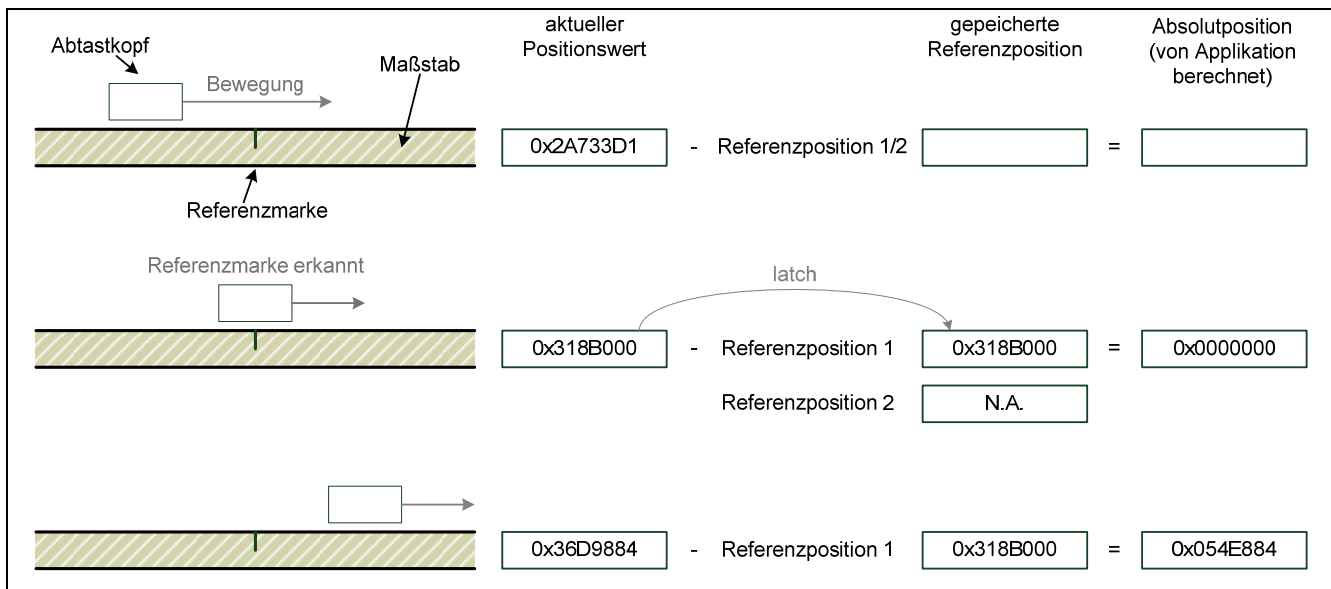
1.3 Behandlung von Referenzmarken

Bei inkrementellen Messgeräten wird die Referenzmarke bzw. werden die Referenzmarken dazu benutzt, einen absoluten Bezug zwischen Achsposition und Anzeigewert herzustellen. Dieser Bezug geht nach einer Unterbrechung der Versorgungsspannung der Achse verloren.

1.3.1 Messgeräte mit einer Referenzmarke

Bei Messgeräten mit einer Referenzmarke hat diese einen eindeutigen Bezug zu einer bestimmten Signalperiode. Diese Signalperiode kann als Bezug zur Bildung von absoluten Positionswerten verwendet werden. Das Überfahren der Referenzmarke hat keinen Einfluss auf den Periodenzähler oder den Interpolationswert. Es wird lediglich der zum Zeitpunkt des Überfahrens gültige Periodenzählerwert in einem Register für die Referenzposition gespeichert. Mit diesem Wert können in der Kunden-Software absolute Positionswerte berechnet werden.

Das folgende Bild zeigt den prinzipiellen Ablauf bei der Ermittlung einer Referenzposition. Die angezeigten Werte sind nur als Beispiel zu verstehen und der Übersicht halber ist nur ein Ausschnitt des Positionswert-Registers gezeigt.



Ein automatisches Speichern der Referenzposition muss per Software aktiviert werden. Nach diesem Kommando wartet die Achse auf die nächste Referenzmarke und speichert dann die Referenzposition. Ein erneutes Speichern muss anschließend wieder aktiviert werden.

Die Referenzposition kann entweder per Kommando abgefragt oder als Positionsdatenpaket übertragen werden.

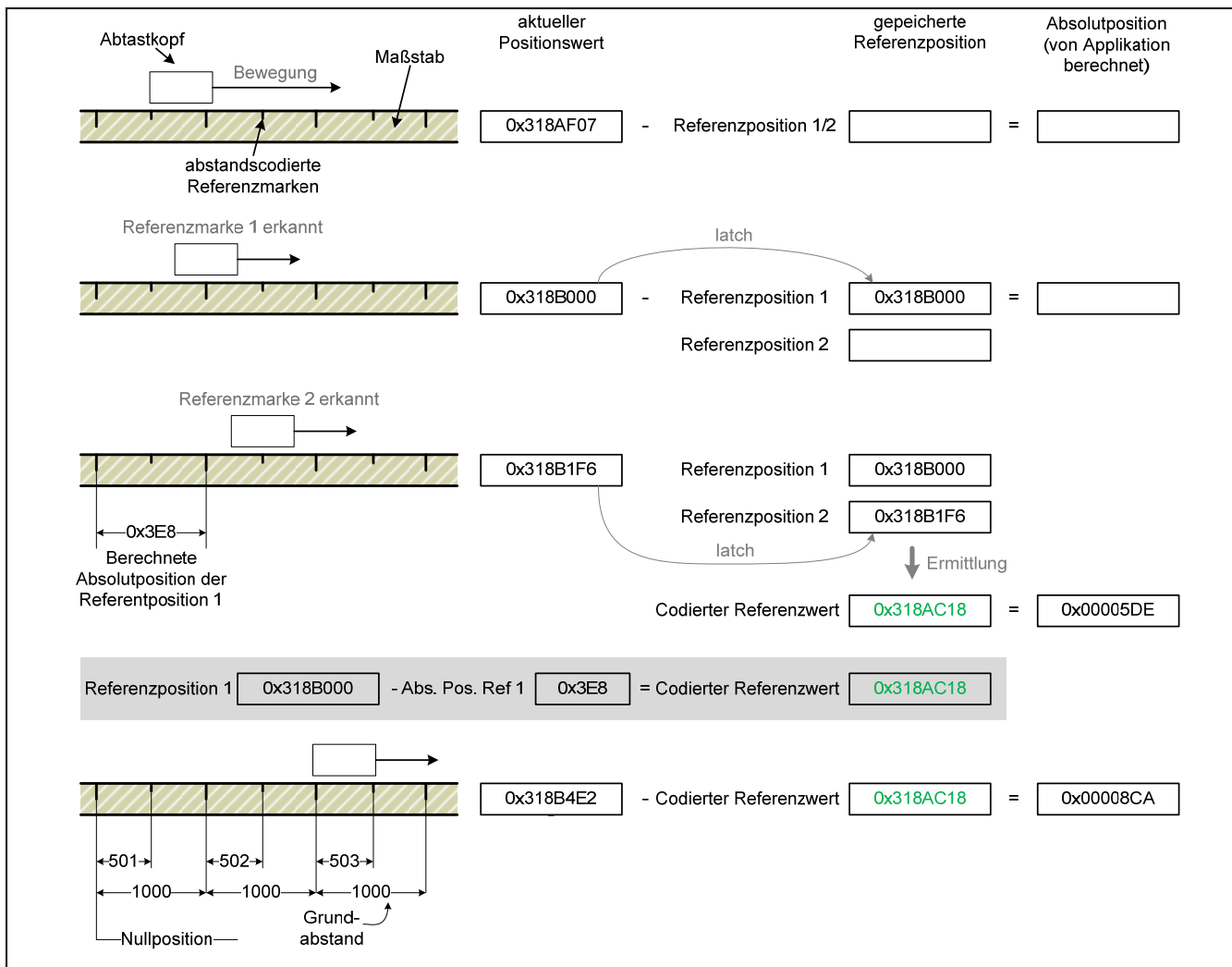
1.3.2 Messgeräte mit abstandscodierten Referenzmarken

Bei Messgeräten mit abstandscodierten Referenzmarken befinden sich über den ganzen Messweg Referenzmarken in einem festen Abstand (Grundabstand in Signalperioden). Zwischen zwei dieser Referenzmarken befindet sich eine dritte, deren Abstand zu den anderen beiden so variiert, dass jeder Abstand ein Vielfaches der Teilungsperiode beträgt und nur einmal über den gesamten Messweg vorkommt (vgl. Bild unten). Der Bezug zur Bildung von absoluten Positionswerten wird aus den Zählerwerten durch den Abstand zweier überfahrener (nebeneinander liegenden) Referenzmarken gewonnen.

Zu diesem Zweck erfolgt das Speichern des Periodenzählers zweimal, jeweils bei Überfahren einer Referenzmarke. Aus dem Abstand der (nebeneinander liegenden) Referenzmarken wird der codierte Referenzwert gebildet und somit der Bezug zur Bildung von absoluten Positionswerten hergestellt.

Dieser Wert wird bei der Berechnung des absoluten Positionswertes durch die Kunden-Softwareapplikation genauso behandelt wie ein gespeicherter Referenz-Positionswert im Fall von Messgeräten mit einer Referenzmarke. Der codierte Referenzwert entspricht somit ebenfalls dem Offset zwischen absolutem Positionswert und ausgegebenem (inkrementellem) Positionswert.

Das folgende Bild zeigt den prinzipiellen Ablauf bei der Ermittlung einer Referenzposition. Die angezeigten Werte sind nur als Beispiel zu verstehen und der Übersicht halber ist nur ein Ausschnitt des Positionswert-Registers gezeigt.



Anmerkungen:

Auch Bänder mit abstandscodierten Referenzmarken, die auf einem Ring montiert sind und somit an der Stoßstelle ein unregelmäßiges Referenzintervall haben können, werden richtig ausgewertet.

1.3.3 Ablauf der Referenzierung

Damit die Referenzierung einer Achse gestartet werden kann müssen folgende Voraussetzungen erfüllt sein:

- (1) Achse ist konfiguriert
- (2) Versorgungsspannung für die Achse ist aktiviert
- (3) Position muss gültig sein

Jetzt kann die Referenzfahrt durch ein Softwarekommando aktiviert werden. Die Referenzposition wird beim Überfahren des nächsten Referenzimpulses gespeichert. Ist die Achse für abstandscodierte Referenzmarken konfiguriert wird mit jedem der zwei folgenden Referenzimpulse ein Positionswert gespeichert. Die gespeicherten Werte entsprechen dem Stand des Periodenzählers bei der jeweiligen Referenzmarke.

Die Referenzfahrt kann durch ein Softwarekommando gestoppt werden. Wurden bereits Referenzimpulse überfahren, so bleiben die zugehörigen Positionswerte erhalten. Wird die Funktion Referenzfahrt starten erneut aufgerufen, werden die alten Referenzpositionswerte ungültig, was durch ein Bit im Status des Positionsdatenpakets gekennzeichnet wird.

Anmerkungen:

Die Referenzierung der Achse wird ebenfalls ungültig (Bit im Statuswort) wenn:

- die Versorgungsspannung der Achse ausgeschaltet wurde
- die Funktion "eib8_set_position_value" aufgerufen wurde
- die Position ungültig wurde

Es gibt unterschiedliche Möglichkeiten den Status der Referenzfahrt abzufragen:

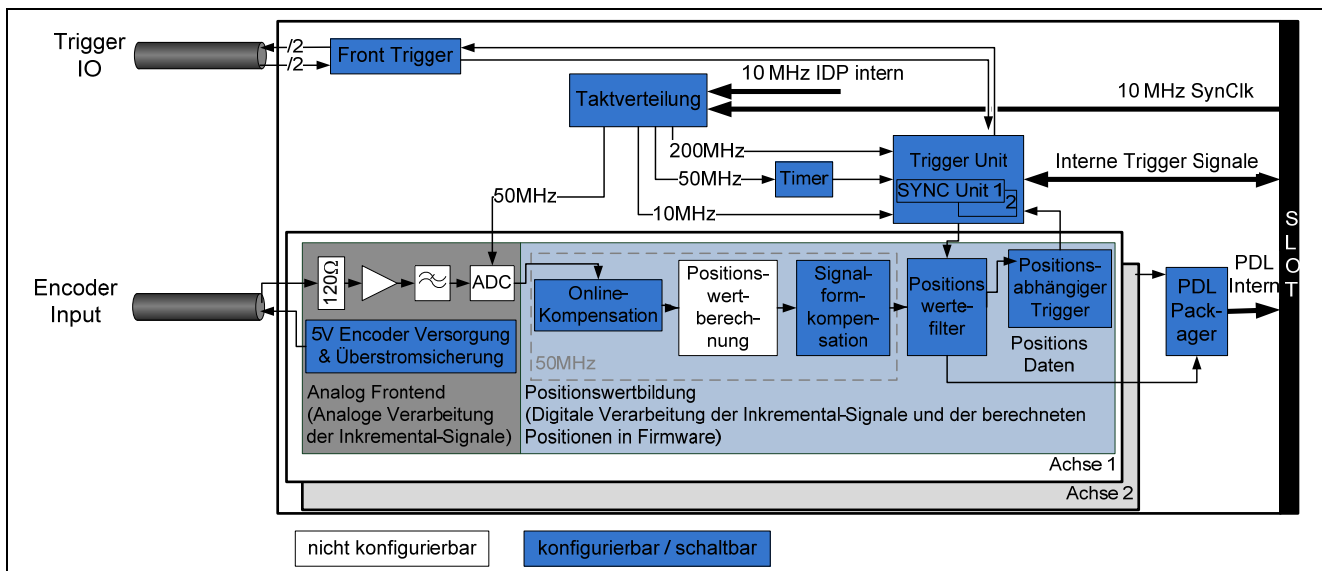
- Durch Abfrage des Parameters "ref_mark:valid" (siehe 2.6.3)
- Im Statuswort der Positionsdatenausgabe

Hier kann allerdings nur festgestellt werden ob eine gültige Referenzmarke gefunden wurde. Außerdem muss die Positionsdatenausgabe vgl. Kapitel "2.7 Positionsdaten PD" entsprechend konfiguriert werden. Bei abstandscodierten Messgeräten wird im Positionsdatenpaket "reference" Paket immer der codierte Referenzwert übertragen.

1.4 Positionswertbildung

Unter Positionswertbildung versteht man den gesamten Signalverarbeitungspfad, den die Abtastsignale eines Positionsmessgeräts für die Ermittlung des Positionswertes durchlaufen. Vom Analogpfad und nachfolgender AD-Wandlung über die Echtzeit-Positionswerteberechnung mit allen aktivierten Kompensations- und Korrekturverfahren hin zur abschließenden Positionswertermittlung im Positionswertfilter.

Blockschaltbild des Signalverarbeitungspfad



1.4.1 Online Kompensation

Die Online Kompensation ist ein automatisch mitlaufender Abgleich von Offset, Phase und Amplitude der abgetasteten Inkrementalsignale. Sie reduziert die Abweichungen innerhalb einer Signalperiode. Die Kompensation kann per Software ein- bzw. ausgeschaltet werden. Die Voraussetzungen für die Gültigkeit der Online Kompensation müssen eingehalten sein, u.a. Signalgrößen und Signalform.

Anmerkungen:

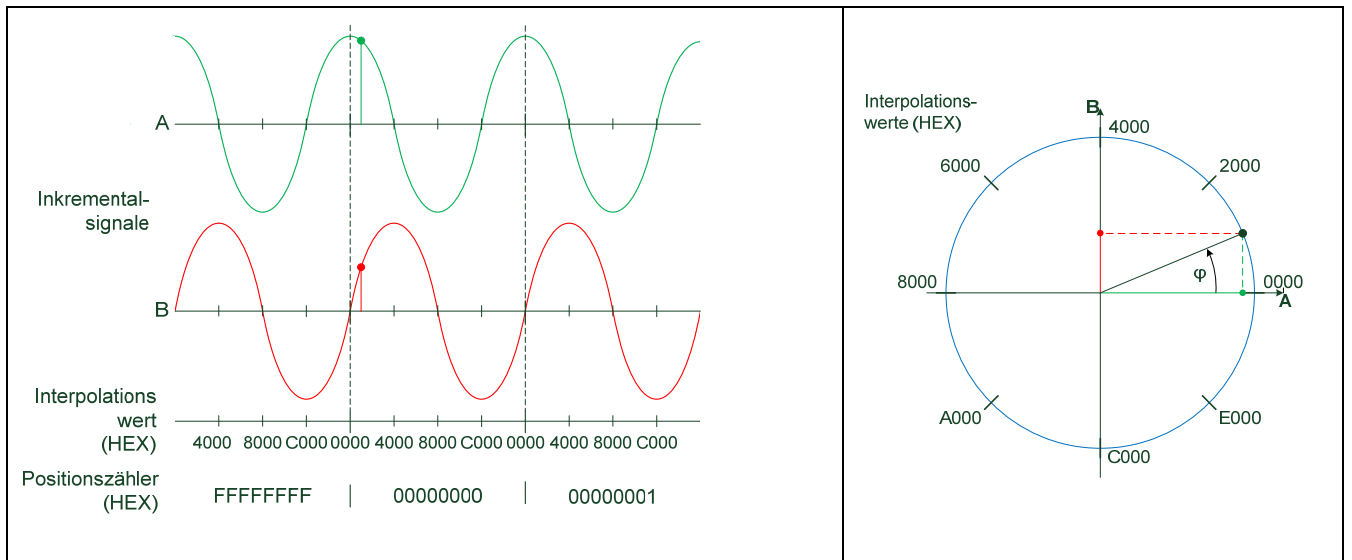
Beim erstmaligen Aktivieren der Versorgungsspannung der Achse wird die Online Kompensation standardmäßig aktiviert, außer sie wurde zuvor explizit deaktiviert. Eine Unterbrechung der Stromzufuhr der EIB aktiviert die Online Kompensation ebenfalls. Wenn die Versorgungsspannung der Achse deaktiviert wurde und erneut aktiviert wird, gilt für die Online Kompensation die zuletzt gültige Konfiguration (ein bzw. aus).

1.4.2 Positionswertberechnung

Zur Bildung des Positionswertes unterteilt die EIB 8791 die Signalperioden der Inkrementalsignale 65 536fach (16 Bit). Der Periodenzähler hat eine Breite von 32 Bit. Der Zählwert wird mit jeder Signalperiode des angeschlossenen Messgerätes um den Wert "1" erhöht oder erniedrigt.

Der Interpolationswert (16 Bit) bildet zusammen mit dem Wert des Periodenzählers (32 Bit) den 48 Bit breiten Positionswert zum Zeitpunkt des Triggerereignisses. Der Positionswert wird in einem 64 Bit breiten Register gespeichert (siehe 2.7.3.1 Position). Die übergeordnete Kunden-Softwareapplikation kann aus diesem Wert abhängig von der Art des Messgerätes (linear bzw. rotativ) die entsprechende Länge bzw. den Winkel berechnen. Der Überlauf des Periodenzählers hat keinen Einfluss auf die Funktionalität des Periodenzählers oder des Interpolators. Der Überlauf muss jedoch durch die übergeordnete Kunden-Softwareapplikation behandelt werden.

Zum Zeitpunkt des Trigger-Ereignisses werden die Inkrementalsignale abgetastet und daraus ein 16 Bit breiter Interpolationswert berechnet. Der Zusammenhang zwischen Interpolationswert und den Inkrementalsignalen (A, B) ergibt sich dabei wie folgt:



1.4.3 Signalformkompensation

Die Signalformkompensation korrigiert Auswirkungen von ortsabhängigen Signalformabweichungen der Abtastsignale eines Positionsmessgeräts auf den Positionswert.

Dabei kann ein beliebiger Abschnitt des Messbereichs des Positionsmessgeräts als der Bereich gewählt werden, in dem die Signalformkompensation während des Betriebs aktiv sein soll.

Für die Signalformkompensation wird eine „Landkarte“ der Signalformabweichungen über den zu kompensierenden Messbereich erstellt und für die betreffende Achse in einem dafür bereitgestellten, nichtflüchtigen Speicherbereich abgelegt.

Die Signalformabweichungen werden hierfür ortsabhängig aus den Abtastsignalen bestimmt und auf ein vorgegebenes Korrekturstützpunkteraster gemittelt, nachdem mit Hilfe eines dazugehörigen Korrekturlaufs (1.4.4) entsprechende Informationen über die Änderungen der Signaleigenschaften gewonnen wurden.

Auch während dieser Korrekturwertaufnahme darf das betreffende Positionsmessgerät als Positionswertgeber im geschlossenen Regelkreis der Antriebsachse für den Korrekturlauf betrieben werden.

Eine Kombination von Signalformkompensation und Onlinekompensation (1.4.1) ist zulässig und geeignet Positionsabweichungen bestmöglich zu reduzieren.

Entsprechend muss dann auch der Korrekturlauf mit aktivierter Onlinekompensation durchgeführt werden.

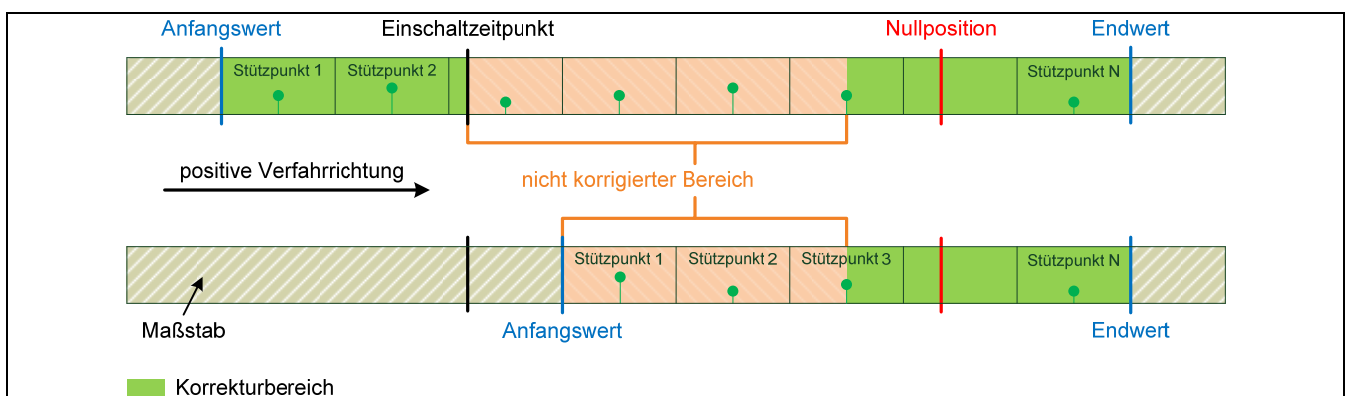
Anmerkung: Die Einstellung für die Online-Kompensation ("ein" bzw. "aus") muss beim Korrekturlauf und späterem Betrieb gleich eingestellt sein.

Liegt ein gültiger Satz an Korrekturwerten im Speicher der betreffenden Achse vor, können diese während des Betriebs des Positionsmessgeräts zu einer automatischen Korrektur der ortsabhängigen Signalformabweichungen herangezogen werden.

Damit die Signalformkompensation aktiviert werden kann müssen folgende Voraussetzungen erfüllt sein:

- (1) Achse ist konfiguriert und Versorgungsspannung für die Achse ist aktiviert
- (2) Achse ist referenziert und Position ist gültig
- (3) Gültige Korrekturwerte vorhanden (Ermittlung der Korrekturwerte siehe Kapitel "1.4.4 Korrekturwertaufnahme zur Signalformkompensation"). Anmerkung: Nur gültig wenn Korrekturlauf mit gleicher Konfiguration vorgenommen wurde.

Beim Funktionsaufruf (Signalformkompensation einschalten) darf die Achse in Bewegung sein und darf sich sowohl außerhalb als auch innerhalb des Korrekturbereiches befinden. Bevor die Positionswerte korrigiert werden können müssen die Korrekturwerte aus dem Speicher geladen werden. Dazu ist es notwendig, dass mindestens vier Korrekturstützpunkte in gleicher Bewegungsrichtung überfahren werden (nur nachdem die Signalformkompensation aktiviert wurde).



Es gibt abhängig von der Anzahl der Stützpunkte eine maximale Verfahrensgeschwindigkeit. Die daraus maximale Signalfrequenz lässt sich in der nachfolgenden Formel bestimmen. Wird diese Bedingung nicht eingehalten, wird eine entsprechende Fehlermeldung generiert und die Positionswerte werden unkorrigiert ausgegeben bleiben aber weiterhin gültig. Die maximal zulässige Verfahrensgeschwindigkeit ist abhängig vom Abstand der Korrekturstützpunkte. Wird die Achse wieder im zulässigen Geschwindigkeitsbereich betrieben kann der Positionswert wieder richtig korrigiert werden.

$$f_{Signal}^{max} = \frac{\text{Abstand der Korrekturstützpunkte in Signalperioden}}{0,2} \text{ [kHz]}$$

1.4.4 Korrekturwertaufnahme zur Signalformkompensation

Messdatenaufnahme für die selbsttätige Ermittlung von Korrekturwerten für die Signalformkompensation.

Bei der Signalformkompensation werden ortsabhängige Fehler (Signalformabweichungen) der Messgerätsignale (Abtastsignale), welche sich auf den Positionswert auswirken, korrigiert. Ermittelt werden diese Fehler (Abweichungen) in einem Korrekturlauf. Der Korrekturlauf kann mit oder ohne Online-Kompensation (ein- bzw. ausgeschaltet) erfolgen. Ein Slot kann nur jeweils für **eine** Achse zeitgleich die Korrekturwertaufnahme durchführen. Somit muss man, wenn beide Kanäle eines Slots korrigiert werden sollen, zwei Korrekturwertläufe durchführen. Es ist aber möglich mehrere Achsen auf unterschiedlichen Slots gleichzeitig für einen Korrekturlauf zu starten. Die Korrekturwertaufnahme kann in positiver und negativer Verfahrrichtung erfolgen.

1.4.4.1 Konfiguration der Korrekturwertaufnahme

Für jede Achse müssen folgende Parameter konfiguriert werden:

waveform_corr_run:start_value	= Anfangswert des Korrekturbereiches bezogen auf die Nullposition
waveform_corr_run:basic_frequency	= Signalgrundfrequenz (phi_1 = eine, phi_2 = zwei Signalperioden)
waveform_corr_run:direction	= Verfahrrichtung (negative, positive)
waveform_corr_run:number_of_correction_points	= Anzahl der Korrekturstützpunkte
waveform_corr_run:distance_of_correction_points	= Abstand der Korrekturstützpunkte in Signalperioden

Bei einer Signalgrundfrequenz phi_2 = zwei Signalperioden muss auch der Abstand der Korrekturstützpunkte in Signalperioden ein durch zwei teilbarer, also gerader Wert sein.

Mit den Parametern start_value, number_of_correction_points und distance_of_correction_points ist der Korrekturwertebereich bestimmt. Der Parameter start_value bezeichnet den Anfangswert des Korrekturbereichs. Bei positiver Verfahrrichtung ist es der Startwert, bei negativer der Endwert des Korrekturbereichs. Zur Veranschaulichung dienen folgende Beispiele.

Beispiele: Rotatives Messgerät

Achsenkonfiguration		
Messgeräte Typ	rotary encoder	
Anzahl der Inkremente	6000	
Referenzmarkentyp	single reference mark	
Konfiguration der Korrekturwertaufnahme		
start_value	+2250	+5250
Direction	positive	
number_of_correction_points	6	
distance_of_correction_points	500	
Korrekturbereich des Messgeräts		

Anmerkungen:

- Positive Drehrichtung für Inkrementalsignale gemäß Anschlussmaßzeichnung des Messgeräts
- bei rotativen Achsen ist der Anfangswert immer auf die positive Verfahrrichtung bezogen
- bei rotativen Achsen darf das Produkt aus Abstand und Anzahl der Stützpunkte nicht größer als die Anzahl der Inkremente pro Umdrehung sein

Achsenkonfiguration	
Messgeräte Typ	linear encoder
Referenzmarkentyp	single reference mark
Konfiguration der Korrekturwertaufnahme	
start_value	-750
number_of_correction_points	8
distance_of_correction_points	200
Korrekturbereich des Messgeräts	

Achsenkonfiguration	
Messgeräte Typ	linear encoder
Referenzmarkentyp	single reference mark
Konfiguration der Korrekturwertaufnahme	
start_value	+1500
Direction	positive
number_of_correction_points	5
distance_of_correction_points	1000
Korrekturbereich des Messgeräts	

Anmerkungen:

- Positive Bewegungsrichtung (Verfahrrichtung) für Inkrementalsignale gemäß Schnittstellen-Beschreibung (ausführliche Beschreibung findet man im Prospekt "Schnittstellen von HEIDENHAIN-Messgeräten")
- bei linearen Achsen muss der Anfangswert immer kleiner als der Endwert sein

1.4.4.2 Ablauf der Korrekturwertaufnahme

Damit die Korrekturwertaufnahme für **eine** Achse gestartet werden kann müssen folgende Voraussetzungen erfüllt sein:

- (1) Signalfrequenzkompensation auf beiden Achsen eines Slots ist ausgeschaltet
- (2) Korrekturwertaufnahme auf beiden Achsen eines Slots ist ausgeschaltet

Um die Korrekturwertaufnahme für **eine** Achse zu starten müssen folgende Schritte ausgeführt werden:

- (1) Achse konfigurieren
- (2) Versorgungsspannung für die Achse aktivieren
- (3) Evtl. Online Kompensation aktivieren
- (4) Korrekturwertaufnahme konfigurieren
- (5) Achse referenzieren
- (6) Abtastkopf, abhängig von der eingestellten Verfahrrichtung, mindestens zwei Signalperioden vor den Startwert bzw. nach den Endwert des Korrekturbereiches bewegen
- (7) Korrekturwertaufnahme mit der Funktion "eib8_exec_waveform_corr_run" starten

Beim Funktionsaufruf (Korrekturwertaufnahme starten) darf die Achse noch im Stillstand sein, muss aber dann am Beginn des Korrekturbereiches die richtige Geschwindigkeit erreicht haben.

Zulässig sind dabei Frequenzen der Abtastsignale in einem weiten Bereich von $125 \text{ Hz} \leq f_{\text{Signal}} \leq 31,25 \text{ kHz}$.

Um eine möglichst hohe Qualität der Korrekturwerte zu erhalten sollte die Signalfrequenz allerdings $f_{\text{Signal}} = 20 \text{ kHz}$ nicht deutlich überschreiten. Wird das Positionsmessgerät während des Korrekturlaufs als Positionswertgeber des geschlossenen Regelkreises der Antriebsachse betrieben, muss die Signalfrequenz des Messgeräts auf jeden Fall (Faktor ca. 5 – 10) oberhalb der Grenzfrequenz des Antriebssystems in Bezug auf sein Führungsübertragungsverhalten liegen.

Obwohl das Verfahren zur Bestimmung der Korrekturwerte gegenüber Geschwindigkeitsschwankungen äußerst tolerant ausgelegt ist, sind vor allem höherfrequente Beschleunigungen während der Korrekturwertaufnahme zu vermeiden.

Verfährt die Achse innerhalb des Korrekturbereiches mit zu hoher oder zu niedriger Geschwindigkeit wird die Korrekturwertaufnahme abgebrochen und der entsprechende Fehlerstatus gesetzt. Wenn der Abstand zum Korrekturbereich (Anfangswert) beim Aktivieren der Korrekturwertaufnahme nicht mindestens zwei Signalperioden groß war wird die Aufnahme ebenfalls abgebrochen. Nachdem der komplette Korrekturbereich überfahren wurde („`waveform_corr_status:data_acquisition_done`“ = 1) kann die Achsbewegung gestoppt werden. Nun bereitet der Slot die Abtastwerte rechnerisch auf, die Berechnung der Korrekturwerte kann je nach Konfiguration bis zu 03:00 (Minuten: Sekunden) dauern. Während dieser Zeit ist `waveform_corr_status:running` gesetzt. Bei der Berechnung können diejenigen Abweichungen der Korrekturwerte detektiert werden, die zum Beispiel auf zu hohe Geschwindigkeitsschwankungen bei Störungen durch Vibrationen zurückzuführen sind. Sollten hierbei die Auswirkungen auf die Korrekturwerte als zu hoch festgestellt werden, wird die Korrekturwertberechnung abgebrochen und `waveform_corr_status:failed` gesetzt. Verläuft die Korrekturwertaufnahme und Berechnung fehlerfrei, so werden zum Schluss die Korrekturwerte im internen Speicher abgelegt. Der Ablauf und die dazu aufzurufenden Funktionen sind im Teil II des Benutzerhandbuches beschrieben.

1.4.5 Positionswertefilter

Konfigurierbarer Filter zur Ermittlung des Positionswertes zum effektiven Messzeitpunkt aus einer Anzahl von Rohpositionen.

Durch die hohe interne Abtastrate von 50 MHz wird empfohlen die Positionswerte vor ihrer Ausgabe noch digital zu filtern.

Dazu wird die Dynamik der Positionswerte zu Gunsten einer Erhöhung der Auflösung der Positionsinformation reduziert.

Gleichzeitig kann auch das Alter der Positionswerte (Data-Age) relativ zum Zeitpunkt der externen Positionsabfrage (Trigger) in weiten Grenzen voreingestellt werden.

Es werden im Filter neben den Positionswerten zusätzlich auch Geschwindigkeiten und/oder Beschleunigungswerte bestimmt und ausgegeben.

Bandbreite:	Legt die Anzahl der, bei der Positionswertemittelung berücksichtigten Rohpositionen fest.
Effektiver Messzeitpunkt:	Alter des Positionswertes relativ zum Zeitpunkt der Positionsabfrage (Trigger)
Charakteristik:	Wahl eines Filters mit linearer (Geschwindigkeit wird berücksichtigt) oder parabolischer (Geschwindigkeit und Beschleunigung wird berücksichtigt) Charakteristik.

1.5 Positionsdaten-Ausgabe

In der EIB 8791 werden die **PositionsDaten (PD)** aufgrund von Trigger-Ereignissen in den Slots 1 bis SLOT 4 generiert. Intern werden diese Positionsdaten über einen **PositionsDatenLink (PDL)** verarbeitet.

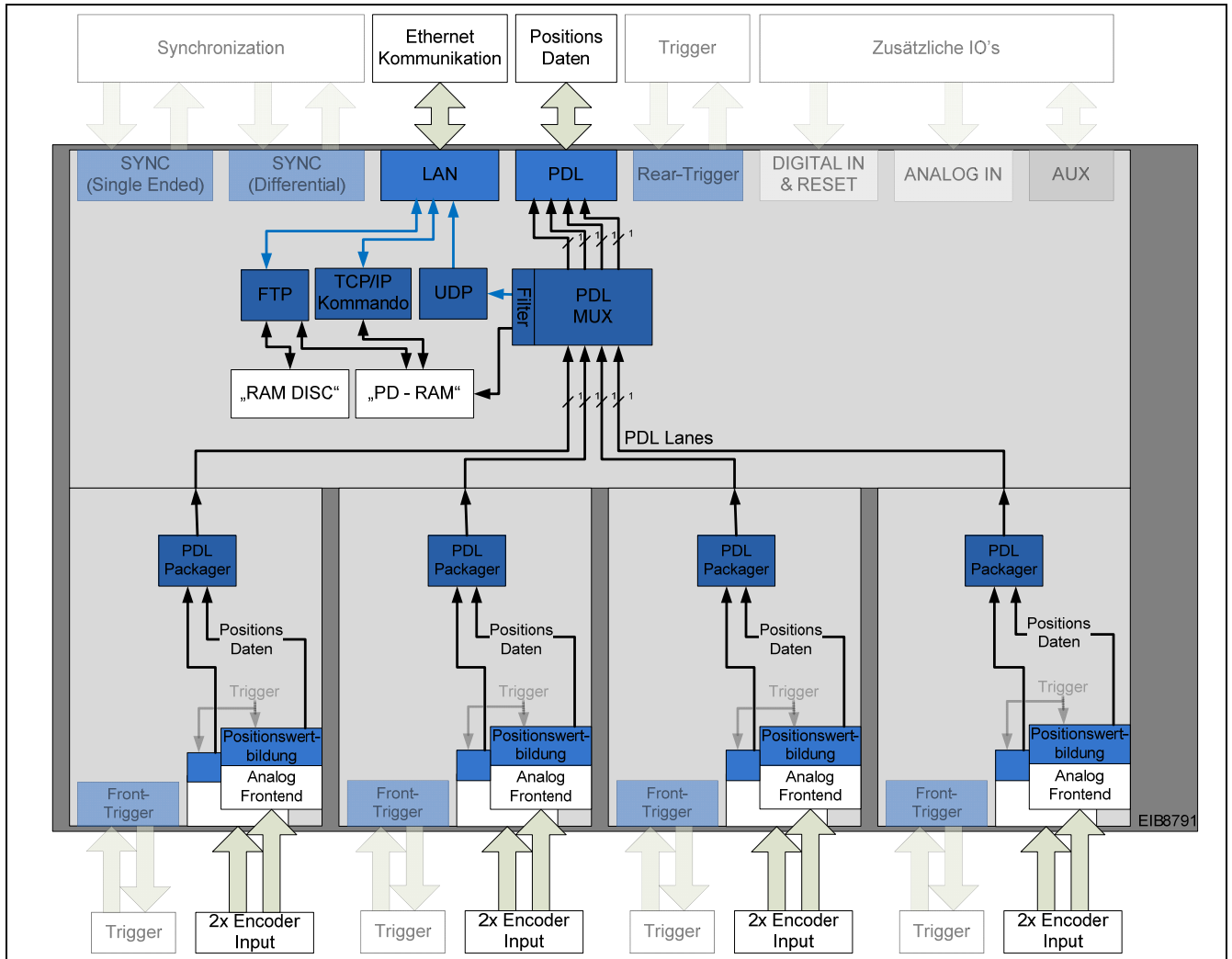
Der PDL ist als Serielle Schnittstelle (full-duplex) für Echtzeitkommunikation mit minimalen Latenzzeiten entwickelt und auch als externe Schnittstelle vorhanden.

Der Anwender hat zwei Möglichkeiten diese Echtzeit-Positionsdaten auszugeben. Als Schnittstelle steht dafür der PDL und die LAN Verbindung zur Verfügung. Über die LAN Schnittstelle können wiederum sowohl Echtzeitdaten über UDP geschickt werden, als auch in einem internen Speicher Aufgezeichnete Positionsdaten (über TCP/IP oder FTP) ausgelesen werden.

Echtzeit-Positionsdaten können über folgende Methoden verarbeitet bzw. ausgegeben werden:

- ➔ LAN Schnittstelle
 - Polling über TCP/IP Kommando
 - UDP Modus
 - Batched Streaming (Soft Realtime)
 - Direct Streaming (Soft Realtime)
 - RAM Recording
 - Single Shot
 - Auslesen über TCP/IP
 - Auslesen über FTP
 - „Rolling“ Betrieb
 - Auslesen über TCP/IP
 - Auslesen über FTP
- ➔ PDL Schnittstelle
 - PDL Schnittstelle mit Proprietären PDL Protokoll (für höchste Übertragungsraten)

Prinzipschaltbild:



1.5.1 Positionspakete über TCP/IP Kommandolink

Es ist bei der EIB 8791 möglich Positionsdaten über ein Kommando auszulesen. Dieser Betrieb wird als Polling bezeichnet. Die Übertragung erfolgt per TCP/IP und ist daher nicht echtzeitfähig.

Konfiguration der Polling Betriebs → Siehe Kapitel: 2.8.5 Polling von Positionsdaten

1.5.2 Positionsdatenausgabe per UDP

Interne Positionsdaten, können von der EIB 8791 auf der LAN Schnittstelle über UDP ausgegeben werden. Dabei können bestimmte Paketnummern gefiltert werden. D.h. es werden nicht zwingend alle Pakete, die über PDL übertragen werden auch über UDP ausgegeben. Es besteht die Möglichkeit einzelne Positionswerte über UDP zu übertragen, wobei die Modi "direct_soft_realtime" und "batch_soft_realtime" unterstützt werden; nähere Angaben dazu siehe Kapitel 2.8.6.1.

Eine Positionsdatenübertragung per UDP ist generell nicht gegen Verlust von Daten geschützt. Es ist daher auf der Host-Seite zu prüfen, ob keine Unterbrechungen in der Übertragung waren bzw. Paketdaten verloren gegangen sind. Dies kann durch Prüfung der Framecounter gemacht werden. Dieser 8Bit Zähler muss fortlaufend hochgezählt werden. Um eine Eindeutige Zuordnung der Daten zu den entsprechenden Achsen zu bekommen sind die Paketnummern eindeutig zu setzen.

Die maximal mögliche (nutzbare) UDP Paketrate hängt meist von der empfangenden Gegenstelle (Host-Rechner) ab. Mit der EIB 8791 können UDP Paketraten von bis zu 400kHz nachgewiesen werden.

Konfiguration des UDP Betriebs → Siehe: „2.8.6 UDP Empfang von Positionsdaten“

1.5.3 Speichern von Positionspaketen auf der EIB 8791 (Recording)

Der Recording-Betrieb ist als eine Unterfunktion des UDP Modus zu sehen. Dabei können interne Positionsdaten von der EIB 8791 in einen internen Speicher abgelegt werden. Es können wie bei der UDP Ausgabe bestimmte Paketnummern gefiltert werden. D.h. es werden nicht zwingend alle Pakete, die über PDL übertragen werden auch im Speicher abgelegt.

Bei der Aufzeichnung können maximal 10 Mio. Positionswerte abgespeichert werden. Werden alle acht Achsen der EIB 8791 aufgezeichnet, so ergibt sich eine maximale Aufzeichnungstiefe von 1,25 Mio Pakete pro Achse.

Die maximale Aufzeichnungsdatenrate beträgt 240 MB/s. Ein Positionspaket ist 12 Byte lang. Es können also bis zu 2 Achsen mit einer maximalen Rate von 10MHz aufgezeichnet werden, oder alle acht Achsen mit bis zu 2,5 MHz. Werden mehrere Pakete pro Achse generiert, können diese mit einer entsprechend reduzierten Rate aufgezeichnet werden. Der Anwender ist selbst dafür verantwortlich, dass die maximale Datenrate zum Recorden eingehalten wird.

Das Auslesen der Recording Speichers (PD-RAM) erfolgt über TCP/IP. Es werden dabei über ein Kommando 1024 Byte große Datenpakete aus dem Speicher ausgelesen und übertragen. Diese Abfrage muss so oft wiederholt werden, bis der Recording-Speicher komplett übertragen wurde.

Auslesen über TCP/IP → Siehe: „2.8.6.3 Lesen der Positionsdaten aus dem Positionsdatenpuffer“

Als Alternative kann auch per FTP das Auslesen erfolgen. Dabei ist aber ein Zwischenschritt notwendig, der die maximale Speichertiefe begrenzt. Die Positionsdaten aus dem PD-RAM müssen hierfür erst in die sog. „RAM-Disc“ kopiert werden. Hier werden sie als File abgelegt und können per einfachem FTP Zugriff abgerufen werden.

Speichern in RAM-Disc → Siehe: „2.8.6.4 Schreiben von Positionsdaten in Datei“

FTP Datei Transfer → Siehe: „1.12 FTP Server der EIB 8791“

1.5.4 Positionsdaten über Position Data Link

Der Positionsdatenlink ist für serielle Echtzeitdatenübertragung entwickelt worden. Diese serielle Schnittstelle mit einem proprietären Protokoll ist optimiert für die Übertragung kleiner Informationseinheiten (Positionsdaten) mit geringer Latenz und minimalem Overhead. Die Kommunikationstopologie erfordert eine Punkt-zu-Punkt Verbindung.

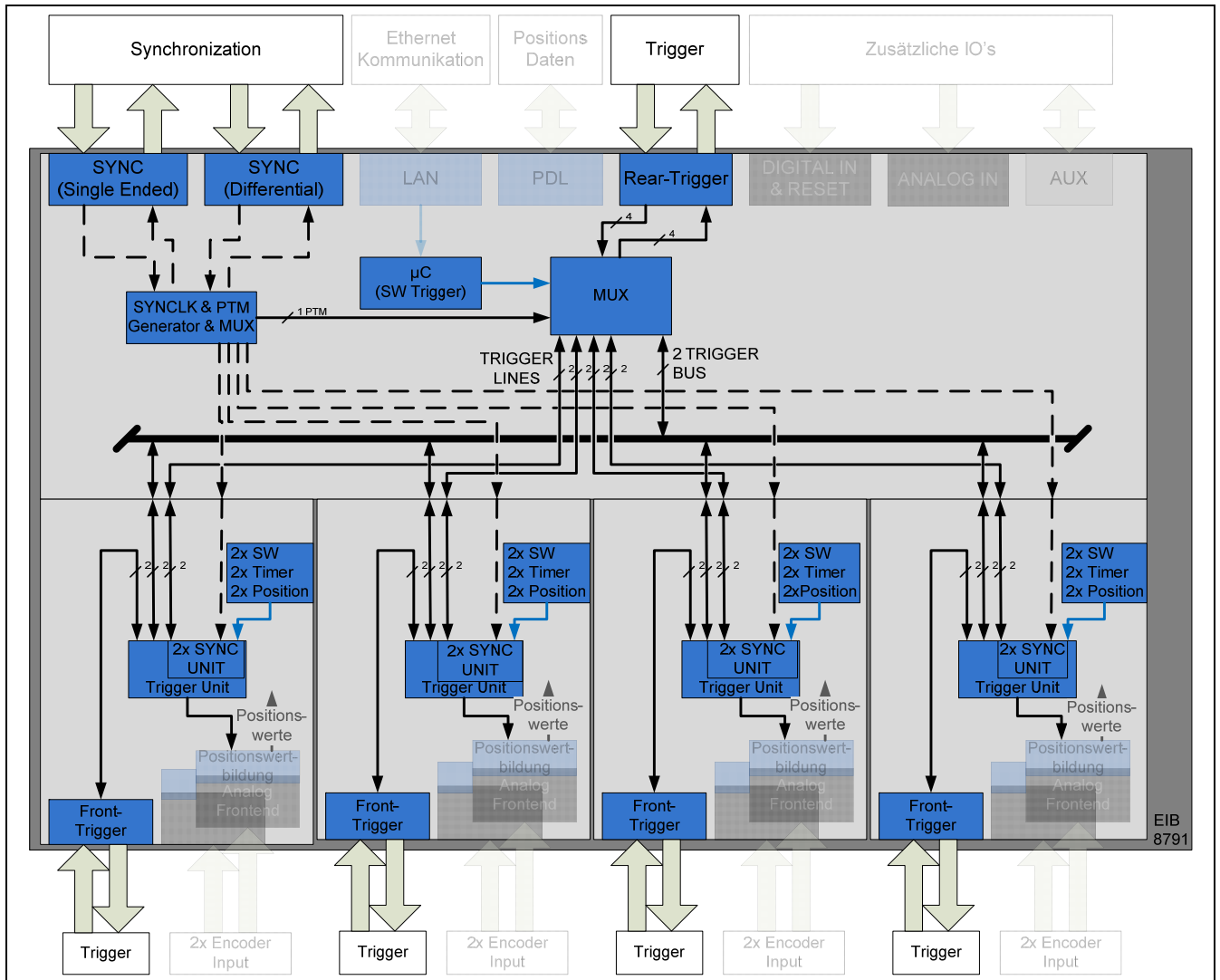
Der Position Data Link ist ein kontinuierlicher serieller Datenstrom. Die Pegel, 8B/10B Kodierung und Datenraten sind identisch zur seriellen Datenübertragung bei PCI -Express (PCIe 1.1, Gen1)

Für weitere Informationen wenden Sie sich bitte an den Messgeräte-Support von HEIDENHAIN.

1.6 Synchronisation und asynchrone Triggerung

Die EIB 8791 bietet die Möglichkeit synchron oder asynchron Positionsdaten zu generieren. Die Triggerung der Positionsdaten kann also in einem festen zeitlichen Raster, gebildet aus einem genauen Synchronisationstakt (SynClk) und einem „Position TriggerMarker“ (PTM), erfolgen. Alternativ können Positionsdaten durch Triggerung auf asynchrone Ereignisse gebildet werden. Es sollte vermieden werden, die beiden Betriebsmodi zu mischen.

Prinzipschaltbild der Synchronisations- und Trigger-Ressourcen:



1.6.1 Asynchroner Betrieb mit Trigger

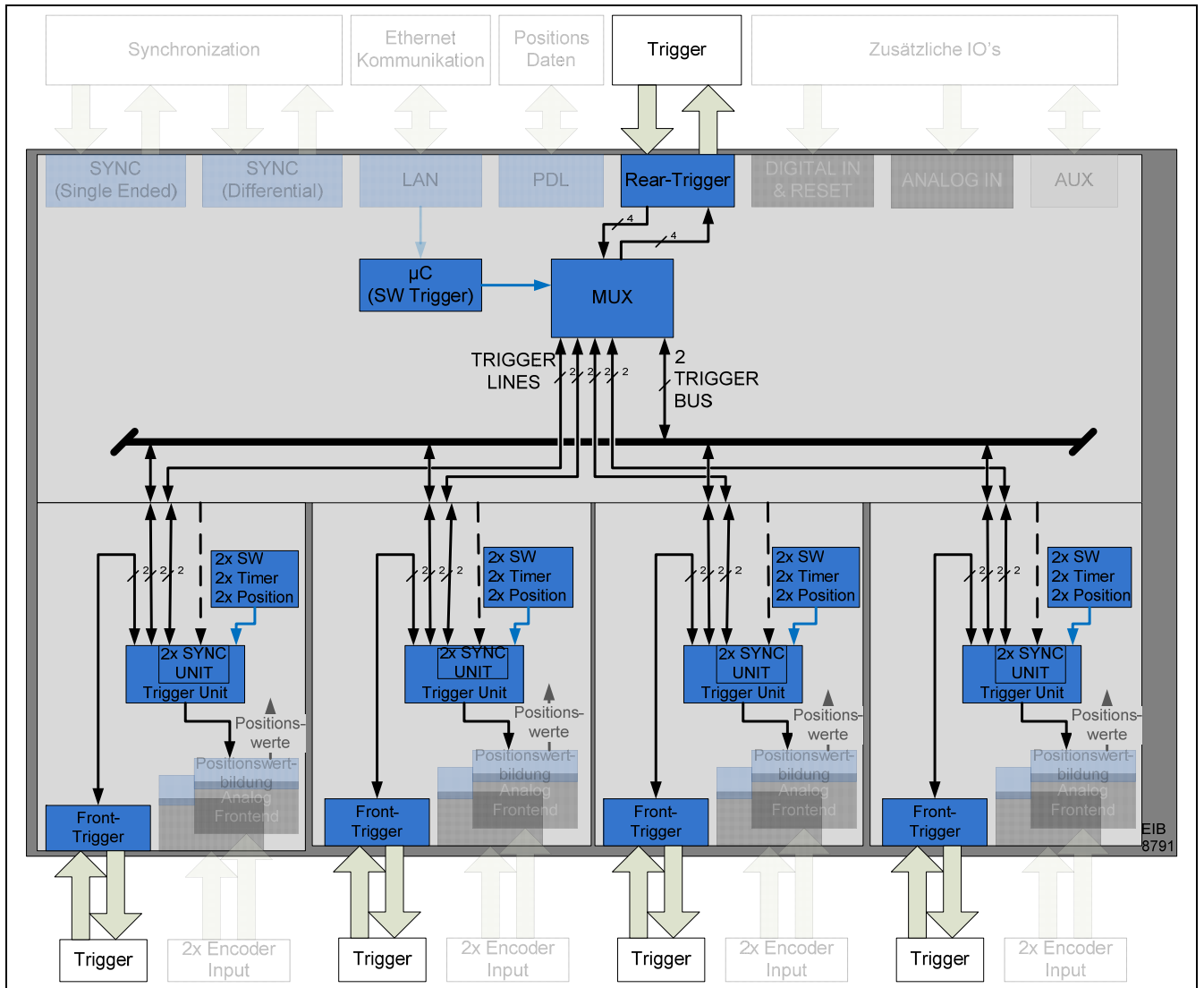
Es stehen bei der EIB 8791 insgesamt 12 asynchrone Triggereingänge und 12 asynchrone Triggereingänge zur Verfügung. Diese sind aufgeteilt in je 4 Trigger IOs auf der Rückseite und 8 Trigger IOs auf der Gerätevorderseite.

Prinzipiell kann jede Achse durch beliebige Trigger- IOs getriggert werden. Beschränkungen bestehen nur in der Auslastung der internen Trigger-Ressourcen. Werden mehrere Eingänge eines Multiplexers auf ein und denselben Ausgang geschaltet, so werden die Eingangssignale „ODER-Verknüpft“. Es stehen pro Slot (mit je 2 Achsen) je 2 Trigger Lanes (Rx und Tx getrennt, daher full-duplex) und 2 Trigger Busse (half-duplex) zur Verfügung. Trigger Lanes sind Punkt zu Punkt Verbindungen zwischen der EIB 8791 Basiskomponente und den Slots. Die Trigger Busse sind Verbindungen zwischen den Slots und der EIB 8791 Basiskomponente, auf die nur ein Busteilnehmer als Quelle geschaltet werden darf, aber beliebig viele Teilnehmer empfangen können.

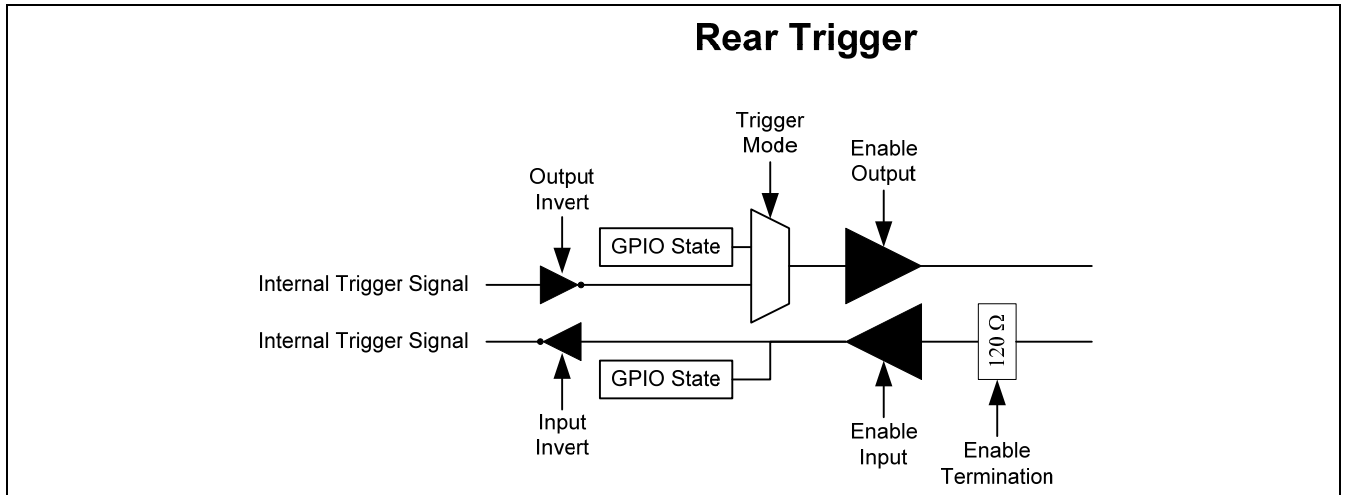
Jede Komponente (EIB 8791 Basiskomponente und SLOT 1 ... 4) hat einen eigenen Trigger-multiplexer bzw. Trigger-Unit, in denen die Verschaltung der eingezeichneten Eingänge und Ausgänge erfolgt.

Konfiguration der Trigger -> siehe: "2.9.2 Konfiguration der Trigger"

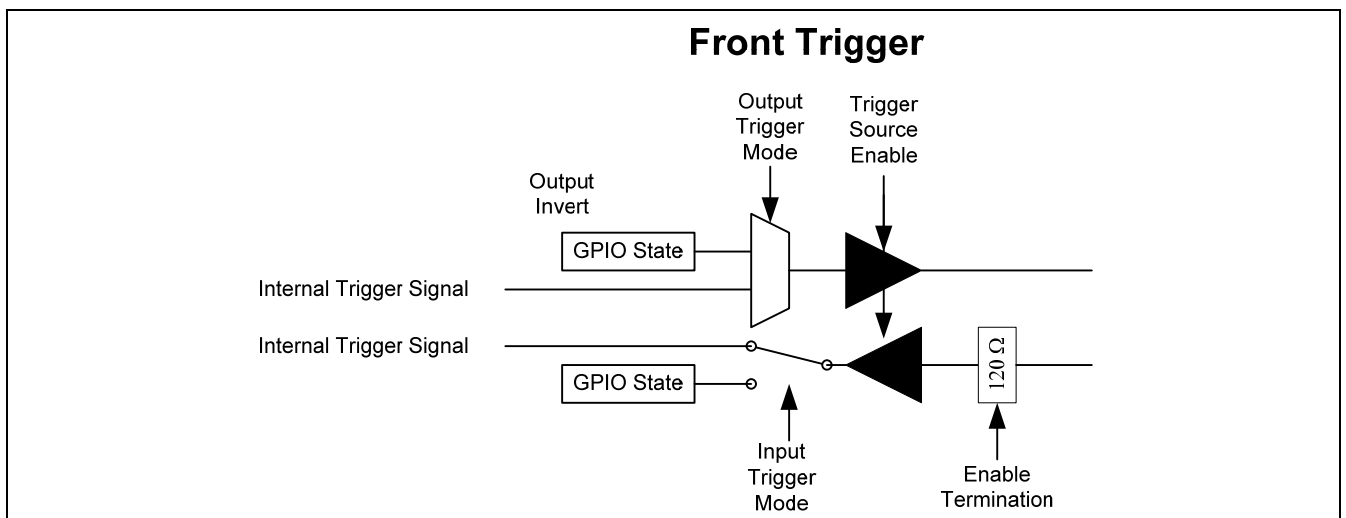
Prinzipschaltbild asynchrone Trigger-Ressourcen:



Blockschaltbild Rear-Trigger IO (4x vorhanden):



Blockschaltbild Front-Trigger IO (8x vorhanden):



Die dargestellten Trigger IOs (Front und Rear) können alternativ auch als GPIO verwendet werden.

1.6.2 Synchroner Betrieb mit SynClk & PTM

Im synchronen Betrieb wird die EIB 8791 auf ein externes Taktsignal synchronisiert. Der PTM ist ein logisches Signal, das dabei steigende Taktflanken des SynClk markiert, zu denen ein Positionswert ermittelt und ausgegeben werden soll.

1.6.2.1 SynClk Takte (Intern und Extern)

Als SynClk dient immer ein 10 MHz Takt, der entweder in der EIB 8791 selbst generiert werden kann, oder über eine Schnittstelle der EIB 8791 zur Verfügung gestellt wird. Als Interface stehen sowohl eine Single-Ended LVTTTL Schnittstelle mit SMA Stecker, als auch eine differenzielle „SYNC“-Schnittstelle, die den SynClk und PTM in Form von LVDS Signalen beinhaltet, zur Verfügung.

Dieser Synchronisationstakt kann auch ausgegeben und zur Synchronisierung weiterer Geräte verwendet werden.

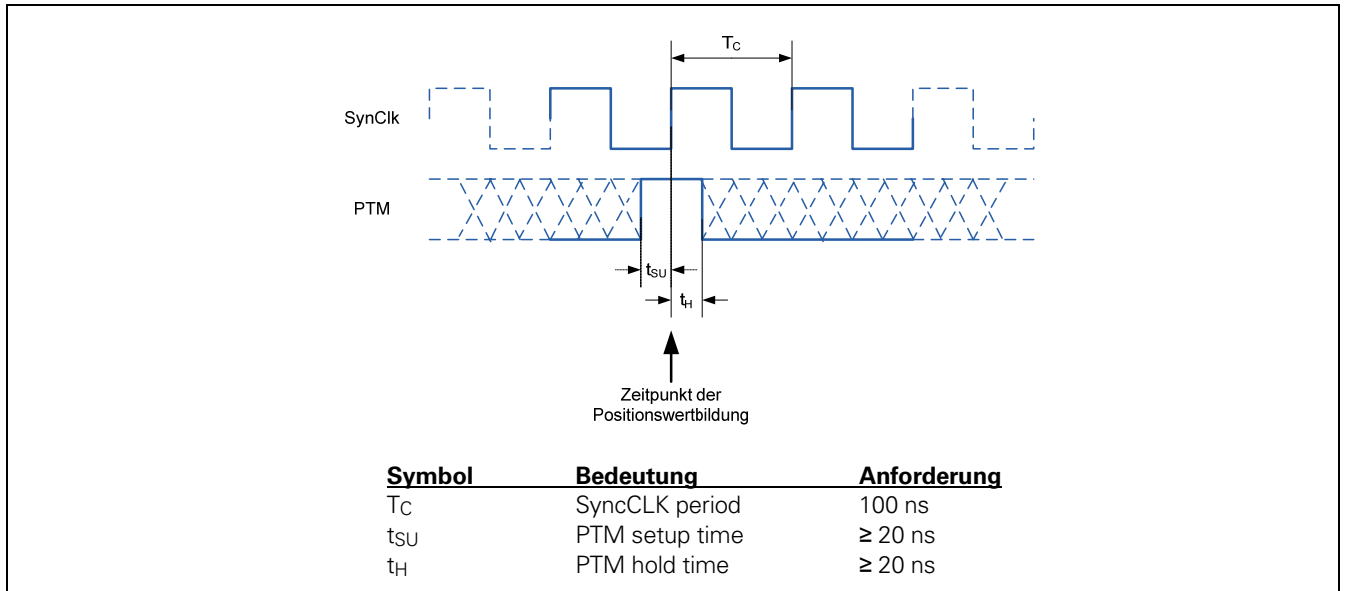
1.6.2.2 PTM – Position Trigger Marker (Intern & Extern)

Als PTM dient ein Signal, das eine steigende Flanke des SynClk markiert. Dieser PTM kann entweder intern generiert werden (nur sinnvoll wenn auch SynClk intern generiert wird) oder über eine Schnittstelle der EIB 8791 zur Verfügung gestellt werden. Als Interface stehen sowohl eine Single-Ended LVTTTL Schnittstelle mit SMA Stecker, als auch eine differenzielle „SYNC“-Schnittstelle, die den SynClk und PTM in Form von LVDS Signalen beinhaltet, zur Verfügung. Im Regelfall ist der PTM ein periodischer Puls, der die Samplerate der Positionswertausgabe definiert. Der genaue Zeitpunkt des Samples ist aber von der Flanke des SynClk die durch den PTM markiert wurde abhängig!

→ Jitter des PTM Signals wirken sich auf die Abtastung nicht aus, sofern die Timinganforderungen eingehalten werden!

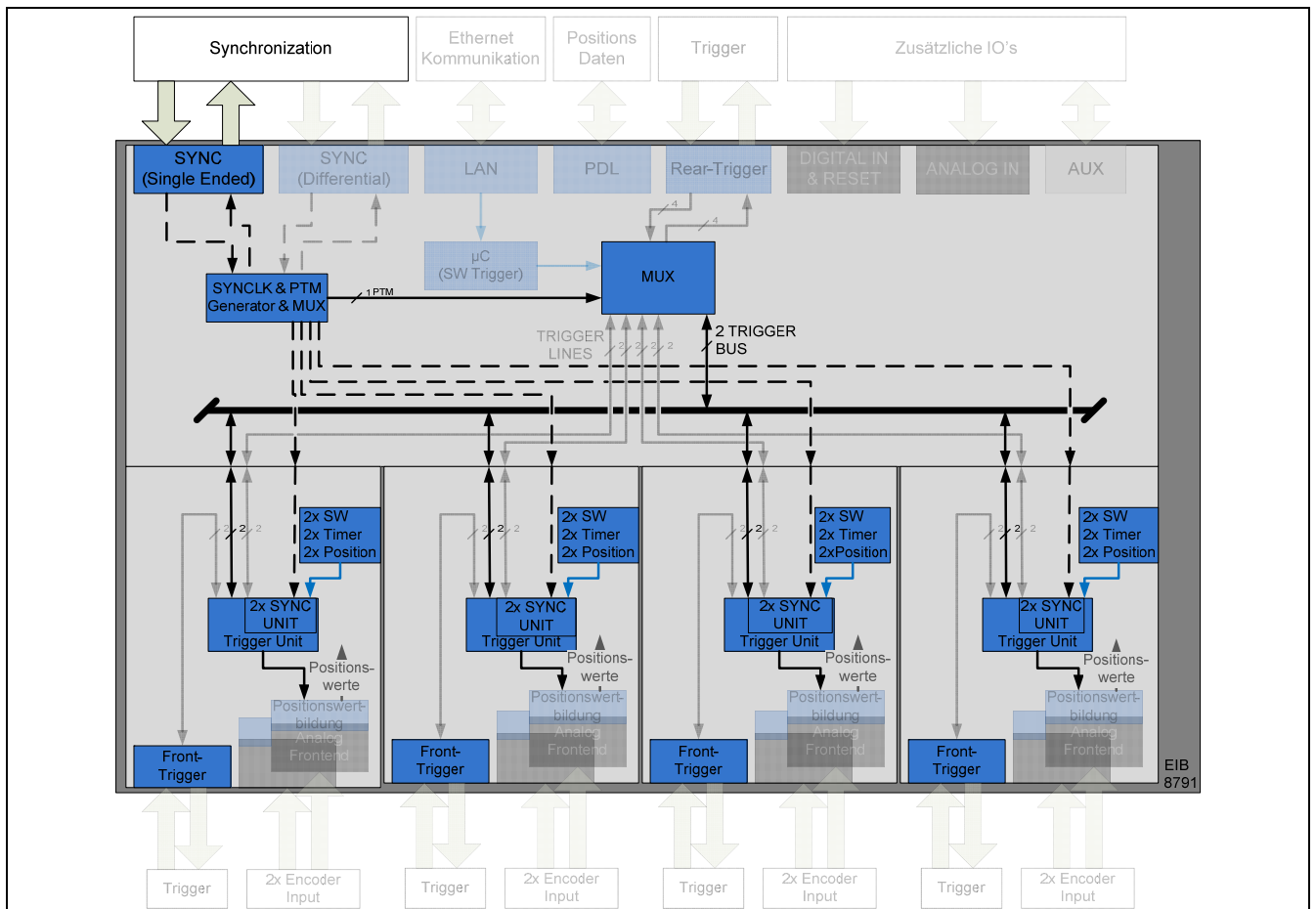
Der PTM kann auch ausgegeben und zur Synchronisierung weiterer Geräte verwendet werden.

Das PTM Signal muss bezüglich des SynClk folgende Timinganforderungen einhalten:

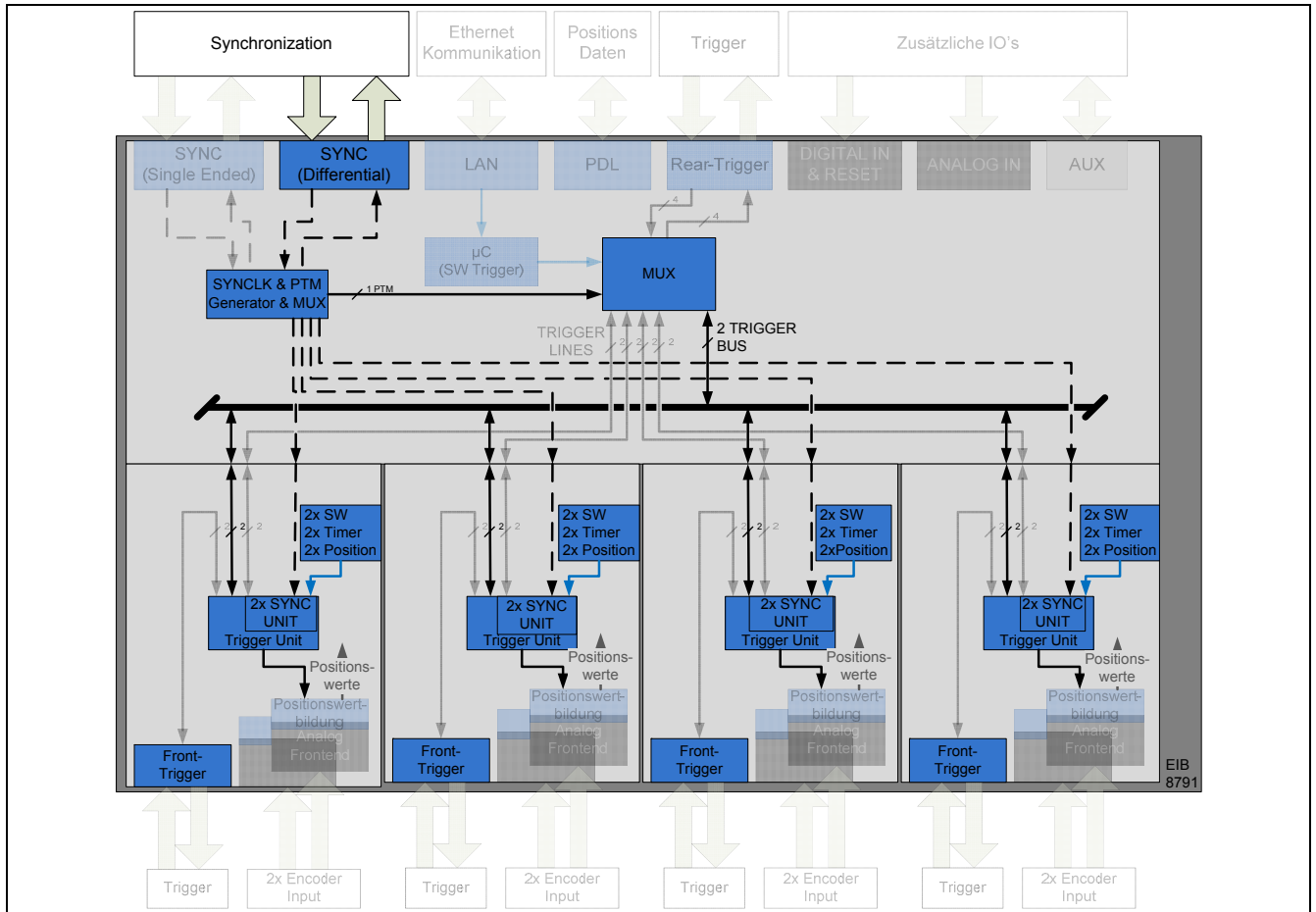


Intern ist der PTM über die zur Verfügung stehenden TriggerLines und Triggerbusse zu den Slots zu routen.

Prinzipschaltbild der Synchronisations-Ressourcen (Single Ended):

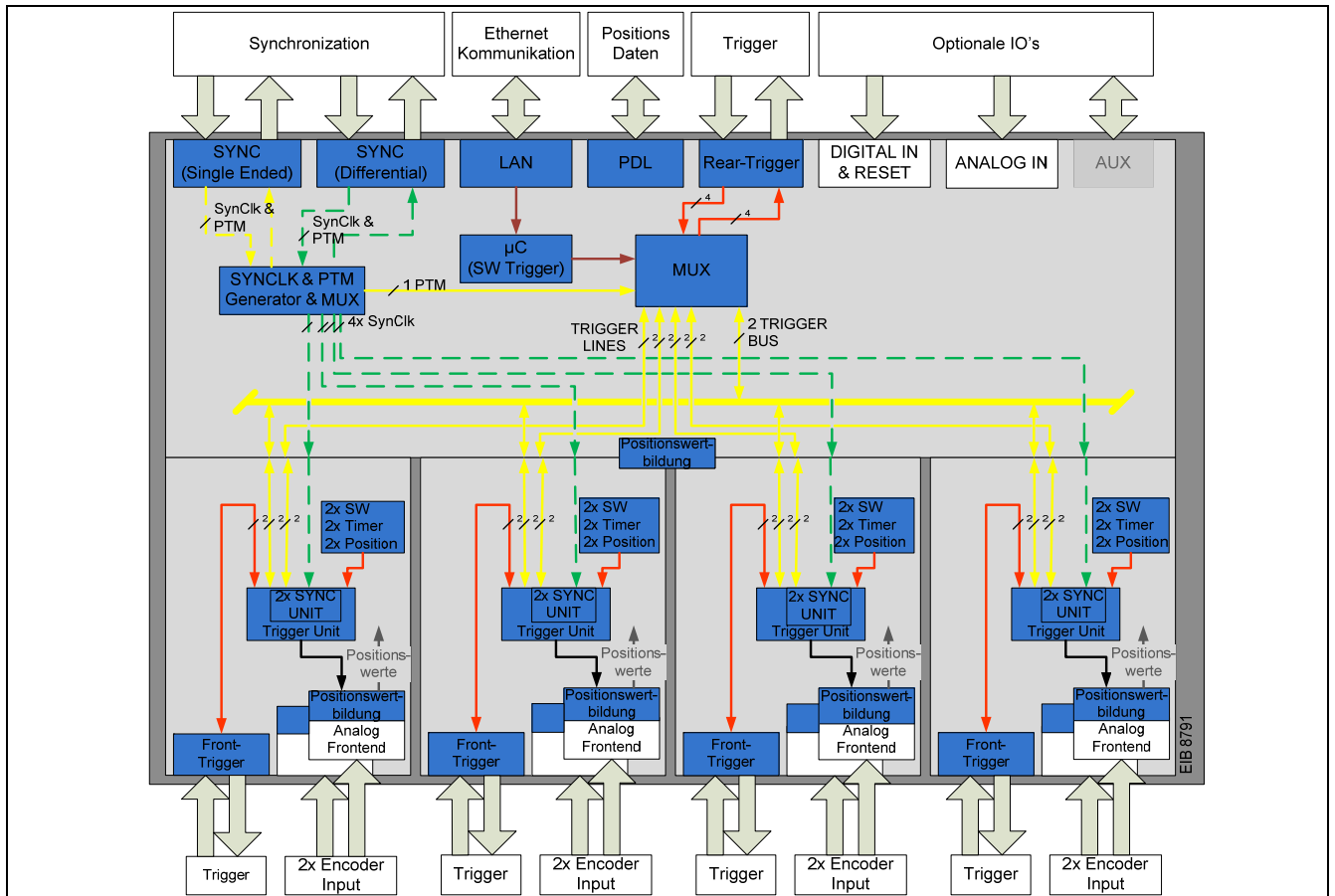


Prinzipschaltbild der Synchronisations-Ressourcen (Differenziell):



Dabei ist zu berücksichtigen, dass die Signalpfade unterschiedliche zeitliche Performance bieten:

Überblick über die Synchronisations- und asynchrone-Trigger Performance:



- Braun:** Geringste Performance der Zeitlichen Genauigkeit
- Rot:** Performance der Genauigkeit (~1ns)
- Gelb:** Performance der Genauigkeit (<1ns)
- Grün:** Beste Performance der Genauigkeit (<50ps)

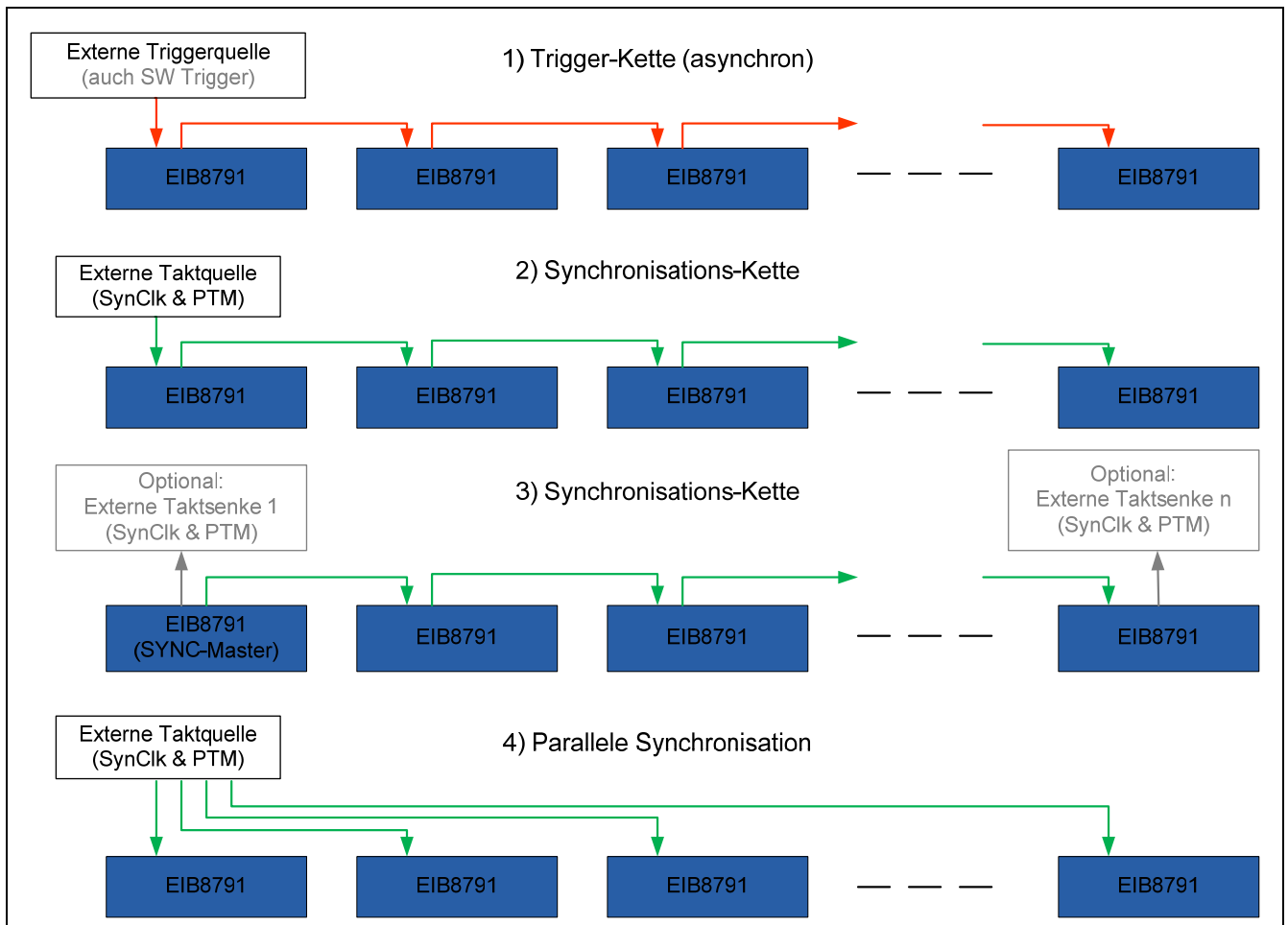
1.6.3 „Synchronisation“ mehrere EIB 8791

Die EIB 8791 kann über verschiedene Arten synchronisiert bzw. getriggert werden.

Es werden für allgemeine Anwendungen verschiedene „Basiskonfigurationen“ empfohlen: „2.9.2 Konfiguration der Trigger“

Werden mehrere Geräte verwendet, lassen sich auch Trigger- bzw. Synchronisationsketten aufbauen. Dabei ist je nach Anwendung die resultierende Performance zu berücksichtigen.

Prinzipschaltbild verschiedener Trigger- oder Synchronisations-Ketten:



- 1) Asynchrone Triggerung ohne Taktsignal
 - a. „Propagation Delay“ pro Gerät: ca. 70 ns
 - b. Drift von bis zu 4 ns pro Gerät
 - c. Hoher additiver Jitter
- 2) Synchronisationskette auf externe Taktquelle
 - a. „Propagation Delay“ pro Geräten < 10 ns bei Single ended bzw. < 1 ns bei Differenziell
 - b. Drift von < 1 ns (Single ended) bzw. < 50 ps (Differenziell) pro Gerät
 - c. Geringer additiver Jitter
- 3) Synchronisationskette auf eine „Master-EIB“
 - a. „Propagation Delay“ pro Geräten < 10 ns bei Single ended bzw. < 1 ns bei Differenziell
 - b. Drift von < 1 ns (Single ended) bzw. < 50 ps (Differenziell) pro Gerät
 - c. Geringer additiver Jitter
- 4) Parallele Synchronisation
 - a. Kein „Propagation Delay“
 - b. Drift von < 1 ns (Single ended) bzw. < 50 ps (Differenziell) pro Gerät
 - c. Minimaler additiver Jitter

1.7 Digitale und Analoge Hilfeingänge

Die EIB 8791 bietet über zusätzliche Hilfeingänge die Möglichkeit, acht digitale und vier analoge Sensoren einzulesen. Der Status dieser Eingänge bzw. der analoge Wert kann ausschließlich per TCP/IP Kommando ausgelesen werden.

1.7.1 Digital IN

Die Abfragerate ist vom TCP/IP Kommandointerface abhängig und kann bis zu ca. 50 Hz erreichen.

Anschlussbelegung und elektrische Kennwerte siehe „Betriebsanleitung“

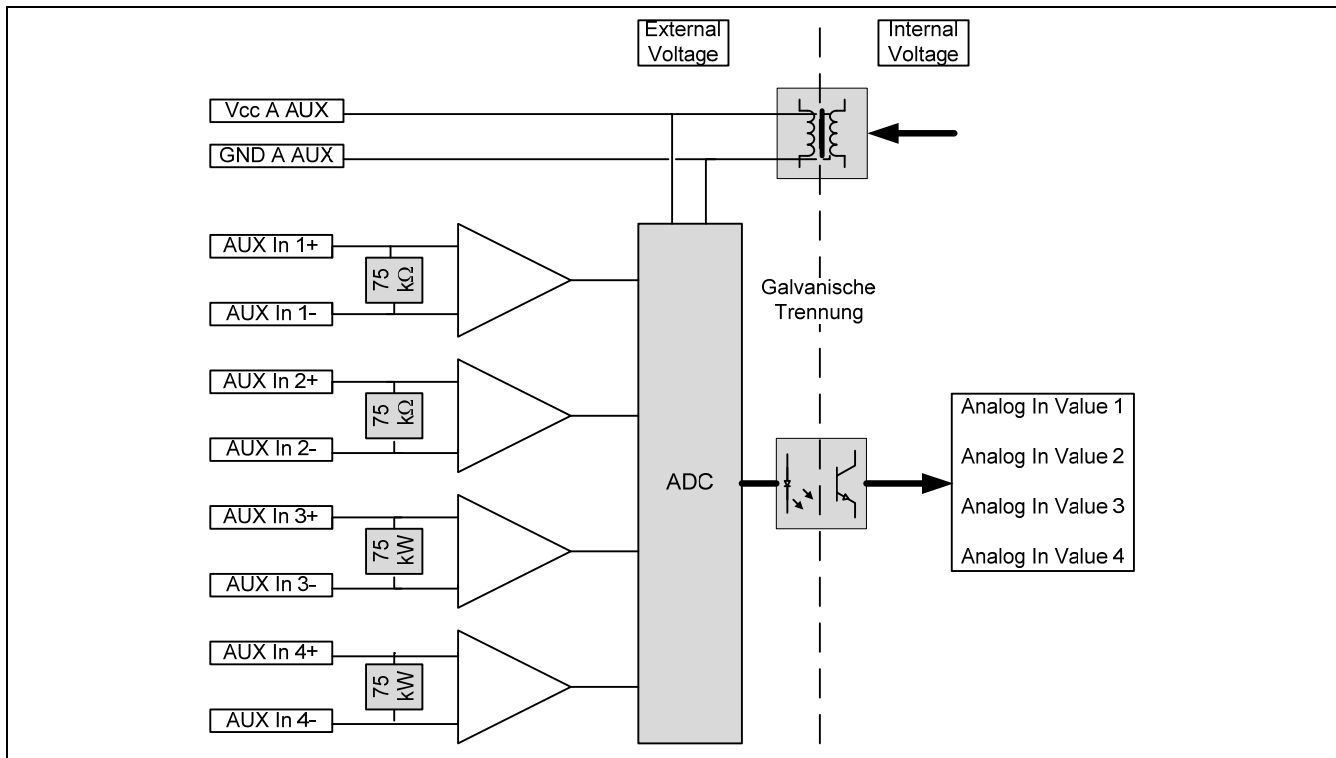
1.7.2 Analog IN

Am „ANALOG IN“ Anschluss können Standard Messumformer angeschlossen werden, die ein Spannungssignal im Bereich zwischen -10 V und +10 V liefern. Die digitale Auflösung beträgt 14 Bit mit einer internen Abtastrate von 2 Samples/s. Der Anschluss „ANALOG IN“ ist gegenüber anderen Anschlüssen galvanisch getrennt. Eine interne Trennung zwischen den vier Kanälen ist nicht vorhanden. Die differenzielle Eingangsimpedanz beträgt typisch 75 kΩ (min. 40 kΩ). Der ausgelesene Rohwert des ADC (Analog In Value [n]) ist als unsigned integer mit 32 Bit zu interpretieren.

Der Analoge Spannungswert ist wie folgt zu berechnen:

$$\text{Analoge Spannung [V]} = \frac{\text{Analog In Value [n]} - 2^{13}}{2^{13}} * 10,24 \text{ V}$$

Prinzipschaltbild des Analog Eingangs:



Anschlussbelegung und elektrische Kennwerte siehe „Betriebsanleitung“

1.7.3 AUX IO

Der Anschluss AUX wird nicht unterstützt und darf nicht verwendet werden.

1.8 Events, Selbsttest und Diagnostik

Die EIB 8791 hat verschiedene Möglichkeiten auftretende Fehler zu erkennen. Beim Einschalten des Gerätes wird automatisch ein Selbsttest (BIST) durchgeführt. Im weiteren Betrieb laufen im Hintergrund permanent Diagnose-Tasks, die die Funktionen und den Betriebszustand des Gerätes überwachen.

Werden Fehler oder unzulässige Betriebszustände erkannt, werden diese als Warning oder Error in eine Event Queue geschrieben. Jedes Hardware-Modul (EIB 8791 Basiskomponente, Slot 1 ... 4) hat eine eigene EventQueue.

Ob Events aufgetreten sind, muss vom Anwender aktiv abgefragt werden. Es wird empfohlen zyklisch jede Sekunde zu prüfen, ob Fehler aufgetreten sind.

Nur Events die in den Slots unmittelbar zu einer gestörten Ausgabe der Positionsdaten über die Echtzeitschnittstelle (PDL oder UDP) führen können, werden auch direkt im Positionsdatenpaket über ein entsprechendes Bit markiert.

Weiteres siehe: *2.8.2 Abfragen der Event Queues*

1.9 Reset

Die EIB 8791 kann sowohl per TCP/IP Kommando, per Reset-Taste oder „Remote Reset“ ein Reset der EIB 8791 ausgelöst werden. Der Software Reset bietet die Möglichkeit einen einfachen Reset auszulösen, nachdem das Gerät im User-Modus startet. Über Reset Taste und Remote-Reset können die verschiedenen Boot-Modi ausgewählt werden. Siehe „Geräte-Resets“ in der Betriebsanleitung.

Die Remote-Reset Leitung ist Teil des „Digital IN“ Anschluss und hat die identischen elektrischen Parameter.

Anschlussbelegung und elektrische Kennwerte siehe „Betriebsanleitung“

1.10 Shutdown

Die EIB 8791 sollte nicht einfach ausgeschaltet werden, damit interne Daten (Diagnosedaten, usw.) korrekt abgespeichert werden können. Hierfür gibt es den Shutdown Befehl. Nachdem der Befehl an die EIB 8791 gesendet wurde speichert die EIB 8791 alle Daten, schaltet die Positionsausgabe und alle weiteren Ausgänge ab. Nachdem die STATUS LED aus ist kann die EIB 8791 ausgeschaltet werden.

1.11 Firmware Update

Ein Update der Firmware der EIB 8791 kann durch den Benutzer mit einem FTP Client durchgeführt werden. Hierfür ist auf der EIB 8791 ein FTP Server vorhanden. Die Konfiguration (Benutzername, Passwort, FTP Port) erfolgt mit den Netzwerkkommandos der EIB 8791 (siehe „2.10.6 Netzwerkparameter setzen“).

Es dürfen nur spezielle Update-Files für die EIB 8791 von HEIDENHAIN verwendet werden.

1.11.1 Update mittels EIB 8791 Application

Das Update der Firmware kann mit der „EIB8 Application“ (Treiber CD) durchgeführt werden (siehe 3.3.8). Hierzu ist ein Windows PC erforderlich.

1.11.2 FTP Update

Da der FTP Server auf der EIB 8791 ein FAT16 Dateisystem verwendet, dürfen die Dateinamen nur im 8.3-Format auf den FTP Server gespielt werden. Das heißt, die von HEIDENHAIN gelieferte Datei (z.B. 816477-06-00-A.bin) muss vor dem Upload z.B. in „update.bin“ umbenannt werden.

Das folgende Beispiel geht für das Firmwareupdate von einem Computer mit dem Betriebssystem „Windows“ aus. Die EIB 8791 muss über Ethernet mit dem Computer verbunden sein. Der Dateiname für das Update ist in diesem Beispiel „816477-06-00-A.bin“.

1.11.2.1 Update mittels Kommandozeile

Zuerst sollte die Firmwaredatei in einem definierten Verzeichnis abgelegt werden, hierbei ist es vorteilhaft die Datei (z.B. 816477-06-00-A.bin) gleich ins 8.3-Format (update.bin) umzubenennen z.B. „C:\temp\EIB\update.bin“.

Nun wird die Windows Kommandozeile gestartet und in das Verzeichnis „C:\temp\EIB“ gewechselt. Das FTP Client Programm wird mit Hilfe des Kommandos „ftp -A 192.168.168.2“ gestartet.

Die Option -A bewirkt eine Anmeldung am FTP Server als „anonym“ (dieses ist der Default). Wenn der FTP Benutzer und / oder das Passwort geändert wurde, muss die Option „-A“ weg gelassen werden, dadurch fragt der FTP-Server der EIB 8791 den Benutzernamen und das Passwort ab.

Die IP Adresse lautet „192.168.168.2“ (Default-Einstellung) oder die kundenspezifische Einstellung verwenden.

Nun erscheint die Kommandozeile des FTP Client Programmes (ftp>). Hier werden nun folgende Befehle eingegeben:

```
ftp> binary
ftp> cd update
ftp> put update.bin
ftp> quit
```

Der Befehl „binary“ schaltet von ASCII auf binären Dateitransfer um. Der Befehl „cd update“ wechselt in das „update“ Verzeichnis des FTP Servers auf der EIB 8791. Der Befehl „put update.bin“ überträgt die Datei „update.bin“ vom Host (im aktuellen Verzeichnis) zum FTP Server der EIB 8791 und speichert sie im aktuellen Verzeichnis („update“) des FTP Servers. Der Befehl „quit“ meldet sich vom FTP Server ab und verlässt das FTP Programm.

Wenn die Datei erfolgreich übertragen wurde, wird eine entsprechende Meldung vom FTP Client in der Kommandozeile ausgegeben. Die Status LED der EIB 8791 beginnt zu blinken solange das Update läuft. Dieser Vorgang kann bis zu 7 Minuten dauern. Während der Phase in der die Status LED blinkt, darf die Spannungsversorgung nicht abgeschaltet werden und es können auch keine Kommandos über die Ethernet Schnittstelle an die EIB 8791 gesandt werden (EIB 8791 blockiert Kommandos solange die Updateprozedur läuft).

Nachdem die Status LED wieder aktiv ist, muss über das entsprechende Software-Kommando abgefragt werden, ob das Update erfolgreich beendet wurde. Der Status des Update Prozesses kann bis zum nächsten Booten der EIB 8791 abgefragt werden. Bei einem Fehler des Updates sollte die Updateprozedur wiederholt werden.

Mit dem nächsten Reset bootet die EIB 8791 die neue Version der Firmware.

Im Falle eines Fehlers während des Firmware-Updates (Spannungsunterbrechung, CRC Fehler während der Übertragung, ungültige Firmwaredatei, ...) wird die EIB 8791 automatisch im Factorymode gestartet, da kein gültiges Userimage vorhanden ist. Im Factorymode muss nun das Update des Userimages wiederholt werden.

Hinweis für die Automatisierung der Updateprozedur:

Im Verzeichnis „C:\temp\EIB“ die Datei [ftp.txt](#) mit folgendem Inhalt ablegen:

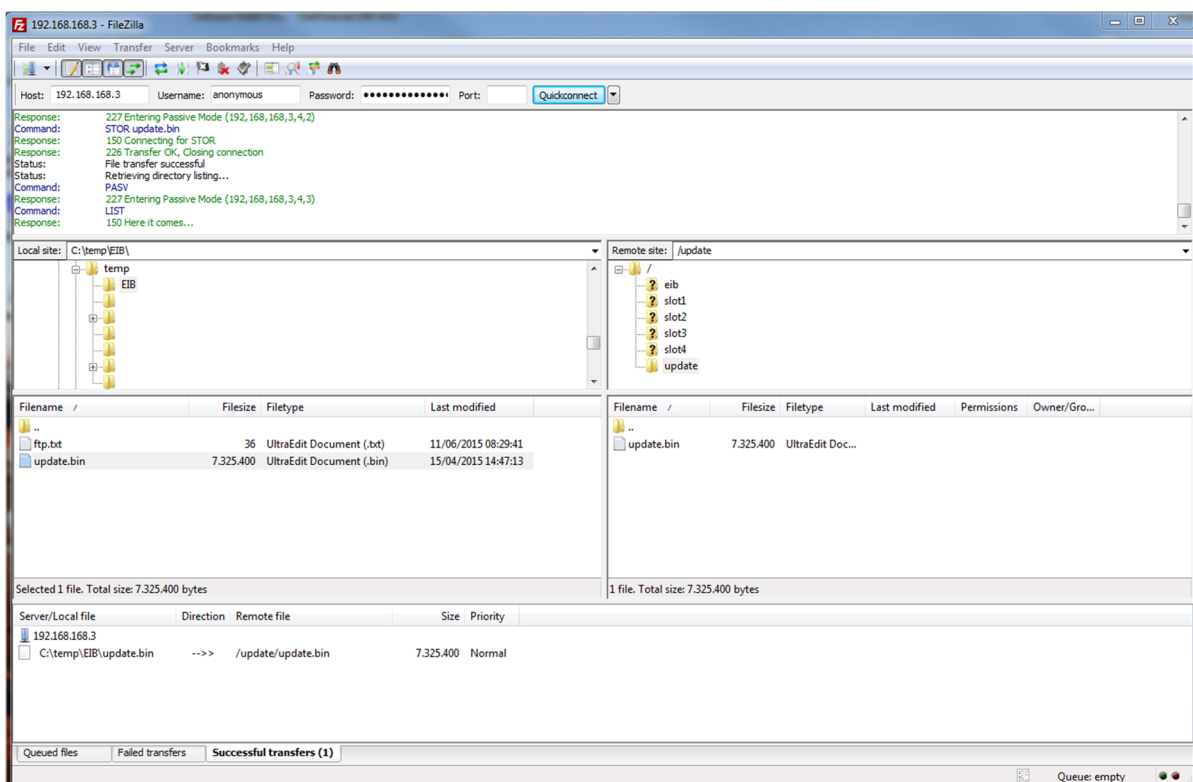
```
binary
cd update
put update.bin
quit
```

Nun kann der Update mit `“ftp -s:ftp.txt -A 192.168.168.2”` gestartet werden, das FTP Clientprogramm führt die Befehle in der Datei [ftp.txt](#) automatisch aus.

1.11.2.2 Update mittels FileZilla

Zuerst sollte die Firmwaredatei in einem definierten Verzeichnis abgelegt werden, hierbei ist es vorteilhaft die Datei (z.B. 816477-06-00-A.bin) gleich im 8.3 Format (update.bin) umzubenennen z.B. `“C:\temp\EIBupdate.bin”`.

Nun wird das FTP Programm FileZilla aufgerufen.



Nun werden die FTP-Daten der EIB 8791 in Host, Username und Password (hier 192.168.168.3, anonymous, „*“) eingegeben und auf „Quickconnect“ gedrückt. Das FTP Programm verbindet sich nun mit der EIB 8791 und zeigt die Verzeichnisstruktur des Host (linke Seite) und der EIB 8791 (rechte Seite) an.

Im Menüpunkt Transfer->Transfer type muss auf „Binary“ umgeschaltet werden.

Nun wechselt man in das Verzeichnis „update“ (rechte Seite) und zieht mit der Maus die Datei „update.bin“ (linke Seite) in dieses Verzeichnis. Nun startet der Datentransfer vom Host zur EIB 8791.

Nach Abschluss der Übertragung beginnt nun die EIB 8791 – wie oben beschrieben – die Updateprozedur.

1.12 FTP Server der EIB 8791

Die EIB 8791 enthält einen FTP Server. Mit Hilfe eines FTP Client-Programmes können Updatedateien und MAP-Daten zur EIB 8791 übertragen und PDL-Daten, Memorydump-Daten und MAP Daten von der EIB 8791 empfangen werden.

Damit die FTP-Verbindung zwischen Host und EIB 8791 funktioniert muss der Steuerkanal (Port 21) bei einer vorhandenen Firewall geöffnet sein. Bei einer aktiven FTP-Verbindung muss zusätzlich der Datenkanal (Port 20) geöffnet werden. Bei einer passiven FTP-Verbindung ist dieses nicht der Fall, da der FTP-Server der EIB 8791 eigenständig keine Verbindung zum Host aufbaut. Der FTP-Port für die Steuerung kann vom Kunden definiert werden (Default 21).

1.12.1 Konfiguration der FTP-Parameter

Mit den Netzwerkbefehlen der EIB 8791 kann die Konfiguration des FTP-Servers verändert bzw. der aktuelle Stand ausgelesen werden.

Folgende FTP-Parameter können angepasst werden (siehe „2.10.6 Netzwerkparameter setzen“):

- FTP Port (Default 21)
- FTP Benutzernamen (Default: „anonymous“)
- FTP Passwort (Default: „*“)

1.12.2 Verzeichnisstruktur des FTP-Servers.

Folgende Verzeichnisse sind auf dem FTP-Server immer vorhanden:

- eib, hier werden PDL Daten und der Memorydump der EIB 8791 Basiskomponente abgelegt.
- slot1, hier werden Memorydumpdaten des Slot1 abgelegt.
- slot2, hier werden Memorydumpdaten des Slot2 abgelegt.
- slot3, hier werden Memorydumpdaten des Slot3 abgelegt.
- slot4, hier werden Memorydumpdaten des Slot4 abgelegt.
- update, in dieses Verzeichnis muss die Firmwaredatei für den EIB 8791-Update abgelegt werden (siehe vorheriges Kapitel).

In jedem Slot-Verzeichnis (slot1, slot2, slot3, slot4) können sich noch Verzeichnisse für die MAP Daten befinden (sid_0000, sid_0001, ...). In diesen Verzeichnissen können MAP Dateien zu den jeweiligen Slots gesendet oder von den Slots abgeholt werden.

1.12.3 Ablage von PDL Daten

Mit Hilfe des Kommandos „EIB8_SaveDataToRD“ können aufgezeichnete PDL Daten in das Verzeichnis „eib“ des FTP Servers übertragen werden. Es können maximal 2 000 000 PDL Pakete im FTP-Server abgelegt werden.

Die PDL Daten müssen sich im RAM der EIB 8791 Basiskomponente befinden (siehe „2.10.1 Recording von Positionsdaten“), nun wird das Kommando „EIB8_SaveDataToRD“ mit dem Dateinamen, dem Slot-Parameter 0x00000000, und der DataID 0x00000002 an die EIB 8791 gesendet. Nach dem Kommando befinden sich die PDL Daten (maximal 2 000 000 PDL Pakete) im „eib“ Verzeichnis des FTP-Servers und können mit einem FTP Client abgeholt werden.

1.12.4 Ablage von Memorydumpdaten

Die Memorydumpdaten enthalten weitergehende Diagnosedaten des EIB 8791 Systems und können von HEIDENHAIN für Supportzwecke verwendet werden.

Mit Hilfe des Kommandos „EIB8_SaveDataToRD“ können Memorydumpdaten von der EIB 8791 Basiskomponente, dem Slot1, Slot2, Slot3 und Slot4 erstellt werden. Die Memorydumpdaten werden im FTP Verzeichnis „eib“ für den EIB-Slot und in „slotX“ für den SlotX abgelegt (wobei X = 1 ... 4).

Das Kommando „EIB8_SaveDataToRD“ wird mit dem Dateinamen, dem entsprechenden Slot-Parameter (0x00000000 für EIB-Slot usw.), und der DataID 0x00000001 an die EIB 8791 gesendet. Nach dem Kommando befinden sich die Memorydumpdaten im entsprechenden Slot-Verzeichnis des FTP-Servers und können mit einem FTP Client abgeholt werden.

1.12.5 Übertragung von MAP Daten zu einem Slot

Mit Hilfe eines FTP-Clients können MAP Daten an die Slots gesendet werden. Der Dateiname der MAP Daten muss dem 8.3-Format entsprechen, da der FTP Server ein FAT16 Dateisystem enthält.

Die Daten werden z.B. in das FTP Serververzeichnis „slot1/sid_0000“ geladen (siehe Kapitel „Firmwareupdate“ → „Daten mittels FTP an die EIB 8791 senden“). Anschließend überträgt die EIB 8791 die MAP Daten an den Slot1 zur StorageID0. Während die EIB 8791 die Daten speichert blinkt die Status LED. Während der Phase, in der die Status LED blinkt, darf die Spannungsversorgung nicht abgeschaltet werden und es können auch keine Kommandos über die Ethernet Schnittstelle an die EIB 8791 gesandt werden (EIB 8791 blockiert Kommandos solange die Updateprozedur läuft).

1.13 DHCP

Die EIB 8791 kann mit statischen IP-Adressen oder alternativ mit dynamischen IP-Adressen, die von einem DHCP-Server bezogen werden, arbeiten. Per Default ist DHCP deaktiviert und die EIB 8791 benutzt statische IP-Adressen. Diese Adresse kann durch den Benutzer gesetzt werden, um sich an die Gegebenheiten eines bestimmten Netzwerkes anzupassen (werksseitig ist hierbei die Adresse 192.168.168.2 voreingestellt).

Wird DHCP aktiviert, versucht die EIB 8791 nach der Bootphase eine IP-Adresse von einem DHCP-Server zu beziehen. Diese Adresse wird so lange benutzt, wie in der Gültigkeitsdauer der „Lease“ angezeigt wird. Falls benötigt, erneuert die EIB 8791 den „Lease“ selbstständig. Wird kein DHCP-Server während der Bootphase gefunden, der eine Adresse zur Verfügung stellt, verwendet die EIB 8791 nach Ablauf eines Timeouts die statische vom Benutzer eingestellte IP-Adresse. Die Bootphase verlängert sich in dem Fall, dass DHCP angewählt ist, aber kein DHCP-Server zur Verfügung steht. Der DHCP-Client fordert eine IP-Adresse, die Subnetzmaske und den Standardgateway an.

Wenn am Anfang – während der Bootphase – eine Adresse mittels DHCP bezogen wurde dann muss nach der Leasetime wieder eine Adresse über DHCP bezogen werden!
Mögliche Abhilfe -> unendliche Leasetime.

Zusätzlich wird der Hostname der EIB 8791 an den DHCP-Server übermittelt. Ist der DHCP-Server mit einem DNS-Server verbunden, dann kann der Hostname anstatt der IP-Adresse zur Kommunikation mit der EIB 8791 verwendet werden. Der Default-Hostname ist individuell für jede EIB 8791 und enthält den Gerätenamen und die eindeutige Seriennummer.

Anbei ein Beispiel für den Hostname: EIB 8791 123456789.

Der Geräte name ist „EIB 8791“ und die Seriennummer ist „123456789“. Die Seriennummer ist auf dem Typenschild auf der Rückseite der EIB 8791 aufgedruckt. Der Hostname kann über ein Software-Kommando geändert werden.

Hinweis:

Für die Änderung der Netzwerk-Einstellung kann z.B. das auf CD mitgelieferte Programm „EIB8 Application“ verwendet werden (siehe 3.3).

2 Konfiguration und Benutzung der EIB 8791 mit der Treibersoftware

2.1 Allgemeine Informationen

Für den Zugriff auf die EIB 8791 aus einer Softwareapplikation werden Funktionen zur Verfügung gestellt. Diese Funktionssammlung wird als DLL für Windows Systeme und als SO-Bibliothek für Linux geliefert. Folgende Betriebssysteme werden unterstützt:

- Windows 7
- Linux/Unix mit Kernel 2.6 – 64 Bit (i386 Systeme)

Für Windows wird die Bibliothek in einer 32 Bit und einer 64 Bit Variante angeboten. Eine 32 Bit Softwareapplikation muss die 32 Bit Bibliothek verwenden, entsprechend muss die 64 Bit Softwareapplikation die 64 Bit Bibliothek einbinden. Für Linux wird nur die 64 Bit Variante bereitgestellt.

Zusätzlich zu den Bibliotheken wird eine Header-Datei geliefert, die eine Integration der Funktionen in C/C++ Programme ermöglicht. Um ein Programm zu erstellen muss die Bibliothek in das Projekt eingebunden werden.

Für erste Tests mit der EIB 8791 kann das mitgelieferte Kommandozeilentool EIB8AppXX.exe für Windows bzw. EIB8App für Linux verwendet werden. Dieses kann ohne Installation gestartet werden (siehe 3.2).

Für LabVIEW™ (Version 2012, Service Pack 1, 32 Bit) werden sog. VIs zur Verfügung gestellt, die als Basis die Windows DLL haben. Die Benennung, die Funktionalität und die Ein- bzw. Ausgabeparameter der VI orientieren sich an den entsprechenden Funktionsaufrufen, die im Anschluss dokumentiert sind. Speziell bei komplexeren Datentypen kann es erforderlich sein, dass die VI neben dem DLL Aufruf noch eine LabVIEW™ spezifische Anpassung enthält. Die entsprechenden Anpassungen sind durch das Öffnen der VI ersichtlich.

Darüber hinaus wird die in LabVIEW™ implementierte EIB 8791 Application (EXE und VI) mitgeliefert, mit der die EIB 8791 betrieben werden kann (siehe 3.3).

2.2 Inhalt der Treiber CD

Auf der Treiber CD befinden sich folgende Verzeichnisse und Dateien:

Verzeichnis	Dateien	Beschreibung
\bin\32Bit	EIB8Driver32.dll	EIB 8791 Treiber DLL für Windows 32 Bit
	EIB8Driver32.lib	EIB 8791 Treiber LIB für Windows 32 Bit
	EIB8LogServer32.exe	Logging Server zur Erfassung von Logging Nachrichten des Treibers
	EIB8TestApp32.exe	Test-Applikation für Spezialanwendungen
	EIB8App32.exe	Kommandozeilentool, das den Treiber benutzt
\bin\64Bit	See \bin\64Bit	Dateien für Windows 64 Bit
\include	eib8_api.h	Include Datei, die von der Softwareapplikation eingebunden wird.
	eib8.h	Deklaration aller Treiberfunktionen und Datentypen für Standardanwendungen
	*.h	Weitere Header Dateien für Spezialanwendungen (müssen immer zusammen mit eib8_api.h verfügbar sein)
\doc	html.zip, Betriebsanleitung	Dokumentation der Treiberschnittstelle; Enthält alle Informationen und Sicherheitshinweise, um das Gerät sachgerecht und bestimmungsgemäß zu betreiben.
	Release Notes	Änderungsdokumentation des Treibers
	Benutzerhandbuch	Benutzerhandbuch zur Treiber-Software; Enthält alle Informationen zu Funktionsumfang, Installation und Funktionsaufrufe der Treiber-Software.
	EIB8_Parameter.xlsx	Tabelle mit allen Parametern der EIB 8791
\example	*.c	Beispielprogramme zur Verwendung des Treibers in einer C / C++ Applikation (siehe 3)
	*.txt	Beispielkonfigurationen, die an die EIB 8791 gesendet werden können (siehe 3)
\labview\32Bit\ EIB8Driver\	EIB8ApiVI*. *	LabVIEW™ VIs, mit denen die Standardfunktionen des Treibers aufgerufen werden können.
	ExampleVI*. *	Beispiel VIs zum Betreiben einer EIB 8791
	HelperSubVI*. *	Hilfs-VIs für die Beispiele und die LabVIEW™ Applikation
	ApplicationVI*. *	VI der LabVIEW™ EIB 8791 Applikation
\labview\32Bit\ EIB8Application	\EIB8 Application EXE*. *	LabVIEW™ EIB 8791 Applikation als ausführbare Datei (exe). Wenn LabVIEW™ auf dem PC vorhanden ist, kann EIB8Application.exe direkt gestartet werden.
	\EIB8 Application Setup*. *	Setup der LabVIEW™ EIB 8791 Applikation mit Installation einer kostenlosen Runtime Version von LabVIEW™
\linux\64Bit\bin	EIB8App	Kommandozeilentool, das den Treiber benutzt
	EIB8LogServer	Logging Server zur Erfassung von Logging Nachrichten des Treibers
	libEIB8Driver.so	EIB 8791 Treiber Library für Linux 64 Bit
\linux\64Bit\include	Siehe \include für Windows	

2.3 Installationsanleitung der Treiber DLL

Die angegebenen Verzeichnisse und Dateien beziehen sich auf die Treiber-CD für die EIB 8791.

2.3.1 Windows

Damit eine Anwendung die DLL laden kann, muss die Datei „EIB8Driver32.dll“ bzw. EIB8Driver64.dll“ in das Windows-Systemverzeichnis kopiert werden (z.B. „C:\Windows\system32“).

Für eine Kompatibilität mit 32 Bit Applikationen sollte zusätzlich die Datei „EIB8DriverXX.dll“ in das Systemverzeichnis "SysWOW64" im Windows-Ordner (z.B. "C:\Windows") kopiert werden. Alternativ kann der Pfad für die DLL im System bekannt gegeben werden oder die DLL in das Verzeichnis der Applikation kopiert werden.

Das Interface der DLL ist über die beiden Dateien „EIB8DriverXX.lib“ und „eib8_api.h“ definiert. Für C/C++ Projekte muss die Header Datei wie folgt eingebunden werden:

```
#include „<Pfad>eib8_api.h“
```

Dabei inkludiert `eib8_api.h` weitere Headerfiles, die im gleichen Pfad verfügbar sein müssen (alle Dateien, die sich im Verzeichnis „\include“ der Treiber CD befinden). Die eigentliche Schnittstelle der DLL ist in „`eib8.h`“ deklariert.

Weiter muss die Bibliothek „EIB8DriverXX.lib“ der Entwicklungsumgebung bekannt gemacht werden. Üblicherweise kann im Bereich des Linkers eine zusätzliche Bibliothek / Library angegeben werden (z.B. unter „linker“ – „input“ – „additional dependencies“).

2.3.2 Linux

Damit eine Anwendung die SO-Bibliothek laden kann, sollte die Datei „`libEIB8Driver.so`“ von der CD ins Verzeichnis „`usr/lib`“ kopiert werden. Das Interface der Bibliothek ist über die Datei „`eib8_api.h`“ definiert. Diese sollte zusammen mit den anderen Headerdateien der Treiber CD („\include“) nach „`usr/local/include`“ kopiert werden und ist in das Softwareprojekt in der Entwicklungsumgebung einzubinden. Die angegebenen Verzeichnisse orientieren sich an dem „Filesystem Hierarchy Standard“ für Linux-Betriebssysteme. Die Bibliothek „`libEIB8Driver.so`“ wurde für i386 Systeme unter Kernel 2.6, 64 Bit kompiliert.

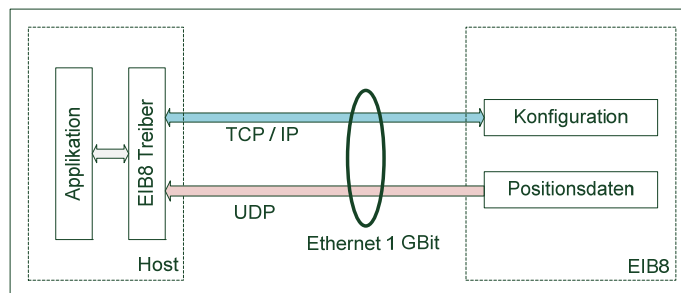
Das Kommandozeilentool `EIB8App` wurde ebenso für i386 Systeme unter Kernel 2.6, 64 Bit kompiliert und kann, wenn sich die „`libEIB8Driver.so`“ im oben angegebenen Verzeichnis befindet, direkt von der Kommandozeile gestartet werden.

2.4 Ethernet-Verbindung mit der EIB 8791 herstellen

Die EIB 8791 wird über Ethernet mit dem Host verbunden. Kommandos werden dabei über TCP/IP vom Host an die EIB 8791 gesendet. Über UDP kann die EIB 8791 zudem Positionsdaten an einen Host übertragen.

Hinweis:

Die Positionsdaten werden üblicherweise an den Host übertragen, der auch die Konfiguration der EIB 8791 durchführt. Möglich ist aber auch, die Positionsdaten an einen zweiten Host zu übertragen. Dazu müssen über einen Switch die entsprechenden Hosts mit der EIB 8791 verbunden werden.



Die IP-Adresse der EIB 8791 ist werksseitig auf 192.168.168.2 eingestellt. Um sich mit der EIB 8791 zu verbinden, muss die IP-Adresse des Hosts so gewählt werden, dass sie sich im gleichen Subnetz mit der Subnetzmaske 255.255.255.0 befindet. Der Host kann beispielsweise auf 192.168.168.100 eingestellt werden.

Um die Verbindung zu testen, kann das auf der CD mitgelieferte Kommandozeilentool `EIB8AppXX.exe` bzw. `EIB8App` genutzt werden (siehe 3.2). Das Kommando `c` (connect) versucht mit der Standard IP Adresse eine Verbindung herzustellen. Mit dem Kommando `i` können dann beispielsweise einige Informationen der EIB 8791 abgefragt werden. `ion` versetzt die LAN LED der EIB 8791 in den Blinkmodus, `ioff` deaktiviert in wieder.

Anschließend kann mit dem Kommando `ipeib` die IP Adresse der EIB 8791 beliebig eingestellt werden. Die Änderung wird in der EIB 8791 erst nach einem Neustart wirksam (Kommando `reset`). Um sich dann mit dem Kommandozeilentool erneut zu verbinden, muss die neue IP Adresse im Kommandozeilentool eingestellt werden (Kommando `iphost`).

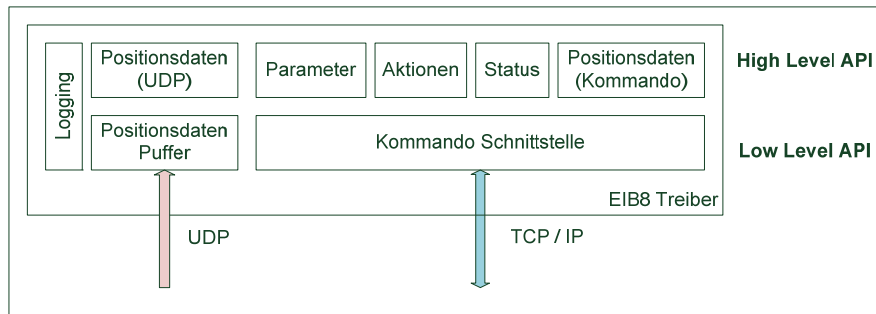
Hinweis:

Sollte die IP Adresse der EIB 8791 unbekannt sein, wird empfohlen, die EIB 8791 im Factory Modus mit Factory Settings (Reset Schalter für mindestens 10 sec drücken) zu starten. Die IP Adresse ist dann wieder 192.168.168.2. Die IP kann dann für den User Modus wie oben beschrieben auf den gewünschten Wert konfiguriert werden.

Tests mit der EIB 8791 können auch komfortabel mit der LabVIEW™ EIB 8791 Applikation durchgeführt werden (siehe Anhang 3.3).

2.5 Überblick EIB 8791 Treiber DLL

2.5.1 Treiberstruktur



Der Treiber kommuniziert mit der EIB 8791 mit Hilfe einer Vielzahl an Kommandos. Die einzelnen Kommandos sind über die sogenannte „Low Level API“ des Treibers verfügbar (Deklarationen in `device_cmd.h`, `device.h`, `err_int.h`), werden jedoch für Standardanwendungen nicht empfohlen. Grundsätzlich sollte die abstrahierte „High Level API“ verwendet werden, die in `eib8.h` definiert ist. Im Folgenden wird ausschließlich auf die „High Level API“ eingegangen.

Positionsdaten überträgt die EIB 8791 über die UDP Schnittstelle. Dabei werden die Daten in der EIB 8791 nicht zwischengespeichert, sondern schnellstmöglich versendet. Der Treiber übernimmt das Puffern der Positionsdaten. Über entsprechende Funktionen kann die Anwendung den Puffer konfigurieren und auf die Positionsdaten zugreifen.

Positionsdaten können auch über die Kommandoschnittstelle abgefragt werden, allerdings nur mit einer Rate von ca. 10Hz.

Insbesondere zur Unterstützung der Fehlersuche bei der Benutzung des Treibers dient die Logging Funktionalität des Treibers. Je nach Logging Level können Fehlermeldungen oder auch der komplette Datenverkehr mit der EIB 8791 geloggt werden. Die Ausgabe der Logging Nachrichten erfolgt wahlweise in eine Datei, auf die Kommandozeile oder auf UDP. Die auf UDP ausgegebenen Logging Nachrichten können mit dem auf der CD mitgelieferten Logging - Server „EIB8LogServerXX.exe“ für Windows bzw. EIB8LogServer für Linux angezeigt werden.

Anmerkung:

Der Empfang der Positionsdaten über die PDL Schnittstelle, für die eine spezielle Hardware benötigt wird, wird seitens des Treibers nicht unterstützt.

2.5.2 Funktionsaufrufe

Die Funktionsaufrufe des Treibers sind in der Regel blockierend. Der Rückgabewert vom Typ `eib8_err_t` gibt an, ob der Funktionsaufruf erfolgreich war (`EIB8_ERROR_OK`, entspricht dem Wert 0) oder ein Fehler aufgetreten ist.

Der Rückgabewert kann mithilfe der Funktion `eib8_get_last_error()` in einen lesbaren String umgewandelt werden.

2.5.3 Strings

Für eine komfortable Benutzung der Funktionen nutzt der Treiber häufig Strings. Um Speicherbereichsüberschreitungen auszuschließen, werden insbesondere Rückgabestrings (z.B. bei der Funktion `eib8_get_driver_version()`) immer in Form des Datentyps `eib8_std_str_t` deklariert, der eine feste maximale Länge hat.

2.5.4 Grundlegende Funktionen

Im Folgenden werden grundlegende Funktionen des Treibers erläutert. Details der angesprochenen Funktionen und Datentypen finden sich in der Schnittstellendefinition („`eib8.h`“) oder der mitgelieferten HTML – Dokumentation.

Das Beispiel „`example_identify`“ auf der Treiber CD zeigt die Verwendung der im Folgenden erläuterten Befehle.

2.5.4.1 Öffnen und Schließen des Treibers

Bevor eine Funktion des Treibers aufgerufen werden kann, muss der Treiber mit der Funktion `eib8_driver_open()` initialisiert werden.

Vor Beendigung der Applikation sollte der Treiber mit der Funktion `eib8_driver_close()`

beendet werden, damit der Treiber den allokierten Speicher wieder freigibt. Mit dem Aufruf der Funktion werden alle EIB 8791 Verbindungen getrennt und alle Handles freigegeben.

Soll in einer Applikation ein bereits mit geöffneter Treiber erneut mit `eib8_driver_open()` geöffnet werden, so muss er vorher mit `eib8_driver_close()` beendet werden.

2.5.4.2 Version des Treibers

Die Treiber Version kann mit Hilfe der Funktion `eib8_get_driver_version()` abgefragt werden.

Beispiel:

2.1.13, Date: Sep 30 2014 12:35:52 [release32], JH-ID: 1065235-02-A-01

2.5.4.3 Logging Funktion des Treibers

Die Logging Funktion des Treibers ist nach dem Öffnen des Treibers deaktiviert. Logging Nachrichten können in die Kommandozeile (stdout)

`eib8_activate_logging_stdout()`

oder in eine Datei

`eib8_activate_logging_file()`

geschrieben werden.

Ebenso können die Logging Nachrichten als UDP Pakete versendet werden (empfohlene Methode):

`eib8_activate_logging_udp()`

Beispiel:

`eib8_activate_logging_udp(EIB8_DRIVER_LOG_LEVEL_WARN, "127.0.0.1", 3000u)`

Im Beispiel werden die Nachrichten an die „interne IP“ 127.0.0.1 und den Port 3000 versendet. Um die Logging Nachrichten sichtbar zu machen, ist der LogServer EIB8LogServerXX.exe auf dem Rechner zu starten, an den die Logging Nachrichten gesendet werden (bei IP 127.0.0.1 der gleiche Rechner, auf dem die Applikation läuft):

`EIB8LogServerXX.exe 3000`

Logging Level:

Bei allen drei Methoden ist der Logging Level (`eib8_driver_log_level_t`) zu übergeben, d.h. ob beispielsweise nur Fehler oder auch Warnungen ausgegeben werden sollen.

Empfohlene Logging Levels:

Logging Level	Bedeutung
<code>EIB8_DRIVER_LOG_LEVEL_NON</code>	Logging wird deaktiviert.
<code>EIB8_DRIVER_LOG_LEVEL_WARN</code>	Alle Warnungen und Fehler werden ausgegeben.
<code>EIB8_DRIVER_LOG_LEVEL_INFO2</code>	Alle Warnungen und Fehler werden ausgegeben. Die komplette Kommunikation mit der EIB 8791 wird protokolliert.

2.5.4.4 EIB 8791 Handles

Der Treiber kann mit bis zu 256 EIB 8791 Geräten (`EIB8_MAX_HANDLES`) gleichzeitig kommunizieren. Dazu wird mit der Funktion

`eib8_get_handle()`

ein Handle für jede EIB 8791 vom Treiber angefordert. Das Handle, das eine EIB 8791 repräsentiert, wird im Folgenden bei jedem Funktionsaufruf, der mit einer EIB 8791 kommuniziert, übergeben.

Mit der Funktion

`eib8_release_handle()`

kann ein Handle wieder freigegeben werden. Die TCP Verbindung zu einer EIB 8791 wird dabei automatisch getrennt. Ein ungültiges Handle hat den Wert `EIB8_INVALID_HANDLE_ID` (0).

Hinweis:

Der Treiber ist „ThreadSafe“ implementiert, d.h. mehrere Threads der Applikation können gleichzeitig auf ein oder mehrere Handles zugreifen.

Erfolgt ein zweiter Zugriff auf ein Handle, mit dem gerade eine Funktion ausgeführt wird, so wird der zweite Zugriff mit dem Fehler `EIB8_ERROR_EIB8_CMD_BUSY` quittiert und die Funktion nicht ausgeführt. Der erste Zugriff ist davon nicht beeinflusst.

2.5.4.5 Verbinden und Trennen der EIB 8791

Um mit der EIB 8791 zu kommunizieren, muss mit der Funktion

eib8_connect_tcp()

eine TCP/IP Verbindung aufgebaut werden.

Beispiel mit den werkseitigen Einstellungen der EIB 8791:

eib8_connect_tcp(&myEIB8handle, „192.168.168.2“, 1050, &CheckResult)

Wurde die Funktion erfolgreich ausgeführt, so ist das Handle mit der entsprechenden EIB 8791 verbunden.

Beim Verbinden überprüft der Treiber grundlegende Systemzustände der EIB 8791 und stellt sie der Applikation in der Struktur

eib8_connect_check_result_t

zur Verfügung. Die Elemente haben dabei folgende Bedeutung:

Eintrag	Bedeutung	Maßnahme
<i>u32BusyUpdate</i>	In der EIB 8791 läuft gerade ein Software Update	Warten und später nochmals versuchen (bis zu 10 min)
<i>u32ErrorUpdate</i>	Beim letzten Update ist ein Fehler aufgetreten	Update Wiederholen. Bei wiederholtem Auftreten setzen Sie sich bitte mit dem Messgeräte-Support bei HEIDENHAIN in Verbindung.
<i>u32FactoryMode</i>	Die EIB 8791 ist im Factory Modus gestartet	Wenn durch den Anwender bewusst veranlasst, nur der Hinweis, dass der volle Funktionsumfang der EIB 8791 zur Verfügung nicht steht. Wenn nicht durch den Anwender veranlasst, dann wurde der letzte Software Update unterbrochen oder es liegt ein Fehler im Flash vor. Software Update wiederholen. Bei wiederholtem Auftreten setzen Sie sich bitte mit dem Messgeräte-Support bei HEIDENHAIN in Verbindung.
<i>u32ErrorHardware</i>	Die EIB 8791 hat einen Hardware Fehler erkannt	EIB 8791 neu starten. Bei wiederholtem Auftreten setzen Sie sich bitte mit dem Messgeräte-Support bei HEIDENHAIN in Verbindung.
<i>u32ErrorBistDiag</i>	Die EIB 8791 hat bei ihrem Selbsttest einen Fehler erkannt	Event Queues auslesen (siehe 2.8.2).
<i>u32PendingEvents</i>	Bei der EIB 8791 sind Einträge in den Event Queues vorhanden	Event Queues auslesen (siehe 2.8.2).
<i>u32IncompDriver</i>	Der Treiber ist nicht kompatibel mit der EIB 8791	Installation einer neuen Treiberversion (Treiberversion muss größer oder gleich der EIB 8791 Software sein).

Ist eine EIB 8791 bereits verbunden, so können die grundlegenden Systemzustände jederzeit in der Applikation mit der Funktion

eib8_connect_result_update()

aktualisiert werden (die Systemzustände können sich aufgrund der in der EIB 8791 eingebauten Diagnosefunktionalität ändern).

Die TCP Verbindung einer EIB 8791 wird mit der Funktion

eib8_disconnect_tcp()

wieder getrennt (erfolgt automatisch auch beim Freigeben des Handles oder beim Schließen des Treibers).

2.5.4.6 „Identify“ an die EIB 8791

Um die Verbindung zur gewünschten EIB 8791 zu überprüfen, kann mit der Funktion

eib8_identify()

die LAN LED an der Frontblende der EIB 8791 in den Blinkmodus versetzt werden.

Beispiel (siehe auch: *example_identify.c* in den Beispielprogrammen):

Blinkmodus Einschalten:

eib8_identify(&myEIB8handle, "EIB8", EIB8_ENABLE)

Blinkmodus Ausschalten:

eib8_identify(&myEIB8handle, "EIB8", EIB8_DISABLE)

2.6 Parameter

2.6.1 Nomenklatur

Der EIB 8791 Treiber stellt eine Schnittstelle zur Verfügung, mit der Parameter zwischen EIB 8791 und Host mittels einer vereinheitlichten Vorgehensweise ausgetauscht werden können.

Dabei gilt folgende Nomenklatur:

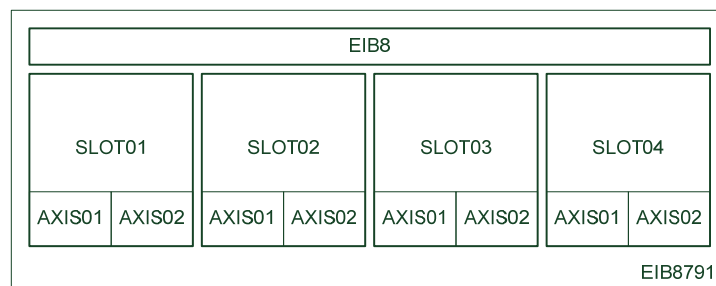
Name	Bedeutung
Konfiguration	Parameter, die an die EIB 8791 gesendet und aus der EIB 8791 gelesen werden können (R/W Parameter)
Status	Parameter, die aus der EIB 8791 nur gelesen werden können (R Parameter)
Parameter	Konfiguration und Status

Ein Parameter ist dabei wie folgt definiert (unabhängig davon, ob es sich um eine Konfiguration oder einen Status handelt):

Bezeichnung	Bedeutung	Beispiel
Node	Knoten	„EIB8“
Param	Parameter	„network_user:ip_address“
Value	Wert	„192.168.168.2“

Knoten:

Der Knoten („Node“) bezeichnet die Komponente in der EIB 8791, für die der Parameter gilt. Innerhalb der EIB 8791 sind folgende Knoten verfügbar:



Beispiele für Knoten:

Knoten	Bedeutung
„EIB8“	EIB8 Basiskomponente
„SLOT01“	Slot (Einschub) 1
„SLOT00“	Alle Slots
„SLOT02_AXIS01“	Achse 1 in Slot 2
„SLOT00:AXIS00“	Alle Achsen in allen Slots

Ein Parameter kann durch die Mehrfachadressierung „SLOT00“ bzw. „AXIS00“ an mehrere Knoten gleichzeitig gesendet werden.

Parameter:

Ein Parameter wird durch seine Parametergruppe und dem eigentlichen Parameternamen bestimmt. Gruppe und Name des Parameters sind durch „:“ getrennt.

Beispiel:

„network_user:ip_address“

Dabei ist „network_user“ die Parametergruppe und „ip_address“ der Parameternamen.

Wert:

Der Wert des Parameters ist das Ergebnis, wenn ein Parameter gelesen wird, oder der Sollwert, wenn ein Parameter geschrieben wird.

Beispiel für die IP Adresse:

„192.168.168.2“

2.6.2 Parameter der EIB 8791

Alle in der EIB 8791 verfügbaren Parameter sind in der Tabelle „EIB8_Parameter.xlsx“ im Verzeichnis „doc“ der Treiber CD mit einer entsprechenden Kurzbeschreibung aufgeführt.

Eine aktuelle Liste kann auch mit Hilfe des Treibers erzeugt werden (siehe 2.6.3). Mit Hilfe des Kommandozeilentools und einer verbundenen EIB 8791 (siehe 2.4) kann mit dem Kommando *pg* für „Parameter get“ die komplette Parameterliste aus der EIB 8791 gelesen werden. Ebenso kann die gesamte Konfiguration (*cg*) oder der komplette Status (*sg*) aus der EIB 8791 gelesen werden.

Hinweis:

Die EIB 8791 verfügt über eine Vielzahl an Parametern, die nur für spezielle Anwendungen relevant sind. Im Folgenden wird v. a. auf die Parameter eingegangen, die im Regelfall Anwendung finden.

2.6.3 Lesen und Schreiben von Parametern

Der Treiber bietet mehrere Funktionen zum Lesen und Schreiben von Parametern. Dabei wird grundsätzlich unterschieden:

- Lesen oder Schreiben eines einzelnen Parameters
- Lesen aller Parameter und Ausgabe als Datei oder Liste
- Schreiben einer Konfigurationsliste: Eingabe über Datei oder Liste

Bei Dateien kann zwischen einem einfachen Textformat oder dem XML Format gewählt werden. Das Textformat ist aufgrund der Übersichtlichkeit zu bevorzugen.

Die folgende Tabelle zeigt eine Übersicht der Funktionen zum Lesen und Schreiben von Parametern:

Funktion	Bedeutung
<i>eib8_set_param_string()</i> <i>eib8_get_param_string()</i>	Lesen oder Schreiben eines Parameters, wobei der Wert ein String ist (immer möglich, auch bei numerischen Werten).
<i>eib8_set_param_bool()</i> <i>eib8_get_param_bool()</i>	Lesen oder Schreiben eines Parameters, wobei der Wert ein bool ist (0 oder 1).
<i>eib8_set_param_hex32()</i> <i>eib8_get_param_hex32()</i>	Lesen oder Schreiben eines Parameters, wobei der Wert ein vorzeichenloser 32 Bit Integer Wert ist.
<i>eib8_set_param_int32()</i> <i>eib8_get_param_int32()</i>	Lesen oder Schreiben eines Parameters, wobei der Wert ein vorzeichenbehafteter 32 Bit Integer ist.
<i>eib8_get_config_all_file()</i> <i>eib8_get_config_all_list()</i>	Lesen der gesamten aktuellen Konfiguration der EIB 8791 und Schreiben in Datei oder eine Liste.
<i>eib8_get_status_all_file()</i> <i>eib8_get_status_all_list()</i>	Lesen des gesamten aktuellen Status der EIB 8791 und Schreiben in Datei oder eine Liste.
<i>eib8_get_parameter_all_file()</i>	Lesen aller Parameter aus der EIB 8791 und Schreiben in eine Datei.
<i>eib8_set_config_file()</i> <i>eib8_set_param_list_string()</i>	Schreiben einer Konfiguration in die EIB 8791: Eingabe über Datei oder Liste.

Beispiele:

Schreiben einer Konfigurationsdatei an die EIB 8791:

```
eib8_set_config_file(&myEIB8Handle, "config_std_EIB 8791.txt", EIB8_FILE_TXT)
```

Dabei ist *"config_std_EIB 8791.txt"* der Dateiname. *"EIB8_FILE_TXT"* legt fest, dass es sich um eine Datei im Text Format handelt.

Schreiben der IP – Adresse:

```
eib8_set_param_string(&myEIB8Handle,  
"EIB8", "network_user:ip_address", "192.168.168.2")
```

Schreiben der internen PTM Frequenz (10 kHz):

```
eib8_set_param_int32(&myEIB8Handle, "EIB8", "trig_ptm_in:freq_config", 10000)
```

Die genaue Verwendung der Funktionen ist in der Schnittstellendefinition („eib8.h“) beschrieben. Beispiele finden sich auch im Verzeichnis „example“ der Treiber CD.

2.6.4 Konfigurationsdateien

Ein Beispiel für eine Konfigurationsdatei befindet sich auf der CD des Treibers: config_std_EIB 8791.txt.

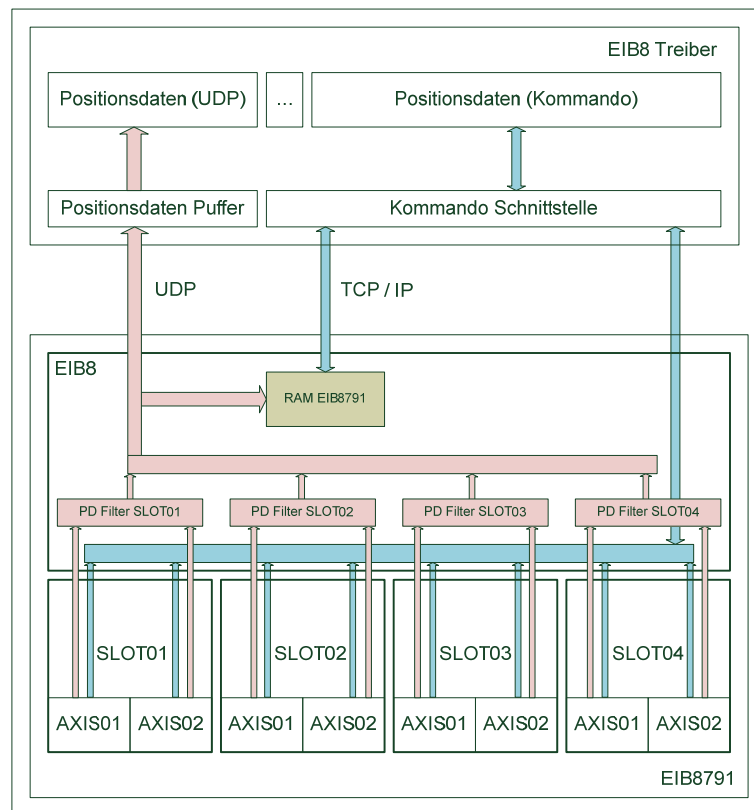
Bezüglich des Formats der Konfigurationsdateien ist folgendes zu beachten:

- Jeder Konfigurationseintrag steht in einer Zeile mit dem Format:
node ; parameter ; value ;
Knoten, Parameter und Wert müssen durch „;“ getrennt sein. Weitere Leerzeichen dazwischen sind erlaubt.
- Zeilen, die keinen Konfigurationseintrag darstellen (Kommentare, Leerzeilen) müssen als erstes Zeichen „#“ haben.
- Die Reihenfolge der Konfigurationseinträge ist unbedeutend. Der Treiber sortiert sie in der Art und Weise, wie sie *eib8_get_config_all_file()* ausgegeben werden.

2.7 Positionsdaten PD

2.7.1 Datenpfad

Positionsdaten werden individuell für jede Achse erzeugt wenn ein entsprechendes Trigger-Ereignis an der Achse auftritt.



Positionsdaten PD können von der EIB 8791 auf verschiedene Art und Weise ausgegeben werden:

- Modus Polling (blauer Datenpfad):
Die Positionsdaten werden über Kommandos von der EIB 8791 geholt. Dabei ist je nach Applikation eine Updaterate von ca. 10 Hz erreichbar.
- Modus UDP (roter Datenpfad):
Die Positionsdaten werden über das UDP Protokoll an einen Host übertragen. Das Ziel des Host wird über seine MAC- und IP-Adresse, sowie über den Port bestimmt, an den die UDP Pakete gesendet werden. Dabei muss das Ziel des UDP Transfers nicht mit dem Host übereinstimmen, der Kommandos an die EIB 8791 sendet. Es ist beispielsweise denkbar, dass die EIB 8791 über einen Switch mit zwei PCs verbunden ist, wobei der eine die Kommandos an die EIB 8791 sendet, während der andere über UDP Positionsdaten empfängt.
Die EIB 8791 ist in der Lage, Positionsdaten bei einer Trigger-Rate von bis zu 400 kHz zu übertragen. Zu prüfen ist hierbei, ob der Host die entsprechende Datenrate verarbeiten kann.
- Modus RAM Recording
Positionsdaten können auch im RAM der EIB 8791 aufgezeichnet werden und nach Beendigung der Aufzeichnung über die Kommandoschnittstelle abgeholt werden. Dabei können bis zu 10 Mio. einzelne Positionen (Positionsdatenpakete) abgespeichert werden.

Hinweise:

- Es können höchstens 32 verschiedene Typen an Positionsdatenpaketen für die Übertragung über UDP bzw. Speichern in das RAM der EIB 8791 konfiguriert werden.
- Positionsdatenpakete können entweder über UDP übertragen oder in das RAM gespeichert werden, aber nicht gleichzeitig beides. Hingegen können jederzeit Positionsdaten über die Kommandoschnittstelle abgefragt werden (auch wenn gleichzeitig Positionsdaten über UDP übertragen oder ins RAM geschrieben werden).

2.7.2 Positionsdatenformat

Die Positionsdaten werden unabhängig vom Typ in einem Positionspaketpaket von 12 Byte übertragen bzw. gespeichert:

Byte Nr.	Bedeutung	Wertebereich	Hinweis
0	Paket Nummer	1 - 255	Die Paket Nummer zur Identifikation des Positionspaketes (<i>u32PacketNumber</i>)
1	Frame - Zähler	0 - 255	Zähler, der bei jedem Trigger-Ereignis um 1 erhöht wird (<i>u32FrameTransCnt</i>). Im Polling Modus ist der Frame-Zähler immer 0.
2	Payload 0	0 - (2 ⁶⁴ - 1)	64 Bit Wert (Payload), der je nach Positionsdatentyp den eigentlichen Positionswert oder die Nutzdaten enthält. Dabei wird der Wert im Format „little endian“ übertragen, d.h. das niederwertige Byte zuerst. (<i>u64Value</i>)
3	Payload 1		
4	Payload 2		
5	Payload 3		
6	Payload 4		
7	Payload 5		
8	Payload 6		
9	Payload 7		
10	Füllbyte		Füllbyte immer mit dem Wert 0xBC
11	Check-Summe		8 Bit CRC über die Bytes 0 - 9. Berechnung nach ITU $x^8 + x^2 + x + 1$

Alle Treiberfunktionen, die Positionsdatenpakete empfangen oder abholen, verwenden folgende Datenstruktur (eib8.h):

```
typedef struct
{
    uint64_t u64Value64;
    uint32_t u32PacketNumber;
    uint32_t u32FrameTransCnt;
    uint32_t u32ChecksumError; /* 0: checksum OK, 1: checksum fail */
    uint32_t u32Dummy; /* Due to 64 bit alignment on various platforms */
} eib8_pdl_packet_t;
```

Die Check – Summe wird dabei bereits vom Treiber ausgewertet.

2.7.3 Positionsdatentypen

In der Regel wird mit dem oben genannten Positionspaket die Position des Messgeräts übertragen. Darüber hinaus kann aber das Paket so konfiguriert werden, dass der 64-Bit-Wert eine andere Bedeutung annimmt. Für eine Achse können damit mehrere Positionspakete erzeugt und übertragen werden. Wird ein Trigger-Ereignis an einer Achse detektiert, so werden alle konfigurierten Positionspakete dieser Achse versendet.

2.7.3.1 Position

Der Datentyp „*position*“ hat folgende Struktur im Positionspaket:

Byte Nr.	Bedeutung	
Payload 0	16 Bit Status	Bit 0 - 3: Reserviert
Payload 1		Bit 4: Limit Signal (L2) aktiv
Payload 2	16 Bit Phase	Bit 5: Homing Signal (L1) aktiv
Payload 3		Bit 6: Referenzmarke gültig
Payload 4		Bit 7 - 13: Reserviert
Payload 5		Bit 14: Event in Event Queue
Payload 6	32 Bit Periodenzähler	Bit 15: Positionsfehler
Payload 7		
		48 Bit Position

Die Position wird als 48 Bit Wert in den oberen 6 Byte der Payload angegeben. Dabei geben die oberen 4 Byte den Wert des Periodenzählers wieder und die unteren 2 Byte die Phase des Inkrementalsignals (Interpolationswert innerhalb der Signalperiode).

Bei Positionspaketen ist insbesondere zu prüfen, ob die Position gültig ist (Bit15 im Status: Positionsfehler). Ist die Position ungültig, beispielsweise weil zunächst kein Messgerät angeschlossen wurde, so kann die Position nur durch das Kommando `eib8_clear_position_error()` wieder gültig gesetzt werden.

Darüber hinaus liefert der Status folgende Informationen:

- Event in Event Queue
Es sind Events im Slot aufgetreten. Die Event Queue sollte ausgelesen werden (siehe 2.8.2).
- Referenzmarke ist gültig:
Das Bit wird gesetzt, wenn eine Referenzmarkensuche erfolgreich war und die Referenzposition gültig ist.
- Homing Signal:
Zustand des Homing Signals
- Limit Signal:
Zustand des Limit Signals.

Der Treiber bietet zur Konvertierung eines allgemeinen Positionsdatenpaketes vom Typ `eib8_pdl_packet_t` eine Funktion, die Position und Status extrahieren:

`eib8_pdl_convert_pos_IDP8791()`

Der Positionswert wird dabei als 64 Bit Wert angegeben, bei dem die die unteren 16 Bit zu Null gesetzt sind.

2.7.3.2 Geschwindigkeit

Der Datentyp „speed“ hat folgende Struktur im Positionsdatenpaket:

Byte Nr.	Bedeutung	
Payload 0	16 Bit Status	Bit 0 - 3: Reserviert
Payload 1		Bit 4: Limit Signal (L2) aktiv Bit 5: Homing Signal (L1) aktiv Bit 6: Referenzmarke gültig Bit 7 - 13: Reserviert Bit 14: Event in Event Queue Bit 15: Positionsfehler
Payload 2	48 Bit Geschwindigkeit	Vorzeichenbehafteter 48 Bit Wert Einheit $\left[\frac{\text{Signalperioden}}{2^{22} \cdot \mu\text{s}} \right]$
Payload 3		
Payload 4		
Payload 5		
Payload 6		
Payload 7		

Der Status entspricht 2.7.3.1 (Position). Entsprechend kann das Paket mit `eib8_pdl_convert_pos_IDP8791()` konvertiert werden. Die Einheit der Geschwindigkeit kann mit `eib8_pdl_convert_velocity_IDP8791` von $\left[\frac{\text{Signalperioden}}{2^{22} \cdot \mu\text{s}} \right]$ in Signalperioden pro Sekunde konvertiert werden.

2.7.3.3 Referenzmarke

Der Datentyp „reference“ hat folgende Struktur im Positionsdatenpaket:

Byte Nr.	Bedeutung	
Payload 0	16 Bit Status	Bit 0 - 3: Reserviert
Payload 1		Bit 4: Limit Signal (L2) aktiv
Payload 2	16 Bit Phase	Bit 5: Homing Signal (L1) aktiv
Payload 3		Bit 6: Referenzmarke gültig
Payload 4	32 Bit Periodenzähler	Bit 7 - 13: Reserviert
Payload 5		Bit 14: Event in Event Queue
Payload 6		Bit 15: Positionsfehler
Payload 7		
		immer 0x00
		Referenz-Position bei Messgeräten mit einer Referenzmarke
		Codierte Referenz-Position bei Messgeräten mit abstandscodierten Referenzmarken

Der Status entspricht 2.7.3.1 (Position). Entsprechend kann das Paket mit `eib8_pdl_convert_pos_IDP8791()` konvertiert werden.

2.7.3.4 Weitere Positionsdatenpakete

Typ	Bedeutung	Anwendung
Konstanter Wert („constant“)	Für eine Achse kann individuell ein konstanter 64 Bit Wert konfiguriert werden, der als Positionsdatenpaket übertragen wird.	Test der Datenübertragung
Inkrementierender Wert („incremental“)	Ein 64 Bit Wert individuell für jede Achse, der bei jedem Trigger-Ereignis inkrementiert wird.	Test der Datenübertragung/ Trigger
Zeitstempel (<i>timestamp</i>)	Ein 64 Bit Wert, der den Zeitstempel des Trigger-Ereignisses, bei dem das Datenpaket ausgelöst wurde, ausgibt. Der Zeitstempel wird beim Start der EIB 8791 auf 0 zurückgesetzt und im 10 MHz Raster inkrementiert.	Zeitstempel der Trigger-Ereignisse, zeitliche Zuordnung der Positionsdatenpakete
Analog Werte („adc“)	Der 64 Bit Wert beinhaltet 4 Analogwerte: Payload 0/1: Spur A Payload 2/3: Spur B Payload 4/5: Spur C (wenn vorhanden) Payload 6/7: Referenzimpuls (wenn vorhanden)	Betrachtung der analogen Eingangssignale vor der Filterkette

2.7.3.5 Paketkonfiguration

Die Paketnummer eines Positionsdatenpaketes wird bei der Konfiguration der EIB 8791 festgelegt. Dabei wird eine Paketnummer einem Positionsdatentyp an einer bestimmten Achse zugeordnet.

Beispiel:

Paketnummer	Achse	Typ	Konfiguration
0x11	Slot 1, Achse 1	Position	<code>SLOT01;pdl_packets:axis_1;0x11:position</code>
0x12 0x13	Slot 1, Achse 2	Position und Konstanter Wert	<code>SLOT01;pdl_packets:axis_2;0x12:position, 0x13:adc</code>

Es werden in dem Beispiel drei Positionsdatenpakete erzeugt: 0x11, 0x12, 0x13. In der Applikation werden Positionsdatenpakete empfangen und müssen anhand der Paketnummer entsprechend interpretiert und weiterverarbeitet werden.

Um die Eindeutigkeit sicherzustellen sollten in einem Gerät keine Paketnummern mehrfach vergeben werden.

2.8 Standard Anwendungsfälle

Anhand von Anwendungsfällen werden die Funktionsaufrufe und Parameter des Treibers erläutert. Wie bereits in 2.5.4 erläutert müssen zunächst der Treiber geöffnet und die EIB 8791 verbunden werden. Dann kann die EIB 8791 konfiguriert werden um Messungen durchzuführen.

Hinweis:

Die Beispiele auf der Treiber CD zeigen die Anwendung der folgenden Funktionen.

2.8.1 Reset – Shutdown der EIB 8791

Um einen Hardware-Reset an der EIB 8791 auszulösen, kann zum einen der Resetschalter an der Frontblende betätigt werden. Wenn möglich sollte aber der Hardware-Reset über ein Kommando ausgelöst werden, damit die EIB 8791 noch die Möglichkeit hat, interne Prozesse abzuschließen. Aus demselben Grund sollte an die EIB 8791 vor dem Betätigen des Ausschalters mit dem Shutdown Befehl heruntergefahren werden. Der Treiber bietet hierzu folgende Funktionen an:

```
eib8_reset()
eib8_shutdown()
```

2.8.2 Abfragen der Event Queues

In der EIB 8791 besitzen die Knoten „EIB8“ und „SLOTXX“ sogenannte „Event Queues“. Dabei werden Fehlermeldungen oder Warnungen, die aufgrund der internen Diagnose – oder Selbsttestfunktionen auftreten, entsprechend protokolliert. (Tritt ein Fehler bei der Ausführung eines Kommandos auf, so wird dieser mit der Quittung des Kommandos gemeldet und im Treiber durch den Rückgabeparameter der Funktion behandelt.)

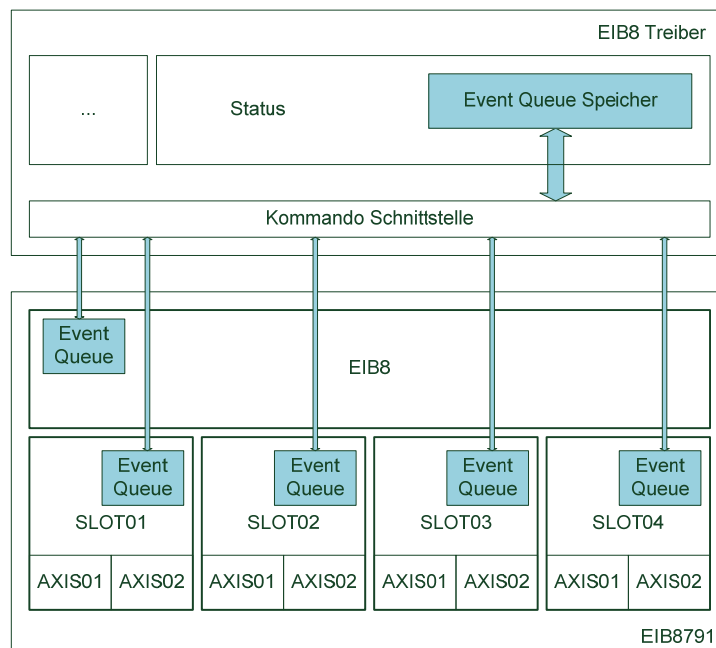
Tritt ein Fehler mehrfach auf, so wird dieser nur einmal in die Event Queue geschrieben. Wurde die Queue aber geleert (d.h. abgefragt), dann werden anliegende Fehler erneut in die Event Queue eingetragen.

Bei den Einträgen in den Event Queues werden grundsätzlich Warnungen und Fehler unterschieden. Bei Fehlern muss die Anwendung sofort reagieren. Warnungen müssen nicht sofort behandelt werden, können aber im Weiteren zu Fehlern führen (z. B. Temperatur übersteigt Warnschwelle, Temperatur übersteigt Fehlerschwelle).

Die Applikation sollte unbedingt die Event Queues zyklisch abfragen (z.B. einmal pro Sekunde) um Störungen in dem Gerät rechtzeitig zu erkennen.

Hinweise:

- Ob sich Events in den Event Queues befinden, wird dies bereits vom Treiber beim Verbinden mit der EIB 8791 ermittelt (u32PendingEvents). Die Events werden dabei noch nicht abgefragt.
- Der Status in den Positionsdatenpaketen enthält die Information „Event in Event Queue“ (siehe 2.7.3.1)



Der Treiber bietet dabei entsprechende Funktionen, um die Abfrage der Event Queues zu vereinfachen. Mit der Funktion:

```
eib8_check_event_queues()
```

werden alle Event Queues in der EIB 8791 abgefragt und im Event Queue Speicher des Treibers abgelegt (die Event Queues der Knoten sind damit gelöscht). Die Funktion gibt dabei die Zahl aller ermittelten Warnung und Fehler aus. Beim erneuten Aufruf von *eib8_check_event_queues()* werden der Event Queue Speicher des Treibers gelöscht und die Event Queues der Knoten erneut abgefragt.

Sind Events im Event Queue Speicher der vorhanden, so können diese von der Applikation einzeln mit der Funktion `eib8_get_event_queue_entry()` ausgelesen werden (beim Lesen der Events werden sie im Event Queue Speicher gelöscht). Der Treiber wandelt dabei die Fehlercodes in lesbare Strings um.

Alternativ kann auch der gesamte Event Queue Speicher des Treibers in eine Textdatei geschrieben werden:

`eib8_event_queue_to_txt()`

Dabei wird der Event Queue Speicher des Treibers nicht gelöscht.

2.8.3 Laden einer Standardkonfiguration

Der Treiber bietet die Möglichkeit, eine Standardkonfiguration an die EIB 8791 zu senden, so dass sofort Messwerte generiert und abgefragt werden können.

`eib8_set_default_conf_EIB 8791()`

Beim Funktionsaufruf müssen bereits MAC- und IP-Adresse, sowie der Port für die Ausgabe von Positionsdaten über UDP angegeben werden.

Beispiel:

`eib8_set_default_conf_EIB 8791 (&myEIB8Handle, „90.e2.ba.03.9c.eb“, „192.168.168.100“, 3051)`

Wenn keine UDP Übertragung konfiguriert werden soll, können die Parameter auch folgendermaßen gesetzt werden:

`eib8_set_default_conf_EIB 8791(&myEIB8Handle, „“, „“, 0)`

Die Standardkonfiguration nimmt dabei folgende Einstellungen vor:

Nr	Konfiguration	Erläuterung
1	<code>EIB8;trig_ptm_in:source;internal EIB8;trig_ptm_in:freq_config;1000</code>	Verwendung des internen PTM Triggers mit einer Frequenz von 1000 Hz
2	<code>EIB8;trig_bus1_out:inputs;trig_ptm_in EIB8;trig_bus1_out:enable;1 EIB8;trig_bus1_out:transmitter;1</code>	Internes Trigger-Routing im Knoten EIB8 (PTM als Trigger)
3	<code>EIB8;pdL_tx_from_slots_to_lane_1:slot_1;1 EIB8;pdL_tx_from_slots_to_lane_1:slot_2;1 EIB8;pdL_tx_from_slots_to_lane_1:slot_3;1 EIB8;pdL_tx_from_slots_to_lane_1:slot_4;1</code>	Internes Routing der Positionsdaten in der EIB 8791 (sollte in Standardanwendungen nicht verändert werden)
4	<code>EIB8;pdL_forwarding_ram_udp:slot_1;0x11,0x12 EIB8;pdL_forwarding_ram_udp:slot_2;0x21,0x22 EIB8;pdL_forwarding_ram_udp:slot_3;0x31,0x32 EIB8;pdL_forwarding_ram_udp:slot_4;0x41,0x42</code>	Positionsdatenfilter für die Paketnummern: Slot 1: 0x11, 0x12 Slot 2: 0x21, 0x22 Slot 3: 0x31, 0x32 Slot 4: 0x41, 0x42
5	<code>EIB8;udp_transfer:udp_dest_mac;90.e2.ba.03.9c.eb EIB8;udp_transfer:udp_dest_ip;192.168.168.100 EIB8;udp_transfer:udp_dest_port;3051</code>	MAC-, IP-Adresse , Port für UDP Übertragung
6	<code>EIB8;pdL_forwarding_ram_udp:mode; batch_soft_realtime</code>	Pfad der Positionsdaten auf UDP Transfer
7	<code>SLOT00;trig_sync1_in:input;trig_bus1_in SLOT00;trig_bus1_in:enable;1 SLOT00;trig_sync1_in:enable;1 SLOT00:AXIS00;trigger:inputs;trig_sync1_in</code>	Internes Trigger-Routing in den Slots und Achsen, alle Achsen identisch (PTM als Trigger)
8	<code>SLOT01;pdL_packets:axis_1;0x11:position SLOT01;pdL_packets:axis_2;0x12:position SLOT02;pdL_packets:axis_1;0x21:position SLOT02;pdL_packets:axis_2;0x22:position SLOT03;pdL_packets:axis_1;0x31:position SLOT03;pdL_packets:axis_2;0x32:position SLOT04;pdL_packets:axis_1;0x41:position SLOT04;pdL_packets:axis_2;0x42:position</code>	Konfiguration der Positionsdatenpakete: Jede Achse liefert ein Paket vom Typ „position“. (Die hier definierten Pakete müssen im PositionData Filter eingetragen sein)
9	<code>SLOT00:AXIS00;encoder_config:type;linear SLOT00:AXIS00;encoder_config:interface;1V_pp_0_90 SLOT00:AXIS00;encoder_config:reference_type;single SLOT00:AXIS00;encoder_config:line_cnt;0 SLOT00:AXIS00;encoder_config:ref_increment;0 SLOT00:AXIS00;encoder_config:limit_signal_1_present;0 SLOT00:AXIS00;encoder_config:limit_signal_2_present;0 SLOT00:AXIS00;encoder_config:homing_signal_present;0 SLOT00:AXIS00;encoder_config:limit_signal_2_present;0</code>	Messgerätkonfiguration auf allen Achsen identisch (siehe 2.9.3)
10	<code>SLOT00:AXIS00;encoder:supply_enable;1</code>	Messgerätversorgung für alle Achsen aktivieren
11	<code>SLOT00:AXIS00;encoder_processing:online_comp_enable;1</code>	Onlinekompensation für alle Achsen aktivieren

Nr	Konfiguration	Erläuterung
12	SLOT00:AXIS00;pos_value_filter:active;1 SLOT00:AXIS00; pos_value_filter:characteristic;linear SLOT00:AXIS00; pos_value_filter:bandwidth;high SLOT00:AXIS00; pos_value_filter:measure_time;0	Positionswertefilter aktivieren
13	SLOT00:AXIS00;pdl:output_active;1	Erzeugung von Positionsdatenpaketen für alle Achsen aktivieren

Hinweise:

- Wird die Funktion `eib8_set_default_conf_EIB 8791()` zur Einstellung der Standardkonfiguration verwendet, so stellt der Treiber sicher, dass die Frame -Zähler aller Achsen bei 0 beginnen (siehe 0).
- In den Programmierbeispielen wird die Konfiguration mittels einer Konfigurationsliste eingestellt (`eib8_set_param_list_string()`). Dabei müssen MAC- und IP-Adresse, sowie der Port für die Ausgabe von Positionsdaten über UDP entsprechend eingetragen werden.
- Ebenso kann die Standardkonfiguration durch Laden der Beispielkonfiguration „`config_std_EIB 8791.txt`“ auf der Treiber-CD mit der Funktion `eib8_set_config_file()` eingestellt werden. Zu beachten ist lediglich, dass der PTM Trigger nach dem Laden der Konfigurationsdatei eigens aktiviert werden muss.
- Bei der Standardkonfiguration werden Positionen für alle 8 Achsen ausgegeben. Ist an einem Messgeräteingang kein Messgerät angeschlossen, so wird ein Event erzeugt, weil die Signalamplituden zu gering sind. Um die Ausgabe der Events von nicht angeschlossenen Messgeräteingängen zu unterbinden, müssen in einer Gesamtkonfiguration (z. B. Konfigurationsdatei) die entsprechenden Achsen komplett entfernt werden oder zumindest folgende Parameter eingestellt werden:
 - `SLOT0X:AXIS0X;trigger:inputs;`
 - `SLOT0X:AXIS0X;encoder:supply_enable;0`

2.8.4 Konfiguration zurücksetzen

Nach dem Einschalten der EIB 8791 befindet sich die Konfiguration im Grundzustand, d.h. es werden keine Positionsdaten, Trigger etc. erzeugt und ausgegeben. Wenn die EIB 8791 konfiguriert wurde, kann sie mit der Funktion

`eib8_reset_idle()`

wieder in einen entsprechenden Grundzustand gebracht werden, ohne einen Hardware-Reset auszuführen.

Insbesondere wenn das Gerät neu konfiguriert werden soll, ist die Ausführung von `eib8_reset_idle()` empfehlenswert.

Hinweis:

`eib8_set_default_conf_EIB 8791()` führt `eib8_reset_idle()` automatisch aus. Auch in den Programmierbeispielen wird vor dem Laden der neuen Konfiguration stets `eib8_reset_idle()` aufgerufen.

2.8.5 Polling von Positionsdaten

Positionsdaten können unabhängig davon, ob Daten über UDP versendet oder im RAM der EIB 8791 aufgezeichnet werden, im Polling Modus abgefragt werden.

Voraussetzungen dazu sind:

- Konfiguration Trigger: Auslösen von Positionsdatenpaketen
- Konfiguration der Positionsdatenpakete
- Konfiguration der Messgeräte und aktivierte Messgerät Versorgung
- Erzeugung von Positionsdatenpaketen aktiviert

Mit der Funktion

`eib8_poll_pdl_packets()`

liefert der Treiber eine Liste aller Positionsdaten (Format siehe 2.7.2), die im Gerät konfiguriert sind (der Eingabeparameter `pNode` ist dabei auf „`SLOT00`“ zu setzen).

Hinweise:

- Die Standardkonfiguration (siehe oben) erfüllt alle Voraussetzungen für das Abfragen von Positionsdaten im Polling Modus
- Wenn kein Trigger-Ereignis stattfindet, so wird die Position (auch die anderen Positionsdatentypen) des letzten Trigger-Ereignisses ausgegeben.
- Das Beispiel „`example_poll.c`“ auf der Treiber-CD zeigt die Verwendung der Funktion `eib8_poll_pdl_packets()` und die Weiterverarbeitung der Positionsdaten.
- Je nach Host erreicht man im Modus Polling eine Updaterate der Messwerte von ca. 10 Hz. Der Polling Modus eignet sich daher beispielsweise für eine Messwertanzeige.

2.8.6 UDP Empfang von Positionsdaten

Wie in 2.7.1 beschrieben können Positionsdaten mit hohen Taktraten über UDP ausgegeben werden.

2.8.6.1 Konfiguration der EIB 8791

Folgende Konfigurationsparameter sind für die Übertragung der Positionsdaten über UDP von Bedeutung:

Konfiguration (Beispiel)	Erläuterung
<i>EIB8;pd_l_tx_from_slots_to_lane_1:slot_1;1</i> <i>EIB8;pd_l_tx_from_slots_to_lane_1:slot_2;1</i> <i>EIB8;pd_l_tx_from_slots_to_lane_1:slot_3;1</i> <i>EIB8;pd_l_tx_from_slots_to_lane_1:slot_4;1</i>	Internes Routing der Positionsdaten in der EIB 8791 (sollte in Standardanwendungen nicht verändert werden)
<i>EIB8;pd_l_forwarding_ram_udp:slot_1;0x11,0x12</i> <i>EIB8;pd_l_forwarding_ram_udp:slot_2;0x21,0x22</i> <i>EIB8;pd_l_forwarding_ram_udp:slot_3;0x31,0x32</i> <i>EIB8;pd_l_forwarding_ram_udp:slot_4;0x41,0x42</i>	Positionsdatenfilter für die Paketnummern: Slot 1: 0x11, 0x12 Slot 2: 0x21, 0x22 Slot 3: 0x31, 0x32 Slot 4: 0x41, 0x42
<i>EIB8;udp_transfer:udp_dest_mac;90.e2.ba.03.9c.eb</i> <i>EIB8;udp_transfer:udp_dest_ip;192.168.168.100</i> <i>EIB8;udp_transfer:udp_dest_port;3051</i>	MAC-, IP-Adresse, Port für UDP Übertragung (Adresse des Hosts, zu dem die Daten übertragen werden sollen)
<i>EIB8;pd_l_forwarding_ram_udp:mode;udp_batch</i>	Pfad der Positionsdaten auf UDP Transfer

Positionsdatenfilter:

Für die einzelnen Slots müssen die Paketnummern eingetragen werden, die über UDP versendet werden sollen. Dabei sollten nur Paketnummern eingetragen werden, die auch auf einer Achse erzeugt werden. Die Zahl der einstellbaren Paketnummern aller Slots zusammen beträgt 32. Die Verteilung auf die Slots ist dabei beliebig.

UDP Destination:

Der EIB 8791 muss das Ziel der UDP Übertragung in Form von MAC- und IP-Adresse, sowie dem Port mitgeteilt werden. Zur Ermittlung der MAC-Adresse einer bekannten IP-Adresse stellt der Treiber die Funktion

eib8_discover_MAC_from_IP()

zur Verfügung.

UDP Modus:

Der Parameter

EIB8;pd_l_forwarding_ram_udp:mode;"Wert"

bestimmt das Verhalten der Positionsdatenausgabe:

Wert	Erläuterung
<i>stop</i>	Keine Positionsdatenausgabe (Default)
<i>batch_soft_realtime</i>	Es werden mehrere Positionsdatenpakete zu einem UDP Paket zusammengefasst. Die Datenübertragung erfolgt mit geringem Overhead. Der Host muss weniger UDP Pakete verarbeiten. Empfohlen für den Synchronen Betrieb bei mehreren Achsen und Datenpaketen.
<i>direct_soft_realtime</i>	Jedes Positionsdatenpaket wird in einem eigenen UDP Paket versendet. Aus technischen Gründen werden aber nicht 12 Byte versendet, sondern 24. Die Bytes 13 – 24 sind immer 0 und haben keine Bedeutung. Insbesondere bei verschiedenen asynchronen Triggerquellen (wenn kein zeitlich fixes oder äquidistantes Zeitraster für verschiedene Achsen vorhanden ist), werden kürzest mögliche Latenzzeiten erreicht.
<i>recording</i>	Positionsdaten werden ins RAM der EIB 8791 geschrieben (siehe 2.10.1)

Hinweise zu *udp_batch_soft_realtime*:

Es werden immer so viele Positionsdatenpakete zusammengefasst, wie Paketnummern für die Positionsdatenfilter konfiguriert sind (maximal 32). Sind beispielsweise 5 Paketnummern in den Filtern konfiguriert, so wartet die EIB 8791 bis 5 Positionsdatenpakete verfügbar sind und versendet diese. Dabei wird nicht berücksichtigt, ob diese zum selben Trigger-Ereignis gehören oder nicht.

Um sicherzustellen, dass alle Positionsdatenpakete in einem UDP Paket zu einem Trigger-Ereignis gehören, ist folgendes sicherzustellen:

- Alle im Filter eingestellten Paketnummern sind für die Achsen konfiguriert und werden mit demselben Trigger-Ereignis erzeugt (synchron)
- Beim ersten Trigger-Ereignis liefern bereits alle Achsen gültige Positionsdatenpakete. Dies ist beispielsweise nicht der Fall, wenn der Trigger bereits anliegt und nacheinander die Achsen konfiguriert werden. Dann liefern die Achsen ihre ersten Positionsdatenpakete zu verschiedenen Trigger-Ereignissen (Sicherstellen eines synchronen Startes der Positionsdatenausgabe siehe 0).

Weiterhin ist zu beachten:

- Damit *udp_batch_soft_realtime* aktiviert werden kann, muss mindestens ein Positionsdatenpaket im Positionsdatenfilter konfiguriert sein
- Der Positionsdatenfilter kann nur konfiguriert werden, wenn *EIB8;pd_l_forwarding_ram_udp:mode* auf „*stop*“ gesetzt wurde. In der Regel wird dies bereits automatisch im Treiber behandelt.

2.8.6.2 UDP Empfang des Treibers Aktivieren und Deaktivieren

Der Treiber bietet entsprechende Funktionen, um Positionsdaten über UDP zu empfangen. Dabei werden die Daten zunächst in einem internen Positionsdatenpuffer des Treibers mit einem hoch priorisierten Task zwischengespeichert (siehe 2.7.1). Die Anwendung holt dann mit entsprechenden Funktionen die Daten nach dem FIFO Prinzip aus dem Puffer wieder ab.

Läuft der Puffer voll, so werden die jüngsten Daten im Puffer verworfen. D.h. neue Daten werden erst wieder im Puffer abgelegt, wenn Speicherplatz verfügbar ist.

Die Aktivierung des Empfangs erfolgt mit:

```
eib8_start_udp_pdl_sampling()
```

Dabei kann der Aufruf mit dem Handle einer verbundenen EIB 8791 erfolgen oder mit einem neuen Handle. Für den UDP Empfang muss das Handle nicht mit einer EIB 8791 verbunden sein. Die Funktion erwartet außerdem folgende Eingabeparameter:

UDP Port:

Dabei ist der Port anzugeben, an den die EIB 8791 sendet. Dieser wurde der EIB 8791 mit beispielsweise

```
EIB8;udp_transfer:udp_dest_port;3051
```

konfiguriert.

UDP Empfang kann mit verschiedenen Handles auf mehreren unterschiedlichen Ports gleichzeitig erfolgen (wenn sich mehrere EIB 8791 im Netzwerk befinden).

Größe des Positionsdatenpuffers:

Die Größe des Positionsdatenpuffers ist in Byte anzugeben. Jedes Positionsdatenpaket benötigt 12 Byte. Werden beispielsweise 8 Positionsdatenpakete mit einer Rate von 10 kHz erzeugt, so entstehen in einer Sekunde ca. 1 MByte Daten.

Dabei hängt es von der Anwendung ab, welche Puffergröße sinnvoll ist: Je größer der Puffer ist, desto unwahrscheinlicher wird ein Datenverlust aufgrund von applikationsseitigen Unterbrechungen beim Abholen der Daten aus dem Puffer. Andererseits befinden sich in einem großen Puffer möglicherweise veraltete Daten.

Beendet wird der UDP Empfang mit:

```
eib8_stop_udp_pdl_sampling()
```

Dabei wird der Positionsdatenpuffer gelöscht.

2.8.6.3 Lesen der Positionsdaten aus dem Positionsdatenpuffer

Der Treiber stellt mehrere Möglichkeiten bereit, Positionsdaten aus dem Positionsdatenpuffer zu lesen (detaillierte Beschreibung in *eib8.h*). Dabei wird in der Regel die Information zurückgegeben, ob ein Überlauf im Positionsdatenpuffer aufgetreten ist.

***eib8_get_udp_pdl_sampling()*:**

Abholen eines Positionsdatenpaketes aus dem Positionsdatenpuffer.

***eib8_get_udp_multiple_pdl()*:**

Abholen mehrerer Positionsdatenpakete aus dem Positionsdatenpuffer. Dabei gibt die Applikation an, wie viele Pakete sie maximal empfangen kann. Der Treiber gibt dann die maximale Anzahl an Paketen an die Applikation zurück. Befinden sich im Puffer weniger Pakete als die Applikation aufnehmen kann, werden alle Pakete aus dem Puffer sofort zurückgegeben (kein Blockieren der Funktion, bis die maximale Anzahl an Paketen verfügbar ist).

***eib8_get_udp_pdl_latest()*:**

Der Treiber liest den kompletten Positionsdatenpuffer und ermittelt von jeder Paketnummer das aktuellste Positionsdatenpaket. Die ermittelten Pakete werden in eine Liste eingetragen und an die Applikation zurückgegeben. Der Positionsdatenpuffer wird dabei komplett geleert.

Damit die Funktion bei hohen Datenraten nicht unkontrolliert blockiert, kann ein Timeout eingestellt werden, bei dem der Treiber das Lesen des Positionsdatenpuffers abbricht. Auch in diesem Fall wird aber eine gültige Liste zurückgegeben.

Diese Funktion eignet sich beispielsweise für Anzeigen, bei denen der aktuellste Wert von Bedeutung ist.

***eib8_get_udp_synchronous_pdl()*:**

Die Applikation gibt vor, wie viele Positionsdatenpakete sie mit gleichem Frame-Zähler erwartet. Der Treiber liest den Positionsdatenspeicher, bis die erwarteten Pakete eingetroffen sind und gibt diese an die Applikation zurück (mit Timeout). Beim nächsten Aufruf prüft der Treiber, ob sich der Frame-Zähler um 1 erhöht hat und wartet wiederum auf die gewünschte Zahl an Paketen. Ein Fehler wird ausgegeben:

- wenn innerhalb eines Timeouts nicht alle Pakete eintreffen
- wenn nicht alle Pakete den gleichen Frame-Zähler haben
- wenn beim nächsten Aufruf sich der Frame-Zähler nicht genau um 1 erhöht

Diese Funktion ist für sogenannte synchrone Systeme gedacht, bei denen eine Trigger-Quelle alle Positionsdaten für alle Achsen auslöst. Der Treiber gibt hier bei einem Aufruf Pakete zurück, die sich auf ein Trigger-Ereignis beziehen und prüft gleichzeitig, ob Daten verloren gegangen sind. Voraussetzung ist hierfür ein synchroner Start der Positionsausgabe (siehe 0).

2.8.6.4 Schreiben von Positionsdaten in Datei

Der Treiber bietet die Möglichkeit, Positionsdaten in einem Hintergrund-Task in eine Datei zu schreiben. Dabei wird ein Positionsdatenpuffer initialisiert und der gewünschte Port für den UDP Empfang entsprechend initialisiert. Die Aufrufe entsprechen 2.8.6.2, lediglich Dateipfad und Dateiformat müssen entsprechend angegeben werden:

```
eib8_start_udp_pdl_dumping()  
eib8_stop_udp_pdl_dumping()
```

Überwacht werden kann der Prozess mit der Funktion:

```
eib8_monitor_udp_pdl_dumping()
```

Ausgegeben wird hierbei beispielweise, wie viele Pakete bereits geschrieben wurden oder ob ein Überlauf des Positionsdatenspeichers aufgetreten ist.

2.8.6.5 Datenverlust beim UDP Empfang

Da UDP ein ungesichertes und verbindungsloses Protokoll ist, gibt es keine Garantie, dass die Daten auch beim Host ankommen. Die EIB 8791 kann nicht erkennen, ob der Host existiert oder ob der Host aufgrund von Überlast Pakete verliert.

Insbesondere hohe Datenraten (die EIB 8791 kann mit einer Rate von bis zu 400 kHz Pakete versenden) führen zu Datenverlust auf dem Host. Dabei gibt es mehrere Ursachen:

- Bereits in der Netzwerkkarte gehen Daten verloren
- Das Betriebssystem kann die Anforderungen der Netzwerkkarte nicht schnell genug bearbeiten
- Der EIB 8791 Treiber kann die Sockets des Betriebssystems nicht schnell genug bedienen
- Der Positionsdatenpuffer läuft über, weil die Applikation nicht schnell genug an der Treiber API abholt.

Das Risiko, Daten zu verlieren hängt also sehr stark mit der verwendeten Hardware (PC und Netzwerkkarte), aber auch mit dem Betriebssystem und der Applikation zusammen. Tests auf verschiedenen Rechnern haben ergeben, dass bei 16 Positionsdatenpaketen mit Modus „*udp_soft_realtime*“ Triggerraten von 10 – 50 kHz möglich sind. Zu Problemen kann es trotzdem kommen, wenn beispielsweise ein Virens Scanner aktiv ist.

Grundsätzlich muss immer damit gerechnet werden, dass Daten, wenn auch sehr sporadisch, verloren gehen können.

Hierzu sollte in die Applikation eine entsprechende Fehlererkennung eingebaut werden. Dazu kann der Frame-Zähler der Positionsdatenpakete analysiert werden. Bei einer Paketnummer muss der Frame-Zähler bei jedem neuen Positionsdatenpaket um 1 inkrementiert sein (Wertebereich 0 - 255). Im synchronen Betrieb (für alle Achsen derselbe Trigger) führt diese Überprüfung bereits die Funktion *eib8_get_udp_synchronous_pdl()* durch.

2.9 Eigene Konfiguration einstellen

2.9.1 Konfiguration des Systemtakts

Wie in 1.6 beschrieben kann für die EIB 8791 der Systemtakt selektiert werden, sowie der interne Systemtakt nach außen geschaltet werden:

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
EIB8	<i>sync_clock:source</i>	<i>internal</i> <i>ext_single</i> <i>ext_diff</i>	Setzen des Systemtakts auf: – intern (Default) – extern an Single-Ended Eingang – extern an differentiellen Eingang
EIB8	<i>sync_clock:output</i>	0 1	Aktivieren Systemtakt am Single-Ended Ausgang: – deaktiviert (Default) – aktiviert (Der differentielle Ausgang des Systemtakts ist immer aktiviert)

2.9.2 Konfiguration der Trigger

Die EIB 8791 besitzt eine Vielzahl an Trigger-Möglichkeiten. Im Folgenden sollen drei typische Applikationen behandelt werden:

2.9.2.1 Synchroner Betrieb mit PTM

Alle Achsen triggern gemeinsam auf einen PTM, der intern erzeugt wird oder von extern eingespeist wird. Das PTM Signal ist, wie in 1.6 beschrieben, auf den Systemtakt der EIB 8791 bezogen. Daher darf der interne PTM nur zusammen mit dem internen Systemtakt und der externe PTM nur zusammen mit dem externen Systemtakt verwendet werden (Konfiguration Systemtakt siehe 2.9.1). Folgende Konfigurationsparameter sind an die EIB 8791 zu senden:

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
EIB8	<i>trig_ptm_in:source</i>	<i>internal</i> <i>ext_single</i> <i>ext_diff</i>	Setzen des PTMs auf: – intern (Default) – extern an single-ended Eingang – extern an differentiellem Eingang
EIB8	<i>trig_ptm_in:freq_config</i>	< Hz >	Frequenz des internen PTMs in Hz (nur relevant bei Verwendung des internen PTMs), Default 0
EIB8	<i>trig_ptm_in:output</i>	0 1	Internen PTM am single-ended Ausgang aktivieren: – aus (Default) – ein (Der differentielle Ausgang des PTMs ist immer aktiviert)
EIB8	<i>trig_bus1_out:inputs</i> <i>trig_bus1_out:enable</i> <i>trig_bus1_out:transmitter</i>	<i>trig_ptm_in</i> 1 1	Internes Trigger-Routing im Knoten EIB 8791 für PTM als Trigger
SLOT00	<i>trig_sync1_in:input</i> <i>trig_bus1_in:enable</i> <i>trig_sync1_in:enable</i>	<i>trig_bus1_in</i> 1 1	Internes Trigger-Routing in den Knoten SLOT für PTM als Trigger (identisch für alle Slots)
SLOT00:AXIS00	<i>trigger:inputs</i>	<i>trig_sync1_in</i>	Internes Trigger-Routing in den Knoten AXIS für PTM als Trigger (identisch für alle Achsen)

Hinweise zum PTM:

- Die Frequenz des internen PTMs wird über den Parameter *trig_ptm_in:freq_config* in Hz eingestellt. Allerdings können nicht alle Frequenzen beliebig eingestellt werden, da die EIB 8791 mit einem 10 MHz Systemtakt arbeitet und die PTM Frequenz mit einem 16 Bit Prescaler abgeleitet wird. Der Treiber stellt bei Eingabe der Frequenz die nächstmögliche Frequenz ein. Um die eingestellte Frequenz auszulesen, kann die Applikation *trig_ptm_in:freq_config* lesen. Der Wert ist allerdings gerundet und entspricht nicht der exakten Frequenz. Den tatsächlichen Wert kann man erhalten, wenn man Systemtakt und aktuellen Prescaler ausliest:

```
EIB8; sync_clock:freq_nominal und
EIB8; trig_ptm_in:prescaler_config
```

Die Frequenz ergibt sich aus $\text{freq_nominal} / \text{prescaler_config}$.

Genau einstellbare Frequenzen sind beispielsweise:

200 Hz, 250 Hz, 500 Hz, 1 kHz, 2 kHz, 5 kHz, 10 kHz, 20 kHz, 40 kHz, 50 kHz, 80 kHz,
100 kHz, 200 kHz, 400 kHz, 500 kHz, 625 kHz, 1 MHz

Die geringste einstellbare Frequenz ist 152,59 Hz.

- Wird für den internen PTM bei *trig_ptm_in:freq_config* der Wert 0 eingestellt, so wird kein PTM erzeugt. Auf diese Weise kann der interne PTM ausgeschaltet werden.
- Wird beim Einschalten des internen PTMs wiederum die Frequenz eingestellt, die zuletzt aktiv war, so wird der PTM phasenrichtig zum letzten PTM aktiviert.
- Der interne PTM (single-ended) kann mit *EIB8;trig_ptm_in:output;1* am Ausgang der EIB 8791 bereitgestellt werden. Beispielsweise können damit mehrere EIB 8791 mit einem gemeinsamen PTM betrieben werden.
- Um den externen PTM definiert zu aktivieren (Trigger werden an die Achsen weitergeleitet) empfiehlt sich folgende Vorgehensweise:
 - Aktivieren des internen PTMs mit Frequenz 0 (*EIB8;trig_ptm_in:source;internal* und *EIB8;trig_ptm_in:freq_config;0*). Damit werden keine Trigger erzeugt.
 - Durchführung der Konfiguration
 - Umschalten des internen PTMs auf den externen PTM (z.B. *EIB8;trig_ptm_in:source;ext_single*)

2.9.2.2 Ein externer Trigger für alle Achsen

Sollen alle Achsen auf denselben externen Triggereingang (siehe 1.6.1) triggern, so sind folgende Konfigurationsparameter zu senden (Tabelle für Triggereingang 1: **ext1**, analog können **ext2**, **ext3** oder **ext4** verwendet werden):

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
EIB8	<i>trig_ext1_in:enable</i>	1	Aktivieren des externen Triggereingangs (Beim Einschalten der EIB 8791 sind die externen Triggereingänge deaktiviert)
EIB8	<i>trig_ext1_in:terminate</i>	0 1	Terminierung des Triggereingangs: – aus (Default) – ein
EIB8	<i>trig_ext1_in:inverted</i>	0 1	Invertierung des Triggereingangs: – aus (Default) – ein
EIB8	<i>trig_bus1_out:inputs</i> <i>trig_bus1_out:enable</i> <i>trig_bus1_out:transmitter</i>	<i>trig_ext1_in</i> 1 1	Internes Trigger-Routing im Knoten EIB 8791 für externen Trigger
SLOT00	<i>trig_bus1_in:enable</i>	1	Internes Trigger-Routing in den Knoten SLOT für externen Trigger (identisch für alle Slots)
SLOT00:AXIS00	<i>trigger:inputs</i>	<i>trig_bus1_in</i>	Internes Trigger-Routing in den Knoten AXIS für externen Trigger (identisch für alle Achsen)

Hinweise:

- Eine Invertierung des externen Triggersignals ist dann notwendig, wenn die Achse auf die fallende Flanke getriggert werden soll. Standardmäßig triggern die Achsen auf die steigende Flanke.
- Um den externen Trigger definiert zu aktivieren, empfiehlt sich, folgende Vorgehensweise:
 - Zunächst den externen Trigger deaktivieren (*EIB8;trig_ext1_in:enable;0*)
 - Durchführung der Konfiguration
 - Aktivieren des externen Triggers (*EIB8;trig_ext1_in:enable;1*)

2.9.2.3 Mehrere externe Trigger

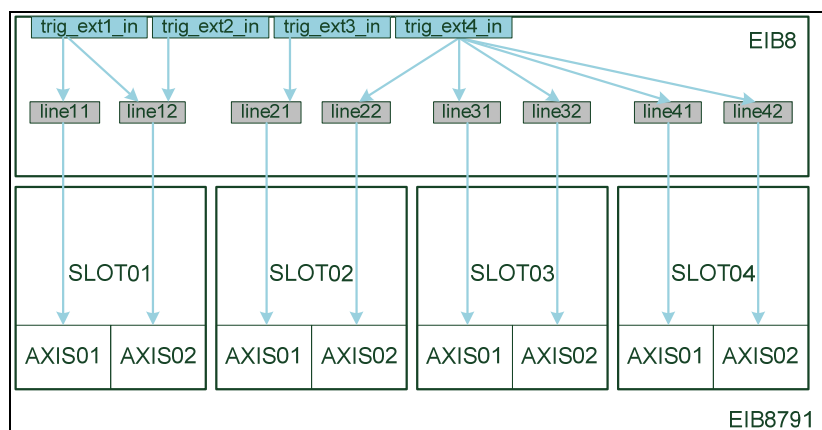
Die Achsen sollen auf verschiedene externe Triggereingänge (siehe 1.6.1) triggern.

Dazu sei folgende Vorgehensweise empfohlen:

- Alle externen Trigger werden konfiguriert und aktiviert (ggf. nur diejenigen, die benötigt werden)
- Jede Achse wird intern auf eine der 8 sogenannten Triggerlines konfiguriert.
- Die externen Trigger werden den Triggerlines zugewiesen. Dabei kann ein externer Trigger mehreren Achsen zugewiesen werden oder eine Achse kann auf mehrere externe Trigger konfiguriert werden.

Ein Beispiel zeigt das folgende Bild:

SLOT01:AXIS01 → trig_line11_out → trig_ext1_in
 SLOT01:AXIS02 → trig_line12_out → trig_ext1_in, trig_ext2_in
 SLOT02:AXIS01 → trig_line21_out → trig_ext3_in
 SLOT02:AXIS02 → trig_line22_out → trig_ext4_in
 SLOT03:AXIS01 → trig_line31_out → trig_ext4_in
 SLOT03:AXIS02 → trig_line32_out → trig_ext4_in
 SLOT04:AXIS01 → trig_line41_out → trig_ext4_in
 SLOT04:AXIS02 → trig_line42_out → trig_ext4_in



Die einzustellenden Konfigurationsparameter zeigt die folgende Tabelle:

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
EIB8	<i>trig_ext1_in:enable</i> <i>trig_ext2_in:enable</i> <i>trig_ext3_in:enable</i> <i>trig_ext4_in:enable</i>	1	Aktivieren der externen Triggereingänge
EIB8	<i>trig_ext1_in:terminate</i> <i>trig_ext2_in:terminate</i> <i>trig_ext3_in:terminate</i> <i>trig_ext4_in:terminate</i>	0 1	Terminierung der Triggereingänge: – aus – ein
EIB8	<i>trig_ext1_in:inverted</i> <i>trig_ext2_in:inverted</i> <i>trig_ext3_in:inverted</i> <i>trig_ext4_in:inverted</i>	0 1	Invertierung des Triggereingänge: – aus – ein
EIB8	<i>trig_line11_out:enable</i> <i>trig_line12_out:enable</i> <i>trig_line21_out:enable</i> <i>trig_line22_out:enable</i> <i>trig_line31_out:enable</i> <i>trig_line32_out:enable</i> <i>trig_line41_out:enable</i> <i>trig_line42_out:enable</i>	1	Internes Trigger-Routing im Knoten EIB8 für einzelne Achsen-Lines
EIB8	<i>trig_line11_out:inputs</i> <i>trig_line12_out:inputs</i> <i>trig_line21_out:inputs</i> <i>trig_line22_out:inputs</i> <i>trig_line31_out:inputs</i> <i>trig_line32_out:inputs</i> <i>trig_line41_out:inputs</i> <i>trig_line42_out:inputs</i>	<i>trig_ext1_in</i> <i>trig_ext1_in, trig_ext2_in</i> <i>trig_ext3_in</i> <i>trig_ext4_in</i> <i>trig_ext4_in</i> <i>trig_ext4_in</i> <i>trig_ext4_in</i> <i>trig_ext4_in</i>	Internes Trigger-Routing: Zuweisung der externen Trigger auf die einzelnen Achsen-Lines (je nach Anwendung, hier nach Beispiel im Bild oben)
SLOT00	<i>trig_line1_in:enable</i> <i>trig_line2_in:enable</i>	1	Internes Trigger-Routing in den Knoten SLOT für einzelne Achsen-Lines (identisch für alle Slots)
SLOT00:AXIS01	<i>trigger:inputs</i>	<i>trig_line1_in</i>	Internes Trigger-Routing in den Knoten AXIS für einzelne Achsen-Lines (identisch für alle Achsen 1)
SLOT00:AXIS02	<i>trigger:inputs</i>	<i>trig_line2_in</i>	Internes Trigger-Routing in den Knoten AXIS für einzelne Achsen-Lines (identisch für alle Achsen 2)

Siehe auch Hinweise in 2.9.2.2.

Weiter ist zu beachten, dass dann, wenn alle externen Trigger gleichzeitig aktiviert werden sollen, die Parameter

EIB8; trig_ext1_in:enable;1

EIB8; trig_ext2_in:enable;1

EIB8; trig_ext3_in:enable;1

EIB8; trig_ext4_in:enable;1

mit einem Funktionsaufruf gesetzt werden sollen. Dies ist beispielsweise mit *eib8_set_config_file()* oder *eib8_set_param_list_string()* möglich (siehe auch 2.6.3).

2.9.3 Konfiguration der Messgeräte

Folgende Parameter können für ein Messgerät eingestellt werden:

Messgerät Konfiguration:

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
SLOT0X:AXIS0X	<i>encoder_config:type</i>	<i>linear</i> <i>rotary</i>	Messgerät Typ: lineares Messgerät Rotatorisches Messgerät
SLOT0X:AXIS0X	<i>encoder_config:interface</i>	<i>1V_pp_0_90</i>	Messgerät Interface: Derzeit nur 1 V _{SS} (0/90° Signale)
SLOT0X:AXIS0X	<i>encoder_config:reference_type</i>	<i>non</i> <i>single</i> <i>distance_coded</i>	Typ der Referenzmarke: Keine Eine Referenzmarke Abstandscodierte Referenzmarke
SLOT0X:AXIS0X	<i>encoder_config:line_cnt</i>	< <i>signal periods</i> >	Zahl der Signalperioden bei rotatorischen Messgeräten
SLOT0X:AXIS0X	<i>encoder_config:ref_increment</i>	< <i>signal periods</i> >	Nominaler Abstand abstandscodierter Referenzmarken in Signalperioden
SLOT0X:AXIS0X	<i>encoder_config:limit_signal_1_present</i>	<i>0</i> <i>1</i>	Limit Signal 1: nicht vorhanden vorhanden
SLOT0X:AXIS0X	<i>encoder_config:limit_signal_2_present</i>	<i>0</i> <i>1</i>	Limit Signal 2: nicht vorhanden vorhanden
SLOT0X:AXIS0X	<i>encoder_config:homing_signal_present</i>	<i>0</i> <i>1</i>	Homing Signal: nicht vorhanden vorhanden

Messgerät Versorgungsspannung:

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
SLOT0X:AXIS0X	<i>encoder:supply_enable</i>	<i>0</i> <i>1</i>	Versorgungsspannung und Positionsberechnung: aus ein

Messgerät Kompensationsalgorithmen:

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
SLOT0X:AXIS0X	<i>encoder_processing:online_comp_enable</i>	<i>0</i> <i>1</i>	Online Kompensation: aus ein
SLOT0X:AXIS0X	<i>encoder_processing:waveform_comp_enable</i>	<i>0</i> <i>1</i>	Signalform Kompensation: aus ein

Messgerät Positionswertefilter:

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
SLOT0X:AXIS0X	<i>pos_value_filter:characteristic</i>	<i>linear</i> <i>parabolic</i>	Filter Charakteristik: linear parabolisch
SLOT0X:AXIS0X	<i>pos_value_filter:bandwidth</i>	<i>low</i> <i>medium</i> <i>high</i>	Filter Bandbreite: niedrig (5 kHz) mittel (10 kHz) hoch (20 kHz)
SLOT0X:AXIS0X	<i>pos_value_filter:measure_time</i>	< value >	Relativer Messzeitpunkt in Bezug auf den Trigger in Nanosekunden. Wertebereich: [-5000 ... 20000]
SLOT0X:AXIS0X	<i>pos_value_filter:active</i>	0 1	Positionswertefilter: aus ein

Dabei ist zu beachten:

- Folgender Ablauf ist einzuhalten:
 - Konfiguration des Messgeräts („*encoder_config*“)
 - „*encoder:supply_enable*“ aktivieren
 - „*encoder_processing:online_comp_enable*“ setzen (optional)
 - „*pos_value_filter:active*“ setzen (optional)
 Die Reihenfolge wird vom Treiber bei Verwendung von Konfigurationsdateien oder Konfigurationslisten automatisch eingehalten.
- encoder_processing:waveform_comp_enable* kann erst aktiviert werden, wenn
 - „*encoder:supply_enable*“ aktiv
 - Vorliegen einer gültigen Referenzposition
- Wird das Messgerät konfiguriert, so wird die Referenzposition ungültig.

2.9.4 Positionsdatenausgabe

Für die Positionsdatenausgabe müssen, wie in obigen Beispielen angeführt, folgende Einstellungen vorgenommen werden:

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
EIB8	<i>pdl_tx_from_slots_to_lane_1:slot_1</i> <i>pdl_tx_from_slots_to_lane_1:slot_1</i> <i>pdl_tx_from_slots_to_lane_1:slot_1</i> <i>pdl_tx_from_slots_to_lane_1:slot_1</i>	1 1 1 1	Internes Routing der Positionsdaten in der EIB 8791. Sollte immer so eingestellt werden.
EIB8			
EIB8	<i>pdl_forwarding_ram_udp:slot_1</i> <i>pdl_forwarding_ram_udp:slot_2</i> <i>pdl_forwarding_ram_udp:slot_3</i> <i>pdl_forwarding_ram_udp:slot_4</i>	„Paketnummern“	Paketnummern, der einzelnen Slots, die per UDP übertragen oder ins RAM geschrieben werden sollen. Format: „0x11,0x12,0x13“
EIB8	<i>pdl_forwarding_ram_udp:mode</i>	„stop“ „recording“ „batch_soft_realtime“ „direct_soft_realtime“	Modus der Positionsdatenausgabe (siehe 2.8.6)
SLOT00:AXIS00	<i>pdl:output_active</i>	0 1	Aktivierung der Positionsdatenausgabe der einzelnen Achsen – aus – ein

Für den UDP Transfer muss darüber hinaus eingestellt werden:

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
EIB8	udp_transfer:udp_dest_mac udp_transfer:udp_dest_ip udp_transfer:udp_dest_port	„IP-Adresse“ „MAC-Adresse“ „UDP-Port“	UDP Ziel in Form von MAC-, IP-Adresse und Port (siehe 2.8.6)

2.9.5 Synchroner Start der Positionsdatenpakete

Positionsdaten werden in der EIB 8791 aufgrund von Trigger-Ereignissen erzeugt. Dabei wird in vielen Anwendungen eine Trigger-Quelle für alle Achsen verwendet. Von Bedeutung ist bei der Auswertung der Positionsdatenpakete, welche Daten zum selben Trigger-Ereignis gehören. Ebenso von Bedeutung ist, ob Pakete von einzelnen Trigger-Ereignissen verloren gegangen sind, weil beispielsweise bei der UDP Übertragung der Host kurzzeitig überlastet war. Im folgenden Abschnitt soll darauf eingegangen werden, was bei der Konfiguration der EIB 8791 beachten werden soll und wie Fehler erkannt werden.

Bei einer gemeinsamen Trigger-Quelle für Achsen sollte es das Ziel sein, dass alle Positionsdatenpakete mit dem gleichen Frame-Zähler generiert werden. Dabei ist der Zähler beim ersten Trigger-Ereignis 0 und wird dann bei jedem weiteren um 1 inkrementiert. Aufgrund des Wertebereichs läuft der Zähler bei 255 über und beginnt wieder bei 0 (im Polling Modus ist der Frame-Zähler immer 0, ein inkrementierender Frame-Zähler kann somit nur im Modus UDP oder RAM-Recording ausgewertet werden).

In der Applikation können somit Positionsdatenpakete mit dem gleichen Frame-Zähler einem Trigger-Ereignis zugeordnet werden. Verloren gegangene Pakete können aufgrund von Lücken im Paketzähler erkannt werden. Für den Empfang von Positionsdaten über UDP liefert die Funktion `eib8_get_udp_synchronous_pdl()` alle Positionsdatenpakete mit einem gleichen Frame-Zähler und signalisiert, wenn Positionsdatenpakete verloren gehen, d.h. wenn Lücken im Frame-Zähler auftreten (siehe 2.8.6.3).

Damit alle Positionsdatenpakete beim ersten Trigger-Ereignis ausgelöst werden und mit dem Frame-Zähler 0 beginnen, ist folgende Vorgehensweise vorteilhaft (genau in der angegebenen Reihenfolge):

Nr	Schritt	Erläuterung	Funktion
1	Zurücksetzen der Konfiguration	Die Konfiguration der EIB 8791 wird zurückgesetzt (Deaktivierung von Triggern, Positionsdatenausgabe, UDP etc.), so dass eine neue Konfiguration ohne Einschränkungen geladen werden kann.	<code>eib8_reset_idle()</code>
2	Neue Konfiguration ohne Aktivierung der Trigger	Wie in den obigen Abschnitten beschrieben, wird die gewünschte Konfiguration in die EIB 8791 geladen. Dabei wird der Trigger, der alle Achsen steuert, noch nicht aktiviert. In 2.9.2 ist beschrieben, welche Parameter die Trigger aktivieren. z.B. <code>EIB8;trig_ptm_in:freq_config;0</code>	Siehe 2.6.3
3	Zurücksetzen der Frame-Zähler	Der Frame – Zähler wird per Kommando auf 0 gesetzt.	<code>eib8_reset_pdl_frame_counter (&myEIB8Handle, „SLOT00:AXIS00“)</code>
4	Starten des UDP Empfang	Starten des UDP Empfangs im Treiber: Positionsdatenpakete werden in den Positionsdatenspeicher geschrieben	<code>eib8_start_udp_pdl_sampling()</code>
5	Aktivierung der Trigger	Mit der Aktivierung des Triggers werden Positionsdaten erzeugt. z.B. <code>EIB8;trig_ptm_in:freq_config;1000</code>	Siehe 2.6.3 z.B. <code>eib8_set_param_string()</code>

Hinweis:

- Die obige Vorgehensweise ist auch im Programmierbeispiel „example_udp.c“ (Treiber CD) umgesetzt.
- Wird die obige Reihenfolge nicht eingehalten, kann es beispielsweise vorkommen, dass ein Trigger bereits aktiv ist und dabei die Achsen konfiguriert werden. Da die Achsen vom Treiber nacheinander konfiguriert werden, würden eine Achse bereits mit der Positionsdatenausgabe beginnen während eine andere erst später startet. Die Frame-Zähler der verschiedenen Achsen wären somit beim gleichen Trigger-Ereignis verschieden.

2.10 Weitere Anwendungsfälle

2.10.1 Recording von Positionsdaten

Positionsdaten können im internen RAM der EIB 8791 gespeichert werden (siehe 2.7.1).

Die maximale Anzahl sind 10 Millionen Positionsdatenpakete. Dabei kann gewählt werden, ob der Speicherbereich einmal vollgeschrieben wird und dann seitens der EIB 8791 die Aufzeichnung automatisch beendet wird (Single Shot) oder ob der Speicherbereich als Ringpuffer betrieben wird. Im Ringpufferbetrieb werden dann, wenn der Speicher voll ist, die ältesten Daten wieder überschrieben, solange bis die Applikation die Aufzeichnung abbricht. Auch im Single Shot Betrieb kann die Aufzeichnung von der Applikation vorzeitig abgebrochen werden.

Um die Daten dann aus dem RAM der EIB 8791 abzuholen, muss die Aufzeichnung beendet sein. Dazu stellt der Treiber entsprechende Funktionen zur Verfügung. Sowohl im Single Shot als auch im Ringpufferbetrieb erhält die Applikation die Daten in chronologischer Reihenfolge.

Zu beachten ist, dass die Downloadgeschwindigkeit der aufgezeichneten Daten ca. 20 000 Positionsdatenpakete je Sekunde beträgt. Bei maximaler Zahl an aufgezeichneten Positionsdatenpaketen dauert somit der Transfer ca. 10 min.

2.10.1.1 Konfiguration

Für das RAM Recording sind die Konfigurationsparameter ähnlich einzustellen wie bei einem UDP Transfer (beispielsweise die Standardkonfiguration). Auch die Hinweise für einen synchronen Start der Positionsdatenpakete gelten genauso (siehe 2.10.1).

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
EIB8	<i>udp_transfer:udp_dest_mac</i> <i>udp_transfer:udp_dest_ip</i> <i>udp_transfer:udp_dest_port</i>	“ “ “	UDP Ziel hier nicht von Bedeutung
EIB8	<i>pdl_forwarding_ram_udp:mode</i>	„recording“	Modus Positionsdatenausgabe auf Recording, Recording ist damit in der EIB 8791 aktiviert.
EIB8	<i>recording:mode_ringbuffer</i>	0 1	Ringpufferbetrieb - aus (Single Shot Betrieb) - ein (Ringpufferbetrieb)
EIB8	<i>recording:packets_buffer</i>	<N>	Zahl der aufzunehmenden Positionsdatenpakete (da die EIB 8791 nur Vielfache von 32 verarbeiten kann, wird der Wert ggf. reduziert bzw. auf mindesten 32 gesetzt)

2.10.1.2 Aufzeichnung beenden

Um die Aufzeichnung seitens der Applikation zu beenden, wird der Modus in der EIB 8791 zurückgesetzt:

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
EIB8	<i>pdl_forwarding_ram_udp:mode</i>	„stop“	Modus Positionsdatenausgabe auf Stop (kein UDP Transfer, kein RAM Recording)

Der Modus *pdl_forwarding_ram_udp:mode* wird von der EIB 8791 automatisch auf „stop“ gesetzt, wenn im Single Shot Betrieb die Zahl der aufzunehmenden Positionsdatenpakete erreicht ist.

2.10.1.3 Status auslesen

Um den Status des aktuellen Recordings auszulesen, stehen folgende Parameter zur Verfügung:

Status			Erläuterung
Knoten	Parameter	Werte	
EIB8	<i>recording:active</i>	0 1	Recording aktiv – nicht aktiv, d.h. nicht aktiv oder beendet – aktiv, d.h. noch nicht beendet
EIB8	<i>recording:buffer_total</i>	<N>	Zahl der aufzunehmenden Positionspakete
EIB8	<i>recording:buffer_fill</i>	<X>	Zahl der bereits aufgenommenen Positionspakete Im Ringpufferbetrieb steigt die Zahl von 0 bis N und verbleibt dann bei N, bis die Aufzeichnung abgeschlossen ist.

2.10.1.4 Daten von der EIB 8791 abholen

Zum Abholen der Daten von der EIB 8791 muss das Recording beendet sein (*recording:active* muss 0 sein).

Der Treiber bietet dazu zwei Funktionen an:

```
eib8_get_rec_pdl_samples_buffer()
eib8_get_rec_pdl_samples_file()
```

Zum einen können die Positionsdatenpakete in einen Puffer (Array) geschrieben werden und zum anderen in eine Textdatei. Beiden Funktionen kann ein Timeout konfiguriert werden, um das Blockieren der Funktion zu verhindern (die bis zum Timeout abgeholten Daten sind gültig). Die Daten werden im Standardformat für Positionsdaten *eib8_pdl_packet_t* bereitgestellt.

Hinweis:

Auf der Treiber CD befindet sich das Beispiel „example_recording.c“, das die Vorgehensweise veranschaulicht.

2.10.2 Referenzfahrt

In 1.3 ist beschrieben, wie Referenzmarken in der EIB 8791 behandelt werden. Voraussetzungen für eine Referenzfahrt sind:

- Messgerät ist konfiguriert:
 - *encoder_config:type*
 - *encoder_config:interface*
 - *encoder_config:reference_type*
 - *encoder_config:line_cnt*
- Für abstandscodierte Referenzmarken zusätzlich:
 - *encoder_config:ref_increment* ist konfiguriert
- *encoder:supply_enable* ist aktiviert.
- *encoder_processing:waveform_comp_enable* muss deaktiviert sein

Die Referenzfahrt kann für eine oder mehrere Achsen gleichzeitig mit folgender Funktion gestartet werden (mit der Funktion kann auch die Referenzmarkensuche abgebrochen werden):

- *eib8_reference_mark_search()*

Der Status der Referenzmarkensuche kann mit folgenden Parametern überwacht werden:

Status			Erläuterung
Knoten	Parameter	Werte	
SLOT0X:AXIS0X	<i>ref_mark:running</i>	0 1	Referenzmarkensuche: nicht aktiv aktiv
SLOT0X:AXIS0X	<i>ref_mark:valid</i>	0 1	Gültige Referenzposition vorhanden: nicht gültig gültig
SLOT0X:AXIS0X	<i>ref_mark:ref_pos_1</i>	< reference pos single >	Referenzposition einer einzelnen Referenzmarke
SLOT0X:AXIS0X	<i>ref_mark:ref_pos_dist_coded</i>	< reference pos distance coded >	Referenzposition einer abstandscodierten Referenzmarke

Ablauf in der Applikation:

- Konfiguration der Achse
- Start der Referenzmarkensuche mit `eib8_reference_mark_search()`
`ref_mark:running` liefert den Zustand „aktiv“ (1)
- Warten bis `ref_mark:running` „nicht aktiv“ (0). Die Referenzmarkensuche wurde beendet.
- Überprüfen, ob die Referenzmarkensuche erfolgreich war: `ref_mark:valid` muss „gültig“ (1) sein.
- Auslesen der Referenzposition
Einzelne Referenzmarke: `ref_mark:ref_pos_1`
Abstandscodierte Referenzmarke: `ref_mark:ref_pos_dist_coded`

Die Referenzposition ist dabei ein 32 Bit Wert (`eib8_get_param_hex32()`), der den Periodenzähler wiedergibt, bei dem die Referenzmarke gefunden wurde. Bei abstandscodierten Referenzmarken wird der Wert des Periodenzählers an der Nullposition angegeben.

Die Referenzposition kann auch als Positionsdatenpaket versendet werden (siehe 2.7.3.3).

Siehe auch Programmierbeispiel „example_ref_pos average.c“.

2.10.3 Positionswert Initialisieren

Der Positionswert der einzelnen Achsen kann mit der Funktion

`eib8_set_position_value()`

initialisiert werden.

Beispiel:

Alle Achsen auf 0:

`eib8_set_position_value(&myEIB8Handle, „SLOT00:AXIS00“, 0)`

Dabei wird nur der Periodenzähler gesetzt. Die Phase (Interpolationswert innerhalb der Signalperiode) bleibt erhalten. Der gewünschte Positionswert ist dabei im Format anzugeben, das in 2.7.2 beschrieben ist.

Beim Setzen des Positionszählers ist zu beachten:

- Das Messgerät muss konfiguriert sein (wie für eine Referenzfahrt, siehe 2.10.2).
- Die Signalformkompensation darf nicht aktiv sein.
- Nach dem Setzen des Positionszählers ist die Referenzposition ungültig.

Das Setzen des Positionszählers sollte vor der Referenzfahrt durchgeführt werden.

2.10.4 Positionsfehler löschen

Tritt bei der Positionsberechnung in der EIB 8791 ein Fehler auf, so wird dies im Positionsdatenpaket angezeigt (2.7.3.1). Der Fehler bleibt solange bestehen, bis er mit

`eib8_clear_position_error()`

wieder gelöscht wird.

Beispiel:

Slot 1 – Achse 1:

`eib8_clear_position_error(&myEIB8Handle, „SLOT01:AXIS01“)`

Alle Achsen:

`eib8_clear_position_error(&myEIB8Handle, „SLOT00:AXIS00“)`

Dabei kann das Kommando auch ausgeführt werden, wenn kein Fehler vorhanden ist.

2.10.5 Korrekturwertaufnahme zur Signalformkompensation

Bevor die Signalformkompensation aktiviert werden kann, muss eine Korrekturwertaufnahme durchgeführt werden (siehe 0). Das Ergebnis der Korrekturwertaufnahme wird im Flash gespeichert und steht somit nach einem erneuten Einschalten der EIB 8791 wieder zur Verfügung.

2.10.5.1 Konfiguration und Voraussetzungen der Korrekturwertaufnahme

Für die Korrekturwertaufnahme müssen folgende Voraussetzungen erfüllt sein:

- Das Messgerät muss konfiguriert sein (wie für eine Referenzfahrt, siehe 2.10.2).
- Die Online Kompensation sollte für das Messgerät aktiviert sein.
- Für das Messgerät muss eine gültige Referenzierung vorliegen.
- Die Signalformkompensation muss für beide Messgeräte (Achsen) eines Slots deaktiviert sein. Es ist empfehlenswert, die Signalformkompensationen für alle Messgeräte der EIB 8791 abzuschalten, wenn eine Korrekturwertaufnahme durchgeführt werden soll.

Darüber hinaus sind für die Korrekturwertaufnahme folgende Parameter zu konfigurieren:

Konfiguration			Erläuterung
Knoten	Parameter	Werte	
SLOT0X:AXIS0X	<i>waveform_corr_run:</i> <i>start_value</i>	< signal periods >	Anfangswert des Korrekturbereichs in Signalperioden bezogen auf die Referenzposition
SLOT0X:AXIS0X	<i>waveform_corr_run:</i> <i>basic_frequency</i>	1 2	Signalgrundfrequenz: eine Signalperiode zwei Signalperioden
SLOT0X:AXIS0X	<i>waveform_corr_run:</i> <i>direction</i>	<i>positive</i> <i>negative</i>	Verfahrrichtung: Positive Zählrichtung Negative Zählrichtung
SLOT0X:AXIS0X	<i>waveform_corr_run:</i> <i>num_correction_points</i>	< num >	Anzahl der Korrekturstützpunkte
SLOT0X:AXIS0X	<i>waveform_corr_run:</i> <i>dist_correction_points</i>	< signal periods >	Abstand der Korrekturstützpunkte in Signalperioden

2.10.5.2 Starten und Überwachen der Korrekturwertaufnahme

Sind die Voraussetzungen erfüllt, kann die Korrekturwertaufnahme mit *eib8_exec_waveform_corr_run()* gestartet werden.

Dabei ist zu beachten:

- An einem Slot kann nur eine Korrekturwertaufnahme gestartet werden. Für mehrere Slots kann die Korrekturwertaufnahme parallel erfolgen. Für die einzelnen Messgeräte (Achsen) eines Slots muss die Korrekturwertaufnahme nacheinander durchgeführt werden.
- Bei der Funktion *eib8_exec_waveform_corr_run()* muss eine sogenannte „StorageID“ angegeben werden. Diese entspricht einem Flashspeicherplatz in dem Slot, für den die Korrekturdaten abgelegt werden.

Folgende Speicherplätze sollten gewählt werden:

SLOT0X:AXIS01 → *StorageID = 0*

SLOT0X:AXIS02 → *StorageID = 1*

- Die Korrekturwertaufnahme kann vor oder nach *eib8_exec_waveform_corr_run()* begonnen werden. Es muss sichergestellt werden, dass nach dem Aufruf des genügend Samples vor der Startposition überfahren werden.

Für die Überwachung der Korrekturwertaufnahme stehen folgende Parameter zur Verfügung:

Status			Erläuterung
Knoten	Parameter	Werte	
SLOT0X:AXIS0X	<i>waveform_corr_status:improvement</i>	< value >	Wert zwischen 0 - 100, der den Verbesserungsfaktor der Kompensation angibt.
SLOT0X:AXIS0X	<i>waveform_corr_status:successful</i>	0 1	1, wenn Korrekturwertaufnahme erfolgreich beendet
SLOT0X:AXIS0X	<i>waveform_corr_status:failed</i>	0 1	1, wenn Korrekturwertaufnahme beendet und fehlerhaft
SLOT0X:AXIS0X	<i>waveform_corr_status:running</i>	0 1	1, wenn Korrekturwertaufnahme läuft (Datenaufnahme oder Datenauswertung)
SLOT0X:AXIS0X	<i>waveform_corr_status:data_acquisition_done</i>	0 1	1, wenn Korrekturwertaufnahme läuft. Die Fahrt kann bereits beendet werden. Berechnungen werden noch auf der EIB 8791 durchgeführt.
Fehlerursachen, wenn <i>waveform_corr_status:failed</i> gesetzt ist (1):			
SLOT0X:AXIS0X	<i>waveform_corr_status:error_underrun</i>	0 1	1, wenn zu wenige Samples vor <i>waveform_corr_run:start_value</i> aufgenommen worden sind.
SLOT0X:AXIS0X	<i>waveform_corr_status:error_direction</i>	0 1	1, wenn die Fahrt in der falschen Richtung durchgeführt wurde (<i>waveform_corr_run:direction</i>).
SLOT0X:AXIS0X	<i>waveform_corr_status:error_speed_low</i>	0 1	1, wenn die Fahrt mit zu geringer Geschwindigkeit durchgeführt wurde.
SLOT0X:AXIS0X	<i>waveform_corr_status:error_speed_high</i>	0 1	1, wenn die Fahrt mit zu hoher Geschwindigkeit durchgeführt wurde.
SLOT0X:AXIS0X	<i>waveform_corr_status:error_internal_bit</i>	0 1	1, wenn ein interner Fehler aufgetreten ist. Bitte HEIDENHAIN Messgeräte-Support kontaktieren.

Ablauf der Überwachung:

- Prüfen, ob *waveform_corr_status:running* und *waveform_corr_status:data_acquisition_done* aktiv.
 - *waveform_corr_status:data_acquisition_done* = 0: die Fahrt kann beendet werden
 - *waveform_corr_status:running* = 0: die Korrekturwertaufnahme ist beendet
- Prüfen, ob der Korrekturwertaufnahme erfolgreich war:
 - *waveform_corr_status:successful* = 1: Korrekturwertaufnahme erfolgreich
 - *waveform_corr_status:failed* = 1: Korrekturwertaufnahme nicht erfolgreich
- Wenn Korrekturwertaufnahme nicht erfolgreich, Fehlerzustände auswerten und die Korrekturwertaufnahme wiederholen:
 - *waveform_corr_status:error_underrun*
 - *waveform_corr_status:error_direction*
 - *waveform_corr_status:error_speed_low*
 - *waveform_corr_status:error_speed_high*
 - *waveform_corr_status:error_internal_bit*
 - *waveform_corr_status:error_acceleration_high*
 - *waveform_corr_status:error_position*
- Wenn Korrekturwertaufnahme erfolgreich, Verbesserungsfaktor (*waveform_corr_status:improvement*) auslesen und bewerten:
 Der Verbesserungsfaktor (0 - 100) gibt an, welche Verbesserung der Genauigkeit bei aktivierter Signalformkompensation zu erwarten ist. Hat der Faktor den Wert 0, so tritt keine Verbesserung der Genauigkeit ein. Die Kompensation sollte nicht verwendet werden.

2.10.5.3 Abschließen und Speichern Korrekturwertaufnahme

Der Speicherplatz für die Korrekturwertaufnahme wurde bereits mit der Funktion `eib8_exec_waveform_corr_run()` festgelegt. Dieser muss dem Messgerät (der Achse) zugewiesen werden.

```
eib8_assign_storage_id()
```

In 2.10.5.2 wurden die Speicherplätze folgendermaßen definiert:

```
SLOT0X:AXIS01 → StorageID = 0
```

```
SLOT0X:AXIS02 → StorageID = 1
```

Somit ist die Funktion für SLOT01 folgendermaßen aufzurufen:

```
/* SLOT01:AXIS01 */
unit32_t u32StorageID = 0u;
eib8_assign_storage_id (&myEIB8Handle , „SLOT01:AXIS01“, u32StorageID, 1u, 1000u);
/* SLOT01:AXIS02 */
unit32_t u32StorageID = 1u;
eib8_assign_storage_id (&myEIB8Handle , „SLOT01:AXIS02“, u32StorageID, 1u, 1000u);
```

Die Signalformkorrektur ist nun gespeichert und steht nach jedem Einschalten der EIB 8791 zur Verfügung.

2.10.5.4 Aktivieren der Signalformkompensation

Voraussetzung für die Aktivierung der Signalformkompensation ist:

- Das Messgerät ist konfiguriert und referenziert (siehe 2.10.2)
- Eine Korrekturwertaufnahme wurde durchgeführt und dem Messgerät (der Achse) zugewiesen.

Die Aktivierung erfolgt mit:

```
encoder_processing:waveform_comp_enable = 1
```

2.10.6 Netzwerkparameter setzen

Die Netzwerkparameter der EIB 8791 können als Konfigurationsparameter gesetzt werden. Sie werden dauerhaft im Gerät gespeichert. Veränderte Netzwerkparameter werden erst nach einem Reset (`eib8_reset()`) oder erneutem Einschalten der EIB 8791 wirksam.

Konfiguration			Erläuterung
Knoten	Parameter	Default	
EIB8	<code>network_user:ip_address</code>	192.168.168.2	IP-Adresse der EIB 8791
EIB8	<code>network_user:netmask</code>	255.255.255.0	Subnetzmaske
EIB8	<code>network_user:gateway</code>	192.168.168.1	Standardgateway
EIB8	<code>network_user:dhcp_enabled</code>	0	DHCP verwenden (0: nein, 1:ja)
EIB8	<code>network_user:dhcp_timeout</code>	30	Timeout Suche DHCP Server [sec] (relevant, wenn DHCP verwendet wird)
EIB8	<code>network_user:tcp_port</code>	1050	TCP Port der EIB 8791 (der Host verbindet sich mit diesem Port bei <code>eib8_connect_tcp()</code>)
EIB8	<code>network_user:tcp_time</code>	100	Standard. Sollte nicht verändert werden.
EIB8	<code>network_user:ftp_port</code>	21	FTP Port der EIB 8791 zur Übertragung von beispielsweise Softwarepaketen
EIB8	<code>network_user:host_name</code>	EIB 8791 12345678	Netzwerkname der EIB 8791
EIB8	<code>network_user:ftp_user</code>	anonymous	User für den FTP Transfer
EIB8	<code>network_user:ftp_password</code>	*	Passwort für den FTP Transfer

Beispiel zum Setzen der IP-Adresse:

```
eib8_set_param_string(&myEIB8Handle, „EIB8“, network_user:ip_address, „192.168.168.10“)
```


2.10.7 Software Update prüfen

Wie in „1.11 Firmware Update“ beschrieben, werden neue Softwarepakete per FTP an die EIB 8791 gesendet. Nach einer erfolgreichen Übertragung, beginnt die EIB 8791 das Softwarepaket zu verarbeiten. Dabei beginnt die LAN LED an der Frontblende zu blinken.

Während der Verarbeitung kann mit der EIB 8791 nicht kommuniziert werden. Die Funktionen des Treibers geben den Fehler *EIB8_ERROR_EIB8_BUSY_UPDATE (5014)* bzw. *"EIB8 busy with update"* aus.

Die EIB 8791 hat das Update beendet, wenn die LAN LED an der Frontblende wieder kontinuierlich leuchten bzw. wenn die oben genannte Fehlermeldung nicht mehr auftritt.

Bevor ein Reset der EIB 8791 durchgeführt wird, sollten folgende Statusparameter zur Überprüfung des Updates ausgelesen werden:

Status			Erläuterung
Knoten	Parameter	Werte	
EIB8	<i>update_info:update_error_eib8</i> <i>update_info:update_error_slot_1</i> <i>update_info:update_error_slot_2</i> <i>update_info:update_error_slot_3</i> <i>update_info:update_error_slot_4</i>	0 1	Updatestatus: – kein Fehler aufgetreten – Fehler aufgetreten

Bei einem erfolgreichen Update haben alle fünf Parameter den Wert 0. Sind Fehler aufgetreten, dann sollte zunächst das Update wiederholt werden (erneutes Senden der Update Datei über FTP an die EIB 8791).

Wenn weiterhin Fehler auftreten, dann sollten die Events ausgelesen werden (siehe 2.8.2, Abspeichern in eine Datei). Bitte setzen Sie sich anschließend mit dem Messgeräte-Support bei HEIDENHAIN in Verbindung.

Zuletzt sollte ein Reset ausgeführt (*eib8_reset()*) und die Softwareversion der EIB 8791 ausgelesen werden (die neue Softwareversion ist erst nach dem Reset sichtbar).

Status			Erläuterung
Knoten	Parameter	Werte	
EIB8	<i>system_info:device_user_software</i>	<Version>	Version der EIB 8791 Software z.B. 816477-04-00-A

Anmerkung:

Wird zum Update der Firmware die „EIB8 Application“ verwendet, so werden dort die Fehler automatisch geprüft.

2.10.8 Analog und Digitale Ein-/Ausgänge

Die analogen und digitalen Eingänge werden im Treiber als Parameter gelesen und geschrieben.

2.10.8.1 Analog Eingänge

Die vier analogen Eingänge (siehe 1.7.2 Analog IN) können mit *eib8_get_param_hex32()* abgefragt werden:

Status			Erläuterung
Knoten	Parameter	Werte	
EIB8	<i>analog_in:analog_1</i> <i>analog_in:analog_2</i> <i>analog_in:analog_3</i> <i>analog_in:analog_4</i>	<Wert>	Die Analogwerte liegen im Bereich 0 – (2 ¹⁴ – 1)

2.10.8.2 Digitale Eingänge

Die acht digitalen Eingänge (siehe 1.7.1 Digital IN) können mit `eib8_get_param_hex32()` abgefragt werden:

Status			Erläuterung
Knoten	Parameter	Werte	
<i>EIB8</i>	<i>digital_in:bits8</i>	<Wert>	Die digitalen Eingänge werden bitcodiert ausgegeben. Wertebereich: 0x0000 0000 – 0x0000 00FF Digital In 1: Maske 0x0000 0001 Digital In 2: Maske 0x0000 0002 Digital In 3: Maske 0x0000 0004 ... Digital 8 In: Maske 0x0000 0080

2.10.8.3 Externe Trigger Ausgänge als digitaler Ausgang

Die vier externen Triggerausgänge an der Rückseite der EIB 8791 (siehe 1.6.1 *Asynchroner Betrieb mit Trigger*) können auch als digitale Ausgänge verwendet werden.

Konfiguration zur Verwendung als digitaler Ausgang (Tabelle für Triggerausgang 1: **ext1**, analog können **ext2**, **ext3** oder **ext4** verwendet werden):

Konfiguration			Erläuterung
Knoten	Parameter	Wert	
<i>EIB8</i>	<i>trig_ext1_out:enable</i>	<i>0</i>	Deaktivierung der Funktion Trigger
<i>EIB8</i>	<i>trig_ext1_out:gpio_mode</i>	<i>1</i>	Aktivierung der Funktion Output

Gesetzt wird dann der Ausgang mit dem Parameter:

Konfiguration			Erläuterung
Knoten	Parameter	Wert	
<i>EIB8</i>	<i>trig_ext1_out:gpio_set</i>	<i>0</i> <i>1</i>	Zustand des Ausgangs: – logisch 0 (low level) – logisch 1 (high level)

3 Anhang

3.1 Beispielprogramme in C

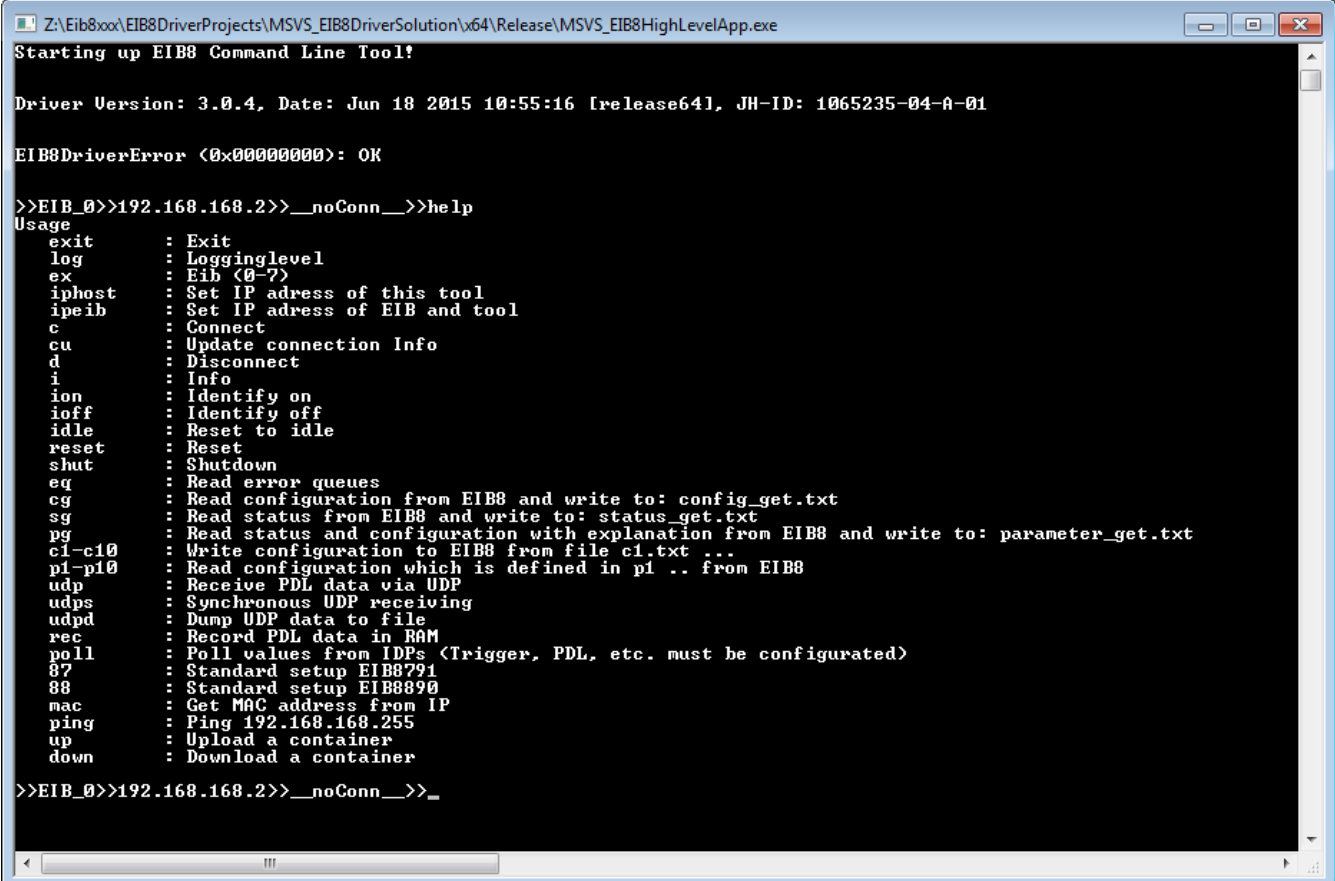
Auf der Treiber CD befinden sich einige Beispiele, die die Anwendung des Treibers zeigen.

Beispiel	Erläuterung
<i>example_identify.c</i>	<ul style="list-style-type: none">• Aktivierung und Deaktivierung des Blinkmodus der LAN LED an der EIB 8791
<i>example_poll.c</i>	<ul style="list-style-type: none">• Erzeugung von Positionsdaten mit einer Standardkonfiguration• Abfrage der Positionsdaten im Polling Modus.
<i>example_udp.c</i>	<ul style="list-style-type: none">• Erzeugung von Positionsdaten mit einer Standardkonfiguration• Übertagung der Pakete per UDP• Lesen der Pakete aus dem Positionsdatenspeichers des Treibers.
<i>example_recording.c</i>	<ul style="list-style-type: none">• Erzeugung von Positionsdaten mit einer Standardkonfiguration• Speichern der Positionsdatenpakete im internen RAM der EIB 8791• Abholen der Pakete durch den Treiber
<i>example_ref_pos_average.c</i>	<ul style="list-style-type: none">• Erzeugung von Positionsdaten mit einer Standardkonfiguration• Positionswert auf 0 setzen• Referenzfahrt• Übertagung der Pakete per UDP• Lesen der Pakete aus dem Positionsdatenspeichers des Treibers.• Berechnung Mittelwert aus 4 Achsen
<i>example_multi_head_system.c</i>	<ul style="list-style-type: none">• Erzeugung von Positionsdaten mit einer Standardkonfiguration• Positionswert auf 0 setzen• Referenzfahrt• Übertagung der Pakete per UDP• Lesen der Pakete aus dem Positionsdatenspeichers des Treibers.• Berechnung Mittelwert aus n Achsen• Korrekturwertaufnahme und Signalformkompensation

Dabei werden bei jedem Beispiel die Event Queues geprüft und Standardkonfigurationen in die EIB 8791 geladen.

3.2 Kommandozeilentool

Das Kommandozeilentool EIB8AppXX.exe (Windows) bzw. EIB8App (Linux), das mit der Treiber CD bereitgestellt wird, benötigt keine Installation und kann für einfache Tests mit der EIB 8791 genutzt werden. Mit der Eingabe ‚help‘ werden die Kommandos des Tools angezeigt.



```
Z:\Eib8xxx\EIB8DriverProjects\MSVS_EIB8DriverSolution\x64\Release\MSVS_EIB8HighLevelApp.exe
Starting up EIB8 Command Line Tool!

Driver Version: 3.0.4, Date: Jun 18 2015 10:55:16 [release64], JH-ID: 1065235-04-A-01

EIB8DriverError (0x00000000): OK

>>EIB_0>>192.168.168.2>>_noConn_>>help
Usage
exit      : Exit
log       : Logginglevel
ex        : Eib (0-7)
iphost    : Set IP address of this tool
ipeib     : Set IP address of EIB and tool
c         : Connect
cu        : Update connection Info
d         : Disconnect
i         : Info
ion       : Identify on
ioff      : Identify off
idle      : Reset to idle
reset     : Reset
shut      : Shutdown
eq        : Read error queues
cg        : Read configuration from EIB8 and write to: config_get.txt
sg        : Read status from EIB8 and write to: status_get.txt
pg        : Read status and configuration with explanation from EIB8 and write to: parameter_get.txt
cl-c10    : Write configuration to EIB8 from file cl.txt ...
pl-p10    : Read configuration which is defined in pl .. from EIB8
udp       : Receive PDL data via UDP
udps      : Synchronous UDP receiving
udpd      : Dump UDP data to file
rec       : Record PDL data in RAM
poll      : Poll values from IDPs (Trigger, PDL, etc. must be configured)
87        : Standard setup EIB8791
88        : Standard setup EIB8890
mac       : Get MAC address from IP
ping      : Ping 192.168.168.255
up        : Upload a container
down      : Download a container

>>EIB_0>>192.168.168.2>>_noConn_>>_
```

Folgende Tabelle erläutert für Standardanwendungen relevante Kommandos:

Kommando	Erläuterung
<i>exit</i>	Beendet das Kommandozeilentool
<i>log</i>	Legt den Logging Level des Tools fest. <ul style="list-style-type: none"> • 0: Logging deaktiviert • 2: Warnungen und Fehler (Default) • 5: Warnungen, Fehler und die komplette Kommunikation mit der EIB 8791 Das Kommandozeilentool schreibt Logging Nachrichten automatisch über UDP an den LogServer. Zum Anzeigen der Nachrichten EIB8LogServerXX.exe starten.
<i>e0</i> <i>e1</i> ... <i>e7</i>	Das Kommandozeilentool kann bis zu 8 EIB 8791 bedienen. Mit dem Kommando <i>e1</i> beispielsweise kann auf die EIB 8791_1 umgeschaltet werden. Die aktive EIB 8791 wird im Prompt angezeigt.
<i>iphost</i>	Setzen der IP-Adresse, mit der sich das Tool mit der EIB 8791 verbindet. (Default 192.168.168.2).
<i>ipeib8</i>	Konfigurieren der IP-Adresse der EIB 8791. Diese wird erst nach einem Neustart des Geräts wirksam (Kommando <i>reset</i>) und dauerhaft im Gerät gespeichert.
<i>c</i>	Verbindung mit der EIB 8791 aufbauen. Anzeigen der grundlegenden Systemzustände (<i>eib8_connect_tcp()</i>).
<i>cu</i>	Aktualisierung der grundlegenden Zustände (<i>eib8_connect_result_update()</i>).
<i>d</i>	Verbindung zur EIB 8791 trennen (<i>eib8_disconnect_tcp()</i>).
<i>i</i>	Softwareversion der EIB 8791 Module auslesen.
<i>ion</i>	Aktivierung Blinkmodus der LAN LED (<i>eib8_identify()</i>).
<i>ioff</i>	Deaktivierung Blinkmodus der LAN LED (<i>eib8_identify()</i>).
<i>idle</i>	Zurücksetzen der Konfiguration der EIB 8791 (<i>eib8_reset_idle()</i>).
<i>reset</i>	Neustart der EIB 8791 (<i>eib8_reset()</i>)
<i>shut</i>	Shutdown der EIB 8791 einleiten (<i>eib8_shutdown()</i>).
<i>eq</i>	Auslesen und Anzeigen der Event Queues. Zudem werden die Events in events.txt im Verzeichnis des Tools abgelegt (Ist die Datei bereits vorhanden, werden die Events an die vorhandenen angehängt).
<i>cg</i>	Gesamtkonfiguration der EIB 8791 auslesen und anzeigen. Zudem wird die Konfiguration in config_get.txt im Verzeichnis des Tools abgelegt.
<i>sg</i>	Gesamtstatus der EIB 8791 auslesen und anzeigen. Zudem wird der Status in status_get.txt im Verzeichnis des Tools abgelegt.
<i>pg</i>	Alle Parameter der EIB 8791 auslesen und anzeigen. Zudem werden die Parameter in parameter_get.txt im Verzeichnis des Tools abgelegt.
<i>c1</i> <i>c2</i> ... <i>c10</i>	Konfigurationsdatei an die EIB 8791 senden. Dabei sendet <i>c1</i> beispielsweise <i>c1.txt</i> im Verzeichnis des Tools an die EIB 8791. Die Dateinamen <i>c1.txt</i> – <i>c10.txt</i> sind fest vorgegeben.
<i>udp</i>	Empfang von Positionsdatenpaketen über UDP (Lesen Positionsdatenspeicher des Treibers). Die Defaultwerte können mit der Return-Taste übernommen werden. Abbruch mit Strg-C EIB 8791 muss beispielsweise mit dem Kommando <i>87</i> entsprechend vorher konfiguriert werden
<i>poll</i>	Abfrage von Positionsdaten im Polling Modus. Abbruch mit Strg-C. EIB 8791 muss beispielsweise mit dem Kommando <i>87</i> entsprechend vorher konfiguriert werden
<i>87</i>	Standardkonfiguration an die EIB 8791 senden. Positionspakete werden erzeugt und können über UDP empfangen werden. IP-Adresse und Port des Rechners, an den die UDP Daten gesendet werden sollen, müssen eingegeben werden (i. d. R. der Rechner, auf dem das Tool läuft).

3.3 LabVIEW™ EIB 8791 Application

Die EIB 8791 Application befindet sich auf der Treiber CD und kann, wenn auf dem Rechner LabVIEW™ (Version 2012, Service Pack 1, 32 Bit), installiert ist, direkt gestartet werden (Nabview\32Bit\EIB8Application\EIB8 Application EXE\EIB8Application.exe).

Befindet sich kein LabVIEW™ auf dem Rechner, so kann das Tool mit einer kostenlosen Runtime von LabVIEW™ installiert werden (Nabview\32Bit\EIB8Application\EIB8 Application Setup\Volume\setup.exe).

Zudem befindet sich das Tool auch als VI auf der Treiber CD (Nabview\32Bit\EIB8Driver\Application\VEIB8 Application Project.lvproj).

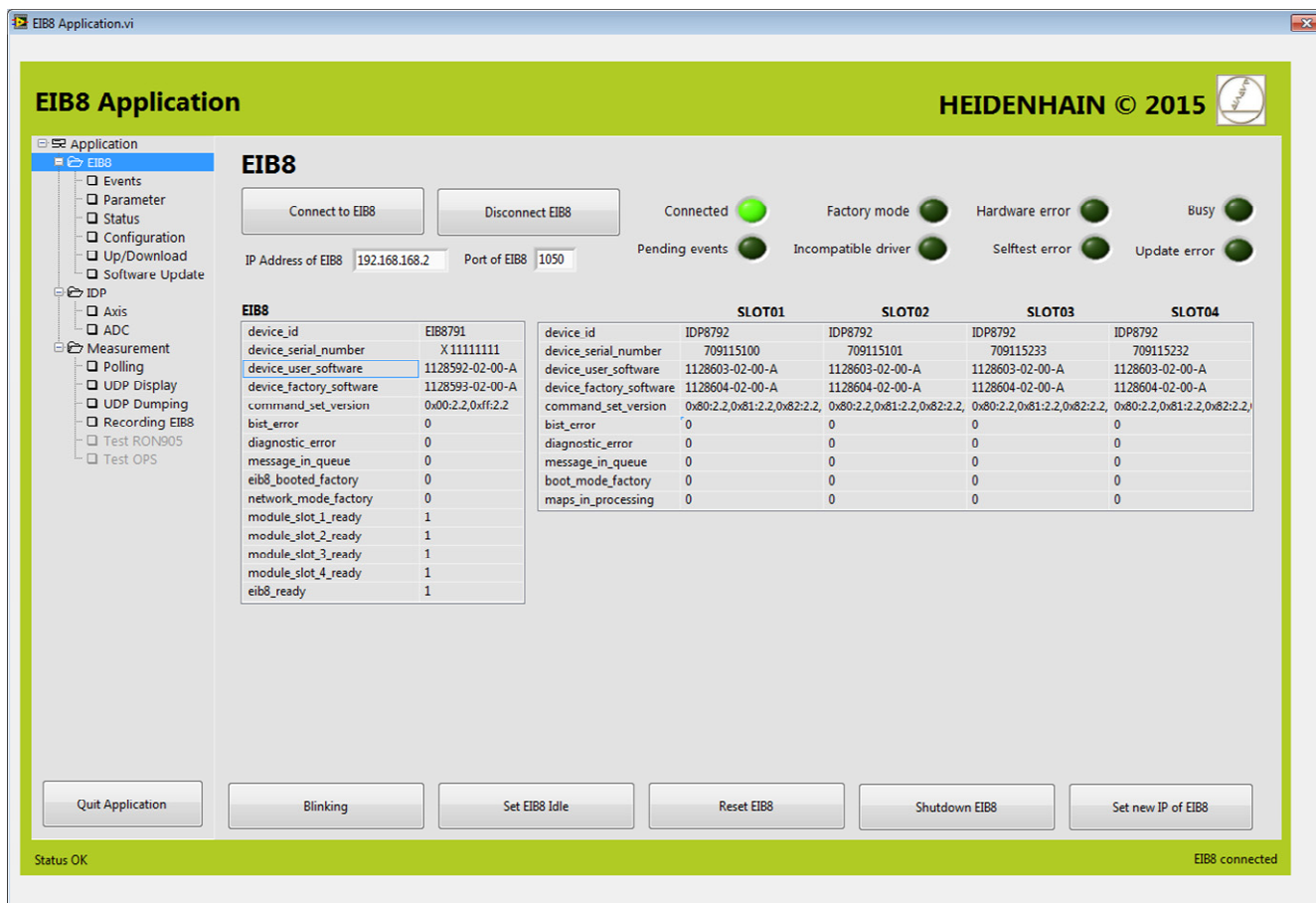
Im Folgenden wird auf Funktionen eingegangen, die für typische Applikationen relevant sind.

3.3.1 Startfenster und verbinden mit der EIB 8791

Die EIB 8791 Application startet mit der Seite *EIB8*. Links im Baum können die verschiedenen Seiten ausgewählt werden. Unten befindet sich eine Statusleiste, die bei einem Fehler einen rot markierten Text ausgibt (i. d. R. ist das der Text, den die Funktion *eib8_get_last_error()* des Treibers ausgibt).



Ist die korrekte IP-Adresse der EIB 8791 in „IP Address of EIB8“ eingetragen, so kann das Gerät mit dem Button „Connect to EIB8“ verbunden werden. Wenn keine Fehler aufgetreten sind, zeigt sich folgendes Fenster:



3.3.2 Seite EIB8

Auf der Seite *EIB8* können nach dem Verbinden mit der EIB 8791 folgende Grundfunktionen ausgeführt werden:

Button	Erläuterung
<i>Connect to EIB8</i>	EIB 8791 verbinden und Zustände aktualisieren. Kann jederzeit wiederholt werden.
<i>Disconnect EIB8</i>	Verbindung zur EIB 8791 trennen (<i>eib8_disconnect_tcp()</i>).
<i>Blinking</i>	Aktivierung/Deaktivierung Blinkmodus der LAN LED (<i>eib8_identify()</i>).
<i>Set EIB8 Idle</i>	Zurücksetzen der Konfiguration der EIB 8791 (<i>eib8_reset_idle()</i>).
<i>Reset EIB8</i>	Neustart der EIB 8791 (<i>eib8_reset()</i>)
<i>Shutdown EIB8</i>	Shutdown der EIB 8791 einleiten (<i>eib8_shutdown()</i>).
<i>Set new IP of EIB8</i>	Konfigurieren der IP-Adresse der EIB 8791. Nach dem Verbinden mit der EIB 8791 wird die gewünschte Adresse in das Feld <i>IP Address of EIB8</i> eingetragen und dann der Button „ <i>Set new IP of EIB8</i> “ betätigt. Die IP-Adresse wird nach einem Neustart des Geräts wirksam (Button <i>Reset EIB8</i>) und dauerhaft im Gerät gespeichert.

3.3.3 Seite Application

Auf der Seite *Application* wird die Version des Treibers angezeigt. Außerdem kann der Logging Level, der von der EIB 8791 Application für das Logging auf UDP verwendet wird, konfiguriert werden (siehe 2.5.4.3).

UDP destination:

Unter *Destination IP*, *Port* und *MAC* wird das Ziel des UDP Transfers für die Standardkonfiguration der EIB 8791 eingestellt (siehe 3.3.7). Die EIB 8791 Application ermittelt beim Programmstart die IP-Adresse des Rechners, sowie die zugehörige MAC-Adresse und trägt diese in die Eingabefelder ein (nicht möglich, wenn die Netzwerkverbindung inaktiv). Je nach Anwendung können die Werte verändert werden, wenn beispielsweise die UDP Pakete an einen anderen Rechner versendet werden sollen.

3.3.4 Seite Events

Button	Erläuterung
<i>Get Events</i>	Auslesen und Anzeigen der Event Queues. Neue Events werden an die Liste angehängt.
<i>Clear Event table</i>	Löscht die Liste der Events im Tool

3.3.5 Seite Parameter

Button	Erläuterung
<i>Get Parameter List from EIB8</i>	Auslesen aller aktuellen Parameter der EIB 8791 mit Erläuterung und Beispiel. Die Parameterliste wird ins Windows temp Verzeichnis unter temp_eib8_params_get.txt gespeichert.

3.3.6 Seite Status

Button	Erläuterung
<i>Get Status List from EIB8</i>	Auslesen aller aktuellen Statusparameter der EIB 8791. Die Statusliste wird ins Windows temp Verzeichnis unter temp_eib8_status_get.txt gespeichert.

3.3.7 Seite Configuration

Button	Erläuterung
<i>Get Config List from EIB8</i>	Auslesen aller aktuellen Konfigurationsparameter der EIB 8791. Anzeige in der „List of current Configuration“. Außerdem wird die Konfiguration ins Windows temp Verzeichnis unter temp_eib8_config_get.txt gespeichert.
<i>Write Config List to EIB8</i>	Senden der „List of current Configuration“ an die EIB 8791.
<i>Get Config List from File</i>	Lesen einer Konfigurationsdatei und in der „List of current Configuration“ anzeigen (zum Senden an die EIB 8791 Button <i>Write Config List to EIB8</i> betätigen).
<i>Write Config List to File</i>	„List of current Configuration“ in eine Datei schreiben.
<i>PTM</i>	Aktivieren / Deaktivieren des internen PTM Generators mit <i>Frequency Hz.</i> (entspricht dem Konfigurationsparameter: <i>EIB8; trig_ptm_in:freq_config</i>) Der Wert ist nur wirksam, wenn die EIB 8791 mit internem PTM betrieben wird.
<i>c1.txt to EIB8</i> <i>c2.txt to EIB8</i> <i>c3.txt to EIB8</i> <i>c4.txt to EIB8</i>	Für eine schnelle Konfiguration wird die Konfigurationsdatei c1.txt (bzw. c2.txt usw.), die sich im Verzeichnis von EIB8Application.exe befinden muss, in der „List of current Configuration“ angezeigt und sofort an die EIB 8791 gesendet.
<i>Set standard Configuration EIB 8791</i>	Schreiben der Standardkonfiguration an die EIB 8791. „List of current Configuration“ bleibt dabei unverändert. Für den UDP Transfer werden dabei die Einstellungen der Seite <i>Application</i> verwendet.
<i>Set EIB8 Idle</i>	Zurücksetzen der Konfiguration der EIB 8791 (<i>eib8_reset_idle()</i>).
<i>Reset PDL frame counter</i>	Rücksetzen des Frame - Zählers aller Achsen (<i>eib8_reset_pdl_frame_counter()</i>).

3.3.8 Seite **Software Update (FTP)**

Zunächst ist über den Dateiauswahldialog (Button Datei rechts) das Softwarepaket (bin Datei) auszuwählen. *Destination* wird auf dem Defaultwert *update* belassen.

Der Button *Start Update* startet den Update Prozess. Dabei werden folgende Schritte ausgeführt (siehe auch 1.11 und 2.10.7):

- Senden des Softwarepakets an die EIB 8791 über FTP (ein Umbenennen der Datei ist hier nicht nötig)
- Warten bis das Update verarbeitet ist.
- Prüfung, ob das Update erfolgreich war.

Sind Fehler aufgetreten, dann sollte zunächst das Update wiederholt werden.

Wenn weiterhin Fehler auftreten, dann sollten die Events ausgelesen werden (siehe 3.3.4). Setzen Sie sich bitte anschließend mit dem Messgeräte-Support bei HEIDENHAIN in Verbindung.

Nach einem erfolgreichen Update, sollte auf der Seite *EIB8* ein Reset ausgelöst werden. Nach erneutem Verbinden der EIB 8791 kann mit dem Parameter *EIB8; device_user_software* überprüft werden, ob die neue Software geladen wurde (der Parameter wird beim Verbinden mit der EIB 8791 automatisch ausgelesen und angezeigt).

3.3.9 Seite **Axis**

Die Kommandos der Seite *Axis* beziehen sich auf immer nur eine Achse. Diese wird in den Auswahlfeldern *Slot and Axis for actions* eingestellt (die EIB 8791 hat zwei Achsen pro Slot). In der Regel muss auch vor Ausführung eines Kommandos eine gültige Konfiguration an die EIB 8791 geladen worden sein (d.h. Konfiguration eines Positionsdatenpaketes, Messgerätkonfiguration und Trigger).

Button	Erläuterung
<i>Referencing / Compensation Check</i>	Aktualisiert den Status <i>Referencing Done</i> , sowie die Konfigurationsparameter <i>Online Compensation</i> und <i>Waveform Compensation</i>
<i>Position</i>	Anzeige von Periodenzähler und Phase (Interpolationswert innerhalb der Signalperiode) der Achse
<i>Referencing Start</i>	Startet den Referenzlauf der Achse. <i>Referencing Done</i> beginnt gelb zu blinken. Wurde die Referenzmarke gefunden, wird <i>Referencing Done</i> grün.
<i>Waveform correction run start</i>	Startet den Korrekturlauf für die Signalformkompensation (siehe 2.10.5).
<i>Set Compensation</i>	Aktivieren der Online- und/oder Signalformkompensation
<i>Clear Position error</i>	Rücksetzen des Fehlerbits in den Positionsdaten (siehe 2.10.4).
<i>Set Position</i>	Setzt den Positionswert der Achse auf einen gewünschten Wert (siehe 2.10.3).

3.3.10 Seite **ADC**

Auf der Seite *ADC* können die Analogwerte der beiden Inkrementalsignale A und B für alle acht Achsen angezeigt werden.

Voraussetzung ist eine gültige Konfiguration mit UDP Ausgabe in der EIB 8791, wie sie beispielsweise mit *Set standard Configuration EIB 8791* auf der Seite *Configuration* erzeugt werden kann.

Der Button *ADC* verändert automatisch die Konfiguration zur Ausgabe von ADC Positionsdatenpaketen und startet den UDP Empfang am Treiber. Zu beachten ist, dass nach Beendigung der ADC Ausgabe die Konfiguration nicht auf die ursprünglichen Werte zurückgestellt wird.

Die weiteren Einstellungen für die ADC Darstellung mit Ausnahme der Skalierung der Werte (*Scaling*) sollten nicht verändert werden.

3.3.11 Seite **Polling**

Auf der Seite *Polling* werden alle konfigurierten Positionsdatenpakete ausgelesen und angezeigt (siehe 2.8.5). Die linken drei Spalten gelten für alle Positionsdatentypen, alle weiteren nur für Pakete vom Datentyp „position“, „speed“ und „reference“.

Voraussetzung ist eine gültige Konfiguration der EIB 8791, wie sie beispielsweise mit *Set standard Configuration EIB 8791* auf der Seite *Configuration* erzeugt werden kann.

3.3.12 Seite **UDP Display**

Auf der Seite *UDP Display* werden alle Positionsdatenpakete angezeigt, die auf dem gewählten Port (*UDP receiving Port*) vom Treiber empfangen werden. Die linken drei Spalten gelten für alle Positionsdattentypen, alle weiteren nur für Pakete vom Datentyp „position“, „speed“ und „reference“. *Overrun occured* signalisiert, dass der Treiber nicht alle Pakete verarbeiten konnte.

Hinweis:

Für die Anzeige wird die Treiberfunktion *eib8_get_udp_pdl_latest()* verwendet, d.h. bei jeder Abfrage des Positionsdatenpuffers wird der aktuellste Wert eines jeden Positionsdattentyps angezeigt.

3.3.13 Seite **UDP Dumping**

Alle Positionsdatenpakete, die auf dem gewählten Port (*UDP receiving Port*) vom Treiber empfangen werden, werden in eine Datei (*File for dumping data*) geschrieben.

Dabei können die Positionsdaten als String (*Dumping format : EIB8_PDL_DUMP_STD_TXT_CHECKS*) oder binär (*Dumping format : EIB8_PDL_DUMP_RAW_BINARY*) abgelegt werden (siehe 2.7.2). Binär bedeutet, dass das Positionsdatenpaket mit 12 Byte abgespeichert wird.

Folgende Zustände werden dabei signalisiert:

- *Total packets:*
Zahl der Pakete, die bereits in die Datei geschrieben wurden.
- *Load (%):*
Rechenleistung, die der Treiber benötigt.
- *Internal queue empty:*
Der Treiber konnte sofort den gesamten Positionsdatenspeicher leeren bzw. es sind keine Positionsdatenpakete empfangen worden.
- *Overrun occured:*
Ein Überlauf des Positionsdatenspeichers ist aufgetreten. Pakete können nicht schnell genug verarbeitet werden (Datenrate der EIB 8791 zu hoch).
- *PD frame counter with gaps:*
Es sind Lücken im Frame-Zähler detektiert worden, d.h. Pakete sind verlorengegangen (in der Regel ist die Datenrate der EIB 8791 zu hoch).
- *CRC errors in PD packets:*
In den Paketen befinden sich CRC Fehler.
- *Aborted by driver:*
Dumping wurde aufgrund eines Fehlers beendet.

3.3.14 Seite **Recording EIB8**

Die Seite *Recording EIB8* unterstützt den Anwender beim Auslesen von Positionsdaten, die im RAM der EIB 8791 aufgezeichnet worden sind. Die Konfiguration der EIB 8791 und das Starten der Aufzeichnung der Positionsdaten ist in 2.10.1 beschrieben.

Button	Erläuterung
<i>Recording state</i>	Überprüft fortlaufend den Stand der Aufzeichnung.
<i>Transfer Recorded PD from EIB8</i>	Abholen der Positionsdatenpakete von der EIB 8791 und Abspeichern in eine Datei (<i>File</i>). Voraussetzung ist, dass die Aufzeichnung beendet ist. Durch eine entsprechende Konfiguration des Timeouts, kann die Zeit für das Abholen der Pakete begrenzt werden. Sollen alle Pakete abgeholt werden, so ist der Timeout entsprechend hoch einzustellen. Für 1 Mio Pakete werden je nach Rechner ca. 60 Sekunden benötigt.

3.4 LabVIEW™ Beispiele

Auf der Treiber CD befinden sich VIs, die die Treiberfunktionen aufrufen (labview\32Bit\EIB8Driver\EIB8ApiVI). Die VIs haben nahezu die gleichen Namen wie die Funktionen des Treibers. Eine Beschreibung der Funktionen und Datentypen finden sich in der Schnittstellendefinition („eib8.h“) oder der mitgelieferten HTML-Dokumentation.

Die LabVIEW™ Beispiele auf der Treiber CD zeigen die Anwendung der genannten VIs. Die Beispiele sind in einem Projekt zusammengefasst (labview\32Bit\EIB8Driver\ExampleVI\Eib8Examples.lvproj).

Beispiel	Erläuterung
<i>ConnectEvents.vi</i>	<ul style="list-style-type: none"> • Auswertung der Statusinformationen beim Connect • Auslesen und Anzeigen der Events
<i>DefaultConfig.vi</i>	<ul style="list-style-type: none"> • Standardkonfiguration EIB 8791: IP-Adresse des Hosts eingeben und mit <i>DISCOVER MAC HOST</i> die MAC-Adresse ermitteln. <i>Button Default EIB 8791</i> sendet die Konfiguration an die EIB 8791 • Reset • Reset Idle • Shutdown
<i>Identify.vi</i>	<ul style="list-style-type: none"> • Aktivierung und Deaktivierung des Blinkmodus der LAN LED der EIB 8791
<i>MultiHeadSystem.vi</i> (mit Konfigurationsdatei <i>config_multi_head_system.txt</i>)	<ul style="list-style-type: none"> • Schreiben Konfigurationsfile (<i>config_multi_head_system.txt</i>) • Referenzierung • Korrekturwertaufnahme und Signalformkompensation • Anzeige von Position und Geschwindigkeit als Mittelwert von bis zu 8 Achsen
<i>Configuration.vi</i>	<ul style="list-style-type: none"> • Lesen und Schreiben von Konfigurationsfiles • Lesen der gesamten Parameterliste aus der EIB 8791
<i>SetIPAdress</i>	<ul style="list-style-type: none"> • Konfigurieren der IP-Adresse der EIB 8791
<i>PollPD.vi</i>	<ul style="list-style-type: none"> • Auslesen von Positionsdaten im Modus Polling
<i>UDPReceive.vi</i>	<ul style="list-style-type: none"> • Auslesen von Positionsdaten aus dem Positionsdatenspeicher des Treibers
<i>UDPSynchReceive.vi</i>	<ul style="list-style-type: none"> • Auslesen von Positionsdaten aus dem Positionsdatenspeicher des Treibers, wobei die Positionsdaten synchron erwartet werden (siehe 2.8.6.3)

HEIDENHAIN

DR. JOHANNES HEIDENHAIN GmbH

Dr.-Johannes-Heidenhain-Straße 5

83301 Traunreut, Germany

☎ +49 8669 31-0

☎ +49 8669 32-5061

E-mail: info@heidenhain.de

Technical support ☎ +49 8669 32-1000

Measuring systems ☎ +49 8669 31-3104

E-mail: service.ms-support@heidenhain.de

NC support ☎ +49 8669 31-3101

E-mail: service.nc-support@heidenhain.de

NC programming ☎ +49 8669 31-3103

E-mail: service.nc-pgm@heidenhain.de

PLC programming ☎ +49 8669 31-3102

E-mail: service.plc@heidenhain.de

APP programming ☎ +49 8669 31-3106

E-mail: service.app@heidenhain.de

www.heidenhain.de