X90DSI00.04-00

1 Bestelldaten

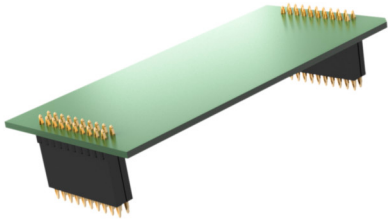
Bestellnummer	Kurzbeschreibung	Abbildung
	Kommunikationsmodule	
X90DSI00.04-00	X90 Optionsplatine IO-Link, Digitales Signalmodul, 4x IO-Link Master, 4 digitale Kanäle wahlweise als Ein- oder Ausgang parametrierbar, 1-Leitertechnik	

Tabelle 1: X90DSI00.04-00 - Bestelldaten

2 Modulbeschreibung

Das Modul ist ein IO-Link Master, mit dem intelligente Sensoren und Aktoren nach dem IO-Link Standard an das X20 System angeschlossen werden können. Das Modul kann bis zu 4 I/O-Link Devices bedienen. Alle IO-Link Kanäle können wahlweise auch als digitale Standard Ein- oder Ausgänge genutzt werden.

Das modulare Steuerungs- und I/O-System X90 mobile eröffnet viele Möglichkeiten in der mobilen Automatisierung. Mit X90 mobile lassen sich flexible Automatisierungskonzepte auf Basis eines standardisierten Gesamtsystems umsetzen.

Die Optionsplatine X90DSI00.04-00 wird im X90 mobile System integriert und erweitert somit die Funktionalität des Gesamtsystems.

Die DS-Optionsplatine ist ein IO-Link Master, mit dem intelligente Sensoren und Aktoren nach dem IO-Link Standard angeschlossen werden können. Die Kommunikation zur Hauptplatine wird via X2X Link ermöglicht.

Funktionen:

- Flatstream-Kommunikation
- NetTime Technology

Flatstream-Kommunikation

Der "Flatstream" wurde für X2X und POWERLINK-Netzwerke konzipiert und ermöglicht einen individuell angepassten Datentransfer. Damit kann die Übertragung der Daten effizienter gestaltet werden als bei der zyklischen Standardabfrage.

NetTime-Zeitstempel IO-Link

Mit Hilfe dieser Zeitstempel können Applikationen am IO-Link Werteänderungen erfassen und Ereignisse auslösen, welche zeitlich höher aufgelöst sind als der IO-Zyklus erlauben würde.

3 Technische Beschreibung

3.1 Technische Daten

Bestellnummer	X90DSI00.04-00
Kurzbeschreibung	
I/O-Modul	IO-Link Master mit 4 IO-Link Schnittstellen
Allgemeines	
B&R ID-Code	0xF9F7
Statusanzeigen	-
Diagnose	
Modul Run/Error	Ja, per Software-Status
Leistungsaufnahme	
I/O-intern	0,6 W
Zusätzliche Verlustleistung durch Aktoren (ohmsch) [W]	-
Zulassungen	
CE	Ja
Sensor-/Aktorversorgung	
Spannung	I/O-Versorgung abzüglich Spannungsabfall am Kurzschlussschutz
Spannungsabfall am Kurzschlussschutz bei 0,3 A	max. 0,3 V
Leistungsaufnahme	max. 12 W pro IO-Link Schnittstelle
kurzschlussfest	Ja
Überlastschutz	
Ausschaltverzögerung	Softwaremäßig einstellbar
Ausschaltdauer	Softwaremäßig einstellbar
IO-Link im Master-Modus	
Übertragungsraten	
COM1	4,8 kBaud
COM2	38,4 kBaud
COM3	230,4 kBaud
Grenzwerte für COM3	
max. Anschlusskapazität	22 nF (Kabel + IO-Link Device)
max. Last	96 Ω / 250 mA
Datenformat	1 Startbit, 8 Datenbits, 1 Paritybit (even), 1 Stoppbit
Buspegel	24 VDC (aktiv), 0 VDC (Ruhepegel)
IO-Link im Master-Modus oder im SIO-Modus "digitaler Ausgang"	
Ausführung	Bipolar, plus- und minus-schaltend
Kurzschlussspitzenstrom	<1,3 A
Restspannung	<0,7 VDC bei Nennstrom 0,25 A
Schaltspannung	I/O-Versorgung abzüglich Spannungsabfall am Kurzschlussschutz und Halbleiterschalter
Spannungsabfall am Halbleiterschalter	max. 0,44 VDC bei 0,2 A
Schaltfrequenz	typ. 25 kHz 300 kHz im IO-Link Mastermodus
Schaltverzögerung	
0 → 1	<1 µs
1 → 0	<1 µs
Einschaltung bei Überlastabschaltung bzw. Kurzschlussabschaltung	Einstellbar durch Software
IO-Link im SIO-Modus "digitaler Ausgang"	
Nennspannung	24 VDC
Ausgangsnennstrom	0,25 A
Summennennstrom	max. 1 A
Ausgangsbeschaltung	Sink oder Source
Schaltfrequenz (ohmsche Last)	max. 500 Hz
Ausgangsschutz [1)]	Thermische Abschaltung bei Überstrom oder Kurzschluss, integrierter Schutz zum Schalten von Induktivitäten
IO-Link im SIO-Modus "digitaler Eingang"	
Nennspannung	24 VDC
Eingangscharakteristik nach EN 61131-2	Typ 1
Eingangsfilter	
Hardware	220 ns
Software	3 µs (ohne Glitchfilter) 4,2 µs (mit Glitchfilter)
Eingangsbeschaltung	Sink
Eingangsspannung	24 VDC -15% / +20%
Eingangsstrom bei 24 VDC	typ. 5,8 mA
Eingangswiderstand	typ. 4,14 kΩ
Schaltschwellen	
Low	<5 VDC
High	>15 VDC

Tabelle 2: X90DSI00.04-00 - Technische Daten

Bestellnummer	X90DSI00.04-00
Elektrische Eigenschaften	
Potenzialtrennung	Bus zu IO-Link getrennt IO-Link zu IO-Link nicht getrennt
Einsatzbedingungen	
Einbaulage	
beliebig	Ja
Schutzart nach EN 60529	bis zu IP69K [2]
Umgebungsbedingungen	
Temperatur	
Lagerung	-40 bis 85°C
Transport	-40 bis 85°C
Luftfeuchtigkeit	
Betrieb	5 bis 100%, kondensierend
Lagerung	5 bis 100%, kondensierend
Transport	5 bis 100%, kondensierend
Mechanische Eigenschaften	
Abmessungen	
Breite	47 mm
Länge	95 mm

Tabelle 2: X90DSI00.04-00 - Technische Daten

1) Abschaltstrom bei Überlast: Zwischen 0,3 A und 0,8 A.
2) In Abhängigkeit der Hauptplatine. Für weitere Details siehe Datenblatt Hauptplatine.

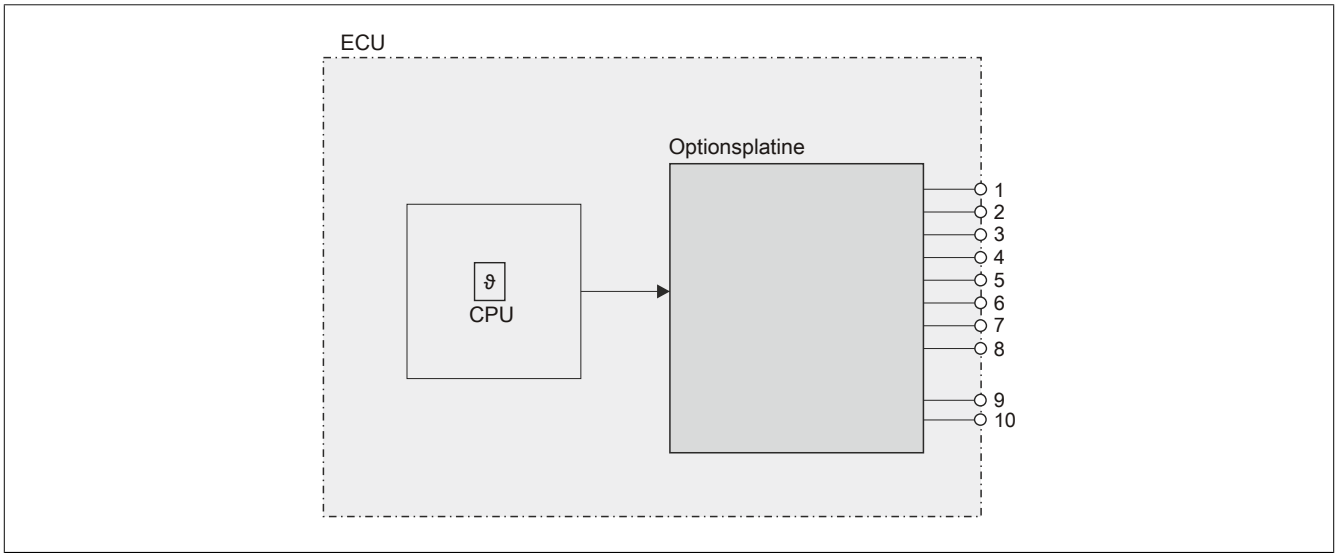
3.2 X2X Link Schnittstelle

Die Kommunikation der Optionsplatine mit der Hauptplatine wird mittels X2X Link realisiert.

3.3 Pinbelegung

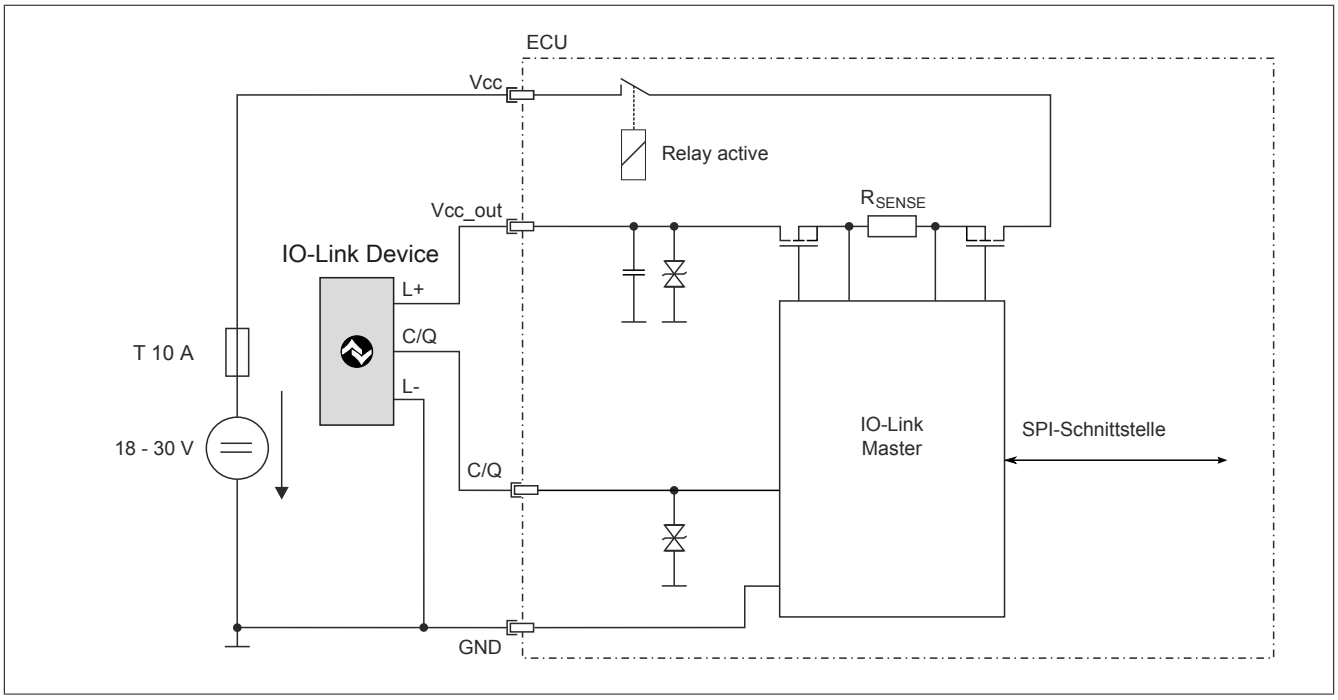
Kanal	Pinbelegung
1	C/Q 1 - DI/DO 1
2	Vcc out (IO-Link)
3	C/Q 2 - DI/DO 2
4	Vcc out (IO-Link)
5	C/Q 3 - DI/DO 3
6	Vcc out (IO-Link)
7	C/Q 4 - DI/DO 4
8	Vcc out (IO-Link)
9	Vcc in (IO-Link)
10	Vcc in (IO-Link)

3.4 Blockschaltbild

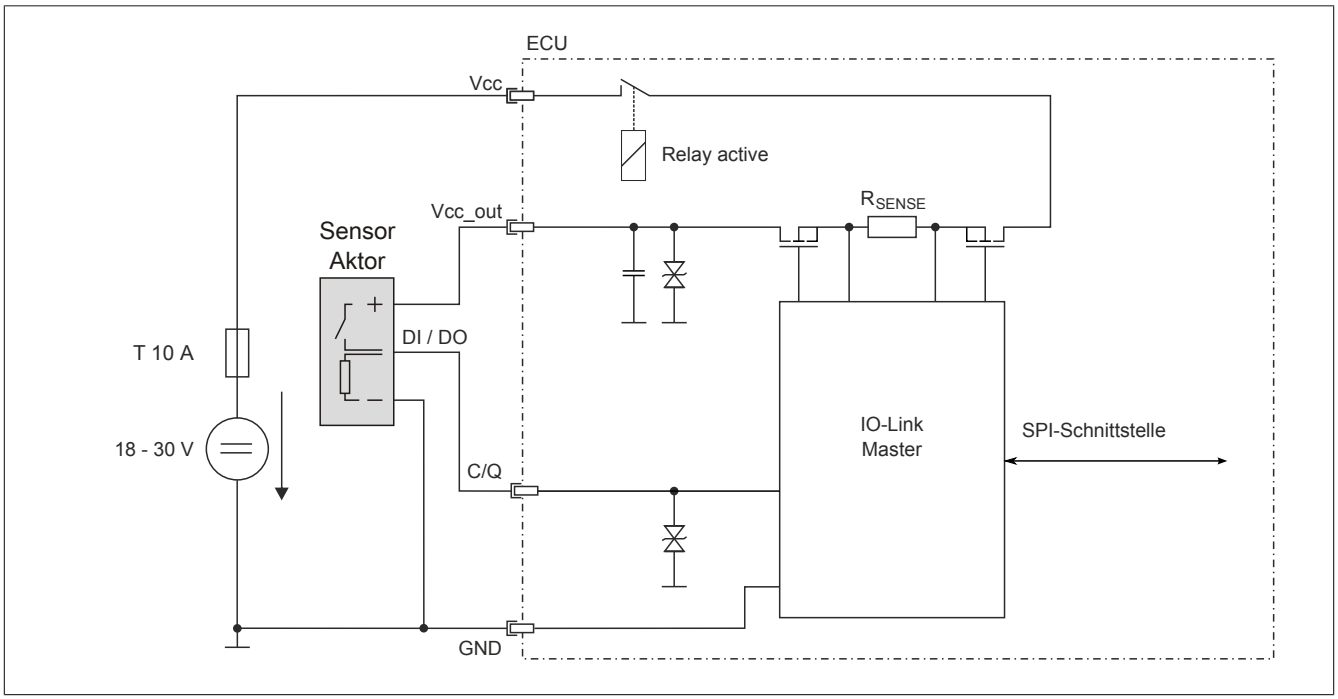


3.4.1 Ein-/Ausgangschema

Verwendung als IO-Link



Verwendung als digitaler Ein- oder Ausgang



4 Funktionsbeschreibung

4.1 Die Flatstream-Kommunikation

4.1.1 Einleitung

Für einige Module stellt B&R ein zusätzliches Kommunikationsverfahren bereit. Der "Flatstream" wurde für X2X und POWERLINK-Netzwerke konzipiert und ermöglicht einen individuell angepassten Datentransfer. Obwohl das Verfahren nicht unmittelbar echtzeitfähig ist, kann die Übertragung effizienter gestaltet werden als bei der zyklischen Standardabfrage.

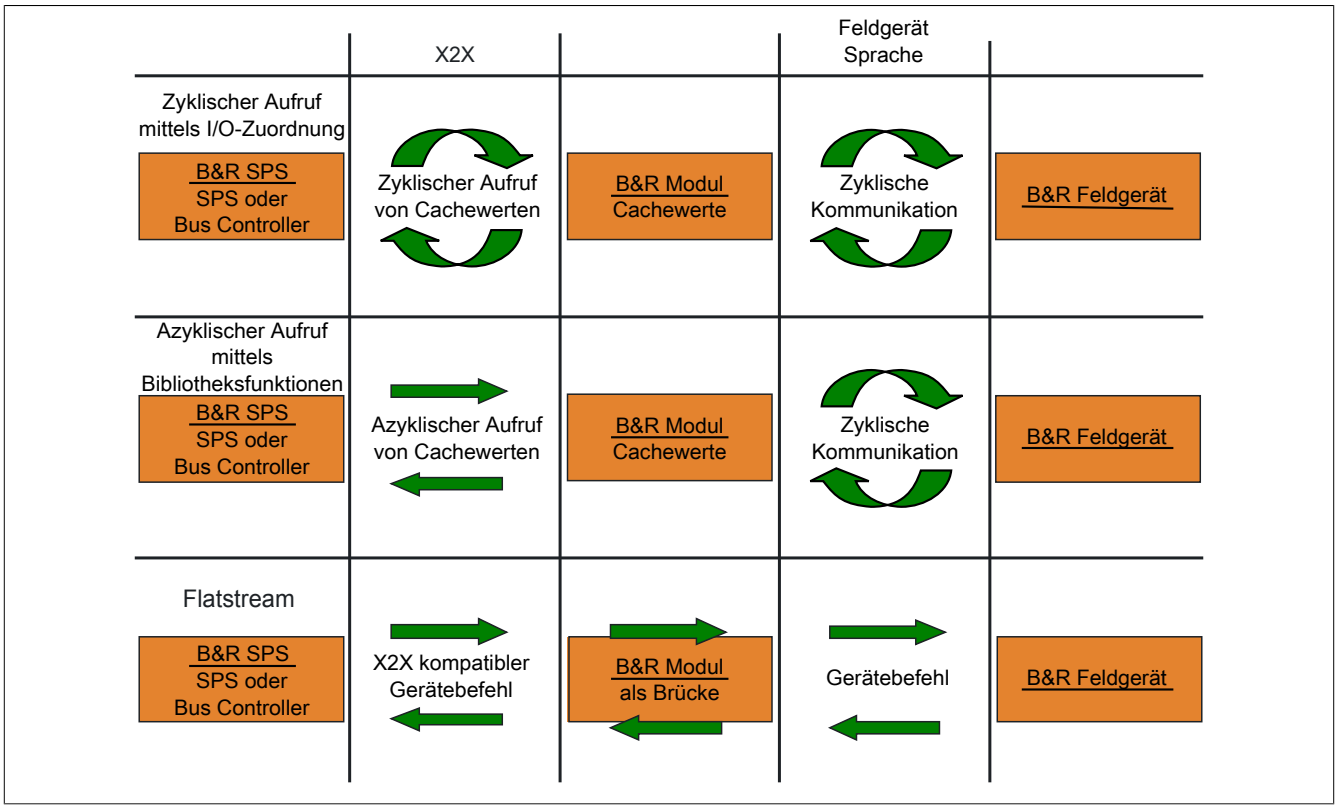


Abbildung 1: 3 Arten der Kommunikation

Durch den Flatstream wird die zyklische bzw. azyklische Abfrage ergänzt. Bei der Flatstream-Kommunikation fungiert das Modul als Bridge. Die Anfragen der Steuerung werden über das Modul direkt zum Feldgerät geleitet.

4.1.2 Nachricht, Segment, Sequenz, MTU

Die physikalischen Eigenschaften des Bussystems begrenzen die Datenmenge, die während eines Buszyklus übermittelt werden kann. Bei der Flatstream-Kommunikation werden alle Nachrichten als fortlaufender Datenstrom (engl. stream) betrachtet. Lange Datenströme müssen in mehrere Teile zerlegt und nacheinander versendet werden. Um zu verstehen, wie der Empfänger die ursprüngliche Information wieder zusammensetzt, werden die Begriffe Nachricht, Segment, Sequenz und MTU unterschieden.

Nachricht

Eine Nachricht ist eine Mitteilung, die zwischen 2 Kommunikationspartnern ausgetauscht werden soll. Die Länge einer solchen Mitteilung wird durch das Flatstream-Verfahren nicht begrenzt. Es müssen allerdings modulspezifische Beschränkungen beachtet werden.

Segment (logische Gliederung einer Nachricht)

Ein Segment ist endlich groß und kann als Abschnitt der Nachricht verstanden werden. Die Anzahl der Segmente pro Nachricht ist beliebig. Damit der Empfänger die übertragenen Segmente wieder korrekt zusammensetzen kann, geht jedem Segment ein Byte mit Zusatzinformationen voraus. Das sogenannte Controlbyte enthält z. B. Informationen über die Länge eines Segments und ob das kommende Segment die Mitteilung vervollständigt. Auf diesem Weg wird der Empfänger in die Lage versetzt, den ankommenden Datenstrom korrekt zu interpretieren.

Sequenz (physikalisch notwendige Gliederung eines Segments)

Die maximale Größe einer Sequenz entspricht der Anzahl der aktivierten Rx- bzw. Tx-Bytes (später: "MTU"). Die sendende Station teilt das Sendearray in zulässige Sequenzen, die nacheinander in die MTU geschrieben, zum Empfänger übertragen und dort wieder aneinandergereiht werden. Der Empfänger legt die ankommenden Sequenzen in einem Empfangsarray ab und erhält somit ein Abbild des Datenstroms.
Bei der Flatstream-Kommunikation werden die abgesetzten Sequenzen gezählt. Erfolgreich übertragene Sequenzen müssen vom Empfänger bestätigt werden, um die Übertragung abzusichern.

MTU (Maximum Transmission Unit) - Physikalischer Transport

Die MTU des Flatstreams beschreibt die aktivierten USINT-Register für den Flatstream. Die Register können mindestens eine Sequenz aufnehmen und zum Empfänger übertragen. Für beide Kommunikationsrichtungen wird eine separate MTU vereinbart. Die OutputMTU definiert die Anzahl der Flatstream-Tx-Bytes und die InputMTU beschreibt die Anzahl der Flatstream-Rx-Bytes. Die MTUs werden zyklisch über den X2X Link transportiert, sodass die Auslastung mit jedem zusätzlich aktivierten USINT-Register steigt.

Eigenschaften

Flatstream-Nachrichten werden nicht zyklisch und nicht unmittelbar in Echtzeit übertragen. Zur Übertragung einer bestimmten Mitteilung werden individuell viele Buszyklen benötigt. Die Rx-/Tx-Register werden zwar zyklisch zwischen Sender und Empfänger ausgetauscht, aber erst weiterverarbeitet, wenn die Übernahme durch die Register "InputSequence" bzw. "OutputSequence" explizit angewiesen wird.

Verhalten im Fehlerfall (Kurzfassung)

Das Protokoll von X2X bzw. POWERLINK Netzwerken sieht vor, dass bei einer Störung die letzten gültigen Werte erhalten bleiben. Bei der herkömmlichen Kommunikation (zyklische/azyklische Abfrage) kann ein solcher Fehler in der Regel ignoriert werden.
Damit auch via Flatstream problemlos kommuniziert werden kann, müssen alle abgesetzten Sequenzen vom Empfänger bestätigt werden. Ohne die Nutzung des Forwards verzögert sich die weitere Kommunikation um die Dauer der Störung.
Falls der Forward genutzt wird, erhält die Empfängerstation einen doppelt inkrementierten Sendezähler. Der Empfänger stoppt, das heißt, er schickt keine Bestätigungen mehr zurück. Anhand des SequenceAck erkennt die Sendestation, dass die Übertragung fehlerhaft war und alle betroffenen Sequenzen wiederholt werden müssen.

4.1.3 Prinzip des Flatstreams

Voraussetzung

Bevor der Flatstream genutzt werden kann, muss die jeweilige Kommunikationsrichtung synchronisiert sein, das heißt, beide Kommunikationspartner fragen zyklisch den SequenceCounter der Gegenstelle ab. Damit prüfen sie, ob neue Daten vorliegen, die übernommen werden müssen.

Kommunikation

Wenn ein Kommunikationspartner eine Nachricht an seine Gegenstelle senden will, sollte er zunächst ein Sendearray anlegen, das den Konventionen des Flatstreams entspricht. Auf diese Weise kann der Flatstream sehr effizient gestaltet werden, ohne wichtige Ressourcen zu blockieren.

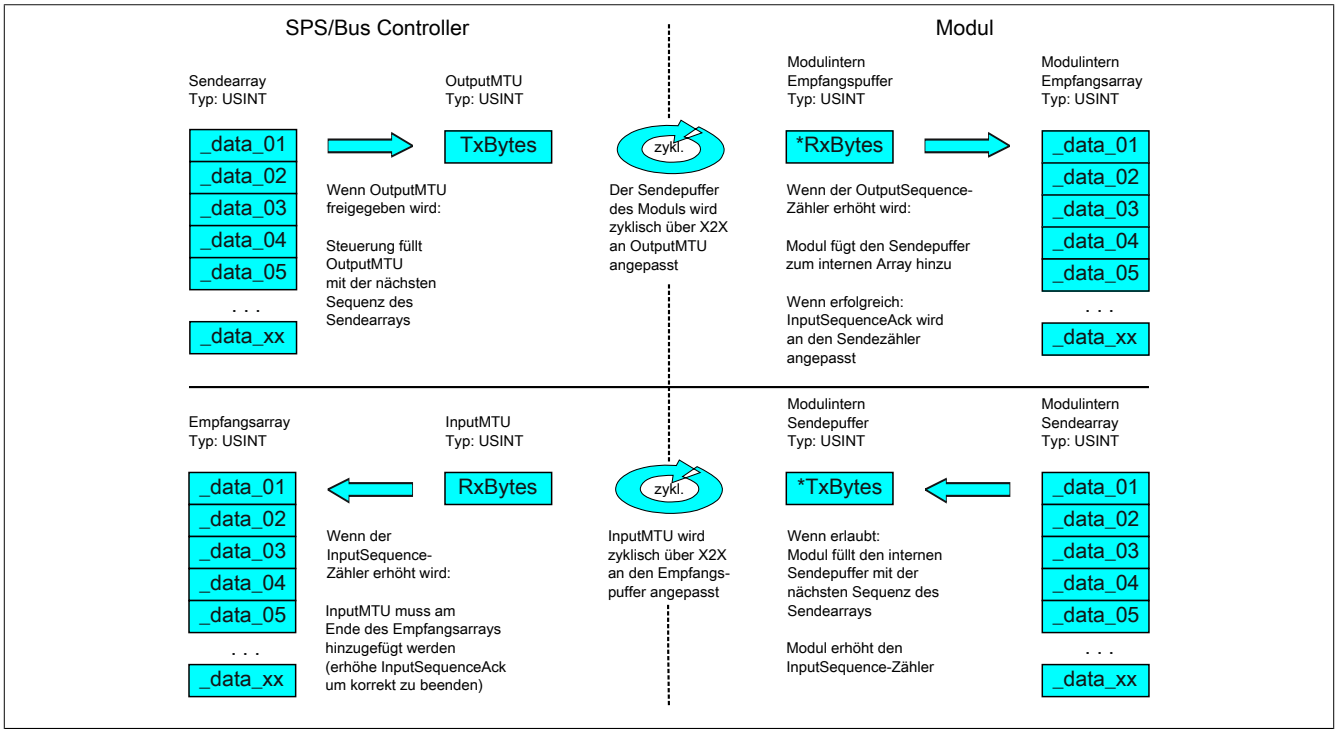


Abbildung 2: Kommunikation per Flatstream

Vorgehensweise

Als erstes wird die Nachricht in zulässige Segmente mit max. 63 Bytes aufgeteilt und die entsprechenden Controlbytes gebildet. Die Daten werden zu einem Datenstrom zusammengefügt, das heißt, je ein Controlbyte und das dazugehörige Segment im Wechsel. Dieser Datenstrom kann in das Sendearray geschrieben werden. Jedes Arrayelement ist dabei max. so groß, wie die freigegebene MTU, sodass ein Element einer Sequenz entspricht.
Wenn das Array vollständig angelegt ist, prüft der Sender, ob die MTU neu befüllt werden darf. Danach kopiert er das erste Element des Arrays bzw. die erste Sequenz auf die Tx-Byte-Register. Die MTU wird zyklisch über den X2X Link zur Empfängerstation transportiert und auf den korrespondierenden Rx-Byte-Registern abgelegt. Als Signal, dass die Daten vom Empfänger übernommen werden sollen, erhöht der Sender seinen SequenceCounter. Wenn die Kommunikationsrichtung synchronisiert ist, erkennt die Gegenstelle den inkrementierten Sequence-Counter. Die aktuelle Sequenz wird an das Empfangsarray angefügt und per SequenceAck bestätigt. Mit dieser Bestätigung wird dem Sender signalisiert, dass die MTU wieder neu befüllt werden kann.

Bei erfolgreicher Übertragung entsprechen die Daten im Empfangsarray exakt denen im Sendearray. Während der Übertragung muss die Empfangsstation die ankommenden Controlbytes erkennen und auswerten. Für jede Nachricht sollte ein separates Empfangsarray angelegt werden. Auf diese Weise kann der Empfänger vollständig übertragene Nachrichten sofort weiterverarbeiten.

4.1.4 Die Register für den Flatstream-Modus

Zur Konfiguration des Flatstreams sind 5 Register vorgesehen. Mit der Standardkonfiguration können geringe Datenmengen relativ einfach übermittelt werden.

Information:

Die Steuerung kommuniziert über die Register "OutputSequence" und "InputSequence" sowie den aktivierten Tx- bzw. RxBytes direkt mit dem Feldgerät. Deshalb benötigt der Anwender ausreichend Kenntnisse über das Kommunikationsprotokoll des Feldgerätes.

4.1.4.1 Konfiguration des Flatstreams

Um den Flatstream zu nutzen, muss der Programmablauf erweitert werden. Die Zykluszeit der Flatstream-Routinen muss auf ein Vielfaches des Buszyklus festgelegt werden. Die zusätzlichen Programmroutinen sollten in Cyclic #1 implementiert werden, um die Datenkonsistenz zu gewährleisten.

Bei der Minimalkonfiguration müssen die Register "InputMTU" und "OutputMTU" eingestellt werden. Alle anderen Register werden beim Start mit Standardwerten belegt und können sofort genutzt werden. Sie stellen zusätzliche Optionen bereit, um Daten kompakter zu übertragen bzw. den allgemeinen Ablauf hoch effizient zu gestalten.

Mit den Forward-Registern wird der Ablauf des Flatstream-Protokolls erweitert. Diese Funktion eignet sich, um die Datenrate des Flatstreams stark zu erhöhen, bedeutet aber erheblichen Mehraufwand bei der Erstellung des Programmablaufs.

Information:

In der weiteren Beschreibung stehen die Bezeichnungen "OutputMTU" und "InputMTU" nicht für die Registernamen, sondern als Synonym für die momentan aktivierten Tx- bzw. Rx-Bytes.

Information:

Die Register sind in Abschnitt "Flatstream-Register" auf Seite 73 beschrieben.

4.1.4.2 Bedienung des Flatstreams

Bei der Verwendung des Flatstreams ist die Kommunikationsrichtung von großer Bedeutung. Für das Senden von Daten an ein Modul (Output-Richtung) werden die Tx-Bytes genutzt. Für den Empfang von Daten eines Moduls (Input-Richtung) sind die Rx-Bytes vorgesehen.
Mit den Registern "OutputSequence" und "InputSequence" wird die Kommunikation gesteuert bzw. abgesichert, das heißt, der Sender gibt damit die Anweisung, Daten zu übernehmen und der Empfänger bestätigt eine erfolgreich übertragene Sequenz.

> **Information:**
>
> **Die Register sind in Abschnitt "Flatstream-Register" auf Seite 73 beschrieben.**

4.1.4.2.1 Format der Ein- und Ausgangsbytes

Name:

"Format des Flatstream" im Automation Studio

Bei einigen Modulen kann mit Hilfe dieser Funktion eingestellt werden, wie die Ein- und Ausgangsbytes des Flatstream (Tx- bzw. Rx-Bytes) übergeben werden.

- **gepackt:** Daten werden als ein Array übergeben
- **byteweise:** Daten werden als einzelne Bytes übergeben

4.1.4.2.2 Transport der Nutzdaten und der Controlbytes

Die Tx- bzw. Rx-Bytes sind zyklische Register, die zum Transport der Nutzdaten und der notwendigen Controlbytes dienen. Die Anzahl aktiver Tx- bzw. Rx-Bytes ergibt sich aus der Konfiguration der Register "OutputMTU" bzw. "InputMTU".
Im Programmablauf des Anwenders können nur die Tx- bzw. Rx-Bytes der Steuerung genutzt werden. Innerhalb des Moduls gibt es die entsprechenden Gegenstücke, welche für den Anwender nicht zugänglich sind. Aus diesem Grund wurden die Bezeichnungen aus Sicht der Steuerung gewählt.

- "T" - "transmit" → Steuerung *sendet* Daten an das Modul
- "R" - "receive" → Steuerung *empfängt* Daten vom Modul

Controlbytes

Neben den Nutzdaten übertragen die Tx- bzw. Rx-Bytes auch die sogenannten Controlbytes. Sie enthalten zusätzliche Informationen über den Datenstrom, damit der Empfänger die übertragenen Segmente wieder korrekt zur ursprünglichen Nachricht zusammensetzen kann.

Bitstruktur eines Controlbytes

Bit	Bezeichnung	Wert	Information
0 - 5	SegmentLength	0 - 63	Bytegröße des folgenden Segments (Standard: max. MTU-Größe - 1)
6	nextCBPos	0	Nächstes Controlbyte zu Beginn der nächsten MTU
		1	Nächstes Controlbyte direkt nach Ende des Segments
7	MessageEndBit	0	Nachricht wird nach dem folgenden Segment fortgesetzt
		1	Nachricht wird durch das folgende Segment beendet

SegmentLength

Die Segmentlänge kündigt dem Empfänger an, wie lang das kommende Segment ist. Wenn die eingestellte Segmentlänge für eine Nachricht nicht ausreicht, muss die Mitteilung auf mehrere Segmente verteilt werden. In diesen Fällen kann das tatsächliche Ende der Nachricht über Bit 7 (Controlbyte) erkannt werden.

> **Information:**
>
> **Bei der Bestimmung der Segmentlänge wird das Controlbyte nicht mitgerechnet. Die Segmentlänge ergibt sich rein aus den Bytes der Nutzdaten.**

nextCBPos

Mit diesem Bit wird angezeigt, an welcher Position das nächste Controlbyte zu erwarten ist. Diese Information ist vor allem bei Anwendung der Option "MultiSegmentMTU" wichtig.
Bei der Flatstream-Kommunikation mit MultiSegmentMTUs ist das nächste Controlbyte nicht mehr auf dem ersten Rx-Byte der darauffolgenden MTU zu erwarten, sondern wird direkt im Anschluss an das Segment übertragen.

MessageEndBit

Das "MessageEndBit" wird gesetzt, wenn das folgende Segment eine Nachricht abschließt. Die Mitteilung ist vollständig übertragen und kann weiterverarbeitet werden.

Information:

In Output-Richtung muss dieses Bit auch dann gesetzt werden, wenn ein einzelnes Segment ausreicht, um die vollständige Nachricht aufzunehmen. Das Modul verarbeitet eine Mitteilung intern nur, wenn diese Kennzeichnung vorgenommen wurde.
Die Größe einer übertragenen Mitteilung lässt sich berechnen, wenn alle Segmentlängen der Nachricht addiert werden.

Flatstream-Formel zur Berechnung der Nachrichtenlänge:

Nachricht [Byte] = Segmentlängen (aller CBs ohne ME) + Segmentlänge (des ersten CB mit ME)	CB	Controlbyte
	ME	MessageEndBit

4.1.4.2.3 Kommunikationsstatus

Der Kommunikationsstatus wird über die Register "OutputSequence" und "InputSequence" ermittelt.

- OutputSequence enthält Informationen über den Kommunikationsstatus der Steuerung. Es wird von der Steuerung geschrieben und vom Modul gelesen.
- InputSequence enthält Informationen über den Kommunikationsstatus des Moduls. Es wird vom Modul geschrieben und sollte von der Steuerung nur gelesen werden.

Beziehung zwischen Output- und InputSequence

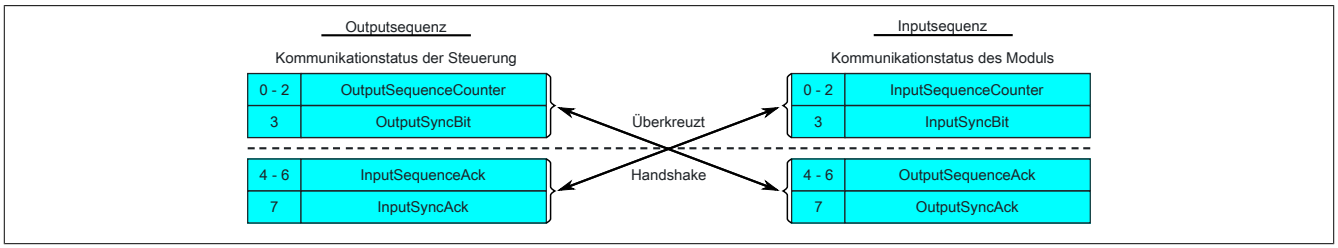


Abbildung 3: Zusammenhang zwischen Output- und InputSequence

Die Register "OutputSequence" und "InputSequence" sind logisch aus 2 Halb-Bytes aufgebaut. Über den Low-Teil wird der Gegenstelle signalisiert, ob ein Kanal geöffnet werden soll bzw. ob Daten übernommen werden können. Der High-Teil dient zur Bestätigung, wenn die geforderte Aktion erfolgreich ausgeführt wurde.

SyncBit und SyncAck

Wenn das SyncBit und das SyncAck einer Kommunikationsrichtung gesetzt sind, gilt der Kanal als "synchronisiert", das heißt, es können Nachrichten in diese Richtung versendet werden. Das Statusbit der Gegenstelle muss zyklisch überprüft werden. Falls das SyncAck zurückgesetzt wurde, muss das eigene SyncBit angepasst werden. Bevor neue Daten übertragen werden können, muss der Kanal resynchronisiert werden.

SequenceCounter und SequenceAck

Die Kommunikationspartner prüfen zyklisch, ob sich das Low-Nibble der Gegenstelle ändert. Wenn ein Kommunikationspartner eine neue Sequenz vollständig auf die MTU geschrieben hat, erhöht er seinen SequenceCounter. Daraufhin übernimmt der Empfänger die aktuelle Sequenz und bestätigt den erfolgreichen Empfang per SequenceAck. Auf diese Weise wird ein Handshake-Verfahren initiiert.

Information:

Bei einer Unterbrechung der Kommunikation werden Segmente von unvollständig übermittelten Mitteilungen verworfen. Alle fertig übertragenen Nachrichten werden bearbeitet.

4.1.4.3 Synchronisieren

Beim Synchronisieren wird ein Kommunikationskanal geöffnet. Es muss sichergestellt sein, dass ein Modul vorhanden und der aktuelle Wert des SequenceCounters beim Empfänger der Nachricht hinterlegt ist.
Der Flatstream bietet die Möglichkeit Vollduplex zu kommunizieren. Beide Kanäle/Kommunikationsrichtungen können separat betrachtet werden. Sie müssen unabhängig voneinander synchronisiert werden, sodass theoretisch auch simplex kommuniziert werden könnte.

Synchronisation der Output-Richtung (Steuerung als Sender)

Die korrespondierenden Synchronisationsbits (OutputSyncBit und OutputSyncAck) sind zurückgesetzt. Aus diesem Grund können momentan keine Nachrichten von der Steuerung an das Modul per Flatstream übertragen werden.

Algorithmus

1) Steuerung muss 000 in OutputSequenceCounter schreiben und OutputSyncBit zurücksetzen. Steuerung muss High-Nibble des Registers "InputSequence" zyklisch abfragen (Prüfung ob 000 in OutputSequenceAck und 0 in OutputSyncAck).
Modul übernimmt den aktuellen Inhalt der InputMTU nicht, weil der Kanal noch nicht synchronisiert ist. *Modul gleicht OutputSequenceAck und OutputSyncAck an die Werte des OutputSequenceCounters bzw. des OutputSyncBits an.*
2) Wenn die Steuerung die erwarteten Werte in OutputSequenceAck und OutputSyncAck registriert, darf sie den OutputSequenceCounter inkrementieren. Die Steuerung fragt das High-Nibble des Registers "OutputSequence" weiter zyklisch ab (Prüfung ob 001 in OutputSequenceAck und 0 in InputSyncAck).
Modul übernimmt den aktuellen Inhalt der InputMTU nicht, weil der Kanal noch nicht synchronisiert ist. *Modul gleicht OutputSequenceAck und OutputSyncAck an die Werte des OutputSequenceCounters bzw. des OutputSyncBits an.*
3) Wenn die Steuerung die erwarteten Werte in OutputSequenceAck und OutputSyncAck registriert, darf sie das OutputSyncBit setzen. Die Steuerung fragt das High-Nibble des Registers "OutputSequence" weiter zyklisch ab (Prüfung ob 001 in OutputSequenceAck und 1 in InputSyncAck). **Hinweis:** Theoretisch könnten ab diesem Moment Daten übertragen werden. Es wird allerdings empfohlen, erst dann Daten zu übertragen, wenn die Output-Richtung vollständig synchronisiert ist.
Modul setzt OutputSyncAck.
Output-Richtung synchronisiert, Steuerung kann Daten an Modul senden.

Synchronisation der Input-Richtung (Steuerung als Empfänger)

Die korrespondierenden Synchronisationsbits (InputSyncBit und InputSyncAck) sind zurückgesetzt. Aus diesem Grund können momentan keine Nachrichten vom Modul an die Steuerung per Flatstream übertragen werden.

Algorithmus

Modul schreibt 000 in InputSequenceCounter und setzt InputSyncBit zurück. *Modul überwacht High-Nibble des Registers "OutputSequence" - erwartet 000 in InputSequenceAck bzw. 0 in InputSyncAck.*
1) Steuerung darf den aktuellen Inhalt der InputMTU nicht übernehmen, weil der Kanal noch nicht synchronisiert ist. Steuerung muss InputSequenceAck und InputSyncAck an die Werte des InputSequenceCounters bzw. des InputSyncBits angleichen.
Wenn das Modul die erwarteten Werte in InputSequenceAck und in InputSyncAck registriert, inkrementiert es den InputSequenceCounter. *Modul überwacht High-Nibble des Registers "OutputSequence" - erwartet 001 in InputSequenceAck bzw. 0 in InputSyncAck.*
2) Steuerung darf den aktuellen Inhalt der InputMTU nicht übernehmen, weil der Kanal noch nicht synchronisiert ist. Steuerung muss InputSequenceAck und InputSyncAck an die Werte des InputSequenceCounters bzw. des InputSyncBits angleichen.
Wenn das Modul die erwarteten Werte in InputSequenceAck und in InputSyncAck registriert, setzt es das InputSyncBit. *Modul überwacht High-Nibble des Registers "OutputSequence" - erwartet 1 in InputSyncAck.*
3) Steuerung darf InputSyncAck setzen. **Hinweis:** Theoretisch könnten bereits in diesem Zyklus Daten übertragen werden. Es gilt: Wenn das InputSyncBit gesetzt ist und der InputSequenceCounter um 1 erhöht wurde, müssen die Informationen der aktivierten Rx-Bytes übernommen und bestätigt werden (siehe dazu auch Kommunikation in Input-Richtung).
Input-Richtung synchronisiert, Modul kann Daten an Steuerung senden.

4.1.4.4 Senden und Empfangen

Wenn ein Kanal synchronisiert ist, gilt die Gegenstelle als empfangsbereit und der Sender kann Nachrichten verschicken. Bevor der Sender Daten absetzen kann, legt er das sogenannte Sendearray an, um den Anforderungen des Flatstreams gerecht zu werden.

Die sendende Station muss für jedes erstellte Segment ein individuelles Controlbyte generieren. Ein solches Controlbyte enthält Informationen, wie der nächste Teil der übertragenen Daten zu verarbeiten ist. Die Position des nächsten Controlbytes im Datenstrom kann variieren. Aus diesem Grund muss zu jedem Zeitpunkt eindeutig definiert sein, wann ein neues Controlbyte übermittelt wird. Das erste Controlbyte befindet sich immer auf dem ersten Byte der ersten Sequenz. Alle weiteren Positionen werden rekursiv mitgeteilt.

Flatstream-Formel zur Berechnung der Position des nächsten Controlbytes:

Position (nächstes Controlbyte) = aktuelle Position + 1 + Segmentlänge

Beispiel

Es werden 3 unabhängige Nachrichten (7 Bytes, 2 Bytes, 9 Bytes) über eine 7-Byte breite MTU übermittelt. Die sonstige Konfiguration entspricht den Standardeinstellungen.

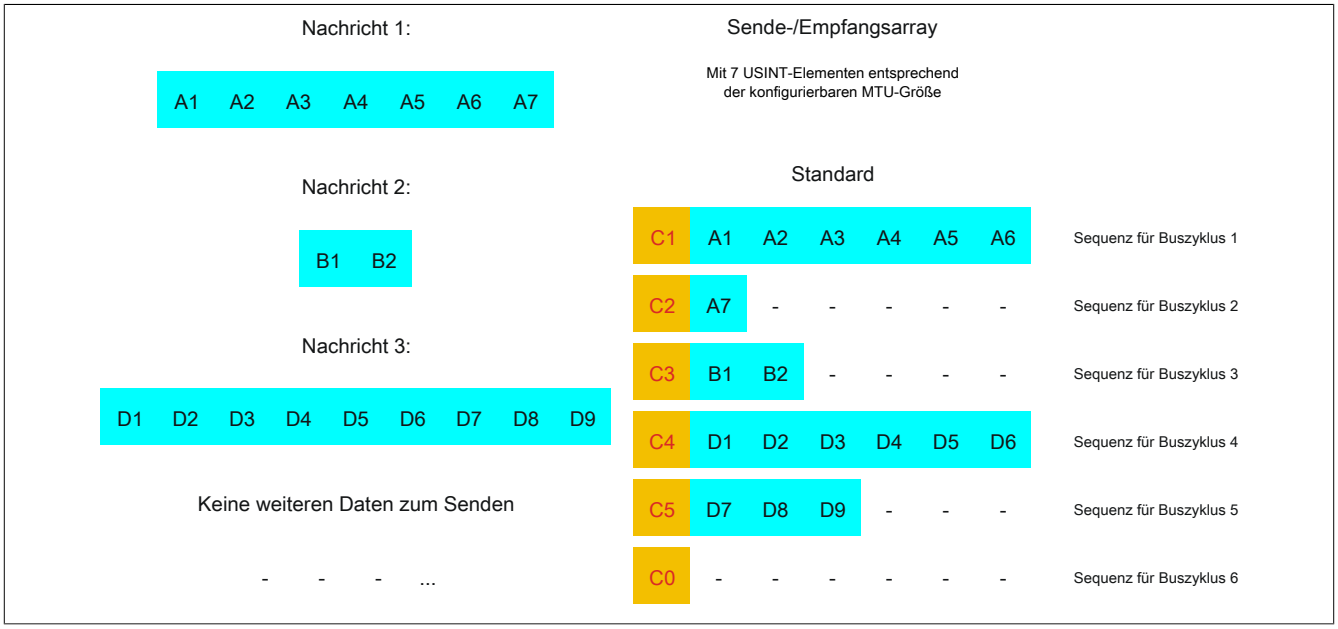


Abbildung 4: Sende-/Empfangsarray (Standard)

Zunächst müssen die Nachrichten in Segmente geteilt werden. Bei der Standardkonfiguration muss sichergestellt sein, dass jede Sequenz ein gesamtes Segment inklusive dem dazugehörigen Controlbyte aufnehmen kann. Die Sequenz ist auf die Größe der aktivierten MTU begrenzt, das heißt, ein Segment muss mindestens um 1 Byte kleiner sein als die aktivierte MTU.

MTU = 7 Bytes → max. Segmentlänge 6 Bytes

- Nachricht 1 (7 Bytes)
  - ⇨ erstes Segment = Controlbyte + 6 Datenbytes
  - ⇨ zweites Segment = Controlbyte + 1 Datenbyte
- Nachricht 2 (2 Bytes)
  - ⇨ erstes Segment = Controlbyte + 2 Datenbytes
- Nachricht 3 (9 Bytes)
  - ⇨ erstes Segment = Controlbyte + 6 Datenbytes
  - ⇨ zweites Segment = Controlbyte + 3 Datenbytes
- Keine weiteren Nachrichten
  - ⇨ C0-Controlbyte

Für jedes gebildete Segment muss ein spezifisches Controlbyte generiert werden. Außerdem wird das Controlbyte C0 generiert, um die Kommunikation auf Standby halten zu können.

C0 (Controlbyte0)			**C1 (Controlbyte1)**			**C2 (Controlbyte2)**		
- SegmentLength (0)	=	0	- SegmentLength (6)	=	6	- SegmentLength (1)	=	1
- nextCBPos (0)	=	0	- nextCBPos (0)	=	0	- nextCBPos (0)	=	0
- MessageEndBit (0)	=	0	- MessageEndBit (0)	=	0	- MessageEndBit (1)	=	128
Controlbyte	Σ	0	Controlbyte	Σ	6	Controlbyte	Σ	129

Tabelle 3: Flatstream-Ermittlung der Controlbytes für Beispiel mit Standardkonfiguration (Teil 1)

C3 (Controlbyte3)			**C4 (Controlbyte4)**			**C5 (Controlbyte5)**		
- SegmentLength (2)	=	2	- SegmentLength (6)	=	6	- SegmentLength (3)	=	3
- nextCBPos (0)	=	0	- nextCBPos (0)	=	0	- nextCBPos (0)	=	0
- MessageEndBit (1)	=	128	- MessageEndBit (0)	=	0	- MessageEndBit (1)	=	128
Controlbyte	Σ	130	Controlbyte	Σ	6	Controlbyte	Σ	131

Tabelle 4: Flatstream-Ermittlung der Controlbytes für Beispiel mit Standardkonfiguration (Teil 2)

4.1.4.4.1 Senden von Daten an ein Modul (Output)

Beim Senden muss das Sendearray im Programmablauf generiert werden. Danach wird es Sequenz für Sequenz über den Flatstream übertragen und vom Modul empfangen.

Information:

Obwohl alle B&R Module mit Flatstream-Kommunikation stets die kompakteste Übertragung in Output-Richtung unterstützen wird empfohlen die Übertragungsarrays für beide Kommunikationsrichtungen gleichermaßen zu gestalten.

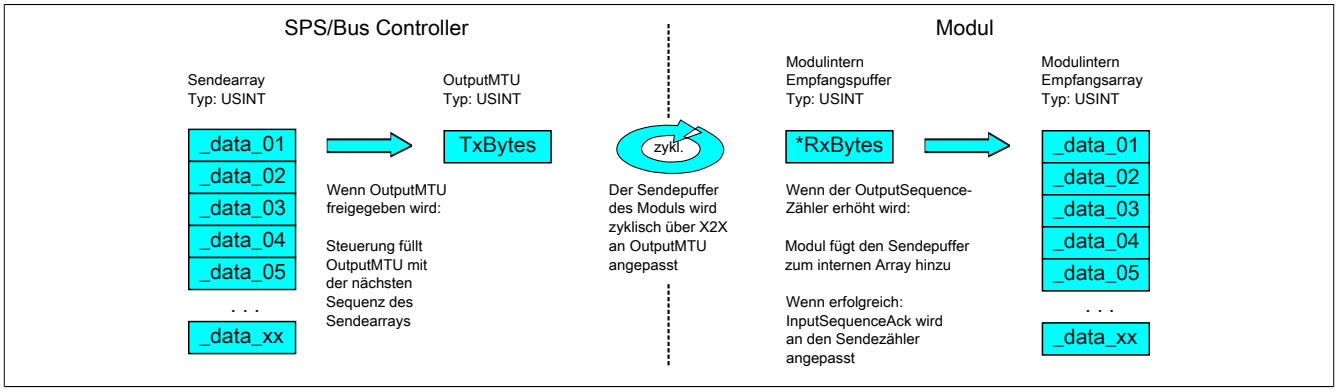


Abbildung 5: Kommunikation per Flatstream (Output)

Nachricht kleiner als OutputMTU

Die Länge der Nachricht sei zunächst kleiner als die OutputMTU. In diesem Fall würde eine Sequenz ausreichen, um die gesamte Nachricht und ein benötigtes Controlbyte zu übertragen.

Algorithmus

Zyklische Statusabfrage: *- Modul überwacht OutputSequenceCounter*
0) Zyklische Prüfungen: - Steuerung muss OutputSyncAck prüfen → falls OutputSyncAck = 0; OutputSyncBit zurücksetzen und Kanal resynchronisieren - Steuerung muss Freigabe der OutputMTU prüfen → falls OutputSequenceCounter > InputSequenceAck; MTU nicht freigegeben, weil letzte Sequenz noch nicht bestätigt
1) Vorbereitung (Sendearray anlegen): - Steuerung muss Nachricht auf zulässige Segmente aufteilen und entsprechende Controlbytes bilden - Steuerung muss Segmente und Controlbytes zu Sendearray zusammenfügen
2) Senden: - Steuerung überträgt das aktuelle Element des Sendearrays in die OutputMTU → OutputMTU wird zyklisch in den Sendepuffer des Moduls übertragen, aber noch nicht weiterverarbeitet - Steuerung muss OutputSequenceCounter erhöhen
Reaktion: *- Modul übernimmt die Bytes des internen Empfangspuffers und fügt sie an das interne Empfangsarray an* *- Modul sendet Bestätigung; schreibt Wert des OutputSequenceCounters auf OutputSequenceAck*
3) Abschluss: - Steuerung muss OutputSequenceAck überwachen → Eine Sequenz gilt erst dann als erfolgreich übertragen, wenn sie über das OutputSequenceAck bestätigt wurde. Um Übertragungsfehler auch bei der letzten Sequenz zu erkennen, muss sichergestellt werden, dass der *Abschluss* lange genug durchlaufen wird. **Hinweis:** Für eine exakte Überwachung der Kommunikationszeiten sollten die Taskzyklen gezählt werden, die seit der letzten Erhöhung des OutputSequenceCounters vergangen sind. Auf diese Weise kann die Anzahl der Buszyklen abgeschätzt werden, die bislang zur Übertragung benötigt wurden. Übersteigt der Überwachungszähler eine vorgegebene Schwelle, kann die Sequenz als verloren betrachtet werden. (Das Verhältnis von Bus- und Taskzyklus kann vom Anwender beeinflusst werden, sodass der Schwellwert individuell zu ermitteln ist.) - Weitere Sequenzen dürfen erst nach erfolgreicher Abschlussprüfung im nächsten Buszyklus versendet werden.

Nachricht größer als OutputMTU

Das Sendearray, welches im Programmablauf erstellt werden muss, besteht aus mehreren Elementen. Der Anwender muss die Control- und Datenbytes korrekt anordnen und die Arrayelemente nacheinander übertragen. Der Übertragungsalgorithmus bleibt gleich und wird ab dem Punkt *zyklische Prüfungen* wiederholt durchlaufen.

Allgemeines Ablaufdiagramm

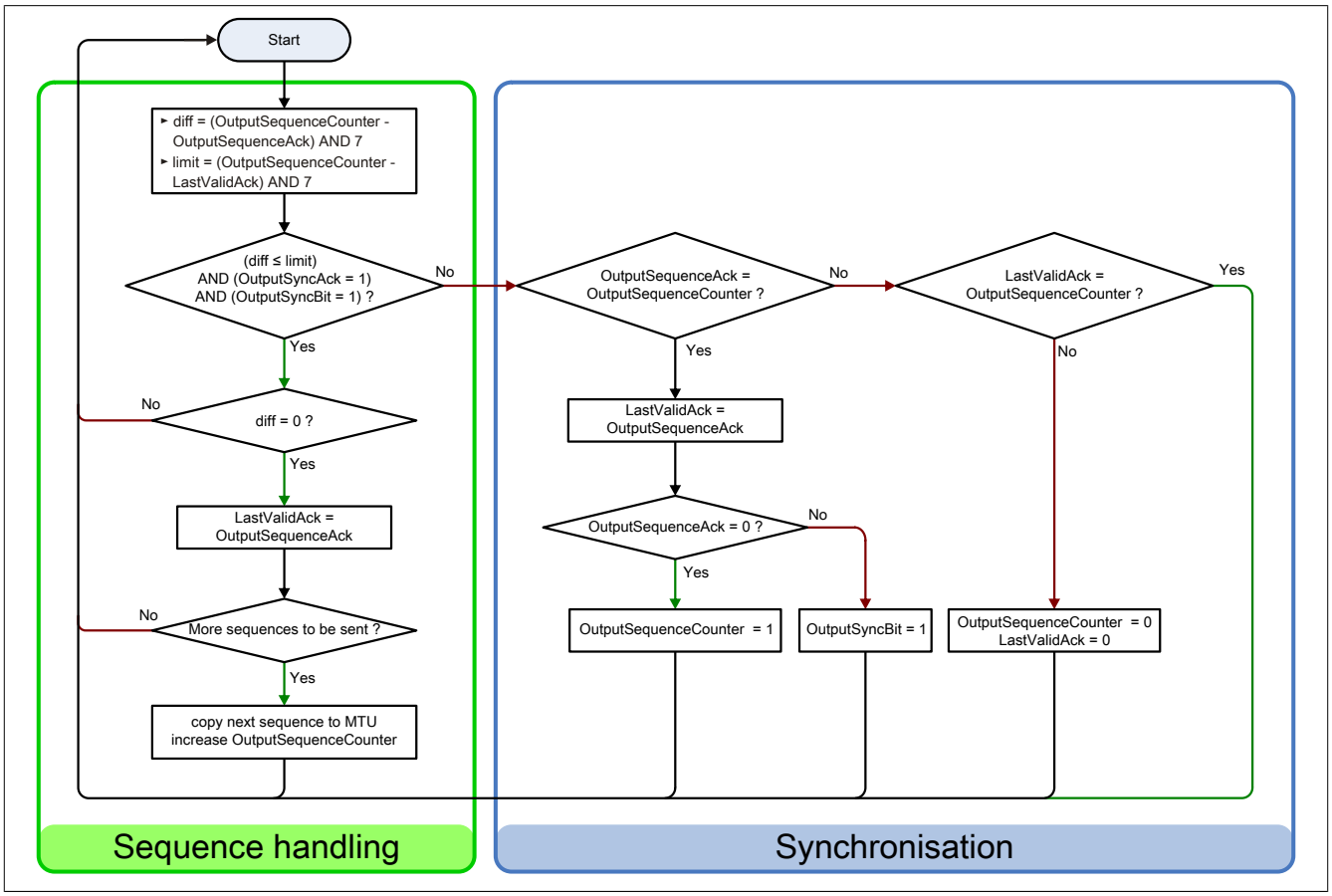


Abbildung 6: Ablaufdiagramm für Output-Richtung

4.1.4.4.2 Empfangen von Daten aus einem Modul (Input)

Beim Empfangen von Daten wird das Sendearray vom Modul generiert, über den Flatstream übertragen und muss auf dem Empfangsarray abgebildet werden. Die Struktur des ankommenden Datenstroms kann über das Modusregister eingestellt werden. Der Algorithmus zum Empfangen bleibt dabei aber unverändert.

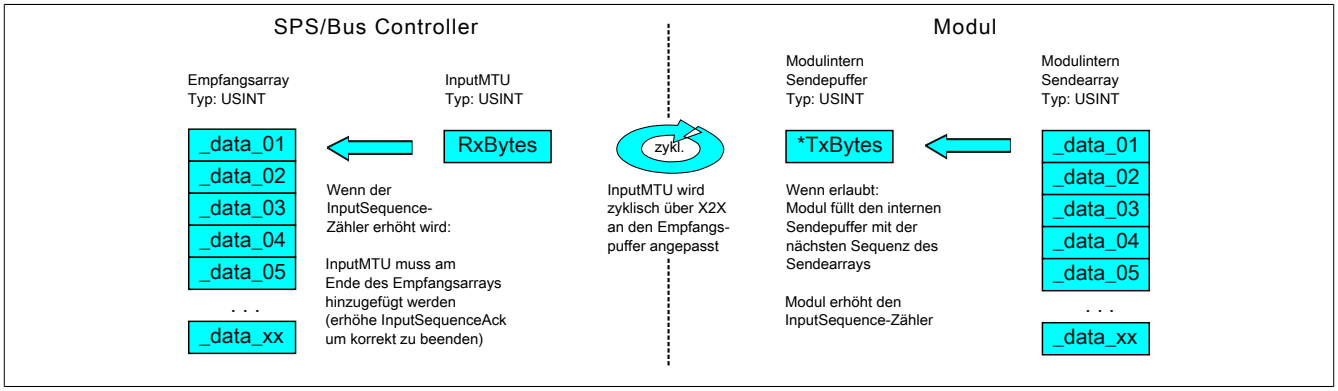


Abbildung 7: Kommunikation per Flatstream (Input)

Algorithmus

0) Zyklische Statusabfrage: - Steuerung muss InputSequenceCounter überwachen
Zyklische Prüfungen: *- Modul prüft InputSyncAck* *- Modul prüft InputSequenceAck*
Vorbereitung: *- Modul bildet Segmente bzw. Controlbytes und legt Sendearray an*
Aktion: *- Modul überträgt das aktuelle Element des internen Sendearrays in den internen Sendepuffer* *- Modul erhöht InputSequenceCounter*
1) Empfangen (sobald InputSequenceCounter erhöht): - Steuerung muss Daten aus InputMTU übernehmen und an das Ende des Empfangsarrays anfügen - Steuerung muss InputSequenceAck an InputSequenceCounter der aktuell verarbeiteten Sequenz angleichen
Abschluss: *- Modul überwacht InputSequenceAck* *→ Eine Sequenz gilt erst dann als erfolgreich übertragen, wenn sie über das InputSequenceAck bestätigt wurde.* *- Weitere Sequenzen werden erst nach erfolgreicher Abschlussprüfung im nächsten Buszyklus versendet.*

Allgemeines Ablaufdiagramm

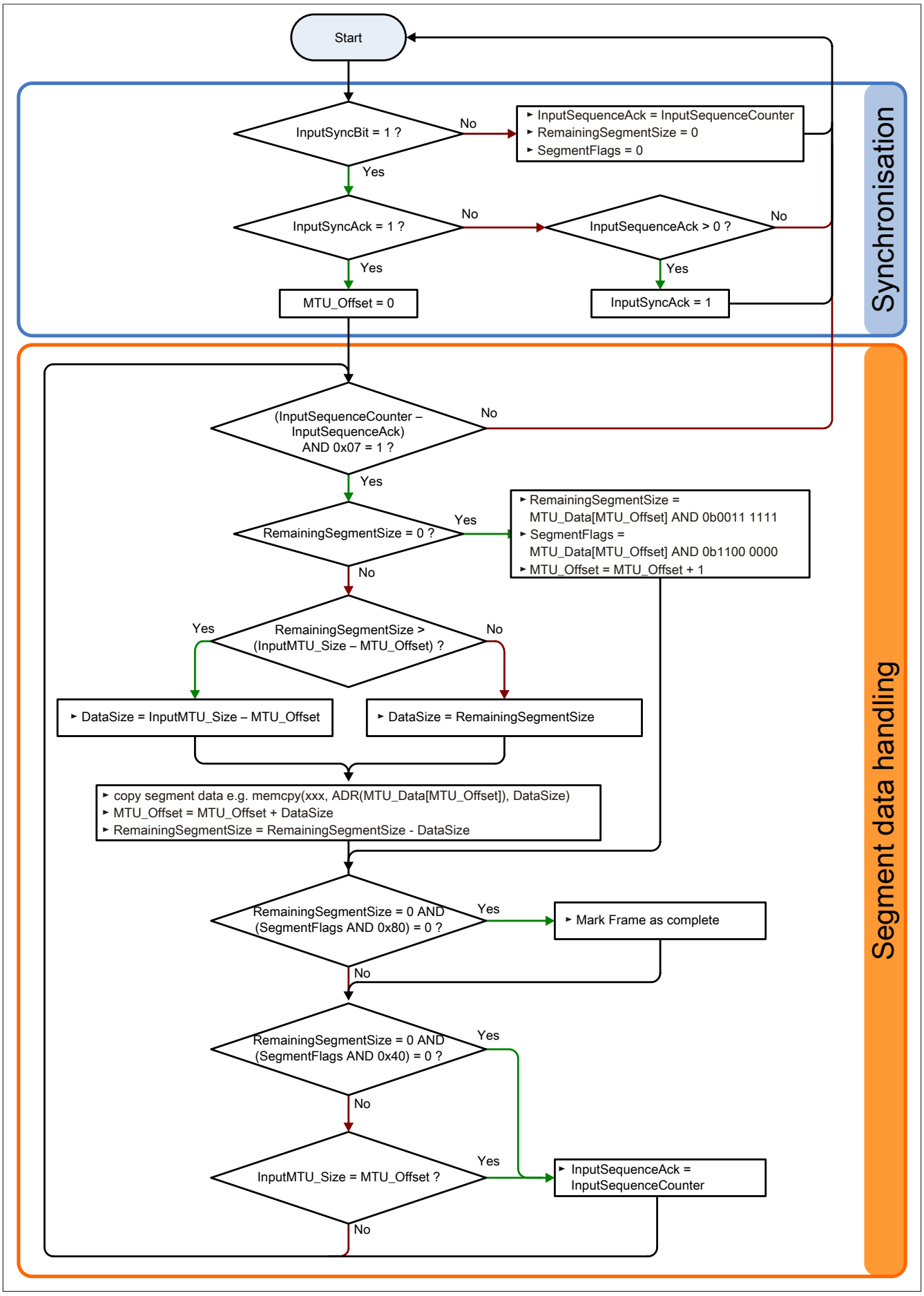


Abbildung 8: Ablaufdiagramm für Input-Richtung

4.1.4.4.3 Details

Es wird empfohlen die übertragenen Nachrichten in separate Empfangsarrays abzulegen

Nach der Übermittlung eines gesetzten MessageEndBits sollte das Folgesegment zum Empfangsarray hinzugefügt werden. Danach ist die Mitteilung vollständig und kann intern weiterverarbeitet werden. Für die nächste Nachricht sollte ein neues/separates Array angelegt werden.

Information:

Bei der Übertragung mit MultiSegmentMTUs können sich mehrere kurze Nachrichten in einer Sequenz befinden. Im Programmablauf muss sichergestellt sein, dass genügend Empfangsarrays verwaltet werden können. Das Acknowledge-Register darf erst nach Übernahme der gesamten Sequenz angepasst werden.

Wenn ein SequenceCounter um mehr als einen Zähler inkrementiert wird, liegt ein Fehler vor

In diesem Fall stoppt der Empfänger. Alle weiteren eintreffenden Sequenzen werden ignoriert, bis die Sendung mit dem korrekten SequenceCounter wiederholt wird. Durch diese Reaktion erhält der Sender keine Bestätigungen mehr für die abgesetzten Sequenzen. Über den SequenceAck der Gegenstelle kann der Sender die letzte erfolgreich übertragene Sequenz identifizieren und die Übertragung ab dieser Stelle fortsetzen.

Information:

Beim Betrieb ohne Forward ist diese Situation sehr unwahrscheinlich.

Bestätigungen müssen auf Gültigkeit geprüft werden

Wenn der Empfänger eine Sequenz erfolgreich übernommen hat, muss sie bestätigt werden. Dazu übernimmt der Empfänger den mitgesendeten Wert des SequenceCounters und gleicht den SequenceAck daran an. Der Absender liest das SequenceAck und registriert die erfolgreiche Übermittlung. Falls dem Absender eine Sequenz bestätigt wird, die noch nicht abgesendet wurde, muss die Übertragung unterbrochen und der Kanal resynchronisiert werden. Die Synchronisationsbits werden zurückgesetzt und die aktuelle/unvollständige Nachricht wird verworfen. Sie muss nach der Resynchronisierung des Kanals erneut versendet werden.

4.1.4.5 Flatstream-Modus

In Input-Richtung wird das Sende-Array automatisch generiert. Dem Anwender werden über den Flatstream-Modus mehrere Optionen zur Verfügung gestellt, um eine kompaktere Anordnung beim eintreffenden Datenstrom zu erlauben. Diese sind:

- Standard
- MultiSegmentMTU erlaubt
- Große Segmente erlaubt

Nach der Aktivierung muss der Programmablauf zur Auswertung entsprechend angepasst werden.

Information:

Alle B&R Module, die den Flatstream-Modus anbieten, unterstützen in Output-Richtung die Optionen "große Segmente" und "MultiSegmentMTU". Nur für die Input-Richtung muss die kompakte Übertragung explizit erlaubt werden.

Standard

Per Standard sind beide Optionen zur kompakten Übertragung in Input-Richtung deaktiviert.

1. Vom Modul werden nur Segmente gebildet, die mindestens ein Byte kleiner sind als die aktivierte MTU. Jede Sequenz beginnt mit einem Controlbyte, sodass der Datenstrom klar strukturiert ist und relativ einfach ausgewertet werden kann.
2. Weil die Länge einer Flatstream-Nachricht beliebig lang sein darf, füllt das letzte Segment der Mitteilung häufig nicht den gesamten Platz der MTU aus. Per Standard werden während eines solchen Übertragungszyklus die restlichen Bytes nicht verwendet.

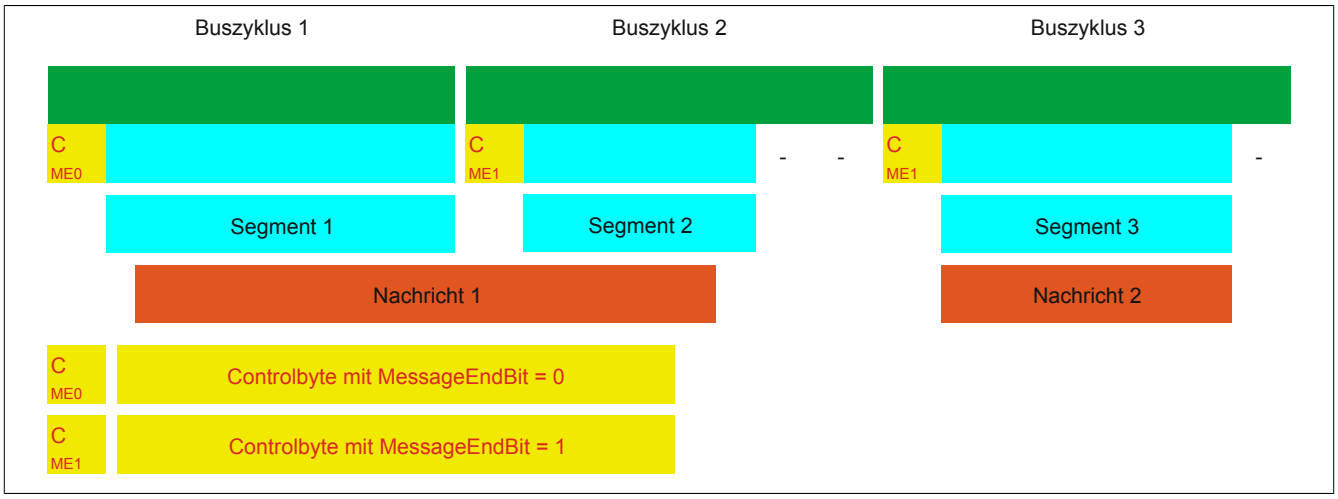


Abbildung 9: Anordnung von Nachrichten in der MTU (Standard)

MultiSegmentMTU erlaubt

Bei dieser Option wird die InputMTU vollständig befüllt (wenn genügend Daten anstehen). Die zuvor frei gebliebenen Rx-Bytes übertragen die nächsten Controlbytes bzw. deren Segmente. Auf diese Weise können die aktivierten Rx-Bytes effizienter genutzt werden.

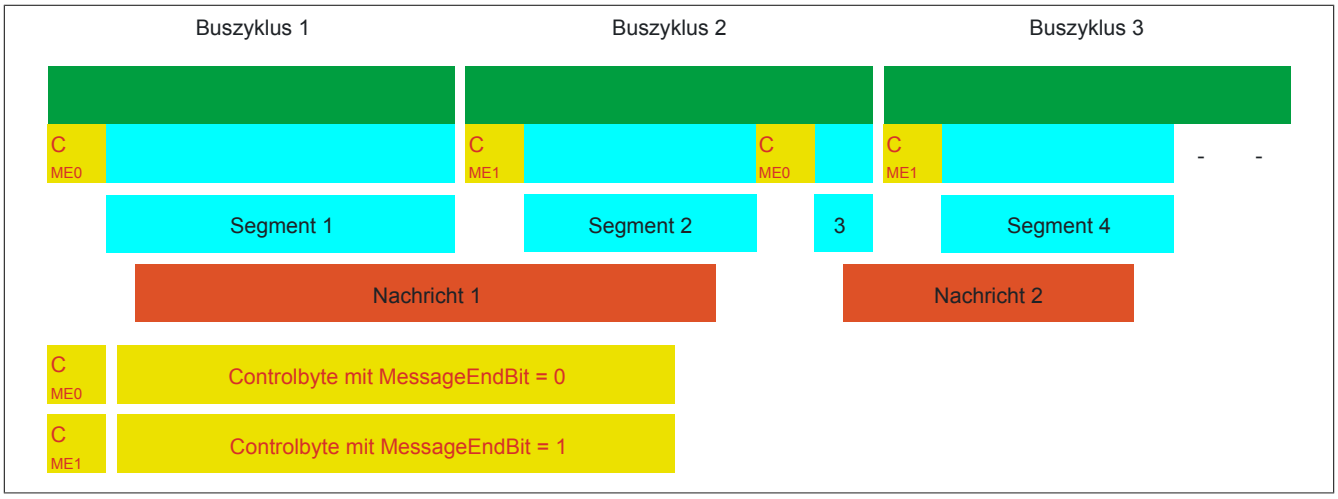


Abbildung 10: Anordnung von Nachrichten in der MTU (MultiSegmentMTU)

Große Segmente erlaubt

Bei der Übertragung sehr langer Mitteilungen bzw. bei der Aktivierung von nur wenigen Rx-Bytes müssen per Standard sehr viele Segmente gebildet werden. Das Bussystem wird stärker belastet als nötig, weil für jedes Segment ein zusätzliches Controlbyte erstellt und übertragen wird. Mit der Option "große Segmente" wird die Segmentlänge unabhängig von der InputMTU auf 63 Bytes begrenzt. Ein Segment darf sich über mehrere Sequenzen erstrecken, das heißt, es können auch reine Sequenzen ohne Controlbyte auftreten.

Information:

Die Möglichkeit eine Nachricht auf mehrere Segmente aufzuteilen bleibt erhalten, das heißt, wird diese Option genutzt und treten Nachrichten mit mehr als 63 Bytes auf, kann die Mitteilung weiterhin auf mehrere Segmente verteilt werden.

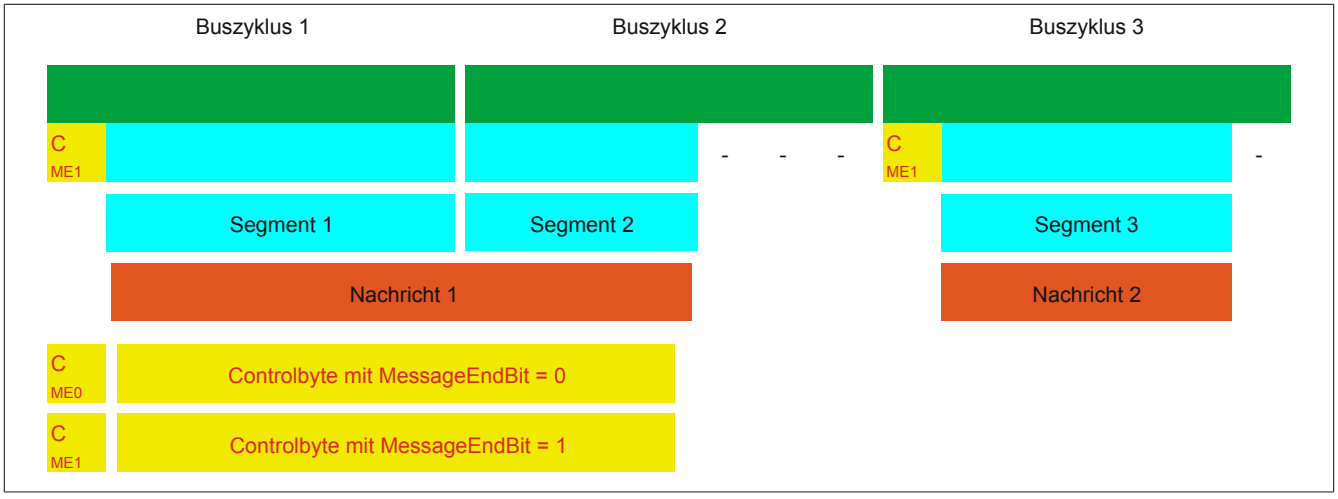


Abbildung 11: Anordnung von Nachrichten in der MTU (große Segmente)

Anwendung beider Optionen

Die beiden Optionen dürfen auch gleichzeitig angewendet werden.

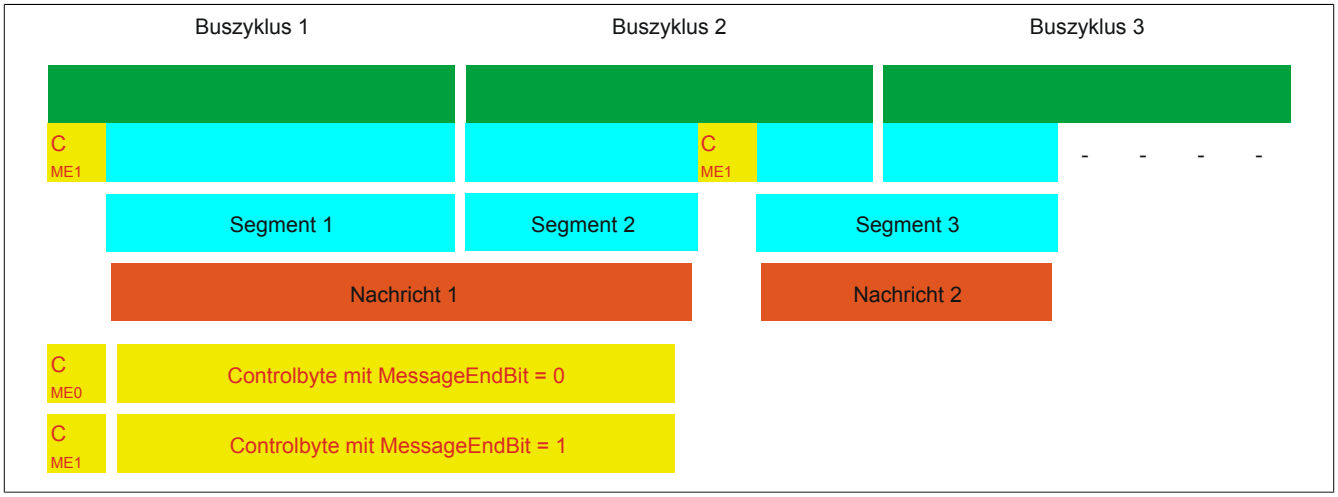


Abbildung 12: Anordnung von Nachrichten in der MTU (große Segmente und MultiSegmentMTU)

4.1.4.6 Anpassung des Flatstreams

Wenn die Strukturierung der Nachrichten verändert wurde, verändert sich auch die Anordnung der Daten im Sende-/Empfangsarray. Für das eingangs genannte Beispiel ergeben sich die folgenden Änderungen.

MultiSegmentMTU

Wenn MultiSegmentMTUs erlaubt sind, können "freie Stellen" in einer MTU genutzt werden. Diese "freien Stellen“ entstehen, wenn das letzte Segment einer Nachricht nicht die gesamte MTU ausnutzt. MultiSegmentMTUs ermöglichen die Verwendung dieser Bits, um die folgenden Controlbytes bzw. Segmente zu übertragen. Im Programmablauf wird das "nextCBPos"-Bit innerhalb des Controlbytes gesetzt, damit der Empfänger das nächste Controlbyte korrekt identifizieren kann.

Beispiel

Es werden 3 unabhängige Nachrichten (7 Bytes, 2 Bytes, 9 Bytes) über eine 7-Byte breite MTU übermittelt. Die Konfiguration erlaubt die Übertragung von MultiSegmentMTUs.

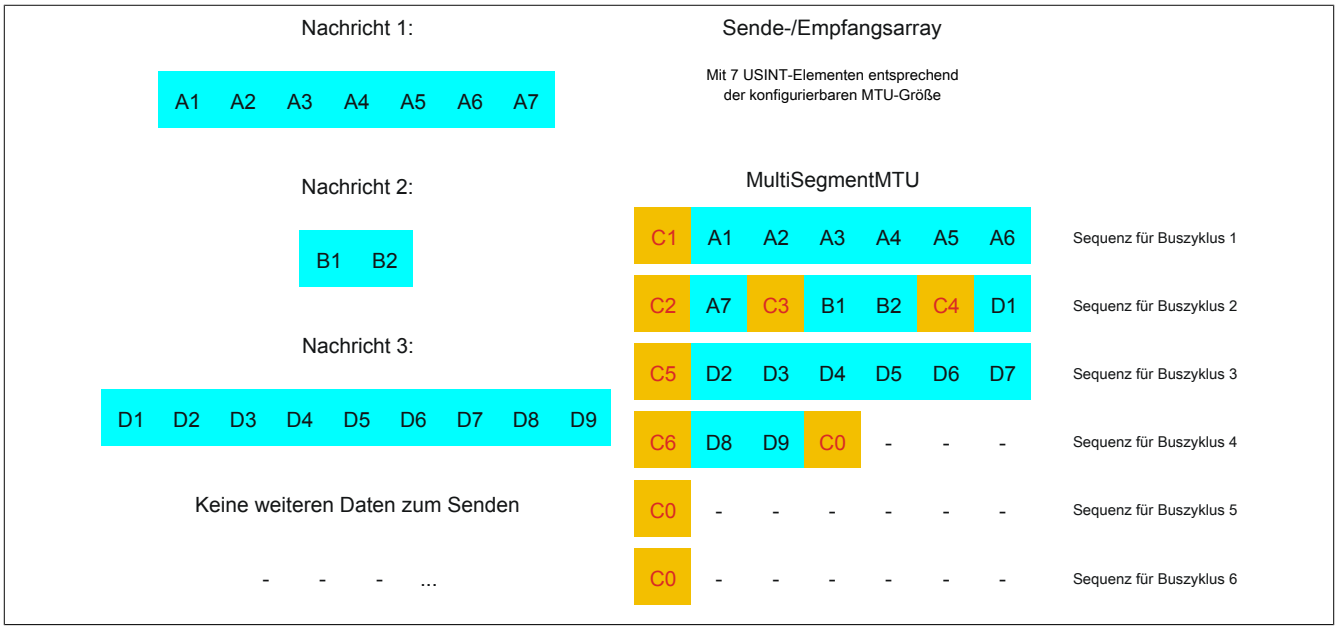


Abbildung 13: Sende-/Empfangsarray (MultiSegmentMTU)

Zunächst müssen die Nachrichten in Segmente geteilt werden. Wie in der Standardkonfiguration muss sichergestellt sein, dass jede Sequenz mit einem Controlbyte beginnt. Die freien Bits in der MTU am Ende einer Nachricht, werden allerdings mit Daten der Folgenachricht aufgefüllt. Bei dieser Option wird das Bit "nextCBPos" immer gesetzt, wenn im Anschluss an das Controlbyte Nutzdaten übertragen werden.

MTU = 7 Bytes → max. Segmentlänge 6 Bytes

- Nachricht 1 (7 Bytes)
  - ⇨ erstes Segment = Controlbyte + 6 Datenbytes (MTU voll)
  - ⇨ zweites Segment = Controlbyte + 1 Datenbyte (MTU noch 5 leere Bytes)
- Nachricht 2 (2 Bytes)
  - ⇨ erstes Segment = Controlbyte + 2 Datenbytes (MTU noch 2 leere Bytes)
- Nachricht 3 (9 Bytes)
  - ⇨ erstes Segment = Controlbyte + 1 Datenbyte (MTU voll)
  - ⇨ zweites Segment = Controlbyte + 6 Datenbytes (MTU voll)
  - ⇨ drittes Segment = Controlbyte + 2 Datenbytes (MTU noch 4 leere Bytes)
- Keine weiteren Nachrichten
  - ⇨ C0-Controlbyte

Für jedes gebildete Segment muss ein spezifisches Controlbyte generiert werden. Außerdem wird das Controlbyte C0 generiert, um die Kommunikation auf Standby halten zu können.

C1 (Controlbyte1)			**C2 (Controlbyte2)**			**C3 (Controlbyte3)**		
- SegmentLength (6)	=	6	- SegmentLength (1)	=	1	- SegmentLength (2)	=	2
- nextCBPos (1)	=	64	- nextCBPos (1)	=	64	- nextCBPos (1)	=	64
- MessageEndBit (0)	=	0	- MessageEndBit (1)	=	128	- MessageEndBit (1)	=	128
Controlbyte	Σ	70	Controlbyte	Σ	193	Controlbyte	Σ	194

Tabelle 5: Flatstream-Ermittlung der Controlbytes für Beispiel mit MultiSegmentMTU (Teil 1)

Warnung!

Die zweite Sequenz darf erst über den SequenceAck bestätigt werden, wenn sie vollständig verarbeitet wurde. Im Beispiel befinden sich 3 verschiedene Segmente innerhalb der zweiten Sequenz, das heißt, im Programmablauf müssen ausreichend Empfänger-Arrays gehandhabt werden können.

C4 (Controlbyte4)			**C5 (Controlbyte5)**			**C6 (Controlbyte6)**		
- SegmentLength (1)	=	1	- SegmentLength (6)	=	6	- SegmentLength (2)	=	2
- nextCBPos (6)	=	6	- nextCBPos (1)	=	64	- nextCBPos (1)	=	64
- MessageEndBit (0)	=	0	- MessageEndBit (1)	=	0	- MessageEndBit (1)	=	128
Controlbyte	Σ	7	Controlbyte	Σ	70	Controlbyte	Σ	194

Tabelle 6: Flatstream-Ermittlung der Controlbytes für Beispiel mit MultiSegmentMTU (Teil 2)

Große Segmente

Die Segmente werden auf maximal 63 Bytes begrenzt. Damit können sie größer sein als die aktive MTU. Diese großen Segmente werden bei der Übertragung auf mehrere Sequenzen aufgeteilt. Es können Sequenzen ohne Controlbyte auftreten, die vollständig mit Nutzdaten befüllt sind.

Information:

Um die Größe eines Datenpakets nicht ebenfalls auf 63 Bytes zu begrenzen, bleibt die Möglichkeit erhalten, eine Nachricht in mehrere Segmente zu untergliedern.

Beispiel

Es werden 3 unabhängige Nachrichten (7 Bytes, 2 Bytes, 9 Bytes) über eine 7-Byte breite MTU übermittelt. Die Konfiguration erlaubt die Übertragung von großen Segmenten.

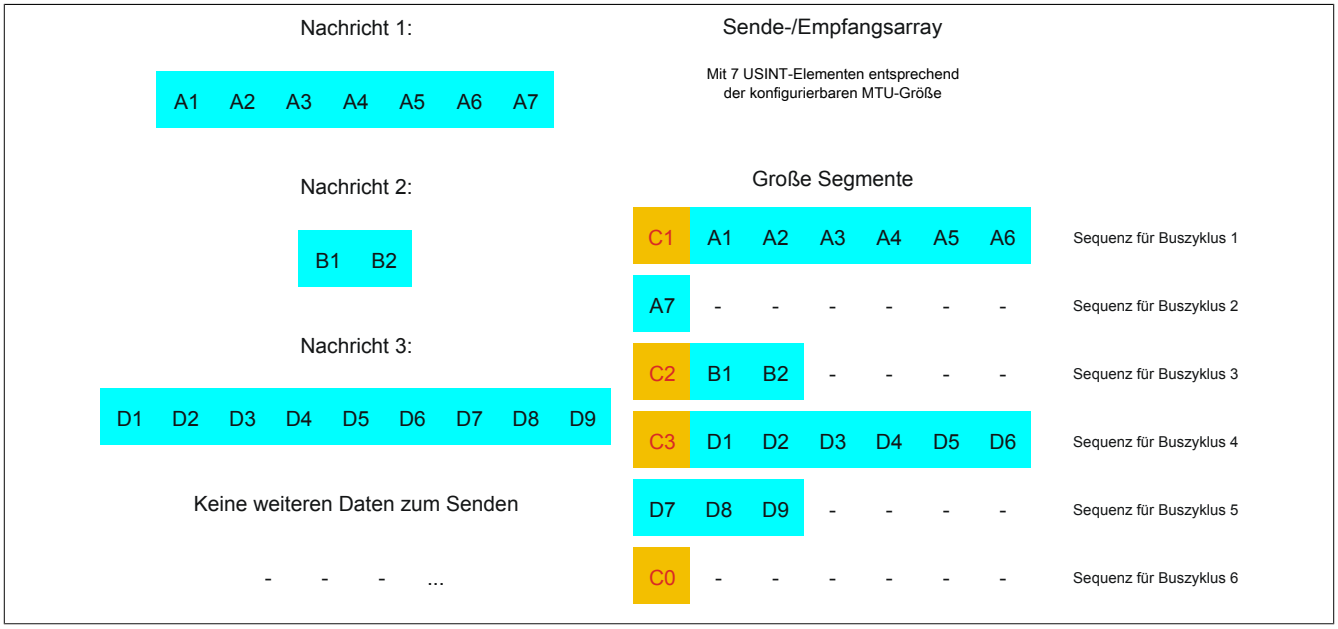


Abbildung 14: Sende-/Empfangsarray (große Segmente)

Zunächst müssen die Nachrichten in Segmente geteilt werden. Durch die Möglichkeit große Segmente zu bilden, müssen Nachrichten seltener geteilt werden, sodass weniger Controlbytes generiert werden müssen.

Große Segmente erlaubt → max. Segmentlänge 63 Bytes

- Nachricht 1 (7 Bytes)
  - ⇨ erstes Segment = Controlbyte + 7 Datenbytes
- Nachricht 2 (2 Bytes)
  - ⇨ erstes Segment = Controlbyte + 2 Datenbytes
- Nachricht 3 (9 Bytes)
  - ⇨ erstes Segment = Controlbyte + 9 Datenbytes
- Keine weiteren Nachrichten
  - ⇨ C0-Controlbyte

Für jedes gebildete Segment muss ein spezifisches Controlbyte generiert werden. Außerdem wird das Controlbyte C0 generiert, um die Kommunikation auf Standby halten zu können.

C1 (Controlbyte1)			**C2 (Controlbyte2)**			**C3 (Controlbyte3)**		
- SegmentLength (7)	=	7	- SegmentLength (2)	=	2	- SegmentLength (9)	=	9
- nextCBPos (0)	=	0	- nextCBPos (0)	=	0	- nextCBPos (0)	=	0
- MessageEndBit (1)	=	128	- MessageEndBit (1)	=	128	- MessageEndBit (1)	=	128
Controlbyte	Σ	135	Controlbyte	Σ	130	Controlbyte	Σ	137

Tabelle 7: Flatstream-Ermittlung der Controlbytes für Beispiel mit großen Segmenten

Große Segmente und MultiSegmentMTU

Beispiel

Es werden 3 unabhängige Nachrichten (7 Bytes, 2 Bytes, 9 Bytes) über eine 7-Byte breite MTU übermittelt. Die Konfiguration erlaubt sowohl die Übertragung von MultiSegmentMTUs als auch von großen Segmenten.

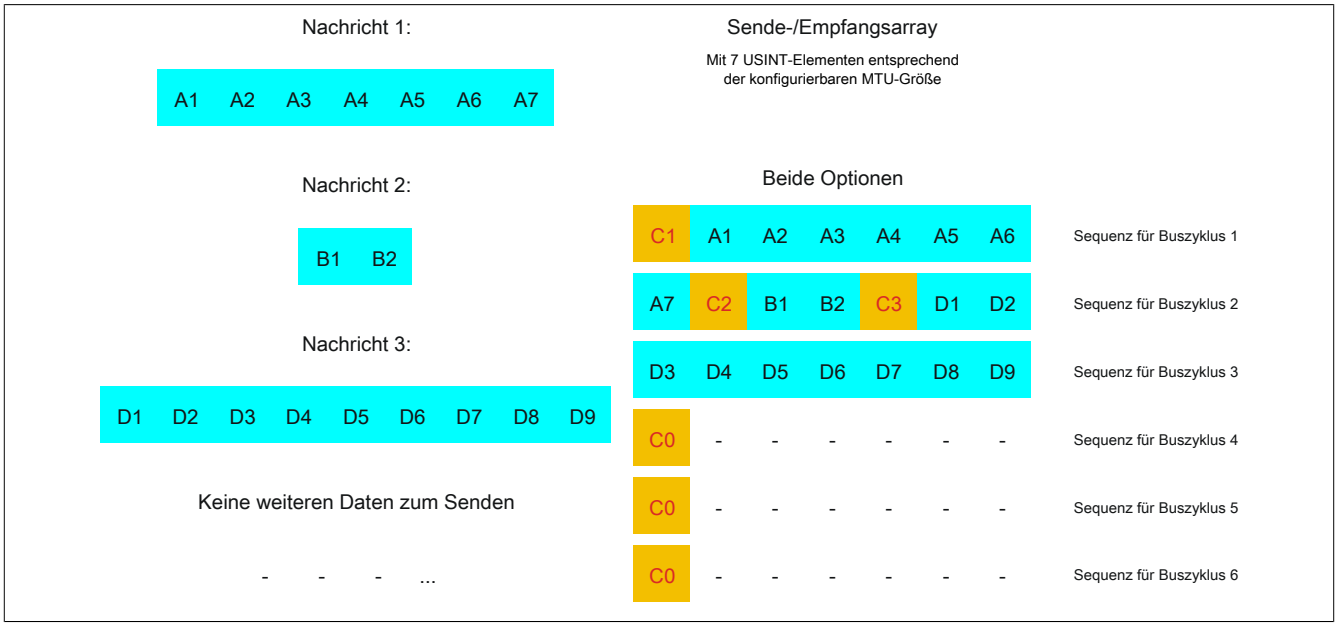


Abbildung 15: Sende-/Empfangsarray (große Segmente und MultiSegmentMTU)

Zunächst müssen die Nachrichten in Segmente geteilt werden. Wenn das letzte Segment einer Nachricht die MTU nicht komplett befüllt, darf sie für weitere Daten aus dem Datenstrom verwendet werden. Das Bit "nextCBPos" muss immer gesetzt werden, wenn das Controlbyte zu einem Segment mit Nutzdaten gehört.
Durch die Möglichkeit große Segmente zu bilden, müssen Nachrichten seltener geteilt werden, sodass weniger Controlbytes generiert werden müssen. Die Generierung der Controlbytes erfolgt auf die gleiche Weise, wie bei der Option "große Segmente".

Große Segmente erlaubt → max. Segmentlänge 63 Bytes

- Nachricht 1 (7 Bytes)
  - ⇨ erstes Segment = Controlbyte + 7 Datenbytes
- Nachricht 2 (2 Bytes)
  - ⇨ erstes Segment = Controlbyte + 2 Datenbytes
- Nachricht 3 (9 Bytes)
  - ⇨ erstes Segment = Controlbyte + 9 Datenbytes
- Keine weiteren Nachrichten
  - ⇨ C0-Controlbyte

Für jedes gebildete Segment muss ein spezifisches Controlbyte generiert werden. Außerdem wird das Controlbyte C0 generiert, um die Kommunikation auf Standby halten zu können.

C1 (Controlbyte1)			**C2 (Controlbyte2)**			**C3 (Controlbyte3)**		
- SegmentLength (7)	=	7	- SegmentLength (2)	=	2	- SegmentLength (9)	=	9
- nextCBPos (0)	=	0	- nextCBPos (0)	=	0	- nextCBPos (0)	=	0
- MessageEndBit (1)	=	128	- MessageEndBit (1)	=	128	- MessageEndBit (1)	=	128
Controlbyte	Σ	135	Controlbyte	Σ	130	Controlbyte	Σ	137

Tabelle 8: Flatstream-Ermittlung der Controlbytes für Beispiel mit großen Segmenten und MultiSegmentMTU

4.1.5 Die "Forward"-Funktion am Beispiel des X2X Link

Bei der "Forward"-Funktion handelt es sich um eine Methode, die Datenrate des Flatstreams deutlich zu erhöhen. Das grundsätzliche Prinzip wird auch in anderen technischen Bereichen angewandt, z. B. beim "Pipelining" für Mikroprozessoren.

4.1.5.1 Das Funktionsprinzip

Bei der Kommunikation mittels X2X Link werden 5 Teilschritte durchlaufen, um eine Flatstream-Sequenz zu übertragen. Eine erfolgreiche Sequenzübertragung benötigt deshalb mindestens 5 Buszyklen.

	Schritt I	Schritt II	Schritt III	Schritt IV	Schritt V
Aktionen	Sequenz aus Sendearray übertragen, SequenceCounter erhöhen	Zyklischer Abgleich MTU und Modulpuffer	Sequenz an Empfangsarray fügen, SequenceAck anpassen	Zyklischer Abgleich MTU und Modulpuffer	Prüfung des SequenceAck
Ressource	Sender (Task zum Versenden)	Bussystem (Richtung 1)	Empfänger (Task zum Empfangen)	Bussystem (Richtung 2)	Sender (Task zur Ack-Prüfung)

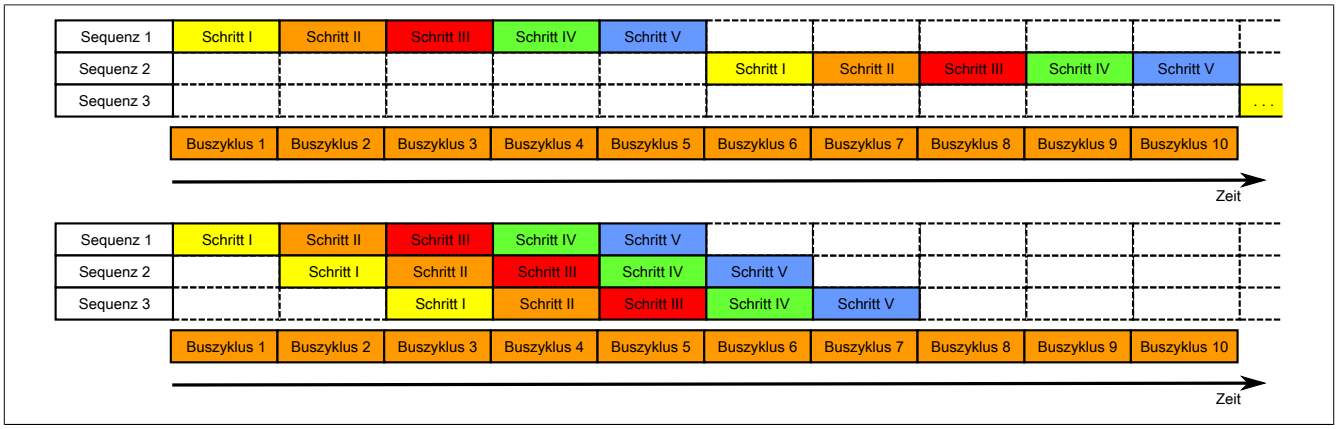


Abbildung 16: Vergleich Übertragung ohne bzw. mit Forward

Jeder der 5 Schritte (Tasks) beansprucht unterschiedliche Ressourcen. Ohne die Verwendung des Forward werden die Sequenzen nacheinander abgearbeitet. Jede Ressource ist nur dann aktiv, wenn sie für die aktuelle Teilaktion benötigt wird.
Beim Forward kann die Ressource, welche ihre Aufgabe abgearbeitet hat, bereits für die nächste Nachricht genutzt werden. Dazu wird die Bedingung zur MTU-Freigabe verändert. Die Sequenzen werden zeitgesteuert auf die MTU gelegt. Die Sendestation wartet nicht mehr auf die Bestätigung durch das SequenceAck und nutzt auf diese Weise die gegebene Bandbreite effizienter.
Im Idealfall arbeiten alle Ressourcen während jedes Buszyklus. Der Empfänger muss weiterhin jede erhaltene Sequenz bestätigen. Erst wenn das SequenceAck angepasst und vom Absender geprüft wurde, gilt die Sequenz als erfolgreich übertragen.

4.1.5.2 Konfiguration

Die Forward-Funktion muss nur für die Input-Richtung freigeschaltet werden. Die Flatstream-Module wurden dahingehend optimiert, diese Funktion unterstützen zu können. In Output-Richtung kann die Forward-Funktion genutzt werden, sobald die Größe der OutputMTU vorgegeben ist.

Information:

Die Register sind in Abschnitt "Flatstream-Register" auf Seite 73 beschrieben.

4.1.5.2.1 Verzögerungszeit

Die Verzögerungszeit wird in µs vorgegeben. Das Modul muss nach dem Versand einer Sequenz diese Zeit abwarten, bevor es im darauf folgenden Buszyklus neue Daten in die MTU schreiben darf. Die Programmroutine zum Empfang von Sequenzen aus einem Modul kann somit auch in einer Taskklasse betrieben werden deren Zykluszeit langsamer ist als der Buszyklus.

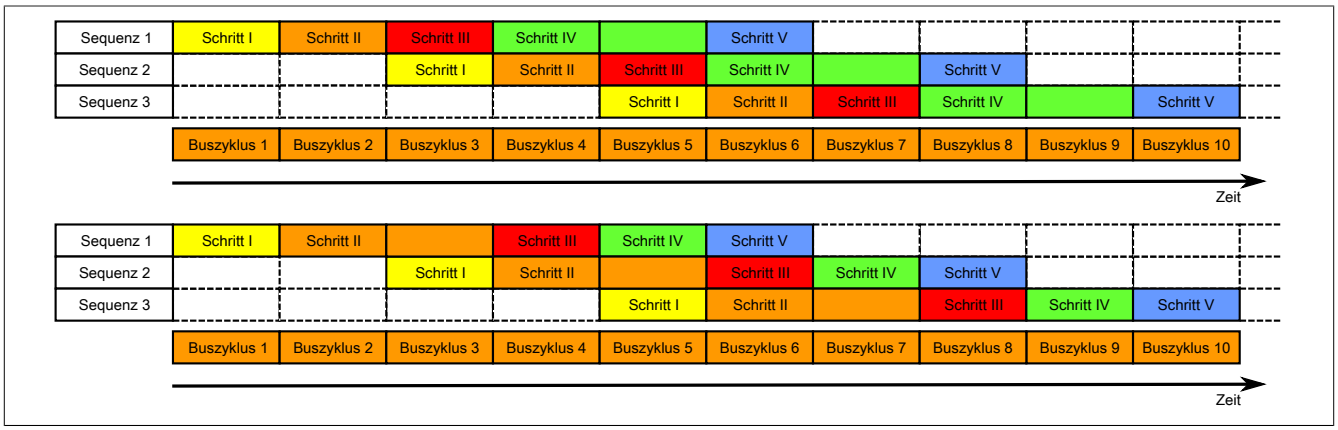


Abbildung 17: Auswirkung des ForwardDelay bei der Flatstream-Kommunikation mit Forward

Im Programmablauf muss sichergestellt werden, dass die Steuerung alle eintreffenden InputSequences bzw. InputMTUs verarbeitet. Der ForwardDelay-Wert bewirkt in Output-Richtung eine verzögerte Bestätigung und in Input-Richtung einen verzögerten Empfang. Auf diese Weise hat die Steuerung länger Zeit die eintreffende InputSequence bzw. InputMTU zu verarbeiten.

4.1.5.3 Senden und Empfangen mit Forward

Der grundsätzliche Algorithmus zum Senden bzw. Empfangen von Daten bleibt gleich. Durch den Forward können bis zu 7 unbestätigte Sequenzen abgesetzt werden. Sequenzen können gesendet werden, ohne die Bestätigung der vorangegangenen Nachricht abzuwarten. Da die Wartezeit zwischen Schreiben und Rückmeldung entfällt, können im gleichen Zeitraum erheblich mehr Daten übertragen werden.

Algorithmus zum Senden

Zyklische Statusabfrage: *- Modul überwacht OutputSequenceCounter*
0) Zyklische Prüfungen: - Steuerung muss OutputSyncAck prüfen → falls OutputSyncAck = 0; OutputSyncBit zurücksetzen und Kanal resynchronisieren - Steuerung muss Freigabe der OutputMTU prüfen → falls OutputSequenceCounter > OutputSequenceAck + 7, in diesem Fall nicht freigegeben, weil letzte Sequenz noch nicht quittiert
1) Vorbereitung (Sendearray anlegen): - Steuerung muss Nachricht auf zulässige Segmente aufteilen und entsprechende Controlbytes bilden - Steuerung muss Segmente und Controlbytes zu Sendearray zusammenfügen
2) Senden: - Steuerung muss aktuellen Teil des Sendearrays in die OutputMTU übertragen - Steuerung muss OutputSequenceCounter erhöhen, damit Sequenz vom Modul übernommen wird - Steuerung darf im nächsten Buszyklus erneut s*enden*, falls MTU freigegeben ist
Reaktion des Moduls, weil OutputSequenceCounter > OutputSequenceAck: *- Modul übernimmt Daten aus internem Empfangspuffer und fügt sie am Ende des internen Empfangsarrays an* *- Modul quittiert; aktuell empfangener Wert des OutputSequenceCounters auf OutputSequenceAck übertragen* *- Modul fragt Status wieder zyklisch ab*
3) Abschluss (Bestätigung): - Steuerung muss OutputSequenceAck zyklisch überprüfen → Eine Sequenz gilt erst dann als erfolgreich übertragen, wenn sie über das OutputSequenceAck bestätigt wurde. Um Übertragungsfehler auch bei der letzten Sequenz zu erkennen, muss sichergestellt werden, dass der Algorithmus lange genug durchlaufen wird. **Hinweis:** Für eine exakte Überwachung der Kommunikationszeiten sollten die Taskzyklen gezählt werden, die seit der letzten Erhöhung des OutputSequenceCounters vergangen sind. Auf diese Weise kann die Anzahl der Buszyklen abgeschätzt werden, die bislang zur Übertragung benötigt wurden. Übersteigt der Überwachungszähler eine vorgegebene Schwelle, kann die Sequenz als verloren betrachtet werden (das Verhältnis von Bus- und Taskzyklus kann vom Anwender beeinflusst werden, sodass der Schwellwert individuell zu ermitteln ist).

Algorithmus zum Empfangen

0) Zyklische Statusabfrage: - Steuerung muss InputSequenceCounter überwachen
Zyklische Prüfungen: *- Modul prüft InputSyncAck* *- Modul prüft InputMTU auf Freigabe* *→ Freigabekriterium: InputSequenceCounter > InputSequenceAck + Forward*
Vorbereitung: *- Modul bildet Controlbytes/Segmente und legt Sendearray an*
Aktion: *- Modul überträgt aktuellen Teil des Sendearrays in den Empfangspuffer* *- Modul erhöht InputSequenceCounter* *- Modul wartet auf neuen Buszyklus, nachdem Zeit aus ForwardDelay abgelaufen ist* *- Modul wiederholt Aktion, falls InputMTU freigegeben ist*
1) Empfangen (InputSequenceCounter > InputSequenceAck): - Steuerung muss Daten aus InputMTU übernehmen und an das Ende des Empfangsarrays anfügen - Steuerung muss InputSequenceAck an InputSequenceCounter der aktuell verarbeiteten Sequenz angleichen
Abschluss: *- Modul überwacht InputSequenceAck* *→ Eine Sequenz gilt erst dann als erfolgreich übertragen, wenn sie über das InputSequenceAck bestätigt wurde.*

Details/Hintergründe

1. SequenceCounter unzulässig groß (Zählerversatz)
   Fehlersituation: MTU nicht freigegeben
   Wenn beim Senden der Unterschied zwischen SequenceCounter und SequenceAck größer wird, als es erlaubt ist, liegt ein Übertragungsfehler vor. In diesem Fall müssen alle unbestätigten Sequenzen mit dem alten Wert des SequenceCounters wiederholt werden.
2. Prüfung einer Bestätigung
   Nach dem Empfang einer Bestätigung muss geprüft werden, ob die bestätigte Sequenz abgesendet wurde und bisher unbestätigt war. Falls eine Sequenz mehrfach bestätigt wird, liegt ein schwerwiegender Fehler vor. Der Kanal muss geschlossen und resynchronisiert werden (gleiches Verhalten wie ohne Forward).

   > **Information:**
   >
   > **In Ausnahmefällen kann das Modul bei der Verwendung des Forward den OutputSequenceAck um mehr als 1 erhöhen.**
   > **In diesem Fall liegt kein Fehler vor. Die Steuerung darf alle Sequenzen bis zur Bestätigten als erfolgreich übertragen betrachten.**

3. Sende- und Empfangsarrays
   Der Forward beeinflusst die Struktur des Sende- und Empfangsarrays nicht. Sie werden auf dieselbe Weise gebildet bzw. müssen auf dieselbe Weise ausgewertet werden.

4.1.5.4 Fehlerfall bei Verwendung des Forward

Im industriellen Umfeld werden in der Regel viele verschiedene Geräte unterschiedlicher Hersteller nebeneinander genutzt. Technische Geräte können sich gegenseitig durch ungewollte elektrische oder elektromagnetische Effekte störend beeinflussen. Unter Laborbedingungen können diese Situationen nur bis zu einem bestimmten Punkt nachempfunden und abgesichert werden.

Für die Übertragung per X2X Link wurden Vorkehrungen getroffen, falls es zu derartigen Beeinflussungen kommen sollte. Tritt beim Datentransfer z. B. eine unzulässige Prüfsumme auf, ignoriert das I/O-System die Daten dieses Buszyklus und der Empfänger erhält die letzten gültigen Daten erneut. Bei den herkömmlichen (zyklischen) Datenpunkten kann dieser Fehler oft ignoriert werden. Im darauffolgenden Zyklus wird der gleiche Datenpunkt wieder abgerufen, angepasst und übertragen.

Bei der Flatstream-Kommunikation mit aktiviertem Forward ist die Situation komplexer. Auch hier erhält der Empfänger ein weiteres mal die alten Daten, das heißt, die vorherigen Werte für SequenceAck/SequenceCounter und die alte MTU.

Ausfall einer Bestätigung (SequenceAck)

Wenn durch den Ausfall ein SequenceAck-Wert verloren geht, wurde die MTU bereits korrekt übertragen. Aus diesem Grund darf die nächste Sequenz vom Empfänger weiterverarbeitet werden. Der SequenceAck wird wieder an den mitgelieferten SequenceCounter angepasst und zum Absender zurückgeschickt. Für die Prüfung der eingehenden Bestätigungen folgt daraus, dass alle Sequenzen bis zur zuletzt Bestätigten erfolgreich übertragen sind (siehe Bild Sequenz 1, 2).

Ausfall einer Sendung (SequenceCounter, MTU)

Wenn durch den Ausfall eines Buszyklus der SequenceCounter-Wert bzw. die befüllte MTU verloren geht, kommen beim Empfänger keine Daten an. Zu diesem Zeitpunkt wirkt sich der Fehler noch nicht auf die Routine zum Absenden aus. Die zeitgesteuerte MTU wird wieder freigegeben und kann neu beschrieben werden.
Der Empfänger erhält SequenceCounter-Werte, die mehrfach inkrementiert sind. Damit das Empfangsarray korrekt zusammengestellt wird, darf der Empfänger nur Sendungen verarbeiten, die einen um eins erhöhten SequenceCounter besitzen. Die eintreffenden Sequenzen müssen ignoriert werden, das heißt, der Empfänger stoppt und gibt keine neuen Bestätigungen zurück.
Wenn die maximale Anzahl an unbestätigten Sequenzen abgesendet wurde und keine Bestätigungen zurück kommen, muss der Sender die betroffenen SequenceCounter und die dazugehörigen MTUs wiederholen (siehe Bild Sequenzen 3 und 4).

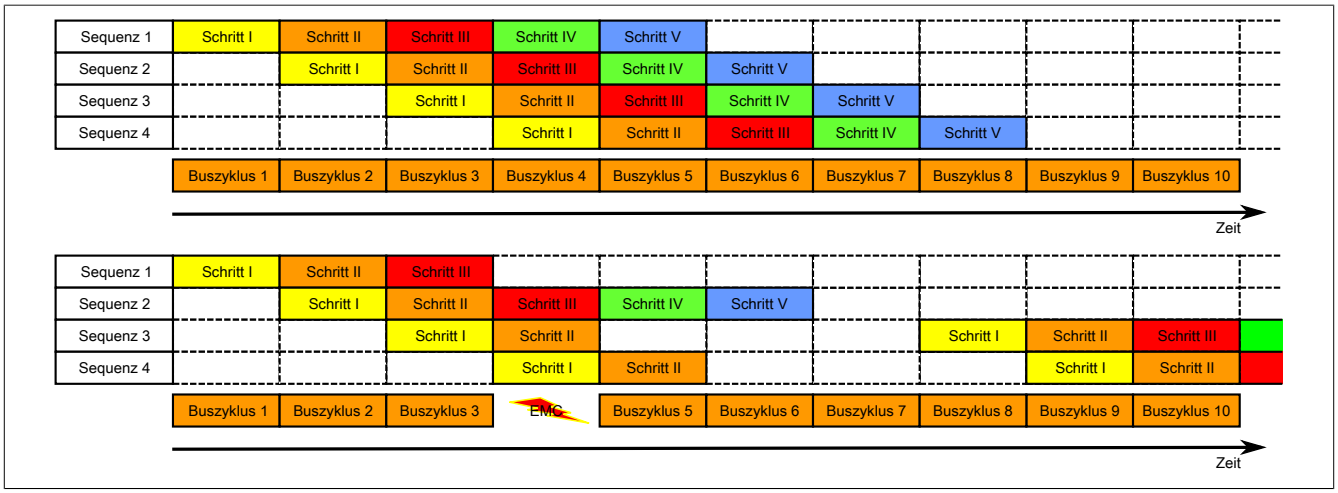


Abbildung 18: Auswirkung eines ausgefallenen Buszyklus

Ausfall der Bestätigung

Bei Sequenz 1 ging aufgrund der Störung die Bestätigung verloren. Im Schritt V der Sequenz 2 werden deshalb die Sequenzen 1 und 2 bestätigt.

Ausfall einer Sendung

Bei Sequenz 3 ging aufgrund der Störung die gesamte Sendung verloren. Der Empfänger stoppt und gibt keine Bestätigungen mehr zurück.
Der Sender sendet zunächst weiter, bis er die max. erlaubte Anzahl an unbestätigten Sendungen abgesetzt hat. Je nach Konfiguration beginnt er frühestens 5 Buszyklen später, die vergeblich abgesendeten Sendungen zu wiederholen.

4.2 NetTime Technology

Unter NetTime versteht man die Möglichkeit Systemzeiten zwischen einzelnen Komponenten der Steuerung bzw. Netzwerks (Steuerung, I/O-Module, X2X Link, POWERKLINK usw.) exakt aufeinander abzustimmen und zu übertragen.

Damit kann von Ereignissen der Zeitpunkt des Auftritts systemweit µs-genau bestimmt werden. Ebenso können anstehende Ereignisse exakt zu einem vorgegebenen Zeitpunkt ausgeführt werden.

NetTime TECHNOLOGY

4.2.1 Zeitinformationen

In der Steuerung bzw. im Netzwerk sind verschiedene Zeitinformationen vorhanden:

- Systemzeit (auf der SPS, APC usw.)
- X2X Link Zeit (für jedes X2X Link Netzwerk)
- POWERLINK-Zeit (für jedes POWERLINK-Netzwerk)
- Zeitdatenpunkte von I/O-Modulen

Die NetTime basiert auf 32 Bit Zähler, welche im µs-Takt erhöht werden. Das Vorzeichen der Zeitinformation wechselt nach 35 min 47 s 483 ms 648 µs und zu einem Überlauf kommt es nach 71 min 34 s 967 ms 296 µs.

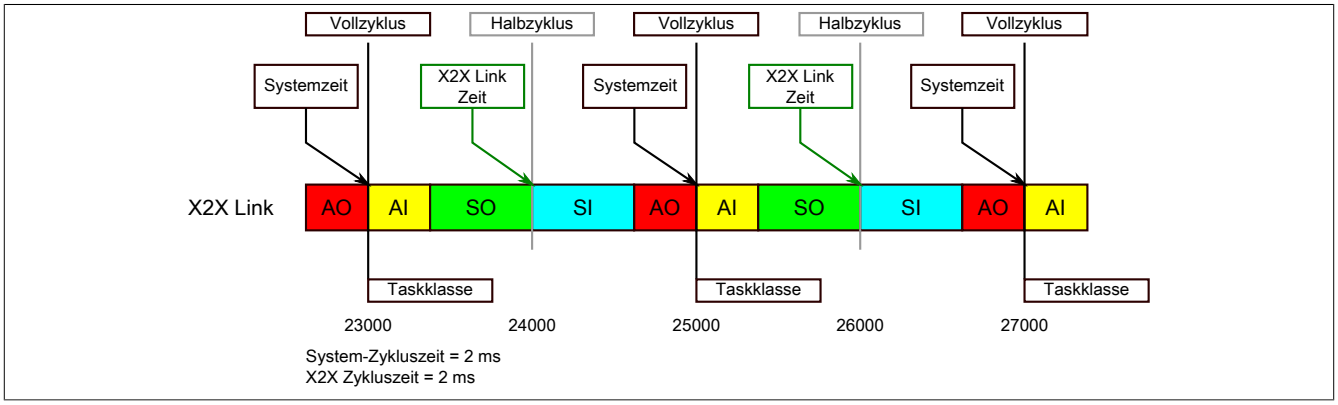
Die Initialisierung der Zeiten erfolgt auf Basis der Systemzeit während des Hochlaufs des X2X Links, der I/O-Module bzw. der POWERLINK-Schnittstelle.

Aktuelle Zeitinformationen in der Applikation können auch über die Bibliothek AsIOTime ermittelt werden.

4.2.1.1 Steuerungs/Controller-Datenpunkte

Die NetTime I/O-Datenpunkte der Steuerung oder des Controllers werden zu jedem Systemtakt gelatcht und zur Verfügung gestellt.

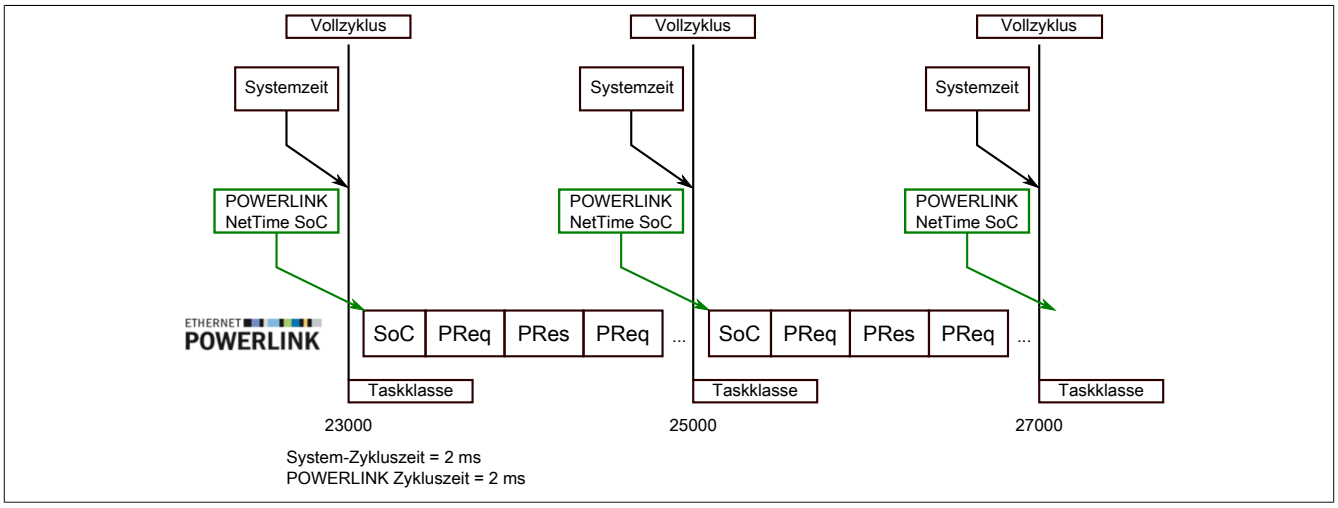
4.2.1.2 Referenzzeitpunkt X2X Link



Der Referenzzeitpunkt am X2X Link wird grundsätzlich zum Halbzyklus des X2X Link Zyklus gebildet. Dadurch ergibt sich beim Auslesen des Referenzzeitpunktes eine Differenz zwischen Systemzeit und X2X Link Referenzzeit.

Im Beispiel oben bedeutet dies einen Unterschied von 1 ms, das heißt, wenn zum Zeitpunkt 25000 im Task die Systemzeit und die X2X Link Referenzzeit miteinander verglichen werden, dann liefert die Systemzeit den Wert 25000 und die X2X Link Referenzzeit den Wert 24000.

4.2.1.3 Referenzzeitpunkt POWERLINK

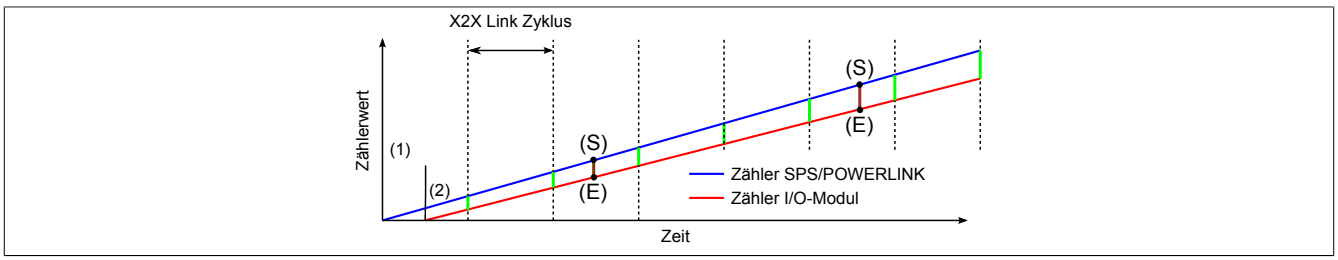


Der Referenzzeitpunkt am POWERLINK wird grundsätzlich beim SoC (Start of Cycle) des POWERLINK-Netzwerks gebildet. Der SoC startet systembedingt 20 µs nach dem Systemtakt. Dadurch ergibt sich folgende Differenz zwischen Systemzeit und POWERLINK-Referenzzeit:

POWERLINK-Referenzzeit = Systemzeit - POWERLINK-Zykluszeit + 20 µs.

Im Beispiel oben bedeutet dies einen Unterschied von 1980 µs, das heißt, wenn zum Zeitpunkt 25000 im Task die Systemzeit und die POWERINK-Referenzzeit miteinander betrachtet werden, dann liefert die Systemzeit den Wert 25000 und die POWERLINK-Referenzzeit den Wert 23020.

4.2.1.4 Synchronisierung von Systemzeit/POWERLINK-Zeit und I/O-Modul



Beim Hochfahren starten die internen Zähler für die Steuerung/POWERLINK (1) und dem I/O-Modul (2) zu unterschiedlichen Zeiten und erhöhen die Werte im µs-Takt.

Am Beginn jedes X2X Link Zyklus wird von der Steuerung bzw. vom POWERLINK-Netzwerk eine Zeitinformation an das I/O-Modul gesendet. Das I/O-Modul vergleicht diese Zeitinformation mit der modulinternen Zeit und bildet eine Differenz (grüne Linie) zwischen beiden Zeiten und speichert diese ab.

Bei Auftreten eines NetTime-Ereignisses (E) wird die modulinterne Zeit ausgelesen und mit dem gespeicherten Differenzwert korrigiert (braune Linie). Dadurch kann auch bei nicht absolut gleichlaufenden Zählern immer der exakte Systemzeitpunkt (S) eines Ereignisses ermittelt werden.

Anmerkung

Die Taktungenauigkeit ist im Bild als rote Linie stark überhöht dargestellt.

4.2.2 Zeitstempelfunktionen

NetTime-fähige Module stellen je nach Funktionsumfang verschiedene Zeitstempelfunktionen zur Verfügung. Tritt ein Zeitstempelereignis auf, so speichert das Modul unmittelbar die aktuelle NetTime. Nach der Übertragung der jeweiligen Daten inklusive dieses exakten Zeitpunkts an die Steuerung kann diese nun, gegebenenfalls mit Hilfe ihrer eigenen NetTime (bzw. Systemzeit), die Daten auswerten.

4.2.2.1 Zeitbasierte Eingänge

Über die NetTime Technology kann der exakte Zeitpunkt einer steigenden Flanke an einem Eingang ermittelt werden. Ebenso kann auch die steigende sowie fallende Flanke erkannt und daraus die Zeitdauer zwischen 2 Ereignissen ermittelt werden.

Information:

Der ermittelte Zeitpunkt liegt immer in der Vergangenheit.

4.2.2.2 Zeitbasierte Ausgänge

Über die NetTime Technology kann der exakte Zeitpunkt einer steigenden Flanke an einem Ausgang vorgegeben werden. Ebenso kann auch die steigende sowie fallende Flanke vorgegeben und daraus ein Pulsmuster generiert werden.

Information:

Die vorgegebene Zeit muss immer in der Zukunft liegen und die eingestellte X2X Link Zykluszeit für die Definition des Zeitpunkts berücksichtigt werden.

4.2.2.3 Zeitbasierte Messungen

Über die NetTime Technology kann der exakte Zeitpunkt einer stattgefundenen Messung ermittelt werden. Es kann dabei sowohl der Anfangs- und/oder der Endzeitpunkt der Messung übermittelt werden.

5 Inbetriebnahme

5.1 IO-Link Device konfigurieren

Um ein IO-Link Device zu konfigurieren, gibt es folgende Möglichkeiten:

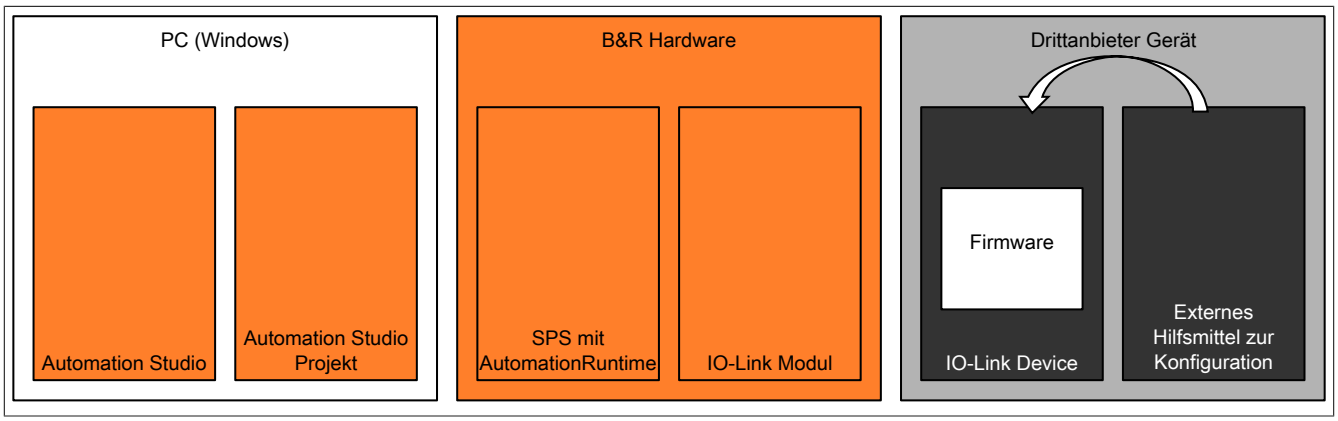
- Direkte Konfiguration
- Konfiguration per IODD/DTM-Unterstützung.
  Dafür muss vom Hersteller eine entsprechende IODD- bzw. DTM-Datei bereitgestellt werden.
- Wiederherstellung einer Konfiguration mittels Parameterserver.
  Dafür muss das IO-Link Device die Funktion "Parameterserver" entsprechend der IO-Link Spezifikation Version 1.1 unterstützten.

Information:

Eine weitere Möglichkeit zur Parametrierung des IO-Link Device bietet die Bibliothek "AsIoLink". Diese Bibliothek ist nicht Bestandteil dieser Beschreibung.

5.1.1 Direkte Konfiguration

Die direkte Konfiguration erfolgt unabhängig von der verwendeten B&R Hard- und Software. Die Eingabe der Parameter kann z. B. über ein zusätzliches Konfigurationsgerät, ein integriertes Display oder weitere Bedienelemente am IO-Link Device geschehen.



Vorteil

Für einzelne Devices vorteilhaft, da die Inbetriebnahme des IO-Link Devices mit den Hilfsmitteln des Herstellers möglich ist.

Falls Probleme bei der Konfiguration des IO-Link Devices auftreten, muss nicht geprüft werden, welche Softwarekomponente die Fehlfunktion verursacht.

Nachteil

Jedes IO-Link Device muss einzeln manuell vorkonfiguriert werden.

Der Anwender muss unter Umständen mehrere Entwicklungsumgebungen auf seinem Computer verwenden.

5.1.2 IODD/DTM Unterstützung

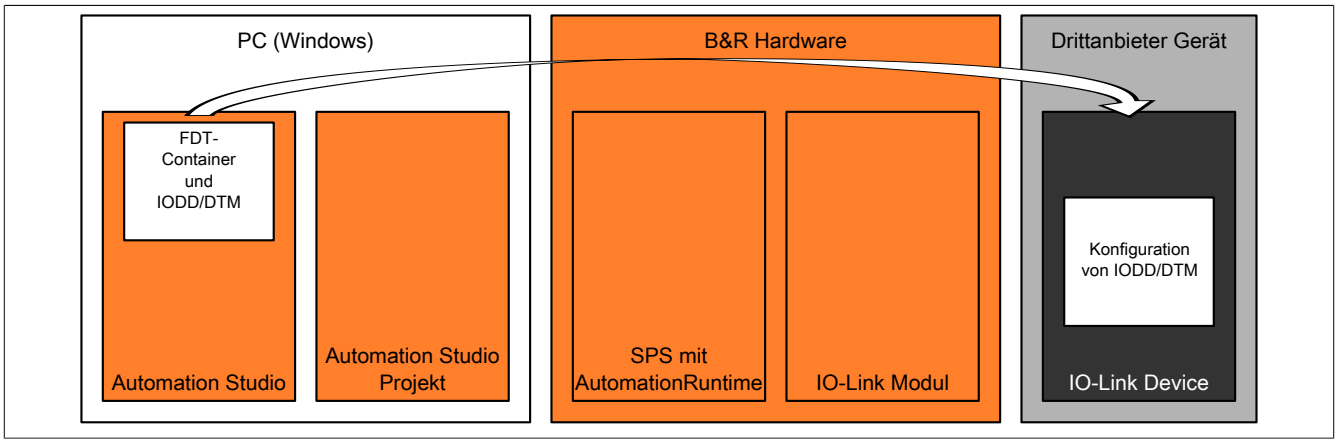
Mit Hilfe des integrierten FDT-Containers können IO-Link Devices mit Hilfe des Automation Studios konfiguriert werden. Die IODD/DTM-Unterstützung für IO-Link Devices kann dabei sowohl online als auch offline geschehen.

Information:

Für die Verwendung des Automation Studios zur Konfiguration von IO-Link Devices muss eine entsprechende Hardwarebeschreibungsdatei (IODD oder DTM) heruntergeladen und installiert werden.

5.1.2.1 IODD/DTM (online)

Bei der Online-Konfiguration kommuniziert der FDT-Container des Automation Studios direkt mit dem IO-Link Device. Nach Aufbau der Verbindung können die Konfigurationsparameter wie gewünscht angepasst werden.



Vorteil

Für die Konfiguration des IO-Link Devices sind in der Regel keine zusätzlichen Geräte notwendig. Alle Einstellungen können vom Anwender in einer einzigen Entwicklungsumgebung vorgenommen werden.

Nachteil

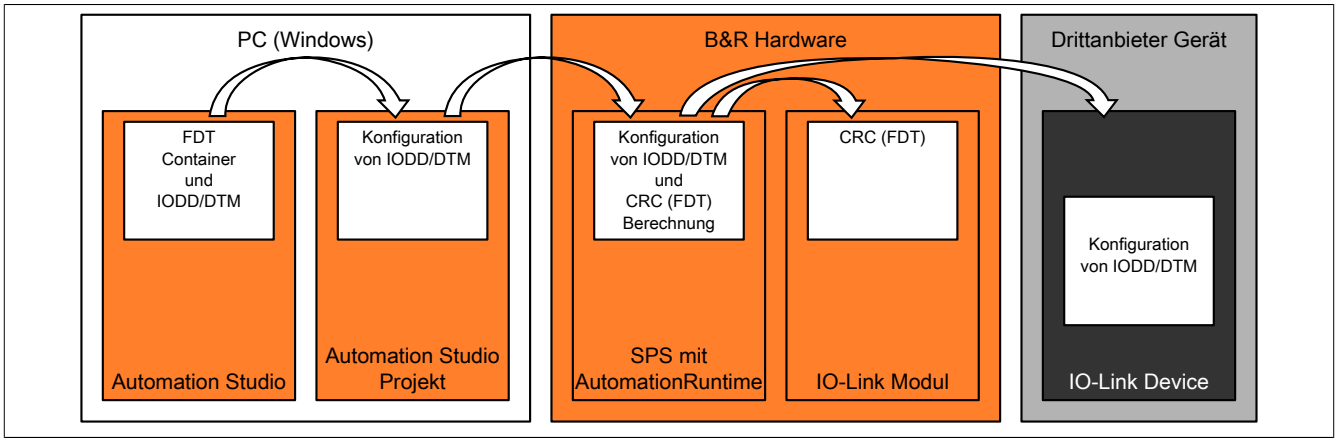
Jedes IO-Link Device muss einzeln konfiguriert werden.

5.1.2.2 IODD/DTM (offline)

Bei der Offline-Konfiguration wird der Parametersatz, der über die IODD- bzw. DTM-Datei eingegeben werden kann, im Automation Studio Projekt hinterlegt. Beim Download wird der Parametersatz für das IO-Link Device auf die Steuerung übertragen und von dort aus über das Modul in das IO-Link Device eingespielt.

Ablauf

1) Beim Start des IO-Link Moduls wird die Checksumme (CRC_{FDT}) für den aktuellen Parametersatz berechnet.
2) Wenn die zuvor hinterlegte Checksumme von der aktuell Berechneten abweicht, wird der Parametersatz an das IO-Link Device übertragen.
3) Nach der Übertragung des Parametersatzes, wird die dazugehörige Checksumme (CRC_{FDT}) am IO-Link Modul gespeichert und kann für zukünftige Vergleiche verwendet werden.
4) Ändert sich der Parametersatz, ergibt sich beim darauffolgenden Neustart der Steuerung eine neue Checksumme (CRC_{FDT}) und die Schritt 2 und 3 wiederholen sich



Vorteil

Die Konfigurationsparameter des IO-Link Devices werden als Teil des Automation Studio Projektes abgelegt. Der Anwender kann mit einer Entwicklungsumgebung arbeiten und alle Einsteillungen festlegen.

Bei Serienmaschinen müssen die später verwendeten IO-Link Devices nicht einzeln vorkonfiguriert werden.

Nachteil

Die Konfigurationsmöglichkeiten für das IO-Link Device hängen vom Umfang der IODD- bzw. DTM-Datei ab.

Information:

Bevor die Übertragung des Parametersatzes an das IO-Link Device ausgeführt wird, prüft die Steuerung, ob das angeschlossene Device die korrekte DeviceID aufweist. Stimmt die DeviceID nicht, wird der Vorgang abgebrochen. Der Parametersatz wird nicht übertragen und die Checksumme nicht gespeichert.

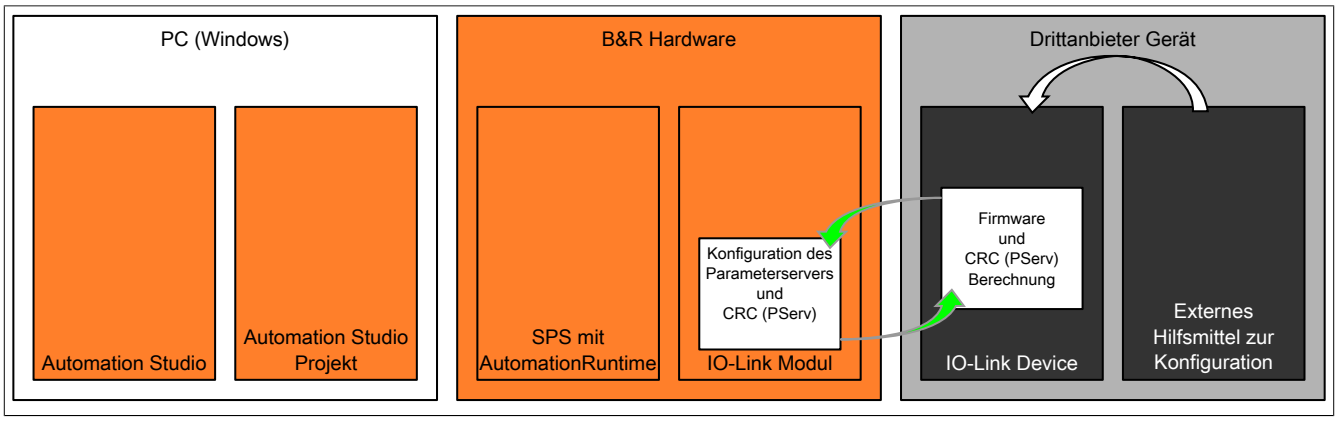
5.1.3 Parameterserver

In der IO-Link Spezifikation wird ab der Version 1.1 die Funktion "Parameterserver" definiert. Durch diese Funktion wird der Austausch eines IO-Link Devices ermöglicht, ohne spezielle Kenntnisse vom Wartungspersonal zu benötigen.

Zu diesem Zweck wird die am IO-Link Device eingespielte Konfiguration am IO-Link Modul hinterlegt. Zusätzlich wird eine Checksumme (CRC_{PServ}) berechnet, um einen einfachen Vergleich der Parametersets zu ermöglichen.

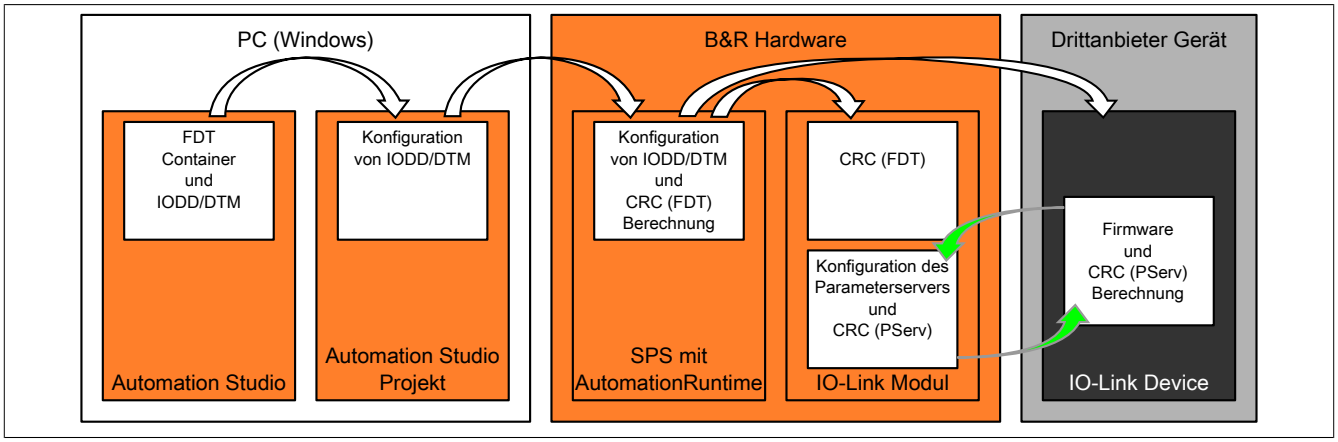
Ablauf

1) Wenn das IO-Link Device die Funktion "Parameterserver" unterstützt, berechnet es beim Hochlauf die Checksumme (CRC_{PServ}) für seinen aktuellen Parametersatz.
2) Wenn die aktuell berechnete Checksumme (CRC_{PServ}) von der zuvor am IO-Link Modul hinterlegten abweicht, unterscheidet sich das Parameterset des IO-Link Devices vom aktuell am Modul hinterlegten.
3) Um zu entscheiden, ob das Parameterset vom Device herauf- oder vom IO-Link Modul heruntergeladen werden muss, werden die Werte der DeviceID und die Seriennummer des IO-Link Devices ausgewertet.
   a) Falls sich die DeviceID geändert hat, wurde ein anderer Devicetyp erkannt. In diesem Fall muss der Parametersatz des IO-Link Devices ausgelesen und am IO-Link Modul gespeichert werden. Außerdem wird die aktuelle Checksumme (CRC_{PServ}) am IO-Link Modul hinterlegt.
   b) Falls die DeviceID unverändert ist aber sich die Seriennummer geändert hat, wird ein Austausch des IO-Link Devices gegen ein Device desselben Typs angenommen.
   In diesem Fall wird der Parametersatz, der im IO-Link Modul hinterlegt ist, auf das IO-Link Device heruntergeladen.
   c) Falls die DeviceID und Seriennummer unverändert sind, wird davon ausgegangen, dass das IO-Link Device eine neue Konfiguration eingespielt bekommen hat. In diesem Fall wird der neue Parametersatz des IO-Link Devices ausgelesen und am IO-Link Modul gespeichert. Außerdem wird die aktuelle Checksumme (CRC_{PServ}) am IO-Link Modul hinterlegt.



5.1.4 Gemeinsame Verwendung von IODD/DTM und Parameterserver

Die IODD/DTM-Unterstützung und der Parameterserver können gemeinsam verwendet werden. Die beiden Funktionen arbeiten unabhängig voneinander, beeinflussen sich aber gegenseitig.



5.1.4.1 Änderung der Konfiguration mittels IODD/DTM-Unterstützung

Wenn das IO-Link Device mittels FDT-Container (IODD/DTM) umkonfiguriert wird, berechnet das IO-Link Device anschließend die neue Checksumme (CRC_{PServ}). Danach werden die geänderten Daten vom Parameterserver des IO-Link Moduls zurückgelesen.

5.1.4.2 Austausch des IO-Link Devices

Wenn das IO-Link Device getauscht wird, prüft das System nur die Checksumme (CRC_{PServ}). Der Parametersatz des FDT-Containers bleibt unbeachtet, weil die Checksumme (CRC_{FDT}) im Projekt auf der Steuerung noch immer mit der hinterlegt Checksumme (CRC_{FDT}) am IO-Link Modul übereinstimmt (Für den Ablauf siehe "Parameterserver" auf Seite 37).

6 Registerbeschreibung

6.1 Systemvoraussetzungen

Um generell alle Funktionen verwenden zu können, werden folgende Mindestversionen benötigt:

- Automation Studio 4.7
- Automation Runtime C4.73 oder höher
- B&R DTM-Bibliothek 1.6.1.1

6.2 Funktionsmodell 0 - Standard

Register	Name	Datentyp	Lesen		Schreiben	
			Zyklisch	Azyklisch	Zyklisch	Azyklisch
Modulkonfiguration						
3073 + N * 1024	CfO_OperatingMode0N (Index N = 1 bis 4)	USINT				•
IO-Link Konfiguration						
3076 + N * 1024	CfO_ChannelMode0N (Index N = 1 bis 4)	UDINT				•
3099 + N * 1024	CfO_IdentificationRevisionId0N (Index N = 1 bis 4)	USINT				•
3102 + N * 1024	CfO_IdentificationVendorId0N (Index N = 1 bis 4)	UINT				•
3108 + N * 1024	CfO_IdentificationDeviceId0N (Index N = 1 bis 4)	UDINT				•
3116 + N * 1024	CfO_PDI_TypeInfo0N (Index N = 1 bis 4)	UDINT				•
3124 + N * 1024	CfO_PDO_TypeInfo0N (Index N = 1 bis 4)	UDINT				•
15372	CfO_TimerCycle	UDINT				•
15366	CfO_TimerOffset	INT				•
3086 + N * 1024	CfO_ReqCycleMultiple0N (Index N = 1 bis 4)	UINT				•
3090 + N * 1024	CfO_ReqCycleDivisor0N (Index N = 1 bis 4)	UINT				•
3094 + N * 1024	CfO_ReqCycleOffset0N (Index N = 1 bis 4)	UINT				•
3082 + N * 1024	CfO_ReqCycleTime0N (Index N = 1 bis 4)	UINT				•
IO-Link Kommunikation						
4473 + N * 8	OutputData01_N (Index N = 1 bis 8)	(U)SINT			•	
4474 + N * 8	OutputData01_N (Index N = 1 bis 8)	(U)SINT				
4476 + N * 8	OutputData01_N (Index N = 1 bis 8)	(U)DINT REAL				
5497 + N * 8	OutputData02_N (Index N = 1 bis 8)	(U)SINT			•	
5498 + N * 8	OutputData02_N (Index N = 1 bis 8)	(U)SINT				
5500 + N * 8	OutputData02_N (Index N = 1 bis 8)	(U)DINT REAL				
6521 + N * 8	OutputData03_N (Index N = 1 bis 8)	(U)SINT			•	
6522 + N * 8	OutputData03_N (Index N = 1 bis 8)	(U)SINT				
6524 + N * 8	OutputData03_N (Index N = 1 bis 8)	(U)DINT REAL				
7545 + N * 8	OutputData04_N (Index N = 1 bis 8)	(U)SINT			•	
7546 + N * 8	OutputData04_N (Index N = 1 bis 8)	(U)SINT				
7548 + N * 8	OutputData04_N (Index N = 1 bis 8)	(U)DINT REAL				
7	SIO oder digitale Ausgänge	USINT			•	
	DigitalOutput01	Bit 0				
	...	...				
	DigitalOutput04	Bit 3				
	DisablePowerSupply01	Bit 4				
	...	...				
	DisablePowerSupply04	Bit 7				
4345 + N * 8	InputData01_N (Index N = 1 bis 8)	(U)SINT	•			
4346 + N * 8	InputData01_N (Index N = 1 bis 8)	(U)INT				
4348 + N * 8	InputData01_N (Index N = 1 bis 8)	(U)DINT REAL				
5369 + N * 8	InputData02_N (Index N = 1 bis 8)	(U)SINT	•			
5370 + N * 8	InputData02_N (Index N = 1 bis 8)	(U)INT				
5372 + N * 8	InputData02_N (Index N = 1 bis 8)	(U)DINT REAL				
6393 + N * 8	InputData03_N (Index N = 1 bis 8)	(U)SINT	•			
6394 + N * 8	InputData03_N (Index N = 1 bis 8)	(U)INT				
6396 + N * 8	InputData03_N (Index N = 1 bis 8)	(U)DINT REAL				
7417 + N * 8	InputData04_N (Index N = 1 bis 8)	(U)SINT	•			
7418 + N * 8	InputData04_N (Index N = 1 bis 8)	(U)INT				
7420 + N * 8	InputData04_N (Index N = 1 bis 8)	(U)DINT REAL				
1	SIO oder digitale Eingänge	USINT	•			
	DigitalInput01	Bit 0				
	...	...				
	DigitalInput04	Bit 3				

Register	Name	Datentyp	Lesen		Schreiben	
			Zyklisch	Azyklisch	Zyklisch	Azyklisch
IO-Link Statusrückmeldung						
3	Sync (Statusbyte)	USINT	•			
	Synchronized01	Bit 0				
	...	...				
	Synchronized04	Bit 3				
	CycleEnd01	Bit 4				
	...	...				
	CycleEnd04	Bit 7				
5	Überlast (Statusbyte)	USINT	•			
	Overload01	Bit 0				
	...	...				
	Overload04	Bit 3				
17 + N * 16	ChannelStatus0N (Index N = 1 bis 4)	USINT	•			
22 + N * 16	FrameCount0N (Index N = 1 bis 4)	SINT	•			
3586 + N * 1024	CycleStartNettime0N (Index N = 1 bis 4)	INT	•			
3588 + N * 1024	CycleStartNettime0N (Index N = 1 bis 4)	DINT				
3594 + N * 1024	CycleEndNettime0N (Index N = 1 bis 4)	INT	•			
3596 + N * 1024	CycleEndNettime0N (Index N = 1 bis 4)	DINT				
IO-Link Event-Schnittstelle						
113	EventPortSeq	USINT	•	•		
115	EventQualifier	USINT	•	•		
118	EventCode	UINT	•	•		
121	EventsLeft	USINT		•		
123	EventQuit	USINT			•	•
123	EventQuitReadBack	USINT		•		
IO-Link Parameterserver						
19 + N * 16	DsControl0N (Index N = 1 bis 4)	USINT			•	•
3140 + N * 1024	Cfo_DS_Config0N (Index N = 1 bis 4)	UDINT				•
3241 + N * 1024	DsProgress0N (Index N = 1 bis 4)	USINT		•		
3148 + N * 1024	CfO_DS_SaveCtrl0N (Index N = 1 bis 4)	UDINT				•
3156 + N * 1024	CfO_DS_SaveData0N (Index N = 1 bis 4)	UDINT				•
IO-Link Zeitstempel						
3610 + N * 1024	IoLinkTimestampIn0N (Index N = 1 bis 4)	INT	•			
3612 + N * 1024	IoLinkTimestampIn0N (Index N = 1 bis 4)	DINT				
3617 + N * 1024	IoLinkTimestampInStatusSeq0N (Index N = 1 bis 4)	USINT	•			
3714 + N * 1024	IoLinkTimestampOut0N (Index N = 1 bis 4)	INT			•	
3716 + N * 1024	IoLinkTimestampOut0N (Index N = 1 bis 4)	DINT				
3721 + N * 1024	IoLinkTimestampOutCtrlSeq0N (Index N = 1 bis 4)	USINT			•	
3619 + N * 1024	IoLinkTimestampOutStatus0N (Index N = 1 bis 4)	USINT	•			
IO-Link Device-IDs						
3202 + N * 1024	VendorId0N (Index N = 1 bis 4)	UINT	•	•		
3212 + N * 1024	DeviceId0N (Index N = 1 bis 4)	UDINT	•	•		
3206 + N * 1024	FunctionId0N (Index N = 1 bis 4)	UINT	•	•		
3218 + N * 1024	CycleTime0N (Index N = 1 bis 4)	UINT	•	•		
3222 + N * 1024	CycleMultible0N (Index N = 1 bis 4)	UINT		•		
3226 + N * 1024	CycleDivisor0N (Index N = 1 bis 4)	UINT		•		
3230 + N * 1024	MinCycleTime0N (Index N = 1 bis 4)	UINT		•		
3233 + N * 1024	PDI_Size0N (Index N = 1 bis 4)	USINT		•		
3235 + N * 1024	PDO_Size0N (Index N = 1 bis 4)	USINT		•		
3237 + N * 1024	Baudrate0N (Index N = 1 bis 4)	USINT		•		
3239 + N * 1024	IoLinkVersionID0N (Index N = 1 bis 4)	USINT		•		
Statistik-Zählregister						
3266 + N * 1024	RetryCnt0N (Index N = 1 bis 4)	UINT		•		
3270 + N * 1024	SpiErrorCnt0N (Index N = 1 bis 4)	UINT		•		
3274 + N * 1024	TransmErrCnt0N (Index N = 1 bis 4)	UINT		•		
3278 + N * 1024	ParityErrCnt0N (Index N = 1 bis 4)	UINT		•		
3282 + N * 1024	FrameErrCnt0N (Index N = 1 bis 4)	UINT		•		
3286 + N * 1024	RxSizeErrCnt0N (Index N = 1 bis 4)	UINT		•		
3290 + N * 1024	RxChksmErrCnt0N (Index N = 1 bis 4)	UINT		•		
3294 + N * 1024	DeviceDlyErrCnt0N (Index N = 1 bis 4) [1)]	UINT		•		
3298 + N * 1024	CycleTimeErrorCnt0N (Index N = 1 bis 4) [1)]	UINT		•		
Kommando-Schnittstelle						
98	ParameterIndexOut	UINT			•	•
101	ParameterSubIndexOut	USINT			•	•
103	ParameterCtrlOut	USINT			•	•
108	ParameterDataOut_0	UDINT			•	•
103	ParameterCtrlIn	USINT	•	•		
108	ParameterDataIn_0	UDINT	•	•		
Flatstream						
193	CfO_OutputMTU	USINT				•
195	CfO_InputMTU	USINT				•
197	CfO_FlatStreamMode	USINT				•
199	CfO_Forward	USINT				•

Register	Name	Datentyp	Lesen		Schreiben	
			Zyklisch	Azyklisch	Zyklisch	Azyklisch
204	CfO_ForwardDelay	UDINT				•
129	InputSequence	USINT	•			
129 + N * 2	RxByteN (Index N = 1 bis 27)	USINT	•			
129	OutputSequence	USINT			•	
129 + N * 2	TxByteN (Index N = 1 bis 27)	USINT			•	

1) Erst ab Firmware-Version 261

6.3 Modulkonfiguration

Das Moduls kann genutzt werden, um IO-Link Devices (IO-Link Version 1.1) anzuschließen. Die Spannungsversorgung für den IO-Link Sensor/Aktor darf vom Modul bezogen werden. Um Schäden an der Hardware zu vermeiden, ist das Modul mit einem Überlastschutz ausgerüstet.

Information:

Je Anschluss-Pin der Sensor/Aktorversorgung ist ein separater Überlastschutz implementiert; d. h. die Versorgung jedes angeschlossenen IO-Link Devices erfolgt zueinander galvanisch getrennt.

Sollte ein IO-Link Device den Überlastschutz am Modul auslösen, wirkt sich die anschließende Begrenzung des Versorgungstroms nur auf dieses Device aus. Alle weiteren angeschlossene Devices werden nicht beeinflusst.

6.3.1 OperatingMode

Name:
CfO_OperatingMode01 bis CfO_OperatingMode04

Dieses Register ist identisch mit dem ersten Byte des Registers "ChannelMode" auf Seite 42 aus der IO-Link Konfiguration. Es enthält alle Einstellungen eines IO-Link Kanals, die während der Laufzeit verändert werden dürfen.

Datentyp	Werte	Bus Controller Default
USINT	Siehe Bitstruktur	0

Bitstruktur:

Bit	Beschreibung	Werte	Information
0 - 1	Modus des Kanals	00	Modus: Inaktiv (Bus Controller Default)
		01	Modus: SIO-Ausgang Der C/Q-Anschluss des Kanals ist als digitaler Ausgang konfiguriert.
		10	Modus: SIO-Eingang Der C/Q-Anschluss des Kanals ist als digitaler Eingang konfiguriert.
		11	Modus: Operate Der C/Q-Anschluss des Kanals ist für die IO-Link Datenübertragung konfiguriert.
2 - 7	Reserviert	-	

6.4 IO-Link Konfiguration

Das Modul baut die Kommunikation zum IO-Link Device auf, wenn das Register "ChannelMode" auf Seite 42 des entsprechenden Kanals konfiguriert ist. Die anderen Register in diesem Abschnitt können genutzt werden, um den Datenaustausch an die Bedürfnisse der Applikation anzupassen.

6.4.1 ChannelMode

Name:
CfO_ChannelMode01 bis CfO_ChannelMode04

Über dieses Register erhält der Anwender die Möglichkeit alle kanalspezifischen Einstellungen einzustellen.

Datentyp	Werte	Bus Controller Default
UDINT	Siehe Bitstruktur	0

Bitstruktur:

Bit	Beschreibung	Werte	Information
0 - 1	Modus des Kanals	00	Modus: Inaktiv (Bus Controller Default)
		01	Modus: SIO-Ausgang Der C/Q-Anschluss des Kanals ist als digitaler Ausgang konfiguriert.
		10	Modus: SIO-Eingang Der C/Q-Anschluss des Kanals ist als digitaler Eingang konfiguriert.
		11	Modus: Operate Der C/Q-Anschluss des Kanals ist für die IO-Link Datenübertragung konfiguriert.
2 - 15	Reserviert	-	
16 - 17	Modus zur Synchronisierung	00	Freilaufend (asynchron) (Bus Controller Default)
		01	Synchron (manuell)
		10	Synchron (automatisch)
		11	Unzulässig
18 - 19	Reserviert	-	
20 - 23	Inspektionsebene	0	Prüfungen deaktiviert (Bus Controller Default)
		1	Prüfung von VendorID und DeviceID
24 - 25	IO-Link Zeitstempel	00	Kein Zeitstempel (Bus Controller Default)
		01	Eingangszeitstempel
		10	Ausgangszeitstempel
		11	Ein- und Ausgangszeitstempel
26	Format des IO-Link Ausgangszeitstempels [1)]	0	32 Bit (DINT) (Bus Controller Default)
		1	16 Bit (INT)
27 - 32	Reserviert	-	

1) Mit diesem Bit wird dem Modul mitgeteilt, in welchem Format der Ausgangszeitstempel IoLinkTimestampOut vorliegt. In Automation Studio wird diese Einstellung in der I/O-Konfiguration implizit zusammen mit der Auswahl des Datentyps für den IO-Link Zeitstempel durchgeführt.

6.4.2 IdentificationRevisionID

Name:
CfO_IdentificationRevisionId01 bis CfO_IdentificationRevisionId04

Wenn beim Hochfahren die Kennungen (IDs) des angeschlossenen Devices verifiziert werden sollen, kann in diesem Register die IO-Link Revision bekanntgegeben werden, mit der die Prüfung stattfindet.

Datentyp	Werte	Information
USINT	0	Es wird die vom Device gelesene Revision verwendet.
	16	Das angeschlossene Device wird gemäß Revision V1.0 geprüft.
	17	Das angeschlossene Device wird gemäß Revision V1.1 geprüft. **Sollte das Device diesen Standard nicht unterstützen, wird im Register "ChannelStatus" auf Seite 54 Fehlercode 41 ausgegeben.**

6.4.3 IdentificationVendorID

Name:
CfO_IdentificationVendorId01 bis CfO_IdentificationVendorId04

Wenn beim Hochfahren die VendorID verifiziert werden soll, muss in diesem Register die erwartete Hersteller-ID eingetragen werden. Die Überprüfung kann durch Setzen der Inspektionsebene im Register "ChannelMode" auf Seite 42 aktiviert werden.

Information:

Stimmt die erwartete ID nicht mit der tatsächlichen ID des angeschlossenen IO-Link Device überein, wird die Kommunikation für diesen Kanal nicht gestartet.

Datentyp	Werte	Information
UINT	0 bis 65.535	Bus Controller Default: 0

6.4.4 IdentificationDeviceID

Name:
CfO_IdentificationDeviceId01 bis CfO_IdentificationDeviceId04

Wenn beim Hochfahren die DeviceID verifiziert werden soll, muss in diesem Register die erwartete ID des IO-Link Device eingetragen werden. Die Überprüfung kann durch Setzen der Inspektionsebene im Register "ChannelMode" auf Seite 42 aktiviert werden.

Information:

Stimmt die erwartete ID nicht mit der tatsächlichen ID des angeschlossenen IO-Link Device überein, wird die Kommunikation für diesen Kanal nicht gestartet.

Datentyp	Werte	Information
UDINT	0 bis 4.294.967.295	Bus Controller Default: 0

6.4.5 PDI_TypeInfo

Name:CfO_PDI_TypeInfo01 bis CfO_PDI_TypeInfo04

Um die Prozessdaten aus dem IO-Link Device in die Steuerung (Applikation) zu übertragen, wird die Information zunächst vom Modul eingelesen und zwischengespeichert. Hierfür werden grundsätzlich 4 Byte je angemeldeter Information reserviert (siehe "IO-Link Kommunikation" auf Seite 50).

Mit diesem Register wird konfiguriert, wie der eingehende IO-Link Prozessdatenstrom (IO-Link Frame) aufgeteilt wird. Gemäß dieser Konfiguration werden die IO-Link Prozessdaten über die entsprechenden InputData-Register der Applikation zur Verfügung gestellt. In der I/O-Zuordnung werden die InputData-Register einzelnen Datenpunkten mit dem entsprechenden Datentyp zugeordnet.

Datentyp	Werte	Bus Controller Default
UDINT	Siehe Bitstruktur	0

Bitstruktur:

Bit	Beschreibung	Wert	Information
0 bis 3	IO-Link Information 1	0000	Array[4] von Bytes (Bus Controller Default)
		0001	USINT
		0010	SINT
		0011	UINT
		0100	INT
		0101	UDINT
		0110	DINT
		0111	REAL
		1000 - 1111	Reserviert
4 - 7	IO-Link Information 2		Mögliche Werte sind identisch mit IO-Link Information 1
8 - 11	IO-Link Information 3		
12 -15	IO-Link Information 4		
16 - 19	IO-Link Information 5		
20 - 23	IO-Link Information 6		
24 - 27	IO-Link Information 7		
28 - 31	IO-Link Information 8		

Information:

Bei Einstellung 0 (Array[4] von Bytes) werden die Bytes aus dem IO-Link Datenstrom unverändert kopiert. In allen anderen Modi wird die Byte-Reihenfolge geändert (von Big Endian auf Little Endian).

6.4.6 PDO_TypeInfo

Name:CfO_PDO_TypeInfo01 bis CfO_PDO_TypeInfo04

Um Prozessdaten zum IO-Link Device zu übertragen, wird mit diesem Register konfiguriert, welcher Datentyp der einzelnen "OutputData" auf Seite 51 Register genutzt werden, um den ausgehenden IO-Link Prozessdatenstrom (IO-Link Frame, siehe "IO-Link Kommunikation" auf Seite 50) zusammenzufügen. Entsprechend dieser Konfiguration werden OutputData-Register im Automation Studio den Datenpunkten mit dem entsprechenden Datentyp zugeordnet (I/O-Zuordnung).

Datentyp	Werte	Bus Controller Default
UDINT	Siehe Bitstruktur	0

Bitstruktur:

Bit	Beschreibung	Wert	Information
0 bis 3	IO-Link Information 1	0000	Array[4] von Bytes (Bus Controller Default)
		0001	USINT
		0010	SINT
		0011	UINT
		0100	INT
		0101	UDINT
		0110	DINT
		0111	REAL
		1000 - 1111	Reserviert
4 - 7	IO-Link Information 2		Mögliche Werte sind identisch mit IO-Link Information 1
8 - 11	IO-Link Information 3		
12 -15	IO-Link Information 4		
16 - 19	IO-Link Information 5		
20 - 23	IO-Link Information 6		
24 - 27	IO-Link Information 7		
28 - 31	IO-Link Information 8		

Information:

Bei Einstellung 0 (Array[4] von Bytes) werden die Bytes aus dem IO-Link Datenstrom unverändert kopiert. In allen anderen Modi wird die Byte-Reihenfolge geändert (von Big Endian auf Little Endian).

6.4.7 Zeitverhalten der IO-Link Kommunikation

Während der Laufzeit muss das Modul Datensätze aus 2 verschiedenen Kommunikationsstandards verwalten. Für eine effiziente Kommunikation am X2X Link muss sichergestellt sein, dass die Zykluszeit aller X2X Module der Buszykluszeit entspricht.

IO-Link spezifizierte Zykluszeiten

In der IO-Link Spezifikation wird festgelegt, dass die Abfrage eines IO-Link Device in bestimmten Zeitabständen erfolgen muss. Dieser Zyklus wird als IO-Link Zyklus bezeichnet.
Gültige IO-Link Zykluszeiten liegen im Bereich von 0,5 ms bis 132,8 ms. Dabei werden 3 verschiedene Bereiche unterschieden.

Bereich	Schrittweite	Berechnung	Gültige Zykluszeiten
0,5 bis 6,3 ms	0,1 ms	Zykluszeit = 0,1 ms * n + 0,4 ms	0,5; 0,6 bis 6,2; 6,3 ms
6,4 bis 32,6 ms	0,4 ms	Zykluszeit = 0,4 ms * n + 6,4 ms	6,4; 6,8; 7,2 bis 32,2; 32,6 ms
32,0 bis 132,8 ms	1,6 ms	Zykluszeit = 1,6 ms * n + 32,0 ms	32,0; 33,6; 35,2 bis 131,2; 132,8 ms

Modultimer

Als Basis für eine Synchronisation der einzelnen Kanäle verfügt das Modul über einen internen Modultimer, der global für alle Kanäle gilt. Mit Hilfe dieser festgelegten Zeitbasis können X2X und IO-Link Kommunikation miteinander synchronisiert werden. Die Periodendauer des Modultimers kann in µs vorgegeben werden. Um die Kommunikation möglichst effizient und deterministisch zu gestalten wird der Modultimer im Automatikmodus standardmäßig mit derselben Zykluszeit konfiguriert, mit der auch der X2X Link betrieben wird. Wenn nötig kann der Start des Modultimers mit Hilfe des "TimerOffsets" auf Seite 48 verschoben werden.

Der Zyklus des Modultimers wird automatisch mit dem X2X-Zyklus synchronisiert. Abhängig vom Verhältnis zwischen X2X- und Modultimer-Zykluszeit entstehen verschiedene Verhältnisse zwischen den Zyklen.

Beispiele

1 zu 1	(X2X Zyklus 1000, Timerzyklus 1000)	→ Immer genau ein Timerzyklus pro X2X Zyklus
2 zu 1	(X2X Zyklus 2000, Timerzyklus 1000)	→ Immer genau zwei Timerzyklen pro X2X Zyklus
1 zu 2	(X2X Zyklus 1000, Timerzyklus 2000)	→ Immer genau ein Timerzyklus pro 2 X2X Zyklen
3 zu 5	(X2X Zyklus 1500, Timerzyklus 2500)	→ Immer genau 3 Timerzyklen pro 5 X2X Zyklen

Synchronbetrieb

Im Gegensatz zum Freilaufenden Betrieb kann in dieser Betriebsart der Synchronbetrieb und die Synchronisations-Zykluszeit für jeden Kanal einzeln eingestellt werden.

Der Betriebsmodus SYNCHRON optimiert das Zusammenspiel von X2X Link und IO-Link Kommunikation. Die Ressourcen des Moduls wurden für diesen Modus ausgelegt, daher soll diese Konfiguration für die Kanäle des Moduls verwendet werden.

- Im Betriebsmodus SYNCHRON (automatisch) berechnet sich das Modul die erforderlichen Zeitparameter selbstständig. Es wird ein IO-Link Zyklus bestimmt, welcher der IO-Link Spezifikation entspricht. Die gewählte IO-Link Zykluszeit entspricht dabei dem kleinstmöglichen Vielfachen der Modultimerzykluszeit, die folgende Bedingungen erfüllt:
  - Gültige IO-Link Zykluszeit
  - Größer oder gleich der minimalen Zykluszeit des Device
- Im Betriebsmodus SYNCHRON (manuell) kann der Anwender das Zeitverhalten des Moduls manuell konfigurieren. Dabei kann der Anwender sowohl die Synchronisations-Zykluszeit als auch den IO-Link Zyklus manuell über einen Faktor festlegen.

Synchronisations-Zykluszeit

Synchronisations-Zykluszeit = Timer Zykluszeit * CfO_ReqCycleMultiple0x

Die Synchronisation stellt sicher, dass Synchronisationszyklen mit derselben Synchronisationszykluszeit parallel laufen und nicht durch Timerzyklen versetzt werden.

IO-Link Zykluszeit

IO-Link Zykluszeit = Synchronisations-Zykluszeit / CfO_ReqCycleDivisor0x

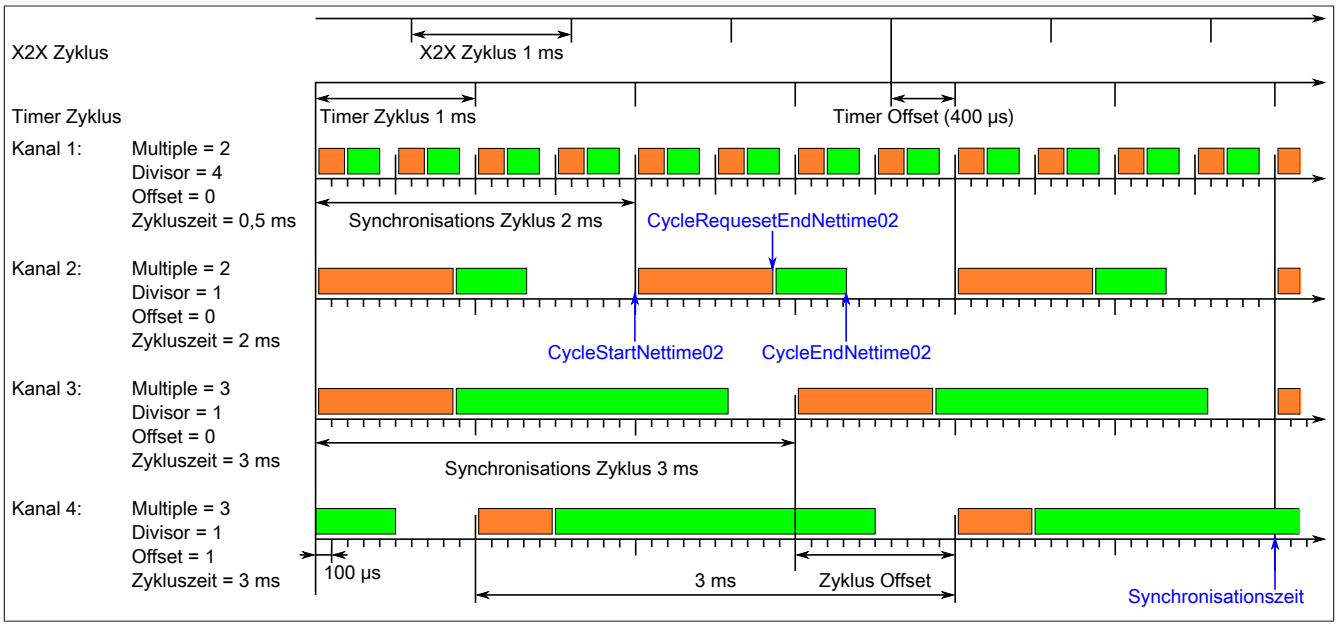
Der IO-Link Zyklus wird für jeden Kanal einzeln eingestellt. Wenn nötig, kann der IO-Link Zyklus des Kanals mit Hilfe eines kanalspezifischen Offsets verschoben werden. Auf diese Weise können Kanäle so aufeinander abgestimmt werden, dass sie ihre Anfragen z. B. zum selben Zeitpunkt beenden.
Bei sehr kurzen Zykluszeiten (<1 ms) kann es vorkommen, dass die Daten nicht schnell genug verarbeitet werden können. In diesem Fall verzögern sich die darauffolgenden Zyklen, was durch ein Rücksetzen des Statusbits für die Synchronisation angezeigt wird.

Information:

Wird der IO-Link Zyklus kleiner als die minimale Zykluszeit des Devices konfiguriert, so wird automatisch ein Zyklus gewählt, der folgende Bedingungen erfüllt:

- **Vielfaches des Modultimerzyklus**
- **Gültige IO-Link Zykluszeit**
- **Größer oder gleich der minimalen Zykluszeit des Device**

Beispiel einer Konfiguration



Der Modultimer in diesem Beispiel

- Die Periodendauer des Modultimers wurde nicht explizit festgelegt. Sie entspricht in diesem Fall der Zykluszeit des X2X Link.
- Der Modultimer wurde mit einem Timer Offset von 400 µs beaufschlagt; d.h., der Modultimerzyklus beginnt um 400 µs versetzt zum X2X Linkzyklus.

Die IO-Link Kommunikation in diesem Beispiel

- Über die Parameter "Multiple" auf Seite 48 und "Divisor" auf Seite 48 ergibt sich die kanalspezifische Zykluszeit für die IO-Link Kommunikation.
- Kanal 1 und 2 haben einen gemeinsamen Synchronisationszyklus von 2 ms.
  Kanal 3 und 4 haben einen gemeinsamen, durch den Offset verschobenen, Synchronisationszyklus von 3 ms.
- Kanäle starten die Abfrage zu Beginn eines gemeinsamen Synchronisationszyklus gleichzeitig.
- Der IO-Link Zyklus des vierten Kanals wurde mit einem Offset von 1 ms verzögert.
- Alle Kanäle haben einen gemeinsamen Synchronisationszyklus von 6 ms.

Freilaufender (asynchroner) Betrieb

Wenn sich die Zykluszeit von IO-Link und X2X Link nicht synchronisieren lassen, kann die IO-Link Zykluszeit direkt vorgegeben werden. Die IO-Link Kommunikation läuft dabei unabhängig von Modultimer und X2X Zyklus. Bis auf "CycleEndNettime" auf Seite 55 können keine weiteren NetTime-Datenpunkte verwendet werden. Die Zykluszeit von freilaufenden IO-Link Kanälen wird über das entsprechende Register direkt vorgegeben. Allerdings können Schwankungen auftreten, wenn die Ressourcen des Moduls erschöpft sind.

6.4.7.1 TimerCycle

Name:
CfO_TimerCycle

Dieses Register kann zur Konfiguration der synchronen IO-Link Kommunikation genutzt werden. Falls der Modultimer nicht mit demselben Zyklus betrieben werden soll, kann in diesem Register die Periodendauer des Modultimers in µs festgelegt werden. Damit können Kanäle untereinander synchronisiert werden, auch wenn eine sehr ungewöhnliche X2X-Zykluszeit verwendet wird.

Datentyp	Werte	Information
UDINT	0 bis 4.294.967.295	Bus Controller Default: Aktuelle X2X-Zykluszeit

6.4.7.2 TimerOffset

Name:
CfO_TimerOffset

Dieses Register kann zur Konfiguration der synchronen IO-Link Kommunikation genutzt werden. Falls der Modultimer zeitversetzt zum X2X Link laufen soll, kann in diesem Register festgelegt werden, um wie viele µs vorher oder nachher der Modultimer versetzt werden soll.

Datentyp	Werte	Information
INT	-32768 bis 32767	Bus Controller Default: 0

6.4.7.3 ReqCycleMultiple

Name:
CfO_ReqCycleMultiple01 bis CfO_ReqCycleMultiple04

Mit diesem Register kann die Synchronisations-Zykluszeit eines Kanals manuell eingestellt werden. Diese Zykluszeit kann zusammen mit dem Register "ReqCycleDivisor" auf Seite 48 genutzt werden, um die IO-Link Zykluszeit festzulegen. Siehe "Synchronbetrieb" auf Seite 46 für ein Beispiel.

Information:

Falls für einen IO-Link Kanal dieses Register nicht festgelegt oder mit null vorgegeben ist, werden die Werte der Register CycleMultiple und CycleDivisor automatisch beim Hochlauf des Moduls berechnet.

Datentyp	Werte	Information
UINT	0 bis 65.535	Bus Controller Default: 0

6.4.7.4 ReqCycleDivisor

Name:
CfO_ReqCycleDivisor01 bis CfO_ReqCycleDivisor04

Dieses Register kann zusammen mit "ReqCycleMultiple" auf Seite 48 genutzt werden, um die IO-Link Zykluszeit festzulegen. Siehe "Synchronbetrieb" auf Seite 46 für ein Beispiel.

Information:

Falls für einen IO-Link Kanal dieses Register nicht festgelegt oder mit null vorgegeben ist, werden die Werte der Register CycleMultiple und CycleDivisor automatisch beim Hochlauf des Moduls berechnet.

Datentyp	Werte	Information
UINT	0 bis 65.535	Bus Controller Default: 0

6.4.7.5 ReqCycleOffset

Name:
CfO_ReqCycleOffset01 bis CfO_ReqCycleOffset04

Mit diesem Register kann der IO-Link Zyklus eines Kanals zum Synchronisationszyklus verschoben werden. Diese Verschiebung kann sinnvoll sein, wenn alle Kanäle mit derselben Zykluszeit laufen. In diesem Fall werden alle Kanäle gleichzeitig fertig, was dazu führen kann, dass das Modul nicht alle Daten rechtzeitig verarbeitet. Über Offsets können solche Engpässe verhindert und das Datenaufkommen gleichmäßiger verteilt werden.

Datentyp	Werte	Information
UINT	0 bis 65.535	Einstellung erfolgt in Timer-Zyklen; Bus Controller Default: 0

6.4.7.6 ReqCycleTime

Name:
CfO_ReqCycleTime01 bis CfO_ReqCycleTime04

Dieses Register wird bei der freilaufenden (asynchronen) IO-Link Kommunikation genutzt. Es enthält die direkt vorgegebene Zykluszeit in µs für die IO-Link Abfrage.

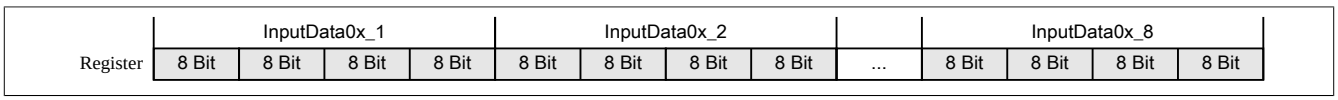
Information:

- **Im freilaufenden Modus dürfen bis auf "CycleEndNettime" auf Seite 55 keine NetTime-Datenpunkte verwendet werden.**
- **Unterschreitet die vorgegebene Zykluszeit der IO-Link Kommunikation die minimale Zykluszeit des Devices, werden die IO-Link Daten mit der minimalen Zykluszeit des Devices abgefragt.**
- **Für eine effiziente IO-Link Kommunikation, sollte der eingestellte Abfragezyklus den spezifizierten IO-Link Zykluszeiten entsprechen. Bei einem unpassenden Wert wird automatisch die nächste passende Zykluszeit verwendet.**

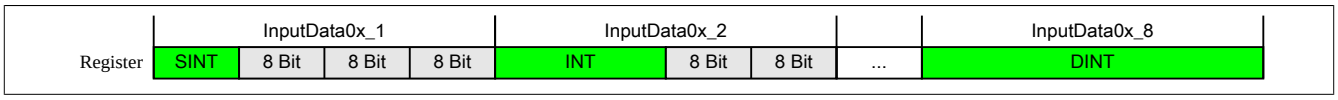
Datentyp	Werte	Information
UINT	0 bis 65.535	In 100 µs Schritten; Bus Controller Default: 0

6.5 IO-Link Kommunikation

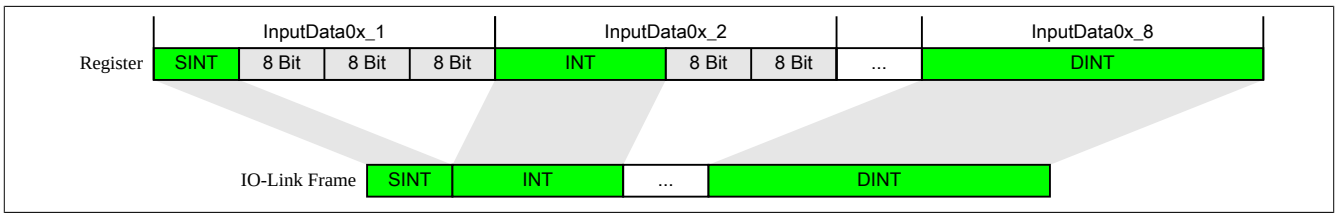
Wenn der entsprechende IO-Link Kanal des Moduls im Automation Studio auf "operate" konfiguriert wurde, versucht das Modul IO-Link Prozessdaten mit dem angeschlossenen IO-Link Device auszutauschen. Für jeden aktiven IO-Link Kanal werden im Speicher des Moduls 8 Register InputData0x_y und 8 Register OutputData0x_y allokiert.



Um den tatsächlichen IO-Link Frame zu definieren, muss festgelegt wie viele der maximal 8 Register genutzt werden und welchen Datentyp die IO-Link Daten haben.



Aus diesen initialisierten Datenpunkten ergibt sich anschließend der IO-Link Frame, für die Kommunikation mit dem IO Link Device.



Um die IO-Link Daten bis zur SPS zu übertragen, muss bei der Datentypdefinition für die IO-Link Kommunikation auch die Bandbreite des X2X Links beachtet werden. Diese Einschränkung kann minimiert werden, wenn statt der Standarddatentypen die sogenannte "OCTET"-Datenpunkte bzw. "mutiplexed OCTET"-Datenpunkte verwendet werden.

Byte-Arrays "OCTET":

Pro Kanal und Richtung sind 8 Register mit bis zu 32 Bit vorgesehen. Auf diese Weise können 8 Datenpunkte übertragen werden. Falls diese Datenmenge nicht ausreicht, kann ein Bytearray für die Erzeugung des IO-Link Frames verwendet werden. Dabei muss der Anwender die Aufteilung der IO-Link Frames innerhalb der Applikation bewerkstelligen und die Byteorder im IO-Link Device beachten.

Byte-Arrays "multiplexed OCTET":

Bei der Übertragung der IO-Link Daten über den X2X Link ist zu beachten, dass nicht alle 32 Byte der 4 IO-Link Kanäle gleichzeitig am zyklischen Teil des X2X Links angemeldet werden können. In Eingangsrichtung wurde das Modul diesbezüglich erweitert. Die Daten können time multiplexed im Hintergrund übertragen werden. Je nach Datenmenge werden dabei mehrere X2X-Zyklen benötigt, um neue Daten zwischen dem Modul und der Steuerung zu übertragen. In Ausgangsrichtung steht dieser Modus nicht zur Verfügung. Deshalb können IO-Link Ausgangsdaten nur mit einer maximalen Länge von 27 Byte erzeugt werden.

SIO-Modus

"SIO" steht für Standard-I/O und beschreibt eine alternative Verwendung für den Anschluss C/Qx. Wenn ein Kanal des Moduls nicht für die IO-Link Kommunikation benötigt wird, kann das Pin als Standard-IO verwendet werden. Der Anwender kann entscheiden, ob er den Standard-IO als Eingang oder Ausgang verwenden möchte. Der IO-Link Standard lässt außerdem zu, dass die IO-Link Kommunikation gestoppt und wieder gestartet werden kann. Sollte die IO-Link Kommunikation während der Laufzeit gestoppt werden, kann der Anschuss C/Qx als Standard-Ausgang verwendet werden.

6.5.1 OutputData

Name:
OutputData01_1 bis OutputData04_8

Ausgangsdaten vom IO-Link Device im IO-Link Kommunikationsmodus. Alternativ kann auch ein Bytearray verwendet werden. Der Anwender muss sich dann selbst um die Aufteilung der Bytes in die benötigten Datentypen kümmern.
Über das Register "PDO_TypeInfo" auf Seite 45 kann konfiguriert werden, wie viele Bytes aus den Ausgangsregistern in den IO-Link Frame übernommen werden sollen.

Datentyp	Werte
USINT	0 bis 255
SINT	-128 bis 127
UINT	0 bis 65535
INT	-32768 bis 32767
UDINT	0 bis 4.294.967.295
DINT	-2.147.483.648 bis 2.147.483.647
REAL	-3.4E38 bis 3.4E38

6.5.2 Digitale SIO-Ausgänge

Name:
DigitalOutput01 bis DigitalOutput04
DisablePowerSupply01 bis DisablePowerSupply04

Wenn ein Kanal im SIO-Modus (SIO-Ausgang) betrieben wird, kann über dieses Register der SIO-Ausgänge des IO-Link Kanals gesteuert werden. Zusätzlich lässt sich die Versorgung jedes IO-Link Kanals individuell ein- bzw. ausschalten.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0	DigitalOutput01	0	Digitalen SIO-Ausgang 01 zurücksetzen
		1	Digitalen SIO-Ausgang 01 setzen
...		...	
3	DigitalOutput04	0	Digitalen SIO-Ausgang 04 zurücksetzen
		1	Digitalen SIO-Ausgang 04 setzen
4	DisablePowerSupply01	0	Versorgung IO-Link Kanal 01 einschalten
		1	Versorgung IO-Link Kanal 01 ausschalten
...		...	
7	DisablePowerSupply04	0	Versorgung IO-Link Kanal 04 einschalten
		1	Versorgung IO-Link Kanal 04 ausschalten

6.5.3 InputData

Name:
InputData01_1 bis InputData04_8

Eingangsdaten vom IO-Link Device im IO-Link Kommunikationsmodus. Alternativ kann auch ein Bytearray verwendet werden. Der Anwender muss sich dann selbst um die Aufteilung der Bytes in die benötigten Datentypen kümmern.

Datentyp	Werte
USINT	0 bis 255
SINT	-128 bis 127
UINT	0 bis 65535
INT	-32768 bis 32767
UDINT	0 bis 4.294.967.295
DINT	-2.147.483.648 bis 2.147.483.647
REAL	-3.4E38 bis 3.4E38

6.5.4 Digitale SIO-Eingänge

Name:
DigitalInput01 bis DigitalInput04

Wenn ein Kanal im SIO-Modus (SOI-Eingang) betrieben wird, kann über dieses Register der Eingangszustand des Kanals eingelesen werden.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0	DigitalInput01	0	Digitaler SIO-Eingang 01 zurückgesetzt
		1	Digitaler SIO-Eingang 01 gesetzt
...		...	
3	DigitalInput04	0	Digitaler SIO-Eingang 04 zurückgesetzt
		1	Digitaler SIO-Eingang 04 gesetzt
4 - 7	Reserviert	-	

6.6 IO-Link Statusrückmeldung

Im nachfolgenden Kapitel werden die Statusregister der IO-Link Kommunikation erläutert. Die Statusinformationen geben Auskunft über die aktuelle Situation zwischen Modul und IO-Link Device. Sie können von der Steuerung abgerufen und im Applikationstask ausgewertet werden.

6.6.1 Sync (Statusbyte)

Name:
Synchronized01 bis Synchronized04
CycleEnd01 bis CycleEnd04

Über dieses Statusregister meldet das Modul, ob während des letzten Modulzyklus eine fehlerfreie Kommunikation mit dem Device möglich war.

- Die CycleEnd-Bits zeigen an, ob die zuletzt an das IO-Link Device gesendeten Daten verarbeitet wurden. Die CycleEnd-Bits werden nach jedem X2X-Zyklus zurückgesetzt.
- Die Synchronized-Bits zeigen an, dass der Kanal fehlerfrei synchronisiert ist.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0	Synchronized01	0	Synchronisation für Kanal 1 nicht OK
		1	Synchronisation für Kanal 1 OK
...		...	
3	Synchronized04	0	Synchronisation für Kanal 4 nicht OK
		1	Synchronisation für Kanal 4 OK
4	CycleEnd01	0	IO-Zyklusende: Keine neuen IO-Link Daten
		1	IO-Zyklusende: Neue Daten gesendet und empfangen
...		...	
7	CycleEnd04	0	IO-Zyklusende: Keine neuen IO-Link Daten
		1	IO-Zyklusende: Neue Daten gesendet und empfangen

6.6.2 Überlast (Statusbyte)

Name:
Overload01 bis Overload04

Über dieses Statusregister meldet das Modul, ob auf der Kanalversorgung oder Datenleitung eine Überlast in Form von Überstrom oder Übertemperatur aufgetreten ist.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0	Overload01	0	Kanal 1: keine Überlast
		1	Kanal 1: Überlast
...		...	
3	Overload04	0	Kanal 4: keine Überlast
		1	Kanal 4: Überlast
4 - 7	Reserviert	-	

6.6.3 ChannelStatus

Name:
ChannelStatus01 bis ChannelStatus04

Mit Hilfe dieses Registers wird der aktuelle Status des IO-Link Kanals angezeigt.

Datentyp	Werte	Information	Zustand
USINT	0	Kanal inaktiv	Deaktiviert
	1	Verwendung als digitaler SIO-Ausgang	SIO-Modus
	2	Verwendung als digitaler SIO-Eingang	
	3	Hochlauf des IO-Link Devices, Modus PREOPERATIONAL	Die Kommunikation läuft, aber es werden keine Prozessdaten ausgetauscht. Azyklische Zugriffe sind jedoch möglich.
	4	Betrieb, Modus OPERATE	Kommunikation läuft
	5	Betrieb, Parameterserverdaten in Ordnung	
	6	Parameterserver: Upload aktiv	Die Kommunikation läuft und es werden Prozessdaten geliefert.
	7	Parameterserver: Download aktiv	
	8	Parameterserver: Löschen aktiv	
	9	IODD-Parameter werden geschrieben	
	10 bis 20	Reserviert	
	21	Genereller Fehler im Parameterserver; z. B. • Parameterserver wird nicht unterstützt • Fehler beim Zugriff auf ein Objekt das vom Parameterserver verwaltet wird • Interner Fehler	Kommunikation läuft. Am Parameterserver ist jedoch ein Fehler aufgetreten. Fehler des Parameterservers können über das Register "DsControl" auf Seite 58 quittiert werden.
	22	Der Parameterserver ist vom IO-Link Device gesperrt.	
	23	Parameterserver leer: Es wurde versucht Daten auf das IO-Link Device zu laden, obwohl keine Daten im EEPROM des DS-Moduls gespeichert sind.	
	24	Neue Seriennummer erkannt: Der Benutzer muss über Register "DsControl " auf Seite 58 entscheiden, was zu tun ist (Upload – Download – Default-Werte wiederherstellen)	
	25	Parameterserver nicht kompatibel (neue DeviceID oder neue VendorID erkannt): Die Daten im EEPROM passen nicht zum angeschlossenen IO-Link Device. Der Benutzer muss über Register "DsControl " auf Seite 58 entscheiden, ob ein Upload durchgeführt werden soll.	
	26	Upload-Anforderung empfangen: Der Benutzer muss über Register "DsControl " auf Seite 58 entscheiden, was zu tun ist (Upload – Download – Default Werte wiederherstellen).	
	27	Die Parameterprüfsumme des IO-Link Device hat sich geändert: Der Benutzer muss über Register "DsControl " auf Seite 58 entscheiden, was zu tun ist (Upload – Download – Default Werte wiederherstellen).	
	28	Fehler beim Senden des SAVE-Kommandos	
	29	Reserviert	
	30	Prozessdaten ungültig	Die Kommunikation läuft. Die Prozessdaten wurden vom Device jedoch als ungültig markiert.
	31 - 39	Reserviert	
	40	Keine Verbindung	Keine Kommunikation
	41	Die konfigurierte RevisionID wird vom angeschlossenen Device nicht unterstützt.	
	42	Die DeviceID oder VendorID des angeschlossenen IO-Link Devices stimmen nicht mit den vorgegebenen IDs überein.	Die Kommunikation läuft, aber es werden keine Prozessdaten ausgetauscht. Azyklische Zugriffe sind jedoch möglich.
	43	Die konfigurierte Seriennummer stimmt nicht mit der Seriennummer des angeschlossenen Devices überein.	
	44	Zeitstempelfehler Das IO-Link Device unterstützt keine IO-Link Zeitstempel.	
	45	Fehler beim Hochlauf des Device.	Keine Kommunikation
	46 bis 255	Reserviert	

6.6.4 FrameCount

Name:
FrameCount01 bis FrameCount04

In diesem Register werden die empfangenen IO-Link Frames gezählt. Im Gegensatz zu den Sync-Bits stellt das FrameCount Register sicher, dass wirklich alle Frames erkannt werden. Selbst bei Verlust von X2X Zyklen, oder wenn der IO-Link Zyklus schneller ist als der X2X Zyklus.

Datentyp	Werte
SINT	-128 bis 127

6.6.5 CycleStartNettime

Name:
CycleStartNettime01 bis CycleStartNettime04

Mit Hilfe dieses Registers kann der Wert der NetTime, zum Startzeitpunkt des letzten IO-Link Zyklus, ausgelesen werden.

Für weitere Informationen zu NetTime und Zeitstempel siehe "NetTime Technology" auf Seite 31.

Datentyp	Werte
INT	-32768 bis 32767
DINT	-2.147.483.648 bis 2.147.483.647

6.6.6 CycleEndNettime

Name:
CycleEndNettime01 bis CycleEndNettime04

Mit Hilfe dieses Registers kann der Wert der NetTime, zum Endzeitpunkt des letzten IO-Link Zyklus, ausgelesen werden.

Für weitere Informationen zu NetTime und Zeitstempel siehe "NetTime Technology" auf Seite 31.

Datentyp	Werte
INT	-32768 bis 32767
DINT	-2.147.483.648 bis 2.147.483.647

6.7 IO-Link Event-Schnittstelle

Bei der Event-Schnittstelle handelt es sich um eine interruptgesteuerte Hintergrund-Kommunikation. Sie ermöglicht den angeschlossenen IO-Link Devices spezielle Nachrichten, sogenannte "Eventcodes", an den Master zu senden.

Das Modul kann bis zu 16 dieser Nachrichten empfangen, zwischenpuffern und für den Abruf von der Steuerung bereitstellen. Im Wesentlichen wird dafür ein FIFO-Speicher genutzt, der unabhängig von der zyklischen Kommunikation verwaltet wird.

> **Information:**
>
> **Wenn eine Nachricht über die Event-Schnittstelle empfangen wird und der FIFO-Speicher voll, wird die älteste Nachricht im Puffer überschrieben. In seltenen Fällen kann das dazu führen, dass Nachrichten verloren gehen, bevor sie ausgewertet wurden.**

Ablauf beim Lesen eines Events

- Ein neues Event wurde vom Device ausgelöst. Dies wird durch die Erhöhung von "EventPortSeq" auf Seite 56 angezeigt.
- Eventdaten können mit Hilfe der Register "EventQualifier" auf Seite 56 und "EventCode" auf Seite 56 ausgelesen werden.
- Das Event muss quittiert werden. Dazu muss die Sequenznummer aus "EventPortSeq" auf Seite 56 in die Sequenznummer von "EventQuit" auf Seite 57 kopiert werden.
- Erst nach der Quittierung des Events wird das nächste Event übergeben.

6.7.1 EventPortSeq

Name:
EventPortSeq

In diesem Register wird die Sequenznummer erhöht, sobald ein neues Ereignis von einem IO-Link Device generiert wurde. Zusätzlich wird noch die betreffende Kanalnummer angezeigt.

Datentyp	Werte
USINT	0 bis 255

Bitstruktur:

Bit	Beschreibung	Wert	Information
0 - 3	Sequenznummer	0 bis 15	
4 - 6	IO-Link Kanalnummer	001	IF1 (Kanal 1)
		010	IF2 (Kanal 2)
		011	IF3 (Kanal 3)
		100	IF4 (Kanal 4)
7	Reserviert	0	

6.7.2 EventQualifier

Name:
EventQualifier

In diesem Register werden zusätzliche Informationen zu dem Ereignis abgebildet.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0 - 2	Instanzschicht, welches das Ereignis generiert hat	000	Unbekannt
		001	Hardware
		010	Datenaustauschschicht des IO-Link Device
		011	Anwendungsschicht des IO-Link Device
		100	Anwendung
3	Ursache des Ereignisses	0	Device
		1	Master
4 - 5	Art des Ereignisses	00	Reserviert
		01	Information
		10	Warnung
		11	Fehler
6 - 7	Modus des Ereignisses	00	Reserviert
		01	Einmaliges Ereignis
		10	Ereignis nicht mehr gemeldet (z. B: Spannung wieder OK)
		11	Ereignis gemeldet (z. B: Spannung zu niedrig)

6.7.3 EventCode

Name:
EventCode

In diesem Register wird der Ereigniscode des übertragenen Ereignisses abgebildet. Die Ereigniscodes können aus herstellerspezifischen bzw. durch die IO-Link Spezifikation vorgegebenen Ereigniscodes bestehen.

Datentyp	Werte
UINT	0 bis 65.535

6.7.4 EventsLeft

Name:
EventsLeft

Dieses Register gibt die Anzahl der noch nicht verarbeiteten Ereignisse im FIFO-Speicher an.

Datentyp	Werte
USINT	0 bis 16

6.7.5 EventQuit

Name:
EventQuit

Mit Hilfe dieses Register können Ereignisse quittiert werden. Dazu muss die Sequenznummer des zu quittierenden Ereignisses in dieses Register kopiert werden.

Datentyp	Werte
USINT	0 bis 15

6.7.6 EventQuitReadBack

Name:
EventQuitReadBack

Dieses Register enthält die Sequenznummer des zuletzt quittierten Ereignisses.

Datentyp	Werte
USINT	0 bis 15

6.8 IO-Link Parameterserver

Beim Parameterserver handelt es sich um eine Funktion, die von der IO-Link Spezifikation definiert wird. Im Modul ist diese Funktion grundsätzlich aktiviert und kann mit Hilfe des Registers "CfO_DS_Config" auf Seite 59 verwaltet werden.

Der Parameterserver erlaubt dem Modul, Konfigurationsparameter des angeschlossenen IO-Link Devices auszulesen. Die Daten des Drittanbietergerätes werden im EEPROM gespeichert und können anschließend z. B. nach dem Austausch des IO-Link Devices automatisch wiederhergestellt werden.

> **Information:**
>
> **Die Auswahl der übertragenen Konfigurationsdaten, hängt vom angeschlossenen IO-Link Device ab. Das Modul fungiert lediglich als Datenspeicher. Es fordert die Konfigurationsdaten des IO-Link Gerätes an, speichert die Antwort und überträgt die erhaltenen Informationen bei Bedarf wieder an das angeschlossene IO-Link Device.**
>
> **Eine Veränderung der ausgelesenen Parameterserver-Daten im Speicher des Moduls ist nicht vorgesehen.**

Der Eventcode 0xFF91

Das Modul ist in der Lage die "Datenspeicher-Upload-Anforderung" (Eventcode 0xFF91) des angeschlossenen IO-Link Devices zu verarbeiten, um den Speicher des Parameterservers im Modul automatisch zu verwalten.

Im Standard wurde nicht genau festgelegt, wann der Eventcode erzeugt werden muss. Die meisten IO-Link Devices erzeugen ihn, sobald sich die Konfigurationsparameter ändern. Bei einigen IO-Link Devices kann es vorteilhaft sein die Up- und Download-Vorgänge manuell anzufordern. Für diesen Zweck bietet das Modul die Möglichkeit, die Übertragung der Parameterserverdaten an die individuellen Bedürfnisse der Applikation anpassen.

> **Information:**
>
> **Die automatische Verwaltung kann genutzt werden, wenn das angeschlossene IO-Link Devices des Drittanbieters die Funktion Parameterserver unterstützt und den Eventcode erzeugen kann.**

Der Parameterserver

Wenn vom IO-Link Device unterstützt, können mit Hilfe des IO-Link Parameterservers z. B. applikationsspezifische Konfiguration des Devices vom IO-Link Master ausgelesen werden. Der Parameterserver des Moduls ist grundsätzlich aktiviert und kann mit Hilfe eines Steuerungsregisters verwendet werden.

Welche Datenspeicherparameter übertragen werden, hängt vom angeschlossenen IO-Link Device ab. Die ausgelesenen Informationen werden im EEPROM des Moduls gespeichert und können z. B. automatisch nach dem Austausch des Device wieder eingespielt werden.

Das Modul ist in der Lage die Datenspeicher-Upload-Anforderung (Ereigniscode 0xFF91) der IO-Link Spezifikation zu verarbeiten. Die Anforderung wird in der Regel ausgelöst, wenn Parameter auf dem Device geändert werden. Je nach Konfiguration kann in diesem Fall ein Upload der Datenspeicherdaten gestartet werden (Standard).

Automatische Verwaltung der Datenspeicherparameter

Die automatische Verwaltung wurde gemäß IO-Link Spezifikation gestaltet. Da der IO-Link Standard an dieser Stelle tolerant gestaltet wurde, kann es sein, dass einige IO-Link Devices eine andere Behandlung erfordern. Diese kann über das Register konfiguriert werden.

Up- bzw. Download wird ausgeführt, wenn folgende Bedingungen zutreffen:

- DsControl0x = 1
- Während des Device-Hochlaufs oder wenn eine Datenspeicher-Upload-Anforderung empfangen wurde.

Offline-Parametrierung

Bei der Offline-Parametrierung werden die im Automation Studio eingestellten Konfigurationsdaten für das Device im Projekt gespeichert und nach dem Download des Projekts bzw. nach erstellen der Speicherkarte für die Steuerung automatisch konfiguriert. Anders als beim Parameterserver, wo die Werte aus einem vorhandenen Device ausgelesen werden, werden die Werte in diesem Fall direkt von der Applikation vorgegeben. Die Werte werden nach dem Download nur ein einziges mal automatisch parametriert. Erst wenn vom Automation Studio eine neue Parameterdatei kommt, das Device getauscht wurde, oder wenn der Download manuell durch die Bibliothek gestartet wird, wird der Vorgang wiederholt.

Diese Funktion arbeitet unabhängig vom Parameterserver. Wenn der Parameterserver jedoch aktiviert ist, so startet dieser bei Bedarf nach der Offline-Parametrierung und speichert die entsprechenden Daten. Bei einem Austausch des Devices werden in diesem Fall die Daten vom Parameterserver auf das Device geladen.

6.8.1 DsControl

Name:
DsControl01 bis DsControl04

Mit Hilfe dieses Registers wird der "Parameterserver" auf Seite 57 manuell gesteuert. Jede Aktion wird beim Setzen des entsprechenden Wertes genau einmal ausgeführt. Soll dieselbe Aktion mehrfach ausgeführt werden, so muss zwischendurch dieses Register auf den Wert 0 gesetzt werden.

Datentyp	Werte	Information
USINT	0	Keine Aktion (Bus Controller Default)
	1	Betriebsmodus des Parameterservers: Automatischer Up- und Download
	2	Upload, wenn Datenspeicherparameter im Device vorhanden
	3	Download, wenn Datenspeicherparameter im Steuerungs-Speicher vorhanden und Device Datenspeicherparameter verarbeiten kann
	4	Fehlerstatus vom Parameterserver quittieren. (siehe "ChannelStatus" auf Seite 54: Fehlermeldungen 21 bis 28)
	5	Datenspeicherparameter im Steuerungs-Speicher löschen
	6	Dummy Upload starten. Startet einen Upload, ohne die Daten zu Speichern. Kann verwendet werden, um eine Upload-Anforderung zu quittieren.
	7 bis 255	Reserviert

6.8.2 DsProgress

Name:
DsProgress01 bis DsProgress04

Mit Hilfe dieser Register meldet das Modul den Fortschritt des Up- bzw. Downloads vom Parameterserver. Die Werte von 0 bis 100 können z. B. zur Implementierung einer Fortschrittsanzeige genutzt werden.

Datentyp	Werte
USINT	0 bis 100

6.8.3 CfO_DS_Config

Name:
CfO_DS_Config01 bis CfO_DS_Config04

Mit Hilfe dieser Register kann das Verhalten des Parameterservers eingestellt werden (bei manuellem Betrieb des Parameterservers). Dabei wird jedem Auslöseereignis eine entsprechende Reaktion zugeordnet.

Datentyp	Werte	Bus Controller Default
UDINT	Siehe Bitstruktur	0

Bitstruktur:

Bit	Ereignis	Wert	Reaktion
0 - 3	Die Device-ID des angeschlossenen Device stimmt nicht mit der Device-ID überein, die zusammen mit den Parametern abgespeichert ist.	000	Keine Reaktion (Bus Controller Default)
		001	Abbruch
		010	Benutzerdefinierte Reaktion. Siehe "ChannelStatus" auf Seite 54: Statusmeldung 25
		011	Upload
4 - 7	Das Device hat eine Upload Anforderung gesendet.	000	Keine Reaktion (Bus Controller Default)
		001	Abbruch
		010	Benutzerdefinierte Reaktion. Siehe "ChannelStatus" auf Seite 54: Statusmeldung 26
		011	Upload
8 - 11	Beim Hochlauf des Devices wurde eine neue Parameterprüfsumme erkannt.	000	Keine Reaktion (Bus Controller Default)
		001	Abbruch
		010	Benutzerdefinierte Reaktion. Siehe "ChannelStatus" auf Seite 54: Statusmeldung 27
		011	Upload
		100	Download
12 - 15	Die Serialnummer des angeschlossenen Device stimmt nicht mit der Serialnummer überein, die zusammen mit den Parametern abgespeichert ist.	000	Keine Reaktion (Bus Controller Default)
		001	Abbruch
		010	Benutzerdefinierte Reaktion. Siehe "ChannelStatus" auf Seite 54: Statusmeldung 24
		011	Upload
		100	Download
16 - 23	Reserviert	-	
24 - 26	Gibt an, in welcher Reihenfolge die einzelnen Ereignisse geprüft werden.	000	DeviceID, Seriennummer, Upload-Request, Parameterprüfsumme (Bus Controller Default)
		001	DeviceID, Seriennummer, Parameterprüfsumme, Upload-Request
		010	DeviceID, Upload-Request, Parameterprüfsumme, Seriennummer
		011	DeviceID, Upload-Request, Seriennummer, Parameterprüfsumme
		100	DeviceID, Parameterprüfsumme, Upload-Request, Seriennummer
		101	DeviceID, Parameterprüfsumme, Seriennummer, Upload-Request
27 - 31	Reserviert	-	

6.8.4 CfO_DS_SaveCtrl

Name:
CfO_DS_SaveCtrl01 bis CfO_DS_SaveCtrl04

Dieses Register wird zusammen mit "CfO_DS_SaveData" auf Seite 60 verwendet.

Einige IO-Link Devices müssen angewiesen werden, eingespielte Datenspeicherparameter nach einem Download remanent zu speichern. Um diese Parameter bei diesen Devices in den remanenten Speicher zu übernehmen, muss der in diesen Registern hinterlegte Index und Subindex zusammen mit den Speicherbefehl gesendet werden (z. B. Wert 163 auf Index 2, Subindex 0).

Datentyp	Werte	Bus Controller Default
UDINT	Siehe Bitstruktur	0

Bitstruktur:

Bit	Beschreibung	Wert	Information
0 - 15	Index	0 bis 255	Device-spezifischer Index für Savebefehl
16 - 24	Subindex	0 bis 255	Device-spezifischer Subindex für Savebefehl
24 - 26	Datenlänge	0	Savebefehl deaktiviert
		1 bis 4	Datenlänge des vom Device erwarteten Savebefehls in Byte
27 - 31	Reserviert		

6.8.5 CfO_DS_SaveData

Name:
CfO_DS_SaveData01 bis CfO_DS_SaveData04

Dieses Register wird zusammen mit "CfO_DS_SaveCtrl" auf Seite 59 verwendet und beinhaltet den Wert, der auf den im Register CfO_DS_SaveCtrl konfigurierten Index geschrieben wird.

Datentyp	Werte	Information
UDINT	0 bis 4.294.967.295	Bus Controller Default: 0

6.9 IO-Link Zeitstempel

Die IO-Link Zeitstempelregister erlauben eine Zuordnung von IO-Link Zeitstempeln zur NetTime einer Steuerung und umgekehrt.

Dadurch können Wertänderungen des IO-Link Device zeitlich genau der NetTime der Steuerung zugeordnet werden und umgekehrt. Ereignisse können mit einer höheren zeitlichen Auflösung erfasst bzw. ausgelöst werden, als es der IO-Link Zyklus erlauben würde. Dies ermöglicht eine zeitgenaue Reaktion der Steuerung auf Signale vom Sensor und umgekehrt. Die Auflösung hängt dabei von den verwendeten Devices ab.

Für weitere Informationen zu NetTime und Zeitstempel siehe "NetTime Technology" auf Seite 31.

Beispiele

- Bei einem Eingangs-Device wird der Zeitstempel direkt vom Device bei Auftreten eines bestimmten Ereignisses (z. B. Lichtschranke unterbrochen) gespeichert und anschließend über IO-Link übertragen. Der IO-Link Master rechnet diesen IO-Link spezifischen Zeitstempel in einen NetTime-Zeitstempel um, welcher Systemweit verwendet werden kann.
- In Ausgangsrichtung wird ein umgerechneter Zeitstempel über IO-Link zum Device übertragen. Das Ausgangs-Device reagiert zum entsprechenden Zeitpunkt und führt das vorgesehene Ereignis aus (z. B. das Schließen eines Schalters).

Information:

- **Die Zeitstempelfunktion ist gerätespezifisch und wird nicht vom jedem IO-Link Device unterstützt.**
- **Diese Funktion kann nicht genutzt werden, wenn der Kanal freilaufend (asynchron) betrieben wird.**

6.9.1 IoLinkTimestampIn

Name:
IoLinkTimestampIn01 bis IoLinkTimestampIn04

In diesem Register wird der NetTime-Zeitpunkt angezeigt, an dem das Applikationsereignis aufgetreten ist.

Für weitere Informationen zu NetTime und Zeitstempel siehe "NetTime Technology" auf Seite 31.

Datentyp	Werte
INT	-32768 bis 32767
DINT	-2.147.483.648 bis 2.147.483.647

6.9.2 IoLinkTimestampInStatusSeq

Name:
IoLinkTimestampInStatusSeq01 bis IoLinkTimestampInStatusSeq04

In diesem Register werden Informationen über den Eingangszeitstempel angezeigt.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0 - 3	Sequenznummer	0 bis 15	Die Sequenznummer wird mit jedem empfangenen gültigen Zeitstempel um 1 erhöht. Tritt der Fall auf, dass die Sequenznummer um mehr als 1 erhöht wurde, ist ein Ereignis verloren gegangen.
4	Durch Applikation ausgelöstes Ereignis 1	x	Signalzustand bei Auftreten des Zeitstempels
5	Durch Applikation ausgelöstes Ereignis 2	x	Signalzustand bei Auftreten des Zeitstempels **Beispiel:** Signalzustand bei Auftreten des Zeitstempels – Lichtschranke wurde unterbrochen → dieses Bit = 0 – Lichtschranke frei → dieses Bit = 1
6	Reserviert	-	
7	Zeitstempel Fehler	0	Kein Fehler
		1	Auf dem IO-Link Device ist ein Fehler aufgetreten. Mögliche Ursachen sind: • Mehr Zeitstempel wurden generiert, als übertragen werden konnten. • Der Wert des IO-Link Zeitstempels hat den zulässigen Wertebereich überschritten. In beiden Fällen kann eine Verringerung der IO-Link Zykluszeit Abhilfe schaffen.

6.9.3 IoLinkTimestampOut

Name:
IoLinkTimestampOut01 bis IoLinkTimestampOut04

In dieses Register kann der Anwender die NetTime für den Ausgangszeitstempel schreiben.
Die NetTime wird automatisch in einen IO-Link Zeitstempel umgerechnet. Das Ereignis wird zur festgelegten NetTime ausgelöst. Die Quittierung erfolgt über das Register "IoLinkTimestampOutStatus" auf Seite 62.

Für weitere Informationen zu NetTime und Zeitstempel siehe "NetTime Technology" auf Seite 31.

Information:

Die NetTime muss mindestens drei IO-Link Zyklen in der Zukunft liegen, ansonsten wird eine Warnung in IoLinkTimestampOutStatus gesetzt.

Der Datentyp dieses Registers muss mit dem im Register "ChannelMode" auf Seite 42, Bit 26 eingestellten Format übereinstimmen.

Datentyp	Werte	Information
INT	-32768 bis 32767	Bus Controller Default: 0
DINT	-2.147.483.648 bis 2.147.483.647	

6.9.4 IoLinkTimestampOutCtrlSeq

Name:
IoLinkTimestampOutCtrlSeq01 bis IoLinkTimestampOutCtrlSeq04

Mit Hilfe dieses Register wird die Übernahme des Zeitstempels gesteuert.

Datentyp	Werte	Bus Controller Default
USINT	Siehe Bitstruktur	0

Bitstruktur:

Bit	Beschreibung	Wert	Information
0 - 3	Sequenznummer	0 bis 15	Der Ausgangszeitstempel sowie die Applikation-Ereignisbits werden übernommen, wenn die Sequenznummer um 1 erhöht wird.
4	Applikation Ereignis 1	x	Ausgangszustand zum Zeitstempel
5	Applikation Ereignis 2	x	Ausgangszustand zum Zeitstempel
6	Warnung quittieren	0	Nicht quittieren (Bus Controller Default)
		1	Warnung quittieren
7	Fehler quittieren	0	Nicht quittieren (Bus Controller Default)
		1	Fehler quittieren

6.9.5 IoLinkTimestampOutStatus

Name:
IoLinkTimestampOutStatus01 bis IoLinkTimestampOutStatus04

In diesem Register wird der Status des Ausgangszeitstempel angezeigt.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0 - 3	Sequenznummer Quittierung	0 bis 15	Wenn ein Ausgangszeitstempel übernommen werden konnte, dann wird die Sequenznummer aus "IoLinkTimestampOutCtrlSeq" auf Seite 62 hier quittiert.
4 - 5	Reserviert	-	
6	Warnung	0	Keine Warnung
		1	Ein Zeitstempel lag nicht wenigstens 3 Zyklen weit in der Zukunft, sodass er möglicherweise verzögert ausgegeben wurde.
7	Fehler	0	Kein Fehler
		1	Es wurden mehr Zeitstempel an das Modul übertragen als ausgegeben werden konnten.

6.10 IO-Link Device-IDs

Die IO-Link Device-IDs werden vom Hersteller des IO-Link Devices festgelegt und können vom Anwender nicht verändert werden. Mit ihrer Hilfe lässt sich ein angeschlossenes IO-Link Device eindeutig identifizieren.

6.10.1 VendorId

Name:
VendorId01 bis VendorId04

Dieses Register enthält die eindeutige Hersteller-ID des IO-Link Device.

Datentyp	Werte
UINT	0 bis 65.535

6.10.2 DeviceId

Name:
DeviceId01 bis DeviceId04

Dieses Register enthält die eindeutige ID des IO-Link Device.

Datentyp	Werte
UDINT	0 bis 4.294.967.295

6.10.3 FunctionId

Name:
FunctionId01 bis FunctionId04

Dieses Register enthält die vom Hersteller vergebene Funktionsklasse des Devices.

Datentyp	Werte
UINT	0 bis 65.535

6.10.4 CycleTime

Name:
CycleTime01 bis CycleTime04

Manche IO-Link Devices kommen mit schnellen Zyklen nicht zurecht und benötigen eine höhere Zykluszeit. Mit Hilfe dieses Registers kann die aktuell angewendete IO-Link Zykluszeit des Kanals zurückgelesen werden. Die zur Kommunikation verwendete Zeit ist immer ein Vielfaches von 100 µs, z. B. 50 für 5 ms Zykluszeit.

Datentyp	Werte	Information
UINT	0 bis 65.535	Angabe in 100 µs Schritten

6.10.5 CycleMultible

Name: CycleMultible01 bis CycleMultible04

Mit Hilfe dieses Registers kann der aktuell angewendete "Multiplikator" auf Seite 48 für den IO-Link Zyklus des Kanals zurückgelesen werden.

Datentyp	Werte
UINT	0 bis 65.535

6.10.6 CycleDivisor

Name:
CycleDivisor01 bis CycleDivisor04

Mit Hilfe dieses Registers kann der aktuell angewendete "Divisor" auf Seite 48 für den IO-Link Zyklus des Kanals zurückgelesen werden.

Datentyp	Werte
UINT	0 bis 65.535

6.10.7 MinCycleTime

Name:
MinCycleTime01 bis MinCycleTime04

In diesem Register kann die minimale IO-Link Zykluszeit zurückgelesen werden. Die minimale IO-Link Zykluszeit hängt vom IO-Link Device ab und wird vom Modul nach Aufbau der Kommunikation mit dem IO-Link Device ausgelesen.

Datentyp	Werte
UINT	0 bis 65.535

6.10.8 PDI_Size

Name:
PDI_Size01 bis PDI_Size04

In diesem Register kann die vom Device vorgegebene Größe der Eingangsprozessdaten zurückgelesen werden. Dieser Wert wird beim Hochfahren des IO-Link Device ausgelesen.

Datentyp	Werte
USINT	0 bis 255

6.10.9 PDO_Size

Name:
PDO_Size01 bis PDO_Size04

In diesem Register kann die vom IO-Link Device definierte Größe der Ausgangsprozessdaten zurückgelesen werden. Dieser Wert wird beim Hochfahren des IO-Link Device ausgelesen.

Datentyp	Werte
USINT	0 bis 255

6.10.10 Baudrate

Name:
Baudrate01 bis Baudrate04

In diesem Register kann die vom IO-Link Device vorgegebene Baudrate zurückgelesen werden. Dieser Wert wird beim Hochfahren des IO-Link Device ausgelesen.

Datentyp	Werte	Information
USINT	1	COM1 = 4,8 kBit/s
	2	COM2 = 38,4 kBit/s
	3	COM3 = 230,4 kBit/s

6.10.11 IoLinkVersionID

Name:
IoLinkVersionID01 bis IoLinkVersionID04

In diesem Register kann die IO-Link Version zurückgelesen werden.

Datentyp	Werte	Information
USINT	16 (= 0x10)	V1.0
	17 (= 0x11)	V1.1

6.11 Fehlercodes

Anfragen können über Register oder den Flatstream durchgeführt werden. Falls eine Anfrage fehlschlägt, wird das Fehlerbit gesetzt und ein Fehlercode generiert. Neben den allgemeinen Fehlercodes können auch Herstellerspezifische auftreten. Diese sind der Bedienungsanleitung des entsprechenden IO-Link Devices zu entnehmen.

Fehleranzeige in den Registern

- In "ParameterCtrlIn" auf Seite 69 wird das Fehlerbit gesetzt und im Parameter Datenlänge wird die Länge des Fehlercodes angezeigt.
- "ParameterDataIn" auf Seite 69 enthält den Fehlercode.

Fehleranzeige im Flatstream

Bei gesetzten Fehlerbit setzen sich die Flatstream-Bytes folgendermaßen zusammen:

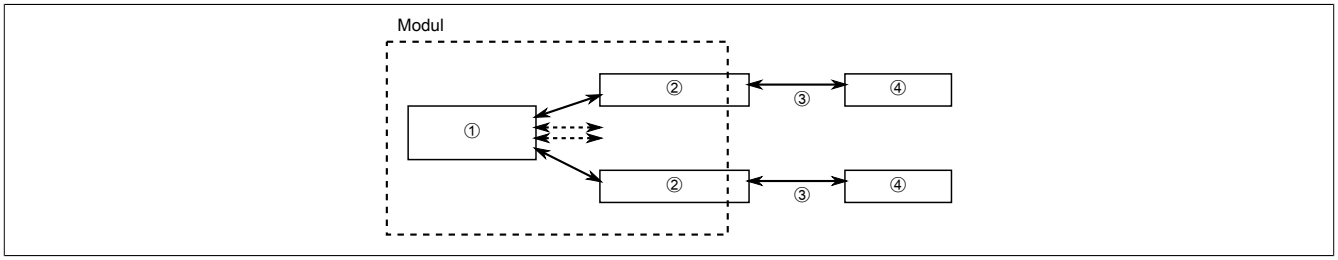
- Byte 1 bis 3: Modulspezifisches Flatstream-Array
- Byte 4: Fehlercode, bei Fehlercode 8 (= vom Device gemeldeter Fehler) enthält Byte 5 und 6 noch zusätzliche Informationen.
- Byte 5 und 6: Fehlercode vom IO-Link Device
- ...

Fehlercodes

Code	Bedeutung
1	Kein Device an diesem Kanal
2	IO-Link deaktiviert
3	Kommunikationsfehler mit Device
4	Anfragepuffer voll
5	Ereigniswarteschlange ist leer
6	Anfrage wird nicht unterstützt
7	Objektzugriff schlug fehl
8	Objektzugriff, vom Device gemeldeter Fehler
9	Falsche Kanalnummer
10	Kein Schreibzugriff möglich
11	Keine Eingangsdaten verfügbar
12	Frame zu kurz
13	Ein oder mehrere Ereignisse wurden verworfen
14	Device verfügt über keine Eingangsdaten
15	Device verfügt über keine Ausgangsdaten

6.12 Statistikzähler

In den Statistikzähler werden Kommunikationsfehler abgebildet, die zwischen den einzelnen IO-Link Komponenten aufgetreten sind.



Legende

① IO-Prozessor
② Kanalspezifische IO-Link Anschaltung
③ IO-Link Kanal
④ IO-Link Device

6.12.1 Anzahl der Befehlswiederholungen

Name:
RetryCnt01 bis RetryCnt04

Diese Register enthalten die Anzahl der Befehlswiederholungen auf Grund von Kommunikationsfehler zwischen IO-Prozessor und IO-Link Device.

Datentyp	Werte
UINT	0 bis 65.535

6.12.2 Anzahl der Prüfsummenfehler zur Steuerung

Name:
SpiErrorCnt01 bis SpiErrorCnt04

Diese Register enthalten die Anzahl der Prüfsummenfehler zwischen dem IO-Prozessor und der kanalspezifischen IO-Link Anschaltung.

Datentyp	Werte
UINT	0 bis 65.535

6.12.3 Anzahl der Kommunikationsfehler

Name:
TransmErrCnt01 bisTransmErrCnt04

Diese Register enthalten die Anzahl der Kommunikationsfehler zwischen dem IO-Prozessor und der kanalspezifischen IO-Link Anschaltung.

Datentyp	Werte
UINT	0 bis 65.535

6.12.4 Anzahl der Parityfehler

Name:
ParityErrCnt01 bis ParityErrCnt04

Diese Register enthalten die Anzahl der Parityfehler zwischen der kanalspezifischen IO-Link Anschaltung und dem IO-Link Device.

Datentyp	Werte
UINT	0 bis 65.535

6.12.5 Anzahl der Protokollfehler

Name:
FrameErrCnt01 bis FrameErrCnt04

Diese Register enthalten die Anzahl der Protokollfehler zwischen der kanalspezifischen IO-Link Anschaltung und dem IO-Link Device.

Datentyp	Werte
UINT	0 bis 65.535

6.12.6 Anzahl der Byteanzahlfehler

Name:
RxSizeErrCnt01 bis RxSizeErrCnt04

Diese Register enthalten die Anzahl der fehlerhaft empfangenen Byteanzahl zwischen der kanalspezifischen IO-Link Anschaltung und dem IO-Link Device.

Datentyp	Werte
UINT	0 bis 65.535

6.12.7 Anzahl der Prüfsummenfehler zum IO-Link Device

Name:
RxChksmErrCnt01 bisRxChksmErrCnt04

Diese Register enthalten die Anzahl der Prüfsummenfehler zwischen der kanalspezifischen IO-Link Anschaltung und dem IO-Link Device.

Datentyp	Werte
UINT	0 bis 65.535

6.12.8 Anzahl der Antwortfehler

Name:
DeviceDlyErrCnt01 bis DeviceDlyErrCnt04

Diese Register enthalten die Anzahl der Antwortfehler. Diese entstehen, wenn das IO-Link Device nicht rechtzeitig auf den Request-Frame des Masters antwortet oder wenn die Pause zwischen den einzelnen Bytes im Antwortframe zu groß ist.

Datentyp	Werte
UINT	0 bis 65.535

6.12.9 Anzahl der Zyklusfehler

Name:
CycleTimeErrorCnt01 bis CycleTimeErrorCnt04

Diese Register enthalten die Anzahl der Zyklusfehler. Diese entstehen, wenn ein IO-Link Zyklus gestartet wird, bevor der vorherige abgeschlossen und verarbeitet werden konnte. Diese Fehler können durch eine niedrigere Zykluszeit-Einstellung behoben werden.

Datentyp	Werte
UINT	0 bis 65.535

6.13 Kommando-Schnittstelle

Die Kommando-Schnittstelle bietet die Möglichkeit, über Index und Subindex auf das Objektverzeichnis des IO-Link Device zugreifen zu können. Der Zugriff kann alternativ auch mit Hilfe der AsIoLink Bibliothek oder dem Flatstream erfolgen.

> **Information:**
>
> **Mit dieser Schnittstelle können maximal die ersten 4 Byte eines Objektes gelesen oder geschrieben werden.**

Vorgang beim Schreibzugriff:

- Setzen des zu schreibenden Objektes mit Hilfe von "ParameterIndexOut" auf Seite 68 und "ParameterSubIndexOut" auf Seite 68
- Die zu schreibenden Daten in "ParameterDataOut" auf Seite 69 schreiben.
- Lesen/Schreiben, IF, und die um 1 erhöhte Sequenznummer im Register "ParameterCtrlOut" auf Seite 69 setzen.
- Warten bis die Sequenzbestätigung in "ParameterCtrlIn" auf Seite 69 gleich der Sequenznummer ist

Vorgang beim Lesezugriff:

- Setzen des zu lesenden Objektes mit Hilfe von "ParameterIndexOut" auf Seite 68 und "ParameterSubIndexOut" auf Seite 68
- In Parameter"ParameterCtrlOut" auf Seite 69 das Bit 7 löschen sowie Kanalnummer setzen und Sequenznummer erhöhen
- Warten bis die Sequenzbestätigung in "ParameterCtrlIn" auf Seite 69 gleich der Sequenznummer ist
- Fehlerstatus aus "ParameterCtrlIn" auf Seite 69 auslesen. Ein Fehler wird durch ein gesetztes Fehlerbit angezeigt.
- Daten aus "ParameterCtrlIn" auf Seite 69 lesen.

Für das Verhalten bei Auftreten eines Fehlers siehe "Fehlercodes" auf Seite 65

6.13.1 ParameterIndexOut

Name:
ParameterIndexOut

Mit diesem Register wird der Index des Objekts im Objektverzeichnis, auf das der Zugriff erfolgen soll, definiert.

Datentyp	Werte
UINT	0 bis 65.535

6.13.2 ParameterSubIndexOut

Name:
ParameterSubIndexOut

Mit diesem Register wird der SubIndex des Objekts im Objektverzeichnis, auf das der Zugriff erfolgen soll, definiert.

Datentyp	Werte
USINT	0 bis 255

6.13.3 ParameterCtrlOut

Name:
ParameterCtrlOut

Mit diesem Register wird die Art des gewünschten Zugriffes definiert.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0 - 1	Sequenznummer	0 bis 3	
2 - 3	IO-Link Kanalnummer	0	IF1 (Kanal 1)
		1	IF2 (Kanal 2)
		2	IF3 (Kanal 3)
		3	IF4 (Kanal 4)
4 - 6	Datenlänge	0 bis 4	
7	Lesen oder Schreiben	0	Lesen
		1	Schreiben

6.13.4 ParameterDataOut

Name:
ParameterDataOut_0

Dieses Register beinhaltet die Daten, die geschrieben werden sollen.

Datentyp	Werte
UDINT	0 bis 4.294.967.295

6.13.5 ParameterCtrlIn

Name:
ParameterCtrlIn

Mit diesem Register wird der Status des Zugriffs überwacht.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0 - 1	Sequenz Bestätigung	0 bis 3	
2 - 3	IO-Link Kanalnummer	0	IF1 (Kanal 1)
		1	IF2 (Kanal 2)
		2	IF3 (Kanal 3)
		3	IF4 (Kanal 4)
4 - 6	Datenlänge	0 bis 4	
7	Fehlerbit	0	Kein Fehler
		1	Fehler; der Fehlercode wird in "ParameterDataIn" auf Seite 69 angezeigt.

6.13.6 ParameterDataIn

Name:
ParameterDataIn_0

Dieses Register enthält nach einem erfolgreichen Lesezugriff die Eingangsdaten oder im Fehlerfall die Fehlercodes.

Datentyp	Werte
UDINT	0 bis 4.294.967.295

Fehleranzeige

- Ist der "Fehlercode" auf Seite 65 ungleich 8 (= vom Device gemeldeter Fehler), dann enthält das LSB den Fehlercode.
- Bei einem vom Device gemeldeter Fehler wird noch zusätzlich der von IO-Link Device übergebene Fehler angezeigt.

UDINT			
MSB			LSB
Reserviert	IO-Link Fehlercode	Zusätzlicher IO-Link Fehlercode	8

6.14 Verwendung des IO-Links am Flatstream

Das Modul bietet dem Anwender die Möglichkeit mit Hilfe des Flatstreams mit dem angeschlossenen IO-Link Device zu kommunizieren.

Die Kommunikation geschieht dabei zeitlich getrennt; d. h., Ausgangsdaten werden vollständig von der Steuerung zum Modul übertragen und geprüft. Im Anschluss initiiert das Modul die eigentliche Kommunikation mit dem IO-Link Device.

In Eingangsrichtung verhält sich das Modul auf dieselbe Weise. Nachrichten des IO-Link Device müssen vollständig im X2X Modul eingegangen sein, bevor die Flatstream-Nachricht generiert und zur Steuerung gesendet wird.

6.14.1 Allgemeine Handhabung des Flatstreams

Ein-/Ausgangssequenz	Rx/Tx-Bytes	
(unverändert)	Kontrollbyte (unverändert)	Nutzdaten-Array für Flatstream (IO-Link Informationen)

Bei der Verwendung des Flatstreams hat der Anwender die Wahl.

- Verwendung des Flatstreams wie in "Die Flatstream-Kommunikation" auf Seite 5 beschrieben.
- Benutzung die Bibliothek "AsFltGen", um Ein- bzw. Ausgangssequenzen und die Flatstream-Kontrollbytes automatisch zu verwalten.

In beiden Fällen muss in der Applikation ein modulspezifisches Array mit Flatstream-Nutzdaten erstellt werden.

6.14.2 IO-Link Informationen für den Flatstream

Um die IO-Link Kommunikation über den Flatstream verwenden zu können, muss in der Applikation ein individuelles Array definiert werden.

Für die Anfrage in Richtung Steuerung → Modul → IO-Link Device muss folgendes definiert werden.

- Kanalnummer des Moduls
- Framenummer für die Anfrage
- Art der Anfrage
- Anschließend sind, abhängig von der Anfrage, die entsprechenden IO-Link Daten anzuhängen.

Die Antwort besteht aus folgenden Teilen

- Die Kanalnummer, Framenummer, Zugriffsart und die Art der Anfrage werden wiederholt.
- Das Modul erzeugt das Fehlerbit und verwaltet das Bestätigungsbit.
- Anschließend werden die erfolgreich empfangen IO-Link Informationen oder der entsprechende "Fehlercode" auf Seite 65 angefügt.

Modulspezifisches Flatstream-Array für die IO-Link Kommunikation

Byte	Name	Wert	Beschreibung
1	Kanalnummer	1 bis 4	
2	Framenummer	0 bis 255	Diese Nummer wird in der Antwort des Moduls wiederholt. Auf diese Weise kann die spätere Antwort des Moduls eindeutig zur Anfrage zugeordnet werden.
3	Siehe Byte 3	x	
...	IO-Link Daten oder Fehlercode		Abhängig von Byte 3

Byte 3

Bit	Beschreibung	Wert	Information
0 - 2	Art der Anfrage	0	Zugriff auf Objektverzeichnis
		1	Zugriff auf Prozessdaten der Eingänge
		2	Zugriff auf Prozessdaten der Ausgänge
		3	Einzelnes Ereignis auslesen
		4	Mehrere Ereignisse auslesen
		5	Ereignisweiterleitung aktivieren
		6	Ereignisweiterleitung deaktivieren
		7	Ankündigung eines automatisch weitergeleiteten Ereignisses
3 - 4	Reserviert	-	
5	Bestätigung	0	Nachricht ohne Anfrage
		1	Antwort auf Anfrage[1)]
6	Statusbit (für Antwort-Frame)	0	Kein Fehler
		1	Fehler
7	Zugriffsart	0	Lesen
		1	Schreiben

1) Bei einer Antwort auf eine Anfrage wird zusätzlich dieses Bestätigungsbit gesetzt. Häufig beinhaltet die Antwort, mit der eine Anfrage bestätigt wird noch weitere Daten, die verarbeitet werden müssen.

6.14.3 IO-Link Daten

Abhängig von der Art der Anfrage ergeben sich unterschiedliche IO-Link Daten, die dem Flatstream-Array angehängt werden müssen.

6.14.3.1 Zugriff auf Objektverzeichnis

Anfrage

Byte	Name	Wert	Beschreibung
1 bis 3	Modulspezifisches Flatstream-Array für die IO-Link Kommunikation		
4	Indexnummer high	0 bis 255	Index des gewünschten IO-Link Parameters
5	Indexnummer low	0 bis 255	
6	Subindexnummer	0 bis 255	Subindex des IO-Link Parameters
7 bis ...	Daten	0 bis 255	Optional, für Schreibzugriff

Antwort

Byte	Name	Wert	Beschreibung
1 bis 3	Modulspezifisches Flatstream-Array für die IO-Link Kommunikation		
4 bis ...	Daten / "Fehlercode" auf Seite 65	0 bis 255	Entfällt, falls Daten erfolgreich geschrieben wurden

6.14.3.2 Zugriff auf Prozessdaten

Anfrage

Byte	Name	Wert	Beschreibung
1 bis 3	Modulspezifisches Flatstream-Array für die IO-Link Kommunikation		
4	Daten	0 bis 255	Optional, für Schreibzugriff

Antwort

Byte	Name	Wert	Beschreibung
1 bis 3	Modulspezifisches Flatstream-Array für die IO-Link Kommunikation		
4	Daten / "Fehlercode" auf Seite 65	0 bis 255	Entfällt, falls Daten erfolgreich geschrieben wurden

6.14.3.3 Zugriff auf Ereignisdaten

Anfrage

Byte	Name	Wert	Beschreibung
1 bis 3	Modulspezifisches Flatstream-Array für die IO-Link Kommunikation		

Antwort

Byte	Name	Wert	Beschreibung
1 bis 3	Modulspezifisches Flatstream-Array für die IO-Link Kommunikation		
4	Ereigniszähler – aktuell	Bit 0 bis 3	Anzahl der angehängten Ereignisse
	Ereigniszähler – anstehend	Bit 4 bis 7	Anzahl der ausstehenden Ereignisse
5	Ereignis 0 – Ereignisqualifier	0 bis 255	Siehe "EventQualifier" auf Seite 56
6	Ereignis 0 – Ereignisdaten High	0 bis 255	
7	Ereignis 0 – Ereignisdaten Low	0 bis 255	
8 - 10	Ereignis 1		
x bis (x + 2)	Ereignis n[1)]		

1) Gilt nur wenn mit Byte 3 (Bits 0 bis 2 = 4) mehrere Ereignisse abgefragt wurden. Mit Byte 3 (Bits 0 bis 2 = 3) kommt nur 1 Ereignis.

oder

Byte	Name	Wert	Beschreibung
1 bis 3	Modulspezifisches Flatstream-Array für die IO-Link Kommunikation		
4	"Fehlercode" auf Seite 65	0 bis 255	

6.14.3.4 Ereignisweiterleitung aktivieren bzw. deaktivieren

Anfrage

Byte	Name	Wert	Beschreibung
1 bis 3	Modulspezifisches Flatstream-Array für die IO-Link Kommunikation		

Antwort

Byte	Name	Wert	Beschreibung
1 bis 3	Modulspezifisches Flatstream-Array für die IO-Link Kommunikation		

oder

Byte	Name	Wert	Beschreibung
1 bis 3	Modulspezifisches Flatstream-Array für die IO-Link Kommunikation		
4	"Fehlercode" auf Seite 65	0 bis 255	

6.14.3.5 Ankündigung eines weitergeleiteten Ereignisses

Nach Aktivierung der Ereignisweiterleitung müssen Ereignisse nicht mehr zyklisch abgefragt werden. Das Modul generiert das Ereignis sobald das entsprechende Ereignis aufgetreten ist.

Nachricht

Byte	Name	Wert	Beschreibung
1 bis 3	Modulspezifisches Flatstream-Array für die IO-Link Kommunikation		
4	Ereigniszähler – aktuell	Bit 0 bis 3	Anzahl der angehängten Ereignisse
	Ereigniszähler – anstehend	Bit 4 bis 7	Anzahl der ausstehenden Ereignisse
5	Ereignis 0 – Ereignisqualifier	0 bis 255	Siehe "EventQualifier" auf Seite 56
6	Ereignis 0 – Ereignisdaten High	0 bis 255	
7	Ereignis 0 – Ereignisdaten Low	0 bis 255	
8 - 10	Ereignis 1		
x bis (x + 2)	Ereignis n[1)]		

1) Gilt nur wenn mit Byte 3 (Bits 0 bis 2 = 4) mehrere Ereignisse abgefragt wurden. Mit Byte 3 (Bits 0 bis 2 = 3) kommt nur 1 Ereignis.

oder

Byte	Name	Wert	Beschreibung
1 bis 3	Modulspezifisches Flatstream-Array für die IO-Link Kommunikation		
4	"Fehlercode" auf Seite 65	0 bis 255	

6.15 Flatstream-Register

Bei der Minimalkonfiguration müssen die Register "InputMTU" und "OutputMTU" eingestellt werden. Alle anderen Register werden beim Start mit Standardwerten belegt und können sofort genutzt werden. Sie stellen zusätzliche Optionen bereit, um Daten kompakter zu übertragen bzw. den allgemeinen Ablauf hoch effizient zu gestalten.

> **Information:**
>
> **Für detaillierte Informationen zum Flatstream siehe "Die Flatstream-Kommunikation" auf Seite 5.**

6.15.1 Anzahl der aktivierten Tx- bzw. Rx-Bytes

Name:
OutputMTU
InputMTU

Diese Register definieren die Anzahl der aktivierten Tx- bzw. Rx-Bytes und somit auch die maximale Größe einer Sequenz. Der Anwender muss beachten, dass mehr freigegebene Bytes auch eine stärkere Belastung für das Bussystem bedeuten.

Datentyp	Werte
USINT	Siehe Registerübersicht

6.15.2 Transport der Nutzdaten und der Controlbytes

Name:
TxByte1 bis TxByteN
RxByte1 bis RxByteN

(Die Größe der Zahl N ist je nach verwendetem Bus Controller Modell unterschiedlich.)

Die Tx- bzw. Rx-Bytes sind zyklische Register, die zum Transport der Nutzdaten und der notwendigen Controlbytes dienen. Die Anzahl aktiver Tx- bzw. Rx-Bytes ergibt sich aus der Konfiguration der Register "OutputMTU" bzw. "InputMTU".

- "T" - "transmit" → Steuerung *sendet* Daten an das Modul
- "R" - "receive" → Steuerung *empfängt* Daten vom Modul

Datentyp	Werte
USINT	0 bis 255

6.15.3 Kommunikationsstatus der Steuerung

Name:
OutputSequence

Dieses Register enthält Informationen über den Kommunikationsstatus der Steuerung. Es wird von der Steuerung geschrieben und vom Modul gelesen.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Bezeichnung	Wert	Information
0 - 2	OutputSequenceCounter	0 - 7	Zähler der in Output abgesetzten Sequenzen
3	OutputSyncBit	0	Output-Richtung deaktiviert (disable)
		1	Output-Richtung aktiviert (enable)
4 - 6	InputSequenceAck	0 - 7	Spiegel des InputSequenceCounters
7	InputSyncAck	0	Input-Richtung nicht bereit (disable)
		1	Input-Richtung bereit (enable)

OutputSequenceCounter

Der OutputSequenceCounter ist ein umlaufender Zähler der Sequenzen, die von der Steuerung abgeschickt wurden. Über den OutputSequenceCounter weist die Steuerung das Modul an, eine Sequenz zu übernehmen (zu diesem Zeitpunkt muss die Output-Richtung synchronisiert sein).

OutputSyncBit

Mit dem OutputSyncBit versucht die Steuerung den Output-Kanal zu synchronisieren.

InputSequenceAck

Der InputSequenceAck dient zur Bestätigung. Der Wert des InputSequenceCounters wird darin gespiegelt, wenn die Steuerung eine Sequenz erfolgreich empfangen hat.

InputSyncAck

Das Bit InputSyncAck bestätigt dem Modul die Synchronität des Input-Kanals. Die Steuerung zeigt damit an, dass sie bereit ist, Daten zu empfangen.

6.15.4 Kommunikationsstatus des Moduls

Name:
InputSequence

Dieses Register enthält Informationen über den Kommunikationsstatus des Moduls. Es wird vom Modul geschrieben und sollte von der Steuerung nur gelesen werden.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Bezeichnung	Wert	Information
0 - 2	InputSequenceCounter	0 - 7	Zähler der in Input abgesetzten Sequenzen
3	InputSyncBit	0	Nicht bereit (disable)
		1	Bereit (enable)
4 - 6	OutputSequenceAck	0 - 7	Spiegel des OutputSequenceCounters
7	OutputSyncAck	0	Nicht bereit (disable)
		1	Bereit (enable)

InputSequenceCounter

Der InputSequenceCounter ist ein umlaufender Zähler der Sequenzen, die vom Modul abgeschickt wurden. Über den InputSequenceCounter weist das Modul die Steuerung an, eine Sequenz zu übernehmen (zu diesem Zeitpunkt muss die Input-Richtung synchronisiert sein).

InputSyncBit

Mit dem InputSyncBit versucht das Modul den Input-Kanal zu synchronisieren.

OutputSequenceAck

Der OutputSequenceAck dient zur Bestätigung. Der Wert des OutputSequenceCounters wird darin gespiegelt, wenn das Modul eine Sequenz erfolgreich empfangen hat.

OutputSyncAck

Das Bit OutputSyncAck bestätigt der Steuerung die Synchronität des Output-Kanals. Das Modul zeigt damit an, dass es bereit ist, Daten zu empfangen.

6.15.5 Flatstream Modus

Name:
FlatstreamMode

Mit Hilfe dieses Registers kann eine kompaktere Anordnung beim eintreffenden Datenstrom erreicht werden.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Bezeichnung	Wert	Information
0	MultiSegmentMTU	0	Nicht erlaubt (Standard)
		1	Erlaubt
1	Große Segmente	0	Nicht erlaubt (Standard)
		1	Erlaubt
2 - 7	Reserviert		

6.15.6 Anzahl der unbestätigten Sequenzen

Name:
Forward

Über das Register "Forward" stellt der Anwender ein, wie viele unbestätigte Sequenzen das Modul abschicken darf.

Empfehlung:
X2X Link: max. 5
POWERLINK: max. 7

Datentyp	Werte
USINT	1 bis 7 Standard: 1

6.15.7 Verzögerungszeit

Name:
ForwardDelay

Mit diesem Register wird die Verzögerungszeit in µs vorgegeben.

Datentyp	Werte
UINT	0 bis 65535 [µs] Standard: 0

6.16 Minimale Zykluszeit

Die minimale Zykluszeit gibt an, bis zu welcher Zeit der Buszyklus heruntergefahren werden kann, ohne dass Kommunikationsfehler auftreten. Es ist zu beachten, dass durch sehr schnelle Zyklen die Restzeit zur Behandlung der Überwachungen, Diagnosen und azyklischen Befehle verringert wird.

	Minimale Zykluszeit
Ohne IO-Link (alle Kanäle im SIO-Modus)	≥200 µs
Mit IO-Link	≥400 µs

6.17 Minimale I/O-Updatezeit

Die minimale I/O-Updatezeit gibt an, bis zu welcher Zeit der Buszyklus heruntergefahren werden kann, so dass in jedem Zyklus ein I/O-Update erfolgt.

	Minimale I/O-Updatezeit
Ohne IO-Link (alle Kanäle im SIO-Modus)	≥200 µs
Mit IO-Link	≥400 µs (abhängig von der minimalen IO-Link Zykluszeit des IO-Link Device)