X20(c)AO2438

1 Allgemeines

Das Modul ist mit 2 Stromausgängen mit 16-Bit digitaler Wandlerauflösung ausgestattet. Es unterstützt den HART-Kommunikationsstandard für Datenübertragung, Parametrierung und Diagnose.

Die beiden Kanäle sind voneinander galvanisch getrennt ausgeführt. Der Anwender kann zwischen den 3 Ausgangsbereichen 4 bis 20 mA, 0 bis 20 mA und 0 bis 24 mA wählen.

- 2 analoge Stromausgänge
- HART-Protokoll integriert
- Unterstützt HART-Variablen
- Galvanisch getrennte Analogkanäle
- 16-Bit digitale Wandlerauflösung
- OSP-Modus
- NetTime-Zeitstempel: HART-Abbild

NetTime-Zeitstempel des HART-Abbildes

Für etliche Applikationen sind nicht nur die HART-Werte bedeutend, sondern auch der exakte Zeitpunkt des Empfangs. Das Modul verfügt dafür über eine NetTime-Zeitstempelfunktion, die den Empfang-Zeitpunkt mit einem µs-genauen genauen Zeitstempel versieht.

Die Zeitstempelfunktion basiert auf synchronisierten Timern. Tritt ein Zeitstempelereignis auf, so speichert das Modul unmittelbar die aktuelle NetTime. Nach der Übertragung der jeweiligen Daten inklusive dieses exakten Zeitpunktes in die CPU kann diese nun, gegebenenfalls mit Hilfe ihrer eigenen NetTime (bzw. Systemzeit), die Daten auswerten.

OSP-Modus

Im Funktionsmodell "OSP" (Operator Set Predefined) definiert der Anwender einen analogen Wert. Dieser OSP-Wert wird immer ausgegeben, sobald die Kommunikation zwischen Modul und Master abbricht. Alternativ kann auch der letzte gültige Ausgabewert erhalten werden.

Damit wird sichergestellt, dass bei einem Kommunikationsausfall das Modul nicht in einen undefinierten Zustand gerät.

2 Coated Module

Coated Module sind X20 Module mit einer Schutzbeschichtung der Elektronikbaugruppe. Die Beschichtung schützt X20c Module vor Betauung und Schadgasen.
Die Elektronik der Module ist vollständig funktionskompatibel zu den entsprechenden X20 Modulen.

In diesem Datenblatt werden zur Vereinfachung nur Bilder und Modulbezeichnungen der unbeschichteten Module verwendet.

Die Beschichtung wurde nach folgenden Normen qualifiziert:

- Betauung: BMW GS 95011-4, 2x 1 Zyklus
- Schadgas: EN 60068-2-60, Methode 4, Exposition 21 Tage

3 Bestelldaten

Bestellnummer	Kurzbeschreibung	Abbildung
	Analoge Ausgänge	
X20AO2438	X20 Analoges Ausgangsmodul, 2 Ausgänge, 4 bis 20 mA / 0 bis 20 mA oder 0 bis 24 mA, 16 Bit Wandlerauflösung, Einzelkanal galvanisch getrennt, unterstützt HART-Protokoll, NetTime-Funktion	
X20cAO2438	X20 Analoges Ausgangsmodul beschichtet, 2 Ausgänge, 4 bis 20 mA / 0 bis 20 mA oder 0 bis 24 mA, 16 Bit Wandlerauflösung, Einzelkanal galvanisch getrennt, unterstützt HART-Protokoll, NetTime-Funktion	
	Erforderliches Zubehör	
	Busmodule	
X20BM11	X20 Busmodul, 24 VDC codiert, interne I/O-Versorgung durchverbunden	
X20BM15	X20 Busmodul, mit Knotennummernschalter, 24 VDC codiert, interne I/O-Versorgung durchverbunden	
X20cBM11	X20 Busmodul, beschichtet, 24 VDC codiert, interne I/O-Versorgung durchverbunden	
	Feldklemmen	
X20TB12	X20 Feldklemme, 12-polig, 24 VDC codiert	

Tabelle 1: X20AO2438, X20cAO2438 - Bestelldaten

4 Technische Daten

Bestellnummer	X20AO2438	X20cAO2438
Kurzbeschreibung		
I/O-Modul	2 analoge Ausgänge 4 bis 20 mA, 0 bis 20 mA oder 0 bis 24 mA	
Allgemeines		
B&R ID-Code	0xB3AA	0xE211
Statusanzeigen	I/O-Funktion pro Kanal, Betriebszustand, Modulstatus, HART	
Diagnose		
Modul Run/Error	Ja, per Status-LED und SW-Status	
Ausgänge	Ja, per Status-LED und SW-Status	
HART Link	Ja, per Status-LED und SW-Status	
HART Fehler	Ja, per Status-LED und SW-Status	
Leistungsaufnahme		
Bus	0,05 W	
I/O-intern	1,65 W	
Zusätzliche Verlustleistung durch Aktoren (ohmsch) [W]	-	
Zulassungen		
CE	Ja	
ATEX	Zone 2, II 3G Ex nA nC IIA T5 Gc IP20, Ta (siehe X20 Anwenderhandbuch) FTZÚ 09 ATEX 0083X	
UL	cULus E115267 Industrial Control Equipment	
HazLoc	cCSAus 244665 Process Control Equipment for Hazardous Locations Class I, Division 2, Groups ABCD, T5	
DNV GL	Temperature: **B** (0 - 55 °C) Humidity: **B** (up to 100%) Vibration: **B** (4 g) EMC: **B** (bridge and open deck)	
LR	ENV1	
KR	Ja	
ABS	Ja	
EAC	Ja	
KC	Ja	-
Analoge Ausgänge		
Ausgang	4 bis 20 mA, 0 bis 20 mA oder 0 bis 24 mA per Software einstellbar	
Digitale Wandlerauflösung	16 Bit	
Einschwingzeit bei Ausgangsänderung über vollen Bereich	2 ms bis 20 s, per Software einstellbar	
Datenausgaberate		
mit Hart	210 ms (Standard)	
analog	1 ms ohne Rampe	
max. Fehler		
Gain		
4 bis 20 mA	0,025% [1]	
0 bis 20 mA	0,022% [1]	
0 bis 24 mA	0,02% [1]	
Offset		
4 bis 20 mA	0,025% [2]	
0 bis 20 mA	0,022% [2]	
0 bis 24 mA	0,02% [2]	
Ausgangsschutz	Kurzschlussfest, Überspannungsschutz (bis 30 VDC)	
Drahtbrucherkennung	Ja, per Hardware und Software	
Datenformat	INT	
Ausgabeformat		
4 bis 20 mA	INT 0x0000 bis 0x7FFF / 1 LSB = 0x0001 = 488,281 nA	
0 bis 20 mA	INT 0x0000 bis 0x7FFF / 1 LSB = 0x0001 = 610,352 nA UINT 0x0000 bis 0xFFFF / 1 LSB = 0x0001 = 305,176 nA	
0 bis 24 mA	INT 0x0000 bis 0x5DC0 / 1 LSB = 0x0001 = 1000 nA	
Belastung je Kanal	max. 600 Ω	
kurzschlussfest	Ja, dauerhaft	
Ausgangsfilter	Aktiver Tiefpass 2. Ordnung / Eckfrequenz 19 Hz Anstiegsgeschwindigkeit einstellbar	
max. Gain-Drift		
4 bis 20 mA	0,0055 %/°C [1]	
0 bis 20 mA	0,005 %/°C [1]	
0 bis 24 mA	0,005 %/°C [1]	
max. Offset-Drift		
4 bis 20 mA	0,0035 %/°C [2]	
0 bis 20 mA	0,002 %/°C [2]	
0 bis 24 mA	0,002 %/°C [2]	

Tabelle 2: X20AO2438, X20cAO2438 - Technische Daten

Bestellnummer	X20AO2438	X20cAO2438
Fehler durch Laständerung [3]		
4 bis 20 mA	0,14%	
0 bis 20 mA	0,1%	
0 bis 24 mA	0,1%	
Nichtlinearität	<0,003% [4]	
Prüfspannung		
Kanal - Kanal	1000 VAC	
Kanal - Bus	1000 VAC	
Kanal - Erde	1000 VAC	
HART		
Übertragungsrate	1200 Bit/s	
Arbeitsfrequenzen	1200 Hz / 2200 Hz	
Burst-Betrieb möglich	Ja	
Multi-Drop-Betrieb		
möglich	Ja	
Teilnehmer	Bis zu 15	
Sendeamplitude		
minimal	400 mV_{pp}	
typisch	500 mV_{pp}	
maximal	600 mV_{pp}	
Empfangsamplitude		
minimal	120 mV_{pp}	
maximal	1500 mV_{pp}	
Elektrische Eigenschaften		
Potenzialtrennung	Kanal zu Kanal und Bus getrennt	
Einsatzbedingungen		
Einbaulage		
waagrecht	Ja	
senkrecht	Ja	
Aufstellungshöhe über NN (Meeresspiegel)		
0 bis 2000 m	Keine Einschränkung	
>2000 m	Reduktion der Umgebungstemperatur um 0,5°C pro 100 m	
Schutzart nach EN 60529	IP20	
Umgebungsbedingungen		
Temperatur		
Betrieb		
waagrechte Einbaulage	-25 bis 60°C	
senkrechte Einbaulage	-25 bis 50°C	
Derating	Siehe Abschnitt "Derating"	
Lagerung	-40 bis 85°C	
Transport	-40 bis 85°C	
Luftfeuchtigkeit		
Betrieb	5 bis 95%, nicht kondensierend	Bis 100%, kondensierend
Lagerung	5 bis 95%, nicht kondensierend	
Transport	5 bis 95%, nicht kondensierend	
Mechanische Eigenschaften		
Anmerkung	Feldklemme 1x X20TB12 gesondert bestellen Busmodul 1x X20BM11 gesondert bestellen	Feldklemme 1x X20TB12 gesondert bestellen Busmodul 1x X20cBM11 gesondert bestellen
Rastermaß	12,5 $^{+0,2}$ mm	

Tabelle 2: X20AO2438, X20cAO2438 - Technische Daten

1) Bezogen auf den aktuellen Ausgabewert.
2) Bezogen auf den jeweiligen Ausgabebereich.
3) Laständerung von 1 Ω → 600 Ω, ohmsch
4) Bezogen auf den gesamten Ausgabebereich.

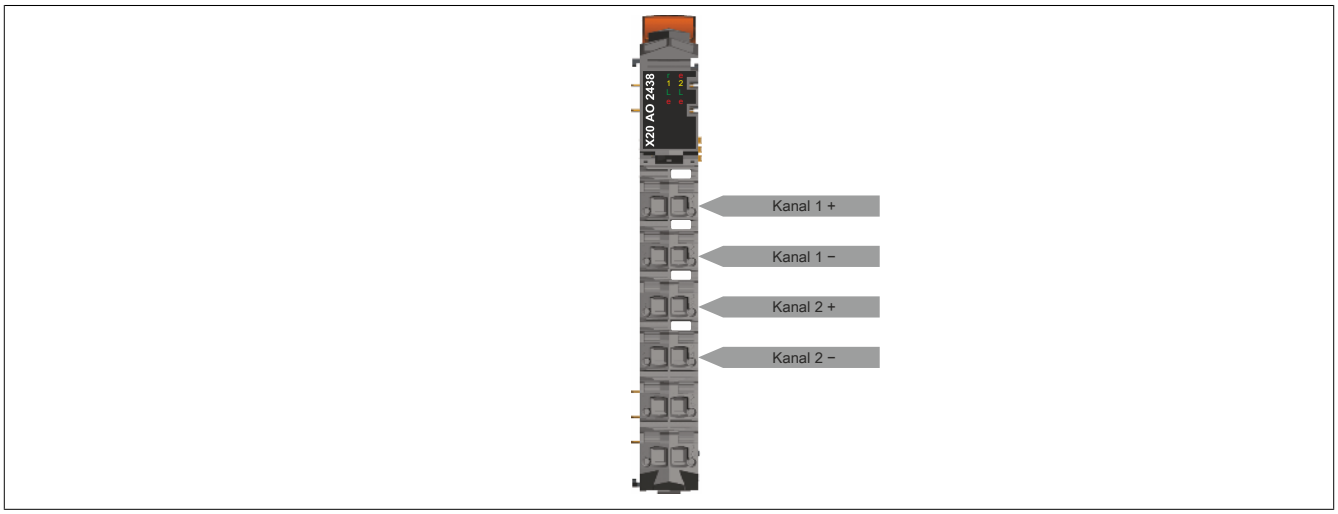
5 Status-LEDs

Für die Beschreibung der verschiedenen Betriebsmodi siehe X20 System Anwenderhandbuch, Abschnitt "Zusätzliche Informationen - Diagnose-LEDs".

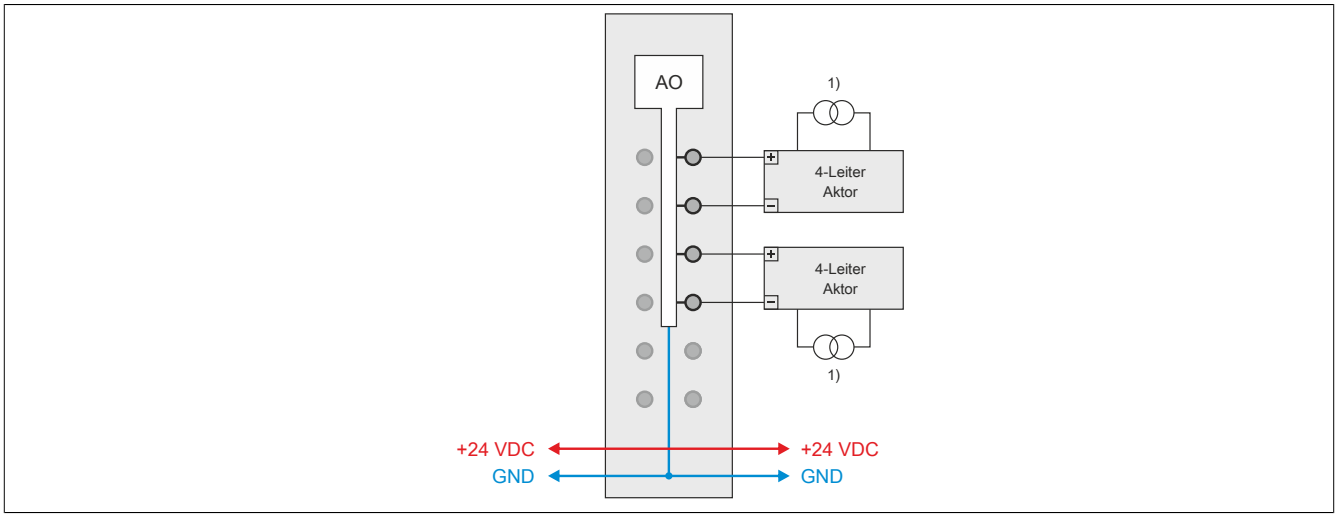
Abbildung	LED	Farbe	Status	Beschreibung
	**Betriebszustand**			
	r	Grün	Aus	Modul nicht versorgt
			Single Flash	Modus UNLINK
			Double Flash	Modus BOOT (während Firmware-Update)[1]
			Schnell blinkend	Modus SYNC
			Langsam blinkend	Modus PREOPERATIONAL
			Ein	Modus RUN
			Flackernd (ca. 10 Hz)	Modul befindet sich im OSP-Zustand
	Modulstatus			
	e	Rot	Aus	Modul nicht versorgt oder alles in Ordnung
			Single Flash	Es ist ein Wandlerfehler aufgetreten. Dieser Status wird zusätzlich zum Double Flash der Kanal-LED des fehlerhaften Analogausgangs ausgegeben.
			Ein	Fehler- oder Resetzustand
	Analogausgang			
	1 - 2	Orange	Aus	Zeigt einen der folgenden Fälle an: • Modul nicht versorgt • Kanal deaktiviert
			Single Flash	Drahtbruch
			Double Flash	Es ist ein Wandlerfehler aufgetreten. Zusätzlich wird an der roten Modulstatus-LED "e" ein Single Flash ausgegeben.
			Ein	Der Digital-/Analogwandler läuft, Wert ist in Ordnung
	HART Link			
	L	Grün	Aus	Zeigt einen der folgenden Fälle an: • Modul nicht versorgt • HART für den jeweiligen Kanal deaktiviert
			Flackernd	Bei aktivem Trägersignal (DCD oder RTS)
	HART Fehler			
	e	Rot	Aus	Zeigt einen der folgenden Fälle an: • Kommunikation läuft fehlerfrei • Modul nicht versorgt • HART für den jeweiligen Kanal deaktiviert
			Ein	Kommunikationsfehler

1) Je nach Konfiguration kann ein Firmware-Update bis zu mehreren Minuten benötigen.

6 Anschlussbelegung



7 Anschlussbeispiel

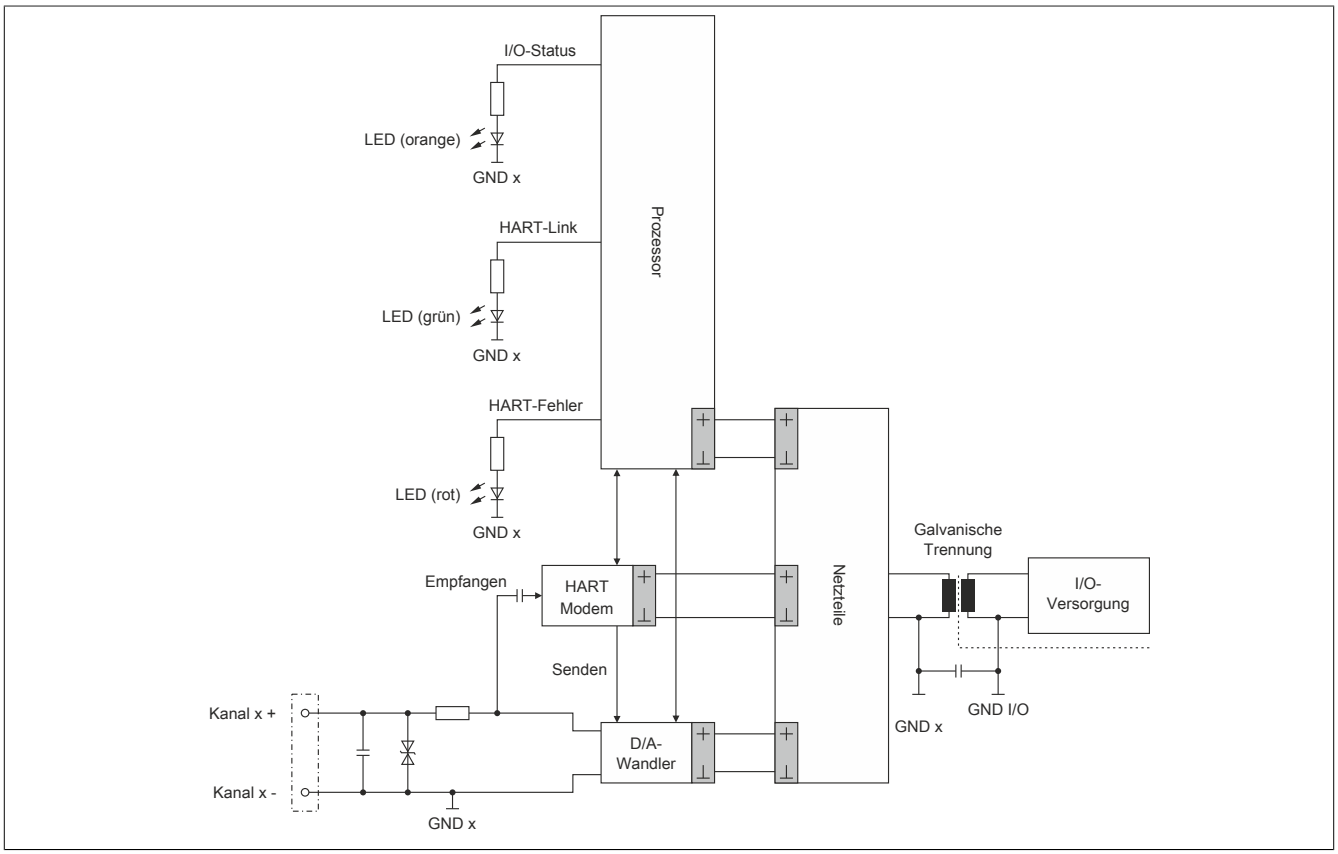


1) Mit externer Versorgung.

8 OSP-Hardwarevoraussetzungen

Um den OSP-Modus sinnvoll einzusetzen, sollte beim Aufbau der Applikation sichergestellt werden, dass die Energieversorgung des Ausgangsmoduls und der CPU voneinander unabhängig gestaltet sind.

9 Ausgangsschema



10 Betrieb

10.1 Derating

Um einen problemlosen Betrieb zu gewährleisten, sind die nachfolgend angeführten Deratings zu beachten.

Waagrechte Einbaulage

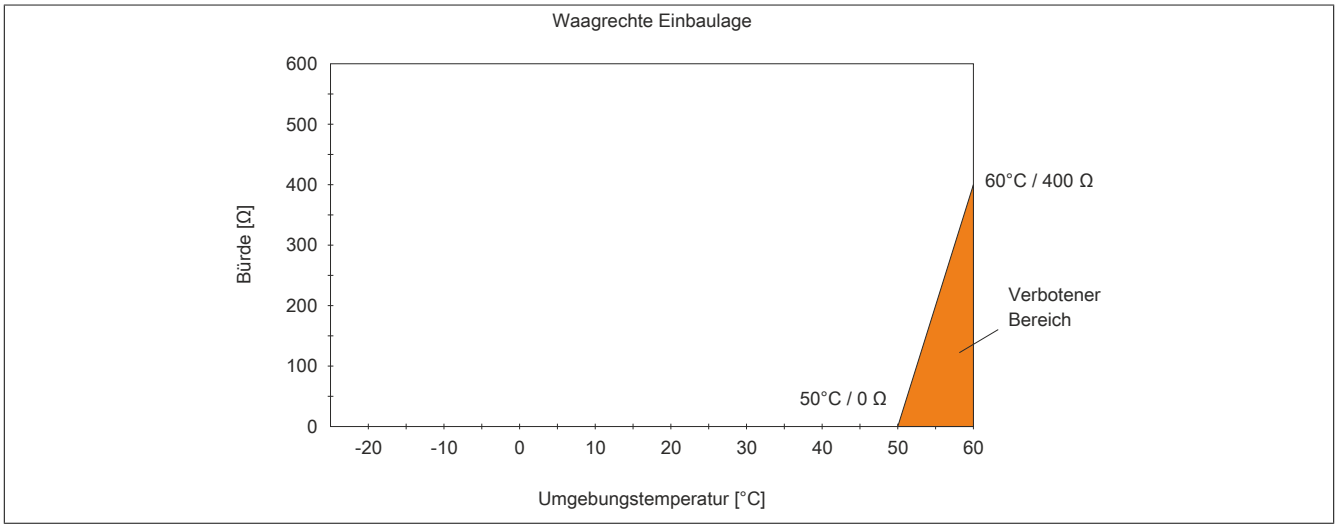


Abbildung 1: Derating der Bürde bei waagrechter Einbaulage

Senkrechte Einbaulage

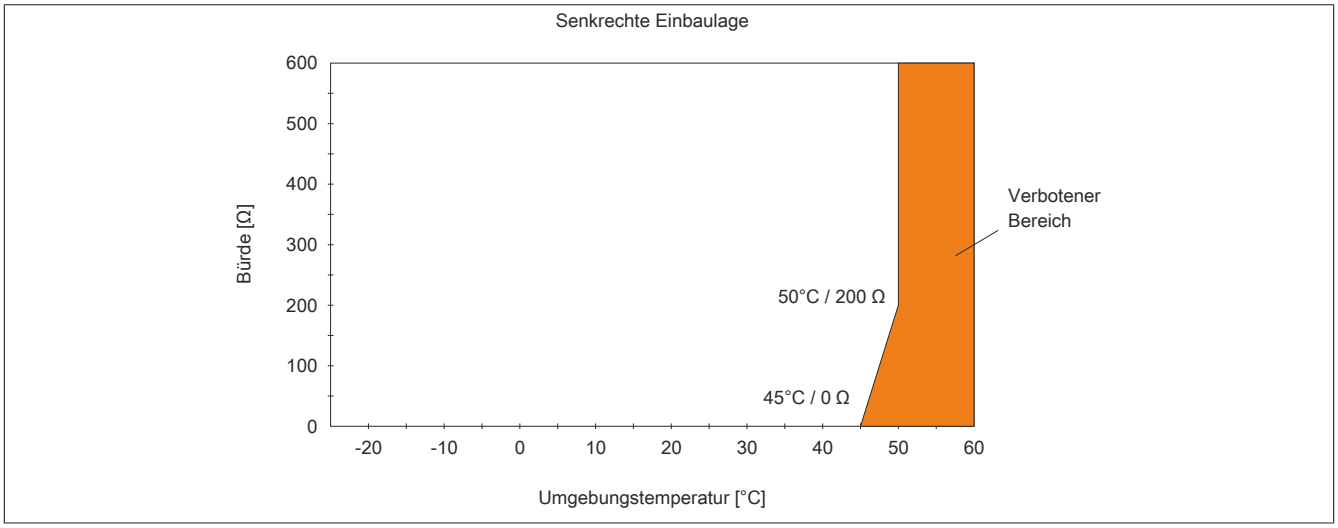


Abbildung 2: Derating der Bürde bei senkrechter Einbaulage

10.2 Verwendung nach X20IF1091-1

Wird dieses Modul nach dem X2X Link Modul X20IF1091-1 betrieben, kann es zu Verzögerungen bei der Flatstream-Übertragung kommen. Für genauere Informationen siehe X20IF1091-1, Abschnitt "Datenübertragung am Flatstream".

10.3 HART-Kommunikationsstandard

Das Modul unterstützt den HART-Kommunikationsstandard für Datenübertragung, Parametrierung und Diagnose. Die HART-Norm bezieht sich auf den Strombereich 4 bis 20 mA. Zu beachten ist, dass die Bürde nicht unter 230 Ω sinken darf.

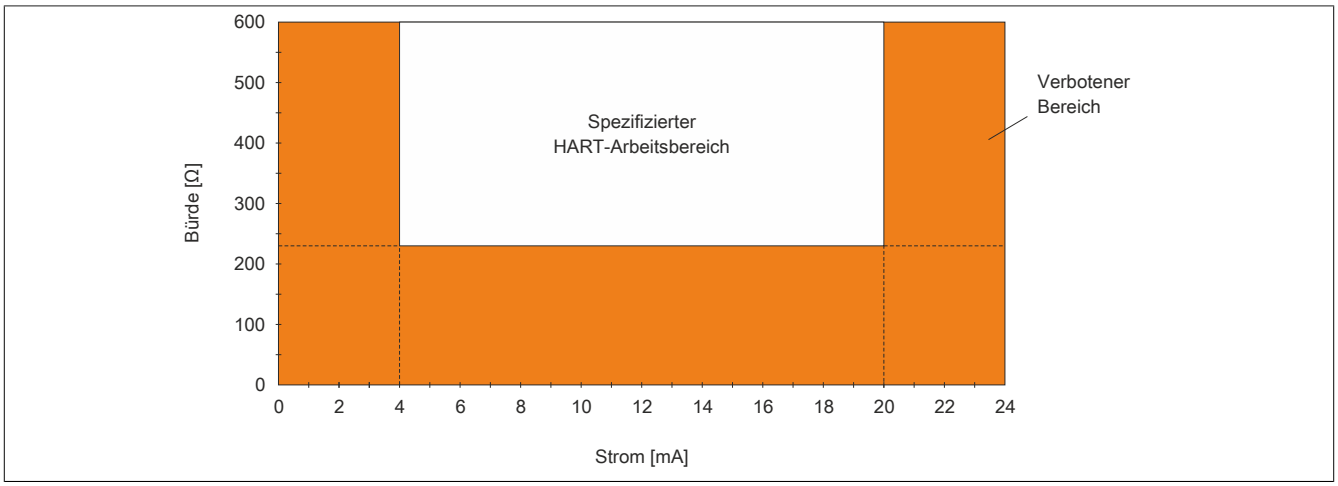


Abbildung 3: Spezifizierter HART-Arbeitsbereich

Vom Modul werden noch die beiden Strombereiche 0 bis 20 mA und 0 bis 24 mA unterstützt. Auch in diesen Bereichen kann die HART-Kommunikation verwendet werden. Es muss allerdings sichergestellt sein, dass sich der Ausgangsstrom im spezifizierten HART-Arbeitsbereich befindet.

11 Registerbeschreibung

11.1 Allgemeine Datenpunkte

Neben den in der Registerbeschreibung beschriebenen Registern verfügt das Modul über zusätzliche allgemeine Datenpunkte. Diese sind nicht modulspezifisch, sondern enthalten allgemeine Informationen wie z. B. Seriennummer und Hardware-Variante.

Die allgemeinen Datenpunkte sind im X20 System Anwenderhandbuch, Abschnitt "Zusätzliche Informationen - Allgemeine Datenpunkte" beschrieben.

11.2 Funktionsmodell 0 - Standard

Register	Name	Datentyp	Lesen		Schreiben	
			Zyklisch	Azyklisch	Zyklisch	Azyklisch
Analogsignal - Konfiguration						
386 394	AnalogMode01 AnalogMode02	UINT				•
390 398	DACSlewrate01 DACSlewrate02	UINT				•
Analogsignal - Kommunikation						
0 2	AnalogOutput01 AnalogOutput02	(U)INT			•	
30 31	AnalogStatus01 AnalogStatus02	USINT	•			
	OpenLineAnalogOutput01 bzw. OpenLineAnalogOutput02	Bit 2				
	ConversionErrorAnalogOutput01 bzw. ConversionErrorAnalogOutput02	Bit 3				
	IoSuppErrorAnalogOutput01 bzw. IoSuppErrorAnalogOutput02	Bit 7				
HART - Konfiguration						
1537 1665	HartNodeCnt_1 HartNodeCnt_2	USINT				•
1539 1667	HartMode_1 HartMode_2	USINT				•
1541 1669	HartBurstNode_1 HartBurtNode_2	USINT				•
HART - Erweiterte Konfiguration						
1558 1668	HartNodeDisable_1 HartNodeDisable_2	UINT				•
1546 1674	HartProtTimeOut_1 HartProtTimeOut_2	UINT				•
1550 1678	HartProtRetry_1 HartProtRetry_2	UINT				•
1554 1682	HartPreamble_1 HartPreamble_2	UINT				•
HART - Kommunikation (P2P)						
612 + Index*24 1124 + Index*24	PvInput01_N (Index N = 01 bis 04) PvInput02_N (Index N = 01 bis 04)	REAL	•	•[1)]		
617 + Index*24 1129 + Index*24	PvUnit01_N (Index N = 01 bis 04) PvUnit02_N (Index N = 01 bis 04)	USINT	•	•[1)]		
628 1140	PvSampleTime01 PvSampleTime02	DINT	•	•[1)]		
626 1138	PvSampleTime01 PvSampleTime02	INT	•			
566 1078	PvNodeComStatus01 PvNodeComStatus02	UINT		•		
HART - Kommunikation (multidrop)						
612 + Index*24 1124 + Index*24	PvInput01_N (Index N = 01 bis 15) PvInput02_N (Index N = 01 bis 15)	REAL	•	•[1)]		
617 + Index*24 1129 + Index*24	PvUnit01_N (Index N = 01 bis 15) PvUnit02_N (Index N = 01 bis 15)	USINT	•	•[1)]		
604 + Index*24 1116 + Index*24	PvSampleTime01_N (Index N = 01 bis 15) PvSampleTime02_N (Index N = 01 bis 15)	DINT	•	•[1)]		
602 + Index*24 1114 + Index*24	PvSampleTime01_N (Index N = 01 bis 15) PvSampleTime02_N (Index N = 01 bis 15)	INT	•			
562 + Index*4 1074 + Index*4	PvNodeComStatus01_N (Index N = 01 bis 15) PvNodeComStatus02_N (Index N = 01 bis 15)	UINT		•		
HART - Erweiterte Kommunikation						
522 1034	PvCountHartRequest01 PvCountHartRequest02	UINT	•			
530 1042	PvCountHartTimeout01 PvCountHartTimeout02	UINT	•			
538 1050	PvCountHartRxError01 PvCountHartRxError02	UINT	•			
546 1058	PvCountHartFrameError01 PvCountHartFrameError02	UINT	•			

Register	Name	Datentyp	Lesen		Schreiben	
			Zyklisch	Azyklisch	Zyklisch	Azyklisch
554 1066	PvNodeFound01 PvNodeFound02	UINT	•			
558 1070	PvNodeError01 PvNodeError02	UINT	•			
Flatstream - Konfiguration						
1793	OutputMTU	USINT				•
1795	InputMTU	USINT				•
1797	FlatstreamMode	USINT				•
1799	Forward	USINT				•
1802	ForwardDelay	UINT				•
Flatstream- Kommunikation						
1857	InputSequence	USINT	•			
1857 + Index*2	RxByteN (Index N = 1 bis 15)	USINT	•			
1889	OutputSequence	USINT			•	
1889 + Index*2	TxByteN (Index N = 1 bis 15)	USINT			•	

1) Diese HART-Register sind mehrfach definiert. Aus diesem Grund können sie azyklisch angesprochen werden, wenn sie nicht im zyklischen Teil der X2X Übertragung angemeldet werden.

11.3 Funktionsmodell 1 - OSP

Register	Name	Datentyp	Lesen		Schreiben	
			Zyklisch	Azyklisch	Zyklisch	Azyklisch
Analogsignal - Konfiguration						
386 394	AnalogMode01 AnalogMode02	UINT				•
390 398	DACSlewrate01 DACSlewrate02	UINT				•
Analogsignal - Kommunikation						
0 2	AnalogOutput01 AnalogOutput02	(U)INT			•	
30 31	AnalogStatus01 AnalogStatus02	USINT	•			
	OpenLineAnalogOutput01 bzw. OpenLineAnalogOutput02	Bit 2				
	ConversionErrorAnalogOutput01 bzw. ConversionErrorAnalogOutput02	Bit 3				
	IoSuppErrorAnalogOutput01 bzw. IoSuppErrorAnalogOutput02	Bit 7				
HART - Konfiguration						
1537 1665	HartNodeCnt_1 HartNodeCnt_2	USINT				•
1539 1667	HartMode_1 HartMode_2	USINT				•
1541 1669	HartBurstNode_1 HartBurtNode_2	USINT				•
HART - Erweiterte Konfiguration						
1558 1668	HartNodeDisable_1 HartNodeDisable_2	UINT				•
1546 1674	HartProtTimeOut_1 HartProtTimeOut_2	UINT				•
1550 1678	HartProtRetry_1 HartProtRetry_2	UINT				•
1554 1682	HartPreamble_1 HartPreamble_2	UINT				•
HART - Kommunikation (P2P)						
612 + Index*24 1124 + Index*24	PvInput01_N (Index N = 01 bis 04) PvInput02_N (Index N = 01 bis 04)	REAL	•	•[1)]		
617 + Index*24 1129 + Index*24	PvUnit01_N (Index N = 01 bis 04) PvUnit02_N (Index N = 01 bis 04)	USINT	•	•[1)]		
628 1140	PvSampleTime01 PvSampleTime02	DINT	•	•[1)]		
626 1138	PvSampleTime01 PvSampleTime02	INT	•			
566 1078	PvNodeComStatus01 PvNodeComStatus02	UINT		•		
HART - Kommunikation (multidrop)						
612 + Index*24 1124 + Index*24	PvInput01_N (Index N = 01 bis 15) PvInput02_N (Index N = 01 bis 15)	REAL	•	•[1)]		
617 + Index*24 1129 + Index*24	PvUnit01_N (Index N = 01 bis 15) PvUnit02_N (Index N = 01 bis 15)	USINT	•	•[1)]		
604 + Index*24 1116 + Index*24	PvSampleTime01_N (Index N = 01 bis 15) PvSampleTime02_N (Index N = 01 bis 15)	DINT	•	•[1)]		
602 + Index*24 1114 + Index*24	PvSampleTime01_N (Index N = 01 bis 15) PvSampleTime02_N (Index N = 01 bis 15)	INT	•			
562 + Index*4 1074 + Index*4	PvNodeComStatus01_N (Index N = 01 bis 15) PvNodeComStatus02_N (Index N = 01 bis 15)	UINT		•		
HART - Erweiterte Kommunikation						

Register	Name	Datentyp	Lesen		Schreiben	
			Zyklisch	Azyklisch	Zyklisch	Azyklisch
522 1034	PvCountHartRequest01 PvCountHartRequest02	UINT	•			
530 1042	PvCountHartTimeout01 PvCountHartTimeout02	UINT	•			
538 1050	PvCountHartRxError01 PvCountHartRxError02	UINT	•			
546 1058	PvCountHartFrameError01 PvCountHartFrameError02	UINT	•			
554 1066	PvNodeFound01 PvNodeFound02	UINT	•			
558 1070	PvNodeError01 PvNodeError02	UINT	•			
Flatstream - Konfiguration						
1793	OutputMTU	USINT				•
1795	InputMTU	USINT				•
1797	FlatstreamMode	USINT				•
1799	Forward	USINT				•
1802	ForwardDelay	UINT				•
Flatstream - Kommunikation						
1857	InputSequence	USINT	•			
1857 + Index*2	RxByteN (Index N = 1 bis 15)	USINT	•			
1889	OutputSequence	USINT			•	
1889 + Index*2	TxByteN (Index N = 1 bis 15)	USINT			•	
Das Funktionsmodell OSP						
32	OSPComByte	USINT			•	
	OSPValid	Bit 0				
401 403	CfgOSPMode01 CfgOSPMode02	USINT				•
34 36	CfgOSPValue01 CfgOSPValue02	INT				•

1) Diese HART-Register sind mehrfach definiert. Aus diesem Grund können sie azyklisch angesprochen werden, wenn sie nicht im zyklischen Teil der X2X Übertragung angemeldet werden.

11.4 Funktionsmodell 254 - Bus Controller

Register	Offset[1)]	Name	Datentyp	Lesen		Schreiben	
				Zyklisch	Azyklisch	Zyklisch	Azyklisch
Analogsignal - Konfiguration							
386 394	- -	AnalogMode01 AnalogMode02	UINT				•
390 398	- -	DACSlewrate01 DACSlewrate02	UINT				•
Analogsignal - Kommunikation							
0 2	0 8	AnalogOutput01 AnalogOutput02	(U)INT			•	
30 31	- -	AnalogStatus01 AnalogStatus02	USINT		•		
		OpenLineAnalogOutput01 bzw. OpenLineAnalogOutput02	Bit 2				
		ConversionErrorAnalogOutput01 bzw. ConversionErrorAnalogOutput02	Bit 3				
		IoSuppErrorAnalogOutput01 bzw. IoSuppErrorAnalogOutput02	Bit 7				
HART - Konfiguration							
1537 1665	- -	HartNodeCnt_1 HartNodeCnt_2	USINT				•
1539 1667	- -	HartMode_1 HartMode_2	USINT				•
1541 1669	- -	HartBurstNode_1 HartBurtNode_2	USINT				•
HART - Erweiterte Konfiguration							
1558 1668	- -	HartNodeDisable_1 HartNodeDisable_2	UINT				•
1546 1674	- -	HartProtTimeOut_1 HartProtTimeOut_2	UINT				•
1550 1678	- -	HartProtRetry_1 HartProtRetry_2	UINT				•
1554 1682	- -	HartPreamble_1 HartPreamble_2	UINT				•
HART - Kommunikation (P2P)							
636 1148	4 12	PvInput01_01 PvInput02_01	REAL	•			
612 + Index*24 1124 + Index*24	- -	PvInput01_N (Index N = 02 bis 04) PvInput02_N (Index N = 02 bis 04)	REAL		•		
641 1153	2 10	PvUnit01_01 PvUnit02_01	USINT	•			
617 + Index*24 1129 + Index*24	- -	PvUnit01_N (Index N = 02 bis 04) PvUnit02_N (Index N = 02 bis 04)	USINT		•		
566 1078	- -	PvNodeComStatus01 PvNodeComStatus02	UINT		•		
HART - Kommunikation (multidrop)							
636 1148	4 12	PvInput01_01 PvInput02_01	REAL	•			
612 + Index*24 1124 + Index*24	- -	PvInput01_N (Index N = 02 bis 15) PvInput02_N (Index N = 02 bis 15)	REAL		•		
641 1153	2 10	PvUnit01_01 PvUnit02_01	USINT	•			
617 + Index*24 1129 + Index*24	- -	PvUnit01_N (Index N = 02 bis 15) PvUnit02_N (Index N = 02 bis 15)	USINT		•		
562 + Index*4 1074 + Index*4	- -	PvNodeComStatus01_N (Index N = 01 bis 15) PvNodeComStatus02_N (Index N = 01 bis 15)	UINT		•		
HART - Erweiterte Kommunikation							
522 1034	- -	PvCountHartRequest01 PvCountHartRequest02	UINT		•		
530 1042	- -	PvCountHartTimeout01 PvCountHartTimeout02	UINT		•		
538 1050	- -	PvCountHartRxError01 PvCountHartRxError02	UINT		•		
546 1058	- -	PvCountHartFrameError01 PvCountHartFrameError02	UINT		•		
554 1066	- -	PvNodeFound01 PvNodeFound02	UINT		•		
558 1070	- -	PvNodeError01 PvNodeError02	UINT		•		

1) Der Offset gibt an, wo das Register im CAN-Objekt angeordnet ist.

11.4.1 Verwendung des Moduls am Bus Controller

Das Funktionsmodell 254 "Bus Controller" wird defaultmäßig nur von nicht konfigurierbaren Bus Controllern verwendet. Alle anderen Bus Controller können, abhängig vom verwendeten Feldbus, andere Register und Funktionen verwenden.

Für Detailinformationen siehe X20 Anwenderhandbuch (ab Version 3.50), Abschnitt "Zusätzliche Informationen - Verwendung von I/O-Modulen am Bus Controller".

11.4.2 CAN-I/O Bus Controller

Das Modul belegt an CAN-I/O 2 analoge logische Steckplätze.

11.5 Analogsignal - Konfiguration

Das Modul verfügt über 2 unabhängige galvanisch getrennte Kanäle mit integrierten HART-Modems. Über beide Kanäle kann sowohl ein Analogsignal ausgegeben werden als auch eine HART-Kommunikation stattfinden. Alle dafür notwendigen Register sind doppelt aufgelegt, sodass die Kanäle unabhängig voneinander konfiguriert und betrieben werden können.

Die Stromausgänge (Standard: 4 bis 20 mA) können als herkömmliche Analogsignale genutzt werden. Die integrierten HART-Modems nutzen physikalisch dieselben Leitungen. Mit Hilfe zusätzlicher Signale höherer Frequenz können digitale Informationen aus dem Speicher des HART-Slaves abgerufen werden.

Je Kanal werden folgende Anschlussvarianten unterschieden

- Point-to-Point (Anschaltung eines HART-Knotens am Kanal):
  → Auswertung des Analogsignals
  und
  → Aufnahme von bis zu 4 HART-Informationen
- Multidrop (Anschaltung von bis zu 15 HART-Knoten am Kanal):
  → Aufnahme einer HART-Information je angeschlossenem Knoten

Spezifische Besonderheiten

- Kanalweise galvanische Trennung
- Bis zu 4 bzw. 15 HART-Eingangsvariablen pro Kanal
- Konfigurierbare Ausgangsrampe (DAC-Slewrate), um HART- und Analogsignal ohne Beeinträchtigung zu übertragen (Standard: 210 ms Fullscale)
- Selektierbare Fehlerstrategie (statischer Ersatzwert oder Erhalt des letzten zulässigen Wertes)
- Zyklisches "HART Status" Polling (HART Befehl 0), die erhaltene Statusinformation wird zur Kanaldiagnose bereitgestellt
- Kompatibel mit zusätzlichem Secondary Master im HART-Netzwerk (Modul fungiert als Primary Master)
- "HART Kommunikationsfehlerbit" (zeigt Abbruch der HART-Verbindung, falls Verbindungsaufbau zuvor erfolgreich)
- Optional: BURST-Mode für einen Knoten pro Kanal
- Optional: Zyklisches "HART Variablen" Polling (HART Befehl 3 oder 9)
- Optional: Flatstream-Funktionalität (Modul als Bridge für HART-Pakete)

11.5.1 AnalogMode

Name:
AnalogMode01bis AnalogMode02

Mit diesen Registern werden die Betriebsparameter vorgegeben, die das Modul für den dazugehörigen Kanal anwendet. Jeder Kanal muss separat aktiviert und konfiguriert werden.

Information:

Die Auswahl des Betriebsmodus "Skalierung 0 bis 20 mA (Auflösung 0 bis 65535)" bewirkt, dass das entsprechende "AnalogOutput"-Register intern nicht mehr als INT, sondern als UINT interpretiert wird. Für die Änderung des Datentyps ist ein Rebuild des gesamten Programms notwendig. Während der Laufzeit kann der Datentyp nicht verändert werden (z. B. über Bibliothek).

Datentyp	Werte	Bus Controller Default
UINT	Siehe Bitstruktur	33

Bitstruktur:

Bit	Bezeichnung	Wert	Information
0	Kanal	0	Deaktiviert
		1	Aktiviert (Bus Controller Default)
1	Prüfung - D/A-Wandler-Konfiguration, -Status	0	Aktiviert (Bus Controller Default)
		1	Deaktiviert
2 - 3	Reserviert	-	
4	Skalierung 0 bis 20 mA (Auflösung 0 bis 32767)	0	Deaktiviert
		1	Aktiviert
5	Skalierung 4 bis 20 mA (Auflösung 0 bis 32767)	0	Deaktiviert
		1	Aktiviert (Bus Controller Default)
6	Skalierung 0 bis 24 mA (Auflösung 0 bis 24000)	0	Deaktiviert
		1	Aktiviert
7	Skalierung 0 bis 20 mA (Auflösung 0 bis 65535)	0	Deaktiviert
		1	Aktiviert
8 - 15	Reserviert	-	

Information:

Die "AnalogMode"-Register bieten die Möglichkeit, die zyklische Prüfung der D/A-Wandler-Konfiguration zu umgehen. Um die Kommunikation zuverlässig zu gestalten, sollte diese Option nur genutzt werden, wenn keine HART-Kommunikation auf dem Kanal stattfindet.

11.5.2 DACSlewrate

Name:
DACSlewrate01 bis DACSlewrate02

Diese Register begrenzen die Änderungsgeschwindigkeit des Analogsignals. Auf diese Weise kann eine Art obere Grenzfrequenz definiert werden.

Es gilt die Formel: f(Analog) = f(Ausgabetakt) * zulässige Änderung / max. Δ(normierter Ausgabewert)

Für eine fehlerfreie Kommunikation muss sichergestellt sein, dass der Frequenzbereich des digitalen HART-Signals nicht vom analogen Ausgang beeinflusst wird. HART kommuniziert im Frequenzbereich von 950 bis 2500 Hz.

Beispiel (Standard): f(Analog) = 152440 Hz * 4 / (32767 - 0)
Schlussfolgerung: f(Analog) = **~20 Hz << 950 Hz** = f(HART)

Datentyp	Werte	Bus Controller Default
UINT	Siehe Bitstruktur	514

Bitstruktur:

Bit	Bezeichnung	Wert	Information
0 - 2	zulässige Änderung pro Takt	000	1 Bit
		001	2 Bit
		010	4 Bit (Bus Controller Default)
		011	8 Bit
		100	16 Bit
		101	32 Bit
		110	64 Bit
		111	128 Bit
3 - 7	Reserviert	-	
8 - 11	Ausgabetakt	0000	257730 Hz
		0001	198410 Hz
		0010	152440 Hz (Bus Controller Default)
		0011	131580 Hz
		0100	115740 Hz
		0101	69440 Hz
		0110	37590 Hz
		0111	25770 Hz
		1000	20160 Hz
		1001	16030 Hz
		1010	10290 Hz
		1011	8280 Hz
		1100	6900 Hz
		1101	5530 Hz
		1110	4240 Hz
		1111	3300 Hz
12 - 14	Reserviert	-	
15	Slewrate enable (Rampenfunktionalität)	0	Deaktiviert (undefiniertes Sprungverhalten)
		1	Aktiviert (definierte Übergänge)

11.6 Analogsignal - Kommunikation

Um das gewünschte Stromsignal (Standard: 4 bis 20 mA) auszugeben, muss dem Modul der normierte Ausgabewert (Standard: 0 bis 32767) vorgegeben werden. Auf diese Weise kann das X20AO2438 als herkömmliches Ausgangsmodul genutzt werden. Das integrierte HART-Modem nutzt physikalisch dieselbe Leitung. Mit Hilfe von Signalen höherer Frequenz kann das Modul mit dem HART-Slave kommunizieren und zusätzlich Informationen abrufen.

11.6.1 AnalogOutput

Name:
AnalogOutput01 bis AnalogOutput02

Über diese Register werden die normierten Ausgabewerte vorgegeben. Je nach Wahl der Skalierung (siehe "AnalogMode" auf Seite 14-Register) kann der Wertebereich und der Datentyp auf die Anforderungen der Applikation angepasst werden. Nach der Übermittlung eines zulässigen Wertes gibt das Modul den entsprechenden Strom aus.

> **Information:**
>
> **Der Wert "0" deaktiviert die Kanalstatus-LED.**

Datentyp	Werte
INT	0 bis 32767
Optional: UINT	0 bis 65535

11.6.2 AnalogStatus

Name:
AnalogStatus01 bis AnalogStatus02

Mit dem Statusregister erhält der Anwender die Rückmeldung, ob der jeweilige Kanal ordnungsgemäß arbeitet.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Bezeichnung	Wert	Information
0 - 1	Reserviert	-	
2	OpenLineAnalogOutput01, 02	0	Verdrahtung okay
		1	Drahtbruch
3	ConversionErrorAnalogOutput01, 02	0	Wandlertemperatur okay
		1	Wandlertemperatur zu hoch
4 - 6	Reserviert	-	
7	IoSuppErrorAnalogOutput01, 02	0	Modulversorgung okay
		1	Modulversorgung fehlerhaft

11.7 HART

HART (Highway Addressable Remote Transducer) ist ein Protokoll zur Kommunikation mit intelligenten Feldgeräten. Das Verfahren wurde konzipiert, um Infrastrukturen zur Übertragung analoger Signale effizienter zu nutzen. Die digitalen HART-Nachrichten werden per Frequenzumtastung (engl.: Frequency Shift Keying, FSK) auf das Analogsignal moduliert. Auf diese Weise kann HART dieselbe physikalische Leitung wie das Analogsignal nutzen, ohne die ursprüngliche Funktion zu beeinflussen.
HART-Slaves sind in der Lage verschiedene Prozessdaten eigenständig zu ermitteln und HART-konform aufzubereiten. Das Protokoll sieht vor, dass der Wert einer Prozessgröße und separat dessen Einheit und Status erfasst werden. Die Feldgeräte liefern in der Regel ihre Informationen nachdem der HART-Master sie anfordert. In neueren Revisionen ist auch die Übertragung von Konfigurationsdaten möglich.

Es werden 2 Arten von HART-Netzwerken unterschieden. Im *Point-to-Point*-Netzwerk wird nur ein Slave an einen HART-Master angeschlossen. Darin können das analoge und das HART-Signal über dieselbe Leitung übertragen werden. Zur Verwaltung mehrerer Slaves wird bei HART ein sogenanntes *Multidrop*-Netzwerk eingerichtet. Jedem HART-Slave wird eine Adresse zugewiesen, die ihn eindeutig beschreibt. Klassische analoge Signale können in Bussystemen nicht eindeutig zugeordnet werden. Deshalb sieht das HART-Protokoll bis einschließlich HART-Revision 5 keine analoge Informationsübertragung in Multidrop-Netzwerken vor.

Information:

Split-Range-Betrieb bei HART-AO-Modulen

Ab HART-Revision 6 werden Busteilnehmer, die ein Analogsignal nach dem Split-Range-Verfahren nutzen, gesondert beschrieben. Das HART-Protokoll sieht für diese Applikationen sowohl die Adressierung gemäß Multidrop als auch die Nutzung des analogen Signals vor.

Das Modul wurde auf der Grundlage der HART-Revision 5 entworfen. Zur Übertragung der Signale steht deshalb ausschließlich das einkanalige FSK-Verfahren zur Verfügung.

Da bei der Nutzung des Flatstream-Interfaces alle HART-Frames applikativ generiert und ausgewertet werden, ist es möglich auch Informationen auszulesen, die in späteren Revisionen spezifiziert wurden.

11.7.1 HART - Konfiguration

Die HART-Module sind analoge Module, die mit einem HART-Modem ausgestattet sind. Je Kanal kann ein separates HART-Netzwerk mit dem Modul als Primary Master verwaltet werden. Nach erfolgreicher Konfiguration werden die HART-Informationen im Modul zwischengespeichert und können im Anschluss von der SPS verwendet werden.

Bei der Konfiguration muss kanalweise die Anzahl der HART-Slaves vorgegeben werden.
Wenn nur ein Slave am HART-Kanal angeschlossen ist, handelt es sich um ein Point-to-Point-Netzwerk. Das Modul bereitet bis zu 4 Prozessvariablen des angeschlossenen Slaves auf.
Der Multidrop-Modus ermöglicht die Anbindung von bis zu 15 HART-Slaves. Je Slave wird die primäre Prozessvariable abgefragt.

11.7.1.1 HartNodeCnt

Name:
HartNodeCnt_1 bis HartNodeCnt_2

In diesen Registern wird dem Modul vorgegeben, wie viele HART-Slaves sich am jeweiligen Kanal befinden.

Information:

Falls sich an einem der HART-Kanäle kein Slave befindet, sollte in diesem Register der Wert „0“ vorgegeben werden. Auf diese Weise verkürzt sich die I/O-Updatezeit und überflüssige Fehlermeldungen werden vermieden.

Datentyp	Werte	Information	
USINT	0	HART-Kommunikation des Kanals deaktiviert	
	1	Point-to-Point	HART-Standard Kommunikation (Bus Controller Default)
	2 bis 15	Multidrop	Anzahl der HART-Slave Knoten

11.7.1.2 HartBurstNode

Name:
HartBurstNode_1 bis HartBurstNode_2

Neben der Art des Netzwerkes kann der Anwender zwischen 2 verschiedenen Kommunikationsverhalten wählen. Bei der herkömmlichen HART-Kommunikation werden die Prinzipien des Pollings angewendet. Das Modul fragt die Daten des HART-Slaves einzeln ab und bekommt vom Slave die entsprechende Information als Antwort. Wenn ein HART-Knoten in kurzen Zeitabständen abgefragt werden soll, kann der Anwender für einen Knoten je Kanal den Burst-Modus konfigurieren. Der Slave sendet in diesem Fall die Informationen dieses Knotens zyklisch, ohne eine erneute Aufforderung durch den Master.
In den "HartBurstNode"-Registern werden kanalweise die Knotennummern (short address) eingetragen, deren Informationen im Burst-Modus abgefragt werden sollen. Aktiviert wird der Burst-Modus über das Register "HartMode" auf Seite 18.

Datentyp	Werte	Information
USINT	0 bis 15	Point-to-Point; Bus Controller Default: 0

11.7.1.3 HartMode

Name:
HartMode_1 bis HartMode_2

Mit diesen Registern kann der Anwender das Kommunikationsverhalten der einzelnen HART-Kanäle konfigurieren. Im Regelfall werden die HART-Knoten einzeln abgefragt (Polling). Falls der Burst-Modus un-/erwünscht ist, kann er mit diesem Register gestoppt/gestartet werden.
Ein Burst-Knoten sendet seine Informationen nicht kontinuierlich, sondern getaktet. Aus diesem Grund ermöglicht der HART-Standard den parallelen Betrieb von Burst-Modus und Polling.

Information:

Bei Burst-Abfrage muss das Register "HartBurstNode" auf Seite 18 korrekt konfiguriert sein.

Datentyp	Werte	Bus Controller Default
UINT	Siehe Bitstruktur	0

Bitstruktur:

Bit	Bezeichnung	Wert	Information
0	Slave Polling-Modus	0	Polling-Modus aktiviert (Bus Controller Default)
		1	Polling-Modus ausgeschalten
1	Slave Burst-Modus starten	0	Keine Reaktion auf Burst (Bus Controller Default)
		1	Aktiviert Burst-Modus in Knoten "HartBurstNode" auf Seite 18
2	Slave Burst-Modus stoppen	0	Keine Reaktion auf Burst (Bus Controller Default)
		1	Deaktiviert Burst-Modus, falls vorhanden
3 - 7	Reserviert	-	

11.7.2 HART - Kommunikation

Nach Abschluss der Konfiguration werden die Informationen automatisch abgerufen und auf die Modulregister übertragen. Für jede Teilinformation ist ein separates Register im Modul implementiert. Die HART-Module sind für die Abfrage von maximal 15 Informationen pro Kanal konzipiert. Das Modul liest die Daten ein, speichert sie zwischen und stellt sie für den Abruf bereit. Beim Zugriff des X2X Masters auf die Modulregister, ist es unerheblich, ob die HART-Daten aus einem Point-to-Point- oder Multidrop-Netzwerk stammen.

Übersicht der modulinternen Zuordnung

	Point-to-Point-Netzwerk (1 HART-Slave)	*Multidrop-Netzwerk (2 bis 15 HART-Slaves)*
(Pv)Input_01	Primäre Information aus HART-Knoten 1	Primäre Information aus HART-Knoten 1
(Pv)Input_02	Sekundäre Information aus HART-Knoten 1	Primäre Information aus HART-Knoten 2
…	…	…
(Pv)Input_04	Quartäre Information aus HART-Knoten 1	Primäre Information aus HART-Knoten 4
(Pv)Input_05	Reserviert	Primäre Information aus HART-Knoten 5
…	…	…
(Pv)Input_15	Reserviert	Primäre Information aus HART-Knoten 15

Die HART-Spezifikation sieht vor, dass Informationen aus einem HART-Knoten in verschiedene Teile untergliedert sind. Der Wert einer Prozessvariablen wird auf das jeweilige Register "PvInput" auf Seite 20 gespeichert und ist gemäß HART-Spezifikation 4 Bytes (REAL) groß. Durch die Längenlimitierung von 30 Bytes am X2X Link ergeben sich Einschränkungen in der Anzahl der möglichen zyklischen Variablen. Es wird empfohlen max. 2 Register "PvInput" auf Seite 20 zyklisch zum X2X Master zu übertragen. Alle weiteren Informationen sollten alternativ ausgelesen werden. Um auf HART-Informationen zuzugreifen, kann der Anwender zwischen folgenden Methoden wählen:

- **Azyklisch** - Bei Verwendung der AsIOAcc-Bibliothek werden die Informationen nur bei Bedarf azyklisch abgefragt, d. h. die Kommunikation kann an den Programmablauf des X2X Masters angepasst werden. Auf diesem Weg können trotz Längenlimitierung am X2X Link alle benötigten Modulregister abgefragt werden. Diese Art des Informationsaustausches ist nicht echtzeitfähig.
- **Zyklisch** - Zyklisch konfigurierte Datenpunkte werden pro Buszyklus einmal gelesen. Diese Vorgehensweise ermöglicht einen echtzeitfähigen Informationsaustausch zwischen Modul und X2X Master. Allerdings können wegen der Längenlimitierung möglicherweise nicht alle Daten zyklisch abgefragt werden.
- **Multiplexed** - Für die Übertragung der HART-Datenpunkte der I/O-Zuordnung kann ein Runtime-Treiber genutzt werden. In diesem Fall werden die HART-Prozessdaten (time multiplexed) abwechselnd übermittelt. Die Kommunikation bleibt weiterhin echtzeitfähig. Es werden allerdings mehrere Buszyklen benötigt, um alle Datenpunkte zu aktualisieren.

  > **Information:**
  >
  > **Bei Verwendung des Moduls nach einem Bus Controller kann dieser Modus nicht benutzt werden.**
  >
  > **Die "multiplxed" Datenübertragung wird ausschließlich für HART-Datenpunkte genutzt.**
  >
  > **Die Informationen der analogen Ein-/Ausgänge werden stets zyklisch (siehe oben) übermittelt.**

- **Flatstream** - Die HART-Module sind mit einer Flatstream-Schnittstelle ausgestattet. Bei der Flatstream-Kommunikation wird das Modul als Bridge zwischen dem X2X Master und HART-Slave genutzt, d. h. der X2X Master kommuniziert direkt mit dem HART-Slave (siehe "Flatstreamkommunikation" auf Seite 24).
  Die Flatstreamkommunikation ist ebenfalls nicht echtzeitfähig. Sie ermöglicht einen unbeschränkten Zugang zum HART-Slave. Der Anwender benötigt ausreichend Kenntnisse über den Befehlssatz des HART-Protokolls und die Fähigkeiten des entsprechenden HART-Slaves.

11.7.2.1 PvInput

Name:
PvInput01_01 bis PvInput01_15
PvInput02_01 bis PvInput02_15

Diese Register liefern den aktuellen Wert der ausgelesenen Prozessvariablen.

Information:

Diese Register sind vom Datentyp REAL, daher kommt es bei zyklischer Verwendung schneller zur Belegung der am X2X Link verfügbaren Bytes. Falls die Informationen von mehreren Slave-Knoten nötig sind, muss die azyklische Abfrage oder der Flatstream genutzt werden.

Datentyp	Werte	Information
REAL	IEEE745 SPF	32 Bit Datentyp bei gültigem Wert
	0x7FA00000	NotANumber (NAN) bei ungültigem Wert

11.7.2.2 PvUnit

Name:
PvUnit01_01 bis PvUnit01_15
PvUnit02_01 bis PvUnit02_15

Diese Register liefern einen HART-spezifischen Code, um die Einheit des Messwertes zu beschreiben. Die Codierung wird in der HART-Spezifikation genau festgelegt.

Datentyp	Werte
USINT	Siehe Beschreibung des HART-Slaves Siehe HART-Spezifikation

11.7.2.3 PvSampleTime

Name:
PvSampleTime01 bis PvSampleTime02
PvSampleTime01_01 bis PvSampleTime01_15
PvSampleTime02_01 bis PvSampleTime02_15

Diese Register liefern den Zeitstempel, zu dem das aktuelle Abbild des Kanals vom Modul eingelesen wurde. Die Werte werden als vorzeichenbehafteter 2- oder 4-Byte Wert bereitgestellt.

Für weitere Informationen zu NetTime und Zeitstempel siehe "NetTime Technology" auf Seite 55.

Datentyp	Werte	Information
INT	-32768 bis 32767	NetTime-Zeitstempel des aktuellen Eingangswertes in µs
DINT	-2.147.483.648 bis 2.147.483.647	NetTime-Zeitstempel des aktuellen Eingangswertes in µs

Es handelt sich dabei um den Zeitpunkt, zu dem der HART-Master die Slave-Antwort empfängt. Auf diese Weise kann geprüft werden, ob seit dem letzten X2X Zyklus neue HART-Informationen eingelesen wurden.

Information:

Die Zykluszeiten im HART-Netzwerk sind verhältnismäßig groß, sodass anhand dieser Information der Zeitpunkt der Messwerterfassung nicht zuverlässig ermittelt werden kann.

11.7.2.4 PvNodeComStatus

Name:
PvNodeComStatus01 bis PvNodeComStatus02
PvNodeComStatus01_01 bis PvNodeComStatus01_15
PvNodeComStatus02_01 bis PvNodeComStatus02_15

Diese Register geben Auskunft, ob ein eingelesener Wert gültig ist. Gemäß HART-Spezifikation besteht ein solches Statusregister aus 2 Teilen. Im High-Byte wird der Rresponse code" und im Low-Byte der "field device status" abgelegt. Der aktuelle Status einer eingelesen Prozessvariablen kann auf diese Weise überprüft werden.

Diese Register können überprüft werden, bevor eine zwischengespeicherte Prozessinformation weiterverarbeitet wird. Wenn der aktuelle Wert gleich 0x0000 ist, wurden keine Fehler bei der HART-Übertragung erkannt und die Information des geprüften Knotens kann verwendet werden. Falls ein anderer Wert vorhanden ist, sollte die Situation im HART-Netzwerk geprüft werden. Zu diesem Zweck können z. B. die Zusatzregister verwendet werden.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Bezeichnung	Wert	Information
0	Qualität - Knoteninformation 2 bis n	0	Digitaler Messwert okay
		1	Messwert außerhalb des zuläss. Arbeitsbereichs
1	Qualität - Knoteninformation 1	0	Digitaler Messwert okay
		1	Messwert außerhalb des zuläss. Arbeitsbereichs
2	Grenzwertverletzung	0	Parameter okay
		1	Unzulässige(r) Messwert(e) oder Geberversorgung
3	statisches Analogsignal	0	Gewöhnliche Wertänderungen/-schwankungen
		1	Konstanter Analogwert an Slave von Knoten 1
4	zusätzl. Stausinformationen (nur von wenigen Slaves unterstützt)	0	Nicht vorhanden
		1	Abrufbar (nur per Flatstream - Kommando #48)
5	Neustart	0	Normalbetrieb
		1	Feldgerät startet neu
6	Geräte-ID	0	Unverändert
		1	Verändert
7	Gerätefehler	0	Messwert okay
		1	zweifelhafte Messwertinformation
8 - 14	Antwort-Code, falls relevant	x	Siehe **HART-spezifischer Antwortcode**
15	Fehler - Kommunikation	0	Kommunikation fehlerfrei (Antwortcode irrelevant)
		1	Kommunikation fehlerhaft (Antwortcode relevant)

HART-spezifischer Antwortcode (Auszug):

0x82 … Überlauf des Empfangspuffers 0x88 … Prüfsumme inkorrekt 0x90 … Protokollaufbau fehlerhaft 0xA0 … Überlauf 0xC0 … Parität unzulässig 0xFF … Zeitüberschreitung	Tritt bei der HART-Kommunikation ein Fehler auf, wird der Antwortcode geschrieben. Dabei wird stets Bit 15 gesetzt.

Abruf der eingelesenen Informationen

Nachdem die Knotendaten erfolgreich auf die Modulregister übertragen wurden, können die Informationen vom Modul abgerufen werden. Für jede Teilinformation wurden separate Register im Modul implementiert.

11.7.2.5 PvCountHartRequest

Name:
PvCountHartRequest01 bis PvCountHartRequest02

Diese Register werden erhöht, sobald das Modul eine Mitteilung auf dem entsprechenden Kanal senden will.

Datentyp	Werte
UDINT	0 bis 4.294.967.295

11.7.2.6 PvCountHartTimeout

Name:
PvCountHartTimeout01 bis PvCountHartTimeout02

Diese Register werden erhöht, wenn der Slave die maximal zulässige Zeit überschreitet, um auf eine Anfrage des Moduls zu reagieren.

Datentyp	Werte
UDINT	0 bis 4.294.967.295

11.7.2.7 PvCountHartRxError

Name:
PvCountHartRxError01 bis PvCountHartRxError02

Diese Register werden erhöht, wenn Kommunikationsfehler auf Schicht 1 des OSI-Modells auftreten (z. B. Übertragungsfehler laut Paritätsbit).

Datentyp	Werte
UDINT	0 bis 4.294.967.295

11.7.2.8 PvCountHartFrameError

Name:
PvCountHartFrameError01 bis PvCountHartFrameError02

Diese Register werden erhöht, wenn Kommunikationsfehler auf Schicht 2 des OSI-Modells auftreten (z. B. fehlerhafter Aufbau des Telegramms).

Datentyp	Werte
UDINT	0 bis 4.294.967.295

11.7.2.9 PvNodeFound

Name:
PvNodeFound01 bis PvNodeFound02

Diese Register geben Auskunft, welche Knoten am jeweiligen Kanal erkannt wurden (Slave wird erfolgreich identifiziert).

Datentyp	Werte
UINT	Siehe Bitstruktur

Bitstruktur:

Bit	Bezeichnung	Wert	Information
0	Knoten 0 (Standard-Modus) Knoten 1 (Multidrop-Modus)	0	Nicht als zulässig erkannt
		1	Als zulässig erkannt
1	Knoten 2 (Multidrop-Modus)	0	Nicht als zulässig erkannt
		1	Als zulässig erkannt
...		...	
13	Knoten 14 (Multidrop-Modus)	0	Nicht als zulässig erkannt
		1	Als zulässig erkannt
14	Knoten 15 (Multidrop-Modus)	0	Nicht als zulässig erkannt
		1	Als zulässig erkannt
15	Reserviert	-	

11.7.2.10 PvNodeError

Name:
PvNodeError01 bis PvNodeError02

Diese Register beinhalten die HART-Kommunikationsfehlerbits. Diese Bits werden gesetzt, wenn die Verbindung zu einem Knoten erfolgreich aufgebaut wurde und im Anschluss dieser Knoten nicht mehr korrekt antwortet (z. B. HART-Slave überschreitet konfigurierte Zeitüberschreitung bzw. konfiguriert Anzahl der Versuche).

Datentyp	Werte
UINT	Siehe Bitstruktur

Bitstruktur:

Bit	Bezeichnung	Wert	Information
0	Knoten 0 (Standard-Modus) Knoten 1 (Multidrop-Modus)	0	Als fehlerfrei erkannt
		1	Fehlerhaft
1	Knoten 2 (Multidrop-Modus)	0	Als fehlerfrei erkannt
		1	Fehlerhaft
...		...	
13	Knoten 14 (Multidrop-Modus)	0	Als fehlerfrei erkannt
		1	Fehlerhaft
14	Knoten 15 (Multidrop-Modus)	0	Als fehlerfrei erkannt
		1	Fehlerhaft
15	Reserviert	-	

11.7.3 Erweiterte Konfiguration

Die zusätzlichen Konfigurationsregister sind beim Start des Moduls mit Werten vorbelegt. In vielen Systemen muss der Anwender keine Anpassungen an ihnen vornehmen. Die Registerwerte sollten nur geändert werden, falls die Kommunikation im HART-Netzwerk nicht zufriedenstellend abläuft.

11.7.3.1 HartNodeDisable

Name:
HartNodeDisable_1 bis HartNodeDisable_2

Diese Register sind z. B. für Wartungsarbeiten vorgesehen. Sie ermöglichen die Abschaltung von projektierten HART-Knoten, um Fehlermeldungen zeitweise zu unterdrücken. Im regulären Betrieb müssen die projektierten Knoten aktiv geschaltet sein, um einen reibungslosen Ablauf zu gewährleisten.

Datentyp	Werte	Bus Controller Default
UINT	Siehe Bitstruktur	0x3FFF

Bitstruktur:

Bit	Bezeichnung	Wert	Information
0	Knoten 0 (Standard-Modus) Knoten 1 (Multidrop-Modus)	0	Aktiviert (Bus Controller Default)
		1	Deaktiviert
1	Knoten 2 (Multidrop-Modus)	0	Aktiviert
		1	Deaktiviert (Bus Controller Default)
...		...	
13	Knoten 14 (Multidrop-Modus)	0	Aktiviert
		1	Deaktiviert (Bus Controller Default)
14	Knoten 15 (Multidrop-Modus)	0	Aktiviert
		1	Deaktiviert (Bus Controller Default)
15	Reserviert	-	

11.7.3.2 HartProtTimeOut

Name:
HartProtTimeOut_1 bis HartProtTimeOut_2

In diesen Registern wird die Zeitspanne festgelegt, nach der ein Slave spätestens reagieren muss, um eine gültige Antwort zu geben.

Datentyp	Werte [ms]	Information
UINT	0 bis 65535	Bus Controller Default: 256 [ms]

11.7.3.3 HartProtRetry

Name:
HartProtRetry_1 bis HartProtRetry_2

Diese Register bestimmen, wie oft der Master eine Anfrage wiederholt, wenn er eine ungültige oder keine Antworten erhält.

Datentyp	Werte	Information
UINT	0 bis 65535	Bus Controller Default: 3 Versuche

11.7.3.4 HartPreamble

Name:
HartPreamble_1 bis HartPreamble_2

In diesen Registern kann die Länge der Preamble eingestellt werden. Die Preamble dient zur Synchronisierung des Empfängers auf den Sender. Je länger die Preamble vereinbart wird, desto geringer ist die Wahrscheinlichkeit eines Kommunikationsfehlers. Allerdings wird während der Synchronisierung kein Nutzsignal übertragen, sodass die Preamble nur so lang wie nötig konfiguriert werden sollte.

Datentyp	Werte	Information
UINT	5 bis 20	Bus Controller Default: 20

11.8 Die Flatstream-Kommunikation

11.8.1 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.

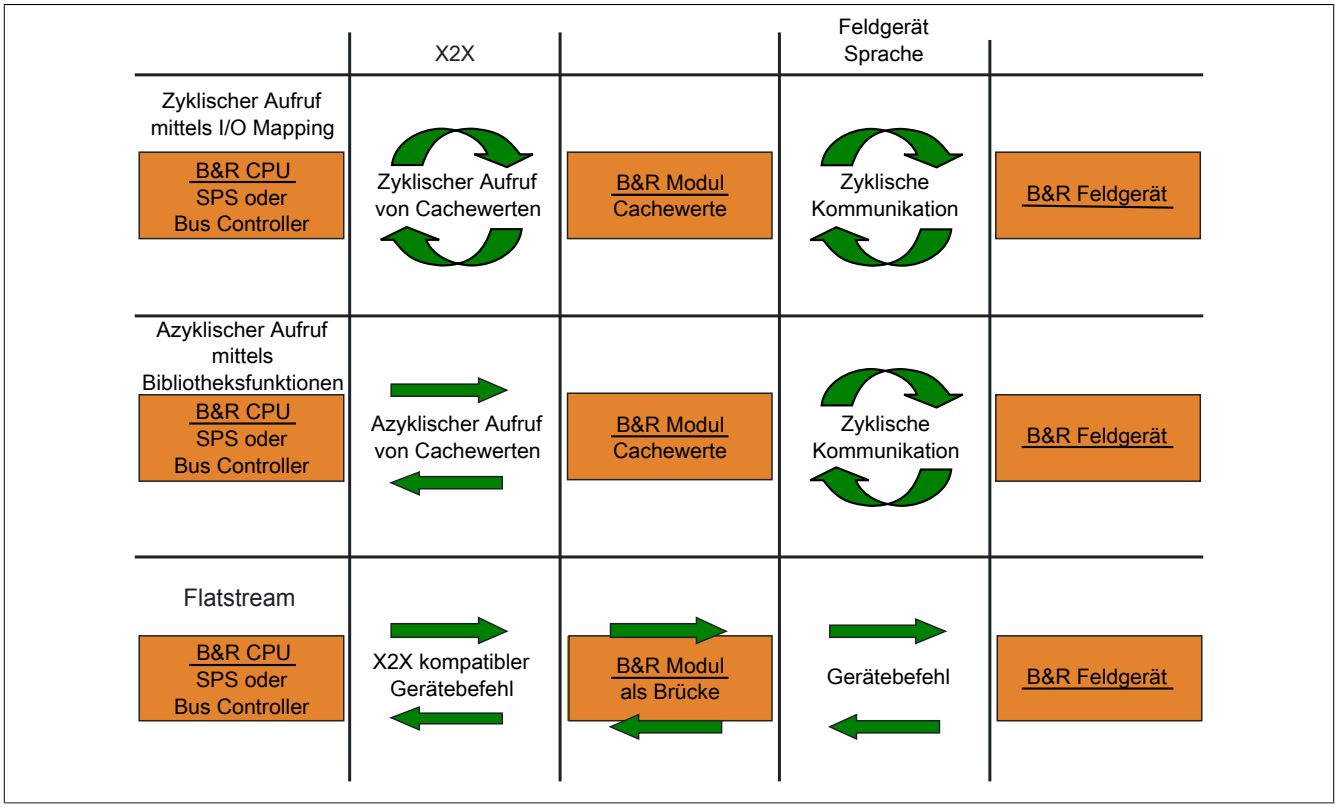


Abbildung 4: 3 Arten der Kommunikation

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

11.8.2 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 Forward 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.

11.8.3 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.

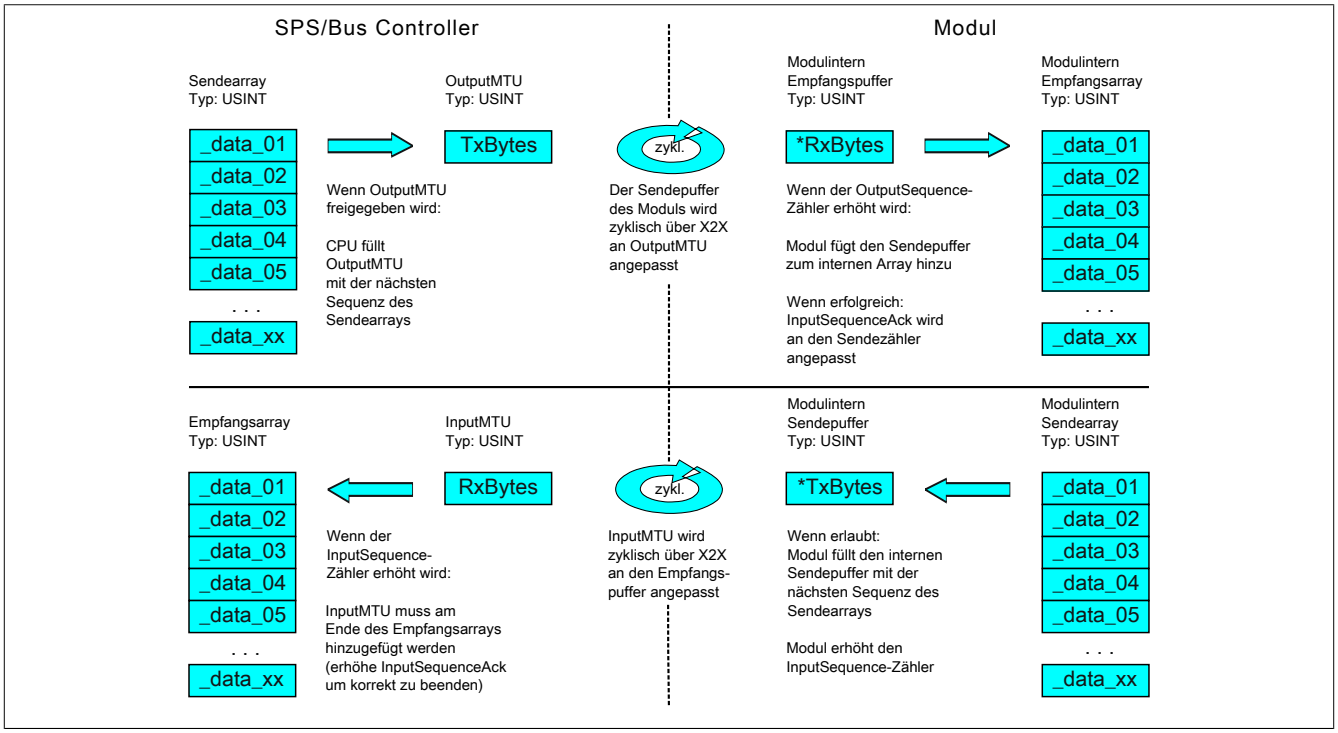


Abbildung 5: Kommunikation per Flatstream

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.

11.8.4 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 CPU 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.

11.8.4.1 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.

11.8.4.1.1 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.

Information:

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

Datentyp	Werte
USINT	Siehe modulspezifische Registerübersicht (theoretisch: 3 bis 27)

11.8.4.2 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.

11.8.4.2.1 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

11.8.4.2.2 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 unerschiedlich.)

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 CPU 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 CPU gewählt.

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

Datentyp	Werte
USINT	0 bis 255

11.8.4.2.3 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

11.8.4.2.4 Kommunikationsstatus der CPU

Name:
OutputSequence

Das Register "OutputSequence" enthält Informationen über den Kommunikationsstatus der CPU. Es wird von der CPU 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 CPU abgeschickt wurden. Über den OutputSequenceCounter weist die CPU das Modul an, eine Sequenz zu übernehmen (zu diesem Zeitpunkt muss die Output-Richtung synchronisiert sein).

OutputSyncBit

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

InputSequenceAck

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

InputSyncAck

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

11.8.4.2.5 Kommunikationsstatus des Moduls

Name:
InputSequence

Das Register "InputSequence" enthält Informationen über den Kommunikationsstatus des Moduls. Es wird vom Modul geschrieben und sollte von der CPU 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 CPU 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 CPU die Synchronität des Output-Kanals. Das Modul zeigt damit an, dass es bereit ist, Daten zu empfangen.

11.8.4.2.6 Beziehung zwischen Output- und InputSequence

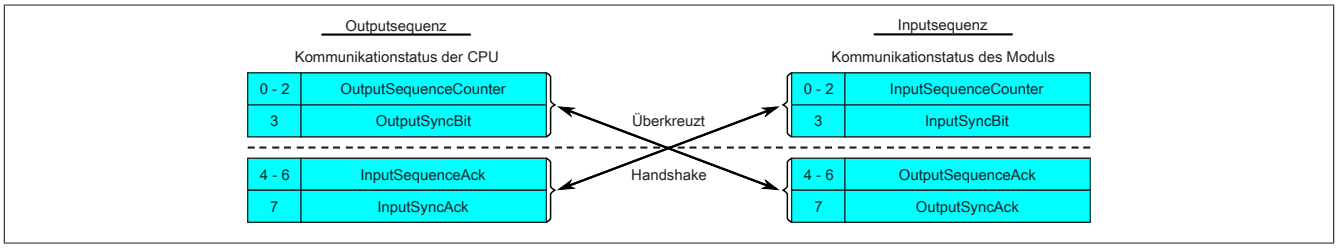


Abbildung 6: Zusammenhang zwischen Output- und InputSequence

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.

11.8.4.3 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 (CPU als Sender)

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

Algorithmus

1) CPU muss 000 in OutputSequenceCounter schreiben und OutputSyncBit zurücksetzen. CPU 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 CPU die erwarteten Werte in OutputSequenceAck und OutputSyncAck registriert, darf sie den OutputSequenceCounter inkrementieren. Die CPU 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 CPU die erwarteten Werte in OutputSequenceAck und OutputSyncAck registriert, darf sie das OutputSyncBit setzen. Die CPU 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, CPU kann Daten an Modul senden.

Synchronisation der Input-Richtung (CPU als Empfänger)

Die korrespondierenden Synchronisationsbits (InputSyncBit und InputSyncAck) sind zurückgesetzt. Aus diesem Grund können momentan keine Nachrichten vom Modul an die CPU 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) CPU darf den aktuellen Inhalt der InputMTU nicht übernehmen, weil der Kanal noch nicht synchronisiert ist. CPU 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) CPU darf den aktuellen Inhalt der InputMTU nicht übernehmen, weil der Kanal noch nicht synchronisiert ist. CPU 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) CPU 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 CPU senden.

11.8.4.4 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.

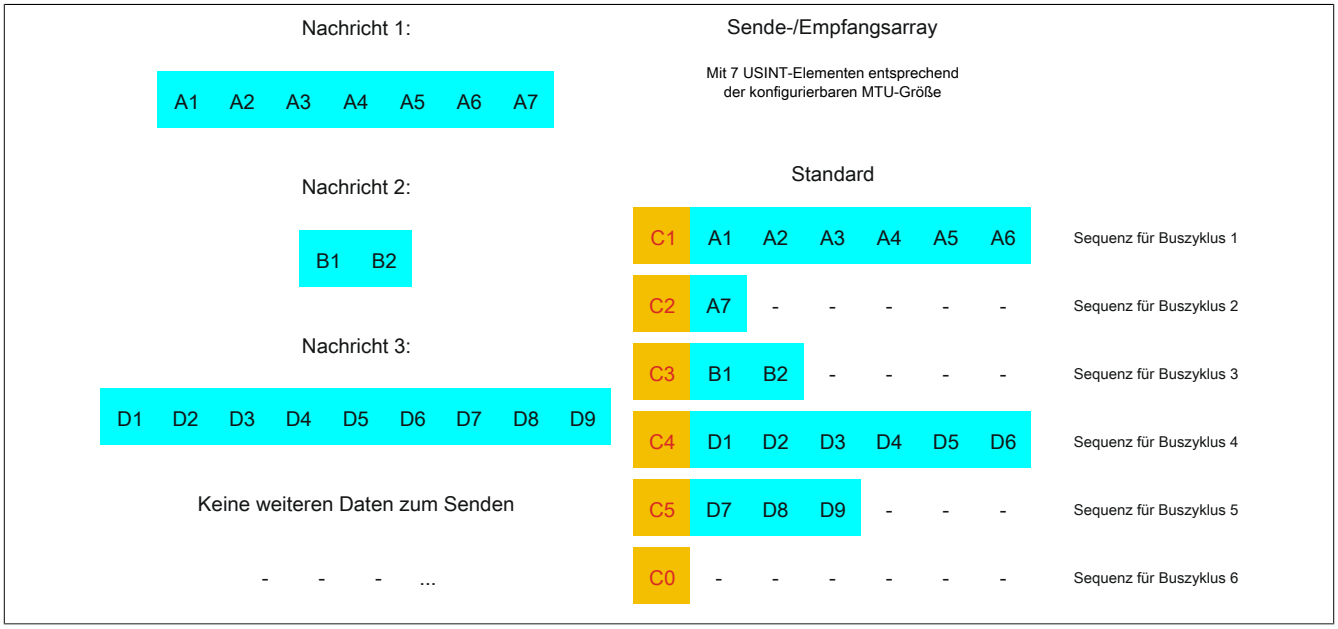


Abbildung 7: Sende-/Empfangsarray (Standard)

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)

11.8.4.5 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.

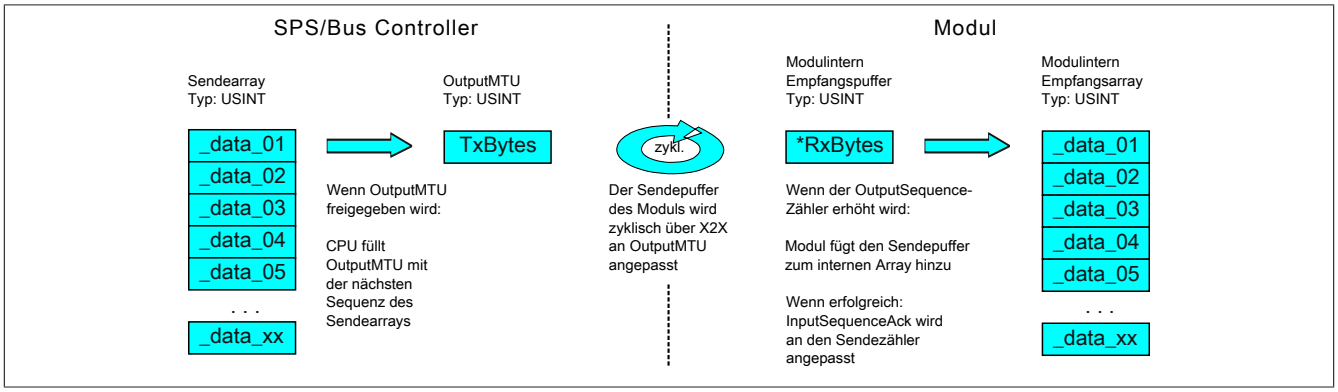


Abbildung 8: Kommunikation per Flatstream (Output)

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: - CPU muss OutputSyncAck prüfen → falls OutputSyncAck = 0; OutputSyncBit zurücksetzen und Kanal resynchronisieren - CPU muss Freigabe der OutputMTU prüfen → falls OutputSequenceCounter > InputSequenceAck; MTU nicht freigegeben, weil letzte Sequenz noch nicht bestätigt
1) Vorbereitung (Sendearray anlegen): - CPU muss Nachricht auf zulässige Segmente aufteilen und entsprechende Controlbytes bilden - CPU muss Segmente und Controlbytes zu Sendearray zusammenfügen
2) Senden: - CPU überträgt das aktuelle Element des Sendearrays in die OutputMTU → OutputMTU wird zyklisch in den Sendepuffer des Moduls übertragen, aber noch nicht weiterverarbeitet - CPU 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: - CPU 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

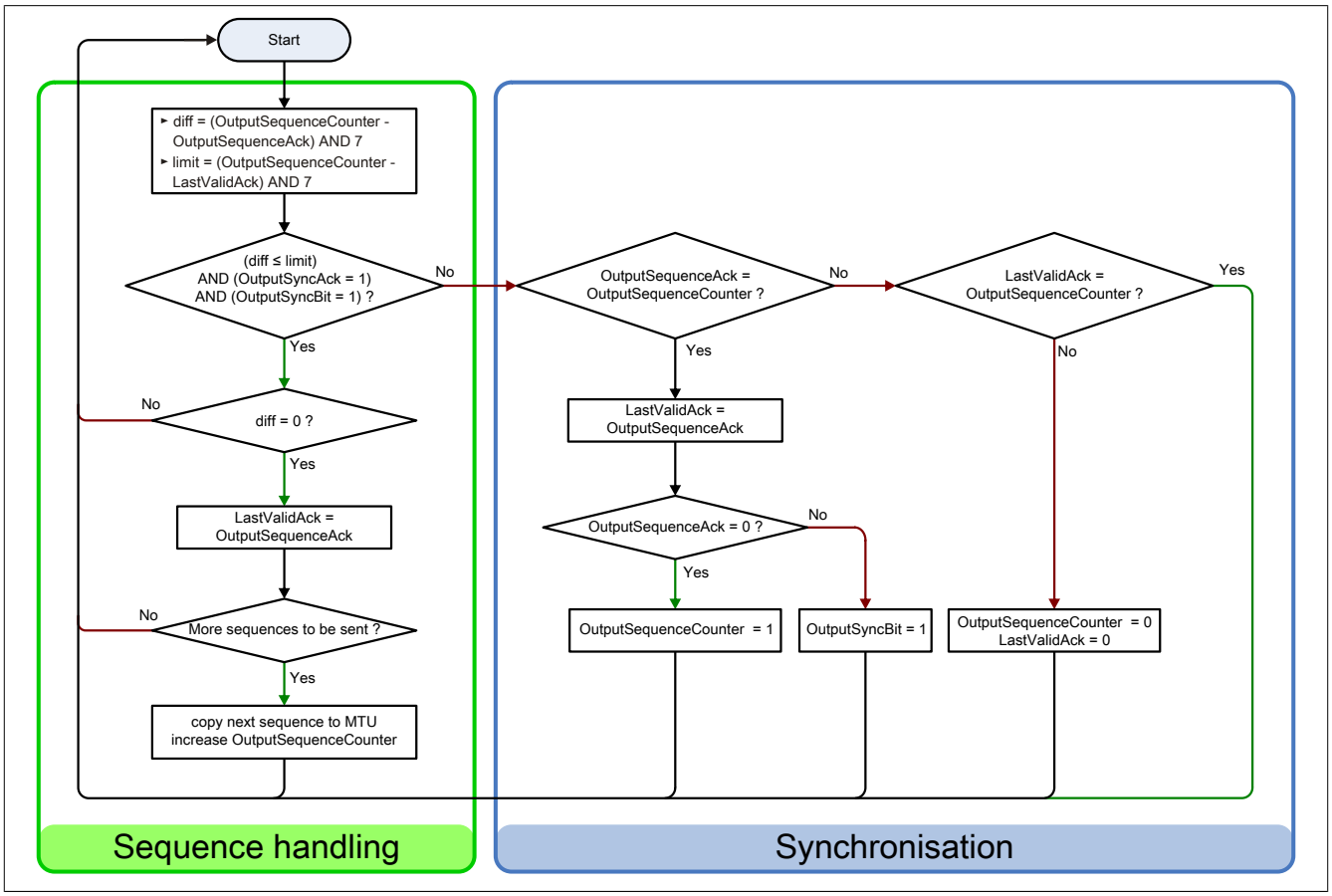


Abbildung 9: Ablaufdiagramm für Output-Richtung

11.8.4.6 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.

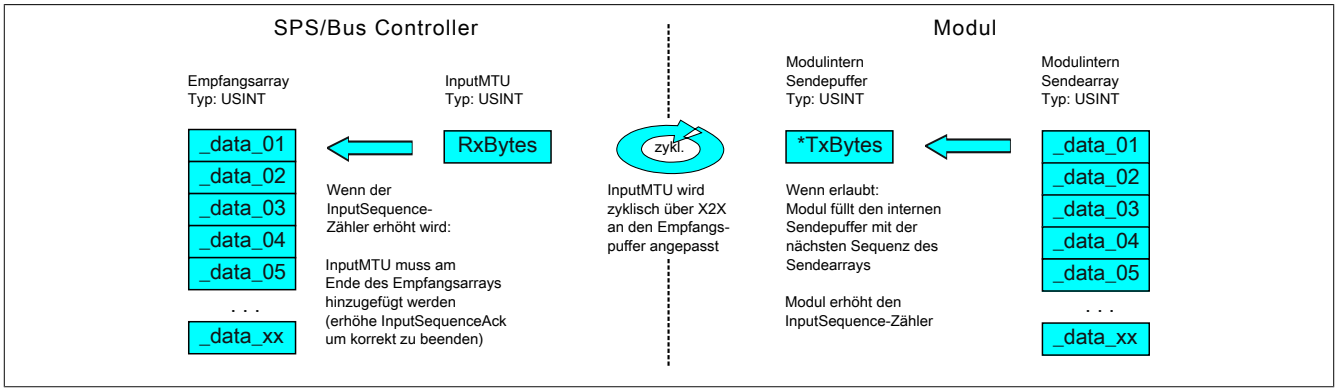


Abbildung 10: Kommunikation per Flatstream (Input)

Algorithmus

0) Zyklische Statusabfrage: - CPU 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): - CPU muss Daten aus InputMTU übernehmen und an das Ende des Empfangsarrays anfügen - CPU 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

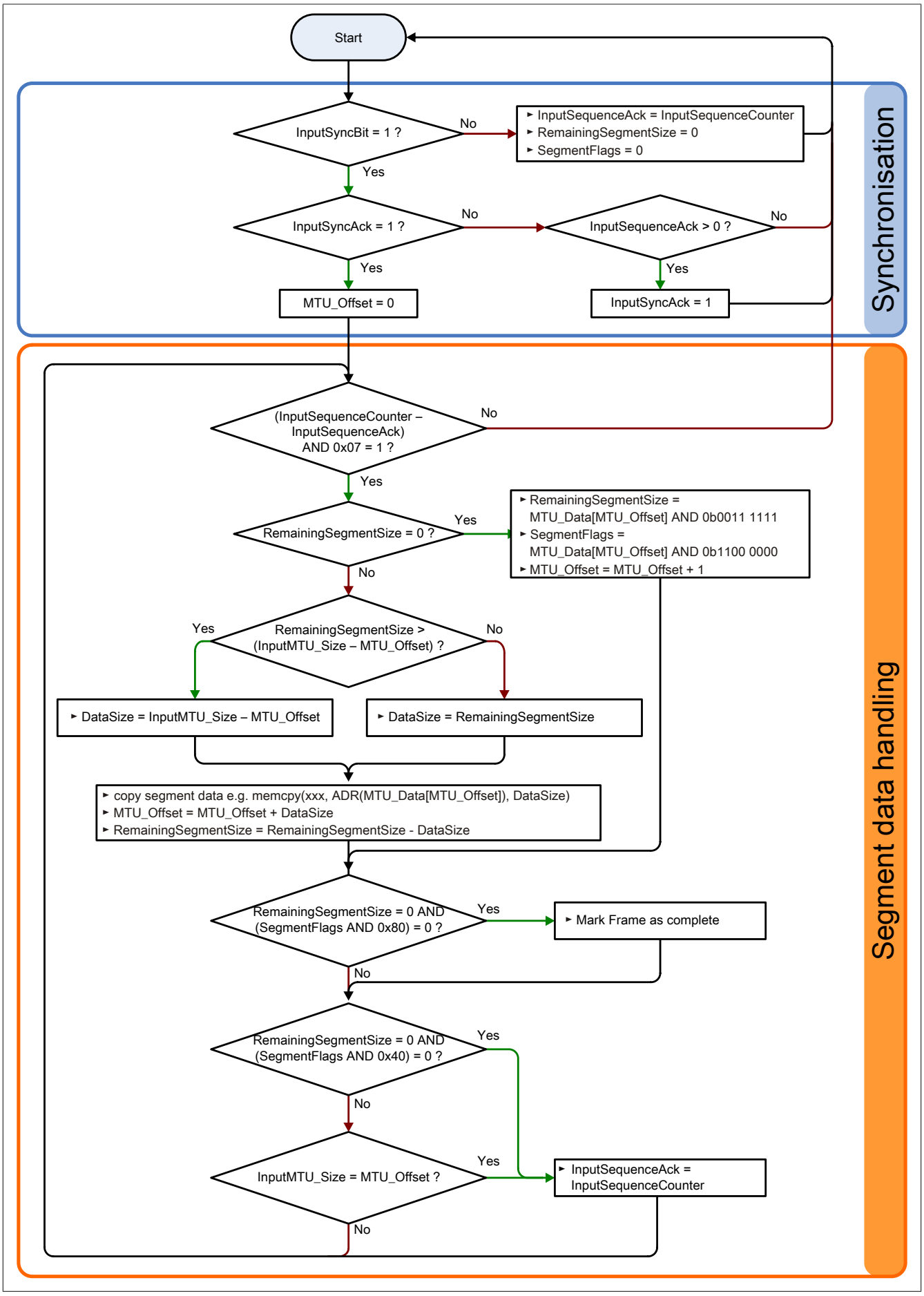


Abbildung 11: Ablaufdiagramm für Input-Richtung

11.8.4.7 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

Anmerkung: Beim Betrieb ohne Forward ist diese Situation sehr unwahrscheinlich.

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.

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.

11.8.4.8 Flatstream Modus

Name:
FlatstreamMode

In Input-Richtung wird das Sende-Array automatisch generiert. Dem Anwender werden über dieses Register 2 Optionen zur Verfügung gestellt, um eine kompaktere Anordnung beim eintreffenden Datenstrom zu erlauben. 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.

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		

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.

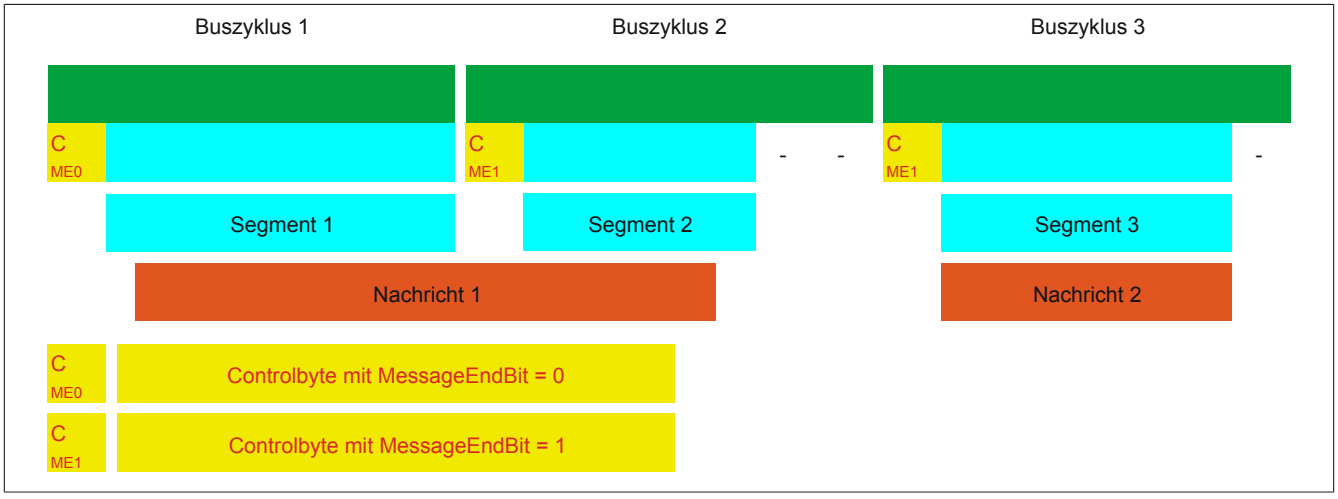


Abbildung 12: Anordnung von Nachrichten in der MTU (Standard)

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.

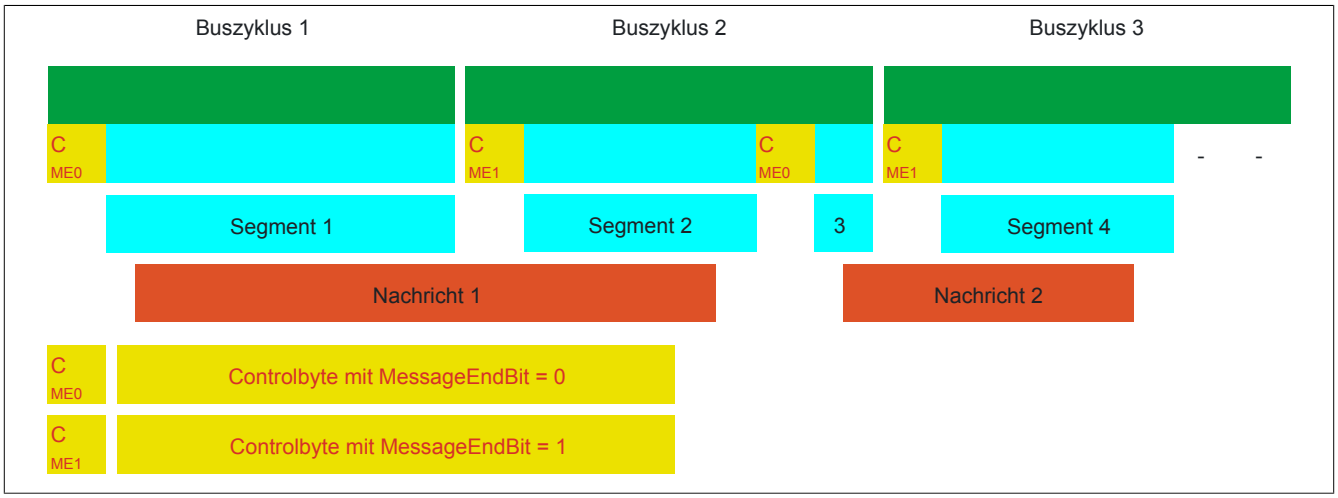


Abbildung 13: Anordnung von Nachrichten in der MTU (MultiSegmentMTU)

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.

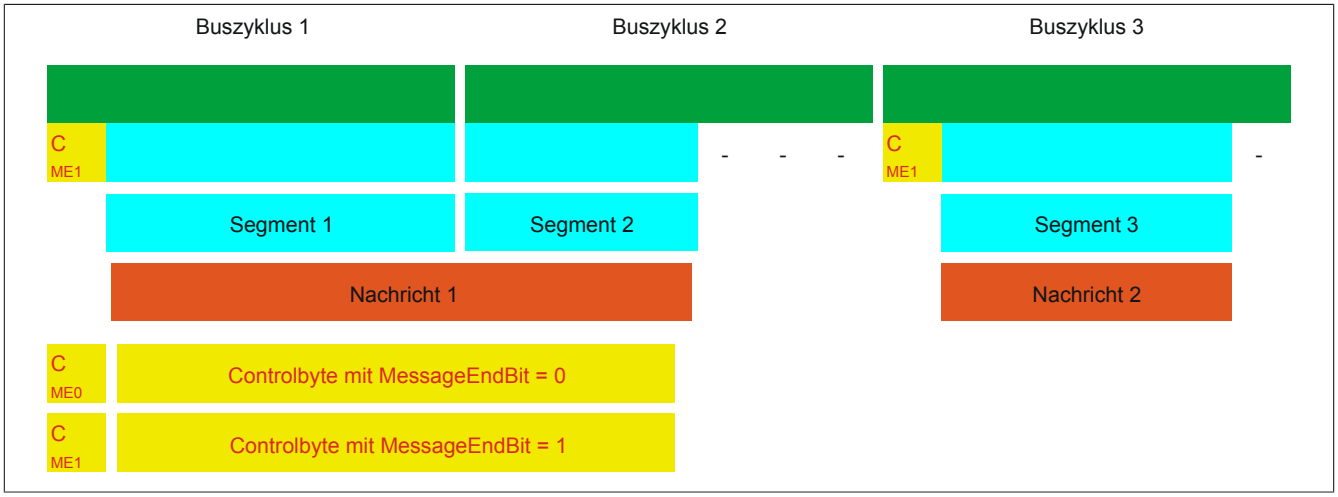


Abbildung 14: Anordnung von Nachrichten in der MTU (große Segmente)

Anwendung beider Optionen

Die beiden Optionen dürfen auch gleichzeitig angewendet werden.

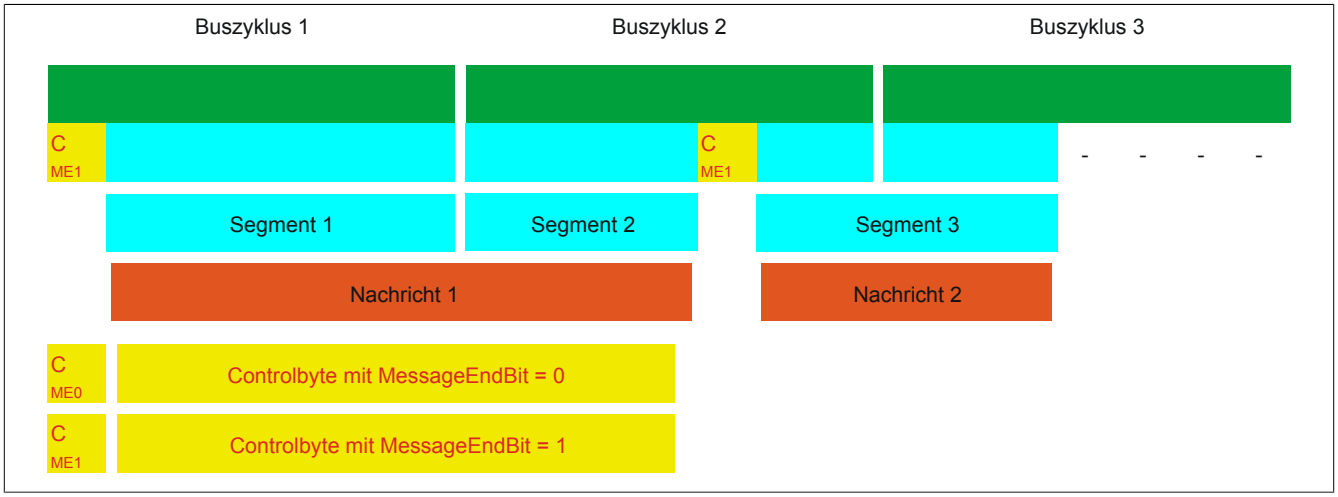


Abbildung 15: Anordnung von Nachrichten in der MTU (große Segmente und MultiSegmentMTU)

11.8.4.9 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.

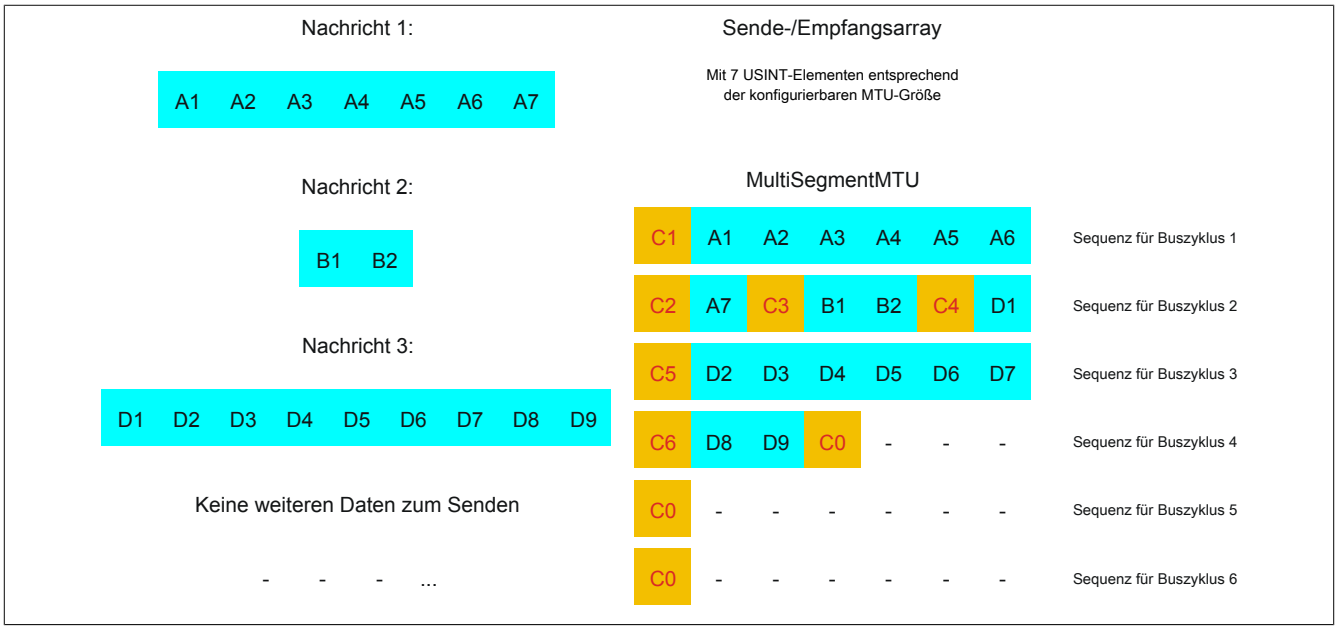


Abbildung 16: Sende-/Empfangsarray (MultiSegmentMTU)

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.

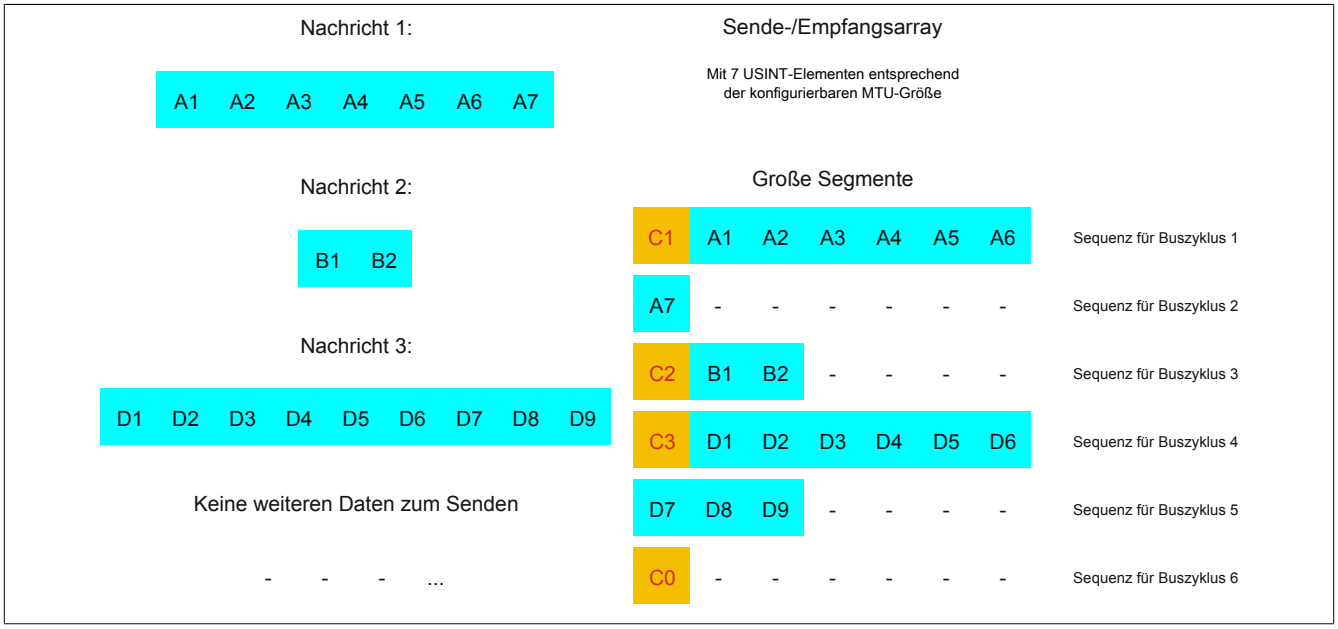


Abbildung 17: Sende-/Empfangsarray (große Segmente)

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.

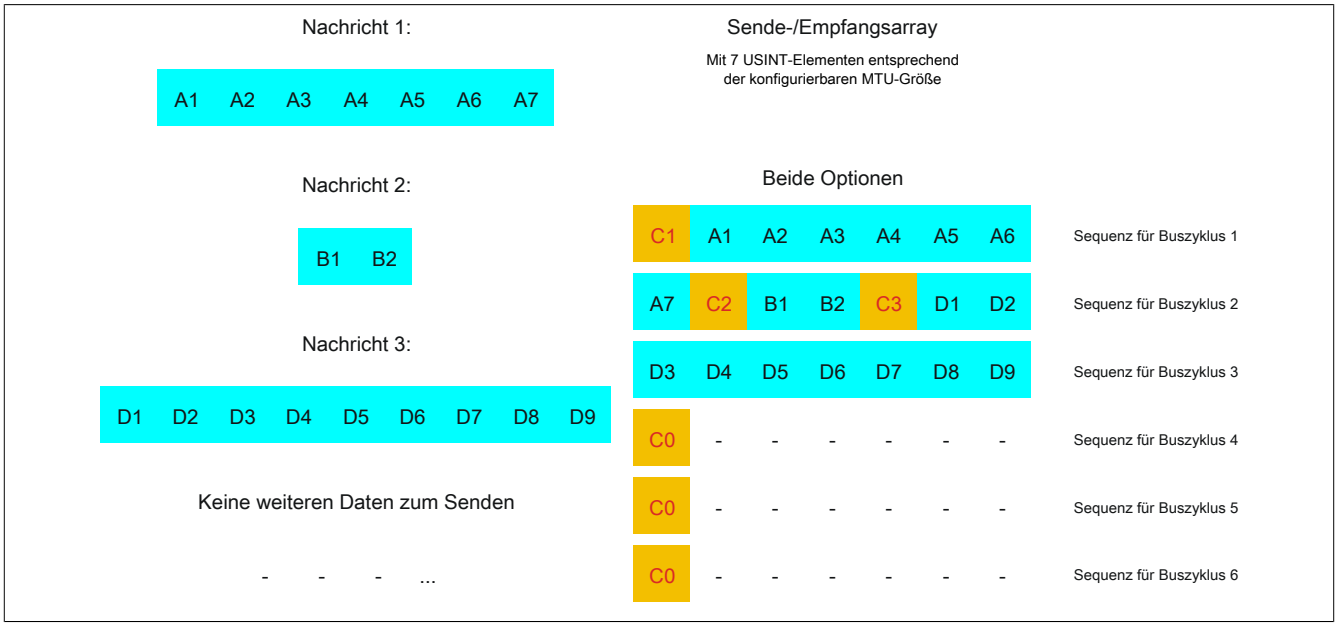


Abbildung 18: Sende-/Empfangsarray (große Segmente und MultiSegmentMTU)

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

11.8.5 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.

11.8.5.1 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)

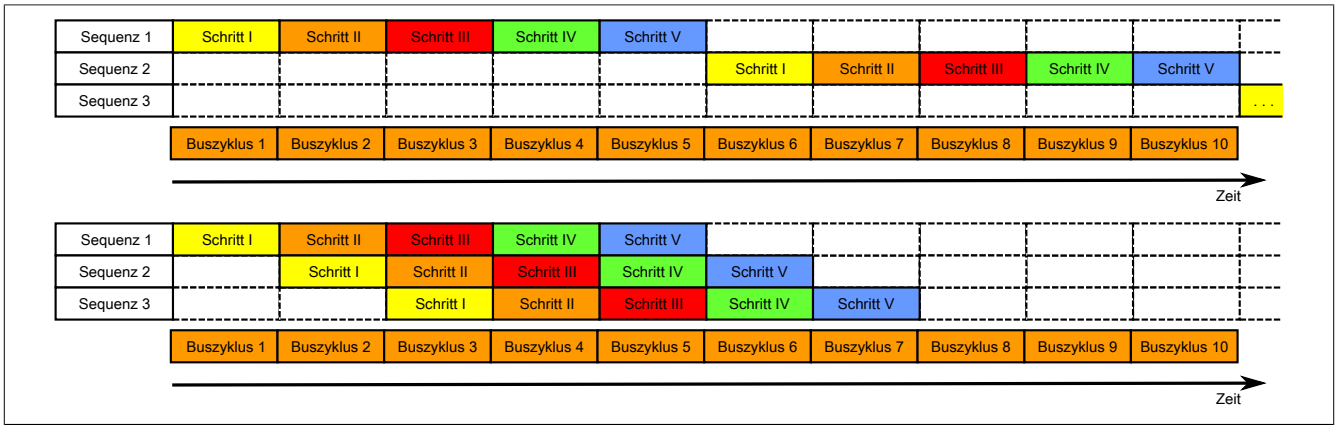


Abbildung 19: Vergleich Übertragung ohne bzw. mit Forward

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.

11.8.5.2 Konfiguration

Die Forward-Funktion muss nur für die Input-Richtung freigeschaltet werden. Zu diesem Zweck sind 2 weitere Register zu konfigurieren. 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.

11.8.5.2.1 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

11.8.5.2.2 Verzögerungszeit

Name:
ForwardDelay

Mit dem Register "ForwardDelay" wird die Verzögerungszeit 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.

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

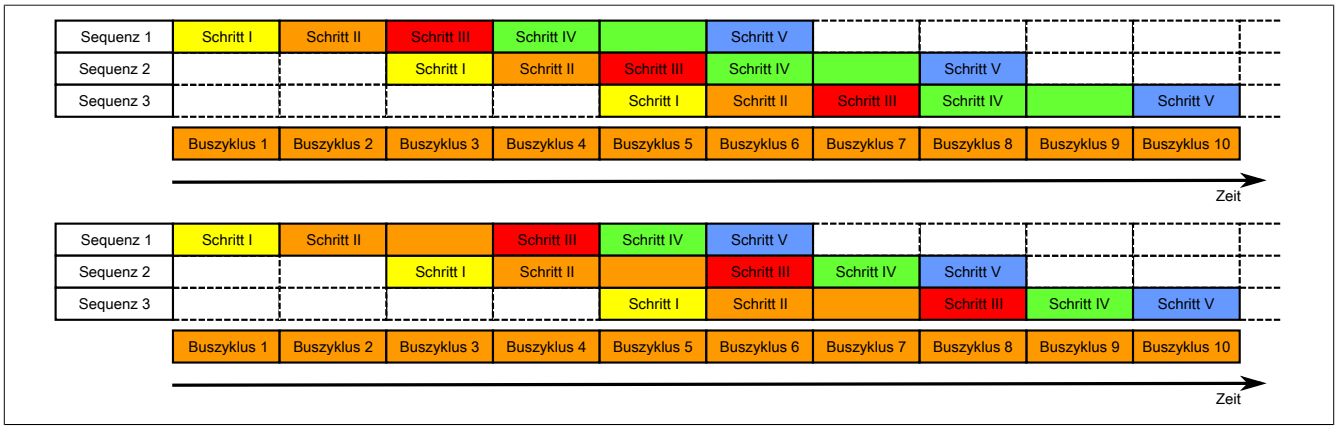


Abbildung 20: Auswirkung des ForwardDelay bei der Flatstream-Kommunikation mit Forward

Im Programmablauf muss sichergestellt werden, dass die CPU 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 CPU länger Zeit die eintreffende InputSequence bzw. InputMTU zu verarbeiten.

11.8.5.3 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: - CPU muss OutputSyncAck prüfen → falls OutputSyncAck = 0; OutputSyncBit zurücksetzen und Kanal resynchronisieren - CPU muss Freigabe der OutputMTU prüfen → falls OutputSequenceCounter > OutputSequenceAck + 7, in diesem Fall nicht freigegeben, weil letzte Sequenz noch nicht quittiert
1) Vorbereitung (Sendearray anlegen): - CPU muss Nachricht auf zulässige Segmente aufteilen und entsprechende Controlbytes bilden - CPU muss Segmente und Controlbytes zu Sendearray zusammenfügen
2) Senden: - CPU muss aktuellen Teil des Sendearrays in die OutputMTU übertragen - CPU muss OutputSequenceCounter erhöhen, damit Sequenz vom Modul übernommen wird - CPU 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): - CPU 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: - CPU 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): - CPU muss Daten aus InputMTU übernehmen und an das Ende des Empfangsarrays anfügen - CPU 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 CPU 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.

11.8.5.4 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).

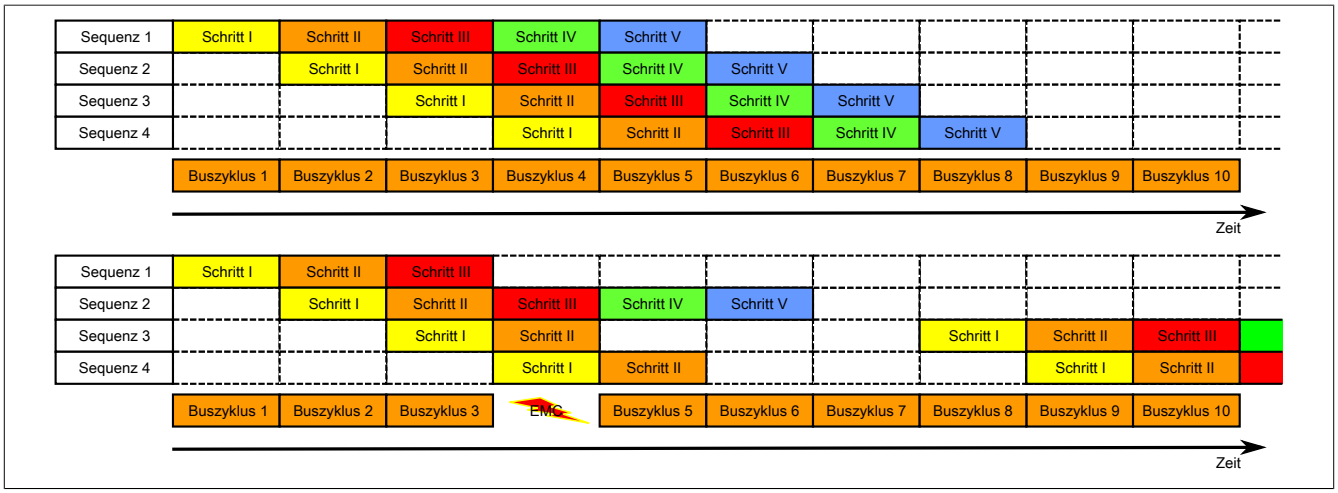


Abbildung 21: Auswirkung eines ausgefallenen Buszyklus

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.

11.9 HART on Flatstream

Bei der Flatstream-Kommunikation arbeitet das Modul als Bridge zwischen dem X2X Master und einem intelligenten Feldgerät, welches an das Modul angeschlossen ist. Der Flatstream-Modus kann sowohl für Point-to-Point-Verbindungen als auch bei Multidrop-Systemen genutzt werden. Spezifische Algorithmen wie Zeitüberschreitungs- oder Prüfsummenüberwachung werden in der Regel automatisch verwaltet. Dem Anwender sind diese Details im Normalbetrieb nicht zugänglich.

Bei HART handelt es sich um ein Master-Slave-Netzwerk, in dem asynchron und halbduplex kommuniziert wird. Um eine fehlerfreie Signalübertragung zu gewährleisten, werden verschiedene Sicherheitsvorkehrungen getroffen. Der Anwender kann z. B. die Länge der Preamble und damit die Sicherheit der Übertragung erhöhen. Allerdings wird dadurch auch das Verhältnis von Nutzdaten und Overhead beeinflusst.

Weiterführende Informationen zu HART sind auf www.HARTcomm.org ersichtlich.

Handhabung

Das Modul verfügt über 2 unabhängige Kanäle. Bei der Nutzung des Flatstreams muss deshalb zusätzlich die Kanalnummer angegeben werden. Die generelle Struktur des Flatstream-Frames wird folgendermaßen erweitert.

In/Output-Sequence	Tx/Rx-Bytes		
(unverändert)	Controlbyte (unverändert)	Kanalnummer	HART-Frame (ohne Preamble und Prüfsumme)

HART-Frame on Flatstream					
Start	ADDR	CMD	BCNT	(STS)	(DATA)

Start Start Kennung
ADDR Adresse innerhalb des HART-Netzwerkes
CMD HART-Befehl
BCNT Bytezähler (Anzahl der verbleibenden Bytes)
*STS Status des letzten empfangenen Befehls. Information über den Arbeitsmodus des HART-Slaves und Kommunikationsfehler (Falls unterstützt, Rückgaben des HART-Slaves)
*DATA Daten (falls für den Befehl benötigt)

Beispiele für HART-Kommandos

Kommando	Bedeutung
0x00	Slave-ID einlesen
0x03	Stromwert und bis zu 4 Variablen einlesen
0x09	Bis zu 4 Variablen inkl. Status einlesen
0x21	Variablen einlesen

11.10 Funktionsmodell "OSP"

Im Funktionsmodell "OSP" (Operator Set Predefined) definiert der Anwender einen analogen Wert bzw. ein digitales Muster. Dieser OSP-Wert wird ausgegeben, sobald die Kommunikation zwischen Modul und Master abbricht.

Funktionsweise

Der Anwender hat die Wahl zwischen 2 OSP-Modi:

- Letzten gültigen Wert halten
- Durch statischen Wert ersetzen

Im ersten Fall behält das Modul den letzten Wert als gültig erkannten Ausgabezustand bei.

Bei Auswahl des Modus "Durch statischen Wert ersetzen" muss auf dem dazugehörigen Value-Register ein plausibler Ausgabewert eingetragen sein. Bei Auftritt eines OSP-Ereignisses wird dieser Wert anstatt des aktuell vom Task angeforderten Wertes ausgegeben.

11.10.1 OSP-Ausgabe im Modul aktivieren

Name:
OSPValid

Dieser Datenpunkt bietet die Möglichkeit die Ausgabe des Moduls zu starten und während des laufenden Betriebs den OSP-Anwendungsfall anzufordern.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Bezeichnung	Wert	Information
0	OSPValid	0	OSP-Betrieb anfordern (nach Erststart oder Modul in Standby)
		1	Normalbetrieb anfordern
1 - 7	Reserviert	0	

Das OSPValid-Bit existiert einmal am Modul und wird vom Anwendertask verwaltet. Zum Start der aktivierten Kanäle muss es gesetzt werden. Solange das OSPValid-Bit im Modul gesetzt bleibt, verhält sich das Modul äquivalent zum Funktionsmodell "Standard".

Ereignet sich ein OSP-Ereignis, z. B. Abbruch der Kommunikation zwischen Modul und Master-CPU, wird modulseitig das OSPValid-Bit zurückgesetzt. Das Modul fällt in den OSP-Zustand und die Ausgabe erfolgt entsprechend der Konfiguration im "OSPMode" auf Seite 54-Register.

Grundsätzlich gilt:

Auch nach Regenerierung des Kommunikationskanals steht der OSP-Ersatzwert weiter an. Der OSP-Zustand wird erst wieder verlassen, wenn ein gesetztes OSPValid-Bit übertragen wird.

Bei Neustart der Master-CPU wird das OSPValid-Bit in der Master-CPU neu initialisiert. Es muss ein weiteres Mal durch die Anwendung gesetzt und über den Bus übertragen werden.
Bei kurzzeitigen Kommunikationsfehlern zwischen Modul und Master-CPU (z. B. durch EMV) fällt der Refresh der zyklischen Register für einige Buszyklen aus. Modulintern wird das OSPValid-Bit zurückgesetzt - in der CPU bleibt das gesetzte Bit hingegen erhalten. Bei der nächsten erfolgreichen Übertragung wird das modulinterne OSPValid-Bit wieder gesetzt und das Modul kehrt automatisch in den Normalbetrieb zurück.

Wird von Seiten des Tasks in der Master-CPU die Information benötigt, in welchem Ausgabemodus sich das Modul momentan befindet, kann das ModulOK-Bit ausgewertet werden.

Warnung!

Wird das OSPValid-Bit modulseitig auf "0" zurückgesetzt, hängt der Ausgabezustand nicht mehr vom zuständigen Task in der Master-CPU ab. Trotzdem erfolgt, je nach Konfiguration des OSP-Ersatzwertes, eine Ausgabe.

11.10.2 OSP-Modus einstellen

Name:
CfgOSPMode01 bis CfgOSPMode02

Dieses Register steuert grundlegend das Verhalten eines Kanals im OSP-Anwendungsfall.

Datentyp	Werte	Bedeutung
USINT	0	Durch statischen Wert ersetzen
	1	Letzten gültigen Wert halten

11.10.3 OSP analogen Ausgabewert festlegen

Name:
CfgOSPValue01 bis CfgOSPValue02

Dieses Register beinhaltet den analogen Ausgabewert, der im Modus "Durch statischen Wert ersetzen" bei OSP-Betrieb ausgegeben wird.

Datentyp	Werte
Entspricht AnalogOutput0x	Entspricht AnalogOutput0x

Warnung!

Der "OSPValue" wird vom Modul nur dann übernommen, wenn das "OSPValid"-Bit im Modul gesetzt wurde.

11.11 NetTime Technology

Unter NetTime versteht man die Möglichkeit Systemzeiten zwischen einzelnen Komponenten der Steuerung bzw. Netzwerks (CPU, 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

11.11.1 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.

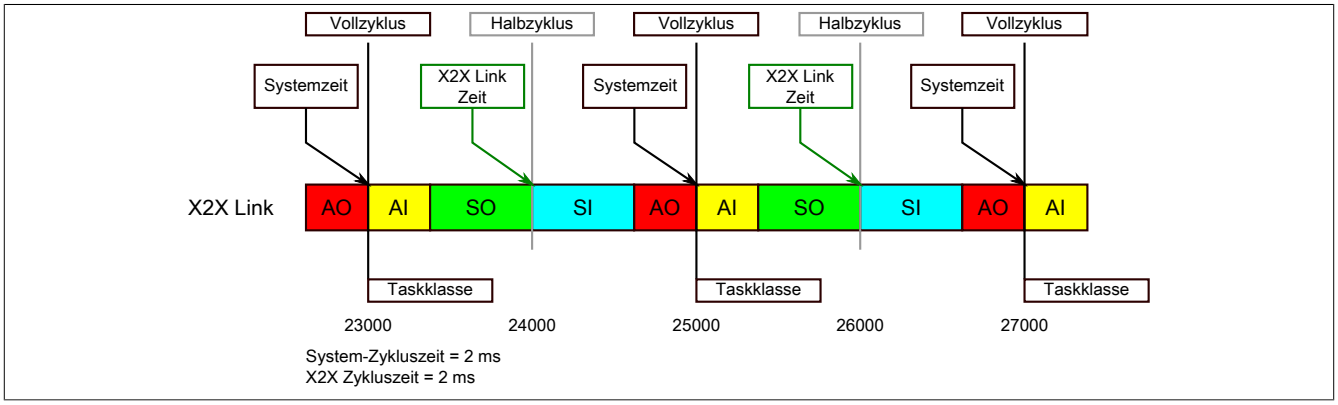
Die Initialisierung der Zeiten erfolgt auf Basis der Systemzeit während des Hochlaufs des X2X Links, der I/O-Module bzw. der POWERLINK-Schnittstelle.

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

11.11.1.1 SPS/Controller-Datenpunkte

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

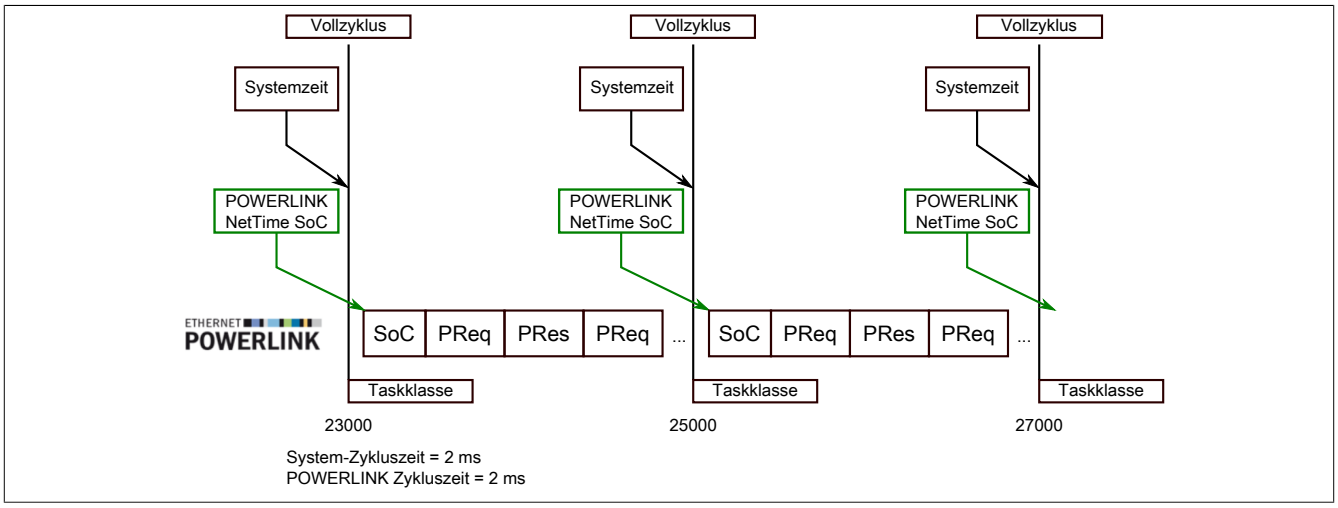
11.11.1.2 Refenzzeitpunkt X2X Link



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.

11.11.1.3 Referenzzeitpunkt POWERLINK

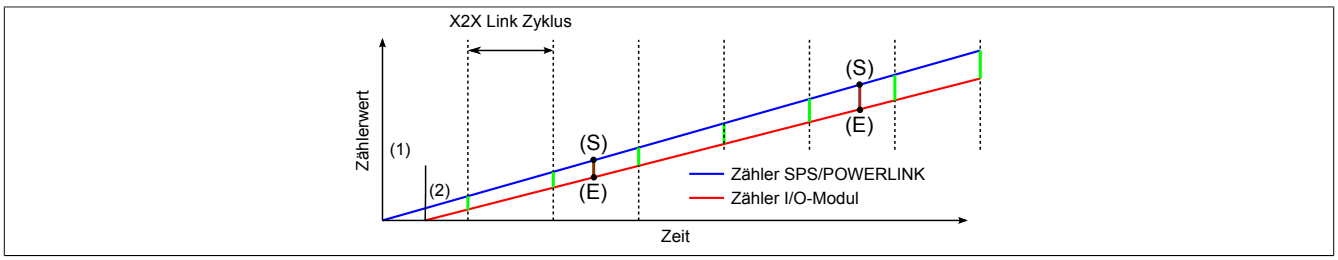


Der Referenzzeitpunkt am POWERLINK wird grundsätzlich beim SoC (Start of Cycle) des POWERLINK-Netzwerks gebildet. Der SoC startet systembedingt 20 µs nach dem Systemtakt. Dadurch ergibt sich folgende Differenz zwischen Systemzeit und POWERLINK-Referenzzeit:

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.

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



Beim Hochfahren starten die internen Zähler für die SPS/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 SPS 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.

11.11.2 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 CPU kann diese nun, gegebenenfalls mit Hilfe ihrer eigenen NetTime (bzw. Systemzeit), die Daten auswerten.

11.11.2.1 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.

11.11.2.2 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.

11.11.2.3 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.

11.12 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
200 µs

11.13 Minimale IO-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	
Analogausgänge	1 ms

Minimale I/O-Updatezeit Hart-Kommunikation	
Point-to-Point	500 ms
Multidrop	500 ms * Stationsanzahl