

X90IF7x0.0x-00

1 Allgemeines

1.1 Mitgeltende Dokumente

Weiterführende und ergänzende Informationen sind den folgenden gelisteten Dokumenten zu entnehmen.

Mitgeltende Dokumente

Dokumentname	Titel
MAX90	X90 mobile System Anwenderhandbuch

1.2 Bestelldaten

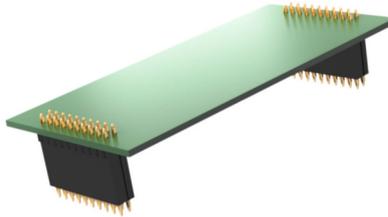
Bestellnummer	Kurzbeschreibung	Abbildung
	Kommunikationsmodule	
X90IF720.04-00	X90 mobile IF Optionsplatine, 3x CAN, 1x RS232 Konfiguration über Software	
X90IF730.04-00	X90 mobile IF Optionsplatine, 3x CAN, 1x RS485, Konfiguration über Software	
X90IF7L0.05-00	X90 mobile IF Optionsplatine, 3x CAN, 2x LIN (Master), Konfiguration über Software	

Tabelle 1: X90IF720.04-00, X90IF730.04-00, X90IF7L0.05-00 - Bestelldaten

1.3 Modulbeschreibung

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

Für die Anbindung von dezentralen Aktoren und Sensoren stehen zusätzlich zu den vorhandenen Schnittstellen auf der Hauptplatine Kommunikations-Optionsplatinen zur Verfügung.

- 3 CAN-Bus Schnittstellen
- serielle Schnittstelle

2 Technische Beschreibung

2.1 Technische Daten

Bestellnummer	X90IF720.04-00	X90IF730.04-00	X90IF7L0.05-00
Kurzbeschreibung			
Kommunikationsmodul	3x CAN-Bus, 1x RS232	3x CAN-Bus, 1x RS485	3x CAN-Bus, 2x LIN
Allgemeines			
B&R ID-Code	0xEF7F	0xEF80	0x2A7E
Statusanzeigen	-		
Leistungsaufnahme	1,6 W	1,71 W	1 W
Zulassungen			
UN ECE-R10	Ja		
CE	Ja		
UKCA	Ja		
Schnittstellen			
Schnittstelle IF1			
Signal	CAN-Bus		
max. Reichweite	1000 m		
Übertragungsrate	max. 1 MBit/s		
Abschlusswiderstand	Extern 120 Ω vorzusehen		
Schnittstelle IF2			
Signal	CAN-Bus		
max. Reichweite	1000 m		
Übertragungsrate	max. 1 MBit/s		
Abschlusswiderstand	Extern 120 Ω vorzusehen		
Schnittstelle IF3			
Signal	CAN-Bus		
max. Reichweite	1000 m		
Übertragungsrate	max. 1 MBit/s		
Abschlusswiderstand	Extern 120 Ω vorzusehen		
Schnittstelle IF4			
Signal	RS232	RS485	LIN
max. Reichweite	900 m	1200 m	40 m
Übertragungsrate	max. 115,2 kBit/s		max. 20 kBit/s
FIFO	1 kByte		-
Abschlusswiderstand	Nein	in Software schaltbar	-
Schnittstelle IF5			
Signal	-	-	LIN
max. Reichweite	-	-	40 m
Übertragungsrate	-	-	max. 20 kBit/s
Elektrische Eigenschaften			
Potenzialtrennung	Nein		
Einsatzbedingungen			
Einbaulage			
beliebig	Ja		
Schutzart nach EN 60529	bis zu IP69K ¹⁾		
Umgebungsbedingungen			
Temperatur			
Betrieb			
waagrechte Einbaulage	-40 bis 85°C Gehäuseoberfläche ¹⁾		
senkrechte Einbaulage	-40 bis 85°C Gehäuseoberfläche ¹⁾		
Lagerung	-40 bis 85°C		
Transport	-40 bis 85°C		
Luftfeuchtigkeit			
Betrieb	5 bis 100%, kondensierend		
Lagerung	5 bis 100%, kondensierend		
Transport	5 bis 100%, kondensierend		
Mechanische Eigenschaften			
Abmessungen			
Breite	47 mm		
Länge	95 mm		

Tabelle 2: X90IF720.04-00, X90IF730.04-00, X90IF7L0.05-00 - Technische Daten

1) In Abhängigkeit der Hauptplatine. Für weitere Details siehe Datenblatt Hauptplatine.

2.2 Bedien- und Anschlusselemente

2.2.1 X2X Link Schnittstelle

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

2.2.2 Anschlusskabel

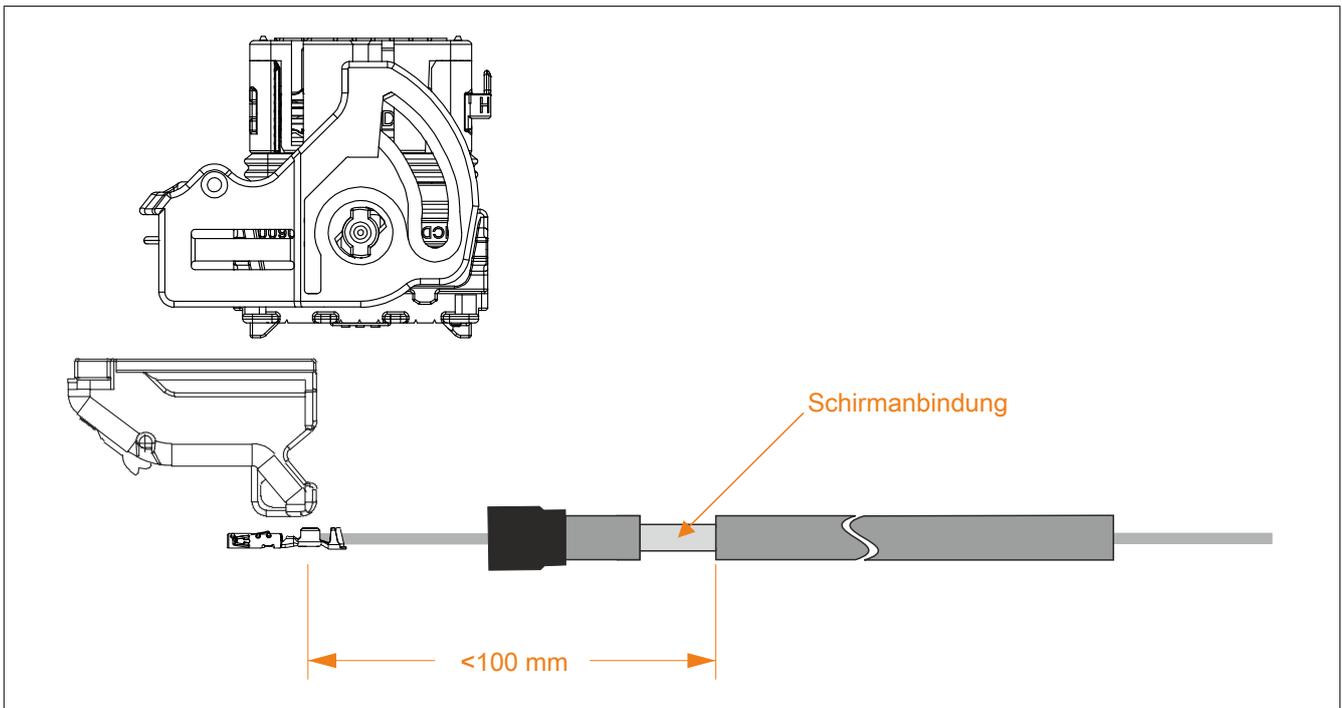
Der Anschluss erfolgt über den Stecker der Hauptplatine.

Hierfür werden Sensorkabel empfohlen.

2.3 Schirmung

Folgende Punkte für die Schirmung der Sensorleitungen sind zu beachten:

- Der Schirm der Sensorleitungen ist im Bereich des LIN-Anschluss großflächig und impedanzarm aufzulegen und auf kürzestem Weg, großflächig und impedanzarm mit dem X90 Gehäuse zu verbinden.
- Der Kabelschirm muss so weit wie möglich bis zum LIN-Anschluss reichen. Der Bereich der freigelegten Adern der Sensorleitung muss so kurz wie möglich gehalten werden (<10 cm).
- Die Leitungslänge zwischen LIN-Anschluss und Kabelschirmauflage muss so kurz wie möglich gehalten werden (<10 cm).
- Zur bestmöglichen Reduzierung der EMV-Störungen ist die Befestigung des X90 Gehäuses auf einer gut leitenden Montageplatte notwendig, auf der auch die Kabelschirme direkt aufzulegen sind.
- Das X90 Gehäuse ist zu erden.



2.4 Anschlussbelegung

Kanal	Anschlussbelegung		
	X90IF720.04-00	X90IF730.04-00	X90IF7L0.05-00
1	CAN2_L	CAN2_L	CAN2_L
2	CAN1_H	CAN1_H	CAN1_H
3	CAN2_H	CAN2_H	CAN2_H
4	CAN1_L	CAN1_L	CAN1_L
5	RS232 RXD	RS485 DATA\	LIN1
6	CAN3_H	CAN3_H	CAN3_H
7	RS232 TXD	RS485 DATA	LIN2
8	CAN3_L	CAN3_L	CAN3_L
9	n.c.	n.c.	n.c.
10	n.c.	n.c.	n.c.

n.c. ... nicht verbunden

3 Registerbeschreibung

3.1 Systemvoraussetzungen

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

- Automation Studio 4.9
- Automation Runtime 4.9

3.2 Registerübersicht

3.2.1 CyclicStream

Das Funktionsmodell "CyclicStream" nutzt einen modulspezifischen Treiber des Betriebssystems der Steuerung. Die CAN-Schnittstellen können mit Hilfe der Bibliotheken "ArCAN" und "CAN_Lib", die serielle Schnittstelle mit Hilfe der Bibliothek "DvFrame" gesteuert und während der Laufzeit umkonfiguriert werden.

Es werden zyklische I/O-Register genutzt, sodass die Kommunikation zeitlich determiniert abläuft.

Register	Name	Datentyp	Lesen		Schreiben	
			Zyklisch	Azyklisch	Zyklisch	Azyklisch
Konfiguration - CyclicStream01 (CAN1)						
0x10F1	Output01MTU	USINT		•		•
0x10F3	Input01MTU	USINT		•		•
Konfiguration - CyclicStream02 (CAN2)						
0x11F1	Output02MTU	USINT		•		•
0x11F3	Input02MTU	USINT		•		•
Konfiguration - CyclicStream03 (CAN3)						
0x12F1	Output03MTU	USINT		•		•
0x12F3	Input03MTU	USINT		•		•
Konfiguration - CyclicStream04 (RS232 / RS485)						
0x13F1	Output04MTU	USINT		•		•
0x13F3	Input04MTU	USINT		•		•
Konfiguration - RS485						
0x1381	TerminationResistor	USINT		•		•

3.2.2 StreamFilter

Je CAN-Schnittstelle können bis zu 4 Stream-Filter konfiguriert werden. Diese bestimmen, welche CAN-IDs über den CyclicStream an die Steuerung weitergeleitet werden.

Die Filter werden in der numerischen Reihenfolge durchlaufen. Der erste zur eintreffenden CAN-Nachricht passende Filter wird verwendet, alle weiteren Filter werden ignoriert. Wenn kein Filter zur eintreffenden CAN Nachricht passt, bestimmt eine globale Konfiguration, ob die Nachricht verworfen oder akzeptiert wird (Default: Nachricht akzeptieren).

Jeder Filter hat eine einstellbare ID und eine einstellbare Filtermaske. Es werden nur jene Bits der ID verglichen, welche in der Maske mit 0 gesetzt sind.

Register	Name	Datentyp	Lesen		Schreiben	
			Zyklisch	Azyklisch	Zyklisch	Azyklisch
Konfiguration - IF1StreamFilter (CAN1)						
0x1041	CfO_IF1DefaultCANFilterMode	USINT		•		•
0x104C + N*10	CfO_IF1CANFilter0N (Index N = 1 bis 4)	UDINT		•		•
0x1054 + N*10	CfO_IF1CANFilterMask0N (Index N = 1 bis 4)	UDINT		•		•
Konfiguration - IF2StreamFilter (CAN2)						
0x1141	CfO_IF2DefaultCANFilterMode	USINT		•		•
0x114C + N*10	CfO_IF2CANFilter0N (Index N = 1 bis 4)	UDINT		•		•
0x1154 + N*10	CfO_IF2CANFilterMask0N (Index N = 1 bis 4)	UDINT		•		•
Konfiguration - IF3StreamFilter (CAN3)						
0x1241	CfO_IF3DefaultCANFilterMode	USINT		•		•
0x124C + N*10	CfO_IF3CANFilter0N (Index N = 1 bis 4)	UDINT		•		•
0x1254 + N*10	CfO_IF3CANFilterMask0N (Index N = 1 bis 4)	UDINT		•		•

3.2.3 LIN-Register

Register	Name	Datentyp	Lesen		Schreiben		
			Zyklisch	Azyklisch	Zyklisch	Azyklisch	
LIN 1							
0x503	LinCtrl01	UINT			•		
0x513	LinStat01	UINT	•				
0x13F1	Output04MTU	USINT		•		•	
0x13F3	Input04MTU	USINT		•		•	
0x0300	Flatstream Rx/Tx cyclic data	USINT[61]	•		•		
LIN 2							
0x523	LinCtrl02	UINT			•		
0x533	LinStat02	UINT	•				
0x14F1	Output05MTU	USINT		•		•	
0x14F3	Input05MTU	USINT		•		•	
0x0400	Flatstream Rx/Tx cyclic data	USINT[61]	•		•		

3.2.4 Statusmeldung

Register	Name	Datentyp	Lesen		Schreiben		
			Zyklisch	Azyklisch	Zyklisch	Azyklisch	
Konfiguration - Statusmeldungen							
6273	CfO_ErrorByte01	USINT				•	
6275	CfO_ErrorByte02	USINT				•	
	CfO_ErrorByte03	USINT				•	
Kommunikation - Statusmeldungen							
6145	ErrorByte01	USINT	•				
	CANIF1warning	Bit 0					
	CANIF1passive	Bit 1					
	CANIF1busoff	Bit 2					
	CANIF1RXoverrun	Bit 3					
	CANIF2warning	Bit 4					
	CANIF2passive	Bit 5					
	CANIF2busoff	Bit 6					
6147	ErrorByte02	USINT	•				
	CANIF3warning	Bit 0					
	CANIF3passive	Bit 1					
	CANIF3busoff	Bit 2					
	CANIF3RXoverrun	Bit 3					
	xxxIF4StartBitError	Bit 4					
	xxxIF4StopBitError	Bit 5					
	xxxIF4ParityError	Bit 6					
6149	ErrorByte03	USINT	•				
	LINIF5StartBitError	Bit 0					
	LINIF5StopBitError	Bit 1					
	LINIF5ParityError	Bit 2					
6209	ErrorQuitByte01	USINT			•		
	QuitCANIF1warning	Bit 0					
	QuitCANIF1passive	Bit 1					
	QuitCANIF1busoff	Bit 2					
	QuitCANIF1RXoverrun	Bit 3					
	QuitCANIF2warning	Bit 4					
	QuitCANIF2passive	Bit 5					
	QuitCANIF2busoff	Bit 6					
6211	ErrorQuitByte02	USINT			•		
	QuitCANIF3warning	Bit 0					
	QuitCANIF3passive	Bit 1					
	QuitCANIF3busoff	Bit 2					
	QuitCANIF3RXoverrun	Bit 3					
	QuitxxxIF4StartBitError	Bit 4					
	QuitxxxIF4StopBitError	Bit 5					
	QuitxxxIF4ParityError	Bit 6					
6213	ErrorQuitByte03	USINT			•		
	QuitLINIF5StartBitError	Bit 0					
	QuitLINIF5StopBitError	Bit 1					
	QuitLINIF5ParityError	Bit 2					

3.3 Konfigurationsregister "CyclicStream"

3.3.1 Output0xMTU

Name:

Output01MTU bis Output05MTU

Diese Register definieren die Anzahl der aktiven Tx-Bytes und somit auch die maximale Größe einer Stream-Sequenz.

Datentyp	Werte
USINT	0 bis 60

3.3.2 Input0xMTU

Name:

Input01MTU bis Input05MTU

Diese Register definieren die Anzahl der aktiven Rx-Bytes und somit auch die maximale Größe einer Stream-Sequenz.

Datentyp	Werte
USINT	0 bis 60

3.3.3 TerminationResistor

Name:

TerminationResistor

Dieses Register ist nur beim X90IF730.04-00 vorhanden.

In diesem Register kann der Abschlusswiderstand der Schnittstelle aus- bzw. eingeschaltet werden.

Datentyp	Werte	Information
USINT	0	Widerstand deaktiviert
	1	Widerstand aktiviert

3.3.4 CfO_IFxDefaultCANFilterMode

Name:

CfO_IF1DefaultCANFilterMode bis CfO_IF3DefaultCANFilterMode

Diese Register geben die default Einstellungen für IDs an, die keinem der eingestellten Filter entsprechen.

Datentyp	Werte	Information
USINT	0	Kein Ansprechen des Filters, PDO Frame wird verworfen.
	1	Kein Ansprechen des Filters, PDO Frame wird über Stream übertragen.

3.3.5 CfO_IFxCANFilter

Name:

CfO_IF1CANFilter01 bis CfO_IF1CANFilter04

CfO_IF2CANFilter01 bis CfO_IF2CANFilter04

CfO_IF3CANFilter01 bis CfO_IF3CANFilter04

In diesen Registern sind die Filtereigenschaften definiert.

Datentyp	Werte
UDINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0 bis 28	Filter-ID	x	Zu filternder Identifier-Wert. ¹⁾
29	Identifier	0	11-Bit Identifier verwenden; Mögliche FilterID-Werte: 0 bis 2047 (0x7FF)
		1	29-Bit Identifier verwenden; Mögliche FilterID-Werte: 0 bis 536 870 911 (0x1FFFFFFF)
30	Reserviert	-	
31	Enable	0	Filter inaktiv
		1	Filter aktiv

1) Dieser Wert wird mit dem Identifier-Wert und dem Maskenwert verknüpft (siehe Beispiel).

Beispiel

Das folgende Beispiel zeigt den Zusammenhang zwischen Filtermaske, Filter-ID und den tatsächlich empfangenen 11-Bit CAN-Nachrichten.

Filtermaske ¹⁾	Filter-ID	CAN-Nachricht-ID	Information
000 0011 1110	110 0100 0000	110 0110 1010	Relevante Bits von Filter-ID und CAN-Nachricht sind identisch → Filter spricht an, der Frame wird entsprechend der Moduseinstellung verworfen oder weitergeleitet.
000 0011 1110	110 0100 0000	110 0110 1011	Relevante Bits nicht identisch → Nächster Filter bzw. Defaultmodus wird bearbeitet
000 0011 1111	110 0100 0000	110 0110 1011	Relevante Bits von Filter-ID und CAN-Nachricht sind identisch → Filter spricht an, der Frame wird entsprechend der Moduseinstellung verworfen oder weitergeleitet
000 0001 1111	110 0100 0000	110 0110 1011	Relevante Bits nicht identisch → Nächster Filter bzw. Defaultmodus wird bearbeitet

1) Rot = relevante bits

3.3.6 CfO_IFxCANFilterMask

Name:

CfO_IF1CANFilterMask01 bis CfO_IF1CANFilterMask04

CfO_IF2CANFilterMask01 bis CfO_IF2CANFilterMask04

CfO_IF3CANFilterMask01 bis CfO_IF3CANFilterMask04

In diesen Registern sind Filtermaske und Filtermode definiert.

Datentyp	Werte
UDINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0 bis 28	Filtermaske	x	Vergleichs-Bitmuster für Filter-ID ¹⁾
29 bis 30	Reserviert	-	
31	Modus	0	Bei Ansprechen des Filters wird der PDO Frame übertragen.
		1	Bei Ansprechen des Filters wird der PDO Frame verworfen.

1) Dieser Wert wird mit dem Identifier-Wert und dem Maskenwert verknüpft (siehe "Beispiel" auf Seite 7).

3.4 LIN-spezifische Register

3.4.1 LinStat0x

Name:

LinStat01 bis LinStat02

In diesen Registern wird der Status der LIN-Schnittstellen abgebildet.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0	CollisionError	0	Kein Fehler oder ausgeschaltet
		1	Kollision beim Senden von Daten erkannt
2	WakeUp	0	Kein Wakeup Signal erkannt oder ausgeschaltet
		1	Wakeup Signal erkannt

3.4.2 LinCtrl0x

Name:

LinCtrl01 bis LinCtrl02

In diesen Registern können die Kollisions- und Wakeup-Erkennung konfiguriert werden.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur:

Bit	Beschreibung	Wert	Information
0	QuitCollisionError	0	Keine Quittierung
		1	Quittierung des Fehlers
1	CollisionEnable	0	Kollisions Erkennung ist ausgeschaltet
		1	Kollisions Erkennung ist eingeschaltet
2	QuitWakeUp	0	Keine Quittierung
		1	Quittierung des Statussignals
3	WakeupEnable	0	Wakeup Erkennung ist ausgeschaltet
		1	Wakeup Erkennung ist eingeschaltet
4	WakeupTrigger	0	Keine Wakeup Trigger Sequenz wird gesendet
		1	Eine Wakeup Trigger Sequenz wird gesendet

3.5 Zyklische Datenpunkte für Error Anzeigen

3.5.1 ErrorByte

Name:

ErrorByte01 bis ErrorByte03

In diesem Register werden die einzelnen Fehlerbits übertragen. Tritt ein Fehler auf, so wird das entsprechende Bit gesetzt und gehalten bis eine Quittierung erfolgt.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur ErrorByte01:

Bit	Beschreibung	Wert	Information
0	CANIF1warning	0	Kein Fehler
		1	Fehler
1	CANIF1passive	0	Kein Fehler
		1	Fehler
2	CANIF1busoff	0	Kein Fehler
		1	Fehler
3	CANIF1RXoverrun	0	Kein Fehler
		1	Fehler
4	CANIF2warning	0	Kein Fehler
		1	Fehler
5	CANIF2passive	0	Kein Fehler
		1	Fehler
6	CANIF2busoff	0	Kein Fehler
		1	Fehler
7	CANIF2RXoverrun ¹⁾	0	Kein Fehler
		1	Fehler

1) Eingehende Daten werden in einem Empfangspuffer gespeichert und über den CyclicStream gesendet. Ist die Menge der eingehenden Daten größer als die Gesendete, kann es zu einem Überlauf kommen. Bei vollem Puffer werden weitere eingehende Daten verworfen.

Bitstruktur ErrorByte02:

Bit	Beschreibung			Wert	Information
	X90IF720.04-00 xxx = RS232	X90IF730.04-00 xxx = RS485	X90IF7L0.05-00 xxx = LIN		
0	CANIF3warning			0	Kein Fehler
				1	Fehler
1	CANIF3passive			0	Kein Fehler
				1	Fehler
2	CANIF3busoff			0	Kein Fehler
				1	Fehler
3	CANIF3RXoverrun			0	Kein Fehler
				1	Fehler
4	xxxIF4StartBitError			0	Kein Fehler
				1	Fehler
5	xxxIF4StopBitError			0	Kein Fehler
				1	Fehler
6	xxxIF4ParityError			0	Kein Fehler
				1	Fehler
7	xxxIF4RXoverrun ¹⁾			0	Kein Fehler
				1	Fehler

1) Eingehende Daten werden in einem Empfangspuffer gespeichert und über den CyclicStream gesendet. Ist die Menge der eingehenden Daten größer als die Gesendete, kann es zu einem Überlauf kommen. Bei vollem Puffer werden weitere eingehende Daten verworfen.

Bitstruktur ErrorByte03:

Bit	Beschreibung	Wert	Information
	X90IF7L0.05-00		
0	LINIF5StartBitError	0	Kein Fehler
		1	Fehler
1	LINIF5StopBitError	0	Kein Fehler
		1	Fehler
2	LINIF5ParityError	0	Kein Fehler
		1	Fehler
3	LINIF5RXoverrun ¹⁾	0	Kein Fehler
		1	Fehler
4 - 7	Reserviert	0	

1) Eingehende Daten werden in einem Empfangspuffer gespeichert und über den CyclicStream gesendet. Ist die Menge der eingehenden Daten größer als die Gesendete, kann es zu einem Überlauf kommen. Bei vollem Puffer werden weitere eingehende Daten verworfen.

3.5.2 ErrorQuitByte

Name:

ErrorQuitByte01 bis ErrorQuitByte03

In diesem Register werden die einzelnen Fehlerbits übertragen um einen auftretenden Fehler zu quittieren. Das Quittierungsbit kann erst rückgesetzt werden, wenn das entsprechende Fehlerstatusbit nicht mehr gesetzt ist.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur ErrorQuitByte01:

Bit	Beschreibung	Wert	Information
0	QuitCANIF1warning	0	Keine Quittierung
		1	Quittierung des Fehlers
1	QuitCANIF1passive	0	Keine Quittierung
		1	Quittierung des Fehlers
2	QuitCANIF1busoff	0	Keine Quittierung
		1	Quittierung des Fehlers
3	QuitCANIF1RXoverrun	0	Keine Quittierung
		1	Quittierung des Fehlers
4	QuitCANIF2warning	0	Keine Quittierung
		1	Quittierung des Fehlers
5	QuitCANIF2passive	0	Keine Quittierung
		1	Quittierung des Fehlers
6	QuitCANIF2busoff	0	Keine Quittierung
		1	Quittierung des Fehlers
7	QuitCANIF2RXoverrun	0	Keine Quittierung
		1	Quittierung des Fehlers

Bitstruktur ErrorQuitByte02:

Bit	Beschreibung			Wert	Information
	X90IF720.04-00 xxx = RS232	X90IF730.04-00 xxx = RS485	X90IF7L0.05-00 xxx = LIN		
0	QuitCANIF3warning			0	Keine Quittierung
				1	Quittierung des Fehlers
1	QuitCANIF3passive			0	Keine Quittierung
				1	Quittierung des Fehlers
2	QuitCANIF3busoff			0	Keine Quittierung
				1	Quittierung des Fehlers
3	QuitCANIF3RXoverrun			0	Keine Quittierung
				1	Quittierung des Fehlers
4	QuitxxxIF4StartBitError			0	Keine Quittierung
				1	Quittierung des Fehlers
5	QuitxxxIF4StopBitError			0	Keine Quittierung
				1	Quittierung des Fehlers
6	QuitxxxIF4ParityError			0	Keine Quittierung
				1	Quittierung des Fehlers
7	QuitxxxIF4RXoverrun			0	Keine Quittierung
				1	Quittierung des Fehlers

Bitstruktur ErrorQuitByte03:

Bit	Beschreibung		Wert	Information
	X90IF7L0.05-00			
0	QuitLINIF5StartBitError		0	Keine Quittierung
			1	Quittierung des Fehlers
1	QuitLINIF5StopBitError		0	Keine Quittierung
			1	Quittierung des Fehlers
2	QuitLINIF5ParityError		0	Keine Quittierung
			1	Quittierung des Fehlers
3	QuitLINIF5RXoverrun		0	Keine Quittierung
			1	Quittierung des Fehlers
4 - 7	Reserviert			

3.6 Konfigurationsregister Error-Anzeigen

3.6.1 CfO_ErrorByte

Name:

CfO_ErrorByte01 bis CfO_ErrorByte03

Mit diesem Register müssen die zu übertragenden Fehlermeldungen zuerst konfiguriert werden. Wenn das entsprechende Aktivierungsbit nicht gesetzt ist, wird beim Auftreten des Fehlers auch kein Fehler an das übergeordnete System gemeldet.

Datentyp	Werte
USINT	Siehe Bitstruktur

Bitstruktur CfO_ErrorByte01:

Bit	Beschreibung	Wert	Information
0	CANIF1warning	0	Fehler ignorieren
		1	Fehler anzeigen
1	CANIF1passive	0	Fehler ignorieren
		1	Fehler anzeigen
2	CANIF1busoff	0	Fehler ignorieren
		1	Fehler anzeigen
3	CANIF1RXoverrun	0	Fehler ignorieren
		1	Fehler anzeigen
4	CANIF2warning	0	Fehler ignorieren
		1	Fehler anzeigen
5	CANIF2passive	0	Fehler ignorieren
		1	Fehler anzeigen
6	CANIF2busoff	0	Fehler ignorieren
		1	Fehler anzeigen
7	CANIF2RXoverrun	0	Fehler ignorieren
		1	Fehler anzeigen

Bitstruktur CfO_ErrorByte02:

Bit	Beschreibung			Wert	Information
	X90IF720.04-00 xxx = RS232	X90IF730.04-00 xxx = RS485	X90IF7L0.05-00 xxx = LIN		
0	CANIF3warning			0	Fehler ignorieren
				1	Fehler anzeigen
1	CANIF3passive			0	Fehler ignorieren
				1	Fehler anzeigen
2	CANIF3busoff			0	Fehler ignorieren
				1	Fehler anzeigen
3	CANIF3RXoverrun			0	Fehler ignorieren
				1	Fehler anzeigen
4	xxxIF4StartBitError			0	Fehler ignorieren
				1	Fehler anzeigen
5	xxxIF4StopBitError			0	Fehler ignorieren
				1	Fehler anzeigen
6	xxxIF4ParityError			0	Fehler ignorieren
				1	Fehler anzeigen
7	xxxIF4RXoverrun			0	Fehler ignorieren
				1	Fehler anzeigen

Bitstruktur CfO_ErrorByte03:

Bit	Beschreibung	Wert	Information
	X90IF7L0.05-00		
0	LINIF5StartBitError	0	Fehler ignorieren
		1	Fehler anzeigen
1	LINIF5StopBitError	0	Fehler ignorieren
		1	Fehler anzeigen
2	LINIF5ParityError	0	Fehler ignorieren
		1	Fehler anzeigen
3	LINIF5ParityError	0	Fehler ignorieren
		1	Fehler anzeigen
4 - 7	Reserviert	-	-

3.7 Die CyclicStream-Kommunikation

Die physikalischen Eigenschaften des Bussystems begrenzen die Datenmenge, die während eines Buszyklus übermittelt werden kann.

MTU (Maximum Transmission Unit) - Physikalischer Transport

Die MTU beschreibt die aktivierten USINT-Register die dem CyclicStream zur Datenübertragung zur Verfügung stehen. Für beide Kommunikationsrichtungen wird eine separate MTU vereinbart.

Die OutputMTU definiert die Anzahl der Tx-Bytes und die InputMTU beschreibt die Anzahl der Rx-Bytes.

Die MTUs werden zyklisch über den X2X Link transportiert, sodass die Auslastung mit jedem zusätzlich aktivierten USINT-Register steigt.

Zusammenhang MTU-Größe und X2X Zykluszeit

Je nachdem wie groß die MTU der einzelnen Schnittstellen eingestellt ist, müssen gewisse Mindestzeiten am X2X Link eingehalten werden, da es ansonsten vorkommen kann, dass das Modul zu wenig Zeit für das X2X Handling hat und ein Reset ausgelöst wird.

Entscheidend ist hierbei die Summe aller MTU-Sizes in eine Richtung (Input/Output) aller Schnittstellen gemeinsam. Je größer die X2X Zykluszeit eingestellt ist, umso größer kann die Summe der MTUs sein.

4 Getting Started mit CpLin

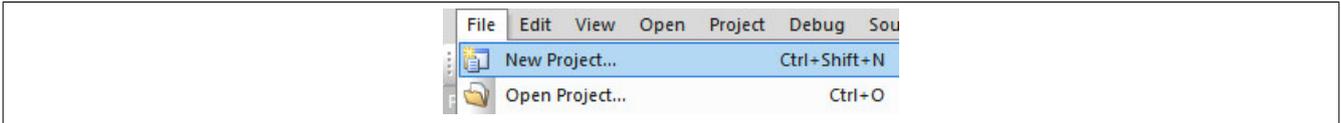
LIN (Local Interconnect Network) dient zum Übertragen von seriellen Frames, die mittels der Bibliothek "DVFrame" von der Applikation (LIN Master Stack oder anwenderspezifische Implementierung) gesendet und empfangen werden sollen. Dazu werden die Daten über zyklischen Register mit dem CyclicStream-Verfahren übertragen.

Das Schnittstellenmodul unterstützt ausschließlich die LIN-Masteranschaltung für das X90 System. Ein Betrieb als Slave ist nicht vorgesehen.

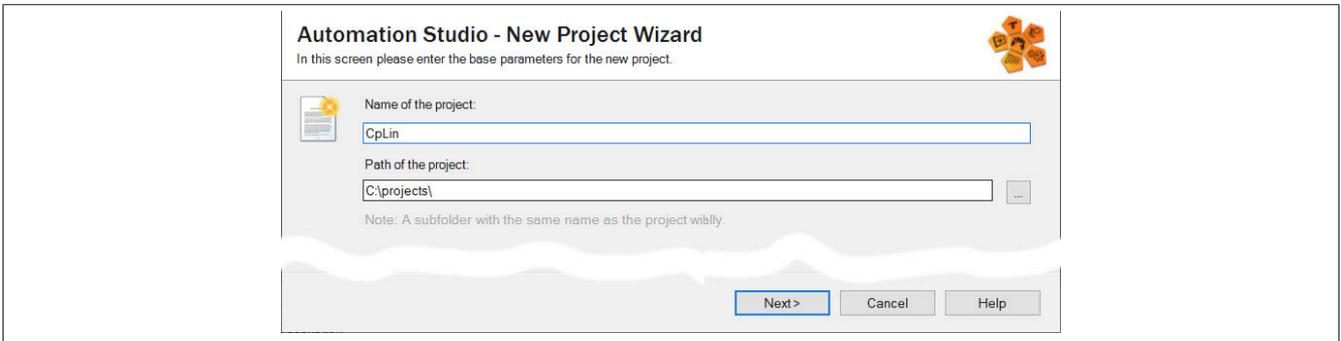
4.1 Grundlegende Schritte

4.1.1 Automation Studio Projekt erstellen

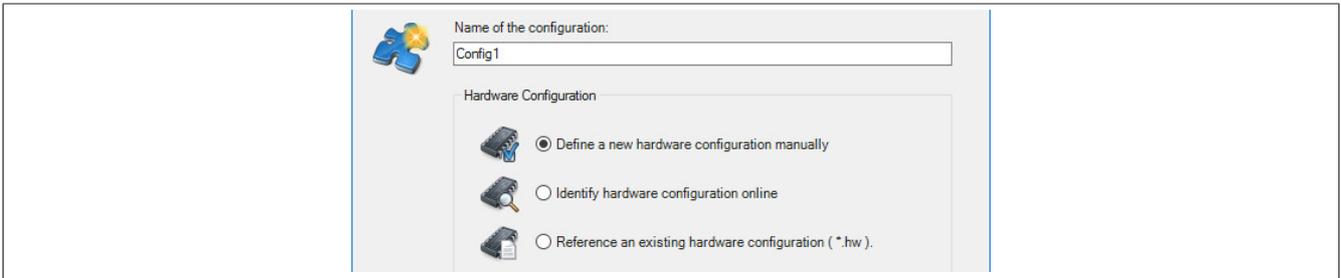
- Ein neues Automation Studio Projekt durch Auswahl von "Neues Projekt" erstellen.



- Projektnamen vergeben und Projektpfad einrichten.

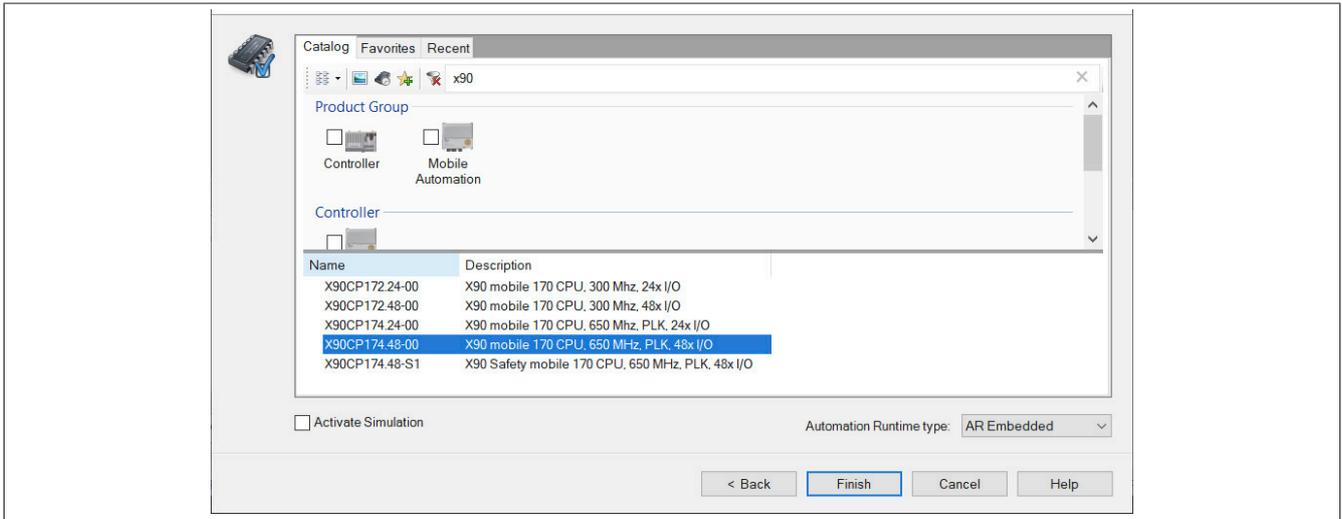


- Die Art der Hardwarekonfiguration auswählen und der Konfiguration einen Namen geben.



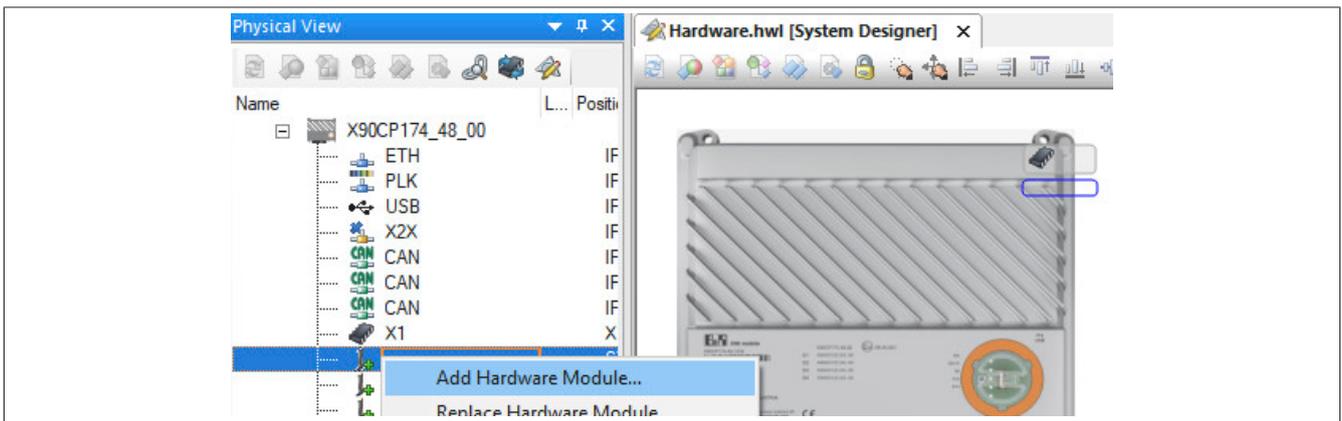
- Wenn "Eine neue Hardwarekonfiguration manuell definieren" ausgewählt wurde, im nächsten Schritt die Hardware auswählen.

Um die Suche zu vereinfachen, können im Hardwarekatalog beliebige Filter gesetzt werden. Abschließend die erforderliche Hardware mit LIN-Schnittstellenunterstützung markieren und das Automation Studio Projekt mit einem Klick auf "Fertigstellen" generieren.

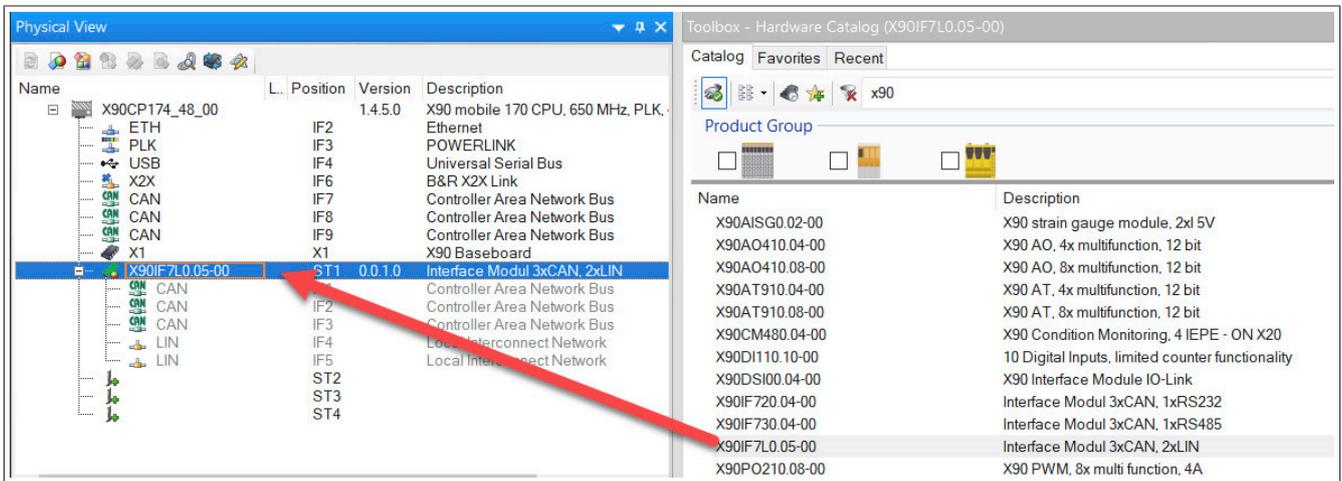


4.1.2 Schnittstellenmodul hinzufügen

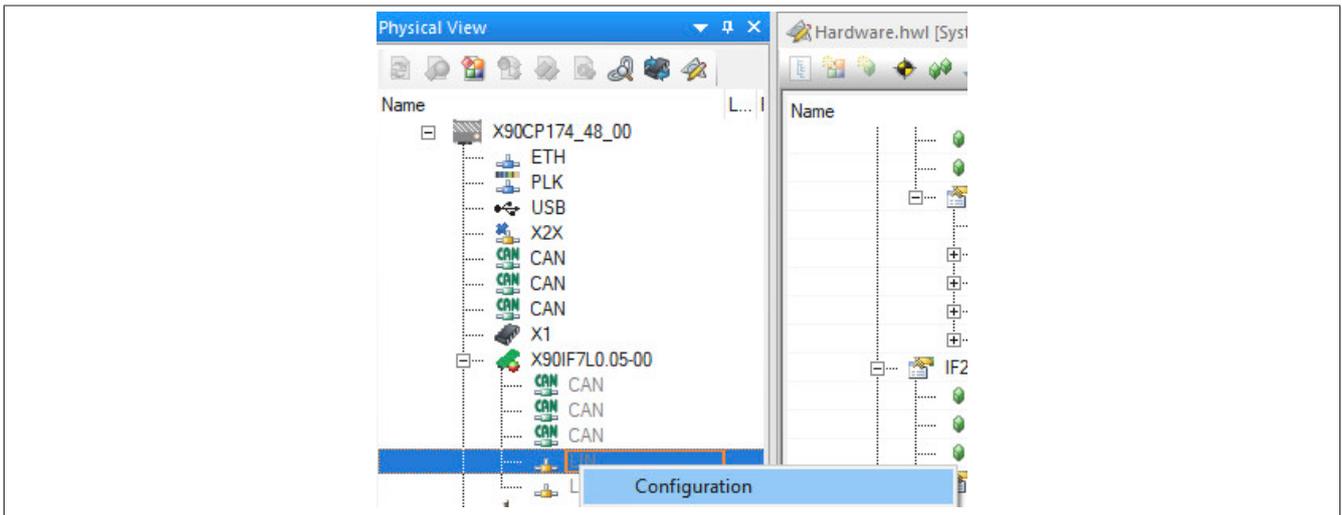
- Die SPS muss um ein LIN-Schnittstellenmodul erweitert werden. Um den Hardwarekatalog zu öffnen, mit der rechten Maustaste in die entsprechende Spalte in der Physical View klicken und "Hardwaremodul hinzufügen" wählen.



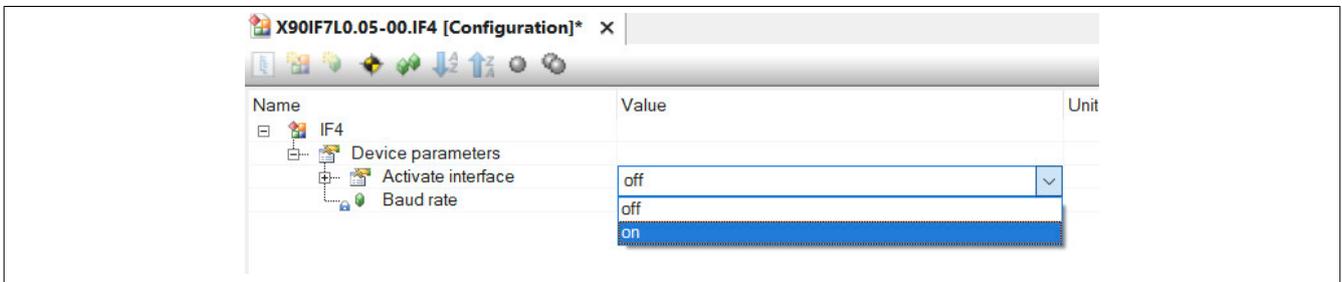
- Mittels Drag & Drop bzw. Doppelklick auf die Schnittstellenkarte das Modul in das Projekt einfügen.



- Mit Rechtsklick auf die LIN-Schnittstelle und Auswahl von "Konfiguration" die Projektierungsumgebung öffnen.



- Den Wert für **Schnittstelle aktivieren** auf **ein** setzen, um die Schnittstelle zu aktivieren.

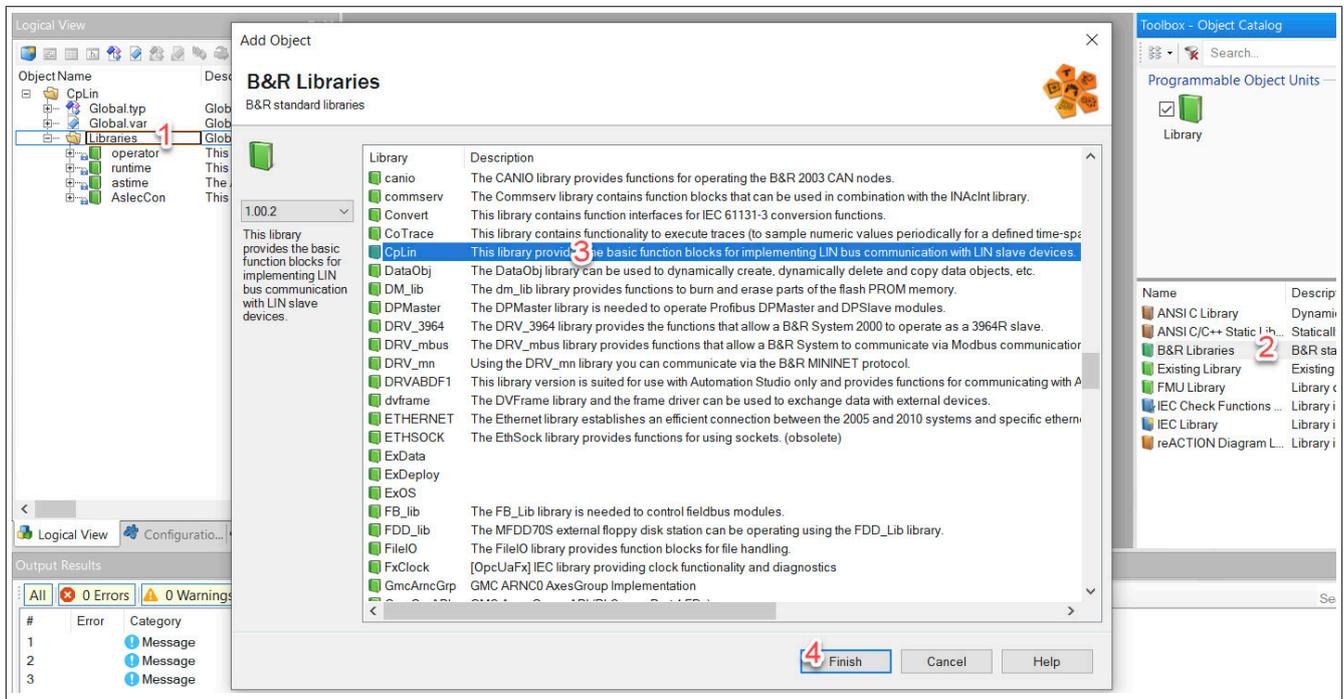


4.1.3 Bibliothek hinzufügen

Die Bibliothek CpLin wird unter dem Knoten "Bibliotheken" in der Logical View über den Objektkatalog hinzugefügt.

- 1) In der Logical View eine Bibliothek auswählen.
- 2) In der Kategorieliste des Objektkatalogs "Programmierbare Organisationseinheiten" auswählen und "B&R Bibliotheken" doppelklicken.
- 3) Die Bibliothek CpLin in der Objektliste des Objektkatalogs auswählen.

Durch Doppelklick auf die Bibliothek CpLin wird diese unter der aktuellen Auswahl hinzugefügt. Wenn Drag & Drop anstelle eines Doppelklicks verwendet wird, kann die Position, an der das Objekt hinzugefügt wird, variieren.



4.2 Beispiel 1: Marquardt-Wippschalter-LED auswählen

Dieses Beispiel beschreibt, wie die Bibliothek CpLin zur Auswahl einer Marquardt-Wippschalter-LED (Serie 3270) verwendet werden kann.

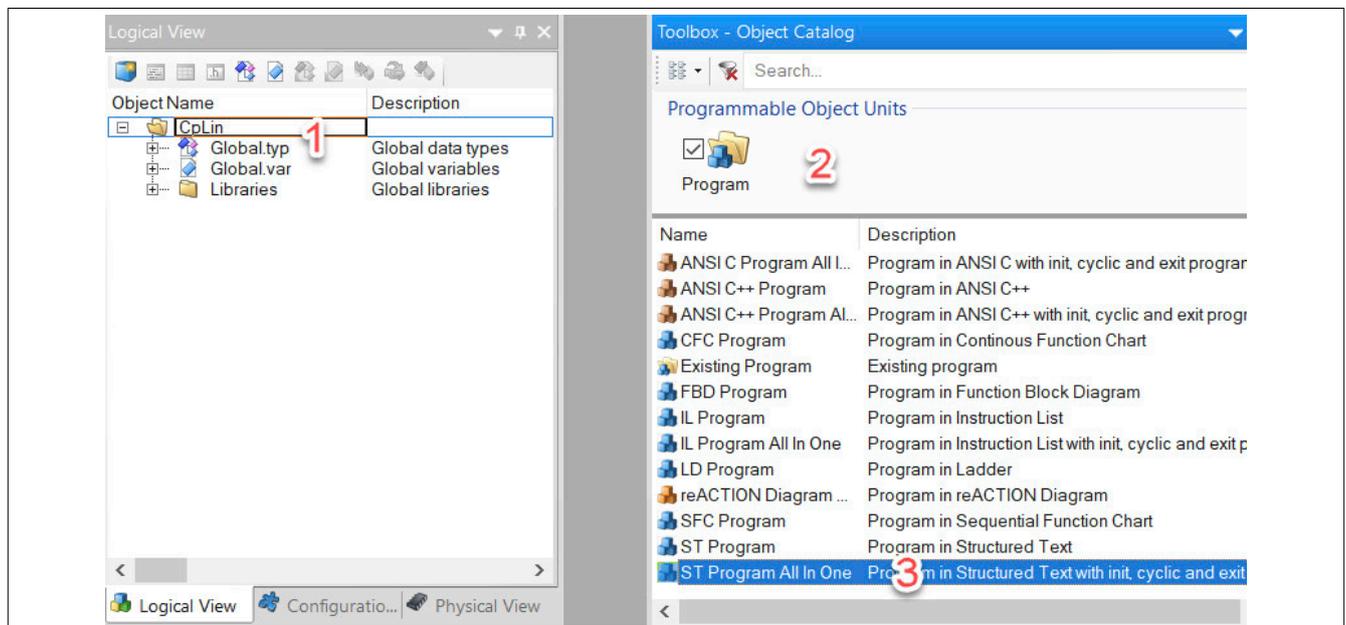
Das Beispielprogramm hat die folgende Funktionalität:

Wenn der obere Kontakt geschlossen ist, leuchtet die obere grüne LED:	Wenn der untere Kontakt geschlossen ist, leuchtet die untere grüne LED:
	

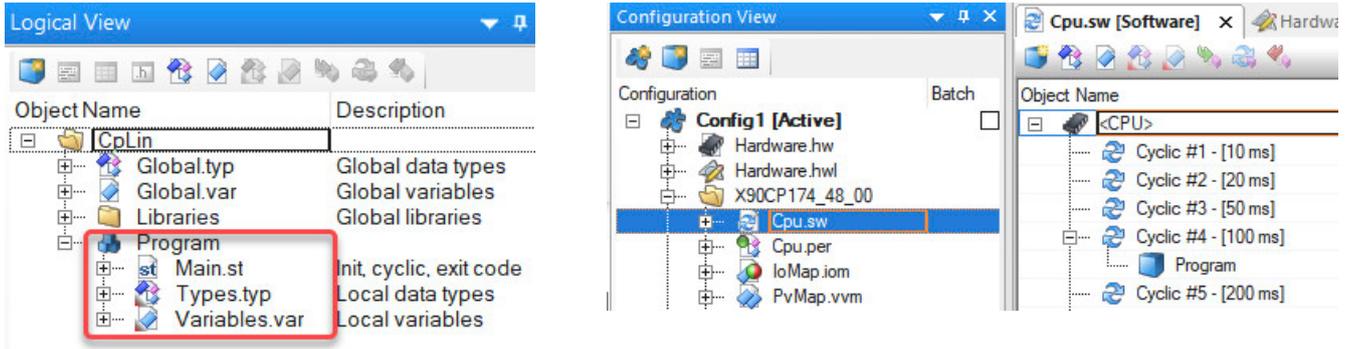
4.2.1 Programm hinzufügen

- Das Programm wird in der Logical View mit Hilfe des Objektkatalogs unter dem Knoten Projektname hinzugefügt.
 - 1) In der Logical View den Projektnamen wählen.
 - 2) In der Kategorieliste des Objektkatalogs "Programmierbare Organisationseinheiten" auswählen und "Programm" wählen.
 - 3) "ST Program All In One" in der Objektliste des Objektkatalogs auswählen.

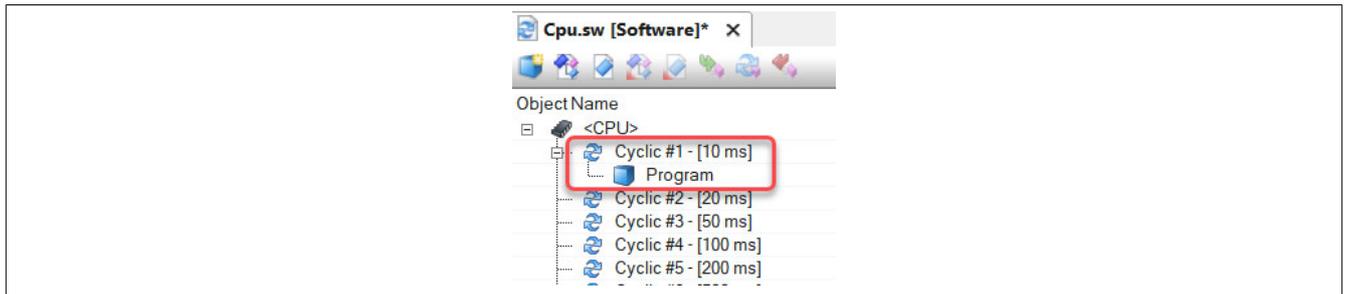
Durch Doppelklick auf das Programm wird es unter der aktuellen Auswahl hinzugefügt. Wenn Drag & Drop anstelle eines Doppelklicks verwendet wird, kann die Position, an der das Objekt hinzugefügt wird, variieren.



Das neue Programm wird in der Logical View und in der Configuration View unter "Software" hinzugefügt.



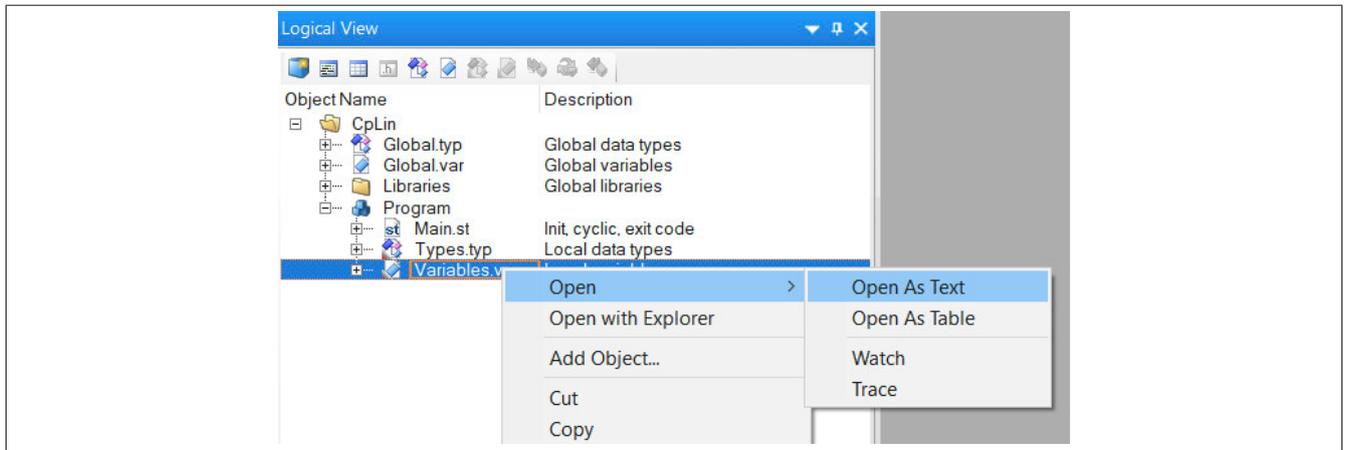
- Die Taskklasse des Programms von "Cyclic #4" auf "Cyclic #1" ändern.



4.2.2 Quellcode hinzufügen

Variablendeklaration

- Das Variablenfenster durch Rechtsklick auf **Variables.var** und Auswahl von "Öffnen → Öffnen als Text" öffnen.

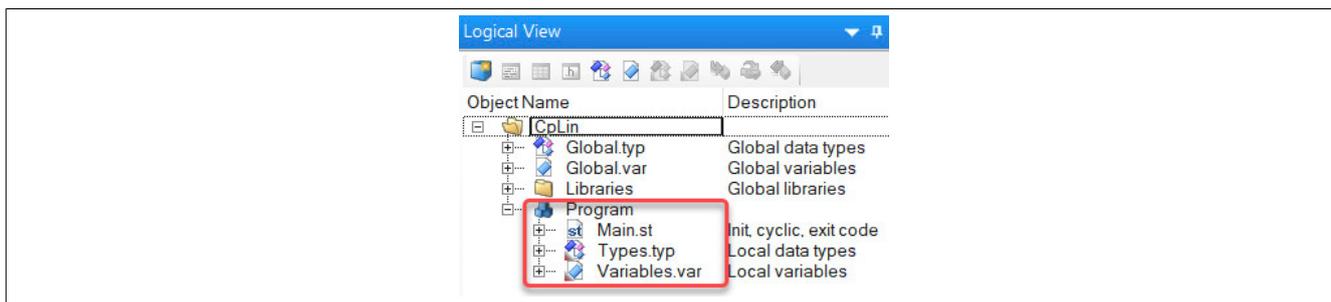


- Den folgenden Inhalt kopieren und in das Variablenfenster einfügen.

```
VAR
  nbReceivedFrames : UDINT; (*number of received frames*)
  CpLinScheduler_0 : CpLinScheduler;
  Schedule         : ARRAY[0..4] OF CpLinScheduleItemType; (*Schedule*)
  nbErrors         : UDINT; (*error counter*)
  lastErrorID     : DINT; (*last error number*)
  lpb_001_a       : BOOL; (*1 = top switch closed*)
  lpb_001_b       : BOOL; (*1 = bottom switch closed*)
  nbErrorFlags    : UDINT; (*counter of error frames*)
  lastErrorFlag   : CpLinErrorFlagsEnum; (*error flag of last error frame*)
  lpb_001_led_b_green_brightness : USINT; (*bottom green LED brightness*)
  lpb_001_led_b_red_brightness   : USINT; (*bottom red LED brightness*)
  lpb_001_led_a_green_brightness : USINT; (*top green LED brightness*)
  lpb_001_led_a_red_brightness   : USINT; (*top red LED brightness*)
  lpb_001_led_b_blue_brightness  : USINT; (*bottom blue LED brightness*)
  lpb_001_led_a_blue_brightness  : USINT; (*top blue LED brightness*)
  lpb_background_lighting        : USINT; (*background lighting*)
END_VAR
```

Programmcode hinzufügen

- Das Programmfenster durch Doppelklick oder Rechtsklick auf **Main.st** und Auswahl von "Öffnen → Öffnen als Text" öffnen.



- Den folgenden Inhalt kopieren und in das Codefenster einfügen.

Der Quellcode ist in 5 Abschnitte unterteilt:

- Scheduler-Konfiguration
- Lesen der Eingangsdaten
- Logische Operationen
- Schreiben der Ausgangsdaten
- Fehlerauswertung

```
(*****
* Copyright: B&R Industrial Automation GmbH
* Author:    B&R
* Created:   March 24, 2023/4:06 PM
* Description: Marquardt Rocker Switch Series 3270 example
*****)

PROGRAM _INIT

    CpLinScheduler_0.Enable := TRUE;

    ...

    (* Error evaluation *)
    IF CpLinScheduler_0.Error THEN
        nbErrors := nbErrors + 1;
        lastErrorID := CpLinScheduler_0.ErrorID;
    END_IF

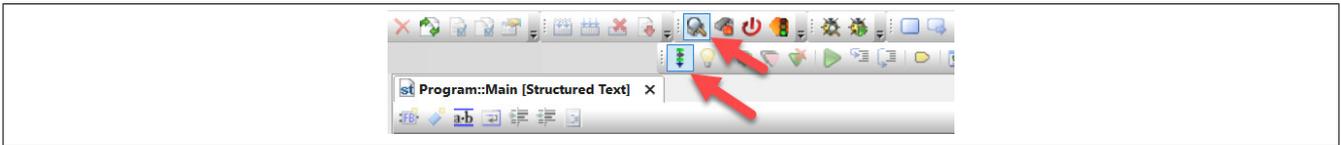
END_PROGRAM
```

Information:

Das vollständige Codebeispiel ist im entsprechenden Abschnitt der Automation Help zu finden.

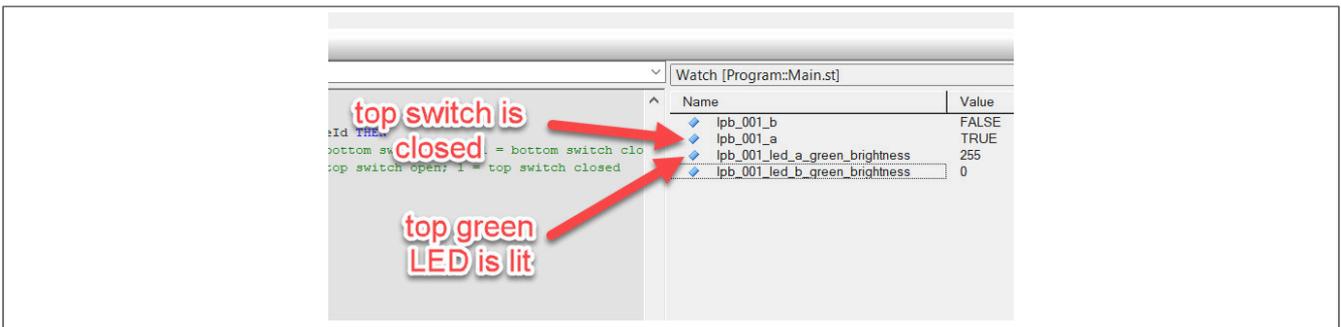
4.2.3 Programm testen

Das Projekt kann nach der Übertragung auf die SPS getestet werden. Den Monitor und die Zeilenüberwachung aktivieren.



Der Zustand der Variablen kann im Variablenmonitor überprüft werden:

- Der Zustand des oberen Kontakts wird vom LIN-Bus in die Variable **lpb_001_a** geschrieben.
- Die erforderliche Helligkeit der oberen grünen LED wird über die Variable **lpb_001_led_a_green_brightness** an den LIN-Bus gesendet.



Wenn der obere Kontakt geschlossen ist, leuchtet die obere grüne LED.



4.3 Beispiel 2: Daten von einem Audiowell-Parksensor lesen

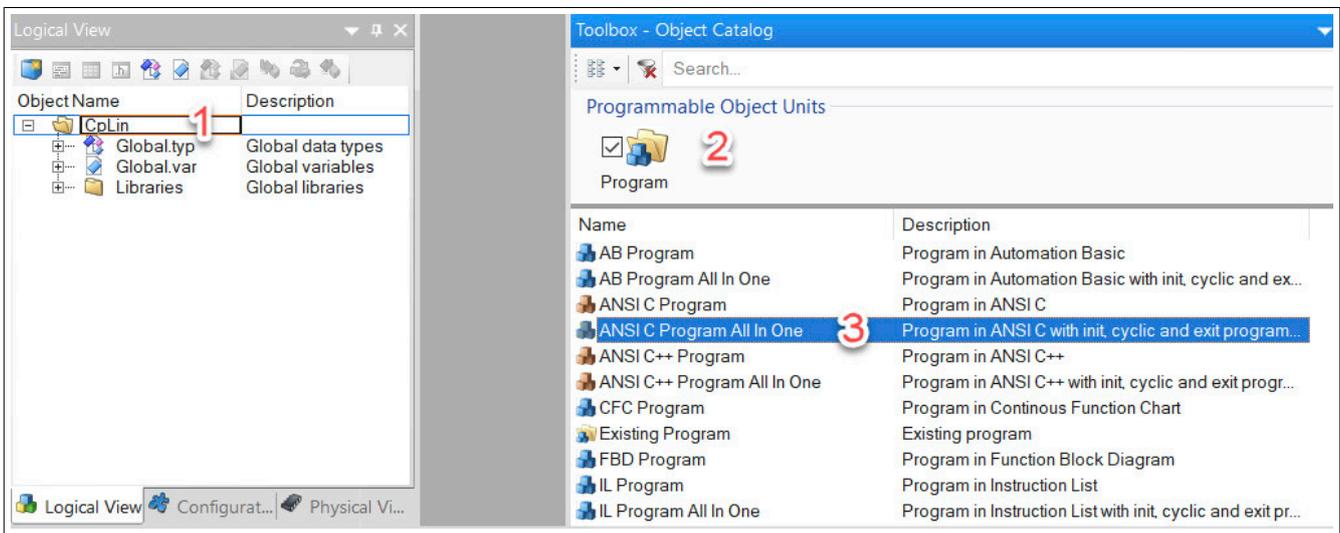
Dieses Beispiel beschreibt, wie die Bibliothek CpLin zum Lesen von Daten von einem Audiowell-Parksensor verwendet werden kann.



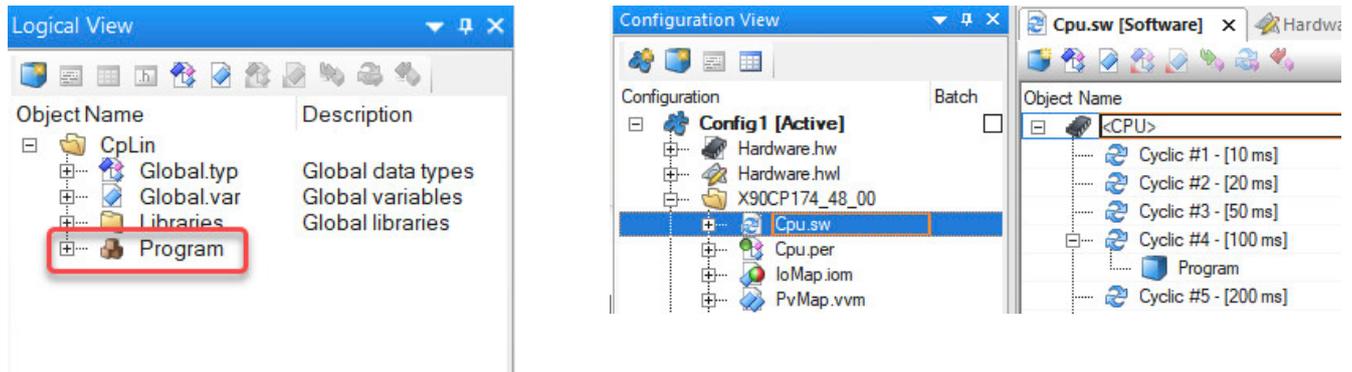
4.3.1 Programm hinzufügen

- Das Programm wird in der Logical View mit Hilfe des Objektkatalogs unter dem Knoten Projektname hinzugefügt.
 - 1) In der Logical View den Projektnamen wählen.
 - 2) In der Kategorieliste des Objektkatalogs "Programmierbare Organisationseinheiten" auswählen und "Programm" wählen.
 - 3) "Ansi C Program All In One" in der Objektliste des Objektkatalogs auswählen.

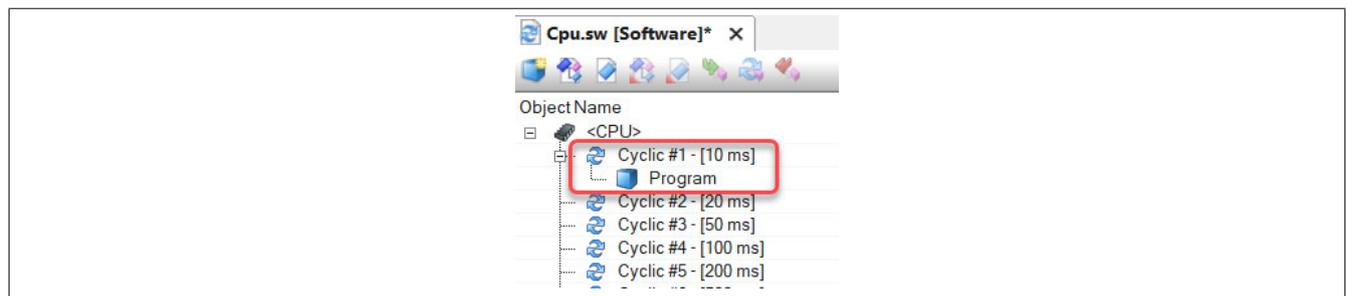
Durch Doppelklick auf das Programm wird es unter der aktuellen Auswahl hinzugefügt. Wenn Drag & Drop anstelle eines Doppelklicks verwendet wird, kann die Position, an der das Objekt hinzugefügt wird, variieren.



Das neue Programm wird in der Logical View und in der Configuration View unter "Software" hinzugefügt.



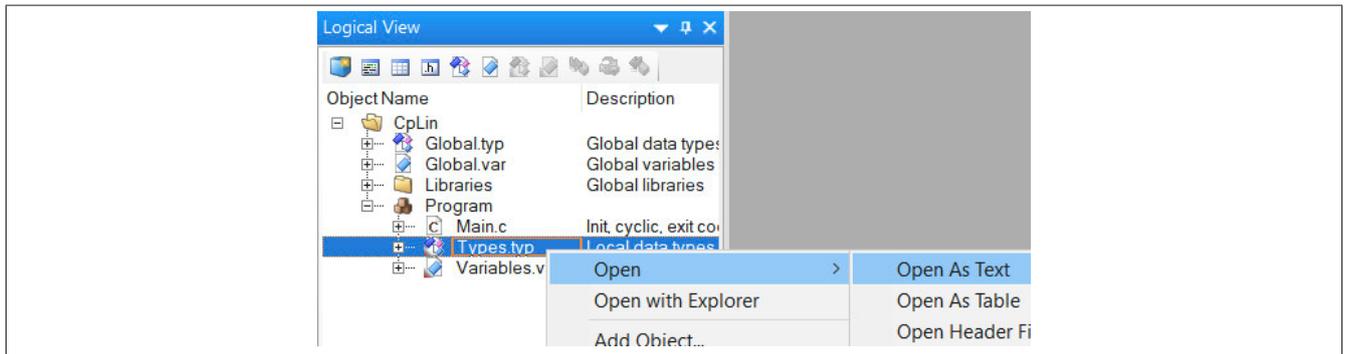
- Die Taskklasse des Programms von "Cyclic #4" auf "Cyclic #1" ändern.



4.3.2 Quellcode hinzufügen

Datentypendeklaration

- Zur Dekodierung der empfangenen Daten müssen Datentypen definiert werden. Das Deklarationsfenster durch Rechtsklick auf **Types.typ** und Auswahl von "Öffnen → Öffnen als Text" öffnen.



- Den folgenden Inhalt kopieren und in das Deklarationsfenster einfügen.

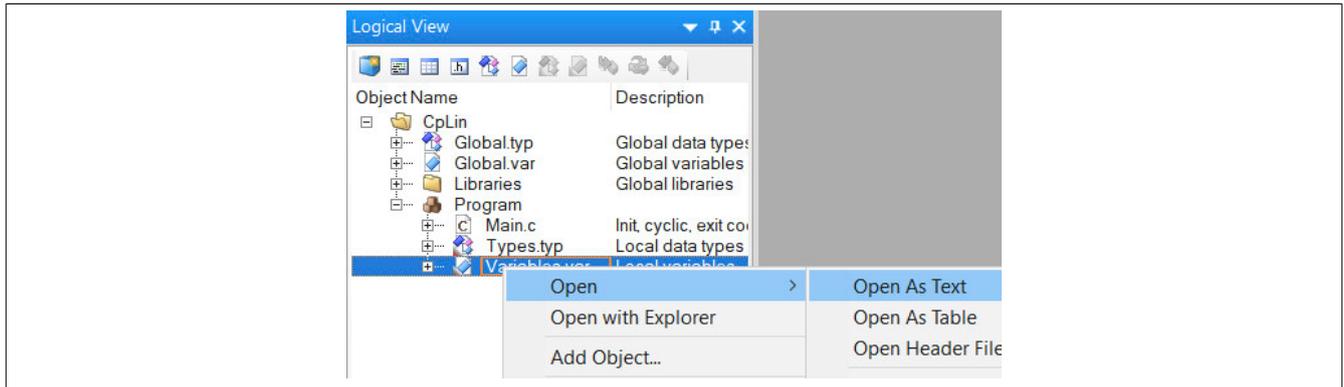
```

TYPE
    UltrasonicSensor_type : STRUCT
        Echo_1_Flight_Time      : REAL;
        Echo_1_Signal_Amplitude : USINT;
        Echo_1_Signal_Width    : REAL;
        Echo_1_Spike           : REAL;
        Echo_2_Flight_Time     : REAL;
        Echo_2_Signal_Amplitude : USINT;
        Echo_2_Signal_Width    : REAL;
        Echo_2_Spike           : REAL;
        AfterShock              : REAL;
        PowerDetection          : SensorPower_enum;
    END_STRUCT;
    SensorPower_enum :
    (
        ABOVE_7V := 0,
        BELOW_7V := 1
    );
END_TYPE

```

Variablendeklaration

- Das Variablenfenster durch Rechtsklick auf **Variables.var** und Auswahl von "Öffnen → Öffnen als Text" öffnen.



- Den folgenden Inhalt kopieren und in das Variablenfenster einfügen.

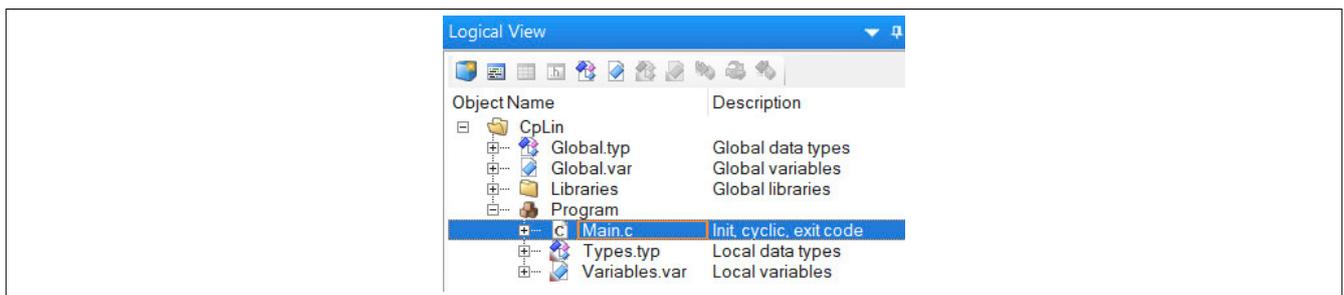
```

VAR
  CplInScheduler_0   : CplInScheduler;
  InitSchedule       : ARRAY[0..9] OF CplInScheduleItemType;
  lastErrorFlag      : CplInErrorFlagsEnum;
  lastErrorID        : DINT;
  lastErrorSlotIndex : USINT;
  LoopSchedule       : ARRAY[0..5] OF CplInScheduleItemType;
  nbErrorFlags       : UDINT;
  nbErrors            : UDINT;
  nbReceivedFrames   : UDINT;
  Sensor_1           : UltrasonicSensor_type;
  Sensor_2           : UltrasonicSensor_type;
  state              : INT;
END_VAR

```

Programmcode hinzufügen

- Das Programmfenster durch Doppelklick oder Rechtsklick auf **Main.c** und Auswahl von "Öffnen → Öffnen als Text" öffnen.



- Den folgenden Inhalt kopieren und in das Codefenster einfügen.

Der Quellcode ist in 4 Abschnitte unterteilt:

- Scheduler-Konfiguration - Initialisierung
- Scheduler-Konfiguration - Zyklischer Datenaustausch
- Lesen der Eingangsdaten
- Fehlerauswertung

```

/*****
 * Copyright: B&R Industrial Automation GmbH
 * Author:    B&R
 * Created:   March 24, 2023/4:06 PM
 * Description: Audiowell parking sensor example
 *****/

#ifdef _DEFAULT_INCLUDES
#include <AsDefault.h>
#endif

#include <string.h>

void parseSensorData (struct UltrasonicSensor_type * sensor, USINT raw[8])
{
    ...
}

/* Error evaluation */

if (CpLinScheduler_0.Error) {
    // general error
    nbErrors = nbErrors + 1;
    lastErrorID = CpLinScheduler_0.ErrorID;
    lastErrorSlotIndex = CpLinScheduler_0.SlotIndex;
}
}

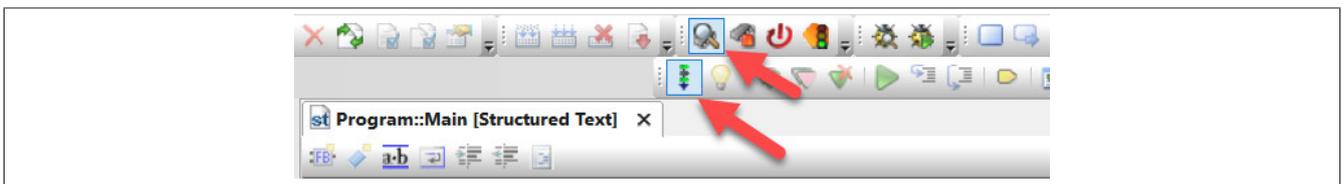
```

Information:

Das vollständige Codebeispiel ist im entsprechenden Abschnitt der Automation Help zu finden.

4.3.3 Programm testen

Das Projekt kann nach der Übertragung auf die SPS getestet werden. Den Monitor und die Zeilenüberwachung aktivieren.



Die vom ersten Sensor gelesenen Daten werden im Variablenmonitor angezeigt.

Watch [Program::Main.c]	
Name	Value
Sensor_1	
Echo_1_Flight_Time	10.624
Echo_1_Signal_Amplitude	252
Echo_1_Signal_Width	1.408
Echo_1_Spike	11.264
Echo_2_Signal_Amplitude	1.196
Echo_2_Signal_Width	0.128
Echo_2_Spike	12.096
AfterShock	1.344
PowerDetection	ABOVE_7V

data from sensor 1