

be in motion **be in motion**

BM4-O-PLC-01

b maXX drive PLC

Applikationshandbuch

D

5.02004.04



Titel	Applikationshandbuch
Produkt	b maXX drive PLC BM4-O-PLC-01
Stand	01. März 2007
Copyright	<p>Dieses Applikationshandbuch darf vom Eigentümer ausschließlich für den internen Gebrauch in beliebiger Anzahl kopiert werden. Für andere Zwecke darf dieses Applikationshandbuch auch auszugsweise weder kopiert noch vervielfältigt werden.</p> <p>Verwertung und Mitteilung von Inhalten dieses Applikationshandbuches sind nicht gestattet.</p> <p>Bezeichnungen bzw. Unternehmenskennzeichen in diesem Applikationshandbuch können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen kann.</p>
Verbindlichkeit	<p>Dieses Applikationshandbuch ist Teil des Gerätes/der Maschine. Dieses Applikationshandbuch muss jederzeit für den Bediener zugänglich und in einem leserlichen Zustand sein. Bei Verkauf/Verlagerung des Gerätes/der Maschine muss dieses Applikationshandbuch vom Besitzer zusammen mit dem Gerät/der Maschine weitergegeben werden. Nach Verkauf des Gerätes/der Maschine sind dieses Original und sämtliche Kopien an den Käufer zu übergeben. Nach Entsorgung oder anderem Nutzungsende sind dieses Original und sämtliche Kopien zu vernichten.</p> <p>Mit der Übergabe des vorliegenden Applikationshandbuches werden entsprechende Applikationshandbücher mit einem früheren Stand außer Kraft gesetzt.</p> <p>Bitte beachten Sie, dass Angaben/Zahlen/Informationen aktuelle Werte zum Druckdatum sind. Zur Ausmessung, Berechnung und Kalkulationen sind diese Angaben nicht rechtlich verbindlich.</p> <p>Die Firma Baumüller Nürnberg GmbH behält sich vor, im Rahmen der eigenen Weiterentwicklung der Produkte die technischen Daten und die Handhabung von Baumüller-Produkten zu ändern.</p> <p>Es kann jedoch keine Gewährleistung bezüglich der Fehlerfreiheit dieses Applikationshandbuches, soweit nicht in den Allgemeinen Verkaufs- und Lieferbedingungen anders beschrieben, übernommen werden.</p>
Hersteller	<p>Baumüller Nürnberg GmbH Ostendstr. 80 - 90 90482 Nürnberg Deutschland Tel. +49 9 11 54 32 - 0 Fax: +49 9 11 54 32 - 1 30 www.baumueller.de</p>



Inhaltsverzeichnis

1	Einleitung	5
1.1	Erste Schritte	5
1.2	Verwendete Begriffe	5
2	Grundlegende Sicherheitshinweise	7
2.1	Gefahrenhinweise und Gebote	7
2.1.1	Struktur eines Gefahrenhinweises	8
2.1.2	Verwendete Gefahrenhinweise	9
2.2	Infozeichen	11
2.3	Rechtliche Hinweise	11
2.4	Bestimmungsgemäße Verwendung	11
2.5	Sachwidrige Verwendung	12
2.6	Schutzeinrichtungen	12
2.7	Ausbildung des Personals	13
2.8	Sicherheitsmaßnahmen im Normalbetrieb	13
2.9	Verpflichtung und Haftung	13
2.9.1	Gefahrenhinweise und Sicherheitshinweise beachten	14
2.9.2	Gefahren im Umgang mit diesem Modul	14
2.9.3	Gewährleistung und Haftung	14
3	Inbetriebnahme	15
3.1	Allgemeine Sicherheitsvorschriften	15
3.2	Anforderungen an das ausführende Personal	15
3.3	Beschreibung/Überprüfung der Sicherheits- und Überwachungseinrichtungen	16
3.4	Beschreibung und Überprüfung der Bedienungs- und Anzeigeelemente	16
3.4.1	LEDs zur Anzeige von Betriebszuständen der BM4-O-PLC-01	16
3.4.2	Schalter/Taster S1 zum Wechsel von Betriebszuständen der BM4-O-PLC-01	17
3.5	Ablauf der Inbetriebnahme	18
3.5.1	Einschalten	18
3.5.2	Testen der Funktion	19
3.5.3	Bootprojekt senden	19
4	Betrieb	25
4.1	Programmiersysteme PROPROG wt II und ProProg wt III	25
4.2	b maXX drive PLC Projekt	27
4.3	b maXX drive PLC Konfiguration	27
4.4	b maXX drive PLC Ressource	28
4.4.1	Kommunikation und Verbindung	30
4.4.2	Kontrolldialog für Ressourcen	33
4.4.3	Datenbereich	41
4.4.4	Die b maXX drive PLC Event-Tasks	45
4.5	b maXX drive PLC Anwenderbibliotheken	47
4.5.1	b maXX drive PLC Firmware	49
4.5.2	b maXX drive PLC Board Funktionen	50
4.5.3	Die b maXX drive PLC Datentypen	55
4.5.4	Die Standard Funktionsbaustein-Bibliotheken	59
4.5.5	Die b maXX drive PLC Technologiebausteine	60
4.5.6	Einfügen einer Anwenderbibliothek in ein Projekt	60
4.6	BACI-Systembeschreibung für die b maXX drive PLC	62
4.6.1	Direkter Zugriff auf die BACI-HW von der b maXX drive PLC aus	62



Inhaltsverzeichnis

4.6.2	Prozessdatenkommunikation zwischen b maXX Regler und b maXX drive PLC, Synchronisierung auf eine gemeinsame BACI-Hardware-Signalquelle	63
4.6.3	Für die Applikation nutzbare Interruptquellen der b maXX drive PLC	65
4.6.4	BACI-Signalgenerierung über den Funktionsbaustein TIMER_A_INIT	67
4.6.5	Funktionsbaustein "TIMER_A_INIT"	68
4.6.6	Konfigurationsbeispiele	74
4.6.7	Beispiel A: Umsetzung einer BACI-Prozessdatenkommunikation innerhalb einer BACI-Event-Task.	76
4.6.8	Beispiel B: Umsetzung einer BACI-Prozessdatenkommunikation über ein selbst generiertes SYNC1-Signal.	79
4.6.9	Beispiel C: Umsetzung einer BACI-Prozessdatenkommunikation über ein externes SYNC1-Signal	83
4.6.10	Beispiel D: Realisierung einer einfachen seriellen RS485-Kommunikation über einen CPU-Timer.	85
4.6.11	Beschreibung der Bausteine für die serielle RS485-Kommunikation über die Schnittstelle X1 (9-pol. Sub-D-Buchse) auf der b maXX drive PLC.	87
4.6.12	Funktionsbaustein TER_INIT.	89
4.6.13	Funktionsbaustein TER_R.	94
4.6.14	Funktionsbaustein TER_S.	97
4.6.15	Funktionsbaustein T64_RSYN	99
4.6.16	Funktionsbaustein T64_REC.	101
4.6.17	Funktionsbaustein TER_USS	104
4.7	Codelaufzeiten	110
4.7.1	Funktionsbaustein TIME_MEASURE_START.	112
4.7.2	Funktionsbaustein TIME_MEASURE_END.	113
4.8	Praktische Hinweise.	114
4.8.1	SPS-Fehler "Watchdog in Task 'x' überschritten!".	114
5	Interne Kommunikationsschnittstelle zum Regler (BACI)	117
5.1	BACI (BAumüller-Component-Interface) allgemein	117
5.2	ProMaster, ProProg wt III und Motion Control	118
5.2.1	Voraussetzungen.	118
5.2.2	Durchzuführende Schritte	119
5.2.3	Inbetriebnahme des b maXX 4400 mit b maXX Regler und b maXX drive PLC	119
5.2.4	Anlegen eines IEC Projekts mit ProProg wt III.	119
5.2.5	Anlegen einer Maschinenkonfiguration in ProMaster	121
5.2.6	Konfigurieren der EtherCAT Kommunikation mit ProEtherCAT.	126
5.2.7	Konfigurieren der BACI-Kommunikation	127
5.2.8	Konfigurieren der PLC	132
5.2.9	Download der Daten auf den EtherCAT-Master und die b maXX drive PLC.	141
5.2.10	Programmieren des IEC Projekts	142
5.3	PROPROG wt II	146
5.3.1	b maXX drive PLC und Regler.	146
5.3.2	Programmierung der BACI-Kommunikation auf der b maXX drive PLC	146
5.3.3	Funktionsbausteine für BACI - Übersicht.	147
Anhang A - Abkürzungen		167
Stichwortverzeichnis		169

EINLEITUNG

Dieses Applikationshandbuch b maXX drive PLC ist ein wichtiger Bestandteil ihres b maXX 4400 Gerätes; lesen Sie daher nicht zuletzt im Interesse Ihrer eigenen Sicherheit diese Dokumentation komplett durch.

Es wird vorausgesetzt, dass Sie die Betriebsanleitung b maXX drive PLC verstanden und angewendet haben.

In diesem Kapitel beschreiben wir die ersten Schritte, die Sie nach Erhalt des Gerätes ausführen sollten. Wir definieren Begriffe, die in dieser Dokumentation durchgängig verwendet werden, und informieren Sie über Verpflichtungen, die beim Einsatz dieses Gerätes beachtet werden müssen.

1.1 Erste Schritte

- ◆ Überprüfen Sie die Lieferung.
- ◆ Leiten Sie alle Unterlagen, die mit dem Steckmodul geliefert wurden, an die entsprechenden Stellen in Ihrem Unternehmen weiter.
- ◆ Stellen Sie das geeignete Personal für Montage und Inbetriebnahme bereit.
- ◆ Übergeben Sie diese Betriebsanleitung an das Personal und stellen Sie sicher, dass insbesondere die hier angegebenen Sicherheitshinweise verstanden und befolgt werden können.

1.2 Verwendete Begriffe

Für das Baumüller-Produkt „**b maXX drive PLC**“ werden wir in dieser Dokumentation auch die Begriffe „Optionsmodul“, „Steckmodul“, „b maXX PLC“ oder nur „PLC“ verwenden. Weiterhin wird der Begriff „BM4-O-PLC-01“ verwendet.

Eine Liste der verwendeten Abkürzungen finden Sie in [►Anhang A - Abkürzungen◄](#) ab Seite 167.

Für das Produkt „Grundgerät b maXX 4400“ wird auch der Begriff „b maXX“ verwendet. Der Regler im Grundgerät wird auch „b maXX Regler“ genannt.

GRUNDLEGENDE SICHERHEITS- HINWEISE

Jedes Baumüller-Steckmodul haben wir nach strengen Sicherheitsvorgaben konstruiert und gefertigt. Trotzdem kann die Arbeit mit dem Steckmodul für Sie gefährlich sein.

In diesem Kapitel beschreiben wir Gefahren, die bei der Arbeit mit dem Baumüller-Steckmodul auftreten können. Gefahren verdeutlichen wir mit Symbolen (Icons). Alle in dieser Dokumentation verwendeten Symbole werden wir auflisten und erklären.

Wie Sie sich vor den einzelnen Gefahren im konkreten Fall schützen können, können wir in diesem Kapitel nicht erklären. In diesem Kapitel geben wir ausschließlich allgemeine Schutzmaßnahmen. Die konkreten Schutzmaßnahmen werden wir in den nachfolgenden Kapiteln immer direkt nach dem Hinweis auf die Gefahr geben.

2.1 Gefahrenhinweise und Gebote



WARNUNG (WARNING)

Folgendes **kann eintreffen**, wenn Sie diesen Gefahrenhinweis nicht beachten:

- schwere Körperverletzung
- Tod

Gefahrenhinweise zeigen Ihnen Gefahren, die zu Verletzungen oder sogar zu Ihrem Tod führen können.

Beachten Sie immer die in dieser Dokumentation angegebenen Gefahrenhinweise.

Eine Gefahr teilen wir immer in eine der drei Gefahrenklassen ein. Jede Gefahrenklasse wird durch eines der folgenden Signalwörter gekennzeichnet:

GEFAHR (DANGER)

- erheblicher Sachschaden
- schwere Körperverletzung
- Tod - **wird** eintreffen

WARNUNG (WARNING)

- erheblicher Sachschaden
- schwere Körperverletzung
- Tod - **kann** eintreffen

VORSICHT (CAUTION)

- Sachschaden • leichte bis mittlere Körperverletzung - **kann** eintreffen

2.1.1 Struktur eines Gefahrenhinweises

Die nachfolgenden zwei Beispiele zeigen den prinzipiellen Aufbau eines Gefahrenhinweises. Ein Dreieck wird verwendet, wenn vor einer Gefahr für Lebewesen gewarnt wird. Fehlt das Dreieck, beziehen sich die Gefahrenhinweise ausschließlich auf Sachschäden.



Ein Dreieck zeigt, dass hier eine Gefahr für Lebewesen ist.
Die Farbe der Umrandung zeigt, wie groß die Gefahr ist - je dunkler die Farbe, desto größer ist die Gefahr.



Das Icon im Viereck stellt die Gefahr dar.
Die Farbe der Umrandung zeigt, wie groß die Gefahr ist - je dunkler die Farbe, desto größer ist die Gefahr.



Das Icon im Kreis stellt ein Gebot dar. Dieses Gebot muss der Anwender befolgen.
(Der Kreis ist gestrichelt dargestellt, weil nicht bei jedem Gefahrenhinweis ein Gebot als Icon vorhanden ist.)



Der Kreis zeigt, dass eine Gefahr für Sachschaden existiert.



Das Icon im Viereck stellt die Gefahr dar.
Die Farbe der Umrandung zeigt, wie groß die Gefahr ist - je dunkler die Farbe, desto größer ist die Gefahr. (Das Viereck ist gestrichelt dargestellt, weil nicht bei jedem Gefahrenhinweis die Gefahr als Icon dargestellt wird)

Der Text neben den Icons ist folgendermaßen aufgebaut:

HIER STEHT DAS SIGNALWORT, WELCHES DEN GRAD DER GEFAHR ANZEIGT




Hier schreiben wir, ob eine oder mehrere der untenstehenden Folgen eintreffen, wenn dieser Warnhinweis nicht beachtet wird.


- hier beschreiben wir die möglichen Folgen. Die schlimmste Folge steht ganz rechts.

Hier beschreiben wir die Gefahr.

Hier beschreiben wir, was Sie tun können, um die Gefahr zu vermeiden.


2.1.2 Verwendete Gefahrenhinweise

Steht vor einem Signalwort ein Gefahrzeichen:  oder  oder , dann bezieht sich der Sicherheitshinweis auf Personenschaden.

Steht vor einem Signalwort ein rundes Gefahrzeichen: , dann bezieht sich der Sicherheitshinweis auf Sachschaden.

2.1.2.1 Gefahrenhinweise vor Personenschaden

Zur optischen Unterscheidung verwenden wir für jede Klasse von Gefahrenhinweisen eine eigenen Umrandung für die dreieckigen Gefahrzeichen und die viereckigen Piktogramme.

Für die Gefahrenklasse **GEFAHR** (DANGER) verwenden wir das Gefahrzeichen . Folgende Gefahrenhinweise dieser Gefahrenklasse verwenden wir in dieser Dokumentation.

GEFAHR (DANGER)



Folgendes **wird eintreffen**, wenn Sie diesen Warnhinweis nicht beachten:

- schwere Körpervletzung
- Tod

*Die Gefahr ist: **Elektrizität**. Hier wird die Gefahr gegebenenfalls genauer beschrieben.*

Hier beschreiben wir, was Sie tun können, um die Gefahr zu vermeiden.

GEFAHR (DANGER)




Folgendes **wird eintreffen**, wenn Sie diesen Gefahrenhinweis nicht beachten:

- schwere Körpervletzung
- Tod

*Die Gefahr ist: **mechanische Einwirkung**. Hier wird die Gefahr gegebenenfalls genauer beschrieben.*

Hier beschreiben wir, was Sie tun können, um die Gefahr zu vermeiden.

Für die Gefahrenklasse **WARNUNG** (WARNING) verwenden wir das Gefahrzeichen . Folgende Gefahrenhinweise dieser Gefahrenklasse verwenden wir in dieser Dokumentation.

WARNUNG (WARNING)




Folgendes **kann eintreffen**, wenn Sie diesen Gefahrenhinweis nicht beachten:

- schwere Körpervletzung
- Tod

*Die Gefahr ist: **Elektrizität**. Hier wird die Gefahr gegebenenfalls genauer beschrieben.*

Hier beschreiben wir, was Sie tun können, um die Gefahr zu vermeiden.

Für die Gefahrenklasse **VORSICHT** (CAUTION) verwenden wir das Gefahrzeichen . Folgende Gefahrenhinweise dieser Gefahrenklasse verwenden wir in dieser Dokumentation.

2.1 Gefahrenhinweise und Gebote



VORSICHT (CAUTION)

Folgendes **kann eintreffen**, wenn Sie diesen Gefahrenhinweis nicht beachten:

- leichte bis mittlere Körperverletzung

*Die Gefahr ist: **scharfe Kanten**. Hier wird die Gefahr gegebenenfalls genauer beschrieben.*

Hier beschreiben wir, was Sie tun können, um die Gefahr zu vermeiden.



VORSICHT (CAUTION)

Folgendes **kann eintreffen**, wenn Sie diesen Warnhinweis nicht beachten:

- Umweltverschmutzung

*Die Gefahr ist: **unsachgemäße Entsorgung**. Hier wird die Gefahr gegebenenfalls genauer beschrieben.*

Hier beschreiben wir, was Sie tun können, um die Gefahr zu vermeiden.

2.1.2.2 Gefahrenhinweise vor Sachschaden

Steht vor einem Signalwort ein rundes Gefahrzeichen: ⓘ dann bezieht sich der Sicherheitshinweis auf Sachschaden.



VORSICHT (CAUTION)

Folgendes **kann eintreffen**, wenn Sie diesen Gefahrenhinweis nicht beachten:

- Sachschaden

*Die Gefahr ist: **elektrostatische Entladung**. Hier wird die Gefahr gegebenenfalls genauer beschrieben.*

Hier beschreiben wir, was Sie tun können, um die Gefahr zu vermeiden.

2.1.2.3 Verwendete Gebotszeichen



Sicherheitshandschuhe tragen



Sicherheitsschuhe tragen

2.2 Infozeichen



HINWEIS

Dieser Hinweis ist eine besonders wichtige Information.

2.3 Rechtliche Hinweise

Diese Dokumentation wendet sich an technisch qualifiziertes Personal, welches speziell ausgebildet ist und gründlich mit allen Warnungen und Instandhaltungsmaßnahmen vertraut ist.

Die Geräte sind nach dem Stand der Technik gefertigt und betriebssicher. Sie lassen sich gefahrlos installieren und in Betrieb setzen und funktionieren problemlos, wenn sichergestellt ist, dass die Hinweise der Dokumentation beachtet werden.

Der Benutzer trägt die Verantwortung für die Durchführung von Service und Inbetriebnahme gemäss den Sicherheitsvorschriften der geltenden Normen und allen anderen relevanten staatlichen oder örtlichen Vorschriften betreffend Leiterdimensionierung und Schutz, Erdung, Trennschalter, Überstromschutz usw.

Für Schäden, die bei der Montage oder beim Anschluss entstehen, haftet der Benutzer.

2.4 Bestimmungsgemäße Verwendung

Sie müssen das Steckmodul immer bestimmungsgemäß verwenden. Untenstehend haben wir einige wichtige Hinweise für Sie zusammengestellt. Die untenstehenden Hinweise sollen Ihnen ein Gefühl für die bestimmungsgemäße Verwendung des Steckmoduls geben. Mit den untenstehenden Hinweisen erheben wir keinen Anspruch auf Vollständigkeit - beachten Sie alle in dieser Betriebsanleitung gegebenen Hinweise.

- Sie dürfen das Steckmodul nur in Geräte der Reihe b maXX 4400 einbauen.
- Projektieren Sie die Anwendung so, dass Sie das Steckmodul immer innerhalb seiner Spezifikationen betreiben.
- Sorgen Sie dafür, dass ausschließlich qualifiziertes Personal mit diesem Steckmodul arbeitet.
- Montieren Sie das Steckmodul nur an dem/den vorgegebenen Steckplatz/Steckplätzen.
- Installieren Sie das Steckmodul so wie in es in dieser Dokumentation vorgegeben ist.
- Sorgen Sie dafür, dass die Anschlüsse immer den vorgegebenen Spezifikationen entsprechen.
- Betreiben Sie das Steckmodul nur, wenn es technisch einwandfrei ist.
- Betreiben Sie das Steckmodul immer in einer Umgebung, wie sie in den „Technischen Daten“ vorgeschrieben ist.
- Betreiben Sie das Steckmodul immer in serienmäßigem Zustand.
Aus Sicherheitsgründen dürfen Sie das Steckmodul nicht umbauen.

- Beachten Sie alle diesbezüglichen Hinweise, falls Sie das Steckmodul lagern.

Sie verwenden das Steckmodul dann bestimmungsgemäß, wenn Sie alle Hinweise und Informationen dieser Betriebsanleitung beachten.

2.5 Sachwidrige Verwendung

Im Folgenden listen wir einige Beispiele sachwidriger Verwendung auf. Die untenstehenden Hinweise sollen Ihnen ein Gefühl dafür geben, was eine sachwidrige Verwendung des Steckmoduls ist. Wir können aber nicht alle erdenklichen sachwidrigen Verwendungen hier auflisten. Alle Verwendungen, bei denen die Hinweise dieser Dokumentation missachtet werden, sind sachwidrig und somit verboten, insbesondere in folgenden Fällen:

- Sie haben das Steckmodul in andere Geräte als die Reihe b maXX 4400 eingebaut.
- Sie haben Hinweise dieser Betriebsanleitung missachtet.
- Sie haben das Steckmodul nicht bestimmungsgemäß verwendet.
- Sie haben das Steckmodul
 - unsachgemäß montiert,
 - unsachgemäß angeschlossen,
 - unsachgemäß in Betrieb genommen,
 - unsachgemäß bedient,
 - von nicht bzw. nicht ausreichend qualifiziertem Personal montieren, anschließen, in Betrieb nehmen und betreiben lassen,
 - überlastet,
- betrieben
 - mit defekten Sicherheitseinrichtungen,
 - mit nicht ordnungsgemäß angebrachten bzw. ohne Sicherheitsvorrichtungen,
 - mit nicht funktionsfähigen Sicherheits- und Schutzvorrichtungen
 - außerhalb der vorgeschriebenen Umgebungsbedingungen
- Sie haben das Steckmodul umgebaut, ohne dass dies schriftlich von der Firma Baumüller Nürnberg GmbH genehmigt wurde.
- Sie haben die Anweisungen bezüglich Wartung in den Komponentenbeschreibungen nicht beachtet.
- Sie haben das Steckmodul unsachgemäß mit Produkten anderer Hersteller kombiniert.
- Sie haben das Antriebssystem mit fehlerhaften und/oder fehlerhaft dokumentierten Produkten anderer Hersteller kombiniert.
- Ihre selbsterstellte Software der PLC enthält Programmierfehler, die zu einer Fehlfunktion führen.

Die „Allgemeinen Verkaufs- und Lieferbedingungen“ Version 1.1 vom 15.02.2002 bzw. die jeweils neueste Version der Firma Baumüller Nürnberg GmbH gelten grundsätzlich. Diese stehen Ihnen spätestens seit Vertragsabschluss zur Verfügung.

2.6 Schutzeinrichtungen

Während des Transports werden die Steckmodule durch ihre Verpackung geschützt. Entnehmen Sie das Steckmodul erst unmittelbar vor der Montage der Transportverpackung.

Die Abdeckhaube des Reglerteils der b maXX Geräte schützt in Schutzklasse IP20 die Steckmodule vor Verschmutzung und Schäden durch statische Entladungen bei Berührungen. Stecken Sie daher nach erfolgter Montage des Steckmoduls die Abdeckhaube wieder auf.

2.7 Ausbildung des Personals



WARNUNG (WARNING)

Folgendes **kann eintreffen**, wenn Sie diesen Gefahrenhinweis nicht beachten:

- schwere Körperverschletzung
- Tod

Geräte der Firma Baumüller Nürnberg GmbH dürfen ausschließlich von qualifiziertem Personal montiert, installiert, betrieben und gewartet werden.

Qualifiziertes Personal (Fachkräfte) wird folgendermaßen definiert:

Qualifiziertes Personal

Von der Firma Baumüller Nürnberg GmbH autorisierte Elektro-Ingenieure und Elektro-Fachkräfte des Kunden oder Dritter, die Installation und Inbetriebnahme von Baumüller-Antriebssystemen erlernt haben und berechtigt sind, Stromkreise und Geräte gemäß den Standards der Sicherheitstechnik in Betrieb zu nehmen, zu erden und zu kennzeichnen.

Qualifiziertes Personal verfügt über eine Ausbildung oder Unterweisung gemäß den örtlich jeweils gültigen Standards der Sicherheitstechnik in Pflege und Gebrauch angemessener Sicherheitsausrüstung.

Anforderungen an das Bedienungs-personal

Die Bedienung des Antriebssystems darf nur von Personen durchgeführt werden, die dafür ausgebildet, eingewiesen und befugt sind.

Störungsbeseitigung, Instandhaltung, Reinigung, Wartung und Austausch dürfen nur durch geschultes oder eingewiesenes Personal durchgeführt werden. Diese Personen müssen die Betriebsanleitung kennen und danach handeln.

Inbetriebnahme und Einweisung dürfen nur vom qualifizierten Personal durchgeführt werden.

2.8 Sicherheitsmaßnahmen im Normalbetrieb

- Beachten Sie am Aufstellort des Gerätes die gültige Sicherheitsbestimmungen für die Anlage, in die dieses Gerät eingebaut ist.
- Versehen Sie das Gerät mit zusätzlichen Überwachungs- und Schutzeinrichtungen, falls Sicherheitsbestimmungen dies fordern.
- Beachten Sie die Sicherheitsmaßnahmen für das Gerät, in das das Steckmodul eingebaut ist.

2.9 Verpflichtung und Haftung

Damit Sie sicherheitsgerecht mit diesem Steckmodul arbeiten können, müssen Sie die Gefahrenhinweise und Sicherheitshinweise dieser Dokumentation kennen und beachten.

2.9 Verpflichtung und Haftung

2.9.1 Gefahrenhinweise und Sicherheitshinweise beachten

Wir verwenden in dieser Betriebsanleitung optisch einheitliche Sicherheitshinweise, die sie vor Personen- und Sachschäden bewahren sollen.



WARNUNG (WARNING)

Folgendes **kann eintreffen**, wenn Sie diesen Gefahrenhinweis nicht beachten:

- schwere Körperverletzung
- Tod

Alle Personen, die an und mit Geräten der Reihe b maXX arbeiten, müssen bei ihren Arbeiten diese Betriebsanleitung verfügbar haben und die hierin enthaltenen Anweisungen und Hinweise - insbesondere die Sicherheitshinweise - beachten.

Außerdem müssen alle Personen, die an diesem Gerät arbeiten, zusätzlich alle Regeln und Vorschriften, die am Einsatzort gelten, kennen und beachten.

2.9.2 Gefahren im Umgang mit diesem Modul

Das Steckmodul wurde nach dem Stand der Technik und unter Einhaltung der geltenden Richtlinien und Normen entwickelt und gefertigt. Dennoch können bei der Verwendung Gefahren entstehen. Eine Übersicht möglicher Gefahren finden Sie im Kapitel [►Grundlegende Sicherheitshinweise◄](#) ab Seite 7 und in der Betriebsanleitung.

Weiterhin warnen wir Sie vor der akuten Gefahr an der entsprechenden Stelle in dieser Dokumentation.

2.9.3 Gewährleistung und Haftung

Alle Angaben in dieser Dokumentation sind unverbindliche Kundeninformationen, unterliegen einer ständigen Weiterentwicklung und werden laufend durch unseren permanenten Änderungsdienst aktualisiert.

Gewährleistungs- und Haftungsansprüche gegen die Firma Baumüller Nürnberg GmbH sind ausgeschlossen, wenn insbesondere eine oder mehrere der von uns in [►Sachwidrige Verwendung◄](#) ab Seite 12 oder unten aufgeführten Ursachen den Schaden bewirkt hat/haben:

- Eintritt eines Katastrophenfalls durch Fremdkörpereinwirkung bzw. höhere Gewalt

INBETRIEBNAHME

In diesem Kapitel beschreiben wir, wie Sie das montierte und installierte (siehe „Betriebsanleitung b maXX drive PLC“) Steckmodul in Betrieb nehmen. Die Inbetriebnahme stellt sicher, dass das Steckmodul richtig funktioniert.

Stellen Sie vor der Inbetriebnahme sicher, dass die folgenden Voraussetzungen erfüllt sind:

- 1 Steckmodul ist korrekt montiert.
- 2 Steckmodul ist korrekt installiert.
- 3 Alle Sicherheitsvorrichtungen sind in Betrieb gesetzt.
- 4 Das b maXX Gerät ist einsatzbereit.

3.1 Allgemeine Sicherheitsvorschriften

► beachten Sie ► [Grundlegende Sicherheitshinweise](#) ◄ ab Seite 7.



GEFAHR (DANGER)

Folgendes **wird eintreffen**, wenn Sie diesen Warnhinweis nicht beachten:

- schwere Körperverschüttung
- Tod



Die Gefahr ist: **mechanische Einwirkung**. *Bei der Inbetriebnahme kann der Antrieb drehen.*

Halten Sie genügend Abstand von sich drehenden Teilen. Beachten Sie, dass von anlaufenden Antrieben Maschinenteile in Bewegung gesetzt werden können. Aktivieren Sie in jedem Fall deren Sicherheitsvorrichtungen.

3.2 Anforderungen an das ausführende Personal

Die Arbeiten zur Inbetriebnahme dürfen nur von fachlich geschultem Personal, das insbesondere die Sicherheitsvorschriften und -hinweise versteht und befolgen kann, durchgeführt werden.

3.3 Beschreibung/Überprüfung der Sicherheits- und Überwachungseinrichtungen

Bevor Sie die b maXX drive PLC in Betrieb nehmen können, müssen Sie eventuell am Gerät b maXX 4400 anstehende Fehler/Fehlermeldungen beseitigen. Diese Fehler können durch fehlerhafte Montage (z. B. defekte Kabel) oder fehlerhafte Installation (z. B. fehlende Spannungsversorgung) begründet sein. Erst nachdem Sie die Fehler beseitigt haben dürfen Sie mit der Inbetriebnahme fortfahren.

3.4 Beschreibung und Überprüfung der Bedienungs- und Anzeigeelemente

3.4.1 LEDs zur Anzeige von Betriebszuständen der BM4-O-PLC-01

Das Optionsmodul b maXX drive PLC weist als Anzeigeelemente vier LEDs (zwei grüne (H1, H3) und zwei rote (H2, H4)) auf.

Die LEDs werden während der Initialisierung (Hochlaufphase) vom Betriebssystem der b maXX drive PLC verwendet.

Während des Betriebs (nach der Hochlaufphase des Betriebssystems) können die LEDs vom Anwender im Applikationsprogramm auf der b maXX drive PLC verwendet werden.

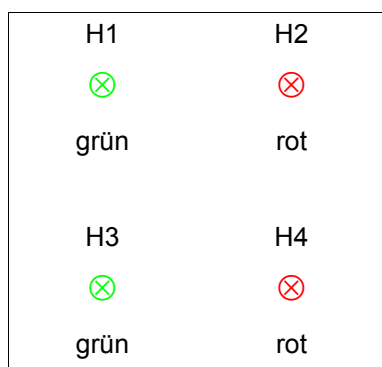


Abbildung 1: LEDs von Steckmodul BM4-O-PLC-01

3.4.1.1 Einschalten und Initialisierung der BM4-O-PLC-01

- Grundsätzlich müssen alle Optionsmodule im Gerät b maXX 4400 nach Zuschaltung der Versorgungsspannung einen gewissen internen Betriebszustand erreicht haben, bevor sie vom b maXX Regler und der b maXX drive PLC angesprochen werden dürfen.

Diese Phase, in der auf die globale Bereitmeldung aller Karten gewartet wird, wird über ein links rotierendes LED-Bitmuster angezeigt. (Also Aufleuchten einer LED in der Reihenfolge H4 -> H2 -> H1 -> H3 -> H4 usw. im Wechsel von 500 ms).

- Nach erfolgter globaler Bereitmeldung aller Optionsmodule muss die b maXX drive PLC warten, bis sie vom b maXX Regler erkannt und von dieser vorinitialisiert worden ist. Diese Phase wird durch ein rechts rotierendes Bitmuster angezeigt. Also LED-Reihenfolge H4 -> H3 -> H1 -> H2 -> H4 usw. im Wechsel von 500 ms.

Die beiden soeben beschriebenen Abläufe können sehr schnell abgeschlossen sein, so dass man die zugehörigen Betriebsanzeigen nicht zwangsläufig beobachten muss.

Danach kann prinzipiell eine PROPROG-Kommunikation über die serielle RS232-Schnittstelle am b maXX Regler zwischen einem PC und der b maXX drive PLC erfolgen.

Ab jetzt ist auch die PROPROG-Kommunikation über TCP/IP möglich, wenn ein Optionsmodul mit Ethernet-Funktionalität vorhanden und für die Kommunikation zur b maXX drive PLC konfiguriert worden ist.

- Falls ein Bootprojekt vorhanden ist, wird jetzt das Bootprojekt geladen (Bootprojekt wird vom Flash gelesen, übersetzt und im SDRAM als ausführbarer Programmcode abgelegt). Das Laden des Bootprojektes wird durch ein schnelles Blinken der oberen beiden LEDs (H2 und H1) angezeigt.

Die LEDs zeigen am Ende der Hochlaufphase folgende PLC-spezifische Betriebszustände an:

- Kein Projekt vorhanden, Zustand „POWER ON“:
→ untere grüne und rote LED (H3 und H4) leuchten.
- Projekt vorhanden, Zustand „STOP“:
→ nur die rechte untere rote LED (H4) leuchtet.
- Projekt vorhanden, Zustand „INIT“, die Steuerung befindet sich in der Kaltstart- bzw. Warmstartphase:
→ nur die linke untere grüne LED (H3) leuchtet.
- Projekt vorhanden, Zustand „RUN“:
→ beide grüne LEDs (unten und oben links, H3 und H1) leuchten.

Im Zustand „RUN“ können die vier LEDs dann vom Anwender frei programmiert werden. Zur Programmierung siehe [►Der Funktionsbaustein LED◄](#) auf Seite 55.

3.4.1.2 Betrieb der BM4-O-PLC-01

Die LEDs zeigen am Ende der Hochlaufphase folgende PLC-spezifische Betriebszustände an:

- Kein Projekt vorhanden, Zustand „POWER ON“:
→ untere grüne und rote LED (H3 und H4) leuchten.
- Projekt vorhanden, Zustand „STOP“:
→ nur die rechte untere rote LED (H4) leuchtet.
- Projekt vorhanden, Zustand „INIT“, die Steuerung befindet sich in der Kaltstart- bzw. Warmstartphase:
→ nur die linke untere grüne LED (H3) leuchtet.
- Projekt vorhanden, Zustand „RUN“:
→ beide grüne LEDs (unten und oben links, H3 und H1) leuchten.

Im Zustand „RUN“ können die vier LEDs dann vom Anwender frei programmiert werden. Zur Programmierung siehe [►Der Funktionsbaustein LED◄](#) auf Seite 55.

3.4.2 Schalter/Taster S1 zum Wechsel von Betriebszuständen der BM4-O-PLC-01

Das Optionsmodul b maXX drive PLC hat zum Ändern der Betriebszustände den Schalter/Taster S1.

3.5 Ablauf der Inbetriebnahme



Taster nach oben: RESET
Schalter Mitte: STOP
Schalter unten: RUN

Abbildung 2: Schalter S1 auf Frontplatte BM4-O-PLC-01



HINWEIS

Das Anwenderprojekt kann nur dann anlaufen, wenn der Schalter/Taster S1 in der unteren Stellung „RUN“ steht.

Mit dem Taster nach oben wird ein Reset **nur** für das Optionsmodul b maXX drive PLC durchgeführt, nicht für den Regler. Nach dem Reset kann die PLC u. U. nicht mehr mit den Optionsmodulen kommunizieren. In diesem Fall ist auch der Neustart des Systems zwingend erforderlich!



GEFAHR (DANGER)

Folgendes **wird eintreffen**, wenn Sie diesen Warnhinweis nicht beachten:

- schwere Körperverletzung
- Tod

Die Gefahr ist: **mechanische Einwirkung**. *Das Bootprojekt kann so programmiert sein, dass der Antrieb bzw. Maschinenteile/Anlagenteile sich drehen oder bewegen.*

Halten Sie genügend Abstand von sich drehenden Teilen. Beachten Sie, dass von anlaufenden Antrieben Maschinenteile in Bewegung gesetzt werden können. Aktivieren Sie in jedem Fall deren Sicherheitsvorrichtungen.

3.5 Ablauf der Inbetriebnahme

Die Inbetriebnahme gliedert sich in folgende Abschnitte:

- 1 Einschalten
- 2 Testen der Funktion

3.5.1 Einschalten

- Lesen und beachten Sie die [►Allgemeine Sicherheitsvorschriften◄](#) ab Seite 15.
- Der Abschnitt „Montage und Installation“ muss korrekt durchgeführt worden sein (siehe hierzu [►Betriebsanleitung b maXX drive PLC◄](#)).
- Stellen Sie den Schalter/Taster S1 an der b maXX drive PLC auf „STOP“ (Mittelstellung).
- Schalten Sie das Grundgerät b maXX 4400 ein.

**HINWEIS**

Das Steckmodul b maXX drive PLC dürfen Sie nicht abziehen oder stecken, wenn das Grundgerät b maXX 4400 eingeschaltet ist. Schalten Sie das Gerät vorher aus.

3.5.2 Testen der Funktion

- Nach dem Einschalten des b maXX Gerätes können zwei Zustände auftreten:
 - Kein Bootprojekt (= kein Anwenderprojekt auf der PLC) vorhanden:
Die LED H2 (oben rechts) leuchtet kurzzeitig auf und nach kurzer Zeit leuchten die unteren LEDs H3 und H4 (rot und grün) andauernd. Dies bedeutet, dass kein Projekt vorhanden ist. In diesem Zustand „POWER ON“ wartet die PLC auf Kommunikation.
 - Bootprojekt vorhanden:
Beim Einschalten wird das Bootprojekt geladen. Dabei blinken die oberen LEDs. Nach kurzer Zeit leuchtet die LED H4 (unten rechts) rot auf. Die PLC befindet sich im Zustand „STOP“.
- Solange der Schalter S1 am Optionsmodul b maXX drive PLC auf "STOP" (Mittelstellung) steht, kann ein vorhandenes Bootprojekt nicht anlaufen.
Wenn Sie ein vorhandenes Bootprojekt starten wollen, indem Sie den Schalter S1 am Optionsmodul b maXX drive PLC auf "RUN" (untere Stellung) stellen, vergewissern Sie sich **vorher**, dass das für Ihre Anwendung **richtige** Bootprojekt für **diese** Anlage in **diesem** Gerät auf der PLC eingespielt ist!

Weitere Informationen, wie Sie dies sicherstellen können oder wie Sie z. B. das „Senden eines Bootprojektes“ zur PLC durchführen können, finden Sie im Kapitel [►Betrieb◄](#) ab Seite 25.

Wie Sie z. B. das "Senden eines Bootprojekts" zur b maXX drive PLC durchführen können, entnehmen Sie [►Bootprojekt senden◄](#) ab Seite 19.

GEFAHR (DANGER)

Folgendes **wird eintreffen**, wenn Sie diesen Warnhinweis nicht beachten:

- schwere Körperverletzung • Tod



Die Gefahr ist: **mechanische Einwirkung**. *Ein auf der PLC vorhandenes Bootprojekt kann anlaufen, wenn der Schalter S1 von „STOP“ (Mittelstellung) auf "RUN" (untere Stellung) gestellt wird oder wenn bei Einschalten des b maXX Gerätes der Schalter S1 auf „RUN“ gestellt ist. Das Bootprojekt kann so programmiert sein, dass der Antrieb bzw. Maschinenteile/Anlagenteile sich drehen oder bewegen.*

Halten Sie genügend Abstand von sich drehenden Teilen. Beachten Sie, dass von anlaufenden Antrieben Maschinenteile in Bewegung gesetzt werden können. Aktivieren Sie in jedem Fall deren Sicherheitsvorrichtungen.

3.5.3 Bootprojekt senden

- 1 Verbinden Sie die serielle Schnittstelle Ihres PC mit der Sub-D-Buchse auf dem **b maXX Regler** (Steckplatz F).



HINWEIS

Nicht mit der Sub-D-Buchse auf der b maXX drive PLC verbinden! Diese hat die Funktion der seriellen Terminal-Schnittstelle RS485 und kann nicht für die Kommunikation mit PROPROG wt II / ProProg wt III verwendet werden.

- 2 Starten Sie auf Ihrem PC die Bediensoftware PROPROG wt II (oder ProProg wt III) und wählen Sie ein Projekt für b maXX drive PLC aus. Dieses Projekt muss bereits erfolgreich kompiliert sein. Sie können das Projekt jetzt auch mit „Code\Projekt neu erzeugen“ erneut kompilieren.
- 3 Klicken Sie im Bedienprogramm auf Ressourcekontrolle im Verzeichnis „Online“.

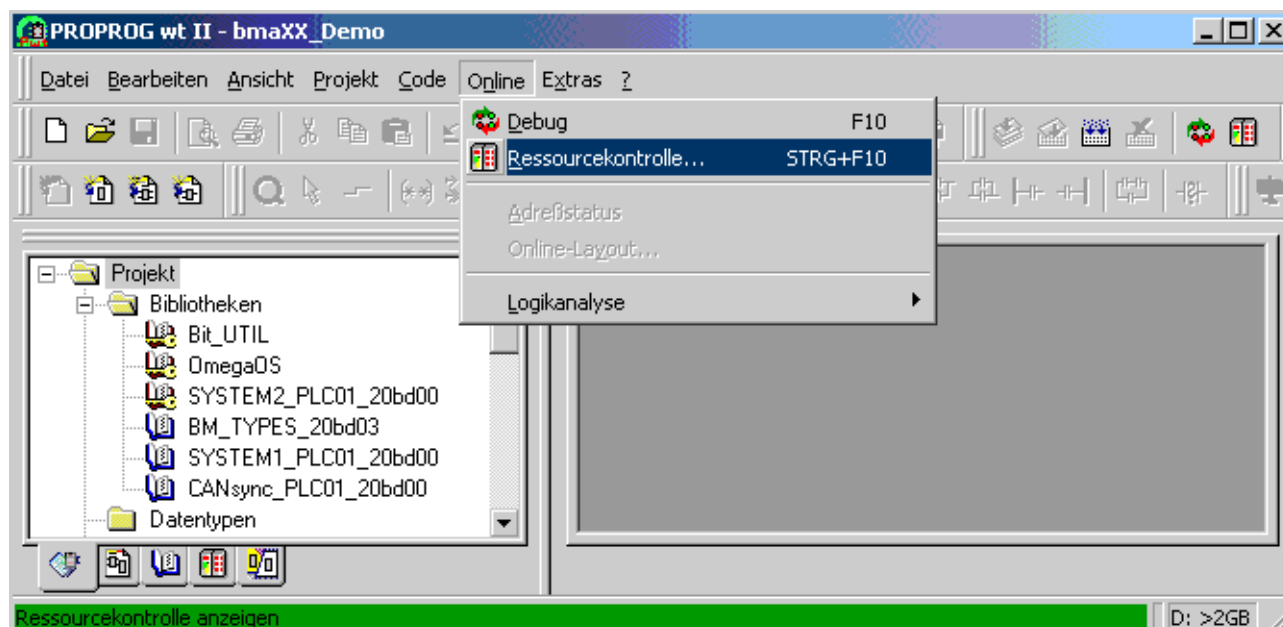


Abbildung 3: PROPROG wt II Bedienoberfläche

Es öffnet sich folgendes Fenster:



Abbildung 4: Ressourcekontrolle „Ein“

4 Drücken Sie den Button „Senden“. Es erscheint folgendes Bild:

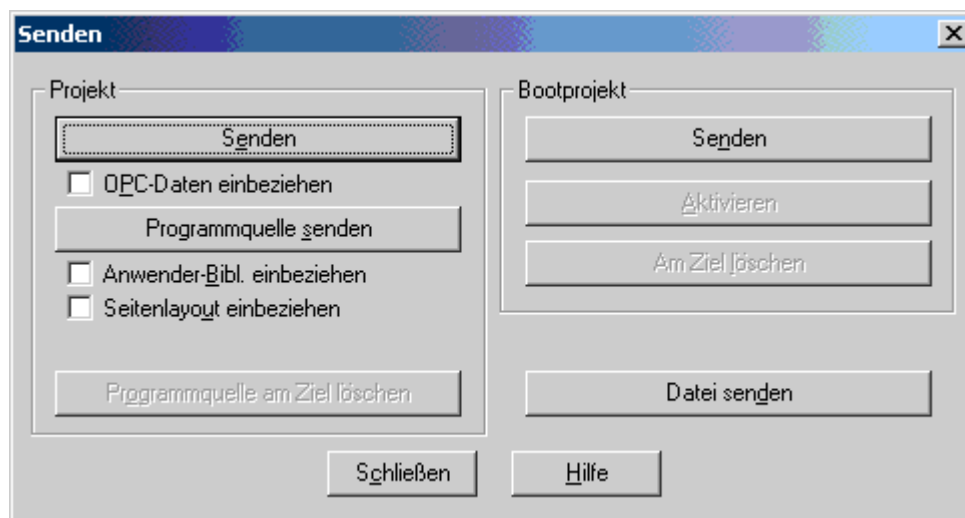


Abbildung 5: Senden

5 Klicken Sie „Bootprojekt Senden“ an.

- Phase 1: Flash löschen:
Ein eventuell vorhandenes Bootprojekt wird gelöscht. Am Bedienprogramm erscheint unten am Bildschirm ein weißer Balken. An der b maXX drive PLC blinken alle 4 LEDs gleichzeitig.
- Phase 2: Projekt senden:
Das Bootprojekt wird ins Flash übertragen. Der Balken unten am Bildschirm wird blau und zeigt den Fortschritt der Übertragung an. An der b maXX drive PLC blinken die LEDs H1 (grün), H2 (rot), H3 (grün), H4 (rot) während der Übertragung diagonal.



HINWEIS

Nachdem der Sendevorgang abgeschlossen wurde, sollte das Grundgerät b maXX 4400 aus- und wieder eingeschaltet werden, damit der b maXX Regler alle Optionsmodule ordnungsgemäß und in richtiger zeitlicher Reihenfolge initialisieren kann.

Alternativ kann es jedoch auch genügen, nur das Projekt auf der b maXX drive PLC entweder über „Senden“ / „Bootprojekt Aktivieren“ und „Kaltstart“ oder über einen Reset an der b maXX drive PLC zu starten. diese Vorgehensweise ist jedoch in hohem Maße applikationsabhängig und liegt somit im Verantwortungsbereich des Inbetriebnehmers.

Die Ausführung eines Projekts auf der b maXX drive PLC kann mit der Ressourcekontrolle gestoppt bzw. neu gestartet werden. Alternativ kann dies auch über den Schalter S1 auf der b maXX drive PLC erreicht werden.



HINWEIS

Der Inbetriebnehmer sollte sich bezüglich dem Gesamtverhalten seines Geräts jedoch immer bewusst sein, dass im Zustand „Stop“ kein Applikationsprojekt ausgeführt wird, also auch keine Kommunikation mehr zum b maXX Regler und zu anderen Optionsmodulen stattfinden kann.



Abbildung 6: Ressourcekontrolle „Betrieb“

Wenn Sie den Button „Info“ drücken erhalten Sie das folgende Bild mit Informationen:



Abbildung 7: Informationsfenster Ressource

**HINWEIS**

Bedeutung der angezeigten Versions-Strings im Info-Fenster:

SPS-String: "OmegaOS V3.1.2078 Oct 15 2002"

-> "OmegaOS": Produktbezeichnung

-> "V3.1.2078": OmegaOS Compiler Info

-> "Oct 15 2002": Datum, an dem die SPS-Version erzeugt wurde.

Firmware-String: "1290.0101/1276.0102/1284.0101"

-> "1290.0101": Version OmegaOS SW 6.1290.0101

-> "1276.0102": Version FPGA SW 6.1276.0102

-> "1284.0101": Version BOOT SW 6.1284.0101"

Im Rahmen der technischen Weiterentwicklung der Produkte können hier auch höhere Versionen mit jüngerem Datum installiert sein.

Quittieren Sie mit „OK“ und kehren Sie zur Ressourcekontrolle zurück (siehe ► [Abbildung 4](#) auf Seite 20).

BETRIEB

4.1 Programmiersysteme PROPROG wt II und ProProg wt III

PROPROG wt ist ein auf der Norm IEC 61131-3 basierendes Standard-Programmiersystem.

Das Programmiersystem verfügt über leistungsstarke Funktionen für die unterschiedlichen Entwicklungsphasen von SPS-Anwendungen wie:

- Editieren
- Kompilieren
- Debuggen
- Drucken

Das Programmiersystem PROPROG wt basiert auf einer modernen 32-Bit-Windows-Technologie und ermöglicht dem Benutzer eine einfache Bedienung durch Werkzeuge wie Zoom, Scrollen, spezielle Symbolleisten, Drag & Drop, einen Shortcut-Manager und andockbare Fenster.

Insbesondere ermöglicht das System die Bearbeitung mehrerer Konfigurationen und Ressourcen innerhalb eines Projektes sowie das Einbinden von Projektbibliotheken. Des weiteren verfügt es über ein leistungsstarkes System zum Debuggen. Mit Hilfe des sehr einfach zu bedienenden Projektbaum-Editors können Projekte angezeigt und editiert werden. Die komplexe Struktur der Norm IEC 61131-3 wird so einfach und transparent wie möglich dargestellt. Dies ermöglicht dem Benutzer auf einfache Weise POEs, Datentypen, Bibliotheken und Konfigurationselemente in den Projektbaum einzufügen und zu editieren.

Das Programmiersystem PROPROG wt besteht aus einem SPS-unabhängigen Kern zum Programmieren in den verschiedenen IEC Programmiersprachen: dazu gehören die Textsprachen Strukturierter Text (ST) und Anweisungsliste (AWL) sowie die graphischen Sprachen Funktionsbaustein-Sprache (FBS), Kontaktplan (KOP) und Ablaufsprache (AS).

Zum Programmieren in den einzelnen Sprachen steht jeweils ein Editor-Assistent zur Verfügung, der ein schnelles und sehr einfaches Einfügen von vorbereiteten Schlüsselwörtern, Anweisungen, Operatoren, Funktionen und Funktionsbausteinen in den einzelnen Arbeitsblättern ermöglicht. Der Editor-Assistent kann auch zum Deklarieren von Variablen und Datentypen verwendet werden. Der unabhängige Kern des Systems wird durch spezielle Teile vervollständigt, die den verschiedenen SPSen angepasst sind.

Das neue, einfache Online-Handling und die 32-Bit-Simulation bieten Möglichkeiten zum Debuggen des Adressstatus und eine Multitask-Testumgebung in Echtzeit.

Ein einfach zu bedienendes Werkzeug für die Projektdokumentation ermöglicht das Drucken des Projektes in zeitsparender optimierter Form (mit weniger Papier) mit einem Seitenlayout, das vom Benutzer festgelegt werden kann.



GEFAHR (DANGER)

Folgendes **wird eintreffen**, wenn Sie diesen Warnhinweis nicht beachten:

- schwere Körperverletzung
- Tod

Die Gefahr ist: **mechanische Einwirkung**. *Ein auf der PLC vorhandenes Bootprojekt kann anlaufen, wenn der Schalter S1 von „STOP“ (Mittelstellung) auf „RUN“ (untere Stellung) gestellt wird oder wenn bei Einschalten des b maXX Gerätes der Schalter S1 auf „RUN“ gestellt ist. Das Bootprojekt kann so programmiert sein, dass der Antrieb bzw. Maschinenteile/Anlagenteile sich drehen oder bewegen.*

Halten Sie genügend Abstand von sich drehenden Teilen. Beachten Sie, dass von anlaufenden Antrieben Maschinenteile in Bewegung gesetzt werden können. Aktivieren Sie in jedem Fall deren Sicherheitsvorrichtungen.



GEFAHR (DANGER)

Folgendes **wird eintreffen**, wenn Sie diesen Warnhinweis nicht beachten:

- schwere Körperverletzung
- Tod

Die Gefahr ist: **mechanische Einwirkung**. *Online-Änderungen werden automatisch an die SPS gesandt und sofort wirksam ohne vorher die Programmausführung auf der SPS zu stoppen.*

Halten Sie genügend Abstand von sich drehenden Teilen. Beachten Sie, dass durch die Änderungen Maschinenteile in Bewegung gesetzt werden können. Aktivieren Sie in jedem Fall deren Sicherheitsvorrichtungen.



GEFAHR (DANGER)

Folgendes **wird eintreffen**, wenn Sie diesen Warnhinweis nicht beachten:

- schwere Körperverletzung
- Tod

Die Gefahr ist: **mechanische Einwirkung**. *Wenn Sie Variablen forcen oder überschreiben, während die SPS läuft, wird das SPS-Programm mit den geforcten oder überschriebenen Variablen ausgeführt.*

Halten Sie genügend Abstand von sich drehenden Teilen. Beachten Sie, dass durch die Änderungen Maschinenteile in Bewegung gesetzt werden können. Aktivieren Sie in jedem Fall deren Sicherheitsvorrichtungen.

**GEFAHR (DANGER)**

Folgendes **wird eintreffen**, wenn Sie diesen Warnhinweis nicht beachten:

- schwere Körperverletzung
- Tod

Die Gefahr ist: **mechanische Einwirkung**. Wenn Sie Breakpoints verwenden, während die SPS läuft, wird das SPS-Programm beim Auflaufen auf einen Breakpoint gestoppt.

Halten Sie genügend Abstand von sich drehenden Teilen. Beachten Sie, dass durch die Änderungen Maschinenteile in Bewegung gesetzt werden können. Aktivieren Sie in jedem Fall deren Sicherheitsvorrichtungen.

4.2 b maXX drive PLC Projekt

Ein neues b maXX drive PLC Projekt wird unter PROPROG wt über das Menü "Datei\Neues Projekt" begonnen. Mit der „Vorlage für BM4_O_PLC01“ wird ein b maXX drive PLC Projekt geöffnet. Der Prozessortyp BM4_O_PLC01 in diesem Projekt ist in der Konfiguration voreingestellt.

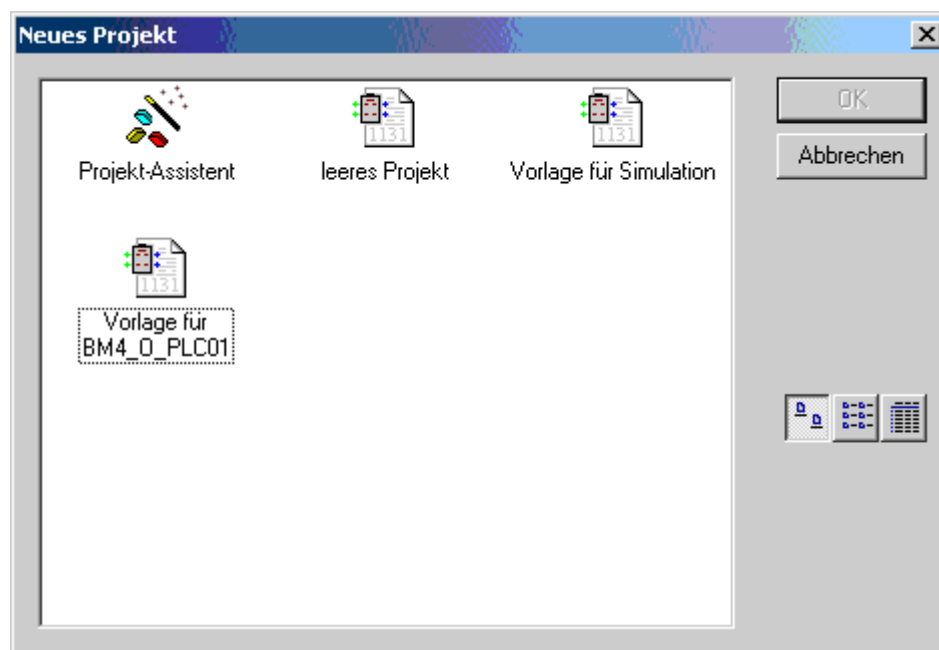


Abbildung 8: BM4_O_PLC01 Projektvorlage unter "Neues Projekt".

4.3 b maXX drive PLC Konfiguration

Als IEC 61131-3 Programmieroberfläche können mit PROPROG wt verschiedene Zielsysteme (CPUs) programmiert werden. Es können auch verschiedene Zielsysteme in einem Projekt programmiert werden. Die Erstellung eines Programms für das Zielsystem BM4_O_PLC01 erfolgt unter "Physical Hardware" mit der Ressource BM4_O_PLC01.

Über das Menü „Neues Projekt“ wird die Vorlage für BM4_O_PLC01 geöffnet. Die Konfiguration CONF1 hat unter Eigenschaften den SPS-Typ SH03_30.

4.4 b maXX drive PLC Ressource

Eine Konfiguration besteht aus mindestens einer Ressource. Die Ressource beinhaltet den spezifischen Datenbereich für die b maXX drive PLC, die Kommunikationsquelle, globale Variablenarbeitsblätter und die Tasks mit den Programmteilen.

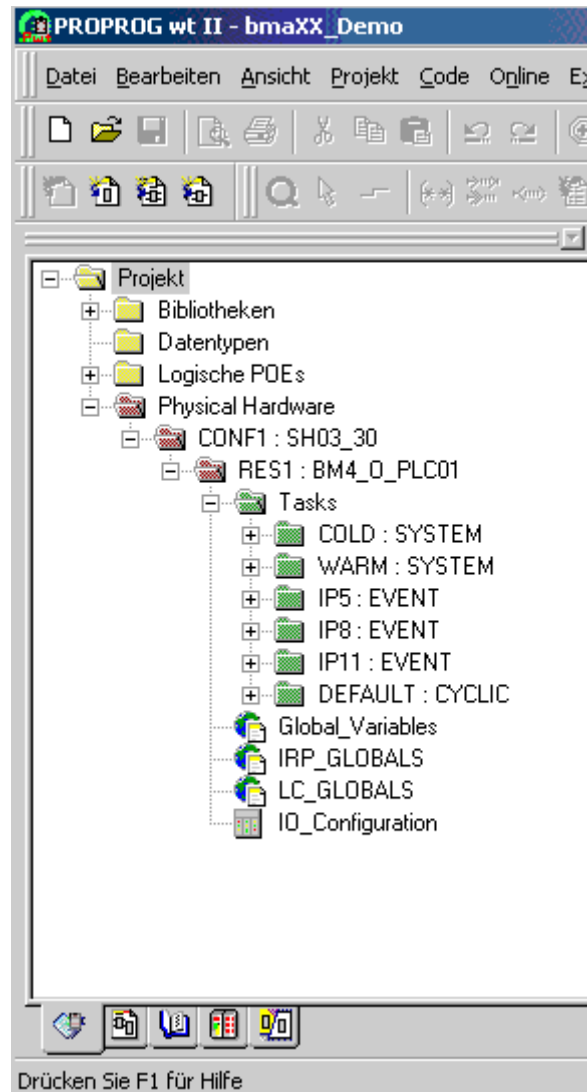


Abbildung 9: Einstellung unter "Physical Hardware" Eigenschaft einer b maXX drive PLC Konfiguration.

Übersicht der Einstellungen innerhalb der Physikalischen Hardware:

Projektvorlage	Konfiguration	Ressource
BM4_O_PLCO1	SH03_30	BM4_O_PLCO1

Ein Projekt kann mehrere Konfigurationen mit jeweils mehreren Ressourcen enthalten.

4.4 b maXX drive PLC Ressource

Die Ressource beinhaltet die b maXX drive PLC spezifischen Einstellungen für ein Programm:

- Datenbereich

- Kommunikationsquelle
- Globale Variablenarbeitsblätter
- Unterschiedliche Tasks mit dem Programm
- Dokumentationsarbeitsblätter und I/O-Konfiguration
(Die I/O-Konfiguration ist bereits fertig eingestellt und braucht nicht verändert werden.)

Einer Konfiguration können mehrere Ressourcen mit unterschiedlichen Ports zugeordnet werden. Somit ist eine komplette Applikation mit mehreren Antrieben in einem Projekt umsetzbar.

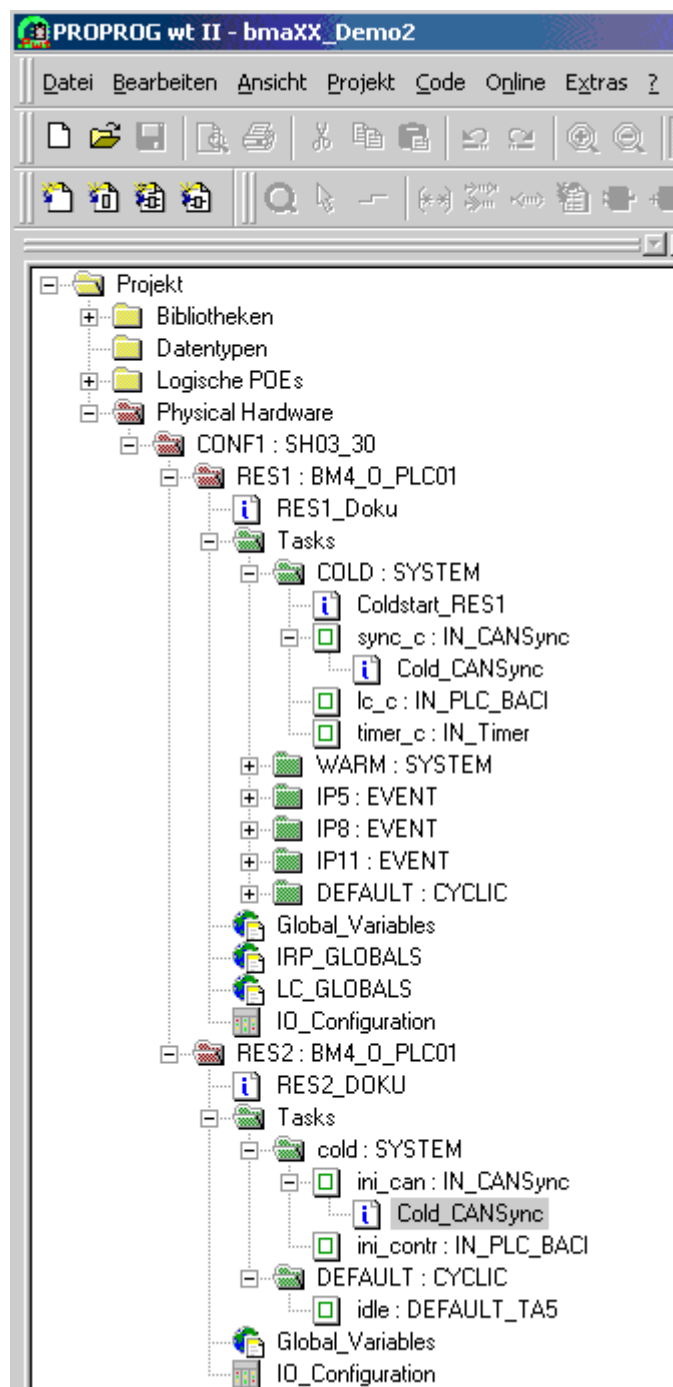


Abbildung 10: Beispiel einer b maXX drive PLC Konfiguration mit mehreren Ressourcen.

4.4.1 Kommunikation und Verbindung

Die Kommunikation zur Daten-Übertragung wird unter "Einstellung" im Kontextmenü der Ressource konfiguriert.

Die Kommunikation über RS232 wird, für den gewählten Port, wie folgt eingestellt:

- Baud: 38400
- Stopbits: 1
- Datenbits: 8
- Parität: Keine
- Timeout: Default 2000 ms; Kommunikationsüberwachung während der Online-Darstellung.

Die Verbindung wird über den seriellen Port COM1 am PC aufgebaut:

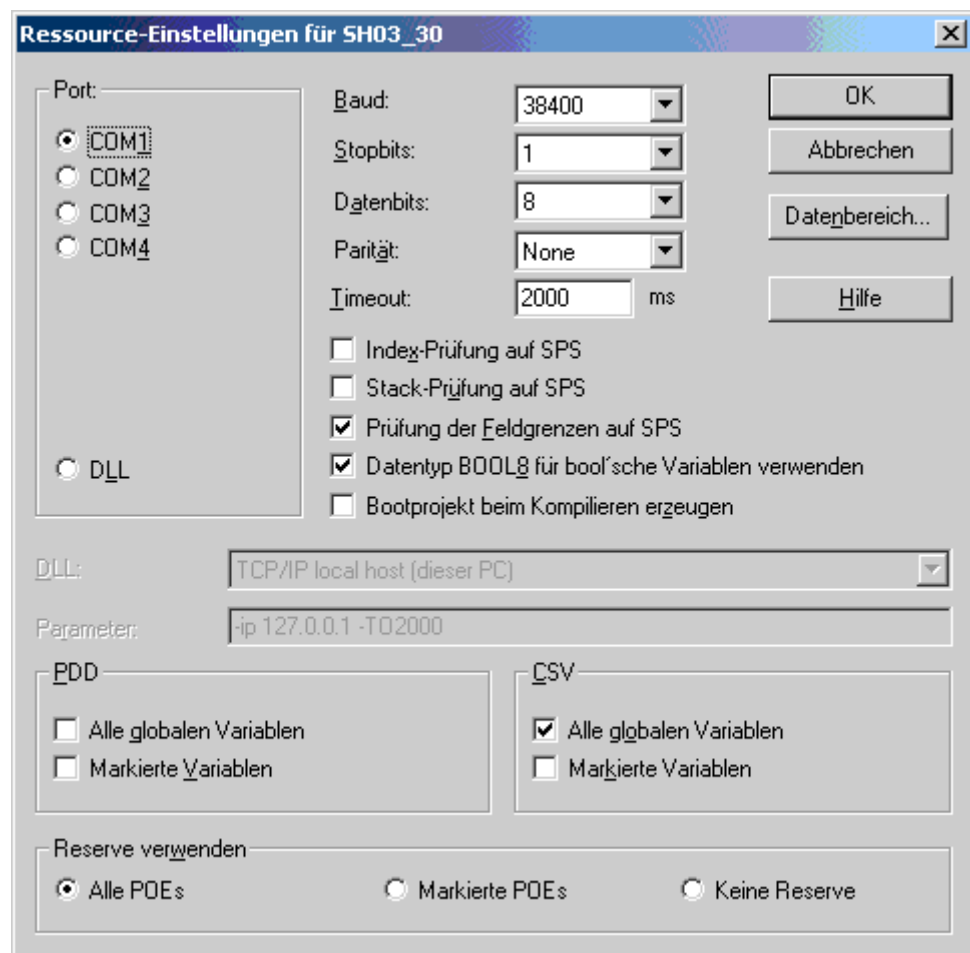


Abbildung 11: Ressource-Einstellung innerhalb einer b maXX drive PLC Konfiguration.

- "Index-Prüfung auf SPS": Deklarierte Feldgröße (Index) eines ARRAY wird zur Laufzeit kontrolliert. Achtung: Code-Ausführungszeit wird erhöht!
- "Stack-Prüfung auf SPS": Überprüfung auf Stacküberlauf zur Laufzeit. Mit der Programm-Task wird Stack-Speicher reserviert. Die Daten auf dem Stack erhöhen sich, wenn z. B. FB-Instanzen geschachtelt programmiert werden. Achtung: Code-Ausführungszeit wird erhöht!

- "Prüfung der Feldgrenzen auf SPS": Bei absoluter Adressierung wird geprüft, ob das Feld die parametrisierten Datenbereichsgrenzen überschreitet. Die Überprüfung erfolgt nur während des Übersetzungsvorganges auf dem PC. Die Code-Ausführungszeit wird nicht erhöht.
- "Datentyp BOOL8 für boolsche Variablen verwenden": 8-Bit Zugriff auf boolsche Variablen aktivieren. Für die b maXX drive PLC muss die Einstellung aktiviert sein.
- "Bootprojekt beim kompilieren erzeugen": Bei aktivierter Funktion wird bereits beim Kompilieren des Projektes für jede Ressource das "bootfile.pro" erzeugt, und nicht erst bei Aktivieren der "Bootprojekt senden"-Funktion über die Ressourcenkontrolle.
- "PDD": Einstellungen zum Prozessdatenverzeichnis. (Siehe auch Programmierhandbuch PROPROG wt II bzw. Online-Hilfesystem ProProg wt III.)
- "CSV": Einstellungen zur Bereitstellung von Variablen für den OPC-Server. (Siehe auch Programmierhandbuch PROPROG wt II bzw. Online-Hilfesystem ProProg wt III.)
- "Reserve verwenden": Speicherreserve um Änderungen in Funktionsbausteine und Funktionen mit der Funktion "Online-Änderung" durchführen zu können. (Siehe auch Programmierhandbuch PROPROG wt II bzw. Online-Hilfesystem ProProg wt III.)

Die Ressource-Einstellung kann für jede b maXX drive PLC - Ressource separat vorgenommen werden. Die serielle oder Ethernet-Kommunikationsquelle

- COM1
- COM2
- COM3
- COM4
- DLL

wird genutzt

- zur Ressourcenkontrolle.
- zur Online-Darstellung von Variablen und Strukturen im Watch-Fenster
- zum Senden des kompilierten Projekts.
- zum Debuggen.
- zur Verbindung zum OPC-Server.

Einstellung der Ethernet-Kommunikationsquelle durch Auswahl bzw. Angabe der TCP/IP-Adresse

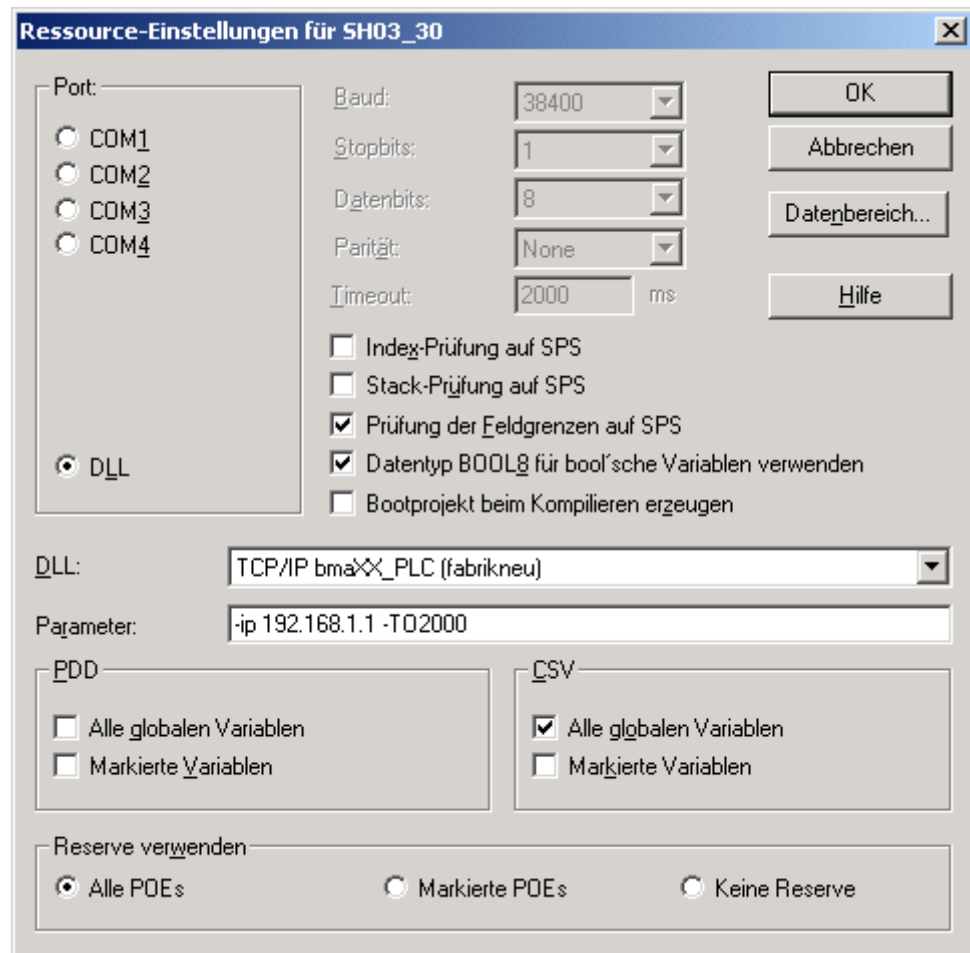


Abbildung 12: Einstellung der TCP/IP Adresse

Nach Umstellen des Ports auf "DLL" hat der Anwender über das DLL-Menü die Möglichkeit, die Anwendungsfälle "Soft-PLC" (=TCP/IP local Host (dieser PC)) und "b maXX drive PLC - Zugriff über Ethernet" (=TCP/IP bmaXX_PLC (fabrikneu)) auszuwählen:

- **Soft-PLC:**
Die voreingestellte TCP/IP-Adresse im Feld "Parameter" muss nicht geändert zu werden, wenn die Soft-PLC auf dem gleichen Rechner installiert ist. Bei Installation der Soft-PLC auf einem anderen Rechner muss der Anwender dessen entsprechende TCP/IP-Zieladresse (bzw. der entsprechende Netzwerk-Name) hier eintragen, um von PROPROG wt aus über TCP/IP auf die Soft-PLC zugreifen zu können.
- **b maXX drive PLC:**
Um auf die b maXX drive PLC über Ethernet zugreifen zu können, muss im Grundgerät b maXX 4400 zusätzlich zur b maXX drive PLC ein Ethernet-fähiges Optionsmodul (z. B. BM4-O-ETH-02) vorhanden sein. Die im Parameterfeld voreingestellte TCP/IP-Adresse "192.168.1.1" entspricht der IP-Adresse, die auf der Ethernet-Karte bei Auslieferung vorinstalliert ist, so dass der Anwender eine erste Verbindung zum Modul für seine Grund-Initialisierung aufnehmen kann, in der er dann die TCP/IP-Adresse für sein TCP/IP-Netz an der Maschine vornehmen muss.

Die genaue Vorgehensweise für die Grund-Initialisierung ist den jeweiligen Betriebsanleitungen und Applikations-Handbüchern der eingesetzten Ethernet-Optionsmodule zu entnehmen.

4.4.2 Kontrolldialog für Ressourcen

Mit dem Kontrolldialog für Ressourcen wird die Programmübermittlung zur b maXX drive PLC und der Betriebszustand der b maXX drive PLC eingestellt.

Sind mehrere Ressourcen in einem Projekt aktiv, muss die gewünschte Ressource ausgewählt werden.

Auswählen einer Ressource in PROPROG wt II:

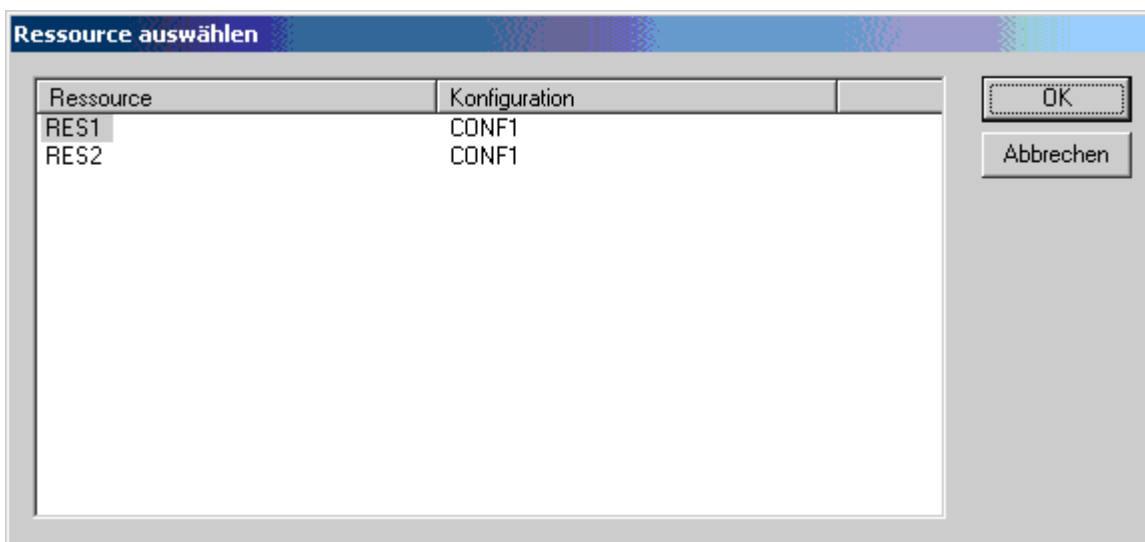


Abbildung 13: Nach Auswahl der Ressource erscheint der Kontrolldialog für die Ressource des zugeordneten b maXX drive PLC.

Der Kontrolldialog wird geöffnet.

Auswählen einer Ressource in ProProg wt III:

Klicken Sie auf die gewünschte Ressource und drücken Sie anschließend auf den Button „Verbinden“.

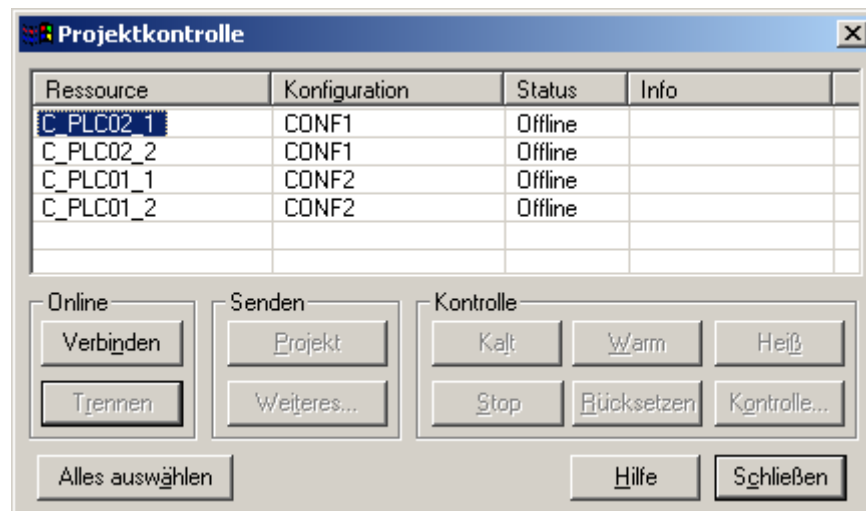


Abbildung 14: Auswahl der Ressource in ProProg wt III - Verbinden

Drücken Sie jetzt auf den Button „Kontrolle...“

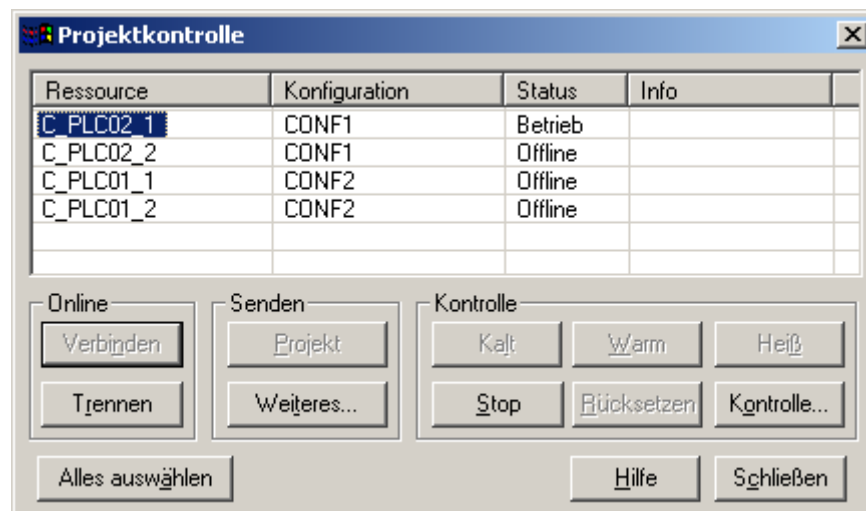


Abbildung 15: Auswahl der Ressource in ProProg wt III - Kontrolldialog öffnen

Der Kontrolldialog wird geöffnet.



Abbildung 16: Funktionen des Kontrolldialogs der ausgewählten Ressource im Zustand "RUN".

Mit **"Senden"** kann das kompilierte Projekt an das Zielsystem übermittelt werden.

Senden des Projektes zum Zielsystem.

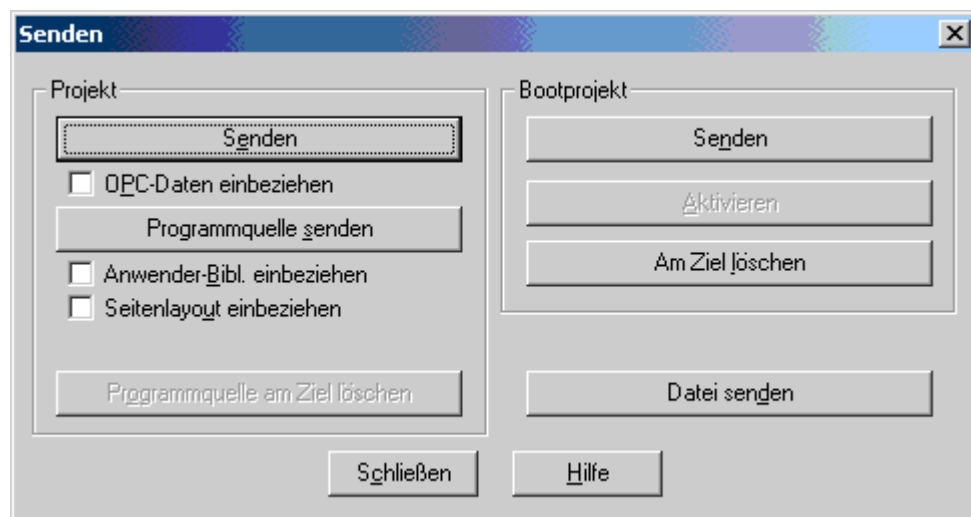


Abbildung 17: b maXX drive PLC Ressource Übertragung zum Flash oder RAM-Speicher.

Mit **"Bootprojekt / Senden"** wird das aktuelle Bootprojekt der Ressource gelöscht, das kompilierte Projekt als Bootprojekt gesendet und im b maXX drive PLC Flash-Speicher abgelegt. Mit **"Aktivieren"** wird das Projekt vom b maXX drive PLC Flash-Speicher in den RAM-Speicher geladen.

Mit **"Am Ziel löschen"** wird das Bootprojekt im Flash gelöscht.

Mit **"Projekt / Senden"** wird das kompilierte Projekt der Ressource direkt ins RAM gesendet und kann über den Kontrolldialog (Button „Kalt“) anschließend gestartet werden. Das Bootprojekt bleibt unverändert im b maXX drive PLC Flash-Speicher. Nach dem nächsten Hardware-Reset oder Steuerung Aus → Ein wird wieder das Bootprojekt geladen!



HINWEIS

Die Menüpunkte "Programmquelle senden", "Datei senden" und das Attribut "OPC-Daten einbeziehen" werden zur Zeit nicht unterstützt und dürfen nicht angewählt werden.

Will man das über „**Bootprojekt / Senden**“ ins Flash abgelegte Projekt aktivieren, geht man folgendermaßen vor:

- direkt Spannung „Aus/Ein“

oder

- über PROPROG wt:

Im Kontrolldialog das geladene Programm (RAM) über „**Stop**“ anhalten und über „**Reset**“ löschen. Der Betriebsstatus wechselt auf den Zustand „**Ein**“.



Abbildung 18: Ressourcenkontrolle Status „Ein“

Klicken Sie im „Senden“-Menü des Kontrolldialogs dann „Bootprojekt / Aktivieren“ an. Dadurch wird das Flash-Projekt auf der b maXX drive PLC im RAM installiert.

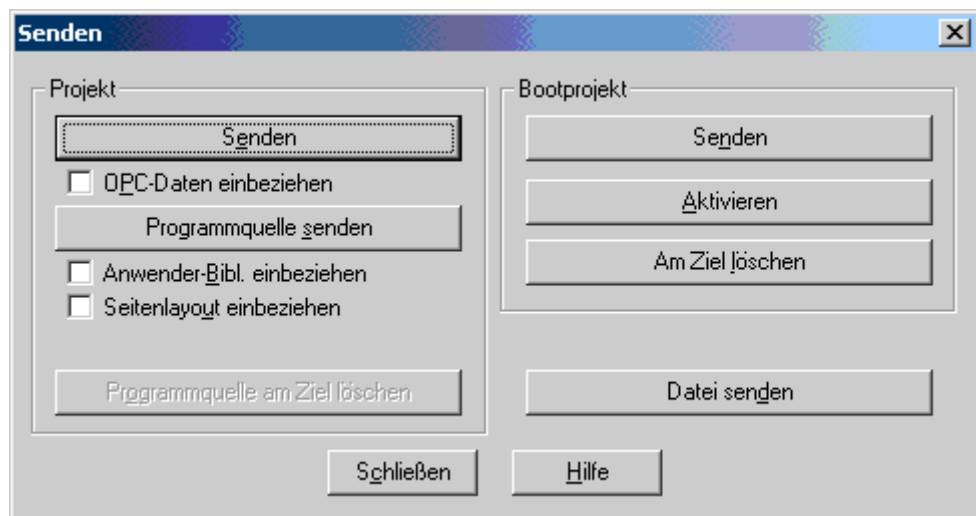


Abbildung 19: Kontrolldialog „Senden“

Dann können Sie das Programm über „Kalt“ im Kontrolldialog starten:



Abbildung 20: Die b maXX drive PLC Ressourcenkontrolle im Zustand "STOP".

Beschreibung der Buttons in der b maXX drive PLC Ressourcenkontrolle

- Button „**Stop**“: Programmausführung wird angehalten, der PLC-Betriebsstatus wechselt von „Run“ nach „Stop“ (Der Status wird oben in der Ressourcenkontrolle angezeigt).
- Button „**Reset**“: Löscht das RAM-Projekt auf der b maXX drive PLC (nicht das im Flash abgespeicherte Bootprojekt!). Der Status wechselt von „Stop“ nach „Ein“.
- Button „**Senden**“: Ruft die Seite für die Programmübermittlung (b maXX drive PLC Ressource Übertragung zum RAM oder Flash) auf.
- Button „**Fehler**“: Hier können in der b maXX drive PLC anstehende Fehler- und Warnmeldungen ausgelesen werden, wenn der Button aktiviert ist. Durch Drücken des aktiven Fehler-Buttons werden die Fehlereinträge von der Steuerung abgefragt und im Fehler- bzw. Warnungsmeldefenster angezeigt.
- Button „**Schließen**“: Die b maXX Ressourcenkontrolle wird wieder geschlossen.
- Buttons „**Kalt/Warm/Heiß**“: Über diese PROPROG-Befehle kann die b maXX drive PLC manuell gestartet werden. Der Status wechselt von „Stop“ nach „Run“. Ein Start der PLC über die Buttons kann jedoch nur erfolgen, wenn der Schalter S1 auf dem Optionsmodul BM4-O-PLC-01 auf „RUN“ eingestellt ist. Sonst bleibt die Steuerung im Status „Stop“.

Die drei Startbuttons unterscheiden sich dabei wie folgt:

- Button „**Kalt**“: Die PLC durchläuft einmalig die PROPROG-System-Task „Kaltstart (SPG1)“, in der die Anwender-Programminitialisierung stattfindet, und wechselt dann in die zyklische Programmbearbeitung.
Der Kaltstart ist dadurch gekennzeichnet, dass alle Variablen mit ihren Default-Werten initialisiert werden. Wurde im Projekt kein Default-Wert vom Anwender angegeben, wird ein Defaultwert „0“ (bzw. „FALSE“ für boolsche Variablen) gesetzt.
- Button „**Warm**“: Die PLC durchläuft einmalig die PROPROG-System-Task „Warmstart (SPG0)“, in der die Anwender-Programminitialisierung stattfindet, und wechselt dann in die zyklische Programmbearbeitung.
Im Unterschied zum Kaltstart behalten die remanenten Merker ihre Werte, sie werden also nicht auf Default-Werte zurückgesetzt.
Falls keine remanenten Merker auf der b maXX drive PLC vorhanden sind, und wenn der gleiche Anwender-Code sowohl im Kaltstart- als auch in der Warmstart-Task eingebunden ist, gibt es keine Unterschiede im Verhalten zwischen Kalt- und Warmstart.

- Button „**Heiß**“: Es wird keine Initialisierungstask durchlaufen, sondern direkt in die zyklische Programmbearbeitung gewechselt.



HINWEIS

Ein Start der PLC über die Buttons (Kalt/Warm/Heiß-Start) bzw. nach Spannung aus/ein kann nur dann erfolgen, wenn der Schalter S1 auf dem Optionsmodul BM4-O-PLC-01 auf „RUN“ eingestellt ist. Sonst bleibt die Steuerung im Status „Stop“, und es wird kein Anwendercode ausgeführt.

Wird ein HW-Reset am Schalter S1 ausgeführt, muss der Schalter S1 anschließend ebenfalls auf "RUN" gestellt werden, damit der Anwendercode abgearbeitet werden kann.

Bei einer b maXX drive PLC mit remanenten Merken wird nach Senden eines neuen Boot-Projektes und anschließendem automatischen Programmstart („Spannung aus/ein“ bzw. HW-Reset am Schalter S1 des Optionsmoduls BM4-O-PLC-01) einmalig ein Kaltstart durchgeführt, in dem auch die remanenten Merker auf die vorgegebenen Default-Werte gesetzt werden. Die Information „Kaltstart wurde durchlaufen“ wird in den remanenten Bereich abgespeichert.

Bei jeden weiteren automatischen Wechsel in den Zustand „Run“ wird diese abgespeicherte Info ausgewertet und immer ein Warmstart ausgeführt, bei dem die remanenten Merker ihre Inhalte beibehalten.

Aktiviert der Anwender den manuellen Start-Button "Kalt" in der b maXX drive PLC Ressourcenkontrolle, wird das explizite Setzen der Default-Werte auch für die remanenten Merker erzwungen.

Bei Abspeichern von Bootprojekten ist das automatische Wiederanlauf-Verhalten davon abhängig, ob sich im remanenten Merkerbereich etwas geändert hat: Wird auf der Steuerung erkannt, dass im Vergleich zum Projekt-Download vorher neue remanente Merker angelegt bzw. gelöscht, oder der Name, der Datentyp oder die Reihenfolge von remanentern Merken geändert wurden, wird daraufhin bei Wechsel in "Run" ebenfalls einmalig ein Kaltstart ausgeführt, um die Konsistenz der remanenten Merker gewährleisten zu können.

Das Ziel bei Projekten ist, remanente Daten auf der Steuerung auch nach einem Projekt-Download wenn möglich zu erhalten. Bei Änderung von Default-Werten alleine wird deswegen kein automatischer Kaltstart ausgeführt und muss gegebenenfalls vom Anwender selbst explizit durchgeführt werden.

Bei einer b maXX drive PLC ohne remanente Merker wird bei automatischen Programmstarts immer ein Kaltstart ausgeführt.

Ob eine PLC remanente Merker unterstützt, kann über den Button „Info“ (siehe unten) abgefragt werden.

- Button "**Hochladen**": Upload-Funktionen in PROPROG wt werden zur Zeit nicht unterstützt.
- Button "**Info**": Allgemeine Informationen zur b maXX drive PLC selbst und zum Projekt-Status auf der b maXX drive PLC (Beschreibung siehe unten).
- Button "**Hilfe**": Die allgemeine und spezifische SPS-Hilfe für die b maXX drive PLC werden aufgerufen.

Beschreibung der "Info"-Seite "Ressource-Informationen".

Diese Seite wird über den „Info“ Button in der b maXX drive PLC Ressourcenkontrolle aufgerufen:

Abbildung 21: Informationsfenster Ressource

- Version: SPS "OmegaOS V3.1.2078 Oct 15 2002"
 - „OmegaOS“: Produktbezeichnung.
 - „V3.1.2078“: OmegaOS Compiler Info.
 - „Oct 15 2002“: Datum, an der die PLC-Version erzeugt wurde.
- Firmware: "1290.0101/1276.0102/1284.0101"
 - „1290.0101“: Version OmegaOS SW 6.1290.0101.
 - „1276.0102“: Version FPGA SW 6.1276.0102.
 - „1284.0101“: Version BOOT SW 6.1284.0101.
- Projekt: Name des aktiven Projekts (Die Darstellung des Projektnamens ist auf 11 Buchstaben begrenzt).
- CPU-Auslastung: Die Auslastung wird vom System automatisch zur Echtzeit ermittelt und wird hier in Prozent angezeigt.
 Es fließen u. A. die Laufzeiten der einzelnen zyklischen Tasks sowie der nicht über Watchdogzeiten überwachten Bypass-Tasks ein. Um wesentliche Betriebssystemfunktionen wie beispielsweise die Online-Kommunikation sicherzustellen, sollte eine Gesamt-Auslastung kleiner 80 % angestrebt werden. Bei 100 % Auslastung wird ein System-Fehler generiert, der zum Stopp der b maXX drive PLC führt.
 Eine Auslastung von 100 % wird z. B. erreicht, wenn die Ausführung der Default-Task nach ca. 10 Sekunden noch nicht abgeschlossen werden konnte und die eingestellte

Watchdogzeit für die Default-Task auf größer 10 Sekunden vom Anwender geändert wurde. (Bei Watchdog-Einstellungen kleiner 10 Sekunden in der Default-Task würde vorher die Watchdog-Überwachung einen System-Fehler auslösen und zurückmelden). Die augenblickliche Default-Taskdauer kann im Fenster rechts neben der CPU-Auslastung abgelesen werden.

Eine Überlastung kann vermieden werden, wenn man die entsprechenden Tasks in mehrere kürzere Einzel-Tasks mit größeren Aufrufintervallen teilt oder einzelne Programmteile (z. B. in den Bypass-Tasks) jeweils überspringt, um die Task-Laufzeiten zu verringern.

- **Default-Task:** Die augenblickliche Zykluszeit der Default-Task kann hier direkt in ms abgelesen werden.
- **Systemtick:** Diese Zeit gibt die kleinste Auflösung von Zyklus- bzw. Überwachungszeiten an, die das PLC-Laufzeitsystem verwalten kann.

Beispiel: Ob man das Intervall in einer Cyclic-Task mit „400 ms“ oder „409 ms“ angibt, ist „egal“: Bei einem Systemtick vom 10 ms wird immer „400 ms“ (also auf die Dauer eines Systemticks abgerundet) aktiviert.

- **Remanente Daten:** Wie bereits im Hinweis oben beschrieben, gibt es b maXX drive PLC-Ausführungen mit und ohne remanente Merker.

Bei einer b maXX drive PLC ohne remanente Merker wird hier der Wert 0 angezeigt. In diesem Fall wird bei Aktivierung des „Projekt/Senden“-Buttons im Fenster „PLC Ressource Übertragung“ vom System ein entsprechender SPS-Fehler rückgemeldet, wenn fälschlicherweise remanente Merker im Anwender-Projekt benutzt werden oder im Anwender-Datenbereich (siehe unten) eingestellt wurden.

Bei Systemen mit remanenten Merkern wird bei der Anzeige „Remanente Daten“ der verfügbare remanente Gesamt-Speicherbereich auf der PLC angezeigt. Da darin außer den remanenten Merkern noch zusätzliche remanente System-Informationen abgelegt werden (z. B. ob ein Kaltstart bei einem Projekt bereits durchlaufen wurde), ist der vom Anwender als remanente Merker nutzbare Bereich kleiner und kann den Datenbereichs-Einstellungen („Remanente Merker/Anfangsadresse Anwender“ und „Ende System“, siehe unten) entnommen werden.

- **Programmspeicher:** Anzeige des noch freien PLC-Programmspeichers des aktiven Projektes in KB.

Mit einem freien Programmspeicher von 2046 KB z. B. kann man:

- max. 400000 AWL-Zeilen (LD/ST-Anweisungen auf globale Variablen) bzw.
- typ. 120000 AWL-Zeilen (Typische AWL-Anweisungen auf Strukturen und Instanz-Variablen) programmieren.

- **Systemdaten:** Dynamischer Speicher (z. B. 1460 KByte) für Debug- und Logic-Analyzer-Funktionen.
- **Breakpoints:** Hier wird angezeigt, ob auf der PLC Breakpoints gesetzt sind. Wenn dies der Fall ist, wird das Kontrollkästchen 'Breakpoints rücksetzen' angezeigt. Aktivieren Sie dieses Kontrollkästchen, wenn Sie alle gesetzten Breakpoints gleichzeitig rücksetzen möchten. (Breakpoints können über das Menü "Online/Debug" verwaltet werden).
- **Forcen:** Zeigt an, ob Variablen geforct wurden. Wenn dies der Fall ist, wird das Kontrollkästchen 'Forceliste rücksetzen' angezeigt. Aktivieren Sie dieses Kontrollkästchen, wenn Sie alle geforcten Variablen gleichzeitig rücksetzen möchten.
- **Freier Speicher Systemdaten Laufzeitsystem** (für Auslastung Oszilloskopfunktion).
- **Zeit zwischen Übertragungen:** Hier kann der Anwender eine Zeit angeben, nach der das Programmiersystems frühestens eine neue Kommunikation an die PLC startet,

nachdem die Letzte abgeschlossen ist. Durch ein Heraufsetzen der Zeit durch den Anwender kann die allgemeine SPS-Belastung durch Kommunikationsaufgaben vermindert werden. Der Wert ist in ms angegeben.

- SPS-Status: Zeigt den augenblicklichen Betriebszustand (z. B. „Run“, „Stop“, etc) der b maXX drive PLC an.
- Fehler: Zeigt an, ob eine SPS-Fehlermeldung im Fehlerkatalog ansteht und über den Button „Fehler“ in der b maXX drive PLC Ressourcenkontrolle ausgelesen werden kann.
- Zugriffsrechte: Zeigt die Zugriffsrechte zum Senden und Debuggen auf der SPS an.
- Durchlaufkontrolle: Zeigt an, ob der Adressstatus mit Durchlaufkontrolle vom Anwender aktiviert wurde (Menüpunkt „Online/Adressstatus“).
- Bootprojekt: Name des (inaktiven) Bootprojekts auf der b maXX drive PLC (Die Darstellung des Projektnamens ist auf 11 Buchstaben begrenzt).
Über die Seite für die Programmübermittlung (b maXX drive PLC Ressource Übertragung) kann durch den Menüpunkt „Aktivieren“ das Bootprojekt in das aktive RAM-Projekt übersetzt werden. (Geschieht immer automatisch bei einem HW-RESET oder Spannung aus/ein).
- Projektcode: Zeigt an, ob ein Source-Projekt (zwt) in der PLC vorhanden ist (Wird aus Speicherplatzgründen zur Zeit nicht unterstützt).
- Logikanalyse: Zeigt an, ob die Logikanalyse-Aufzeichnung gerade aktiv ist.

4.4.3 Datenbereich

Der Datenbereich enthält die b maXX drive PLC spezifische Einstellung von physikalischen Adressbereichen in Form von Merkern, auf die der Anwender und der Übersetzer in PROPROGRAMM zugreifen können.

Die Einstellungen werden mit dem Projekt abgespeichert.

Das „Datenbereichs“-Formular kann im Projektbaum über die „Einstellungen/Datenbereich...“ (rechte Maustaste) in der Ressource „Physical Hardware/RES1“) erreicht werden:

Datenbereiche für SH03_30

Nicht remanente Merker

Anfangsadresse Anwender: 0

Ende Anwender / Start System: 80000

Ende System (max 2097147): 2097147

Reserve pro POE: 500

Remanente Merker

Anfangsadresse Anwender: 10000000

Ende Anwender / Start System: 10000000

Ende System (max 10057323): 10057323

Reserve pro POE: 10%

☒ Autom. Deklaration von Anwenderspeicher in I/O-Konfiguration

OK Abbrechen Hilfe

Abbildung 22: Ressource b maXX drive PLC spezifischer Datenbereich ohne Anwenderbereich Remanente Merker.

Die Einteilung des Datenbereichs erfolgt dabei in zweierlei Hinsicht:

Erstens untergliedert sich der Datenbereich in „Nicht remanente Merker“ und „Remanente Merker“.

Zweitens wird in beiden Bereichen unterschieden, ob der Anwender oder der Übersetzer in PROPROGRAM auf diese Merker zugreifen dürfen.

Die beiden auf dem Bild sichtbaren Einstellungen "Reserve pro POE" dient der Compiler-Funktionalität „Online-Änderungen senden“ und reserviert Programmspeicher.

Begriffserklärung „Nicht remanente“ und „Remanente“ Merker:

Remanente Merker behalten im Gegensatz zu den nicht remanenten Merkern ihren Wert auch nach Wegnahme der Spannungsversorgung, so dass mit diesen Werten beim nächsten Wiederanlauf der PLC weitergearbeitet werden kann. Bevorzugt werden darin anlagenspezifische Daten abgelegt, die erst während des Betriebes ermittelt werden und nicht aus anderen Werten neu berechnet werden können.

Als Beispiel dafür sei ein Wickeldurchmesser genannt, der sich über die Geschwindigkeit und der Maschinenlaufzeit kontinuierlich verändert.



HINWEIS

Die Standardausführung des Optionsmoduls BM4-O-PLC-01 enthält keine remanenten Merker. In diesem Fall sollten die bestehenden Standard-Einstellungen für remanente Merker im Datenbereich nicht geändert werden.

Bei einer b maXX drive PLC ohne remanente Merker wird im Online-Info-Fenster beim Eintrag „Remanente Daten“ der Wert 0 angezeigt. In diesem Fall wird bei Projekt-Download über „Projekt/Senden“-Buttons im Fenster „PLC Ressource Übertragung“ vom System ein entsprechender SPS-Fehler rückgemeldet, wenn fälschlicherweise remanente Merker im Anwender-Datenbereich eingestellt waren.

Bei Systemen mit remanenten Merkern wird bei der Anzeige "Remanente Daten" der verfügbare remanente Gesamt-Speicherbereich auf der PLC angezeigt.

Begriffserklärung **Anwender-Merker** („Anfangsadresse-Anwender/ Ende Anwender"):

Will der Anwender in seinem Projekt direkt auf Merker zugreifen, können bei den nicht remanenten und remanenten Merkern dafür Bereiche vorab reserviert werden.

Man sollte jedoch darauf achten, dass die Anwenderbereiche nicht zu groß eingestellt werden, da sonst bei größeren Projekten u. U. zu wenig Systemspeicher für den PRO-PROG-Übersetzer vorhanden ist.

Wieviele Bytes des Speichers vom Projekt für diese Merker verbraucht werden, ist unter Infos im Meldungsfenster ersichtlich. Vorher muss eine Projekt-Code-Erzeugung erfolgen, wobei in den Ressource-Einstellungen „Keine Reserve“ aktiviert sein muss.

Begriffserklärung **System-Merker** („Start System/Ende System"):

In PROPROG wt kann der Anwender symbolische Variablen definieren und in seinem Programmcode verwenden, ohne eigens Merker dafür anlegen zu müssen. Das wird beim Übersetzungsvorgang in PROPROG wt vom System automatisch erledigt, indem diese Variablen auf die Merker im Systembereich (= vom System vergebene Merker) abgebildet werden.

Da dies bei jeder Projekt-Code-Erzeugung erfolgt, können die Teilungsgrenzen auch im späteren Projektverlauf noch angepasst werden.

Wieviele Bytes des Speichers vom Projekt für diese Merker verbraucht werden, ist unter Infos im Meldungsfenster ersichtlich. Vorher muss eine Projekt-Code-Erzeugung erfolgen, wobei in den Ressource-Einstellungen „Keine Reserve“ aktiviert sein muss.

Der voreingestellte Standard-Anwenderbereich für die Vergabe absoluter Merker-Adressen innerhalb einer b maXX Ressource im nicht remanenten Bereich ist:

- %MB 0 - %MB 79999

Der restliche Bereich von %MB 80000 auf z. B. %MB 2097147 bleibt dann dem System vorbehalten für die automatische Umsetzung der nicht remanenten symbolischen Anwender-Variablen in diesen Merkerbereich.

Der Standard-Anwenderbereich für die Vergabe absoluter Merker-Adressen innerhalb einer b maXX Ressource im remanenten Bereich ist deaktiviert.

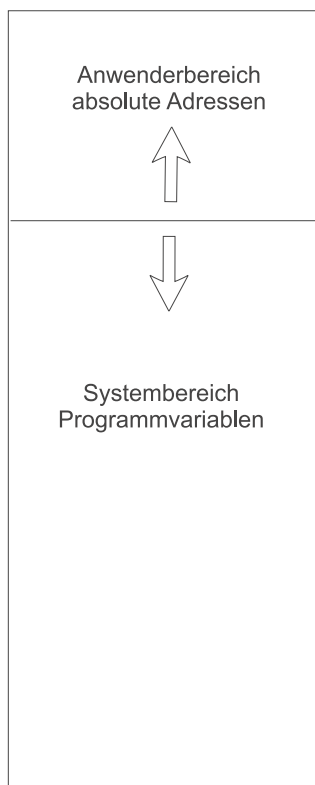
Bei PLC-Ausführungen mit remanenten Merkern kann der Anwender der jeweiligen Projektanforderung nach die Teilgrenze "Ende Anwender/Start System" nach oben verschieben (z. B. von 10000000 auf 10020000).

Dies würde dann einem remanenten Merkerbereich %MB 10000000 - %MB 10019999 entsprechen, den der Anwender dann in seinem Projekt als absolute remanente Merker direkt ansprechen kann.

Der restliche Bereich von %MB 10020000 auf z. B. %MB 10057323 ist dann dem System vorbehalten für die automatische Umsetzung von remanenten symbolischen Anwender-Variablen in diesen Merkerbereich.

Werden absolute Merker-Adressen vom Anwender verwendet, die im Systembereich liegen, erfolgt beim Übersetzen des Projektcodes eine entsprechende Fehlermeldung.

Nicht Remanenter Datenbereich



Remanenter Datenbereich (optional)

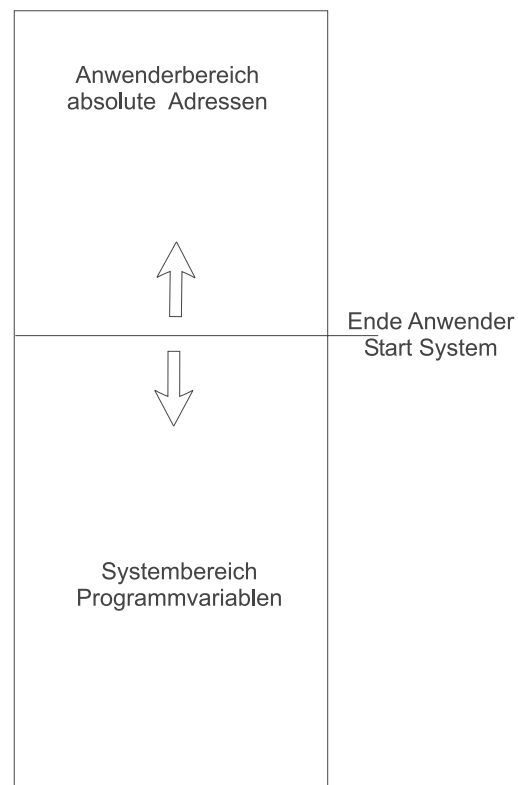


Abbildung 23: Aufteilung und Einstellung des b maXX drive PLC Datenbereichs

Vergabe von absoluten Merker-Adressen im Programm

Eine absolute b maXX drive PLC-Adresse oder ein Variablenfeld mit einem Datentyp

- 16-Bit (WORD) kann nur für eine ohne Rest durch zwei zu dividierende Adresse und Null vergeben werden.
- 32-Bit (DWORD) kann nur für eine ohne Rest durch vier zu dividierende Adresse und Null vergeben werden.

Beispiel:

Eine Variable vom Datentyp DWORD soll im nicht remanenten Datenbereich deklariert werden.

PROPROG wt II:

```
d_abs_adr AT %MD12 : DWORD;      (* Symbolische Variable auf
                                   absoluter Adresse *)
```

oder auch:

```
di_abs_adr AT %MD16 : DINT;      (* Symbolische Variable auf
                                   absoluter Adresse *)
```

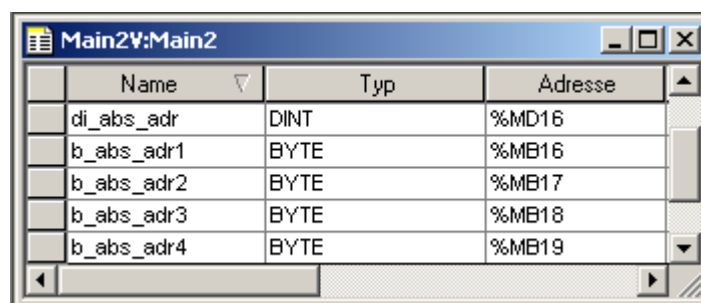
```
b_abs_adr1 AT %MB16 : BYTE;
```

```
b_abs_adr2 AT %MB17 : BYTE;
```

```
b_abs_adr3 AT %MB18 : BYTE;
```

```
b_abs_adr4 AT %MB19 : BYTE;
```

ProProg wt III:



Name	Typ	Adresse
di_abs_adr	DINT	%MD16
b_abs_adr1	BYTE	%MB16
b_abs_adr2	BYTE	%MB17
b_abs_adr3	BYTE	%MB18
b_abs_adr4	BYTE	%MB19

Abbildung 24: Variablendeklaration

Ein Vorteil bei der Verwendung von absoluten Merkern kann u. U. der schnelle und einfache Zugriff auf die einzelnen Variablenteile ohne zusätzliche Maskier- und Konvertierungsfunktionen sein:

Wird beispielsweise der Wert "DINT#16#98765432" auf die angelegte absolute Variable "di_abs_adr" geschrieben, kann der Variablen-Wert direkt als Einzel-Bytes weiterbearbeitet werden (INTEL-Format: b_abs_adr1 = BYTE#16#32, ..., b_abs_adr4 = BYTE#16#98).

Aber Vorsicht! Diese Art der Programmierung ist dann nicht mehr instanziiert, und die Sequenz könnte von einem Interrupt unterbrochen und von dort nochmal aufgerufen werden, so dass nach Beendigung des Interrupts mit falschen Werten weiter gearbeitet wird!

4.4.4 Die b maXX drive PLC Event-Tasks

Die b maXX drive PLC Event-Tasks dienen dem ereignisgesteuerten Programmaufruf (Interrupt). Sie bestimmen durch ihre Art und Codelaufzeit das Echtzeitverhalten.

Die Realisierung des Echtzeitverhaltens ist von der Art der Sollwertvorgabe und der Betriebsart des b maXX Reglers abhängig. Die Sollwertvorgabe kann beispielsweise umgesetzt werden:

- In einem „stand alone“ Antrieb über eine „BACI-Prozessdaten“ Event-Task.

- In einem vernetzten Antrieb über „SYNC Signal 1 Optionsmodul“ Event-Task.

Die Eigenschaft der Event-Task wird über die Ereignis-Nummer der Task zugewiesen:

Event	Bezeichnung	Level
0	CPU-Timer 1	14
1	reserviert	
2	CPU-Timer 2	13
3	reserviert	
4	BACI-Prozessdaten	13
5	Timer A	14
6	Timer A	13
7	reserviert	
8	BACI-Prozessdaten	14
9, 10	reserviert	
11	Sync-Signal 1 Optionsmodul	14
12	Sync-Signal 2 Optionsmodul	14

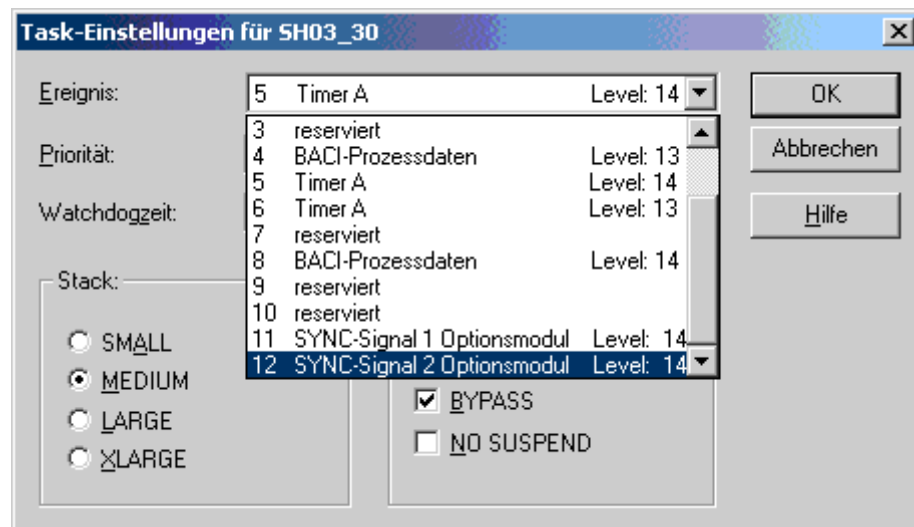


Abbildung 25: Die Ereignisse der b maXX drive PLC Event-Tasks.

**HINWEIS**

Alle b maXX drive PLC Tasks mit dem Task-Typ „EVENT“ sind von der Ressource abhängig und benötigen das Attribut **"BYPASS"**. Initialisierung und Aufruf der deklarierten Event-Task sowie deren Priorität erfolgen über die Initialisierungs-FBs innerhalb des Programms. Daher sind Priorität, Watchdogzeit, "SAVE FPU" und "NO SUSPEND" bei Bypass-Event-Tasks ohne Bedeutung.

Ein höherer Interrupt-Level bedeutet höhere Priorität.

Der Stack ist bei mehreren Event-Tasks, die sich gegenseitig unterbrechen können, oder Funktionsbausteinen, die mehrfach geschachtelt sind, auf "Large" bzw. "XLarge" anzupassen.

4.5 b maXX drive PLC Anwenderbibliotheken

Die PROPROG wt Anwenderbibliotheken gliedern sich in Firmware und Anwenderbibliotheken, die hardwareabhängig oder hardwareunabhängig sein können. Hardwareabhängige Bibliotheken können nur in Ressourcen des angegebenen Zielsystems verwendet werden. Die Hardwareabhängigkeit von b maXX drive PLC Bibliotheken ist mit ***_PLC01_** in der Bibliotheksbezeichnung gekennzeichnet.

PROPROG wt II: Die Version wird wie folgt angegeben: 20bd00.

- 20 ist der inkompatible Versionsstand.
- 00 ist der kompatible Versionsstand.

Der Versionsstand wird bei Kompatibilität der Ein- und Ausgangsvariablen der Funktionsbausteine um eins erhöht, z. B. 20bd01, 20bd02, usw. Wird an einem FB der Bibliothek z. B. ein Ein- oder Ausgang hinzugefügt, wird der Versionsstand vor dem „bd“ um eins erhöht und nach dem „bd“ auf Null gesetzt, z. B. 20bd03 auf 21bd00.

ProProg wt III: Die Version wird wie folgt angegeben: 30bd00.

- 30 ist der inkompatible Versionsstand
- 00 ist der kompatible Versionsstand.

Der Versionsstand wird bei Kompatibilität der Ein- und Ausgangsvariablen der Funktionsbausteine um eins erhöht, z. B. 30bd01, 30bd02, usw. Wird an einem FB der Bibliothek z. B. ein Ein- oder Ausgang hinzugefügt, wird der Versionsstand vor dem "bd" um eins erhöht und nach dem "bd" auf Null gesetzt, z.B. 30bd03 auf 31bd00.

Die Anwenderbibliotheken gliedern sich in:

- Firmware: b maXX drive PLC Board Funktionalität
z. B. Starten einer PROPROG wt Bypass Event-Task.
- Datentypen: b maXX drive PLC spezifische zusammengesetzte Datentypen und Felder, z. B. Registerstrukturen von Optionskarten.
- Standard-FBs: Elementare FBs für antriebsnahe Steuerungstechnik.
- Technologiebausteine: Komplett einsetzbare Antriebsfunktionalität.

Übersicht von PLC Bibliotheken die vollkommen hardwareunabhängig sind PROPROG wt II:

CD-ROM	Bibliothek	Bemerkung
Standard-Bibliotheken	BM_TYPES_20bd06	Baumüller Datentypen
	UNIVERSAL_20bd01	Basis-Funktionsbausteine
	MC_SYS_20bd01	Firmwarebausteine (werden von anderen Bibliotheken verwendet)

ProProg wt III:

CD-ROM	Bibliothek	Bemerkung
Standard-Bibliotheken	BM_TYPES_30bd01	Baumüller Datentypen
	UNIVERSAL_30bd00	Basis-Funktionsbausteine
	MC_SYS_30bd00	Firmwarebausteine (werden von anderen Bibliotheken verwendet)

Übersicht von PLC Bibliotheken die b maXX gruppenabhängig sind und somit sowohl von der b maXX controller PLC als auch von der b maXX drive PLC verwendet werden können (ab dem unten angegebenen Versionsstand)

PROPROG wt II:

CD-ROM	Bibliothek	Bemerkung
TB Kurvenscheibe	CAM_PLC01_21bd00	FBs für Technologiebaustein "Kurvenscheibe"
TB Wickler	WINDER_PLC01_20bd01	FBs für Technologiebaustein "Wickler"
CANopen	CANopen_PLC01_20bd03	FBs für Modul BM4-O-ETH-02 bzw. BM4-O-CAN-04 (CANopen-Master) und BM4-O-CAN-03 (CANopen-Slave)
Ethernet	TCP_PLC01_21bd01	FBs für Modul BM4-O-ETH-01 bzw. BM4-O-ETH-02 (Ethernet)

ProProg wt III:

CD-ROM	Bibliothek	Bemerkung
TB Kurvenscheibe	CAM_PLC01_30bd00 CAM_PLC02_30bd00	FBs für Technologiebaustein "Kurvenscheibe"
TB Wickler	WINDER_PLC01_30bd00 WINDER_PLC02_30bd00	FBs für Technologiebaustein "Wickler"

CD-ROM	Bibliothek	Bemerkung
CANopen	CANopen_PLC01_30bd00 CANopen_PLC02_30bd00	FBs für Modul BM4-O-ETH-02 bzw. BM4-O-CAN-04 (CANopen-Master) und BM4-O-CAN-03 (CANopen-Slave)
Ethernet	TCP_PLC01_30bd01 TCP_PLC02_30bd01	FBs für Modul BM4-O-ETH-01 bzw. BM4-O-ETH-02 (Ethernet)

Übersicht von PLC Bibliotheken die hardwareabhängig sind und nur von der b maXX drive PLC verwendet werden können

PROPROG wt II:

CD-ROM	Bibliothek	Bemerkung
Standard-Bibliotheken	SYSTEM1_PLC01_20bd00	Funktionsbausteine für BM4-O-PLC-01
	SYSTEM2_PLC01_20bd00	Firmwarebausteine für BM4-O-PLC-01

ProProg wt III:

CD-ROM	Bibliothek	Bemerkung
Standard-Bibliotheken	SYSTEM1_PLC01_30bd00	Funktionsbausteine für BM4-O-PLC-01
	SYSTEM2_PLC01_30bd00	Firmwarebausteine für BM4-O-PLC-01

Der Verzeichnispfad für die Bibliotheken ist unter PROPROG wt, Optionen, Verzeichnisse anzugeben. Die b maXX drive PLC Bibliotheken werden im Projektbaum von PROPROG wt unter Bibliotheken eingefügt.

Zu jedem FB ist eine HTML-Hilfe aufrufbar, die eine Beschreibung der Ein- und Ausgänge bereitstellt (siehe Programmierhandbuch PROPROG wt II bzw. Online-Hilfesystem ProProg wt III).

4.5.1 b maXX drive PLC Firmware

Die b maXX drive PLC Firmware besteht aus Funktionsbausteinen (FBs), die über Parameterübergabe mit Funktionen auf der b maXX drive PLC-CPU kommunizieren. Diese FBs sind nur ressourcenabhängig einsetzbar, d. h. vom Zielsystem abhängig.

PROPROG wt II:

Die BM4-O-PLC-01 Firmware ist mit der Bibliothek

SYSTEM2_PLC01_20bd00 (oder höher)

in ein Projekt einzufügen. Die Bibliothek beinhaltet folgenden Funktionsumfang:

- Bypass Event-Task starten und freiprogrammierbare LEDs an der b maXX drive PLC.
- P-, PI-Regler, 48-Bit Division über elektronisches Getriebe, Integrieren, Differenzieren.
- Funktionsbausteine zur Anschaltung an USS[®]- und 3964R[®]-Protokoll.



HINWEIS

Die b maXX drive PLC Firmware wird von einigen b maXX drive PLC Anwenderbibliotheken genutzt. Daher kann es erforderlich sein, mit einer b maXX drive PLC Anwenderbibliothek, die geforderte Firmware-Bibliothek SYSTEM2_PLC01_20bd00 oder höher einzufügen (siehe Beschreibung der jeweiligen Anwenderbibliothek). Die jeweils aktuellste Version ist auf der PROPROG-Programm-CD integriert und wird bei der Installation von PROPROG wt II automatisch mitinstalliert.

Bei Auswahl der „Vorlage für BM4_O_PLC01“ unter „Datei\neues Projekt“ ist die Bibliothek SYSTEM2_PLC01_20bd00 (oder höher) automatisch angelegt. Die Standard-Anwenderbibliothek SYSTEM1_PLC01_20bd00 (oder höher) wird im Regelfall ebenfalls in Projekten notwendig sein und wird auf einer eigenen CD ausgeliefert (siehe Übersicht in Abschnitt [►b maXX drive PLC Anwenderbibliotheken◄](#) auf Seite 47).

Dies gilt analog auch für ProProg wt III und die zugehörigen Bibliotheken.

ProProg wt III:

Die BMC-M-PLC-01 Firmware ist mit der Bibliothek

SYSTEM2_PLC01_30bd00 (oder höher)

in ein Projekt einzufügen. Die Bibliothek beinhaltet folgenden Funktionsumfang:

- Bypass Event-Task starten, freiprogrammierbare LEDs an der b maXX drive PLC.
- P-, PI-Regler, 48-Bit Division über elektronisches Getriebe, Integrieren, Differenzieren.
- Funktionsbausteine zur Anschaltung an USS[®]- und 3964R[®]-Protokoll.

4.5.2 b maXX drive PLC Board Funktionen

Die Firmware-Bibliothek SYSTEM2_PLC01_20bd00 (oder höher) enthält Funktionsbausteine (FBs) zur Kontrolle von Event Signalen für Interrupts und Board-LEDs. Um b maXX drive PLC Event-Tasks (Bypass) zu initialisieren und zu starten wird der Funktionsbaustein INTR_SET benutzt. Um freiprogrammierbare LEDs zu nutzen wird der Funktionsbaustein LED benutzt.

Für die Optimierung von Code-Laufzeiten innerhalb von b maXX drive PLC Ressourcen werden zwei FBs zur Code-Laufzeitmessung bereitgestellt.

PROPROG wt II: BM4-O-PLC-01:

Die FBs werden über die Anwenderbibliothek **SYSTEM1_PLC01_20bd00** und höher eingefügt.

ProProg wt III: BM4-O-PLC-01:

Die FBs werden über die Anwenderbibliothek **SYSTEM1_PLC01_30bd00** und höher eingefügt.

Der zu messende Code-Block innerhalb einer Task ist mit der Platzierung der FBs TIME_MEASURE_START und TIME_MEASURE_END einzugrenzen. Das Ergebnis der Laufzeit des gemessenen Code-Blocks wird am FB TIME_MEASURE_END als Zeitdifferenz in µs ausgegeben (siehe auch Online Beschreibung der FBs

TIME_MEASURE_START und TIME_MEASURE_END). Mit diesen Bausteinen können Laufzeiten bis max. 10 ms (BM4-O-PLC-01) ausgemessen werden.

4.5.2.1 Der Funktionsbaustein INTR_SET

Der Funktionsbaustein INTR_SET startet in einer Anlauf-Task eine Bypass Event-Task. Die PROPROGRAM Event-Task mit dem Programm muss auf das Ereignis und auf das Attribut "Bypass" gestellt werden.

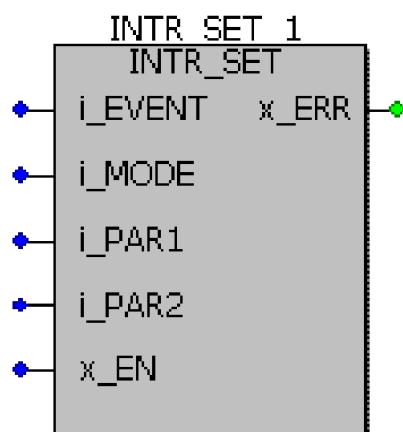


Abbildung 26: Bypass Event-Task über Funktionsbaustein INTR_SET initialisieren und freigeben.

Parameter	Eingang	Wertebereich
i_EVENT	Interrupt-Hardware-Programmnummer	8 bit signed
i_MODE	Reserve	16 bit signed
i_PAR1	CPU-Timer(1,2)-Wert-Multiplikator zur Basis 50µs	16 bit signed
i_PAR2	Reserve	16 bit signed
x_EN	Sperren/Freigabe des Interrupts	1 bit

Parameter	Ausgang	Wertebereich
x_ERR	Fehler-Bit	1 bit

Beschreibung:

Mit dem FB INTR_SET kann der Anwender diverse systemeigene Interrupt-Quellen konfigurieren und aktivieren. Diese Interrupts können dann im Programm zur Event-Task Aktivierung benutzt werden.

Am Eingang `i_EVENT` muss eine Ereignis Nummer angeschlossen werden. Diese Nummer spezifiziert die Interrupt-Quelle der Event-Task. Handelt es sich um einen CPU-Timer-Interrupt muss zusätzlich ein Faktor zur Zeitbasis 50 µs am Eingang `i_PAR1` angegeben werden.

Mit `x_EN = FALSE` kann ein vorher aktivierter Interrupt gesperrt werden.

Verzeichnis der Ereignis Nummern für Event-Tasks:

Event	Bezeichnung	Level
0	CPU-Timer 1	14
2	CPU-Timer 2	13
4	BACI-Prozessdaten	13
5	Timer A	14
6	Timer A	13
8	BACI-Prozessdaten	14
11	Sync-Signal 1 Optionsmodul	14
12	Sync-Signal 2 Optionsmodul	14

Die Ereignis-Nummer am Eingang `i_Event` muss identisch mit der Einstellung der Event-Task innerhalb der Ressource sein. Das Attribut "Bypass" muss aktiviert sein.



HINWEIS

Ein in der Priorität höherer Interrupt unterbricht einen niedrigeren. Für die Ereignisse 0 und 2, CPU-Timer 1 (oder 2) Interrupt, ist zusätzlich am Eingang `i_PAR1` der Faktor für die Zeitbasis 50 µs anzugeben.

Der höherpriorie Timer "CPU-Timer 1" (`i_EVENT = 0`) und der niederpriorie Timer "CPU-Timer 2" (`i_EVENT = 2`) können gleichzeitig verwendet werden.

Der "Board-Timer A" hohe Priorität (`i_EVENT = 5`) und "Board-Timer A" niedrige Priorität (`i_EVENT = 6`) sind nicht gleichzeitig verwendbar.

Die Ereignisse "SYNC-Signal 1 Optionsmodul" (`i_EVENT = 11`) und "SYNC-Signal 2 Optionsmodul" (`i_EVENT = 12`) sind nicht gleichzeitig verwendbar. Die "BACI-Prozessdaten" niedrige Priorität (`i_EVENT = 4`) und "BACI-Prozessdaten" hohe Priorität (`i_EVENT = 8`) sind nicht gleichzeitig verwendbar.

Für die Ereignisse `i_EVENT = 4, 8, 11` und `12` kann im Allgemeinen vollständig auf den Einsatz von `INTR_SET` verzichtet werden, da der `INTR_SET`-Baustein bereits im FB "BACI_INIT" (siehe Standard Bibliothek "SYSTEM1_PLC01_20bd00" (PROPROG wt II) bzw. "SYSTEM1_PLC01_30bd00" (ProProg wt III) oder höher) fest integriert ist.

Beispiel 1:

Die Anwender-POE (z. B. mit dem Namen "Timer") soll mit dem höherpriorien CPU-Timer 1 (`i_EVENT = 0`) verknüpft werden und zyklisch mit 5 ms Intervall aufgerufen werden.

Umsetzung:

In der Kalt- und Warmstart-Task wird eine INI-POE aufgerufen, die das konfigurierte INTR_SET enthält.

Für das Timer-Intervall gilt folgende Formel:

$$\text{Timer-Intervall} = i_PAR1 \cdot 50 \mu s$$

Für das gewünschte Intervall von 5 ms ergibt sich: $i_PAR1 := (5000 \mu s / 50 \mu s) = INT\#100$.

Darstellung des konfigurierten und freigegebenen INTR_SET:

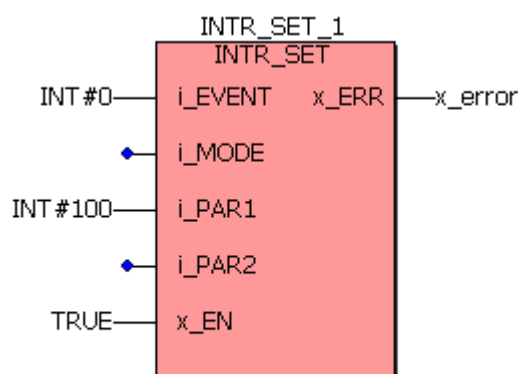


Abbildung 27: Funktionsbaustein INTR_SET: Start eines CPU Timer 1-Interrupts mit dem Aufrufintervall 5 ms.

Damit wird der Interrupt während des SPS-Startvorgangs eingerichtet und aktiviert.

Die Anwender-POE "Timer" muss in der Gruppe Task innerhalb der BM4_O_PLC01-Ressource als Task-Typ "EVENT" mit dem Ereignis "CPU_Timer 1" mit der Einstell-Option "BYPASS" verknüpft werden:

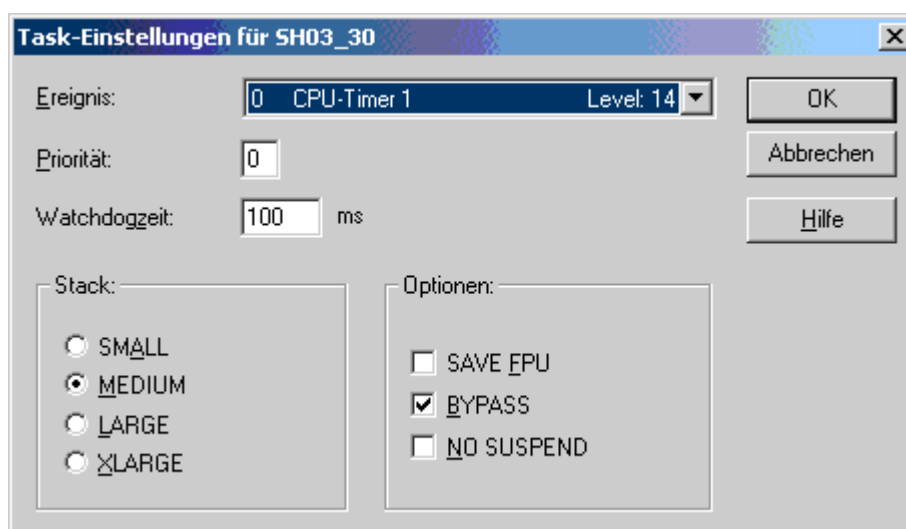


Abbildung 28: Verknüpfung mit dem Ereignis "CPU-Timer 1" als Bypass-Task.

Beispiel 2:

Start des „Timer A“ -Interrupts mit 5 ms und Generierung des Triggersignals am BACI-Signal-Ausgang "TRIGGER1":

Im Gegensatz zum CPU-Timer 1 (oder 2) kann der Board-Timer A auch als Triggersignal für Optionsmodule genutzt werden, die die Triggersignale benötigen. Denn nur mit dem „Timer A“ kann sowohl ein zyklischer Interrupt ausgelöst werden als auch eine Triggerrung erfolgen.

Zunächst erfolgt die Einrichtung der Event-Task innerhalb der BM4_O_PLC01-Ressource über das Event „Timer A“ mit der Ereignisnummer 5 (oder 6).

Innerhalb der Kalt/Warmtask wird zunächst zweimal der FB „TIMER_A_INIT“ (siehe Standard Bibliothek „SYSTEM1_PLC01_20bd00“ (PROPROP wt II) bzw. „SYSTEM1_PLC01_30bd00“ (ProProg wt III) oder höher) und dann der FB „INTR_SET“ aufgerufen.

- Mit dem 1. Aufruf von „TIMER_A_INIT“ wird der Timer für 5 ms eingerichtet.
- Mit dem 2. Aufruf wird das generierte 5 ms-Signal auf den BACI-Ausgang „TRIGGER1“ gelegt, das dann von den Optionsmodulen verarbeitet werden kann.
- Mit dem Aufruf „INTR_SET“ wird dann der Timer A - Interrupt eingerichtet und aktiviert, so dass der Anwender seine POE synchron zum generierten 5 ms-Triggersignal ablaufen lassen kann.

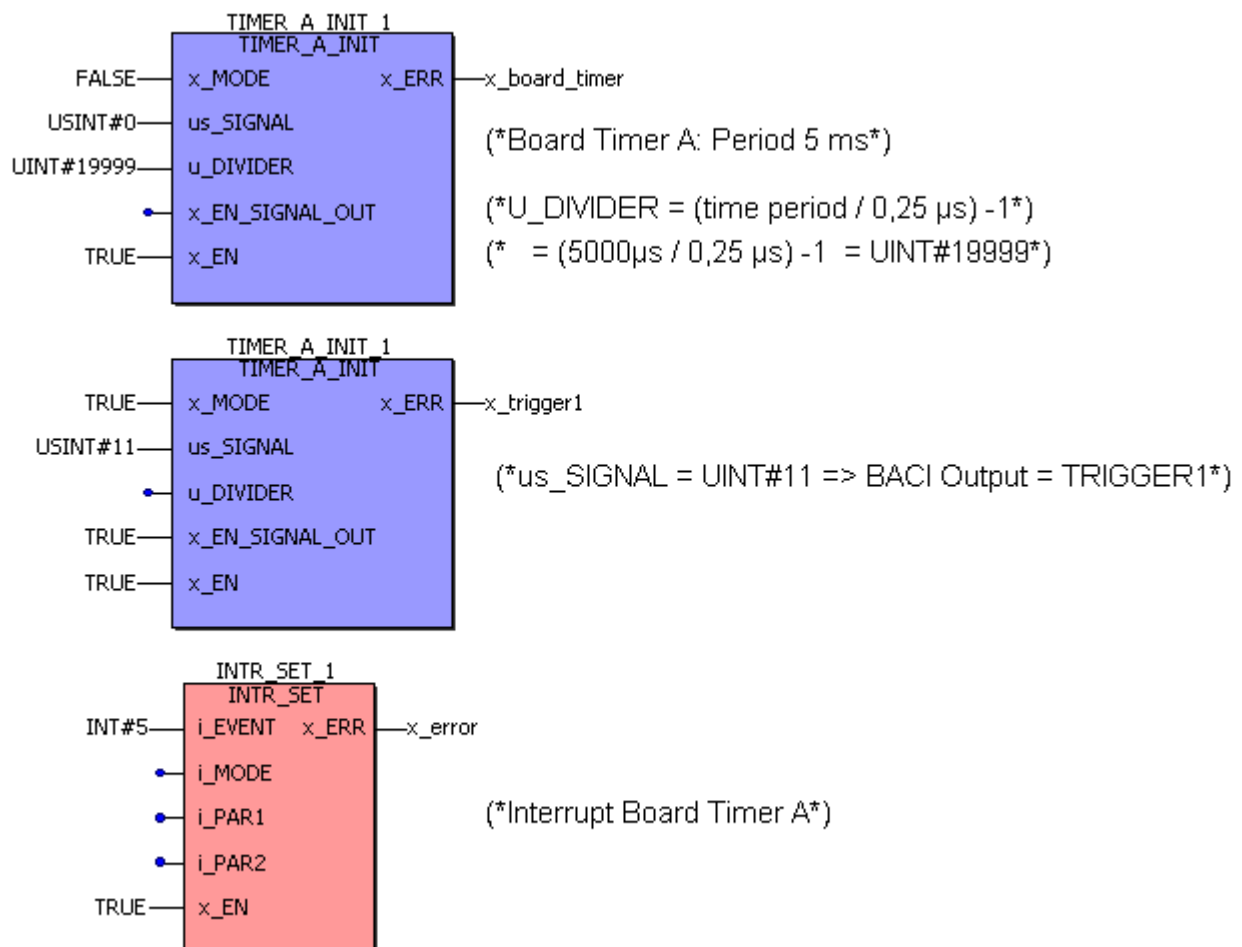


Abbildung 29: Start der "Timer A" Bypass Event Task mit der Periode 5 ms und gleichzeitige Nutzung von "Timer A" als Triggersignal für BACI-Optionsmodule.

4.5.2.2 Der Funktionsbaustein LED

Über den Funktionsbaustein LED aus der Firmware-Bibliothek **SYSTEM2_PLC01_20bd00** (PROPROG wt II) bzw. **SYSTEM1_PLC01_30bd00** (ProProg wt III) (oder höher), sind die b maXX drive PLC LEDs am Board programmierbar.

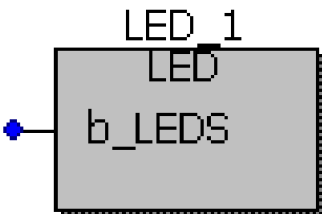


Abbildung 30: Funktionsbaustein LED

Parameter	Eingang	Wertebereich
b_LEDS	LED Setz-Maske	8 bit

Beschreibung:

Die linken LEDs am b maXX drive PLC-Board leuchten grün, die rechten rot. Die Bits 0-3 des 8-Bit-Musters am LED-Eingang werden auf die vier LEDs wie folgt ausgegeben:

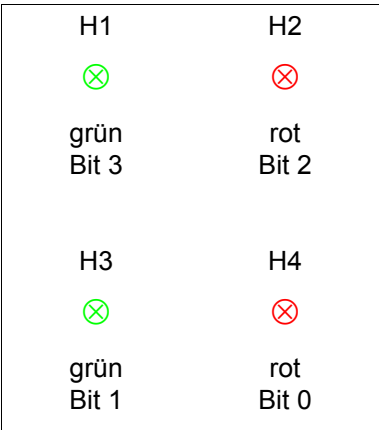


Abbildung 31: LEDs von Optionsmodul BM4-O-PLC-01

4.5.3 Die b maXX drive PLC Datentypen

In einem PROPROG wt Projekt erfolgt der Datenaustausch zwischen der b maXX drive PLC und den Optionsmodulen für b maXX drive PLC über Variablen vom entsprechenden Datentyp. Die Anwenderbibliothek **BM_TYPES_20bd03** (PROPROG wt II) bzw. **BM_TYPES_30bd01** (ProProg wt III) oder höher stellt bereits eine Vielzahl von vorgefertigten Datentypen (Strukturen und Arrays) zur Verfügung. Diese strukturierten Datentypen werden in der Regel in den ebenfalls vorgefertigten Standard- und Technologie-

Bibliotheken verwendet, um somit auf sehr einfache Weise mit den Optionsmodulen kommunizieren zu können.

Dazu sind im Template BM4_O_PLCC01 ("Datei/Neues Projekt.../Vorlage für BM4_O_PLCC01") bereits die wichtigsten Variablen für die verschiedenen Funktionen und Steckplätze der einzelnen Optionsmodule vordefiniert worden.

Der Anwender braucht somit nur noch die steckplatzabhängige Variable aus dem Template für sein gestecktes Optionsmodul auswählen und an die entsprechenden Ein- und Ausgänge der eingesetzten Funktionsbausteine (die in den verschiedenen Bibliotheken zu den Optionsmodulen zur Verfügung stehen) anschließen.

In den Bibliotheken wird an den relevanten Stellen auf die eingesetzten Datentypen der BM_TYPES_20bd03 (PROPROP wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher verwiesen.

Die vordefinierten Datentypen in BM_TYPES_20bd03 (PROPROP wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher können und sollen selbstverständlich auch vom Anwender verwendet werden:

Das Arbeitsblatt der BM_TYPES_20bd03 (PROPROP wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher ist im Projektbaum nicht direkt aufrufbar; die darin enthaltenen Daten-Typen können jedoch eingesehen werden, indem der Projektbaum-Editor über die an der Fußzeile des Editors angebrachten Reiter von "Projekt" auf "Bibliotheken" umgestellt und auf das Arbeitsblatt BM_TYPES_20bd03 (PROPROP wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher doppelgeklickt wird:

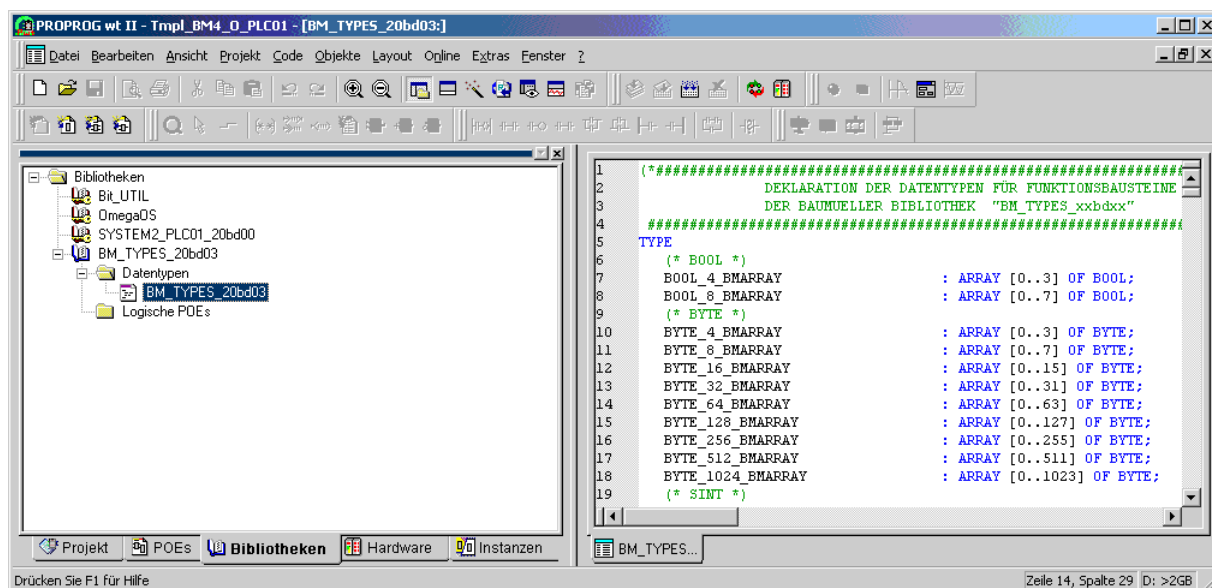


Abbildung 32: Einblick in die in Ihrem Projekt eingebundenen Baumüller-Datentypen "BM_TYPES_20bd03".

Auf diese Datentypen kann man dann bei der Vergabe von Variablen über den Variablendialog unter "Eigenschaften/Automatische Variablen-Deklaration" zugreifen.

PROPROP wt II:

In PROPROP wt II klicken Sie im Variablendialog auf den Button „Eigenschaften...“. Das Fenster „Automatische Variablen-Deklaration“ wird geöffnet:

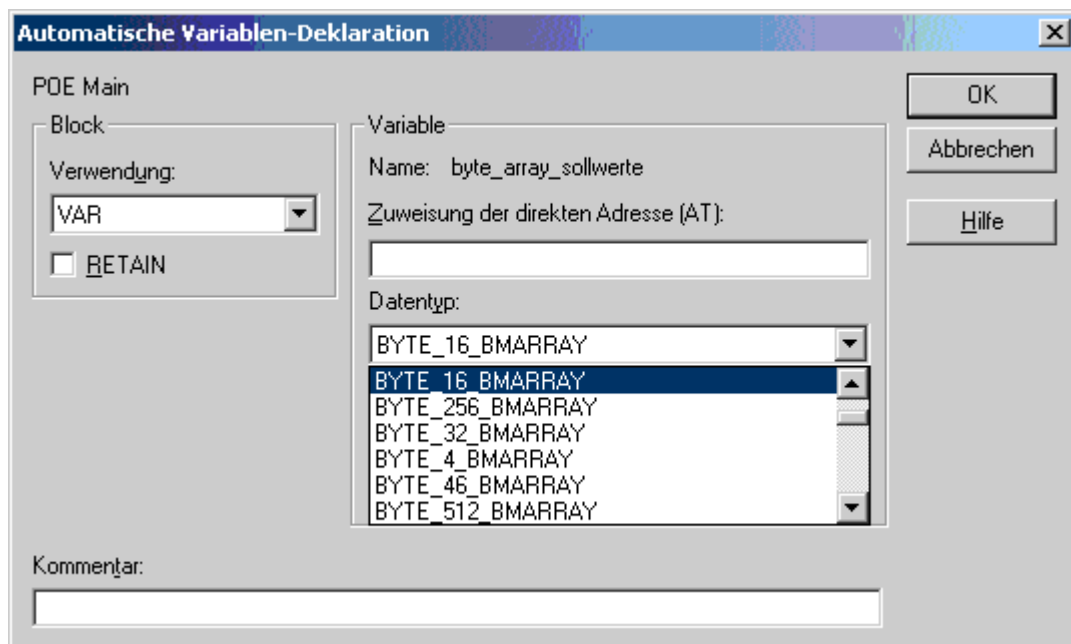


Abbildung 33: Automatische Variablen-Deklaration.

ProProg wt III:

In ProProg wt III ist die automatische Variablendeklaration im Variablendialog integriert.

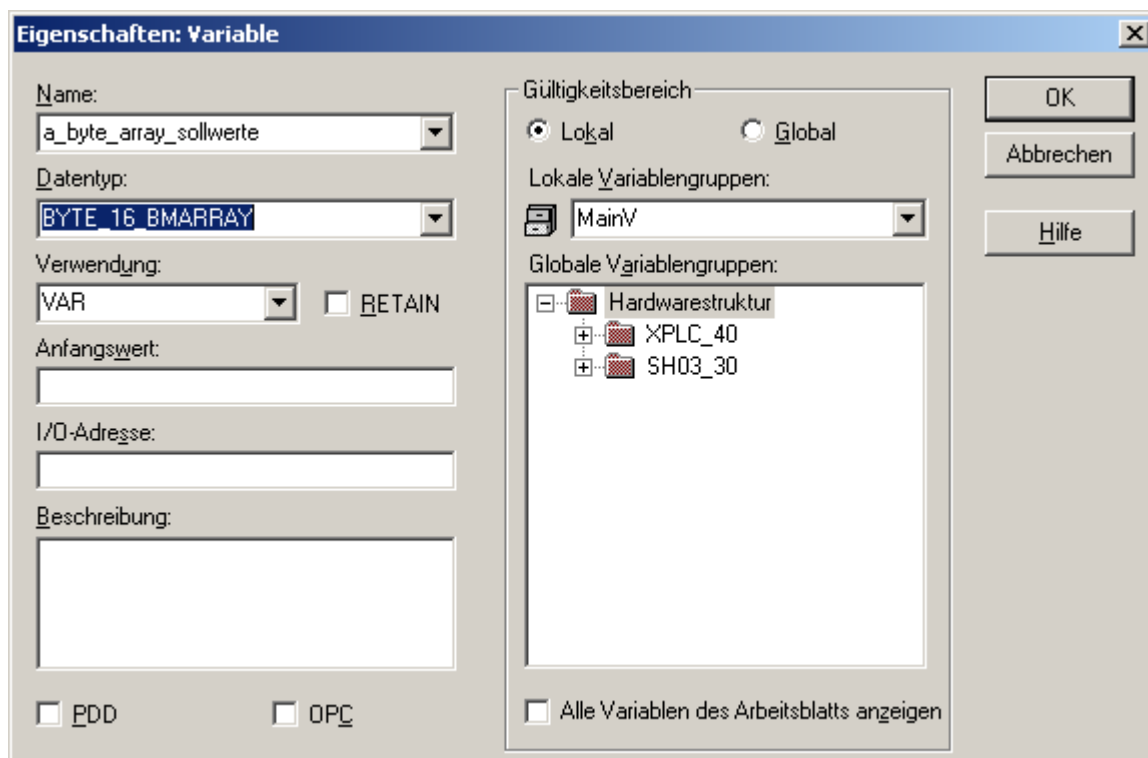


Abbildung 34: Variablendeklaration

Wenn Sie ein PROPROGRAMM Projekt erstellen und dazu das Template BM4_O_PLC01 verwenden, sind im globalen Variablen-Arbeitsblatt bereits die Variablen für den Datenaustausch deklariert. Für die Optionsmodule finden Sie eine (manchmal auch zwei) Variablen pro Steckplatz des Optionsmoduls.

Beispiel:

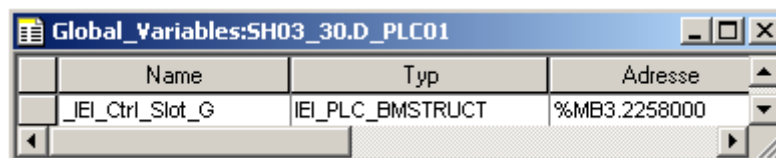
Das Optionsmodul BM4-O-IEI-01 benötigt zur Initialisierung Einstellungen in verschiedenen Registern.

Mit dem Optionsmodul auf Steckplatz Slot G ergibt sich folgende (im Template bereits vorgefertigte) Variablendeklaration:

PROPROGRAMM wt II:

```
_IEI_Ctrl_Slot_G AT      %MB3.2258000 : IEI_PLC_BMSTRUCT;
                        (* Option module IEI (BM4-O-IEI-01) *)
```

ProProg wt III:



Name	Typ	Adresse
_IEI_Ctrl_Slot_G	IEI_PLC_BMSTRUCT	%MB3.2258000

Abbildung 35: Variablendeklaration

Der komplette Registeraufbau der Struktur IEI_PLC_BMSTRUCT ist mit seinen Elementen und verwendeten Datentypen in den BM_TYPES_20bd03 (PROPROGRAMM wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher hinterlegt (Auszug):

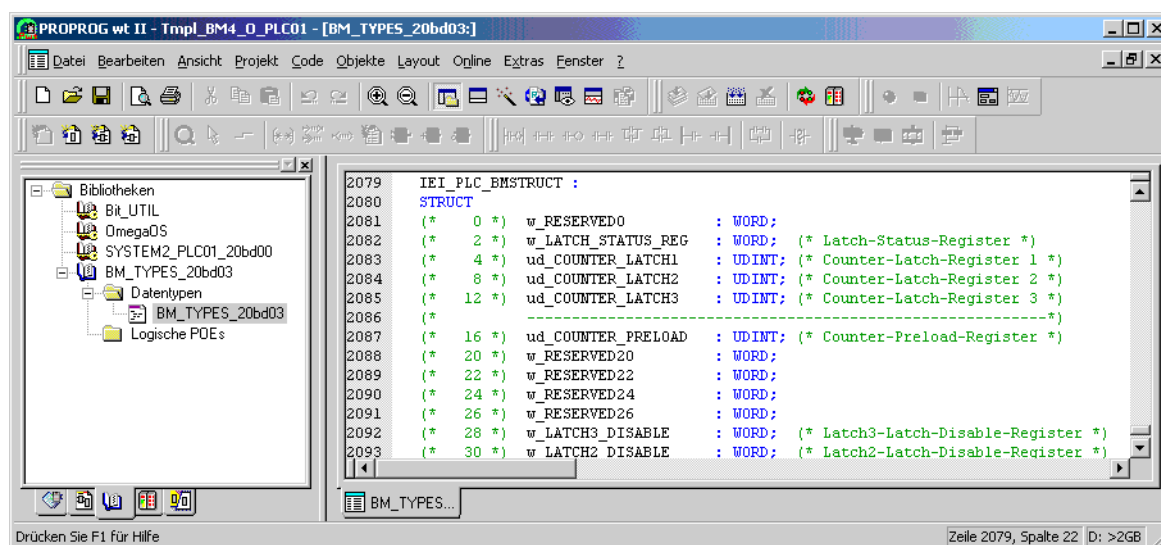


Abbildung 36: Auszug aus Registeraufbau der Struktur IEI_PLC_BMSTRUCT

Die Elemente der Variable `_IEI_Ctrl_Slot_G` bilden jetzt die Register des Optionsmoduls auf Steckplatz G ab. Weitere Informationen zum Einsatz des Optionsmoduls siehe Betriebsanleitung BM4-O-IEI-01.



HINWEIS

Im Variablendialog werden die Datentypen zur Zuweisung über Auswahlfenster bereitgestellt. Die einzelnen Datentypen der Bibliothek `BM_TYPES_20bd03` (PROPROG wt II) bzw. `BM_TYPES_30bd01` (ProProg wt III) oder höher sind dabei in der Regel mit dem Kürzel „_BM“ (`_BMARRAY`, `_BMSTRUCT`, usw.) gekennzeichnet. Das Arbeitsblatt der jeweiligen Bibliothek ist schreibgeschützt und kann über den Bibliotheksbaum (siehe oben) eingesehen werden.

4.5.4 Die Standard Funktionsbaustein-Bibliotheken

Die Standardbibliotheken enthalten Funktionsbausteine (FBs) mit Grundfunktionalitäten einer antriebsnahen Programmierung und Projektierung des Echtzeitverhaltens.

Diese Funktionsbausteine finden sich in:

PROPROG wt II:

- **SYSTEM1_PLC01_20bd00** (oder höher)
 - BACI-Kommunikation (Prozessdaten, Bedarfsdaten)
 - Funktionsbaustein für BACI (Input, Timer, Output, Trigger)
 - USS®-Protokoll-Anschaltung
 - Code-Laufzeitmessung
- **SYSTEM2_PLC01_20bd00** (oder höher)
 - b maXX drive PLC Firmware
 - Firmwarebaustein LED
 - Firmwarebaustein INTR_SET
 - 3964R®-Protokoll-Anschaltung
 - Terminal-Bausteine TER_x
- **UNIVERSAL_20bd01** oder höher (unabhängig von der Hardware)
 - Antriebs-Zustand und -Kontrolle durch den FB DRIVE1
 - Extrapolatoren, Rampengeneratoren, Positionsgeneratoren, Min-Max und Begrenzungs FBs
 - Virtuelle Leitachse FB TRAJECTORY_GEN1.

ProProg wt III:

- **SYSTEM1_PLC01_30bd00** (oder höher; für BM4-O-PLC-01)
 - BACI-Kommunikation (Prozessdaten, Bedarfsdaten)
 - Funktionsbaustein für BACI (Input, Timer, Output, Trigger)
 - USS®-Protokoll-Anschaltung
 - Code-Laufzeitmessung
- **SYSTEM2_PLC01_30bd00** (oder höher; für BM4-O-PLC-01)
 - b maXX drive PLC Firmware

- Firmwarebaustein LED
- Firmwarebaustein INTR_SET
- 3964R®-Protokoll-Anschaltung
- Terminal-Bausteine TER_x
- **UNIVERSAL_30bd00** oder höher (unabhängig von der Hardware)
 - Antriebs-Zustand und -Kontrolle durch den FB DRIVE1
 - Extrapolatoren, Rampengeneratoren, Positionsgeneratoren, Min-Max und Begrenzungs FBs
 - Virtuelle Leitachse FB TRAJECTORY_GEN1.

4.5.5 Die b maXX drive PLC Technologiebausteine

Die Standard-Anwenderbibliotheken lassen sich um komplette Antriebsfunktionalitäten, den Technologiebausteinen, erweitern. Diese sind:

- Technologiebaustein Kurvenscheibe: PROPROG wt II: Anwenderbibliothek
CAM_PLC01_21bd00 (oder höher)
ProProg wt III: Anwenderbibliothek
CAM_PLC01_30bd00 (oder höher)
- Technologiebaustein Registerregelung: PROPROG wt II: Anwenderbibliothek
REGISTER_PLC01_20bd00 (oder höher)
ProProg wt III: Anwenderbibliothek
REGISTER_PLC01_30bd00 (oder höher)
- Technologiebaustein Wickler: PROPROG wt II: Anwenderbibliothek
WINDER_PLC01_20bd00 (oder höher)
ProProg wt III: Anwenderbibliothek
WINDER_PLC01_30bd00 (oder höher)

Die Technologiebausteine bieten Antriebsfunktionalitäten, die durch die Beschaltung und mehrfache Instanziierung eine Vielzahl von Applikationslösungen bereitstellen.



HINWEIS

Alle Anwenderbibliotheken der Technologiebausteine benötigen zur Einbindung die Datentypen ab BM_TYPES_20bd03 (PROPROG wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher.

4.5.6 Einfügen einer Anwenderbibliothek in ein Projekt

Anwenderbibliotheken werden in PROPROG wt im Projektbaum unter Bibliotheken eingefügt. Handelt es sich um Firmware, ist bei der Anwahl das Dateiformat .fwl einzustellen.

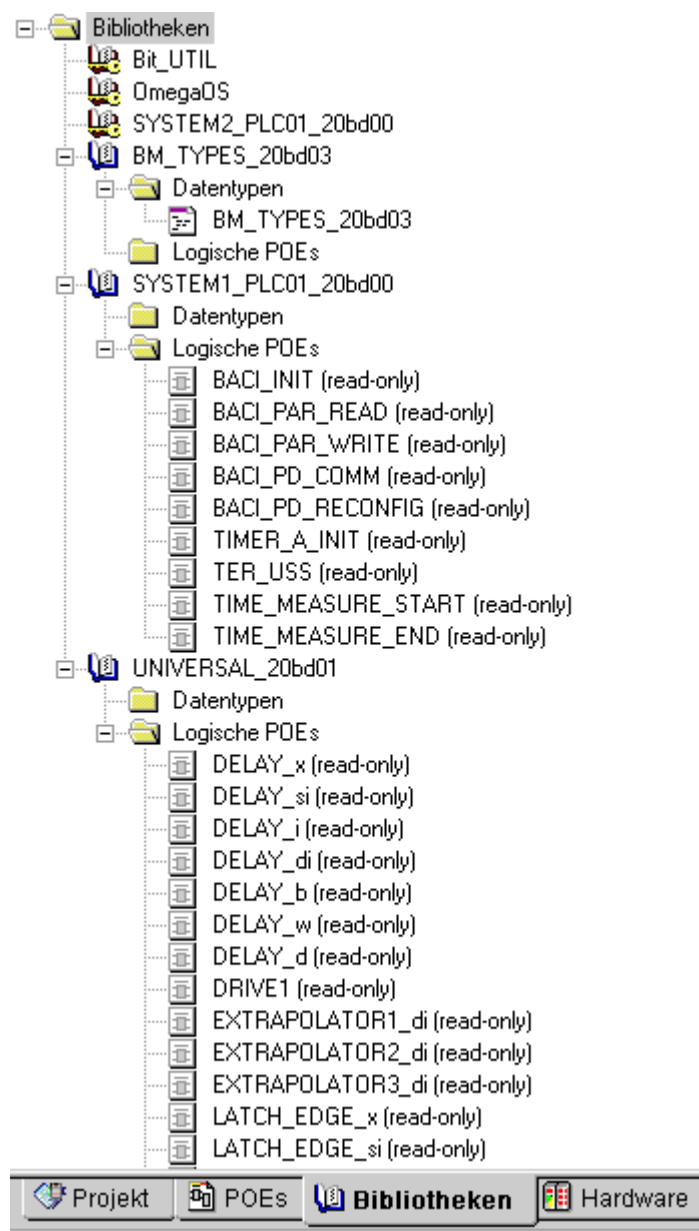


Abbildung 37: Standard-Auswahl von Anwenderbibliotheken, Firmware und Datentypen mit dem Filter Bibliotheken

HINWEIS



Die Auslieferung einer Bibliothek für PROPROGRAMT erfolgt gepackt (ZWT-File). Das ZWT-File ist unter PROPROGRAMT zu entpacken. Beim Entpacken wird die schreibgeschützte Bibliothek automatisch im angegebenen PROPROGRAMT Bibliothekspfad (Menü Extras>Optionen) abgelegt. In PROPROGRAMT II erscheint ein "Untitled"-Projekt, welches ohne Speicherung zu schließen ist. In ProProgramt III wird ein Projekt mit dem gleichen Namen wie das ZWT-File angelegt. Dieses Projekt wird nicht weiter benötigt.

Die Firmware-Bibliotheken werden dabei automatisch in das Firmware-Bibliotheks-Verzeichnis von PROPROGRAMT entpackt.

4.6 BACI-Systembeschreibung für die b maXX drive PLC

4.6.1 Direkter Zugriff auf die BACI-HW von der b maXX drive PLC aus

Die BACI ist ein Bussystem mit mehreren Steckplätzen, zwischen denen Daten ausgetauscht werden müssen. Jedes Modul ist dabei neben den BACI-Signalen in der Regel über ein DPRAM (=Dual Ported RAM) an den gemeinsamen BACI-BUS angeschlossen.

Es wird zwischen BACI-Master (b maXX Regler und b maXX drive PLC) und BACI-Slave unterschieden. Nur die BACI-Master dürfen auf die ihnen zugeordneten Optionsmodule über den BACI-BUS zugreifen. Zugriffe von BACI-Slave-Optionsmodulen untereinander oder gar auf die BACI-Master sind nicht möglich. Jedes einzelne Optionsmodul kann die vom Master geschriebenen Daten über das eigene DPRAM abfragen und auswerten.

Die b maXX drive PLC stellt gegenüber dem b maXX Regler einen BACI-Slave dar, so dass nur der Regler über den BACI-BUS auf die PLC zugreifen kann. Die vom Regler geschriebenen Daten kann die PLC über den eigenen DPRAM-Bereich auswerten.

Steckplatzübersicht BACI der Optionsmodule im Gerät b maXX 4400:

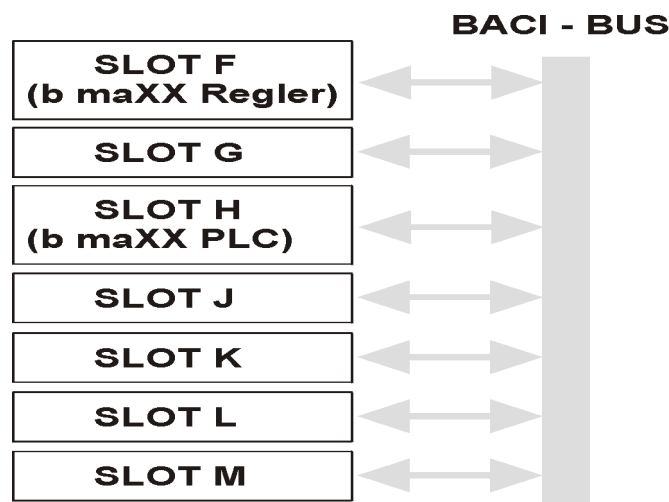


Abbildung 38: Steckplatzübersicht

Aus Sicht der PLC können in der Applikation nur die BACI-Steckplätze SLOT G bis SLOT M angesprochen werden, um mit den ihr zugewiesenen Optionsmodulen zu kommunizieren. Auch der Zugriff auf SLOT H (= PLC-Vorzugssteckplatz) über die BACI ist möglich, wenn die PLC selbst auf einem anderen SLOT steckt.

Steckplatz	Zugeordneter IEC 61131-3 - Bereich (max.)
SLOT G	%MB 3.20000000 - %MB 3.20262143
SLOT H	%MB 3.30000000 - %MB 3.30262143
SLOT J	%MB 3.40000000 - %MB 3.40262143
SLOT K	%MB 3.50000000 - %MB 3.50262143
SLOT L	%MB 3.60000000 - %MB 3.60262143
SLOT M	%MB 3.70000000 - %MB 3.70262143
Eigener DPRAM-Bereich der b maXX drive PLC	%MB 3.90000000 - %MB 3.90262143

HINWEIS



Ein SLOT-Zugriff von der PLC aus kann immer nur auf ein anderes Optionsmodul erfolgen. Um die vom Regler stammenden Daten auszuwerten, darf die PLC nicht den eigenen Steckplatz-Bereich ansprechen, in der sich die PLC selbst befindet, sondern muss immer über den eigenen DPRAM-Bereich (%MB 3.90000000 - %MB 3.90262143) gehen.

Eine Software-Struktur, die den Speicher- und Register-Aufbau eines Optionsmoduls beschreibt und die in einer vorgefertigten Bibliothek verwendet wird, kann dann in der Applikation einer Variable zugewiesen werden, die auf den verwendeten Steckplatz zeigt.

Solche globalen Variablen sind bereits in der BM4_O_PLC01 - Vorlage in PROPROG wt im globalen Variablenblatt "Global_Variables" entsprechend vordefiniert worden, so dass der Anwender diese je nach eingesetzten Optionsmodul und dem verwendeten Steckplatz nur noch an die eingesetzten Bibliotheksbausteine anschließen muss. In Zukunft werden diese Aufgaben vom Konfigurator automatisch übernommen, der in einer späteren Ausbauphase von PROPROG wt für die b maXX drive PLC integriert sein wird, so dass der Anwender dann mit den oben genannten SLOT-Bereichen nicht mehr in Berührung kommen wird.

Weitere Erläuterungen zu Strukturen und Funktionen der Optionsmodule finden sich in den Betriebsanleitungen und Applikationshandbüchern der jeweiligen Optionsmodule.

4.6.2 Prozessdatenkommunikation zwischen b maXX Regler und b maXX drive PLC, Synchronisierung auf eine gemeinsame BACI-Hardware-Signalquelle

Grundsätzlich unterscheidet man für die Prozessdatenkommunikation zwischen b maXX Regler und b maXX drive PLC zwei Anwendungsfälle:

1. Die Prozessdatenkommunikation zwischen b maXX Regler und b maXX drive PLC findet direkt statt, wobei Regler und PLC nicht auf ein BACI-Hardware-Signal synchronisiert sind. Dies bedeutet, dass der Regler die Interrupts auf der PLC für die Kommunikation unter Berücksichtigung der eingestellten Istwert-Periode jedesmal selbst auslöst, nachdem die Bearbeitung des eigenen Regler-Istwerte-Kanals abgeschlossen ist. Dadurch wird ein zeitgleicher Zugriff von Regler und PLC auf die BACI

vermieden, und die Kommunikation läuft immer synchron zueinander ab. Für diesen Fall ist am FB "BACI_INIT" der Bibliothek SYSTEM1_PLC01_20bd00 (PROPROP wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher am Eingang i_EVENT das Ereignis INT#4 (= Prioritäts-Level 13) oder INT#8 (= Prioritäts-Level 14) anzuschließen sowie eine Task mit dem gleichen Event anzulegen, in der die Prozessdatenkommunikation zum b maXX Regler über den FB "BACI_PD_COMM" ablaufen soll.

Prozessdatenkommunikation direkt über b maXX Regler (MasterCS_Actual1):

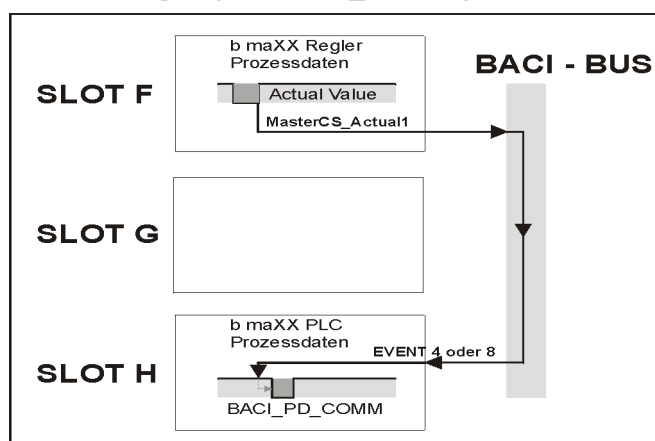


Abbildung 39: Prozessdatenkommunikation

2. Direkte Synchronisation auf eine gemeinsame Quelle:

Meistens stellt sich die System-Anforderung, dass mehrere Regler, wenn sie z. B. als Verbund zusammen geschlossen werden, synchron zueinander laufen müssen. Dies erreicht man, indem alle Regler auf ein gemeinsames BUS-Signal (z. B. über einen gemeinsamen CANsync-Ring) synchronisiert werden, welches an einer Stelle generiert und von den Optionsmodulen an die jeweils lokalen BACI-Busse der b maXX Regler durchgeschaltet wird (vorzugsweise auf das BACI-Signal SYNC1).

Ausgehend von Fall 1 würde dann ein Optionsmodul den b maXX Regler synchronisieren, der wiederum nach Abarbeitung des zugeordneten Istwerte-Kanals mit seiner PLC Prozessdaten ebenfalls synchron austauscht.

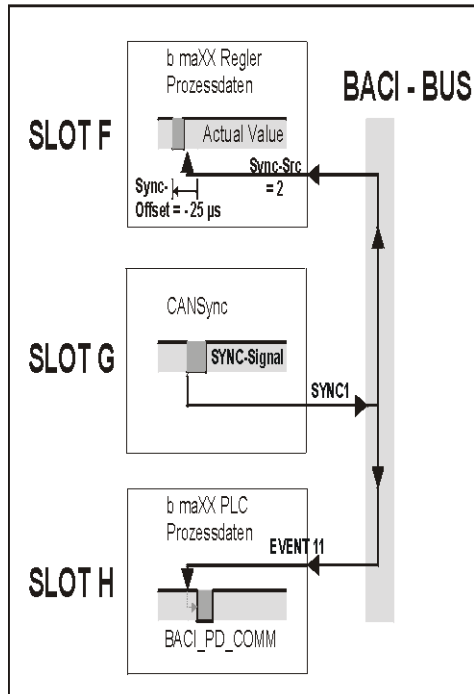
Da damit aber aus der Sicht der PLC eine gewisse Totzeit wegen der Bearbeitungsdauer im Regler verbunden ist, und meistens sowieso mit dem Optionsmodul, welches das Signal zum b maXX Regler weiterleitet bzw. generiert, (Feldbus-)Daten synchron ausgetauscht werden müssen, wird in der Praxis ein anderer Weg gewählt: Man synchronisiert sowohl den Regler als auch die PLC direkt auf das gleiche Hardware-Signal (meist BACI-Signal SYNC1).

Dadurch wird unter Echtzeit-Gesichtspunkten eine bessere, da nur von Hardware abhängige Synchronisierung aller Regler und PLCs erreicht.

Für diesen Fall 2 ist am FB "BACI_INIT" der Bibliothek SYSTEM1_PLC01_20bd00 (PROPROP wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher am Eingang i_EVENT das Ereignis INT#11 (= Prioritäts-Level 14, SYNC1-Signal von der BACI verwenden) bzw. INT#12 (= Prioritäts-Level 14, SYNC2-Signal von der BACI verwenden) anzuschließen sowie eine Ereignis-gleiche Task anzulegen, in der die Prozessdatenkommunikation zum b maXX Regler über den FB "BACI_PD_COMM" ablaufen soll und u. U.

zusätzlich eine Feldbus-Kommunikation zum Event-auslösenden Optionsmodul stattfindet.

Prozessdatenkommunikation über SYNC1-Signal:



Prozessdatenkommunikation über SYNC2-Signal:

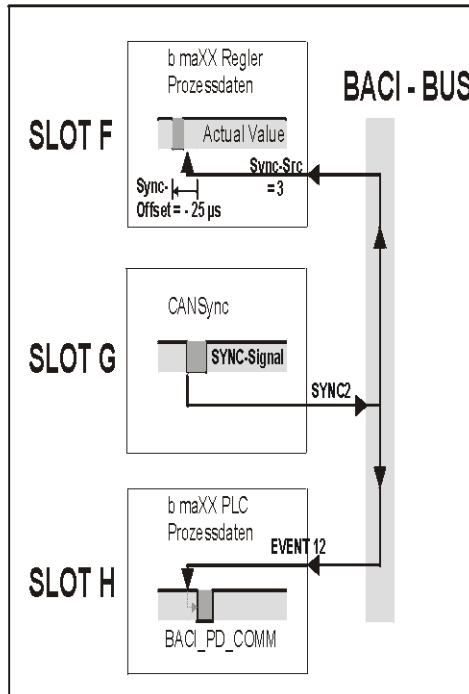


Abbildung 40: Prozessdatenkommunikation über SYNC1- bzw. SYNC2-Signal

Eines sollte man jedoch beachten: Bei einem per Default voreingestellten Regler-Parameter "Sync-Offset = 0" versuchen beim Prozessdatenaustausch sowohl der Regler als auch die PLC gleichzeitig auf die BACI -Daten zuzugreifen, und es kommt zu einem Zugriffskonflikt. Um die Zugriffszeitpunkte von Regler und PLC auf die BACI gegeneinander zu verschieben, muss im Regler ein negativer Sync-Offset von $-25\ \mu\text{s}$ eingestellt werden. Dadurch ist die Bearbeitung des Istwerte-Kanals im Regler immer abgeschlossen, und aktuelle Istwerte vom Regler stehen der PLC zur Verfügung, wenn die PLC den vom BACI-Signal ausgelösten Interrupt bearbeitet.

Die für die Synchronisation notwendigen Einstellungen der beiden Fälle im Regler sowie der PLC kann den Beschreibungen zu den FBs "BACI_INIT" und "BACI_PD_COMM" der Bibliothek SYSTEM1_PLC01_20bd00 (PROPROP wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher entnommen werden.

4.6.3 Für die Applikation nutzbare Interruptquellen der b maXX drive PLC

Für die b maXX PLC können Sie Hardware-Signale von der BACI, Schreibzugriffe vom Regler auf die PLC, und diverse PLC-interne Quellen wie CPU-Timer, Interruptquellen darstellen, die über die Ereignis-Auswahl der EVENT-BYPASS-Tasks mit spezifischen Programmteilen einer Applikation verbunden werden können.



HINWEIS ZU EVENT-BYPASS-EVENT-TASKS

Wird in PROPROG wt II eine Task vom Typ EVENT eingefügt, muss auch das Attribut "BYPASS" gesetzt werden.

Da EVENT-BYPASS-Tasks direkt durch das Ereignis aufgerufen und nicht vom Laufzeitsystem überwacht werden, sind die Einstellungen "Intervall" und "Watchdogzeit" in der Taskverwaltung ohne Bedeutung und werden ignoriert.

Weiterhin dürfen in den Programmteilen, die mit einer EVENT-BYPASS-Task verknüpft ist, prinzipiell keine Breakpoints gesetzt und auch kein Adressstatus aktiviert werden.

Eine Interruptquelle muss während der Kalt/Warmstart-Phase der PLC entweder über den Funktionsbaustein "BACI_INIT" (für die Ereignisse „4“, „8“, „11“ und „12“, siehe Standard-Bibliothek SYSTEM1_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher) oder "INTR_SET" für die Ereignisse „0“, „2“, „5“, „6“ (siehe Standard-Bibliothek SYSTEM2_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher) eingerichtet werden. Erst dann kann im SPS-Zustand "RUN" bei Auftreten des Ereignisses der in der EVENT-BYPASS-Task zugeordnete Programmteil aufgerufen werden.

Durch die Zuordnung von Ereignis zu Programmteil wird dieser Teil dann immer synchron und mit der gleichen Periodizität wie die Ereignis-Quelle abgearbeitet.

Darstellung aller EVENTS auf dem Optionsmodul BM4-O-PLC-01, die einen Interrupt auslösen können:

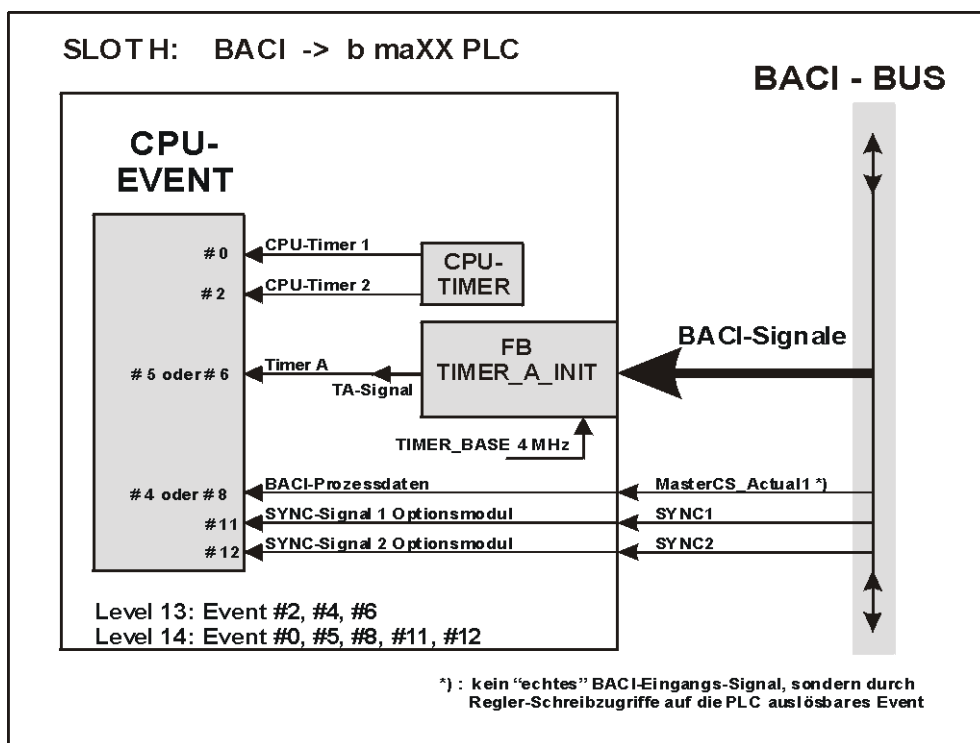


Abbildung 41: Darstellung aller EVENTS

Detail Konfigurationsmöglichkeiten für das EVENT Timer A durch Auswahl des BACI-Signals über die INPUT-Funktion von Baustein "TIMER_A_INIT" und Aufschaltung auf das interne TA-Signal:

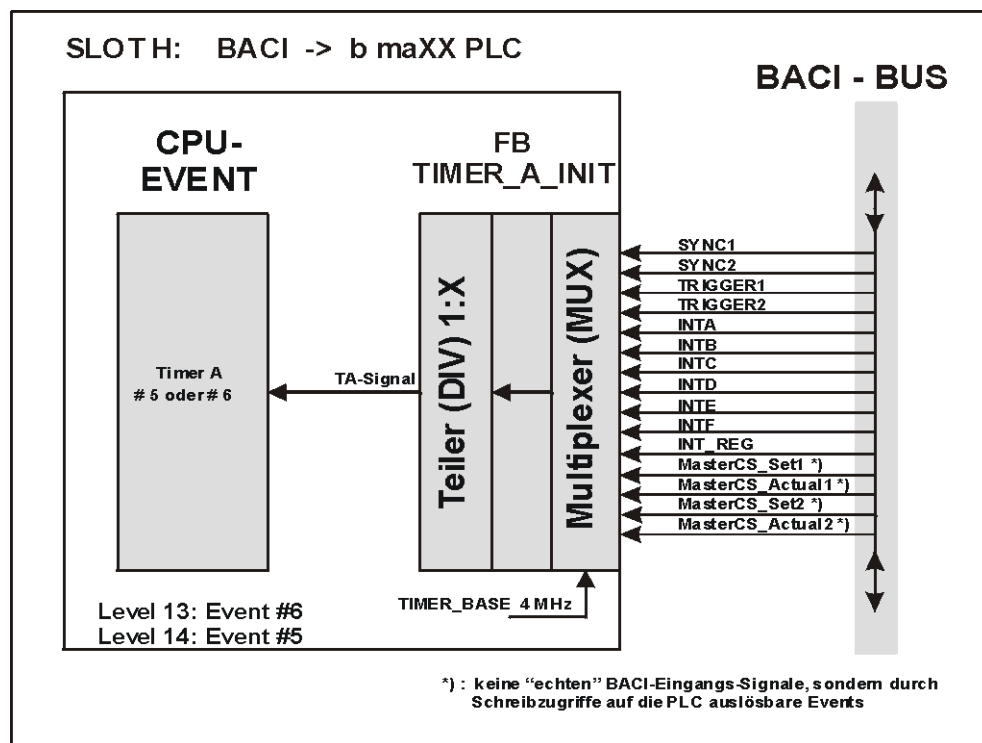


Abbildung 42: Detail Konfigurationsmöglichkeiten

4.6.4 BACI-Signalgenerierung über den Funktionsbaustein TIMER_A_INIT

Die b maXX drive PLC kann auch für eine BACI-Signalausgabe (b maXX drive PLC -> BACI) konfiguriert werden, wo sie dann z. B. als Triggerquelle für andere Optionsmodule fungieren kann.

Die Signalquelle wird dabei entweder auf der PLC selbst erzeugt oder stammt von einem anderen BACI-Signal, das über die PLC umgeleitet und gegebenenfalls heruntergeteilt wird (siehe Beschreibung zu Funktionsbaustein "TIMER_A_INIT" der Bibliothek SYSTEM1_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher).

Konfigurationsmöglichkeiten zur BACI-Signalgenerierung durch Aufschalten des intern generierten TA-Signals auf die BACI-Signalleitung über die OUTPUT-Funktion von Baustein "TIMER_A_INIT":

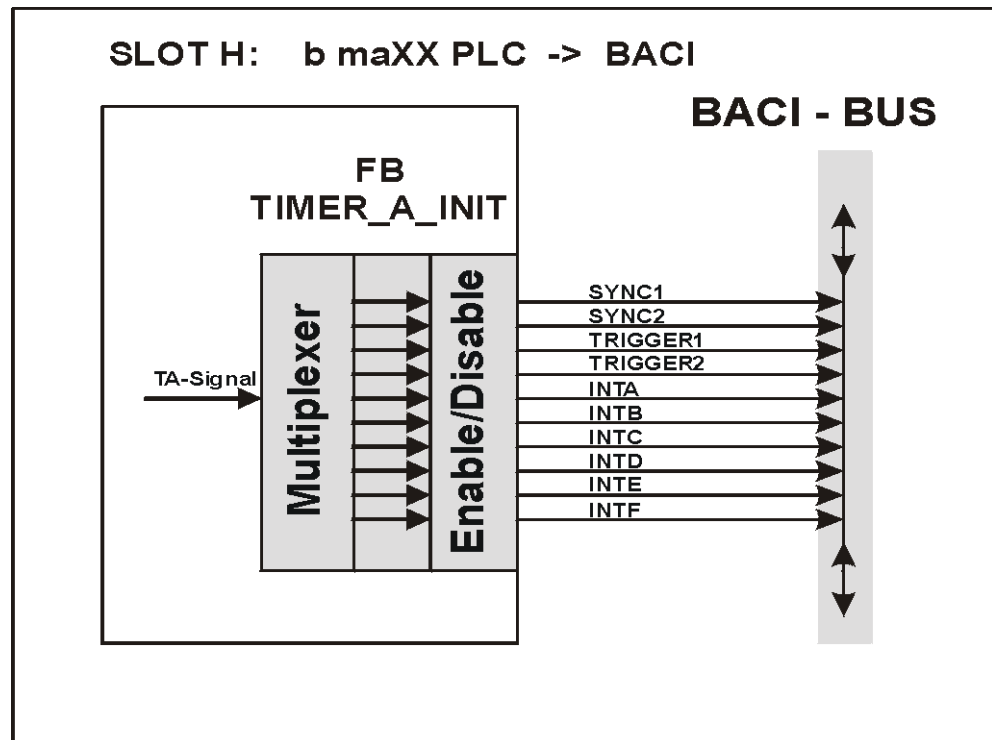


Abbildung 43: Konfigurationsmöglichkeiten zur BACI-Signalgenerierung

Durch Mehrfachaufruf der OUTPUT-Funktion kann das interne TA-Signal auch gleichzeitig auf verschiedene BACI-Signale durchgeschaltet werden.

Eine Umleitung von BACI-Signalen über die PLC (bei Bedarf mit einer Signal-Teilung) kann man erreichen, indem man die INPUT-Funktion und die OUTPUT-Funktion von "TIMER_A_INIT" entsprechend miteinander kombiniert (siehe "TIMER_A_INIT"-Beschreibung).

4.6.5 Funktionsbaustein "TIMER_A_INIT"

Beschreibung des Funktionsbausteins "TIMER_A_INIT" der Bibliothek SYSTEM1_PLC01_20bd00 (PROPROP wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher

Dieser Funktionsbaustein für BACI besitzt mehrere Funktionen:

Funktion	Beschreibung
INPUT	Sie können ein BACI-Signal auswählen, teilen und auf die interne TA-Leitung durchschalten sowie bei Bedarf damit einen TIMER_A Interrupt auslösen.
TIMER	INPUT-Funktion: Durch Auswahl der Source-Quelle "4MHz" (= Zeitbasis Timer_A) und Angabe eines Teilerwertes kann nahezu jedes beliebige Periodensignal erzeugt werden. Bei Bedarf kann damit ein TIMER_A Interrupt ausgelöst werden.

Funktion	Beschreibung
OUTPUT	Die interne TA-Leitung wird auf einen auswählbaren BACI-Ausgang geschaltet.
TRIGGER	Kombiniert man die TIMER-Funktion mit der OUTPUT-Funktion, kann das erzeugte Periodensignal auf einen oder mehrere BACI-Ausgänge geschaltet werden.
OUTPUT-Signal freigeben / sperren	BACI-Ausgänge können selektiv gesperrt und wieder freigegeben werden.

**HINWEIS**

zum Baustein `TIMER_A_INIT`:

Einsatz: Kann sowohl in der Initialisierungstask (Kalt/Warmstart-Task) als auch in der DE-Fault-Task oder in einer beliebigen Cyclisch Task eingebunden werden.

Der Baustein kann dabei auch mehrere Male an verschiedenen Stellen platziert oder auch öfters hintereinander aufgerufen werden, um die entsprechende Einzelfunktion zu installieren.

Wird der Baustein im zyklischen Teil verwendet, muss nach Aktivierung der Funktion der Eingang `x_EN` wieder auf `FALSE` zurückgestellt werden, damit der TIMER ablaufen bzw. der Teiler teilen kann!

Parameter Eingang	Datentyp	Beschreibung
<code>x_MODE</code>	BOOL	Auswahl der Grund-Funktion (Signal-Richtung)
<code>us_SIGNAL</code>	USINT	Signal-Quelle / Ziel
<code>u_DIVIDER</code>	UINT	Teiler-Wert
<code>x_EN_SIGNAL_OUT</code>	BOOL	Freigeben/Sperren des Ausgangssignals bei gewählter OUTPUT-Funktion
<code>x_EN</code>	BOOL	Freigabe des Bausteins

Parameter Ausgang	Datentyp	Beschreibung
<code>x_ERR</code>	BOOL	Fehlerbit

Eingang **`x_MODE`**:

Über `x_MODE` kann die grundlegende Funktion ausgewählt werden:

`x_MODE = FALSE` (`us_SIGNAL` = Signal-Quelle): Funktionen INPUT, TIMER. => Es wird ein internes TA-Signal generiert.

`x_MODE = TRUE` (`us_SIGNAL` = Signal-Ziel): OUTPUT, TRIGGER, OUTPUT-Signal freigeben/sperren. Wurde der BACI-Ausgang freigegeben, wird das interne TA-Signal auf das angegebene BACI-Signal-Ziel durchgeschaltet.

Eingang **us_SIGNAL**:

Für x_MODE := FALSE: us_SIGNAL :=	Auswahl der Signal-Quelle:
USINT#0	4MHz
USINT#1	MasterCS_Set1
USINT#2	MasterCS_Actual1
USINT#3	MasterCS_Set2
USINT#4	MasterCS_Actual2
USINT#5	INTA
USINT#6	INTB
USINT#7	INTC
USINT#8	INTD
USINT#9	INTE
USINT#10	INTF
USINT#11	TRIGGER1
USINT#12	TRIGGER2
USINT#13	SYNC1
USINT#14	SYNC2
USINT#15	INT_REG
> 15	Nicht erlaubt

Für x_MODE := TRUE: us_SIGNAL :=	Auswahl des BACI-Signal-Ziels:
USINT#0 - USINT#4	Nicht erlaubt
USINT#5	INTA
USINT#6	INTB
USINT#7	INTC
USINT#8	INTD
USINT#9	INTE
USINT#10	INTF
USINT#11	TRIGGER1
USINT#12	TRIGGER2
USINT#13	SYNC1
USINT#14	SYNC2
> 14	Nicht erlaubt

Eingang u_DIVIDER:

Wird nur bei der INPUT- und TIMER-Funktion ausgewertet ($x_MODE = FALSE$).

u_DIVIDER gibt an, in welchem Verhältnis das Eingangs-Signal geteilt wird. Dabei gilt folgende Teilungsformel:

$$\text{Internes TA-Signal} := \text{us_SIGNAL} / (\text{u_DIVIDER} + 1)$$

Bei u_DIVIDER = 0 wird somit das externe BACI-Signal direkt 1:1 auf das interne TA-Signal durchgeschaltet, es erfolgt keine Teilung des Signals.

Eingang x_EN_SIGNAL_OUT:

Wird nur bei der OUTPUT- und TRIGGER-Funktion ausgewertet ($x_MODE = TRUE$).

Das interne TA-Signal wird auf den ausgewählten BACI-Ausgang geschaltet ($x_EN_SIGNAL_OUT := TRUE$) bzw. wieder gesperrt ($x_EN_SIGNAL_OUT := FALSE$).

Durch mehrere Baustein-Aufrufe hintereinander und unterschiedlichen BACI-Zielen kann das interne TA-Signal auf mehrere BACI-Ausgänge ausgegeben werden.

Eingang x_EN:

Mit $x_EN = TRUE$ wird die jeweilige Funktion installiert. Bei $x_EN := FALSE$ erfolgt keine Bearbeitung des Bausteins. Wenn der Baustein zyklisch ausgeführt wird, sollte nach Aktivierung einer Einzel-Funktion der Eingang x_EN wieder auf FALSE zurückgestellt werden, damit der TIMER ablaufen bzw. das BACI-Eingangssignal geteilt werden kann.

Ausgang x_ERR := TRUE:

Auswahl des BACI-Ziels wird nicht unterstützt!

Erläuterung der Funktionen im Detail:**INPUT-Sonderfunktion: TIMER ($x_MODE := FALSE$ und $\text{us_SIGNAL} := \text{USINT\#0}$):**

Als Timer-Basis sind 4 MHz fest vorgegeben. Formel für die Berechnung der Zeitperiode:

$$\text{Zeitperiode} = (\text{u_DIVIDER} + 1) \times 0,25 \mu\text{s}$$

Minimalwert für "u_DIVIDER": UINT#999 => Zeit = 250 μs

Maximalwert für "u_DIVIDER": UINT#65535 => Zeit = 16384 μs

Formel für die Berechnung von u_DIVIDER bei gegebener Zeitperiode:

$$\text{u_DIVIDER} = (\text{Zeitperiode} / 0,25 \mu\text{s}) - 1$$

Wenn der Timer über das intern erzeugte TA-Signal einen Interrupt auslösen soll, muss nach Aufruf der TIMER-Funktion über "TIMER_A_INIT" der System-Funktionsbaustein "INTR_SET" (siehe FW-LIB SYSTEM2_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher) ausgeführt werden mit den Parametern: i_EVENT := INT#5 (bzw. i_EVENT := INT#6) für TIMER A. (Die anderen Parameter i_MODE, i_PAR1 und i_PAR2 werden ignoriert).

Dabei ist darauf zu achten, dass die Zeitperiode nicht zu klein bzw. zu schnell gewählt wird, sonst kann der nächste zyklische Interrupt bereits anstehen, noch bevor der gerade Laufende beendet ist. Da die Interrupts als Bypass-Interrupts mit hoher Priorität einge-

richtet sind, läuft nur noch dieser Interrupt, und die SPS geht offline (-> Die übrige Anwendung "reagiert nicht mehr").

Um dies zu vermeiden, wird ein Interruptintervall von mindestens 250 µs empfohlen. Dies entspricht einem "u_DIVIDER"-Wert von \geq UINT#999.

Sonderfunktion: TRIGGER

Wenn man die TIMER-Funktion mit der OUTPUT-Funktion kombiniert, kann das erzeugte Periodensignal als Triggersignal auf einen oder mehrere BACI-Ausgänge geschaltet werden:

1. Aufruf der TIMER-Funktion:

Als Timer-Basis sind 4 MHz fest vorgegeben. Wenn ein Teilerwert am Eingang "u_DIVIDER" kleiner UINT#3 angeschlossen wird, wird ein dauerhaftes LOW-Signal am internen Timerausgang TA erzeugt. Wenn das interne TA-Signal über die OUTPUT-Funktion anschließend auf die BACI ausgegeben werden soll (z. B. PLC-Funktion als BACI-Trigger-Quelle), sollte zuvor bei der INPUT-Funktion mindestens ein Wert größerer UINT#7 eingegeben werden, um das BACI-BUS-Timing nicht zu verletzen.

Formel für die Berechnung der Zeitperiode:

$$\text{Zeitperiode} = (\text{u_DIVIDER} + 1) \times 0,25 \mu\text{s}.$$

Minimalwert für "u_DIVIDER": UINT#8 => Zeit = 2,25 µs

Maximalwert für "u_DIVIDER": UINT#65535 => Zeit = 16384 µs

Formel für die Berechnung von u_DIVIDER:

$$\text{u_DIVIDER} = (\text{Zeitperiode} / 0,25 \mu\text{s}) - 1.$$

Soll der Triggerimpuls auch einen Interrupt auslösen, wird ein Interruptintervall von mindestens 250 µs empfohlen. Dies entspricht einem "u_DIVIDER"-Wert von \geq UINT#999.

2. Aufruf der OUTPUT-Funktion (BACI-Ziel = USINT#11 (Trigger 1)): siehe unten

OUTPUT-Sonderfunktion: BACI-Signal freigeben/sperren:

Falls das intern generierte TA-Signal an mehrere BACI-Ausgängen gleichzeitig ausgegeben werden soll, kann der Baustein mehrere Male hintereinander mit dem jeweiligen BACI-Ziel und "x_MODE := TRUE" sowie "x_EN_SIGNAL_OUT := TRUE" für die Treiberfreigabe aufgerufen werden. Mit "x_EN_SIGNAL_OUT := FALSE" kann die Freigabe selektiv wieder gesperrt werden.

Internes TA-Signal soll einen Interrupt generieren:

Nach Parametrierung von "TIMER_A_INIT" muss der System-Funktionsbaustein "INTR_SET" (siehe FW-LIB SYSTEM2_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher) ausgeführt werden mit den Parametern: i_EVENT := INT#5 (bzw. i_EVENT := INT#6) für TIMER A. (Die anderen Parameter i_MODE, i_PAR1 und i_PAR2 werden ignoriert).

Anwendungsbeispiel 1:

Für ein Optionsmodul BM4-O-IEI-01 soll die b maXX drive PLC auf der SYNC1-Leitung ein Triggersignal mit 2 ms Periode erzeugen. Synchron zum generierten Triggersignal soll ein SPS-Programmteil ausgeführt werden.

1. Schritt: Internes TA-Signal erzeugen über die Timer-Funktion mit der Zeitperiode 2 ms:
FB "TIMER_A_INIT" ausführen mit: x_MODE: = FALSE; us_SIGNAL := USINT#0; (für Timer-Funktion); u_DIVIDER = $(2000 \mu\text{s} / 0,25 \mu\text{s}) - 1$ = UINT#7999;
2. Schritt: TA-Signal als Triggersignal an SYNC1 ausgeben:
Baustein "TIMER_A_INIT" ausführen mit: x_MODE: = TRUE; us_SIGNAL := USINT#13; (Für Ziel-Auswahl SYNC1); x_EN_SIGNAL_OUT := TRUE;
3. Schritt: Baustein "INTR_SET" ausführen mit i_EVENT := INT#5.
=> Durch Anlegen einer EVENT-Bypass-Task mit dem Ereignis „5“ kann ein SPS-Programm synchron zum Triggersignal ausgeführt werden.

Anwendungsbeispiel 2:

Das Eingangs-BACI-Signal TRIGGER1 soll durch den Wert 10 geteilt und dann wieder auf die Ausgangs-BACI-Leitung TRIGGER2 geschaltet werden; außerdem soll das geteilte Signal einen Interrupt auslösen.

1. Schritt: TRIGGER1 -> 1:10 -> internes TA-Signal:
FB "TIMER_A_INIT" ausführen mit: x_MODE: = FALSE; us_SIGNAL := USINT#11; (Für Auswahl TRIGGER1) u_DIVIDER := UINT#9; (Für Teilverhältnis 1:10)
2. Schritt: Internes TA-Signal -> TRIGGER2:
Baustein "TIMER_A_INIT" ausführen mit: x_MODE: = TRUE; us_SIGNAL := USINT#12; (Für Auswahl TRIGGER2); x_EN_SIGNAL_OUT := TRUE;
3. Schritt: Baustein "INTR_SET" ausführen mit i_EVENT := INT#5.
=> Durch Anlegen einer EVENT-Bypass-Task mit dem Ereignis „5“ kann ein SPS-Programm synchron zum geteilten Signal ausgeführt werden.

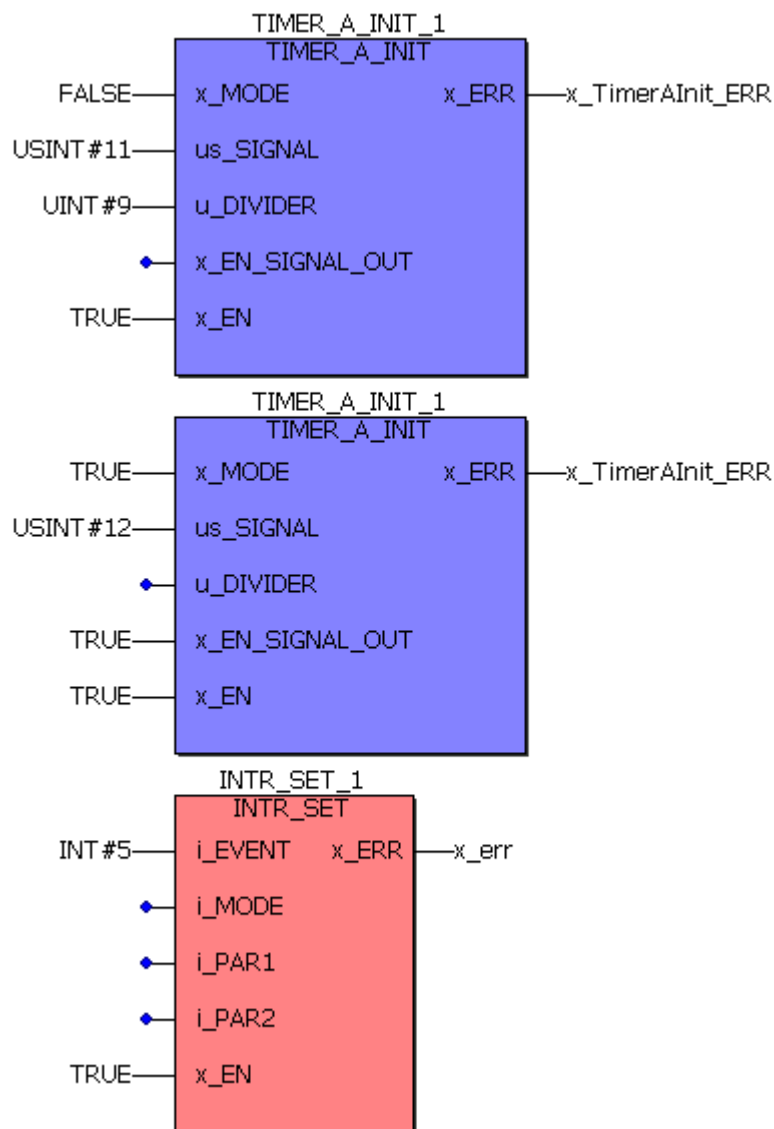


Abbildung 44: Das Signal TRIGGER1 löst jedes 10. mal das Signal TRIGGER2 aus und das Signal TRIGGER1 löst die Bypass Event-Task 5 aus

4.6.6 Konfigurationsbeispiele

Je nach verwendeten Optionsmodulen und Synchronisierungsanforderungen werden verschiedene Konfigurationen für die Prozessdatenkommunikation zwischen b maXX Regler und PLC, die Verwaltung von Interruptquellen sowie die Bereitstellung und Verarbeitung von Trigger- und Sync-Signalen benötigt.

Für die Mehrzahl der Anwendungsfälle sind folgende Beispiele verwendbar:

Beispiel A:

Umsetzung einer Prozessdatenkommunikation zwischen b maXX Regler und PLC.

Die Synchronisierung erfolgt durch die Regler-Funktionalität "MasterCS_Actual1" (Istwerte-Kanal des Reglers).

Über die gleiche Quelle soll auf der PLC ein SYNC1-Signal gebildet werden, mit der wiederum dann das Optionsmodul BM4-O-IEI-01 getriggert wird.



HINWEIS

Die Synchronisierung eines Optionsmoduls auf das Ereignis "MasterCS_Actual1" (Istwerte-Kanal des Reglers) kann über die PLC durch die Umlegung dieses Ereignisses auf ein BACI-Signal (z. B. SYNC1) erreicht werden!

Beispiel B:

Umsetzung einer hochgenauen BACI-Prozessdatenkommunikation durch Generierung des SYNC1-Signals und gleichzeitige Triggerung eines Optionsmoduls BM4-O-IEI-01 sowie des b maXX Reglers über das SYNC1-Signal.

Beispiel C:

Umsetzung einer BACI-Prozessdatenkommunikation innerhalb einer CANsync-Event-Task und Triggerung eines Optionsmoduls BM4-O-IEI-01 sowie des b maXX Reglers über das Synchronisierungssignal SYNC1.

Beispiel D:

Einrichten einer Timer-Event-Task für den zyklischen Aufruf, in der eine serielle RS485-Kommunikation (über die 9-pol. Sub-D-Buchse auf der b maXX drive PLC BM4-O-PLC-01 Schnittstelle X1) bearbeitet werden soll. Die Funktionsbausteine zur Kommunikation werden in einer POE platziert, die dieser Task zugeordnet sind.



HINWEIS

Für die Optimierung der Durchlaufzeiten von Prozessdaten gemäß den Konfigurationsbeispielen wird eine Code-Laufzeitmessung empfohlen. (Details dazu siehe [►Codelaufzeiten◄](#) ab Seite 110)

Diese kann mit den FBs TIME_MEASURE_START und TIME_MEASURE_END aus der Bibliothek SYSTEM1_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher durchgeführt werden.

4.6 BACI-Systembeschreibung für die b maXX drive PLC

4.6.7 Beispiel A: Umsetzung einer BACI-Prozessdatenkommunikation innerhalb einer BACI-Event-Task

Kalt-/Warmstart-Task:

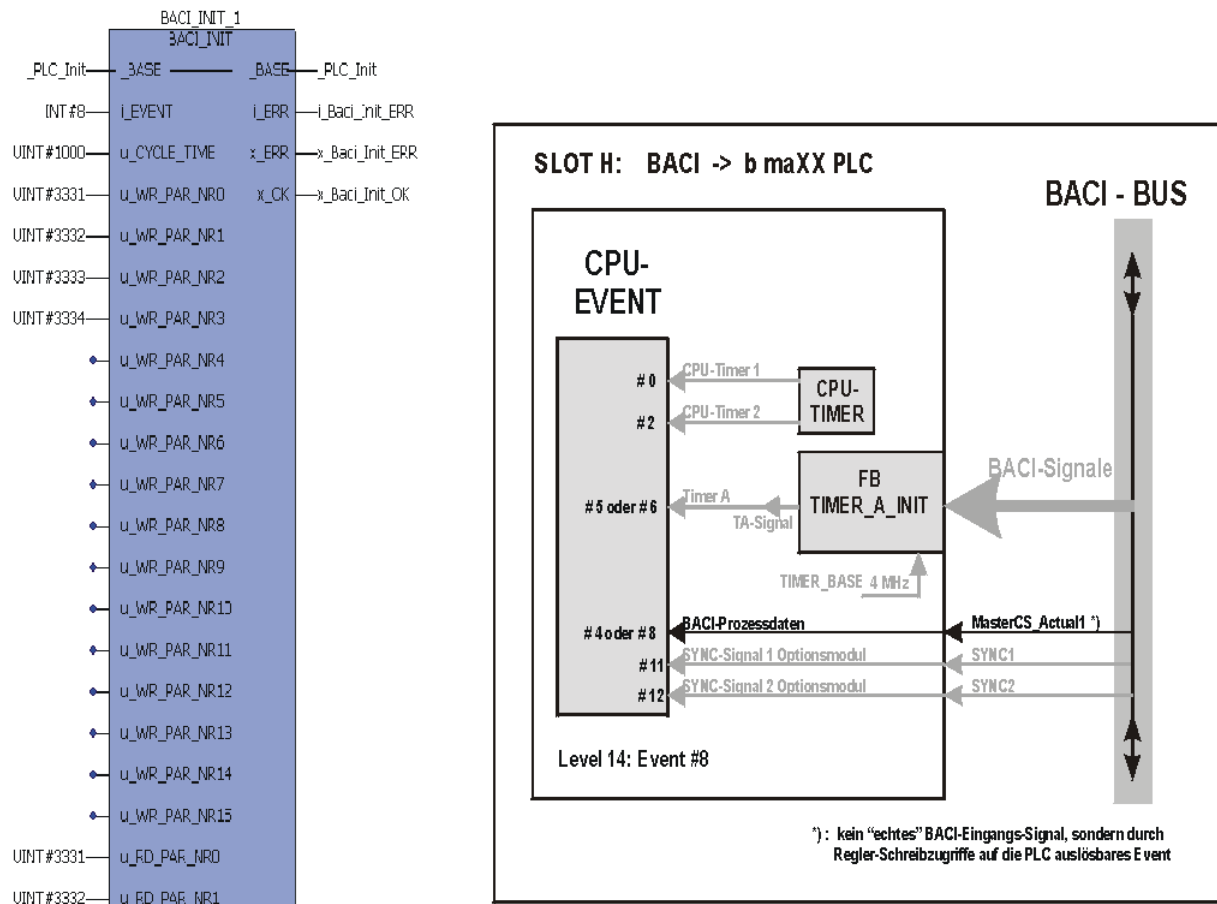


Abbildung 45: Prozessdatenkommunikation zwischen b maXX Regler und b maXX drive PLC über die BACI-Schnittstelle initialisieren

Der Bausteinaufruf erfolgt einmalig während der Initialisierungsphase (Kalt/Warmstart-Task).

Beschaltung:

Die globale Variable "_PLC_Init" ist bereits in der BM4_O_PLCO1 - Vorlage in PROPROGRAMM im globalen Variablenblatt "Global_Variables" entsprechend vordefiniert und ist Platzhalter für den eigenen DPRAM-Bereich der b maXX drive PLC (wird für die Regler-Kommunikation verwendet).

Mit u_CYCLE_TIME = UINT#1000 wird eine Zykluszeit von 1000 µs eingestellt.

Mit i_EVENT = 8 wird das Ereignis "BACI-Prozessdaten" und Interrupt-Level 14 (hohe Priorität) initialisiert.

Wirkung: Für die Prozessdaten-Kommunikation sind die Parameternummern 3331-3334 als Soll- und Istwerte eingerichtet. Jedesmal wenn der Regler das Ereignis "MasterCS_Actual1" (= Istwerte-Kanal des Reglers) bedient, wird das Interrupt-Ereignis „8“ ausgeführt (siehe Darstellung rechts neben dem "BACI_INIT"-Aufruf).

PROPROG wt II:

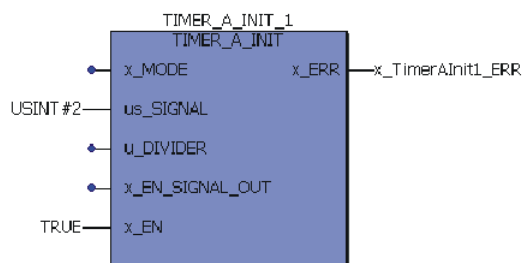
Der FB "BACI_INIT" ist in der Bibliothek SYSTEM1_PLC01_20bd00 oder höher abgelegt und verwendet die Bibliothek BM_TYPES_20bd03 oder höher und die Firmware-Bibliothek SYSTEM2_PLC01_20bd00 oder höher. Der Firmware-Baustein "INTR_SET" ist Bestandteil des Bausteins "BACI_INIT".

ProProg wt III:

Der FB "BACI_INIT" ist in der Bibliothek SYSTEM1_PLC01_30bd00 oder höher abgelegt und verwendet die Bibliothek BM_TYPES_30bd01 oder höher und die Firmware-Bibliothek SYSTEM2_PLC01_30bd00 oder höher. Der Firmware-Baustein "INTR_SET" ist Bestandteil des Bausteins "BACI_INIT".

(* TIMER_A_INIT: INPUT-Funktion

BACI-Signal "MasterCS_Actual1" --> TA-Signal *)



(* TIMER_A_INIT: OUTPUT-Funktion

TA-Signal --> BACI-Signal "SYNC1" *)

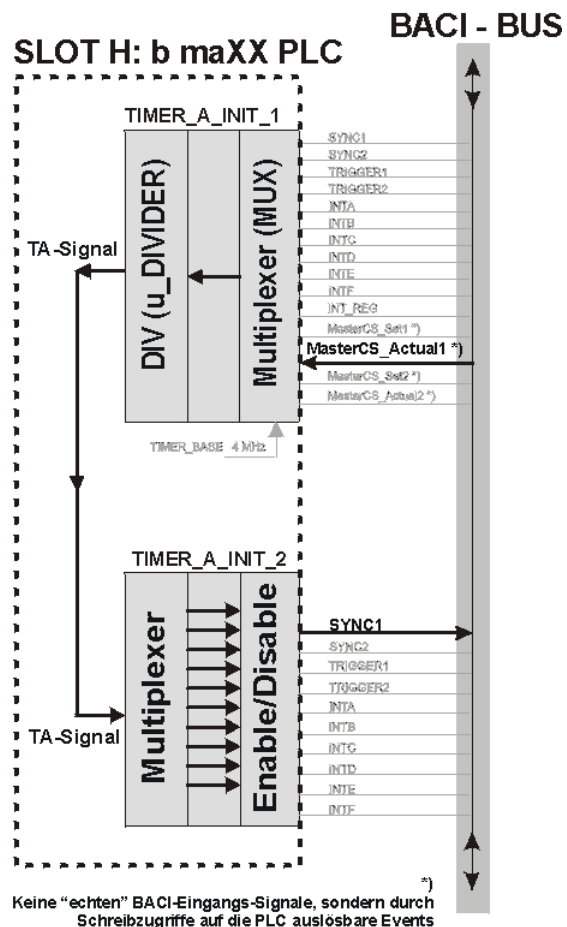
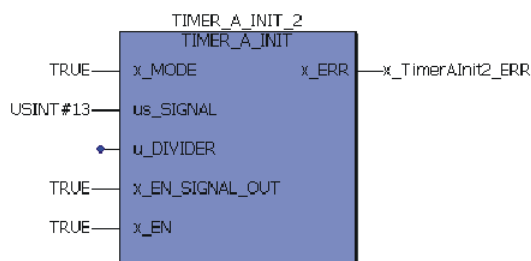


Abbildung 46: Gesamtfunktion der TIMER_A_INIT-Aufrufe: Jedesmal wenn der Regler das Ereignis "MasterCS_Actual1" (= Istwerte-Kanal des Reglers) bedient, wird von der PLC ein BACI-SYNC1-Signal für die Triggerung des Optionsmoduls BM4-O-IEI-01 erzeugt.

Die Bausteinaufrufe erfolgen einmalig während der Initialisierungsphase (Kalt/Warmstart-Task).

Rechts neben den Aufrufen ist dargestellt, wie die Signal-Weichen auf der PLC geschaltet werden, um die Signal-Umstellung über die PLC-Hardware zu erreichen.

Schritt 1 (Aufruf "TIMER_A_INIT_1"):

Beschaltung: x_MODE unbeschaltet (= FALSE) und us_SIGNAL = USINT#2:

4.6 BACI-Systembeschreibung für die b maXX drive PLC

Wirkung: Das BACI-Signal "MasterCS_Actual1" ungeteilt auf das PLC-interne TA-Signal geschaltet:

Schritt 2 (Aufruf "TIMER_A_INIT_2"):

Beschaltung: $x_MODE = TRUE$, $x_EN_SIGNAL_OUT = TRUE$ und $us_SIGNAL = USINT\#13$:

Wirkung: Das interne TA-Signal wird zum BACI-SYNC1-Signal verschaltet und auf die BACI ausgegeben.

Der FB "TIMER_A_INIT" ist in der Bibliothek SYSTEM1_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher abgelegt.

- Triggerung Optionsmodul BM4-O-IEI-01 über SYNC1-Signal: siehe Technische Beschreibung des Moduls BM4-O-IEI-01.

Prozessdatenkommunikation in der BYPASS-Task BACI-Prozessdaten:

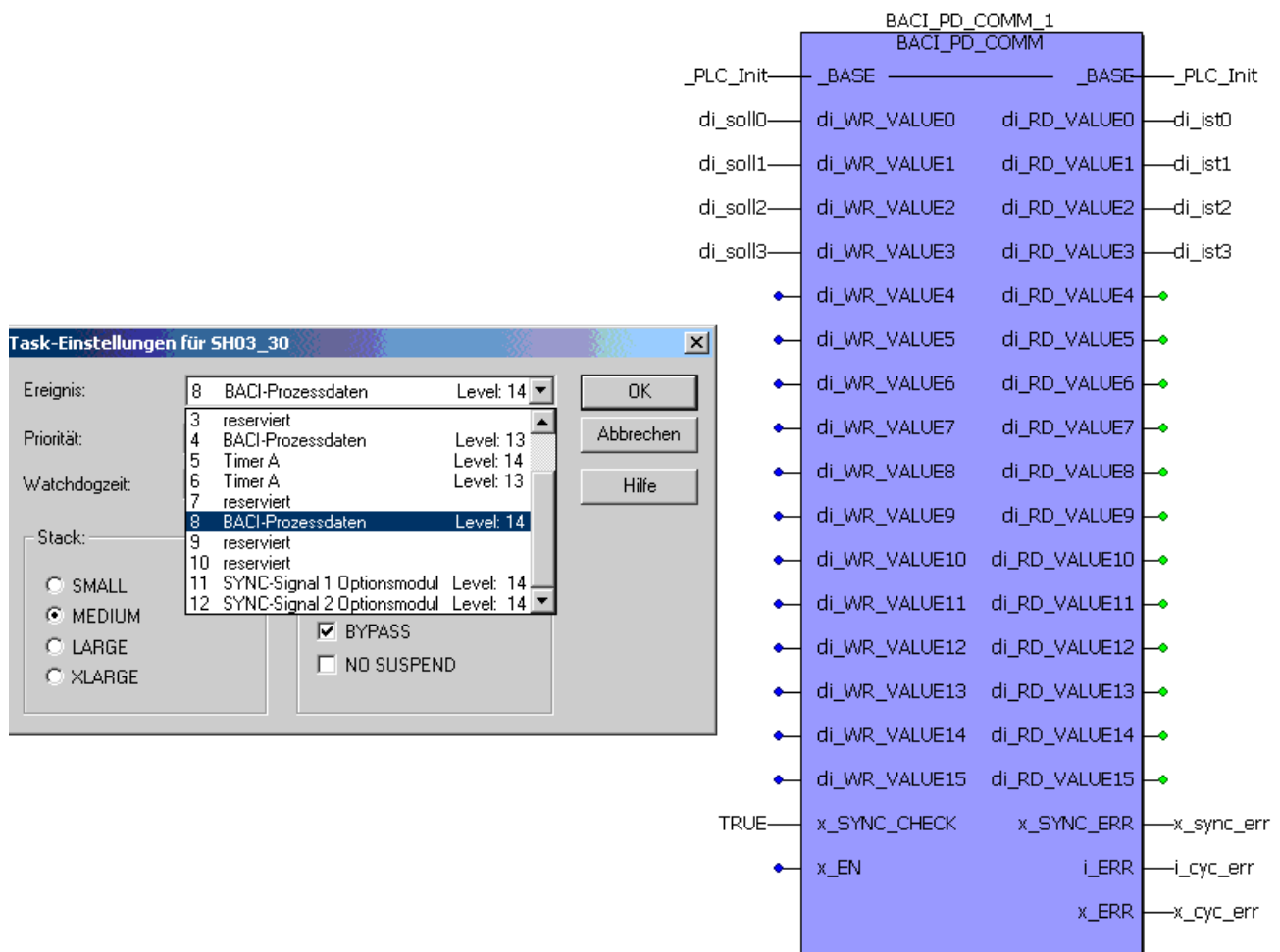


Abbildung 47: Durchführung der Prozessdaten-Kommunikation

Über den Baustein "BACI_PD_COMM" in einer POE, die dem Ereignis „8 Prozessdaten Level 14" zugewiesen wurde, werden nun zyklisch alle 1000 μs die Soll- und Istwerte der

eingestellten Parameter (in unserem Beispiel A sind das die Parameter 3331-3334) übertragen.

Der FB "BACI_PD_COMM" ist in der Bibliothek SYSTEM1_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher abgelegt.

4.6.8 Beispiel B: Umsetzung einer BACI-Prozessdatenkommunikation über ein selbst generiertes SYNC1-Signal

Umsetzung einer hochgenauen BACI-Prozessdatenkommunikation durch Generierung des SYNC1-Signals und gleichzeitige Triggerung eines Optionsmoduls BM4-O-IEI-01 sowie des b maXX Reglers über das SYNC1-Signal.

Der Timer A wird über den TIMER_A_INIT-Baustein mit 1 ms eingerichtet und auf das BACI-SYNC1-Signal geschaltet, damit sich der Regler und das Optionsmodul BM4-O-IEI-01 (und auch die PLC!) auf dieses Signal synchronisieren kann:

Mit dem SYNC1-Signal wird das Ereignis „11 SYNC-Signal 1 Optionsmodul Level 14" aktiviert und in der zugeordneten POE die Prozessdatenkommunikation zum b maXX Regler ausgeführt.

Kalt-/Warmstart-Task:

(* Timer_A_INIT: Timer-Funktion

Timer A mit 1 mSec => $u_DIVIDER = (Zeitperiode / 0,25 \mu s) - 1 = 3999$ *)

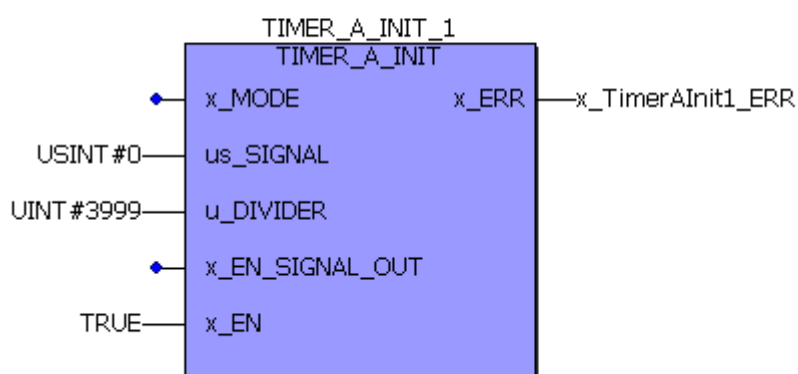


Abbildung 48: Timer A mit 1 ms Periode einrichten

Der Bausteinaufruf erfolgt einmalig während der Initialisierungsphase (Kalt/Warmstart-Task).

(* Timer_A_INIT: OUTPUT-Funktion
TA-Signal -> BACI-Signal "SYNC1" *)

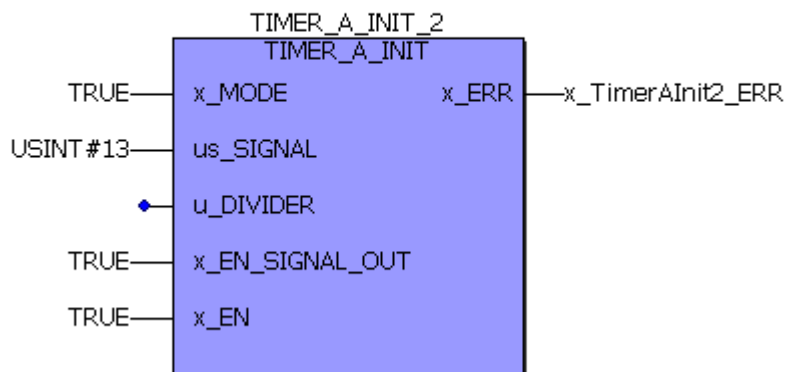


Abbildung 49: an SYNC1-BACI-Signal ausgeben:

Der Bausteinanruf erfolgt einmalig während der Initialisierungsphase (Kalt/Warmstart-Task).

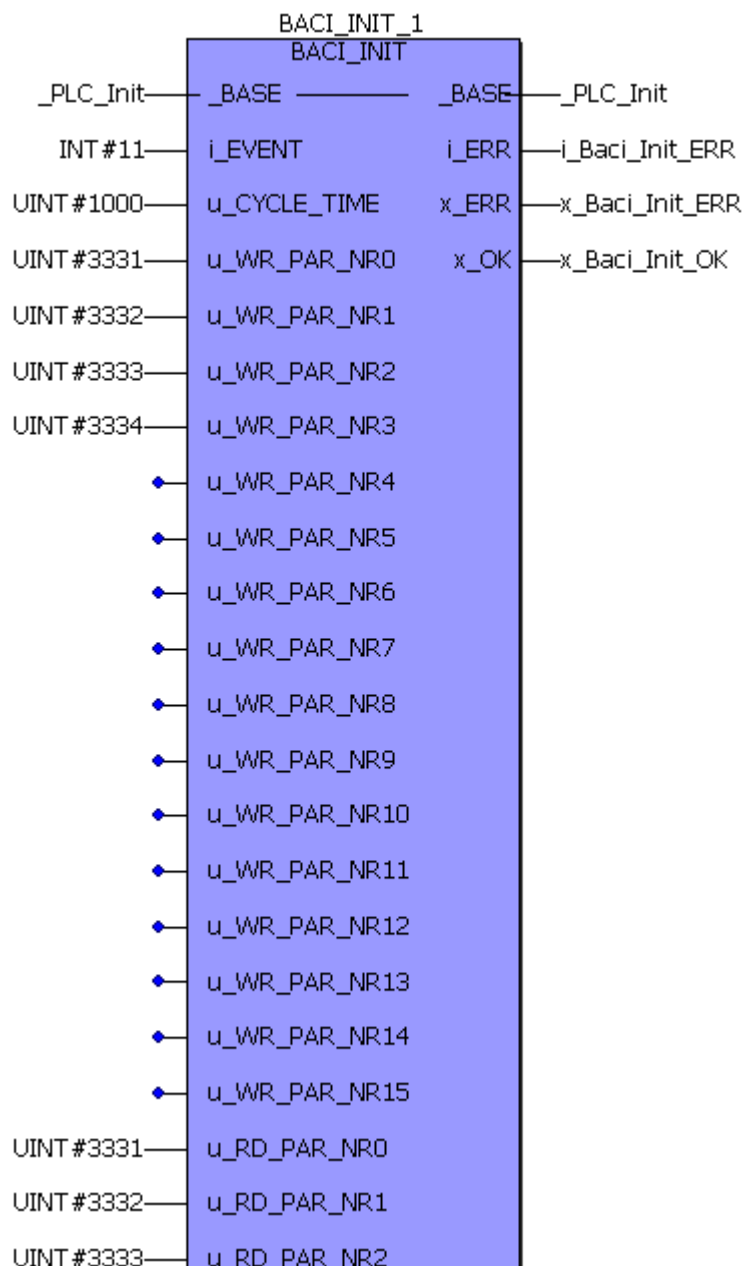


Abbildung 50: Prozessdatenkommunikation zum Regler über das SYNC1-Ereignis einrichten. Mit $u_CYCLE_TIME = \text{UINT}\#1000$ wird eine Zykluszeit von $1000\ \mu\text{s}$ eingestellt. Mit $i_EVENT = 11$ wird das Ereignis „SYNC-Signal 1 Optionsmodul Level 14 (hohe Priorität)“ initialisiert

Der Bausteinaufruf erfolgt einmalig während der Initialisierungsphase (Kalt/Warmstart-Task).

Prozessdatenkommunikation in der BYPASS-Task SYNC-Signal 1 Optionsmodul:

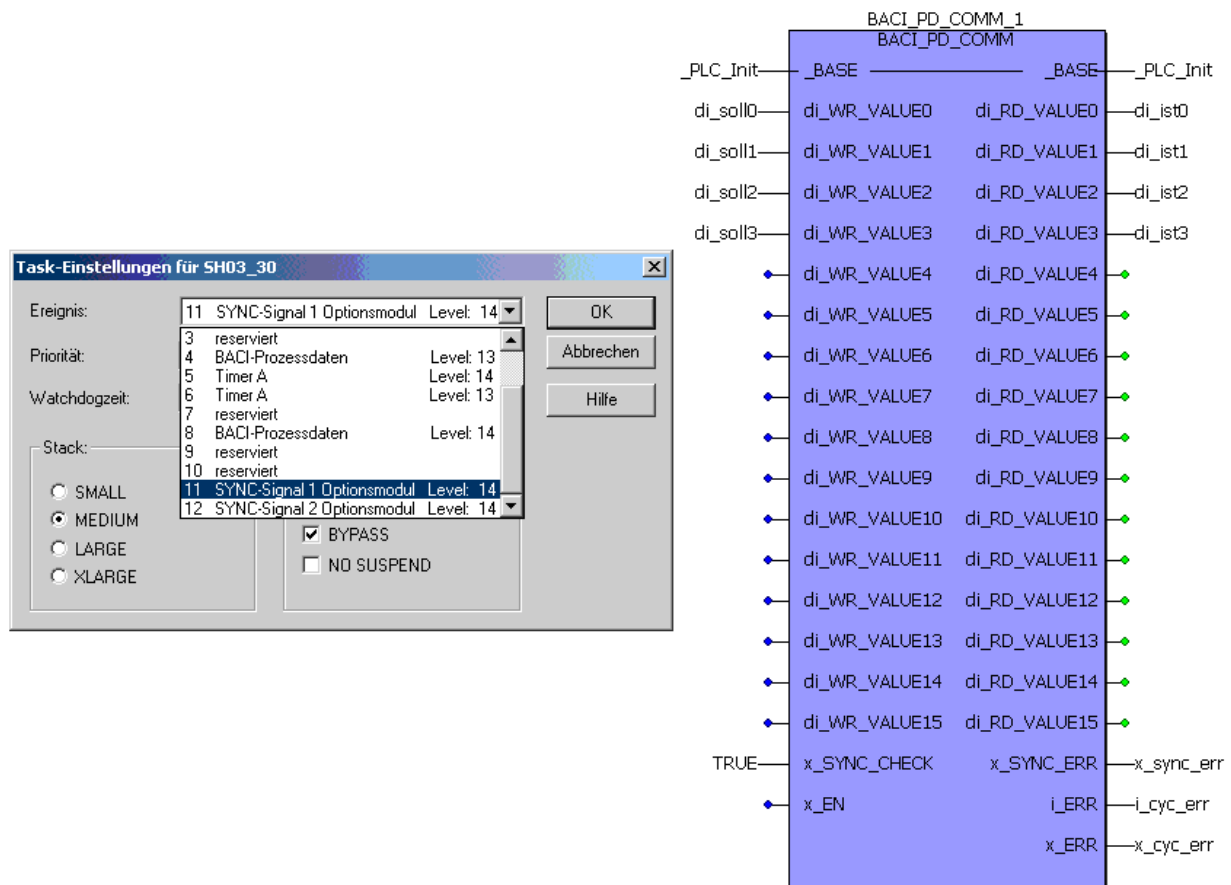


Abbildung 51: Durchführung der Prozessdaten-Kommunikation

Über den Baustein "BACI_PD_COMM" in einer POE, die dem Ereignis „11 SYNC-Signal 1 Optionsmodul Level 14" zugewiesen wurde, werden nun zyklisch alle 1000 µs die Soll- und Istwerte der eingestellten Parameter übertragen.

Der b maXX Regler muss ebenfalls auf das SYNC1-Signalquelle synchronisiert werden. Dazu notwendige b maXX Reglereinstellungen auf WinBASS II Seite "Synchronisation":

- Quelle für Sync-Signal:
"Sync 1 Signal von der BACI verwenden" (i_EVENT = INT#11)
- Sync Intervall 1 ms auswählen (=> Die gleiche Zeit-Periode muss am Eingang "u_CYCLE_TIME" in µs angeschlossen werden).
- Sync Offset: -25.0 µs.
- Sync Toleranz: 2.0 µs.

Um die Zugriffszeitpunkte von Regler und PLC auf die BACI gegeneinander zu verschieben, muss im Regler ein negativer Sync-Offset von -25 µs eingestellt werden. Dadurch ist die Bearbeitung des Istwerte-Kanals im Regler immer abgeschlossen, und aktuelle Istwerte vom Regler stehen der PLC zur Verfügung, wenn die PLC den vom BACI-Signal ausgelösten Interrupt bearbeitet.

-> Triggerung Optionsmodul BM4-O-IEI-01 über SYNC1-Signal: siehe Betriebsanleitung des Optionsmoduls BM4-O-IEI-01.

4.6.9 Beispiel C: Umsetzung einer BACI-Prozessdatenkommunikation über ein externes SYNC1-Signal

Umsetzung einer BACI-Prozessdatenkommunikation innerhalb einer CANsync-Event-Task und Triggerung eines Optionsmoduls BM4-O-IEI-01 sowie des b maXX Reglers über das Synchronisierungssignal SYNC1.

Jetzt wird das SYNC1-Signal nicht selbst generiert, sondern kommt von einem CANsync-Optionsmodul (BM4-O-CAN-05 für CAN-Slave bzw. BM4-O-CAN-06 für CAN-Master). Programmerstellung und Initialisierung der Optionsmodule: siehe die zugehörigen Betriebsanleitungen und Applikationshandbüchern.

Kalt-/Warmstart-Task:

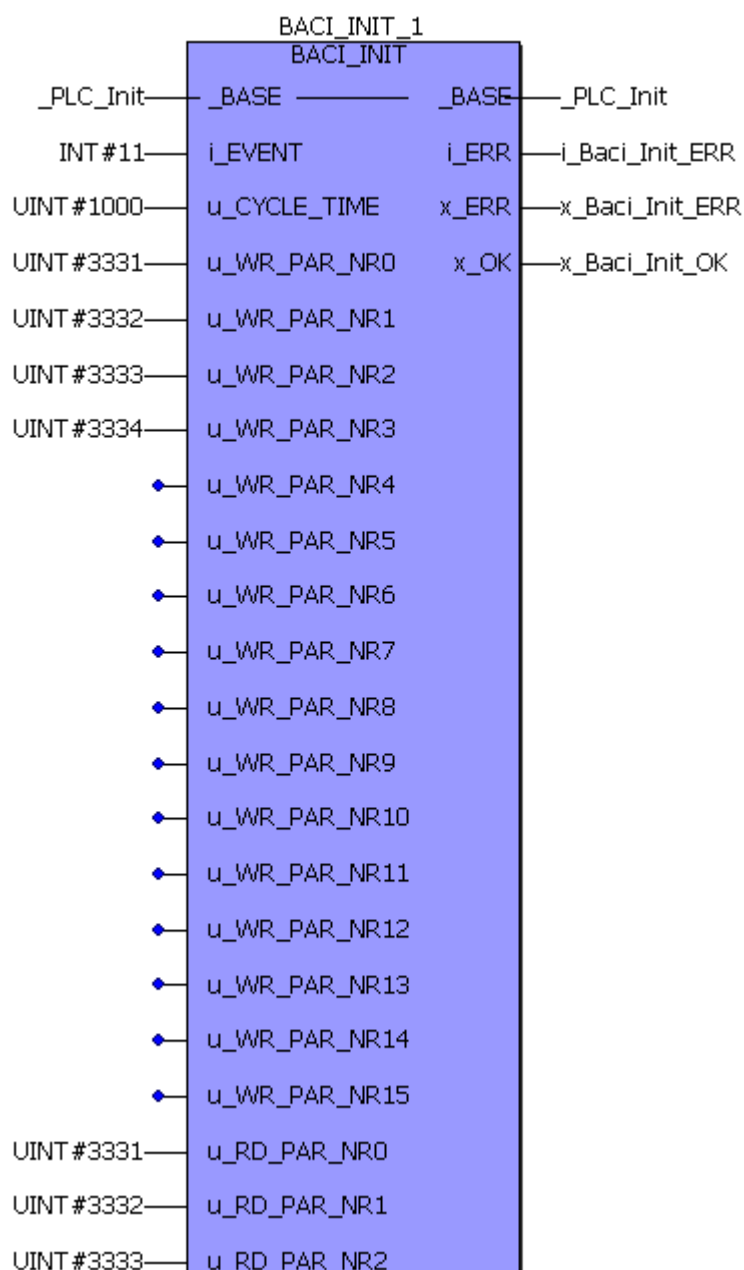


Abbildung 52: Prozessdatenkommunikation zum Regler über das SYNC1-Ereignis einrichten. Mit **u_CYCLE_TIME = UINT#1000** wird eine Zykluszeit von 1000 µs eingestellt. Mit **i_EVENT = 11** wird das Ereignis „SYNC-Signal 1 Optionsmodul Level 14 (hohe Priorität)“ initialisiert

Der Bausteinaufruf erfolgt einmalig während der Initialisierungsphase (Kalt/Warmstart-Task).

Prozessdatenkommunikation in der BYPASS-Task SYNC-Signal 1 Optionsmodul:

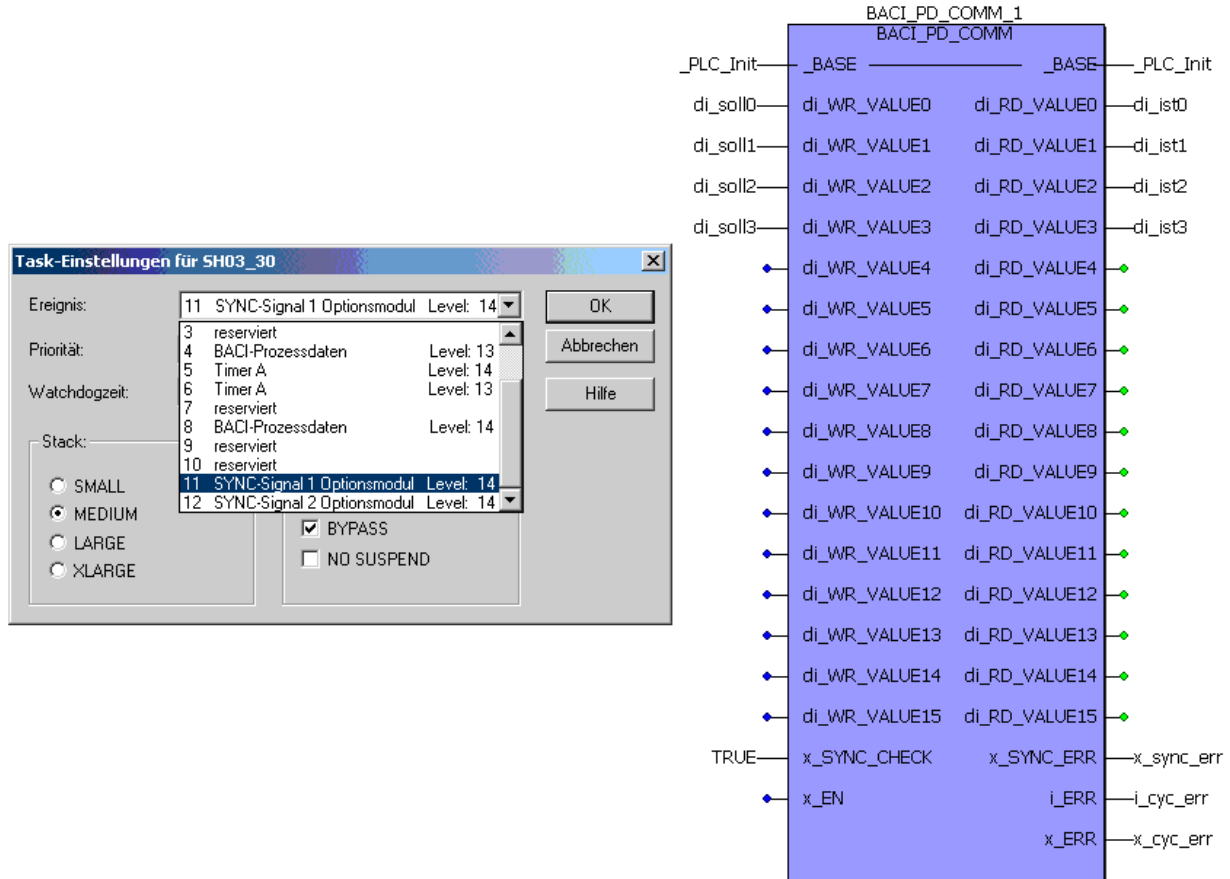


Abbildung 53: Durchführung der Prozessdaten-Kommunikation

Über den Baustein "BACI_PD_COMM" in einer POE, die dem Ereignis „11 SYNC-Signal 1 Optionsmodul Level 14" zugewiesen wurde, werden nun zyklisch alle 1000 µs die Soll- und Istwerte der eingestellten Parameter übertragen.

Der b maXX Regler muss ebenfalls auf das SYNC1-Signalquelle synchronisiert werden. Dazu notwendige b maXX Reglereinstellungen auf WinBASS II Seite "Synchronisation":

- Quelle für Sync-Signal:
"Sync 1 Signal von der BACI verwenden" (i_EVENT = INT#11)
- Sync Intervall 1 ms auswählen (=> Die gleiche Zeit-Periode muss am Eingang "u_CYCLE_TIME" in µs angeschlossen werden).
- Sync Offset: -25.0 µs.
- Sync Toleranz: 2.0 µs.

Um die Zugriffszeitpunkte von Regler und PLC auf die BACI gegeneinander zu verschieben, muss im Regler ein negativer Sync-Offset von -25 µs eingestellt werden. Dadurch ist die Bearbeitung des Istwerte-Kanals im Regler immer abgeschlossen, und aktuelle Istwerte vom Regler stehen der PLC zur Verfügung, wenn die PLC den vom BACI-Signal ausgelösten Interrupt bearbeitet.

-> Triggerung Optionsmodul BM4-O-IEI-01 über SYNC1-Signal: siehe Betriebsanleitung des Optionsmoduls BM4-O-IEI-01.

4.6.10 Beispiel D: Realisierung einer einfachen seriellen RS485-Kommunikation über einen CPU-Timer

Einrichten einer Timer-Event-Task für den zyklischen Aufruf, in der eine serielle RS485-Kommunikation (über die 9-pol. Sub-D-Buchse auf der b maXX drive PLC BM4-O-PLC-01 Schnittstelle X1) bearbeitet werden soll. Die Funktionsbausteine zur Kommunikation werden in einer POE platziert, die dieser Task zugeordnet sind. (Details zur Anschlussbelegung und Kabel der X1-Schnittstelle siehe Betriebsanleitung b maXX drive PLC BM4-O-PLC-01).

Kalt-/Warmstart-Task:

```

UINT #0—terminal_init[0]    (* protocol type: interface to ASCII-protocol *)
UINT #0—terminal_init[1]    (* protocol mode: RS485-Pull-Up-resistor on (activ)
                             RS485-terminator-resistors off (inactiv) *)
UINT #0—terminal_init[2]    (* bidirectional communication activated *)
UINT #10—terminal_init[3]   (* baudrate: 38400 Bd *)
UINT #2—terminal_init[4]    (* parity: even *)
UINT #2—terminal_init[5]    (* 8 databits *)
UINT #1—terminal_init[6]    (* 1 Stopbit *)

```

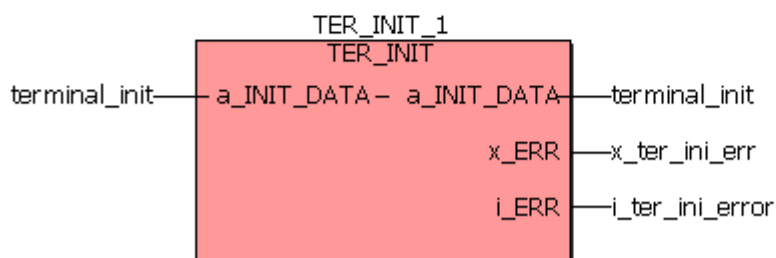


Abbildung 54: Initialisierung der seriellen Schnittstelle

Der Bausteinanruf von "TER_INIT" erfolgt einmalig während der Initialisierungsphase (Kalt/Warmstart-Task). (Details zu "TER_INIT" und den seriellen Schnittstellenparametern siehe [Beschreibung der Bausteine für die serielle RS485-Kommunikation über die Schnittstelle X1 \(9-pol. Sub-D-Buchse\) auf der b maXX drive PLC](#) ab Seite 87 oder On-line-Beschreibung in SYSTEM1_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher).

(* Event #2: CPU-Timer 2 Level 13;
i_PAR1=#200 => 10 mSec Interrupt intervall time *)

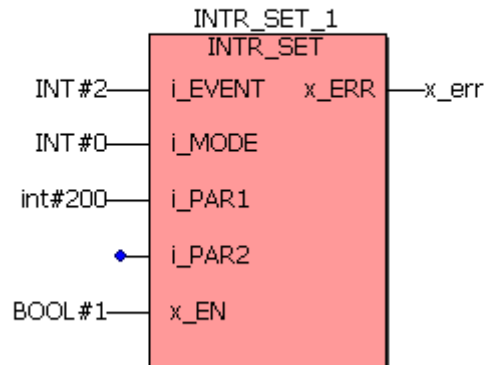


Abbildung 55: Einrichten eines 10 ms Bypass-Interrupts Level 13, in der die Sende- und Empfangsbausteine bearbeitet werden sollen:

Der Bausteinaufruf erfolgt einmalig während der Initialisierungsphase (Kalt/Warmstart-Task).

Serielle Kommunikation in der BYPASS-Task CPU-Timer 2:

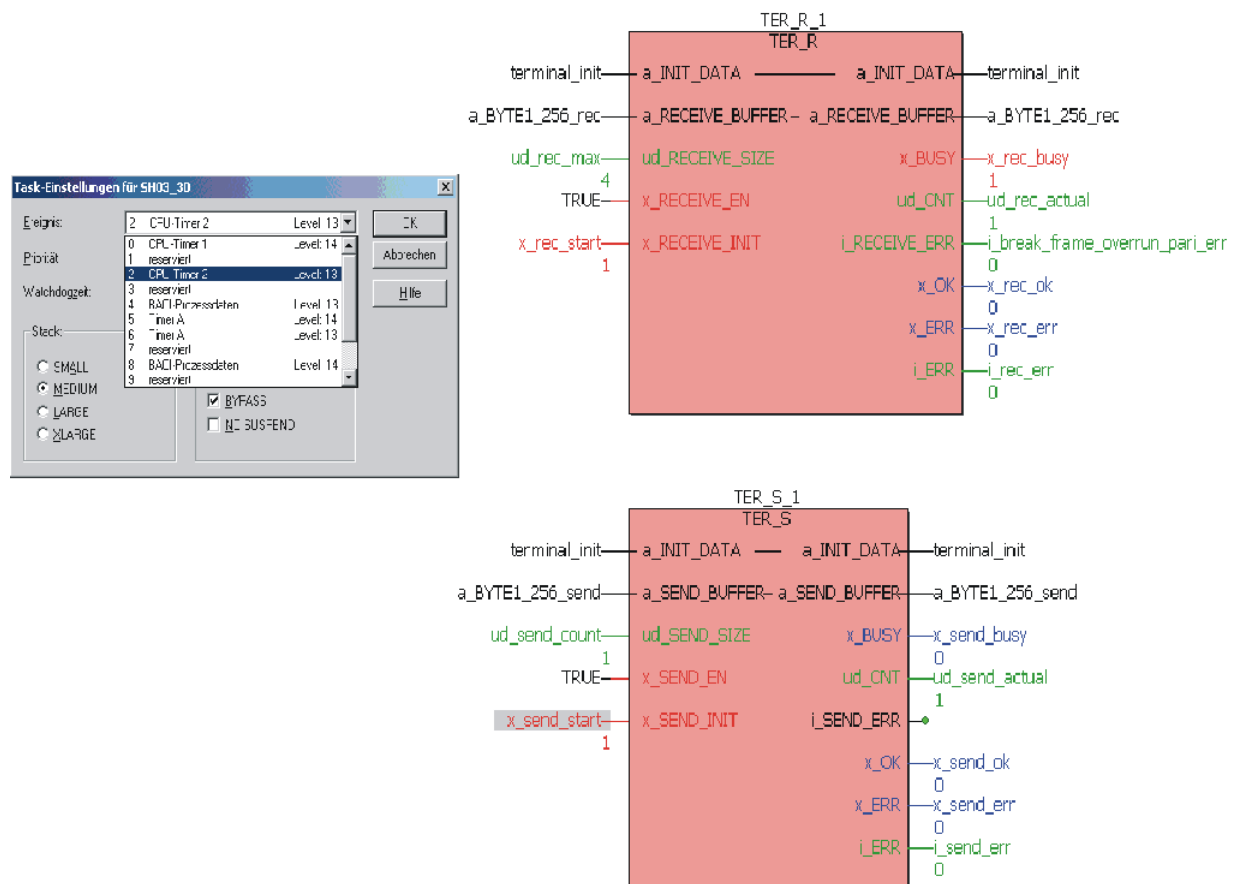


Abbildung 56: Aufruf der Sende- und Empfangsbausteine im Bypass-Interrupt (alle 10 ms)

Kurzbeschreibung

(Details zu den FW-Bausteinen siehe [►Beschreibung der Bausteine für die serielle RS485-Kommunikation über die Schnittstelle X1 \(9-pol. Sub-D-Buchse\) auf der b maXX drive PLC](#) ab Seite 87 oder Online-Beschreibungen in SYSTEM1_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher):

Empfangsbaustein "TER_R":

Über eine steigende Flanke an x_RECEIVE_INIT wird der Empfangsauftrag aktiviert. Bis nicht alle Zeichen empfangen worden sind (Anzahl an "ud_RECEIVE_SIZE"), bleibt x_BUSY auf TRUE, und am Ausgang "ud_CNT" werden die bisher empfangenen und im Array "a_BYTE1_256_rec" abgelegten Zeichen angezeigt.

Sendebaustein "TER_S":

Über eine steigende Flanke an x_SEND_INIT wird der Sendeauftrag aktiviert: Alle "ud_SEND_SIZE" Zeichen im Sendebuffer-Array "a_BYTE1_256_send" werden gesendet. Sobald der Sendeauftrag komplett ausgeführt worden ist, wird x_BUSY zurückgesetzt und x_OK gesetzt.

4.6.11 Beschreibung der Bausteine für die serielle RS485-Kommunikation über die Schnittstelle X1 (9-pol. Sub-D-Buchse) auf der b maXX drive PLC

Details zur Anschlussbelegung und Kabel der X1-Schnittstelle siehe Betriebsanleitung "b maXX drive PLC".



VORSICHT (CAUTION)

Folgendes **kann eintreffen**, wenn Sie diesen Gefahrenhinweis nicht beachten:

- Sachschaden

*Die Gefahr ist: **Kurzschluss** in PLC oder Beschädigung der am Pin 2 der Schnittstelle X1 angeschlossen Teilnehmer.*

Die + 5V an Pin 2 der Schnittstelle X1 an der BM4-O-PLC-01 SUB-D-Buchse ist nur für die Versorgung von externen Baumüller-eigenen RS485/RS232-Umsetzern vorgesehen und darf nicht kurzgeschlossen oder im Ring miteinander durchverbunden werden.

**HINWEIS**

zur RS485-Schnittstelle

Die RS485-Schnittstelle X1 ist galvanisch getrennt und optisch entkoppelt.

Die RS485 ist eine Erweiterung der RS422 für Mehrsenderbetrieb (= Kommunikation erfolgt wahlweise mit mehreren Sendern je Übertragungsweg, wobei zeitgleich immer nur max. ein Sender je Übertragungsweg aktiv sein darf.).

Üblicherweise werden die differentiellen Einzelleitungen miteinander verdreht, womit eingekoppelte elektromagnetische Störungen auf beide Einzelleitungen gleichmäßig wirken. Durch die Differenzauswertung am Empfangsbaustein können diese Störungen dann unterdrückt werden (Gleichtaktunterdrückung). Die Verdrehung steigert somit die Übertragungsqualität.

Empfehlung: Zusätzlich zur Zwei/Vierdrahtleitung kann auch die Masse-Leitung mitgeführt werden, womit beide Treiberbausteine (Sende-Treiber auf der einen Seite und Empfangs-Treiber auf der anderen Seite der Verbindung) das gleiche absolute Bezugspotential besitzen. Dadurch werden Überschreitungen des Gleichtakteingangsspannungsbereiches vermieden.

Die Masse-Leitung darf nirgends auf der gesamten Verbindungsstrecke mit dem Schirm verbunden werden!

- Konfigurationsmöglichkeit bei der b maXX drive PLC über die Software: Verwendung der auf der PLC integrierten Pull-Up/Down- und Abschlusswiderständen:

Beim Einsatz differentieller Schnittstellen, deren Sende-Treiber abgeschaltet werden können, besteht oftmals die Notwendigkeit, Pull-Up/Down-Widerstände und/oder Abschlusswiderstände den Verbindungskabeln zuzufügen.

Durch den Einsatz von Pull-Up/Down-Widerständen soll sichergestellt werden, dass an einer differentiellen Empfangseinheit stets ein definiertes Potential anliegt, auch wenn gerade keine Zeichen übertragen werden und die zugeordnete Sendeeinheit des gegenüberliegenden Partners seine Treiber hochohmig schaltet (protokollabhängig).

Fehlen solche Widerstände, so kann die Leitung floaten mit der Folge, dass sporadische Übertragungsfehler auftreten wie "Break Error", "Framing Error", "Parity Error" etc.

Besonders bei Verwendung von längeren Leitungen und höheren Übertragungsraten können die Kabelwellenwiderstände nicht mehr vernachlässigt werden. Deshalb werden die differentiellen Leitungen an den beiden Bus-Enden der Übertragungsstrecke dann mit Abschlusswiderständen abgeschlossen. Fehlen diese, können Reflexionen auftreten, welche zu Empfangsfehlern führen. Die Verwendung von Busabschlüssen verbessert somit die Übertragungsqualität.

Bei netzartigen Strukturen mit mehreren Teilnehmern auf einer Leitung sollte man jedoch darauf achten, dass auf dieser Leitung nicht mehrere Pull-Up/Down-Widerstände und nicht zuviele Abschlusswiderstände (höchstens die beiden an den Bus-Enden) parallel liegen.

Um die umständliche Prozedur "Pull-Up/Down- und Abschlusswiderstände direkt in den Stecker oder in das Kabel integrieren" zu verhindern, sind Pull-Up/Down- und Abschlusswiderstände in der b maXX drive PLC bereits vorhanden und können jeweils getrennt voneinander über die Applikations-Software zu- und abgeschaltet werden.

Die genaue Vorgehensweise dazu kann der FB-Beschreibung "TER_INIT" in der PROPROG wt Firmware-Bibliothek SYSTEM2_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher (siehe [►Funktionsbaustein TER_INIT](#) ab Seite 89) bzw. deren Online-Beschreibung entnommen werden.

- Konfigurationsmöglichkeit bei der b maXX drive PLC: Zweidraht / Vierdraht

Unterschied Zweidraht/Vierdraht:

Über eine Vierdrahtleitung (+ Masseleitung empfohlen) können Zeichen vollduplex gesendet werden. Vollduplex bedeutet immer, dass Zeichen über getrennte Kanäle zeitgleich in beiden Richtungen gesendet und empfangen werden können. Das 3964R®-Protokoll ist z. B. ein solches Vollduplex-Protokoll.

Im Unterschied dazu gibt es auch Protokolle im Halbduplexverfahren, die ihre Sende- und Empfangsaufträge über eine gemeinsame Zweidrahtleitung (+ Masseleitung empfohlen) abwickeln können. Bei einer Zweidrahtleitung sind an den beiden Enden des Verbindungskabels die TxD+ mit der RxD+-Leitung kurzgeschlossen, sowie die TxD- mit der RxD- Leitung. Dadurch brauchen im Gegensatz zum Vierdraht zwei Einzelleitungen weniger verlegt werden (=Materialersparnis).

Das Fehlen von den beiden Einzelleitungen muss aber durch einen exakten Protokollablauf "kompensiert" werden:

Das Protokoll muss zu jedem Zeitpunkt sicherstellen, dass Zeichen nie in beiden Richtungen gleichzeitig übertragen werden und dass immer nur maximal eine Sendeeinheit aktiviert ist und dabei die andere gegenüberliegende Sendeeinheit dann abgeschaltet (hochohmig) ist.

Hierzu gibt es die folgende Konfigurationsmöglichkeit bei der b maXX drive PLC:

Es kann zwischen Zweidraht-und Vierdrahtmodus ausgewählt werden.

Unterschied: Im Zweidraht-Modus erfolgt das Abschalten der Sendeeinheit, nachdem alle Zeichen gesendet wurden. Zu beachten ist dabei, dass durch das Abschalten die Verbindungsleitungen hochohmig werden, falls keine Pull-Up/Down-Widerstände programmiert wurden.

Die genaue Vorgehensweise für die Einstellmöglichkeiten kann der FB-Beschreibung "TER_INIT" in der PROPROG wt Standard-Bibliothek SYSTEM2_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher bzw. deren Online-Beschreibung entnommen werden.

4.6.12 Funktionsbaustein TER_INIT

Der Firmware-FB TER_INIT initialisiert die Terminal-Schnittstelle (serielle Schnittstelle) des Optionsmoduls b maXX drive PLC mit den notwendigen Schnittstellen-Parametern wie Baudrate, Parity usw...

Der FB TER_INIT ist in der FW-Bibliothek SYSTEM2_PLC_20bd00 (PROPROG wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher enthalten.

Parameter Eingang	Datentyp	Beschreibung
a_INIT_DATA	UINT_256_BMARRAY	Initialisierungs-Array für die Terminalschnittstelle

Parameter Ausgang	Datentyp	Beschreibung
a_INIT_DATA	UINT_256_BMARRAY	Initialisierungs-Array für die Terminalschnittstelle
x_ERR	BOOL	Fehlerbit
i_ERR	INT	Fehlerwort 0, 1, 2, 3, 4



HINWEIS

Es müssen alle Eingänge belegt werden, da sonst kein definierter Wert übergeben wird!

Typdefinition:

```
UINT_256_BMARRAY : ARRAY [0..255] OF UINT
```

Beschreibung:

Bevor eine Kommunikation über die Terminal-Schnittstelle (serielle Schnittstelle) des Optionsmoduls b maXX drive PLC erfolgen kann, muss diese parametrieren werden. Dies erfolgt über den Anschluss a_INIT_DATA. Die dort angeschlossene Variable enthält die notwendigen Schnittstellen-Parameter wie Baudrate, Parity usw... Die eigentliche Kommunikation erfolgt über entsprechende Kommunikations-FBs (siehe unten).

Da die Terminal-Schnittstelle nur einmal initialisiert wird, muss der Aufruf von TER_INIT in der Kalt- und in der Warmstart-Task erfolgen.

Eingang a_INIT_DATA:

Am Eingang a_INIT_DATA wird eine Variable vom Typ UINT_256_BMARRAY angeschlossen. Die einzelnen Einträge von a_INIT_DATA haben folgende Bedeutung:

a_INIT_DATA[Index]	Bedeutung
a_INIT_DATA[0]	Protokoll 0: beliebiges ASCII-Protokoll 1: Anschaltung an Prozedur 3964R [®] auf Datenbausteine
a_INIT_DATA[1]	Betriebsart des Protokolls 0: nur RS485-Pull-Up-Widerstände sind aktiv (on) 1: nur RS485-Abschlusswiderstand ist aktiv (on) 2: RS485-Pull-Up-Widerstände und RS485-Abschlusswiderstand sind aktiv (on) 3: RS485-Pull-Up-Widerstände und RS485-Abschlusswiderstand sind nicht aktiv (off)
a_INIT_DATA[2]	Betriebsart RS485 0: Vierdraht-Betrieb RS485 1: Zweidraht-Betrieb RS485 ¹⁾

a_INIT_DATA[Index]	Bedeutung
a_INIT_DATA[3]	Baudrate 0: 50 Bit/s ¹⁾ 1: 110 Bit/s ¹⁾ 2: 150 Bit/s ¹⁾ 3: 300 Bit/s ¹⁾ 4: 600 Bit/s ¹⁾ 5: 1200 Bit/s ¹⁾ 6: 2400 Bit/s ¹⁾ 7: 4800 Bit/s 8: 9600 Bit/s 9: 19200 Bit/s 10: 38400 Bit/s
a_INIT_DATA[4]	Parity 0: kein Parity-Bit 1: ungerade (odd) 2: gerade (even)
a_INIT_DATA[5]	Zeichen-Länge 0: Reserviert 1: 7 Bit je Zeichen 2: 8 Bit je Zeichen
a_INIT_DATA[6]	Stopp-Bit 0: 1 Stopp-Bit 1: 1 Stopp-Bit 2: 1 Stopp-Bit 3: 2 Stopp-Bit
a_INIT_DATA[7]	Reserviert
a_INIT_DATA[8]	max. Anzahl der Kommunikationsversuche je Telegramm ²⁾
a_INIT_DATA[9]	Priorität ²⁾ 0: niedrige Priorität 1: hohe Priorität
a_INIT_DATA[10]	Zeichenverzugszeit (ZVZ) in ms ²⁾
a_INIT_DATA[11]	Quittungsverzugszeit (QVZ) in ms ²⁾
a_INIT_DATA[12]	Blockverzugszeit (BVZ) in ms ²⁾
a_INIT_DATA[13]	Reserviert
...	...
a_INIT_DATA[255]	Reserviert

¹⁾ nicht für Anschaltung an Prozedur 3964R[®] auf Datenbausteine

²⁾ nur für Anschaltung an Prozedur 3964R[®] auf Datenbausteine

Die Prozedur 3964R[®] ist ein eingetragenes Warenzeichen der Firma Siemens AG.

a) ASCII-Protokoll

Die Kommunikation mit dem Protokoll "beliebiges ASCII-Protokoll" erfolgt über die FBs TER_S und TER_R. Der Anwender kann damit n Zeichen an einen Kommunikations-Partner senden und n Zeichen von einem Kommunikations-Partner empfangen.

4.6 BACI-Systembeschreibung für die b maXX drive PLC

Die FBs TER_S und TER_R sind in der FW-Bibliothek SYSTEM2_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher enthalten.

Typische Initialisierungswerte für das ASCII-Protokoll sind:

a_INIT_DATA[Index]	Bedeutung
a_INIT_DATA[0]	Protokoll 0: beliebiges ASCII-Protokoll
a_INIT_DATA[1]	Betriebsart des Protokolls 0: nur RS485-Pull-Up-Widerstände sind aktiv (on) 1: nur RS485-Abschlusswiderstand ist aktiv (on) 2: RS485-Pull-Up-Widerstände und RS485-Abschlusswiderstand sind aktiv (on) 3: RS485-Pull-Up-Widerstände und RS485-Abschlusswiderstand sind nicht aktiv (off)
a_INIT_DATA[2]	Betriebsart RS485 0: Vierdraht-Betrieb RS485 1: Zweidraht-Betrieb RS485
a_INIT_DATA[3]	Baudrate 0: 50 Bit/s 1: 110 Bit/s 2: 150 Bit/s 3: 300 Bit/s 4: 600 Bit/s 5: 1200 Bit/s 6: 2400 Bit/s 7: 4800 Bit/s 8: 9600 Bit/s 9: 19200 Bit/s 10: 38400 Bit/s
a_INIT_DATA[4]	Parity 0: kein Parity-Bit 1: ungerade (odd) 2: gerade (even)
a_INIT_DATA[5]	Zeichen-Länge 1: 7 Bit je Zeichen 2: 8 Bit je Zeichen
a_INIT_DATA[6]	Stopp-Bit 0: 1 Stopp-Bit 1: 1 Stopp-Bit 2: 1 Stopp-Bit 3: 2 Stopp-Bit
a_INIT_DATA[7]	Reserviert 0
...	...
a_INIT_DATA[255]	Reserviert 0

b) Anschaltung an Prozedur 3964R®

Die Anschaltung an die Prozedur 3964R® auf Datenbausteine erfolgt über die Firmware-FBs T64_RSYN und T64_REC (sind in der FW-Bibliothek SYSTEM2_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher enthalten).

Typische Initialisierungswerte für die Anschaltung an die Prozedur 3964R® auf Datenbausteine sind:

a_INIT_DATA[Index]	Bedeutung
a_INIT_DATA[0]	Protokoll 1: Anschaltung an Prozedur 3964R® auf Datenbausteine
a_INIT_DATA[1]	Betriebsart des Protokolls 0: nur RS485-Pull-Up-Widerstände sind aktiv (on)
a_INIT_DATA[2]	Betriebsart RS485 0: Vierdraht-Betrieb RS485
a_INIT_DATA[3]	Baudrate (Baud=Bit/s) 7: 4800 Bit/s 8: 9600 Bit/s 9: 19200 Bit/s
a_INIT_DATA[4]	Parity 2: gerade (even)
a_INIT_DATA[5]	Zeichen-Länge 2: 8 Bit je Zeichen
a_INIT_DATA[6]	Stopp-Bit 1: 1 Stopp-Bit
a_INIT_DATA[7]	Reserviert 0
a_INIT_DATA[8]	max. Anzahl der Kommunikationsversuche je Telegramm 2: max. 2 Kommunikationsversuche je Telegramm
a_INIT_DATA[9]	Priorität 0: niedrige Priorität
a_INIT_DATA[10]	Zeichenverzugszeit (ZVZ) in ms 100
a_INIT_DATA[11]	Quittungsverzugszeit (QVZ) in ms 500
a_INIT_DATA[12]	Blockverzugszeit (BVZ) in ms 2000
a_INIT_DATA[13]	Reserviert 0
...	...
a_INIT_DATA[255]	Reserviert 0

Fehlerrauswertung:

Wird eine falsche Parametrierung festgestellt, wird das Fehlerbit x_ERR gesetzt und der Fehler über die Fehlernummer i_ERR genauer spezifiziert:

i_ERR	Fehler serielle Schnittstelle
0	kein Fehler
1 bis 3	Reserviert
4	eingestellte Baudrate nicht möglich

4.6.13 Funktionsbaustein TER_R

Der Firmware-FB TER_R empfängt Zeichen über die Terminal-Schnittstelle (serielle Schnittstelle) des Optionsmoduls b maXX drive PLC und legt die Zeichen in einem Array ab.

Der FB TER_R ist in der FW-Bibliothek SYSTEM2_PLC01_20bd00 (PROPROP wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher enthalten.

Parameter Eingang	Datentyp	Beschreibung
a_INIT_DATA	UINT_256_BMARRAY	Initialisierungs-Array für die Terminalschnittstelle
a_RECEIVE_BUFFER	ANY ¹⁾	Array in das eingelesen wird
ud_RECEIVE_SIZE	UDINT	Anzahl der Bytes, die empfangen werden soll
x_RECEIVE_EN	BOOL	Schnittstellen-Empfangs-Interrupt sperren/freigeben
x_RECEIVE_INIT	BOOL	Empfangs-Auftrag absetzen

Parameter Ausgang	Datentyp	Beschreibung
a_INIT_DATA	UINT_256_BMARRAY	Initialisierungs-Array für die Terminalschnittstelle
a_RECEIVE_BUFFER	ANY ¹⁾	Array in das eingelesen wird
x_BUSY	BOOL	zeigt an, ob Auftrag aktiv ist
ud_CNT	UDINT	Anzahl der bereits empfangenen Zeichen
i_RECEIVE_ERR	INT	Anzeige serieller Empfangs-Fehler
x_OK	BOOL	zeigt erfolgreich aktivierten Auftrag an
x_ERR	BOOL	Fehlerbit
i_ERR	INT	Fehlernummer

¹⁾ Es ist ein Array von der Größe 256 Byte anzuschließen. Es eignen sich z. B. der Datentyp BYTE_256_BMARRAY (: ARRAY [0..255] OF BYTE) oder der Datentyp DINT_64_BMARRAY (: ARRAY [0..63] OF DINT) aus der Bibliothek BM_TYPES_20bd03 (PROPROP wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher.

**HINWEIS**

Es müssen alle Eingänge belegt werden, da sonst kein definierter Wert übergeben wird!

Typdefinition:

```
UINT_256_BMARRAY : ARRAY [0..255] OF UINT
```

Beschreibung:

Der Firmware-FB TER_R muss zyklisch im Anwenderprogramm aufgerufen werden.

Für die Kommunikation mit den Firmware-FBs TER_S und TER_R sind die Hardware-spezifischen Schnittstellen-Parameter am Firmware-FB TER_INIT einzustellen (siehe Beschreibung Firmware-FB TER_INIT).

Mit dem Protokoll "beliebiges ASCII-Protokoll" kann der Anwender mit dem Firmware-FB TER_S n Zeichen an einen Kommunikations-Partner senden. Mit dem Firmware-FB TER_R kann der Anwender n Zeichen von einem Kommunikations-Partner empfangen.

Ein-/Ausgang a_INIT_DATA:

An a_INIT_DATA wird das parametrisierte Initialisierungs-Array für die Terminal-Schnittstelle angeschlossen (siehe Beschreibung Firmware-FB TER_INIT). D. h. an a_INIT_DATA ist die gleiche globale Variable anzuschließen wie am Firmware-FB TER_INIT.

Ein-/Ausgang a_RECEIVE_BUFFER:

An a_RECEIVE_BUFFER wird ein Array (256 Byte, siehe oben) angeschlossen. Die empfangenen (und im systeminternen Terminal-Schnittstellen-Empfangsbuffer eingetragenen) Daten/Zeichen werden abgeholt und in diesem Array abgelegt.

Eingang ud_RECEIVE_SIZE:

Am Eingang ud_SEND_SIZE wird die Anzahl der Zeichen angegeben, die vom systeminternen Empfangsbuffer der Terminal-Schnittstelle in das Array am Ein-/Ausgang a_RECEIVE_BUFFER abgelegt werden.

Eingang x_RECEIVE_EN:

Der Terminal-Schnittstellen-Empfangs-Interrupt kann bei Aktivierung eines Auftrags (steigende Flanke an x_RECEIVE_INIT) entweder freigegeben (x_RECEIVE_EN = TRUE) oder gesperrt (x_RECEIVE_EN = FALSE) werden. Ist x_RECEIVE_EN = FALSE werden im systeminternen Terminal-Schnittstellen-Empfangsbuffer alle bisher empfangenen Zeichen (unabhängig vom Zustand des Eingangs x_RECEIVE_INIT) gelöscht. Bei gesperrtem Interrupt erfolgt kein Zeichenempfang in dem systeminternen Empfangsbuffer mehr.

Eingang **x_RECEIVE_INIT**:

Durch eine steigende Flanke am Eingang **x_RECEIVE_INIT** wird ein neuer Auftrag zum Empfang von "**ud_RECEIVE_SIZE**" Zeichen (aus dem systeminternen Terminal-Schnittstellen-Empfangsbuffer) aktiviert.

Ausgang **x_BUSY**:

Der Ausgang **x_BUSY** ist TRUE solange ein Auftrag ausgeführt wird. Wurden alle Zeichen aus dem systeminternen Terminal-Schnittstellen-Empfangsbuffer abgeholt und in das Array am Ein-Ausgang **a_RECEIVE_BUFFER** eingetragen, ist der Auftrag erfolgreich ausgeführt, und **x_BUSY** wird FALSE.

Ausgang **ud_CNT**:

Am Ausgang **ud_CNT** wird die Anzahl der bis zu diesem Zeitpunkt (aus dem Terminal-Schnittstellen-Empfangsbuffer) empfangenen Zeichen angezeigt.

Ausgang **x_OK**:

Der Ausgang **x_OK** zeigt mit **x_OK** = TRUE einen Zyklus lang an, ob ein neuer Auftrag (steigende Flanke am Eingang **x_RECEIVE_INIT**) übernommen wurde. Ein Auftrag wird auch dann übernommen, wenn gerade ein Empfangsauftrag aktiv (**x_BUSY** = TRUE) ist.

D. h. der laufende Auftrag wird sofort abgebrochen und der neue aktiviert.

Fehlerauswertung:

Der Ausgang **i_RECEIVE_ERR** zeigt Empfangs-Fehler an.

i_RECEIVE_ERR	Fehler serielle Schnittstelle
0	kein Fehler
1	BREAK-Fehler
2	FRAME-Fehler
3	PARITY-Fehler
4	OVERRUN-Fehler
5	Interner Empfangsfehler RxRDY

Ausgänge **x_ERR**, **i_ERR**:

Wird eine falsche Parametrierung festgestellt, wird der Ausgang **x_ERR** = TRUE gesetzt und der Fehler über die Fehlernummer (Ausgang **i_ERR**) genauer spezifiziert:

i_ERR	Fehler serielle Schnittstelle
0	kein Fehler
1 bis 2	Reserviert
3	Terminal-Schnittstelle wurde nicht initialisiert. Bitte überprüfen Sie den Aufruf von TER_INIT in der Kalt- und in der Warmstart-Task sowie alle a_INIT_DATA -Anschlüsse

4.6.14 Funktionsbaustein TER_S

Der Firmware-FB TER_S sendet Zeichen aus einem Array über die Terminal-Schnittstelle (serielle Schnittstelle) des Optionsmoduls b maXX drive PLC an einen Kommunikations-Partner.

Der FB TER_S ist in der FW-Bibliothek SYSTEM2_PLC01_20bd00 (PROPROP wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher enthalten.

Parameter Eingang	Datentyp	Beschreibung
a_INIT_DATA	UINT_256_BMARRAY	Initialisierungs-Array für die Terminalschnittstelle
a_SEND_BUFFER	ANY ¹⁾	Array aus dem gesendet wird
ud_SEND_SIZE	UDINT	Anzahl der Bytes, die gesendet werden soll
x_SEND_EN	BOOL	Schnittstellen-Sende-Interrupt sperren/freigeben
x_SEND_INIT	BOOL	Sende-Auftrag absetzen

Parameter Ausgang	Datentyp	Beschreibung
a_INIT_DATA	UINT_256_BMARRAY	Initialisierungs-Array für die Terminalschnittstelle
a_SEND_BUFFER	ANY ¹⁾	Array aus dem gesendet wird
x_BUSY	BOOL	zeigt an, ob Auftrag aktiv ist
ud_CNT	UDINT	Anzahl der bereits gesendeten Zeichen
i_SEND_ERR	INT	Anzeige serieller Sende-Fehler (zur Zeit nicht benutzt, also immer 0)
x_OK	BOOL	zeigt erfolgreich aktivierten Auftrag an
x_ERR	BOOL	Fehlerbit
i_ERR	INT	Fehlernummer

- ¹⁾ Es ist ein Array von der Größe 256 Byte anzuschließen. Es eignen sich z. B. der Datentyp BYTE_256_BMARRAY (: ARRAY [0..255] OF BYTE) oder der Datentyp DINT_64_BMARRAY (: ARRAY [0..63] OF DINT) aus der Bibliothek BM_TYPES_20bd03 (PROPROP wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher.



HINWEIS

Es müssen alle Eingänge belegt werden, da sonst kein definierter Wert übergeben wird!

Typdefinition:

```
UINT_256_BMARRAY : ARRAY [0..255] OF UINT
```

Beschreibung:

Der Firmware-FB TER_S muss zyklisch im Anwenderprogramm aufgerufen werden.

Für die Kommunikation mit den Firmware-FBs TER_S und TER_R sind die Hardware-spezifischen Schnittstellen-Parameter am Firmware-FB TER_INIT einzustellen (siehe Beschreibung Firmware-FB TER_INIT).

Mit dem Protokoll "beliebiges ASCII-Protokoll" kann der Anwender mit dem Firmware-FB TER_S n Zeichen an einen Kommunikations-Partner senden. Mit dem Firmware-FB TER_R kann der Anwender n Zeichen von einem Kommunikations-Partner empfangen.

Ein-/Ausgang **a_INIT_DATA**:

An a_INIT_DATA wird das parametrierte Initialisierungs-Array für die Terminal-Schnittstelle angeschlossen (siehe Beschreibung Firmware-FB TER_INIT). D. h. an a_INIT_DATA ist die gleiche globale Variable anzuschließen wie am Firmware-FB TER_INIT.

Ein-/Ausgang **a_SEND_BUFFER**:

An a_SEND_BUFFER wird ein Array (256 Byte, siehe oben) angeschlossen. In diesem Array werden vom Anwender die zu sendenden Daten/Zeichen abgelegt.

Eingang **ud_SEND_SIZE**:

Die Anzahl der zu sendenden Zeichen wird an ud_SEND_SIZE angegeben.

Eingang **x_SEND_EN**:

Der Schnittstellen-Sende-Interrupt kann bei Aktivierung eines Auftrags (steigende Flanke an x_SEND_INIT) entweder freigegeben (x_SEND_EN = TRUE) oder gesperrt (x_SEND_EN = FALSE) werden. Bei gesperrtem Interrupt erfolgt keine Bearbeitung eines gerade laufenden oder neu ausgelösten Sendeauftrags mehr. Wenn die Betriebsart RS485 auf "Zweidraht" eingestellt ist (siehe TER_INIT), wird die RS485-Sende-Schnittstelle zusätzlich physikalisch gesperrt (= hochohmiger Zustand des Sende-Ausgangs für Mehr-Punkt-Verbindungen).

Eingang **x_SEND_INIT**:

Durch eine steigende Flanke am Eingang x_SEND_INIT wird ein neuer Auftrag zum Senden von "ud_SEND_SIZE" Zeichen aktiviert. Die RS485-Sende-Schnittstelle wird physikalisch freigegeben.

Ausgang **x_BUSY**:

Der Ausgang x_BUSY ist TRUE solange ein Auftrag ausgeführt wird. Wurden alle Zeichen aus dem Array a_SEND_BUFFER gesendet, ist der Auftrag erfolgreich ausgeführt, und x_BUSY wird FALSE.

Ausgang **ud_CNT**:

Am Ausgang ud_CNT wird die Anzahl der bis zu diesem Zeitpunkt gesendeten Zeichen angezeigt.

Ausgang i_SEND_ERR:

Dieser Ausgang zeigt mit i_SEND_ERR = INT#0 an, dass kein Sendefehler aufgetreten ist. Andere Werte von i_SEND_ERR sind reserviert.

Ausgang x_OK:

Der Ausgang x_OK zeigt mit x_OK = TRUE einen Zyklus lang an, ob ein neuer Auftrag (steigende Flanke am Eingang x_SEND_INIT) übernommen wurde. Ein Auftrag kann erst dann übernommen werden, wenn der zuvor laufende Sendevorgang bereits vollkommen abgeschlossen ist (x_BUSY = FALSE) oder vom Anwender aktiv abgebrochen wurde (x_SEND_EN = FALSE). Solange der Ausgang x_BUSY = TRUE ist, werden Sendeaufträge ignoriert (x_OK bleibt FALSE).

Fehlerauswertung:

Wird eine falsche Parametrierung festgestellt, wird der Ausgang x_ERR = TRUE gesetzt und der Fehler über die Fehlernummer am Ausgang i_ERR genauer spezifiziert:

i_ERR	Fehler serielle Schnittstelle
0	kein Fehler
1 bis 2	Reserviert
3	Terminal-Schnittstelle wurde nicht initialisiert. Bitte überprüfen Sie den Aufruf von TER_INIT in der Kalt- und in der Warmstart-Task sowie alle a_INIT_DATA-Anschlüsse

4.6.15 Funktionsbaustein T64_RSYN

Der Firmware-FB T64_RSYN bildet einen Datenbaustein einer Anschaltung an die Prozedur 3964R® auf ein Array ab. Die eigentliche Kommunikationsabwicklung erfolgt über den Firmware-FB T64_REC.

Der FB T64_RSYN ist in der FW-Bibliothek SYSTEM2_PLC01_20bd00 (PROPROP wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher enthalten.

Die Prozedur 3964R® ist ein eingetragenes Warenzeichen der Firma Siemens AG.

Parameter Eingang	Datentyp	Beschreibung
a_DB_BMARRAY	ANY ¹⁾	Array, das für die Kommunikation mit der Anschaltung an die Prozedur 3964R® auf Datenbausteine freigegeben werden soll.
i_DB_NR	INT	Nummer des Datenbausteins, mit der das Array vom Kommunikations-Partner angesprochen werden soll

Parameter Ausgang	Datentyp	Beschreibung
a_DB_BMARRAY	ANY ¹⁾	Array, das für die Kommunikation mit der Anschaltung an die Prozedur 3964R® auf Datenbausteine freigegeben werden soll.
i_ERR	INT	Fehlernummer

¹⁾ Es ist ein Array von der Größe 256 Byte anzuschließen. Es eignen sich z. B. der Datentyp SINT_256_BMARRAY (: ARRAY [0..255] OF SINT) oder der Datentyp DINT_64_BMARRAY (: ARRAY [0..63] OF DINT) aus der Bibliothek BM_TYPES_20bd03 (PROPROG wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher.



HINWEIS

Es müssen alle Eingänge belegt werden, da sonst kein definierter Wert übergeben wird!

Beschreibung:

Für die Anschaltung an die Prozedur 3964R® über Datenbausteine müssen in der Kalt- und in der Warmstart-Task einmalig die Zuordnungen der unterschiedlichen Datenbaustein-Nummern zu den unterschiedlichen Arrays getroffen werden.

Es sind bis zu 20 Zuordnungen möglich, d. h. der Firmware-FB T64_RSYN kann höchstens 20 mal aufgerufen werden. Mit jedem Aufruf wird die angeschlossene Nummer und das zugeordnete Array in eine system-intern verwaltete Tabelle eingetragen; im zyklischen Programmteil genügt dann der Aufruf des Firmware-FB T64_REC, mit dem dann über diese interne Tabelle die eigentliche Kommunikation abgewickelt wird.

Ein-/Ausgang a_DB_BMARRAY:

An a_DB_BMARRAY wird eine Variable vom Datentyp ARRAY mit der Größe 256 Byte angeschlossen (siehe oben).

Eingang i_DB_NR:

Mit i_DB_NR wird dem angeschlossenen Array eine entsprechende Datenbaustein-Nummer zugeordnet.

Fehlerauswertung:

i_ERR	Fehlerbeschreibung
0	kein Fehler
1	Es wurden zuviele Zuordnungen getroffen (maximal 20 möglich).
2	Fehlender Eintrag von a_DB_BMARRAY.
3	Kein Array-Typ an a_DB_BMARRAY angeschlossen.

4.6.16 Funktionsbaustein T64_REC

Über den Firmware-FB T64_REC können Daten zwischen der Terminal-Schnittstelle (serielle Schnittstelle) am Optionsmodul b maXX PLC und einem weiteren Teilnehmer ausgetauscht werden.

Der FB T64_REC ist in der FW-Bibliothek SYSTEM2_PLC01_20bd00 (PROPROP wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher enthalten.

Als Übertragungsprotokoll wird die Anschaltung an die Prozedur 3964R[®] eingesetzt mit der Vereinfachung, dass der Datentransfer nur über Datenbausteine erfolgen kann und keine Folgetelegramme unterstützt werden.

Die Prozedur 3964R[®] ist ein eingetragenes Warenzeichen der Firma Siemens AG.

Die Terminal-Schnittstelle am Optionsmodul b maXX drive PLC dient dabei immer als Slave-Anschaltung an die Prozedur 3964R[®] auf Datenbausteine.

Für die Anschaltung an die Prozedur 3964R[®] auf Datenbausteine müssen vor der Verwendung von T64_REC die entsprechenden Zuordnungen zwischen den von der Kommunikation verwendeten Datenbaustein-Nummern und den einzelnen Array-Speicherbereichen getroffen worden sein (siehe Firmware-FB T64_RSYN).

Parameter Eingang	Datentyp	Beschreibung
x_STRT	BOOL	Freigabe für genau einen Kommunikationszyklus bei steigender Flanke
x_AUTO	BOOL	Wenn TRUE erfolgen alle folgenden Kommunikationszyklen automatisch

Parameter Ausgang	Datentyp	Beschreibung
i_STAT	INT	Status-Meldung (-1, 0, 1, 2, 3, 4)
i_ERR	INT	Fehlernummer

HINWEIS



Es müssen alle Eingänge belegt werden, da sonst kein definierter Wert übergeben wird!

Beschreibung:

Die Kommunikation findet zwischen zwei Teilnehmern statt (Punkt-Zu-Punkt-Verbindung). Der angeschlossene Partner ist dabei immer der Master, von dem der Kommunikationsaufbau ausgeht. Die Optionsmodul b maXX drive PLC Anschaltung an die Prozedur 3964R[®] auf Datenbausteine stellt den reagierenden Slave dar.

Für die Übertragung mit RS485-Schnittstellen am Optionsmodul b maXX drive PLC ist eine Vierdrahtleitung (plus Bezugsmasse) Voraussetzung, da die Daten bidirektional übertragen werden müssen. Außerdem ist darauf zu achten, dass die differentiellen Lei-

tungen mit entsprechenden Pull-Up-Widerständen abgeschlossen sind (siehe Firmware-FB TER_INIT).

Der Firmware-FB T64_REC sollte in regelmäßigen Zeitabständen (Richtwert: ≥ 10 ms) aufgerufen werden. Daher ist es sinnvoll, den Aufruf in einer zyklischen Task zu implementieren oder über eine Timer-Event-Task abzuwickeln, falls die Programmlaufzeit der zyklischen Task zu groß ist.

Für die Initialisierung der Terminal-Schnittstelle werden die Firmware-FBs TER_INIT und T64_RSYN in der Kalt- und in der Warmstart-Task eingesetzt (siehe Firmware-FBs TER_INIT und T64_RSYN).

Eingang **x_STRT**:

Mit einer steigenden Flanke am Eingang x_STRT des Firmware-FB T64_REC wird die Kommunikation einmalig freigegeben. Die Übertragung zum Partner wird jedoch nach dem ersten Kommunikationszyklus wieder gesperrt. Diese Funktion kann zu Testzwecken benutzt werden, um genau einen Kommunikationszyklus zu bearbeiten.

Eingang **x_AUTO**:

Wenn hier TRUE angeschlossen wird, werden auch alle nachfolgenden Kommunikationsaufträge vom Partner bearbeitet. Auch dann, wenn ein Fehler am Ausgang i_STAT (= -1) gemeldet wurde.

Üblicherweise sollte der Anwender nach dem ersten Aufruf vom Firmware-FB T64_REC diesen Eingang fest auf TRUE setzen, damit alle Kommunikationsanforderungen vom Partner an die Steuerung bearbeitet werden können.

Ausgang **i_STAT**:

Nach Start der Kommunikation mit x_AUTO = TRUE und einer steigenden Flanke am Eingang x_STRT wird die Kommunikation freigegeben. Sobald der Ausgang i_STAT = 2 angezeigt, können vom Kommunikations-Partner ankommende Lese- und Schreibaufträge bearbeitet werden. Der Kommunikations-Partner kann dann direkt auf die Array-Inhalte über die zugeordneten Datenbaustein-Nummern zugreifen (siehe Firmware-FB T64_RSYN).

Wurde ein Auftrag vom Kommunikations-Partner korrekt empfangen, wechselt der Ausgang i_STAT von 2 auf 3 und der Auftrag wird vom Optionsmodul b maXX drive PLC bearbeitet. Beim nächsten Aufruf des Firmware-FB T64_REC wird die Rückantwort zum Kommunikations-Partner gesendet und der Ausgang i_STAT wechselt von 3 auf 4. Nach Abschluss des Sende-Vorgangs wird wieder selbstständig (also ohne notwendige Flanke am Eingang x_STRT) der Ausgang i_STAT = 2 gesetzt, so dass ein neuer Auftrag vom Kommunikations-Partner an das Optionsmodul b maXX drive PLC bearbeitet werden kann. Tritt während der Kommunikation ein Empfangsfehler auf, wird am Ausgang i_STAT der Fehlerzustand "-1" angezeigt.

i_STAT	interner Zustand der Kommunikation der Anschaltung an Prozedur 3964R® auf Datenbausteine
0	Firmware-FB T64_REC ist nicht aktiv oder Initialisierung nicht durchlaufen
1	RECEIVE-Funktion wurde angestoßen

i_STAT	interner Zustand der Kommunikation der Anschaltung an Prozedur 3964R® auf Datenbausteine
2	Ab jetzt können Aufträge vom Kommunikations-Partner empfangen werden
3	Telegramm erhalten, Auftrag wird ausgeführt; "Antwort zurückschicken" einleiten
4	Warten, bis Antwort abgeschickt: Dann über Zustand 1 in Zustand 2 wechseln
-1	Fehler aufgetreten! Fehlerursache in i_ERR näher spezifiziert

Fehlerauswertung:

Tritt während der Kommunikation ein Empfangsfehler auf, wird am Ausgang i_STAT der Fehlerzustand "-1" angezeigt, wobei die genaue Fehlerursache am Ausgang i_ERR angezeigt wird. Beim nächsten Aufruf des Firmware-FB T64_RSYN wird wieder selbstständig in den Empfangsmodus i_STAT = 2 gewechselt.

i_ERR	Fehlernummer
0	kein Fehler
1	Reserviert
2	Terminal-Schnittstelle nicht initialisiert (siehe Firmware-FB TER_INIT)
3	BREAK-Fehler Terminal-Schnittstelle
4	FRAME-Fehler Terminal-Schnittstelle
5	PARITY-Fehler Terminal-Schnittstelle
6	OVERRUN-Fehler Terminal-Schnittstelle
7 bis 15	Reserviert
16	max. Anzahl der Kommunikationsversuche erreicht
17	falsche ENDE-Kennung nach Blockübertragung erhalten
18	Zeichenverzugszeit (ZVZ) überschritten
19	Blockverzugszeit (BVZ) überschritten
20	falsches Zeichen als Verbindungszeichen (kein STX)
21 bis 49	Reserviert
50	empfangene Telegrammlänge des Kommunikations-Partners zu kurz (< 10 Zeichen)
51	Auftrag nicht implementiert (weder "SEND" noch "FETCH"), oder vom Kommunikations-Partner angeforderter Quell-/Zielbereich wird nicht unterstützt
52	Zugriffsbereich vom Anwender nicht freigegeben (siehe Firmware-FB T64_RSYN)
ab 53	Reserviert

4.6.17 Funktionsbaustein TER_USS

Mit dem FB TER_USS wird eine Kommunikation mit Anschaltung an das USS-Protokoll® realisiert. Die Verbindung erfolgt über die Terminal-Schnittstelle (serielle RS485 X1-Schnittstelle) der b maXX drive PLC.

Die b maXX drive PLC ist der alleinige Master und kann mit allen Slaves, die das USS-Protokoll® verarbeiten können, kommunizieren.



HINWEIS

Die genaue Funktionsweise, der Telegrammaufbau und die Bedeutung von Parametern des USS-Protokolls® ist in der Beschreibung "Baumotronic Kommunikations-Software" nachzulesen.



HINWEIS

Der FB TER_USS ist in der Standard-Bibliothek SYSTEM1_PLC01_20bd00 (PROPROP wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher enthalten und verwendet den Firmware-FB USS_SR aus der Firmwarebibliothek SYSTEM2_PLC01_20bd00 (PROPROP wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher. In dieser Bibliothek befindet sich auch der Firmware-FB TER_INIT, welcher zur Initialisierung der Terminal-Schnittstelle benötigt wird. Zudem werden Datentypen aus der Bibliothek BM_TYPES_20bd03 (PROPROP wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher verwendet.

Das USS-Protokoll® ist ein eingetragenes Warenzeichen der Firma Siemens AG.

Parameter Eingang	Datentyp	Beschreibung
a_INIT_DATA	UINT_256_BMARRAY	Array mit den Schnittstellenparametern
ud_MSEC_DELAY_MAX	UDINT	maximale Verzugszeit für zu empfangende Zeichen
us_MSG_MIN	USINT	minimale Anzahl der Komm-Versuche
us_MSG_MAX	USINT	maximale Anzahl der Komm-Versuche
us_ADR	USINT 0 bis 31	Adresse des USS-Teilnehmers
x_BROADCAST	BOOL	Broadcast-Telegramm an alle Teilnehmer
x_MIRROR	BOOL	Spiegeltelegramm
us_SIZE_PIV	USINT 0, 3, 4	PKW-Anzahl
us_JOB_IDENT	USINT 0 bis 4	Auftrags-Kennung
u_PARA_IDENT_WRITE	UINT	Parameter-Nummer

Parameter Eingang	Datentyp	Beschreibung
ud_PARA_VALUE_WRITE	UDINT	Parameterwert schreiben
u_INDEX_WRITE	UINT	Index-Nummer schreiben
us_SIZE_PD	USINT 0 bis 3	PZD_Anzahl
w_CONTROLWORD	WORD	Steuerwort Prozessdaten
ud_PROC_DATA_WRITE	UDINT	Sollwert Prozessdaten schreiben
x_COM_INIT	BOOL	Startbit für Telegramm (steigende Flanke)
x_COM_CYCL	BOOL	Anschluss für zyklische Datenübertragung

Parameter Ausgang	Datentyp	Beschreibung
a_INIT_DATA	UINT_256_BMARRAY	Array mit den Schnittstellenparametern
x_BUSY	BOOL	Anzeige bit für laufende Kommunikation
x_VALID	BOOL	Anzeige bit für Empfangsdaten gültig
us_REPLY_IDENT	USINT	empfangene Auftrags-Kennung
u_PARA_IDENT_READ	UINT	empfangene Parameter-Nummer
u_INDEX_READ	UINT	Index-Nummer lesen
ud_PARA_VALUE_READ	UDINT	empfangener Parameterwert
w_STATUSWORD	WORD	empfangenes Statuswort Prozessdaten
ud_PROC_DATA_READ	UDINT	empfangener Istwert Prozessdaten
si_SER_ERR	SINT	Fehlernummer serielle Schnittstelle
si_MSG_ERR	SINT	Fehlernummer Telegramm
si_PARA_ERR	SINT	Fehlernummer Auftrag
x_ERR	BOOL	globales Fehlerbit

HINWEIS

Es müssen alle Eingänge belegt werden, da sonst kein definierter Wert übergeben wird!

Beschreibung:

Der FB TER_USS sollte zyklisch (Richtwert: ≥ 10 ms) aufgerufen werden. Daher ist es sinnvoll, den Aufruf des FB TER_USS in einer zyklischen Task zu implementieren oder über eine Timer-Event-Task abzuwickeln, falls die Programmlaufzeit der zyklischen Task durch darin befindliche andere Programmteile zu hoch oder zu unregelmäßig ist.

Über die Anbindung an das USS-Protokoll[®] können Bedarfsdaten über den cPKW-Bereich zwischen dem Master (b maXX drive PLC) und der daran angeschlossenen USS-Slaves ausgetauscht werden.

Die Prozessdatenübertragung ("schneller Soll/Istwert-Kanal") erfolgt im USS-Protokoll® über den cPZD-Bereich.

An `a_INIT_DATA` muss das gleiche Array (global deklariert) wie am Firmware-FB `TER_INIT` angeschlossen werden, da während der Kommunikation auf dieses Array zugegriffen wird.

Bei einer steigenden Flanke am Eingang `x_COM_INIT` wird die USS-Kommunikation gestartet. Sollen automatisch neue Kommunikations-Zyklen angestoßen werden, muss an dem Eingang `x_COM_CYCL` = TRUE eingestellt sein.

Der Ausgang `x_BUSY` bleibt solange gesetzt, bis eine Master-Slave-Kommunikation zum Teilnehmer vollständig abgeschlossen ist. Der Abschluss einer Kommunikation wird durch `x_VALID` = TRUE angezeigt. Dieser bleibt nach fallender `x_BUSY`-Flanke einen Programm-Zyklus lang TRUE.

Am Eingang `ud_MSEC_DELAY_MAX` wird die maximale Timeoutzeit in ms angeschlossen, bis wann eine Master-Slave-Kommunikation abgeschlossen sein muss.

Wird diese Zeit nach Anstoßen der Kommunikation überschritten, erfolgt eine "Timeout"-Fehlermeldung. Üblicherweise sollte dort ein Wert von ca. 220 ms angeschlossen werden.

Falls kein Fehler während der Kommunikation aufgetreten ist, werden bei `x_VALID` = TRUE die Ausgänge mit den empfangenen Werten (`w_STATUSWORD`, `ud_PROC_DATA_READ`, `us_REPLY_IDENT`, `u_PARA_IDENT_READ`, `u_INDEX_READ`, `ud_PARA_VALUE_READ`) beschrieben.

Vor der Parameterausgabe erfolgt eine Überprüfung der gesendeten und empfangenen Parameternummern `u_PARA_IDENT_WRITE` und `u_PARA_IDENT_READ`.

Dabei wird das Telegramm genau `us_MSG_MIN`-mal angefordert, bis eine Parameterausgabe erfolgt.

Bei einem Fehler (z. B. "Timeout") wird der Telegramm-Aufbau `us_MSG_MAX`-mal versucht, bevor der Ausgang `x_VALID` = TRUE wird und ein Fehler angezeigt wird. Je nachdem, was für ein Fehler aufgetreten ist, können folgende Ausgänge ausgewertet werden:

Parameter- und Empfangs-Fehler, Timeout-Fehler:

Das `x_ERR`-Bit und die Fehlernummern (`si_SER_ERR`, `si_MSG_ERR`, `si_PARA_ERR`) werden ausgegeben, die anderen Ausgänge behalten ihren vorherigen Instanzwert!

Auftrags-Fehler:

Das Bit `x_ERR`, `si_SER_ERR`, `si_MSG_ERR`-Zustand, die Auftrags-Fehlernummer `si_PARA_ERR`, `us_REPLY_IDENT`, `u_PARA_IDENT_READ` werden neu beschrieben, `u_INDEX_READ` und `ud_PARA_VALUE_READ` behalten ihren vorherigen Instanzwert!

Die Prozessdaten (`w_STATUSWORD` und `ud_PROC_DATA_READ`) werden bei jedem geglückten Telegramm-Aufbau ausgegeben, auch wenn ein sonstiger Parameter-Auftragsfehler aufgetreten ist. Die Prozessdaten-Ausgabe ist deswegen von `us_MSG_MIN` unabhängig!

Die Eingänge `us_MSG_MIN` und `us_MSG_MAX` legen fest, wie oft eine Kommunikation mindestens (`us_MSG_MIN`) und wie oft höchstens (`us_MSG_MAX`, z. B. bei Timeout-Meldung) stattfinden darf, bevor eine entsprechende Reaktion am Ausgang durch die vom Slave zurückgelieferten Daten oder eine Fehlermeldung erfolgt.

Normalerweise sollte der Anwender folgende Initialisierungswerte verwenden:

us_MSG_MIN = 1;
us_MSG_MAX = 2;

Die Organisation des Inhalts im Telegrammaufbau des USS-Protokoll® wird über die PKW- und PZD-Anzahl festgelegt. Diese Einstellungen müssen bei allen Teilnehmern im USS-Ring identisch sein, damit der Inhalt des ausgetauschten Telegramms richtig interpretiert werden kann.

us_SIZE_PIV	PKW-Anzahl: Aufteilung der Bedarfsdaten im USS-Protokoll
4	Doppelwort-Parameter (32 Bit-Typen)
3	Wort-Parameter (16 Bit-Typen)
0	Telegramm enthält keine Bedarfsdaten!

us_SIZE_PD	PZD-Anzahl: Aufteilung der Prozessdaten im USS-Protokoll
3	Steuer/Statuswort und Doppelwort-Soll/Istwerte (32 Bit-Typen)
2	Steuer/Statuswort und Wort-Soll/Istwerte (16 Bit-Typen)
1	nur Steuer/Statuswort
0	Telegramm enthält keine Prozess-Daten!

us_JOB_IDENT	Auftragskennung (Master -> Slave)
0	kein Auftrag
1	Parameterwert lesen (Antwort in ud_PARA_VALUE_READ)
2	Parameterwert ud_PARA_VALUE_WRITE schreiben vom Typ Wort (16 Bit)
3	Parameterwert ud_PARA_VALUE_WRITE schreiben vom Typ Doppelwort (32 Bit)
4	Ein Element aus der Parameter-Beschreibung lesen (welches, steht in u_INDEX_WRITE)
5 bis ...	(reserviert)

us_REPLY_IDENT	Antwortkennung (Slave -> Master)
0	kein Auftrag
1	Parameter-Wort (16 Bit) wurde geschrieben (us_JOB_IDENT = 2) bzw. gelesen (us_JOB_IDENT = 1) (in ud_PARA_VALUE_READ steht der übertragene Wert)
2	Parameter-Doppelwort (32 Bit) wurde geschrieben (us_JOB_IDENT = 3) bzw. gelesen (us_JOB_IDENT = 1) (in ud_PARA_VALUE_READ steht der übertragene Wert)

4.6 BACI-Systembeschreibung für die b maXX drive PLC

us_REPLY_IDENT	Antwortkennung (Slave -> Master)
3	Element aus der Parameter-Beschreibung wurde gelesen (Antwort steht in ud_PARA_VALUE_READ)
4 bis 6	(reserviert)
7	Parameter-Fehler: Auftrag nicht ausführbar! (Fehlerkennung siehe si_PARA_ERR)
8 bis ...	(reserviert)

Fehlerauswertung:

si_SER_ERR	Fehler serielle Schnittstelle
0	kein Fehler
1	(reserviert)
2	Falsches Serial-Initialisierungsarray a_INIT_DATA angeschlossen
3	Schnittstelle nicht initialisiert
4 bis 9	(reserviert)
10	Break beim Empfangen
11	Frame-Fehler beim Empfangen
12	Parity-Fehler beim Empfangen
13	Overrun-Fehler beim Empfangen

si_MSG_ERR	Telegramm-Fehler beim Senden/Empfangen
0	kein Fehler
1	Der am FB angeschlossene us_SIZE_PIV-Bereich wird nicht unterstützt
2	Der am FB angeschlossene us_SIZE_PD-Bereich wird nicht unterstützt
3	us_MSG_MIN > us_MSG_MAX oder us_MSG_MIN bzw. us_MSG_MAX = 0
4	an us_ADR ungültige Adresse angeschlossen (nur 0..31 zulässig)
5	gesendete und empfangene Adresse unterschiedlich
6	gesendete und empfangene Parameternummer u_PARA_IDENT_WRITE / u_PARA_IDENT_READ unterschiedlich!
7	us_SIZE_PIV-Bereich falsch
8	us_SIZE_PD-Bereich falsch
9	STX-Zeichen nicht empfangen!
10	empfangene und erwartete Längenangabe (us_SIZE_PIV/us_SIZE_PD) unterschiedlich!
11	BCC-Checksummen-Fehler
12	Timeout-Fehler beim Empfangen

si_PARA_ERR	Parameter-Fehler
0	kein Fehler
-1	Parameter-Nummer wird nicht unterstützt (unzulässige u_PARA_IDENT_WRITE)
1	Parameter nicht änderbar
2	MIN/MAX-Begrenzung
3	fehlerhafter Index u_INDEX_WRITE
4	kein Array-Typ
5	falscher Datentyp
6	kein Setzen erlaubt
7	Beschreibungselement nicht änderbar
8 bis 100	(reserviert)
101	Unbestimmter Fehler
102	Dienst nicht implementiert
103	Parameterformat zu groß für PKW-Bereich
104	PBE-Element nicht vorhanden

4.7 Codelaufzeiten

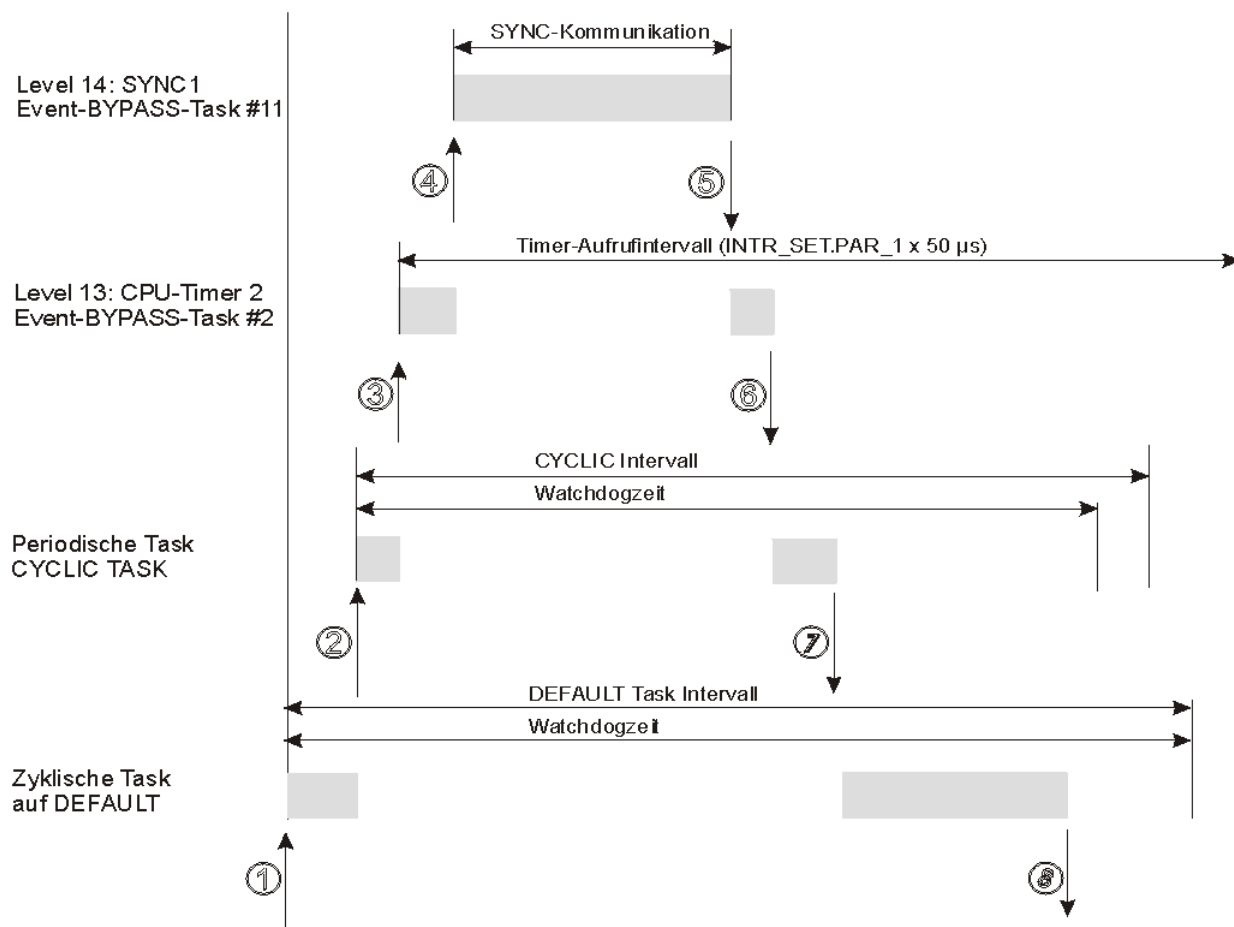


Abbildung 57: Aufteilung von Codelaufzeiten auf die Ressource BM4_O_PLCO1



HINWEIS

Alle Events müssen als Tasks mit der Eigenschaft "BYPASS" angelegt werden.

Eine höherpriorige Event-Task unterbricht eine niederpriorige Event-Task, daher sollten Timer-Event-Tasks, die zeitgleich zu synchronen Bussystemen laufen könnten, immer mit dem niedrigeren Level 13 eingesetzt werden, damit sie unterbrochen werden können.

Bei zyklischen Tasks, die keine BYPASS-Tasks sind, muss in den "Einstellungen" der Task die zyklische Abtastzeit (= Einstellung "Intervall") und eine Überwachungszeit (= Einstellung "Watchdogzeit") vom Anwender angegeben werden. Diese zyklischen Tasks sowie die Default-Task werden hinsichtlich der Watchdogzeit während des Betriebes durch das Laufzeitsystem überwacht.

Dabei ist darauf zu achten, dass die eingestellte Watchdogzeit nicht nur die reine Laufzeit dieser Task selbst berücksichtigt, sondern auch die Summe der maximalen Einzel-Laufzeiten enthält, die diese Task unterbrechen können.

Ist die Überwachungszeit zu klein eingestellt, wird der SPS-Fehler "Watchdog in Task 'x' überschritten!" zurückgemeldet, um den Anwender auf die aufgetretene System-Überlast hinzuweisen, und die Steuerung geht gleichzeitig in den sicheren Zustand "STOP".

Die Watchdogzeiten in BYPASS-Tasks werden ignoriert.

Weitere Einfluss-Faktoren auf die Watchdogzeiten in den zyklischen Nicht-Bypass-Tasks und der Default-Task:

Für die Überwachungszeiten sollte man zusätzlich eine gewisse Reserve einkalkulieren für sonstige im Hintergrund ablaufende Systemfunktionen, wie z. B. Task-Aufrufschalen sowie Funktionen, die für die Online-Darstellungen von Variablen in den Watch-Listen und/oder in den globalen Variablen-Arbeitsblättern aufgerufen werden und die Systemlast erhöhen.

Außerdem können bestimmte Überwachungsfunktionen vom Anwender eingeschaltet sein, die während der Laufzeit aktiv sind:

Dazu gehören die Einstellungen in der Ressource für "Index-Prüfung auf SPS" und "Stack Prüfung auf SPS". Diese Einstellungen können eventuell nach Abschluss einer Inbetriebnahme wieder zurückgesetzt werden, um Laufzeit-Ressourcen zu sparen.

Für die BYPASS-Tasks lässt sich die tatsächliche Systembelastung während des Betriebes unter einer "worst case"-Betrachtung durch Einsatz der "Time_Measure"- Bausteine aus der Bibliothek SYSTEM1_PLC01_20bd00 (PROProg wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher exakt ermitteln. Mit den FBs lassen sich Laufzeiten von max. 10 ms mit einer sehr hohen Auflösung erfassen.

Wenn in den niederpriorigen zyklischen Tasks größere Zeitabstände ausgemessen werden müssen, kann in einem BYPASS-Interrupt, der mit einer festen Zeitbasis läuft, ein kontinuierlicher globaler Zählwert gebildet werden, der dann direkt in den zyklischen Tasks über Differenzbildung des Zählwertes ausgewertet werden kann. (Wie man einen zyklischen CPU-Timer als BYPASS-Task einrichtet, siehe Beschreibung zu FB "INTR_SET" in der FW-Lib SYSTEM2_PLC01_20bd00 (PROProg wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher).

Ein BYPASS-Event kann über den FB "INTR_SET" der FW-Bibliothek SYSTEM2_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher mit x_EN = FALSE auch wieder während der Laufzeit gesperrt werden.

Vermeidung einer System-Überlastung:

Bei Watchdog-Überschreitungen müssen die Einstellungen (Intervall bzw. Watchdogzeit) nach oben gesetzt werden und/oder die Laufzeiten in den BYPASS-Interrupts minimiert werden, entweder durch Programm-Aufteilung im Interrupt selbst (= abwechselnde Programmteil-Abläufe je Interrupt) oder durch Verlagerung in niederepriorige Nicht-BYPASS-Tasks mit größeren Aufruf-Intervallen bzw. durch Platzierung von Programmteilen direkt in die Default-Task.

Soll in zyklischen Nicht-Bypass-Tasks keine Laufzeit-Überwachung stattfinden, kann auch eine Watchdogzeit eingestellt werden, die größer als das für diese Task eingestellte Intervall ist.

4.7.1 Funktionsbaustein TIME_MEASURE_START

Diesen Funktionsbaustein für TIME_MEASURE können Sie zusammen mit dem Funktionsbaustein TIME_MEASURE_END verwenden, um Code-Laufzeiten und Aufrufzeiten für Tasks zu ermitteln.

Der Funktionsbaustein TIME_MEASURE_START ist in der Standard-Bibliothek SYSTEM1_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher enthalten.

Ausgangsparameter	Datentyp	Beschreibung
ud_CNT_START	UDINT	aktueller Systemtimer-Wert in Systemeinheiten

Beschreibung

Der FB TIME_MEASURE_START wird zusammen mit dem FB TIME_MEASURE_END für Zeitmessungen auf der b maXX drive PLC eingesetzt. Ermittelt wird die Laufzeit des Programmcodes, der sich zwischen dem Aufruf des FB TIME_MEASURE_START (Beginn der Zeitmessung) und dem Aufruf des FB TIME_MEASURE_END (Ende der Zeitmessung) befindet.

Der FB TIME_MEASURE_START liest von einem Systemtimer den aktuellen Wert in Systemeinheiten und gibt diesen an ud_CNT_START aus. Der FB TIME_MEASURE_END liest wiederum den aktuellen Systemtimer-Wert in Systemeinheiten und bildet die Differenz zum vom FB TIME_MEASURE_START gelesenen Wert. An beiden FBs muss daher an ud_CNT_START die gleiche Variable angeschlossen werden.

**HINWEIS**

Eine korrekte Zeitmessung ist nur möglich, wenn die beiden FBs in einer Event-Task mit der höchsten Priorität eingesetzt werden. Es können Laufzeiten bis max. 10 ms ermittelt werden. Bei Programmcode, der mehr als 10 ms Laufzeit hat, werden keine korrekten Werte geliefert.

4.7.2 Funktionsbaustein TIME_MEASURE_END

Diesen Funktionsbaustein für TIME_MEASURE können Sie zusammen mit dem Funktionsbaustein TIME_MEASURE_START verwenden, um Code-Laufzeiten und Aufrufzeiten für Tasks zu ermitteln.

Der Funktionsbaustein TIME_MEASURE_END ist in der Standard-Bibliothek SYSTEM1_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher enthalten.

Eingangsparameter	Datentyp	Beschreibung
ud_CNT_START	UDINT	Systemtimer-Wert von FB TIME_MEASURE_START

Ausgangsparameter	Datentyp	Beschreibung
ud_CNT_END	UDINT	aktueller Systemtimer- Wert in Systemeinheiten
di_DIFF_in_counts	DINT	Differenz zwischen ud_CNT_END und ud_CNT_START
di_DIFF_in_mysec	DINT	Zeitdifferenz in ms

Beschreibung

Der FB TIME_MEASURE_START wird zusammen mit dem FB TIME_MEASURE_END für Zeitmessungen auf der b maXX drive PLC eingesetzt. Ermittelt wird die Laufzeit des Programmcodes, der sich zwischen dem Aufruf des FB TIME_MEASURE_START (Beginn der Zeitmessung) und dem Aufruf des FB TIME_MEASURE_END (Ende der Zeitmessung) befindet.

Der FB TIME_MEASURE_START liest von einem Systemtimer den aktuellen Wert in Systemeinheiten und gibt diesen an ud_CNT_START aus. Der FB TIME_MEASURE_END liest wiederum den aktuellen Systemtimer-Wert in Systemeinheiten und bildet die Differenz zum vom FB TIME_MEASURE_START gelesenen Wert. An beiden FBs muss daher an ud_CNT_START die gleiche Variable angeschlossen werden.

Die Zeitdifferenz wird an di_DIFF_in_mysec ausgegeben, an di_DIFF_in_counts steht die Differenz der Systemtimer-Werte in Systemeinheiten zur Verfügung. Der vom FB TIME_MEASURE_END ausgelesene Systemtimer-Wert kann an ud_CNT_END abgelesen werden.

Um die Aufrufzeit einer Task zu ermitteln, werden die beiden FBs an den Beginn einer POE gesetzt, wobei der FB `TIME_MEASURE_END` vor dem FB `TIME_MEASURE_START` zu platzieren ist. Die Beschaltung mit Variablen erfolgt genauso wie bei einer Code-Laufzeitmessung. An `di_DIFF_in_mysec` kann dann die Aufrufzeit abgelesen werden.



HINWEIS

Eine korrekte Zeitmessung ist nur möglich, wenn die beiden FBs in einer Event-Task mit der höchsten Priorität eingesetzt werden. Es können Laufzeiten bis max. 10 ms ermittelt werden. Bei Programmcode, der mehr als 10 ms Laufzeit hat, werden keine korrekten Werte geliefert.

4.8 Praktische Hinweise

4.8.1 SPS-Fehler "Watchdog in Task 'x' überschritten!"

Die SPS-Fehlermeldung ist ein vom Laufzeitsystem generierter Fehler, um den Anwender auf folgende System-Überlastung hinzuweisen:

Bei zyklischen Tasks, die keine BYPASS-Tasks sind, muss in den "Einstellungen" der Task die zyklische Abtastzeit (= Einstellung "Intervall") und eine Überwachungszeit (= Einstellung "Watchdogzeit") vom Anwender angegeben werden. Diese Tasks werden während des Betriebes durch das Laufzeitsystem überwacht. Wenn nun eine Verletzung der gewählten Einstellungen durch das Laufzeitsystem festgestellt wird, geht die SPS in den Zustand "Stop" und meldet den Fehler "Watchdog in Task 'x' überschritten!" zurück.

Mögliche Gründe für eine Verletzung:

Werden BYPASS-Tasks vom Anwender eingesetzt, die sehr hoch ausgelastet sind (z. B. reine Interrupt-Laufzeit von 1,5 ms bei 2 ms Abtastzeit) und sich womöglich noch gegenseitig unterbrechen (z. B. eine weitere BYPASS-Timer-Task mit 300 ms reiner Laufzeit), stellt dies eine zusätzliche Belastung der CPU-Rechenzeit dar, die unter einer "worst case"-Laufzeitbetrachtung zu diesen oben genannten SPS-Fehler in einer niederprioritären, vom Betriebssystem überwachten zyklischen Task (= Nicht-BYPASS-Task) führen kann.

Womit man "rechnen" sollte:

Bei der Einrichtung der Überwachungszeit muss man also nicht nur die reine Laufzeit der einzurichtenden Task alleine berücksichtigen, sondern auch die Maximal-Laufzeiten von höherprioritären Tasks und BYPASS-Tasks hinzuaddieren sowie eine gewisse Reserve berücksichtigen für sonstige ablaufende Systemfunktionen, z. B. für die im Hintergrund ablaufenden Task-Aufrufschalen sowie den Online-Darstellungen von Variablen in den Watch-Listen und/oder in den globalen Variablen-Arbeitsblättern. Auch sollte man nicht vergessen, dass bestimmte Überwachungsfunktionen vom Anwender eingeschaltet sein können, die während der Laufzeit aktiv sind (z. B. Die Einstellungen in der Ressource für "Index-Prüfung auf SPS" oder "Stack Prüfung auf SPS") und die eventuell nach Abschluss einer Inbetriebnahme wieder zurückgesetzt werden können, um Ressourcen zu sparen.

Für die BYPASS-Tasks lässt sich die tatsächliche Systembelastung während des Betriebes unter einer "worst case"-Betrachtung durch Einsatz der "Time_Measure"- Bausteine aus der Bibliothek SYSTEM1_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher exakt ermitteln. Mit den FBs lassen sich Laufzeiten von max. 10 ms mit einer sehr hohen Auflösung erfassen.

Wenn in den niederpriorigen zyklischen Tasks größere Zeitabstände ausgemessen werden müssen, kann in einem BYPASS-Interrupt, der mit einer festen Zeitbasis läuft, ein kontinuierlicher globaler Zählwert gebildet werden, der dann direkt in den zyklischen Tasks über Differenzbildung des Zählwertes ausgewertet werden kann. (Wie man einen zyklischen CPU-Timer als BYPASS-Task einrichtet, siehe Beschreibung zu FB "INTR_SET" in der FW-Lib SYSTEM2_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher).

Vermeidung einer Überlastung:

Bei Watchdog-Überschreitungen müssen entweder die Einstellungen (Intervall bzw. Watchdogzeit) nach oben gesetzt werden und/oder die Laufzeiten in den BYPASS-Interrupts minimiert werden, entweder durch Programm-Aufteilung im Interrupt selbst (= abwechselnde Programmteil-Abläufe je Interrupt) oder durch Verlagerung in niedrigere Nicht-BYPASS-Tasks mit größeren Aufruf-Intervallen bzw. durch Platzierung von Programmteilen direkt in die Default-Task.

INTERNE KOMMUNIKATIONS- SCHNITTSTELLE ZUM REGLER (BACI)

5.1 BACI (Baumüller-Component-Interface) allgemein

Die BACI ist eine interne Kommunikationsschnittstelle für den Datenaustausch zwischen dem b maXX Regler und der b maXX drive PLC.

Bei der Kommunikation wird zwischen Prozessdatenkommunikation und Bedarfsdatenkommunikation unterschieden.

Die Prozessdatenkommunikation umfasst das Schreiben und Lesen der zeitkritischen Soll- und Istwerte sowie des Status- und Steuerworts in einem definierbaren Zeitraster.

Die Bedarfsdatenkommunikation umfasst das Schreiben und Lesen einzelner, zeitunkritischer Parameter des Reglers.

Um einen Datenaustausch über BACI zwischen b maXX drive PLC und b maXX Regler zu erreichen, stehen Ihnen verschiedene Wege zur Verfügung.

- ProMaster, ProProg wt III und Motion Control

Einfache und schnelle Konfiguration des Datenaustauschs in ProMaster. Einfache und schnelle Programmierung der Maschinenfunktion in ProProg wt III mit den Motion Control Funktionsbausteinen.

Beispiel Projekte:

ProMaster Projekt: Example_1_3_2.bmxml

IEC Projekt in ProProg wt III: Example_BM4_O_ECT02_MA_2.mwt

Siehe [►ProMaster, ProProg wt III und Motion Control◄](#) ab Seite 118.

- PROPROG wt II

Programmierung des Datenaustauschs (BACI-Kommunikation) und der Maschinenfunktion in PROPROG wt II durch die Verwendung der IEC 61131-3 Programmiersystems PROPROG wt II.

Siehe [►PROPROG wt II◄](#) ab Seite 146.

5.2 ProMaster, ProProg wt III und Motion Control

Beim Anlegen eines Projekts in ProMaster werden per Default Netzwerkvariablen für Motion Control angelegt.

Im IEC Projekt schreiben oder lesen die Motion Control Funktionsbausteine diese Default-Netzwerkvariablen. Über die Motion Control Funktionsbausteine kann die gesamte Maschinenfunktion im IEC Projekt programmiert werden.

Bei der Konfiguration des Datenaustauschs in ProMaster Projekt können zusätzliche Netzwerkvariablen angelegt werden.

Im IEC Projekt in ProProg wt III können diese zusätzlichen Netzwerkvariablen geschrieben oder gelesen werden.

Das Intervall in dem die (Prozess-) Daten zwischen b maXX drive PLC und b maXX Regler ausgetauscht werden wird in ProMaster konfiguriert.

Falls es notwendig ist Bedarfsdaten zu schreiben oder zu lesen, werden die entsprechenden Motion Control Funktionsbausteine verwendet.

5.2.1 Voraussetzungen

Die Voraussetzungen sind

- PC mit
 - Engineering Framework ProMaster
 - IEC 61131-3 Programmiersystem ProProg wt III mit den Firmware Bibliotheken
 - Bit_UTIL_30bd00
 - OmegaOS_30bd00
 - SYSTEM2_PLC01_30bd00
 - MC_SYS_30bd00
 - und den Bibliotheken
 - BM_TYPES_30bd00
 - SYSTEM1_PLC01_30bd00
 - MOTION_TYPES_30bd02
 - MOTION_CONTROL_30bd03
 - MOTION_MULTI_AXIS_30bd03 (wenn Multi Axis Motion Control verwendet wird)
- Physikalische Inbetriebnahme der Komponenten.

Im nachfolgenden Beispiel wird ein kleines EtherCAT-Netzwerk angelegt. In unserem Beispiel sind die Komponenten der EtherCAT-Master

 - b maXX 4400 mit
 - b maXX PLC (BM4-O-PLC-0x-...)
 - Optionsmodul EtherCAT-Master (BM4-O-ECT-02-...)
 - und der EtherCAT-Slave
 - b maXX 4400 mit
 - Optionsmodul EtherCAT-Slave (BM4-O-ECT-01-...)

Siehe hierzu die jeweilige Betriebsanleitung.

5.2.2 Durchzuführende Schritte

Um für Motion Control die b maXX drive PLC und den b maXX Regler verwenden zu können sind folgende Schritte durchzuführen.

- 1 Inbetriebnahme des b maXX 4400 mit b maXX Regler und b maXX drive PLC
- 2 Anlegen eines IEC 61131-3 Projekts für ProProg wt III mit einem Motion Control Template
- 3 Anlegen einer Maschinenkonfiguration in ProMaster
(im nachfolgenden Beispiel wird hier ein kleines EtherCAT-Netzwerk angelegt)
- 4 Konfigurieren der EtherCAT-Kommunikation und anschließender Download auf den EtherCAT-Master
(ggf. Testen des EtherCAT Netzwerkes mit dieser Konfiguration)
- 5 Konfigurieren der BACI-Kommunikation (lokale Achse)
- 6 Konfigurieren der PLC mit Verbinden von IEC Projekt und ProMaster, Konfigurieren der Kurvenscheibendaten
- 7 Ggf. Programmieren des IEC 61131-3 Projekts für ProProg wt III (Applikation) und anschließender Download auf die b maXX drive PLC
- 8 Betrieb der Applikation

Natürlich können sie auch mit Schritt 2 beginnen und erst vor Schritt 6 die physikalische Inbetriebnahme (Schritt 1) durchführen.

Dabei entstehen das ProMaster Projekt **Example_1_3_2.bmxml** und das IEC 61131-3 Projekt (ProProg wt III Projekt) **Example_BM4_O_ECT02_MA_2.mwt/.zwt**.

5.2.3 Inbetriebnahme des b maXX 4400 mit b maXX Regler und b maXX drive PLC

Einzelheiten zur Inbetriebnahme des b maXX 4400 mit b maXX Regler sowie des Optionsmoduls b maXX drive PLC entnehmen sie bitte den jeweiligen Betriebsanleitungen.

Einzelheiten zur Inbetriebnahme des EtherCAT-Netzwerkes entnehmen Sie bitte der Betriebsanleitung zum Optionsmodul EtherCAT-Master für b maXX drive PLC und den Betriebsanleitungen der übrigen EtherCAT-Netzwerkknoten.

5.2.4 Anlegen eines IEC Projekts mit ProProg wt III

Im Folgenden erläutern wir, wie ein Motion Control IEC Projekt angelegt wird.

In den ProMaster Default Einstellungen für Motion Control ist Multi Axis Motion Control eingestellt. Daher ist es sinnvoll, zunächst ein IEC Projekt für Multi Axis Motion Control zu erstellen.

Öffnen Sie ProProg wt III und legen Sie ein neues Projekt an, indem Sie über das Menü "Datei\Projekt öffnen / Projekt entpacken..." ein IEC-Template Projekt auswählen. Für unser Beispiel verwenden wir das IEC-Template Projekt "Tmpl_PLC01_MA_2_0104.zwt" (für BM4_O_PLC01).

Dieses Projekt finden Sie im Verzeichnis <ProMaster Installationsverzeichnis>\projects\IEC-Templates, z B. in ProMaster Standardinstallationsverzeichnis C:\Baumueller\ProMasterNET\projects\IEC-Templates.

Das IEC-Template Projekt "Tmpl_PLC01_MA_2_0104.zwt" entpacken wir als "Example_BM4_O_ECT02_MA_2.mwt".

Die Vorlage enthält die Bibliotheken

- Bit_UTIL_30bd00
- BM_TYPES_30bd01
- MC_SYS_30bd00
- MOTION_TYPES_30bd02
- MOTION_CONTROL_30bd03
- MOTION_MULTI_AXIS_30bd03

Speichern Sie das Projekt. In unserem Beispiel verwenden wir den Projektnamen "Example_BM4_O_ECT02_MA_2.mwt".

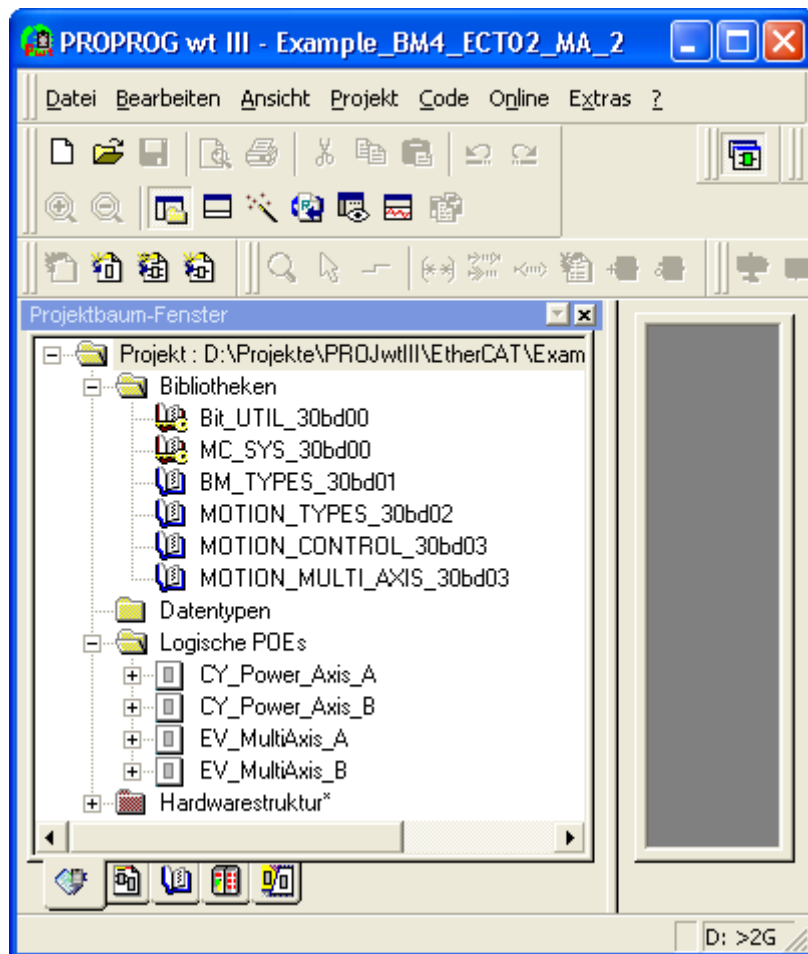


Abbildung 58: ProProg wt III - Anlegen des Projekts "Example_BM4_O_ECT02_MA_2.mwt"

Jetzt müssen noch die Kommunikationseinstellungen konfiguriert werden. Per Default ist die serielle Schnittstelle COM1 eingestellt. Falls Sie Ethernet verwenden wollen, siehe Kapitel [►Kommunikation und Verbindung◄](#) ab Seite 30.

Für die Verwendung eines IEC Projekts in ProMaster ist es unbedingt erforderlich, dass bei ProProg wt III im Fenster "Ressource - Einstellungen" (Ressource markieren, Kontextmenü "Einstellungen" anwählen) die Check-Box "Bootprojekt beim Kompilieren erzeugen" aktiviert ist. Dadurch wird die Datei "bootfile.pro" erzeugt, welche für den Download auf die b maXX drive PLC benötigt wird.



HINWEIS

In ProProg wt III muss unter "Ressource - Einstellungen" die Check-Box "Bootprojekt beim kompilieren erzeugen" aktiviert sein.

Das Projekt "Example_BM4_O_ECT02_MA_2.mwt" wird später, im Kapitel [►Konfigurieren der PLC◄](#) ab Seite 132, mit dem ProMaster Projekt verbunden.

Schließen Sie jetzt ProProg wt III über Menü „Datei\Beenden“.

5.2.5 Anlegen einer Maschinenkonfiguration in ProMaster

In diesem Abschnitt erläutern wir, wie ein Projekt in ProMaster angelegt wird und wie eine Maschinenkonfiguration erstellt wird.

Öffnen Sie ProMaster.



Abbildung 59: ProMaster ohne Projekte

Legen Sie ein neues ProMaster-Projekt an, indem Sie über Menü "Datei\NeuesProjekt" das Fenster "Projekt Einstellungen" öffnen.

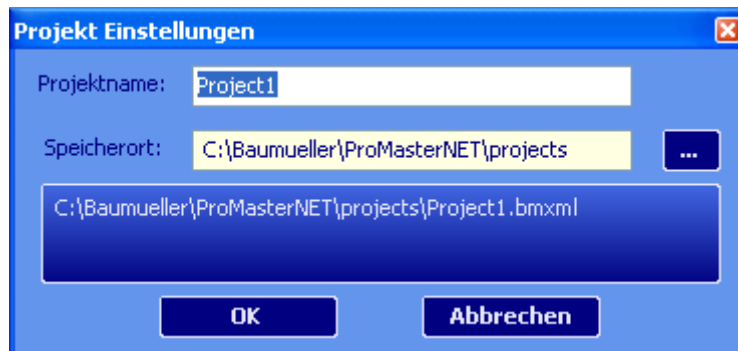


Abbildung 60: ProMaster - Projektname

Geben Sie den Namen (in der Edit-Box) und den Speicherort (über die Pfadauswahl) des Projektes an. Für unser Beispielprojekt verwenden wir den Namen "Example_1_3_2.bmxml".

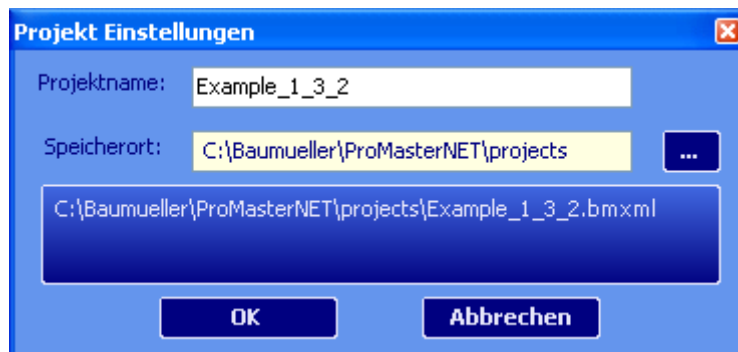


Abbildung 61: ProMaster - Projektname vergeben

Übernehmen Sie die Einstellungen aus dem Fenster "Projekt Einstellungen" durch Klicken auf den Button "OK". ProMaster wechselt jetzt in die Netzwerkansicht.

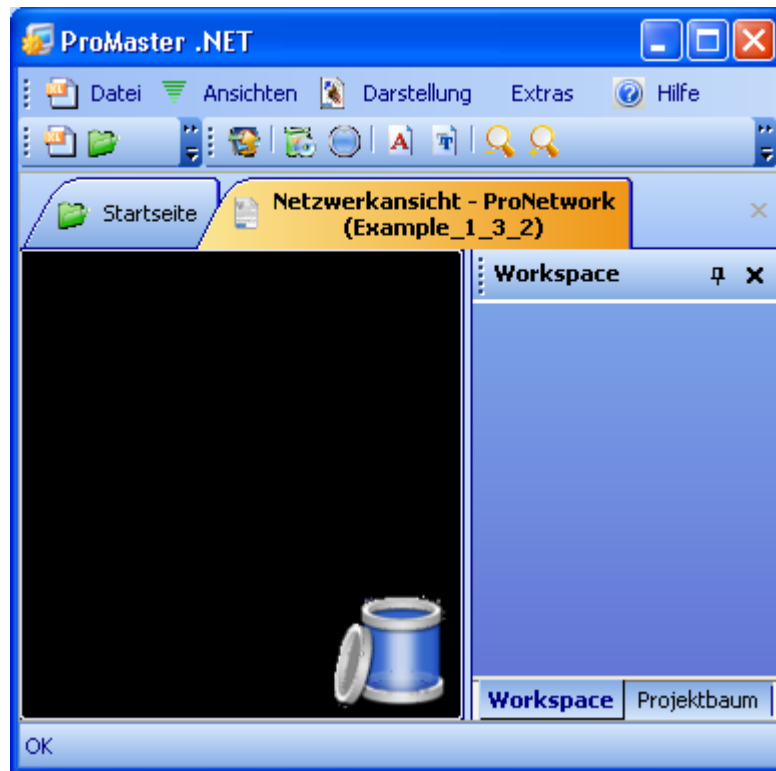


Abbildung 62: ProMaster - Projektname vergeben

Öffnen Sie jetzt über das Menü "Ansichten/Katalog" den Baumüller Katalog. Der Baumüller Katalog enthält die Geräte, Bussystem und Komponenten, die Baumüller Ihnen per Default zur Verfügung stellt.

Für unser Beispiel verwenden wir

- die Geräte
 - "bmaXX 4000 EtherCAT-Master (2-reihig)" (aus der Gruppe "bmaXX 4000 Antrieb");
b maXX 4000 mit zwei Steckplätzen,
Optionsmodul Ethernet mit EtherCAT-Master in Steckplatz G,
Optionsmodul b maXX drive PLC in Steckplatz H
 - "b maXX 4000 EtherCAT-Slave (2-reihig)" (aus der Gruppe "b maXX 4000 Antrieb");
b maXX 4000 mit zwei Steckplätzen,
Optionsmodul Ethernet mit EtherCAT-Slave in Steckplatz G
- das Bussystem
 - "EtherCAT Bus" wird automatisch angelegt.

Ziehen Sie per Drag&Drop das Gerät "bmaXX 4000 EtherCAT-Master (2-reihig)" aus dem Baumüller Katalog in die ProMaster Netzwerkansicht. Das Bussystem ist automatisch angeschlossen worden. Ziehen Sie jetzt per Drag&Drop das Gerät "bmaXX 4000 EtherCAT-Slave (2-reihig)" auf den Bus EtherCAT (Drop ist erst möglich, wenn der Bus die Farbe wechselt).

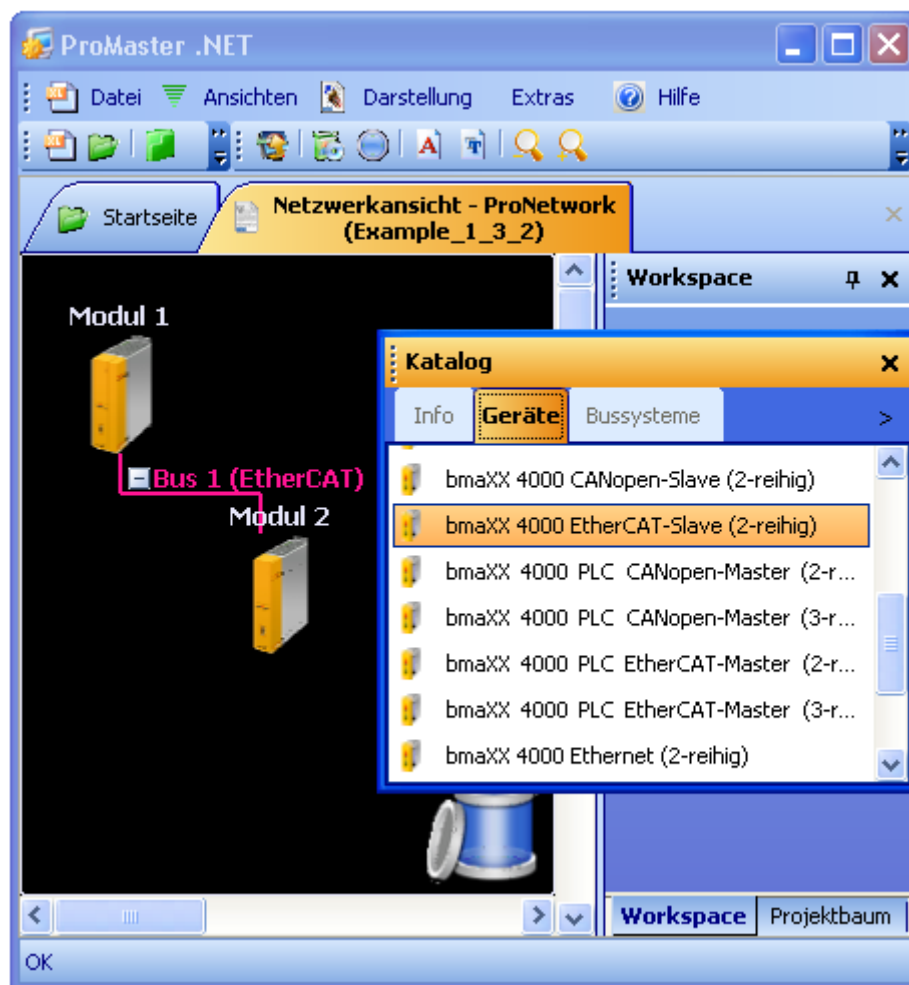


Abbildung 63: ProMaster - Maschinenkonfiguration angelegt

Im Bild sehen Sie jetzt den EtherCAT-Master (Modul 1) und den EtherCAT-Slave (Modul 2). Weitere Informationen zu den Modulen erhalten Sie über den jeweiligen Tooltip oder das "Projektbaum Fenster" (Project tree). Der Übersichtlichkeit halber wird jetzt das Fenster "Katalog" geschlossen.

Die Namen (Modul 1 und Modul 2) werden geändert indem man auf das jeweilige Modul klickt und dann über das Kontextmenü und „Eigenschaften“ das Eigenschaften-Fenster des jeweiligen Moduls öffnet.



Abbildung 64: ProMaster - Modul-Eigenschaften

Für unser Beispiel vergeben wir dem EtherCAT-Master den Namen `_Axis_A` und für den EtherCAT-Slave den Namen `_Axis_B` (`_Axis_B` ist der Default Achsvariablenname der Motion Control Achse in unserem IEC Projekt). Die jeweiligen Bilder (*.bmp) behalten wir. In der ProMaster Netzwerkansicht und im Projektbaum stehen nun die neuen Namen.

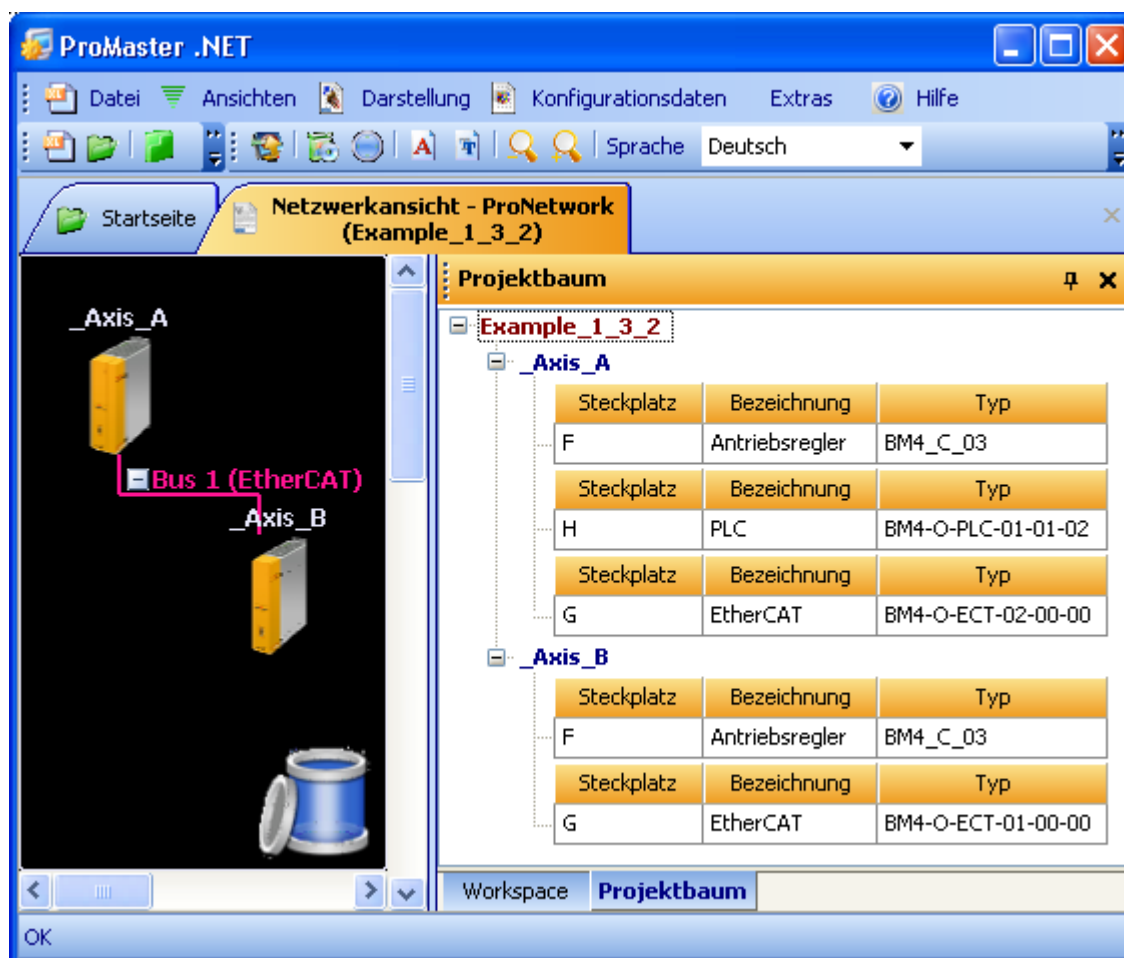


Abbildung 65: ProMaster - Maschinenkonfiguration - Module umbenannt

Die Kommunikationseinstellungen für die Kommunikation von ProMaster zur b maXX drive PLC können Sie einstellen in dem Sie auf das Gerät „_Axis_A“ klicken und über das Kontextmenü „Kommunikationseinstellungen“ anwählen. Speichern Sie anschließend das Projekt über Menü „Datei\Projekt speichern“.

In ProMaster ist für die lokale Achse sowie für die Komponenten mit EtherCAT (EtherCAT-Master, EtherCAT-Slave-Antrieb) per Default Motion Control aktiviert. Dadurch verringern sich die zu konfigurierenden Einstellungen auf ein Minimum.

Falls Sie die Default Einstellungen verwenden wollen, brauchen Sie keine Konfiguration der BACI-Kommunikation mit „Motion Control (lokale Achse)“ und der EtherCAT-Kommunikation mit ProEtherCAT vornehmen.

In diesem Fall geht es weiter mit [Konfigurieren der PLC](#) ab Seite 132.

5.2.6 Konfigurieren der EtherCAT Kommunikation mit ProEtherCAT

In ProMaster ist für die Komponenten mit EtherCAT (EtherCAT-Master, EtherCAT-Slave-Antrieb) per Default Motion Control aktiviert. Dadurch verringern sich die zu konfigurierenden Einstellungen auf ein Minimum.

Falls Sie die Default Einstellungen verwenden wollen, brauchen Sie keine Konfiguration der EtherCAT-Kommunikation mit ProEtherCAT vornehmen.

In diesem Fall geht es weiter mit [►Konfigurieren der PLC◄](#) ab Seite 132.

Wie Sie die Einstellungen der EtherCAT Kommunikation ändern können entnehmen Sie bitte den jeweiligen Applikationshandbüchern, in unserem Beispiel (mit EtherCAT) dem Applikationshandbuch EtherCAT-Master für b maXX drive PLC.

5.2.7 Konfigurieren der BACI-Kommunikation

Öffnen Sie das Fenster "Motion Control (Lokale Achse)" für unseren b maXX Regler (lokale Achse) in dem Sie auf das Gerät `_Axis_A` klicken und dann über das Kontextmenü "Konfigurationsdaten (Komponenten)\Regler[...]\Motion Control (Lokale Achse)" anwählen.

Die Lokale Achse (d. h. die BACI-Kommunikation zwischen b maXX Regler und b maXX drive PLC) wurde bereits beim Anlegen der Maschinenkonfiguration per Default so eingestellt, dass die BACI Einstellungen für Motion Control konfiguriert sind (Drag&Drop eines b maXX mit PLC).

Das Fenster "Motion Control (Lokale Achse)" wird geöffnet.

5.2.7.1 Register Ident

Im Register Ident werden Ihnen b maXX Regler spezifische Informationen wie (Regler-Tabellen-) Version, Regler-Typ (Controller), sowie die Regler Firmware Version angezeigt. Falls Ihr b maXX Regler eine andere als die eingestellte (Regler-Tabellen-) Version hat, können Sie in der Combo-Box Version die entsprechende (Regler-Tabellen-) Version auswählen.

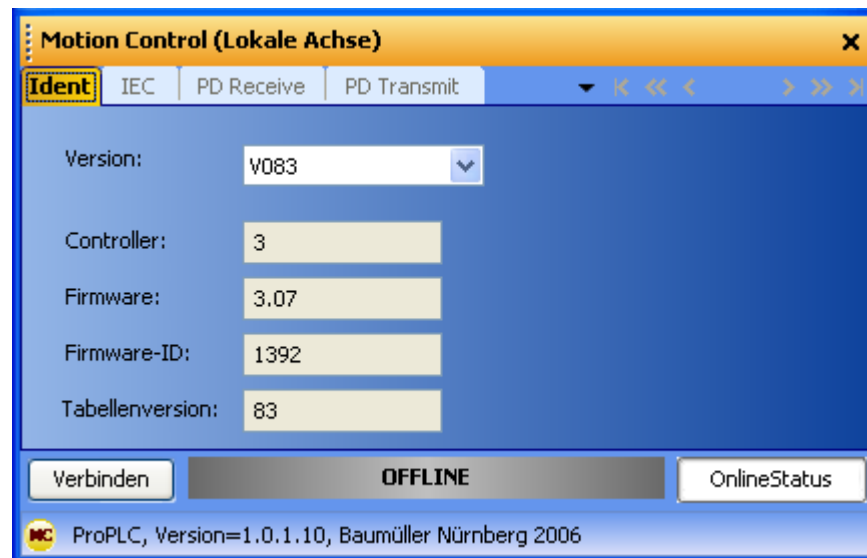


Abbildung 66: Motion Control (Lokale Achse) - Register Ident

5.2.7.2 Register IEC

Im Register IEC werden die Netzwerkvariablen angezeigt, die für die Kommunikation zwischen b maXX drive PLC und b maXX Regler verwendet werden. Die Netzwerkvariablen

unterscheiden sich in die Standard Netzwerkvariablen für Motion Control und, sofern zusätzliche Parameter in den Registern PD Receive und PD Transmit angelegt wurden, in die zusätzlichen Netzwerkvariablen für die BACI-Kommunikation.

Hierbei ist unbedingt folgendes zu beachten:

Die Standard Netzwerkvariablen für Motion Control dürfen vom Anwender gelesen, aber nicht geschrieben werden.

Dies sind die Standard Motion Control Netzwerkvariablen für die Sollwerte:

"w_controlword" für das Steuerwort der Achse
 "ud_PosIpSetAngle1" für den Gleichlauf Lage Winkel Sollwert der Achse

Dies sind die Standard Motion Control Netzwerkvariablen für die Istwerte:

"w_statusword" für das Statuswort der Achse
 "ud_Enc1ActAngle1" für den Gleichlauf Lage Winkel Istwert der Achse
 "i_OperationModeAct" für die Betriebsart der Achse



HINWEIS

Die Standard Netzwerkvariablen für Motion Control dürfen vom Anwender gelesen, aber nicht geschrieben werden.

Die Nummer hinter den Netzwerkvariablen ist eine interne Nummer um ggf. gleich lautende, automatisch generierte Netzwerkvariablen im IEC Projekt zu unterscheiden.

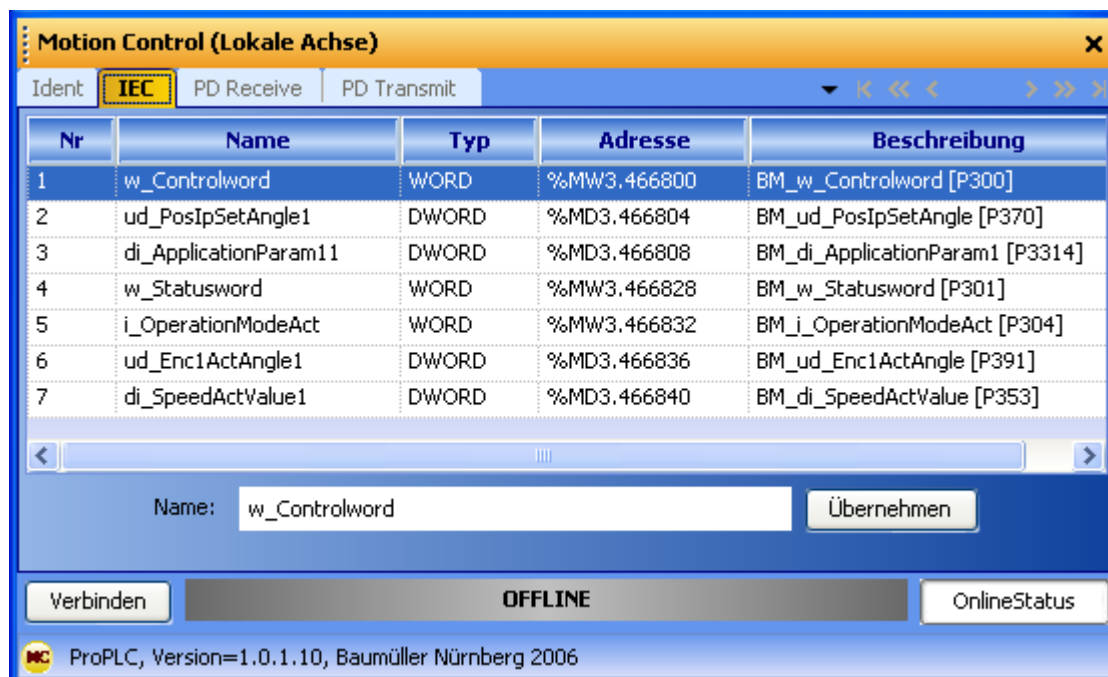


Abbildung 67: Motion Control (Lokale Achse) - Register IEC

Falls Sie bei der Konfiguration der BACI-Kommunikation in den Abschnitten [Register PD Receive](#) ab Seite 129 und [Register PD Transmit](#) ab Seite 130 die dort vorgeschlagenen zusätzlichen Netzwerkvariablen (und deren Verknüpfung zu den Parametern des b maXX Reglers angelegt haben, sehen Sie die zusätzlichen Netzwerkvariablen.

Die zusätzliche Netzwerkvariable für die Sollwerte ist:

"di_ApplicationParam1.." für den Applikationsparameter 1 der Achse

Die zusätzliche Netzwerkvariable für die Istwerte ist:

"di_SpeedActValue.." für den Drehzahl-Istwert der Achse

5.2.7.3 Register PD Receive

Im Register PD Receive wird die Kommunikation von der b maXX drive PLC zum b maXX Regler konfiguriert.

Für Motion Control ist die Default Konfiguration bereits eingestellt. Sie können die Konfiguration erweitern. Beachten Sie dabei, dass die Motion Control Konfiguration eine höhere Priorität hat als die Konfiguration des Anwenders.

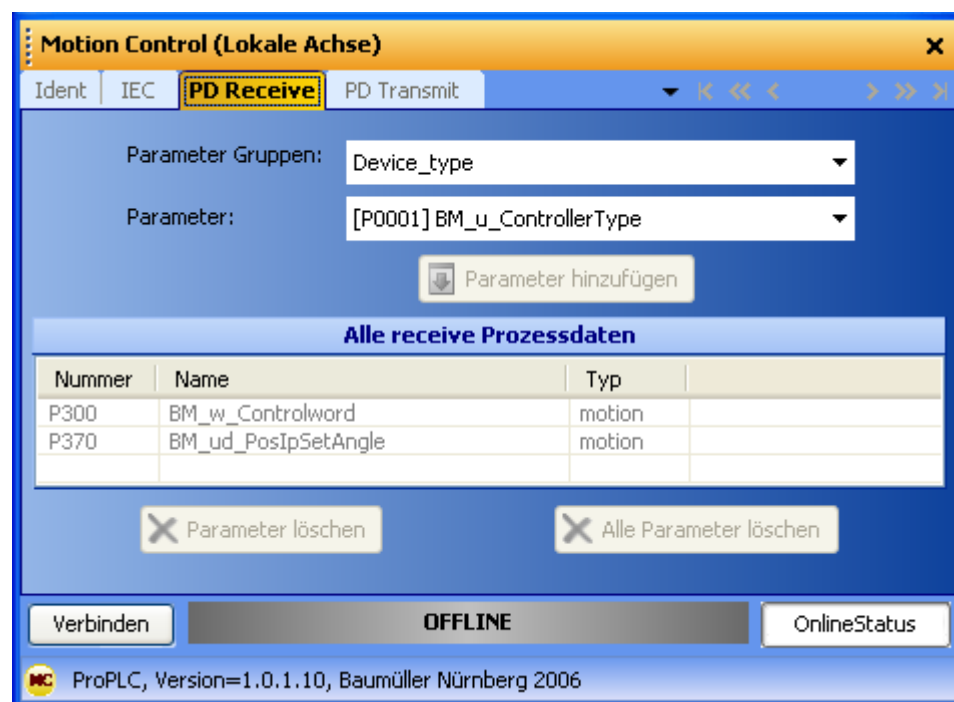


Abbildung 68: Motion Control (Lokale Achse) - Register PD Receive mit Default Motion Control Konfiguration

Hier findet die Verknüpfung eines Parameters im b maXX Regler mit einer Netzwerkvariablen im IEC Projekt statt.

Vorgehensweise:

- Auswahl einer Parameter Gruppe des b maXX Regler.
Beispiel: Wir wählen in der Combo-Box "Parameter Gruppe" die Gruppe "Configuration_Application" aus.
- Auswahl eines Parameters aus dieser Parameter Gruppe

Beispiel: Wir wählen in der Combo-Box "Parameter" den Parameter 3314 "[3314] BM_di_ApplicationParam1" aus.

- Mit dem Button "Parameter hinzufügen" wird die eben erstellte Verknüpfung auf Konformität zu den Einträgen überprüft und in die Liste eingetragen.

Der b maXX Regler Parameter 3314 (Applikationsparameter 1) ist jetzt mit einer Netzwerkvariablen im IEC Projekt verknüpft. In unserem Fall ist der Name der Netzwerkvariablen im IEC Projekt "di_ApplicationParam11". Im IEC Projekt wird auf diese Netzwerkvariable geschrieben. Der geschriebene Wert wird über die BACI-Kommunikation im Regler auf den Parameter 3314 (Applikationsparameter 1) geschrieben.

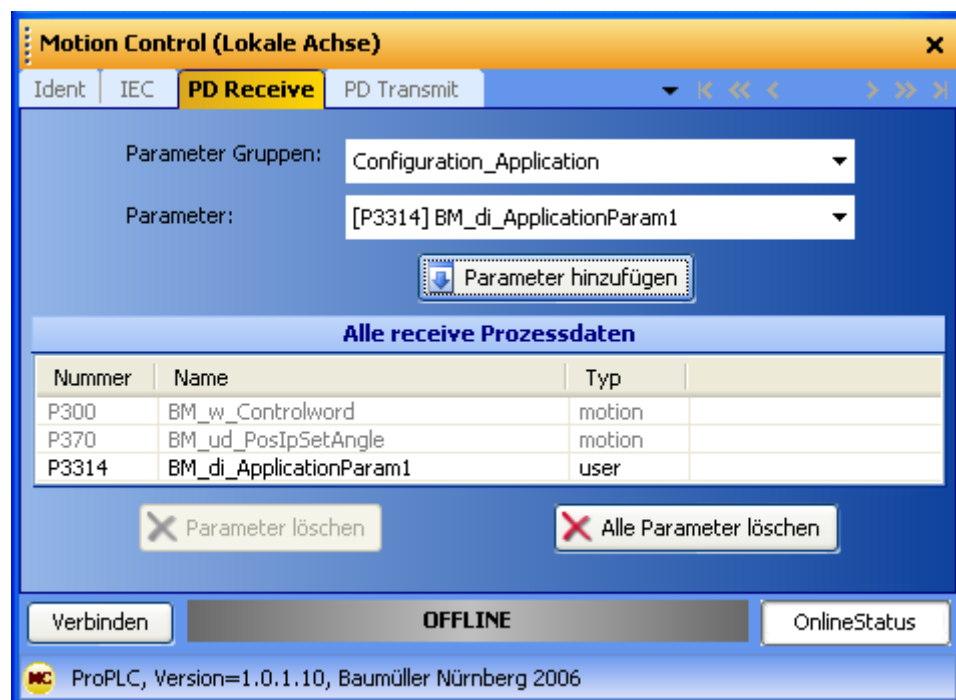


Abbildung 69: Motion Control (Lokale Achse) - Register PD Receive mit Default Motion Control Konfiguration und zusätzlichem Parameter



HINWEIS

ProMaster berücksichtigt automatisch die Grenze von 30 Zeichen für einen Variablennamen im IEC Projekt in ProProg wt III.

Die Namen der Variablen aus der zusätzlichen Konfiguration können im [Register IEC](#) ab Seite 127 geändert werden.

5.2.7.4 Register PD Transmit

Im Register PD Transmit wird die Kommunikation vom b maXX Regler zur b maXX drive PLC konfiguriert.

Für Motion Control ist die Default Konfiguration bereits eingestellt. Sie können die Konfiguration erweitern. Beachten Sie dabei, dass die Motion Control Konfiguration eine höhere Priorität hat als die Konfiguration des Anwenders.

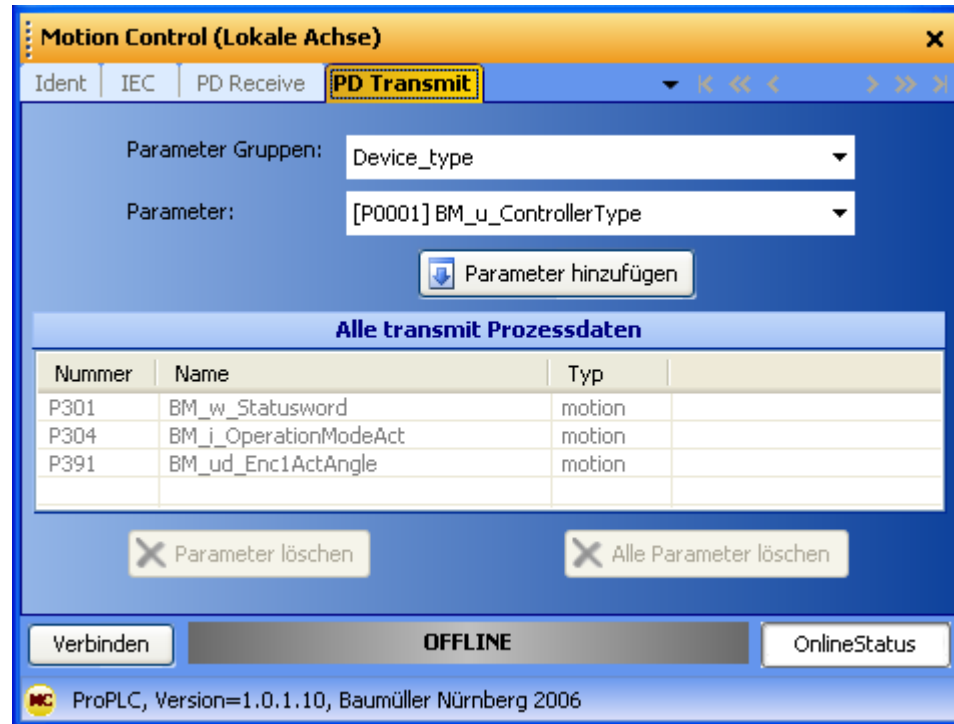


Abbildung 70: Motion Control (Lokale Achse) - Register PD Receive mit Default Motion Control Konfiguration

Hier findet die Verknüpfung eines Parameters im b maXX Regler mit einer Netzwerkvariablen im IEC Projekt statt.

Vorgehensweise:

- Auswahl einer Parameter Gruppe des b maXX Regler.
Beispiel: Wir wählen in der Combo-Box "Parameter Gruppe" die Gruppe "Speed_controller" aus.
- Auswahl eines Parameters aus dieser Parameter Gruppe
Beispiel: Wir wählen in der Combo-Box "Parameter" den Parameter 353 "[0353] BM_di_SpeedActValue" aus.
- Mit dem Button "Parameter hinzufügen" wird die eben erstellte Verknüpfung auf Konformität zu den Einträgen überprüft und in die Liste eingetragen.

Der b maXX Regler Parameter 353 (Drehzahl-Istwert) ist jetzt mit einer Netzwerkvariablen im IEC Projekt verknüpft. In unserem Fall ist der Name der Netzwerkvariablen im IEC Projekt "di_SpeedActValue1". Im IEC Projekt wird diese Netzwerkvariablen gelesen. Der Wert wird über die BACI-Kommunikation vom Regler Parameter 353 (Drehzahl-Istwert) gelesen.

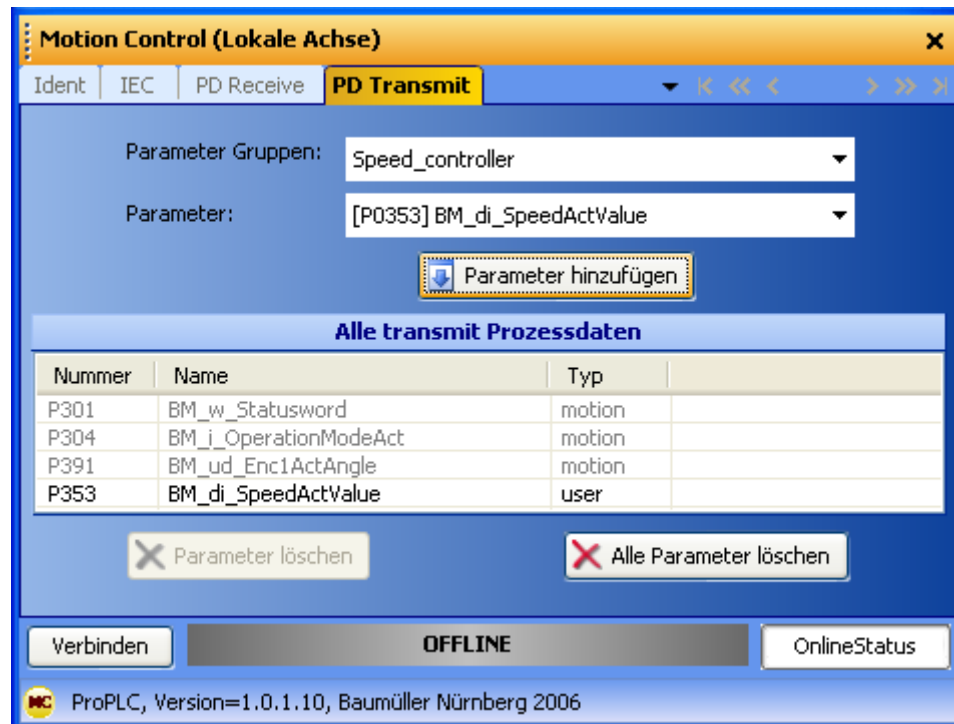


Abbildung 71: Motion Control (Lokale Achse) - Register PD Receive mit Default Motion Control Konfiguration und zusätzlichem Parameter



HINWEIS

ProMaster berücksichtigt automatisch die Grenze von 30 Zeichen für einen Variablennamen im IEC Projekt in ProProg wt III.

Die Namen der Variablen aus der zusätzlichen Konfiguration können im [Register IEC](#) ab Seite 127 geändert werden.

5.2.8 Konfigurieren der PLC

Nach der Konfiguration der BACI-Kommunikation, der Kommunikation des EtherCAT-Netzwerks und der Erstellung des IEC Projekts werden nun die Daten für die b maXX drive PLC konfiguriert.

Dabei werden

- das IEC Projekt in das ProMaster Projekt eingebunden
- die Kurvendatensätze ausgewählt.

Außerdem können

- mit dem Kurvenscheiben Editor ProCAM eigene Kurvenscheibendaten erstellt werden

Anschließend klicken Sie im Fenster "ProPLC" auf den Button "Gesamtes IEC-Projekt aktualisieren". Die Daten für die Maschinenkonfiguration, sowohl für das EtherCAT-Netzwerk, als auch für die b maXX drive PLC und die lokale Achse werden dann erzeugt.



HINWEIS

Die Netzwerkvariablen aus der Konfiguration der BACI-Kommunikation (Lokale Achse) werden in ProPLC nicht angezeigt.

Über den Button "Gesamtes IEC Projekt aktualisieren" werden auch diese Netzwerkvariablen in das globale Variablen Arbeitsblatt des IEC Projekts in ProProg wt III geschrieben (in den Abschnitt LocalAxisVariables).

5.2.8.1 IEC

In diesem Abschnitt erläutern wir, wie ein IEC Projekt mit ProMaster verbunden wird.

Öffnen Sie im ProMaster Projekt in der Netzwerkansicht für unsere lokale Achse (und EtherCAT-Master) das Fenster "ProPLC" indem Sie auf das Gerät „_Axis_A“ klicken und dann über das Kontextmenü "Konfigurationsdaten (Komponenten)\PLC (Steckplatz H)\PLC - Konfiguration (ProPLC)" anwählen und dort das Register "IEC" auswählen.

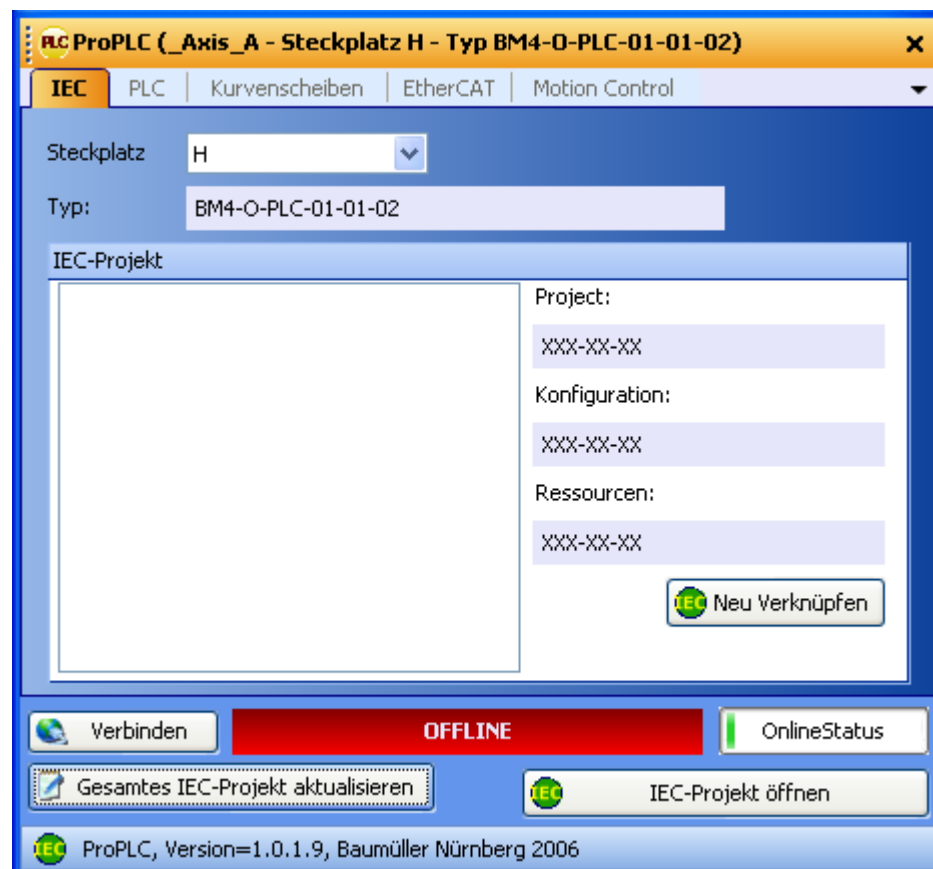


Abbildung 72: ProPLC - Verbinden mit Beispiel IEC Projekt



HINWEIS

Mit ProMaster können nur ProProg wt III Projekte geöffnet werden. Falls Sie ein bestehendes PROPROG wt II Projekt verwenden wollen, müssen Sie dieses zuerst mit ProProg wt III öffnen (und dabei konvertieren) und können es danach in ProMaster öffnen und verwenden.

Beachten Sie, dass dabei Ihre PROPROG wt II-Bibliotheken ebenfalls konvertiert werden!

Klicken Sie auf den Button "Neu Verknüpfen" und wählen Sie unser Beispiel IEC Projekt "Example_BM4_O_ECT02_MA_2.mwt", welches wir in Kapitel [5.2.4 Anlegen eines IEC Projekts mit ProProg wt III](#) ab Seite 119 angelegt haben, aus.

Im Register "IEC" wird das eben ausgewählte IEC Projekt angezeigt.

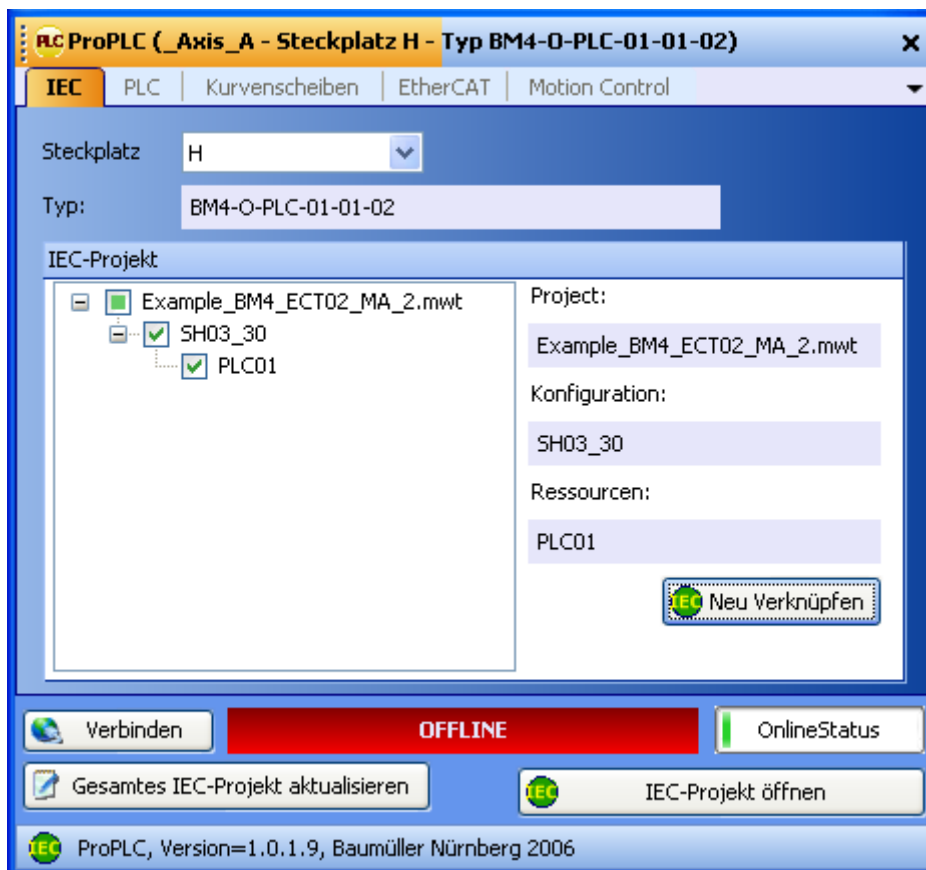


Abbildung 73: ProPLC - Register IEC

Falls Sie eine vorhandene Applikation (oder eine Vorlage) aus ProProg wt III mit dem ProMaster Projekt verbinden, kann es sein, dass die Namen der Geräte in ProMaster und im IEC Projekt nicht gleich sind. In diesem Fall siehe Kapitel [5.2.4 Anlegen eines IEC Projekts](#) ab Seite 142.

5.2.8.2 PLC

Nach dem Download der Daten auf EtherCAT-Master und b maXX drive PLC, siehe [►Download der Daten auf den EtherCAT-Master und die b maXX drive PLC◄](#) ab Seite 141, kann im Register PLC das IEC Projekt auf der b maXX drive PLC aktiviert und gestartet werden (Bereich "PLC Status").

Weiterhin werden im Bereich "PLC Info" Daten der b maXX drive PLC wie Betriebssystem-Version, Firmware-Version, IEC Projekt und Boot Projekt angezeigt. Im Bereich "Flash Directory" werden (nach "Verbinden" und "Aktualisieren") Informationen zu den Dateien im Flash-Speicher der b maXX drive PLC angezeigt, z. B. zu den Kurvendatensätzen (siehe [►5.2.8.3. Kurvenscheiben◄](#))

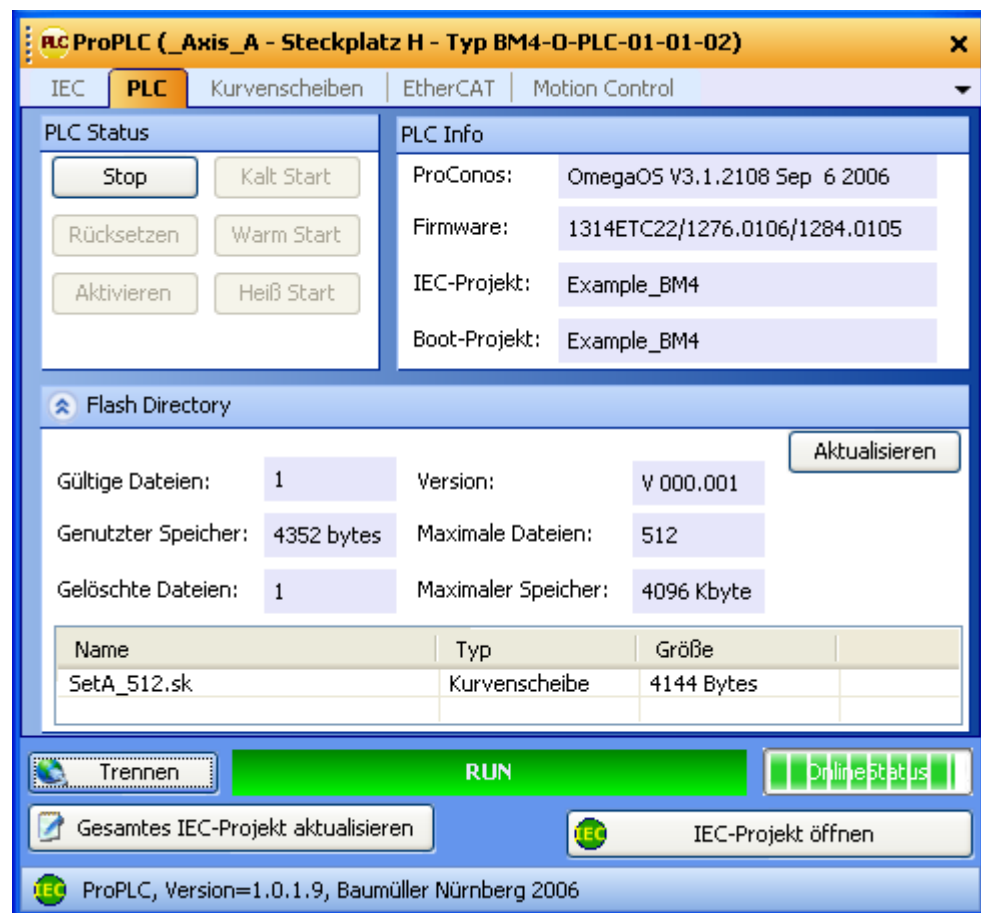


Abbildung 74: ProPLC - Register PLC nach Download der Kurvenscheibe (Register Kurvenscheibe)

5.2.8.3 Kurvenscheiben

In diesem Abschnitt erläutern wir, wie ein Kurvenscheibendatensatz erstellt und mit Pro-Master verbunden wird.

Öffnen Sie im ProMaster Projekt in der Netzwerkansicht für unsere lokale Achse (und EtherCAT-Master) das Fenster "PLC Konfiguration" in dem Sie auf das Gerät „_Axis_A“ klicken und dann über das Kontextmenü "Konfigurationsdaten (Komponenten)\PLC (Steckplatz H)\ PLC - Konfiguration (ProPLC)" anwählen und dort das Register "Kurvenscheiben" auswählen.

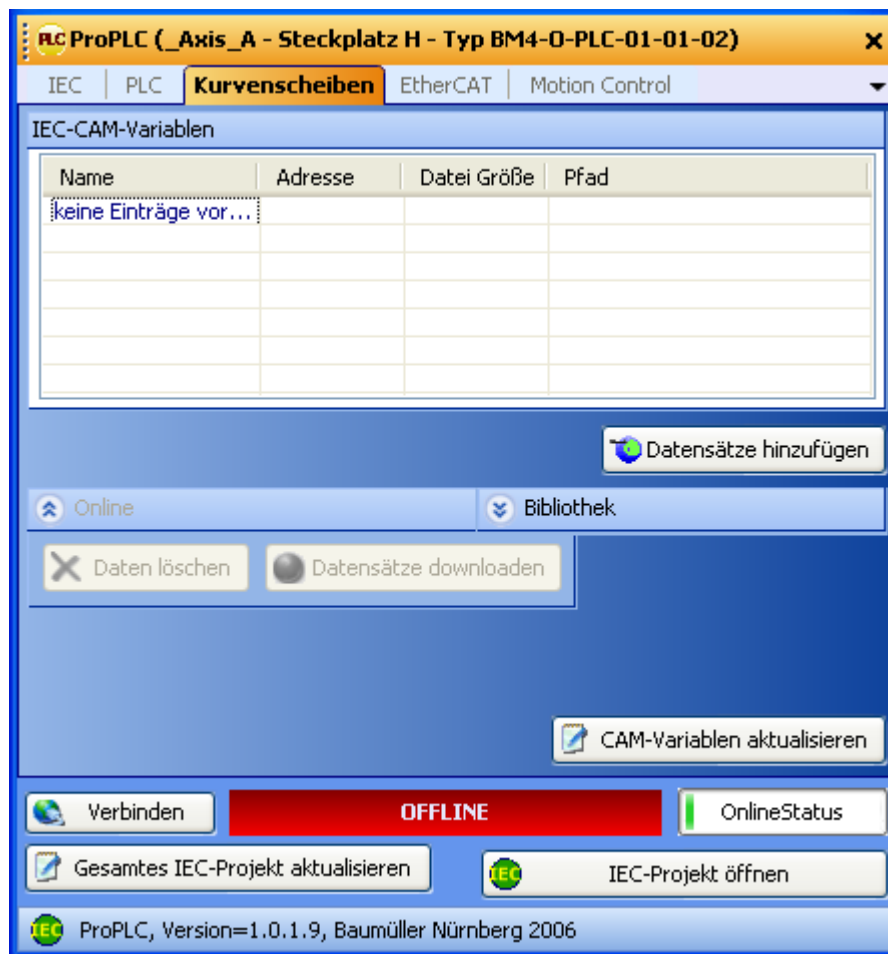


Abbildung 75: ProPLC - Register „Kurvenscheiben“ ohne Kurvendatensatz

Drücken Sie den Button "Datensätze hinzufügen". Das Fenster "ProMaster.NET - Kurvenscheiben" wird geöffnet.

Drücken Sie im Bereich "Festplatte" den Button "Hinzufügen" und wählen Sie auf Ihrer Festplatte den Pfad aus, in dem Ihre Kurvendaten stehen.



HINWEIS

Beispielkurvendaten finden Sie im Installationsverzeichnis von ProCAM im Unterordner "examples" (z. B. C:\...\Baumüller\ProCam 2\examples).

Falls Sie eigen Kurven generieren wollen, klicken Sie auf den Button "ProCAM" und öffnen damit den Kurvenscheiben Editor. Dort können Sie Ihre Kurven editieren.

Stellen Sie jetzt per Drag&Drop Ihren Kurvendatensatz (im Bereich "*.sk Dateien") aus den verschiedenen Kurvenscheibendaten (im Bereich "Festplatte") zusammen.

Für unser Beispiel wählen wir die Kurvenscheibendaten "ExampleGerade_MC.kbin" und "ExamplePendelP5_MC.kbin", die in den Kurvenscheibendatensatz „SetA_512“ zusammengefasst werden.



HINWEIS

Sie können sich die Kurvenscheibendaten (im jeweiligen Bereich) über den Button "Vorschau" grafisch darstellen lassen. Dies erleichtert die Kurvenauswahl.

Klicken Sie anschließend auf den Button "Speichern". Dadurch wird der Kurvendatensatz erstellt und ProMaster zur Verfügung gestellt.

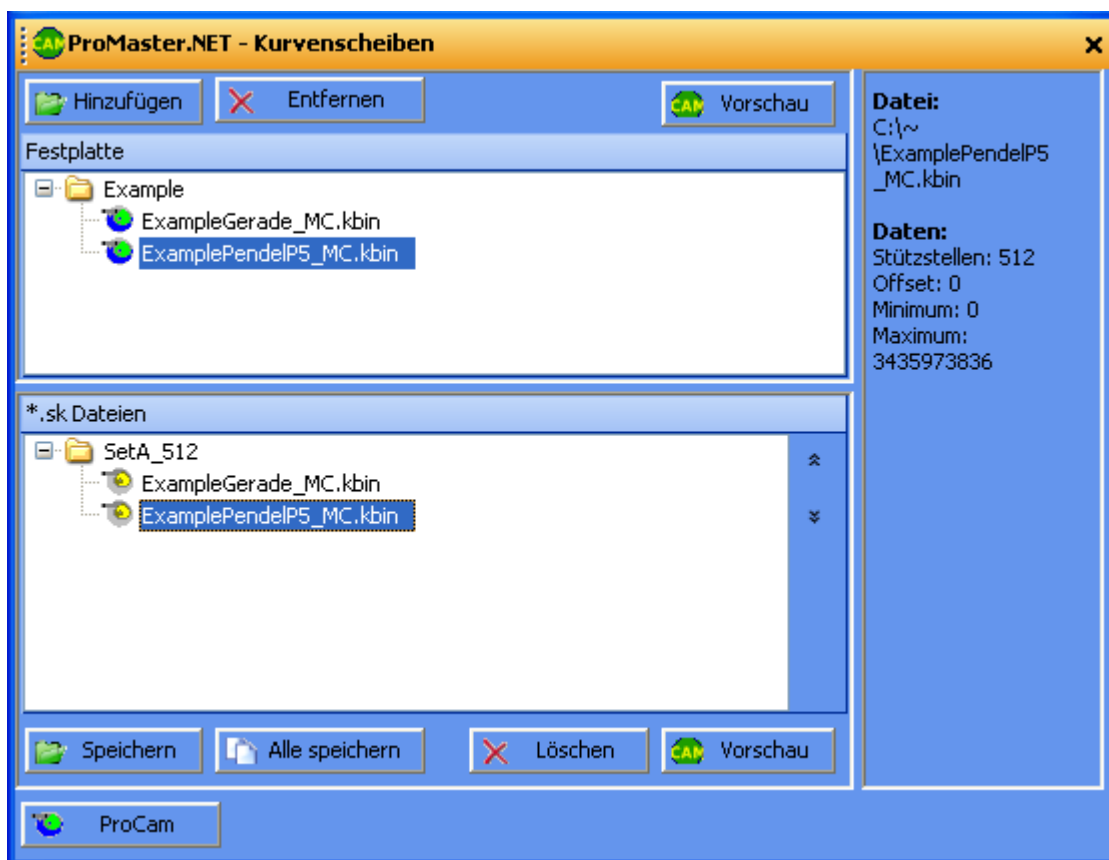


Abbildung 76: ProPLC (- Register Kurvenscheiben -) Kurvendatensatz erstellen

Schließen Sie jetzt das Fenster "ProMaster.NET - Kurvenscheiben", indem Sie oben rechts auf "x" klicken.

Im Register "Kurvenscheiben" sehen Sie jetzt den Namen der Variablen des Kurvenscheibendatensatzes im IEC Projekt, ihre Adresse im IEC Projekt und den Dateinamen des Kurvenscheibendatensatzes auf der Festplatte.

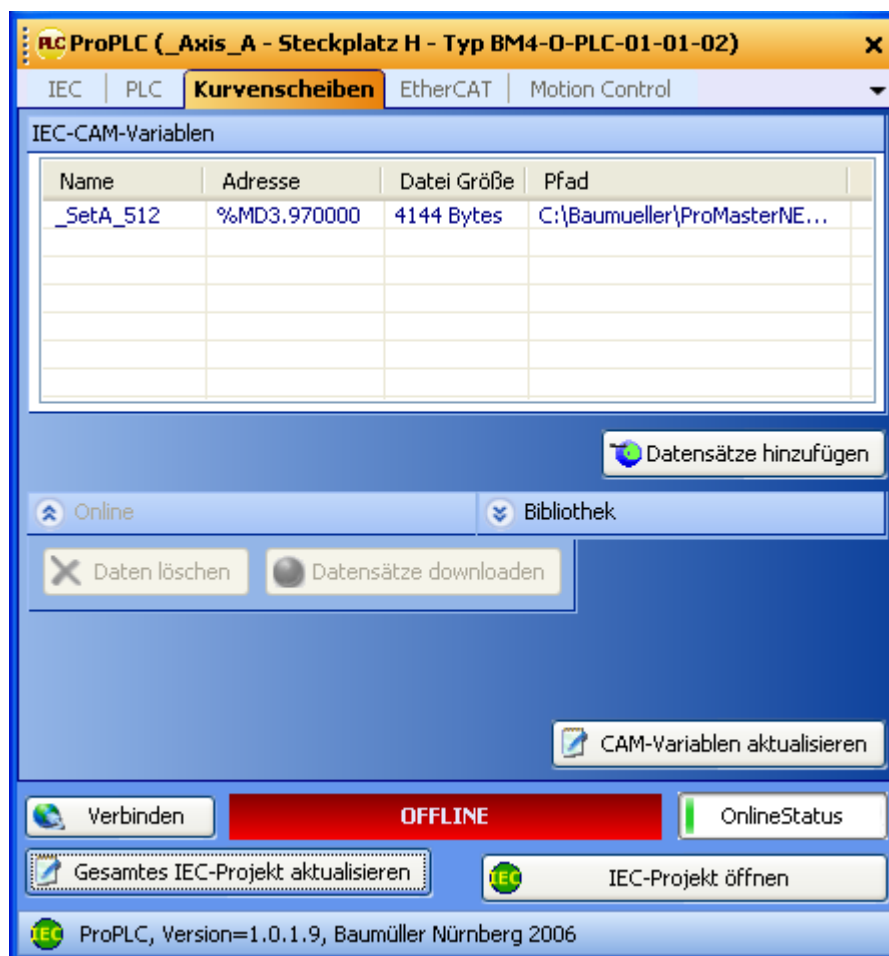


Abbildung 77: ProPLC - Register „Kurvenscheiben“ mit Kurvendatensatz

Falls Sie eine vorhandene Applikation (oder eine Vorlage) aus ProProg wt III mit dem ProMaster Projekt verbinden, kann es sein, dass der Kurvendatensatzname in ProMaster und im IEC Projekt nicht gleich sind. In diesem Fall siehe Kapitel [►Programmieren des IEC Projekts](#) ab Seite 142.

Sie können vom Register "Kurvenscheiben" aus, über den Button "Flash löschen" (Bereich "↓ Online"), den Flash-Speicher auf der b maXX drive PLC löschen. Dies kann notwendig sein, falls durch diverse Downloads (sowohl Kurvenscheibendatensätze als auch IEC Projekte) der Flash-Speicher "voll" ist.

Weiterhin können Sie die Kurvenscheibendatensätze vom Register "Kurvenscheiben" aus, über den Button "Datensätze downloaden", auf die b maXX drive PLC einzeln downloaden.

Laden Sie jetzt die Kurvenscheibendatensätze über den Button „Datensätze downloaden“ (Bereich "↓ Online") auf die b maXX drive PLC.

5.2.8.4 EtherCAT

Im Register "EtherCAT" werden der Name, der Datentyp, die Adresse und der Kommentar der Netzwerkvariablen der EtherCAT-Slaves im IEC Projekt dargestellt.

Bei der Verwendung von Motion Control erfolgt die Kommunikation über die Achsvariable. ProMaster legt nach der Betätigung des Buttons „gesamtes IEC-Projekt aktualisieren“ siehe Kapitel [►Download der Daten auf den EtherCAT-Master und die b maXX drive PLC◄](#) ab Seite 141 im IEC Projekt die Achsvariable mit dem Gerätenamen des EtherCAT-Slave neu an. Zusätzlich legt ProMaster im IEC Projekt Netzwerkvariablen an. Diese unterscheiden sich in die Standard Netzwerkvariablen für Motion Control und, sofern zusätzliche Objekte bei der EtherCAT-Slave Konfiguration angelegt wurden, in zusätzliche Netzwerkvariablen für EtherCAT.

Im Register "EtherCAT" werden Ihnen diese Netzwerkvariablen, sowohl die Standard Netzwerkvariablen für Motion Control als auch die zusätzlichen Netzwerkvariablen für EtherCAT, angezeigt.

Hier ist unbedingt folgendes zu beachten:

Die Standard Netzwerkvariablen für Motion Control dürfen vom Anwender gelesen, aber nicht geschrieben werden.

Dies sind die Standard Motion Control Netzwerkvariablen für die Sollwerte:

"u_controlword.."	für das Steuerwort der Achse
"ud_PoslpSetAngel.."	für den Gleichlauf Lage Winkel Sollwert der Achse

Dies sind die Standard Motion Control Netzwerkvariablen für die Istwerte:

"u_statusword.."	für das Statuswort der Achse
"ud_Enc1ActAngle.."	für den Gleichlauf Lage Winkel Istwert der Achse
"si_modes_of_operation_display.."	für die Betriebsart der Achse



HINWEIS

Die Standard Netzwerkvariablen für Motion Control dürfen vom Anwender gelesen, aber nicht geschrieben werden.



HINWEIS

Die Standard Netzwerkvariablen müssen in der Motion Control Event-Task gelesen werden.

Die Nummer hinter den Netzwerkvariablennamen ist eine interne Nummer um gleich laufende, automatisch generierte Netzwerkvariablennamen zu unterscheiden.

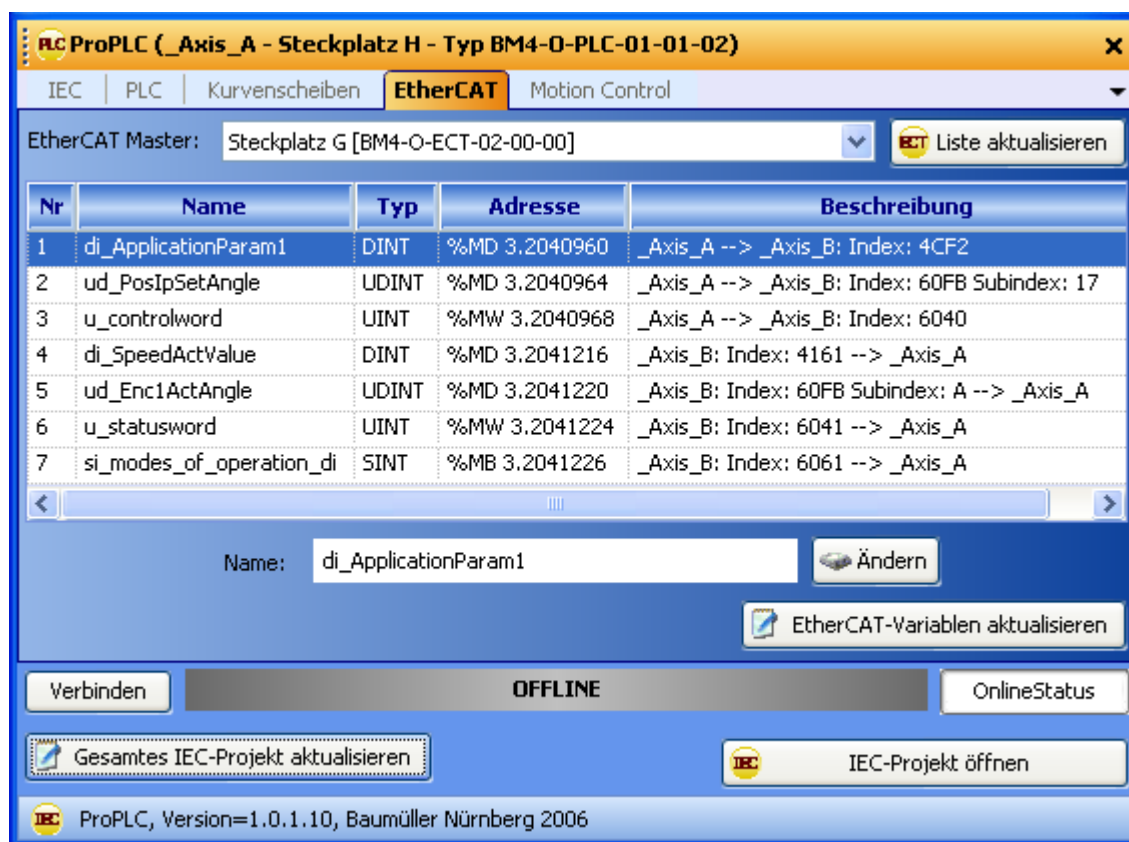


Abbildung 78: ProPLC - Register EtherCAT

Falls Sie bei der Konfiguration der EtherCAT-Slave Kommunikation im Abschnitt [Konfigurieren der EtherCAT Kommunikation mit ProEtherCAT](#) ab Seite 126 zusätzliche Netzwerkvariable (und deren Verknüpfung zu EtherCAT-Slaves) angelegt haben, sehen Sie die zusätzlichen Netzwerkvariablen.

Z. B. ist eine zusätzliche Netzwerkvariable für die Sollwerte:

"di_ApplicationParam1.." für den Applikationsparameter 1 der Achse

Z. B. ist eine zusätzliche Netzwerkvariable für die Istwerte:

"di_SpeedActValue.." für den Drehzahl-Istwert der Achse

Beachten Sie bei der Änderung von Netzwerkvariablenamen, dass jeder Variablenname nur einmal im IEC Projekt vergeben sein kann. Dies ist insbesondere bei der Verwendung mehrerer Optionsmodule EtherCAT-Master an der b maXX drive PLC zu beachten.



HINWEIS

Jeder Variablenname darf nur einmal im IEC Projekt vergeben werden.



HINWEIS

Die zusätzlichen Netzwerkvariablen müssen in der Motion Control Event-Task gelesen und geschrieben werden.

Weiter Informationen entnehmen Sie bitte dem Applikationshandbuch EtherCAT-Master für b maXX drive PLC.

5.2.8.5 Motion Control

Im Register "Motion Control" werden zukünftig die allgemeinen Einstellungen für Motion Control konfiguriert.

Wenn Sie die Default Motion Control Einstellungen verwenden wollen, brauchen Sie hier keine Einstellungen vornehmen.

5.2.9 Download der Daten auf den EtherCAT-Master und die b maXX drive PLC

Wie Sie den Download der Daten auf den EtherCAT-Master durchführen entnehmen Sie bitte dem Applikationshandbuch EtherCAT-Master für b maXX drive PLC.

Der Download der Kurvendatensätze für die b maXX drive PLC erfolgt zur Zeit über Register „Kurvenscheiben“ der PLC-Konfiguration. Siehe hierzu Kapitel [Kurvenscheiben](#) ab Seite 135.

Der Download des IEC-Projektes für die b maXX drive PLC erfolgt zur Zeit wie gewohnt über ProProg wt III.

Klicken Sie jetzt im Fenster "PLC Konfiguration" auf den Button "Gesamtes IEC-Projekt aktualisieren". Die Daten für die b maXX drive PLC werden jetzt erzeugt. Dieser Vorgang kann etwas länger dauern, da unter anderem ProProg wt III geöffnet wird und im IEC-Projekt die konfigurierten IEC-Variablen in das globale Variablenarbeitsblatt geschrieben werden.

Anschließend können Sie das IEC-Projekt kompilieren indem Sie die entsprechende Abfrage mit „Ja“ bestätigen.

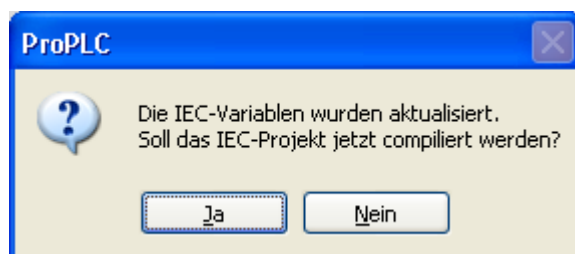


Abbildung 79: ProMaster - aktualisiertes IEC-Projekt kompilieren

Alternativ können Sie das IEC-Projekt auch in ProProg wt III über das ProProg wt III Menü „Code\Projekt neu erzeugen“ kompilieren.

Führen Sie jetzt den Download des IEC-Projektes auf die b maXX drive PLC durch (ProProg wt III - Menü „Online\Projektkontrolle...“ → „Senden“ → Bootprojekt „Senden“).

Führen Sie jetzt an der b maXX drive PLC einen Reset aus und schalten Sie danach die PLC in den Zustand „RUN“ (Alternativ können Sie auch das gesamte EtherCAT-Netzwerk aus- und wieder einschalten). Jetzt können Sie über das IEC-Projekt in ProProg wt III die lokale Achse `_Axis_A` und die EtherCAT-Slave Achse `_Axis_B` steuern.

5.2.10 Programmieren des IEC Projekts

5.2.10.1 Allgemeines

Wie Sie eine Motion Control Applikation im IEC Projekt in ProProg wt III programmieren können entnehmen Sie bitte dem Applikationshandbuch Motion Control und der Online-Hilfe von ProProg wt III.

Über das Kontextmenü auf `_Axis_A` "Konfigurationsdaten (Komponenten)\PLC (Steckplatz H)\IEC - Programmierung (ProProg wt III)" wird ProProg wt III mit unserem IEC Projekt "Example_BM4_O_ECT02_MA_1.mwt" geöffnet. Sie können das IEC Projekt wie gewohnt bearbeiten.

ProMaster hat

- die Motion Control Achsvariable (in den Abschnitt MC_Axis_Variables)
- den Kurvenscheibendatensatz (in den Abschnitt MC_CamDataSet)
- die Netzwerkvariablen für EtherCAT (in den Abschnitt EtherCATVariables)
- die Netzwerkvariablen der lokalen Achse (in den Abschnitt LocalAxisVariables)
(Standard Netzwerkvariablen für Motion control und zusätzliche Netzwerkvariablen der lokalen Achse; siehe [►5.2.10.2 Datenaustausch◄](#) ab Seite 144)

in das globale Variablenarbeitsblatt "Global_Variables" geschrieben.

Global_Variables:SH03_30.PLC01					
Name	Typ	Verwendung	Beschreibung	Adresse	
Global_Variables					
MyApplication					
_MyMaster	AXI...	VAR_GLO...	Not from ProMaster		
MC_CamDataSet					
s_File_SetA_512	FileS...	VAR_GLO...		%MD3.970000	
_SetA_512	MC_...	VAR_GLO...		%MD3.970000	
MC_AxisVariables					
_Axis_A	AXI...	VAR_GLO...	1. Achse	%MD3.451500	
_Axis_B	AXI...	VAR_GLO...	2. Achse	%MD3.457300	
MC_MasterRef					
MC_SystemReserved					
MC_AxisRef					
MC_BacInitRef					
MC_AxisPDORef					
MC_TriggerRef					
MC_IRPRef					
EtherCAT Variables					
LocalAxisVariables					
w_Controlword	WORD	VAR_GLO...	BM_w_Controlword [P300]	%MW3.466800	
ud_PoslpSetAngle1	UDINT	VAR_GLO...	BM_ud_PoslpSetAngle [P370]	%MD3.466804	
di_ApplicationParam11	DINT	VAR_GLO...	BM_di_ApplicationParam1 [P3314]	%MD3.466808	
w_Statusword	WORD	VAR_GLO...	BM_w_Statusword [P301]	%MW3.466828	
i_OperationModeAct	INT	VAR_GLO...	BM_i_OperationModeAct [P304]	%MW3.466832	
ud_Enc1ActAngle1	UDINT	VAR_GLO...	BM_ud_Enc1ActAngle [P391]	%MD3.466836	
di_SpeedActValue1	DINT	VAR_GLO...	BM_di_SpeedActValue [P353]	%MD3.466840	

Abbildung 80: Globales Variablenarbeitsblatt mit den Daten von ProMaster

Besonders muss folgendes beachtet werden:

Der Gerätenamen der Geräte am EtherCAT-Bus (lokale Achse/EtherCAT-Master, EtherCAT-Slave(s)) in ProMaster ist gleichzeitig der Name der Achsen im IEC Projekt in ProProg wt III. D. h. falls Sie eine vorhandene Applikation (oder eine Vorlage) aus ProProg wt III mit dem ProMaster Projekt verbunden haben, kann es sein, dass die Namen der Geräte in ProMaster und im IEC Projekt nicht gleich sind. In diesem Fall gibt es zwei Lösungsmöglichkeiten:

- 1 ProMaster ändert in ProProg wt III im globalen Variablenarbeitsblatt "Global_Variables", im Abschnitt "MC_AxisVariables", die Achsnamen auf die Gerätenamen in ProMaster und der Anwender ändert (über die ProProg wt III Funktion "Globales Ersetzen") die Achsnamen in den POEs auf die Gerätenamen in ProMaster.
- 2 Sie ändern die Gerätenamen in ProMaster auf die Achsnamen in ProProg wt III.

In unserem Beispiel haben wir in ProMaster der lokalen Achse den Gerätenamen _Axis_A und dem EtherCAT-Slave den Gerätenamen _Axis_B gegeben. Dies sind auch die Achsnamen aus unserer Motion Control Vorlage, die wir für unser IEC-Projekt verwendet haben.

Ähnliches gilt auch für den Kurvendatensatznamen im IEC Projekt in ProProg wt III. D. h. falls Sie eine vorhandene Applikation (oder eine Vorlage) aus ProProg wt III mit dem Pro-

Master Projekt verbinden, kann es sein, dass Kurvendatensatzname in ProMaster und im IEC Projekt nicht gleich sind. In diesem Fall gilt folgendes:

ProMaster ändert in ProProg wt III im globalen Variablenarbeitsblatt "Global_Variables", im Abschnitt "MC_CamDataSet", die Kurvendatensatznamen auf die Kurvendatensatznamen in ProMaster und der Anwender ändert (über die ProProg wt III Funktion "Globales Ersetzen") die Kurvendatensatznamen in den POEs auf die Kurvendatensatznamen in ProMaster.

5.2.10.2 Datenaustausch

Für den Datenaustausch zwischen

- den Funktionsbausteinen im IEC Projekt auf der b maXX drive PLC (BM4-O-PLC-0x) und der lokalen Achse
- den Funktionsbausteinen im IEC Projekt auf der b maXX drive PLC (BM4-O-PLC-0x), dem Optionsmodul EtherCAT-Master (BM4-O-ETH-02) und (über den EtherCAT-Feldbus mit) dem EtherCAT-Slave

wird jeweils eine globale Variable benötigt. Diese Variablen werden auch Achsvariablen genannt und haben im IEC-Projekt einen Achsnamen.

Unsere Achsvariable für die lokale Achse hat den Achsnamen `_Axis_A`.

Unsere Achsvariable für die EtherCAT-Slave Achse hat den Achsnamen `_Axis_B`.

Wir haben in ProMaster unserer lokalen Achse den Gerätenamen `_Axis_A` und unserem EtherCAT-Slave den Gerätenamen `_Axis_B` gegeben, deshalb brauchen wir hier keine Anpassung vornehmen.

Das Gleiche gilt für die Variable für die Kurvendaten, die den Kurvendatensatznamen `_SetA_512` in IEC Projekt und ProMaster hat.

Die Achsvariable und die Kurvendatenvariable werden im IEC Projekt an den Motion Control Funktionsbausteinen angeschlossen. Mit den Motion Control Funktionsbausteinen wird die Maschinenfunktion programmiert.

Hinweise zur Verwendung der Motion Control Funktionsbausteine finden Sie im Applikationshandbuch Motion Control.

Bei der Verwendung von Motion Control erfolgt die Kommunikation über die Achsvariable. ProMaster legt im IEC Projekt Netzwerkvariablen an. Diese unterscheiden sich in die Standard Netzwerkvariablen für Motion Control und, sofern zusätzliche Objekte bei der Konfiguration der BACI-Kommunikation (lokale Achse) angelegt wurden, in zusätzliche Netzwerkvariablen für die lokale Achse.

Hier ist unbedingt folgendes zu beachten:

Die Standard Netzwerkvariablen für Motion Control dürfen vom Anwender gelesen, aber nicht geschrieben werden.

`_Axis_A` (lokale Achse):

Dies sind die Standard Motion Control Netzwerkvariablen für die Sollwerte:

<code>"w_controlword"</code>	für das Steuerwort der Achse
<code>"ud_PoslpSetAngle1"</code>	für den Gleichlauf Lage Winkel Sollwert der Achse

Dies sind die Standard Motion Control Netzwerkvariablen für die Istwerte:

"w_statusword"	für das Statuswort der Achse
"ud_Enc1ActAngle1"	für den Gleichlauf Lage Winkel Istwert der Achse
"i_OperationModeAct"	für die Betriebsart der Achse

Die zusätzlichen Netzwerkvariablen für die lokale Achse werden vom Anwender geschrieben (Sollwerte) und gelesen (Istwerte).

Falls Sie bei der Konfiguration der BACI-Kommunikation in den Abschnitten [►Register PD Receive◄](#) ab Seite 129 und [►Register PD Transmit◄](#) ab Seite 130 die dort vorgeschlagenen zusätzlichen Netzwerkvariablen (und deren Verknüpfung zu den Parametern des b maXX Reglers angelegt haben, ist die zusätzliche Netzwerkvariable für die Sollwerte:

"di_ApplicationParam1.." für den Applikationsparameter 1 der Achse

Die zusätzliche Netzwerkvariable für die Istwerte ist:

"di_SpeedActValue.." für den Drehzahl-Istwert der Achse

_Axis_B (EtherCAT-Slave Achse):

Dies sind die Standard Motion Control Netzwerkvariablen für die Sollwerte:

"u_controlword.."	für das Steuerwort der Achse
"ud_PosIpSetAngel.."	für den Gleichlauf Lage Winkel Sollwert der Achse

Dies sind die Standard Motion Control Netzwerkvariablen für die Istwerte:

"u_statusword.."	für das Statuswort der Achse
"ud_Enc1ActAngle.."	für den Gleichlauf Lage Winkel Istwert der Achse
"si_modes_of_operation_display.."	für die Betriebsart der Achse

Falls Sie bei der Konfiguration der EtherCAT-Slave Kommunikation im Abschnitt [►Konfigurieren der EtherCAT Kommunikation mit ProEtherCAT◄](#) ab Seite 126 zusätzliche Netzwerkvariable (und deren Verknüpfung zu EtherCAT-Slaves) angelegt haben, sehen Sie die zusätzlichen Netzwerkvariablen.

Z. B. ist eine zusätzliche Netzwerkvariable für die Sollwerte:

"di_ApplicationParam1.." für den Applikationsparameter 1 der Achse

Z. B. ist eine zusätzliche Netzwerkvariable für die Istwerte:

"di_SpeedActValue.." für den Drehzahl-Istwert der Achse

Die Nummer hinter den Netzwerkvariablenamen ist eine interne Nummer um gleich lautende, automatisch generierte Netzwerkvariablenamen zu unterscheiden.



HINWEIS

Die Standard Netzwerkvariablen für Motion Control dürfen vom Anwender gelesen, aber nicht geschrieben werden.

Die zusätzlichen Netzwerkvariablen für EtherCAT werden vom Anwender geschrieben (Sollwerte) und gelesen (Istwerte).



HINWEIS

Bei der Verwendung von Motion Control erfolgt der Netzwerkstart automatisch durch Motion Control.



HINWEIS

Die Netzwerkvariablen müssen in der Motion Control Event-Task gelesen und geschrieben werden.

5.3 PROPROG wt II

5.3.1 b maXX drive PLC und Regler

Prozessdaten:

In einem definierbaren Zeitraster übernimmt die b maXX drive PLC die Istwerte und das Statuswort des Reglers von der BACI und übergibt Sollwerte und Steuerwort über die BACI an den Regler.

Bei der Übergabe der Istwerte und des Statusworts wird im b maXX drive PLC das Hardware-Ereignis "BACI-Prozessdaten" ausgelöst. Dieses Ereignis kann im b maXX drive PLC eine Event-Task auslösen, in der die BACI-Prozessdatenkommunikation durchgeführt werden kann.

Die Einstellung des Zeitpunkts bzw. des Zeitrasters für die Kommunikation erfolgt bei der Initialisierung der Prozessdatenkommunikation im Anwenderprogramm mit dem FB BACI_INIT.

Bedarfsdaten:

Die b maXX drive PLC übergibt einen Parameter-Lesen- oder Parameter-Schreiben-Auftrag an die BACI. Der Regler liest den Auftrag von der BACI, bearbeitet den entsprechenden Parameter im Regler (siehe jeweilige Regler-Beschreibung) und gibt das Ergebnis an die BACI zurück. Die b maXX drive PLC liest dann das Ergebnis der Kommunikation von der BACI.

5.3.2 Programmierung der BACI-Kommunikation auf der b maXX drive PLC

Aus Kompatibilitätsgründen stehen die Funktionsbausteine der Bibliotheken für PROPROG wt II auch in Bibliotheken für ProProg wt III zur Verfügung. Diese Funktionsbausteine werden nachfolgend beschrieben.



HINWEIS

Für neue Projekte mit ProMaster, ProProg wt III und Motion Control werden nachfolgende, auch für ProProg wt III beschriebene, Funktionsbausteine nicht verwendet.

Die Programmierung der b maXX drive PLC erfolgt mit dem Programmiersystem PROPROG wt II bzw. ProProg wt III (siehe Programmierhandbuch PROPROG wt II bzw. Online-Hilfesystem ProProg wt III).

Für die Programmierung der BACI Kommunikation stehen die Baumüller Anwender Bibliotheken SYSTEM1_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) und SYSTEM2_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher zur Verfügung.

In der Bibliothek SYSTEM1_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM1_PLC01_30bd00 (ProProg wt III) oder höher sind Funktionsbausteine für die Initialisierung der Prozessdatenkommunikation, für die Prozessdatenkommunikation und für die Bedarfsdatenkommunikation vorhanden.

Für die Prozessdatenkommunikation werden folgende FBs benötigt:

BACI_INIT	Initialisierung der Prozessdatenkommunikation.
BACI_PD_COMM	Prozessdatenkommunikation

Für die Bedarfsdatenkommunikation werden folgende FBs benötigt:

BACI_PAR_READ	Bedarfsdatenkommunikation (Parameter-Lesen-Auftrag)
BACI_PAR_WRITE	Bedarfsdatenkommunikation (Parameter-Schreiben-Auftrag)

Die Bibliothek SYSTEM2_PLC01_20bd00 (PROPROG wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher enthält u.a. den Funktionsbaustein:

INTR_SET	wird vom FB BACI_INIT verwendet (FB zur Verknüpfung und Aktivierung eines Hardware-Signal mit dem Ereignis BACI-Prozessdaten.)
----------	---

5.3.3 Funktionsbausteine für BACI - Übersicht

Zusätzlich zu den Standardfunktionsbausteinen können Sie herstellerdefinierte Funktionsbausteine verwenden, wenn Sie herstellerdefinierte Bibliotheken in einem Projekt angemeldet haben.

Anmerkung: Das Anmelden von Bibliotheken ist in der allgemeinen Hilfe beschrieben.

Folgende Funktionsbausteine für BACI sind verfügbar:

Funktionsbaustein	Kurzbeschreibung
BACI_INIT	Initialisierung der BACI (BA umüller C omponent I nter- face) in der b maXX drive PLC
BACI_PAR_READ	BACI Parameter lesen
BACI_PAR_WRITE	BACI Parameter schreiben
BACI_PD_COMM	Prozessdatenkommunikation über die BACI in der b maXX drive PLC für max. 8 Soll- und Istwerte
BACI_PD_RECONFIG	BACI-Prozessdatenkommunikation während des laufen- den Betriebs rekonfigurieren

5.3.3.1 BACI_INIT

Diesen Funktionsbaustein für BACI können Sie verwenden, um die Prozessdatenkommunikation zwischen b maXX Regler und b maXX drive PLC über die BACI-Schnittstelle zu initialisieren.



Hinweis

Der FB BACI_INIT verwendet die Bibliothek BM_TYPES_20bd03 (PROPROP wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher und die Firmware-Bibliothek SYSTEM2_PLC01_20bd00 (PROPROP wt II) bzw. SYSTEM2_PLC01_30bd00 (ProProg wt III) oder höher.

Einsatz: Während der Initialisierungsphase (Kalt/Warmstart-Task).



Hinweis

Notwendige b maXX Reglereinstellungen für i_EVENT = INT#11 (=SYNC1-Signalquelle) bzw. i_EVENT = INT#12 (=SYNC2-Signalquelle) auf WinBASS II Seite "Synchronisation":

- Quelle für Sync-Signal:
 - "Sync 1 Signal von der BACI verwenden" (i_EVENT = INT#11) bzw.
 - "Sync 2 Signal von der BACI verwenden" (i_EVENT = INT#12).
- Sync Intervall auswählen (=> Die gleiche Zeit-Periode muss am Eingang "u_CYCLE_TIME" in µs angeschlossen werden).
 - Sync Offset: -25.0 µs.
 - Sync Toleranz: 2.0 µs.

Durch die Einstellung des SyncOffsets wird gewährleistet, dass die Bearbeitung des Istwert-Zyklus vom Regler bereits beendet ist, bevor die PLC ihren Kommunikations-Zyklus beginnt.

Ansonsten kann es zu BACI-Zugriffskonflikten bei der Sollwert-und Istwertübergabe im Baustein BACI_PD_COMM kommen.

Parameter Eingang	Datentyp	Beschreibung
_BASE	BACI_CTRL_BMSTRUCT	Betriebsdaten für BACI
i_EVENT	INT 0, 4, 8, 11, 12	Ereignis
u_CYCLE_TIME	UINT	BACI Zykluszeit in µs
u_WR_PAR_NR0	UINT	Sollwert-Parameternummer 0
u_WR_PAR_NR1	UINT	Sollwert-Parameternummer 1
u_WR_PAR_NR2	UINT	Sollwert-Parameternummer 2
u_WR_PAR_NR3	UINT	Sollwert-Parameternummer 3
u_WR_PAR_NR4	UINT	Sollwert-Parameternummer 4
u_WR_PAR_NR5	UINT	Sollwert-Parameternummer 5

Parameter Eingang	Datentyp	Beschreibung
u_WR_PAR_NR6	UINT	Sollwert-Parameternummer 6
u_WR_PAR_NR7	UINT	Sollwert-Parameternummer 7
u_WR_PAR_NR8	UINT	Sollwert-Parameternummer 8
u_WR_PAR_NR9	UINT	Sollwert-Parameternummer 9
u_WR_PAR_NR10	UINT	Sollwert-Parameternummer 10
u_WR_PAR_NR11	UINT	Sollwert-Parameternummer 11
u_WR_PAR_NR12	UINT	Sollwert-Parameternummer 12
u_WR_PAR_NR13	UINT	Sollwert-Parameternummer 13
u_WR_PAR_NR14	UINT	Sollwert-Parameternummer 14
u_WR_PAR_NR15	UINT	Sollwert-Parameternummer 15
u_RD_PAR_NR0	UINT	Istwert-Parameternummer 0
u_RD_PAR_NR1	UINT	Istwert-Parameternummer 1
u_RD_PAR_NR2	UINT	Istwert-Parameternummer 2
u_RD_PAR_NR3	UINT	Istwert-Parameternummer 3
u_RD_PAR_NR4	UINT	Istwert-Parameternummer 4
u_RD_PAR_NR5	UINT	Istwert-Parameternummer 5
u_RD_PAR_NR6	UINT	Istwert-Parameternummer 6
u_RD_PAR_NR7	UINT	Istwert-Parameternummer 7
u_RD_PAR_NR8	UINT	Istwert-Parameternummer 8
u_RD_PAR_NR9	UINT	Istwert-Parameternummer 9
u_RD_PAR_NR10	UINT	Istwert-Parameternummer 10
u_RD_PAR_NR11	UINT	Istwert-Parameternummer 11
u_RD_PAR_NR12	UINT	Istwert-Parameternummer 12
u_RD_PAR_NR13	UINT	Istwert-Parameternummer 13
u_RD_PAR_NR14	UINT	Istwert-Parameternummer 14
u_RD_PAR_NR15	UINT	Istwert-Parameternummer 15
t_TIME	TIME	Überwachungszeit in Sekunden

Parameter Ausgang	Datentyp	Beschreibung
_BASE	BACI_CTRL_BMSTRUCT	Betriebsdaten für BACI
i_ERR	INT	Fehlernummer
x_ERR	BOOL	Fehlerbit
x_OK	BOOL	OK-Bit

Beschreibung

Der FB BACI_INIT ermöglicht die Initialisierung der Prozessdatenkommunikation zwischen dem b maXX Regler und der b maXX drive PLC.

Als FB für die Prozessdatenkommunikation steht der FB BACI_PD_COMM zur Verfügung für den Einsatz in der zugeordneten Bypass-Interrupt (siehe Eingang i_EVENT).

Ein-/Ausgang _BASE:

An _BASE muss eine globale Variable vom Datentyp BACI_CTRL_BMSTRUCT angeschlossen werden.

Diese globale Variable ist bereits in der BM4_O_PLC01 - Vorlage in PROPROG wt im globalen Variablenblatt "Global_Variables" als _PLC_Init vordefiniert worden.

Alternativ kann diese Variable vom Anwender auch selbst angelegt werden:

Sie muss dann über die Deklaration der globalen Variablen auf die Basisadresse des DPRAM der b maXX drive PLC, %MB3.9000000, gelegt werden.

Beispiel:

```
_PLC_Init AT %MB3.9000000 : BACI_CTRL_BMSTRUCT;
```

dabei ist:

_PLC_Init	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
BACI_CTRL_BMSTRUCT	der Datentyp
%MB3.9000000	die Basisadresse des DPRAM der b maXX drive PLC

Eingang i_EVENT:

Mit i_EVENT = 4 wird das Ereignis "BACI-Prozessdaten" mit Interrupt-Level 13 (niedrige Priorität) initialisiert.

Mit i_EVENT = 8 wird das Ereignis "BACI-Prozessdaten" mit Interrupt-Level 14 (hohe Priorität) initialisiert.

Mit i_EVENT = 11 wird das Ereignis "SYNC-Signal 1 Optionsmodul" mit Interrupt-Level 14 (hohe Priorität) initialisiert.

Mit i_EVENT = 12 wird das Ereignis "SYNC-Signal 2 Optionsmodul" mit Interrupt-Level 14 (hohe Priorität) initialisiert.

Wird der Eingang i_EVENT nicht belegt oder ist i_EVENT = 0, wird kein Ereignis initialisiert. In diesem Fall werden zwar neue Parameternummern rekonfiguriert, jedoch keine Interrupt-Quelle eingerichtet.

Soll während des Betriebs eine Rekonfigurierung von Parameternummern durchgeführt werden, dann sollte der Baustein BACI_PD_RECONFIG verwendet werden.

Ist i_EVENT nicht 0, 4, 8, 11 oder 12, wird i_ERR auf den Wert INT#3 gesetzt.

Eingang u_CYCLE_TIME:

Am Eingang u_CYCLE_TIME wird die Zykluszeit der Prozessdatenkommunikation festgelegt. Die Zykluszeit der Prozessdatenkommunikation entspricht dem Intervall in dem das Ereignis "BACI-Prozessdaten" einen Interrupt auf der b maXX drive PLC auslöst.

Der Wert `u_CYCLE_TIME` muss ein Vielfaches von 125 μ s sein und mindestens 250 μ s betragen.

Ist `u_CYCLE_TIME` < 250, wird 250 μ s als Zykluszeit der Prozessdatenkommunikation eingestellt.

Ist `u_CYCLE_TIME` < 125 * x, wird aufgerundet und 125 μ s * x als Zykluszeit der Prozessdatenkommunikation eingestellt.

Beispiel:

`u_CYCLE_TIME` = 100: es wird 250 μ s eingestellt.

`u_CYCLE_TIME` = 1900: es wird 2000 μ s (= 2 ms) eingestellt.

Eingänge `u_WR_PAR_NR0` bis `u_WR_PAR_NR15`:

Die Sollwert-Parameternummern der zyklisch zu übertragenden Sollwerte werden an den Eingängen `u_WR_PAR_NR0` bis `u_WR_PAR_NR15` angegeben. Die Sollwerte selbst werden dann in der zugeordneten Bypass-Task zyklisch über den Baustein `BACI_PD_COMM` übertragen.

Eingänge `u_RD_PAR_NR0` bis `u_RD_PAR_NR15`:

Die Istwert-Parameternummern der zyklisch zu übertragenden Istwerte werden an den Eingängen `u_RD_PAR_NR0` bis `u_RD_PAR_NR15` angegeben. Die Istwerte selbst werden dann in der zugeordneten Bypass-Task zyklisch über den Baustein `BACI_PD_COMM` übertragen.

Eingang `t_TIME`:

Die Überwachungszeit wird am Eingang `t_TIME` in s eingestellt. Wird der Eingang `t_TIME` nicht belegt, ergibt sich eine Voreinstellung von TIME#3s. Kann innerhalb der Überwachungszeit die Initialisierung nicht durchgeführt werden, erfolgt eine entsprechende Timeout-Fehlermeldung.

Ausgang `x_OK`:

Der Ausgang `x_OK` ist auf TRUE gesetzt, wenn die Prozessdatenkommunikation zwischen b maXX Regler und b maXX drive PLC über die BACI-Schnittstelle richtig initialisiert werden konnte.

Ausgänge `x_ERR`, `i_ERR`:

Falls ein Fehler auftritt, wird das Fehlerbit `x_ERR` auf TRUE gesetzt und die Fehlernummer `i_ERR` ausgegeben.

Fehlernummer i_ERR:

i_ERR	Fehler
-32768 bis -1	Reserviert
0	Error i_EVENT Initialisierung
1	Timeout basic init b maXX controller: warten auf das ConfDone-Bit bzw. config flag bit.
2	Timeout Reconfig b maXX controller: warten auf das ConfDone-Bit.
3	Timeout Reconfig b maXX drive PLC: Fehler in den PLC Einstellungen.
4	Eingang i_EVENT ist ungleich 0, 4, 8, 11 oder 12.
5 bis 32767	Reserviert

5.3.3.2 BACI_PAR_READ

Diesen Funktionsbaustein für BACI können Sie verwenden, um einen Bedarfsdatenwert (= Parameterwert) vom b maXX Regler über die BACI-Schnittstelle zu lesen.



Hinweis

Der FB BACI_PAR_READ verwendet die Bibliothek BM_TYPES_20bd03 (PROPROG wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher.

Einsatz: Wird vorzugsweise in der DEFAULT-Task oder in einer beliebigen Cyclic Task eingebunden.

Parameter Eingang	Datentyp	Beschreibung
_BASE	BACI_CTRL_BMSTRUCT	Betriebsdaten für BACI
u_PAR_NR	UINT	Parameternummer
t_TIME	TIME	Überwachungszeit
x_EN	BOOL	Freigabe
x_RESET	BOOL	Reset

Parameter Ausgang	Datentyp	Beschreibung
_BASE	BACI_CTRL_BMSTRUCT	Betriebsdaten für BACI
di_PAR_VALUE	DINT	gelesener Parameterwert
x_PAR_FORMAT	BOOL	Format des Parameters (16/32 Bit)
x_BUSY	BOOL	Kommunikation ist aktiv
i_ERR	INT	Fehlernummer
x_ERR	BOOL	Fehlerbit
x_OK	BOOL	OK-Bit

Beschreibung

Der FB BAPS_PAR_READ sendet über den Parameternummer-Eingang u_PAR_NR einen Parameter-Lese-Auftrag an den b maXX Regler. Der vom b maXX Regler angeforderte Parameterwert wird am Ausgang ud_PAR_VALUE und das Format des Parameters wird am Ausgang x_PAR_FORMAT angezeigt. Falls ein Fehler bei der Ausführung des Parameter-Lesen-Auftrags auftritt, wird das Fehlerbit x_ERR gesetzt und die Fehlernummer i_ERR ausgegeben.

Der FB BACI_PAR_READ kann ebenso wie der FB BACI_PAR_WRITE mehrfach eingesetzt (instanziiert) werden. Es darf jedoch nur ein Parameter-Auftrag zur Zeit aktiv sein (d. h. es darf nur ein FB BACI_PAR_READ oder BACI_PAR_WRITE zur selben Zeit aktiv sein).

Ein-/Ausgang _BASE:

An _BASE muss eine globale Variable vom Datentyp BACI_CTRL_BMSTRUCT angeschlossen werden.

Diese globale Variable ist bereits in der BM4_O_PLC01 - Vorlage in PROPROG wt im globalen Variablenblatt "Global_Variables" als _PLC_Init vordefiniert worden.

Alternativ kann diese Variable vom Anwender auch selbst angelegt werden:

Sie muss dann über die Deklaration der globalen Variablen auf die Basisadresse des DPRAM der b maXX drive PLC, %MB3.9000000, gelegt werden.

Beispiel:

```
_PLC_Init AT %MB3.9000000 : BACI_CTRL_BMSTRUCT;
```

dabei ist:

_PLC_Init	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
BACI_CTRL_BMSTRUCT	der Datentyp
%MB3.9000000	die Basisadresse des DPRAM der b maXX drive PLC

Eingang u_PAR_NR:

Am Eingang u_PAR_NR wird die Parameternummer des Parameters angegeben, dessen Wert gelesen werden soll.

Eingang t_TIME:

Die Überwachungszeit wird am Eingang t_TIME in s eingestellt. Wird der Eingang t_TIME nicht belegt, ergibt sich eine Voreinstellung von 3 s. Kann innerhalb der Überwachungszeit der Auftrag nicht abgeschlossen, erfolgt eine Timeout-Meldung.

Eingang x_EN:

Die Kommunikation wird mit x_EN = TRUE gestartet. Wird x_EN auf FALSE gesetzt bevor x_BUSY = FALSE ist, wird von einem bewussten Abbruch der Kommunikation ausgegangen. Der FB BACI_PAR_READ muss dann mit x_RESET = TRUE zurückgesetzt werden.

Eingang x_RESET:

Mit x_RESET = TRUE wird der FB zurückgesetzt.

Ausgang di_PAR_VALUE:

Der gelesene Parameter-Wert wird am Ausgang di_PAR_VALUE ausgegeben.

Ausgang x_PAR_FORMAT:

Am Ausgang x_PAR_FORMAT wird das Format des Parameters ausgegeben.

Bei einem 16-Bit-Parameter ist x_PAR_FORMAT = FALSE, bei einem 32-Bit-Parameter ist x_PAR_FORMAT = TRUE.

Bei anderen Parameterformaten (z. B. STRING-Parameter, zur Zeit nicht unterstützt) bleibt `x_PAR_FORMAT = FALSE` und es wird an `i_ERR` eine Nummer ausgegeben.

Ausgang `x_BUSY`:

Der Ausgang `x_BUSY` zeigt mit `TRUE` an, dass die Kommunikation aktiv ist.

Ausgang `x_OK`:

Der Ausgang `x_OK` wird auf `TRUE` gesetzt, wenn die Kommunikation erfolgreich abgeschlossen ist.

Ausgänge `x_ERR`, `i_ERR`:

Falls ein Fehler auftritt, wird das Fehlerbit `x_ERR` auf `TRUE` gesetzt und der Fehler am Ausgang `i_ERR` spezifiziert.

Fehlernummer `i_ERR`:

i_ERR	Fehler
-32768 bis -3	Reserviert
-2	Timeout
-1	Parameter-Lese-Auftrag auf der BACI-Schnittstelle wurde abgebrochen (evtl. verursacht durch eine andere Instanz von FB BACI_PAR_READ oder FB BACI_PAR_WRITE)
0	Kein Fehler
1 bis 4	Interne Meldung vom b maXX Regler 1 - 4
5 bis 127	Reserviert
128 bis 141	Interne Meldung vom b maXX Regler 128 - 141
142 bis 255	Reserviert
256 bis 273	Interne Meldung vom b maXX Regler 256 - 273
274 bis 511	Reserviert
512	Kein Datentyp ermittelbar
513	Datentyp wird nicht unterstützt
514	Datentyp ist STRING (wird zur Zeit nicht unterstützt).
515 bis 32767	Reserviert

5.3.3.3 BACI_PAR_WRITE

Diesen Funktionsbaustein für BACI können Sie verwenden, um einen Bedarfsdatenwert (= Parameterwert) zum b maXX Regler über die BACI-Schnittstelle zu schreiben.



Hinweis

Der FB BACI_PAR_WRITE verwendet die Bibliothek BM_TYPES_20bd03 (PROPROP wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher.

Einsatz: Wird vorzugsweise in der DEFAULT-Task oder in einer beliebigen Cyclic Task eingebunden.

Parameter Eingang	Datentyp	Beschreibung
_BASE	BACI_CTRL_BMSTRUCT	Betriebsdaten für BACI
u_PAR_NR	UINT	Parameternummer
di_PAR_VALUE	DINT	Parameterwert
t_TIME	TIME	Überwachungszeit
x_EN	BOOL	Freigabe
x_RESET	BOOL	Reset

Parameter Ausgang	Datentyp	Beschreibung
_BASE	BACI_CTRL_BMSTRUCT	Betriebsdaten für BACI
x_BUSY	BOOL	Kommunikation ist aktiv
i_ERR	INT	Fehlernummer
x_ERR	BOOL	Fehlerbit
x_OK	BOOL	OK-Bit

Beschreibung

Der FB BACI_PAR_WRITE sendet über den Parameternummer-Eingang u_PAR_NR und den Parameterwert di_PAR_VALUE einen Parameter-Schreib-Auftrag an den b maXX Regler. Falls ein Fehler bei der Ausführung des Parameter-Schreiben-Auftrags auftritt, wird das Fehlerbit x_ERR gesetzt und die Fehlernummer i_ERR ausgegeben.

Der FB BACI_PAR_WRITE kann ebenso wie der FB BACI_PAR_READ mehrfach eingesetzt (instanziiert) werden. Es darf jedoch nur ein Parameter-Auftrag zur Zeit aktiv sein (d. h. es darf nur ein FB BACI_PAR_READ oder BACI_PAR_WRITE zur selben Zeit aktiv sein).

Ein-/Ausgang _BASE:

An _BASE muss eine globale Variable vom Datentyp BACI_CTRL_BMSTRUCT angeschlossen werden.

Diese globale Variable ist bereits in der BM4_O_PLC01 - Vorlage in PROPROG wt im globalen Variablenblatt "Global_Variables" als _PLC_Init vordefiniert worden.

Alternativ kann diese Variable vom Anwender auch selbst angelegt werden:

Sie muss dann über die Deklaration der globalen Variablen auf die Basisadresse des DPRAM der b maXX drive PLC, %MB3.9000000, gelegt werden.

Beispiel:

```
_PLC_Init AT %MB3.9000000 : BACI_CTRL_BMSTRUCT;
```

dabei ist:

_PLC_Init	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
BACI_CTRL_BMSTRUCT	der Datentyp
%MB3.9000000	die Basisadresse des DPRAM der b maXX drive PLC

Eingang u_PAR_NR:

Am Eingang u_PAR_NR wird die Parameter-Nummer des Parameters angegeben, dessen Wert geschrieben werden soll.

Eingang di_PAR_VALUE:

Der zu schreibende Parameterwert wird am Eingang di_PAR_VALUE angegeben.

Eingang t_TIME:

Die Überwachungszeit wird am Eingang t_TIME in s eingestellt. Wird der Eingang t_TIME nicht belegt, ergibt sich eine Voreinstellung von 3 s. Kann der Auftrag innerhalb der Überwachungszeit nicht abgeschlossen werden, erfolgt eine Timeout-Meldung.

Eingang x_EN:

Die Kommunikation wird mit x_EN = TRUE gestartet. Wird x_EN auf FALSE gesetzt bevor x_BUSY = FALSE ist, wird von einem bewussten Abbruch der Kommunikation ausgegangen. Der FB BACI_PAR_WRITE muss dann jeweils mit x_RESET = TRUE zurückgesetzt werden.

Eingang x_RESET:

Mit x_RESET = TRUE wird der FB zurückgesetzt.

Ausgang x_BUSY:

Der Ausgang x_BUSY zeigt mit TRUE an, dass die Kommunikation aktiv ist.

Ausgang x_OK:

Der Ausgang x_OK wird auf TRUE gesetzt, wenn die Kommunikation erfolgreich abgeschlossen ist.

Ausgänge x_ERR, i_ERR:

Falls ein Fehler auftritt, wird das Fehlerbit x_ERR auf TRUE gesetzt und der Fehler am Ausgang i_ERR spezifiziert.

Fehlernummer i_ERR:

i_ERR	Fehler
-32768 bis -3	Reserviert
-2	Timeout
-1	Parameter-Schreiben-Auftrag auf der BACI-Schnittstelle wurde abgebrochen (eventuell verursacht durch eine andere Instanz von FB BACI_PAR_WRITE oder FB BACI_PAR_READ)
0	kein Fehler
1 bis 4	Interne Meldung b maXX Regler 1 bis 4
5 bis 127	Reserviert
128 bis 141	Interne Meldung b maXX Regler 128 bis 141
142 bis 255	Reserviert
256	Ungültige Parameter-Nummer
257	Ungültiger Datentyp
258	Wert kleiner als Minimalwert
259	Wert größer als Maximalwert
260	Parameter ist read-only, d. h. Parameter Schreiben ist nicht erlaubt
261	Interne Meldung b maXX Regler 261
262	Parameter kann aufgrund Betriebszustand nicht geändert werden
263	Parameterwert ist ungültig
264 bis 273	Interne Meldung b maXX Regler 264 bis 511
274 bis 281	Reserviert
282	Parameter schreiben nicht möglich, da „BACI, Schreibzugriffe Bedarfsdaten“ im Antriebsmanager (WinBASS II) nicht freigegeben ist
283 bis 32767	Reserviert

5.3.3.4 BACI_PD_COMM

Diesen Funktionsbaustein für BACI können Sie verwenden, um die Prozessdatenkommunikation zwischen b maXX Regler und b maXX drive PLC über die BACI-Schnittstelle durchzuführen.



Hinweis

Der FB BACI_PD_COMM verwendet die Bibliothek BM_TYPES_20bd03 (PROPROG wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher.

Einsatz: In einer Bypass-Task, in der die Prozessdatenkommunikation stattfinden soll (also Event 4, 8, 11 oder 12).

Parameter Eingang	Datentyp	Beschreibung
_BASE	BACI_CTRL_BMSTRUCT	Betriebsdaten für BACI
di_WR_VALUE0	DINT	Sollwert 0
di_WR_VALUE1	DINT	Sollwert 1
di_WR_VALUE2	DINT	Sollwert 2
di_WR_VALUE3	DINT	Sollwert 3
di_WR_VALUE4	DINT	Sollwert 4
di_WR_VALUE5	DINT	Sollwert 5
di_WR_VALUE6	DINT	Sollwert 6
di_WR_VALUE7	DINT	Sollwert 7
di_WR_VALUE8	DINT	Sollwert 8
di_WR_VALUE9	DINT	Sollwert 9
di_WR_VALUE10	DINT	Sollwert 10
di_WR_VALUE11	DINT	Sollwert 11
di_WR_VALUE12	DINT	Sollwert 12
di_WR_VALUE13	DINT	Sollwert 13
di_WR_VALUE14	DINT	Sollwert 14
di_WR_VALUE16	DINT	Sollwert 15
x_SYNC_CHECK	BOOL	Freigabe Synchronisationsüberwachung
x_EN	BOOL	Freigabe

Parameter Ausgang	Datentyp	Beschreibung
_BASE	BACI_CTRL_BMSTRUCT	Betriebsdaten für BACI
di_RD_VALUE0	DINT	Istwert 0
di_RD_VALUE1	DINT	Istwert 1
di_RD_VALUE2	DINT	Istwert 2
di_RD_VALUE3	DINT	Istwert 3
di_RD_VALUE4	DINT	Istwert 4
di_RD_VALUE5	DINT	Istwert 5
di_RD_VALUE6	DINT	Istwert 6
di_RD_VALUE7	DINT	Istwert 7
di_RD_VALUE8	DINT	Istwert 8
di_RD_VALUE9	DINT	Istwert 9
di_RD_VALUE10	DINT	Istwert 10
di_RD_VALUE11	DINT	Istwert 11
di_RD_VALUE12	DINT	Istwert 12
di_RD_VALUE13	DINT	Istwert 13
di_RD_VALUE14	DINT	Istwert 14
di_RD_VALUE15	DINT	Istwert 15
x_SYNC_ERR	BOOL	Synchronisationsfehlerbit
i_ERR	INT	Fehlernummer
x_ERR	BOOL	Fehlerbit

Beschreibung

Die Sollwerte werden an den b maXX Regler gesendet, die Istwerte werden vom b maXX Regler empfangen und ausgegeben.

Ein-/Ausgang _BASE:

An _BASE muss eine globale Variable vom Datentyp BACI_CTRL_BMSTRUCT angeschlossen werden.

Diese globale Variable ist bereits in der BM4_O_PLC01 - Vorlage in PROPROG wt im globalen Variablenblatt "Global_Variables" als _PLC_Init vordefiniert worden.

Alternativ kann diese Variable vom Anwender auch selbst angelegt werden:

Sie muss dann über die Deklaration der globalen Variablen auf die Basisadresse des DPRAM der b maXX drive PLC, %MB3.9000000, gelegt werden.

Beispiel:

```
_PLC_Init AT %MB3.9000000 : BACI_CTRL_BMSTRUCT;
```

dabei ist:

<code>_PLC_Init</code>	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
<code>BACI_CTRL_BMSTRUCT</code>	der Datentyp
<code>%MB3.9000000</code>	die Basisadresse des DPRAM der b maXX drive PLC

Eingänge `di_WR_VALUE0` bis `di_WR_VALUE15`:

An den Eingängen `di_WR_VALUE0` bis `di_WR_VALUE15` werden die Sollwerte angeschlossen, deren Parameternummern am FB `BACI_INIT` (Eingänge `u_WR_PAR_NR0` bis `u_WR_PAR_NR15`) in der Initialisierung der Prozessdatenkommunikation zwischen b maXX Regler und b maXX drive PLC über die BACI-Schnittstelle angegeben wurden (bzw. den über `BACI_PD_RECONFIG` geänderten Parameternummern).

Eingang `x_SYNC_CHECK`:

Mit `x_SYNC_CHECK = TRUE` wird die Synchronisationsüberwachung zwischen b maXX Regler und b maXX drive PLC über die BACI-Schnittstelle freigegeben. Voreinstellung für `x_SYNC_CHECK = TRUE`, d. h. die Synchronisationsüberwachung ist freigegeben.

Eingang `x_EN`:

Mit `x_EN = TRUE` wird die Kommunikation freigegeben. Voreinstellung ist `x_EN = TRUE`, d. h. die Kommunikation ist freigegeben.

Ausgänge `di_RD_VALUE0` bis `di_RD_VALUE15`:

An den Ausgängen `di_RD_VALUE0` bis `di_RD_VALUE15` werden die Istwerte ausgegeben, deren Parameternummern am FB `BACI_INIT` (Eingänge `u_RD_PAR_NR0` bis `u_RD_PAR_NR15`) in der Initialisierung der Prozessdatenkommunikation zwischen b maXX Regler und b maXX drive PLC über die BACI-Schnittstelle angegeben wurden (bzw. den über `BACI_PD_RECONFIG` geänderten Parameternummern).

Ausgang `x_SYNC_ERR`:

Wenn die Synchronisationsüberwachung aktiviert wurde, wird der Alive Counter überwacht. Wenn ein Synchronfehler auftritt (= ein oder mehrere Kommunikationszyklen fehlen), wird das Synchronisationsfehlerbit `x_SYNC_ERR` auf `TRUE` gesetzt.

Ausgänge `x_ERR`, `i_ERR`:

Falls ein Fehler auftritt, wird das Fehlerbit `x_ERR` auf `TRUE` gesetzt und die Fehlernummer `i_ERR` ausgegeben.

Fehlernummer i_ERR:

Wenn Fehler auftreten, überprüfen Sie die Einstellung in WinBASS II Seite "Synchronisation"

(z. B. SyncOffset = -25,0 µs).

i_ERR	Fehler
-32768 bis -6	Reserviert
-5	BACI-Zugriffskonflikt bei der Istwertübergabe: Der b maXX Regler hat sein "data not valid flag" und "data access flag" gesetzt. => Die b maXX drive PLC darf keine Istwerte vom b maXX Regler lesen.
-4	BACI-Zugriffskonflikt bei der Istwertübergabe: Der b maXX Regler hat sein "data access flag" gesetzt, nachdem die b maXX drive PLC sein Eigenes gesetzt hatte. => Die b maXX drive PLC darf keine Istwerte vom b maXX Regler lesen.
-3	BACI-Zugriffskonflikt bei der Istwertübergabe: Der b maXX Regler hat sein "data access flag" gesetzt. => Die b maXX drive PLC darf keine Istwerte vom b maXX Regler lesen.
-2	BACI-Zugriffskonflikt bei der Sollwertübergabe: Der b maXX Regler hat sein "data access flag" gesetzt, nachdem die b maXX drive PLC sein Eigenes gesetzt hatte. => Die b maXX drive PLC darf keine Sollwerte zum b maXX Regler schreiben.
-1	BACI-Zugriffskonflikt bei der Sollwertübergabe: Der b maXX Regler hat sein "data access flag" gesetzt. => Die b maXX drive PLC darf keine Sollwerte zum b maXX Regler schreiben.
0	Kein Fehler
1 bis 32767	Reserviert

5.3.3.5 BACI_PD_RECONFIG

Diesen Funktionsbaustein für BACI können Sie verwenden, um die Prozessdatenkommunikation zwischen b maXX Regler und b maXX drive PLC über die BACI-Schnittstelle während des laufenden Betriebs zu rekonfigurieren.



Hinweis

Der FB BACI_PD_RECONFIG verwendet die Bibliothek BM_TYPES_20bd03 (PROPROG wt II) bzw. BM_TYPES_30bd01 (ProProg wt III) oder höher.

Während der Umkonfigurierung muss der FB BACI_PD_COMM gesperrt sein (x_EN := FALSE).

Im Gegensatz zum Baustein BACI_INIT können nur die Parameternummern geändert werden, die über BACI_INIT eingestellte Zykluszeit und Interruptquelle bleiben weiterhin bestehen.

Einsatz: Wird aus Laufzeitgründen vorzugsweise in der DEFAULT-Task oder in einer beliebigen cyclic Task eingebunden. Kann aber auch in einer Bypass-Task (z. B. Event 4, 8, 11 oder 12) laufen.

Parameter Eingang	Datentyp	Beschreibung
_BASE	BACI_CTRL_BMSTRUCT	Betriebsdaten für BACI
u_WR_PAR_NR0	UINT	Sollwert-Parameternummer 0
u_WR_PAR_NR1	UINT	Sollwert-Parameternummer 1
u_WR_PAR_NR2	UINT	Sollwert-Parameternummer 2
u_WR_PAR_NR3	UINT	Sollwert-Parameternummer 3
u_WR_PAR_NR4	UINT	Sollwert-Parameternummer 4
u_WR_PAR_NR5	UINT	Sollwert-Parameternummer 5
u_WR_PAR_NR6	UINT	Sollwert-Parameternummer 6
u_WR_PAR_NR7	UINT	Sollwert-Parameternummer 7
u_WR_PAR_NR8	UINT	Sollwert-Parameternummer 8
u_WR_PAR_NR9	UINT	Sollwert-Parameternummer 9
u_WR_PAR_NR10	UINT	Sollwert-Parameternummer 10
u_WR_PAR_NR11	UINT	Sollwert-Parameternummer 11
u_WR_PAR_NR12	UINT	Sollwert-Parameternummer 12
u_WR_PAR_NR13	UINT	Sollwert-Parameternummer 13
u_WR_PAR_NR14	UINT	Sollwert-Parameternummer 14
u_WR_PAR_NR15	UINT	Sollwert-Parameternummer 15
u_RD_PAR_NR0	UINT	Istwert-Parameternummer 0
u_RD_PAR_NR1	UINT	Istwert-Parameternummer 1

Parameter Eingang	Datentyp	Beschreibung
u_RD_PAR_NR2	UINT	Istwert-Parameternummer 2
u_RD_PAR_NR3	UINT	Istwert-Parameternummer 3
u_RD_PAR_NR4	UINT	Istwert-Parameternummer 4
u_RD_PAR_NR5	UINT	Istwert-Parameternummer 5
u_RD_PAR_NR6	UINT	Istwert-Parameternummer 6
u_RD_PAR_NR7	UINT	Istwert-Parameternummer 7
u_RD_PAR_NR8	UINT	Istwert-Parameternummer 8
u_RD_PAR_NR9	UINT	Istwert-Parameternummer 9
u_RD_PAR_NR10	UINT	Istwert-Parameternummer 10
u_RD_PAR_NR11	UINT	Istwert-Parameternummer 11
u_RD_PAR_NR12	UINT	Istwert-Parameternummer 12
u_RD_PAR_NR13	UINT	Istwert-Parameternummer 13
u_RD_PAR_NR14	UINT	Istwert-Parameternummer 14
u_RD_PAR_NR15	UINT	Istwert-Parameternummer 15
x_CFG_INIT	BOOL	Freigabe
t_TIME	TIME	Überwachungszeit in Sekunden

Parameter Ausgang	Datentyp	Beschreibung
_BASE	BACI_CTRL_BMSTRUCT	Betriebsdaten für BACI
x_ERR	BOOL	Fehlerbit
x_OK	BOOL	OK-Bit

Beschreibung

Der FB BACI_PD_RECONFIG ermöglicht die Umkonfigurierung der Sollwert- und Istwert-Parameternummern für die Prozessdatenkommunikation zwischen dem b maXX Regler und der b maXX drive PLC bei laufendem Betrieb.

Eine Umkonfigurierung kann nur erfolgen, wenn der FB BACI_PD_COMM gesperrt ist (x_EN := FALSE).

Ein-/Ausgang _BASE:

An _BASE muss eine globale Variable vom Datentyp BACI_CTRL_BMSTRUCT angeschlossen werden.

Diese globale Variable ist bereits in der BM4_O_PLC01 - Vorlage in PROPROG wt im globalen Variablenblatt "Global_Variables" als _PLC_Init vordefiniert worden.

Alternativ kann diese Variable vom Anwender auch selbst angelegt werden:

Sie muss dann über die Deklaration der globalen Variablen auf die Basisadresse des DPRAM der b maXX drive PLC, %MB3.9000000, gelegt werden.

Beispiel:

```
_PLC_Init AT %MB3.9000000 : BACI_CTRL_BMSTRUCT;
```

dabei ist:

<code>_PLC_Init</code>	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
<code>BACI_CTRL_BMSTRUCT</code>	der Datentyp
<code>%MB3.9000000</code>	die Basisadresse des DPRAM der b maXX drive PLC

Eingänge `u_WR_PAR_NR0` bis `u_WR_PAR_NR15`:

Die Sollwert-Parameternummern der über den FB `BACI_PD_COMM` zyklisch zu übertragenden Sollwerte werden an den Eingängen `u_WR_PAR_NR0` bis `u_WR_PAR_NR15` angegeben.

Eingänge `u_RD_PAR_NR0` bis `u_RD_PAR_NR15`:

Die Istwert-Parameternummern der über den FB `BACI_PD_COMM` zyklisch zu übertragenden Istwerte werden an den Eingängen `u_RD_PAR_NR0` bis `u_RD_PAR_NR15` angegeben.

Eingang `t_TIME`:

Die Überwachungszeit wird am Eingang `t_TIME` in s eingestellt. Wird der Eingang `t_TIME` nicht belegt, ergibt sich eine Voreinstellung von `TIME 3s`. Kann die Rekonfigurierung innerhalb der Überwachungszeit nicht abgeschlossen werden, wird des Fehlerbit `x_ERR` gesetzt.

Eingang `x_CFG_INIT`:

Mit `x_CFG_INIT = TRUE` wird die Umkonfigurierung gestartet.

Ausgang `x_ERR`:

Während der Umkonfigurierung ist ein Timeout aufgetreten. Mögliche Fehlerursache: Der Baustein FB `BACI_PD_COMM` wurde nicht an `x_EN` gesperrt oder wird übersprungen.

Ausgang `x_OK`:

Der Ausgang `x_OK` ist auf `TRUE` gesetzt, wenn der b maXX Regler die neue Liste der Sollwert- und Istwert-Parameternummern übernommen hat.

Jetzt kann `x_CFG_INIT` wieder zurückgesetzt und die Prozessdatenkommunikation über den Baustein FB `BACI_PD_COMM` erneut freigegeben (`x_EN = TRUE`) werden.



ANHANG A - ABKÜRZUNGEN

AA	Funktionsmodul Analoge Ausgänge	DIN	Deutsches Institut für Normung e.V.
Abs.	Absatz	DOPPELW	Doppelwort (32 Bit)
AC	Wechselstrom	DW	Datenwort (16 Bit)
ADR	Adreßbyte	DWort	Doppelwort (32 Bit)
AE	Funktionsmodul Analoge Eingänge	EMK	Elektromagnetische Konstante
AK	Auftrags-/Antwort-Kennung	EMV	Elektromagnetische Verträglichkeit
BACI	Baumüller Component Interface	EN	Europäische Norm
BAPS	Baumüller Antriebe parallele Schnittstelle	EOF	End of File
BASS	Baumüller Antriebe serielle Schnittstelle	EXT, ext	Extern
BB	Betriebsbereit	FI	Fehlerstrom
BBext	Betriebsbereitschaft (extern)	HLG	Funktionsmodul Hochlaufgeber
BBint	Betriebsbereitschaft (intern)	HM	Hauptmenü
BEDAS	Betriebsdatenspeicher	HW	Highword
BOF	Begin of File	I/O	Input/Output, Eingang und Ausgang
BSE	Bezug extern für 24 V-Steuereingänge	IKG	Funktionsmodul Inkrementalgeber
BUB	Ballast-Einheit	ID-Nr.	Identifikations-Nummer
BUC	Baumüller Ein-/Rückspeise-Einheit	Inc	Zähleinheit der Position
BUG	Baumüller Umrichter Grund-Einspeise-Einheit	IND	Index
BUM	Baumüller Einzel-Leistungs-Einheit	Ink	Strichzahl des Inkrementalgebers
BUS	Baumüller Leistungs-Modul	INK.	Inkremental
CPU	Central Processing Unit	IW	Istwert
DA	Digital/Analog	IWK	Istwertkanal
DAC	Digital/Analog Wandler	LED	Leuchtdiode
DB	Datenbyte (8 Bit)	LW	Lowword
DC	► Gleichstrom ► Drive-Control	16#	Präfix für Hexadezimalzahl
		mtr.	Mittelträge
		n = 0	Drehzahl = 0
		NN	Höhe über Normal Null
		PKE	Parameter-Kennung

PKW	Parameter-Kennung-Wert
PZD	Prozeßdaten
R	reserviert
SE	Schirmerde
SL	Schutzleiter
SM	Synchronmotor
STX	Start of Text
SW	Sollwert
SWG	Funktionsmodul Sollwertgenerator
SWK	Sollwertkanal
USS	Funktionsmodul USS-Protokoll
USS®	Warenzeichen Siemens, universelle serielle Schnittstelle
VBG	Verwaltungs-Berufsgenossen- schaft
VDE	Verband deutscher Elektrotechni- ker
ZK	Zwischenkreis



Stichwortverzeichnis

A			
Anwenderbibliothek	47	Grundfunktionalität	59
Einfügen in Projekt	60	I	
B		I/O-Konfiguration	29
b maXX drive PLC		INTR_SET	51
Firmware	49	K	
BACI-Schnittstelle		Kommunikation über RS232	30
Parameter lesen	154	Kommunikationsquelle	29
Parameter schreiben	157	Kontrolldialog für Ressourcen	33
Prozessdatenkommunikation	160	L	
BACI-Schnittstelle initialisieren	149	Level	
Baumüller	7	Interrupt	46, 52
Begriffe		N	
Definition	5	Neues Projekt	27
Bibliotheken		P	
Übersicht	48	Personal	13
Board-Funktion	50	qualifiziert	13
C		Programmiersprachen	25
CAM_PLC_21bd00	60	Projekt	
CAM_PLC01_30bd00	60	neu, öffnen	27
D		Projekt zum Zielsystem senden	35
Datenbereich	28, 41	PROPROG wt II	25
Dokumentationsarbeitsblätter	29	PROPROG wt II Bibliothek	61
E		Prozessdatenkommunikation	149, 160
Echtzeitverhalten	45	Rekonfigurieren	164
Eigenschaft		Q	
Event-Task	46	Qualifiziertes Personal	13
Event-Task	45	R	
F		REGISTER_PLC_20bd00	60
Fachkraft	13	REGISTER_PLC_30bd00	60
Firmware-Bibliothek	47	Ressource	28
Funktionsbaustein	49	Einstellungen	30
INTR_SET	51	Ressource BM4_O_PLC01	27
LED	55	Ressourcen	
T64_REC	101	Kontrolldialog	33
T64_RSYN	99	S	
TER_INIT	89	Standardbibliothek	59
TER_R	94	SYSTEM1_C_PLC01_20bd00	50
TER_S	97	SYSTEM1_C_PLC01_30bd00	50
TER_USS	104	SYSTEM1_PLC_20bd00	59
TIME_MEASURE_END	50, 113	SYSTEM1_PLC_30bd00	59
TIME_MEASURE_START	50, 112	SYSTEM2_C_PLC01_30bd00	50
TIMER_A_INIT	67, 68	SYSTEM2_PLC_20bd00	59
Funktionsbausteine BACI		SYSTEM2_PLC_30bd00	59
Übersicht	147	T	
G		T64_REC	101
Gewährleistung und Haftung	14		
Globale Variablenarbeitsblätter	29		



Stichwortverzeichnis

T64_RSYN	99
Technologiebaustein	
Kurvenscheibe	60
Registerregelung	60
Wickler	60
TER_INIT	89
TER_R	94
TER_S	97
TER_USS	104
TIME_MEASURE_END	50, 113
TIME_MEASURE_START	50, 112
TIMER_A_INIT	67, 68

U

UNIVERSAL_20bd00	59
UNIVERSAL_30bd00	60

V

Verpflichtung und Haftung	13
Verzeichnispfad für Bibliotheken	49

W

Watchdog	39, 40, 111, 112, 114
WINDER_PLC_20bd00	60
WINDER_PLC_30bd00	60

Z

ZWT-File	61
----------	----

be in motion



Alle Angaben in dieser Betriebsanleitung sind unverbindliche Kundeninformationen, unterliegen einer ständigen Weiterentwicklung und werden fortlaufend durch unseren permanenten Änderungsdienst aktualisiert. Bitte beachten Sie, dass Angaben/Zahlen/Informationen aktuelle Werte zum Druckdatum sind.
Zur Ausmessung, Berechnung und Kalkulationen sind diese Angaben nicht rechtlich verbindlich. Bevor Sie in dieser Betriebsanleitung aufgeführte Informationen zur Grundlage eigener Berechnungen und/oder Verwendungen machen, informieren Sie sich bitte, ob Sie den aktuellsten Stand der Informationen besitzen.
Eine Haftung für die Richtigkeit der Informationen wird daher nicht übernommen.