BAUMÜLLER

be in motion be in motion

Drive-Line II

Steuerungstechnik Ωmega

Betriebsanleitung

D	5.00005.03a

Titel	Betriebsanleitung
Produkt	Ωmega Drive-Line II
Stand	24. Juni 2005
Artikelnummer	340357

Verbindlichkeit

Diese Betriebsanleitung ist Teil des Gerätes/der Maschine. Diese Betriebsanleitung muss jederzeit für den Bediener zugänglich und in einem leserlichen Zustand sein. Bei Verkauf/Verlagerung des Gerätes/der Maschine muss diese Betriebsanleitung 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.

Mit der Übergabe der vorliegenden Betriebsanleitung werden entsprechende Betriebsanleitungen mit einem früheren Stand außer Kraft gesetzt.
Bitte beachten Sie, dass Angaben/Zahlen/Informationen **aktuelle Werte zum Druckdatum** sind. Zur Ausmessung, Berechnung und Kalkulationen sind diese Angaben **nicht rechtlich verbindlich**.
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.

Es kann jedoch keine Gewährleistung bezüglich der Fehlerfreiheit dieser Betriebsanleitung, soweit nicht in den Allgemeinen Verkaufs- und Lieferbedingungen anders beschrieben, übernommen werden.

Hersteller

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

INHALTSVERZEICHNIS

1 SICHERHEITSHINWEISE

Allgemeine Hinweise

Diese Betriebsanleitung enthält die erforderlichen Informationen für den bestimmungsgemäßen Gebrauch der darin beschriebenen Produkte. Sie wendet sich an technisch qualifiziertes Personal, welches speziell ausgebildet ist und gründlich mit allen Warnungen und Instandhaltungsmaßnahmen vertraut ist.

Die Einheiten 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 Betriebsanleitung beachtet werden.

Gefahrenhinweise

Die Hinweise dienen einerseits der persönlichen Sicherheit des Anwenders und andererseits der Sicherheit vor Beschädigung der beschriebenen Produkte oder angeschlossenen Geräte.

Die verwendeten Begriffe haben im Sinne der Betriebsanleitung und der Hinweise auf den Produkten selbst folgende Bedeutung:

GEFAHR

Bedeutet, daß **Tod**, **schwere Körperverletzung** oder **erheblicher Sachschaden** eintreten **werden**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

WARNUNG

bedeutet, daß **Tod**, **schwere Körperverletzung** oder **erheblicher Sachschaden** eintreten **können**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

HINWEIS

ist eine **wichtige Information** über das Produkt, die Handhabung des Produktes oder den jeweiligen Teil der Dokumentation, auf den besonders aufmerksam gemacht werden soll.

Qualifiziertes Personal

Qualifiziertes Personal im Sinne der sicherheitsbezogenen Hinweise in dieser Betriebsanleitung oder auf den Produkten selbst sind Personen, die mit Montage, Inbetriebsetzung und Betrieb des Produktes vertraut sind und über die ihrer Tätigkeit entsprechenden Qualifikation verfügen:

- Ausbildung oder Unterweisung bzw. Berechtigung Stromkreise und Geräte gemäß den Standards der Sicherheitstechnik in Betrieb zu nehmen, zu erden und zu kennzeichnen.
- Ausbildung oder Unterweisung gemäß den Standards der Sicherheitstechnik in Pflege und Gebrauch angemessener Sicherheitsausrüstung.

Bestimmungsgemäßer Gebrauch

WARNUNG

Die Einheit / das System darf nur für die in der Betriebsanleitung vorgesehenen Einsatzfälle und nur in Verbindung mit von der BAUMÜLLER NÜRNBERG GmbH empfohlenen bzw. zugelassenen Fremdgeräten und -komponenten verwendet werden.

Eigenmächtige Umbauten und Veränderungen an der Einheit sind aus Sicherheitsgründen nicht gestattet. Der Bediener ist verpflichtet, eintretende Veränderungen, die die Sicherheit der Einheit / des Systems beeinträchtigen könnten, sofort zu melden.

2 TECHNISCHE DATEN

2.1 Allgemeines

Das Ωmega Drive-Line II ist eine antriebsintegrierte SPS zur Umsetzung dezentraler intelligenter Antriebstechnik. Der V-Regler kann dazu optional um das Ωmega Drive-Line II erweitert werden.

Die Funktionalitäten einer antriebsintegrierten SPS wie IEC 61131 Programmierung, projektierbare Regelungstechnik, Kurvenscheibe, Positionserfassung, digitale und analoge Ein- und Ausgänge oder synchrones Bussystem werden mit dem Ωmega Drive-Line II System umgesetzt.

Für die Bus-Kommunikation auf Antriebsebene ist das synchrone Bussystem CANsync vorhanden. Über eine optionale CAN-Master-Karte ist die Anschaltung externer Peripherie-Baugruppen z. B. I/O's möglich. Die Anbindung von HMI-Interfaces (Bedientableaus, Touch Screens etc.) kann über die integrierte RS485-Schnittstelle über eine SW-Anschaltung an die Prozedur 3964R® (Datenbaustein-Anbindung) als Slave-Anschaltung erfolgen. Alternativ kann diese Schnittstelle über eine SW-Anschaltung an das USS-Protokoll® betrieben werden, wobei das Ωmega Drive-Line II als Master fungiert, mit dem mehrere USS-Protokoll®-fähige Slaves angesprochen werden können.

Die Prozedur 3964R® und das USS-Protokoll® sind eingetragene Warenzeichen der Firma Siemens AG.

Die Steuerungs-und Regelungsprogrammierung des Ωmega Drive-Line II kann entweder über die Standard-RS232-Schnittstelle als Punkt-zu-Punkt-Verbindung oder über die optionale Ethernet-Schnittstelle (TCP/IP mit den entsprechenden Vernetzungsmöglichkeiten mehrerer Ωmega Drive-Lines II in der Maschinenanlage) erfolgen.

Darüberhinaus kann der Funktionsumfang über Optionskarten auf zwei Optionssteckplätzen erweitert werden. Als Optionskarten sind beispielsweise erhältlich:

- MFM-01 für digitale und analoge Ein- und Ausgänge.
- IEI-02 für Positions- und Druckmarkenerfassung über zwei Kanäle.
- CAN-M-01 für CAN-Busankopplung.
- PROFIBUS-DP-Slave-Anschaltung

Die Steuerungs- und Regelungsprogrammierung erfolgt modularisiert mit der IEC 61131-3 Programmieroberfläche PROPROG wt II in den Programmiersprachen:

- Ablaufsprache AS
- Strukturierter Text ST
- Anweisungsliste AWL
- Funktionsplan FUP
- Kontaktplan KOP

Darüberhinaus sind über Bibliotheken intelligente Antriebsfunktionalitäten implementierbar, wie:

- Kurvenscheibe
- Registerregelung
- Wickler

Neben der IEC 61131-3 Programmieroberfläche PROPROG wt II kann ein OPC-Server zur Anbindung von Visualisierungsaufgaben und Parametrierungen über OPC-Clients in das Gesamt-Maschinenkonzept integriert werden.

2.2 Funktionalität

- 32-Bit RISC-CPU 120 MHz
- Programmspeicher 2046 kByte für
  - max. 400000 AWL-Zeilen (LD/ST-Anweisungen auf globale Variablen)
  - typ. 120000 AWL-Zeilen (typische AWL-Anweisungen auf Strukturen und Instanz-Variablen)
- Zykluszeit 100 µs pro 1000 Linien Anweisungsliste (AWL)
- 56 kB NOVRAM Nichtflüchtiger Datenspeicher (= Gesamtbereich „Remante Merker“)
- Variabler RAM-Speicher von 2 MByte (= Gesamtbereich „Nicht Remante Merker“)
- Dynamischer Speicher 1460 kByte für Debug- und Logic-Analyzer-Funktionen
- Serielle Programmier-Schnittstelle RS232 mit 38400 Baud, vom Ωmega Drive-Line II optisch entkoppelt
- Serielle Terminal-Schnittstelle RS485 mit max.38400 Baud, vom Ωmega Drive-Line II optisch entkoppelt
- 2 Optionssteckplätze zur Erweiterung des Systems

  es können 2 Arten von Optionskarten bestückt werden:

  I/O - Karten z. B. IEI (Inkrementalzählerbaugruppe), MFM (digitale und analoge Ein- und Ausgänge)

  Feldbuskarten z. B. CAN, PROFIBUS-DP

  folgende Kombinationen von Optionskarten sind möglich.

  zwei I/O - Karten

  eine I/O - Karte und eine Feldbuskarte

  Die Kombination von 2 Feldbuskarten ist nicht möglich.
- 2 CANsync-Knoten max. 500 kBit/s, vom Ωmega Drive-Line II optisch entkoppelt
- Ethernet Schnittstelle 10/100 MBit/s (optional)
- Leistungsaufnahme 5,5 W

Das Ωmega Drive-Line II ist

- mit einem Betriebssystem ohne Aktivierung des CPU-internen Caches
- mit einem Betriebssystem mit Aktivierung des CPU-internen Caches

lieferbar.

Ohne Aktivierung des CPU-internen Caches ist für eine lineare Befehlsfolge zu jedem Zeitpunkt die gleiche Bearbeitungszeit gewährleistet.

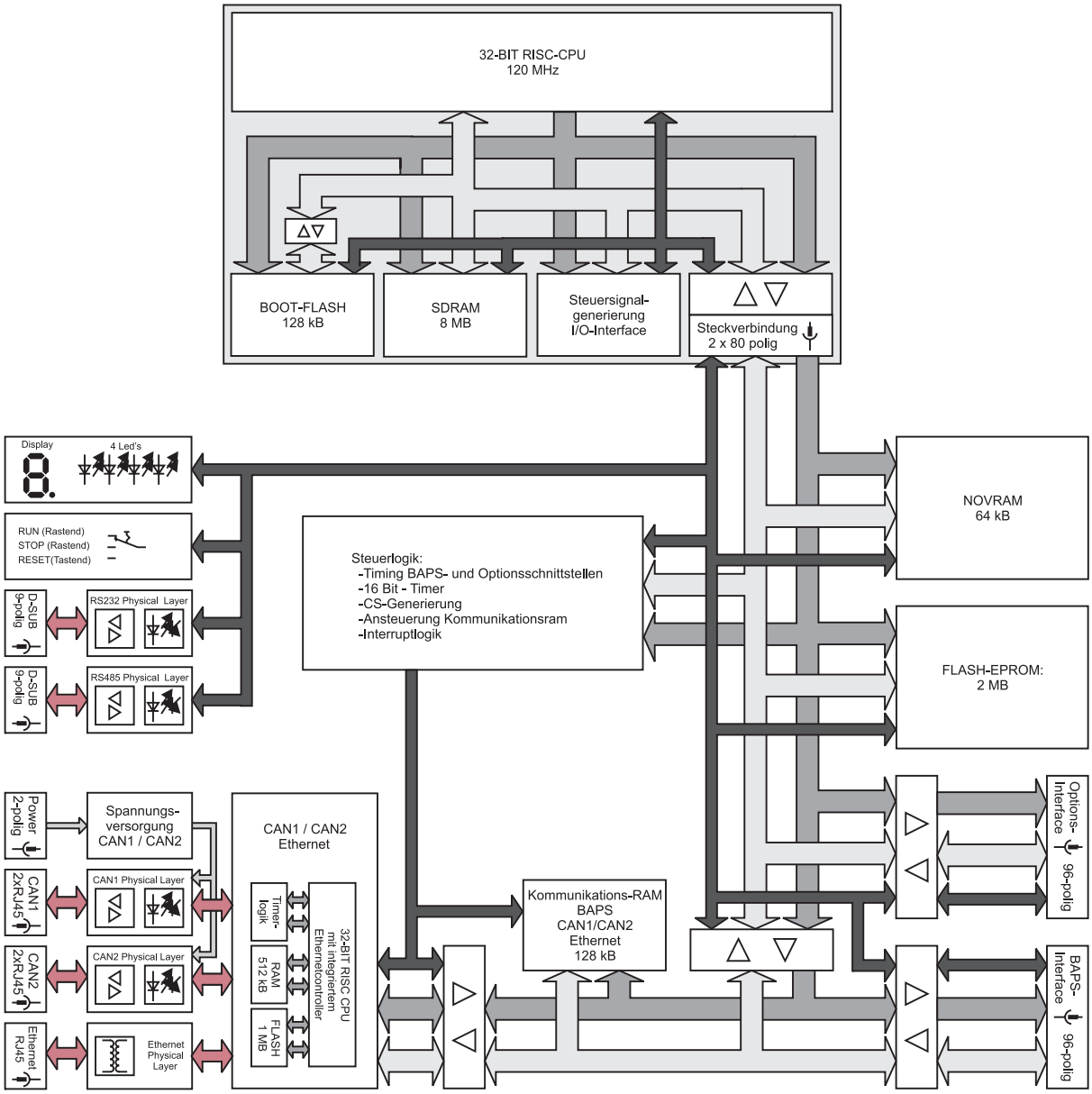
Die Aktivierung des CPU-internen Caches führt zu einer erheblich schnelleren Bearbeitungszeit für eine lineare Befehlsfolge, garantiert aber nicht mehr zu jedem Zeitpunkt die gleiche Bearbeitungszeit für eine lineare Befehlsfolge!

Funktionalität	Typenschlüssel - Funktion Steuerung
Betriebssystem, ohne Cache-Aktivierung, ohne Ethernet	I002
Betriebssystem, ohne Cache-Aktivierung, mit Ethernet	I102
Betriebssystem, mit Cache-Aktivierung, ohne Ethernet	K002
Betriebssystem, mit Cache-Aktivierung, mit Ethernet	K102

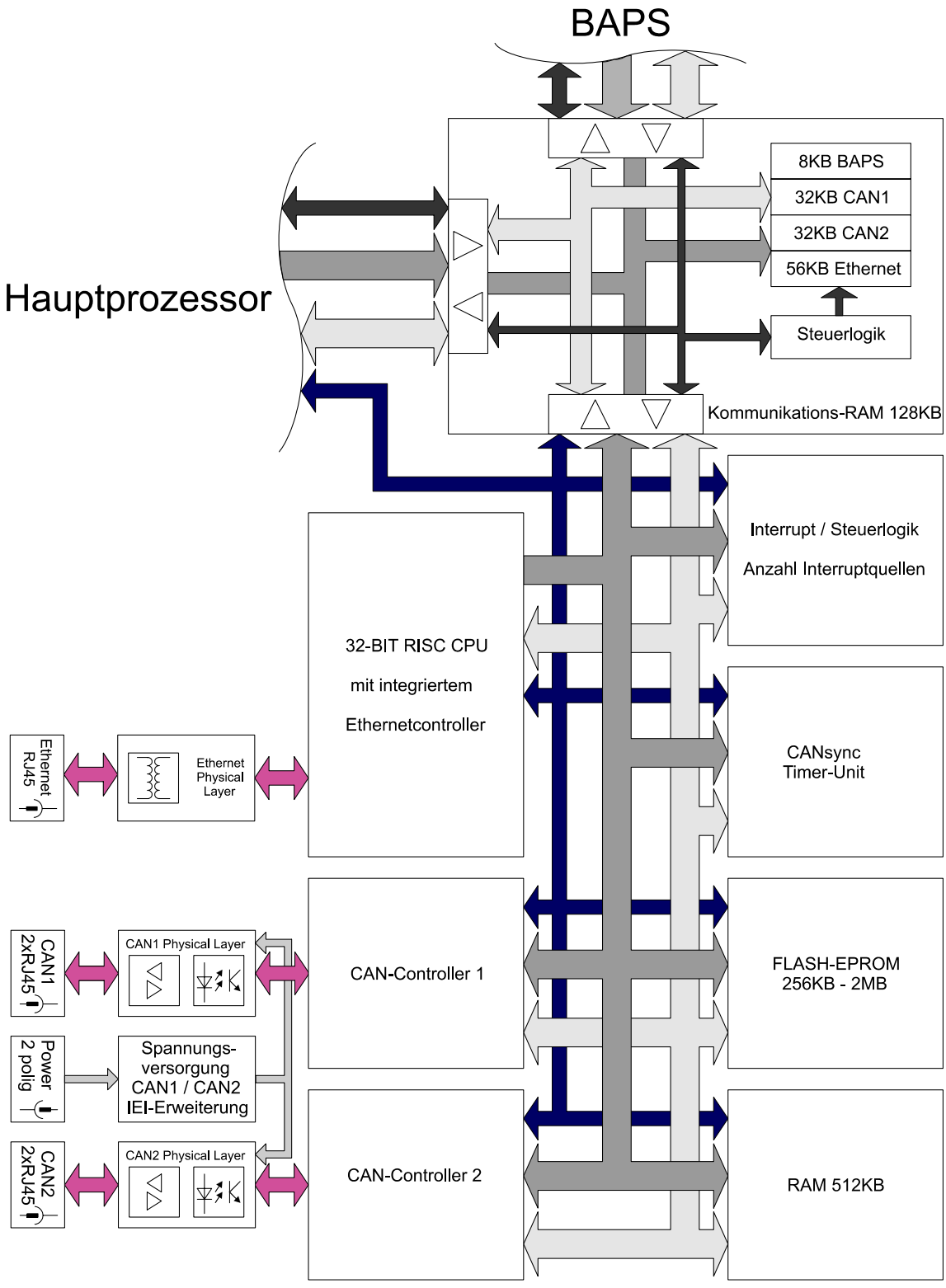
HINWEIS

Die Aktivierung des CPU-internen Caches führt zu einer erheblich schnelleren Bearbeitungszeit für eine lineare Befehlsfolge, garantiert aber nicht mehr zu jedem Zeitpunkt die gleiche Bearbeitungszeit für eine lineare Befehlsfolge!

2.3 Funktionsstruktur



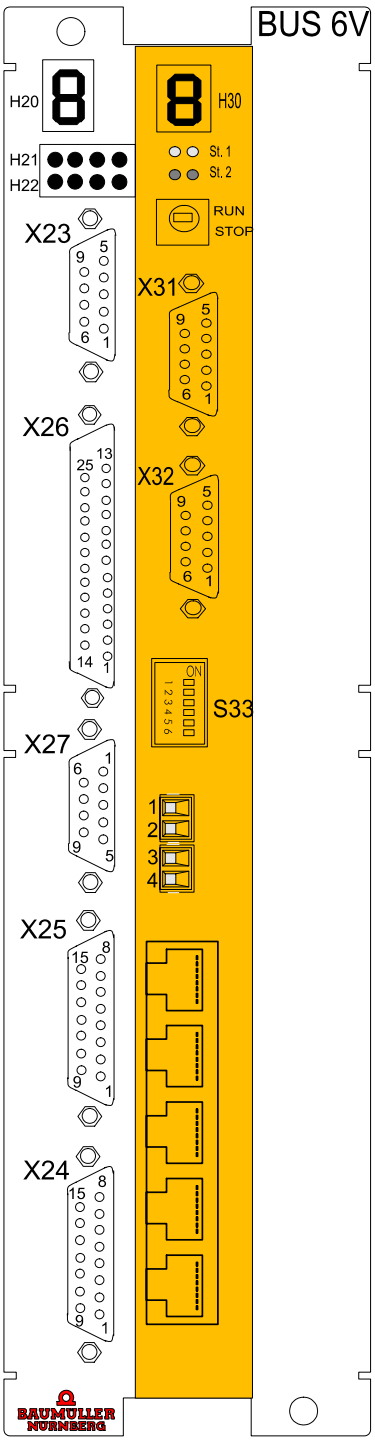
Blockschaltbild Ωmega Drive-Line II



Blockschaltbild Ethernet

3 INSTALLATION

3.1 Anzeigen- und Bedienelemente



Betriebs-
zustands-
anzeige

LED-Anzeigen

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

X31
RS232

X32
RS485

S33
Adresse CANsync

X34 reserviert
reserviert
Masse CANsync
+24 V CANsync

X35
CANsync 1. Knoten

X36
CANsync 1. Knoten

X37
CANsync 2. Knoten

X38
CANsync 2. Knoten

X39
Ethernet (optional)

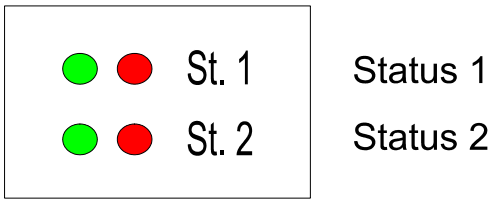
Ωmega Drive-Line II

3.2 Display

3.2.1 Siebensegmentanzeige

Zur Bedeutung der angezeigten Zahlen und Buchstaben siehe „Die Ωmega Drive-Line II Board Siebensegmentanzeige“ auf Seite 40.

3.2.2 LED-Anzeige



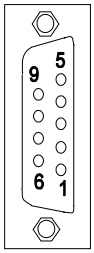
LEDs von Ωmega Drive-Line II

Die vier LEDs (links grün und rechts rot) sind frei programmierbar. Zur Programmierung siehe „Die Ωmega Drive-Line II Board Funktionen“ auf Seite 47.

3.3 Steckerbelegung

RS232 Schnittstelle (PROPROG wt II)

X 31 SUB-D-Buchse



Sub-D-Buchse

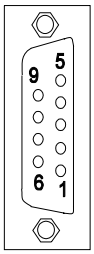
Pin Nr.	Belegung
1	nicht belegt
2	TxD (Transmit Data)
3	RxD (Receive Data)
4	verbunden mit Pin 6
5	GND (Signal Ground)
6	verbunden mit Pin 4
7	CTS (Clear to Send)
8	RTS (Request to Send)
9	nicht belegt

HINWEIS

Signal Ground der RS232- und RS485-Schnittstelle sind miteinander verbunden.

RS485 Schnittstelle (Terminal-Schnittstelle)

X 32 SUB-D-Buchse



Sub-D-Buchse

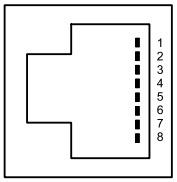
Pin Nr.	Belegung
1	TxD- (Transmit Data negativ)
2	VCC (+5V Ausgang zur Versorgung für externe Sender / Empfänger)
3	GND (Signal Ground RS232 / RS485)
4	GND (Signal Ground RS232 / RS485)
5	RxD- (Receive Data negativ)
6	RxD+ (Receive Data positiv)
7	GND (Signal Ground RS232 / RS485)
8	GND (Signal Ground RS232 / RS485)
9	TxD+ (Transmit Data positiv)

HINWEIS

Signal Ground der RS232- und RS485-Schnittstelle sind miteinander verbunden.

CANsync-Knoten 1

X 35 / X36 RJ45-Buchse



RJ45-Buchse

Die Pins von X35 und X36 mit der gleichen Nummer sind auf der Leiterplatte miteinander verbunden, d. h. die CANsync-Schnittstelle wird an beiden Buchsen zur Verfügung gestellt.

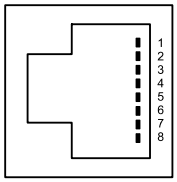
Pin Nr.	Belegung
1	GND-CAN (Signal Ground CAN)
2	GND-CAN (Signal Ground CAN)
3	nicht belegt
4	CAN1-SYNC- (SYNC-Signal negativ Knoten 1)
5	CAN1-SYNC+ (SYNC-Signal positiv Knoten 1)
6	nicht belegt
7	CAN1H (CAN-Bus-Leitung dominant High Knoten 1)
8	CAN1L (CAN-Bus-Leitung dominant Low Knoten 1)

HINWEIS

Signal Ground von CANsync-Knoten 1 und Knoten 2 sind miteinander verbunden.

CANsync-Knoten 2

X37 / X38 RJ45-Buchse



RJ45-Buchse

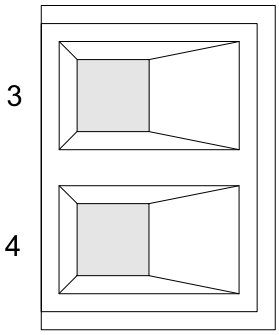
Die Pins von X37 und X38 mit der gleichen Nummer sind auf der Leiterplatte miteinander verbunden, d. h. die CANsync-Schnittstelle wird an beiden Buchsen zur Verfügung gestellt.

Pin Nr.	Belegung
1	GND-CAN (Signal Ground CAN)
2	GND-CAN (Signal Ground CAN)
3	nicht belegt
4	CAN2-SYNC- (SYNC-Signal negativ Knoten 2)
5	CAN2-SYNC+ (SYNC-Signal positiv Knoten 2)
6	nicht belegt
7	CAN2H (CAN-Bus-Leitung dominant High Knoten 2)
8	CAN2L (CAN-Bus-Leitung dominant Low Knoten 2)

HINWEIS

Signal Ground von CANsync-Knoten 1 und Knoten 2 sind miteinander verbunden.

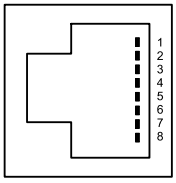
24 V Versorgungsspannung Ωmega Drive-Line II (X34) für SYNC-Signal



Pin Nr.	Belegung
3	Masse 24 V
4	+24 V-Versorgungsspannung

Ethernet Anschaltung 10/100 MBit/s (optional)

X39 RJ45-Buchse

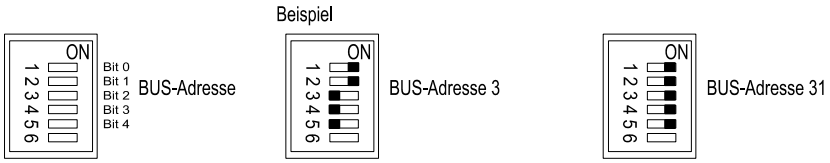


RJ45-Buchse

Pin Nr.	Belegung
1	TX+ (Transmitleitung +)
2	TX- (Transmitleitung -)
3	RX+ (Receiveleitung +)
4	nicht belegt
5	nicht belegt
6	RX- (Receiveleitung -)
7	nicht belegt
8	nicht belegt

3.3.1 Einstellung der Slavenummer

Über die DIP-Schalter 1 - 5 (S 33) auf der Ωmega-Platine (mittlere Platine in der Kassette) wird die Slavenummer binär codiert eingestellt mit der die CANsync-Slave-Anschaltung auf dem Ωmega Drive-Line II angesprochen wird. Für jede CANsync-Slave-Anschaltung an einem CANsync-Bus muß eine von den anderen CANsync-Slave-Anschaltungen verschiedene Slavenummer eingestellt werden (1 CANsync-Master-Anschaltung und bis zu 32 CANsync-Slave-Anschaltungen).



3.3.2 Hinweise zur Konfiguration

Auf der Leiterplatte befinden sich 2 CANsync-Anschaltungen. Sie dienen zur Verbindung von einem CANsync-Bus zu einem CANsync-Bus einer Subebene. Damit wird ein Netzwerk mit mehreren Ebenen gebildet.

Auf der Leiterplatte muß eine Anschaltung als CANsync-Slave und eine Anschaltung als CANsync-Master (Slave für SYNC-Signal) konfiguriert werden.

Die Funktionalität der CANsync-Master- und CANsync-Slave-Anschaltung ist im Kapitel 6 beschrieben.

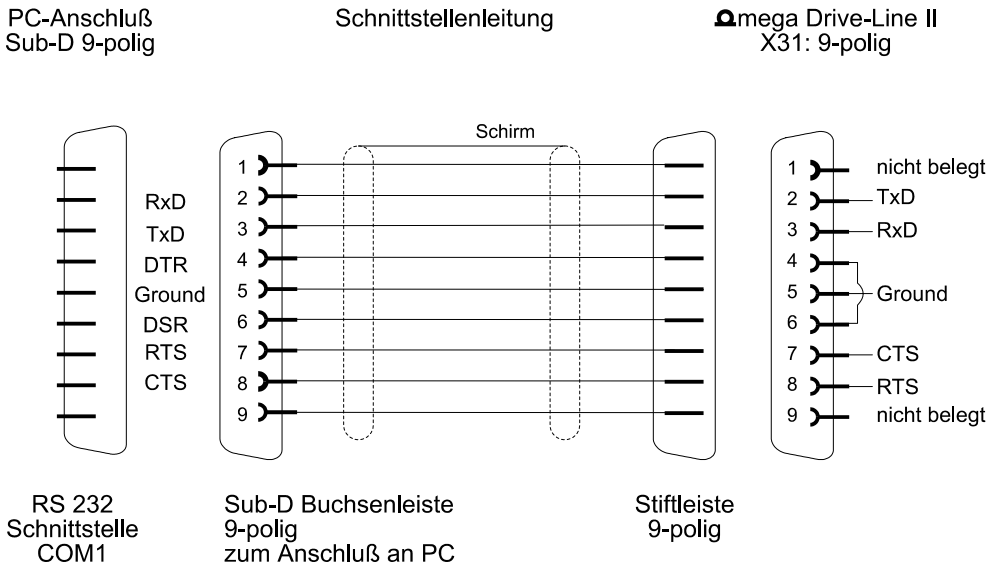
HINWEIS

Die voreingestellte IP-Adresse für Ethernet (192.168.1.“1+Dipschalter“, Dipschalter = Bit 1 bis Bit 5) kann per Software geändert werden (siehe “Einstellen von IP-Adresse und IP-Maske” auf Seite 78 und “Kommunikation über Ethernet (optional)” auf Seite 34).

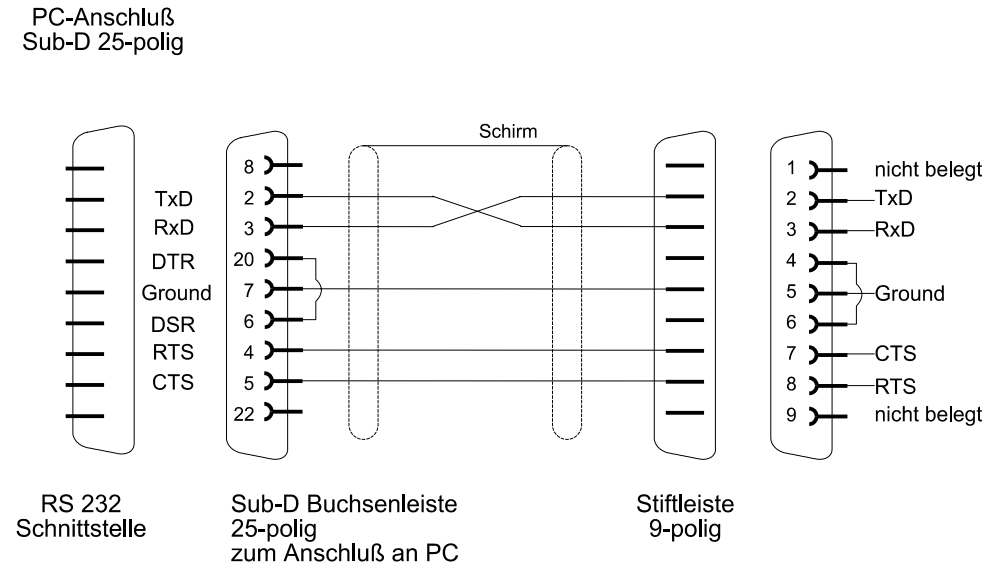
3.4 Anschlußkabel

Serielles Anschlußkabel für PC - Ωmega Drive-Line II

- PC-Anschluß 9-polig

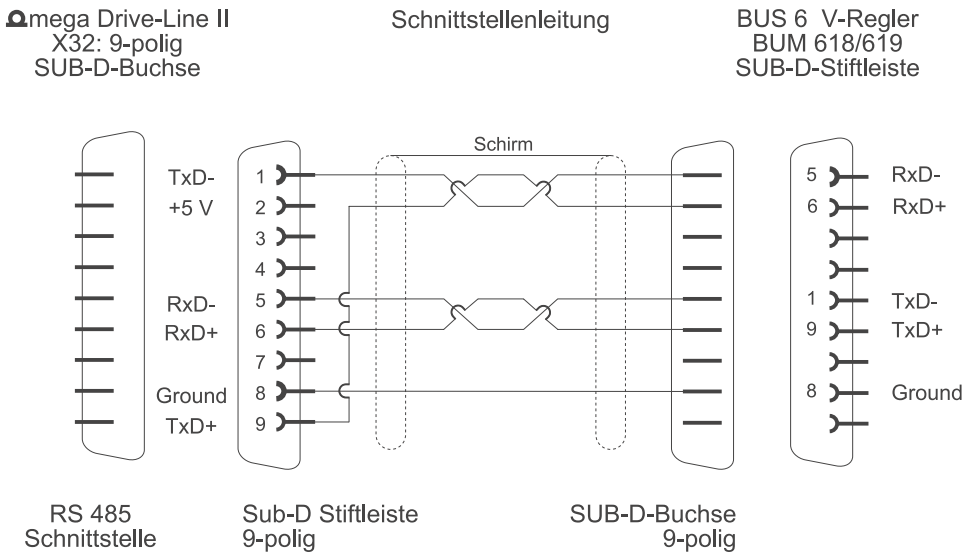


- PC-Anschluß 25-polig



Serielles Anschlußkabel für RS485

- Ωmega ⇔ BUS 6 V-Regler (4-Draht-Verbindung und Masse mit 9-pol. Stecker)



WARNUNG

Die + 5V an Pin 2 der Drive-Line II 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

Die RS485-Schnittstelle ist galvanisch getrennt und optisch entkoppelt.

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

Mehrere Geräte werden über die RS485-Schnittstelle miteinander verbunden, so daß ein Bussystem (Bus) mit mehreren Teilnehmern (Geräten) entsteht. Dieser Bus kann eine lineare Topologie, eine Sterntopologie oder eine Mischung aus beiden haben.

HINWEIS

Üblicherweise werden die differentiellen Einzelleitungen des Busses miteinander verdrillt, wodurch eingekoppelte elektromagnetische Störungen auf beide Einzelleitungen gleichmäßig wirken. Durch die Differenzauswertung am Empfangsbaustein können diese Störungen dann unterdrückt werden (Gleichtaktunterdrückung). Die Verdrillung steigert somit die Übertragungsqualität.

Die Fa. Baumüller empfiehlt, zusätzlich zu den Leitungen der 2-Draht- bzw. 4-Draht-Verbindung auch die Masse-Leitung mitzuführen, wodurch 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 (oder nur an einer einzigen Stelle) auf der gesamten Verbindungsstrecke mit dem Schirm verbunden werden!

Verwendung von Pull-Up- / Pull-Down- und Abschlußwiderständen

Beim Einsatz differentieller Schnittstellen besteht oftmals die Notwendigkeit, Pull-Up- / Pull-Down-Widerstände und/oder Abschlußwiderstände (auch Bus-Abschlußwiderstände) den Verbindungskabeln hinzuzufügen.

Durch den Einsatz von Pull-Up- / Pull-Down-Widerständen wird sichergestellt, daß an einer differentiellen Empfangseinheit stets ein definiertes Potential anliegt, auch wenn gerade keine Zeichen übertragen werden und die Sendeeinheiten der Geräte am Bus ihre Treiber hochohmig schalten (protokollabhängig).

Fehlen solche Widerstände, so kann die Leitung ein undefiniertes Potential haben 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 des Busses mit Abschlußwiderständen abgeschlossen. Fehlen diese, können Reflexionen auftreten, welche zu Empfangsfehlern führen. Die Verwendung von Abschlußwiderständen verbessert somit die Übertragungsqualität.

Bei einem Bus mit mehreren Teilnehmern muß man darauf achten, daß auf den Busleitungen nicht zu viele Pull-Up- / Pull-Down- und Abschlußwiderstände parallel liegen.

Im Gerät Ωmega Drive-Line II sind die erforderlichen Pull-Up- / Pull-Down- und Abschlußwiderstände bereits vorhanden und können jeweils getrennt voneinander über die Applikations-Software zu- und abgeschaltet werden.

Die genaue Vorgehensweise dazu kann der Online-Beschreibung des FBs "TER_INIT" in der Bibliothek "SYSTEM2_DLII_20bd00" oder höher entnommen werden.

Verwendung von 2-Draht- bzw. 4-Draht-Verbindung

Über eine 4-Draht-Verbindung werden Zeichen vollduplex gesendet und empfangen. Vollduplex bedeutet, daß Zeichen über ein Leitungspaar gesendet und über das andere Leitungspaar empfangen werden können.

Das 3964R-Protokoll® ist z. B. ein Protokoll, über das Zeichen vollduplex gesendet und empfangen werden und das eine 4-Draht-Verbindung voraussetzt.

Im Unterschied dazu werden über eine 2-Draht-Verbindung die Zeichen halbduplex gesendet und empfangen. Halbduplex bedeutet, daß Zeichen über ein gemeinsames Leitungspaar durch Umschalten der Richtung gesendet und empfangen werden können.

HINWEIS

Bei einer 2-Draht-Verbindung muß die Verbindung von TxD+ mit RxD+ und TxD- mit RxD- an den Enden des Verbindungskabels erfolgen.

Bei der 2-Draht-Verbindung werden, im Gegensatz zur 4-Draht-Verbindung, zwei Leitungen weniger benötigt. Das Fehlen der beiden Leitungen muß durch einen exakten Protokollablauf beim Senden und Empfangen "kompensiert" werden:

Das Protokoll muß zu jedem Zeitpunkt sicherstellen, daß Zeichen nie in beide Richtungen gleichzeitig gesendet werden, d. h. daß nur eine Sendeeinheit zur Zeit aktiviert ist und daß alle anderen Sendeeinheiten dann abgeschaltet (hochohmig) sind.

Beim Ωmega Drive-Line II wird 2-Draht- bzw. 4-Draht-Verbindung über die Applikations-Software ausgewählt.

Zu beachten ist dabei, daß durch das Deaktivieren der jeweiligen Sendeeinheiten die Verbindungsleitungen hochohmig werden, falls keine Pull-Up- / Pull-Down- Widerstände programmiert wurden.

Die genaue Vorgehensweise dazu kann der Online-Beschreibung des FBs "TER_INIT" in der Bibliothek "SYSTEM2_DLII_20bd00" oder höher entnommen werden.

3.5 Zubehör

Programmierkabel (seriell RS232)

Leitungstyp: K-SS-01-xx (Sub-D 9 pol., Sub-D 9 pol.):

Type	Länge [m]	Artikelnummer
K-SS-01-03	3	213 846
K-SS-01-05	5	213 283
K-SS-01-15	15	231 086

Kabel für CANsync / CAN

Leitungstyp: K-CAN-33-xx (RJ-Stecker, RJ-Stecker):

Type	Länge [m]	Artikelnummer
K-CAN-33-0-0,5	0,5	324 497
K-CAN-33-0-01	1	324 498
K-CAN-33-0-02	2	324 499
K-CAN-33-0-03	3	324 500
K-CAN-33-0-04	4	324 501
K-CAN-33-0-05	5	324 502
K-CAN-33-0-10	10	324 503

Leitungstyp: K-CAN-13-xx (RJ-Stecker, Sub-D-Stecker):

Type	Länge [m]	Artikelnummer
K-CAN-13-0-0,5	0,5	324 504
K-CAN-13-0-01	1	324 505
K-CAN-13-0-02	2	324 506
K-CAN-13-0-03	3	324 507
K-CAN-13-0-04	4	324 508
K-CAN-13-0-05	5	324 509
K-CAN-13-0-10	10	324 510

Abschlußstecker für CANsync / CAN

K-CAN-T1-S (D-SUB 9-pol., male) Art.-Nr. 313 910

K-CAN-T2-S (D-SUB 9-pol., female) Art.-Nr. 313 911

K-CAN-RJ Art.-Nr. 313 264

Kabel für Ethernet

Leitungstyp: K-ETH-33-xx (RJ-Stecker, RJ-Stecker, Kabel CAT5):

Type	Länge [m]	Artikelnummer
K-ETH-33-0-0,5	0,5	325 160
K-ETH-33-0-01	1	325 161
K-ETH-33-0-02	2	325 162
K-ETH-33-0-03	3	325 163
K-ETH-33-0-04	4	325 317
K-ETH-33-0-05	5	325 164
K-ETH-33-0-10	10	325 165

4 ΩMEGA DRIVE-LINE II UND PROPROG WT II

4.1 PROPROG wt II - ein effizientes, leistungsfähiges und umfassendes Programmierwerkzeug

PROPROG wt II 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 II 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 II 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).

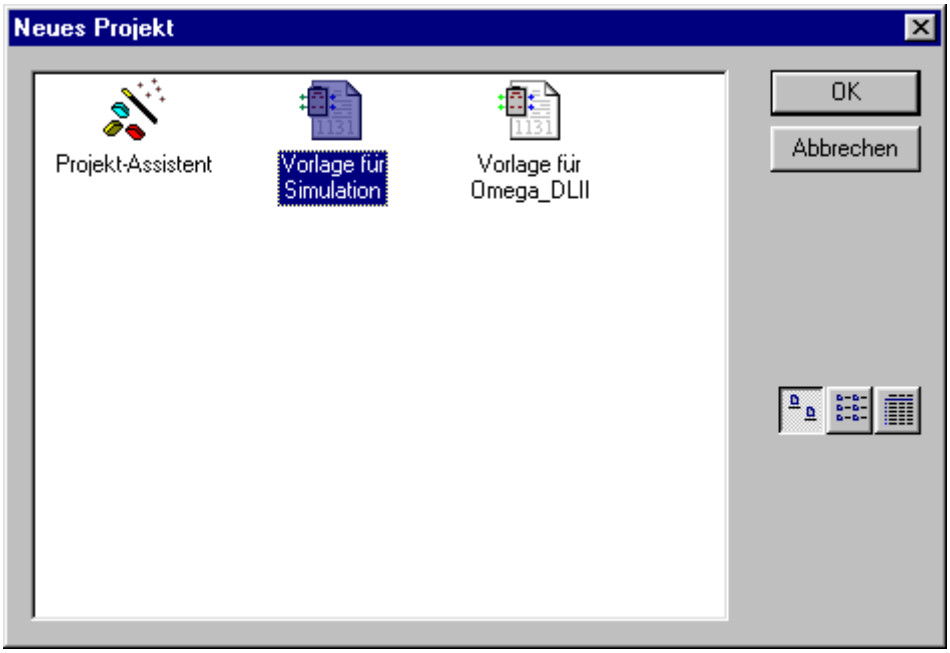
Zum Programmieren in den einzelnen Sprachen steht jeweils ein Editor-Assistent zur Verfügung, der ein schnelles und sehr einfaches Einfügen von vorbereiteten Schlüsselwörtern, Anweisungen, Operatoren, Funktionen und Funktionsbausteinen in den einzelnen Arbeitsblättern ermöglicht. Der Editor-Assistent 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.

4.2 Ωmega Drive-Line II Projekt

Ein neues Ωmega Drive-Line II Projekt wird unter PROPROG wt II über das Menü "Neues Projekt" begonnen. Mit der „Vorlage für Omega_DLII“ wird ein Ωmega Drive-Line II Projekt geöffnet. Die Ωmega Drive-Line II CPU in diesem Projekt ist in der Konfiguration voreingestellt.



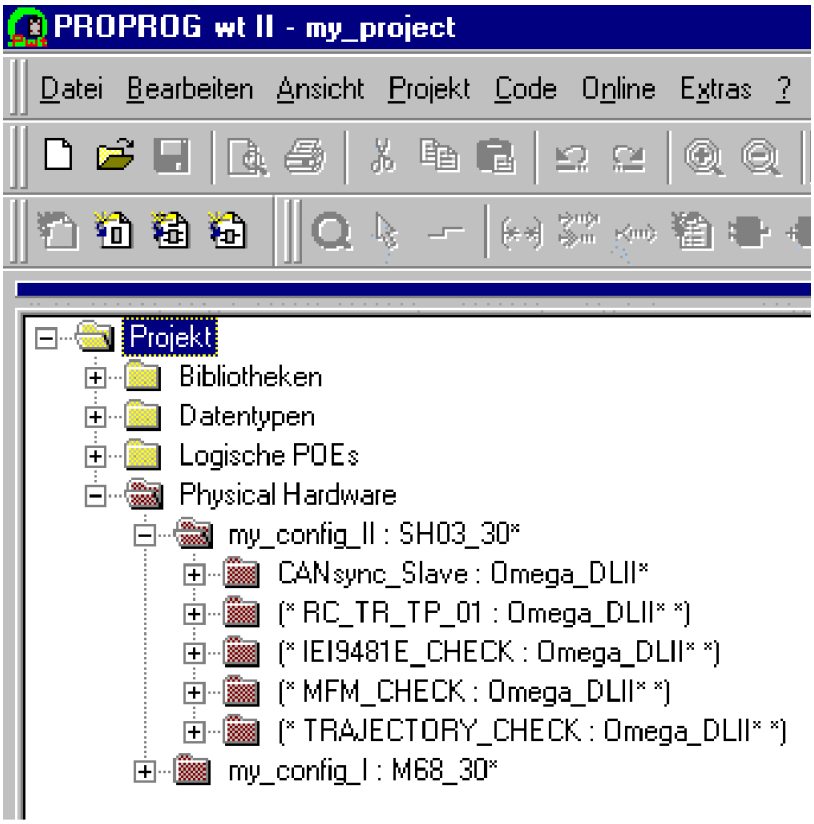
Ωmega Drive-Line II Projektvorlage unter "Neues Projekt".

4.3 Ωmega Drive-Line II Konfiguration

Als IEC 61131-3 Programmieroberfläche können mit PROPROG wt II verschiedene Zielsysteme (CPUs) programmiert werden. Es können auch verschiedene Zielsysteme in einem Projekt programmiert werden. Die Erstellung eines Programms für das Zielsystem Ωmega Drive-Line II erfolgt unter "Physikalischer Hardware" mit der Ωmega Drive-Line II Konfiguration.

Mit der Vorlage wird eine Ωmega Drive-Line II Konfiguration geöffnet. Die Ωmega Drive-Line II Konfiguration hat unter Eigenschaften den SPS-Typ SH03_30.

Eine Konfiguration besteht aus mindestens einer Ressource. Die Ressource beinhaltet den Ωmega Drive-Line II spezifischen Datenbereich, die Kommunikationsquelle, globale Variablenarbeitsblätter und die Tasks mit dem Programm.



Einstellung unter "Physical Hardware" Eigenschaft einer Ωmega Drive-Line II Konfiguration.

Übersicht der Einstellungen, für das Einfügen eines Ωmega Drive-Line II Zielsystems, innerhalb der Physikalischen Hardware:

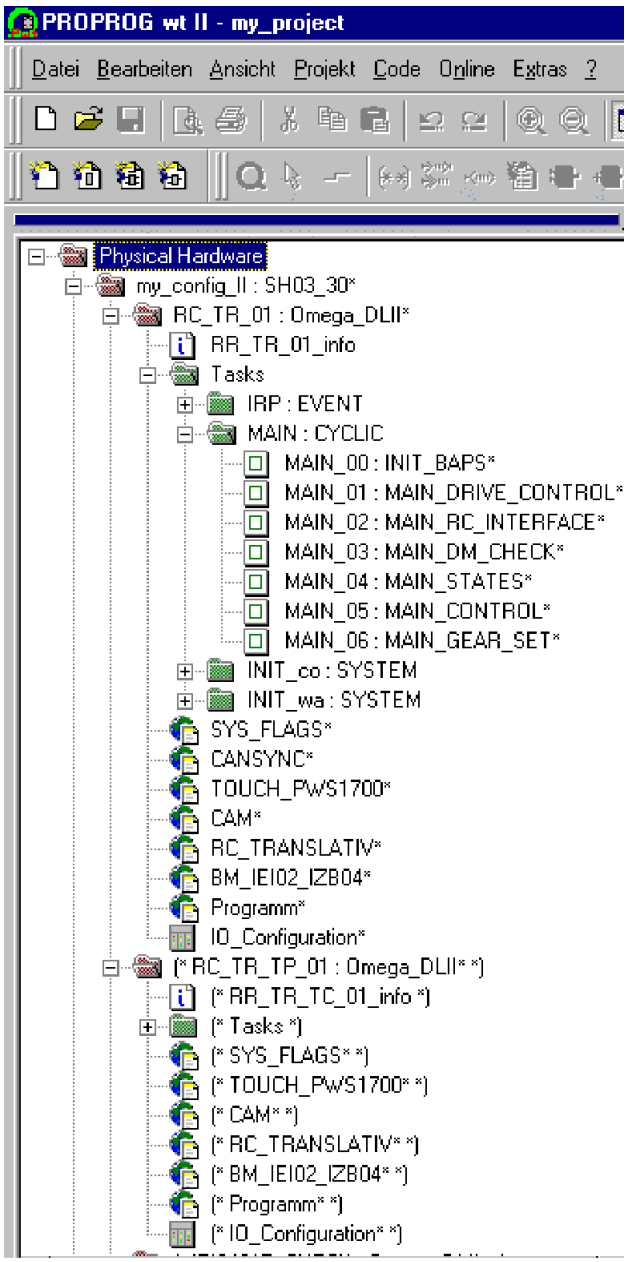
Projektvorlage	Konfiguration	Ressource
Omega_DLII	SH03_30	Omega_DLII

4.4 Ωmega Drive-Line II Ressource

Die Ressource beinhaltet die Ωmega Drive-Line II 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.)

Der Ωmega Drive-Line II Konfiguration können mehrere Ressourcen zugeordnet werden. Somit ist eine komplette Applikation mit mehreren Antrieben in einem Projekt umsetzbar.



Beispiel einer Ωmega Drive-Line II Konfiguration mit mehreren Ressourcen.

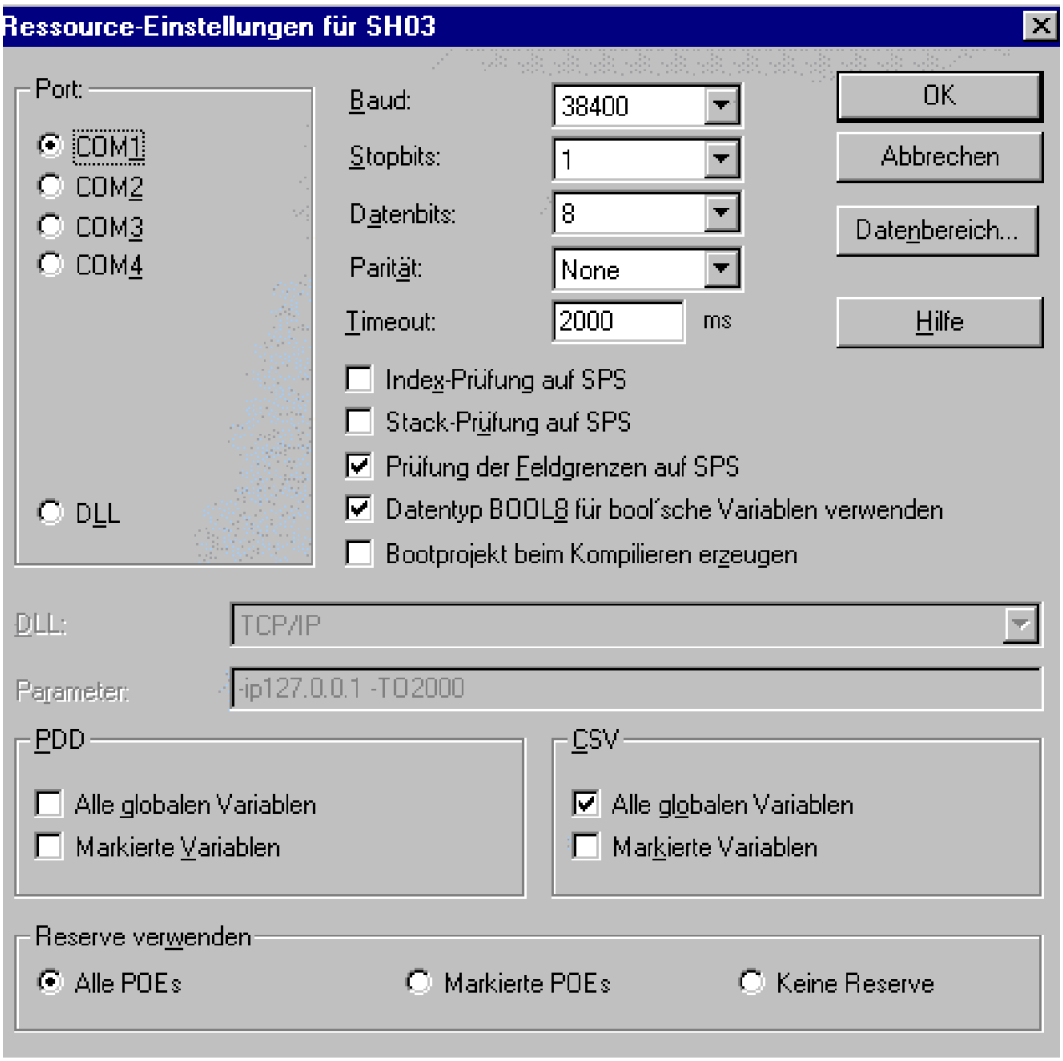
4.4.1 Kommunikation und Verbindung

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

Die Verbindung wird über X31 am Ωmega Drive-Line II aufgebaut.



Ressource-Einstellung innerhalb einer Ωmega Drive-Line II 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. Der 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 parametrierten Datenbereichsgrenzen (remanent, nicht remanent) nicht verletzt. 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 den Ωmega Drive-Line II muß die Einstellung aktiviert sein.
- "Bootprojekt beim kompilieren erzeugen": Bei aktivierter Funktion wird bereits beim Compilieren 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 Handbuch PROPROG wt II.)
- "CSV": Einstellungen zur Bereitstellung von Variablen für den OPC-Server. (Siehe auch PROPROG wt II Programmierhandbuch.)
- "Reserve verwenden": Speicherreserve um Änderungen in Funktionsbausteine und Funktionen mit der Funktion "Online-Änderung" durchführen zu können. (Siehe auch PROPROG wt II Programmierhandbuch.)

Die Ressource-Einstellung kann für jede Ωmega Drive-Line II - Ressource separat vorgenommen werden. Die serielle oder Ethernet-Kommunikationsquelle

- COM1
- COM2
- COM3
- COM4
- DLL

wird genutzt

  – zur Ressourcenkontrolle.
  – zum Senden des kompilierten Projekts.
  – zum Debuggen.
  – zur Verbindung zum OPC-Server.

Kommunikation über Ethernet (optional)

Die Kommunikation über Ethernet (Anwendung siehe “Ethernet (optional)” auf Seite 75) an der Ωmega Drive-Line II Buchse X39 wird wahlweise wie folgt eingestellt:

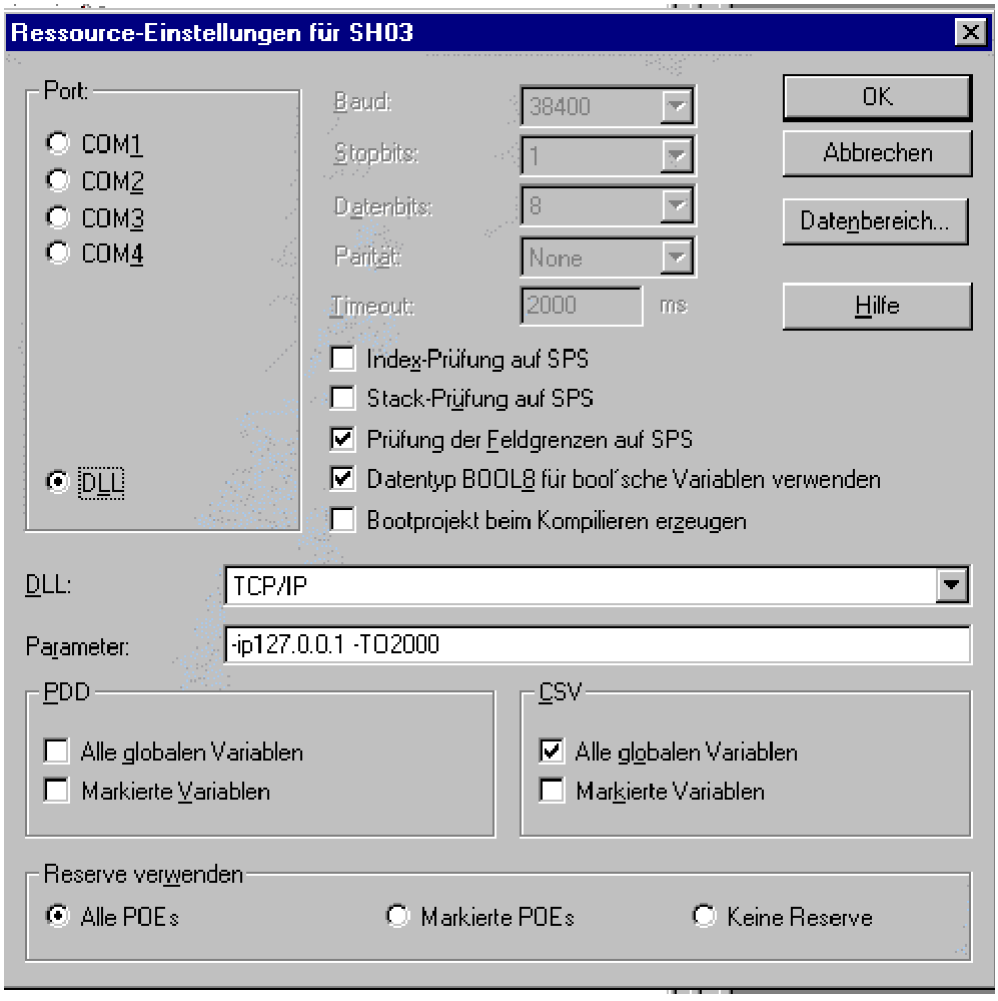
- In Einstellungen muß auf DLL geschaltet werden. Im Fenster "DLL" kann die mit Namen hinterlegte IP-Adresse ausgewählt werden oder im Fenster "Parameter" kann die IP-Adresse manuell eingegeben werden. (Z. B. -ip 192.168.1.1 für den Auslieferungszustand und DIP-Schalter = 0.)
- Um IP-Adressen mit Namen zu hinterlegen, müssen in der PROPROG wt II Datei (%INSTALLATIONSVERZEICHNIS%\PROPROG\) **mwt.ini** die IP-Adressen eingetragen werden. Für diese sind unter dem Länder-Code, z. B. englisch 001, deutsch 049, Namen einzutragen.

Beispiel: mwt.ini mit Ethernet-Adresseinträge zur Auswahl über Synonym.

```
[COMMUNICATION]
DLL001=plc\socomm.dll -ip 127.0.0.1 -TO2000
DLL003=plc\socomm.dll -ip 192.168.1.1 -TO2000
DLL002=plc\socomm.dll -ip 192.075.191.184 -TO2000
DLL004=plc\socomm.dll -ip 192.075.191.185 -TO2000

[COMMUNICATION001]
NAME001=TCP/IP
NAME002=TCP/IP Omega DL II (factory default)
NAME003=TCP/IP my lab sample
NAME004=TCP/IP testmachine

[COMMUNICATION049]
NAME001=TCP/IP
NAME002=TCP/IP Omega DL II (Werkseinstellung)
NAME003=TCP/IP mein Testgerät im Labor
NAME004=TCP/IP Omega DL II Testmaschine
```



Einstellung DLL für Kommunikation über Ethernet mit Adresse über Synonym oder manueller Eingabe.

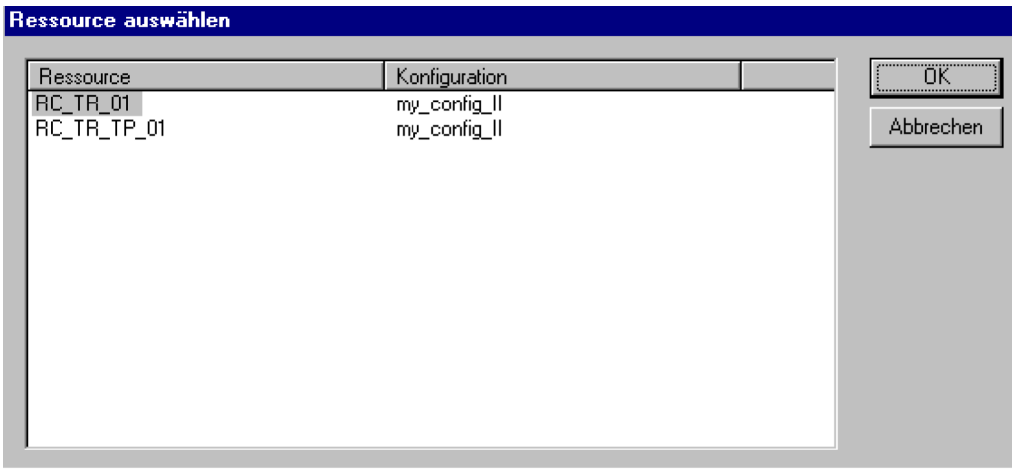
HINWEIS

Bitte beachten Sie auch die Einstellungen der Netzwerk-Verbindungen unter Windows-Systemsteuerung (siehe Seite 82).

4.4.2 Kontrolldialog für Ressourcen

Mit dem Kontrolldialog für Ressourcen wird die Programmübermittlung zum Ωmega Drive-Line II und der Betriebszustand des Ωmega Drive-Line II eingestellt.

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

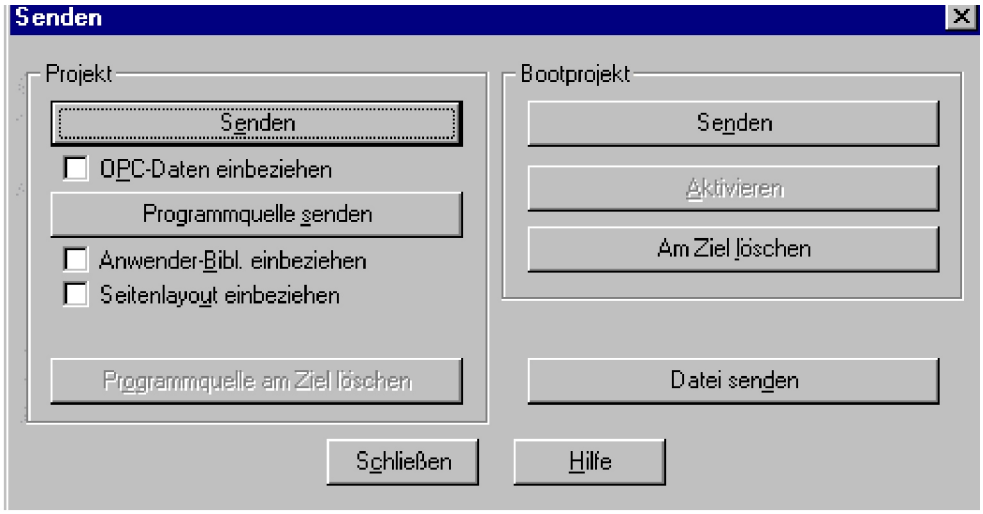


Nach Auswahl der Ressource erscheint der Kontrolldialog für die Ressource des zugeordneten Ωmega Drive-Line II.

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.



Ωmega Drive-Line II 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 Ωmega Drive-Line II Flash-Speicher abgelegt. Mit "**Aktivieren**" wird das Projekt vom Ωmega Drive-Line II 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 gesendet. Das Bootprojekt bleibt unverändert im Ωmega Drive-Line II Flash-Speicher. Nach dem nächsten Hardware-Reset oder Steuerung Aus -> Ein ist wieder das Bootprojekt aktiv!

HINWEIS

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

Nach "**Projekt senden**" kann das Projekt auf dem Ωmega Drive-Line II gestartet werden.
Wurde ein Bootprojekt gesendet wird es unter „**Senden**“ mit „**Bootprojekt aktivieren**“ im Ωmega Drive-Line II geladen und kann dann gestartet werden:

Die Ωmega Drive-Line II Ressourcenkontrolle im Zustand "STOP".

- **Stop**: Programmausführung wird angehalten.
- **Reset**: Löscht das Projekt auf dem Ωmega Drive-Line II (nicht das Bootprojekt!).
- **Senden**: Ruft die Programmübermittlung auf.
- **Kaltstart**: Es wird die System-Task Kaltstart durchgeführt, wobei alle Variablen mit ihrem Default-Wert initialisiert werden. Wird ein neues Projekt gesendet, wird einmalig ein Kaltstart durchgeführt, wenn der Hardware-Schalter auf "RUN" steht. Retain-Variablen werden dabei auf ihren Initialisierungswert gesetzt.
- **Warmstart**: Es wird die System-Task Warmstart mit Hardwareschalter auf RUN durchgeführt. Die globalen Retain-Variablen behalten ihren letzten Wert. Nur möglich, wenn der Hardware-Schalter auf "RUN" steht.

HINWEIS

Wird nach "Projekt senden" der Hardware-Schalter von STOP auf RUN gestellt, wird einmalig der Kaltstart durchgeführt. Bei jeden weiteren Wechsel von STOP auf RUN wird der Warmstart durchgeführt.

- **Heißstart**: Es wird keine Initialisierungstask, also weder Kalt- noch Warmstart, durchlaufen, sondern direkt das zyklische Projekt ausgeführt.
- **Fehler**: Hier können von der Steuerung anstehende Fehler- und Warnmeldungen ausgelesen werden, wenn der Button aktiviert ist. Die Aktivierung erfolgt, wenn eine Laufzeitfehlermeldung generiert wird. Durch Drücken des aktiven Fehler-Buttons werden die Fehlereinträge von der Steuerung abgefragt und im Fehler- bzw. Warnungsmeldefenster angezeigt.
- **Hochladen**: Upload-Funktion wird z. Z. nicht unterstützt.
- **Info**: Informationen über die OmegaOS-Version, die Firmware-Nummer, der verwendete Speicherausbau sowie die internen Systemzustände der Steuerung.

Der Kontrolldialog für die Ressource bietet unter "Info" allgemeine Informationen zum Projekt auf dem Ωmega Drive-Line II. Folgende Inhalte sind dargestellt:

- "Version-SPS" mit Versionsstand des Ωmega Drive-Line II Laufzeitsystems.
- "Version-Firmware" mit Versionsstand der Ωmega Drive-Line II Firmware zu der Firmwarebibliothek.
- "Projekt" mit dem Namen des aktiven Projekts.
- Betriebs- und Fehlerstatus sowie aktiver Debug-Modus Breakpoints, Forcen.
- Gesamter Speicherbereich Remanente Daten.
- Freier Programmspeicher.
- Freier Speicher Systemdaten Laufzeitsystem (für Auslastung Oszilloskopfunktion).
- CPU-Auslastung, z. B. als Kontrolle für Schleifendurchläufe in einer parametrierten Taskzeit.
- Aktualisierungszeit der Online-Darstellung im Debug-Online-Modus.

Ωmega Drive-Line II Ressource-Informationen.

4.4.3 Die Ωmega Drive-Line II Board Siebensegmentanzeige

Die Start/Stop-Funktionen sind entweder über den Kontrolldialog der Ressource von PROPROG wt II oder über den RUN/STOP Schalter ausführbar. Soll ein Start über PROPROG wt II ausgeführt werden, muß dazu der RUN/STOP Schalter auf Stellung "RUN" stehen.

Der RUN/STOP Schalter auf dem Ωmega Drive-Line II Board hat folgende Schaltstellungen:

Taster-/Schalterstellung	Zustand	Siebensegmentanzeige
Unten (Taster)	**RESET**	dunkel solange der Taster gedrückt bleibt. Nach Loslassen wechselt der Taster automatisch in die Mittelstellung "STOP".
Mitte (Schalter)	**STOP**	
	Ohne Projekt	**0**
	Mit Projekt	**1**
Oben (Schalter)	**RUN**	
	(Kontrolle Ressourcenmanager PROPROG wt II aktiv)	
	Kein Projekt auf der Steuerung.	**0**
	Über den Kontrolldialog der Ressource wurde ein Stop ausgeführt oder ein neues Projekt aktiviert.	**1**
	Initialisierung wird durchlaufen. (Kalt- bzw. Warmstart)	**2**
	Initialisierungsphase abgeschlossen, die zyklischen Programmteile werden gerade ausgeführt.	**3**
	Das Bootprojekt auf der Steuerung wird gerade gelöscht.	**C**
	Ein neues Bootprojekt wird gerade ins Flash kopiert.	**L. < ---- > L**
	Der Vorgang Bootprojekt ins Flash kopieren ist abgeschlossen. Jetzt kann das Bootprojekt aktiviert werden.	**A**

Die Siebensegmentanzeige kann u. U. nach Einschalten der Versorgungsspannung oder auch in seltenen Fällen während des Betriebes verschiedene andere Symbole zeigen, die auf einen Geräte-Defekt hinweisen. Das Gerät muss dann in der Regel zur Reparatur gegeben werden!

Diese Anzeigen können sein:

Taster-/Schalterstellung	Bedeutung	Siebensegmentanzeige
Unabhängig von Schalterstellung	Querbalken im Display Vermutlich liegt ein defektes SDRAM vor	"**-**", "**C**", "**E**" oder "**F**"
	Fehlendes Betriebssystem Vermutlich liegt ein defektes FLASH vor	"**U**" oder das Display bleibt komplett dunkel.

4.4.4 Datenbereich

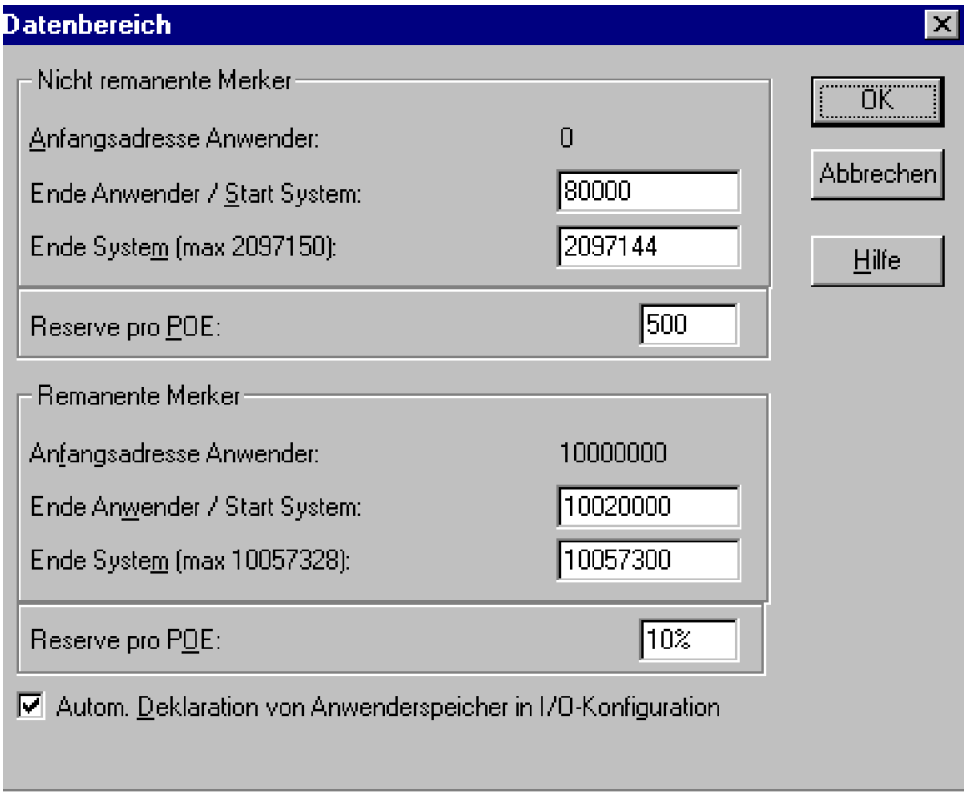
Der Datenbereich enthält die Ωmega Drive-Line II spezifische Einstellung des physikalischen Adressbereichs.

Der Datenbereich gliedert sich in **Nicht Remanente Merker** und **Remanente Merker**. In beiden Bereichen befindet sich ein Systembereich. Im Systembereich erfolgt die Adressvergabe der symbolischen Programmvariablen über den Compiler.

In den beiden Anwenderbereichen, nicht remanent und remanent, können vom Anwender absolute Adressen vergeben werden.

Darüberhinaus gibt es einen Ωmega Drive-Line II spezifischen Bereich für Schnittstellen, z. B. Optionsschnittstellen (siehe “Die Basisadressen der Optionsschnittstellen” auf Seite 60).

Die Reserve dient der Compiler-Funktionalität "Online-Änderungen" senden und reserviert Programmspeicher.



Ressource Ωmega Drive-Line II spezifischer Datenbereich ohne Anwenderbereich Remanente Merker.

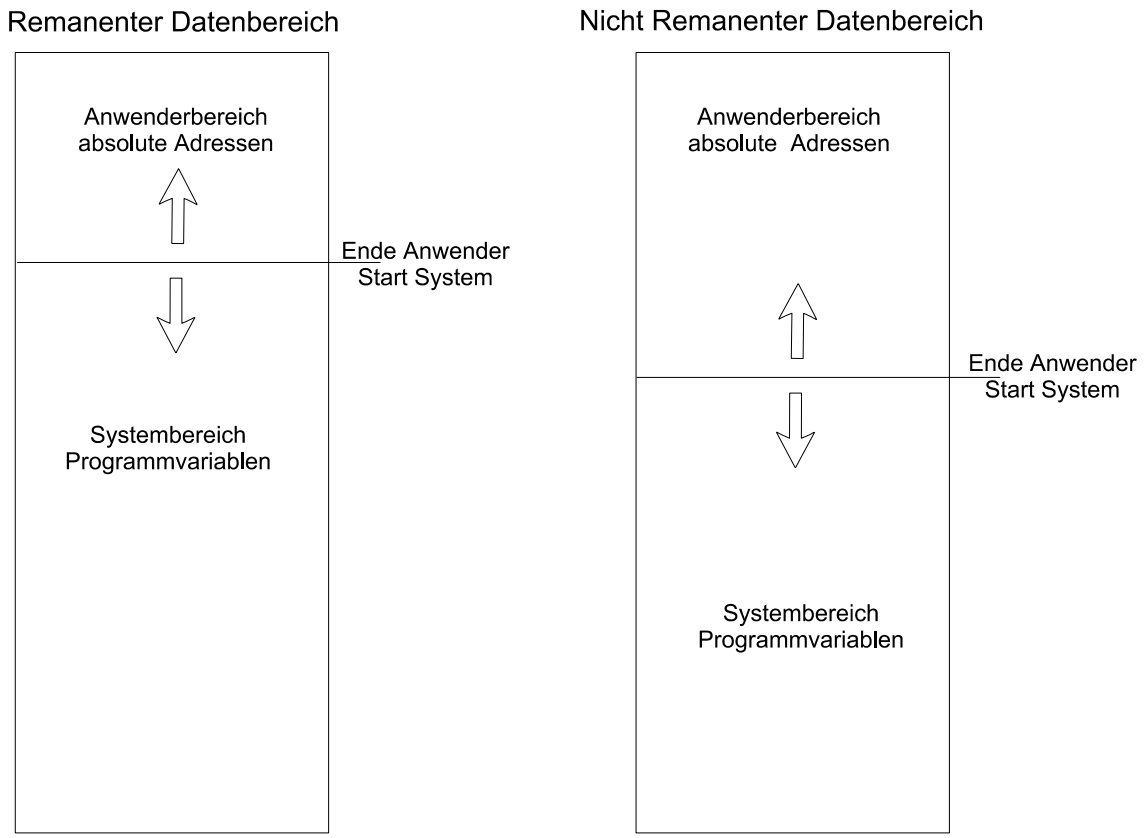
Der Ωmega Drive-Line II Datenbereich gliedert sich in die Bereiche remanente und nicht remanente Daten. Der Bereich der absoluten Adressen ist abhängig vom Projekt einzustellen.

Der Standard-Anwenderbereich für die Vergabe absoluter Adressen innerhalb einer Ωmega Drive-Line II Ressource im **nicht remanenten** Bereich ist:

- %MB 0 - %MB 79999

Der Standard-Anwenderbereich für die Vergabe absoluter Adressen innerhalb einer Ωmega Drive-Line II Ressource im **remanenten** Bereich ist:

- %MB 10000000 - %MB 10019999



Aufteilung und Einstellung des Ωmega Drive-Line II Datenbereichs.

Adressvergabe im Programm:

Eine absolute Ωmega Drive-Line II-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.

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

4.4.5 Die Ωmega Drive-Line II Event-Tasks

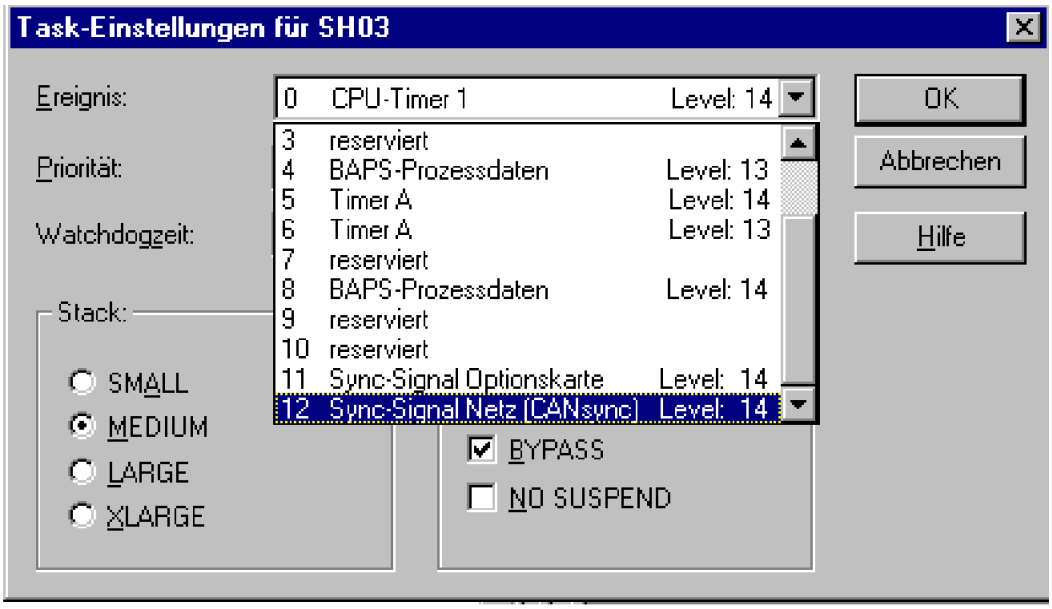
Die Ωmega Drive-Line II Event-Tasks dienen dem ereignisgesteuerten Programmaufruf (Interrupt). Sie bestimmen durch ihre Art und Codelaufzeit das Echtzeitverhalten.

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

- In einem "stand alone" Antrieb über eine BAPS Event-Task.
- In einem vernetzten Antrieb über CANsync Event-Task.

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

Eigenschaft	**Ωmega Drive-Line II Event**	**Ωmega Drive-Line II Interrupt-Level**
Ereignis 0	CPU-Timer 1	Level 14
Ereignis 1	reserviert	
Ereignis 2	CPU-Timer 2	Level 13
Ereignis 3	reserviert	
Ereignis 4	BAPS-Prozeßdaten	Level 13
Ereignis 5	Timer A	Level 14
Ereignis 6	Timer A	Level 13
Ereignis 7	reserviert	
Ereignis 8	BAPS-Prozeßdaten	Level 14
Ereignis 9, 10	reserviert	
Ereignis 11	Sync-Signal Optionskarte	Level 14
Ereignis 12	Sync-Signal Netzwerk (CANsync)	Level 14



Die Ereignisse der Ωmega Drive-Line II Event-Tasks.

HINWEIS

Alle Ωmega Drive-Line II Event-Tasks 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 Ωmega Drive-Line II Anwenderbibliotheken

Die PROPROG wt II 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 Ωmega Drive-Line II Bibliotheken ist mit **_DLII_** in der Bibliotheksbezeichnung gekennzeichnet.

Alle Ωmega Drive-Line II Anwenderbibliotheken beginnen mit der Version 2.0 Build 0. 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.

Die Anwenderbibliotheken gliedern sich in:

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

Übersicht der Ωmega Drive-Line II Bibliotheken unter PROPROG wt II (Änderungen vorbehalten):

Datentypen:

BM_TYPES_20bd00 Zusammengesetzte Datentypen und Felder.

Ωmega Drive-Line II System-FBs:

SYSTEM1_DLII_20bd00 BAPS und serielle Kommunikation, Triggersignalbeschaltung, Codelaufzeitmessung.

Ωmega Drive-Line II Firmware:

SYSTEM2_DLII_20bd00 Die gesamte Ωmega Drive-Line II Firmware.

Antriebsnahe Steuerungstechnik:

UNIVERSAL_20bd00 Hardware unabhängige FBs wie Regler oder Sollwertgeneratoren.

IEI-02 Optionskarte (optional):

IEI_DLII_20bd00 IEI-02 Initialisierung.

Asynchroner CAN-Bus mit CAN-M-01 Optionskarte (optional):

CAN_DLII_20bd00 FBs für eine CAN Bus Ankopplung mit CAN-M-01 Optionskarte.

Synchrones Bussystem CANsync:

CANsync_DLII_20bd00	FBs für eine CANsync Bus Ankopplung.

Technologiebausteine (optional):

WINDER_DLII_20bd00	Wickler.
CAM_DLII_20bd00	Kurvenscheibe.
REGISTER_DLII_20bd00	Registerregelung.

Der Verzeichnispfad für die Bibliotheken ist unter PROPROG wt II, Optionen, Verzeichnisse anzugeben. Die Ωmega Drive-Line II Bibliotheken werden im Projektbaum von PROPROG wt II unter Bibliotheken eingefügt.

Zu jedem FB ist eine HTML-Hilfe aufrufbar, die eine Beschreibung der Ein- und Ausgänge bereitstellt (siehe PROPROG wt II Handbuch).

4.5.1 Ωmega Drive-Line II Firmware

Die Ωmega Drive-Line II Firmware besteht aus Funktionsbausteinen (FBs), die über Parameterübergabe mit Funktionen auf der Ωmega Drive-Line II-CPU kommunizieren. Diese FBs sind nur ressourcenabhängig einsetzbar, d. h. vom Zielsystem abhängig.

Die Ωmega Drive-Line II Firmware ist mit der Bibliothek

SYSTEM2_DLII_20bd00 (oder höher)

in ein Projekt einzufügen. Die Bibliothek beinhaltet folgenden Funktionsumfang:

- Bypass Event-Task starten und freiprogrammierbare LEDs am Ωmega Drive-Line II.
- P, PI-Regler, 48-Bit Division über elektronisches Getriebe, Integrieren, Differenzieren.
- Funktionsbausteine zur Anschaltung an USS®- und 3964R®-Protokoll.

HINWEIS

Die Ωmega Drive-Line II Firmware wird von einigen Ωmega Drive-Line II Anwenderbibliotheken genutzt. Daher kann es erforderlich sein, mit einer Ωmega Drive-Line II Anwenderbibliothek, die geforderte Firmware-Bibliothek SYSTEM2_DLII_20bd00 oder höher einzufügen (siehe Beschreibung der Anwenderbibliothek).
Die Firmware SYSTEM2_DLII_20bd00 (oder höher) sollte neben der Anwenderbibliothek SYSTEM1_DLII_20bd00 (oder höher) immer in ein Ωmega Drive-Line II Projekt eingefügt werden, um Grundfunktionalitäten nutzen zu können.

4.5.2 Die Ωmega Drive-Line II Board Funktionen

Die Firmware-Bibliothek SYSTEM2_DLII_20bd00 (oder höher) enthält Funktionsbausteine (FBs) zur Kontrolle von Event Signalen für Interrupts und Board-LEDs. Um Ωmega Drive-Line II Event-Tasks (Bypass) zu initialisieren und zu starten wird der Funktionsbaustein INTR_SET benutzt. Um freiprogrammierbare LEDs zu nutzen wird der Funktionsbaustein LED benutzt.

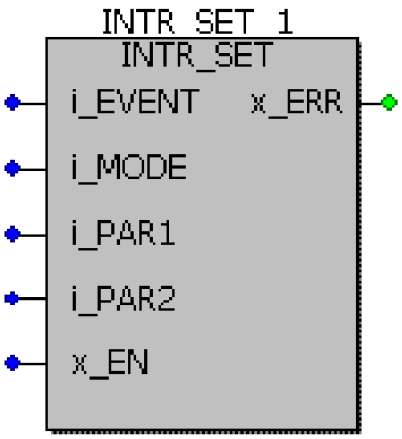
Für die Optimierung von Code-Laufzeiten innerhalb von Ωmega Drive-Line II Ressourcen werden zwei FBs zur Code-Laufzeitmessung bereitgestellt. Die FBs werden über die Anwenderbibliothek **SYSTEM1_DLII_20bd00** und höher eingefügt.

Der zu messende Code-Block innerhalb einer Task ist mit der Plazierung 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).

Der Funktionsbaustein INTR_SET

Der Funktionsbaustein INTR_SET startet in einer Anlauf-Task eine Bypass Event-Task.

Die PROPROG wt II Event-Task mit dem Programm muß auf das Ereignis und auf das Attribut "Bypass" gestellt werden.



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:

i_EVENT	HW-Ereigniss(e)	Interrupt-Level
0	CPU-Timer 1	Level 14
2	CPU-Timer 2	Level 13
4	BAPS-Prozeßdaten	Level 13
5	Board-Timer A	Level 14
6	Board-Timer A	Level 13
8	BAPS-Prozeßdaten	Level 14
11	Sync-Signal Optionskarte	Level 14
12	Sync-Signal Netzwerk (CANsync)	Level 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.
Die Ereignisse Sync-Signal Optionskarte (11) und Sync-Signal Netzwerk (CANsync, 12) sind nicht gleichzeitig verwendbar. Die BAPS-Prozeßdaten niedrige Priorität (4) und BAPS-Prozeßdaten hohe Priorität (8) sind nicht gleichzeitig verwendbar.

Beispiel 1:

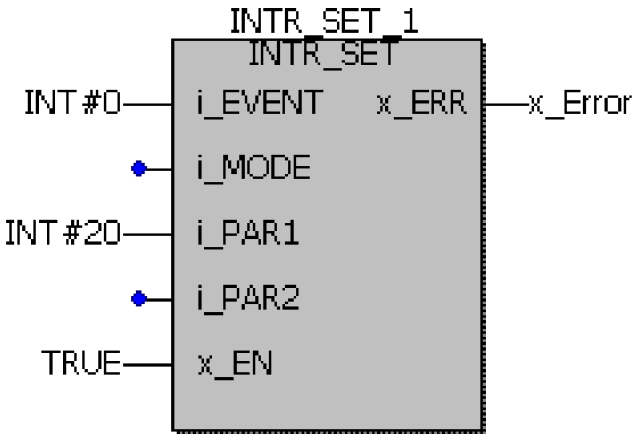
Start eines CPU-Timer-Interrupts mit der Zeit 1 ms.

Umsetzung:

Zunächst erfolgt die Einrichtung der Event-Task, innerhalb der Ωmega Drive-Line II Ressource, mit der Ereignisnummer 0, bzw. CPU-Timer 1.

Innerhalb einer Anlauf-Task wird der FB INTR_SET implementiert. Die Ein-/Ausgangsbelegung sieht wie folgt aus:

Interruptzeit = i_PAR1 • 50 µs



Funktionsbaustein INTR_SET: Start eines CPU Timer 1-Interrupt mit der Interrupt-Zeit 1 ms.

Beispiel 2:

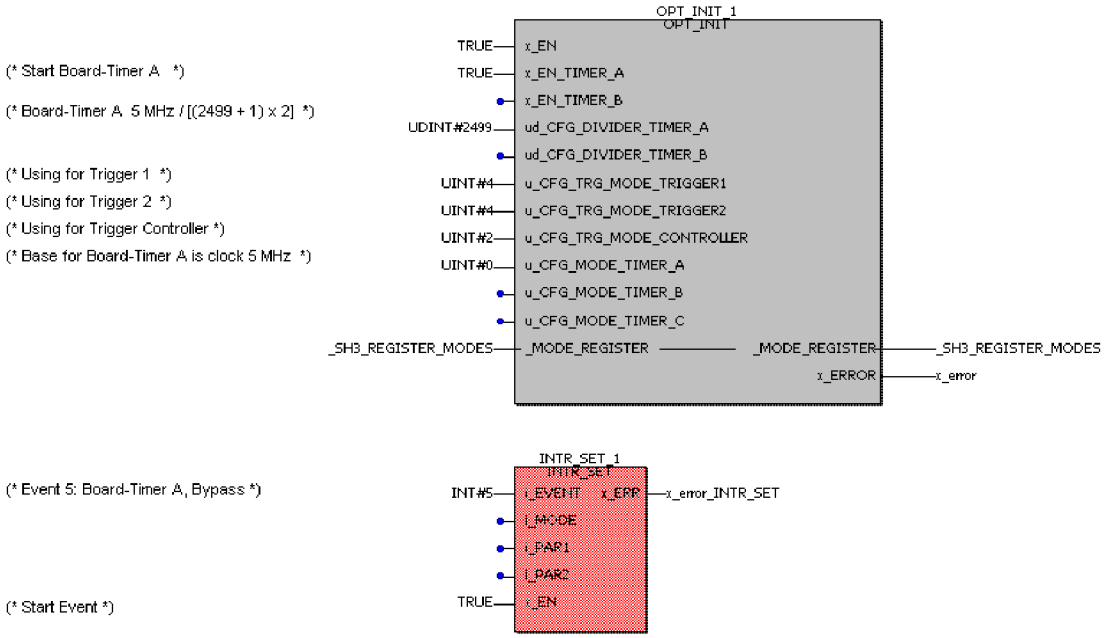
Start eines Board-Timer A-Interrupts mit der Zeit 1 ms.

Im Gegensatz zum CPU-Timer 1 (oder 2) kann der Board-Timer A auch als Triggersignal für Baugruppen, die Triggersignale benötigen, genutzt werden, denn nur mit dem Board-Timer A kann sowohl ein Interrupt als auch eine Triggerung erfolgen.

Zunächst erfolgt die Einrichtung der Event-Task, innerhalb der Ωmega Drive-Line II Ressource, mit der Ereignisnummer 5 (oder 6), bzw. Event Board-Timer A.

Innerhalb einer Anlauf-Task wird zunächst der FB OPT_INIT und dann der FB INTR_SET implementiert.

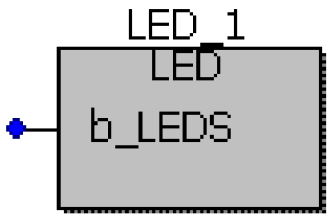
Interruptzeit 1 ms	=	Frequenzteiler für Timer A	an FB OPT_INIT einrichten
1 ms	=	((2499 + 1) • 2) / 5 MHz	an FB OPT_INIT freigeben
Interrupt	=	Event Board-Timer A	an FB INTR_SET angeben



Start einer Board-Timer A Event-Task mit der Zeit 1 ms und gleichzeitige Nutzung als Triggersignal für Baugruppen die Triggersignale benötigen innerhalb einer Anlauf-Task.

Der Funktionsbaustein LED

Über den Funktionsbaustein LED aus der Firmware-Bibliothek **SYSTEM2_DLII_20bd00** (oder höher), sind die Ωmega Drive-Line II LEDs am Board programmierbar.



Parameter	Eingang	Wertebereich
b_LEDS	LED Setz-Maske	8 bit

Beschreibung:

Die linken LEDs am Ωmega Drive-Line II-Board leuchten rot, die rechten grün. Die Bits 0-3 des 8-Bit-Musters werden auf die vier LEDs wie folgt ausgegeben:

rote LEDs	grüne LEDs
⊗	⊗
Bit 0	Bit 1
⊗	⊗
Bit 2	Bit 3

4.5.3 Die Ωmega Drive-Line II Datentypen

Ωmega Drive-Line II Anwenderbibliotheken benötigen in der Ein- und Ausgangsbelegung ihrer Funktionsbausteine (FBs) sehr oft zusammengesetzte Datentypen. Diese Datentypen sind in der Anwenderbibliothek

- **BM_TYPES_20bd00** (oder höher)

hinterlegt.

Die Anwenderbibliothek BM_TYPES_20bd00 (oder höher) stellt die Datentypen zur Verfügung, um die Ωmega Drive-Line II Anwenderfunktionsbausteine und Firmwarebausteine einzusetzen. Ωmega Drive-Line II Anwenderfunktionsbausteine und Firmwarebausteine nutzen diese Datentypen, um Registerstrukturen von Optionskarten abzubilden oder Initialisierungswerte Task-übergreifend weiterzuleiten.

Um Standardbibliotheken und Technologiebausteine nutzen zu können, muß der Anwender die BM_TYPES_20bd00 (oder höher) in sein Projekt einfügen. Das Arbeitsblatt der BM_TYPES_20bd00 ist nicht aufrufbar. Die Datentypen stehen im Variablendialog unter Eigenschaften der Zuweisung zur Verfügung. Datentypen der BM_TYPES_20bd00 sind mit „**_BM**“ gekennzeichnet.

Die Standardbibliotheken und Technologiebausteine verweisen in ihrer Beschreibung auf die Datentypen der BM_TYPES_20bd00 (oder höher). Bei der Implementierung von FBs aus diesen Bibliotheken sind Variablen dieser Datentypen zu deklarieren und eventuell auf eine absolute Adresse, z. B. Optionsschnittstelle zu legen.

Beispiel:

Die Optionskarte IEI-02 benötigt zur Initialisierung Einstellungen in verschiedenen Registern. Der komplette Registeraufbau ist mit seiner Datenbreite und seinen Elementen in den BM_TYPES_20bd00 hinterlegt. Mit dem IEI-02 auf Optionssteckplatz 1 ergibt sich folgende Variablendeklaration.

```
_IEI_write_register  AT  %MD3.1000000        : IEI_WRITE_BMSTRUCT;
```

Die Elemente der Variable `_IEI_write_register` bilden jetzt die Register der Optionskarte ab. Die Register können über die Elemente der Variable symbolisch programmiert werden (siehe Technische Beschreibung Optionskarte IEI-02 für Ωmega Drive-Line II).

HINWEIS

Im Variablendialog werden die Datentypen zur Zuweisung über Auswahlfenster bereitgestellt. Die Datentypen der Bibliothek BM_TYPES_20bd00 (oder höher) sind dabei mit dem Kürzel „_BM“ (_BMARRAY, _BMSTRUCT, usw.) gekennzeichnet. Das Arbeitsblatt der Bibliothek ist schreibgeschützt und nicht einsehbar.

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:

- **SYSTEM1_DLII_20bd00** (oder höher)

  BAPS-Kommunikation

  Triggersignalbeschaltung FB OPT_INIT

  Code-Laufzeitmessung

  3964R®-Protokoll-Anschaltung

  USS®-Protokoll-Anschaltung
- **SYSTEM2_DLII_20bd00** (oder höher)

  Ωmega Drive-Line II Firmware
- **UNIVERSAL_20bd00** 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 Ωmega Drive-Line II Technologiebausteine

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

- Technologiebaustein Kurvenscheibe: Anwenderbibliothek CAM_DLII_20bd00 (oder höher)
- Technologiebaustein Registerregelung: Anwenderbibliothek REGISTER_DLII_20bd00 (oder höher)
- Technologiebaustein Wickler: Anwenderbibliothek WINDER_DLII_20bd00 (oder höher)

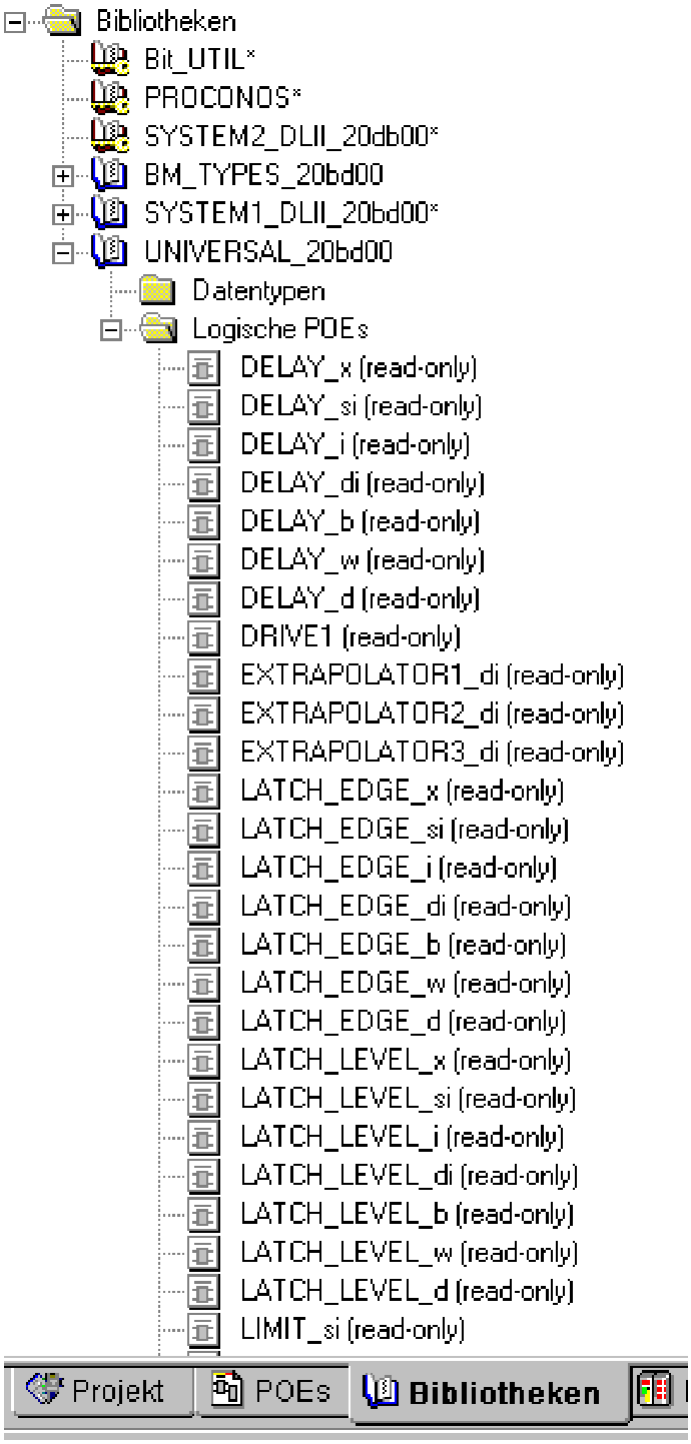
Die Technologiebausteine bieten Antriebsfunktionalitäten, die durch die Beschaltung und mehrfache Instanziierung eine Vielzahl von Applikationslösungen bereitstellen.

HINWEIS

Alle Anwenderbibliotheken der Technologiebausteine benötigen zur Einbindung die Datentypen ab BM_TYPES_20bd00.

4.5.6 Einfügen einer Anwenderbibliothek in ein Projekt

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



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

HINWEIS

Die Auslieferung einer PROPROG wt II Bibliothek erfolgt gepackt (ZWT-File). Das ZWT-File ist unter PROPROG wt II zu entpacken. Beim Entpacken wird die schreibgeschützte Bibliothek automatisch im angegebenen PROPROG wt II Bibliothekspfad (Menü Extras>Optionen) abgelegt und es erscheint ein "Untitled"-Projekt, welches ohne Speicherung zu schließen ist.

Die Firmware-Bibliotheken werden in das Firmware-Bibliotheks-Verzeichnis von PROPROG wt II entpackt.

4.6 Ωmega Drive-Line II Optionsschnittstellen, Interruptquellen und Triggersignale

4.6.1 Die Interruptquellen und Triggersignale

Im Ωmega Drive-Line II können Baugruppen, Timer und Optionskarten **Interruptquellen** sein und Event gesteuerte Tasks auslösen. Die benötigten Programm-Interrupts sind applikationsabhängig. Damit die Ein- und Ausgangswerte aller Ωmega Drive-Line II-Komponenten echtzeitfähig verarbeitet werden können, müssen alle in der gesetzten Event-Task verwendeten Baugruppenwerte zu diesem synchronisiert werden. Dies geschieht mit einem **Triggersignal**, das die Baugruppe liefert oder benötigt.

Baugruppen, die Triggersignale benötigen sind:

- **Trigger 1**: IEI-02 Inkrementalgeberkarte, zum Latchen der Lageistwerte von Inkrementalgebern.
- **Trigger 2**: MFM-01 Digital/Analog Ein- und Ausgabekarte, zum Start der analog/digital Wandlung.
- **Trigger Controller**: V-Regler-zur Synchronisierung der Regelungszeitscheiben.

Taktsignale, die Baugruppen bereitstellen:

- **Basistakt 5 MHz**: Takt für Frequenzteiler bzw. Board-Timer

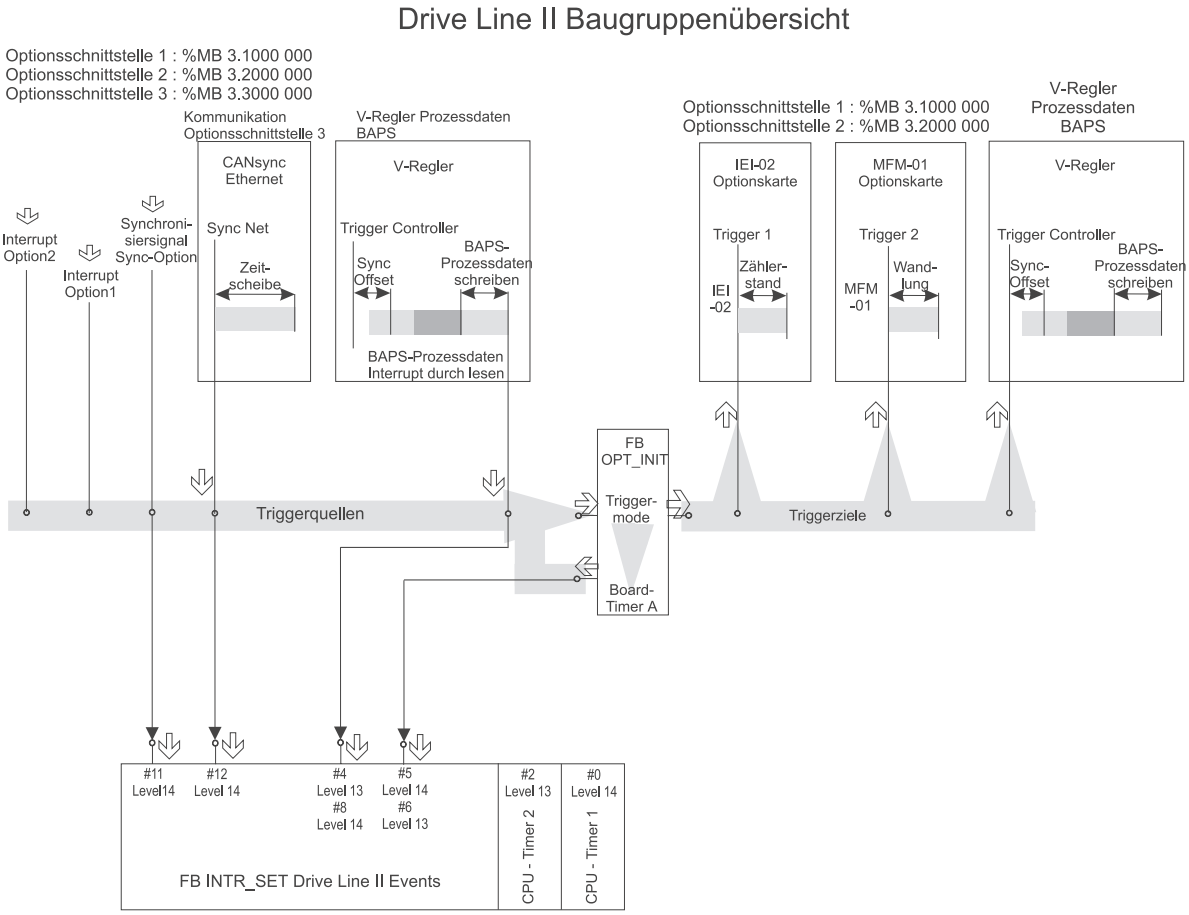
Interruptsignale, die Baugruppen bereitstellen:

- **Interrupt Option 1**: Interrupt von einer Optionskarte
- **Interrupt Option 2**: Interrupt von einer Optionskarte
- **Interrupt Net**: reserviert
- **Interrupt Timer A**: Timer A Interrupt
- **Interrupt Timer B**: reserviert

Synchronisations-Signale, die Baugruppen bereitstellen:

- **Sync Net**: SYNC-Signal von CANsync.
- **Sync Option**: Sync-Signal von der Optionskarte.

Das Ωmega Drive-Line II bietet die Möglichkeit Triggersignale, Sync-Signale und Interruptsignale auf Baugruppen, die Triggersignale benötigen, zu schalten (Funktionsbaustein OPT_INIT).



Baugruppenübersicht und Triggersignale

4.6.2 Die Triggersignalbeschaltung und Timerkonfiguration über den Funktionsbaustein OPT_INIT

Die Baugruppen, die Triggersignale benötigen, sind über den Funktionsbaustein (FB) OPT_INIT zu beschalten. Der FB wird über die Bibliothek SYSTEM1_DLII_20bd00 (oder höher) in das Projekt eingebunden. Baugruppen, die Triggersignale benötigen, sind:

- **Trigger 1**: IEI-02: Zum Latchen der Lageistwerte auf der Optionskarte.
- **Trigger 2**: MFM-01: Zum Start der analog/digital Wandlung auf der Optionskarte.
- **Trigger Controller**: V-Regler: Zur Synchronisierung der Regelungszeitscheiben auf dem V-Regler.

Wird ein Triggersignal nicht benötigt, kann die Beschaltung offen bleiben. Als Quelle für die zu beschaltenen Triggersignale können von anderen Baugruppen bereitgestellte Triggersignale, Interruptsignale oder Sync-Signale dienen.

Benötigter Trigger 1 für die Optionskarte IEI-02 auf Optionssteckplatz 1 oder 2:

Mode für Trigger 1	Synchronisierung/Triggerung mit	Quelle
0	reserviert	reserviert
1	reserviert	reserviert
2	Sync-Signal von der Optionskarte	Sync Option
3	CANsync Synchronisier-Signal	Sync Net
4	Boardtimer A über OPT_INIT	Interrupt Timer A
5	reserviert	Interrupt Timer B
6	Interrupt von einer Optionskarte	Interrupt Option 1

Benötigter Trigger 2 für die Optionskarte MFM-01 auf Optionssteckplatz 1 oder 2:

Mode für Trigger 2	Synchronisierung/Triggerung mit	Quelle
0	reserviert	reserviert
1	reserviert	reserviert
2	Sync-Signal von der Optionskarte	Sync Option
3	CANsync Synchronisier-Signal	Sync Net
4	Boardtimer A über OPT_INIT	Interrupt Timer A
5	reserviert	Interrupt Timer B
6	Interrupt von einer Optionskarte	Interrupt Option 1

Benötigter Trigger Controller (V-Regler-BAPS):

Mode für Trigger Controller	Synchronisierung mit	Quelle
0	Sync-Signal von der Optionskarte	Sync Option
1	CANsync Synchronisier-Signal	Sync Net
2	Boardtimer A über OPT_INIT	Interrupt Timer A
3	reserviert	Interrupt Timer B
4	Interrupt von einer Optionskarte	Interrupt Option 1
5	Interrupt von einer Optionskarte	Interrupt Option 2

Der Ωmega Drive-Line II Funktionsbaustein OPT_INIT wird für zwei Aufgaben eingesetzt:

1. Triggersignale schalten für Baugruppen, die Triggersignale benötigen, um Daten aus den Baugruppen beim Interrupt-Aufruf bereitzustellen.
2. Zeitintervall des Board-Timer A festlegen und starten:
   Zeitintervall = [(Divisor +1) • 2] / Taktsignalfrequenz

   Beispiel: 1 ms = ((2499 + 1) • 2) / 5 MHz (Basistakt 5 MHz als Taktsignal)

Die Timer A und B können auf eine feste Frequenz eingestellt und separat gestartet werden. Der Timer A kann als Event für Bypass Event-Tasks genutzt werden.

HINWEIS

Wird in PROPROG wt II eine Task vom Typ Event eingefügt und das Attribut Bypass gesetzt, ist es möglich über den Funktionsbaustein INTR_SET (Firmware SYSTEM2_DLII_20bd00 oder höher) einen Interrupt aus einer Anlauf-Task, zu starten. Voraussetzung ist, daß die Event-Task in der IEC 61131-3 Ressource auf das Ereignis eingestellt ist.

4.6.3 Einsatz des Funktionsbausteins OPT_INIT

Der Funktionsbaustein (FB) OPT_INIT wird in den Initialisierungstasks (Task Kaltstart, Warmstart) oder in einer zyklischen Task plaziert, wobei der FB nur einmal durchlaufen werden darf.

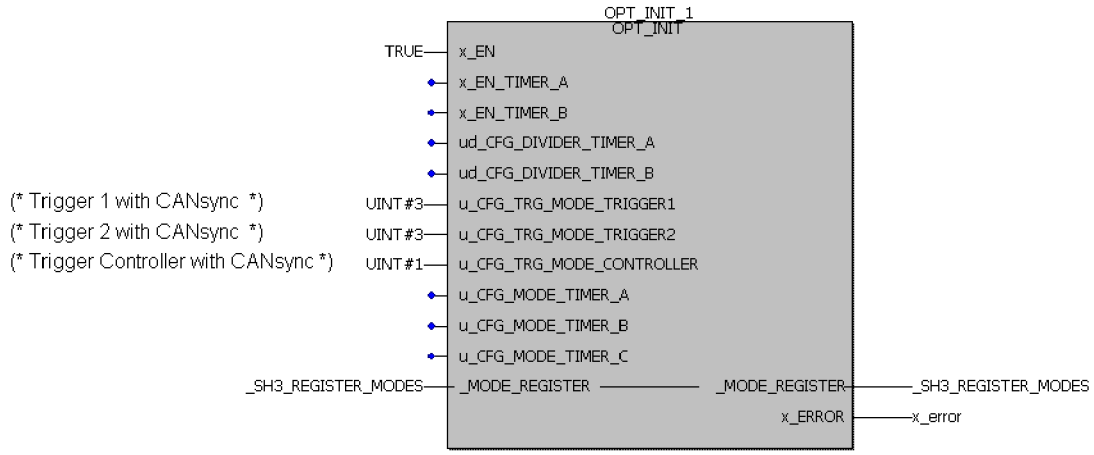
Der Aufruf kann für die Belegung eines benötigten Triggersignals (z. B. IEI-02, MFM-01, V-Regler) **nach** der Initialisierung der Baugruppe erfolgen.

Bei der Einrichtung eines Board-Timer (A, B) über einen Frequenzteiler und Synchronisierung auf ein Taktsignal (z. B. Basistakt 5 MHz) ist die Teilung wie folgt einzurichten:

Zeitintervall = [(Divisor +1) • 2] / Taktsignalfrequenz

Beispiel: FB OPT_INIT Einsatz im Initialisierungsteil eines CANsync-Slave-Projekts zur Triggerung der eingesetzten Optionskarten und des V-Reglers.

- Es ist der FB OPT_INIT aus der Bibliothek SYSTEM1_DLII_20bd00 (oder höher) in das Programm einzubinden. Der Mode für Baugruppen, die Triggersignale benötigen, ist auf Sync Net (CANsync) zu schalten. Der Mode kann der Online-Hilfe des FB entnommen werden.
- Der FB OPT_INIT wird einmal durchlaufen und speichert die an den Eingängen angelegte Konfiguration auf die Elemente in der Konfigurations-Struktur (Ein-/Ausgangsvariable _MODE_REGISTER).



Ωmega Drive-Line II Einsatz mit CANsync Slave Event-Task und Triggerung von IEI-02, MFM-01 und V-Regler mit dem SYNC-Signal vom CANsync-Bus.

4.6.4 Die Basisadressen der Optionsschnittstellen

Den Optionsschnittstellen 1-3 sind Basisadressen zugeordnet. Die Register der Optionskarte sind, entsprechend ihrer Datentypbreite, Offsetadressen zu dieser Basisadresse.

Ωmega Drive-Line II	Zugriff in PROPROG wt II: Anfang bis Ende	Zugeordnete Hardware-Adresse
Optionsschnittstelle 1:	**%MB 3.1000000** - %MB3.1262143 auf Optionssteckplatz 1.	16#B4000000 - 16#B403FFFF
Optionsschnittstelle 2:	**%MB 3.2000000** - %MB3.2262143 auf Optionssteckplatz 2.	16#B4040000 - 16#B407FFFF
Optionsschnittstelle 3: (Kommunikationskarten)	**%MB 3.3000000** - %MB3.3262143 auf Optionssteckplatz 2.	16#B4080000 - 16#B40AFFFF

HINWEIS

Um in einem PROPROG wt II Projekt auf Register der Optionskarten zugreifen zu können, sind Datentypen definiert, die die Registerstruktur abbilden. Mit diesen Datentypen werden Variablen deklariert, die auf die Adresse der benutzten Optionsschnittstelle gelegt werden. So ist es möglich über die Elemente der deklarierten Variablen auf die Registerstruktur der Optionskarte zuzugreifen. Weitere Erläuterungen zur Registerstruktur und Funktion finden sie hierzu in den Technischen Beschreibungen der jeweiligen Optionskarte.

4.6.5 Das Steuerungsspezifische Mapping der Hardware-Bereiche

Für die BAPS-, CANsync- und Ethernet-Kommunikation ist im Ωmega Drive-Line II für PROPROG wt II folgendes Mapping der Hardware-Adressen (Basisadressen) umgesetzt:

Bezeichnung	**Zugriff in PROPROG wt II Anfang - Ende**	**Zugeordnete Hardware-Adresse**
DPRAM BAPS	**%MB3.80000** - %MB3.88191	16#B4140000 - 16#B4141FFF
CANsync-Knoten 1	**%MB3.100000** - %MB3.132767	16#B4142000 - 16#B4149FFF
Ethernet-Konfiguration	**%MB3.180016** - %MB3.180031	16#B0000010 - 16#B000001F
CANsync-Knoten 2	**%MB3.200000** - %MB3.232767	16#B414A000 - 16#B4151FFF
Ethernet	**%MB3.300000** - %MB3.357343	16#B4152000 - 16#B415FFFF

Den Adressbereichen sind Strukturen zugeordnet. Die genaue Verwendung der Strukturen ist bei Bedarf der entsprechenden Dokumentation zu den Bibliotheken zu entnehmen.

BAPS:

Die V-Regler Schnittstelle BAPS mit dem Kommunikations-RAM der BAPS wird umgesetzt mit:

- **BAPS_CTRL_BMSTRUCT** (in Bibliothek BM_TYPES_20bd00 oder höher definiert.)

CANsync:

Mit CANsync Knoten 1 wird die Kommunikation für einen CANsync-Slave umgesetzt. Mit CANsync Knoten 2 wird die Kommunikation für einen CANsync-Master umgesetzt. Folgende Strukturen sind dafür in der Bibliothek BM_TYPES_20bd00 (oder höher) definiert:

- **CANsync_INIT_BMSTRUCT**
- **CANsync_MA_CTRL_BMSTRUCT**
- **CANsync_SL_CTRL_BMSTRUCT**

CAN:

Mit der Optionsschnittstelle 3 ist eine CAN-Kommunikation umsetzbar: Folgende Strukturen sind dafür in der Bibliothek BM_TYPES_20bd00 (oder höher) definiert:

- **CAN_INIT_BMSTRUCT**
- **CAN_CTRL_BMSTRUCT**

Ethernet:

Zur Ethernet-**Konfiguration** durch den Anwender, z. B. Ethernet-IP-Adressen und Ethernet-IP-Maske, kann folgende Struktur in der Bibliothek BM_TYPES_20bd00 (oder höher) benutzt werden:

- **ETHERNET_CONFIG_BMSTRUCT** (AT %MB3.180016)

(Der Ethernet-Konfigurations-Bereich liegt im ausfallsicheren NOVRAM-Bereich)

Für die Ethernet-**Diagnose** kann folgende Struktur aus der Bibliothek BM_TYPES_20bd00 (oder höher) benutzt werden:

- **ETHERNET_DIAGNOSE_BMSTRUCT** (AT %MB3.300000)

4.6.6 Konfigurationsbeispiele

Je nach verwendeten Optionskarten und Synchronisierungsanforderungen werden verschiedene Konfigurationen für die BAPS-Kommunikation, die Triggersignale und Interruptquellen benötigt.

Für die Mehrzahl der Anwendungsfälle sind die folgenden Beispiele verwendbar:

Beispiel A:

Umsetzung einer BAPS-Prozeßdatenkommunikation innerhalb einer BAPS-Event-Task. Die Synchronisierung der BAPS-Prozeßdatenkommunikation erfolgt durch das Ereignis BAPS-Prozeßdaten vom V-Regler aus.

HINWEIS

In diesem Fall kann keine Optionskarte über Triggersignale synchronisiert werden.

Beispiel B:

Umsetzung einer hochgenauen BAPS-Prozeßdatenkommunikation innerhalb einer Timer-Event-Task (Board-Timer A) und Triggerung von MFM-01, IEI-02 und V-Regler über diesen Timer.

Beispiel C:

Umsetzung einer BAPS-Prozeßdatenkommunikation innerhalb einer CANsync-Event-Task und Triggerung des V-Reglers, MFM-01 und IEI-02 über das Synchronisiersignal von CANsync.

Für die Übermittlung von Daten über 3964R®-Protokoll oder USS®-Protokoll benötigt man eine Timer-Event-Task.

Beispiel D:

Umsetzung einer Timer-Event-Task für eine zyklische serielle Kommunikation. Die Funktionsbausteine zur zyklischen Kommunikation werden in einer POE, die dieser Task zugeordnet wird, plaziert. Falls Optionskartensteuersignale benötigt werden, können diese eingerichtet werden.

HINWEIS

Für die Optimierung der Durchlaufzeiten von Prozeßdaten, gemäß den Konfigurationsbeispielen, wird eine Code-Laufzeitmessung empfohlen.
Diese kann mit den FBs TIME_MEASURE_START und TIME_MEASURE_END aus der Bibliothek SYSTEM1_DLII_20bd00 (oder höher) durchgeführt werden.

4.6.7 Umsetzung einer BAPS in einer BAPS-Event Task

Beispiel A:

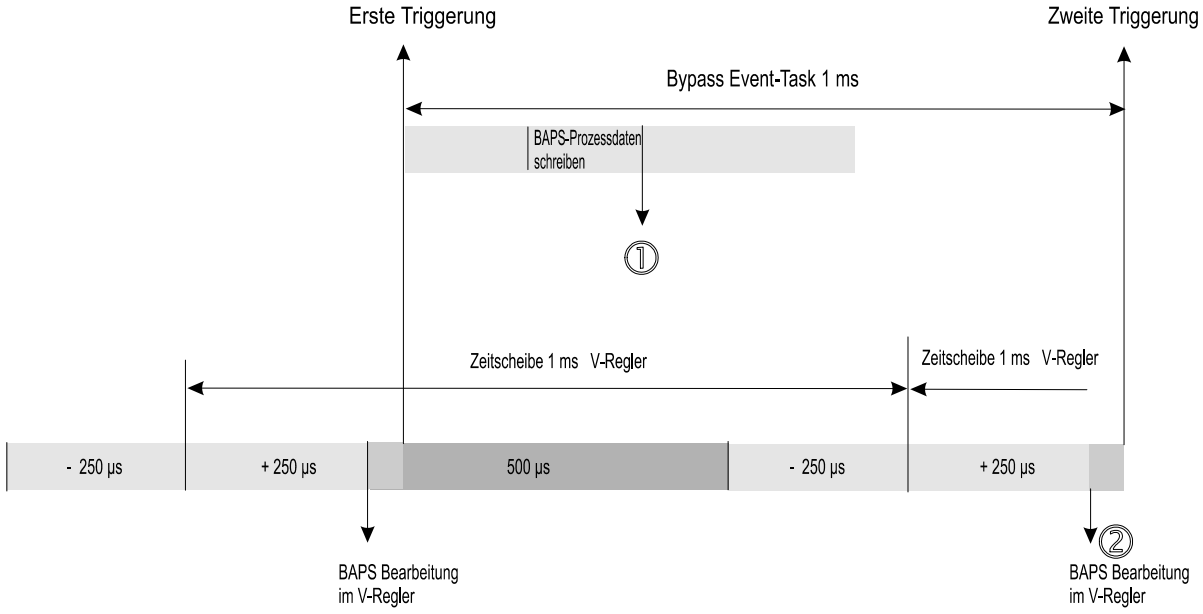
Das Ωmega Drive-Line II generiert Sollwerte in einer V-Regler Event-Task mit BAPS-Prozeßdatenkommunikation zum V-Regler. Das Event für die Task kommt vom V-Regler. Eine direkte Triggerung von Optionskarten auf das BAPS-Event ist nicht möglich.

Die BAPS Prozeßdatenkommunikation erfolgt direkt in einer Event-Task mit dem BAPS-Prozeßdaten-Event 4 oder 8 (V-Regler). Die Event-Task BAPS-Prozeßdatenkommunikation wird über den FB BAPS_INIT initialisiert und gestartet (siehe “BAPS_INIT” auf Seite 86). Der FB BAPS_INIT wird dazu in einer POE, die einer zyklischen Task zugeordnet ist, plaziert. Es muß dabei sichergestellt werden, daß die Freigabe des FBs erst nach fünf Programmzyklen erfolgt und der FB nur einmal durchlaufen wird.

- Die BAPS wird in einer zyklischen Task über den FB BAPS_INIT auf die gewünschte Zeitscheibe und Parameter initialisiert. Der Firmwarebaustein INTR_SET ist Bestandteil des FB BAPS_INIT.
- Für die Prozeßdatenkommunikation über den FB BAPS_PD_COMM wird in der Ressource eine Bypass-Event-Task mit dem Ereignis 4 oder 8 eingefügt.

Die Zeitscheiben-Synchronisierung entspricht der folgenden Zeichnung.

Zeitscheibenkonfiguration Bypass Event Task auf Event BAPS (Beispiel A)



Bypass-Event Task im Ωmega Drive-Line II auf Event BAPS aus dem V-Regler

HINWEIS

Die Wandlung der MFM-01 Werte kann in der Betriebsart "Wandlung durch Schreiben" innerhalb der BAPS-Event-Task synchronisiert werden.

In der V-Regler Betriebsart "Gleichlauf mit synchroner Sollwertvorgabe" sind bei der BAPS-Prozeßdatenkommunikation über den Parameter P258 (32-Bit Winkel) je nach gewählter BAPS-Zeitscheibe die untersten Bits des Übergabewertes auszumaskieren (FALSE setzen).
Vorgehensweise siehe Tabelle auf Seite 74

4.6.8 Umsetzung einer hochgenauen BAPS in einer Timer-Event Task

Beispiel B:

Das Ωmega Drive-Line II generiert die Sollwerte, die in einer Event Task mit BAPS-Prozeßdatenkommunikation zum V-Regler übermittelt werden und benötigt synchronisiert getriggerte Optionskarten, z. B. Applikation Kurvenscheibe mit Registerregelung.

Interrupt-Signal und Triggersignal für Baugruppen die Triggersignale benötigen müssen identisch sein. Diese Umsetzung ist nur mit dem Board-Timer A möglich. Die BAPS Prozeßdatenkommunikation erfolgt in der Event-Task Board-Timer A, d. h. der Event Board-Timer A wird auf die Zeitscheibe der benötigten BAPS-Prozeßdatenkommunikation eingestellt.

- Triggersignalbeschaltung und starten der Event-Task auf Board Timer A erfolgen in einer Anlauf-Task. Der **FB OPT_INIT** konfiguriert die Board-Timer A Zeit auf die gewünschte BAPS Zeitscheibe, z. B. 1 ms.

Benötigtes Event Board-Timer A starten mit Frequenzteiler auf Basistakt 5 MHz:

u_DIVIDER_TIMER_A	x_EN_TIMER_A	ud_CFG_MODE_TIMER_A
2499	TRUE	0
1 ms = [(2499 +1) • 2] / 5 MHz	Timer A gestartet	Basistakt 5 MHz

Benötigter Trigger 1 für die Optionskarte IEI-02 auf Optionssteckplatz 1 oder 2:

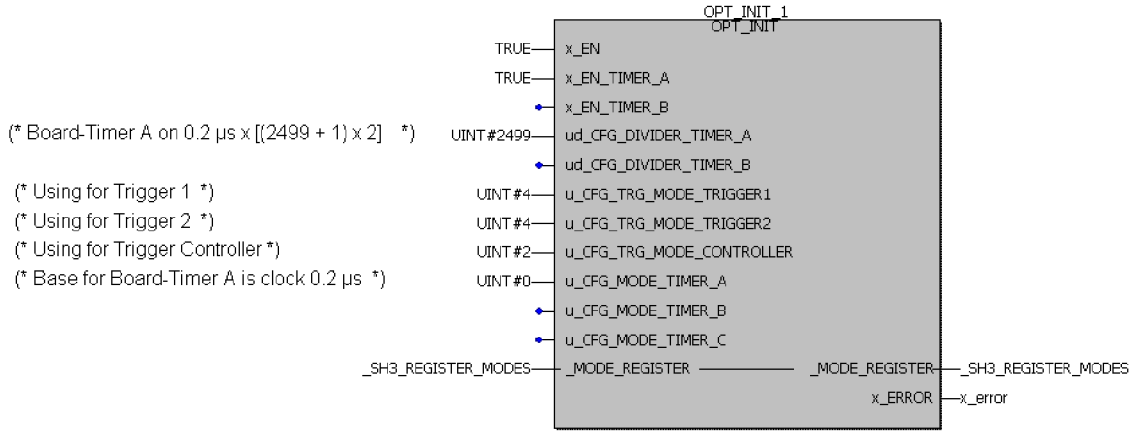
Mode für Trigger 1	Synchronisierung/Triggerung mit	Quelle
4	Board-Timer A über OPT_INIT	Interrupt Timer A

Benötigter Trigger 2 für die Optionskarte MFM-01 auf Optionssteckplatz 1 oder 2:

Mode für Trigger 2	Synchronisierung/Triggerung mit	Quelle
4	Board-Timer A über OPT_INIT	Interrupt Timer A

Benötigter Trigger Controller (V-Regler-BAPS):

Mode für Trigger Controller	Synchronisierung mit	Quelle
2	Board-Timer A über OPT_INIT	Interrupt Timer A

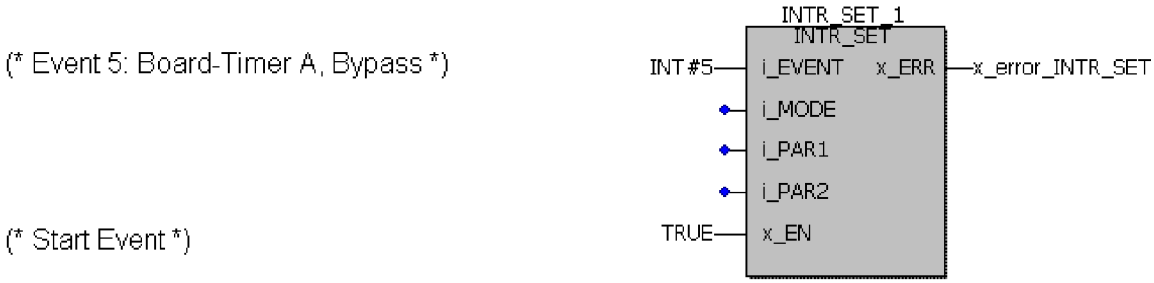


FB OPT_INIT zum Start des Board-Timer A und Triggersignalbeschaltung in einer Anlauf-Task.

- Dem FB OPT_INIT folgend wird der **FB INTR_SET** eingefügt:

I_EVENT	HW-Ereigniss(e)	Interrupt-Level
5	Board-Timer A	Level 14

i_MODE, i_PAR1, i_PAR2 bleiben unbeschaltet.



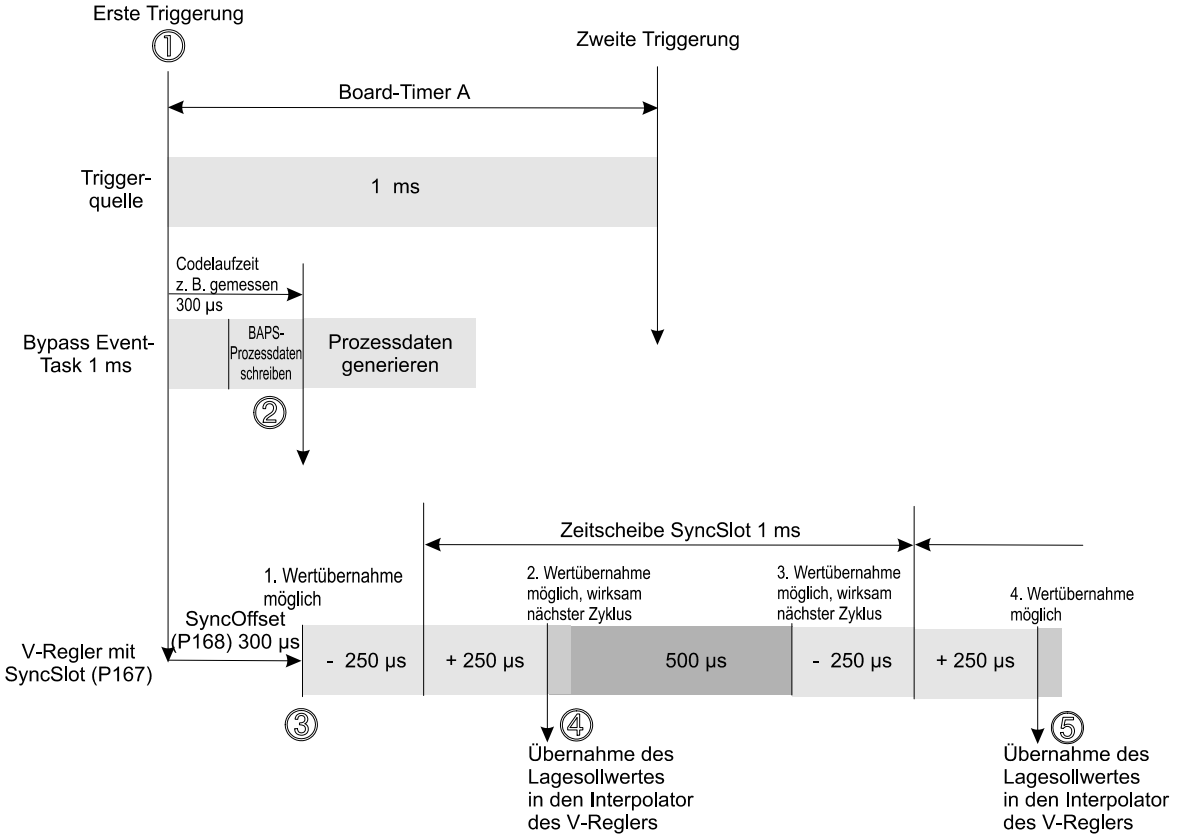
FB INTR_SET zum Start einer Bypass Event-Task auf Board-Timer A in einer Anlauf-Task.

- In der Ωmega Drive-Line II Ressource ist eine **Event-Task** auf das Ereignis **Board-Timer A** einzufügen. In der Task wird als erstes die Auswertung der Optionskarten (IEI-02/MFM-01) und die BAPS-Prozeßdatenkommunikation eingefügt. Anschließend erfolgt die Sollwertgenerierung.
- Im V-Regler wird über WinBASS innerhalb der Service-Oberfläche die Sollwertsynchronisierung angepaßt: Die SyncSlotzeit (P167) wird auf die **Board-Timer A Zeit** eingestellt und der Parameter SyncOffset (P168) gemäß Erläuterung durch die Zeichnung.

HINWEIS

In der V-Regler Betriebsart "Gleichlauf mit synchroner Sollwertvorgabe" sind bei der BAPS-Prozeßdatenkommunikation über den Parameter P258 (32-Bit Winkel) je nach gewählter BAPS-Zeitscheibe die untersten Bits des Übergabewertes auszumaskieren (FALSE setzen).
Vorgehensweise siehe Tabelle auf Seite 74

Zeitscheibenkonfiguration hochgenaue BAPS (Beispiel B)



Prozeßdatenkommunikation zum V-Regler in Board-Timer A Event-Task und Triggerung der Baugruppen auf Board-Timer A-Event.

Der SyncOffset dient zur Optimierung der Durchlaufzeit bis der Lagesollwert im V-Regler übernommen wird. Wenn es ausreicht, daß der Wert im nächsten Zyklus f übernommen wird, dann ist der SyncOffset auf NULL zu setzen. Anderenfalls ist die Codelaufzeit einschließlich der BAPS Prozeßdatenkommunikation zu messen und als SyncOffset im V-Regler zu parametrieren. Dann wird der Wert in Punkt Ð übernommen.

Dabei ist zu beachten, daß die Werte, die erst bei der 2. Werteübernahme über die BAPS übertragen werden, nicht in Punkt Ð, sondern in Punkt f wirksam werden.

Im V-Regler werden alle Parameter, außer den Lagesollwerten, direkt nach der Wertübernahme wirksam.

4.6.9 Umsetzung einer BAPS innerhalb des synchronen Bussystems CANsync

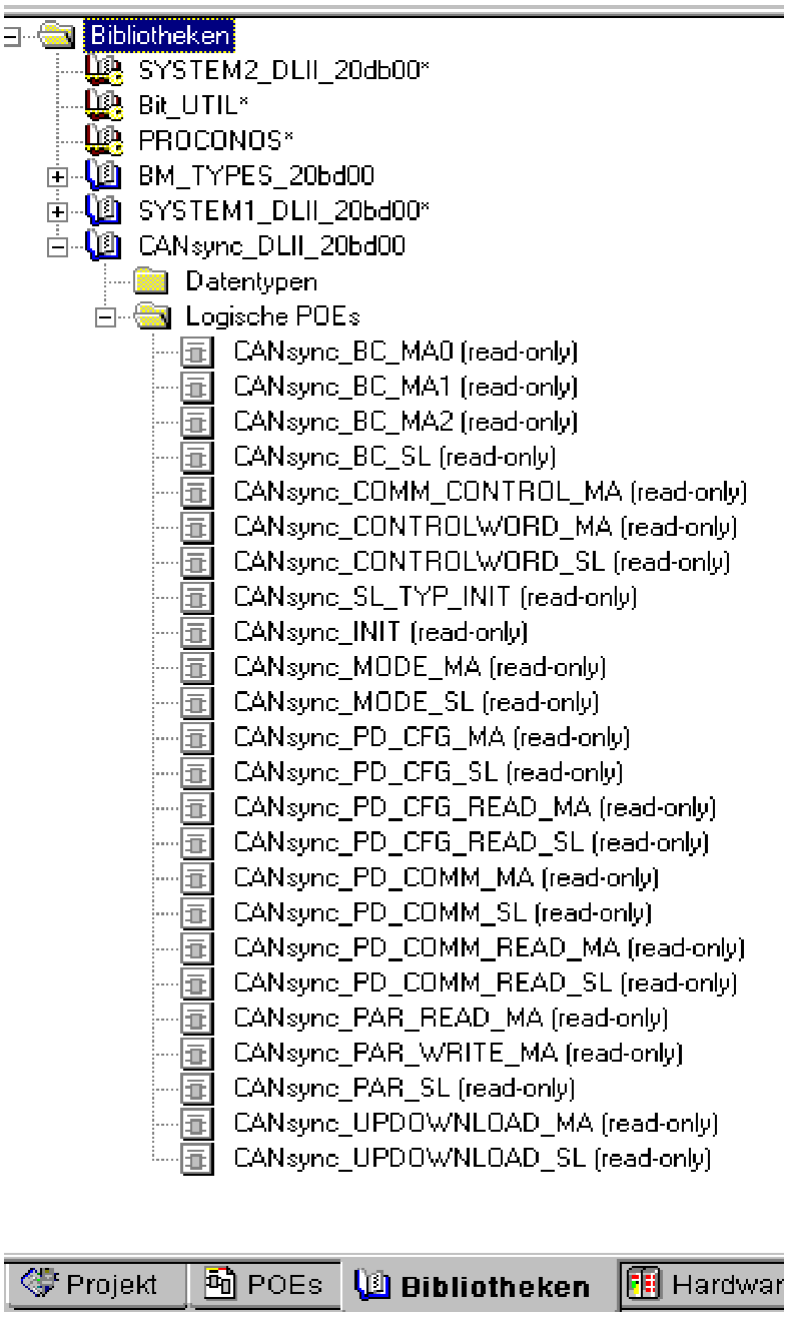
Beispiel C:

Das Ωmega Drive-Line II ist Teil im synchronen Bussystem CANsync (Master oder Slave). In diesem Fall muß die Prozeßdatenkommunikation über CANsync und die BAPS-Prozeßdatenkommunikation zum V-Regler in einer Task erfolgen.

Das Event für die Task ist das Synchronisiersignal vom CANsync Bus. Die Ωmega Drive-Line II Ressource muß eine Bypass Event-Task mit dem Ereignis CANsync (12) haben. In diese Bypass Event-Task wird die BAPS-Prozeßdatenkommunikation eingefügt.

Die Baugruppen die Triggersignale benötigen, V-Regler und Optionskarten, werden ebenfalls mit dem Synchronisiersignal vom CANsync Bus getriggert.

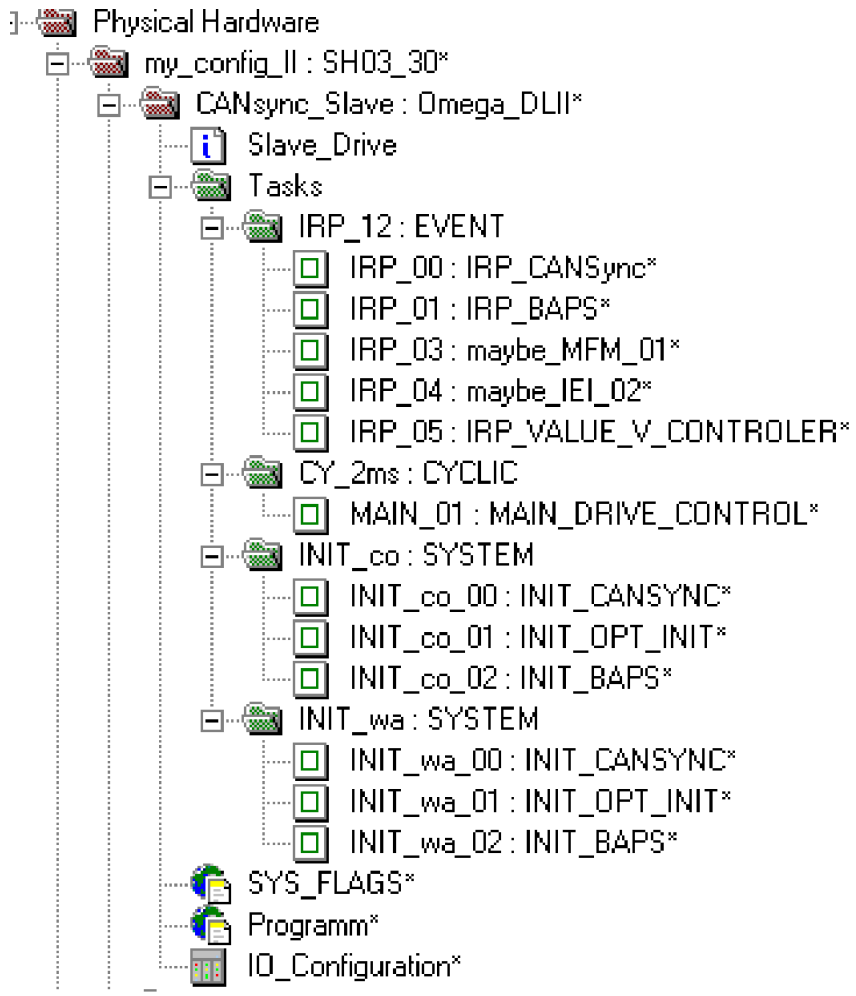
- In der Ωmega Drive-Line II Ressource wird die Bypass Event-Task mit dem Ereignis **CANsync (12)** eingefügt.
- Über WinBASS wird die Betriebsart des V-Reglers auf **Gleichlauf** mit **synchroner Sollwertvorgabe parametriert**. Der Parameter **SyncSlot (P167)** ist auf die CANsync-Zeitscheibe einzustellen. Der **SyncOffset (P168)** ist Null.
- Für die Umsetzung der BAPS-Prozeßdatenkommunikation innerhalb der CANsync Event-Task und CANsync-Prozeßdatenkommunikation sind folgende Ωmega Drive-Line II Bibliotheken einzufügen:
  - BM_TYPES_20bd00 oder höher: Datentypen für CANsync und BAPS.
  - SYSTEM1_DLII_20bd00 oder höher: BAPS-FBs und FB OPT_INIT.
  - SYSTEM2_DLII_20bd00 oder höher: FB INTR_SET für die CANsync-Bibliothek.
  - CANsync_DLII_20bd00 oder höher: CANsync Event-Task und Busankopplung initialisieren.



Bibliotheken für die Umsetzung einer BAPS-Zeitscheibe synchron zur CANsync-Prozeßdatenkommunikation

Mit den FBs der Bibliotheken wird die CANsync Prozeßdaten- und Bedarfsdatenkommunikation umgesetzt. Die Bypass Event-Task CANsync wird vom FB CANsync_INIT gestartet. Die BAPS Prozeßdatenkommunikation wird anschließend mit dem FB BAPS_INIT initialisiert.

Der Einsatz der Funktionsbausteine für die CANsync- und BAPS- Implementierung ist der jeweiligen Baugruppenbeschreibung zu entnehmen. Die Umsetzung sollte zu folgender Task- und Programmstruktur führen.



Grundstruktur einer Ressource mit CANsync-Busankopplung und BAPS-Prozeßdatenkommunikation.

In diese Grundstruktur werden die Programm-POE´s der Sollwertvorgabe zur BAPS eingefügt. Die Initialisierung der CANsync-Anschaltung und der BAPS wird mit der Beschaltung der Triggersignale, für Baugruppen die Triggersignale benötigen, über den FB OPT_INIT vervollständigt.

Alle Baugruppen werden mit dem Synchronisiersignal von CANsync getriggert.

Benötigter Trigger 1 für die Optionskarte IEI-02 auf Optionssteckplatz 1 oder 2

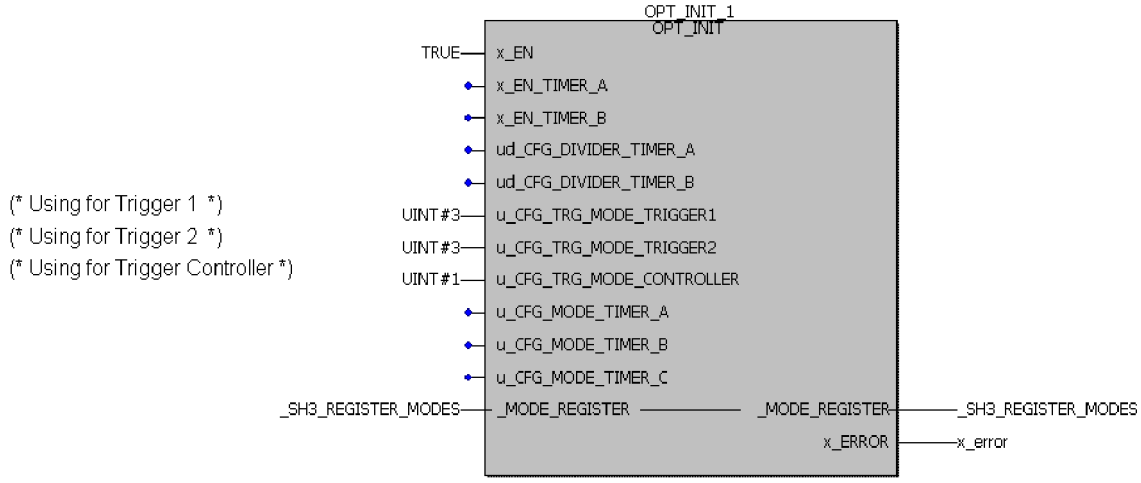
Mode für Trigger 1	Synchronisierung/Triggerung mit	Quelle
3	CANsync Synchronisier-Signal	Sync Net

Benötigter Trigger 2 für die Optionskarte MFM-01 auf Optionssteckplatz 1 oder 2

Mode für Trigger 2	Synchronisierung/Triggerung mit	Quelle
3	CANsync Synchronisier-Signal	Sync Net

Benötigter Trigger Controller (V-Regler-BAPS)

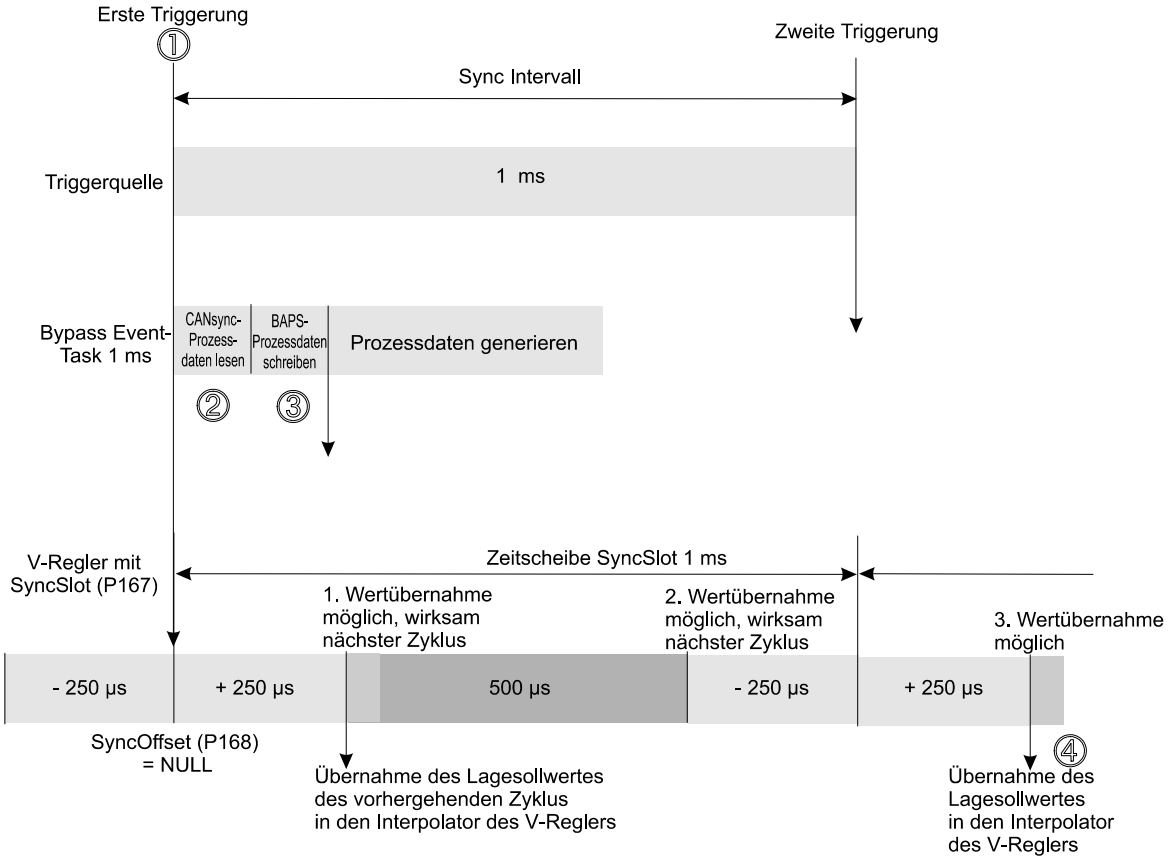
Mode für Trigger Controller	Synchronisierung mit	Quelle
1	CANsync Synchronisier-Signal	Sync Net



FB OPT_INIT mit Triggersignalbeschaltung auf Sync-Signal vom synchronen Bus CANsync.

Mit dieser Umsetzung ist das Zeitscheibenverhalten wie in der folgenden Zeichnung synchronisiert. Es ist sichergestellt, daß die Übernahme des Sollwertes über die BAPS zum V-Regler immer nach der CANsync-Prozeßdatenkommunikation erfolgt.

Zeitscheibenkonfiguration Bypass Event -Task auf CANsync (Beispiel C)



CANsync Standard Zeitscheibenkonfiguration.
Da mit SyncOffset gleich NULL der Lagesollwert erst im nächsten Zyklus wirksam wird, kann die BAPS-Prozeßdatenkommunikation auch am Ende der Event-Task ausgeführt werden.

HINWEIS

In der V-Regler Betriebsart "Gleichlauf mit synchroner Sollwertvorgabe" erfolgt die Übernahme des Lagesollwertes innerhalb einer 500 µs Zeitscheibe. Andere Parameter, z. B. die Momentengrenze oder der Drehzahlsollwert werden sofort übernommen.
In der V-Regler Betriebsart "Gleichlauf mit synchroner Sollwertvorgabe" sind bei der BAPS-Prozeßdatenkommunikation über den Parameter P258 (32-Bit Winkel) je nach gewählter BAPS-Zeitscheibe die untersten Bits des Übergabewertes auszumaskieren (FALSE setzen).
Vorgehensweise siehe Tabelle auf Seite 74

4.6.10 Umsetzung einer Timer-Event Task für eine zyklische serielle Kommunikation

Beispiel D:

In die Ωmega Drive-Line II Ressource soll eine Bypass Event-Task auf einen Timer eingerichtet werden. Es gibt folgende Events für Timer.

i_EVENT	HW-Ereigniss(e)	Interrupt-Level	Triggerquelle FB OPT_INIT
0	CPU-Timer 1	Level 14	nein
2	CPU-Timer 2	Level 13	nein
5	Board-Timer A	Level 14	ja
6	Board-Timer A	Level 13	ja

Bei der Auswahl des Timer-Events ist zu beachten ob mit diesem eine Baugruppe getriggert werden soll (IEI-02, MFM-01, V-Regler). Eine Triggerung von Baugruppen ist nur über den Timer A möglich, nicht über die internen CPU-Timer 1 (oder 2).

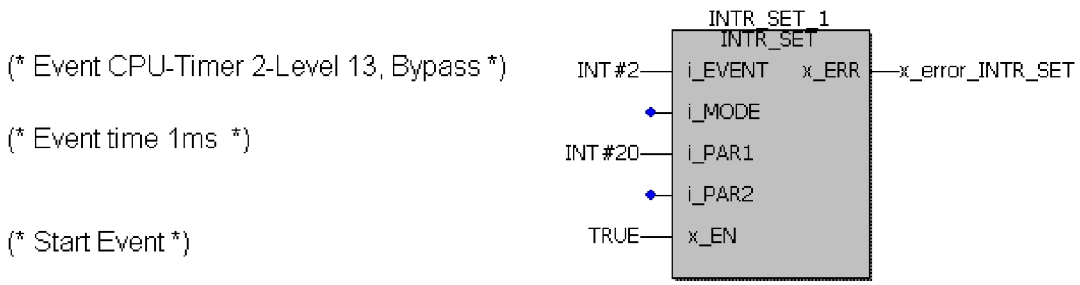
Daraus folgt:

- Bypass Event-Task über FB INTR_SET in einer Anlauf-Task starten. Ist das Event ein **CPU-Timer 1** (oder 2) wird die Zeit am Eingang i_PAR1 parametriert. Eine Triggerung von Baugruppen, die Triggersignale benötigen, ist nicht möglich, daher wird der FB OPT_INIT nicht verwendet. Die Event-Task läuft völlig asynchron zu der Wertewandlung der Baugruppen. Einsatzbeispiel: Zeitscheibe für 3964R®-Protokoll.
- Bypass Event-Task über FB INTR_SET in einer Anlauf-Task starten. Ist das Event ein **Board-Timer A** bleibt der Eingang i_PAR1 an FB INTR_SET unbeschaltet. Die Timer-Zeit wird über den FB OPT_INIT eingerichtet. Eine Triggerung von Baugruppen, die Triggersignale benötigen, ist möglich, daher werden diese Baugruppen mit dem FB OPT_INIT auf Timer A beschaltet. Einsatzbeispiel: Hochgenaue BAPS Event-Task.

Eine feste Zeitscheibe ist in der einfachsten Anwendung wie folgt einzufügen:

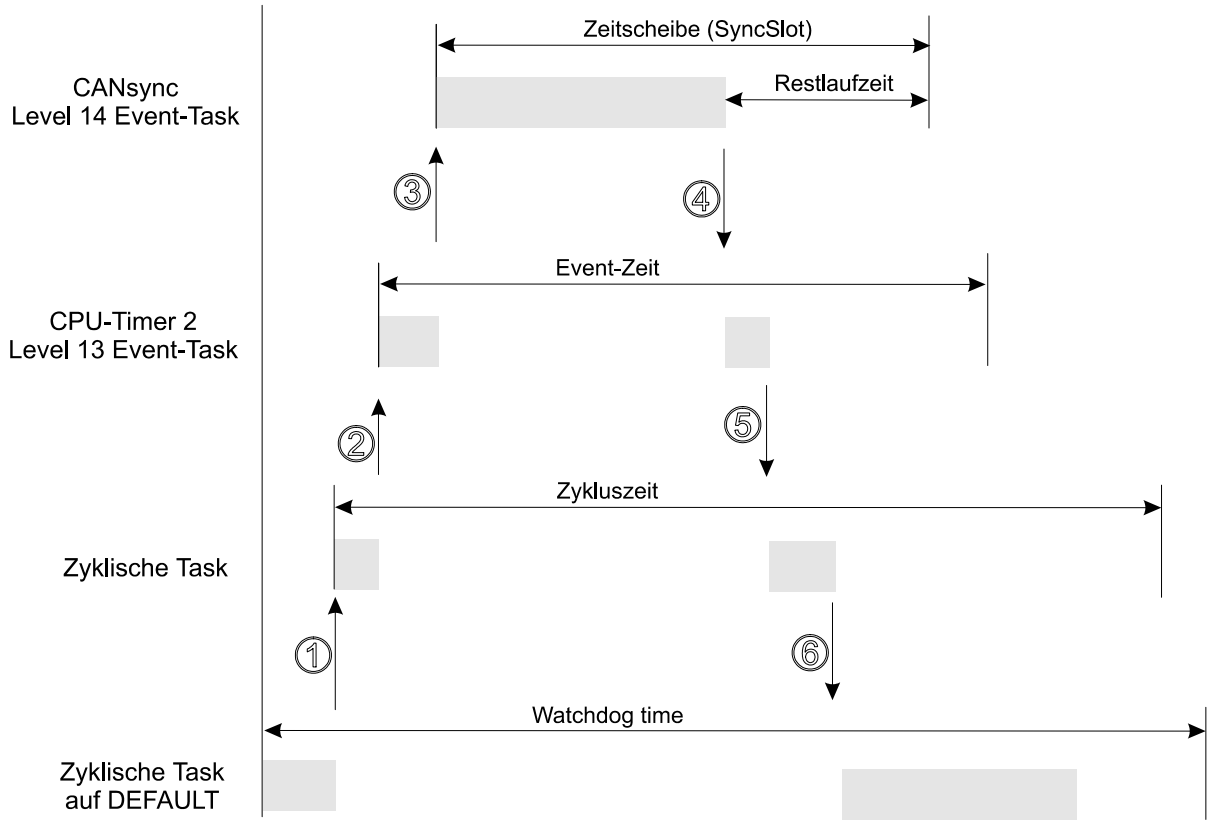
- In der Ωmega Drive-Line II Ressource ist eine Bypass Event-Task mit dem Ereignis CPU-Timer 1 (oder 2) mit dem Level 14 oder 13 einzugeben. Die Programm-POEs werden in diese Event-Task implementiert.
- Da es sich um eine Bypass Event-Task handelt, wird das Event mit dem FB INTR_SET in einer Anlauf-Task gestartet.

Interruptzeit = i_PAR1 • 50 µs



Funktionsbaustein INTR_SET in einer Anlauf-Task: Start eines CPU Timer 2 Level 13 -Interrupt mit der Interrupt-Zeit 1 ms.

Codelauf Timer Level 13 Event-Task mit Level 14 CANsync Event-Task (Beispiel D)



Codelauf einer Ωmega Drive-Line II Ressource mit Event-Tasks und zyklischen Tasks.

HINWEIS

Eine höherpriore Event-Task unterbricht eine niederpriore Event-Task, daher sollten Timer Event-Tasks, die parallel zu synchronen Bussystemen gestartet sind, immer mit dem Level 13 eingesetzt werden. Der FB INTR_SET kann mit x_EN = FALSE ein zuvor gestartetes Event sperren.

4.7 Prozeßdatensollwert zum Gleichlauf mit synchroner Sollwertvorgabe

In der Betriebsart Gleichlauf mit synchroner Sollwertvorgabe ist folgendes zu beachten:

Abhängig von den ausgewählten Parametern P 167 BS Sync.-Slot und P 103 PWM Frequenz wird eine bestimmte Anzahl von Bits im Parameter P 258 ausgeblendet. Dies ist notwendig, um für die interne Interpolation vom eingestellten BS Sync.-Slot auf das Regelungsintervall immer ganze Zahlen zu bekommen.

P 167 BS Sync.-Slot [µs]	P 103 PWM Frequenz [kHz]	Maske über P 258
0	4	FFF8 0000
500	4	FFFF FFFC
1000	4	FFFF FFF8
2000	4	FFFF FFF0
4000	4	FFFF FFE0
8000	4	FFFF FFC0
0	8	FFF8 0000
500	8	FFFF FFF8
1000	8	FFFF FFF0
2000	8	FFFF FFE0
4000	8	FFFF FFC0
8000	8	FFFF FF80

Der tatsächlich wirksame Wert Phi-Sollwert wird in Parameter P 258 angezeigt.

5 ETHERNET (OPTIONAL)

5.1 Allgemeines

Nicht alle Geräte Ωmega Drive-Line II werden mit der Option Ethernet ausgeliefert.

HINWEIS

Sie können anhand der Softwareversionsnummer (SV) auf dem 2.Typenschild erkennen, ob Sie ein Gerät mit oder ohne Ethernetfunktionalität haben.
Beispiel: SV: 0003-I002-0000
Die zweite 4er-Gruppe (I002) gibt die Funktion der Steuerung an.
I0xx (auch **K0**xx) bedeutet, daß Sie ein Gerät **ohne** Ethernetfunktion haben.
I1xx (auch **K1**xx) bedeutet, daß Sie ein Gerät **mit** Ethernetfunktion haben.

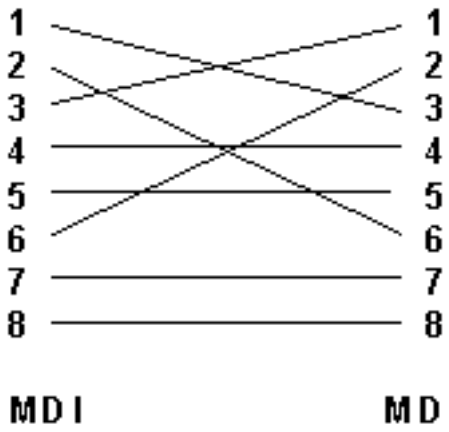
5.2 Anschlüsse und Verkabelung

Für Kommunikation über Ethernet steht der RJ45-Anschluss X39 zur Verfügung.

Das Ωmega Drive-Line II erkennt die Netztypen 10BaseT (10Mbit) und 100BaseTX (100Mbit, Fast-Ethernet) am laufenden Verkehr und stellt sich automatisch ein.

Die Verkabelung zum Sternkoppler (Hub bzw. Switch) erfolgt mit CAT5-Twisted-Pair-Kabeln (siehe “Zubehör” auf Seite 27).

Optional können Sie auch einen PC und ein Ωmega Drive-Line II direkt verbinden (ohne Hub). In diesem Fall müssen Sie jedoch ein "gekreuztes" Kabel (Cross-Link-Kabel) verwenden.



Cross-Link-Kabel für Ethernet

Eine busförmige (linienförmige) Verkabelung mit Koaxial-Kabel (10Base2, Thin Ethernet, Cheapernet) ist nicht vorgesehen.

5.3 TCP/IP-Einstellungen

Bei der Kommunikation über das TCP/IP-Protokoll muß für jedes Gerät eine andere IP-Adresse eingestellt werden. Siehe auch “Kommunikation über Ethernet (optional)” auf Seite 34. Die IP-Adresse ist nicht identisch mit der MAC-Adresse. Eine MAC-Adresse ist weltweit einmalig und wird vom Geräteher-steller in einem Gerät eingestellt.

Sie müssen für die TCP/IP-Kommunikation eine IP-Adresse einstellen.

Werkseitig voreingestellt ist die Adresse 192.168.1.1 + DIP-Schalter-Stellung

und die IP-Maske (Subnet-Mask) 255.255.255.0
(siehe “Voreingestellte, variable IP-Adresse (Auslieferungszustand)” auf Seite 80)

Eine anwenderdefinierte Einstellung der IP-Adresse und der IP-Maske braucht nur einmalig vorgenommen zu werden. Die IP-Adresse und die IP-Maske wird im NOVRAM (nichtflüchtiger Speicher) des Ωmega Drive-Line II gespeichert.

HINWEIS

Voraussetzungen für das Einstellen von IP-Adresse und IP-Maske sind:
- Programm PROPROG wt II
- Bibliothek BM_TYPES_20bd00 (oder höher),

sowie Kenntnisse der Hardware des Ωmega Drive-Line II (siehe Kap. 2) und Ethernet-Kenntnisse (IP-Adresse, IP-Maske und Router)

Für die Einstellung der IP-Adresse und der IP-Maske des Ωmega Drive-Line II steht unter PROPROG wt II die Struktur `ETHERNET_CONFIG_BMSTRUCT` zur Verfügung.

Zur Einstellung der IP-Adresse und der IP-Maske muß der Anwender in einem PROPROG wt II-Projekt eine globale Variable vom Datentyp

```
ETHERNET_CONFIG_BMSTRUCT
```

anlegen und auf die Adresse

```
%MB3.180016
```

legen.

Beispiel:

```
_Ethernet_config  AT     %MB3.180016 : ETHERNET_CONFIG_BMSTRUCT
```

dabei ist:

`_Ethernet_config`	der Variablenname mit der Datentyp-Kurzbezeichnung „_“ für STRUCT
`ETHERNET_CONFIG_BMSTRUCT`	der Datentyp der Variable
`%MB3.180016`	die Adresse

Die Struktur `ETHERNET_CONFIG_BMSTRUCT` ist wie folgt definiert:

```
ETHERNET_CONFIG_BMSTRUCT      :      Struct
      d_IP_CONFIG             :      DWORD;
```

```
    a_IP_ADDRESS            :     USINT_4_BMARRAY;
    a_IP_MASK               :     USINT_4_BMARRAY;
    d_ROUTER                :     DWORD;
END_STRUCT;
```

Beispiel für den Zugriff auf ein Element der Struktur:

```
_Ethernet_config.d_IP_CONFIG
```

dabei ist:

`_Ethernet_config`	der Variablenname
`d_IP_CONFIG`	das Element der Struktur mit der Datentyp-Kurzbezeichnung „d“ für DWORD

In der nachfolgenden Beschreibung wird der Variablenname durch * ersetzt.

Die Struktur-Elemente `*.a_IP_ADDRESS` und `*.a_IP_MASK` sind jeweils vom Datentyp `USINT_4_BMARRAY`. Der Datentyp `USINT_4_BMARRAY` ist ein Feld mit 4 Einträgen des Datentyps Unsigned Short Integer:

```
    USINT_4_BMARRAY             ARRAY[0..3]  OF    USINT;
```

Eine IP-Adresse wird in der Variablen `*` im Eintrag `a_IP_ADDRESS` an den Stellen 0 bis 3 eingetragen.

Beispiel: Eintragen der IP-Adresse 192.168.75.190 (in Structured Text (ST))

```
    *.a_IP_ADDRESS[0]       :=    USINT#192;
    *.a_IP_ADDRESS[1]       :=    USINT#168;
    *.a_IP_ADDRESS[2]       :=    USINT#75;
    *.a_IP_ADDRESS[3]       :=    USINT#190;
```

Eine IP-Maske wird in der Variablen `*` im Eintrag `*.a_IP_MASK` an den Stellen 0 bis 3 eingetragen.

Beispiel: Eintragen der IP-Maske 255.255.255.0 (in Structured Text (ST))

```
    *.a_IP_MASK[0]          :=    USINT#255;
    *.a_IP_MASK[1]          :=    USINT#255;
    *.a_IP_MASK[2]          :=    USINT#255;
    *.a_IP_MASK[3]          :=    USINT#0;
```

Das Ωmega Drive-Line II bietet verschiedene Möglichkeiten zur IP-Adressen-Auswahl und der Nutzung der Dip-Schalter des Ωmega Drive-Line II zur automatischen IP-Adressen-Numerierung in einem Netzwerk (mit mehreren Ωmega Drive-Line II). Die Auswahl erfolgt über den Eintrag

```
    *.d_IP_CONFIG
```

der Struktur

`ETHERNET_CONFIG_BMSTRUCT.`

5.4 Einstellen von IP-Adresse und IP-Maske

Vorgehensweise:

– Eintragen der IP-Adresse
  `*.a_IP_ADRESS[0] .. *.a_IP_ADRESS[3]`
– Eintragen der IP-Maske
  `*.a_IP_MASK[0] .. *.a_IP_MASK[3]`
– Speichern der IP-Adresse und IP-Maske durch Beschreiben von
  `*.d_IP_CONFIG`
– Einstellen des Verhaltens von Routern im Netz
  `*.d_ROUTER`

Für die Einstellung der IP-Adresse gibt es folgende Möglichkeiten, die nachfolgend beschrieben werden:

- Selbst gewählte, feste (von Dip-Schalter-Stellung unabhängige) IP-Adresse
- Selbst gewählte, variable (von Dip-Schalter-Stellung abhängige) IP-Adresse
- Voreingestellte, variable (von Dip-Schalter-Stellung abhängige) IP-Adresse

5.4.1 Selbst gewählte, feste IP-Adresse

`*.d_IP_CONFIG` gleich `DWORD#16#12345678`

Die IP-Adresse ist von der Dip-Schalter-Stellung **unabhängig**.

Die IP-Adresse aus

`*.a_IP_ADDRESS[0] .. *.a_IP_ADDRESS[3]`

und die IP-Maske aus

`*.a_IP_MASK[0] .. *.a_IP_MASK[3]`

wird im Ωmega Drive-Line II gespeichert.

Beispiel (Reihenfolge muß eingehalten werden):

– Eintragen der IP-Adresse 192.168.75.190 (Darstellung in Structured Text (ST))

```
*.a_IP_ADDRESS[0] :=     USINT#192;
*.a_IP_ADDRESS[1] :=     USINT#168;
*.a_IP_ADDRESS[2] :=     USINT#75;
*.a_IP_ADDRESS[3] :=     USINT#190;
```

– Eintragen der IP-Maske 255.255.255.0 (Darstellung in Structured Text (ST))

```
*.a_IP_MASK[0]    :=     USINT#255;
*.a_IP_MASK[1]    :=     USINT#255;
*.a_IP_MASK[2]    :=     USINT#255;
*.a_IP_MASK[3]    :=     USINT#0;
```

– Speichern der IP-Adresse und der IP-Maske im Ωmega Drive-Line II

```
*.d_IP_CONFIG     :=     DWORD#16#12345678;
```

Das Ωmega Drive-Line II hat dann (unabhängig von der Dip-Schalter-Einstellung) die IP-Adresse 192.168.75.190.

HINWEIS

IP-Adressen xxx.yyy.zzz.0 und xxx.yyy.zzz.255 sind für Geräte nicht erlaubt.

5.4.2 Selbst gewählte, variable IP-Adresse

```
*.d_IP_CONFIG gleich DWORD#16#12345600;
```

Die IP-Adresse ist von der Dip-Schalter-Stellung **abhängig**.

Die IP-Adresse aus

```
*.a_IP_ADDRESS[0],
*.a_IP_ADDRESS[1],
*.a_IP_ADDRESS[2] und
*.a_IP_ADDRESS[3] + ”Nummer Dip-Schalter”
```

und die IP-Maske aus

```
*.a_IP_MASK[0] .. *.a_IP_MASK[3]
```

wird im Ωmega Drive-Line II gespeichert.

Beispiel (Reihenfolge muß eingehalten werden):

– Eintragen der IP-Adresse 192.168.75.190 (Darstellung in Structured Text (ST))

```
*.a_IP_ADDRESS[0] :=    USINT#192;
*.a_IP_ADDRESS[1] :=    USINT#168;
*.a_IP_ADDRESS[2] :=    USINT#75;
*.a_IP_ADDRESS[3] :=    USINT#190;
```

– Eintragen der IP-Maske 255.255.255.0 (Darstellung in Structured Text (ST))

```
*.a_IP_MASK[0]    :=    USINT#255;
*.a_IP_MASK[1]    :=    USINT#255;
*.a_IP_MASK[2]    :=    USINT#255;
*.a_IP_MASK[3]    :=    USINT#0;
```

– Speichern der IP-Adresse und der IP-Maske im Ωmega Drive-Line II

```
*.d_IP_CONFIG     :=    DWORD#16#12345600;
```

Bei dieser eingestellten IP-Basisadresse 192.168.75.190 gilt dann:

Das Ωmega Drive-Line mit der Dip-Schalter-Stellung “**0**” hat die IP-Adresse 192.168.75.19**0**,

das Ωmega Drive-Line mit der Dip-Schalter-Stellung “**1**” hat die IP-Adresse 192.168.75.19**1**,

...,

das Ωmega Drive-Line mit der Dip-Schalter-Stellung “**30**” hat die IP-Adresse 192.168.75.2**20**,

das Ωmega Drive-Line mit der Dip-Schalter-Stellung “**31**” hat die IP-Adresse 192.168.75.2**21**.

HINWEIS

Die IP-Adressen xxx.yyy.zzz.0 und xxx.yyy.zzz.255 sind für Geräte nicht erlaubt.
Dip-Schalter-Stellungen können 0 ... 31 sein (Dip-Switch 1 – Dip-Switch 5).
Siehe “Einstellung der Slavenummer” auf Seite 22.

Hierbei ist zu beachten:
IP-Adresse xxx.yyy.zzz.224 + 31 entspricht IP-Adresse xxx.yyy.zzz.255 und ist für ein Gerät nicht erlaubt.

HINWEIS

Das Ωmega Drive-Line II übernimmt die Dip-Schalterstellung nach dem Einschalten oder nach einem Reset. Spätere Änderungen werden erst wieder nach Aus-/Einschalten bzw. Reset übernommen.

Achtung:
Die Dip-Schalterstellung wird auch für die CANsync-Adressierung verwendet!

5.4.3 Voreingestellte, variable IP-Adresse (Auslieferungszustand)

Die IP-Adresse ist von der Dip-Schalter-Stellung **abhängig**.

Die im Ωmega Drive-Line II voreingestellte

IP-Adresse 192.168.1.1 + “Nummer Dip-Schalter”

und die voreingestellte

IP-Maske 255.255.255.0 wird verwendet.

Bei dieser eingestellten IP-Basisadresse 192.168.1.1 gilt dann:

Das Ωmega Drive-Line mit der Dip-Schalter-Stellung “**0**” hat die IP-Adresse 192.168.1.**1**,

das Ωmega Drive-Line mit der Dip-Schalter-Stellung “**1**” hat die IP-Adresse 192.168.1.**2**,

...,

das Ωmega Drive-Line mit der Dip-Schalter-Stellung “**30**” hat die IP-Adresse 192.168.1.**31**,

das Ωmega Drive-Line mit der Dip-Schalter-Stellung “**31**” hat die IP-Adresse 192.168.1.**32**.

HINWEIS

Dip-Schalter-Stellungen können 0 .. 31 sein (Dip-Switch 1 – Dip-Switch 5). Siehe “Einstellung der Slavenummer” auf Seite 22.

HINWEIS

Das Ωmega Drive-Line II übernimmt die Dip-Schalterstellung nach dem Einschalten oder nach einem Reset. Spätere Änderungen werden erst wieder nach Aus-/Einschalten bzw. Reset übernommen.

Achtung:
Die Dip-Schalterstellung wird auch für die CANsync-Adressierung verwendet!

5.5 Einstellung des Verhaltens bei Routern im Netz

Das Ωmega Drive-Line II kann im Ethernet mit Geräten, deren IP-Adresse außerhalb des Subnetzes liegt, kommunizieren. Für diese Kommunikation kann eingestellt werden, ob ein Router verwendet werden soll, der nach dem Router Discovery Verfahren (RFC 1256) vom Ωmega Drive-Line II gefunden wird.

Mit der Einstellung (= Auslieferungszustand)

*.d_ROUTER ungleich DWORD#16#4E6F5F52

verwendet das Ωmega Drive-Line II Router nachdem es diese im Netz erkannt hat.

Mit der Einstellung

*.d_ROUTER gleich DWORD#16#4E6F5F52

verwendet das Ωmega Drive-Line II **keine** Router.

HINWEIS

Konsultieren Sie Ihren Netzwerk-Administrator über die in Ihrem LAN gültigen Routing-Regeln.

Die Adressen 192.168.yyy.zzz beispielsweise werden prinzipiell nicht geroutet.

5.6 Kommunikation zwischen Ωmega Drive-Line II und PROPROG wt II über Ethernet

Nachdem die IP-Adresse und die IP-Maske eingestellt und gespeichert sind, ist die Kommunikation zwischen dem Ωmega Drive-Line II und einem PC mit PROPROG wt II prinzipiell möglich. In PROPROG wt II muß dazu unter Ressource/Einstellungen die Kommunikation auf DLL und die IP-Adresse des Ωmega Drive-Line II eingestellt werden. Siehe “Kommunikation über Ethernet (optional)” auf Seite 34.

Voraussetzung ist jedoch, daß auf dem PC die TCP/IP-Netzwerkunterstützung des MS-Windows-Betriebssystems installiert und richtig eingestellt ist.

Ωmega Drive-Line II und PC müssen sich im selben Subnetz befinden; anderenfalls muß die Verbindung geroutet werden (siehe “Einstellung des Verhaltens bei Routern im Netz” auf Seite 81).

Wenn Sie eine direkte Verbindung zwischen Ωmega Drive-Line II und PC mit Cross-Link-Kabel (ohne Hub) benutzen, müssen Sie auch am PC eine feste IP-Adresse und dieselbe IP-Maske wie am Ωmega Drive-Line II einstellen.

Die IP-Maske definiert die Sichtbarkeit zweier Teilnehmer entsprechend der Regel:

IP-Adresse_Teilnehmer_A OR IP-Maske = IP-Adresse_Teilnehmer_B OR IP-Maske.

Wenn Sie über einen Hub (oder Switch) an Ihr LAN angeschlossen sind, können Sie PC-seitig auch die automatische Adreßvergabe (DHCP) benutzen. Sie müssen jedoch sicherstellen, daß der DHCP-Server Ihres Netzwerkes dem PC nur eine solche IP-Adresse zuweist, die zu der am Ωmega Drive-Line II eingestellten IP-Maske und IP-Adresse paßt.

DHCP macht bei Ωmega Drive-Line II keinen Sinn, weil die eindeutige Kenntnis der IP-Adresse für die Identifizierung des jeweiligen Ωmega Drive-Line II im Netz und den Aufbau der Verbindung zu PROPROG wt II oder dem Ωmega OPC-Server notwendig ist. Eine Identifizierung mit Namen über höhere Protokolle (wie SMB im Windows-LAN-Manager-Netz) wird nicht unterstützt. Sie können jedoch in der lmhost-Datei Ihres PC's eine feste Zuordnung zwischen der IP-Adresse des Ωmega Drive-Line II und einem logischen Namen, den Sie in PROPROG wt II verwenden möchten, vornehmen.

6 BAPS - BAUMÜLLER ANTRIEBE PARALLELE SCHNITTSTELLE

6.1 BAPS Allgemein

Die BAPS ist eine interne Kommunikationsschnittstelle für den Datenaustausch zwischen der V-Regler-Baugruppe und der Ωmega Drive-Line II - Baugruppe.

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 V-Reglers.

Ωmega Drive-Line II und V-Regler

Prozeßdaten:

In einem definierbaren Zeitraster übernimmt das Ωmega Drive-Line II die Istwerte und das Statuswort des V-Reglers von der BAPS und übergibt Sollwerte und Steuerwort über die BAPS an den V-Regler.

Bei der Übergabe der Istwerte und des Statusworts wird im Ωmega Drive-Line II das Hardware-Ereignis "BAPS-Prozeßdaten" ausgelöst. Dieses Ereignis kann im Ωmega Drive-Line II eine Event-Task auslösen, in der die BAPS-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 BAPS_INIT.

Bedarfsdaten:

Das Ωmega Drive-Line II übergibt einen Parameter-Lesen- oder Parameter-Schreiben-Auftrag an die BAPS. Der V-Regler liest den Auftrag von der BAPS, bearbeitet den entsprechenden Parameter im V-Regler (siehe jeweilige V-Regler-Beschreibung) und gibt das Ergebnis an die BAPS zurück. Das Ωmega Drive-Line II liest dann das Ergebnis der Kommunikation von der BAPS.

Programmierung der BAPS-Kommunikation auf dem Ωmega Drive-Line II

Die Programmierung des Ωmega Drive-Line II erfolgt mit dem Programmiersystem PROPROG wt II (siehe Handbuch PROPROG wt II).

Für die Programmierung der BAPS Kommunikation stehen die Baumüller Anwender Bibliotheken SYSTEM1_DLII_20bd00 und SYSTEM2_DLII_20bd00 (oder höher) zur Verfügung.

In der Bibliothek SYSTEM1_DLII_20bd00 (oder höher) sind Funktionsbausteine für die Initialisierung der Prozeßdatenkommunikation, für die Prozeßdatenkommunikation und für die Bedarfsdatenkommunikation vorhanden.

Für die Prozeßdatenkommunikation werden folgende FBs benötigt:

BAPS_INIT	Initialisierung der Prozeßdatenkommunikation.

BAPS_PD_COMM8	Prozeßdatenkommunikation (maximal 8 Soll- und 8 Istwerte)

oder

BAPS_PD_COMM24	Prozeßdatenkommunikation (maximal 2 Soll- und 4 Istwerte).

oder

BAPS_PD_COMM2	Prozeßdatenkommunikation (maximal 2 Soll- und 2 Istwerte).

Zur Überwachung der Prozeßdatenkommunikation (optional):

BAPS_PD_CONTROL	Überwachung des Aufrufs des FB BAPS_PD_COMMxx (und damit indirekt Überwachung des Aufrufs der Event-Task)

Für die Bedarfsdatenkommunikation werden folgende FBs benötigt:

BAPS_PAR_READ	Bedarfsdatenkommunikation (Parameter-Lesen-Auftrag)
BAPS_PAR_WRITE	Bedarfsdatenkommunikation (Parameter-Schreiben-Auftrag)

und

BAPS_SD_CONTROL	Aufbau und Überwachung der Bedarfsdatenkommunikation.

Die Bibliothek SYSTEM2_DLII_20bd00 (oder höher) enthält u.a. den Funktionsbaustein:

INTR_SET	wird vom FB BAPS_INIT verwendet (FB zur Verknüpfung und Aktivierung eines Hardware-Signal mit dem Ereignis BAPS-Prozeßdaten.)

6.2 Funktionsbausteine für BAPS - 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 BAPS sind verfügbar:

Funktionsbaustein	**Kurzbeschreibung**
BAPS_INIT	Initialisierung der BAPS (Baumüller Antrieb Parallele Schnittstelle) im Ωmega Drive-Line II
BAPS_PAR_READ	BAPS Parameter lesen
BAPS_PAR_WRITE	BAPS Parameter schreiben
BAPS_PD_COMM2	Prozeßdatenkommunikation über die BAPS im Ωmega Drive-Line II für max. 2 Soll- und Istwerte
BAPS_PD_COMM24	Prozeßdatenkommunikation über die BAPS im Ωmega Drive-Line II für max. 2 Soll- und 4 Istwerte vorzugsweise im Modus "2 Soll- und 4 Istwerte im selben Zyklus" (siehe w_COMMAND_REG)
BAPS_PD_COMM8	Prozeßdatenkommunikation über die BAPS im Ωmega Drive-Line II für max. 8 Soll- und Istwerte
BAPS_PD_CONTROL	BAPS-Prozeßdatenkommunikations-Überwachung
BAPS_SD_CONTROL	BAPS-Bedarfsdatenkommunikation

6.2.1 BAPS_INIT

Beschreibung

Diesen Funktionsbaustein für BAPS können Sie verwenden, um die Prozeßdatenkommunikation (zyklische Kommunikation) zwischen V-Regler und Ωmega Drive-Line II über die BAPS-Schnittstelle zu initialisieren.

HINWEIS

Der FB BAPS_INIT verwendet die Bibliothek SYSTEM2_DLII_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
us_HW_TYPE	USINT 0	Ωmega Drive-Line II
i_EVENT	INT 0, 4, 8	Ereignis
w_COMMAND_REG	WORD	BAPS Steuerregister
us_MODE	USINT 0, 1, 2	Modus
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_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
t_ TIME	TIME	Überwachungszeit in ms
x_EN	BOOL	Freigabe Initialisierung
x_RESET	BOOL	Reset

Parameter Ausgang	Datentyp	Beschreibung
w_STATUS_REG	WORD	BAPS Statusregister
x_BUSY	BOOL	Busy-Bit
b_SL_QUIT	BYTE	V-Reglerquittung

Parameter Ausgang	Datentyp	Beschreibung
b_ERR	BYTE	Fehlerbyte
x_ERR	BOOL	Fehlerbit
x_OK	BOOL	OK-Bit

Der FB BAPS_INIT ermöglicht die Initialisierung und die Re-Initialisierung der BAPS für die zyklische Kommunikation.

Als FB für die zyklische Kommunikation stehen die FBs BAPS_PD_COMM8, BAPS_PD_COMM2 oder BAPS_PD_COMM24 (im folgenden auch BAPS_PD_COMMxx) zur Verfügung.

Die zyklische Kommunikation kann

für bis zu 2 Soll- und 2 Istwerte mit dem FB BAPS_PD_COMM2,

für bis zu 8 Soll- und 8 Istwerte mit dem FB BAPS_PD_COMM8,

für bis zu 2 Soll- und 4 Istwerte mit dem FB BAPS_PD_COMM24 [a)]

initialisiert werden.

Eingang us_HW_TYPE:

Am Eingang us_HW_TYPE wird mit us_HW_TYPE = 0 angegeben, daß der FB BAPS_INIT im Ωmega Drive-Line II eingesetzt wird (us_HW_TYPE ≠ 0 nicht implementiert).

HINWEIS

Wird der Eingang us_HW_TYPE **nicht** belegt, ergibt sich als Voreinstellung us_HW_TYPE = 0 (FB BAPS_INIT im Ωmega Drive-Line II).

Der FB der zyklischen Kommunikation BAPS_PD_COMMxx kann in jedem Zyklus im Hauptprogramm, einer Event-Task (auf ein beliebiges Ereignis) oder in einer Event-Task auf das Ereignis „BAPS-Prozeßdaten“ abgearbeitet werden. Für den letzten Fall wird das Ereignis "BAPS-Prozeßdaten" durch den FB BAPS_INIT initialisiert.

Das Ereignis "BAPS-Prozeßdaten" ist mit einem Interrupt-Level 13 (niedrige Priorität) oder 14 (hohe Priorität) initialisierbar.

Eingang i_EVENT:

Mit i_EVENT = 4 wird das Ereignis "BAPS-Prozeßdaten" mit Interrupt-Level 13 (niedrige Priorität) initialisiert.
Mit i_EVENT = 8 wird das Ereignis "BAPS-Prozeßdaten" mit Interrupt-Level 14 (hohe Priorität) initialisiert.

Wird der Eingang i_EVENT nicht belegt oder ist i_EVENT = 0, wird kein Ereignis initialisiert. Ist i_EVENT nicht 0, 4 oder 8 wird das Fehlerbit 3 am Ausgang b_ERR gesetzt.

[a)] ab V-Regler Softwareversion 000309

Eingänge u_WR_PAR_NR0 bis u_WR_PAR_NR7:

Die Sollwert-Parameternummern der zyklisch zu übertragenden Sollwerte werden an den Eingängen

u_WR_PAR_NR0 bis u_WR_PAR_NR7 (bei Nutzung von BAPS_PD_COMM8),

u_WR_PAR_NR0 und u_WR_PAR_NR1 (bei Nutzung von BAPS_PD_COMM2 oder BAPS_PD_COMM24)

angegeben.

Eingänge u_RD_PAR_NR0 bis u_RD_PAR_NR7:

Die Istwert-Parameternummern der zyklisch zu übertragenden Istwerte werden an den Eingängen

u_RD_PAR_NR0 **bis** u_RD_PAR_NR7 (bei Nutzung von BAPS_PD_COMM8),

u_RD_PAR_NR0 **und** u_RD_PAR_NR1 (bei Nutzung von BAPS_PD_COMM2),

u_RD_PAR_NR0 **bis** u_RD_PAR_NR3 (bei Nutzung von BAPS_PD_COMM24)

angegeben.

Eingang w_COMMAND_REG:

Am Eingang w_COMMAND_REG findet

- die Auswahl der Soll- und Istwertübertragung,
- die Auswahl der Berechnungsmethode der Kommunikations-Zykluszeit der Soll- und Istwertübertragung sowie die
- die Einstellung der Kommunikations-Zykluszeit der Soll- und Istwertübertragung

statt.

Bit-Nr.	Bedeutung		
0	Reserviert		
1, 2	Bit 2	Bit 1	Auswahl der Soll- und Istwertübertragung ($\rightarrow t_{cyc\ P}$)
	0	0	Zeitscheibenverfahren
	0	1	Reserviert
	1	0	Zwei Soll- und zwei (vier) Istwerte im selben Zyklus
	1	1	Reserviert
3	Bit3		Auswahl der Berechnungsmethode der Kommunikations-Zykluszeit der Soll- und Istwertübertragung ($\rightarrow t_{cyc\ K}$)
	0		Zähler
	1		Zeitscheibe
4, 5, 6, 7			Bit 3 = 0:
	1…15:		Wert des Zählers Alle 500µs wird intern ein Zähler inkrementiert. Stimmt der Stand dieses Counters mit dem Wert, der aus den Bits 4 bis 7 gebildet wird, überein, so wird eine Prozeßdaten-Kommunikation ausgeführt (sofern ein Ereignis vom V-Regler ansteht).
			Bit 3 = 1:
	1…15:		Nummer des Zeitschlitzes Prozeßdaten-Kommunikation findet jeweils 500µs nach dem Zeitschlitz statt, dessen Nummer in den Bits 4 bis 7 eingetragen ist (sofern ein Ereignis vom V-Regler ansteht).
8…15	Reserviert		

Zeitscheibe des V-Reglers für die zyklische Kommunikation: 500 µs

$t_{cyc\ K}$ - Zeit zwischen zwei Kommunikationen über die BAPS-Schnittstelle

$t_{cyc\ P}$ - Zeit mit der die Parameter (Soll- und Istwerte) an Position x aktualisiert werden

Ermittlung $t_{cyc\ K}$: → Bit 3 und 4 bis 7

Zähler: → Bit 3 = 0

$t_{cyc\ K}$ = 0,5 ms * "Wert aus Bit 4 bis 7"

Minimalwert:	0,5 ms * 1	= 0,5 ms
Beispiel:	0,5 ms * 2	= 1,0 ms
Beispiel:	0,5 ms * 3	= 1,5 ms
Beispiel:	0,5 ms * 4	= 2,0 ms
	usw.	
Maximalwert:	0,5 ms * 15	= 7,5 ms

Zeitscheibe: → Bit 3 = 1

$t_{cyc\ K}$ = 0,5 ms * $2^{\text{"Wert aus Bit 4...7"}}$

Minimalwert:	0,5 ms * 2^1	= 1 ms
Beispiel:	0,5 ms * 2^2	= 2 ms
Beispiel:	0,5 ms * 2^3	= 4 ms
Beispiel:	0,5 ms * 2^4	= 8 ms
Beispiel:	0,5 ms * 2^5	= 16 ms
	usw.	
Maximalwert:	0,5 ms * 2^{15}	= 16384 ms

HINWEIS

Nach jeder Zykluszeit $t_{cyc\ K}$ muß im Ωmega Drive-Line II der jeweilige Funktionsbaustein zur BAPS-Prozeßdatenkommunikation (FB BAPS_PD_COMMxx) aufgerufen werden (z. B. in einer Event-Task auf das Ereignis „BAPS-Prozeßdatenkommunikation oder in einer Event-Task auf das Ereignis „Sync-Signal Netzwerk (CANsync)“).

HINWEIS

Wenn die BAPS-Prozeßdatenkommunikation über ein Synchronisier-Signal getriggert wird oder die Betriebsart „Synchrone Lage-Sollwert-Vorgabe“ verwendet wird, muß im V-Regler der Parameter 167 (Sync.-Slot) auf $t_{cyc\ K}$ in µs eingestellt werden.

Ermittlung $t_{cyc\ P}$: Bit 1 und 2:

Zeitscheibenverfahren (bis 8 Sollwerte und 8 Istwerte):

$t_{cyc\ P\ x} = (2 * t_{cyc\ K}) * 2^x$; x: Parameternummer am FB: 0 bis 7

$t_{cyc\ P\ 0} = (2 * t_{cyc\ K}) * 2^0 =$	$2 * t_{cyc\ K}$	=	1 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 1} = (2 * t_{cyc\ K}) * 2^1 =$	$4 * t_{cyc\ K}$	=	2 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 2} = (2 * t_{cyc\ K}) * 2^2 =$	$8 * t_{cyc\ K}$	=	4 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 3} = (2 * t_{cyc\ K}) * 2^3 =$	$16 * t_{cyc\ K}$	=	8 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 4} = (2 * t_{cyc\ K}) * 2^4 =$	$32 * t_{cyc\ K}$	=	16 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 5} = (2 * t_{cyc\ K}) * 2^5 =$	$64 * t_{cyc\ K}$	=	32 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 6} = (2 * t_{cyc\ K}) * 2^6 =$	$128 * t_{cyc\ K}$	=	64 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 7} = (2 * t_{cyc\ K}) * 2^7 =$	$256 * t_{cyc\ K}$	=	128 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 0} = (2 * t_{cyc\ K}) * 2^0 =$	$2 * t_{cyc\ K}$	=	4 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 1} = (2 * t_{cyc\ K}) * 2^1 =$	$4 * t_{cyc\ K}$	=	8 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 2} = (2 * t_{cyc\ K}) * 2^2 =$	$8 * t_{cyc\ K}$	=	16 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 3} = (2 * t_{cyc\ K}) * 2^3 =$	$16 * t_{cyc\ K}$	=	32 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 4} = (2 * t_{cyc\ K}) * 2^4 =$	$32 * t_{cyc\ K}$	=	64 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 5} = (2 * t_{cyc\ K}) * 2^5 =$	$64 * t_{cyc\ K}$	=	128 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 6} = (2 * t_{cyc\ K}) * 2^6 =$	$128 * t_{cyc\ K}$	=	256 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 7} = (2 * t_{cyc\ K}) * 2^7 =$	$256 * t_{cyc\ K}$	=	512 ms	(bei $t_{cyc\ K}$ = 2 ms)

2 Sollwerte und 2 (4) Istwerte im selben Zyklus:

$t_{cyc\ P\ 0} = t_{cyc\ P\ 1}\ (= t_{cyc\ P\ 2} = t_{cyc\ P\ 3}) = t_{cyc\ K}$

Eingang us_MODE:

Mit us_MODE = 0 wird angegeben, daß der FB BAPS_INIT zur Prozeßdatenkonfigurierung einmalig aufgerufen wird. Das ist beim ersten Initialisieren der BAPS-Schnittstelle nach Reset, Warmstart oder Kaltstart notwendig.

Mit us_MODE = 1 wird angegeben, daß der FB BAPS_INIT zur Änderung der Prozeßdatenkonfigurierung bei gesperrter Prozeßdatenkommunikation eingesetzt wird.

Ab V-Regler-Softwareversion 000309 gilt zusätzlich:

Mit us_MODE = 2 können die Sollwert-Parameternummern der zyklisch zu übertragenden Sollwerte 0 und 1 sowie die Istwert-Parameternummern der zyklisch zu übertragenden Istwerte 0, 1, 2 und 3 überschrieben werden. Dies ist für die Re-Initialisierung bzw. Umkonfigurierung der BAPS bei laufender Prozeßdatenkommunikation vorgesehen.

HINWEIS

Eine Anfangsinitialisierung der BAPS ist mit us_MODE = 2 **nicht** möglich. Der FB BAPS_INIT gibt im us_MODE = 2 **keine** OK- oder Fehlermeldung aus!

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.

Eingang x_EN:

Mit x_EN = TRUE wird die Initialisierung bzw. die Prozeßdatenkonfigurierung der BAPS freigegeben. Voreinstellung ist x_EN = TRUE.

Eingang x_RESET:

Mit x_RESET = TRUE wird der FB BAPS_INIT zurückgesetzt.

Ausgang w_STATUS_REG:

Der Ausgang w_STATUS_REG zeigt mit dem gesetzten Bit 0 an, daß der V-Regler auf das Signal Trigger Controller synchronisiert ist (siehe “Die Interruptquellen und Triggersignale” auf Seite 56).

Ausgang x_BUSY:

Bei us_MODE = 1 zeigt der Ausgang x_BUSY mit TRUE an, daß die Initialisierung der Prozeßdatenkonfigurierung aktiv ist, bei us_MODE = 0 bzw. = 2 bleibt x_BUSY = FALSE.

Ausgang b_SL_QUIT:

Am Ausgang b_SL_QUIT wird die V-Reglerquittung nach Abschluß der Initialisierung gemeldet.

Wert b_SL_QUIT	**Bedeutung**
16#00	ohne Bedeutung
16#01	Sollwert gelesen / Istwert geschrieben
16#02	Konfiguration / Initialisierung korrekt durchgeführt
16#03 – 16#7F	reserviert
16#80	nicht interpretierbares Kommando empfangen
16#81	keine Konfiguration / Initialisierung durchgeführt
16#82	Istwert läßt sich nicht lesen
16#83	Sollwert läßt sich nicht schreiben
16#84 – 16#FE	reserviert
16#FF	ohne Bedeutung

Ausgang x_OK:

Der Ausgang x_OK ist auf TRUE gesetzt wenn die BAPS und ggf. das Ereignis BAPS-Prozeßdaten richtig initialisiert sind.

Ausgänge x_ERR, b_ERR:

Falls ein Fehler auftritt, wird das Fehlerbit x_ERR auf TRUE gesetzt und das Fehlerbyte b_ERR ausgegeben (gilt NICHT bei us_MODE = 2).

Fehlerbyte b_ERR:

Bit-nummer	**Fehler**
0	Ausgang b_SL_QUIT ≠ 16#02
1	reserviert
2	reserviert
3	Eingang i_EVENT nicht 4 oder 8 (oder 0 für kein Ereignis)
4	Timeout
5	Fehler beim Einrichten des Ereignis
6 – 7	reserviert

6.2.2 BAPS_PAR_READ

Beschreibung

Diesen Funktionsbaustein für BAPS können Sie verwenden, um einen Bedarfsdatenwert (Parameter) vom V-Regler über die BAPS-Schnittstelle und den FB BAPS_SD_CONTROL zu lesen.

HINWEIS

Der FB BAPS_PAR_READ verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BAPS_SD_DATA	BAPS_BMSTRUCT	Kommunikations-Daten
u_PAR_NR	UINT	Parameternummer
us_PAR_ELEMENT	USINT 7	Parameterelement
x_EN	BOOL	Freigabe
x_RESET	BOOL	Reset

Parameter Ausgang	Datentyp	Beschreibung
_BAPS_SD_DATA	BAPS_BMSTRUCT	Kommunikations-Daten
ud_PAR_VALUE	UDINT	gelesener Parameterwert
x_PAR_FORMAT	BOOL	Format des Parameterwertes
x_BUSY	BOOL	Kommunikation ist aktiv
i_ERR_DETAIL	INT	Betriebssystem-Fehler
i_ERR_COMM	INT	Kommunikations-Fehler
x_ERR	BOOL	Fehlerbit
x_OK	BOOL	OK-Bit

Der FB BAPS_PAR_READ übergibt mit den Werten der Eingänge u_PAR_NR und us_PAR_ELEMENT einen Parameter-Lesen-Auftrag an den FB BAPS_SD_CONTROL. Der FB BAPS_SD_CONTROL gibt den Parameter-Lesen-Auftrag an den V-Regler weiter und gibt die vom V-Regler zurückgegebenen Daten an den FB BAPS_PAR_READ zurück. Das vom V-Regler angeforderte Parameterelement wird am Ausgang ud_PAR_VALUE, das Format am Ausgang x_PAR_FORMAT angezeigt. Falls Fehler bei der Ausführung des Parameter-Lesen-Auftrags auftreten, werden sie an den Fehlerausgängen vom FB BAPS_PAR_READ angezeigt und näher spezifiziert.

Der FB BAPS_PAR_READ kann mehrfach eingesetzt (instanziiert) werden. Der FB BAPS_SD_CONTROL wird nur einmal eingesetzt und bearbeitet jeweils einen Parameter-Lesen-Auftrag (oder Parameter-Schreiben-Auftrag, siehe FB BAPS_PAR_WRITE).

Ein-/Ausgang _BAPS_SD_DATA:

An _BAPS_SD_DATA muß eine globale Variable vom Datentyp BAPS_BMSTRUCT angeschlossen werden.

Beispiel:

```
_BAPS_SD_DATEN : BAPS_BMSTRUCT;
```

dabei ist:

`_BAPS_SD_DATEN`	der Variablenname mit der Datentypkurzbezeichnung "_" für STRUCT
`BAPS_BMSTRUCT`	der Datentyp

Über diese Variable erfolgt der Datenaustausch mit dem FB BAPS_SD_CONTROL. Diese Variable wird bei den FBs der Bedarfsdatenkommunikation der BAPS angeschlossen - dies gilt auch, wenn die FBs BAPS_PAR_READ und/oder BAPS_PAR_WRITE mehrfach eingesetzt werden.

Eingang u_PAR_NR:

Am Eingang u_PAR_NR wird die Parameternummer des Parameters angegeben, dessen Wert gelesen werden soll.

Eingang us_PAR_ELEMENT:

Am Eingang us_PAR_ELEMENT wird das Element des zu lesenden Parameters angegeben. Wird us_PAR_ELEMENT nicht belegt, ergibt sich die Voreinstellung us_PAR_ELEMENT = 7 (≡ Wert des Parameters).

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 BAPS_PAR_READ und der FB BAPS_SD_CONTROL müssen dann jeweils mit x_RESET = TRUE zurückgesetzt werden.

Eingang x_RESET:

Mit x_RESET = TRUE wird der FB zurückgesetzt.

Ausgang ud_PAR_VALUE:

Der gelesene Parameter-Wert wird am Ausgang ud_PAR_VALUE ausgegeben.

Ausgang x_PAR_FORMAT:

Das Format des Parameter-Werts wird am Ausgang x_PAR_FORMAT zur Verfügung gestellt. x_PAR_FORMAT = FALSE bedeutet Format Wort, x_PAR_FORMAT = TRUE bedeutet Format Doppelwort.

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_DETAIL, i_ERR_COMM:

Falls ein Fehler auftritt, wird das Fehlerbit x_ERR auf TRUE gesetzt und der Fehler an den Ausgängen i_ERR_DETAIL und i_ERR_COMM spezifiziert.

Fehlernummer i_ERR_DETAIL:

i_ERR_DETAIL	**Fehler**
0	kein Fehler
-1	nicht näher spezifizierter Datenfehler
-2	Wert kleiner als Minimalwert
-3	Wert größer als Maximalwert
-4	Element darf nicht beschrieben werden
-5	kein Element vorhanden
-6	Element wegen Berechnung zur Zeit nicht verfügbar
-7	falsches Übergabe-Datenformat
-8	falsche Anzahl der Elemente beim Schreiben

Fehlernummer i_ERR_COMM:

i_ERR_COMM	**Fehler**
0	kein Fehler
-1	Kommunikationsfehler

6.2.3 BAPS_PAR_WRITE

Beschreibung

Diesen Funktionsbaustein für BAPS können Sie verwenden, um einen Bedarfsdatenwert (Parameter) vom V-Regler über die BAPS-Schnittstelle und den FB BAPS_SD_CONTROL zu schreiben.

HINWEIS

Der FB BAPS_PAR_WRITE verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BAPS_SD_DATA	BAPS_BMSTRUCT	Kommunikations-Daten
u_PAR_NR	UINT	Parameternummer
x_PAR_FORMAT	BOOL	Format des Parameterwertes
ud_PAR_VALUE	UDINT	Parameterwert
x_EN_ERR_FORMAT	BOOL	Anzeige Formatfehler ein/aus
x_EN	BOOL	Freigabe
x_RESET	BOOL	Reset

Parameter Ausgang	Datentyp	Beschreibung
_BAPS_SD_DATA	BAPS_BMSTRUCT	Kommunikations-Daten
x_BUSY	BOOL	Kommunikation ist aktiv
i_ERR_DETAIL	INT	Betriebssystem-Fehler
i_ERR_COMM	INT	Kommunikations-Fehler
x_ERR	BOOL	Fehlerbit
x_OK	BOOL	OK-Bit

Der FB BAPS_PAR_WRITE übergibt mit den Werten der Eingänge u_PAR_NR, x_PAR_FORMAT und ud_PAR_VALUE einen Parameter-Schreiben-Auftrag an den FB BAPS_SD_CONTROL. Der FB BAPS_SD_CONTROL gibt den Parameter-Schreiben-Auftrag an den V-Regler weiter und gibt das vom V-Regler zurückgegebene Ergebnis der Kommunikation an den FB BAPS_PAR_WRITE zurück. Falls Fehler bei der Ausführung des Parameter-Schreiben-Auftrags auftreten, werden sie an den Fehlerausgängen vom FB BAPS_PAR_WRITE angezeigt und näher spezifiziert.

Der FB BAPS_PAR_WRITE kann mehrfach eingesetzt (instanziiert) werden. Der FB BAPS_SD_CONTROL wird nur einmal eingesetzt und bearbeitet jeweils einen Parameter-Schreiben-Auftrag (oder Parameter-Lesen-Auftrag, siehe FB BAPS_PAR_READ).

Ein-/Ausgang _BAPS_SD_DATA:

An _BAPS_SD_DATA muß eine globale Variable vom Datentyp BAPS_BMSTRUCT angeschlossen werden.

Beispiel:

```
_BAPS_SD_DATEN : BAPS_BMSTRUCT;
```

dabei ist:

`_BAPS_SD_DATEN`	der Variablenname mit der Datentypkurzbezeichnung "_" für STRUCT
`BAPS_BMSTRUCT`	der Datentyp

Über diese Variable erfolgt der Datenaustausch mit dem FB BAPS_SD_CONTROL. Diese Variable wird bei den FBs der Bedarfsdatenkommunikation der BAPS angeschlossen - dies gilt auch, wenn die FBs BAPS_PAR_READ und/oder BAPS_PAR_WRITE mehrfach eingesetzt werden.

Eingang u_PAR_NR:

Am Eingang u_PAR_NR wird die Parameter-Nummer des Parameters angegeben, dessen Wert geschrieben werden soll.

Eingang x_PAR_FORMAT:

Am Eingang x_PAR_FORMAT wird das Format des zu schreibenden Wertes angegeben. x_PAR_FORMAT = FALSE bedeutet Format Wort, x_PAR_FORMAT = TRUE bedeutet Format Doppelwort.

Eingang ud_PAR_VALUE:

Der zu schreibende Parameterwert wird am Eingang ud_PAR_VALUE angegeben.

Eingang x_EN_ERR_FORMAT:

Über den Eingang x_EN_ERR_FORMAT kann die Anzeige eines Formatfehlers i_ERR_DETAIL = -7 im Fehlerbit x_ERR eingestellt werden. Wenn die Anzeige nicht stattfinden soll, wird x_EN_ERR_FORMAT auf FALSE gesetzt. Wird der Eingang x_EN_ERR_FORMAT nicht belegt, ergibt sich eine Voreinstellung von x_EN_ERR_FORMAT = TRUE und i_ERR_DETAIL = -7 wird in x_ERR angezeigt.

HINWEIS

Ist x_EN_ERR_FORMAT = FALSE wird ein Formatfehler in i_ERR_DETAIL (= -7) angezeigt, jedoch wird der Formatfehler nicht im Fehlerbit x_ERR angezeigt!
Das OK-Bit ist in diesem Fall (trotz des Formatfehlers) auf TRUE gesetzt!

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 BAPS_PAR_WRITE und der FB BAPS_SD_CONTROL müssen 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_DETAIL, i_ERR_COMM:

Falls ein Fehler auftritt, wird das Fehlerbit x_ERR auf TRUE gesetzt und der Fehler an den Ausgängen i_ERR_DETAIL und i_ERR_COMM spezifiziert.

Fehlernummer i_ERR_DETAIL:

i_ERR_DETAIL	**Fehler**
0	kein Fehler
-1	nicht näher spezifizierter Datenfehler
-2	Wert kleiner als Minimalwert
-3	Wert größer als Maximalwert
-4	Element darf nicht beschrieben werden
-5	kein Element vorhanden
-6	Element wegen Berechnung zur Zeit nicht verfügbar!
-7	falsches Übergabe-Datenformat
-8	falsche Anzahl der Elemente beim Schreiben

Fehlernummer i_ERR_COMM:

i_ERR_COMM	**Fehler**
0	kein Fehler
-1	Kommunikationsfehler

6.2.4 BAPS_PD_COMM2

Beschreibung

Diesen Funktionsbaustein für BAPS können Sie verwenden, um die Prozeßdatenkommunikation zwischen V-Regler und Ωmega Drive-Line II über die BAPS-Schnittstelle durchzuführen.

Parameter Eingang	Datentyp	Beschreibung
us_HW_TYPE	USINT 0	Ωmega Drive-Line II
w_CONTROLWORD	WORD	Steuerwort
w_COMMAND_REG	WORD	Steuerregister
us_MODE	USINT 0, 1	Mode
ud_WR_VALUE0	UDINT	Sollwert 0
ud_WR_VALUE1	UDINT	Sollwert 1
x_EN	BOOL	Freigabe

Parameter Ausgang	Datentyp	Beschreibung
w_STATUSWORD	WORD	Statuswort
w_STATUS_REG	WORD	Statusregister
ud_RD_VALUE0	UDINT	Istwert 0
ud_RD_VALUE1	UDINT	Istwert 1
b_SL_QUIT	BYTE	Reglerquittung
b_ERR	BYTE	Fehlerbyte
x_ERR	BOOL	Fehlerbit

Die Sollwerte und das Steuerwort werden an den V-Regler gesendet, die Istwerte und das Statuswort werden vom V-Regler empfangen und ausgegeben.

Eingang us_HW_TYPE:

Am Eingang us_HW_TYPE wird mit us_HW_TYPE = 0 angegeben, daß der FB BAPS_PD_COMM2 im Ωmega Drive-Line II eingesetzt wird (us_HW_TYPE ≠ 0 nicht implementiert).

HINWEIS

Wird der Eingang us_HW_TYPE **nicht** belegt, ergibt sich als Voreinstellung us_HW_TYPE = 0 (FB BAPS_PD_COMM2 im Ωmega Drive-Line II).

Eingang w_CONTROLWORD:

Am Eingang w_CONTROLWORD wird das Steuerwort angegeben, das an den V-Regler gesendet werden soll.

Eingang w_COMMAND_REG:

Am Eingang w_COMMAND_REG findet

- die Auswahl der Soll- und Istwertübertragung,
- die Auswahl der Berechnungsmethode der Kommunikations-Zykluszeit der Soll- und Istwertübertragung sowie die
- die Einstellung der Kommunikations-Zykluszeit der Soll- und Istwertübertragung

statt.

Bit-Nr.	Bedeutung		
0	Reserviert		
1, 2	Bit 2	Bit 1	Auswahl der Soll- und Istwertübertragung (→ $t_{cyc\ P}$)
	0	0	Zeitscheibenverfahren
	0	1	Reserviert
	1	0	Zwei Soll- und zwei Istwerte im selben Zyklus
	1	1	Reserviert
3	Bit3		Auswahl der Berechnungsmethode der Kommunikations-Zykluszeit der Soll- und Istwertübertragung (→ $t_{cyc\ K}$)
	0		Zähler
	1		Zeitscheibe
4, 5, 6 ,7	1…15:		Bit 3 = 0 : Wert des Zählers Alle 500µs wird intern ein Zähler inkrementiert. Stimmt der Stand dieses Zählers mit dem Wert, der aus den Bits 4 bis 7 gebildet wird, überein, so wird eine Prozeßdaten-Kommunikation ausgeführt (sofern ein Ereignis vom V-Regler ansteht). Bit 3 = 1: Nummer des Zeitschlitzes Prozeßdaten-Kommunikation findet jeweils 500µs nach dem Zeitschlitz statt, dessen Nummer in den Bits 4 bis 7 eingetragen ist (sofern ein Ereignis vom V-Regler ansteht).
	1…15:		
8…15	Reserviert		

Zeitscheibe des V-Reglers für die zyklische Kommunikation: 500 µs

$t_{cyc\ K}$ - Zeit zwischen zwei Kommunikationen über die BAPS

$t_{cyc\ P}$ - Zeit mit der die Parameter (Soll- und Istwerte) an Position x (Eingänge ud_WR_VALUEx und Ausgänge ud_RD_VALUEx) aktualisiert werden

Ermittlung $t_{cyc\ K}$: → Bit 3 und 4 bis 7

Zähler: → Bit 3 = 0

$t_{cyc\ K}$ = 0,5 ms * “Wert aus Bit 4 bis 7”

Minimalwert:	0,5 ms * 1	= 0,5 ms
Beispiel:	0,5 ms * 2	= 1,0 ms
Beispiel:	0,5 ms * 3	= 1,5 ms
Beispiel:	0,5 ms * 4	= 2,0 ms
	usw.	
Maximalwert:	0,5 ms * 15	= 7,5 ms

Zeitscheibe: → Bit 3 = 1

$t_{cyc\,K}$ = 0,5 ms * 2^"Wert aus Bit 4 bis 7"

Minimalwert:	0,5 ms * 2^1	=	1 ms
Beispiel:	0,5 ms * 2^2	=	2 ms
Beispiel:	0,5 ms * 2^3	=	4 ms
Beispiel:	0,5 ms * 2^4	=	8 ms
Beispiel:	0,5 ms * 2^5	=	16 ms
	usw.		
Maximalwert:	0,5 ms * 2^{15}	=	16384 ms

HINWEIS

Nach jeder Zykluszeit $t_{cyc\,K}$ muß im Ωmega Drive-Line II der FB BAPS_PD_COMM2 aufgerufen werden (z.B. in einer Event-Task auf das Ereignis "BAPS-Prozeßdaten" oder in einer Event-Task auf das Ereignis "Sync-Signal Netzwerk (CANsync)").

HINWEIS

Wenn die (BAPS-) Prozeßdatenkommunikation über ein Synchronisier-Signal getriggert wird oder die Betriebsart „Synchrone Lage-Sollwert-Vorgabe“ verwendet wird, muß im V-Regler der Parameter 167 (Sync.-Slot) auf $t_{cyc\,K}$ in µs eingestellt werden.

Ermittlung $t_{cyc\,P}$: Bit 1 und 2:

Zeitscheibenverfahren (bis 2 Sollwerte und 2 Istwerte):

$t_{cyc\,P\,x} = (2 * t_{cyc\,K}) * 2^x$; x: Soll-/Istwertnummer am FB: 0 bis 1 (Eingänge ud_WR_VALUEx und Ausgänge ud_RD_VALUEx)

$t_{cyc\,P\,0} = (2 * t_{cyc\,K}) * 2^0 =$	$2 * t_{cyc\,K}$	=	1 ms	(bei $t_{cyc\,K}$ = 0,5 ms)
$t_{cyc\,P\,1} = (2 * t_{cyc\,K}) * 2^1 =$	$4 * t_{cyc\,K}$	=	2 ms	(bei $t_{cyc\,K}$ = 0,5 ms)
$t_{cyc\,P\,0} = (2 * t_{cyc\,K}) * 2^0 =$	$2 * t_{cyc\,K}$	=	4 ms	(bei $t_{cyc\,K}$ = 2 ms)
$t_{cyc\,P\,1} = (2 * t_{cyc\,K}) * 2^1 =$	$4 * t_{cyc\,K}$	=	8 ms	(bei $t_{cyc\,K}$ = 2 ms)

2 Sollwerte und 2 Istwerte im selben Zyklus:

$t_{cyc\,P\,0} = t_{cyc\,P\,1} = t_{cyc\,K}$

Eingang us_MODE:

Mit us_MODE = 0 wird angegeben, daß der FB BAPS_PD_COMM2 im zyklischen Hauptprogramm oder in einer Event-Task auf ein beliebiges Ereignis (z.B. auf das Ereignis "Sync-Signal Netzwerk (CANsync)") aufgerufen wird.

Mit us_MODE = 1 wird angegeben, daß der FB BAPS_PD_COMM2 in einer Event-Task auf das Ereignis "BAPS-Prozeßdaten" aufgerufen wird (siehe Beschreibung FB BAPS_INIT, Eingang i_EVENT).

Wird us_MODE nicht belegt ergibt sich die Voreinstellung us_MODE = 0 (Einsatz des FB BAPS_PD_COMM2 im zyklischen Hauptprogramm oder in einer Event-Task auf ein beliebiges Ereignis).

Eingänge ud_WR_VALUE0, ud_WR_VALUE1:

An den Eingängen ud_WR_VALUE0 und ud_WR_VALUE1 werden die Sollwerte angeschlossen, deren Parameternummern am FB BAPS_INIT (Eingänge u_WR_PAR_NR0 und u_WR_PAR_NR1) in der Initialisierung der BAPS-Prozeßdatenkommunikation angegeben wurden.

Eingang x_EN:

Mit x_EN = TRUE wird die Kommunikation freigegeben. Voreinstellung ist x_EN = TRUE, d.h. die Kommunikation ist freigegeben.

Ausgang w_STATUSWORD:

Am Ausgang w_STATUSWORD wird das Statuswort des V-Reglers ausgegeben.

Ausgang w_STATUS_REG:

Der Ausgang w_STATUS_REG (BAPS Status-Register) zeigt mit dem gesetzten Bit 0 an, daß der V-Regler auf das Signal Trigger Controller synchronisiert ist (siehe “Die Interruptquellen und Triggersignale” auf Seite 56).

Ausgänge ud_RD_VALUE0, ud_RD_VALUE1:

An den Ausgängen ud_RD_VALUE0 und ud_RD_VALUE1 werden die Istwerte ausgegeben, deren Parameternummern am FB BAPS_INIT (Eingänge u_RD_PAR_NR0 und u_RD_PAR_NR1) in der Initialisierung der BAPS-Prozeßdatenkommunikation angegeben wurden.

Ausgang b_SL_QUIT:

Am Ausgang b_SL_QUIT wird die V-Reglerquittung nach Abschluß der Kommunikation gemeldet.

Wert b_SL_QUIT	Bedeutung
16#00	ohne Bedeutung
16#01	Sollwert gelesen / Istwert geschrieben
16#02	Konfiguration / Initialisierung korrekt durchgeführt
16#03 – 16#7F	reserviert
16#80	nicht interpretierbares Kommando empfangen
16#81	keine Konfiguration / Initialisierung durchgeführt
16#82	Istwert lässt sich nicht lesen
16#83	Sollwert lässt sich nicht schreiben
16#84 – 16#FE	reserviert
16#FF	ohne Bedeutung

Ausgänge x_ERR, b_ERR:

Falls ein Fehler auftritt, wird das Fehlerbit x_ERR auf TRUE gesetzt und das Fehlerbyte b_ERR ausgegeben.

Fehlerbyte b_ERR:

Bit-nummer	Fehler
0	Ausgang b_SL_QUIT ≠ 16#01
1 – 7	reserviert

6.2.5 BAPS_PD_COMM24

Beschreibung

Diesen Funktionsbaustein für BAPS können Sie verwenden, um die Prozeßdatenkommunikation zwischen V-Regler und Ωmega Drive-Line II über die BAPS-Schnittstelle durchzuführen. [a)]

Parameter Eingang	Datentyp	Beschreibung
us_HW_TYPE	USINT 0	Ωmega Drive-Line II
w_CONTROLWORD	WORD	Steuerwort
w_COMMAND_REG	WORD	Steuerregister
us_MODE	USINT 0, 1	Mode
ud_WR_VALUE0	UDINT	Sollwert 0
ud_WR_VALUE1	UDINT	Sollwert 1
t_RE_INIT_TIME	TIME	Überwachungszeit in ms für Re-Initialisierung
x_RE_INIT_START	BOOL	Start (-flanke) für Re-Initialisierung
x_EN	BOOL	Freigabe

Parameter Eingang	Datentyp	Beschreibung
w_STATUSWORD	WORD	Statuswort
w_STATUS_REG	WORD	Statusregister
ud_RD_VALUE0	UDINT	Istwert 0
ud_RD_VALUE1	UDINT	Istwert 1
ud_RD_VALUE2	UDINT	Istwert 2
ud_RD_VALUE3	UDINT	Istwert 3
b_SL_QUIT	BYTE	Reglerquittung
b_ERR	BYTE	Fehlerbyte
x_ERR	BOOL	Fehlerbit
x_RE_INIT_ERR	BOOL	Fehlerbit Re-Initialisierung
x_RE_INIT_OK	BOOL	OK-Bit Re-Initialisierung

Die Sollwerte und das Steuerwort werden an den V-Regler gesendet, die Istwerte und das Statuswort werden vom V-Regler empfangen und ausgegeben.

Eingang us_HW_TYPE:

Am Eingang us_HW_TYPE wird mit us_HW_TYPE = 0 angegeben, daß der FB BAPS_PD_COMM24 im Ωmega Drive-Line II eingesetzt wird (us_HW_TYPE ≠ 0 nicht implementiert).

[a)] ab V-Regler Softwareversion 000309

HINWEIS

Wird der Eingang us_HW_TYPE **nicht** belegt, ergibt sich als Voreinstellung us_HW_TYPE = 0 (FB BAPS_PD_COMM24 im Ωmega Drive-Line II).

Eingang w_CONTROLWORD:

Am Eingang w_CONTROLWORD wird das Steuerwort angegeben, das an den V-Regler gesendet werden soll.

Eingang w_COMMAND_REG:

Am Eingang w_COMMAND_REG findet

- die Auswahl der Soll- und Istwertübertragung,
- die Auswahl der Berechnungsmethode der Kommunikations-Zykluszeit der Soll- und Istwertübertragung sowie die
- die Einstellung der Kommunikations-Zykluszeit der Soll- und Istwertübertragung

statt.

Bit-Nr.	**Bedeutung**		
0	Reserviert		
1, 2	Bit 2	Bit 1	Auswahl der Soll- und Istwertübertragung ($\rightarrow t_{cyc\ P}$)
	0	0	Zeitscheibenverfahren
	0	1	Reserviert
	1	0	Zwei Soll- und zwei (vier) Istwerte im selben Zyklus
	1	1	Reserviert
3	Bit3		Auswahl der Berechnungsmethode der Kommunikations-Zykluszeit der Soll- und Istwertübertragung ($\rightarrow t_{cyc\ K}$)
	0		Zähler
	1		Zeitscheibe
4, 5, 6 ,7			Bit 3 = 0:
	1…15:		Wert des Zählers Alle 500µs wird intern ein Zähler inkrementiert. Stimmt der Stand dieses Zählers mit dem Wert, der aus den Bits 4 bis 7 gebildet wird, überein, so wird eine Prozeßdaten-Kommunikation ausgeführt (sofern ein Ereignis vom V-Regler ansteht).
			Bit 3 = 1:
	1…15:		Nummer des Zeitschlitzes Prozeßdaten-Kommunikation findet jeweils 500µs nach dem Zeitschlitz statt, dessen Nummer in den Bits 4 bis 7 eingetragen ist (sofern ein Ereignis vom V-Regler ansteht).
8…15	Reserviert		

Zeitscheibe des V-Reglers für die zyklische Kommunikation: 500 µs

$t_{cyc\ K}$ - Zeit zwischen zwei Kommunikationen über die BAPS

$t_{cyc\ P}$ - Zeit mit der die Parameter (Soll- und Istwerte) an Position x (Eingänge ud_WR_VALUEx und Ausgänge ud_RD_VALUEx) aktualisiert werden

Ermittlung $t_{cyc\ K}$: → Bit 3 und 4 bis 7

Zähler: → Bit 3 = 0

$t_{cyc\ K}$ = 0,5 ms * "Wert aus Bit 4 bis 7"

Minimalwert:	0,5 ms * 1	= 0,5 ms
Beispiel:	0,5 ms * 2	= 1,0 ms
Beispiel:	0,5 ms * 3	= 1,5 ms
Beispiel:	0,5 ms * 4	= 2,0 ms
	usw.	
Maximalwert:	0,5 ms * 15	= 7,5 ms

Zeitscheibe: → Bit 3 = 1

$t_{cyc\ K}$ = 0,5 ms * $2^{\text{"Wert aus Bit 4 bis 7"}}$

Minimalwert:	0,5 ms * 2^1	= 1 ms
Beispiel:	0,5 ms * 2^2	= 2 ms
Beispiel:	0,5 ms * 2^3	= 4 ms
Beispiel:	0,5 ms * 2^4	= 8 ms
Beispiel:	0,5 ms * 2^5	= 16 ms
	usw.	
Maximalwert:	0,5 ms * 2^{15}	= 16384 ms

HINWEIS

Nach jeder Zykluszeit $t_{cyc\ K}$ muß im Ωmega Drive-Line II der FB BAPS_PD_COMM24 aufgerufen werden (z.B. in einer Event-Task auf das Ereignis "BAPS-Prozeßdaten" oder in einer Event-Task auf das Ereignis "Sync-Signal Netzwerk (CANsync)").

HINWEIS

Wenn die (BAPS-) Prozeßdatenkommunikation über ein Synchronisier-Signal getriggert wird oder die Betriebsart „Synchrone Lage-Sollwert-Vorgabe“ verwendet wird muß im V-Regler der Parameter 167 (Sync.-Slot) auf $t_{cyc\ K}$ in µs eingestellt werden.

Ermittlung $t_{cyc\ P}$: Bit 1 und 2:

Zeitscheibenverfahren (bis 2 Sollwerte und 4 Istwerte):

$t_{cyc\ P\ x} = (2 * t_{cyc\ K}) * 2^{x}$; x: Soll-/Istwertnummer am FB: 0 bis 1 bzw. 3 (Eingänge ud_WR_VALUEx und Ausgänge ud_RD_VALUEx)

$t_{cyc\ P\ 0} = (2 * t_{cyc\ K}) * 2^{0} =$	$2 * t_{cyc\ K}$	=	1 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 1} = (2 * t_{cyc\ K}) * 2^{1} =$	$4 * t_{cyc\ K}$	=	2 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 2} = (2 * t_{cyc\ K}) * 2^{2} =$	$8 * t_{cyc\ K}$	=	4 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 3} = (2 * t_{cyc\ K}) * 2^{3} =$	$16 * t_{cyc\ K}$	=	8 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 0} = (2 * t_{cyc\ K}) * 2^{0} =$	$2 * t_{cyc\ K}$	=	4 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 1} = (2 * t_{cyc\ K}) * 2^{1} =$	$4 * t_{cyc\ K}$	=	8 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 2} = (2 * t_{cyc\ K}) * 2^{2} =$	$8 * t_{cyc\ K}$	=	16 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 3} = (2 * t_{cyc\ K}) * 2^{3} =$	$16 * t_{cyc\ K}$	=	32 ms	(bei $t_{cyc\ K}$ = 2 ms)

2 Sollwerte und 4 Istwerte im selben Zyklus:

$t_{cyc\ P\ 0} = t_{cyc\ P\ 1} = t_{cyc\ P\ 2} = t_{cyc\ P\ 3} = t_{cyc\ K}$

Eingang us_MODE:

Mit us_MODE = 0 wird angegeben, daß der FB BAPS_PD_COMM24 im zyklischen Hauptprogramm oder in einer Event-Task auf ein beliebiges Ereignis (z.B. auf das Ereignis "Sync-Signal Netzwerk (CANsync)") aufgerufen wird.

Mit us_MODE = 1 wird angegeben, daß der FB BAPS_PD_COMM24 in einer Event-Task auf das Ereignis "BAPS-Prozeßdaten" aufgerufen wird (siehe Beschreibung FB BAPS_INIT, Eingang i_EVENT).

Wird us_MODE nicht belegt ergibt sich die Voreinstellung us_MODE = 0 (Einsatz des FB BAPS_PD_COMM24 im zyklischen Hauptprogramm oder in einer Event-Task auf ein beliebiges Ereignis).

Eingänge ud_WR_VALUE0, ud_WR_VALUE1:

An den Eingängen ud_WR_VALUE0 und ud_WR_VALUE1 werden die Sollwerte angeschlossen, deren Parameternummern am FB BAPS_INIT (Eingänge u_WR_PAR_NR0 und u_WR_PAR_NR1) in der Initialisierung der BAPS-Prozeßdatenkommunikation angegeben wurden. [a)]

Eingang x_EN:

Mit x_EN = TRUE wird die Kommunikation freigegeben. Voreinstellung ist x_EN = TRUE, d. h. die Kommunikation ist freigegeben.

Eingänge t_RE_INIT_TIME und x_RE_INIT_START siehe weiter unten.

---

a) Ab V-Regler Softwareversion 000309 ist die Funktion 2 Sollwerte und 4 Istwerte im selben Zyklus implementiert.

Ausgang w_STATUSWORD:

Am Ausgang w_STATUSWORD wird das Statuswort des V-Reglers ausgegeben.

Ausgang w_STATUS_REG:

Der Ausgang w_STATUS_REG (BAPS Status-Register) zeigt mit dem gesetzten Bit 0 an, daß der V-Regler auf das Signal Trigger Controller synchronisiert ist (siehe “Die Interruptquellen und Triggersignale” auf Seite 56).

Ausgänge ud_RD_VALUE0 bis ud_RD_VALUE3:

An den Ausgängen ud_RD_VALUE0 bis ud_RD_VALUE3 werden die Istwerte ausgegeben, deren Parameternummern am FB BAPS_INIT (Eingänge u_RD_PAR_NR0 bis u_RD_PAR_NR3) in der Initialisierung der BAPS-Prozeßdatenkommunikation angegeben wurden. [a)]

Ausgang b_SL_QUIT:

Am Ausgang b_SL_QUIT wird die V-Reglerquittung nach Abschluß der Kommunikation gemeldet.

Wert b_SL_QUIT	**Bedeutung**
16#00	ohne Bedeutung
16#01	Sollwert gelesen / Istwert geschrieben
16#02	Konfiguration / Initialisierung korrekt durchgeführt
16#03	Re-Initialisierung aktiv
16#04 – 16#7F	reserviert
16#80	nicht interpretierbares Kommando empfangen
16#81	keine Konfiguration / Initialisierung durchgeführt
16#82	Istwert lässt sich nicht lesen
16#83	Sollwert lässt sich nicht schreiben
16#84	reserviert
16#85	reserviert
16#86	Sollwert lässt sich nicht schreiben (Re-Initialisierung)
16#87	Istwert lässt sich nicht lesen (Re-Initialisierung)
16#88 – 16#FE	reserviert
16#FF	ohne Bedeutung

Ausgänge x_RE_INIT_ERR und x_RE_INIT_OK siehe weiter unten.

Ausgänge x_ERR, b_ERR:

Falls ein Fehler auftritt, wird das Fehlerbit x_ERR auf TRUE gesetzt und das Fehlerbyte b_ERR ausgegeben.

[a)] Ab V-Regler Softwareversion 000309 ist die Funktion 2 Sollwerte und 4 Istwerte im selben Zyklus implementiert.

Fehlerbyte b_ERR:

Bit-nummer	Fehler
0	Ausgang b_SL_QUIT ≠ 16#01
1	Timeout (Re-Initialisierung)
2 – 7	reserviert

Ab V-Regler Softwareversion 000309 ist die Re-Initialisierung bzw. Umparametrierung der BAPS-Prozeßdatenkonfiguration bei laufender Prozeßdatenkommunikation möglich. Hierzu muß der FB BAPS_INIT in einer zyklischen Task im us_MODE = 2 durchlaufen werden. Näheres hierzu in der Beschreibung des FB BAPS_INIT und weiter unten.

Eingang t_RE_INIT_TIME:

Am Eingang t_RE_INIT_TIME wird die Überwachungszeit für die Re-Initialisierung angegeben. Voreinstellung ist t_RE_INIT_TIME = 1 s. Wird innerhalb dieser Zeit die Re-Initialisierung nicht abgeschlossen, wird im Fehlerbyte b_ERR das Bit 1 gesetzt.

Eingang x_RE_INIT_START:

Am Eingang x_RE_INIT_START wird mit x_RE_INIT_START = TRUE die Re-Initialisierung gestartet.

Ein Timeout bei der Re-Initialisierung wird durch b_ERR = 16#02 angegeben und durch das gesetzte Fehlerbit x_ERR signalisiert. Wird bei der Re-Initialisierung eine ungültige Parameternummer angegeben, wird **kein** Bit im Fehlerbyte b_ERR gesetzt und das Fehlerbit x_ERR bleibt FALSE. Die Angabe einer ungültigen Parameternummer wird durch b_SL_QUIT = 16#86 bzw. b_SL_QUIT = 16#87 und x_RE_INIT_ERR = TRUE gemeldet.

Funktion der Re-Initialisierung:

Ab V-Regler Softwareversion 000309 können bei laufender Prozeßdatenkommunikation (siehe Eingang w_COMMAND_REG, Auswahl der Soll- und Istwertübertragung = 2 Soll- und 4 Istwerte im selben Zyklus) die Parameternummern der Soll- und/oder Istwerte geändert (→ re-initialisiert) werden.

Innerhalb der Re-Initialisierungphase sind die Soll- und Istwerte, die re-initialisiert werden, **nicht** definiert. Die Soll- und Istwerte, die nicht re-initialisiert werden bleiben gültig.

Ablauf der Re-Initialisierung:

Für die Re-Initialisierung der BAPS-Prozeßdatenkommunikation muß dem V-Regler mitgeteilt werden, welche Parameternummern wie zu ändern sind. Hierzu wird der FB BAPS_INIT im us_MODE = 2 aufgerufen.

HINWEIS

Eine Instanz des FB BAPS_INIT steht hierzu in einer POU, die zyklisch aufgerufen wird und nicht in der Event-Task, in der der FB BAPS_PD_COMM24 aufgerufen wird.

FB BAPS_INIT, Eingänge u_WR_PAR_NR0, u_WR_PAR_NR1:

An den Eingängen u_WR_PAR_NR0 und u_WR_PAR_NR1 des FB BAPS_INIT werden die neuen Parameternummern für Sollwert 0 und Sollwert 1 angegeben.

FB BAPS_INIT, Eingänge u_RD_PAR_NR0 bis u_RD_PAR_NR3:

An den Eingängen u_RD_PAR_NR0 bis u_RD_PAR_NR3 des FB BAPS_INIT werden die neuen Parameternummern für Istwert 0 bis Istwert 3 angegeben.

Sollen Parameternummern nicht geändert werden, gibt man die bisherige Parameternummer an.

Wird einer der Eingänge (u_WR_PAR_NR0, u_WR_PAR_NR1, u_RD_PAR_NR0, u_RD_PAR_NR1, u_RD_PAR_NR2, u_RD_PAR_NR3) nicht belegt, muß als Parameternummer "0" eingetragen werden.

FB BAPS_INIT, Eingang x_EN:

Sind die Eingänge entsprechend beschaltet, wird mit x_EN = TRUE (des FB BAPS_INIT) das Eintragen der Parameternummern in die entsprechenden Register der BAPS-Schnittstelle gestartet. Der FB BAPS_INIT muß hierzu nur einmal durchlaufen werden.

HINWEIS

Der FB BAPS_INIT gibt im us_MODE = 2 **keine** OK- oder Fehlermeldungen aus!

Eingang x_RE_INIT_TRUE, Ausgang x_RE_INIT_OK:

Nachdem der FB BAPS_INIT durchlaufen wurde, wird die Re-Initialisierung am FB BAPS_PD_COMM24 mit x_RE_INIT_START = TRUE gestartet. Ab jetzt sind die Werte der zu re-initialisierenden Soll- und Istwertparameter nicht definiert, bis das Ende der Re-Initialisierungsphase mit x_RE_INIT_OK = TRUE (oder x_RE_INIT_ERR = TRUE) signalisiert wird.

Im Ok-Fall (x_RE_INIT_OK = TRUE) werden jetzt die Werte der Sollwerte 0 und 1 an die re-initialisierten Sollwertparameter im V-Regler übertragen und die Werte der Istwerte 0 bis 3 von den re-initialisierten Istwertparameternummern im V-Regler "geholt". Alle Werte sind wieder definiert.

Ausgang x_RE_INIT_ERR, b_ERR, b_SL_QUIT:

Im Fehler-Fall wird nach der Zeit x_RE_INIT_TIME mit dem gesetzten Bit 1 im Fehlerbyte b_ERR (des FB BAPS_PD_COMM24) und x_ERR = TRUE ein Timeout signalisiert.

Bei einer (oder zwei) ungültigen Sollwertparameternummer(n) wird am Ausgang b_SL_QUIT der Wert 16#86 ausgegeben und x_RE_INIT_ERR = TRUE gesetzt.

Bei einer (oder mehreren) ungültigen Istwertparameternummer(n) wird am Ausgang b_SL_QUIT der Wert 16#87 ausgegeben und x_RE_INIT_ERR = TRUE gesetzt.

Bei einer (oder zwei) ungültigen Sollwertparameternummer(n) **und** einer (oder mehreren) ungültigen Istwertparameternummer(n) wird am Ausgang b_SL_QUIT der Wert 16#86 ausgegeben und x_RE_INIT_ERR = TRUE gesetzt.

6.2.6 BAPS_PD_COMM8

Beschreibung

Diesen Funktionsbaustein für BAPS können Sie verwenden, um die Prozeßdatenkommunikation zwischen V-Regler und Ωmega Drive-Line II über die BAPS-Schnittstelle durchzuführen.

Parameter Eingang	Datentyp	Beschreibung
us_HW_TYPE	USINT 0	Ωmega Drive-Line II
w_CONTROLWORD	WORD	Steuerwort
w_COMMAND_REG	WORD	Steuerregister
us_MODE	USINT 0, 1	Mode
ud_WR_VALUE0	UDINT	Sollwert 0
ud_WR_VALUE1	UDINT	Sollwert 1
ud_WR_VALUE2	UDINT	Sollwert 2
ud_WR_VALUE3	UDINT	Sollwert 3
ud_WR_VALUE4	UDINT	Sollwert 4
ud_WR_VALUE5	UDINT	Sollwert 5
ud_WR_VALUE6	UDINT	Sollwert 6
ud_WR_VALUE7	UDINT	Sollwert 7
x_EN	BOOL	Freigabe

Parameter Ausgang	Datentyp	Beschreibung
w_STATUSWORD	WORD	Statuswort
w_STATUS_REG	WORD	Statusregister
ud_RD_VALUE0	UDINT	Istwert 0
ud_RD_VALUE1	UDINT	Istwert 1
ud_RD_VALUE2	UDINT	Istwert 2
ud_RD_VALUE3	UDINT	Istwert 3
ud_RD_VALUE4	UDINT	Istwert 4
ud_RD_VALUE5	UDINT	Istwert 5
ud_RD_VALUE6	UDINT	Istwert 6
ud_RD_VALUE7	UDINT	Istwert 7
b_SL_QUIT	BYTE	Reglerquittung
b_ERR	BYTE	Fehlerbyte
x_ERR	BOOL	Fehlerbit

Die Sollwerte und das Steuerwort werden an den V-Regler gesendet, die Istwerte und das Statuswort werden vom V-Regler empfangen und ausgegeben.

Eingang us_HW_TYPE:

Am Eingang us_HW_TYPE wird mit us_HW_TYPE = 0 angegeben, daß der FB BAPS_PD_COMM8 im Ωmega Drive-Line II eingesetzt wird (us_HW_TYPE $\neq$ 0 nicht implementiert).

HINWEIS

Wird der Eingang us_HW_TYPE **nicht** belegt, ergibt sich als Voreinstellung us_HW_TYPE = 0 (FB BAPS_PD_COMM8 im Ωmega Drive-Line II).

Eingang w_CONTROLWORD:

Am Eingang w_CONTROLWORD wird das Steuerwort angegeben, das an den V-Regler gesendet werden soll.

Eingang w_COMMAND_REG:

Am Eingang w_COMMAND_REG findet

- die Auswahl der Soll- und Istwertübertragung,
- die Auswahl der Berechnungsmethode der Kommunikations-Zykluszeit der Soll- und Istwertübertragung sowie die
- die Einstellung der Kommunikations-Zykluszeit der Soll- und Istwertübertragung

statt.

Bit-Nr.	Bedeutung		
0	Reserviert		
1, 2	Bit 2	Bit 1	Auswahl der Soll- und Istwertübertragung (→ $t_{cyc\ P}$)
	0	0	Zeitscheibenverfahren
	0	1	Reserviert
	1	0	Zwei Soll- und zwei (vier) Istwerte im selben Zyklus
	1	1	Reserviert
3	Bit3		Auswahl der Berechnungsmethode der Kommunikations-Zykluszeit der Soll- und Istwertübertragung (→ $t_{cyc\ K}$)
	0		Zähler
	1		Zeitscheibe
4, 5, 6 ,7	1…15:		Bit 3 = 0: Wert des Zählers Alle 500µs wird intern ein Zähler inkrementiert. Stimmt der Stand dieses Zählers mit dem Wert, der aus den Bits 4 bis 7 gebildet wird, überein, so wird eine Prozeßdaten-Kommunikation ausgeführt (sofern ein Ereignis vom V-Regler ansteht).
	1…15:		Bit 3 = 1: Nummer des Zeitschlitzes Prozeßdaten-Kommunikation findet jeweils 500µs nach dem Zeitschlitz statt, dessen Nummer in den Bits 4 bis 7 eingetragen ist (sofern ein Ereignis vom V-Regler ansteht).
8…15	reserviert		

Zeitscheibe des V-Reglers für die zyklische Kommunikation: 500 µs

$t_{cyc\ K}$ - Zeit zwischen zwei Kommunikationen über die BAPS

$t_{cyc\ P}$ - Zeit mit der die Parameter (Soll- und Istwerte) an Position x (Eingänge ud_WR_VALUEx und Ausgänge ud_RD_VALUEx) aktualisiert werden

Ermittlung $t_{cyc\ K}$: → Bit 3 und 4 bis 7

Zähler: → Bit 3 = 0

$t_{cyc\ K}$ = 0,5 ms * "Wert aus Bit 4 bis 7"

Minimalwert:	0,5 ms * 1	= 0,5 ms
Beispiel:	0,5 ms * 2	= 1,0 ms
Beispiel:	0,5 ms * 3	= 1,5 ms
Beispiel:	0,5 ms * 4	= 2,0 ms
	usw.	
Maximalwert:	0,5 ms * 15	= 7,5 ms

Zeitscheibe: → Bit 3 = 1

$t_{cyc\ K}$ = 0,5 ms * $2^{\text{"Wert aus Bit 4 bis 7"}}$

Minimalwert:	0,5 ms * 2^1	= 1 ms
Beispiel:	0,5 ms * 2^2	= 2 ms
Beispiel:	0,5 ms * 2^3	= 4 ms
Beispiel:	0,5 ms * 2^4	= 8 ms
Beispiel:	0,5 ms * 2^5	= 16 ms
	usw.	
Maximalwert:	0,5 ms * 2^{15}	= 16384 ms

HINWEIS

Nach jeder Zykluszeit $t_{cyc\ K}$ muß im Ωmega Drive-Line II der FB BAPS_PD_COMM8 aufgerufen werden (z.B. in einer Event-Task auf das Ereignis "BAPS-Prozeßdaten" oder in einer Event-Task auf das Ereignis "Sync-Signal Netzwerk (CANsync)").

HINWEIS

Wenn die (BAPS-) Prozeßdatenkommunikation über ein Synchronisier-Signal getriggert wird oder die Betriebsart „Synchrone Lage-Sollwert-Vorgabe“ verwendet wird muß im V-Regler der Parameter 167 (Sync.-Slot) auf $t_{cyc\ K}$ in µs eingestellt werden.

Ermittlung $t_{cyc\ P}$: Bit 1 und 2:

Zeitscheibenverfahren (bis 8 Sollwerte und 8 Istwerte):

$t_{cyc\ P\ x} = (2 * t_{cyc\ K}) * 2^{x}$; x: Soll-/Istwertnummer am FB: 0 bis 7 (Eingänge ud_WR_VALUEx und Ausgänge ud_RD_VALUEx)

$t_{cyc\ P\ 0} = (2 * t_{cyc\ K}) * 2^{0} =$	$2 * t_{cyc\ K}$	=	1 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 1} = (2 * t_{cyc\ K}) * 2^{1} =$	$4 * t_{cyc\ K}$	=	2 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 2} = (2 * t_{cyc\ K}) * 2^{2} =$	$8 * t_{cyc\ K}$	=	4 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 3} = (2 * t_{cyc\ K}) * 2^{3} =$	$16 * t_{cyc\ K}$	=	8 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 4} = (2 * t_{cyc\ K}) * 2^{4} =$	$32 * t_{cyc\ K}$	=	16 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 5} = (2 * t_{cyc\ K}) * 2^{5} =$	$64 * t_{cyc\ K}$	=	32 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 6} = (2 * t_{cyc\ K}) * 2^{6} =$	$128 * t_{cyc\ K}$	=	64 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 7} = (2 * t_{cyc\ K}) * 2^{7} =$	$256 * t_{cyc\ K}$	=	128 ms	(bei $t_{cyc\ K}$ = 0,5 ms)
$t_{cyc\ P\ 0} = (2 * t_{cyc\ K}) * 2^{0} =$	$2 * t_{cyc\ K}$	=	4 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 1} = (2 * t_{cyc\ K}) * 2^{1} =$	$4 * t_{cyc\ K}$	=	8 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 2} = (2 * t_{cyc\ K}) * 2^{2} =$	$8 * t_{cyc\ K}$	=	16 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 3} = (2 * t_{cyc\ K}) * 2^{3} =$	$16 * t_{cyc\ K}$	=	32 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 4} = (2 * t_{cyc\ K}) * 2^{4} =$	$32 * t_{cyc\ K}$	=	64 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 5} = (2 * t_{cyc\ K}) * 2^{5} =$	$64 * t_{cyc\ K}$	=	128 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 6} = (2 * t_{cyc\ K}) * 2^{6} =$	$128 * t_{cyc\ K}$	=	256 ms	(bei $t_{cyc\ K}$ = 2 ms)
$t_{cyc\ P\ 7} = (2 * t_{cyc\ K}) * 2^{7} =$	$256 * t_{cyc\ K}$	=	512 ms	(bei $t_{cyc\ K}$ = 2 ms)

2 Sollwerte und 4 Istwerte im selben Zyklus:

$t_{cyc\ P\ 0} = t_{cyc\ P\ 1} = t_{cyc\ P\ 2} = t_{cyc\ P\ 3} = t_{cyc\ K}$

Eingang us_MODE:

Mit us_MODE = 0 wird angegeben, daß der FB BAPS_PD_COMM8 im zyklischen Hauptprogramm oder in einer Event-Task auf ein beliebiges Ereignis (z.B. auf das Ereignis "Sync-Signal Netzwerk (CANsync)") aufgerufen wird.

Mit us_MODE = 1 wird angegeben, daß der FB BAPS_PD_COMM8 in einer Event-Task auf das Ereignis "BAPS-Prozeßdaten" aufgerufen wird (siehe Beschreibung FB BAPS_INIT, Eingang i_EVENT).

Wird us_MODE nicht belegt ergibt sich die Voreinstellung us_MODE = 0 (Einsatz des FB BAPS_PD_COMM8 im zyklischen Hauptprogramm oder in einer Event-Task auf ein beliebiges Ereignis).

Eingänge ud_WR_VALUE0 bis ud_WR_VALUE7:

An den Eingängen ud_WR_VALUE0 bis ud_WR_VALUE7 werden die Sollwerte angeschlossen, deren Parameternummern am FB BAPS_INIT (Eingänge u_WR_PAR_NR0 bis u_WR_PAR_NR7) in der Initialisierung der BAPS-Prozeßdatenkommunikation angegeben wurden.

Eingang x_EN:

Mit x_EN = TRUE wird die Kommunikation freigegeben. Voreinstellung ist x_EN = TRUE, d.h. die Kommunikation ist freigegeben.

Ausgang w_STATUSWORD:

Am Ausgang w_STATUSWORD wird das Statuswort des V-Reglers ausgegeben.

Ausgang w_STATUS_REG:

Der Ausgang w_STATUS_REG (BAPS Status-Register) zeigt mit dem gesetzten Bit 0 an, daß der V-Regler auf das Signal Trigger Controller synchronisiert ist (siehe “Die Interruptquellen und Triggersignale” auf Seite 56).

Ausgänge ud_RD_VALUE0 bis ud_RD_VALUE7:

An den Ausgängen ud_RD_VALUE0 bis ud_RD_VALUE7 werden die Istwerte ausgegeben, deren Parameternummern am FB BAPS_INIT (Eingänge u_RD_PAR_NR0 bis u_RD_PAR_NR7) in der Initialisierung der BAPS-Prozeßdatenkommunikation angegeben wurden.

Ausgang b_SL_QUIT:

Am Ausgang b_SL_QUIT wird die V-Reglerquittung nach Abschluß der Kommunikation gemeldet.

Wert b_SL_QUIT	**Bedeutung**
16#00	ohne Bedeutung
16#01	Sollwert gelesen / Istwert geschrieben
16#02	Konfiguration / Initialisierung korrekt durchgeführt
16#03 – 16#7F	reserviert
16#80	nicht interpretierbares Kommando empfangen
16#81	keine Konfiguration / Initialisierung durchgeführt
16#82	Istwert lässt sich nicht lesen
16#83	Sollwert lässt sich nicht schreiben
16#84 – 16#FE	reserviert
16#FF	ohne Bedeutung

Ausgänge x_ERR, b_ERR:

Falls ein Fehler auftritt, wird das Fehlerbit x_ERR auf TRUE gesetzt und das Fehlerbyte b_ERR ausgegeben.

Fehlerbyte: b_ERR

Bit-nummer	**Fehler**
0	Ausgang b_SL_QUIT ≠ 16#01
1 – 7	reserviert

6.2.7 BAPS_PD_CONTROL

Beschreibung

Diesen Funktionsbaustein für BAPS können Sie verwenden, um den Aufruf der Prozeßdatenkommunikation zu kontrollieren.

Die Überwachung des ordnungsgemäßen Ablaufs der Prozeßdatenkommunikation findet im FB BAPS_PD_COMM2, FB BAPS_PD_COMM24 oder FB BAPS_PD_COMM8 (im folgenden BAPS_PD_COMMxx) statt.

Parameter Eingang	Datentyp	Beschreibung
x_RESET	BOOL	Reset
t_TIME	TIME	Überwachungszeit

Parameter Ausgang	Datentyp	Beschreibung
x_ERR	BOOL	Fehlerbit

Auf der BAPS-Schnittstelle wird bei jedem Aufruf der BAPS-Prozeßdatenkommunikation durch den FB BAPS_PD_COMMxx ein bestimmtes Register (Timeout-Register) geändert. Durch die Überwachung der Änderung dieses Registers ist die Überwachung des Aufrufs der BAPS-Prozeßdatenkommunikation möglich.

Die Änderung dieses Registers ist unabhängig vom Ergebnis der Kommunikation!

Eingang x_RESET:

Mit x_RESET = TRUE kann der FB BAPS_PD_CONTROL zurückgesetzt werden.

Eingang t_TIME:

Die Überwachungszeit wird am Eingang t_TIME eingestellt. Wird der Eingang t_TIME nicht belegt, ergibt sich eine Voreinstellung von 3 s.

Ausgang x_ERR:

Mit x_ERR = TRUE wird angezeigt, daß auf der BAPS-Schnittstelle das o. g. Timeout-Register der BAPS-Schnittstelle innerhalb der Überwachungszeit (t_TIME) nicht mehr geändert wurde. Die Ursachen hierfür können sein, daß der FB BAPS_PD_COMMxx nicht aufgerufen wird oder, daß die Event-Task mit dem FB BAPS_PD_COMMxx nicht ausgelöst wird.

HINWEIS

Der Ausgang x_ERR ist TRUE wenn keine Prozeßdatenkommunikation über die BAPS innerhalb von t_TIME durchgeführt wurde.
Der FB BAPS_PD_CONTROL wird nicht in einer Event-Task auf das Ereignis BAPS-Prozeßdaten eingesetzt (siehe Beschreibung FB BAPS_INIT, FB BAPS_PD_COMMxx).

6.2.8 BAPS_SD_CONTROL

Beschreibung

Diesen Funktionsbaustein für BAPS können Sie verwenden, um einen Bedarfsdatenwert (Parameter) vom V-Regler über die BAPS-Schnittstelle zu lesen oder zu schreiben.

Der Austausch der Daten und die Steuerung des FB BAPS_SD_CONTROL erfolgt über eine Struktur, der von den FBs BAPS_PAR_READ und/oder BAPS_PAR_WRITE beschreiben und gelesen wird.

HINWEIS

Der FB BAPS_SD_CONTROL verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BAPS_SD_DATA	BAPS_BMSTRUCT	Kommunikationsdaten
x_RESET	BOOL	Reset
t_TIME	TIME	Überwachungszeit

Parameter Ausgang	Datentyp	Beschreibung
_BAPS_SD_DATA	BAPS_BMSTRUCT	Kommunikationsdaten
b_ERR	BYTE	Fehlerbyte
x_ERR	BOOL	Fehlerbit

Der FB BAPS_SD_CONTROL führt die Bedarfsdatenkommunikation über die BAPS-Schnittstelle durch. Die Aufträge für diese Kommunikation erhält der FB BAPS_SD_CONTROL von den FBs BAPS_PAR_READ und/oder BAPS_PAR_WRITE.

Der Parameter-Lesen-Auftrag vom FB BAPS_PAR_READ wird an den V-Regler weitergegeben und die vom V-Regler zurückgegebenen Daten an den FB BAPS_PAR_READ zurückgegeben.

Der Parameter-Schreiben-Auftrag vom FB BAPS_PAR_WRITE wird an den V-Regler weitergegeben und das vom V-Regler zurückgegebene Ergebnis der Kommunikation wird an den FB BAPS_PAR_WRITE zurückgegeben.

Fehler in der Kommunikation über die BAPS-Schnittstelle als auch im Datenaustausch mit den FBs BAPS_PAR_READ und BAPS_PAR_WRITE werden an den Fehlerausgängen angezeigt.

Ein-/Ausgang _BAPS_SD_DATA:

An _BAPS_SD_DATA muß eine globale Variable vom Datentyp BAPS_BMSTRUCT angeschlossen werden.

Beispiel:

```
_BAPS_SD_DATEN : BAPS_BMSTRUCT;
```

dabei ist:

`_BAPS_SD_DATEN`	der Variablenname mit der Datentypkurzbezeichnung "_" für STRUCT
`BAPS_BMSTRUCT`	der Datentyp

Über diese Variable erfolgt der Datenaustausch mit den FBs BAPS_PAR_READ und BAPS_PAR_WRITE. Diese Variable wird bei den FBs der Bedarfsdatenkommunikation der BAPS angeschlossen - dies gilt auch, wenn die FBs BAPS_PAR_READ und/oder BAPS_PAR_WRITE mehrfach eingesetzt werden.

Eingang x_RESET:

Mit x_RESET = TRUE kann der FB BAPS_SD_CONTROL zurückgesetzt werden. Das ist notwendig wenn die Kommunikation über die BAPS-Schnittstelle oder der Datenaustausch mit den FBs BAPS_PAR_READ und BAPS_PAR_WRITE fehlerhaft war. Auch die FBs BAPS_PAR_READ und/oder BAPS_PAR_WRITE müssen zurückgesetzt werden, damit der Datenaustausch mit diesen FBs neu gestartet werden kann.

Eingang t_TIME:

Die Überwachungszeit wird am Eingang t_TIME eingestellt. Wird der Eingang t_TIME nicht belegt, ergibt sich eine Voreinstellung von 3 s.

Ausgänge x_ERR, b_ERR:

Falls ein Fehler auftritt, wird das Fehlerbit x_ERR auf TRUE gesetzt und das Fehlerbyte b_ERR ausgegeben.

Fehlerbyte b_ERR:

Bit-nummer	**Fehler**
0, 1	Datenaustausch mit den FBs BAPS_PAR_READ / BAPS_PAR_WRITE gestört
2	BAPS-Schnittstelle: Timeout, Master- oder Slave-Semaphore ungleich 0, kein Kommunikationsstart möglich
3	BAPS-Schnittstelle: Timeout, Master- oder Slave-Semaphore ungleich 1, Kommunikation unterbrochen
4	BAPS-Schnittstelle: Timeout, Master- oder Slave-Semaphore ungleich 0, Kommunikation nicht beendet
5	BAPS-Schnittstelle: allgemeiner Betriebssystem- oder Kommunikationsfehler
6	BAPS-Schnittstelle: Formatfehler
7	reserviert

HINWEIS

Das Master-Semaphore wird vom Ωmega Drive-Line II, das Slave-Semaphore wird vom V-Regler beschrieben.

7 CANSYNC

HINWEIS

In diesem Kapitel genannte Funktionsbausteine befinden sich in den Bibliotheken SYSTEM1_DLII_20bd00 (oder höher), SYSTEM2_DLII_20bd00 (oder höher) und CANsync_DLII_20db00 (oder höher).
In diesem Kapitel genannten Datentypen sind in der Bibliothek BM_TYPES_20bd00 (oder höher) definiert.
Zur Programmierung des CANsync unter PROPROG wt II werden diese Bibliotheken in ein Projekt eingebunden.

7.1 Allgemeines

7.1.1 Überblick

Der CANsync-Feldbus ist eine Entwicklung der Baumüller Nürnberg GmbH. Das Ziel, mechanische Königswellen durch eine elektronische Leitachse zu ersetzen, wurde dadurch erreicht, daß allen angeschlossenen Antrieben (⇒ CANsync-Slaves) der Leitachswert zum selben Zeitpunkt zur Verfügung gestellt wird (zeitsynchrone Übertragung).

Physikalische Basis stellt der CAN-Bus dar. Dieser wurde erweitert durch die Hinzufügung eines Synchronisations-Signals (SYNC-Signal). Das SYNC-Signal wird auf zwei zusätzlichen Adern im CAN-Kabel übertragen. Das SYNC-Signal dient zur Hardwaresynchronisation des CANsync-Masters mit allen am CANsync-Bus befindlichem CANsync-Slaves. Dadurch ist es im Gegensatz zum CAN-Bus möglich, die Telegramme zu definierten Zeitpunkten zu senden und zu empfangen. Es wird ein garantierter, hoher Datendurchsatz erzielt, der zudem einen festen Zeitbezug auf dem CANsync-Bus besitzt.

Der CANsync-Bus ist ein Master-Slave-Bus mit einem CANsync-Master und bis zu 32 CANsync-Slaves. Zur Unterscheidung der CANsync-Slaves wird jedem CANsync-Slave eine Slavenummer zugeordnet. Die Slavenummer wird über DIP-Schaltereinstellung festgelegt (siehe “Einstellung der Slavenummer” auf Seite 22).

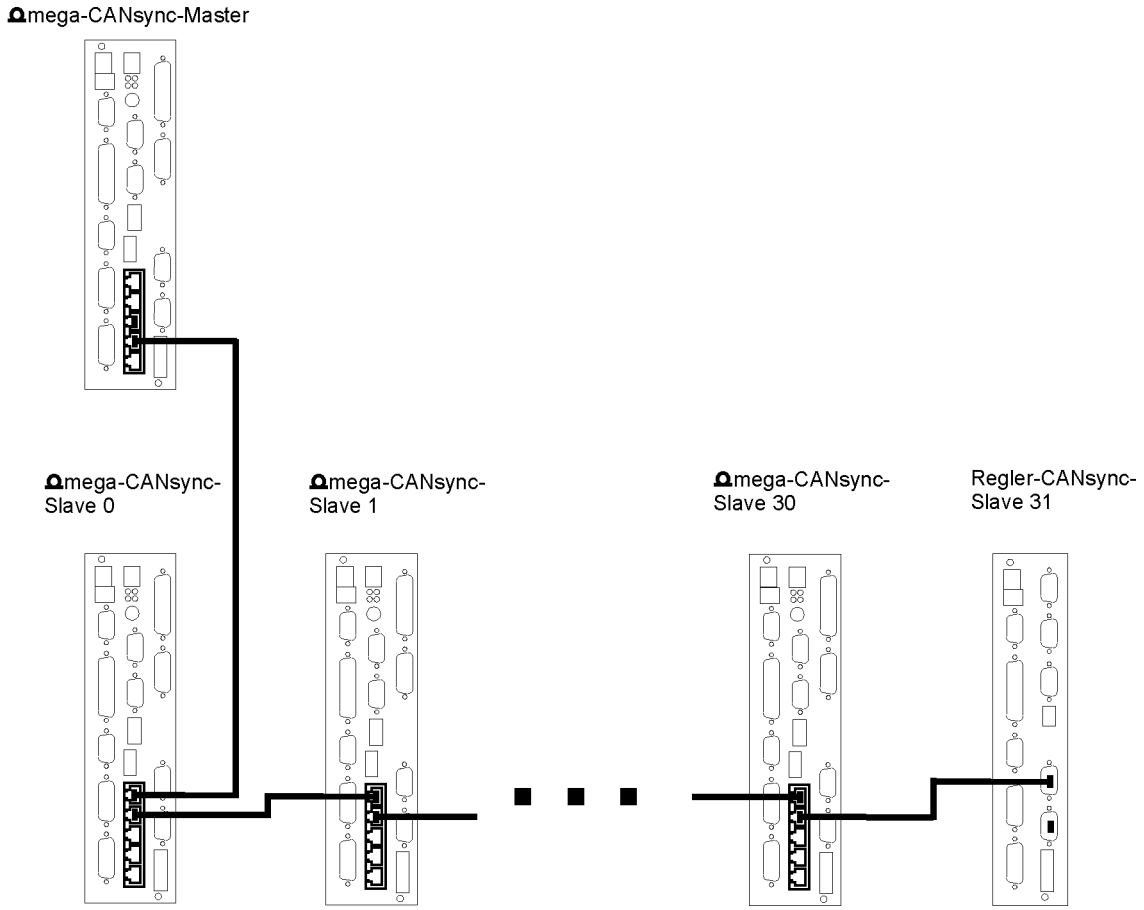


Abbildung 7-1: CANsync-Bus, CANsync-Master mit CANsync-Slaves 0, 1, .., 30,31

Zur Erweiterung des CANsync-Busses auf mehr als 32 CANsync-Slaves wurde das Clustering realisiert. Clustering bedeutet, daß neue CANsync-Netzwerke abgehend von einem sich im CANsync-Bus befindlichen CANsync-Slave eingebunden werden können. Dieser stellt für den Cluster den CANsync-Master dar. Dadurch ist es ebenfalls möglich, Folgeachsen zu realisieren, die selber als Leitachse für die in dem Cluster befindlichen Antriebe dienen.

Das SYNC-Signal steht jedem Teilnehmer des CANsync-Busses, auch in den Clustern, zur Synchronisation zur Verfügung, der Zeitpunkt der Übernahme bestimmter Telegramme ist jedem Teilnehmer bekannt!

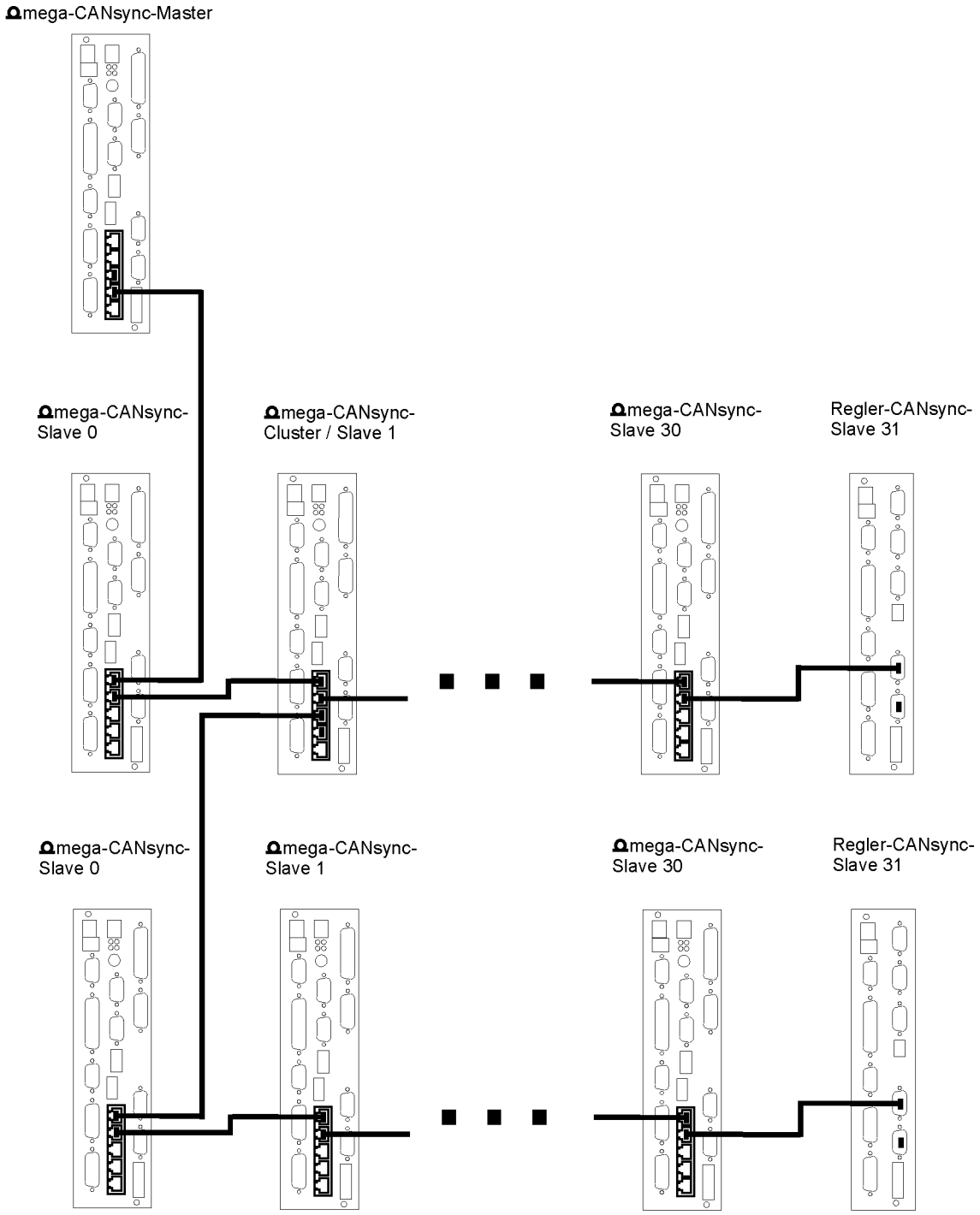


Abbildung 7-2: Beispiel für den Aufbau eines CANsync-Clusters.

Die maximale Teilnehmeranzahl beträgt theoretisch 32^{16}. Da neben der über die DIP-Schalter eingestellte Slavenummer (siehe “Einstellung der Slavenummer” auf Seite 22) aber ebenfalls noch eine CANsync-Nummer vergeben sein muß (eindeutige Identifizierung des Teilnehmers für den CANsync-Bus), findet eine Beschränkung auf maximal 65535 Teilnehmer statt.

Das CANsync-Synchronisationssignal (SYNC-Signal)

- ist ein spezifiziertes Hardwaresignal,
- wird auf der CANsync-Anschaltung des CANsync-Masters generiert,

- wird über zwei zusätzliche Leitungen im CANsync-Bus übertragen,
- wird auch zu den Clustern übertragen.

Je nach Betriebsart und Übertragungsgeschwindigkeit (Baudrate) auf dem CANsync-Bus wird das SYNC-Signal in einem bestimmten Raster, der CANsync-Zykluszeit (Zeit zwischen zwei fallenden Flanken des SYNC-Signals), generiert.

Baudrate	CANsync-Zykluszeit
500 kBit/s	2 ms
250 kBit/s	4 ms
125 kBit/s	8 ms

Der Telegrammverkehr des CANsync-Busses findet in einer festgelegten Reihenfolge statt, so daß sich die einzelnen Telegramme immer in definierten Zeitfenstern befinden, diese werden auch Kanäle genannt. Folgende Kanäle sind im CANsync definiert und werden vom CANsync-Master gesendet:

- Sollwert-Telegramme, WRC1 und WRC2 (**Wr**ite**C**hannel)
- Broadcast-Telegramme, CC (**C**ommand**C**hannel)
- Parameter-Telegramme, CC
- Up-Download-Telegramme, CC

Die CANsync-Slaves müssen ihre Antworten dem Zeitschema des CANsync-Masters anpassen, ihnen stehen folgende Kanäle zur Verfügung:

- Istwert-Telegramme, RDC1 und RDC 2 (**R**ea**dC**hannel)
- Parameter-Antwort-Telegramme, RC (**R**esponse**C**hannel)
- Up-/Download-Antwort-Telegramme, RC

HINWEIS

WRC1 und WRC2 werden nachfolgend auch Sollwertkanal 1 und Sollwertkanal 2 genannt.
RDC1 und RDC2 werden nachfolgend auch Istwertkanal 1 und Istwertkanal 2 genannt.
CC wird nachfolgend auch Kommandokanal genannt.
RC wird nachfolgend auch Antwortkanal genannt.

Das folgende Beispiel zeigt das Zeitschema der jeweiligen Telegramme mit der Übertragungsgeschwindigkeit 500 kbit/s (CANsync-Intervall mit der CANsync-Zykluszeit 2 ms):

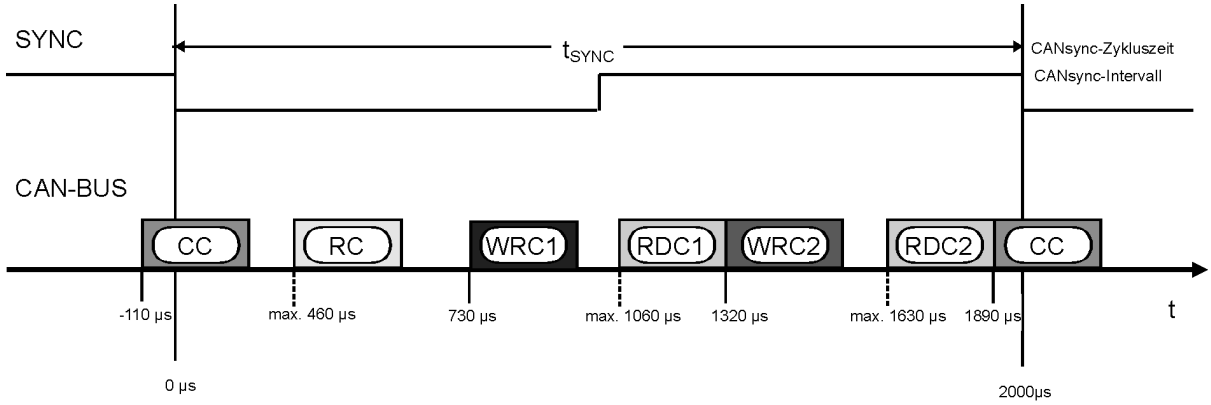


Abbildung 7-3: Zeitschema des CANsync-Intervalls mit der Übertragungsgeschwindigkeit 500 kbit/s

Die Zeitpunkte für die Kanäle sind in Abhängigkeit von der Baudrate in der nachfolgenden Tabelle angegeben:

Tabelle: Zuordnung Übertragungsgeschwindigkeit (Baudrate), CANsync-Zykluszeit t_{SYNC} in ms und maximale Buslänge in m. Die angegebenen Zeiten sind Maximalwerte .

Baudrate	**CANsync-Zykluszeit in µs t_{SYNC}**	**t_{WRC1}**	**t_{RDC1}**	**t_{WRC2}**	**t_{RDC2}**	**t_{CC}**	**t_{RC}**	**max. CANsync-Buslänge**
500 kBit/s	2000	730	1060	1320	1630	1890	460	134 m
250 kBit/s	4000	1430	2100	2620	3290	3810	900	300 m
125 kBit/s	8000	3020	4310	5350	6640	7680	1920	600 m

Prinzipieller Ablauf der Kommunikation

Prozeßdatenkommunikation

Für die Prozeßdatenkommunikation werden die Sollwertkanäle WRC1 und WRC2 (WriteChannel), die Istwertkanäle RDC1 und RDC2 (ReadChannel) sowie der Kommandokanal CC (CommandChannel) verwendet. In jedem CANsync-Zyklus werden sie in der vorgegebenen Reihenfolge gesendet.

Die Sollwert-Telegramme in den Kanälen WRC1 und WRC2 werden vom Master verschickt. Sie besitzen eine fest definierte Länge an Nutzdaten von jeweils 64 Bit ≙ 8 Byte ≙ 4 Worte!

Sie werden von allen Slaves/Teilnehmern empfangen, dabei entscheidet jeder Slave eigenständig, aufgrund seiner eingestellten Konfiguration (→ Mapping), welche Daten für ihn Sollwerte darstellen. Getroffen wird diese Einstellung für die V-Regler-Slaves in den Zusatzkartenparametern (ZK-Parameter, siehe Betriebsanleitung Optionskarte CANsync-Interface), für die Ωmega-Slaves in dem zu programmierenden Mapping.

Als zusätzliche Information wird dem Sollwert-Telegramm hinzugefügt, welcher Slave in dem selben CANsync-Zyklus sein bereits vorbereitetes Istwert-Telegramm an den Master zurücksenden soll. So ist es möglich, die Istwert-Telegramme von maximal zwei Slaves pro CANsync-Zyklus zu erhalten, da nur

zwei Istwert-Telegramme zur Verfügung stehen (in den Kanälen RDC1 und RDC2). Auch diese Telegramme besitzen die Nutzdatengröße von jeweils 64 Bit ≙ 8 Byte ≙ 4 Worte.

Eine automatische Abfrage aller Slaves nacheinander ist ebenfalls programmierbar.

In der Regel wird nur ein CANsync-Slave pro CANsync-Zyklus auf seine Istwerte abgefragt, so dass die Istwert-Telegramme in den Kanälen RDC1 und RDC2 von einem Slave stammen.

HINWEIS

Bei maximaler Buskonfiguration mit 32 Slaves (ohne zusätzlichen Cluster!) benötigt man mindestens 16 CANsync-Zyklen um alle aktuellen Istwerte im Master zur Verfügung zu haben, wenn pro CANsync-Zyklus zwei Slaves ihre Istwerte an den Master zurückmelden!

Beim V-Regler gehört zu den Prozeßdaten das Steuerwort und das Statuswort. Während das Statuswort wie ein gewöhnlicher Istwert in die Istwert-Telegramme 1 oder 2, je nach ZK-Parametereinstellung, eingetragen wird, muß das Steuerwort gesondert behandelt werden. Es ist als Broadcastkommando definiert, welches im CC gesendet wird.

Mapping

Im folgenden wird das Prinzip des Mapping am Beispiel der CANsync-Master-Anschaltung beschrieben. Für die CANsync-Slave-Anschaltung erfolgt das Mapping ähnlich (siehe auch FBs CANsync_PD_CFG_SL bzw. CANsync_PD_CFG_READ_SL.). Für V-Regler-Slaves erfolgt das Mapping in den Zusatzkartenparametern (siehe Betriebsanleitung Optionskarte CANsync-Interface).

Jeweils 64 Bit stehen als Nutzdaten für die Sollwert-Telegramme 1 und 2 zur Verfügung. Damit ergeben sich für die Sollwertvorgabe durch den Master folgende Möglichkeiten:

4 Wort-Sollwerte (je 16 Bit)

2 Doppelwort-Sollwerte (je 32 Bit)

1 Doppelwort-Sollwert (32 Bit) und 2 Wort-Sollwerte (je 16 Bit)

Die Sollwerte werden in ein Feld (Array) eingetragen, dass bei der Prozeßdatenkommunikation am FB CANsync_PD_COMM_MA angeschlossen wird. Es besteht aus 8 Einträgen mit jeweils 32 Bit Länge:

Sollwerte für das Sollwert-Telegramm 1	Sollwert 0	Highword	Sollwert 0	Lowword
	Sollwert 1	Highword	Sollwert 1	Lowword
	Sollwert 2	Highword	Sollwert 2	Lowword
	Sollwert 3	Highword	Sollwert 3	Lowword
Sollwerte für das Sollwert-Telegramm 2	Sollwert 4	Highword	Sollwert 4	Lowword
	Sollwert 5	Highword	Sollwert 5	Lowword
	Sollwert 6	Highword	Sollwert 6	Lowword
	Sollwert 7	Highword	Sollwert 7	Lowword

Diese Sollwerte werden im Kommunikations-RAM der CANsync-Anschaltung als Doppelworte zwischengespeichert. Aus diesem Grund ist es notwendig der CANsync-Anschaltung mitzuteilen, welche

Sollwerte und welches Wort, High- oder Lowword, für das Sollwert-Telegramm zu verwenden sind. Zu treffen ist diese Einstellung am Funktionsbaustein CANsync_PD_CFG_MA. Da sich diese Einstellung in der Regel nicht mehr ändert, ist sie während der Initialisierung des CANsync-Busses von Anwender vorzunehmen.

HINWEIS

Doppelwort-Parameter im V-Regler müssen in der Reihenfolge:
Erst Lowword, dann Highword übertragen werden (wie in den folgenden Beispielen angegeben).

Beispiel:

Sie wollen einen Doppelwort-Sollwert als Sollwert 0 im Sollwert-Telegramm 1 (im WRC1) schreiben und zusätzlich zwei Wortsollwerte als Sollwert 2 und Sollwert 3:

Zu treffende Einstellung: a_WRC1 = [0, 0, 2, 3]
a_HL_WRC1 = [FALSE, TRUE, FALSE, FALSE]

Sollwertkanal WRC1 mit Sollwert-Telegramm 1	Sollwert 0	Highword	Sollwert 0	Lowword
	----		----	
	----		Sollwert 2	Lowword
	----		Sollwert 3	Lowword

Die verwendeten Arrays werden an die Eingänge des FB CANsync_PD_CFG_MA angeschlossen. Das Array a_WRC1 definiert die Positionen der zu übertragenen Sollwerte, Doppelwort auf Sollwert 0, Wortsollwerte auf Sollwert 2 und 3. Das Array a_HL_WRC1 gibt dem System an, ob das High- (TRUE) oder Lowword (FALSE) aus dem Kommunikations-RAM genommen werden soll.

Auch für die beiden Istwert-Telegramme 1 und 2 stehen jeweils 64 Bit Nutzdaten zur Verfügung. Es können so

4 Wort-Istwerte (je 16 Bit)

2 Doppelwort-Istwerte (je 32 Bit)

1 Doppelwort-Istwert (32 Bit) und 2 Wort-Istwerte (je 16 Bit)

übertragen werden.

Diese Istwerte werden ebenfalls im Kommunikations-RAM als Doppelworte abgespeichert, so daß es auch hier notwendig ist, eine Zuordnung zu treffen.

Istwerte aus dem Istwert-Telegramm 1	Istwert 0	Highword	Istwert 0	Lowword
	Istwert 1	Highword	Istwert 1	Lowword
	Istwert 2	Highword	Istwert 2	Lowword
	Istwert 3	Highword	Istwert 3	Lowword

Istwerte aus dem Istwert-Telegramm 2	Istwert 4	Highword	Istwert 4	Lowword
	Istwert 5	Highword	Istwert 5	Lowword
	Istwert 6	Highword	Istwert 6	Lowword
	Istwert 7	Highword	Istwert 7	Lowword

Beispiel:

Sie wollen aus dem Istwert-Telegramm 1 (im RDC1) einen Doppelwort-Istwert als Istwert 0 (Position im Telegramm Wort 0 und 1) und einen zweiten Doppelwort-Istwert (Wort 2 und 3) als Istwert 2 lesen:

Zu treffende Einstellung:

```
a_RDC1    = [     0,     0,     2,     2]
a_HL_RDC1 = [FALSE,  TRUE, FALSE,  TRUE]
```

Istwertkanal RDC1 mit Istwert-Telegramm 1	Istwert 0	Highword	Istwert 0	Lowword
	----		----	
	Istwert 2	Highword	Istwert 2	Lowword
	----		----	

Die verwendeten Arrays werden an die Eingänge des FB CANsync_PD_CFG_READ_MA angeschlossen. Das Array a_RDC1 definiert die Positionen der zu übertragenen Istwerte, Doppelwort auf Istwert 0, Doppelwort auf Istwert 2. Das Array a_HL_RDC1 gibt dem System an, ob das High- (TRUE) oder Lowword (FALSE) in das Kommunikations-RAM geschrieben werden soll.

Im CANsync-Master werden die Istwerte der einzelnen CANsync-Slaves in unterschiedliche Bereiche des Kommunikations-RAM geschrieben, jedem CANsync-Slave wird ein eigener Bereich zugewiesen. Dieses Feld besteht aus 32 (da maximale Teilnehmerzahl ohne Cluster) • 8 (da 8 Istwerte) Einträgen, die jeweils 32 Bit (da Doppelwort-Istwerte) Daten beinhalten.

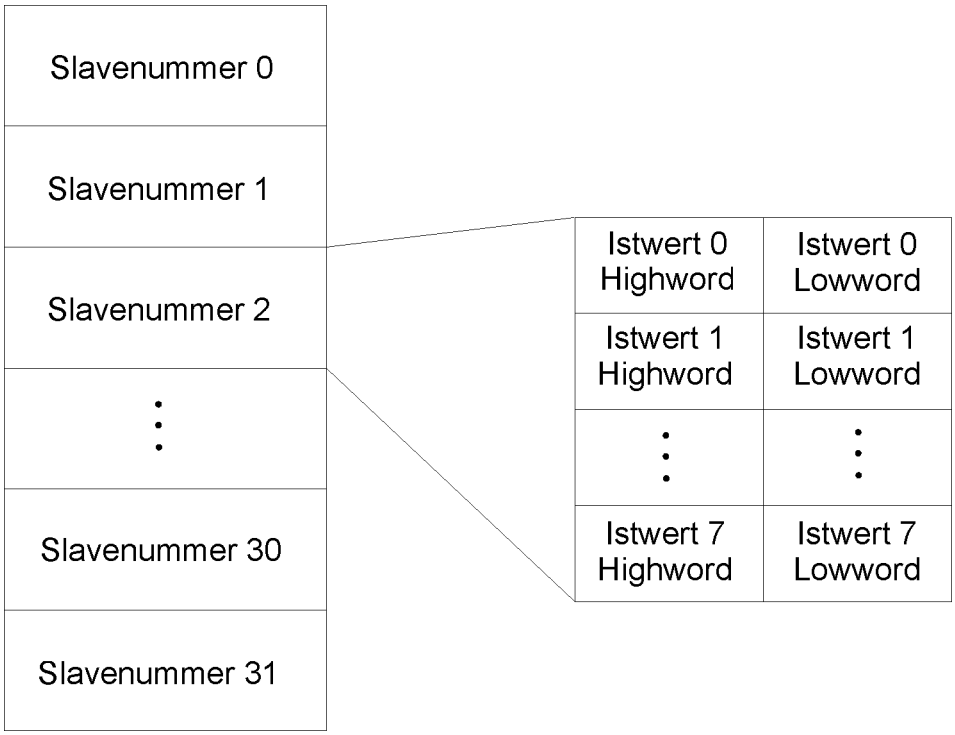


Abbildung 7-4: Istwerteintragung in das Kommunikations-RAM des CANsync-Masters der CANsync-Slaves, beispielhaft für Slave 2

Da in jedem CANsync-Istwert-Telegramm nur ein CANsync-Slave seine Istwerte und eventuell sein Statuswort sendet, ist es für alle anderen CANsync-Slaves möglich, diese Telegramme mitzuhören. Dadurch entsteht die Möglichkeit, dass diese Istwerte als Sollwerte für einen oder mehrere andere CANsync-Slaves genutzt werden. Ist dieses gewünscht, so muß die Einstellung lediglich im Mapping getroffen werden.

Zusammenfassung der Prozeßdatenkommunikation

Der CANsync-Master sendet in jedem CANsync-Intervall das Sollwert-Telegramm 1 und 2 und fordert (in der Regel) mit den Sollwert-Telegrammen die Istwert-Telegramme 1 und 2 von einem CANsync-Slave an.

Alle CANsync-Slaves empfangen die Sollwert-Telegramme des CANsync-Masters und senden nach der entsprechenden Aufforderung ihre Istwert-Telegramme an den CANsync-Master. Die Einstellung, welche Sollwerte in den Sollwert-Telegrammen für den CANsync-Slave relevant sind, wird im Mapping getroffen. Ebenfalls wird dort definiert, welche Istwert-Konstellation in das Istwert-Telegramm einzutragen ist.

Alle CANsync-Slaves können die vom Master angeforderten Istwert-Telegramme der anderen CANsync-Slaves auswerten.

Die Konfiguration des Mappings erfolgt mit folgenden FBs:

a) Im Ωmega CANsync-Master:

CANsync_PD_CFG_MA (FB wird 1 mal eingesetzt)	Mapping der zu sendenden Sollwert-Telegramme 1 und 2 im Ωmega CANsync-Master
CANsync_PD_CFG_READ_MA (FB wird je CANsync-Slave eingesetzt, max. 32 mal)	Mapping der empfangenen Istwert-Telegramme 1 und 2 eines CANsync-Slaves

b) Im Ωmega CANsync-Slave:

CANsync_PD_CFG_SL (FB wird 1 mal eingesetzt)	Mapping der empfangenen Sollwert-Telegramme 1 und 2 und der zu sendenden Istwert-Telegramme 1 und 2 im Ωmega CANsync-Slave
CANsync_PD_CFG_READ_SL (FB wird je weiteren CANsync-Slave eingesetzt, max. 31 mal)	Mapping der empfangenen Istwert-Telegramme 1 und 2 eines CANsync-Slaves im Ωmega CANsync-Slave

c) Im V-Regler CANsync-Slave:

In den Zusatzkartenparametern (siehe Betriebsanleitung Optionskarte CANsync-Interface)

Bedarfsdaten

Die Bedarfsdatenkommunikation findet im CANsync über den CommandCannel (CC) und den ResponseChannel (RC) statt. Dabei werden vom CANsync-Master im Kommandokanal Telegramme gesendet, welche Aktionen in einem oder mehreren CANsync-Slaves auslösen.

Es stehen mehrere Telegramme zur Verfügung:

- Broadcast-Telegramme
- Steuerwort-Telegramme (Sonderfall eines Broadcast-Telegramm)
- Parameter-Telegramme
- Up-/Download-Telegramme

Der CANsync-Slave sendet seine Antwort im Antwortkanal (RC), es kann sich um

- Parameter-Antwort-Telegramme
- Up-/Download-Antwort-Telegramme

handeln.

HINWEIS

Der CANsync-Master kann in jedem CANsync-Intervall Kommando-Telegramme senden. Die CANsync-Slaves antworten nur nach Aufforderung durch den CANsync-Master.

Die unterschiedlichen Telegramme im Kommandokanal werden in einer durch Prioritäten vorgegebenen Reihenfolge abgearbeitet:

Tabelle: Prioritätenvergabe der Telegramme im Kommandokanal (CC)

Telegrammtyp	Priorität
Broadcast-Telegramm 0	höchste
Broadcast-Telegramm 1	↑
Broadcast-Telegramm 2	|
Steuerwort-Telegramme	|
Parameter-Telegramme	↓
Up-/Download-Telegramme	niedrigste

Die Konsequenz dieser Prioritäten ist, daß kein anderes Telegramm verschickt werden kann, wenn ein höherpriores gesendet wird. Sendet man z. B. in jedem CANsync-Intervall das Steuerwort-Telegramm, so kann nie ein Parameter- oder Up-/Download-Telegramm verschickt werden!

In jedem CANsync-Intervall kann eines der folgenden Telegrame gesendet werden:

ein Broadcast-Telegramm an **alle** CANsync-Slaves oder
ein Steuerwort-Telegramm an **einen** CANsync-Slave oder
ein Parameter-Telegramm an **einen** CANsync-Slave oder
ein Up-/Download-Telegramm an **einen** CANsync-Slave

Die Steuerwort-, Parameter- oder Up-/Download-Telegramme werden, sofern kein Broadcast-Telegramm zu senden ist, in jedem CANsync-Intervall an einen CANsync-Slave gesendet.

Dabei kann im CANsync-Master eingestellt werden, ob an einen bestimmten CANsync-Slave ein Telegramm gesendet wird, oder ob er automatisch nacheinander an alle CANsync-Slaves ein Telegramm sendet.

Da nicht immer die Maximalkonfiguration an CANsync-Slaves (32 Stück ohne Cluster) vorhanden sein muß, kann dem CANsync-Master die maximale Slavenummer zur Steuerwort-, Parameter-, Up-/Download-Telegramm-Versendung sowie Istwert-Telegramm-Anforderung mitgeteilt werden. Sie wird an dem Funktionsbaustein CANsync_COMM_CONTROL_MA oder CANsync_PD_COMM_MA getroffen.

7.1.2 Hinweise zur Programmierung

Auf dem Ωmega Drive-Line II befinden sich 2 CANsync-Anschaltungen (CANsync-Knoten 1 und 2, auch Node 1 und 2). Sie dienen zur Verbindung von einem CANsync-Bus zu einem CANsync-Bus einer Subebene. Damit wird ein Netzwerk mit mehreren Ebenen gebildet.

Initialisierung

Wird das Ωmega Drive-Line II als CANsync-Master verwendet, muß die CANsync-Anschaltung auf dem CANsync-Knoten 2 (Hardwareadresse: %MB3.200000) als CANsync-Master initialisiert werden.

Anschließend wird der aktive Betrieb auf der CANsync-Master-Anschaltung freigegeben.

Benötigte FBs und deren Reihenfolge bei der Initialisierung als CANsync-Master:

CANsync_SL_TYP_INIT

CANsync_INIT

CANsync_COMM_CONTROL_MA

CANsync_PD_CFG_MA

CANsync_PD_CFG_READ_MA (je CANsync-Slave einmal)

OPT_INIT

INTR_SET

CANsync_MODE_MA

Wird das Ωmega Drive-Line II als CANsync-Slave verwendet, muß die CANsync-Anschaltung auf dem CANsync-Knoten 1 (Hardwareadresse: %MB3.100000) als CANsync-Slave initialisiert werden.

Anschließend wird der aktive Betrieb auf der CANsync-Slave-Anschaltung freigegeben.

Benötigte FBs und deren Reihenfolge bei der Initialisierung als CANsync-Slave:

CANsync_INIT

CANsync_PD_CFG_SL

CANsync_PD_CFG_READ_SL (je weiteren CANsync-Slave einmal)

OPT_INIT

INTR_SET

CANsync_MODE_SL

Wird das Ωmega Drive-Line II als CANsync-Cluster verwendet, muß zuerst die CANsync-Anschaltung auf dem CANsync-Knoten 1 (Hardwareadresse: %MB3.100000) als CANsync-Slave initialisiert werden.

Danach wird die CANsync-Anschaltung auf dem CANsync-Knoten 2 (Hardwareadresse: %MB3.200000) als CANsync-Master initialisiert.

Anschließend wird der aktive Betrieb auf der CANsync-Master-Anschaltung freigegeben.

Benötigte FBs und deren Reihenfolge bei der Initialisierung als CANsync-Cluster:

CANsync_INIT

CANsync_PD_CFG_SL

CANsync_PD_CFG_READ_SL (je weiteren CANsync-Slave einmal)

CANsync_SL_TYP_INIT

CANsync_INIT

CANsync_COMM_CONTROL_MA

CANsync_PD_CFG_MA

CANsync_PD_CFG_READ_MA (je CANsync-Slave einmal)

OPT_INIT

INTR_SET

CANsync_MODE_MA

Hinweise zur Prozeßdatenkommunikation

Die FBs der CANsync-Prozeßdatenkommunikation werden in einer POE plaziert, die der CANsync Event-Task zugeordnet ist. Die FBs der Prozeßdatenkommunikation müssen nach dem Aufruf der Event-Task sofort aufgerufen werden, damit Sollwerte und/oder Istwerte noch im gleichen Aufruf der Event-Task gesendet und/oder empfangen werden.

Wird das Ωmega Drive-Line II als CANsync-Master verwendet, muß der FB

CANsync_PD_COMM_MA

aufgerufen werden.

Wird das Ωmega Drive-Line II als CANsync-Slave verwendet, muß der FB

CANsync_PD_COMM_SL

aufgerufen werden.

Wird das Ωmega Drive-Line II als CANsync-Cluster verwendet, müssen die FBs in dieser Reihenfolge

CANsync_PD_COMM_SL

CANsync_PD_COMM_MA

aufgerufen werden.

7.2 Detaillierte Informationen zum CANsync

HINWEIS

Dieses Kapitel enthält detaillierte Informationen zum CANsync. Diese Informationen werden nicht benötigt, wenn zur Programmierung die Funktionsbausteine der Bibliothek CANsync_DLII_20bd00 oder höher verwendet werden.

t_{RSPTO} - Response Timeout: Zeit innerhalb der der CANsync-Slave während der Initialisierung eine Antwort senden muß.

Baudrate	t_{RSPTO}
500 kBit/s	600 µs
250 kBit/s	1100 µs
125 kBit/s	2100 µs

HINWEIS

Alle folgenden Zeitangaben beziehen sich auf eine Baudrate von 500 kBit/s.

Anlauf und Initialisierung

Das Anlaufverhalten gliedert sich in die folgenden Schritte:

⇨ Initialisierung mit FB CANsync_INIT:

Nachdem der FB CANsync_INIT erfolgreich durchlaufen wurde gibt der CANsync-Master das SYNC-Signal im Zeitraster von 2 ms und das Aktionskommando „SYNC-Modus“ mit dem Dateninhalt: „SYNC-Betrieb einschalten“ an alle CANsync-Slaves aus. Die CANsync-Slaves beginnen ihre Regelungs-Task auf das SYNC-Signal zu synchronisieren.

Der CANsync-Master fordert von jedem CANsync-Slave, den er am CANsync-Bus erwartet über das Parameter-Kommando „Parameter Lesen“ dessen Statuswort an.

Der CANsync-Slave muß innerhalb von t_{RSPTO} mit der Parameter-Antwort antworten. Der CANsync-Master überwacht, ob der CANsync-Slave innerhalb dieser Zeit antwortet.

Der CANsync-Master gibt während der Synchronisierung noch keine Sollwerte aus.

⇨ Übergang in den aktiven Betrieb

Mit dem FB CANsync_Mode (Eingang x_CANsync_RUN = TRUE) startet man direkt den aktiven Betrieb. Es werden dann auch Sollwerte und Istwerte übertragen.

Es ist dabei zu beachten, daß die CANsync-Slaves einige Zeit (Sekunden) benötigen, um sich auf das SYNC-Signal zu synchronisieren. Das Statuswort eines V-Regler CANsync-Slaves enthält die Information (Bit 15 = TRUE), ob der V-Regler CANsync-Slave synchronisiert ist.

Wenn die Übertragung sychroner Lagesollwerte gewünscht wird, darf das Applikationsprogramm die Maschine erst starten, wenn alle CANsync-Slaves synchronisiert sind (= Zustand "synchronisiert"). Andere Sollwerte, bzw. andere Betriebsarten sind auch schon davor zulässig.

Zustand „synchronisiert“

Im Zustand „synchronisiert“ laufen alle Antriebe (Ωmega CANsync-Master, Ωmega CANsync-Slaves, V-Regler CANsync-Slaves) im gleichen CANsync-Intervall, d. h. alle CANsync-Slaves bearbeiten gleichzeitig in ihren Regelungs-Task und übernehmen die Sollwerte zeitgleich.

Der CANsync-Master sendet seine Aufträge in definierter Reihenfolge mit zugeordneten Zeitfenstern. Die CANsync-Slaves müssen ihre Antworten diesem Schema anpassen (siehe “Überblick” auf Seite 121 und Abbildung 6-3).

Der Sollwert, der im vorherigen CANsync-Intervall empfangen wurde, wird im V-Regler in der Regelungs-Task 750 µs nach dem SYNC-Signal aktiv.

Der CANsync-Master muß bis spätestens 730 µs nach dem SYNC-Signal das nächste Sollwert-Telegramm 1 senden.

Synchronisierungs-Verlust bei V-Regler CANsync-Slaves

Fällt das SYNC-Signal in einem CANsync-Intervall aus, wird das nächste CANsync-Intervall noch durchlaufen. In diesem Fall kann das Istwert-Telegramm des CANsync-Slaves entfallen, wenn dieser nicht mit der nötigen Zeitpräzision senden kann.

Fällt das SYNC-Signal eine einstellbare Zeit in Folge aus, befindet sich der CANsync-Slave im nichtsynchronisierten Zustand. Für diesen Fall wird ein entsprechender Fehler gemeldet. Die Reaktion darauf ist vom Anwender im V-Regler über die Kommunikationsüberwachung einstellbar (Schnellhalt, Reglersperre,...).

Sollwert-Verlust bei V-Regler CANsync-Slaves

Wird in einem CANsync-Intervall kein neuer Sollwert empfangen, führt der CANsync-Slave eine Extrapolation mit dem zuletzt empfangenen Sollwert durch.

Bei einer einstellbaren Anzahl von Sollwertaussetzern (ZK 26) in Folge wird ein entsprechender Fehler (Kommunikationsüberwachung) gemeldet. Die Reaktion darauf ist vom Anwender im V-Regler einstellbar (Schnellhalt, Reglersperre,...).

7.2.1 Aufbau der Telegramme

⇨ **CANsync-Telegramme**

Die Telegrammlängen sind variabel (0 bis 8 Datenbytes). Sie ergeben sich aus der vorangegangenen Auflistung und hängen bei manchen Telegrammen auch von dem jeweiligen Betriebszustand ab.

⇨ **Datenformat**

Die Daten werden in den Telegrammen im Intelformat (Lowbyte / Highbyte) abgelegt.

⇨ **Statuswort**

Das Statuswort eines CANsync-Slaves gibt dessen Antriebzustand an. Im obersten Bit (15) muß der SYNC-Zustand angezeigt werden. Wenn das Bit gesetzt ist, ist der CANsync-Slave synchronisiert.

Sollwertkanäle

Sollwertkanal 1

Im Sollwertkanal 1 sendet der CANsync-Master das Sollwert-Telegramm 1.

Der CANsync-Master überträgt damit an alle CANsync-Slaves einen oder mehrere (max. 4) Sollwerte. Die Nummer NNNNNNN im Identifier gibt an, welcher CANsync-Slave im Anschluß an das Sollwert-Telegramm 1 sein Istwert-Telegramm 1 senden muß.

<IDENTIFIER><SOLLWERTE>			
<IDENTIFIER>	::=	0010NNNNNNN	Identifier Sollwertkanal 1
<SOLLWERTE>	::=	W_SOLL | DW_SOLL | W_DW_SOLL	
<W_SOLL>	::=	<W_SOLL_1> | <W_SOLL_1..2> | <W_SOLL_1..3> | <W_SOLL_1..4>	nur Wort-Sollwerte
<DW_SOLL>	::=	<DW_SOLL_1> | <DW_SOLL_1..2>	nur DWort-Sollwerte
<W_DW_SOLL>	::=	<DW_SOLL_1><W_SOLL_3> | <DW_SOLL_1><W_SOLL_3><W_SOLL_4>	Wort- und DWort-Sollwerte
<W_SOLL_1>	::=	<Word>	Wort-Sollwert 1 = CAN-DB 0..1
<W_SOLL_2>	::=	<Word>	Wort-Sollwert 2 = CAN-DB 2..3
<W_SOLL_3>	::=	<Word>	Wort-Sollwert 3 = CAN-DB 4..5
<W_SOLL_4>	::=	<Word>	Wort-Sollwert 4 = CAN-DB 6..7
<DW_SOLL_1>	::=	<DWord>	DWort-Sollwert 1 = CAN-DB 0..3
<DW_SOLL_2>	::=	<DWord>	DWort-Sollwert 2 = CAN-DB 4..7
			CAN-DB: CAN-Daten-Byte

Sollwertkanal 2

Im Sollwertkanal 2 sendet der CANsync-Master das Sollwert-Telegramm 2.

Der CANsync-Master überträgt damit auf diesem Sollwertkanal bis zu 4 zusätzliche Sollwerte.

Der Aufbau und die Funktion entsprechen dem Sollwertkanal 1. Das Sollwert-Telegramm 2 unterscheidet sich im Identifier und der Zuordnung der Sollwerte vom Sollwert-Telegramm 1. Dem Sollwertkanal 2 sind die Wort-Sollwerte 5...8 und die Doppelwort-Sollwerte 3...4 zugeordnet. Die Nummer NNNNNNN im Identifier gibt die Slave-Nummer des CANsync-Slaves an, der mit dem Istwert-Telegramm 2 antwortet.

<IDENTIFIER>	::=	0011NNNNNNN	Identifier Sollwertkanal 2

Istwertkanäle

Istwertkanal 1

Im Istwertkanal 1 sendet der CANsync-Slave das Istwert-Telegramm 1.

Der Identifier der Sollwert-Telegramm 1 gibt an, welcher CANsync-Slave sein Istwert-Telegramm 1 als direkte Antwort auf das Sollwert-Telegramm 1 schicken darf.

<IDENTIFIER><ISTWERTE>			
<IDENTIFIER>	::=	0110NNNNNNN	NNNNNNN = CANsync-Slave-Nummer
<ISTWERTE>	::=	W_IST | DW_IST | W_DW_IST	
<W_IST>	::=	<W_IST_1> | <W_IST_1..2> | <W_IST_1..3> | <W_IST_1..4>	nur Wort-Istwerte
<DW_IST>	::=	<DW_IST_1> | <DW_IST_1..2>	nur DWort-Istwerte
<W_DW_IST>	::=	<DW_IST_1><W_IST_3> | <DW_IST_1><W_IST_3><W_IST_4>	Wort- und DWort-Istwerte
<W_IST_1>	::=	<Word>	Wort-Istwert 1 = CAN-DB 0..1
<W_IST_2>	::=	<Word>	Wort-Istwert 2 = CAN-DB 2..3
<W_IST_3>	::=	<Word>	Wort-Istwert 3 = CAN-DB 4..5
<W_IST_4>	::=	<Word>	Wort-Istwert 4 = CAN-DB 6..7
<DW_IST_1>	::=	<DWord>	DWort-Istwert 1 = CAN-DB 0..3
<DW_IST_2>	::=	<DWord>	DWort-Istwert 2 = CAN-DB 4..7
			CAN-DB: CAN-Daten-Byte

Istwertkanal 2

Im Istwertkanal 2 sendet der CANsync-Slave das Istwert-Telegramm 2.

Der Aufbau und die Funktion des Telegramms entsprechen dem Istwert-Telegramm 1. Es sind maximal 4 weitere Istwerte übertragbar. Das Istwert-Telegramm 2 unterscheidet sind im Identifier und der Zuordnung der Istwerte. Dem Istwertkanal 2 sind die Wort-Istwerte 5...8 und die Doppelwort-Istwerte 3...4 zugeordnet.

<IDENTIFIER>	::=	0111NNNNNNN	NNNNNNN = CANsync-Slave-Nummer

Kommandokanal & Antwortkanal

Der Kommandokanal und der zugehörige Antwortkanal bestehen funktional aus drei Gruppen von Telegrammen, wobei in einem CANsync-Intervall immer nur je ein Kommando/Antwort vorkommen kann.

- ⇨ Aktionskommandos dienen der Initialisierung und Steuerung der CANsync-Slaves und werden an einen oder mehrere CANsync-Slaves gesendet, ohne daß vom CANsync-Master eine Antwort erwartet wird.
- ⇨ Parameterkommandos werden zum Lesen oder Schreiben eines Parameters verwendet und sind immer an einen CANsync-Slave gerichtet. Der CANsync-Master erwartet immer eine Antwort.
- ⇨ Up/Download-Kommandos dienen der Übertragung größerer Datenmengen (Programmcode, Datensatz) und sind immer an einen CANsync-Slave gerichtet. Der CANsync-Master erwartet immer eine Antwort.

Parameter- und Up-/Download-Kommandos werden mit dem gleichen Identifier gesendet.

Aktionskommando

Ein Aktionskommando wird vom CANsync-Master an einen einzelnen CANsync-Slave oder eine Gruppe von CANsync-Slaves gesendet. Die Auswahl erfolgt über eine Bitleiste (SLAVE_GROUP) in der jedem CANsync-Slave ein Bit zugeordnet ist. Wenn das Bit gesetzt ist, muß der zugeordnete CANsync-Slave dieses Kommando ausführen. In einem Broadcast-Kommando an alle CANsync-Slaves sind alle Bits in der Bitleiste gesetzt.

Die unterschiedlichen Kommandos unterscheiden sich im Datenbyte COMMAND. In Abhängigkeit vom Kommando folgen unterschiedlich viele weitere Datenbytes, die Daten zum Kommando enthalten.

<IDENTIFIER><SLAVE_GROUP><COMAND><DATA>			
<IDENTIFIER>	::=	00000010000	Identifier Aktionskommando
<SLAVE_GROUP>	::=	<DWord>	Slave-Bits 0..30 = CAN-DB 0..3
<COMMAND>	::=	<Byte> 1 = Steuerwort Schreiben	= CAN-DB 4
<DATA>	::=	<DATA_1>	vom Kommando abhängige Daten
<DATA_1>	::=	<res><Wert>	
<res>	::=	<Byte>	CAN-DB 5
<Wert>	::=	<Word>	Steuerwort = CAN-DB 6..7
			CAN-DB: CAN-Daten-Byte

Parameter Lesen

Mit dem Parameter Lesen Kommando, fordert der CANsync-Master einen Parameter des CANsync-Slaves zum Lesen an. Der CANsync-Slave erkennt an der Telegrammlänge (= 4 Datenbytes), daß es sich um ein Parameter-Lese-Kommando handelt. Der CANsync-Slave muß die Elementauswahl nicht unterstützen, er antwortet dann immer mit dem aktuellen Datenwert.

Der CANsync-Slave muß innerhalb der vorgegeben Antwortzeit t_{RSPTO} antworten. Falls er bis dahin noch nicht den Auftrag beenden kann, antwortet er mit der Parameter-Antwort, in der die Parameternummer des Auftrags eingetragen ist und das BUSY-Bit gesetzt ist. Wenn der CANsync-Master das nächste Mal das Parameter-Lese-Kommando auf den Parameter wiederholt und der CANsync-Slave den Auftrag inzwischen bearbeitet hat, dann antwortet er mit den angeforderten Daten und das BUSY-Bit ist auf Null gesetzt.

Falls der Leseauftrag nicht bearbeitet werden kann oder ein Fehler auftritt, setzt der CANsync-Slave das ERR-Bit und gibt in den Datenbytes einen Fehlercode an.

Die Parameternummer kann von 0 bis 4095 gehen.

Auftrag:

<IDENTIFIER><CONTROL><PARA_NUM_L><SUB-ADRESSE>

<IDENTIFIER>	::=	1010NNNNNNN	NNNNNNN = CANsync-Slave-Nummer
<CONTROL>	::=	<P><ELEMENT><PARA_NUM_H>	CAN-DB 0
<P>	::=	<Bit7>	0 = Kennung: Parameter -Kommando
<ELEMENT>	::=	<Bit6..4>	Elementauswahl des Parameter
<PARA_NUM_H>	::=	<Bit3..0>	Bit11..8 der Parameter-Nr.
<PARA_NUM_L>	::=	<Byte>	Bit7..0 der Parameter-Nr. = CAN-DB 1
<SUB-ADRESSE>	::=	<Word>	Sub-Slave-Adresse = CAN-DB 2..3

Antwort:

<IDENTIFIER><STATUS><PARA_NUM_L><DATA><SUB-ADRESSE>

<IDENTIFIER>	::=	1011NNNNNNN	NNNNNNN = CANsync-Slave-Nummer
<STATUS>	::=	<P><BUSY><ERR> <FREI><PARA_NUM_H>	CAN-DB 0
<P>	::=	<Bit7>	0 = Kennung: Parameter -Antwort
<BUSY>	::=	<Bit6>	0 = Antwort gültig, 1 = Auftrag wird bearbeitet
<ERR>	::=	<Bit5>	0 = kein Fehler, 1 = Fehler
<FREI>	::=	<Bit4>	frei
<PARA_NUM_H>	::=	<Bit3..0>	Bit11..8 der Parameter-Nr.
<PARA_NUM_L>	::=	<Byte>	Bit7..0 der Parameter-Nr. = CAN-DB 1
<DATA>	::=	<2_BYTE> | <4_BYTE> | <ERR_CODE>	
<2_BYTE>	::=	<Word>	Wort-Parameter = CAN-DB 2..3
<4_BYTE>	::=	<Dword>	DWort-Parameter = CAN-DB 2..5
<ERR_CODE>	::=	<Word>	2 Byte Fehlercode = CAN-DB 2..3
<SUB-ADRESSE>	::=	<Word>	Sub-Slave-Adresse = CAN-DB 4..5 / 6..7 CAN-DB: CAN-Daten-Byte

Parameter Schreiben

Mit dem Parameter Schreiben Kommando, schreibt der CANsync-Master einen Parameter in einem CANsync-Slave. Der CANsync-Slave erkennt an der Telegrammlänge (= 6 oder 8 Datenbytes), ob es sich um ein Wort oder einen Doppelwort-Parameter handelt. Beim Schreiben ist zur Zeit nur das Element 7, der Parameterwert zulässig.

Der CANsync-Slave muß innerhalb der vorgegeben Antwortzeit t_{RSPTO} antworten. Falls er bis dahin noch nicht den Auftrag beenden kann, antwortet er mit der Parameter-Antwort, in der die Parameternummer des Auftrags eingetragen ist und das BUSY-Bit gesetzt ist. Wenn der CANsync-Master das nächste Mal das Parameter-Schreiben-Kommando auf den Parameter wiederholt und der CANsync-Slave den Auftrag inzwischen bearbeitet hat, dann antwortet er mit der Parameternummer und das BUSY-Bit ist auf Null gesetzt.

Falls der Schreibauftrag nicht bearbeitet werden kann oder ein Fehler auftritt, setzt der CANsync-Slave das ERR-Bit und gibt in den Datenbytes einen Fehlercode an.

Die Parameternummer kann von 0 bis 4095 gehen.

Auftrag:

<IDENTIFIER><CONTROL><PARA_NUM_L><DATA><SUB-ADRESSE>

<IDENTIFIER>	::=	1010NNNNNNN	NNNNNNN = CANsync-Slave-Nummer
<CONTROL>	::=	<P><ELEMENT><PARA_NUM_H>	CAN-DB 0
<P>	::=	<Bit7>	0 = Kennung: Parameter-Kommando
<ELEMENT>	::=	<Bit6..4>	Elementauswahl des Parameters
<PARA_NUM_H>	::=	<Bit3..0>	Bit11..8 der Parameter-Nr.
<PARA_NUM_L>	::=	<Byte>	Bit7..0 der Parameter-Nr. = CAN-DB 1
<DATA>	::=	<2_BYTE> | <4_BYTE>	
<2_BYTE>	::=	<Word>	Wort-Parameter = CAN-DB 2..3
<4_BYTE>	::=	<Dword>	DWort-Parameter = CAN-DB 2..5
<SUB-ADRESSE>	::=	<Word>	Sub-Slave-Adresse = CAN-DB 4..5/6..7

Antwort:

<IDENTIFIER><STATUS><PARA_NUM_L><DATA><SUB-ADRESSE>

<IDENTIFIER>	::=	1011NNNNNNN	NNNNNNN = CANsync-Slave-Nummer
<STATUS>	::=	<P><BUSY><ERR> <FREI> <PARA_NUM_H>	CAN-DB 0
<P>	::=	<Bit7>	0 = Kennung: Parameter-Antwort
<BUSY>	::=	<Bit6>	0 = Auftrag fertig, 1 = Auftrag wird bearbeitet
<ERR>	::=	<Bit5>	0 = kein Fehler, 1 = Fehler
<FREI>	::=	<Bit4>	noch nicht belegt
<PARA_NUM_H>	::=	<Bit3..0>	Bit11..8 der Parameter-Nr.
<PARA_NUM_L>	::=	<Byte>	Bit7..0 der Parameter-Nr. = CAN-DB 1
<DATA>	::=	<0_BYTE> | <ERR_CODE>	
<0_BYTE>	::=	keine Daten-Bytes wenn fehlerfrei	
<ERR_CODE>	::=	<Word>	2 Byte Fehlercode = CAN-DB 2..3
<SUB-ADRESSE>	::=	<Word>	Sub-Slave-Adresse = CAN-DB 2..3/4..5
			CAN-DB: CAN-Daten-Byte

Beginn eines Up- oder Downloads

Mit der Upload- oder Download-Übertragung können größere zusammenhängende Datenbereiche vom CANsync-Master zum CANsync-Slave oder umgekehrt übertragen werden.

Die Konfiguration der Übertragung erfolgt mit einem Initialisierungs-Telegramm.

Der CANsync-Slave muß innerhalb der vorgegeben Antwortzeit t_{RSPTO} antworten. Falls er bis dahin noch nicht den Auftrag beenden kann, antwortet er mit der Up/Download-Antwort, in der das BUSY-Bit gesetzt ist. Wenn der CANsync-Master das nächste Mal das Up/Download-Telegramm wiederholt und der CANsync-Slave den Auftrag inzwischen bearbeitet hat, dann antwortet er mit Antwort, in der das BUSY-Bit auf Null gesetzt ist.

Falls der Upload oder Download-Auftrag nicht bearbeitet werden kann oder ein Fehler auftritt, setzt der CANsync-Slave das ERR-Bit und gibt in den Datenbytes einen Fehlercode an.

Die Startadresse ist eine Doppelwort-Adresse. Die maximale Upload oder Downloadlänge ist 4096 Bytes. Größere Datenbereiche müssen über mehrere Up/Download-Initialisierungen übertragen werden. Als Option kann eine Sub-Slave-Adresse angegeben werden. Diese Adresse gibt an, daß der folgenden Up/Download sich nicht direkt auf den angesprochenen CANsync-Slave bezieht, sondern daß die Up/Download-Telegramme an einen Sub-Slave weitergereicht werden. Diese Sub-Slave-Adresse bleibt bis zum Ende des Up/Downloads gültig. Für den nächsten Up/Download-Auftrag muß die Adresse wieder neu angegeben werden. Wenn die Sub-Adresse gleich Null ist, wird direkt der CANsync-Slave angesprochen und kein Sub-Slave.

Auftrag:

<IDENTIFIER><CONTROL><OFFSET_L><ADRESSE><SUB-ADRESSE>

<IDENTIFIER>	::=	1010NNNNNNN	NNNNNNN = CANsync-Slave-Nummer
<CONTROL>	::=	<L><U/D><MODE><OFFSET_H>	CAN-DB 0
<L>	::=	<Bit7>	1 = Kennung: Up/Download-Auftrag
<U/D>	::=	<Bit6>	0 = Upload, 1 = Download
<MODE>	::=	<Bit5..4>	01 = Initialisierung
<OFFSET_H>	::=	<Bit3..0>	Blocklänge in Bytes Bit 11..8 a)
<OFFSET_L>	::=	<Byte>	Blocklänge in Bytes Bit 7..0 = CAN-DB 1 a)
<ADRESSE>	::=	<Dword>	absolute Startadresse = CAN-DB 2..5
<SUB-ADRESSE>	::=	<Word>	Sub-Slave-Adresse = CAN-DB 6..7

Antwort:

<IDENTIFIER><STATUS><OFFSET_L><DATA>

<IDENTIFIER>	::=	1011NNNNNNN	NNNNNNN = CANsync-Slave-Nummer
<STATUS>	::=	<L><BUSY><ERR><FREI><OFFSET_H>	CAN-DB 0
<L>	::=	<Bit7>	1 = Kennung: Up/Download-Antwort
<BUSY>	::=	<Bit6>	0 = Auftrag fertig, 1 = Auftrag wird bearbeitet
<ERR>	::=	<Bit5>	0 = kein Fehler, 1 = Fehler
<FREI>	::=	<Bit4>	noch nicht belegt
<OFFSET_H>	::=	<Bit3..0>	Blocklänge in Bytes Bit 11..8 a)
<OFFSET_L>	::=	<Byte>	Blocklänge in Bytes Bit 7..0 = CAN-DB 1 a)
<DATA>	::=	<0_BYTE> | <ERR_CODE>	
<0_BYTE>	::=	keine Daten-Bytes, wenn fehlerfrei	
<ERR_CODE>	::=	<Word>	Bit15..0 des Fehlercodes = CAN-DB 2..3 CAN-DB: CAN-Daten-Byte

a) Bei einem Upload-Auftrag kann die Länge statt durch den CANsync-Master auch durch den CANsync-Slave festgelegt werden. In diesem Fall wird die Länge in der Antwort in <OFFSET_H> und <OFFSET_L> angegeben. Ansonsten stehen diese Werte in der Antwort auf Null.

Laufender Upload & Ende eines Uploads

Der Uploadvorgang besteht aus aufeinanderfolgenden Upload-Telegrammen in denen der CANsync-Master von der in der Initialisierung eingestellten Startadresse ab, aufeinanderfolgende Datenblöcke anfordert. Die Offsetadresse steigt fortlaufend als Byteadresse. Mit jedem Telegramm werden 6 Bytes Nutzdaten übertragen. Das heißt das erste Telegramm beginnt mit der Offsetadresse 0, das zweite fordert die Daten mit der Offsetadresse 6 an, usw.

Der CANsync-Slave muß innerhalb der vorgegeben Antwortzeit t_{RSPTO} antworten. Falls er bis dahin noch nicht den Auftrag beenden kann, antwortet er mit der Upload-Antwort, in der die Offsetadresse des Auftrags eingetragen ist und das BUSY-Bit gesetzt ist. Wenn der CANsync-Master das nächste Mal das Upload-Telegramm wiederholt und der CANsync-Slave den Auftrag inzwischen bearbeitet hat, dann antwortet er mit dem angeforderten Datenblock und das BUSY-Bit ist auf Null gesetzt.

Im letzten Telegramm ist MODE auf 11 gesetzt. Der CANsync-Slave überprüft, ob er auch am Ende des eingestellten Datenbereich angekommen ist und sendet den letzten Datenblock immer mit 6 Datenbytes. Wenn der zuladende Speicherbereich nicht mehr so viele Daten umfaßt wird mit irrelevanten Daten aufgefüllt. Falls der CANsync-Slave nicht am Ende angekommen ist, antwortet er mit gesetztem ERR-Bit und einem Fehlercode.

Auch wenn der Upload-Auftrag nicht bearbeitet werden kann, oder der CANsync-Slave eine Lücke in den angeforderten Offsetadressen feststellt, setzt der CANsync-Slave das ERR-Bit und gibt in den Datenbytes einen Fehlercode an. Der CANsync-Master kann gegebenenfalls das ausgefallene Telegramm wiederholen oder er bricht den Upload ab, in dem er MODE auf 01 setzt und als Basisadresse und Blocklänge 0 einträgt.

Auftrag:

<IDENTIFIER><CONTROL><OFFSET_L>

<IDENTIFIER>	::=	1010NNNNNNN	NNNNNNN = CANsync-Slave-Nummer
<CONTROL>	::=	<UD><U/D><MODE><OFFSET_H>	CAN-DB 0
<UD>	::=	<Bit7>	1 = Kennung: Up/Download-Auftrag
<U/D>	::=	<Bit6>	0 = Upload
<MODE>	::=	<Bit5..4>	10 = Mittelstück
			11 = letzter Block
<OFFSET_H>	::=	<Bit3..0>	Bit11..8 der Offset-Adresse
<OFFSET_L>	::=	<Byte>	Bit7..0 der Offset-Adresse = CAN-DB 1

Antwort:

<IDENTIFIER><STATUS><OFFSET_L><DATA>

<IDENTIFIER>	::=	1011NNNNNNN	NNNNNNN = CANsync-Slave-Nummer
<STATUS>	::=	<UD><BUSY><ERR><FREI><OFFSET_H>	CAN-DB 0
<UD>	::=	<Bit7>	1 = Kennung: Up/Download-Antwort
<BUSY>	::=	<Bit6>	0 = Auftrag fertig,
			1 = Auftrag wird bearbeitet
<ERR>	::=	<Bit5>	0 = kein Fehler, 1 = Fehler
<FREI>	::=	<Bit4>	noch nicht belegt
<OFFSET_H>	::=	<Bit3..0>	Bit11..8 der Offset-Adresse
<OFFSET_L>	::=	<Byte>	Bit7..0 der Offset-Adresse = CAN-DB 1
<DATA>	::=	<DATEN> | <ERR_CODE>	
<DATEN>	::=	<Word><Word><Word>	6 Byte Daten = CAN-DB 2..7
<ERR_CODE>	::=	<Word>	Fehlercode = CAN-DB 2..3
			CAN-DB: CAN-Daten-Byte

Laufender Download & Ende eines Downloads

Der Downloadvorgang besteht aus aufeinanderfolgenden Download-Telegrammen in denen der CANsync-Master von der in der Initialisierung eingestellten Startadresse ab, aufeinanderfolgende Datenblökke sendet. Die Offsetadresse steigt fortlaufend als Byteadresse. Das heißt das erste Telegramm beginnt mit der Offsetadresse 0, das zweite sendet die Daten mit der Offsetadresse 6 an, usw.

Der CANsync-Slave muß innerhalb der vorgegeben Antwortzeit t_{RSPTO} antworten. Falls er bis dahin noch nicht den Auftrag beenden kann, antwortet er mit der Download-Antwort, in der die Offsetadresse des Auftrags eingetragen ist und das BUSY-Bit gesetzt ist. Wenn der CANsync-Master das nächste Mal das Download-Telegramm wiederholt und der CANsync-Slave den Auftrag inzwischen bearbeitet hat, dann antwortet er mit dem Antwort-Telegramm, in dem das BUSY-Bit auf Null gesetzt ist.

Im letzten Telegramm ist MODE auf 11 gesetzt und es enthält auch 6 Datenbytes. Der CANsync-Slave darf jedoch nur die Datenbytes übernehmen, die der vorher eingestellten Downloadlänge entsprechen. Falls der CANsync-Slave noch nicht am Ende angekommen ist, antwortet er mit gesetztem ERR-Bit und einem Fehlercode.

Auch wenn der Download-Auftrag nicht bearbeitet werden kann, oder der CANsync-Slave eine Lücke in den gesendeten Offsetadressen feststellt, setzt der CANsync-Slave das ERR-Bit und gibt in den Datenbytes einen Fehlercode an. Der CANsync-Master kann gegebenenfalls das ausgefallene Telegramm wiederholen oder er bricht den Download ab, in dem er MODE auf 01 setzt und als Basisadresse und Blocklänge 0 einträgt.

Auftrag:
<IDENTIFIER><CONTROL><OFFSET_L><DATA>

<IDENTIFIER>	::=	1010NNNNNNN	NNNNNNN = CANsync-Slave-Nummer
<CONTROL>	::=	<UD><U/D><MODE><OFFSET_H>	CAN-DB 0
<UD>	::=	<Bit7>	1 = Kennung: Up/Download-Auftrag
<U/D>	::=	<Bit6>	1 = Download
<MODE>	::=	<Bit5..4>	10 = Mittelstück
			11 = letzter Block
<OFFSET_H>	::=	<Bit3..0>	Bit11..8 der Offset-Adresse
<OFFSET_L>	::=	<Byte>	Bit7..0 der Offset-Adresse = CAN-DB 1
<DATA>	::=	<Word><Word><Word>	6 Byte Nutzdaten = CAN-DB 2..7

Antwort:
<IDENTIFIER><STATUS><OFFSET_L><DATA>

<IDENTIFIER>	::=	1011NNNNNNN	NNNNNNN = CANsync-Slave-Nummer
<STATUS>	::=	<UD><BUSY><ERR><FREI><ERR_CODE_H>	CAN-DB 0
<UD>	::=	<Bit7>	1 = Kennung: Up/Download
<BUSY>	::=	<Bit6>	0 = Auftrag fertig,
			1 = Auftrag wird bearbeitet
<ERR>	::=	<Bit5>	0 = kein Fehler, 1 = Fehler
<FREI>	::=	<Bit4>	noch nicht belegt
<OFFSET_H>	::=	<Bit3..0>	Bit11..8 der Offset-Adresse
<OFFSET_L>	::=	<Byte>	Bit7..0 der Offset-Adresse = CAN-DB 1
<DATA>	::=	<0_BYTE> | <ERR_CODE>	
<0_BYTE>	::=	keine Daten-Bytes, wenn fehlerfrei	
<ERR_CODE>	::=	<Word>	Bit15..0 des Fehlercodes = CAN-DB 2..3
			CAN-DB: CAN-Daten-Byte

7.2.2 Registerstruktur und Funktion des Ωmega CANsync-Master

Im folgenden wird die Registerstruktur des Kommunikations-RAM im Omega-CANsync-Master erläutert.

Um im PROPROG wt II Projekt auf die Register des Kommunikations-RAM zugreifen zu können, sind Datentypen definiert, die die Registerstruktur abbilden. Mit diesen Datentypen werden Variablen deklariert, die auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Anschließend ist es möglich über die Strukturelemente der deklarierten Variablen auf die Register des Kommunikations-RAM zuzugreifen.

Bei der Initialisierung der CANsync-Master-Anschaltung haben die Register im Kommunikations-RAM eine andere Bedeutung als nach der Initialisierung, im zyklischen Betrieb.

Deshalb gibt es für die Initialisierung die Struktur

CANsync_INIT_BMSTRUCT

und für den zyklischen Betrieb die Struktur

CANsync_MA_CTRL_BMSTRUCT

Diese Strukturen sind ab der Bibliothek BM_TYPES_20bd00 definiert. Nachdem die Bibliothek BM_TYPES_20bd00 im Projekt eingebunden ist, stehen die Datentypen zur Verfügung.

Diese Strukturen enthalten

- 8-Bit-Elemente,
- 16-Bit-Elemente,
- 32-Bit-Elemente,
- Strukturen aus den o.g. Elementen
- Felder (ARRAY) und Strukturen aus den o.g. Elementen und Strukturen

Den in einer Struktur verwendeten Datentypen (8-, 16-, 32-Bit-Elemente, Strukturen und Feldern) wurden Kurzbezeichnungen vorangefügt. Dies dient der Übersichtlichkeit bei der Verwendung der Strukturen in der Programmierung.

Datentyp	Kurzbezeichnung	Anzahl der Bits
BYTE	b	8
WORD	w	16
DWORD (double word)	d	32
SINT (short integer)	si	8
DINT (double integer)	di	32
USINT (unsigned short integer)	us	8
UINT (unsigned integer)	u	16
UDINT (unsigned double integer)	ud	32
STRUCT	_ (underline)	-
ARRAY	a	-

Weitere, nicht in den Strukturen verwendete Datentypen sind:

Datentyp	Kurzbezeichnung	Anzahl der Bits
BOOL (bit)	x	1
TIME	t	-

Erläuterung zur Deklaration der globalen Variablen

Für die Initialisierung wird eine globale Variable vom Datentyp CANsync_INIT_BMSTRUCT angelegt. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) im Ωmega Drive-Line II,

```
_CANsync_INIT_MA        AT     %MB3.200000 : CANsync_INIT_BMSTRUCT;
```

dabei ist:

`_CANsync_INIT_MA`	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
`CANsync_INIT_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 am Ωmega Drive-Line II.

Für den zyklischen Betrieb wird eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angelegt. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) im Ωmega Drive-Line II,

```
_CANsync_CTRL_MA        AT     %MB3.200000 : CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 am Ωmega Drive-Line II.

HINWEIS

In den nachfolgenden Tabellen wird der Variablenname durch * ersetzt.

Auf das Register `*.w_CPU_CONTROL` greift man demzufolge über

`_CANsync_INIT_MA.w_CPU_CONTROL` zu,

auf `*.w_OPTION_STATUS` greift man über

`_CANsync_INIT_MA.w_OPTION_STATUS` zu.

Dabei ist:

_CANsync_INIT_MA	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
w_CPU_CONTROL	das Steuerregister der CANsync-Anschaltung mit der Datentypkurzbezeichnung „w“ für WORD

Die Register *.w_CPU_CONTROL und *.w_OPTION_STATUS können auch über die Struktur für den zyklischen Betrieb angesprochen werden. Der Zugriff ist dann über

_CANsync_CTRL_MA.w_CPU_CONTROL und

_CANsync_CTRL_MA.w_OPTION_STATUS möglich.

Dabei ist

_CANsync_CTRL_MA	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
w_OPTION_STATUS	das Statusregister der CANsync-Anschaltung mit der Datentypkurzbezeichnung „w“ für WORD

Beispiel für den Zugriff auf ein Element eines Feldes, das in der Struktur verwendet wird:

laut Tabelle: *.a_WR_VALUE[3]

Zugriff: _CANsync_CTRL_MA.a_WR_VALUE[3]

Dabei ist

_CANsync_CTRL_MA	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
a_WR_VALUE[3]	das Register für den Sollwert 3 mit der Datentypkurzbezeichnung „a“ für ARRAY. Der Datentyp der Elemente des Feldes (der Sollwerte) wird der entsprechenden Tabelle und der Beschreibung entnommen.

Beispiel für den Zugriff auf ein Element eines zweidimensionalen Feldes, das in der Struktur verwendet wird:

laut Tabelle: *.a_RD_VALUE[5][7]

Zugriff: _CANsync_CTRL_MA.a_RD_VALUE[5][7]

Dabei ist

_CANsync_CTRL_MA	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
a_RD_VALUE[5][7]	das Register für den Istwert 7 des CANsync-Slaves 5 mit der Datentypkurzbezeichnung „a“ für ARRAY. Der Datentyp der Elemente des Feldes (der Sollwerte) wird der entsprechenden Tabelle und der Beschreibung entnommen.

Beispiel für den Zugriff auf ein Element einer (Sub-) Struktur, die selbst Element eines Feldes ist, das in der Struktur verwendet wird:

laut Tabelle: `*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3`

Zugriff: `_CANsync_CTRL_MA.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3`

Dabei ist

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
`a_CFG_RDC_WORD[31]`	das Feld mit den Konfigurierungsdaten für das Mapping der Worte der Istwert-Telegramme des CANsync-Slaves 31 mit der Datentypkurzbezeichnung „a“ für ARRAY
`b_CFG_RDC2_WORD3`	das Register für die Konfigurierungsdaten für das Mapping des 3. Wortes des Istwert-Telegramms 2 (des CANsync-Slaves 31) mit der Datentypkurzbezeichnung „b“ für BYTE

Beispiel für den Zugriff auf ein Element einer (Sub-) Struktur, die in der Struktur verwendet wird:

laut Tabelle: `*._CFG_WRC_WORD.b_CFG_WRC1_WORD0`

Zugriff: `_CANsync_CTRL_MA._CFG_WRC_WORD.b_CFG_WRC1_WORD0`

Dabei ist

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
`_CFG_WRC_WORD`	die Struktur mit den Konfigurierungsdaten für das Mapping der Worte der Sollwert-Telegramme mit der Datentypkurzbezeichnung „_“ für STRUCT
`b_CFG_WRC1_WORD0`	das Register für die Konfigurierungsdaten für das Mapping des 0. Wortes des Sollwert-Telegramms 1 mit der Datentypkurzbezeichnung „b“ für BYTE

Allgemeine Register der CANsync-Anschaltung

Register	Inhalt
*.w_CANsync_STATUS	CANsync-Status
*.w_OMEGA_NR	über Dip-Schalter eingestellte Ωmega-Nummer
*.i_SW1_NR	Karten-Softwarenummer
*.i_SW1_RELEASE	Softwarestand inkompatibel und kompatibel

CANsync-Status

Bei jedem Durchlaufen des CANsync-Prozessorzyklusses wird der CANsync-Status auf *.w_CANsync_STATUS ausgegeben.

Bedeutung:

Bit-Nr.	Bedeutung (Bit = TRUE)
0	reserviert
1	Overrun: eine CANsync-Nachricht konnte nicht empfangen werden
2	CANsync-Sendepuffer ist frei
3	CANsync-Sendeauftrag wurde erfolgreich durchgeführt
4	es wird gerade eine CANsync-Meldung empfangen
5	es wird gerade eine CANsync-Meldung gesendet
6	Fehler vorhanden (Warnung)
7	CANsync-Knoten ist deaktiviert (BUS-off)
8-15	reserviert

Ωmega-Nummer

Im Register *.w_OMEGA_NR wird die über die DIP-Schalter (S33) eingestellte Ωmega-Nummer angezeigt. Bei einer CANsync-Master-Anschaltung ist diese Nummer ohne Bedeutung.

Softwarenummer und Softwarestand

Im Register *.i_SW1_NR wird die Nummer der CANsync-Software auf dem Ωmega Drive-Line II angezeigt.

Im Register *.i_SW1_RELEASE wird der inkompatible und der kompatible Stand der CANsync-Software auf dem Ωmega Drive-Line II angezeigt.

Initialisierung

Für die Initialisierung wird eine globale Variable vom Datentyp CANsync_INIT_BMSTRUCT angelegt. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) im Ωmega Drive-Line II,

```
_CANsync_INIT_MA        AT     %MB3.200000 : CANsync_INIT_BMSTRUCT;
```

dabei ist:

`_CANsync_INIT_MA`	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
`CANsync_INIT_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 am Ωmega Drive-Line II.

Zusätzlich konfiguriert man für den Synchron-Betrieb:

Bedeutung		**Register**	**Wert**
Baudrate	z. B.: 500 kBit/s	*.b_BT_0 *.b_BT_1	16#00 16#1C
CANsync- Intervall	z. B.: 2 ms	*.b_TIME_PATTERN	16#02
Acceptance Code	alle Telegramme	*.b_AC	16#FF
Acceptance Mask		*.b_AM	16#FF
Output-Control	16#FA	*.b_OUTPUT_CONTROL	16#FA
Clock-Divider	16#07	*.b_CLOCK_DIVIDER	16#07
Slave/Master	Master	*.b_MA_SL_MODE	16#00
Slave-Typen	CANsync-Slave mit Slavenummer x nicht vorhanden / vorhanden	*.a_SL_TYP[x]	16#00 / 16#01

Für jede am CANsync-Bus vorhandene CANsync-Slave-Anschaltung (Slavenummer über Dip-Schalter eingestellt) gibt man den CANsync-Slavetyp an. Zur Zeit gibt es nur den CANsync-Slavetyp 16#01. Wenn der Wert auf 16#00 gestellt wird, bedeutet dieses, daß kein CANsync-Slave mit dieser Slavenummer vorhanden ist, bzw. vom CANsync-Master erwartet wird.

Die Betriebsarteneinstellung erfolgt über das Register *.w_CPU_CONTROL. Die Anzeige der gerade aktiven Betriebsart erfolgt im Register *.w_OPTION_STATUS. Auch nach dem erfolgreichen Starten einer Betriebsart kann die Betriebsart wieder geändert werden.

Register	Inhalt
*.w_CPU_CONTROL	Steuerregister CANsync-Anschaltung
*.w_OPTION_STATUS	Statusregister CANsync-Anschaltung

(* entspricht zum Beispiel bei der Initialisierung _CANsync_INIT_MA, nach der Initialisierung zum Beispiel _CANsync_CTRL_MA)

Steuerregister CANsync-Anschaltung	Bedeutung
16#0000	Neuanlauf
16#0001	Handshake
16#0002	Initialisierungsdaten übernehmen
16#0012	reserviert
16#0013	reserviert
16#0020	Synchron-Betrieb starten
16#0040	Aktiven Betrieb freigeben
16#0080	(Bit 7 = TRUE) Reset CAN-Controller

Statusregister CANsync-Anschaltung	Bedeutung
16#0001	Anlauf
16#0002	Warten auf Initialisierungsdaten übernehmen
16#0003	Warten auf Start
16#0011	reserviert
16#0012	reserviert
16#0013	*Synchron-Betrieb CANsync-Slave wird eingerichtet*
16#0020	*Synchron-Betrieb CANsync-Slave ist aktiv*
16#0041	reserviert
16#0042	reserviert
16#0043	Synchron-Betrieb CANsync-Master wird eingerichtet
16#0080	Synchron-Betrieb CANsync-Master ist aktiv

Die Initialisierung wird mit den Kommandos 16#0000, 16#0001 und 16#0002 auf *.w_CPU_CONTROL durchgeführt. Damit wird der Einrichtbetrieb gestartet. Als nächstes wird der Slavestatus der initialisierten CANsync-Slaves gemeldet (siehe “Kommando- und Antwortkanal” auf Seite 160). Wenn alle CANsync-Slaves sich gemeldet haben und den Zustand „synchronisiert“ haben, muß der aktive Betrieb freigegeben werden. Dies geschieht durch Setzen des Bit 6 in *.w_CPU_CONTROL (*.w_CPU_CONTROL = 16#0040).

Die Freigabe darf auch dann erfolgen, wenn sich nicht alle CANsync-Slaves gemeldet haben, dies aber von der Applikation verwaltet werden kann.

Wenn man das Bit 7 von *.w_CPU_CONTROL setzt (*.w_CPU_CONTROL = 16#0080), wird der CAN-Controller zurückgesetzt und das Bit wieder gelöscht. Dadurch kann der BUS-OFF-Zustand des CAN-Controllers zurückgesetzt werden und wieder CANsync-Telegramme gesendet und empfangen werden. Die Anzeige über den BUS-OFF-Zustand erfolgt in *.w_CANsync_STATUS (siehe “Allgemeine Register der CANsync-Anschaltung” auf Seite 147).

In der folgenden Tabelle sind die Register angegeben, die in der Initialisierung bedient werden.

Register	Inhalt
*.b_MA_SL_MODE	Betriebsart: Master / Slave (SYNC-OUT / SYNC-IN)
*.b_AC	Acceptance Code des CANsync-Controllers
*.b_AM	Acceptance Mask des CANsync-Controllers
*.b_BT_0	Bit-Timing-Register 0 des CANsync-Controllers
*.b_BT_1	Bit-Timing-Register 1 des CANsync-Controllers
*.b_OUTPUT_CONTROL	Output-Control-Register des CANsync-Controllers
*.b_CLOCK_DIVIDER	Clock-Divider des CANsync-Controllers
*.b_TIME_PATTERN	CANsync-Intervall in ms
*.a_SL_TYP[0]	CANsync-Slave-Typ 0
*.a_SL_TYP[1]	CANsync-Slave-Typ 1
...	...
*.a_SL_TYP[31]	CANsync-Slave-Typ 31

(* entspricht zum Beispiel bei der Initialisierung der Struktur _CANsync_INIT_MA)

Sollwerte

Das Senden der Sollwerte erfolgt im CANsync Event-Task.

Register	Inhalt
*.b_CTRLREG_WRC1	Kontrollregister Sollwertkanal 1
*.b_CTRLREG_WRC2	Kontrollregister Sollwertkanal 2
*.b_CTRLREG_WRC3	reserviert
*.b_CTRLREG_WRC4	reserviert
*.b_CTRLREG_WRC5	reserviert
*.b_CTRLREG_WRC6	reserviert
*.b_CTRLREG_WRC7	reserviert
*.b_CTRLREG_WRC8	reserviert

(* entspricht nach der Initialisierung zum Beispiel der Variablen _CANsync_CTRL_MA)

Im Kontrollregister wird mit 16#05 gekennzeichnet, daß neue Sollwerte für das jeweilige Sollwert-Telegramm (auch Sollwertkanal (SWK oder WRC)) eingetragen wurden und das Telegramm gesendet wird. Die CANsync-Anschaltung quittiert den Befehl mit 16#04.

Die Konfigurierung der Sollwert-Telegramme erfolgt in folgenden Registern.

Register	Inhalt
*._CFG_WRC_WORD.b_CFG_WRC1_WORD0	Konfigurierung Sollwert-Telegramm 1 Wort 0
*._CFG_WRC_WORD.b_CFG_WRC1_WORD1	Konfigurierung Sollwert-Telegramm 1 Wort 1
*._CFG_WRC_WORD.b_CFG_WRC1_WORD2	Konfigurierung Sollwert-Telegramm 1 Wort 2
*._CFG_WRC_WORD.b_CFG_WRC1_WORD3	Konfigurierung Sollwert-Telegramm 1 Wort 3
*._CFG_WRC_WORD.b_CFG_WRC2_WORD0	Konfigurierung Sollwert-Telegramm 2 Wort 0
*._CFG_WRC_WORD.b_CFG_WRC2_WORD1	Konfigurierung Sollwert-Telegramm 2 Wort 1
*._CFG_WRC_WORD.b_CFG_WRC2_WORD2	Konfigurierung Sollwert-Telegramm 2 Wort 2
*._CFG_WRC_WORD.b_CFG_WRC2_WORD3	Konfigurierung Sollwert-Telegramm 2 Wort 3
*._CFG_WRC_WORD.b_CFG_WRC3_WORD0	reserviert
*._CFG_WRC_WORD.b_CFG_WRC3_WORD1	reserviert
*._CFG_WRC_WORD.b_CFG_WRC3_WORD2	reserviert
*._CFG_WRC_WORD.b_CFG_WRC3_WORD3	reserviert
...	...
*._CFG_WRC_WORD.b_CFG_WRC8_WORD0	reserviert
*._CFG_WRC_WORD.b_CFG_WRC8_WORD1	reserviert
*._CFG_WRC_WORD.b_CFG_WRC8_WORD2	reserviert
*._CFG_WRC_WORD.b_CFG_WRC8_WORD3	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Mit der Konfigurierung gibt man an, welcher Sollwert an welcher Stelle (..._WORD0, ..., ..._WORD3) in ein Sollwerttelegramm (..._WRC1_..., ..._WRC2_...) eingetragen wird.

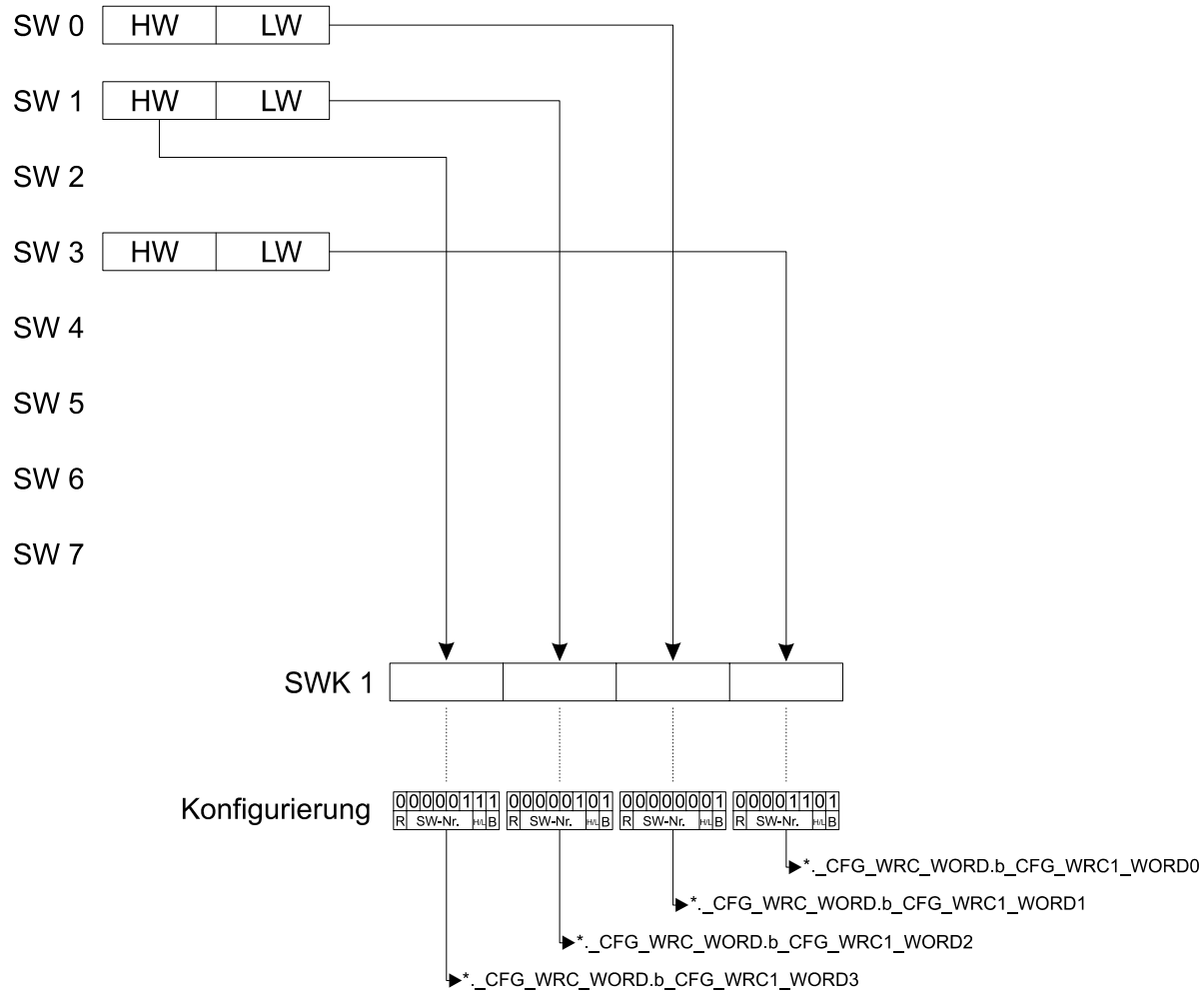
Bedeutung

Bit 0	**Belegt** Wenn = 1, wird dieses Wort des Telegramms verwendet
Bit 1	**Highword/Lowword** Wenn = 1, wird das Highword des Sollwertes eingetragen
Bit 2	**Sollwertnummer** Nummer des Sollwertes 0 bis 31
Bit 3	
Bit 4	
Bit 5	
Bit 6	
Bit 7	reserviert, muß auf Null gesetzt werden

HINWEIS

Ein Wort der Nachricht wird nur dann nicht verwendet, wenn Sollwertnummer = 0, Highword/Lowword = 0 und Belegt = 0 gesetzt wird.

Beispiel:



*._CFG_WRC_WORD.b_CFG_WRC1_WORD0 = 16#0D

0	0	0	0	1	1	0	1
R	SW-Nr.					H/L	B

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Mit dieser Einstellung wird das Lowword des Sollwertes 3 im Sollwert-Telegramm 1 im Wort 0 eingetragen.

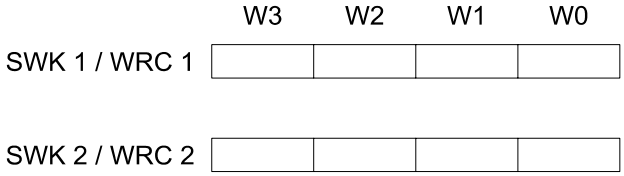
Register	Inhalt
*.a_WR_VALUE[0]	Sollwert 0
*.a_WR_VALUE[1[	Sollwert 1
*.a_WR_VALUE[2]	Sollwert 2
*.a_WR_VALUE[3]	Sollwert 3
*.a_WR_VALUE[4]	Sollwert 4
*.a_WR_VALUE[5[	Sollwert 5
*.a_WR_VALUE[6]	Sollwert 6
*.a_WR_VALUE[7]	Sollwert 7
*.a_WR_VALUE[8]	reserviert
...	...
*.a_WR_VALUE[31]	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Die Sollwerte können Wort- oder Doppelwort-Sollwerte sein.

Erläuterung zur Benutzung der Sollwertkanäle

Im Synchron-Betrieb stehen zwei Sollwertkanäle (Kanal 1 und 2; auch WRC1 und WRC2) zur Verfügung. In den Sollwertkanälen 1 und 2 werden die Sollwert-Telegramme 1 und 2 gesendet.



Beide Sollwert-Telegramme bestehen aus jeweils vier Worten (W0 bis W3). Nach der CAN-Initialisierung muß zumindest einmal angegeben werden, wie die Belegung dieser Worte ist. Dies kann im Initialisierungsprogramm erfolgen. Dazu trägt man im Bereich
*._CFG_WRC_WORD.b_CFG_WRC1_WORD0 bis *._CFG_WRC_WORD.b_CFG_WRC2_WORD3
für jedes Wort der Sollwert-Telegramme ein, welcher Sollwert an dieser Stelle übertragen wird. Die Sollwertnummern sind im Bereich von 0 bis 7 gültig. Für Doppelwort-Sollwerte müssen zwei Worte im Telegramm verwendet werden.

Die Konfigurierung kann auch während des aktiven Betriebs geändert werden. Die Änderung wird spätestens im nächsten CANsync-Intervall übernommen. Zu Beginn des CANsync-Intervalls werden die Konfigurierungsdaten von der CANsync-Anschaltung gelesen.

Die Sollwertgenerierung wird im Synchron-Betrieb in der CANsync Event-Task erfolgen. Die Sollwerte für das Sollwert-Telegramm 1 müssen bis 490 µs nach dem Start der CANsync Event-Task im Kommunikations-RAM als Sollwert (*.a_WR_VALUE[0] bis *.a_WR_VALUE[7]) eingetragen werden, weil dann die CANsync-Anschaltung die Generierung des Sollwert-Telegramms beginnt. Als Kennzeichnung, daß neue Sollwerte eingetragen wurden, muß im entsprechenden Kontrollregister (*.b_CTRLREG_WRC1 oder *.b_CTRLREG_WRC2) 16#05 eingetragen werden. Dies ist die Freigabe für die CANsync-Anschaltung, daß die Sollwerte gelesen werden dürfen und das Sollwert-Telegramm erzeugt wird. Falls bis

490 µs nach dem Start der CANsync Event-Task diese Freigabe nicht erfolgt, fällt in diesem CANsync-Intervall das Sollwert-Telegramm 1 aus. Als Quittung für die Erzeugung des Sollwert-Telegramms wird im Kontrollregister 16#04 von der CANsync-Anschaltung eingetragen. Dadurch kann der Anwender überprüfen, ob die Sollwertgenerierung rechtzeitig fertig geworden ist oder nicht. Falls die Sollwertgenerierung länger dauert, - vom Start der CANsync Event-Task bis zur Ausführung des Applikationsprogramms vergehen ca. 80 µs -, muß die Sollwertgenerierung immer für den nächsten Start der CANsync Event-Task erfolgen und zu Beginn der neuen CANsync Event-Task müssen die vorberechneten Sollwerte nur noch an die entsprechenden Stellen im Kommunikations-RAM kopiert werden.

Der Zeitpunkt bis zu dem die Sollwerte für das Sollwert-Telegramm 2 eingetragen werden müssen verzögert sich um die Zeit zur Erzeugung des Sollwert-Telegramms 1. Die Zeitdauer hängt von der Zahl der einzutragenden Sollwertworte ab (mindestens 15 µs, höchstens 60 µs). Die Signalisierung im Kontrollregister erfolgt wie für das Sollwert-Telegramm 1.

Mit dem Sollwert-Telegramm wird das Istwert-Telegramm eines CANsync-Slaves angefordert. Die Nummer der CANsync-Slave-Anschaltung wird im Kontrollregister Istwertanforderung (*.b_CTRLREG_RD_ORDER_RDC1 oder *.b_CTRLREG_RD_ORDER_RDC2) eingetragen (weitere Beschreibung siehe “Istwerte” auf Seite 154). Das Sollwert-Telegramm 1 fordert das Istwert-Telegramm 1 an, usw.

Istwerte

Die Istwert-Telegramme werden durch eine bestimmte Information im Identifier der Sollwert-Telegramme angefordert (siehe “Sollwertkanal 1” auf Seite 136).

Register	Inhalt
*.b_MAX_SL_NR	maximale Slavenummer (Bus-Adresse des CANsync-Slaves

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Sie gibt die größte Slavenummer des CANsync-Slaves für die automatische Istwertanforderung (siehe *.b_CTRLREG_RD_ORDER_RDC1 ff.) an.

Register	Inhalt
*.b_CTRLREG_RD_ORDER_RDC1	Kontrollregister Istwertanforderung Istwertkanal 1
*.b_CTRLREG_RD_ORDER_RDC2	Kontrollregister Istwertanforderung Istwertkanal 2
*.b_CTRLREG_RD_ORDER_RDC3	reserviert
*.b_CTRLREG_RD_ORDER_RDC4	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Im Kontrollregister kann die Slavenummer des CANsync-Slaves angegeben werden, der im entsprechenden Istwert-Telegramm seine Istwerte melden soll. Wenn im Kontrollregister 16#80 eingegeben wird, wird die Slavenummer in jedem Zyklus um eins erhöht, bis die maximale Slavenummer (*.b_MAX_SL_NR) erreicht ist. Dann wird wieder mit der Slavenummer 0 gestartet.

Register	Inhalt
*.b_STATREG_RDC1	Statusregister Istwertkanal 1
*.b_STATREG_RDC2	Statusregister Istwertkanal 2
*.b_STATREG_RDC3	reserviert
*.b_STATREG_RDC4	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Im Statusregister wird die Slavenummer des CANsync-Slaves eingetragen, von dem ein Istwert-Telegramm empfangen wurde.

Register	Inhalt
*.a_STATREG_RDC[0].b_STATREG_RDC1	Istwertquittung Istwertkanal 1 Slave 0
*.a_STATREG_RDC[0].b_STATREG_RDC2	Istwertquittung Istwertkanal 2 Slave 0
*.a_STATREG_RDC[0].b_STATREG_RDC3	reserviert
*.a_STATREG_RDC[0].b_STATREG_RDC4	reserviert
*.a_STATREG_RDC[1].b_STATREG_RDC1	Istwertquittung Istwertkanal 1 Slave 1
*.a_STATREG_RDC[1].b_STATREG_RDC2	Istwertquittung Istwertkanal 2 Slave 1
*.a_STATREG_RDC[1].b_STATREG_RDC3	reserviert
*.a_STATREG_RDC[1].b_STATREG_RDC4	reserviert
*.a_STATREG_RDC[2].b_STATREG_RDC1	Istwertquittung Istwertkanal 1 Slave 2
*.a_STATREG_RDC[2].b_STATREG_RDC2	Istwertquittung Istwertkanal 2 Slave 2
*.a_STATREG_RDC[2].b_STATREG_RDC3	reserviert
*.a_STATREG_RDC[2].b_STATREG_RDC4	reserviert
...	...
*.a_STATREG_RDC[31].b_STATREG_RDC1	Istwertquittung Istwertkanal 1 Slave 31
*.a_STATREG_RDC[31].b_STATREG_RDC2	Istwertquittung Istwertkanal 2 Slave 31
*.a_STATREG_RDC[31].b_STATREG_RDC3	reserviert
*.a_STATREG_RDC[31].b_STATREG_RDC4	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Wenn das entsprechende Istwert-Telegramm für einen CANsync-Slave angekommen ist, wird eine 16#02 in das Istwertquittungsregister eingetragen.

Die Konfigurierung der Istwert-Telegramme erfolgt in folgenden Registern.

Register	Inhalt
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD0	Konfigurierung Istwert-Telegramm 1 Wort 0 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD1	Konfigurierung Istwert-Telegramm 1 Wort 1 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD2	Konfigurierung Istwert-Telegramm 1 Wort 2 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD3	Konfigurierung Istwert-Telegramm 1 Wort 3 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD0	Konfigurierung Istwert-Telegramm 2 Wort 0 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD1	Konfigurierung Istwert-Telegramm 2 Wort 1 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD2	Konfigurierung Istwert-Telegramm 2 Wort 2 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD3	Konfigurierung Istwert-Telegramm 2 Wort 3 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD0	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD1	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD2	reserviert

Register	Inhalt
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD3	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD0	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD1	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD2	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD3	reserviert
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD0	Konfigurierung Istwert-Telegramm 1 Wort 0 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD1	Konfigurierung Istwert-Telegramm 1 Wort 1 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD2	Konfigurierung Istwert-Telegramm 1 Wort 2 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD3	Konfigurierung Istwert-Telegramm 1 Wort 3 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD0	Konfigurierung Istwert-Telegramm 2 Wort 0 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD1	Konfigurierung Istwert-Telegramm 2 Wort 1 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD2	Konfigurierung Istwert-Telegramm 2 Wort 2 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD3	Konfigurierung Istwert-Telegramm 2 Wort 3 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC3_WORD0	reserviert
*.a_CFG_RDC_WORD[1].b_CFG_RDC3_WORD1	reserviert
*.a_CFG_RDC_WORD[1].b_CFG_RDC3_WORD2	reserviert
*.a_CFG_RDC_WORD[1].b_CFG_RDC3_WORD3	reserviert
*.a_CFG_RDC_WORD[1].b_CFG_RDC4_WORD0	reserviert
*.a_CFG_RDC_WORD[1].b_CFG_RDC4_WORD1	reserviert
*.a_CFG_RDC_WORD[1].b_CFG_RDC4_WORD2	reserviert
*.a_CFG_RDC_WORD[1].b_CFG_RDC4_WORD3	reserviert
...	...
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD0	Konfigurierung Istwert-Telegramm 1 Wort 0 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD1	Konfigurierung Istwert-Telegramm 1 Wort 1 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD2	Konfigurierung Istwert-Telegramm 1 Wort 2 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD3	Konfigurierung Istwert-Telegramm 1 Wort 3 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD0	Konfigurierung Istwert-Telegramm 2 Wort 0 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD1	Konfigurierung Istwert-Telegramm 2 Wort 1 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD2	Konfigurierung Istwert-Telegramm 2 Wort 2 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3	Konfigurierung Istwert-Telegramm 2 Wort 3 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD0	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD1	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD2	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD3	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD0	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD1	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD2	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD3	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Mit der Konfigurierung gibt man an, welcher Istwert an welcher Stelle im Istwert-Telegramm übertragen wird.

Bedeutung

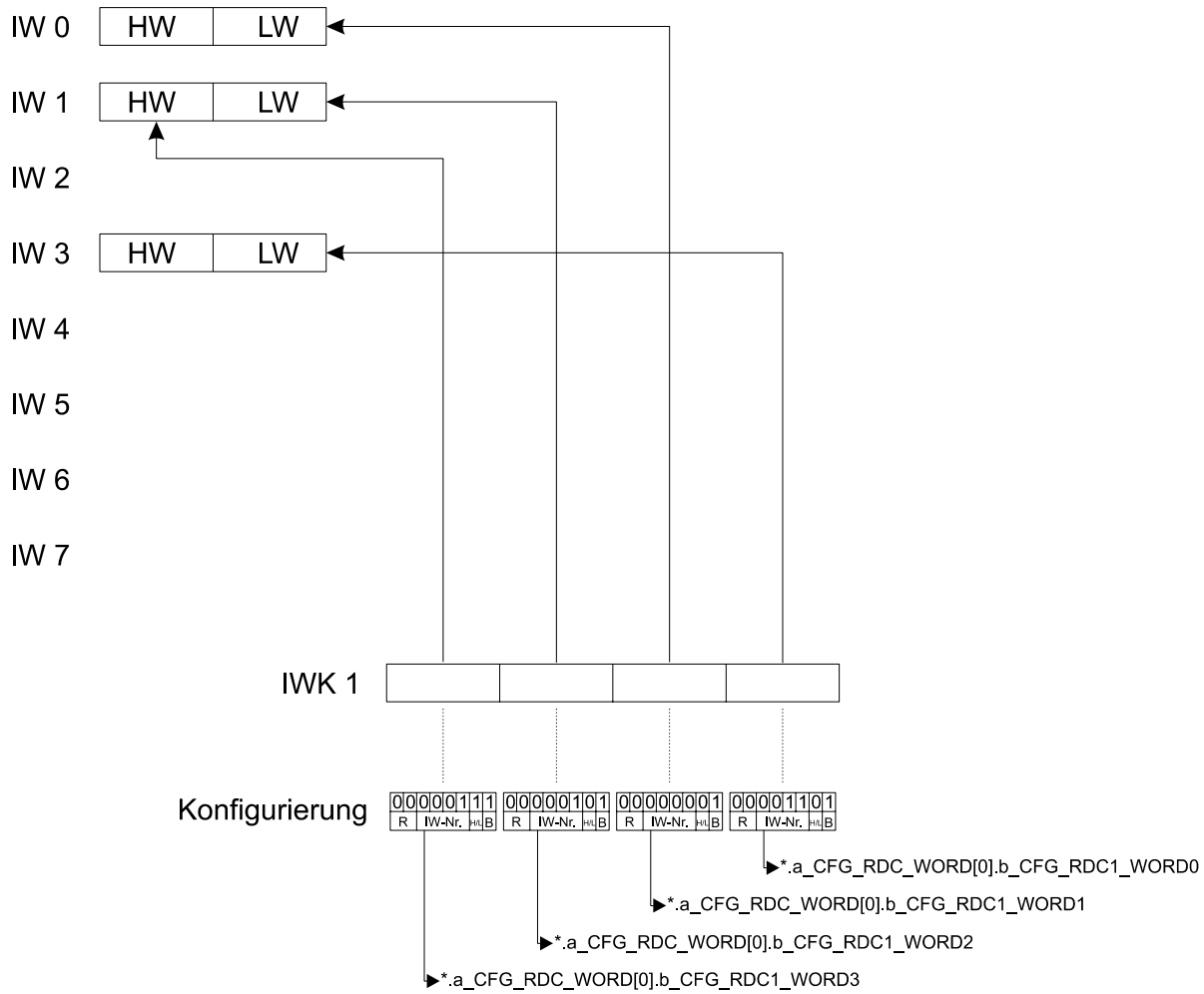
Bit 0	**Belegt** Wenn = 1, wird dieses Wort des Telegramms verwendet
Bit 1	**Highword/Lowword** Wenn = 1, wird das Highword des Istwertes gelesen

Bit 2	**Istwertnummer** Nummer des Istwertes 0 bis 15
Bit 3	
Bit 4	
Bit 5	
Bit 6	reserviert
Bit 7	

HINWEIS

Ein Wort des Telegramms wird nur dann nicht verwendet, wenn Istwertnummer = 0, Highword/Lowword = 0 und Belegt = 0 gesetzt wird.

Beispiel:



*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD0 = 16#0D

0	0	0	0	1	1	0	1
R		IW-Nr.				H/L	B

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Damit wird aus dem Istwert-Telegramm 1 vom CANsync-Slave 0 das Wort 0 gelesen und in das Low-word des Istwertes 3 geschrieben.

Register	Inhalt
*.a_RD_BMARRAY[0][0]	Istwert 0 Slave 0
*.a_RD_BMARRAY[0][1]	Istwert 1 Slave 0
*.a_RD_BMARRAY[0][2]	Istwert 2 Slave 0
*.a_RD_BMARRAY[0][3]	Istwert 3 Slave 0
*.a_RD_BMARRAY[0][4]	Istwert 4 Slave 0
*.a_RD_BMARRAY[0][5]	Istwert 5 Slave 0
*.a_RD_BMARRAY[0][6]	Istwert 6 Slave 0
*.a_RD_BMARRAY[0][7]	Istwert 7 Slave 0
*.a_RD_BMARRAY[0][8]	reserviert
...	...
*.a_RD_BMARRAY[0][15]	reserviert
*.a_RD_BMARRAY[1][0]	Istwert 0 Slave 1
*.a_RD_BMARRAY[1][1]	Istwert 1 Slave 1
*.a_RD_BMARRAY[1][2]	Istwert 2 Slave 1
*.a_RD_BMARRAY[1][3]	Istwert 3 Slave 1
*.a_RD_BMARRAY[1][4]	Istwert 4 Slave 1
*.a_RD_BMARRAY[1][5]	Istwert 5 Slave 1
*.a_RD_BMARRAY[1][6]	Istwert 6 Slave 1
*.a_RD_BMARRAY[1][7]	Istwert 7 Slave 1
*.a_RD_BMARRAY[1][8]	reserviert
...	...
*.a_RD_BMARRAY[1][15]	reserviert
...	...
*.a_RD_BMARRAY[31][0]	Istwert 0 Slave 31
*.a_RD_BMARRAY[31][1]	Istwert 1 Slave 31
*.a_RD_BMARRAY[31][2]	Istwert 2 Slave 31
*.a_RD_BMARRAY[31][3]	Istwert 3 Slave 31
*.a_RD_BMARRAY[31][4]	Istwert 4 Slave 31
*.a_RD_BMARRAY[31][5]	Istwert 5 Slave 31
*.a_RD_BMARRAY[31][6]	Istwert 6 Slave 31
*.a_RD_BMARRAY[31][7]	Istwert 7 Slave 31
*.a_RD_BMARRAY[31][8]	reserviert
...	...
*.a_RD_BMARRAY[31][15]	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_MA)

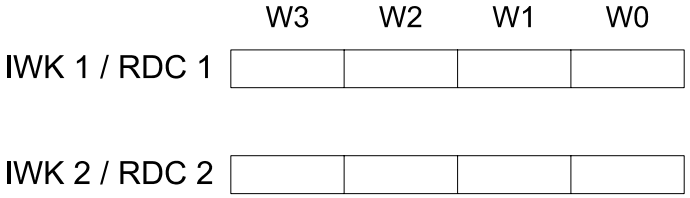
Die Istwerte können Wort- oder Doppelwort-Istwerte sein.

HINWEIS

Bei der Anzeige von _CANsync_CTRL_MA im PROPROG wt II Watch-Fenster kann zwischen den eckigen Klammern ein Punkt stehen.

Erläuterung zur Benutzung der Istwertkanäle

Im Synchron-Betrieb stehen zwei Istwertkanäle (Kanal 1 und 2; auch RDC1 und RDC2) zur Verfügung. In den Istwertkanälen 1 und 2 werden die Istwert-Telegramme 1 und 2 gesendet.



Beide Istwert-Telegramme bestehen aus jeweils vier Worten (W0 bis W3). Nach der CANsync-Initialisierung muß zumindest einmal angegeben werden, wie die Belegung dieser Worte ist. Dies kann im Initialisierungsprogramm erfolgen.
Dazu trägt man im Bereich *.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD0 bis *.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3 für jedes belegte Wort der Istwert-Telegramme (jedes CANsync-Slaves) ein, welcher Istwert an dieser Stelle übertragen wird. Istwertnummern sind im Bereich von 0 bis 7 gültig. Für Doppelwort-Istwerte müssen zwei Worte im Telegramm verwendet werden.

Die Konfigurierung kann auch während des aktiven Betriebs geändert werden. Die Änderung wird spätestens im nächsten CANsync-Intervall übernommen. (Die Konfigurierungsdaten werden von der CANsync-Anschaltung gelesen, wenn das entsprechende Istwert-Telegramm empfangen wird.)

Die Istwert-Telegramm-Auswertung erfolgt zu Beginn der CANsync Event-Task. Dazu gibt es zwei Auswertungsmethoden: Bei der ersten Methode fragt man die Statusregister der Istwertkanäle ab (*.b_STATREG_RDC1 oder *.b_STATREG_RDC2). In diesen Registern wird die Slavenummer des CANsync-Slaves eingetragen, von dem das entsprechende Istwert-Telegramm empfangen wurde. Dann kann man die Istwerte aus den Registern (*.a_RD_BMARRAY[0][0] bis *.a_RD_BMARRAY[31][7]) dieses CANsync-Slaves auslesen und das Statusregister auf z. B. 16#FF setzen, um die nächste Eintragung richtig erkennen zu können.

Bei der zweiten Methode fragt man direkt die Istwertquittung
(*.a_STATREG_RDC[0].b_STATREG_RDC1 bis *.a_STATREG_RDC[31].b_STATREG_RDC2) für einen CANsync-Slave ab. In diesem Quittungsregister wird der Empfang eines Istwert-Telegramms mit 16#02 gekennzeichnet. Dann kann man die Istwerte aus den Registern (*.a_RD_BMARRAY[0][0] bis *.a_RD_BMARRAY[31][7]) dieses CANsync-Slaves auslesen. Auch hier muß man das Statusregister anschließend mit einem anderen Wert beschreiben, um das Wiedereintragen der Quittung zu erkennen.

Die Zuordnung welche Istwerte gelesen werden dürfen, wenn ein Istwert-Telegramm empfangen wurde, muß entsprechend der Istwertkonfigurierung im Applikationsprogramm erfolgen.

Die Anforderung an einen CANsync-Slave, sein Istwert-Telegramm zu senden, erfolgt über das Sollwert-Telegramm.

Die Nummer des CANsync-Slaves wird im Kontrollregister Istwertanforderung
(*.b_CTRLREG_RD_ORDER_RDC1 und *.b_CTRLREG_RD_ORDER_RDC2) eingetragen. Das Sollwert-Telegramm 1 fordert das Istwert-Telegramm 1 an, usw.

Die Anforderungsnummer für das Istwert-Telegramm 1 und Istwert-Telegramm 2, usw. können gleich sein oder auch verschieden, d. h. Istwert-Telegramm 1 kann von CANsync-Slave x und Istwert-Telegramm 2 kann von CANsync-Slave y angefordert werden.

Für die Anforderungsnummer des CANsync-Slaves gibt es zwei Möglichkeiten: Bei der ersten Möglichkeit gibt man direkt die Slavenummer im Kontrollregister an. Solange dieses Register nicht verändert wird, wird in jedem CANsync-Intervall der gleiche CANsync-Slave angesprochen.

Wenn man als Anforderungsnummer 16#80 einträgt, wird die Slavenummer von der CANsync-Anschaltung automatisch in jedem CANsync-Intervall um eins erhöht, bis die maximale Slavenummer (*.b_MAX_SL_NR) erreicht ist. Dann beginnt die Abfrage wieder mit Slavenummer 0. Damit ist eine automatische Anforderung der Istwert-Telegramme aller vorhandenen CANsync-Slaves möglich. Wenn die Slavenummern der CANsync-Slaves mit Lücken vergeben werden, wird zwar auch eine Istwertanforderung für die nicht vorhandenen CANsync-Slaves erzeugt, dies führt jedoch zu keinen Funktionsproblemen.

Kommando- und Antwortkanal

Konfigurierung Kommandokanal

Die Konfigurierung der Kommandokanalbelegung erfolgt in folgenden Registern:

Register	Inhalt
*.b_SL_NR_CONTROLWORD	Slavenummer des CANsync-Slaves für Steuerwortkommando
*.b_SL_NR_ PARAMETER	Slavenummer des CANsync-Slaves für Parameterkommando
*.b_SL_NR_UPDOWN	Slavenummer des CANsync-Slaves für Up-/Downloadkommando

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Erläuterung zur Konfigurierung des Kommandokanals

Über die Konfigurierungsregister *.b_SL_NR_CONTROLWORD, *.b_SL_NR_PARAMETER und *.b_SL_NR_UPDOWN kann die Benutzung des Kommandokanals eingestellt werden. Die Normaleinstellung ist, daß alle drei Register auf 16#80 eingestellt werden. Dann werden die Slavenummern der CANsync-Slaves, für die gegebenenfalls Kommando-Telegramme erzeugt werden, zyklisch erhöht. Die maximale Slavenummer ist die gleiche wie für die Istwert-Anforderung (*.b_MAX_SL_NR).

Alternativ kann auch in jedem Register eine explizite Vorgabe der Slavenummer erfolgen. Dann wird nur geprüft, ob für diesen CANsync-Slave das entsprechende Kommando erzeugt werden soll.

In einem CANsync-Intervall kann nur ein Kommando gesendet werden. Das folgende Schema gibt an, welche Bereiche in welcher Reihenfolge im Synchron-Betrieb abgefragt werden, ob ein Auftrag für die Erzeugung eines Kommandos eingetragen ist. Die verschiedenen Kommandos sind in den nächsten Abschnitten erläutert.

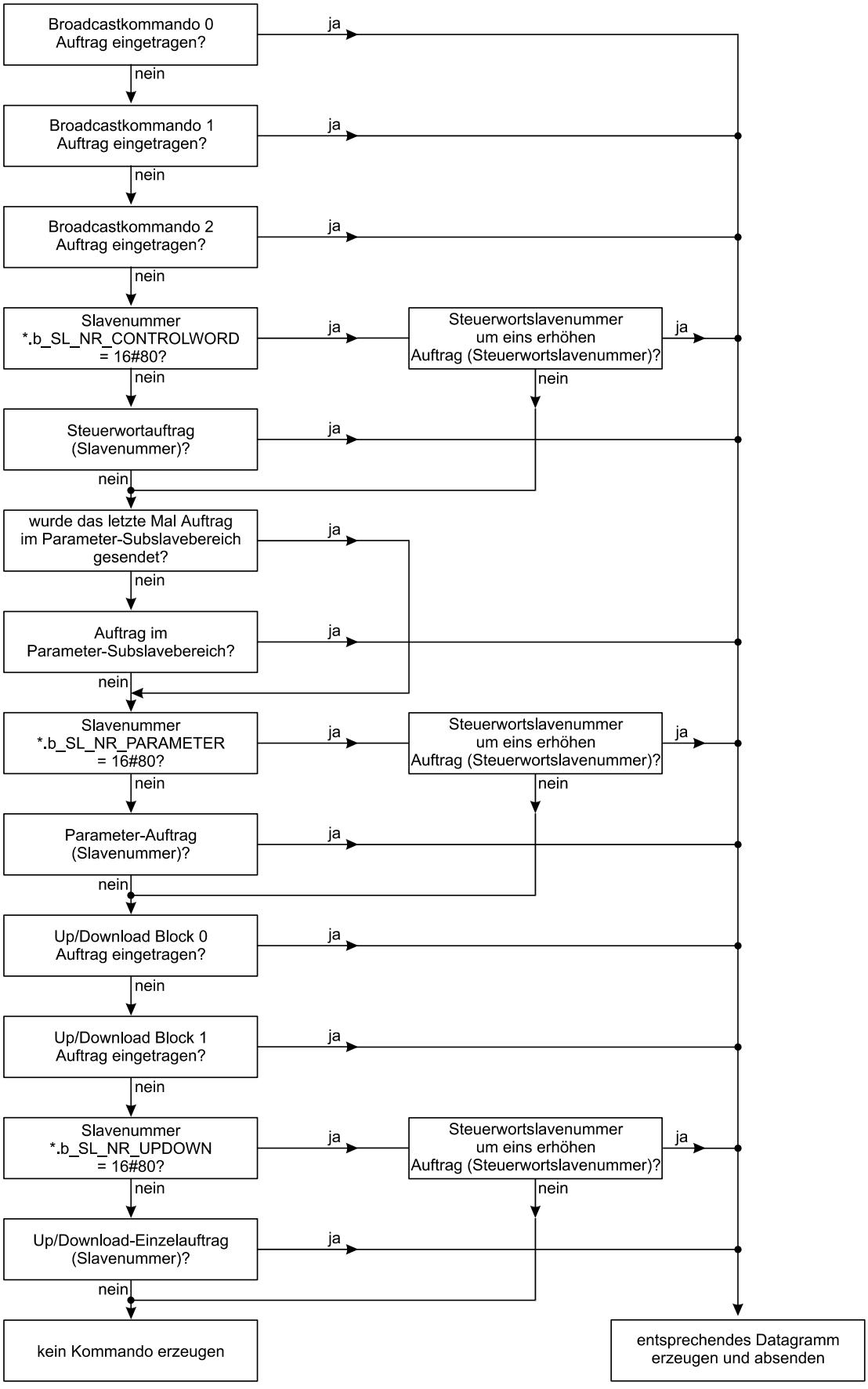


Abbildung 7-5: Ablauf und Priorität von Aufträgen (* entspricht zum Beispiel _CANsync_CTRL_MA)

Aus dem Schema kann man erkennen, welche Aufträge vorrangig behandelt werden (z. B. Broadcast-Kommandos), welche Aufträge andere Aufträge blockieren können und nach welcher Anzahl von CANsync-Intervallen spätestens ein Auftrag bearbeitet wird. Das Broadcastkommando 0 hat die höchste Priorität, es wird sofort in diesem CANsync-Intervall gesendet. Nur wenn in diesem Bereich kein Auftrag eingetragen ist, wird der nächste Bereich abgeprüft. Die Vorgeschichte wird nicht berücksichtigt. Wenn also in jedem CANsync-Intervall ein Auftrag im Broadcastbereich 0 eingetragen ist, wird kein anderes Kommando mehr gesendet.

Broadcastkommando

Broadcastkommandos können mit den folgenden Registern gesendet werden:

Register	Inhalt
*.b_CTRL_REG_BC0	Kontrollregister Broadcastkommando 0
*.d_SL_MASK_BC0	Bitleiste Broadcastkommando 0
*.b_CMD_BC0	Kommandobyte Broadcastkommando 0
*.b_DATA_BYTE_BC0	Datenbyte 0 Broadcastkommando 0 (DB)
*.w_DATA_WORD_BC0	Datenwort 1 Broadcastkommando 0 (DW)

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Register	Inhalt
*.b_CTRL_REG_BC1	Kontrollregister Broadcastkommando 1
*.d_SL_MASK_BC1	Bitleiste Broadcastkommando 1
*.b_CMD_BC1	Kommandobyte Broadcastkommando 1
*.b_DATA_BYTE_BC1	Datenbyte 0 Broadcastkommando 1 (DB)
*.w_DATA_WORD_BC1	Datenwort 1 Broadcastkommando 1 (DW)

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Register	Inhalt
*.b_CTRL_REG_BC2	Kontrollregister Broadcastkommando 2
*.d_SL_MASK_BC2	Bitleiste Broadcastkommando 2
*.b_CMD_BC2	Kommandobyte Broadcastkommando 2
*.b_DATA_BYTE_BC2	Datenbyte 0 Broadcastkommando 2 (DB)
*.w_DATA_WORD_BC2	Datenwort 1 Broadcastkommando 2 (DW)

(* entspricht zum Beispiel _CANsync_CTRL_MA)

In der Bitleiste *.d_SL_MASK_BC0, *.d_SL_MASK_BC1, *.d_SL_MASK_BC2 wird angegeben für welche CANsync-Slaves das Aktionskommando bestimmt ist. Jedem CANsync-Slave ist ein Bit zugeordnet (Bit 0 = Slave 0; Bit 1 = Slave 1; ... Bit 31 = Slave 31).

Wenn dieses Bit = TRUE ist, wird der zugeordnete CANsync-Slave das Kommando ausführen.

Zu einem Befehl können noch ein Datenbyte (*.b_DATA_BYTE...) und ein Datenwort (*.w_DATA_WORD...) übertragen werden.

Um ein Kommando abzusenden, muß in das Kontrollregister 16#05 eingetragen werden. Als Bestätigung für das Senden wird dann von der CANsync-Anschaltung 16#04 gemeldet.

Liste der Kommandos

Kommando-byte	Kommando
16#01	Steuerwort DB = nicht benutzt DW = Steuerwort
16#02 - 16#FF	reserviert

Steuerwortkommando

Das Steuerwortkommando wird mit den folgenden Registern gesendet.

Register	Inhalt
*.a_STEUREG_CONTROLWORD[0]	Kontrollregister Steuerwort für Slave 0
*.a_STEUREG_CONTROLWORD[1]	Kontrollregister Steuerwort für Slave 1
*.a_STEUREG_CONTROLWORD[2]	Kontrollregister Steuerwort für Slave 2
*.a_STEUREG_CONTROLWORD[3]	Kontrollregister Steuerwort für Slave 3
...	...
*.a_STEUREG_CONTROLWORD[31]	Kontrollregister Steuerwort für Slave 31

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Um an einen CANsync-Slave ein Steuerwort zu senden, muß das Steuerwort in den Bereich ab *.a_CONTROLWORD_SL[0] eingetragen werden und anschließend auf das entsprechende Kontrollregister ab *.a_STEUREG_CONTROLWORD[0] 16#05 eingetragen werden. Wenn das Steuerwort gesendet wurde wird als Quittung 16#04 gemeldet.

Beispiel: Steuerwort an CANsync-Slave 3 senden:

1. Steuerwort in *.a_CONTROLWORD_SL[3] eintragen.
2. In Kontrollregister *.a_STEUREG_CONTROLWORD[3] 16#05 eintragen.

   Quittung nach dem Senden: 16#04 auf *.a_STEUREG_CONTROLWORD[3].

Register	Inhalt
*.a_CONTROLWORD_SL[0]	Steuerwort für Slave 0
*.a_CONTROLWORD_SL[1]	Steuerwort für Slave 1
*.a_CONTROLWORD_SL[2]	Steuerwort für Slave 2
*.a_CONTROLWORD_SL[3]	Steuerwort für Slave 3
...	...
*.a_CONTROLWORD_SL[31]	Steuerwort für Slave 31

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Auf dem CANsync-Bus wird das Steuerwortkommando in ein Aktionskommando umgesetzt, bei dem in der Maske (Bitleiste) nur das Bit für den einen CANsync-Slave gesetzt ist, das Kommandobyte auf 16#01 gesetzt ist und als Datenwort (DW) das Steuerwort eingetragen ist.

Parameterkommando

Parameter können mit den folgenden Registern geschrieben und gelesen werden:

Register	Inhalt
*.a_CTRLREG_PAR_CMD_SL[0].b_STEUREG_PAR_CMD	Steuerregister Parameterkommando 0
*.a_CTRLREG_PAR_CMD_SL[0].b_STATREG_PAR_CMD	Statusregister Parameterkommando 0
*.a_CTRLREG_PAR_CMD_SL[1].b_STEUREG_PAR_CMD	Steuerregister Parameterkommando 1
*.a_CTRLREG_PAR_CMD_SL[1].b_STATREG_PAR_CMD	Statusregister Parameterkommando 1
...	...
*.a_CTRLREG_PAR_CMD_SL[31].b_STEUREG_PAR_CMD	Steuerregister Parameterkommando 31
*.a_CTRLREG_PAR_CMD_SL[31].b_STATREG_PAR_CMD	Statusregister Parameterkommando 31

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Bedeutung des Steuer- und des Statusregisters:

Bit 0	**Aktiv** muß auf 1 gesetzt werden, um Befehl zu starten wird auf 0 gesetzt, wenn Antwort empfangen wurde
Bit 1	**Read** =1: Parameter lesen
Bit 2	**Write** =1: Parameter schreiben
Bit 3	**Send** =1: Anzeige, daß Befehl gesendet wurde
Bit 4	**Format** 0: Parameter hat Format Wort 1: Parameter hat Format Doppelwort
Bit 5	**Fehleranzeige** 1: Fehler aufgetreten 0: kein Fehler
Bit 6	**Busy** = 1 CANsync-Slave bearbeitet Auftrag, Daten noch nicht gültig
Bit 7	reserviert

Register	Inhalt
*.a_DATA_PARAMETER[0].d_PAR_VALUE *.a_DATA_PARAMETER[0].w_PAR_NR *.a_DATA_PARAMETER[0].w_SUBSL_ADR	Daten CANsync-Slave 0 Parameternummer 0 reserviert
*.a_DATA_PARAMETER[1].d_PAR_VALUE *.a_DATA_PARAMETER[1].w_PAR_NR *.a_DATA_PARAMETER[1].w_SUBSL_ADR	Daten CANsync-Slave 1 Parameternummer 1 reserviert
...	...
*.a_DATA_PARAMETER[31].d_PAR_VALUE *.a_DATA_PARAMETER[31].w_PAR_NR *.a_DATA_PARAMETER[31].w_SUBSL_ADR	Daten CANsync-Slave 31 Parameternummer 31 reserviert

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Unter „Daten CANsync Slave x“ wird der Parameterwert eingetragen bzw. empfangen. Dieser Wert kann ein Wort oder Doppelwort sein.

Die Parameternummer wählt einen Parameter in dem angesprochenen CANsync-Slave aus. Die Bedeutung des Parameters muß aus der Beschreibung des CANsync-Slaves entnommen werden.

Ablauf eines Parameterzugriffs

Die Parameterkommandos können sowohl im CANsync Event-Task als auch im restlichen Programm verwendet werden. Zur Sicherung der Datenkonsistenz erfolgt die Auftragserteilung und die Auswertung der Rückantwort über das Steuerregister und das Statusregister. Um eine konfliktfreie Durchführung des Auftrags zu gewährleisten, muß vom Applikationsprogramm aus immer zuerst das Steuerregister und dann das Statusregister gelesen werden und wenn ein neuer Wert geschrieben wird, immer zuerst das Statusregister und dann das Steuerregister gesetzt werden. (Für die CANsync-Anschaltung ist das Steuerregister das entscheidende Register.)

Die folgende Erläuterung bezieht sich in den Adressen auf den CANsync-Slave 0.

Für einen Parameter-Schreibauftrag wird
der **Datenwert** in *.a_DATA_PARAMETER[0].d_PAR_VALUE (Doppelwort) und
die **Parameternummer** in *.a_DATA_PARAMETER[0].w_PAR_NR eingetragen.

Anschließend trägt man im Statusregister *.a_CTRLREG_PAR_CMD_SL[0].b_STATREG_PAR_CMD und im Steuerregister *.a_CTRLREG_PAR_CMD_SL[0].b_STEUREG_PAR_CMD den Wert 16#05 ein, wenn ein Wortparameter beschrieben wird, bzw. 16#15, wenn ein Doppelwort-Parameter beschrieben wird.

Wenn der Auftrag (entsprechend der “Konfigurierung Kommandokanal” auf Seite 160) von der CAN-Anschaltung übernommen wurde, wird als Bestätigung das Bit 3 (Send) im Steuerregister und im Statusregister gesetzt.

Wenn der CANsync-Slave den Auftrag akzeptiert, aber noch nicht beendet hat, wird das Bit 6 (Busy) gesetzt.

Falls im CANsync-Slave bei der Ausführung des Auftrags ein Fehler auftritt, wird das Bit 5 (Fehleranzeige) gesetzt, das Bit 6 (Busy) und das Bit 0 (Aktiv) wird gelöscht, dann ist die Fehlernummer im Datenbereich *.a_DATA_PARAMETER[0].d_PAR_VALUE zu lesen.

Wenn der CANsync-Slave den Auftrag beendet hat, wird das Bit 6 (Busy) und das Bit 0 (Aktiv) gelöscht (im Byte ergibt sich 16#04 für Wort schreiben, bzw. 16#14 für Doppelwort schreiben).

Die Auswertung im Applikationsprogramm kann sich darauf beschränken, das Bit 0 abzufragen, solange es gesetzt ist, wird der Auftrag noch bearbeitet, wenn es zurückgesetzt wird, ist der Auftrag beendet.

Es kann sein, daß das Bit 6 (Busy) nicht gesetzt wird, wenn der CANsync-Slave den Auftrag sofort beantworten kann.

Der Parameter-Leseauftrag läuft dazu ähnlich ab, nur mit dem Unterschied, daß der Befehl 16#03 für Wort lesen und 16#13 für Doppelwort lesen lautet und daß die Daten nicht vor dem Auftrag eingetragen werden, sondern nachdem das Bit 0 (Aktiv) gelöscht wurde, zum Auslesen zur Verfügung stehen. Die Quittung im Byte gesehen lautet dann 16#02 für Wortzugriff und 16#12 für Doppelwortzugriff. Das Format wird aus der Antwort des CANsync-Slaves übernommen.

Fehlernummer des CANsync-Slaves

Wert	Bedeutung
16#0000	Kein Fehler aufgetreten
16#FFFF	Fehler aufgetreten
16#FFFE	Wert kleiner Minimalwert
16#FFFD	Wert größer Maximalwert

Wert	Bedeutung
16#FFFC	Element nicht veränderbar
16#FFFB	Element nicht vorhanden
16#FFFA	Daten nicht verfügbar (z. B. in Bearbeitung)
16#FFF9	Fehler beim Datenformat

Up/Download Blockbereich

Upload- und Download-Aufträge können mit den folgenden Registern gesteuert werden:

Register	Inhalt
*.b_STEUREG_UPDOWNBLK0	reserviert
*.b_STATREG_UPDOWNBLK0	reserviert
*.b_SL_NR_UPDOWNBLK0	reserviert
*.w_ERR_NR_UPDOWNBLK0	reserviert
*.w_SUBSL_NR_UPDOWNBLK0	reserviert
*.d_BASE_ADR_UPDOWNBLK0	reserviert
*.w_LENGTH_UPDOWNBLK0	reserviert
*.w_COUNTER_UPDOWNBLK0	reserviert
*.a_DATA_UPDOWNBLK0[0 bis 74]	reserviert
*.b_STEUREG_UPDOWNBLK1	Steuerregister Up/down-Block 1
*.b_STATREG_UPDOWNBLK1	Statusregister Up/down-Block 1
*.b_SL_NR_UPDOWNBLK1	CANsync-Slavenummer Up/down-Block 1
*.w_ERR_NR_UPDOWNBLK1	Fehlernummer Up/down-Block 1
*.w_SUBSL_NR_UPDOWNBLK1	Subslavenummer Up/down-Block 1
*.d_BASE_ADR_UPDOWNBLK1	Basisadresse Up/down-Block 1
*.w_LENGTH_UPDOWNBLK1	Länge in Bytes Up/down-Block 1
*.w_COUNTER_UPDOWNBLK1	Zähler Up/down-Block 1
*.a_DATA_UPDOWNBLK1[0 bis 74]	Datenblock Up/down-Block 1 (75 Doppelworte)

(* entspricht zum Beispiel _CANsync_CTRL_MA)

HINWEIS

Der Block 0 ist für OmegaOS reserviert.

Bedeutung des Steuer- und des Statusregisters

Bit 0	**Aktiv** muß auf 1 gesetzt werden, um Befehl zu starten wird auf 0 gesetzt, wenn Befehl abgearbeitet wird
Bit 1	**Send** =1: Anzeige, daß Befehl gesendet wurde

Bit 2 Bit 3	**Mode** Bit3 Bit2 0 0: reserviert 0 1: Initialisierung 1 0: laufender Up/Download 1 1: Blockende
Bit 4	**Up/Download** 0: Upload 1: Download
Bit 5	**Fehleranzeige** 1: Fehler aufgetreten 0: kein Fehler
Bit 6	**Busy** = 1 CANsync-Slave bearbeitet Auftrag, Daten noch nicht gültig
Bit 7	**Reset** Abbruch eines Auftrags

Ablauf eines Up/Download-Auftrags im Block 1

Die Up/Downloadkommandos können sowohl im CANsync Event-Task als auch im restlichen Programm verwendet werden. Zur Sicherung der Datenkonsistenz erfolgt die Auftragserteilung und die Auswertung der Rückantwort über das Steuerregister und das Statusregister. Um eine konfliktfreie Durchführung des Auftrags zu gewährleisten, muß vom Applikationsprogramm aus immer zuerst das Steuerregister und dann das Statusregister gelesen werden und wenn ein neuer Wert geschrieben wird, immer zuerst das Statusregister und dann das Steuerregister gesetzt werden. (Für die CANsync-Anschaltung ist das Steuerregister das entscheidende Register.)

Für einen Download wird zuerst der Datenblockbereich *.a_DATA_UPDOWNBLK1[0] bis *.a_DATA_UPDOWNBLK1[74] mit den zu übertragenden Daten gefüllt. Die maximale Blocklänge sind 300 Bytes (75 Doppelworte), die mit einem Downloadauftrag gesendet werden können.

Die Nummer des CANsync-Slaves, der den Download empfangen soll wird im *.b_SL_NR_UPDOWNBLK1 und die Basisadresse im *.d_BASE_ADR_UPDOWNBLK1 eingetragen.

Die Basisadressen 16#0000_0000 bis 16#0000_00FF sind für Aufträge des Betriebssystems reserviert.

Anschließend trägt man im Statusregister *.b_STATREG_UPDOWNBLK1 und im Steuerregister *.b_STEUREG_UPDOWNBLK1 den Wert 16#15 ein.

Die CANsync-Anschaltung wird nun versuchen den gesamten Block zum CANsync-Slave zu übertragen. Die aufeinanderfolgenden Modi werden automatisch generiert. Der Fortschritt kann im Byte-Zähler im *.w_COUNTER_UPDOWNBLK1 abgelesen werden.

Wenn der Auftrag (siehe “Konfigurierung Kommandokanal” auf Seite 160) von der CANsync-Anschaltung übernommen wurde, wird als Bestätigung das Bit 1 (Send) im Steuerregister und im Statusregister gesetzt.

Wenn der CANsync-Slave den Auftrag akzeptiert, aber noch nicht beendet hat, wird das Bit 6 (Busy) gesetzt.

Falls im CANsync-Slave bei der Ausführung des Auftrags ein Fehler auftritt, wird das Bit 5 (Fehleranzeige) gesetzt, das Bit 6 (Busy) und das Bit 0 (Aktiv) wird gelöscht, dann ist die Fehlernummer im *.b_SL_NR_UPDOWNBLK1 zu lesen.

Wenn der CANsync-Slave den Auftrag beendet hat, wird das Bit 6 (Busy) und das Bit 0 (Aktiv) gelöscht (im Steuer- und Statusregister ergibt sich 16#1C).

Die Auswertung im Applikationsprogramm kann sich darauf beschränken, das Bit 0 (Aktiv) abzufragen. Solange es gesetzt ist, wird der Auftrag noch bearbeitet, wenn es zurückgesetzt wird, ist der Auftrag beendet.

Es kann sein, daß das Bit 6 (Busy) nicht gesetzt wird, wenn der CANsync-Slave den einzelnen Modi des Auftrags sofort beantworten kann.

Der Upload-Auftrag läuft dazu ähnlich ab, nur mit dem Unterschied, daß der Befehl 16#05 lautet und daß die Daten nicht vor dem Auftrag eingetragen werden, sondern nachdem das Bit 0 (Aktiv) gelöscht wurde, zum Auslesen zur Verfügung stehen. Die Quittung im Steuer- und Statusregister gesehen lautet dann 16#0C.

Der Abbruch eines Up/Download-Auftrags ist möglich, indem man das Bit 7 (Reset) auf TRUE setzt. An den CANsync-Slave wird dann ein Upload-Initialisierungstelegramm mit Adresse = 0 und Länge = 0 gesendet.

Fehlernummern Up/Download

Fehlernummer	Bedeutung
16#0001	CANsync-Slave quittiert falsche Blocknummer
16#0002	eingetragenen Länge > 300 Bytes
...	
16#0100	CANsync-Slave erwartet Block mit Nummer, die im Zähler eingetragen ist
16#0101	CANsync-Slave erwartet Block-Ende
16#0102	CANsync-Slave erwartet noch nicht Block-Ende
16#0103	CANsync-Slave bricht Up/Download ab
16#0104	Up/Download nicht möglich
16#0105	Basisadresse nicht erlaubt
16#0106	reserviert
16#0107	Blocklänge > maximale Blocklänge des CANsync-Slaves
16#0108	Telegramm-Modefehler (in dieser Phase nicht erlaubter Mode)

CANsync-Slavestatus

Register	Inhalt
*.a_STAT_SL[0]	CANsync-Slavestatus 0
*.a_STAT_SL[1]	CANsync-Slavestatus 1
*.a_STAT_SL[2]	CANsync-Slavestatus 2
...	...
*.a_STAT_SL[31]	CANsync-Slavestatus 31

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Bedeutung CANsync-Slavestatus:

Bit 0	**Response** = 1: CANsync-Slave meldet sich
Bit 1	**Synchron** = 1: CANsync-Slave ist synchronisiert

Bit 2	reserviert
Bit 3	
Bit 4	
Bit 5	
Bit 6	
Bit 7	

Im Einrichtbetrieb (siehe “Initialisierung” auf Seite 148) wird der Slavestatus von allen initialisierten CANsync-Slaves abgefragt und hier eingetragen. Das Bit 0 (Response) zeigt an, daß der CANsync-Slave sich am CANsync-Bus gemeldet hat. Das Bit 1 (Synchron) zeigt an, daß der CANsync-Slave synchronisiert ist und der Synchronbetrieb gestartet werden kann.

7.2.3 Registerstruktur und Funktion des Ωmega CANsync-Slave

Im folgenden wird die Registerstruktur des Kommunikations-RAM im Ωmega-CANsync-Slave erläutert.

Um im PROPROG wt II Projekt auf die Register des Kommunikations-RAM zugreifen zu können, sind Datentypen definiert, die die Registerstruktur abbilden. Mit diesen Datentypen werden Variablen deklariert, die auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Anschließend ist es möglich über die Strukturelemente der deklarierten Variablen auf die Register des Kommunikations-RAM zuzugreifen.

Bei der Initialisierung der CANsync-Slave-Anschaltung haben die Register im Kommunikations-RAM eine andere Bedeutung als nach der Initialisierung, im zyklischen Betrieb.

Deshalb gibt es für die Initialisierung die Struktur

CANsync_INIT_BMSTRUCT

und für den zyklischen Betrieb die Struktur

CANsync_SL_CTRL_BMSTRUCT

Diese Strukturen sind ab der Bibliothek BM_TYPES_20bd00 definiert. Nachdem die Bibliothek BM_TYPES_20bd00 im Projekt eingebunden ist, stehen die Datentypen zur Verfügung.

Diese Strukturen enthalten

- 8-Bit-Elemente,
- 16-Bit-Elemente,
- 32-Bit-Elemente,
- Strukturen aus den o.g. Elementen
- Felder (ARRAY) und Strukturen aus den o.g. Elementen und Strukturen

Den in einer Struktur verwendeten Datentypen (8-, 16-, 32-Bit-Elemente, Strukturen und Feldern) wurden Kurzbezeichnungen vorangefügt. Dies dient der Übersichtlichkeit bei der Verwendung der Strukturen in der Programmierung.

Datentyp	Kurzbezeichnung	Anzahl der Bits
BYTE	b	8
WORD	w	16
DWORD (double word)	d	32
SINT (short integer)	si	8
DINT (double integer)	di	32
USINT (unsigned short integer)	us	8
UINT (unsigned integer)	u	16
UDINT (unsigned double integer)	ud	32
STRUCT	_ (underline)	-
ARRAY	a	-

Weitere, nicht in den Strukturen verwendete Datentypen sind:

Datentyp	Kurzbezeichnung	Anzahl der Bits
BOOL (bit)	x	1
TIME	t	-

Erläuterung zur Deklaration der globalen Variablen

Für die Initialisierung wird eine globale Variable vom Datentyp CANsync_INIT_BMSTRUCT angelegt. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) im Ωmega Drive-Line II,

```
_CANsync_INIT_SL        AT     %MB3.100000 : CANsync_INIT_BMSTRUCT;
```

dabei ist:

`_CANsync_INIT_SL`	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
`CANsync_INIT_BMSTRUCT`	der Datentyp
`%MB3.100000`	die Basisadresse der CANsync-Anschaltung 1 am Ωmega Drive-Line II.

Für den zyklischen Betrieb wird eine globale Variable vom Datentyp CANsync_SL_CTRL_BMSTRUCT angelegt. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) im Ωmega Drive-Line II,

```
_CANsync_CTRL_SL        AT     %MB3.100000 : CANsync_SL_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_SL`	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT

`CANsync_SL_CTRL_BMSTRUCT`	der Datentyp
`%MB3.100000`	die Basisadresse der CANsync-Anschaltung 1 am Ωmega Drive-Line II.

HINWEIS

In den nachfolgenden Tabellen wird der Variablenname durch * ersetzt.

Auf das Register `*.w_CPU_CONTROL` greift man demzufolge über

`_CANsync_INIT_SL.w_CPU_CONTROL` zu,

auf `*.w_OPTION_STATUS` greift man über

`_CANsync_INIT_SL.w_OPTION_STATUS` zu.

Dabei ist:

`_CANsync_INIT_SL`	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
`w_CPU_CONTROL`	das Steuerregister der CANsync-Anschaltung mit der Datentypkurzbezeichnung „w“ für WORD

Die Register `*.w_CPU_CONTROL` und `*.w_OPTION_STATUS` können auch über die Struktur für den zyklischen Betrieb angesprochen werden. Der Zugriff ist dann über

`_CANsync_CTRL_SL.w_CPU_CONTROL` und

`_CANsync_CTRL_SL.w_OPTION_STATUS` möglich.

Dabei ist

`_CANsync_CTRL_SL`	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
`w_OPTION_STATUS`	das Statusregister der CANsync-Anschaltung mit der Datentypkurzbezeichnung „w“ für WORD

Beispiel für den Zugriff auf ein Element eines Feldes, das in der Struktur verwendet wird:

laut Tabelle: `*.a_WR_VALUE_RECEIVE[3]`

Zugriff: `_CANsync_CTRL_SL.a_WR_VALUE_RECEIVE[3]`

Dabei ist

`_CANsync_CTRL_SL`	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT

a_WR_VALUE_RECEIVE[3]	das Register für den Sollwert 3 mit der Datentypkurzbezeichnung „a“ für ARRAY. Der Datentyp der Elemente des Feldes (der Sollwerte) wird der entsprechenden Tabelle und der Beschreibung entnommen.

Beispiel für den Zugriff auf ein Element eines zweidimensionalen Feldes, das in der Struktur verwendet wird:

laut Tabelle: `*.a_RD_VALUE[5][7]`

Zugriff: `_CANsync_CTRL_SL.a_RD_VALUE[5][7]`

Dabei ist

_CANsync_CTRL_SL	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
a_RD_VALUE[5][7]	das Register für den Istwert 7 des CANsync-Slaves 5 mit der Datentypkurzbezeichnung „a“ für ARRAY. Der Datentyp der Elemente des Feldes (der Sollwerte) wird der entsprechenden Tabelle und der Beschreibung entnommen.

Beispiel für den Zugriff auf ein Element einer (Sub-) Struktur, die selbst Element eines Feldes ist, das in der Struktur verwendet wird:

laut Tabelle: `*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3`

Zugriff: `_CANsync_CTRL_SL.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3`

Dabei ist

_CANsync_CTRL_SL	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
a_CFG_RDC_WORD[31]	das Feld mit den Konfigurierungsdaten für das Mapping der Worte der Istwert-Telegramme des CANsync-Slaves 31 mit der Datentypkurzbezeichnung „a“ für ARRAY
b_CFG_RDC2_WORD3	das Register für die Konfigurierungsdaten für das Mapping des 3. Wortes des Istwert-Telegramms 2 (des CANsync-Slaves 31) mit der Datentypkurzbezeichnung „b“ für BYTE

Beispiel für den Zugriff auf ein Element einer (Sub-) Struktur, die in der Struktur verwendet wird:

laut Tabelle: `*._CFG_WRC_WORD.b_CFG_WRC1_WORD0`

Zugriff: `_CANsync_CTRL_SL._CFG_WRC_WORD.b_CFG_WRC1_WORD0`

Dabei ist

_CANsync_CTRL_SL	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT

_CFG_WRC_WORD	die Struktur mit den Konfigurierungsdaten für das Mapping der Worte der Sollwert-Telegramme mit der Datentypkurzbezeichnung „_“ für STRUCT
b_CFG_WRC1_WORD0	das Register für die Konfigurierungsdaten für das Mapping des 0. Wortes des Sollwert-Telegramms 1 mit der Datentypkurzbezeichnung „b“ für BYTE

Allgemeine Register der CANsync-Anschaltung

Register	Inhalt
*.w_CANsync_STATUS	CANsync-Status
*.w_OMEGA_NR	über Dip-Schalter eingestellte Ωmega-Nummer
*.i_SW1_NR	Karten-Softwarenummer
*.i_SW1_RELEASE	Softwarestand inkompatibel und kompatibel

CANsync-Status

Bei jedem Durchlaufen des CANsync-Prozessorzyklusses wird der CANsync-Status auf *.w_CANsync_STATUS ausgegeben.

Bedeutung:

Bit-Nr.	Bedeutung (Bit = TRUE)
0	reserviert
1	Overrun: eine CANsync-Nachricht konnte nicht empfangen werden
2	CANsync-Sendepuffer ist frei
3	CANsync-Sendeauftrag wurde erfolgreich durchgeführt
4	es wird gerade eine CANsync-Meldung empfangen
5	es wird gerade eine CANsync-Meldung gesendet
6	Fehler vorhanden (Warnung)
7	CANsync-Knoten ist deaktiviert (BUS-off)
8-15	reserviert

Ωmega-Nummer

Im Register *.w_OMEGA_NR wird die über die DIP-Schalter (S33) eingestellte Ωmega-Nummer angezeigt. Bei einer CANsync-Slave-Anschaltung ist diese Nummer die Slavenummer.

Softwarenummer und Softwarestand

Im Register *.i_SW1_NR wird die Nummer der CANsync-Software auf dem Ωmega Drive-Line II angezeigt.

Im Register *.i_SW1_RELEASE wird der inkompatible und der kompatible Stand der CANsync-Software auf dem Ωmega Drive-Line II angezeigt.

Initialisierung

Für die Initialisierung wird eine globale Variable vom Datentyp CANsync_INIT_BMSTRUCT angelegt. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) im Ωmega Drive-Line II,

```
_CANsync_INIT_SL        AT     %MB3.100000 : CANsync_INIT_BMSTRUCT;
```

dabei ist:

`_CANsync_INIT_SL`	der Variablenname mit der Datentypkurzbezeichnung „_“ für STRUCT
`CANsync_INIT_BMSTRUCT`	der Datentyp
`%MB3.100000`	die Basisadresse der CANsync-Anschaltung 1 am Ωmega Drive-Line II.

Zusätzlich konfiguriert man für den Synchron-Betrieb:

Bedeutung		**Register**	**Wert**
Baudrate	z. B.: 500 kBit/s	*.b_BT_0 *.b_BT_1	16#00 16#1C
CANsync- Intervall	z. B.: 2 ms	*.b_TIME_PATTERN	16#02
Acceptance Code	alle Telegramme	*.b_AC	16#FF
Acceptance Mask		*.b_AM	16#FF
Output-Control	16#FA	*.b_OUTPUT_CONTROL	16#FA
Clock-Divider	16#07	*.b_CLOCK_DIVIDER	16#07
Slave/Master	Slave	*.b_MA_SL_MODE	16#01

Die Betriebsarteneinstellung erfolgt über das Register *.w_CPU_CONTROL. Die Anzeige der gerade aktiven Betriebsart erfolgt im Register *.w_OPTION_STATUS. Auch nach dem erfolgreichen Starten einer Betriebsart kann die Betriebsart wieder geändert werden.

Register	Inhalt
*.w_CPU_CONTROL	Steuerregister CANsync-Anschaltung
*.w_OPTION_STATUS	Statusregister CANsync-Anschaltung

(* entspricht zum Beispiel bei der Initialisierung _CANsync_INIT_SL, nach der Initialisierung zum Beispiel _CANsync_CTRL_SL)

Steuerregister CANsync-Anschaltung	Bedeutung
16#0000	Neuanlauf
16#0001	Test Handshake
16#0002	Initialisierungsdaten übernehmen
16#0012	reserviert
16#0013	reserviert
16#0020	Synchron-Betrieb starten
16#0040	Aktiven Betrieb freigeben
16#0080	(Bit 7 = TRUE) Reset CAN-Controller

Statusregister CANsync-Anschaltung	Bedeutung
16#0001	Anlauf
16#0002	Warten auf Initialisierungsdaten übernehmen
16#0003	Warten auf Start
16#0011	reserviert
16#0012	reserviert
16#0013	Synchron-Betrieb CANsync-Slave wird eingerichtet
16#0020	Synchron-Betrieb CANsync-Slave ist aktiv
16#0041	reserviert
16#0042	reserviert
16#0043	*Synchron-Betrieb CANsync-Master wird eingerichtet*
16#0080	*Synchron-Betrieb CANsync-Master ist aktiv*

Die Initialisierung wird mit den Kommandos 16#0000, 16#0001 und 16#0002 auf *.w_CPU_CONTROL durchgeführt. Damit wird der Einrichtbetrieb gestartet. Anschließend kann der aktive Betrieb freigegeben werden. Dies geschieht durch Setzen des Bit 6 in *.w_CPU_CONTROL (*.w_CPU_CONTROL = 16#0040).

Wenn man das Bit 7 von *.w_CPU_CONTROL setzt, wird der CANsync-Controller zurückgesetzt und das Bit wieder gelöscht. Dadurch kann der BUS-OFF-Zustand des CANsync-Controllers zurückgesetzt werden und wieder CANsync-Telegramme gesendet und empfangen werden. Die Anzeige über den BUS-OFF-Zustand erfolgt in *.w_CANsync_STATUS (siehe “Allgemeine Register der CANsync-Anschaltung” auf Seite 173).

In der folgenden Tabelle sind die Register angegeben, die in der Initialisierung bedient werden.

Register	Inhalt
*.b_MA_SL_MODE	Betriebsart: Master / Slave (SYNC-OUT / SYNC-IN)
*.b_AC	Acceptance Code des CANsync-Controllers
*.b_AM	Acceptance Mask des CANsync-Controllers
*.b_BT_0	Bit-Timing-Register 0 des CANsync-Controllers
*.b_BT_1	Bit-Timing-Register 1 des CANsync-Controllers
*.b_OUTPUT_CONTROL	Output-Control-Register des CANsync-Controllers
*.b_CLOCK_DIVIDER	Clock-Divider des CANsync-Controllers
*.b_TIME_PATTERN	CANsync-Intervall in ms

(* entspricht zum Beispiel bei der Initialisierung der Struktur _CANsync_INIT_SL;

Sollwerte

Das Empfangen der Sollwerte erfolgt in der CANsync Event-Task.

Register	Inhalt
*.b_CTRLREG_WRC1	Statusregister Sollwertkanal 1
*.b_CTRLREG_WRC2	Statusregister Sollwertkanal 2
*.b_CTRLREG_WRC3	reserviert
*.b_CTRLREG_WRC4	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Im Statusregister der Sollwertkanäle wird mit 16#02 gemeldet, daß das jeweilige Sollwert-Telegramm empfangen wurde.

Die Konfigurierung der Sollwert-Telegramme erfolgt mit den folgenden Registern.

Register	Inhalt
*._CFG_WRC_WORD.b_CFG_WRC1_WORD0	Konfigurierung Sollwert-Telegramm 1 Wort 0
*._CFG_WRC_WORD.b_CFG_WRC1_WORD1	Konfigurierung Sollwert-Telegramm 1 Wort 1
*._CFG_WRC_WORD.b_CFG_WRC1_WORD2	Konfigurierung Sollwert-Telegramm 1 Wort 2
*._CFG_WRC_WORD.b_CFG_WRC1_WORD3	Konfigurierung Sollwert-Telegramm 1 Wort 3
*._CFG_WRC_WORD.b_CFG_WRC2_WORD0	Konfigurierung Sollwert-Telegramm 2 Wort 0
*._CFG_WRC_WORD.b_CFG_WRC2_WORD1	Konfigurierung Sollwert-Telegramm 2 Wort 1
*._CFG_WRC_WORD.b_CFG_WRC2_WORD2	Konfigurierung Sollwert-Telegramm 2 Wort 2
*._CFG_WRC_WORD.b_CFG_WRC2_WORD3	Konfigurierung Sollwert-Telegramm 2 Wort 3
*._CFG_WRC_WORD.b_CFG_WRC3_WORD0	reserviert
*._CFG_WRC_WORD.b_CFG_WRC3_WORD1	reserviert
*._CFG_WRC_WORD.b_CFG_WRC3_WORD2	reserviert
*._CFG_WRC_WORD.b_CFG_WRC3_WORD3	reserviert
*._CFG_WRC_WORD.b_CFG_WRC4_WORD0	reserviert
*._CFG_WRC_WORD.b_CFG_WRC4_WORD1	reserviert
*._CFG_WRC_WORD.b_CFG_WRC4_WORD2	reserviert
*._CFG_WRC_WORD.b_CFG_WRC4_WORD3	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Mit der Konfigurierung gibt man an, welcher Sollwert an welcher Stelle (..._WORD0, ..., ..._WORD3) im Sollwert-Telegramm (..._WRC1_..., ..._WRC2_...) gelesen wird.

Bedeutung

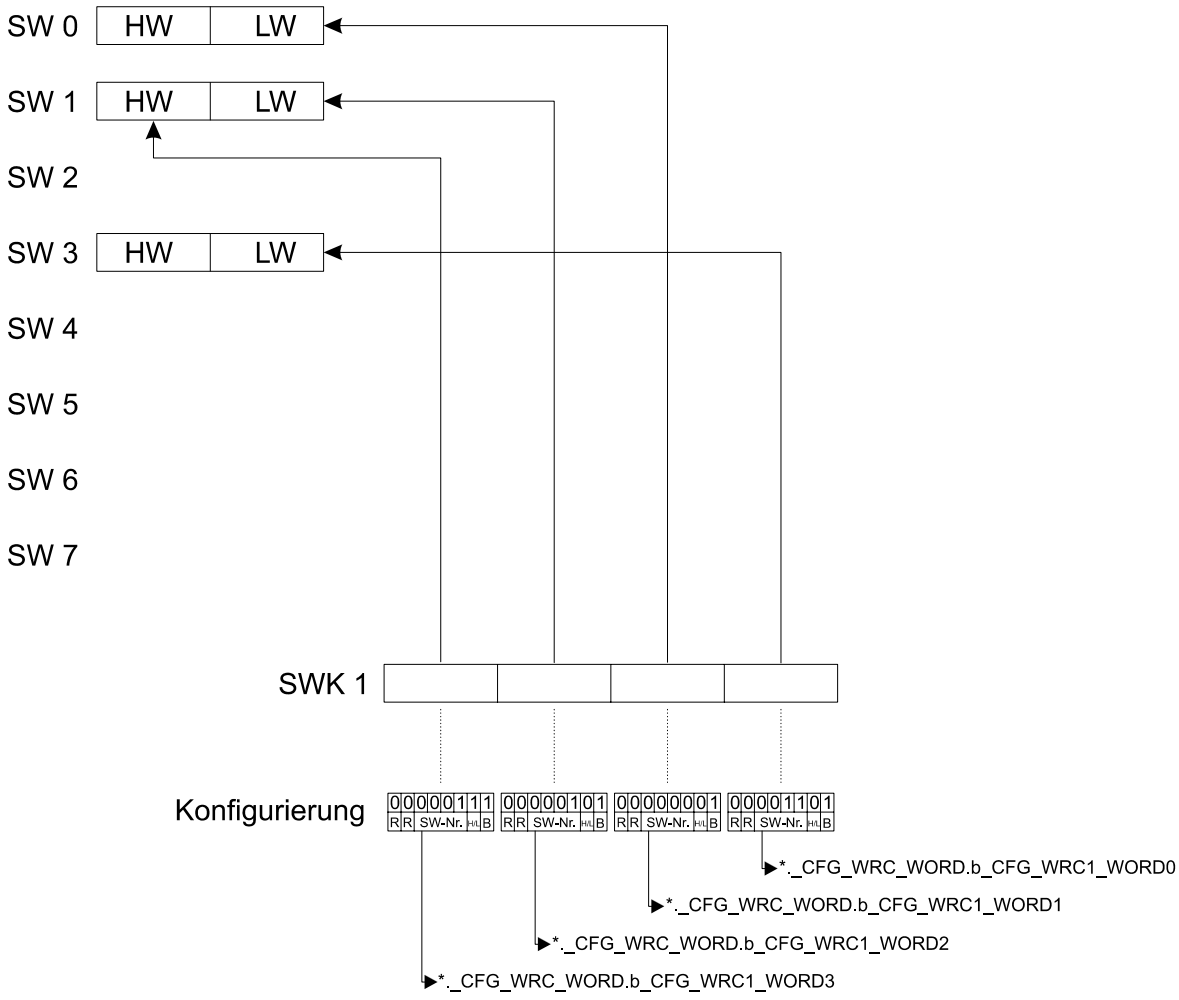
Bit 0	**Belegt** Wenn = 1, wird dieses Wort des Telegramms verwendet
Bit 1	**Highword/Lowword** Wenn = 1, wird das Highword des Sollwertes eingetragen
Bit 2	**Sollwertnummer** Nummer des Sollwertes 0 bis 15
Bit 3	
Bit 4	
Bit 5	

Bit 6	reserviert
Bit 7	

HINWEIS

Ein Wort des Telegramms wird nur dann nicht verwendet, wenn Sollwertnummer = 0, Highword/Lowword = 0 und Belegt = 0 gesetzt wird.

Beispiel:



*._CFG_WRC_WORD.b_CFG_WRC1_WORD0 = 16#0D

0	0	0	0	1	1	0	1
R	R	SW-Nr.				H/L	B

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Mit dieser Einstellung wird das Wort 0 des Sollwert-Telegramms 1 als Lowword des Sollwertes 3 eingetragen.

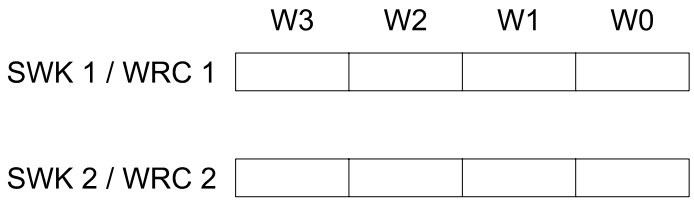
Register	Inhalt
*.a_WR_VALUE_RECEIVE[0]	Sollwert 0
*.a_WR_VALUE_RECEIVE[1]	Sollwert 1
*.a_WR_VALUE_RECEIVE[2]	Sollwert 2
*.a_WR_VALUE_RECEIVE[3]	Sollwert 3
*.a_WR_VALUE_RECEIVE[4]	Sollwert 4
*.a_WR_VALUE_RECEIVE[5]	Sollwert 5
*.a_WR_VALUE_RECEIVE[6]	Sollwert 6
*.a_WR_VALUE_RECEIVE[7]	Sollwert 7
*.a_WR_VALUE_RECEIVE[8]	reserviert
...	...
*.d_WR_VALUE_RECEIVE[15]	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Die Sollwerte können Wort- oder Doppelwort-Sollwerte sein.

Erläuterung zur Benutzung der Sollwertkanäle

Im Synchron-Betrieb stehen zwei Sollwertkanäle (Kanal 1 und 2; auch WRC1 und WRC2) zur Verfügung. In den Sollwertkanälen 1 und 2 werden die Sollwert-Telegramme 1 und 2 des CANsync-Masters empfangen.



Beide Sollwert-Telegramme bestehen aus jeweils vier Worten (W0 bis W3). Nach der CANsync-Initialisierung muß zumindest einmal angegeben werden, wie die Belegung dieser Worte ist.

Dies kann im Initialisierungsprogramm erfolgen. Dazu trägt man im Bereich *._CFG_WRC_WORD.b_CFG_WRC1_WORD0 bis *._CFG_WRC_WORD.b_CFG_WRC2_WORD3 für jedes Wort der Sollwert-Telegramme ein, welcher Sollwert an dieser Stelle empfangen wird. Sollwertnummern sind im Bereich von 0 bis 7 gültig. Für Doppelwort-Sollwerte müssen zwei Worte im Telegramm verwendet werden.

Die Konfigurierung kann auch während des aktiven Betriebs geändert werden. Die Änderung wird spätestens im nächsten CANsync-Intervall übernommen. Zu Beginn des CANsync-Intervalls werden die Konfigurierungsdaten von der CANsync-Anschaltung gelesen.

Die Sollwert-Telegramm-Auswertung erfolgt zu Beginn des CANsync-Intervalls. Dazu fragt man die Statusregister (Kontrollregister) der Sollwertkanäle ab (*.b_CTRLREG_WRC1 bis *.b_CTRLREG_WRC2). In diesen Registern wird 16#02 eingetragen, wenn das entsprechende Sollwert-Telegramm empfangen wurde. Dann kann man die Sollwerte aus den Registern *.a_WR_VALUE_RECEIVE[0] bis *.a_WR_VALUE_RECEIVE[7] auslesen und das Statusregister auf z. B. 16#00 setzen, um die nächste Eintragung richtig erkennen zu können.

Die Zuordnung welche Sollwerte gelesen werden dürfen, wenn ein Sollwert-Telegramm empfangen wurde, muß entsprechend der Sollwertkonfigurierung im Applikationsprogramm erfolgen.

Falls ein Sollwert-Telegramm ausfällt, das z. B. einen Lagesollwert von der virtuellen Leitachse enthält, muß im Applikationsprogramm eine Extrapolation ausgehend vom letzten Lagesollwert erfolgen. Fällt der Sollwert mehrere Male hintereinander aus, muß eine Fehlerreaktion angestoßen werden.

Istwerte des CANsync-Slaves

Das Senden der Istwerte erfolgt in der CANsync Event-Task.

Register	Inhalt
*.b_CTRLREG_RD_RDC1	Kontrollregister Istwertkanal 1
*.b_CTRLREG_RD_RDC2	Kontrollregister Istwertkanal 2
*.b_CTRLREG_RD_RDC3	reserviert
*.b_CTRLREG_RD_RDC4	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Im Kontrollregister wird mit 16#05 gekennzeichnet, daß neue Istwerte für das jeweilige Istwert-Telegramm eingetragen wurden und das Telegramm gesendet wird. Die CANsync-Anschaltung quittiert den Befehl mit 16#04.

Die Konfigurierung der Istwert-Telegramme erfolgt mit den folgenden Registern.

Register	Inhalt
*._CFG_RDC_WORD.b_CFG_RDC1_WORD0	Konfigurierung Istwert-Telegramm 1 Wort 0
*._CFG_RDC_WORD.b_CFG_RDC1_WORD1	Konfigurierung Istwert-Telegramm 1 Wort 1
*._CFG_RDC_WORD.b_CFG_RDC1_WORD2	Konfigurierung Istwert-Telegramm 1 Wort 2
*._CFG_RDC_WORD.b_CFG_RDC1_WORD3	Konfigurierung Istwert-Telegramm 1 Wort 3
*._CFG_RDC_WORD.b_CFG_RDC2_WORD0	Konfigurierung Istwert-Telegramm 2 Wort 0
*._CFG_RDC_WORD.b_CFG_RDC2_WORD1	Konfigurierung Istwert-Telegramm 2 Wort 1
*._CFG_RDC_WORD.b_CFG_RDC2_WORD2	Konfigurierung Istwert-Telegramm 2 Wort 2
*._CFG_RDC_WORD.b_CFG_RDC2_WORD3	Konfigurierung Istwert-Telegramm 2 Wort 3
*._CFG_RDC_WORD.b_CFG_RDC3_WORD0	reserviert
*._CFG_RDC_WORD.b_CFG_RDC3_WORD1	reserviert
*._CFG_RDC_WORD.b_CFG_RDC3_WORD2	reserviert
*._CFG_RDC_WORD.b_CFG_RDC3_WORD3	reserviert
*._CFG_RDC_WORD.b_CFG_RDC4_WORD0	reserviert
*._CFG_RDC_WORD.b_CFG_RDC4_WORD1	reserviert
*._CFG_RDC_WORD.b_CFG_RDC4_WORD2	reserviert
*._CFG_RDC_WORD.b_CFG_RDC4_WORD3	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_MA)

Mit der Konfigurierung gibt man an, welcher Istwert an welcher Stelle im Istwert-Telegramm eingetragen wird.

Bedeutung

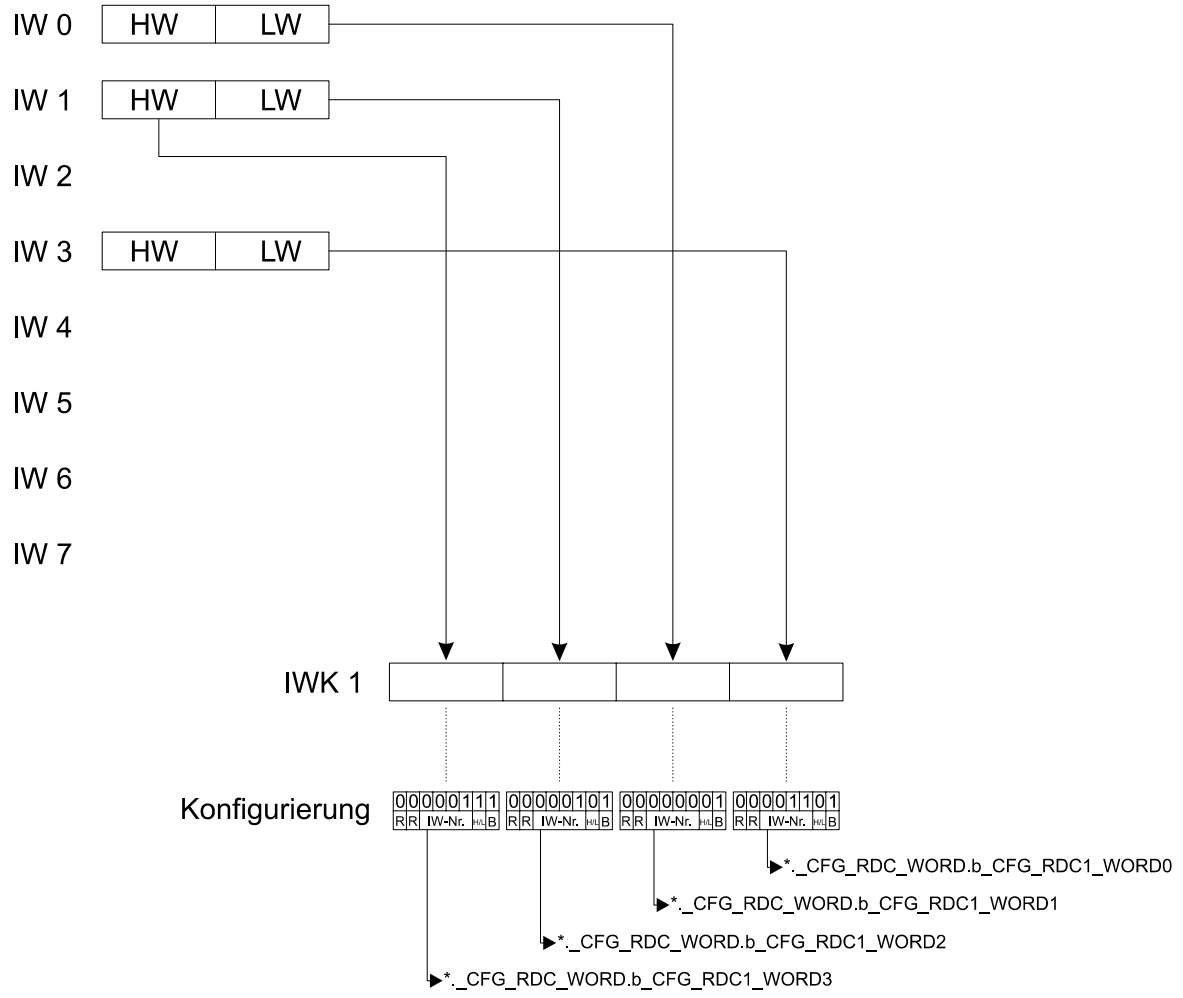
Bit 0	**Belegt** Wenn = 1, wird dieses Wort des Telegramms verwendet
Bit 1	**Highword/Lowword** Wenn = 1, wird das Highword des Istwertes eingetragen
Bit 2	**Istwertnummer** Nummer des Istwertes 0 bis 15
Bit 3	
Bit 4	
Bit 5	

Bit 6	reserviert
Bit 7	

HINWEIS

Ein Wort des Telegramms wird nur dann nicht verwendet, wenn Istwertnummer = 0, Hignword/Lowword = 0 und Belegt = 0 gesetzt wird.

Beispiel:



*._CFG_RDC_WORD.b_CFG_RDC1_WORD0 = 16#0D

0	0	0	0	1	1	0	1
R	R	IW-Nr.				H/L	B

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Mit dieser Einstellung wird das Lowword des Istwertes 3 in das Istwert-Telegramm 1 im Wort 0 eingetragen.

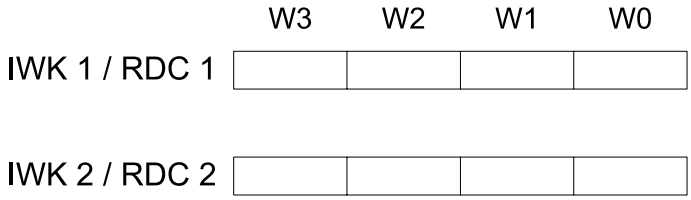
Register	Inhalt
*.a_RD_VALUE_SEND[0]	Istwert 0
*.a_RD_VALUE_SEND[1]	Istwert 1
*.a_RD_VALUE_SEND[2]	Istwert 2
*.a_RD_VALUE_SEND[3]	Istwert 3
*.a_RD_VALUE_SEND[4]	Istwert 4
*.a_RD_VALUE_SEND[5]	Istwert 5
*.a_RD_VALUE_SEND[6]	Istwert 6
*.a_RD_VALUE_SEND[7]	Istwert 7
*.a_RD_VALUE_SEND[8]	reserviert
...	...
*.a_RD_VALUE_SEND[15]	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Die Istwerte können Wort- oder Doppelwort-Istwerte sein.

Erläuterung zur Benutzung der Istwertkanäle

Im Synchron-Betrieb stehen zwei Istwertkanäle (Kanal 1 und 2; auch RDC1 und RDC2) zur Verfügung. In den Istwertkanälen 1 und 2 werden die Istwert-Telegramme 1 und 2 dieses CANsync-Slaves an den CANsync-Master gesendet.



Beide Istwert-Telegramme bestehen aus jeweils vier Worten (W0 bis W3). Nach der CANsync-Initialisierung muß zumindest einmal angegeben werden, wie die Belegung dieser Worte ist.

Dies kann im Initialisierungsprogramm erfolgen. Dazu trägt man im Bereich *._CFG_RDC_WORD.b_CFG_RDC1_WORD0 bis *._CFG_RDC_WORD.b_CFG_RDC2_WORD3 für jedes Wort der Istwert-Telegramme ein, welcher Istwert an dieser Stelle übertragen wird. Istwertnummern sind im Bereich von 0 bis 7 gültig. Für Doppelwort-Istwerte müssen zwei Worte im Telegramm verwendet werden.

Die Konfigurierung kann auch während des aktiven Betriebs geändert werden. Die Änderung wird spätestens im nächsten CANsync-Intervall übernommen. Die Konfigurierungsdaten von der CANsync-Anschaltung werden gelesen, wenn das entsprechende Istwert-Telegramm empfangen wird.

Der CANsync-Slave kann nicht von sich aus das Senden seiner Istwerte auslösen, sondern der CANsync-Master fordert über den Identifier der Sollwert-Telegramme die Istwerte eines CANsync-Slaves an.

Damit immer die aktuellen Istwerte gesendet werden, muß die Istwerteintragung zu Beginn der CANsync Event-Task erfolgen. Die Istwerte für das Istwert-Telegramm 1 müssen bis ca. 490 µs nach dem Start der CANsync Event-Task in die Register *.a_RD_VALUE_SEND[0] bis *.a_RD_VALUE_SEND[7] eingetragen werden, weil dann die CANsync-Anschaltung die Vorbereitung der Istwert-Telegramme beginnt. Als Kennzeichnung, daß neue Istwerte eingetragen wurden, muß im entsprechenden Kontrollregister des Istwertkanals *.b_CTRLREG_RD_RDC1 bis *.b_CTRLREG_RD_RDC2 16#05 eingetragen werden. Dies ist die Freigabe für die CANsync-Anschaltung, daß die Istwerte gelesen werden dürfen und das Istwert-Telegramm erzeugt wird. Falls bis 490 µs

nach dem Start der CANsync Event-Task diese Freigabe nicht erfolgt, fällt in diesem CANsync-Intervall das Istwert-Telegramm 1 aus. Als Quittung für die Erzeugung des Istwert-Telegramms wird im Kontrollregister 16#04 von der CANsync-Anschaltung eingetragen. Dadurch kann der Anwender überprüfen, ob die Istwerteintragung rechtzeitig fertig geworden ist oder nicht. Falls die Istwertgenerierung länger dauert, – vom Start der CANsync Event-Task bis zur Ausführung des Applikationsprogramms vergehen ca. 80 µs –, muß die Istwertgenerierung immer für das nächste CANsync-Intervall erfolgen und zu Beginn des neuen CANsync-Intervalls müssen die vorberechneten Istwerte nur noch an die entsprechenden Register kopiert werden.

Der Zeitpunkt bis zu dem die Istwerte für das Istwert-Telegramm 2 eingetragen werden müssen, ergibt sich aus dem Empfangszeitpunkt des Sollwert-Telegramms 1 plus die Verarbeitungszeit des Sollwert-Telegramms 1. Die Zeitdauer hängt von der Zahl der einzutragenden Sollwertworte ab (insgesamt zwischen 800 µs und 1000 µs nach Start der CANsync Event-Task). Die Signalisierung im Kontrollregister erfolgt wie für das Istwert-Telegramm 1.

Istwerte anderer CANsync-Slaves

Das Empfangen der Istwerte der anderen CANsync-Slaves erfolgt in der CANsync Event-Task.

Register	Inhalt
*.b_STATREG_RDC1	Statusregister Istwertkanal 1
*.b_STATREG_RDC2	Statusregister Istwertkanal 2
*.b_STATREG_RDC3	reserviert
*.b_STATREG_RDC4	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Im Statusregister wird die Slavenummer der CANsync-Slaves eingetragen, von dem ein Istwert-Telegramm empfangen wurde.

Register	Inhalt
*.a_STATREG_RDC[0].b_STATREG_RDC1	Istwertquittung Istwertkanal 1 Slave 0
*.a_STATREG_RDC[0].b_STATREG_RDC2	Istwertquittung Istwertkanal 2 Slave 0
*.a_STATREG_RDC[0].b_STATREG_RDC3	reserviert
*.a_STATREG_RDC[0].b_STATREG_RDC4	reserviert
*.a_STATREG_RDC[1].b_STATREG_RDC1	Istwertquittung Istwertkanal 1 Slave 1
*.a_STATREG_RDC[1].b_STATREG_RDC2	Istwertquittung Istwertkanal 2 Slave 1
*.a_STATREG_RDC[1].b_STATREG_RDC3	reserviert
*.a_STATREG_RDC[1].b_STATREG_RDC4	reserviert
...	...
*.a_STATREG_RDC[31].b_STATREG_RDC1	Istwertquittung Istwertkanal 1 Slave 31
*.a_STATREG_RDC[31].b_STATREG_RDC2	Istwertquittung Istwertkanal 2 Slave 31
*.a_STATREG_RDC[31].b_STATREG_RDC3	reserviert
*.a_STATREG_RDC[31].b_STATREG_RDC4	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Wenn das entsprechende Istwert-Telegramm für einen CANsync-Slave angekommen ist, wird 16#02 in das Istwertquittungsregister eingetragen.

Die Konfigurierung der Istwert-Telegramme der anderen CANsync-Slaves erfolgt mit den folgenden Registern.

Register	Inhalt
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD0	Konfigurierung Istwert-Telegramm 1 Wort 0 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD1	Konfigurierung Istwert-Telegramm 1 Wort 1 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD2	Konfigurierung Istwert-Telegramm 1 Wort 2 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD3	Konfigurierung Istwert-Telegramm 1 Wort 3 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD0	Konfigurierung Istwert-Telegramm 2 Wort 0 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD1	Konfigurierung Istwert-Telegramm 2 Wort 1 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD2	Konfigurierung Istwert-Telegramm 2 Wort 2 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC2_WORD3	Konfigurierung Istwert-Telegramm 2 Wort 3 Slave 0
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD0	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD1	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD2	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC3_WORD3	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD0	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD1	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD2	reserviert
*.a_CFG_RDC_WORD[0].b_CFG_RDC4_WORD3	reserviert
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD0	Konfigurierung Istwert-Telegramm 1 Wort 0 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD1	Konfigurierung Istwert-Telegramm 1 Wort 1 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD2	Konfigurierung Istwert-Telegramm 1 Wort 2 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC1_WORD3	Konfigurierung Istwert-Telegramm 1 Wort 3 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD0	Konfigurierung Istwert-Telegramm 2 Wort 0 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD1	Konfigurierung Istwert-Telegramm 2 Wort 1 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD2	Konfigurierung Istwert-Telegramm 2 Wort 2 Slave 1
*.a_CFG_RDC_WORD[1].b_CFG_RDC2_WORD3	Konfigurierung Istwert-Telegramm 2 Wort 3 Slave 1
...	...
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD0	Konfigurierung Istwert-Telegramm 1 Wort 0 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD1	Konfigurierung Istwert-Telegramm 1 Wort 1 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD2	Konfigurierung Istwert-Telegramm 1 Wort 2 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC1_WORD3	Konfigurierung Istwert-Telegramm 1 Wort 3 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD0	Konfigurierung Istwert-Telegramm 2 Wort 0 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD1	Konfigurierung Istwert-Telegramm 2 Wort 1 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD2	Konfigurierung Istwert-Telegramm 2 Wort 2 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3	Konfigurierung Istwert-Telegramm 2 Wort 3 Slave 31
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD0	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD1	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD2	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC3_WORD3	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD0	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD1	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD2	reserviert
*.a_CFG_RDC_WORD[31].b_CFG_RDC4_WORD3	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Mit der Konfigurierung gibt man an, welcher Istwert an welcher Stelle im Istwert-Telegramm (des anderen CANsync-Slaves) übertragen wird.

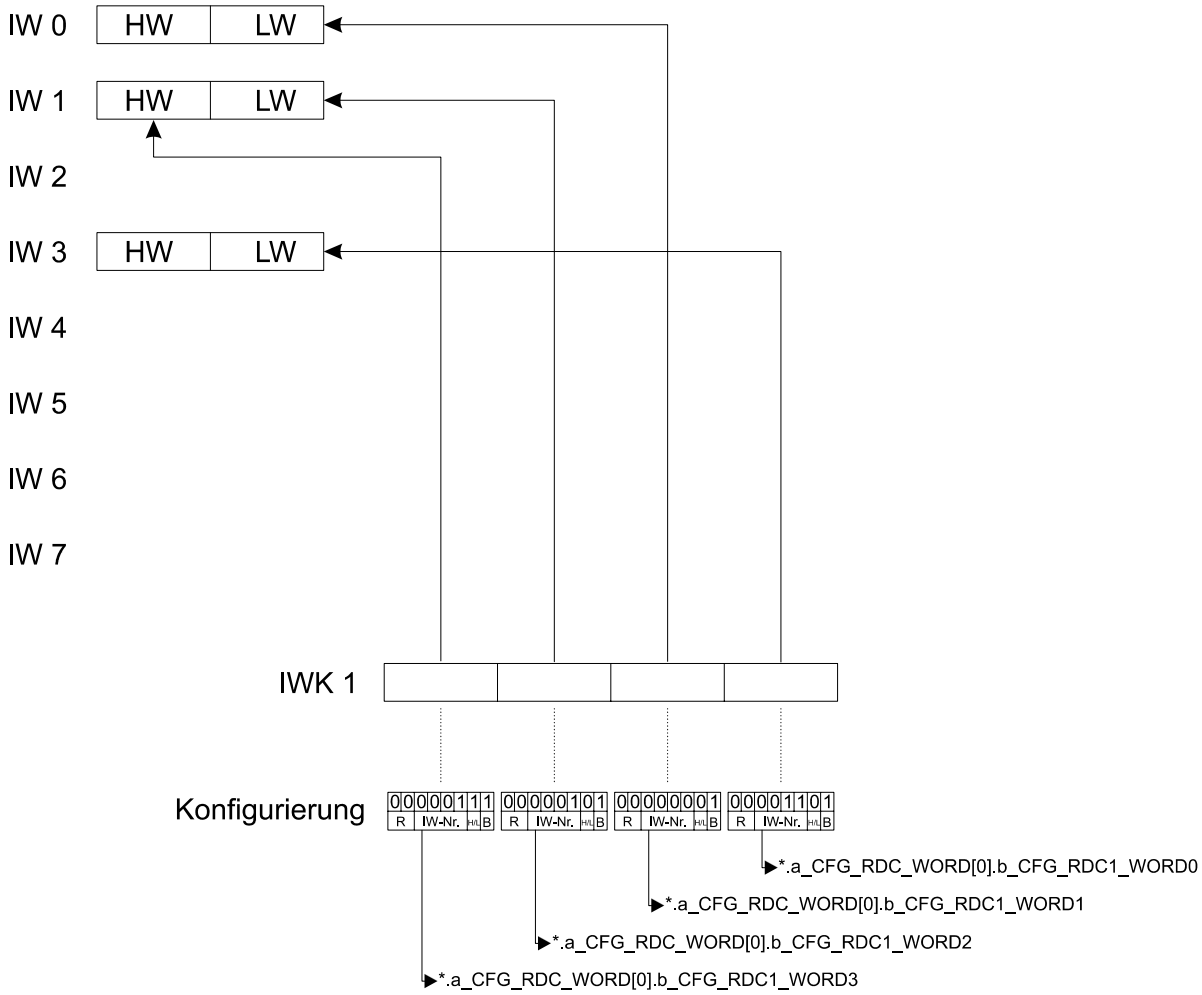
Bedeutung

Bit 0	**Belegt** Wenn = 1, wird dieses Wort des Telegramms verwendet
Bit 1	**Highword/Lowword** Wenn = 1, wird das Highword des Istwertes eingetragen
Bit 2	**Istwertnummer** Nummer des Istwertes 0 bis 7
Bit 3	
Bit 4	
Bit 5	
Bit 6	reserviert
Bit 7	

HINWEIS

Ein Wort des Telegramms wird nur dann nicht verwendet, wenn Istertnummer = 0, Highword/Lowword = 0 und Belegt = 0 gesetzt wird.

Beispiel:



*.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD0 = 16#0D

0	0	0	0	1	1	0	1
R		IW-Nr.				H/L	B

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Damit wird aus dem Istwert-Telegramm 1 des CANsync-Slaves 0 das Wort 0 gelesen und in das Low-word des Istwertes 3 geschrieben.

Register	Inhalt
*.a_RD_BMARRAY[0][0]	Istwert 0 Slave 0
*.a_RD_BMARRAY[0][1]	Istwert 1 Slave 0
*.a_RD_BMARRAY[0][2]	Istwert 2 Slave 0
*.a_RD_BMARRAY[0][3]	Istwert 3 Slave 0
*.a_RD_BMARRAY[0][4]	Istwert 4 Slave 0
*.a_RD_BMARRAY[0][5]	Istwert 5 Slave 0
*.a_RD_BMARRAY[0][6]	Istwert 6 Slave 0
*.a_RD_BMARRAY[0][7]	Istwert 7 Slave 0
*.a_RD_BMARRAY[0][8]	reserviert
...	...

Register	Inhalt
*.a_RD_BMARRAY[0][15]	reserviert
*.a_RD_BMARRAY[1][0]	Istwert 0 Slave 1
*.a_RD_BMARRAY[1][1]	Istwert 1 Slave 1
*.a_RD_BMARRAY[1][2]	Istwert 2 Slave 1
*.a_RD_BMARRAY[1][3]	Istwert 3 Slave 1
*.a_RD_BMARRAY[1][4]	Istwert 4 Slave 1
*.a_RD_BMARRAY[1][5]	Istwert 5 Slave 1
*.a_RD_BMARRAY[1][6]	Istwert 6 Slave 1
*.a_RD_BMARRAY[1][7]	Istwert 7 Slave 1
*.a_RD_BMARRAY[1][8]	reserviert
...	...
*.a_RD_BMARRAY[31][0]	Istwert 0 Slave 31
*.a_RD_BMARRAY[31][1]	Istwert 1 Slave 31
*.a_RD_BMARRAY[31][2]	Istwert 2 Slave 31
*.a_RD_BMARRAY[31][3]	Istwert 3 Slave 31
*.a_RD_BMARRAY[31][4]	Istwert 4 Slave 31
*.a_RD_BMARRAY[31][5]	Istwert 5 Slave 31
*.a_RD_BMARRAY[31][6]	Istwert 6 Slave 31
*.a_RD_BMARRAY[31][7]	Istwert 7 Slave 31
*.a_RD_BMARRAY[31][8]	reserviert
...	...
*.a_RD_BMARRAY[31][15]	reserviert

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Die Istwerte können Wort- oder Doppelwort-Istwerte sein.

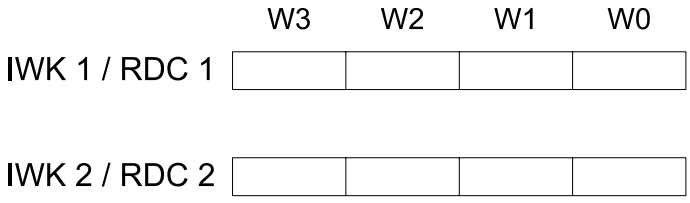
HINWEIS

Bei der Anzeige von _CANsync_CTRL_SL im PROPROG wt II Watch-Fenster kann zwischen den eckigen Klammern ein Punkt stehen.

Erläuterung zur Benutzung der Istwertkanäle

Ein CANsync-Slave kann auch alle Istwert-Telegramme der anderen CANsync-Slaves auswerten. Die Anforderung ist jedoch nur vom CANsync-Master aus möglich.

Im Synchron-Betrieb stehen zwei Istwertkanäle (Kanal 1 und 2; auch RDC1 und RDC2) zur Verfügung. In den Istwertkanälen 1 und 2 werden die Istwert-Telegramme 1 und 2 gesendet.



Beide Istwert-Telegramme bestehen aus jeweils vier Worten (W0 bis W3). Nach der CANsync-Initialisierung muß zumindest einmal angegeben werden, wie die Belegung dieser Worte ist. Dies kann im Initialisierungsprogramm erfolgen.
Dazu trägt man im Bereich *.a_CFG_RDC_WORD[0].b_CFG_RDC1_WORD0 bis *.a_CFG_RDC_WORD[31].b_CFG_RDC2_WORD3 für jedes belegte Wort der Istwert-Telegramme (jedes CANsync-Slaves) ein, welcher Istwert der anderen CANsync-Slaves an dieser Stelle übertragen wird. Istwertnummern sind im Bereich von 0 bis 7 gültig. Für Doppelwort-Istwerte müssen zwei Worte im Telegramm verwendet werden.

Die Konfigurierung kann auch während des aktiven Betriebs geändert werden. Die Änderung wird spätestens im nächsten CANsync-Intervall übernommen. Die Konfigurierungsdaten werden von der CANsync-Anschaltung gelesen, wenn das entsprechende Istwert-Telegramm empfangen wird.

Die Istwert-Telegramm-Auswertung erfolgt zu Beginn der CANsync Event-Task. Dazu gibt es zwei Auswertungsmethoden: Bei der ersten Methode fragt man die Statusregister der Istwertkanäle ab (*.b_STATREG_RDC1 oder *.b_STATREG_RDC2). In diesen Registern wird die Slavenummer des CANsync-Slaves eingetragen, von dem das entsprechende Istwert-Telegramm empfangen wurde. Dann kann man die Istwerte aus den Registern (*.a_RD_BMARRAY[0][0] bis *.a_RD_BMARRAY[31][7]) dieses CANsync-Slaves auslesen und das Statusregister auf z. B. 16#FF setzen, um die nächste Eintragung richtig erkennen zu können.

Bei der zweiten Methode fragt man direkt die Istwertquittung
(*.a_STATREG_RDC[0].b_STATREG_RDC1 bis *.a_STATREG_RDC[31].b_STATREG_RDC2) für einen CANsync-Slave (dessen Istwert-Telegramme mitgelesen werden) ab. In diesem Quittungsregister wird der Empfang eines Istwert-Telegramms mit 16#02 gekennzeichnet. Dann kann man die Istwerte aus den Registern (*.a_RD_BMARRAY[0][0] bis *.a_RD_BMARRAY[31][7]) dieses CANsync-Slaves auslesen. Auch hier muß man das Statusregister anschließend mit einem anderen Wert beschreiben, um das Wiedereintragen der Quittung zu erkennen.

Die Zuordnung welche Istwerte gelesen werden dürfen, wenn ein Istwert-Telegramm empfangen wurde, muß entsprechend der Istwertkonfigurierung im Applikationsprogramm erfolgen.

Die Anforderung an einen CANsync-Slave, sein Istwert-Telegramm zu senden, erfolgt über das Sollwert-Telegramm des CANsync-Masters. Die Einstellung kann nur am CANsync-Master erfolgen.

Kommando- und Antwortkanal

Aktionskommando

Aktionskommandos werden in den folgenden Registern gemeldet.

Register	Inhalt
*.b_STATREG_AKT_CMD	Statusregister Aktionskommando
*.b_AKT_CMD	Aktionskommando
*.b_DATA_BYTE_AKT_CMD	Datenbyte 0 (DB)
*.w_DATAWORD_AKT_CMD	Datenwort 1 (DW)
*.b_STATREG_CONTROLWORD	Statusregister Steuerwort
*.w_CONTROLWORD	Steuerwort
*.b_STATREG_SYNC_MODUS	Statusregister SYNC-Modus
*.b_SYNC_MODUS	SYNC-Modus

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Das Aktionskommando, das als Broadcastkommando vom CANsync-Master gesendet wird, wird nur gemeldet, wenn der CANsync-Slave angesprochen wird, d. h. das entsprechende Bit in der Bitleiste gesetzt ist (siehe “Registerstruktur und Funktion des Ωmega CANsync-Master” auf Seite 144).

Wenn das Aktionskommando ein Steuerwortkommando (Kommandobyte = 16#01) ist, wird das Datenwort des Kommandos im *.w_CONTROLWORD als Steuerwort und im *.b_STATREG_CONTROLWORD als Empfangsanzeige 16#02 eingetragen.

Bei alle anderen Aktionskommandos werden die Daten in *.b_DATA_BYTE_AKT_CMD und *.w_DATAWORD_AKT_CMD eingetragen. Die Quittung (Empfangsanzeige 16#02) steht dann im *.b_STATREG_AKT_CMD. Da in jedem CANsync-Intervall ein neues Aktionskommando übertragen werden kann, sollte das Statusregister des Aktionskommandos in jeder CANsync Event-Task ausgewertet werden.

Parameterkommando

Parameterkommandos werden in folgenden Registern gemeldet.

Register	Inhalt
*.b_STEUREG_PAR_CMD	Steuerregister Parameterkommando
*.b_STATREG_PAR_CMD	Statusregister Parameterkommando
*.w_ERR_NR_PAR_CMD	Fehlernummer Parameterkommando
*.d_DATA_PAR_CMD	Daten Parameterkommando
*.w_PAR_NR	Parameternummer

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Bedeutung der Steuer- und Statusregister

Bit 0	**Aktiv** wird auf 1 gesetzt, wenn Auftrag empfangen wurde muß auf 0 gesetzt werden, wenn Auftrag bearbeitet wurde
Bit 1	**Read** = 1: Parameter schreiben
Bit 2	**Write** = 1: Parameter lesen
Bit 3	**Change** = 1: Anzeige, daß Parameterkommando vom CANsync-Master geändert wurde, muß mit Reset quittiert werden
Bit 4	**Format** 0: Wortparameter 1: Doppelwort-Parameter
Bit 5	**Fehleranzeige** muß auf 1 gesetzt werden, wenn der Auftrag einen Fehler erzeugt
Bit 6	**Busy** = 1: Antwort ist noch nicht gesendet = 0: Antwort wurde an CANsync-Master gesendet
Bit 7	**Reset** = 1 bestätigt, daß die Kommandoänderung am Bit Change gesehen wurde und das alte Kommando nicht weiter bearbeitet wird.

Ablauf eines Parameterzugriffs

Die Bearbeitung von Parameterkommandos kann sowohl in der CANsync Event-Task als auch im restlichen Programm erfolgen. Zur Sicherung der Datenkonsistenz erfolgt die Auftragserteilung und die Auswertung der Rückantwort über das Steuerregister und das Statusregister. Um eine konfliktfreie Durchführung des Auftrags zu gewährleisten, muß vom Applikationsprogramm aus immer zuerst das Steuerregister und dann das Statusregister gelesen werden und wenn ein neuer Wert geschrieben wird, immer zuerst das Statusregister und dann das Steuerregister gesetzt werden. (Für die CANsync-Anschaltung ist das Steuerregister das entscheidende Register.)

Wenn ein Parameter-Schreibauftrag empfangen wurde, steht in *.d_DATA_PAR_CMD der zu schreibende Wert und in *.w_PAR_NR die Parameternummer. Im Steuerregister (*.b_STEUREG_PAR_CMD) und im Statusregister (*.b_STATREG_PAR_CMD) wird mit 16#45 ein Wortschreibkommando und mit 16#55 ein Doppelwort-Schreibkommando gemeldet.

Wenn vom Applikationsprogramm der geforderte Parameter fehlerfrei beschrieben wurde, muß dies im Statusregister und im Steuerregister durch das Löschen des Bit 0 (Aktiv) angezeigt werden.

Sobald das Antwort-Telegramm zum CANsync-Master gesendet wurde, wird das Bit 6 (Busy) von der CANsync-Anschaltung gelöscht.

Wenn das Applikationsprogramm den Schreibzugriff nicht ausführen kann, muß eine Fehlernummer in *.w_ERR_NR_PAR_CMD eingetragen werden und in den Registern das Bit 5 (Fehleranzeige) gesetzt werden und das Bit 0 (Aktiv) gelöscht werden. Dann wird an den CANsync-Master ein Antwort-Telegramm mit der eingetragenen Fehlernummer gesendet.

Sobald das Antwort-Telegramm zum CANsync-Master gesendet wurde, wird das Bit 6 (Busy) von der CANsync-Anschaltung gelöscht.

Der Parameter-Leseauftrag läuft dazu ähnlich ab, nur mit dem Unterschied, daß der Befehl unabhängig vom Format 16#43 lautet und daß zur Antwort die Daten in *.d_DATA_PAR_CMD eingetragen werden müssen. Vor der Quittierung durch das Löschen des Bit 0 (Aktiv) muß das tatsächliche Format des Parameters im Bit 4 (Format) eingetragen werden, damit die Rückantwort an den CANsync-Master richtig erzeugt werden kann.

Wenn der CANsync-Master das Parameterkommando ändert (Parameternummer) wird dies durch das gesetzte Bit 3 (Change) angezeigt. Dann muß der alte Befehl nicht mehr beantwortet werden. Das Applikationsprogramm muß die Änderung erkennen und dies durch das Setzen des Bit 7 (Reset) quittieren. Dann wird das aktuelle Parameterkommando eingetragen.

Fehlercode

Wert	Bedeutung
16#0000	Kein Fehler aufgetreten
16#FFFF	Fehler aufgetreten
16#FFFE	Wert kleiner Minimalwert
16#FFFD	Wert größer Maximalwert
16#FFFC	Element nicht veränderbar
16#FFFB	Element nicht vorhanden
16#FFFA	Daten nicht verfügbar (z. B. in Bearbeitung)
16#FFF9	Fehler beim Datenformat

Up/Download-Kommando

Upload- und Download-Kommandos werden in den folgenden Registern angezeigt.

Register	Inhalt
*.b_STEUREG_UPDOWNBLK0	reserviert
*.b_STATREG_UPDOWNBLK0	reserviert
*.w_ERR_NR_UPDOWNBLK0	reserviert
*.w_SUBSL_NR_UPDOWNBLK0	reserviert
*.d_BASE_ADR_UPDOWNBLK0	reserviert
*.w_LENGTH_UPDOWNBLK0	reserviert
*.w_COUNTER_UPDOWNBLK0	reserviert
*.a_DATA_UPDOWNBLK0[0 bis 74]	reserviert
*.b_STEUREG_UPDOWNBLK1	Steuerregister Up/Download Block 1
*.b_STATREG_UPDOWNBLK1	Statusregister Up/Download Block 1
*.b_EN_UPDOWNBLK1	Freigabe Block 1
*.w_ERR_NR_UPDOWNBLK1	Fehlernummer Up/Download Block 1
*.d_BASE_ADR_UPDOWNBLK1	Basisadresse Up/Download Block 1
*.w_LENGTH_UPDOWNBLK1	Länge in Bytes Up/Download Block 1
*.w_COUNTER_UPDOWNBLK1	Zähler Up/Download Block 1
*.a_DATA_UPDOWNBLK1[0 bis 74]	Datenblock Up/Download Block 1

(* entspricht zum Beispiel _CANsync_CTRL_SL)

HINWEIS

Der Block 0 ist für OmegaOS reserviert.

Bedeutung der Steuer- und Statusregister

Bit 0	**Aktiv** wird auf 1 gesetzt, wenn Auftrag empfangen wurde muß auf 0 gesetzt werden, wenn Auftrag bearbeitet wurde
Bit 1	**Change** = 1: Anzeige, daß Parameterkommando vom CANsync-Master geändert wurde, muß mit Reset quittiert werden
Bit 2 Bit 3	**Mode** Bit3 Bit2 0 0: reserviert 0 1: Initialisierung 1 0: laufender Up/Download 1 1: Blockende
Bit 4	**Up/Download** = 0: Upload = 1: Download
Bit 5	**Fehleranzeige** muß auf 1 gesetzt werden, wenn der Auftrag einen Fehler erzeugt
Bit 6	**Busy** = 1: Antwort ist noch nicht gesendet = 0: Antwort wurde an CANsync-Master gesendet
Bit 7	**Reset** = 1 bestätigt, daß die Kommandoänderung am Bit Change gesehen wurde und das alte Kommando nicht weiter bearbeitet wird.

Ablauf eines Up/Download-Auftrags im Block 1

Up/Download-Aufträge für den CANsync-Slave werden im Block 1 oder im Einzeltelegramm-Bereich empfangen.

Wenn die Blocklänge > 300 Bytes (75 Doppelworte) ist, wird immer der Einzeltelegramm-Bereich verwendet. Bei einer kürzeren Blocklänge wird die Freigabe auf *.b_EN_UPDOWNBLK1 ausgewertet. Wenn dort ein Wert ungleich Null steht, wird der Auftrag im Block 1 gemeldet, sonst wird er im Einzeltelegramm-Bereich eingetragen.

Die Auswertung von Up/Downloadkommandos können sowohl in der CANsync Event-Task als auch im restlichen Programm verwendet werden. Zur Sicherung der Datenkonsistenz erfolgt die Auftragserteilung und die Auswertung der Rückantwort über das Steuerregister und das Statusregister. Um eine konfliktfreie Durchführung des Auftrags zu gewährtleisten, muß vom Applikationsprogramm aus immer zuerst das Steuerregister und dann das Statusregister gelesen werden und wenn ein neuer Wert geschrieben wird, immer zuerst das Statusregister und dann das Steuerregister gesetzt werden. (Für die CANsync-Anschaltung ist das Steuerregister das entscheidende Register.)

Wenn ein Download gemeldet wird, wird die Basisadresse im *.d_BASE_ADR_UPDOWNBLK1 und die Blocklänge im *.w_LENGTH_UPDOWNBLK1 eingetragen.

Die Basisadressen 16#00000000 bis 16#000000FF sind für Aufträge des Betriebssystems reserviert.

Im Statusregister *.b_STATREG_UPDOWNBLK1 und im Steuerregister *.b_STEUREG_UPDOWNBLK1 wird der Wert 16#55 gemeldet. Wenn der Download erlaubt ist, muß das Applikationsprogramm das Bit 0 (Aktiv) löschen. Dann wird der gesamte Block vom CANsync-Master empfangen (der Fortschritt kann im Byte-Zähler in *.w_COUNTER_UPDOWNBLK1 abgelesen werden) und am Ende in den Registern 16#5D gemeldet. Wenn die Daten aus dem Bereich *.a_DATA_UPDOWNBLK1[0] bis *.a_DATA_UPDOWNBLK1[74] übernommen wurden, muß man das Bit 0 (Aktiv) löschen. Als Anzeige, daß das Antwort-Telegramm zum CANsync-Master gesendet wurde, wird das Bit 6 (Busy) gelöscht.

Falls im CANsync-Slave bei der Ausführung des Auftrags ein Fehler auftritt, muß die Fehlernummer in *.w_ERR_NR_UPDOWNBLK1 eingetragen werden, das Bit 5 (Fehleranzeige) gesetzt und das Bit 0 (Aktiv) gelöscht. Dann wird an den CANsync-Master ein entsprechendes Fehlertelegramm gesendet und damit der Download abgebrochen.

Der Upload-Auftrag läuft dazu ähnlich ab, nur mit dem Unterschied, daß der Befehl 16#05 lautet und daß die Daten vor der Quittierung des Initialisierungs-Telegramms in den Blockbereich eingetragen werden müssen.

Wenn der CANsync-Master das Up/Downloadkommando ändert wird dies durch das gesetzte Bit 1 (Change) angezeigt. Dann muß der alte Befehl abgebrochen werden. Das Applikationsprogramm muß die Änderung erkennen und dies durch das Setzen des Bit 7 (Reset) quittieren. Dann wird das aktuelle Up/Downloadkommando eingetragen.

Die Fehlernummern, die vom CANsync-Slave eingetragen werden können, sind > 16#00FF.

Fehlernummern Up/Download, die die CANsync-Slave-Anschaltung an den CANsync-Master sendet; Anzeige nur in CANsync-Master:

Fehlernummer	Bedeutung
16#0001	CANsync-Slave quittiert falsche Blocknummer
16#0002	eingetragenen Länge > 300 Bytes
16#0100	CANsync-Slave erwartet Block mit Nummer, im Zähler eingetragen ist
16#0101	CANsync-Slave erwartet Block-Ende
16#0102	CANsync-Slave erwartet noch nicht Block-Ende
16#0103	CANsync-Slave bricht Up/Download ab
16#0104	Up/Download nicht möglich
16#0105	Basisadresse nicht erlaubt
16#0106	reserviert
16#0107	Blocklänge > maximale Blocklänge des CANsync-Slaves
16#0108	Telegramm-Modefehler (in dieser Phase nicht erlaubter Mode)

Einzeltelegrammbereich

Register	Inhalt
*.b_STEUREG_UPDOWN_SINGLE	Steuerregister Up/Download
*.b_STATREG_UPDOWN_SINGLE	Statusregister Up/Download
*.w_ERR_NR_UPDOWN_SINGLE	Fehlernummer Up/Download
*.d_BASE_ADR_UPDOWN_SINGLE	Basisadresse Up/Download
*.w_LENGTH_UPDOWN_SINGLE	Länge in Bytes Up/Download
*.w_COUNTER_UPDOWN_SINGLE	Zähler Up/Download
*.d_DATA_DW_UPDOWN_SINGLE	Daten Up/Download
*.w_DATA_W_UPDOWN_SINGLE	Daten Up/Download

(* entspricht zum Beispiel _CANsync_CTRL_SL)

Bedeutung der Steuer- und Statusregister

Bit 0	**Aktiv** wird auf 1 gesetzt, wenn Auftrag empfangen wurde muß auf 0 gesetzt werden, wenn Auftrag bearbeitet wurde
Bit 1	**Change** = 1: Anzeige, daß Parameterkommando vom CANsync-Master geändert wurde, muß mit Reset quittiert werden
Bit 2 Bit 3	**Mode** Bit3 Bit2 0 0: reserviert 0 1: Initialisierung 1 0: laufender Up/Download 1 1: Blockende
Bit 4	**Up/Download** = 0: Upload = 1: Download
Bit 5	**Fehleranzeige** muß auf 1 gesetzt werden, wenn der Auftrag einen Fehler erzeugt
Bit 6	**Busy** = 1: Antwort ist noch nicht gesendet = 0: Antwort wurde an CANsync-Master gesendet
Bit 7	**Reset** = 1 bestätigt, daß die Kommandoänderung am Bit Change gesehen wurde und das alte Kommando nicht weiter bearbeitet wird.

Die Verwendung des Einzeltelegrammbereich erfolgt analog zum Block 1. Nur mit dem Unterschied, daß jedes Telegramm einzeln quittiert werden muß und daß der entsprechende Mode eingetragen werden muß.

7.3 CANsync-Funktionsbausteine

7.3.1 Funktionsbausteine für den synchronisierten CAN - Übersicht

Zusätzlich zu den Standardfunktionen können Sie herstellerdefinierte Funktionen verwenden, wenn Sie herstellerdefinierte Bibliotheken in einem Projekt angemeldet haben.

Anmerkung: Das Anmelden von Bibliotheken ist in der allgemeinen Hilfe beschrieben.

Folgende Funktionsbausteine für synchronisierten CAN sind verfügbar:

Funktion	Kurzbeschreibung
CANsync_BC_MA0	sendet Broadcastkommando des CANsync-Masters in Sendebereich 0 (höchste Priorität)
CANsync_BC_MA1	sendet Broadcastkommando des CANsync-Masters in Sendebereich 1 (mittlere Priorität)
CANsync_BC_MA2	sendet Broadcastkommando des CANsync-Masters in Sendebereich 2 (niedrige Priorität)
CANsync_BC_SL	empfängt Broadcastkommandos des CANsync-Masters
CANsync_COMM_CONTROL_MA	konfiguriert die Benutzung des Kommandokanals einer CANsync-Master-Anschaltung
CANsync_CONTROLWORD_MA	sendet Steuerwortkommandos des CANsync-Masters
CANsync_CONTROLWORD_SL	empfängt Steuerwort des CANsync-Masters in einer CANsync-Slave-Anschaltung
CANsync_INIT	initialisiert eine CANsync-Anschaltung
CANsync_MODE_MA	stellt Betriebsart einer CANsync-Master-Anschaltung ein
CANsync_MODE_SL	stellt Betriebsart einer CANsync-Slave-Anschaltung ein
CANsync_PAR_READ_MA	Ωmega-Master fordert über CANsync einen Parameterwert vom Ωmega-Slave an
CANsync_PAR_SL	erkennt Parameteranforderung oder übertragenen Parameter
CANsync_PAR_WRITE_MA	Ωmega-Master sendet über CANsync einen Parameterwert an Ωmega-Slave
CANsync_PD_CFG_MA	konfiguriert Belegung der Sollwertkanäle der CAN-Anschaltung für einen CANsync-Master
CANsync_PD_CFG_READ_MA	konfiguriert Belegung der Istwertkanäle der CAN-Anschaltung für einen CANsync-Master
CANsync_PD_CFG_READ_SL	konfiguriert Belegung der Istwertkanäle der CAN-Anschaltung für einen CANsync-Slave
CANsync_PD_CFG_SL	konfiguriert Belegung der Sollwert- und der Istwertkanäle der CAN-Anschaltung für einen CANsync-Slave
CANsync_PD_COMM_MA	kopiert die Prozeßdaten (Sollwerte und Istwerte der CANsync-Anschaltung für einen CANsync-Master
CANsync_PD_COMM_READ_MA	kopiert in einer CANsync Anschaltung (Master) die Prozeßdaten-Istwerte von einem CANsync-Slave
CANsync_PD_COMM_READ_SL	kopiert in einer CANsync-Anschaltung (Slave) die Prozeßdaten-Istwerte von einem CANsync-Slave
CANsync_PD_COMM_SL	kopiert die Prozeßdaten (Soll- und Istwerte) der CANsync-Anschaltung für einen CANsync-Slave
CANsync_SL_TYP_INIT	CANsync Slave-Typen (Initialisierung)
CANsync_UPDOWNLOAD_MA	CANsync Updownload Master
CANsync_UPDOWNLOAD_SL	CANsync Updownload Slave

7.3.2 CANsync_BC_MA0

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um ein Broadcastkommando des CANsync-Masters im Sendebereich 0 (höchste Priorität) zu senden.

HINWEIS

Der FB CANsync_BC_MA0 verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
us_BC_CMD_NR	USINT	Kommandonummer des Broadcast-kommandos
si_BC_BYTE	SINT	Datenbyte des Broadcastkomman-dos
i_BC_WORD	INT	Datenwort des Broadcastkomman-dos
d_SL_MASK	DWORD	Bitleiste an welche CANsync-Slaves das Broadcast-Kommando geschickt werden soll
x_EN	BOOL	Freigabe

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
x_OK	BOOL	OK-Bit

Es gibt beim CANsync 3 Sendebereiche für die Broadcast-Telegramme mit den Broadcast-Kommandos (Sendebereich 0 bis 2). In einem CANsync-Intervall kann nur ein Broadcast-Telegramm gesendet werden. Die höchste Priorität hat der Sendebereich 0. Der FB CANsync_BC_MA0 nutzt den Sendebereich 0 zum Senden des Broadcast-Telegramms.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Eingang us_BC_CMD_NR:

Am Eingang us_BC_CMD_NR wird die Kommandonummer des Broadcast-Kommandos angegeben.

Eingänge si_BC_BYTE, i_BC_WORD:

An den Eingängen si_BC_BYTE und i_BC_WORD sind je nach Kommandonummer die zugehörigen Daten anzuschließen (siehe weiter unten Liste der Broadcast-Kommandos).

Eingang d_SL_MASK:

Am Eingang d_SL_MASK wird angegeben, welche CANsync-Slaves das Kommando empfangen sollen. Dazu muß für jeden CANsync-Slave die seiner Slavenummer entsprechende Bitnummer auf TRUE gesetzt werden. Für den CANsync-Slave mit der Slavenummer 0 wird das Bit 0 von d_SL_MASK auf TRUE gesetzt, für den CANsync-Slave mit der Slavenummer 1 wird das Bit 1 von d_SL_MASK auf TRUE gesetzt usw. Wenn alle CANsync-Slaves angesprochen werden sollen, muß d_SL_MASK = 16#FFFFFFFF gesetzt werden.

Eingang x_EN:

Wenn der Eingang x_EN auf TRUE gesetzt wird, wird das Broadcast-Kommando zum Senden eingetragen. Die CANsync-Anschaltung sendet das Broadcast-Kommando dann im Broadcast-Telegramm an die ausgewählten CANsync-Slaves (Eingang d_SL_MASK). Solange x_EN auf TRUE bleibt, wird bei jedem Aufruf des FB das Broadcast-Kommando zum Senden eingetragen. Damit würde jedoch der Kommandokanal des CANsync dauernd belegt. Deshalb sollte der Eingang x_EN nur für ein CANsync-Intervall auf TRUE gesetzt sein, um das Broadcast-Kommando einmal abzuschicken.

Ausgang x_OK:

Der Ausgang x_OK zeigt mit TRUE an, daß das letzte Broadcast-Telegramm gesendet wurde. Der Ausgang x_OK ist FALSE wenn kein Broadcast-Telegramm gesendet wurde.

Liste der Broadcastkommandos:

Kommando-nummer	**Bedeutung**
1	Steuerwort si_BC_BYTE: nicht benutzt i_BC_WORD: Steuerwort
2 bis 127	reserviert
128 – 255	stehen dem Anwender zur Verfügung

7.3.3 CANsync_BC_MA1

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um ein Broadcastkommando des CANsync-Masters im Sendebereich 1 (mittlere Priorität) zu senden.

HINWEIS

Der FB CANsync_BC_MA1 verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
us_BC_CMD_NR	USINT	Kommandonummer des Broadcast-kommandos
si_BC_BYTE	SINT	Datenbyte des Broadcastkomman-dos
i_BC_WORD	INT	Datenwort des Broadcastkomman-dos
d_SL_MASK	DWORD	Bitleiste an welche CANsync-Slaves das Broadcast-Kommando geschickt werden soll
x_EN	BOOL	Freigabe

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
x_OK	BOOL	OK-Bit

Es gibt beim CANsync 3 Sendebereiche für die Broadcast-Telegramme mit den Broadcast-Kommandos (Sendebereich 0 bis 2). In einem CANsync-Intervall kann nur ein Broadcast-Telegramm gesendet werden. Die mittlere Priorität hat der Sendebereich 1. Der FB CANsync_BC_MA1 nutzt den Sendebereich 1 zum Senden des Broadcast-Telegramms.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Eingang us_BC_CMD_NR:

Am Eingang us_BC_CMD_NR wird die Kommandonummer des Broadcast-Kommandos angegeben.

Eingänge si_BC_BYTE, i_BC_WORD:

An den Eingängen si_BC_BYTE und i_BC_WORD sind je nach Kommandonummer die zugehörigen Daten anzuschließen (siehe weiter unten Liste der Broadcast-Kommandos).

Eingang d_SL_MASK:

Am Eingang d_SL_MASK wird angegeben, welche CANsync-Slaves das Kommando empfangen sollen. Dazu muß für jeden CANsync-Slave die seiner Slavenummer entsprechende Bitnummer auf TRUE gesetzt werden. Für den CANsync-Slave mit der Slavenummer 0 wird das Bit 0 von d_SL_MASK auf TRUE gesetzt, für den CANsync-Slave mit der Slavenummer 1 wird das Bit 1 von d_SL_MASK auf TRUE gesetzt usw. Wenn alle CANsync-Slaves angesprochen werden sollen, muß d_SL_MASK = 16#FFFFFFFF gesetzt werden.

Eingang x_EN:

Wenn der Eingang x_EN auf TRUE gesetzt wird, wird das Broadcast-Kommando zum Senden eingetragen. Die CANsync-Anschaltung sendet das Broadcast-Kommando dann im Broadcast-Telegramm an die ausgewählten CANsync-Slaves (Eingang d_SL_MASK). Solange x_EN auf TRUE bleibt, wird bei jedem Aufruf des FB das Broadcast-Kommando zum Senden eingetragen. Damit würde jedoch der Kommandokanal des CANsync dauernd belegt. Deshalb sollte der Eingang x_EN nur für ein CANsync-Intervall auf TRUE gesetzt sein, um das Broadcast-Kommando einmal abzuschicken.

Ausgang x_OK:

Der Ausgang x_OK zeigt mit TRUE an, daß das letzte Broadcast-Telegramm gesendet wurde. Der Ausgang x_OK ist FALSE wenn kein Broadcast-Telegramm gesendet wurde.

Liste der Broadcastkommandos:

Kommando-nummer	**Bedeutung**
1	Steuerwort si_BC_BYTE: nicht benutzt i_BC_WORD: Steuerwort
2 bis 127	reserviert
128 – 255	stehen dem Anwender zur Verfügung

7.3.4 CANsync_BC_MA2

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um ein Broadcastkommando des CANsync-Masters im Sendebereich 2 (niedrigste Priorität) zu senden.

HINWEIS

Der FB CANsync_BC_MA2 verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
us_BC_CMD_NR	USINT	Kommandonummer des Broadcast-kommandos
si_BC_BYTE	SINT	Datenbyte des Broadcastkomman-dos
i_BC_WORD	INT	Datenwort des Broadcastkomman-dos
d_SL_MASK	DWORD	Bitleiste an welche CANsync-Slaves das Broadcast-Kommando geschickt werden soll
x_EN	BOOL	Freigabe

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
x_OK	BOOL	OK-Bit

Es gibt beim CANsync 3 Sendebereiche für die Broadcast-Telegramme mit den Broadcast-Kommandos (Sendebereich 0 bis 2). In einem CANsync-Intervall kann nur ein Broadcast-Telegramm gesendet werden. Die niedrigste Priorität hat der Sendebereich 2. Der FB CANsync_BC_MA2 nutzt den Sendebereich 2 zum Senden des Broadcast-Telegramms.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Eingang us_BC_CMD_NR:

Am Eingang us_BC_CMD_NR wird die Kommandonummer des Broadcast-Kommandos angegeben.

Eingänge si_BC_BYTE, i_BC_WORD:

An den Eingängen si_BC_BYTE und i_BC_WORD sind je nach Kommandonummer die zugehörigen Daten anzuschließen (siehe weiter unten Liste der Broadcast-Kommandos).

Eingang d_SL_MASK:

Am Eingang d_SL_MASK wird angegeben, welche CANsync-Slaves das Kommando empfangen sollen. Dazu muß für jeden CANsync-Slave die seiner Slavenummer entsprechende Bitnummer auf TRUE gesetzt werden. Für den CANsync-Slave mit der Slavenummer 0 wird das Bit 0 von d_SL_MASK auf TRUE gesetzt, für den CANsync-Slave mit der Slavenummer 1 wird das Bit 1 von d_SL_MASK auf TRUE gesetzt usw. Wenn alle CANsync-Slaves angesprochen werden sollen, muß d_SL_MASK = 16#FFFFFFFF gesetzt werden.

Eingang x_EN:

Wenn der Eingang x_EN auf TRUE gesetzt wird, wird das Broadcast-Kommando zum Senden eingetragen. Die CANsync-Anschaltung sendet das Broadcast-Kommando dann im Broadcast-Telegramm an die ausgewählten CANsync-Slaves (Eingang d_SL_MASK). Solange x_EN auf TRUE bleibt, wird bei jedem Aufruf des FB das Broadcast-Kommando zum Senden eingetragen. Damit würde jedoch der Kommandokanal des CANsync dauernd belegt. Deshalb sollte der Eingang x_EN nur für ein CANsync-Intervall auf TRUE gesetzt sein, um das Broadcast-Kommando einmal abzuschicken.

Ausgang x_OK:

Der Ausgang x_OK zeigt mit TRUE an, daß das letzte Broadcast-Telegramm gesendet wurde. Der Ausgang x_OK ist FALSE wenn kein Broadcast-Telegramm gesendet wurde.

Liste der Broadcastkommandos:

Kommando-nummer	**Bedeutung**
1	Steuerwort si_BC_BYTE: nicht benutzt i_BC_WORD: Steuerwort
2 bis 127	reserviert
128 – 255	stehen dem Anwender zur Verfügung

7.3.5 CANsync_BC_SL

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um mit einem CANsync-Slave ein Broadcastkommando des CANsync-Masters zu empfangen.

HINWEIS

Der FB CANsync_BC_SL verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
us_BC_CMD_NR	USINT	Kommandonummer des Broadcast-kommandos
si_BC_BYTE	SINT	Datenbyte des Broadcastkomman-dos
i_BC_WORD	INT	Datenwort des Broadcastkomman-dos
si_BC_RECEIVED	SINT 0, 2	Anzeige, daß ein Kommando emp-fangen wurde

Dieser FB zeigt an si_BC_RECEIVED an, daß ein Broadcast-Kommando des CANsync-Masters empfangen wurde und gibt die Kommandonummer (us_BC_CMD_NR) sowie den Inhalt des Broadcast-Kommandos (si_BC_BYTE, i_BC_WORD) aus.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_SL_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) auf Ωmega Drive-Line II

```
_CANsync_CTRL_SL  AT  %MB3.100000  :  CANsync_SL_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_SL`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_SL_CTRL_BMSTRUCT`	der Datentyp

%MB3.100000	die Basisadresse der CANsync-Anschaltung 1 auf Ωmega Drive-Line II

Ausgang us_BC_CMD_NR:

Am Ausgang us_BC_CMD_NR wird die Kommandonummer des Broadcast-Kommandos ausgegeben.

Ausgänge si_BC_BYTE, i_BC_WORD:

An den Ausgängen si_BC_BYTE und i_BC_WORD sind je nach Kommandonummer die zugehörigen Daten zu lesen.

Ausgang si_BC_RECEIVED:

Am Ausgang si_BC_RECEIVED wird angezeigt, ob ein Broadcast-Telegramm (mit einem Broadcast-Kommando) empfangen wurde. Dann zeigt si_BC_RECEIVED eine 2 an. Andernfalls zeigt si_BC_RECEIVED eine 0 an. Nur wenn das Broadcast-Telegramm empfangen wurde, werden die zugehörigen Daten (us_BC_CMD_NR, si_BC_BYTE, i_BC_WORD) ausgelesen. Ansonsten werden die alten Werte weiter angezeigt.

Liste der Broadcastkommandos:

Kommando-nummer	**Bedeutung**
1	Steuerwort si_BC_BYTE: nicht benutzt i_BC_WORD: Steuerwort
2 bis 127	reserviert
128 – 255	stehen dem Anwender zur Verfügung

7.3.6 CANsync_COMM_CONTROL_MA

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um die Benutzung des Kommandokanals einer CANsync-Anschaltung zu konfigurieren.

HINWEIS

Der FB CANsync_COMM_CONTROL_MA verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
si_SL_NR_CTRL	SINT -128, 0 bis 31	Slavenummer des CANsync-Slaves mit Steuerwort-Senden-Auftrag
x_EN_CTRL	BOOL	Freigabe für si_SL_NR_CTRL
si_SL_NR_PAR	SINT -128, 0 bis 31	Slavenummer des CANsync-Slaves mit Parameter-Auftrag
x_EN_PAR	BOOL	Freigabe für si_SL_NR_PAR
si_SL_NR_UDL	SINT -128, 0 bis 31	Slavenummer des CANsync-Slaves mit Up/Download-Auftrag
x_EN_UDL	BOOL	Freigabe für si_SL_NR_UDL
si_MAX_SL_NR [a)]	SINT -1, 0 bis 31	maximale Slavenummer für automatische Erhöhung [a)]

[a)] Dieser Eingang entspricht dem Eingang si_MAX_SL_NR am FB CANsync_PD_COMM_MA als maximale Slavenummer für automatische Istwert-Telegramm-Anforderung.

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung

Mit diesem FB wird die Slavenummer des CANsync-Slaves angegeben, für den abgefragt wird ob Steuerwort-Senden-Aufträge (si_SL_NR_CTRL, x_EN_CTRL), Parameter-Aufträge (si_SL_NR_PAR, x_EN_PAR) und Up-/Download-Aufträge (si_SL_NR_UDL, x_EN_UDL) vorhanden sind.

HINWEIS

Der CANsync-Master kann in jedem CANsync-Intervall ein Kommando-Telegramm senden.

Die unterschiedlichen Telegramme im Kommandokanal werden in einer durch Prioritäten vorgegebenen Reihenfolge abgearbeitet:

Telegrammtyp	Priorität
Broadcast-Telegramm 0	höchste
Broadcast-Telegramm 1	↑
Broadcast-Telegramm 2	|
Steuerwort-Telegramme	|
Parameter-Telegramme	↓
Up-/Download-Telegramme	niedrigste

Die Konsequenz dieser Prioritäten ist, daß kein anderes Telegramm verschickt werden kann, wenn ein höherpriores gesendet wird. Sendet man z. B. in jedem CANsync-Intervall das Steuerwort-Telegramm, so kann nie ein Parameter-Telegramm oder Up-/Download-Telegramm verschickt werden!

Ist die Abfrage nach vorhanden Aufträgen zu mehreren CANsync-Slaves notwendig, wird die CANsync-Anschaltung so eingestellt:

In jedem CANsync-Intervall wird die Slavenummer des CANsync-Slaves (für den abgefragt wird ob Aufträge vorhanden sind) automatisch um eins erhöht. Diese Erhöhung wird bis zur si_MAX_SL_NR durchgeführt. Danach wird wieder mit der Abfrage für den CANsync-Slave mit der Slavenummer 0 begonnen usw.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Eingänge si_SL_NR_CTRL, x_EN_CTRL:

Am Eingang si_SL_NR_CTRL wird die Slavenummer des CANsync-Slaves angegeben, für den abgefragt werden soll, ob ein Steuerwort-Senden-Auftrag vorhanden ist (FB CANsync_CONTROLWORD_MA).

Mit si_SL_NR_CTRL = -128 wird eingestellt, daß in jedem CANsync-Intervall die Slavenummer des CANsync-Slaves (für den abgefragt werden soll, ob ein Steuerwort-Senden-Auftrag vorhanden ist) automatisch um eins erhöht wird, bis si_MAX_SL_NR erreicht ist. Danach wird wieder mit der Abfrage für den CANsync-Slave mit der Slavenummer 0 begonnen usw.

Voreinstellung ist si_SL_NR_CTRL = -128, d. h. automatisches Erhöhen bis si_MAX_SL_NR.

Die Einstellung an si_SL_NR_CTRL (auch wenn nicht belegt) wird nur übernommen, wenn x_EN_CTRL = TRUE gesetzt ist. Wenn x_EN_CTRL = FALSE ist, wird dieser Teil der Kommandokanal-Konfigurierung nicht verändert.

Eingänge si_SL_NR_PAR, x_EN_PAR:

Am Eingang si_SL_NR_PAR wird die Slavenummer des CANsync-Slaves angegeben, für den abgefragt werden soll, ob ein Parameter-Auftrag vorhanden ist (FB CANsync_PAR_WRITE_MA oder CANsync_PAR_READ_MA).

Mit si_SL_NR_PAR = -128 wird eingestellt, daß in jedem CANsync-Intervall die Slavenummer des CANsync-Slaves (für den abgefragt werden soll, ob ein Parameter-Auftrag vorhanden ist) automatisch um eins erhöht wird, bis si_MAX_SL_NR erreicht ist. Danach wird wieder mit der Abfrage für den CANsync-Slave mit der Slavenummer 0 begonnen usw.

Voreinstellung ist si_SL_NR_PAR = -128, d. h. automatisches Erhöhen bis si_MAX_SL_NR.

Die Einstellung an si_SL_NR_PAR (auch wenn nicht belegt) wird nur übernommen, wenn x_EN_PAR = TRUE gesetzt ist. Wenn x_EN_PAR = FALSE ist, wird dieser Teil der Kommandokanal-Konfigurierung nicht verändert.

Eingänge si_SL_NR_UDL, x_EN_UDL:

Am Eingang si_SL_NR_UDL wird die Slavenummer des CANsync-Slaves angegeben, für den abgefragt werden soll, ob ein Up-/Download-Auftrag vorhanden ist (FB CANsync_UPDOWNLOAD_MA).

Mit si_SL_NR_UDL = -128 wird eingestellt, daß in jedem CANsync-Intervall die Slavenummer des CANsync-Slaves (für den abgefragt werden soll, ob ein Up-/Download-Auftrag vorhanden ist) automatisch um eins erhöht wird, bis si_MAX_SL_NR erreicht ist. Danach wird wieder mit der Abfrage für den CANsync-Slave mit der Slavenummer 0 begonnen usw.

Voreinstellung ist si_SL_NR_UDL = -128, d. h. automatisches Erhöhen bis si_MAX_SL_NR.

Die Einstellung an si_SL_NR_UDL (auch wenn nicht belegt) wird nur übernommen, wenn x_EN_PAR = TRUE gesetzt ist. Wenn x_EN_UDL = FALSE ist, wird dieser Teil der Kommandokanal-Konfigurierung nicht verändert.

Eingang si_MAX_SL_NR:

Die höchste Slavenummer eines CANsync-Slaves wird am Eingang si_MAX_SL_NR angegeben. Bis zu dieser Slavenummer wird abgefragt ob Steuerwort-Senden-Aufträge, Parameter-Aufträge und Up-/Download-Aufträge vorhanden sind. Dazu werden dann si_SL_NR_CTRL, si_SL_NR_PAR, si_SL_UDL nicht belegt und si_EN_CTRL, si_EN_PAR und si_EN_UDL auf TRUE gesetzt.

Ist si_MAX_SL_NR = -1, bleibt auf der CANsync-Anschaltung der Wert unverändert. Voreinstellung ist si_MAX_SL_NR = -1, d. h. auf der CANsync-Anschaltung bleibt der Wert unverändert.

HINWEIS

Dieser Eingang entspricht dem Eingang si_MAX_SL_NR am FB CANsync_PD_COMM_MA. D. h. mit dem Eingang si_MAX_SL_NR am FB CANsync_COMM_CONTROL_MA **oder** dem Eingang si_MAX_SL_NR am FB CANsync_PD_COMM_MA wird die höchste Slavenummer eines CANsync-Slaves angegeben. Es darf nur einer von beiden Eingängen verwendet werden!

7.3.7 CANsync_CONTROLWORD_MA

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um ein Steuerwortkommando des CANsync-Masters zu senden.

HINWEIS

Für den Einsatz dieses FBs ist es notwendig, für den Kommandokanal (CC) die Steuerwortkommandos freizugeben. Diese Freigabe erfolgt mit dem FB CANsync_COMM_CONTROL_MA.
Der FB CANsync_CONTROLWORD_MA verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
si_SL_NR_CTRL	SINT 0 bis 31	Slavenummer des CANsync-Slaves, an den das Steuerwort gesendet wird
w_CONTROLWORD	WORD	Steuerwort
x_EN	BOOL	Freigabe

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
x_OK	BOOL	OK-Bit

Das Steuerwort (w_CONTROLWORD), die Freigabe zum Senden (x_EN) und die Slavenummer des CANsync-Slaves an den das Steuerwort gesendet wird (si_SL_NR_CTRL), wird der CANsync-Anschaltung übergeben. Nachdem die CANsync-Anschaltung den Auftrag erkannt hat (siehe FB CANsync_COMM_CONTROL_MA) wird das Steuerwort an den CANsync-Slave mit einem Steuerwort-Telegramm gesendet und das Senden am Ausgang x_OK quittiert.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Eingang w_CONTROLWORD:

An dem Eingang w_CONTROLWORD wird das zu sendende Steuerwort angeschlossen.

Eingang x_EN:

Wenn der Eingang x_EN auf TRUE gesetzt wird, wird das Steuerwort zum Senden eingetragen. Die CANsync-Anschaltung sendet das Steuerwort dann im Steuerwort-Telegramm an den CANsync-Slave si_SL_NR_CTRL. Solange x_EN auf TRUE bleibt, wird bei jedem Aufruf des FB das Steuerwort zum Senden eingetragen Dies wird bei jedem Aufruf des FBs neu durchgeführt, solange x_EN auf TRUE bleibt. Damit würde jedoch der Kommandokanal des CANsync dauernd belegt. Deshalb sollte der Eingang x_EN nur für ein CANsync-Intervall auf TRUE gesetzt sein, um das Steuerwort einmal abzuschikken.

Ausgang x_OK:

Der Ausgang x_OK zeigt mit TRUE an, daß das letzte Steuerwort-Telegramm gesendet wurde. Der Ausgang x_OK ist FALSE wenn kein Steuerwort-Telegramm gesendet wurde.

7.3.8 CANsync_CONTROLWORD_SL

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um ein Steuerwort des CANsync-Masters in einer CANsync-Slave-Anschaltung zu empfangen.

HINWEIS

Der FB CANsync_CONTROLWORD_SL verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
w_CONTROLWORD	WORD	Steuerwort
si_RECEIVED	SINT	Anzeige, daß ein Steuerwort empfangen wurde

Empfängt der CANsync-Slave ein Steuerwort-Telegramm vom CANsync-Master gibt der FB CANsync_CONTROLWORD_SL das empfangene Steuerwort am Ausgang w_CONTROLWORD aus.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_SL_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) auf Ωmega Drive-Line II

```
_CANsync_CTRL_SL  AT  %MB3.100000  :  CANsync_SL_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_SL`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_SL_CTRL_BMSTRUCT`	der Datentyp
`%MB3.100000`	die Basisadresse der CANsync-Anschaltung 1 auf Ωmega Drive-Line II

Ausgang w_CONTROLWORD:

Am Ausgang w_CONTROLWORD wird das empfangene Steuerwort ausgegeben.

Ausgang si_RECEIVED:

Am Ausgang si_RECEIVED wird angezeigt, ob ein Steuerwort-Telegramm empfangen wurde. Dann zeigt der Ausgang eine 2 an. Andernfalls wird am Ausgang si_RECEIVED eine 0 angezeigt.

Das Steuerwort-Telegramm ist ein Spezialfall des Broadcast-Telegramms (mit Broadcast-Kommandonummer 1).

7.3.9 CANsync_INIT

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um eine CANsync-Anschaltung (Master oder Slave) zu initialisieren.

HINWEIS

Der FB CANsync_INIT verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_INIT_BMSTRUCT	Initialisierungsdaten für die CANsync-Anschaltung
x_SL	BOOL	Auswahl CANsync-Slave / CANsync-Master
x_SYNC_IN	BOOL	Konfigurierung SYNC-Signal empfangen / senden
x_SYNC_MODE	BOOL	Synchron-Betrieb einrichten
x_ASYNC_MODE0	BOOL	reserviert
x_ASYNC_MODE1	BOOL	reserviert
a_SL_TYP	BYTE_32_BMARRAY	Initialisierungsdaten „Slave-Typen“
b_ACCEPT_CODE	BYTE	Acceptance Code
b_ACCEPT_MASK	BYTE	Acceptance Mask
b_BIT_TIMING0	BYTE	Bus-Timing 0
b_BIT_TIMING1	BYTE	Bus-Timing 1
us_BAUDRATE	USINT 3, 4, 5	Baudrate
us_SYNC_INTERVAL	USINT 8, 4, 2	CANsync-Intervall (Zyklusschema) in ms

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_INIT_BMSTRUCT	Initialisierungsdaten für die CANsync-Anschaltung
w_ERR	WORD	Fehlerwort
x_ERR	BOOL	Fehlerbit
x_OK	BOOL	OK-Bit

Der FB CANsync_INIT bietet mehrere Konfigurierungsmöglichkeiten für die Initialisierung einer CANsync-Anschaltung. Der FB wird verwendet wenn eine CANsync-Master-Anschaltung initialisiert wird oder wenn eine CANsync-Slave-Anschaltung initialisiert wird. Werden sowohl CANsync-Master- als auch CANsync-Slave-Anschaltung (→ CANsync-Cluster) initialisiert, wird der FB zweimal, mit unterschiedlicher Eingangsbelegung eingesetzt (siehe weiter unten).

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_INIT_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) auf Ωmega Drive-Line II

```
_CANsync_INIT_SL  AT  %MB3.100000  :  CANsync_INIT_BMSTRUCT;
```

dabei ist:

`_CANsync_INIT_SL`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_INIT_BMSTRUCT`	der Datentyp
`%MB3.100000`	die Basisadresse der CANsync-Anschaltung 1 auf Ωmega Drive-Line II

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_INIT_MA  AT  %MB3.200000  :  CANsync_INIT_BMSTRUCT;
```

dabei ist:

`_CANsync_INIT_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_INIT_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Eingang x_SL:

Am Eingang x_SL wird ausgewählt, ob die CANsync-Anschaltung als CANsync-Slave-Anschaltung oder als CANsync-Master-Anschaltung initialisiert wird.

Mit x_SL = FALSE wird die CANsync-Anschaltung als CANsync-Master-Anschaltung initialisiert, mit x_SL = TRUE wird die CANsync-Anschaltung als CANsync-Slave-Anschaltung initialisiert.

Eingang x_SYNC_IN:

Werden die beiden CANsync-Anschaltungen auf dem Ωmega Drive-Line II als CANsync-Cluster (d. h. CANsync-Slave und CANsync-Master) initialisiert, muß eingestellt werden, daß das empfangene SYNC-Signal der CANsync-Slave-Anschaltung als SYNC-Signal der CANsync-Master-Anschaltung verwendet wird.

Wird nur die CANsync-Master-Anschaltung betrieben (kein CANsync-Cluster), muß die CANsync-Master-Anschaltung ein eigenes SYNC-Signal generieren und x_SYNC_IN bleibt FALSE.

Mit x_SYNC_IN = FALSE generiert die CANsync-Master-Anschaltung ein eigenes SYNC-Signal, mit x_SYNC_IN = TRUE übernimmt die CANsync-Master-Anschaltung das SYNC-Signal der CANsync-Slave-Anschaltung.

Eingänge x_SYNC_MODE, x_ASYNC_MODE0, x_ASYNC_MODE1:

An diesen drei Eingängen wird die Betriebsart der CANsync-Anschaltung eingestellt. Es darf nur einer der drei Eingänge gleich TRUE sein.

Mit x_SYNC_MODE = TRUE wird der Synchron-Betrieb eingestellt.

Die Eingänge x_ASYNC_MODE0 und x_ASYNC_MODE1 sind reserviert und werden nicht belegt.

Erläuterung der Betriebsart siehe “Allgemeines” auf Seite 121.

Wenn alle drei Eingänge gleich FALSE sind, werden nur die Initialisierungsdaten an die CANsync-Anschaltung übergeben und keine Betriebsart eingestellt. Die Betriebsart kann dann mit dem FB CANsync_MODE_MA (CANsync-Master-Anschaltung) oder CANsync_MODE_SL (CANsync-Slave-Anschaltung) eingestellt werden.

HINWEIS

Die Betriebsart wird mit dem FB CANsync_MODE_MA bzw. CANsync_MODE_SL freigegeben.

Eingang a_SL_TYP:

Dieser Eingang wird nur belegt, wenn die CANsync-Anschaltung als CANsync-Master-Anschaltung initialisiert wird. Hier wird angegeben welche CANsync-Slaves am CANsync-Bus angeschlossen sind. Diese Angabe kann auch mit dem FB CANsync_SL_TYP_INIT erfolgen.

Am Eingang a_SL_TYP wird eine Variable vom Datentyp BYTE_32_BMARRAY angeschlossen. Der Datentyp BYTE_32_BMARRAY ist ein Feld von 32 Einträgen des Datentyps Byte:

```
BYTE_32_BMARRAY   : ARRAY [0..31] OF BYTE;
```

Beispiel:

```
a_Slave_Typen     : BYTE_32_BMARRAY;
```

dabei ist:

`a_Slave_Typen`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`BYTE_32_BMARRAY`	der Datentyp.

In den einzelnen Einträgen des Feldes wird der Slave-Typ der CANsync-Slaves am CANsync-Bus angegeben. Im Eintrag [0] steht der Slave-Typ des CANsync-Slaves mit der Slavenummer 0, im Eintrag [1] steht der Slave-Typ des CANsync-Slaves mit der Slavenummer 1 usw.

Eine 0 im Eintrag [x] bedeutet, daß kein CANsync-Slave mit der Slavenummer x am CANsync-Bus vorhanden ist.

Ein Wert $\neq 0$ im Eintrag [x] bedeutet, daß ein CANsync-Slave mit der Slavenummer x am CANsync-Bus vorhanden ist.

Bedeutung der Slave-Typen

Slave-Typ	Bedeutung
0	kein CANsync-Slave
1	CANsync-Slave-Anschaltung eines Ωmega Drive-Line II, V-Regler mit CANsync-Interface
2 - 255	Reserviert

Eingänge b_ACCEPT_MASK, b_ACCEPT_CODE:

An den Eingängen b_ACCEPT_MASK und b_ACCEPT_CODE kann der Akzeptanzfilter der CANsync-Anschaltung eingestellt werden. Werden die Eingänge nicht belegt, ergeben sich die Voreinstellungen b_ACCEPT_MASK = 16#FF und b_ACCEPT_CODE = 16#FF, d. h. alle Objekte werden berücksichtigt.

Andere Einstellungen werden beim CANsync nicht benötigt.

Diese Eingänge sind aus Kompatibilitätsgründen vorhanden.

Eingänge b_BIT_TIMING0, b_BIT_TIMING1, us_BAUDRATE, us_SYNC_INTERVAL:

Am Eingang us_BAUDRATE wird die Baudrate für den CANsync-Bus eingestellt. Es darf maximal die Baudrate eingestellt werden, die alle Teilnehmer am CANsync-Bus "verstehen".

HINWEIS

Einschränkungen der Baudrate entnehmen Sie bitte der jeweiligen Technischen Beschreibung.

Für drei verschiedene Baudrates ist das Bus-Timing berechnet und wird vom FB CANsync_INIT der CANsync-Anschaltung übergeben.

Baudrate	us_BAUDRATE	b_BIT_TIMING0	b_BIT_TIMING1
125 kBit/s	3	16#03	16#1C
250 kBit/s	4	16#01	16#1C
500 kBit/s	5	16#00	16#1C

Ist der Wert us_BAUDRATE kleiner 3 oder größer 5, wird die Baudrate auf 125 kBit/s eingestellt und im Fehlerwort w_ERR das Bit 1 auf TRUE gesetzt.

An den Eingängen b_BIT_TIMING0 und b_BIT_TIMING1 kann das Bus-Timing der CANsync-Anschaltung individuell eingestellt werden. Die Werte hierfür entnehmen Sie bitte der jeweiligen Technischen Beschreibung.

Die Einstellungen dieser Eingänge wird übernommen, wenn der Eingang us_BAUDRATE = 0 ist.

Die Einstellungen dieser Eingänge wird ignoriert, wenn der Eingang us_BAUDRATE mit einem Wert von 3 bis 5 belegt ist.

Voreinstellung ist us_BAUDRATE = 0, d. h. wenn us_BAUDRATE nicht belegt ist, werden die Einstellungen von b_BIT_TIMING0 und b_BIT_TIMING1 übernommen.

HINWEIS

Die Werte an den Eingängen b_BIT_TIMING0 und b_BIT_TIMING1 werden nur übernommen, wenn der Eingang us_BAUDRATE = 0 oder nicht belegt ist.

Am Eingang us_SYNC_INTERVAL wird die Dauer des CANsync-Intervalls, die CANsync-Zykluszeit, in ms angegeben.

Zusammen mit dem Eingang us_BAUDRATE sind die folgenden Kombinationen zulässig:

CANsync-Baudrate und CANsync-Intervalldauer	us_BAUDRATE	us_SYNC_INTERVAL
500 kBit/s und 2 ms	5	2
250 kBit/s und 4 ms	4	4
125 kBit/s und 8 ms	3	8

Ausgang x_OK:

Der Ausgang x_OK wird auf TRUE gesetzt wenn die CANsync-Anschaltung erfolgreich initialisiert wurde. Der Ausgang x_OK bleibt FALSE wenn die CANsync-Anschaltung nicht initialisiert wurde oder bei der Initialisierung ein Fehler aufgetreten ist.

Ausgänge x_ERR, w_ERR:

Falls ein Fehler auftritt wird das Fehlerbit x_ERR auf TRUE gesetzt und das Fehlerwort w_ERR ausgegeben. Der Ausgang x_OK bleibt dann FALSE.

Fehlerwort w_ERR:

Bit-Nr.	Fehler
0	Timeout beim Handshake mit der CANsync-Anschaltung
1	Eingabefehler bei b_BIT_TIMING0, b_BIT_TIMING1 oder us_BAUDRATE
2 - 10	Reserviert
11	Initialisierung der CANsync-Anschaltung nicht abgeschlossen
12 - 15	Reserviert

7.3.10 CANsync_MODE_MA

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um die Betriebsart einer CANsync-Anschaltung einzustellen.

HINWEIS

Der FB CANsync_MODE_MA verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
x_RESET_SOFTWARE	BOOL	Software-Reset
x_SET_INIT_DATA	BOOL	Initialisierungsdaten übernehmen
x_CANsync_RUN	BOOL	Aktiven Betrieb freigeben
x_RESET_CANsync_CONTROLLER	BOOL	Reset CANsync-Controller (Bus-Off-Reset)
x_SYNC_MODE	BOOL	Synchron-Betrieb einrichten
x_ASYNC_MODE0	BOOL	reserviert
x_ASYNC_MODE1	BOOL	reserviert

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
x_HS_ACTIV	BOOL	Handshake aktiv
x_INIT_POSSIBLE	BOOL	Initialisierung möglich
x_WAIT	BOOL	Warten auf Befehl zur Einstellung der Betriebsart
x_PREPARE_ACTIV	BOOL	Einstellung der Betriebsart aktiv
x_CANsync_ACTIV	BOOL	Betrieb aktiv
x_SYNC_MODE_ACTIV	BOOL	Synchron-Betrieb
x_ASYNC_MODE_ACTIV	BOOL	Asynchron-Betrieb
x_SL	BOOL	Slave-Betrieb
x_MA	BOOL	Master-Betrieb

Der FB CANsync_MODE_MA ermöglicht die Einstellung der Betriebsarten auf der CANsync-Anschaltung. Die Eingänge entsprechen Befehlen. Die Ausgänge zeigen den aktuellen Ist-Zustand an. Die Signale sind jeweils aktiv, wenn sie TRUE sind. Wenn alle Eingänge FALSE sind, wird kein Befehl ausgeführt, es bleibt dann der letzte Zustand aktiv.

HINWEIS

Der FB CANsync_MODE_MA wartet nicht auf die Statusmeldung der CANsync-Anschaltung. D. h. wenn der FB in der Kalt- und Warmstart-Task aufgerufen wird, kann es passieren daß die Ausgänge nicht gesetzt sind. Falls eine Anzeige des Status benötigt wird, muß der FB noch einmal aufgerufen werden. Die Eingänge müssen dann auf FALSE gesetzt sein. An den Ausgängen wird dann der Status der CANsync-Anschaltung angezeigt.

Einsatz in Kalt- und Warmstart-Task:

Das Starten einer Betriebsart ist möglich. Dazu wird eine Betriebsart eingestellt (z. B. für Synchron-Betrieb wird x_SYNC_MODE auf TRUE gesetzt).

Die Eingestellte Betriebsart wird mit x_CANsync_RUN = TRUE gestartet.

Einsatz im zyklischen Programm:

Die CANsync-Anschaltung kann neu initialisiert werden. Dazu setzt man die CANsync-Anschaltung mit x_RESET_SOFTWARE = TRUE zurück (FB CANsync_MODE_MA).

Anschließend wird eine neue Initialisierung der CANsync-Anschaltung durchgeführt (FB CANsync_INIT), eine Betriebsart eingestellt und freigegeben (FB CANsync_MODE_MA).

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Eingang x_RESET_CANsync_CONTROLLER:

Mit x_RESET_CANsync_CONTROLLER = TRUE setzt man den CANsync-Controller zurück. Dadurch verläßt der CANsync-Controller den Bus-Off Zustand und kann wieder am CANsync-Bus aktiv werden.

Eingang x_SET_INIT_DATA:

Mit x_SET_INIT_DATA = TRUE übernimmt die CANsync-Anschaltung neue Initialisierungsdaten.

Dieser Eingang wird nur bei expliziter Programmierung der Initialisierung ohne den FB CANsync_INIT benötigt.

Eingang x_CANsync_RUN:

Mit x_CANsync_RUN = TRUE aktiviert man die unter x_SYNC_MODE, x_ASYNC_MODE0 oder x_ASYNC_MODE1 eingestellte Betriebsart.

Eingänge x_SYNC_MODE, x_ASYNC_MODE0, x_ASYNC_MODE1:

An diesen drei Eingängen wird die Betriebsart der CANsync-Anschaltung eingestellt. Es darf nur einer der drei Eingänge gleich TRUE sein.

Mit x_SYNC_MODE = TRUE wird der Synchron-Betrieb eingestellt.

Die Eingänge x_ASYNC_MODE0 und x_ASYNC_MODE1 sind reserviert und bleiben gleich FALSE.

Nachdem die Betriebsart eingestellt ist, wird mit x_CANsync_RUN = TRUE der aktive Betrieb freigeben.

An den Ausgängen bekommt man mit TRUE eine entsprechende Rückmeldung für die Befehle. Anderenfalls sind die Ausgänge FALSE.

Ausgang x_HS_ACTIV:
Der Ausgang x_HS_ACTIV zeigt mit TRUE an, daß der Handshakebetrieb aktiv ist. Dies wird vom FB CANsync_INIT verwendet.

Ausgang x_INIT_POSSIBLE:
Der Ausgang x_INIT_POSSIBLE zeigt mit TRUE an, daß die CANsync-Anschaltung im Initialisierungszustand ist. Sie kann dann neue Initialisierungsdaten oder den Befehl zur Einstellung der Betriebsart empfangen.

Ausgang x_WAIT:
Der Ausgang x_WAIT zeigt mit TRUE an, daß die CANsync-Anschaltung die Initialisierungsdaten übernommen hat und auf den Befehl zur Einstellung der Betriebsart wartet.

Ausgang x_PREPARE_ACTIV:
Der Ausgang x_PREPARE_ACTIV zeigt mit TRUE an, daß eine Betriebsart eingerichtet wird.

Ausgang x_CANsync_ACTIV:
Der Ausgang x_CANsync_ACTIV zeigt mit TRUE an, daß eine Betriebsart aktiv ist.

Ausgang x_SYNC_MODE_ACTIV:
Der Ausgang x_SYNC_MODE_ACTIV zeigt mit TRUE an, daß Synchronbetrieb eingestellt wurde.

Ausgang x_ASYNC_MODE_ACTIV:
Der Ausgang x_ASYNC_MODE_ACTIV zeigt mit TRUE an, daß Asynchronbetrieb (Mode 0 oder Mode 1) eingestellt wurde.

Ausgang x_SL:
Der Ausgang x_SL zeigt mit TRUE an, daß die CANsync-Anschaltung als Slave konfiguriert wurde.

Ausgang x_MA:
Der Ausgang x_MA zeigt mit TRUE an, daß die CANsync-Anschaltung als Master konfiguriert wurde.

Es können auch Kombinationen von Ausgängen gesetzt sein. Wenn z. B. x_CANsync_ACTIV, x_SYNC_MODE_ACTIV und x_MA auf TRUE gesetzt ist, heißt das, daß die CANsync-Anschaltung als CANsync-Master im aktiven Synchron-Betrieb ist.

7.3.11 CANsync_MODE_SL

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um die Betriebsart einer CANsync-Anschaltung einzustellen.

HINWEIS

Der FB CANsync_MODE_SL verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
x_RESET_SOFTWARE	BOOL	Software-Reset
x_SET_INIT_DATA	BOOL	Initialisierungsdaten übernehmen
x_CANsync_RUN	BOOL	Aktiven Betrieb freigeben
x_RESET_CANsync_CONTROLLER	BOOL	Reset CANsync-Controller (Bus-Off-Reset)
x_SYNC_MODE	BOOL	Synchron-Betrieb einrichten
x_ASYNC_MODE0	BOOL	reserviert
x_ASYNC_MODE1	BOOL	reserviert

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
x_HS_ACTIV	BOOL	Handshake aktiv
x_INIT_POSSIBLE	BOOL	Initialisierung möglich
x_WAIT	BOOL	Warten auf Befehl zur Einstellung der Betriebsart
x_PREPARE_ACTIV	BOOL	Einstellung der Betriebsart aktiv
x_CANsync_ACTIV	BOOL	Betrieb aktiv
x_SYNC_MODE_ACTIV	BOOL	Synchron-Betrieb
x_ASYNC_MODE_ACTIV	BOOL	Asynchron-Betrieb
x_SL	BOOL	Slave-Betrieb
x_MA	BOOL	Master-Betrieb

Der FB CANsync_MODE_SL ermöglicht die Einstellung der Betriebsarten auf der CANsync-Anschaltung. Die Eingänge entsprechen Befehlen. Die Ausgänge zeigen den aktuellen Ist-Zustand an. Die Signale sind jeweils aktiv, wenn sie TRUE sind. Wenn alle Eingänge FALSE sind, wird kein Befehl ausgeführt, es bleibt dann der letzte Zustand aktiv.

HINWEIS

Der FB CANsync_MODE_SL wartet nicht auf die Statusmeldung der CANsync-Anschaltung. D. h. wenn der FB in der Kalt- und Warmstart-Task aufgerufen wird, kann es passieren daß die Ausgänge nicht gesetzt sind. Falls eine Anzeige des Status benötigt wird, muß der FB noch einmal aufgerufen werden. Die Eingänge müssen dann auf FALSE gesetzt sein. An den Ausgängen wird dann der Status der CANsync-Anschaltung angezeigt.

Einsatz in Kalt- und Warmstart-Task:

Das Starten einer Betriebsart ist möglich. Dazu wird eine Betriebsart eingestellt (z. B. für Synchron-Betrieb wird x_SYNC_MODE auf TRUE gesetzt).

Die Eingestellte Betriebsart wird mit x_CANsync_RUN = TRUE gestartet.

Einsatz im zyklischen Programm:

Die CANsync-Anschaltung kann neu initialisiert werden. Dazu setzt man die CANsync-Anschaltung mit x_RESET_SOFTWARE = TRUE zurück (FB CANsync_MODE_SL).

Anschließend wird eine neue Initialisierung der CANsync-Anschaltung durchgeführt (FB CANsync_INIT) und eine Betriebsart eingestellt und freigegeben (FB CANsync_MODE_SL).

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_SL_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) auf Ωmega Drive-Line II

```
_CANsync_CTRL_SL  AT  %MB3.100000  :  CANsync_SL_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_SL`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_SL_CTRL_BMSTRUCT`	der Datentyp
`%MB3.100000`	die Basisadresse der CANsync-Anschaltung 1 auf Ωmega Drive-Line II

Eingang x_RESET_CANsync_CONTROLLER:

Mit x_RESET_CANsync_CONTROLLER = TRUE setzt man den CANsync-Controller zurück. Dadurch verläßt der CANsync-Controller den Bus-Off Zustand und kann wieder am CANsync-Bus aktiv werden.

Eingang x_SET_INIT_DATA:

Mit x_SET_INIT_DATA = TRUE übernimmt die CANsync-Anschaltung neue Initialisierungsdaten.

Dieser Eingang wird nur bei expliziter Programmierung der Initialisierung ohne den FB CANsync_INIT benötigt.

Eingang x_CANsync_RUN:

Mit x_CANsync_RUN = TRUE aktiviert man die unter x_SYNC_MODE, x_ASYNC_MODE0 oder x_ASYNC_MODE1 eingestellte Betriebsart.

Eingänge x_SYNC_MODE, x_ASYNC_MODE0, x_ASYNC_MODE1:

An diesen drei Eingängen wird die Betriebsart der CANsync-Anschaltung eingestellt. Es darf nur einer der drei Eingänge gleich TRUE sein.

Mit x_SYNC_MODE = TRUE wird der Synchron-Betrieb eingestellt.

Die Eingänge x_ASYNC_MODE0 und x_ASYNC_MODE1 sind reserviert und bleiben gleich FALSE.

Nachdem die Betriebsart eingestellt ist, wird mit x_CANsync_RUN = TRUE der aktive Betrieb freigeben.

An den Ausgängen bekommt man mit TRUE eine entsprechende Rückmeldung für die Befehle. Anderenfalls sind die Ausgänge FALSE.

Ausgang x_HS_ACTIV:
Der Ausgang x_HS_ACTIV zeigt mit TRUE an, daß der Handshakebetrieb aktiv ist. Dies wird vom FB CANsync_INIT verwendet.

Ausgang x_INIT_POSSIBLE:
Der Ausgang x_INIT_POSSIBLE zeigt mit TRUE an, daß die CANsync-Anschaltung im Initialisierungszustand ist. Sie kann dann neue Initialisierungsdaten oder den Befehl zur Einstellung der Betriebsart empfangen.

Ausgang x_WAIT:
Der Ausgang x_WAIT zeigt mit TRUE an, daß die CANsync-Anschaltung die Initialisierungsdaten übernommen hat und auf den Befehl zur Einstellung der Betriebsart wartet.

Ausgang x_PREPARE_ACTIV:
Der Ausgang x_PREPARE_ACTIV zeigt mit TRUE an, daß eine Betriebsart eingerichtet wird.

Ausgang x_CANsync_ACTIV:
Der Ausgang x_CANsync_ACTIV zeigt mit TRUE an, daß eine Betriebsart aktiv ist.

Ausgang x_SYNC_MODE_ACTIV:
Der Ausgang x_SYNC_MODE_ACTIV zeigt mit TRUE an, daß Synchronbetrieb eingestellt wurde.

Ausgang x_ASYNC_MODE_ACTIV:
Der Ausgang x_ASYNC_MODE_ACTIV zeigt mit TRUE an, daß Asynchronbetrieb (Mode 0 oder Mode 1) eingestellt wurde.

Ausgang x_SL:
Der Ausgang x_SL zeigt mit TRUE an, daß die CANsync-Anschaltung als Slave konfiguriert wurde.

Ausgang x_MA:
Der Ausgang x_MA zeigt mit TRUE an, daß die CANsync-Anschaltung als Master konfiguriert wurde.

Es können auch Kombinationen von Ausgängen gesetzt sein. Wenn z. B. x_CANsync_ACTIV, x_SYNC_MODE_ACTIV und x_SL auf TRUE gesetzt ist, heißt das, daß die CANsync-Anschaltung als CANsync-Slave im aktiven Synchron-Betrieb ist.

7.3.12 CANsync_PAR_READ_MA

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um vom CANsync-Master über den CANsync einen Parameterwert vom CANsync-Slave anzufordern.

HINWEIS

Für den Einsatz dieses FBs ist es notwendig, für den Kommandokanal (CC) die Steuerwortkommandos freizugeben. Diese Freigabe erfolgt mit dem FB CANsync_COMM_CONTROL_MA.

Dieser FB kann mehrfach instanziiert werden, wenn jeweils verschiedene CANsync-Slaves angesprochen werden.

Der FB CANsync_PAR_READ_MA verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
si_SL_NR	SINT 0 bis 31	Slavenummer des CANsync-Slaves, an den der Parameter-Lesen-Auftrag gerichtet ist
u_PAR_NR	UINT	Parameter-Nummer
x_PAR_FORMAT	BOOL	Parameter-Format
i_SUB_SL	INT 0 bis 31	Sub-Slave-Adresse (reserviert)
t_TIME	TIME	Überwachungszeit
x_EN	BOOL	Freigabe
x_RESET	BOOL	Reset

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
ud_PAR_VALUE	UDINT	gelesener Parameter-Wert
x_PAR_FORMAT_READ	BOOL	gelesenes Parameter-Format
x_BUSY	BOOL	Kommunikation ist aktiv
b_ERR	BYTE	Fehlerbyte
i_ERR	INT	Fehlernummer
x_ERR	BOOL	Fehlerbit
x_OK	BOOL	OK-Bit

Der FB CANsync_PAR_READ_MA übergibt mit den Werten der Eingänge u_PAR_NR, x_PAR_FORMAT und i_SUB_SL einen Parameter-Lesen-Auftrag an den CANsync-Slave mit der Slavenummer si_SL_NR. Der CANsync-Slave bearbeitet den Parameter-Lesen-Auftrag und gibt das Ergebnis der Kommunikation zurück. Der gelesene Parameter-Wert wird am Ausgang ud_PAR_VALUE ausgegeben.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Eingang si_SL_NR:

Am Eingang si_SL_NR wird die Slavenummer des CANsync-Slaves am CANsync-Bus angegeben, von dem der Parameter-Wert gelesen wird.

Eingang u_PAR_NR:

Die Parameter-Nummer für den Parameter-Lesen-Auftrag wird am Eingang u_PAR_NR angegeben.

Eingang x_PAR_FORMAT:

Das Format des angeforderten Parameter-Werts wird am Eingang x_PAR_FORMAT eingestellt. x_PAR_FORMAT = FALSE bedeutet Format Wort, x_PAR_FORMAT = TRUE bedeutet Format Doppelwort.

Eingang i_SUB_SL:

Dieser Eingang ist reserviert und wird nicht belegt.

Eingang t_TIME:

Am Eingang t_TIME wird die Überwachungszeit angegeben, innerhalb der der Parameter-Lesen-Auftrag ausgeführt sein soll. Wird der Parameter-Lesen-Auftrag nicht innerhalb der Überwachungszeit beendet, wird im Fehlerbyte b_ERR das Bit 1 auf TRUE gesetzt.

Voreinstellung für t_TIME ist 3 s.

Eingang x_EN:

Die Kommunikation wird mit x_EN = TRUE gestartet. Der Eingang x_EN darf erst wieder auf FALSE gesetzt werden, wenn der Ausgang x_BUSY nach Abschluß der Kommunikation auf FALSE fällt. Anderenfalls wird von einem bewußten Abbruch der Kommunikation ausgegangen und der FB muß zurückgesetzt werden (x_RESET = TRUE).

Eingang x_RESET:

Der FB CANsync_PAR_READ_MA wird mit x_RESET = TRUE zurückgesetzt. Dies ist z. B. nach einem Abbruch der Kommunikation (durch x_EN = FALSE) oder nach einer Fehlermeldung notwendig. Anschließend muß x_RESET wieder auf FALSE gesetzt werden.

Ausgang ud_PAR_VALUE:

Der gelesene Parameter-Wert wird am Ausgang ud_PAR_VALUE zur Verfügung gestellt.

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 der Parameter-Lesen-Auftrag korrekt ausgeführt wurde. Der Ausgang x_OK ist FALSE wenn kein Parameter-Lesen-Auftrag ausgeführt wurde oder der Auftrag nicht korrekt ausgeführt wurde.

Ausgänge x_ERR, b_ERR, i_ERR:

Falls ein Fehler auftritt, wird das Fehlerbit x_ERR auf TRUE gesetzt und das Fehlerbyte b_ERR ausgegeben. Der Ausgang x_OK bleibt dann FALSE.

Wenn vom CANsync-Slave eine Fehlernummer gemeldet wird, wird am Ausgang i_ERR eine Fehlernummer ausgegeben. Der Inhalt der Fehlernummer wird durch die Applikation im CANsync-Slave bestimmt.

Fehlerbyte b_ERR:

Bit-Nr.	Fehler
0	Kommunikationsfehler, Fehlernummer steht in i_ERR
1	Timeout
2, 3	Reserviert
4	Ungültige Slavenummer des CANsync-Slaves am CANsync-Bus
5 - 7	Reserviert

7.3.13 CANsync_PAR_SL

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um einen Parameter-Lesen-Auftrag oder einen Parameter-Schreiben-Auftrag zu erkennen. Der FB ist für den Einsatz mit den BAPS-Bedarfsdaten-FBs BAPS_PAR_READ und BAPS_PAR_WRITE geeignet.

HINWEIS

Der FB CANsync_PAR_SL verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
ud_PAR_VALUE_READ	UDINT	Parameter-Wert (Lesen)
x_PAR_FORMAT_READ	BOOL	Parameter-Format (Lesen)
i_ERR	INT	Fehlernummer (Applikation)
x_ERR_IN	BOOL	Fehlerbit (Applikation)
x_OK_IN	BOOL	OK-Bit (Applikation)
x_EN	BOOL	Freigabe
x_RESET	BOOL	Reset

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
x_RESET_JOB	BOOL	Reset (Auftragswechsel)
x_READ	BOOL	Parameter-Lesen-Auftrag
x_WRITE	BOOL	Parameter-Schreiben-Auftrag
u_PAR_NR	UINT	Parameter-Nummer
x_PAR_FORMAT_WRITE	BOOL	Parameter-Format (Schreiben)
ud_PAR_VALUE_WRITE	UDINT	Parameter-Wert (Schreiben)
x_ACTIV	BOOL	Auftrag gemeldet, warten auf Ergebnis der Applikation
x_BUSY	BOOL	Kommunikation ist aktiv
x_OK	BOOL	OK-Bit

Der FB CANsync_PAR_SL erkennt einen Parameter-Lesen-Auftrag oder einen Parameter-Schreiben-Auftrag und stellt die jeweiligen Daten an den Ausgängen zur Verfügung. Die Applikation bearbeitet die Aufträge und übergibt die Ergebnisse dem FB über die Eingänge. Der FB meldet dann die Ergebnisse an die CANsync-Slave-Anschaltung und zeigt an daß die Ergebnisse an den CANsync-Master gesendet wurden.

Bei einem Parameter-Lesen-Auftrag wird mit x_READ = TRUE der Auftrag signalisiert und die Parameter-Nummer an u_PAR_NR ausgegeben. Von der Applikation wird der Parameter-Wert an ud_PAR_VALUE_READ und das Parameter-Format an x_PAR_FORMAT_READ erwartet. Mit x_OK_IN = TRUE werden Parameter-Wert und Parameter-Format übernommen und an den CANsync-Master gesendet. Wird von der Applikation ein Fehler gemeldet, kann eine Fehlernummer an i_ERR angeschlossen und x_ERR_IN = TRUE gesetzt werden. In diesem Fall wird die Fehlernummer an den CANsync-Master gesendet.

Bei einem Parameter-Schreiben-Auftrag wird mit x_WRITE = TRUE der Auftrag signalisiert, die Parameter-Nummer an u_PAR_NR, das Parameter-Format an x_PAR_FORMAT_WRITE und der Parameter-Wert an ud_PAR_VALUE_WRITE ausgegeben. Von der Applikation wird das Ergebnis der Kommunikation erwartet. Mit x_OK_IN = TRUE wird dem CANsync-Master gemeldet, daß der Parameter-Schreiben-Auftrag erfolgreich ausgeführt wurde. Wird von der Applikation ein Fehler gemeldet, kann eine Fehlernummer an i_ERR angeschlossen und x_ERR_IN = TRUE gesetzt werden. In diesem Fall wird die Fehlernummer an den CANsync-Master gesendet.

HINWEIS

Im Ωmega Drive-Line II mit CANsync-Slave-Anschaltung können Parameter-Lesen-Aufträge und Parameter-Schreiben-Aufträge an den V-Regler weitergeleitet werden. Hierfür werden die FBs der BAPS-Bedarfsdatenkommunikation verwendet.
Im folgenden Abschnitt ist bei einigen Ein- und Ausgängen in Klammern der jeweilige Ein- oder Ausgang der FBs der BAPS-Bedarfsdatenkommunikation angegeben.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_SL_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) auf Ωmega Drive-Line II

```
_CANsync_CTRL_SL  AT  %MB3.100000  :  CANsync_SL_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_SL`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_SL_CTRL_BMSTRUCT`	der Datentyp
`%MB3.100000`	die Basisadresse der CANsync-Anschaltung 1 auf Ωmega Drive-Line II

Eingang ud_PAR_VALUE_READ:

An ud_PAR_VALUE_READ wird bei einem Parameter-Lesen-Auftrag der Parameter-Wert erwartet. Diesen Parameter-Wert stellt das Applikationsprogramm zur Verfügung. (Der Eingang ud_PAR_VALUE_READ kann mit FB BAPS_PAR_READ, Ausgang ud_PAR_VALUE verbunden sein.)

Eingang x_PAR_FORMAT_READ:

An x_PAR_FORMAT_READ wird bei einem Parameter-Lesen-Auftrag das Parameter-Format erwartet. Mit x_PAR_FORMAT_READ = FALSE wird angegeben, daß der Parameter-Wert an ud_PAR_VALUE_READ vom Format Wort (16 Bit) ist, mit x_PAR_FORMAT_READ = TRUE wird angegeben, daß der Parameter-Wert an ud_PAR_VALUE_READ vom Format Doppelwort (32 Bit) ist.

Das Parameter-Format stellt das Applikationsprogramm zur Verfügung. (Der Eingang x_PAR_FORMAT_READ kann mit FB BAPS_PAR_READ, Ausgang x_PAR_FORMAT verbunden sein.)

Eingänge x_ERR_IN, i_ERR:

Falls das Applikationsprogramm den Parameter-Auftrag des CANsync-Masters nicht bearbeiten oder erfüllen kann, kann an i_ERR eine Fehlernummer angegeben und x_ERR_IN = TRUE gesetzt werden. In diesem Fall wird die Fehlernummer an den CANsync-Master gesendet.

Die Fehlernummer i_ERR und das Fehlerbit x_ERR_IN stellt das Applikationsprogramm zur Verfügung. (Der Eingang x_ERR_IN kann mit FB BAPS_PAR_WRITE, Ausgang x_ERR und/oder FB BAPS_PAR_READ, Ausgang x_ERR verbunden sein. Der Eingang i_ERR kann mit FB BAPS_PAR_WRITE, Ausgang i_ERR_COMM und/oder FB BAPS_PAR_READ, Ausgang i_ERR_COMM verbunden sein. Siehe Abb. 7-6: Nr. 1)

Eingang x_OK_IN:

Hat das Applikationsprogramm den Parameter-Auftrag des CANsync-Master erfüllt, wird der Eingang x_OK_IN = TRUE gesetzt.

Bei einem Parameter-Lesen-Auftrag wird an ud_PAR_VALUE_READ der gelesene Parameter-Wert und an x_PAR_FORMAT_READ das Format des gelesenen Parameters erwartet. Bei einem Parameter-Schreiben-Auftrag wird kein weiterer Wert erwartet.

Das OK-Bit muß vom Applikationsprogramm zur Verfügung gestellt werden. (Der Eingang x_OK kann mit FB BAPS_PAR_WRITE, Ausgang x_OK und/oder FB BAPS_PAR_READ, Ausgang x_OK verbunden sein. Siehe Abb. 7-6: Nr. 2)

Eingang x_EN:

Der FB CANsync_PAR_SL wird mit x_EN = TRUE aktiviert. Nur wenn der FB aktiviert ist, werden Parameter-Aufträge gemeldet und Antworten an den CANsync-Master gesendet.

Falls der FB CANsync_PAR_SL deaktiviert wird (x_EN = FALSE), muß abgewartet werden bis der letzte Parameter-Auftrag bearbeitet und an den CANsync-Master gesendet wurde (x_BUSY = FALSE). Anderenfalls wird von einem bewußten Abbruch der Kommunikation ausgegangen und der FB muß dann mit x_RESET = TRUE zurückgesetzt werden.

Eingang x_RESET:

Mit x_RESET = TRUE kann der FB zurückgesetzt werden. Dies ist z. B. nach einem Abbruch der Kommunikation (durch x_EN = FALSE) oder nach einer Fehlermeldung notwendig. Anschließend muß x_RESET wieder auf FALSE gesetzt werden.

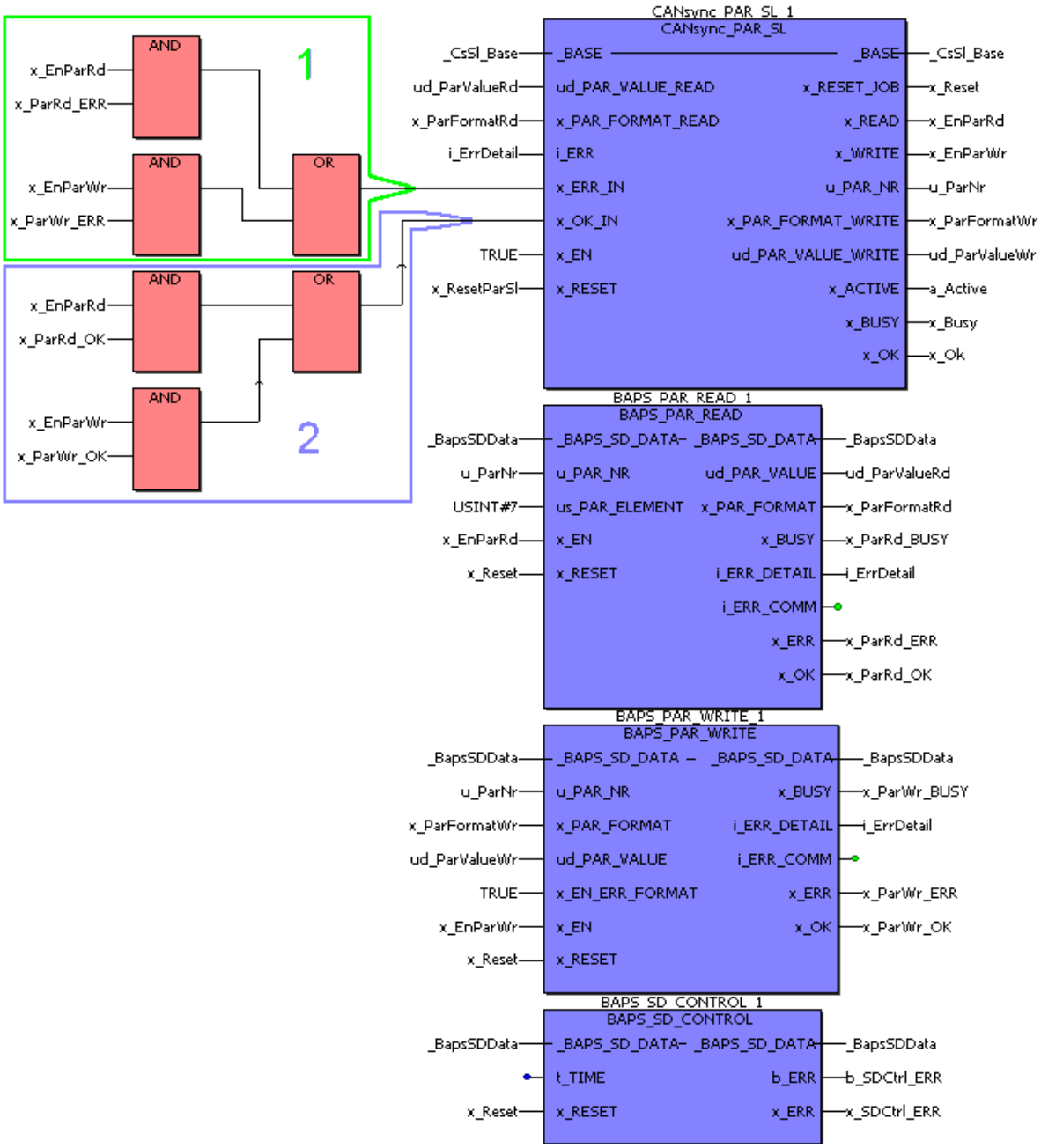


Abbildung 7-6: Beispiel für die Verwendung des FBs CANsync_PAR_SL in Verbindung mit den FBs BAPS_PAR_READ, BAPS_PAR_WRITE und BAPS_SD_CONTROL

Ausgang x_RESET_JOB:

Der CANsync-Master kann einen Parameter-Auftrag abbrechen. In diesem Fall wird der Ausgang x_RESET_JOB auf TRUE gesetzt. Der Ausgang x_RESET_JOB wird wieder auf FALSE gesetzt, wenn ein neuer Parameter-Auftrag vom CANsync-Master gestartet wird. (Der Ausgang x_RESET_JOB kann mit FB BAPS_PAR_WRITE, Eingang x_RESET, FB BAPS_PAR_READ, Eingang x_RESET und/oder FB BAPS_SD_CONTROL, Eingang x_RESET verbunden sein.)

Ausgang x_READ:

Der Ausgang x_READ wird auf TRUE gesetzt, wenn ein Parameter-Lesen-Auftrag ansteht. (Der Ausgang x_READ kann mit FB BAPS_PAR_READ, Eingang x_EN verbunden sein).

Wenn kein Parameter-Lesen-Auftrag ansteht wird der Ausgang x_READ auf FALSE gesetzt.

Ausgang x_WRITE:

Der Ausgang x_WRITE wird auf TRUE gesetzt, wenn ein Parameter-Schreiben-Auftrag ansteht. (Der Ausgang x_WRITE kann mit FB BAPS_PAR_WRITE, Eingang x_EN verbunden sein). Wenn kein Parameter-Schreiben-Auftrag ansteht wird der Ausgang x_WRITE auf FALSE gesetzt.

Ausgang u_PAR_NR:

Am Ausgang u_PAR_NR wird die Parameter-Nummer des Parameter-Auftrags ausgegeben. (Der Ausgang u_PAR_NR kann mit FB BAPS_PAR_WRITE, Eingang u_PAR_NR oder FB BAPS_PAR_READ, Eingang u_PAR_NR verbunden sein.)

Ausgang x_PAR_FORMAT:

Am Ausgang x_PAR_FORMAT wird bei einem Parameter-Schreiben-Auftrag das Parameter-Format des Parameters u_PAR_NR ausgegeben.

x_PAR_FORMAT = FALSE bedeutet Format Wort (16 Bit), x_PAR_FORMAT = TRUE bedeutet Format Doppelwort (32 Bit). (Der Ausgang x_PAR_FORMAT kann mit FB BAPS_PAR_WRITE, Eingang x_PAR_FORMAT verbunden sein.)

Ausgang ud_PAR_VALUE:

Am Ausgang ud_PAR_VALUE wird bei einem Parameter-Schreiben-Auftrag der Parameter-Wert des Parameters u_PAR_NR ausgegeben. (Der Ausgang ud_PAR_VALUE kann mit FB BAPS_PAR_WRITE, Eingang ud_PAR_VALUE verbunden sein.)

Ausgang x_ACTIV:

Der Ausgang x_ACTIV zeigt mit TRUE an, daß der FB CANsync_PAR_SL während eines Parameter-Auftrags auf das Ergebnis des Parameter-Auftrags wartet (Eingang x_OK_IN oder x_ERR_IN). Anderenfalls ist der Ausgang x_ACTIV auf FALSE gesetzt.

Ausgang x_BUSY:

Der Ausgang x_BUSY zeigt mit TRUE an, daß der FB CANsync_PAR_SL auf das Ergebnis des Parameter-Schreiben oder -Lesen wartet und daß die Antwort an den CANsync-Master fertig ist, jedoch der CANsync-Master die Antwort noch nicht angefordert hat. Anderenfalls ist der Ausgang x_BUSY auf FALSE gesetzt.

Ausgang x_OK:

Der Ausgang x_OK wird auf TRUE gesetzt, wenn der CANsync-Master die Antwort abgeholt hat. Es ist dabei unerheblich ob es sich um die Fehler-Antwort oder die Antwort für die ordnungsgemäße Abarbeitung des Auftrags handelt. Der Ausgang x_OK ist FALSE wenn noch kein Parameter-Auftrag ausgeführt wurde, der Parameter-Auftrag nicht ausgeführt wurde oder der CANsync-Master die Antwort nicht abgeholt hat.

7.3.14 CANsync_PAR_WRITE_MA

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um vom CANsync-Master über den CANsync-Bus einen Parameterwert an den CANsync-Slave zu senden.

HINWEIS

Für den Einsatz dieses FBs ist es notwendig, für den Kommandokanal (CC) die Steuerwortkommandos freizugeben. Diese Freigabe erfolgt mit dem FB CANsync_COMM_CONTROL_MA.

Dieser FB kann mehrfach instanziiert werden, wenn jeweils verschiedene CANsync-Slaves angesprochen werden.

Der FB CANsync_PAR_WRITE_MA verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
si_SL_NR	SINT 0 bis 31	Slavenummer des CANsync-Slaves, an den der Parameter-Schreiben-Auftrag gerichtet ist
u_PAR_NR	UINT	Parameter-Nummer
x_PAR_FORMAT	BOOL	Parameter-Format
i_SUB_SL	INT 0 bis 31	Sub-Slave-Adresse (reserviert)
ud_PAR_VALUE	UDINT	zu schreibender Parameter-Wert
t_TIME	TIME	Überwachungszeit
x_EN	BOOL	Freigabe
x_RESET	BOOL	Reset

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
x_BUSY	BOOL	Kommunikation ist aktiv
i_ERR	INT	Fehlernummer
b_ERR	BYTE	Fehlerbyte
x_ERR	BOOL	Fehlerbit
x_OK	BOOL	OK-Bit

Der FB CANsync_PAR_WRITE_MA übergibt mit den Werten der Eingänge u_PAR_NR, x_PAR_FORMAT, i_SUB_SL und ud_PAR_VALUE einen Parameter-Schreiben-Auftrag an den

CANsync-Slave mit der Slavenummer si_SL_NR. Der CANsync-Slave bearbeitet den Parameter-Schreiben-Auftrag und gibt das Ergebnis der Kommunikation zurück.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Eingang si_SL_NR:

Am Eingang si_SL_NR wird die Slavenummer des CANsync-Slaves am CANsync-Bus angegeben, an den der Parameter-Wert gesendet werden soll.

Eingang u_PAR_NR:

Die Parameter-Nummer für den Parameter-Schreiben-Auftrag wird am Eingang u_PAR_NR angegeben.

Eingang x_PAR_FORMAT:

Das Format des zu übertragenden Parameter-Werts wird am Eingang x_PAR_FORMAT eingestellt. x_PAR_FORMAT = FALSE bedeutet Format Wort, x_PAR_FORMAT = TRUE bedeutet Format Doppelwort.

Eingang i_SUB_SL:

Dieser Eingang ist reserviert und wird nicht belegt.

Eingang ud_PAR_VALUE:

Der zu übertragende Parameter-Wert wird am Eingang ud_PAR_VALUE angegeben.

Eingang t_TIME:

Am Eingang t_TIME wird die Überwachungszeit angegeben, innerhalb der der Parameter-Schreiben-Auftrag ausgeführt sein soll. Wird der Parameter-Schreiben-Auftrag nicht innerhalb der Überwachungszeit beendet, wird im Fehlerbyte b_ERR das Bit 1 auf TRUE gesetzt.

Voreinstellung für t_TIME ist 3 s.

Eingang x_EN:

Die Kommunikation wird mit x_EN = TRUE gestartet. Der Eingang x_EN darf erst wieder auf FALSE gesetzt werden, wenn der Ausgang x_BUSY nach Abschluß der Kommunikation auf FALSE fällt. Anderenfalls wird von einem bewußten Abbruch der Kommunikation ausgegangen und der FB muß zurückgesetzt werden (x_RESET = TRUE).

Eingang x_RESET:

Der FB CANsync_PAR_WRITE_MA wird mit x_RESET = TRUE zurückgesetzt. Dies ist z. B. nach einem Abbruch der Kommunikation (durch x_EN = FALSE) oder nach einer Fehlermeldung notwendig. Anschließend muß x_RESET wieder auf FALSE gesetzt werden.

Ausgang x_BUSY:

Der Ausgang x_BUSY zeigt mit TRUE an, daß die Kommunikation aktiv ist. Anderenfalls ist x_BUSY = FALSE.

Ausgang x_OK:

Der Ausgang x_OK wird auf TRUE gesetzt, wenn der Parameter-Schreiben-Auftrag korrekt ausgeführt wurde. Der Ausgang x_OK ist FALSE wenn kein Parameter-Schreiben-Auftrag ausgeführt wurde oder der Auftrag nicht korrekt ausgeführt wurde.

Ausgänge x_ERR, b_ERR, i_ERR:

Falls ein Fehler auftritt, wird das Fehlerbit x_ERR auf TRUE gesetzt und das Fehlerbyte b_ERR ausgegeben. Der Ausgang x_OK bleibt dann FALSE.

Wenn vom CANsync-Slave eine Fehlernummer gemeldet wird, wird am Ausgang i_ERR eine Fehlernummer ausgegeben. Der Inhalt der Fehlernummer wird durch die Applikation im CANsync-Slave bestimmt.

Fehlerbyte b_ERR:

Bit-Nr.	Fehler
0	Kommunikationsfehler, Fehlernummer steht in i_ERR
1	Timeout
2, 3	Reserviert
4	Ungültige Slavenummer des CANsync-Slaves am CANsync-Bus
5 - 7	Reserviert

7.3.15 CANsync_PD_CFG_MA

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um die Belegung der Sollwert-Telegramme der CANsync-Anschaltung für einen CANsync-Master zu konfigurieren.

HINWEIS

Der FB CANsync_PD_CFG_MA verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
a_WRC1	SINT_4_BMARRAY	Sollwertnummern für Sollwert-Telegramm 1
a_HL_WRC1	BOOL_4_BMARRAY	Zuordnung High- oder Lowword für Sollwert-Telegramm 1
a_WRC2	SINT_4_BMARRAY	Sollwertnummern für Sollwert-Telegramm 2
a_HL_WRC2	BOOL_4_BMARRAY	Zuordnung High- oder Lowword für Sollwert-Telegramm 2

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung

Mit dem FB CANsync_PD_CFG_MA erfolgt:

- die Zuordnung von 8 Sollwerten á 32 Bit
  zu den Sollwert-Telegrammen 1 und 2 (Senden)

in einem CANsync-Master.

Die Applikation kann im CANsync-Master 8 Sollwerte á 32 Bit schreiben. Die Sollwerte haben die Sollwertnummern 0 bis 7.

Ein Sollwert setzt sich aus Lowword (Bit 0 bis 15) und Highword (Bit 16 bis 31) zusammen.

In den 2 Sollwert-Telegrammen können (jeweils) 4 * 16 Bit Daten an CANsync-Slaves gesendet werden. Das sind 4 Worte mit den Wortnummern 0 bis 3.

Mit dem FB CANsync_PD_CFG_MA wird festgelegt,
welche Daten in die 4 Worte (Wort 0 bis 3) des Sollwert-Telegramm 1 und
welche Daten in die 4 Worte (Wort 0 bis 3) des Sollwert-Telegramm 2 eingetragen werden.

Die Daten für Sollwert-Telegramm 1 können aus den Sollwerten 0 bis 3 gewählt werden,
die Daten für Sollwert-Telegramm 2 können aus den Sollwerten 4 bis 7 gewählt werden.

Jedem Wort in einem Sollwert-Telegramm kann ein Low- oder Highword eines Sollwerts zugeordnet werden. Bei der Übertragung eines 32 Bit-Sollwerts werden 2 Worte im Sollwert-Telegramm benötigt.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Eingang a_WRC1:

Die Zuordnung

"Sollwert (0 bis 3)" -> "Wort im Sollwert-Telegramm 1" erfolgt in a_WRC1.

a_WRC1[Wortnummer] := Sollwertnummer

Eingang a_HL_WRC1:

Die Zuordnung

"Low- oder Highword des ausgewählten Sollwerts" -> "Wort im Sollwert-Telegramm 1" erfolgt in a_HL_WRC1.

a_HL_WRC1[Wortnummer] := FALSE (wenn Lowword)

a_HL_WRC1[Wortnummer] := TRUE (wenn Highword)

Beispiel:

Wortnummer im Sollwert-Telegramm 1	ausgewählter Sollwert	Anschluß an Eingang a_WRC1	Anschluß an Eingang a_HL_WRC1
0	Sollwert 1 Lowword	a_WRC1[0] = SINT# 1	a_HL_WRC1[0] = FALSE
1	Sollwert 1 Highword	a_WRC1[1] = SINT# 1	a_HL_WRC1[1] = TRUE
2	Sollwert 0 Word	a_WRC1[2] = SINT# 0	a_HL_WRC1[2] = FALSE oder offen
3	Sollwert 2 Word	a_WRC1[3] = SINT# 2	a_HL_WRC1[3] = FALSE oder offen

Eingang a_WRC2:

Die Zuordnung

"Sollwert (4 bis 7)" -> "Wort im Sollwert-Telegramm 2" erfolgt in a_WRC2.

a_WRC2[Wortnummer] := Sollwertnummer

Eingang a_HL_WRC2:

Die Zuordnung

"Low- oder Highword des ausgewählten Sollwerts" -> "Wort im Sollwert-Telegramm 2" erfolgt in a_HL_WRC2.

a_HL_WRC2[Wortnummer] := FALSE (wenn Lowword)

a_HL_WRC2[Wortnummer] := TRUE (wenn Highword)

Wenn einem Wort im Sollwert-Telegramm 1 oder 2 kein Sollwert zugeordnet werden soll, gibt man am entsprechenden Eintrag in a_WRC1 bzw. a_WRC2 als Sollwertnummer eine -1 an.

Die entsprechende Einstellung in a_HL_WRC1 bzw. a_HL_WRC2 ist in diesem Fall ohne Bedeutung.

a_WRC1[Wortnummer] := SINT#-1

a_WRC2[Wortnummer] := SINT#-1

Beispiel:
Dem Wort 1 des Sollwert-Telegramm 2 soll kein Sollwert zugeordnet werden.

Wortnummer im Soll-wert-Telegramm 2	ausgewählter Sollwert	Anschluß an Eingang a_WRC2	Anschluß an Eingang a_HL_WRC2
1	kein	a_WRC2[1] = SINT# -1	a_HL_WRC2[1] ohne Bedeutung

Im CANsync-Slave findet eine Zuordnung der Worte der Sollwert-Telegramme zu Sollwerten statt. Diese Zuordnung erfolgt standardmäßig analog zum CANsync-Master. Eine Überprüfung zwischen der Zuordnung im CANsync-Master und im CANsync-Slave findet nicht statt, weil es auch sinnvolle Anwendungen für abweichende Zuordnungen gibt.

7.3.16 CANsync_PD_CFG_READ_MA

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um die Belegung der Istwert-Telegramme der CANsync-Slaves in der CANsync-Anschaltung für einen CANsync-Master zu konfigurieren.

HINWEIS

Der FB CANsync_PD_CFG_READ_MA verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
si_SL_NR	SINT 0 bis 31	Slavenummer des CANsync-Slaves, von dem Istwerte empfangen werden
a_RDC1	SINT_4_BMARRAY	Istwertnummern für Istwert-Telegramm 1
a_HL_RDC1	BOOL_4_BMARRAY	Zuordnung High- oder Lowword für Istwert-Telegramm 1
a_RDC2	SINT_4_BMARRAY	Istwertnummern für Istwert-Telegramm 2
a_HL_RDC2	BOOL_4_BMARRAY	Zuordnung High- oder Lowword für Istwert-Telegramm 2

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung

Mit dem FB CANsync_PD_CFG_READ_MA erfolgt:

- die Zuordnung der Istwert-Telegramme 1 und 2 eines CANsync-Slaves (Empfang) zu 8 Istwerten á 32 Bit

in einem CANsync-Master.

Die Applikation kann im CANsync-Master 8 Istwerte á 32 Bit jedes CANsync-Slaves lesen.

Die CANsync-Slaves haben die Slavenummern 0 bis 31.

Die Istwerte haben die Istwertnummern 0 bis 7.

Ein Istwert setzt sich aus Lowword (Bit 0 bis 15) und Highword (Bit 16 bis 31) zusammen.

Im CANsync-Master können je CANsync-Slave 2 Istwert-Telegramme mit (jeweils) 4 * 16 Bit Daten empfangen werden. Das sind (jeweils) 4 Worte mit den Wortnummern 0 bis 3.

Mit dem FB CANsync_PD_CFG_READ_MA wird für einen CANsync-Slave festgelegt, welchen Istwerten die Daten aus den 4 Worten (Wort 0 bis 3) des Istwert-Telegramm 1 und welchen Istwerten die Daten aus den 4 Worten (Wort 0 bis 3) des Istwert-Telegramm 2 des CANsync-Slaves zugeordnet werden.

Die Daten aus dem Istwert-Telegramm 1 können den Istwerten 0 bis 3 zugeordnet werden, die Daten aus dem Istwert-Telegramm 2 können den Istwerten 4 bis 7 zugeordnet werden.

Jedes Wort in einem Istwert-Telegramm kann nur einem Low- oder Highword eines Istwerts zugeordnet werden.

Bei der Übertragung eines 32 Bit-Istwerts werden 2 Worte im Istwert-Telegramm benötigt (ein Wort aus dem Istwert-Telegramm wird dem Lowword eines Istwerts zugeordnet, ein anderes Wort dieses Istwert-Telegramms wird dem Highword dieses Istwerts zugeordnet).

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Eingang si_SL_NR:

Am Eingang si_SL_NR wird die Slavenummer des CANsync-Slaves am CANsync-Bus angegeben, dessen Istwert-Telegramme konfiguriert werden.

Eingang a_RDC1:

Die Zuordnung

"Wort im Istwert-Telegramm 1" -> "Istwert (0 bis 3)" erfolgt in a_RDC1.

a_RDC1[Wortnummer] := Istwertnummer

Eingang a_HL_RDC1:

Die Zuordnung

"Wort im Istwert-Telegramm 1" -> "Low- oder Highword des ausgewählten Istwerts" erfolgt in a_HL_RDC1.

a_HL_RDC1[Wortnummer] := FALSE (wenn Lowword)

a_HL_RDC1[Wortnummer] := TRUE (wenn Highword)

Beispiel:

Wortnummer im Ist-wert-Telegramm 1	ausgewählter Istwert	Anschluß an Eingang a_RDC1	Anschluß an Eingang a_HL_RDC1
0	Istwert 1 Lowword	a_RDC1[0] = SINT# 1	a_HL_RDC1[0] = FALSE
1	Istwert 1 Highword	a_RDC1[1] = SINT# 1	a_HL_RDC1[1] = TRUE
2	Istwert 0 Word	a_RDC1[2] = SINT# 0	a_HL_RDC1[2] = FALSE
3	Istwert 2 Word	a_RDC1[3] = SINT# 2	a_HL_RDC1[3] = FALSE

Eingang a_RDC2:

Die Zuordnung

"Wort im Istwert-Telegramm 2" -> "Istwert (4 bis 7)" erfolgt in a_RDC2.

a_RDC2[Wortnummer] := Istwertnummer

Eingang a_HL_RDC2:

Die Zuordnung

"Wort im Istwert-Telegramm 2" -> "Low- oder Highword des ausgewählten Istwerts" erfolgt in a_HL_RDC2.

a_HL_RDC2[Wortnummer] := FALSE (wenn Lowword)

a_HL_RDC2[Wortnummer] := TRUE (wenn Highword)

Wenn einem Istwert kein Wort aus dem Istwert-Telegramm 1 oder 2 zugeordnet werden soll, gibt man am entsprechenden Eintrag in a_RDC1 bzw. a_RDC2 als Istwertnummer eine -1 an. Die entsprechende Einstellung in a_HL_RDC1 bzw. a_HL_RDC2 ist in diesem Fall ohne Bedeutung.

a_RDC1[Wortnummer] := SINT# -1

a_RDC2[Wortnummer] := SINT# -1

Beispiel:
Das Wort 1 des Istwert-Telegramm 2 soll keinem Istwert zugeordnet werden.

Wortnummer im Ist-wert-Telegramm 2	ausgewählter Istwert	Anschluß an Eingang a_RDC2	Anschluß an Eingang a_HL_RDC2
1	kein	a_RDC2[1] = SINT# -1	a_HL_RDC2[1] ohne Bedeutung

Im CANsync-Slave findet eine Zuordnung von Istwerten zu den Worten der Istwert-Telegramme statt. Diese Zuordnung erfolgt standardmäßig analog zum CANsync-Master. Eine Überprüfung zwischen der Zuordnung im CANsync-Master und im CANsync-Slave findet nicht statt, weil es auch sinnvolle Anwendungen für abweichende Zuordnungen gibt.

7.3.17 CANsync_PD_CFG_READ_SL

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um die Belegung der Istwert-Telegramme der CANsync-Slaves in der CANsync-Anschaltung für einen CANsync-Slave zu konfigurieren.

HINWEIS

Der FB CANsync_PD_CFG_READ_SL verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
si_SL_NR	SINT 0 bis 31	Slavenummer des CANsync-Slaves, von dem Istwerte empfangen werden
a_RDC1	SINT_4_BMARRAY	Istwertnummern für Istwert-Telegramm 1
a_HL_RDC1	BOOL_4_BMARRAY	Zuordnung High- oder Lowword für Istwert-Telegramm 1
a_RDC2	SINT_4_BMARRAY	Istwertnummern für Istwert-Telegramm 2
a_HL_RDC2	INOUTPUT4_BOOL_BMARRAY	Zuordnung High- oder Lowword für Istwert-Telegramm 2

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung

Mit dem FB CANsync_PD_CFG_READ_SL erfolgt:

- die Zuordnung der Istwert-Telegramme 1 und 2 eines anderen CANsync-Slaves (Empfang) zu 8 Istwerten á 32 Bit

in einem CANsync-Slave.

Jeder CANsync-Slave kann die Istwert-Telegramme der anderen CANsync-Slaves mitlesen und die Daten in den Istwert-Telegrammen verwenden.

Die Applikation kann im CANsync-Slave 8 Istwerte á 32 Bit jedes anderen CANsync-Slaves am CANsync-Bus lesen.

Die CANsync-Slaves haben die Slavenummern 0 bis 31.

Die Istwerte haben die Istwertnummern 0 bis 7.

Ein Istwert setzt sich aus Lowword (Bit 0 bis 15) und Highword (Bit 16 bis 31) zusammen.

Im CANsync-Slave können die 2 Istwert-Telegramme jedes anderen CANsync-Slaves mit je 4 * 16 Bit Daten empfangen werden. Das sind je 4 Worte mit den Wortnummern 0 bis 3.

Mit dem FB CANsync_PD_CFG_READ_SL wird für einen anderen CANsync-Slave festgelegt, welchen Istwerten die Daten aus den 4 Worten (Wort 0 bis 3) des Istwert-Telegramm 1 und welchen Istwerten die Daten aus den 4 Worten (Wort 0 bis 3) des Istwert-Telegramm 2 des anderen CANsync-Slaves zugeordnet werden.

Die Daten aus dem Istwert-Telegramm 1 können den Istwerten 0 bis 3 zugeordnet werden, die Daten aus dem Istwert-Telegramm 2 können den Istwerten 4 bis 7 zugeordnet werden.

Jedes Wort in einem Istwert-Telegramm kann nur einem Low- oder Highword eines Istwerts zugeordnet werden.

Bei der Übertragung eines 32 Bit-Istwerts werden 2 Worte im Istwert-Telegramm benötigt (ein Wort aus dem Istwert-Telegramm wird dem Lowword eines Istwerts zugeordnet, ein anderes Wort dieses Istwert-Telegramms wird dem Highword dieses Istwerts zugeordnet).

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_SL_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) auf Ωmega Drive-Line II

```
_CANsync_CTRL_SL  AT  %MB3.100000  :  CANsync_SL_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_SL`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_SL_CTRL_BMSTRUCT`	der Datentyp
`%MB3.100000`	die Basisadresse der CANsync-Anschaltung 1 auf Ωmega Drive-Line II

Eingang si_SL_NR:

Am Eingang si_SL_NR wird die Slavenummer des CANsync-Slaves am CANsync-Bus angegeben, dessen Istwert-Telegramme konfiguriert werden.

Eingang a_RDC1:

Die Zuordnung

"Wort im Istwert-Telegramm 1" -> "Istwert (0 bis 3)" erfolgt in a_RDC1.

a_RDC1[Wortnummer] := Istwertnummer

Eingang a_HL_RDC1:

Die Zuordnung

"Wort im Istwert-Telegramm 1" -> "Low- oder Highword des ausgewählten Istwerts" erfolgt in a_HL_RDC1.

a_HL_RDC1[Wortnummer] := FALSE (wenn Lowword)

a_HL_RDC1[Wortnummer] := TRUE (wenn Highword)

Beispiel:

Wortnummer im Ist-wert-Telegramm 1	ausgewähler Istwert	Anschluß an Eingang a_RDC1	Anschluß an Eingang a_HL_RDC1
0	Istwert 1 Lowword	a_RDC1[0] = SINT# 1	a_HL_RDC1[0] = FALSE
1	Istwert 1 Highword	a_RDC1[1] = SINT# 1	a_HL_RDC1[1] = TRUE
2	Istwert 0 Word	a_RDC1[2] = SINT# 0	a_HL_RDC1[2] = FALSE
3	Istwert 2 Word	a_RDC1[3] = SINT# 2	a_HL_RDC1[3] = FALSE

Eingang a_RDC2:

Die Zuordnung

"Wort im Istwert-Telegramm 2" -> "Istwert (4 bis 7)" erfolgt in a_RDC2.

a_RDC2[Wortnummer] := Istwertnummer

Eingang a_HL_RDC2:

Die Zuordnung

"Wort im Istwert-Telegramm 2" -> "Low- oder Highword des ausgewählten Istwerts" erfolgt in a_HL_RDC2.

a_HL_RDC2[Wortnummer] := FALSE (wenn Lowword)

a_HL_RDC2[Wortnummer] := TRUE (wenn Highword)

Wenn einem Istwert kein Wort aus dem Istwert-Telegramm 1 oder 2 zugeordnet werden soll, gibt man am entsprechenden Eintrag in a_RDC1 bzw. a_RDC2 als Istwertnummer eine -1 an. Die entsprechende Einstellung in a_HL_RDC1 bzw. a_HL_RDC2 ist in diesem Fall ohne Bedeutung.

a_RDC1[Wortnummer] := SINT# -1

a_RDC2[Wortnummer] := SINT# -1

Beispiel:
Das Wort 1 des Istwert-Telegramm 2 soll keinem Istwert zugeordnet werden.

Wortnummer im Ist-wert-Telegramm 2	ausgewähler Istwert	Anschluß an Eingang a_RDC2	Anschluß an Eingang a_HL_RDC2
1	kein	a_RDC2[1] = SINT# -1	a_HL_RDC2[1] ohne Bedeutung

Im anderen CANsync-Slave findet eine Zuordnung von Istwerten zu den Worten der Istwert-Telegramme statt. Diese Zuordnung erfolgt standardmäßig analog zum (empfangenden) CANsync-Slave. Eine Überprüfung zwischen der Zuordnung im (empfangenden) CANsync-Slave und im anderen CANsync-Slave findet nicht statt, weil es auch sinnvolle Anwendungen für abweichende Zuordnungen gibt.

7.3.18 CANsync_PD_CFG_SL

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um die Belegung der Sollwert- und der Istwert-Telegramme der CANsync-Anschaltung für einen CANsync-Slave zu konfigurieren.

HINWEIS

Der FB CANsync_PD_CFG_SL verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
a_WRC1	SINT_4_BMARRAY	Sollwertnummern für Sollwert-Telegramm 1
a_HL_WRC1	BOOL_4_BMARRAY	Zuordnung High- oder Lowword für Sollwert-Telegramm 1
a_WRC2	SINT_4_BMARRAY	Sollwertnummern für Sollwert-Telegramm 2
a_HL_WRC2	BOOL_4_BMARRAY	Zuordnung High- oder Lowword für Sollwert-Telegramm 2
a_RDC1	SINT_4_BMARRAY	Istwertnummern für Istwert-Telegramm 1
a_HL_RDC1	BOOL_4_BMARRAY	Zuordnung High- oder Lowword für Istwert-Telegramm 1
a_RDC2	SINT_4_BMARRAY	Istwertnummern für Istwert-Telegramm 2
a_HL_RDC2	BOOL_4_BMARRAY	Zuordnung High- oder Lowword für Istwert-Telegramm 2

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung

Mit dem FB CANsync_PD_CFG_SL erfolgt:

- die Zuordnung der Sollwert-Telegramme 1 und 2 (Empfang)
  zu 8 Sollwerten á 32 Bit

und

- die Zuordnung von 8 Istwerten á 32 Bit
  zu den Istwert-Telegrammen 1 und 2 (Senden)

in einem CANsync-Slave.

Sollwerte empfangen:

Die Applikation kann im CANsync-Slave 8 Sollwerte á 32 Bit lesen. Die Sollwerte haben die Sollwertnummern 0 bis 7.

Ein Sollwert setzt sich aus Lowword (Bit 0 bis 15) und Highword (Bit 16 bis 31) zusammen.

Im CANsync-Slave können 2 Sollwert-Telegramme mit je 4 * 16 Bit Daten empfangen werden. Das sind je 4 Worte mit den Wortnummern 0 bis 3.

Mit dem FB CANsync_PD_CFG_SL wird festgelegt,
welchen Sollwerten die Daten aus den 4 Worten (Wort 0 bis 3) des Sollwert-Telegramm 1 und
welchen Sollwerten die Daten aus den 4 Worten (Wort 0 bis 3) des Sollwert-Telegramm 2
zugeordnet werden.

Die Daten aus dem Sollwert-Telegramm 1 können den Sollwerten 0 bis 3 zugeordnet werden,
die Daten aus dem Sollwert-Telegramm 2 können den Sollwerten 4 bis 7 zugeordnet werden.

Jedes Wort in einem Sollwert-Telegramm kann nur einem Low- oder Highword eines Sollwerts zugeordnet werden.

Bei der Übertragung eines 32 Bit-Sollwerts werden 2 Worte im Sollwert-Telegramm benötigt (ein Wort aus dem Sollwert-Telegramm wird dem Lowword eines Sollwerts zugeordnet, ein anderes Wort dieses Sollwert-Telegramms wird dem Highword dieses Sollwerts zugeordnet).

Istwerte senden:

Weiterhin kann die Applikation im CANsync-Slave 8 Istwerte á 32 Bit schreiben. Die Istwerte haben die Istwertnummern 0 bis 7.

Ein Istwert setzt sich aus Lowword (Bit 0 bis 15) und Highword (Bit 16 bis 31) zusammen.

In den 2 Istwert-Telegrammen können je 4 * 16 Bit Daten an den CANsync-Master gesendet werden. Das sind 4 Worte mit den Wortnummern 0 bis 3.

Mit dem FB CANsync_PD_CFG_SL wird festgelegt,
welche Daten in die 4 Worte (Wort 0 bis 3) des Istwert-Telegramm 1 und
welche Daten in die 4 Worte (Wort 0 bis 3) des Istwert-Telegramm 2 eingetragen werden.

Die Daten für Istwert-Telegramm 1 können aus den Istwerten 0 bis 3 gewählt werden,
die Daten für Istwert-Telegramm 2 können aus den Istwerten 4 bis 7 gewählt werden.

Jedem Wort in einem Istwert-Telegramm kann ein Low- oder Highword eines Istwerts zugeordnet werden. Bei der Übertragung eines 32 Bit-Istwerts werden 2 Worte im Istwert-Telegramm benötigt.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_SL_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) auf Ωmega Drive-Line II

```
_CANsync_CTRL_SL  AT  %MB3.100000  :  CANsync_SL_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_SL`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_SL_CTRL_BMSTRUCT`	der Datentyp
`%MB3.100000`	die Basisadresse der CANsync-Anschaltung 1 auf Ωmega Drive-Line II

Eingang a_WRC1:

Die Zuordnung

"Wort im Sollwert-Telegramm 1" -> "Sollwert (0 bis 3)" erfolgt in a_WRC1.

a_WRC1[Wortnummer] := Sollwertnummer

Eingang a_HL_WRC1:

Die Zuordnung

"Wort im Sollwert-Telegramm 1" -> "Low- oder Highword des ausgewählten Sollwerts" erfolgt in a_HL_WRC1.

a_HL_WRC1[Wortnummer] := FALSE (wenn Lowword)

a_HL_WRC1[Wortnummer] := TRUE (wenn Highword)

Beispiel:

Wortnummer im Soll-wert-Telegramm 1	ausgewählter Sollwert	Anschluß an Eingang a_WRC1	Anschluß an Eingang a_HL_WRC1
0	Sollwert 1 Lowword	a_WRC1[0] = SINT# 1	a_HL_WRC1[0] = FALSE
1	Sollwert 1 Highword	a_WRC1[1] = SINT# 1	a_HL_WRC1[1] = TRUE
2	Sollwert 0 Word	a_WRC1[2] = SINT# 0	a_HL_WRC1[2] = FALSE
3	Sollwert 2 Word	a_WRC1[3] = SINT# 2	a_HL_WRC1[3] = FALSE

Eingang a_WRC2:

Die Zuordnung

"Wort im Sollwert-Telegramm 2" -> "Sollwert (4 bis 7)" erfolgt in a_WRC2.

a_WRC2[Wortnummer] := Sollwertnummer

Eingang a_HL_WRC2:

Die Zuordnung

"Wort im Sollwert-Telegramm 2" -> "Low- oder Highword des ausgewählten Sollwerts" erfolgt in a_HL_WRC2.

a_HL_WRC2[Wortnummer] := FALSE (wenn Lowword)

a_HL_WRC2[Wortnummer] := TRUE (wenn Highword)

Wenn einem Sollwert kein Wort aus dem Sollwert-Telegramm 1 oder 2 zugeordnet werden soll, gibt man am entsprechenden Eintrag in a_WRC1 bzw. a_WRC2 als Sollwertnummer eine -1 an.

Die entsprechende Einstellung in a_HL_WRC1 bzw. a_HL_WRC2 ist in diesem Fall ohne Bedeutung.

a_WRC1[Wortnummer] := SINT# -1

a_WRC2[Wortnummer] := SINT# -1

Beispiel:

Das Wort 1 des Sollwert-Telegramm 2 soll keinem Sollwert zugeordnet werden.

Wortnummer im Sollwert-Telegramm 2	ausgewählter Sollwert	Anschluß an Eingang a_WRC2	Anschluß an Eingang a_HL_WRC2
1	kein	a_WRC2[1] = SINT# -1	a_HL_WRC2[1] ohne Bedeutung

Im CANsync-Master findet eine Zuordnung von Sollwerten zu den Worten der Sollwert-Telegramme statt. Diese Zuordnung erfolgt standardmäßig analog zum CANsync-Slave. Eine Überprüfung zwischen der Zuordnung im CANsync-Master und im CANsync-Slave findet nicht statt, weil es auch sinnvolle Anwendungen für abweichende Zuordnungen gibt.

Eingang a_RDC1:

Die Zuordnung

"Istwert (0 bis 3)" -> "Wort im Istwert-Telegramm 1" erfolgt in a_RDC1.

a_RDC1[Wortnummer] := Istwertnummer

Eingang a_HL_RDC1:

Die Zuordnung

"Low- oder Highword des ausgewählten Istwerts" -> "Wort im Istwert-Telegramm 1" erfolgt in a_HL_RDC1.

a_HL_RDC1[Wortnummer] := FALSE (wenn Lowword)

a_HL_RDC1[Wortnummer] := TRUE (wenn Highword)

Beispiel:

Wortnummer im Istwert-Telegramm 1	ausgewähler Istwert	Anschluß an Eingang a_RDC1	Anschluß an Eingang a_HL_RDC1
0	Istwert 1 Lowword	a_RDC1[0] = SINT# 1	a_HL_RDC1[0] = FALSE
1	Istwert 1 Highword	a_RDC1[1] = SINT# 1	a_HL_RDC1[1] = TRUE
2	Istwert 0 Word	a_RDC1[2] = SINT# 0	a_HL_RDC1[2] = FALSE
3	Istwert 2 Word	a_RDC1[3] = SINT# 2	a_HL_RDC1[3] = FALSE

Eingang a_RDC2:

Die Zuordnung

"Istwert (4 bis 7)" -> "Wort im Istwert-Telegramm 2" erfolgt in a_RDC2.

a_RDC2[Wortnummer] := Istwertnummer

Eingang a_HL_RDC2:

Die Zuordnung

"Low- oder Highword des ausgewählten Istwerts" -> "Wort im Istwert-Telegramm 2" erfolgt in a_HL_RDC2.

a_HL_RDC2[Wortnummer] := FALSE (wenn Lowword)

a_HL_RDC2[Wortnummer] := TRUE (wenn Highword)

Wenn einem Wort im Istwert-Telegramm 1 oder 2 kein Istwert zugeordnet werden soll, gibt man am entsprechenden Eintrag in a_RDC1 bzw. a_RDC2 als Istwertnummer eine -1 an. Die entsprechende Einstellung in a_HL_RDC1 bzw. a_HL_RDC2 ist in diesem Fall ohne Bedeutung.

a_RDC1[Wortnummer] := SINT# -1

a_RDC2[Wortnummer] := SINT# -1

Beispiel:
Dem Wort 1 des Istwert-Telegramm 2 soll kein Istwert zugeordnet werden.

Wortnummer im Istwert-Telegramm 2	ausgewähler Istwert	Anschluß an Eingang a_RDC2	Anschluß an Eingang a_HL_RDC2
1	kein	a_RDC2[1] = SINT# -1	a_HL_RDC2[1] ohne Bedeutung

Im CANsync-Master findet eine Zuordnung der Worte der Istwert-Telegramme zu Istwerten statt. Diese Zuordnung erfolgt standardmäßig analog zum CANsync-Slave. Eine Überprüfung zwischen der Zuordnung im CANsync-Master und im CANsync-Slave findet nicht statt, weil es auch sinnvolle Anwendungen für abweichende Zuordnungen gibt.

7.3.19 CANsync_PD_COMM_MA

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um die Prozeßdatenkommunikation (Sollwerte und Istwerte) der CANsync-Master-Anschaltung durchzuführen.

HINWEIS

Der FB CANsync_PD_COMM_MA verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
a_RD_ARRAY	CANsync_RD_BMARRAY	Ausgabearray für Istwerte der CANsync-Slaves
a_WR_VALUES_SEND	DINT_8_BMARRAY	Sollwerte, die gesendet werden sollen
si_WRC1_SEND	SINT 5	Kommando, daß Sollwert-Telegramm 1 gesendet werden soll
si_WRC2_SEND	SINT 5	Kommando, daß Sollwert-Telegramm 2 gesendet werden soll
si_RD_SL_NR1_RECEIVE	SINT -1, 0 bis 31	Slavenummer des CANsync-Slaves, von dem Istwert-Telegramm 1 angefordert werden soll
si_RD_SL_NR2_RECEIVE	SINT -1, 0 bis 31	Slavenummer des CANsync-Slaves, von dem Istwert-Telegramm 2 angefordert werden soll
si_MAX_SL_NR	SINT 0 bis 31	maximale Slavenummer für automatische Erhöhung [a)]
x_COPY_TO_RD_ARRAY	BOOL	Angabe, ob Istwerte in das RD_ARRAY kopiert werden sollen

a) Dieser Eingang entspricht dem Eingang si_MAX_SL_NR am FB CANsync_COMM_CONTROL_MA als maximale Slavenummer für automatische Abfrage nach Steuerwort-Senden-Aufträgen, Parameter-Aufträgen und/oder Up-/Download-Aufträgen.

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
a_RD_ARRAY	CANsync_RD_BMARRAY	Ausgabearray für Istwerte der CANsync-Slaves
si_RD_SL_NR1_RECEIVED	SINT -1, 0 bis 31	Anzeige der Slavenummer des CANsync-Slaves, von dem Istwert-Telegramm 1 empfangen wurde
si_RD_SL_NR2_RECEIVED	SINT -1, 0 bis 31	Anzeige der Slavenummer des CANsync-Slaves, von dem Istwert-Telegramm 2 empfangen wurde

Mit diesem FB werden an die CANsync-Anschaltung Sollwerte (a_WR_VALUES_SEND) übergeben, die mittels der Sollwert-Telegramme an die CANsync-Slaves gesendet werden [a)]. Weiterhin werden von den CANsync-Slaves die Istwert-Telegramme angefordert [a)] und die Istwerte (in a_RD_ARRAY) ausgegeben.

In jedem CANsync-Intervall werden die Sollwert-Telegramme 1 und 2 an die CANsync-Slaves gesendet.

In jedem CANsync-Intervall wird das Istwert-Telegramm 1 vom CANsync-Slave mit der Slavenummer si_RD_SL_NR1_RECEIVE angefordert und in jedem CANsync-Intervall wird das Istwert-Telegramm 2 vom CANsync-Slave mit der Slavenummer si_RD_SL_NR2_RECEIVE angefordert.

Die Anforderung der Istwert-Telegramme der CANsync-Slaves läuft automatisch wie folgt ab:

In jedem CANsync-Intervall wird die Slavenummer des CANsync-Slaves (von dem Istwert-Telegramm 1 und 2 angefordert werden) automatisch um eins erhöht. Diese Erhöhung wird bis zur si_MAX_SL_NR durchgeführt. Danach wird wieder mit der Abfrage für den CANsync-Slave mit der Slavenummer 0 begonnen usw. (si_RD_SL_NR1_RECEIVE und si_RD_SL_NR2_RECEIVE werden dann nicht belegt).

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Ein-/Ausgang a_RD_ARRAY (Istwerte 0 bis 7 der CANsync-Slaves 0 bis 31):

An a_RD_ARRAY wird eine Variable vom Datentyp CANsync_RD_BMARRAY angeschlossen.

a) sofern eine entsprechende Konfigurierung für die Sollwert-Telegramme 1 und 2 mit dem FB CANsync_PD_CFG_MA bzw. die Konfigurierung für die Istwert-Telegramme 1 und 2 von jedem CANsync-Slave mit dem FB CANsync_PD_CFG_READ_MA erfolgte.

Der Datentyp CANsync_RD_BMARRAY ist ein 2-dimensionales Feld von 32 (CANsync-Slaves) á 16 Istwerten [a)].

D. h. der Datentyp CANsync_RD_BMARRAY ist ein Feld von 32 Einträgen des Datentyps DINT_16_BMARRAY. Der Datentyp DINT_16_BMARRAY ist ein Feld von 16 Einträgen des Datentyps Double Integer:

```
DINT_16_BMARRAY              : ARRAY [0..15] OF DINT;
CANsync_RD_BMARRAY           : ARRAY [0..31] OF DINT_16_BMARRAY
```

Beispiel:

```
a_Istwerte                   : CANsync_RD_BMARRAY;
```

dabei ist:

`a_Istwerte`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`CANsync_RD_BMARRAY`	der Datentyp

Der Zugriff auf die einzelnen Istwerte erfolgt nach dem Muster:

Variablenname[Slavenummer des CANsync-Slaves][Nummer des Istwerts]

HINWEIS

Zwischen "Variablenname" und den eckigen Klammern sowie zwischen den eckigen Klammern steht kein Punkt.

Beispiel: Die Variable di_Istwert_21_6 wird mit dem Istwert 6 des CANsync-Slaves mit der Slavenummer 21 beschrieben (in Structured Text (ST)):

```
di_Istwert_21_6 := a_Istwerte[21][6];
```

Eingänge a_WR_VALUES_SEND, si_WRC1_SEND und si_WRC2_SEND:

An dem Eingang a_WR_VALUES_SEND wird eine Variable vom Datentyp DINT_8_BMARRAY angeschlossen. Der Datentyp DINT_8_BMARRAY ist ein Feld mit 8 Einträgen des Datentyps Double Integer:

```
DINT_8_BMARRAY               : ARRAY [0..7] OF DINT;
```

Beispiel:

```
a_Sollwerte                  : DINT_8_BMARRAY;
```

dabei ist:

`a_Sollwerte`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`DINT_8_BMARRAY`	der Datentyp

---

a) zur Zeit werden die Istwerte 0 bis 3 (Istwert-Telegramm 1) und die Istwerte 4 bis 7 (Istwert-Telegramm 2) unterstützt.

Die Sollwerte 0 bis 7 werden dann z. B. in den Feld-Elementen `a_Sollwerte[0]` bis `a_Sollwerte[7]` erwartet.

Am Eingang si_WRC1_SEND wird angegeben wann die Sollwerte 0 bis 3 gültig sind. Dann kann das Sollwert-Telegramm 1 gesendet werden. Der Wert 5 gibt an, daß die Sollwerte 0 bis 3 gültig sind, während jeder andere Wert angibt, daß die Sollwerte 0 bis 3 nicht gültig sind.

Am Eingang si_WRC2_SEND wird angegeben wann die Sollwerte 4 bis 7 gültig sind. Dann kann das Sollwert-Telegramm 2 gesendet werden. Der Wert 5 gibt an, daß die Sollwerte 4 bis 7 gültig sind, während jeder andere Wert angibt, daß die Sollwerte 4 bis 7 nicht gültig sind.

HINWEIS

Im Sollwert-Telegramm 1 werden die Sollwerte 0 bis 3 gesendet, im Sollwert-Telegramm 2 werden die Sollwerte 4 bis 7 gesendet. Lücken bei den Sollwertnummern sind zulässig.

Eingänge si_RD_SL_NR1_RECEIVE, si_RD_SL_NR2_RECEIVE und si_MAX_SL_NR:

Am Eingang si_RD_SL_NR1_RECEIVE wird die Slavenummer des CANsync-Slaves angegeben, von dem das Istwert-Telegramm 1 angefordert wird.

Zum automatischen Anfordern des Istwert-Telegramm 1 der CANsync-Slaves wird si_RD_SL_NR1_RECEIVE nicht belegt (oder gleich -128 gesetzt) und die höchste Slavenummer an si_MAX_SL_NR angegeben.

Am Eingang si_RD_SL_NR2_RECEIVE wird die Slavenummer des CANsync-Slaves angegeben, von dem das Istwert-Telegramm 2 angefordert wird.

Zum automatischen Anfordern des Istwert-Telegramm 2 der CANsync-Slaves wird si_RD_SL_NR2_RECEIVE nicht belegt (oder gleich -128 gesetzt) und die höchste Slavenummer an si_MAX_SL_NR angegeben.

Dann wird in jedem CANsync-Intervall die Slavenummer des CANsync-Slaves (von dem Istwert-Telegramm 1 und/oder 2 automatisch angefordert werden) um eins erhöht, bis die Nummer am Eingang si_MAX_SL_NR erreicht ist.

Danach wird wieder mit der Abfrage für den CANsync-Slave mit der Slavenummer 0 begonnen usw.

Die explizite Vorgabe und die automatische Erhöhung kann für die Istwert-Telegramme 1 und 2 auch gemischt werden.

Die höchste Slavenummer eines CANsync-Slaves von dem Istwert-Telegramme angefordert werden (Eingang si_MAX_SL_NR) wird auch für die Bedarfsdatenkommunikation (Steuerwort, Parameter, Up-/Download) verwendet.

Ist si_MAX_SL_NR = -1, bleibt auf der CANsync-Anschaltung der Wert unverändert.

Voreinstellung ist si_MAX_SL_NR = -1, d. h. auf der CANsync-Anschaltung bleibt der Wert unverändert.

HINWEIS

Dieser Eingang entspricht dem Eingang si_MAX_SL_NR am FB CANsync_COMM_CONTROL_MA. D. h. mit dem Eingang si_MAX_SL_NR am FB CANsync_COMM_CONTROL_MA oder dem Eingang si_MAX_SL_NR am FB CANsync_PD_COMM_MA wird die höchste Slavenummer eines CANsync-Slaves angegeben.
Es darf nur einer von beiden Eingängen verwendet werden!

Eingang x_COPY_RD_ARRAY:

Am Eingang x_COPY_TO_RD_ARRAY wird mit TRUE angegeben daß die empfangenen Istwerte in das zweidimensionale Feld an a_RD_ARRAY eingetragen werden.

Wird an x_COPY_TO_RD_ARRAY FALSE angegeben, oder ist x_COPY_TO_RD_ARRAY nicht belegt, werden die empfangenen Istwerte nicht in das zweidimensionale Feld an a_RD_ARRAY eingetragen.

Ausgänge si_RD_SL_NR1_RECEIVED, si_RD_SL_NR2_RECEIVED:

Am Ausgang si_RD_SL_NR1_RECEIVED wird die Slavenummer des CANsync-Slaves angezeigt von dem im letzten CANsync-Intervall das Istwert-Telegramm 1 empfangen wurde. Wenn in einem CANsync-Intervall kein Istwert-Telegramm 1 empfangen wurde, wird -128 an si_RD_SL_NR1_RECEIVED angezeigt.

Am Ausgang si_RD_SL_NR2_RECEIVED wird die Slavenummer des CANsync-Slaves angezeigt von dem im letzten CANsync-Intervall das Istwert-Telegramm 2 empfangen wurde. Wenn in einem CANsync-Intervall kein Istwert-Telegramm 2 empfangen wurde, wird -128 an si_RD_SL_NR2_RECEIVED angezeigt.

7.3.20 CANsync_PD_COMM_READ_MA

HINWEIS

Der FB CANsync_PD_COMM_READ_MA ist aus Kompatibiltitätsgründen vorhanden und sollte in neuen Projekten nicht eingesetzt werden. Die Istwerte aller CANsync-Slaves werden mit dem FB CANsync_PD_COMM_MA in a_RD_BMARRAY ausgegeben.

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um in einer CANsync-Master-Anschaltung die Prozeßdaten-Istwerte von einem CANsync-Slave auszugeben.

HINWEIS

Der FB CANsync_PD_COMM_READ_MA verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
si_SL_NR	SINT 0 bis 31	Slavenummer des CANsync-Slaves, von dem Istwerte gelesen werden

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
a_RD_VALUES	DINT_8_BMARRAY	Ausgabearray für Istwerte die vom CANsync-Slave empfangen wurden
si_RD_SL_NR1_RECEIVED	SINT	Anzeige daß Istwert-Telegramm 1 empfangen wurde
si_RD_SL_NR2_RECEIVED	SINT	Anzeige daß Istwert-Telegramm 2 empfangen wurde

Mit diesem FB werden die Istwerte eines CANsync-Slaves ausgegeben (a_RD_VALUES). Die Anforderung der Istwert-Telegramme des CANsync-Slaves muß durch den FB CANsync_PD_COMM_MA erfolgen.

Der Einsatz des FB CANsync_PD_COMM_READ_MA ist nur sinnvoll, wenn am FB CANsync_PD_COMM_MA die Istwerte nicht ausgegeben werden, d. h. wenn FB CANsync_PD_COMM_MA, Eingang x_COPY_TO_RD_ARRAY = FALSE ist.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Eingang si_SL_NR:

Am Eingang si_SL_NR wird die Slavenummer des CANsync-Slaves am CANsync-Bus angegeben, dessen Istwerte ausgegeben werden.

Ausgang a_RD_VALUES (Istwerte 0 bis 7 des CANsync-Slaves si_SL_NR):

An a_RD_VALUES wird eine Variable vom Datentyp DINT_8_BMARRAY angeschlossen. Der Datentyp DINT_8_BMARRAY ist ein Feld von 8 Einträgen des Datentyps Double Integer:

```
DINT_8_BMARRAY                  : ARRAY [0..7] OF DINT;
```

Beispiel:

```
a_Istwerte_3 : DINT_8_BMARRAY;
```

dabei ist:

`a_Istwerte_3`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`DINT_8_BMARRAY`	der Datentyp

Der Zugriff auf die einzelnen Istwerte erfolgt nach dem Muster:

Variablenname[Nummer des Istwerts]

Beispiel: Die Variable `di_Istwert_3_6` wird mit dem Istwert 6 des CANsync-Slaves mit der Slavenummer 3 beschrieben (in Structured Text (ST)):

```
di_Istwert_3_6 := a_Istwerte_3[6];
```

Zur Zeit werden die Istwerte 0 bis 3 (Istwert-Telegramm 1) und die Istwerte 4 bis 7 (Istwert-Telegramm 2) unterstützt.

Ausgänge si_RD_SL_NR1_RECEIVED, si_RD_SL_NR2_RECEIVED:

Am Ausgang si_RD_SL_NR1_RECEIVED wird mit 2 angezeigt, ob das Istwert-Telegramm 1 empfangen wurde. Wenn das Istwert-Telegramm 1 nicht empfangen wurde, ist si_RD_SL_NR1_RECEIVED = 0. Nur wenn das Istwert-Telegramm 1 empfangen wurde, werden die neuen Istwerte 0 bis 3 an a_RD_VALUES ausgegeben.

Am Ausgang si_RD_SL_NR2_RECEIVED wird mit 2 angezeigt, ob das Istwert-Telegramm 2 empfangen wurde. Wenn das Istwert-Telegramm 2 nicht empfangen wurde, ist si_RD_SL_NR2_RECEIVED = 0. Nur wenn das Istwert-Telegramm 2 empfangen wurde, werden die neuen Istwerte 4 bis 7 an a_RD_VALUES ausgegeben.

7.3.21 CANsync_PD_COMM_READ_SL

HINWEIS

Der FB CANsync_PD_COMM_READ_SL ist aus Kompatibiltitätsgründen vorhanden und sollte in neuen Projekten nicht eingesetzt werden. Die mitgelesenen Istwerte der anderen CANsync-Slaves werden mit dem FB CANsync_PD_COMM_SL in a_RD_BMARRAY ausgegeben.

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um in einer CANsync-Slave-Anschaltung die Prozeßdaten-Istwerte von einem CANsync-Slave auszugeben.

HINWEIS

Der FB CANsync_PD_COMM_READ_SL verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
si_SL_NR	SINT 0 bis 31	Slavenummer des CANsync-Slaves, von dem Istwerte mitgelesen werden

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
a_RD_VALUES	DINT_8_BMARRAY	Ausgabearray für Istwerte die vom CANsync-Slave empfangen wurden
si_RD_SL_NR1_RECEIVED	SINT	Anzeige daß Istwert-Telegramm 1 empfangen wurde
si_RD_SL_NR2_RECEIVED	SINT	Anzeige daß Istwert-Telegramm 2 empfangen wurde

Mit diesem FB werden die mitgelesenen Istwerte eines anderen CANsync-Slaves ausgegeben (a_RD_VALUES). Die Anforderung der Istwert-Telegramme des anderen CANsync-Slaves muß durch den CANsync-Master mit dem FB CANsync_PD_COMM_MA erfolgen.

Der Einsatz des FB CANsync_PD_COMM_READ_SL ist nur sinnvoll, wenn am FB CANsync_PD_COMM_SL die Istwerte nicht ausgegeben werden, d. h. wenn FB CANsync_PD_COMM_SL, Eingang x_COPY_TO_RD_ARRAY = FALSE ist.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_SL_CTRL_BMSTRUCT angeschlossen werden.Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) auf Ωmega Drive-Line II

```
_CANsync_CTRL_SL  AT  %MB3.100000  :  CANsync_SL_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_SL`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_SL_CTRL_BMSTRUCT`	der Datentyp
`%MB3.100000`	die Basisadresse der CANsync-Anschaltung 1 auf Ωmega Drive-Line II

Eingang si_SL_NR:

Am Eingang si_SL_NR wird die Slavenummer des CANsync-Slaves am CANsync-Bus angegeben, dessen mitgelesene Istwerte ausgegeben werden.

Ausgang a_RD_VALUES (Istwerte 0 bis 7 des CANsync-Slaves si_SL_NR):

An a_RD_VALUES wird eine Variable vom Datentyp DINT_8_BMARRAY angeschlossen. Der Datentyp DINT_8_BMARRAY ist ein Feld von 8 Einträgen des Datentyps Double Integer:

```
DINT_8_BMARRAY                  : ARRAY [0..7] OF DINT;
```

Beispiel:

```
a_Istwerte_3 : DINT_8_BMARRAY;
```

dabei ist:

`a_Istwerte_3`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`DINT_8_BMARRAY`	der Datentyp

Der Zugriff auf die einzelnen Istwerte erfolgt nach dem Muster:

Variablenname[Nummer des Istwerts]

Beispiel: Die Variable `di_Istwert_22_6` wird mit dem Istwert 6 des CANsync-Slaves mit der Slavenummer 22 beschrieben (in Structured Text (ST)):

```
di_Istwert_22_6 := a_Istwerte_22[6];
```

Zur Zeit werden die Istwerte 0 bis 3 (Istwert-Telegramm 1) und die Istwerte 4 bis 7 (Istwert-Telegramm 2) unterstützt.

Ausgänge si_RD_SL_NR1_RECEIVED, si_RD_SL_NR2_RECEIVED:

Am Ausgang si_RD_SL_NR1_RECEIVED wird mit 2 angezeigt, ob das Istwert-Telegramm 1 empfangen wurde. Wenn das Istwert-Telegramm 1 nicht empfangen wurde, ist si_RD_SL_NR1_RECEIVED = 0. Nur wenn das Istwert-Telegramm 1 empfangen wurde, werden die neuen Istwerte 0 bis 3 an a_RD_VALUES ausgegeben.

Am Ausgang si_RD_SL_NR2_RECEIVED wird mit 2 angezeigt, ob das Istwert-Telegramm 2 empfangen wurde. Wenn das Istwert-Telegramm 2 nicht empfangen wurde, ist si_RD_SL_NR2_RECEIVED = 0. Nur wenn das Istwert-Telegramm 2 empfangen wurde, werden die neuen Istwerte 4 bis 7 an a_RD_VALUES ausgegeben.

7.3.22 CANsync_PD_COMM_SL

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um die Prozeßdatenkommunikation (Sollwerte und Istwerte) der CANsync-Slave-Anschaltung durchzuführen.

HINWEIS

Der FB CANsync_PD_COMM_SL verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
a_RD_ARRAY	CANsync_RD_BMARRAY	Ausgabearray für Istwerte der CANsync-Slaves
a_RD_VALUES_SEND	DINT_8_BMARRAY	Istwerte, die an den CANsync-Master gesendet werden
si_RDC1_SEND	SINT 5	Kommando, daß Istwert-Telegramm 1 gesendet werden soll
si_RDC2_SEND	SINT 5	Kommando, daß Istwert-Telegramm 2 gesendet werden soll
x_COPY_TO_RD_ARRAY	BOOL	Angabe, ob empfangene Istwerte in das RD_ARRAY kopiert werden sollen

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
a_RD_ARRAY	CANsync_RD_BMARRAY	Ausgabearray für Istwerte der CANsync-Slaves
a_WR_VALUES_RECEIVED	DINT_8_BMARRAY	Sollwerte, die vom CANsync-Master empfangen wurden
si_WRC1_RECEIVED	SINT	Anzeige daß Sollwert-Telegramm 1 empfangen wurde
si_WRC2_RECEIVED	SINT	Anzeige daß Sollwert-Telegramm 2 empfangen wurde
si_RD_SL_NR1_RECEIVED	SINT	Anzeige Slavenummer des CANsync-Slaves, von dem Istwert-Telegramm 1 empfangen wurde
si_RD_SL_NR2_RECEIVED	SINT	Anzeige Slavenummer des CANsync-Slaves, von dem Istwert-Telegramm 2 empfangen wurde

Mit diesem FB werden an die CANsync-Anschaltung Istwerte (a_RD_VALUES_SEND) übergeben, die mittels der Istwert-Telegramme an den CANsync-Master gesendet werden [a)]. Weiterhin werden von den anderen CANsync-Slaves am CANsync-Bus die Istwert-Telegramme mitgelesen [a)] und die Istwerte (in a_RD_ARRAY) ausgegeben.

In jedem CANsync-Intervall werden die Sollwert-Telegramme 1 und 2 an die CANsync-Slaves gesendet und von diesem FB an a_WR_VALUES_RECEIVED ausgegeben.

In jedem CANsync-Intervall fordert der CANsync-Master von einem CANsync-Slave das Istwert-Telegramm 1 an. Nur wenn der CANsync-Master von diesem CANsync-Slave das Istwert-Telegramm 1 anfordert, wird das Istwert-Telegramm 1 gesendet.

In jedem CANsync-Intervall fordert der CANsync-Master von einem CANsync-Slave das Istwert-Telegramm 2 an. Nur wenn der CANsync-Master von diesem CANsync-Slave das Istwert-Telegramm 2 anfordert, wird das Istwert-Telegramm 2 gesendet.

Dieser FB kann die Istwert-Telegramme 1 und 2 der anderen CANsync-Slaves mitlesen und auswerten wenn der CANsync-Master diese Istwert-Telegramme anfordert.

HINWEIS

Ein CANsync-Slave kann Istwert-Telegramme anderer CANsync-Slaves auswerten aber nicht anfordern!

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_SL_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) auf Ωmega Drive-Line II

```
_CANsync_CTRL_SL  AT  %MB3.100000  :  CANsync_SL_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_SL`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_SL_CTRL_BMSTRUCT`	der Datentyp
`%MB3.100000`	die Basisadresse der CANsync-Anschaltung 1 auf Ωmega Drive-Line II

Ein-/Ausgang a_RD_ARRAY (Istwerte 0 bis 7 der (anderen) CANsync-Slaves 0 bis 31):

An a_RD_ARRAY wird eine Variable vom Datentyp CANsync_RD_BMARRAY angeschlossen.

---

a) sofern eine entsprechende Konfigurierung für die Sollwert-Telegramme 1 und 2 mit dem FB CANsync_PD_CFG_SL bzw. die Konfigurierung für die Istwert-Telegramme 1 und 2 von jedem anderen (außer diesem) CANsync-Slave mit dem FB CANsync_PD_CFG_READ_SL erfolgte.

Der Datentyp CANsync_RD_BMARRAY ist ein 2-dimensionales Feld von 32 (CANsync-Slaves) á 16 Istwerten [a)].

D. h. der Datentyp CANsync_RD_BMARRAY ist ein Feld von 32 Einträgen des Datentyps DINT_16_BMARRAY. Der Datentyp DINT_16_BMARRAY ist ein Feld von 16 Einträgen des Datentyps Double Integer:

```
DINT_16_BMARRAY                 : ARRAY [0..15] OF DINT;
CANsync_RD_BMARAY               : ARRAY [0..31] OF DINT_16_BMARRAY
```

Beispiel:

```
a_Istwerte : CANsync_RD_BMARRAY;
```

dabei ist:

`a_Istwerte`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`CANsync_RD_BMARRAY`	der Datentyp

Der Zugriff auf die einzelnen Istwerte erfolgt nach dem Muster:

Variablenname[Slavenummer des CANsync-Slaves][Nummer des Istwerts]

HINWEIS

Zwischen "Variablenname" und den eckigen Klammern sowie zwischen den eckigen Klammern steht kein Punkt.

Beispiel: Die Variable `di_Istwert_21_6` wird mit dem Istwert 6 des CANsync-Slaves mit der Slavenummer 21 beschrieben (in Structured Text (ST)):

```
di_Istwert_21_6 := a_Istwerte[21][6];
```

HINWEIS

Die mitgelesenen Istwerte der anderen CANsync-Slaves werden in a_RD_ARRAY eingetragen, jedoch nicht die Istwerte dieses CANsync-Slaves.

Eingänge a_RD_VALUES_SEND, si_RDC1_SEND und si_RDC2_SEND:

An dem Eingang a_RD_VALUES_SEND wird eine Variable vom Datentyp DINT_8_BMARRAY angeschlossen. Der Datentyp DINT_8_BMARRAY ist ein Feld mit 8 Einträgen des Datentyps Double Integer:

```
DINT_8_BMARRAY                  : ARRAY [0..7] OF DINT;
```

[a)] zur Zeit werden die Istwerte 0 bis 3 (Istwert-Telegramm 1) und die Istwerte 4 bis 7 (Istwert-Telegramm 2) unterstützt.

Beispiel:

```
a_Istwerte_Senden : DINT_8_BMARRAY;
```

dabei ist:

`a_Istwerte_Senden`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`DINT_8_BMARRAY`	der Datentyp

Die Istwerte 0 bis 7 werden in die Einträge 0 bis 7 (z. B. `a_Istwerte_Senden[0]` bis `a_Istwerte_Senden[7]`) eingetragen.

Am Eingang si_RDC1_SEND wird angegeben wann die Istwerte 0 bis 3 gültig sind. Dann kann das Istwert-Telegramm 1 gesendet werden, sofern der CANsync-Master es anfordert. Der Wert 5 gibt an, daß die Istwerte 0 bis 3 gültig sind, während jeder andere Wert angibt, daß die Istwerte 0 bis 3 nicht gültig sind.

Am Eingang si_RDC2_SEND wird angegeben wann die Sollwerte 4 bis 7 gültig sind. Dann kann das Istwert-Telegramm 2 gesendet werden, sofern der CANsync-Master es anfordert. Der Wert 5 gibt an, daß die Istwerte 4 bis 7 gültig sind, während jeder andere Wert angibt, daß die Istwerte 4 bis 7 nicht gültig sind.

HINWEIS

Im Istwert-Telegramm 1 werden die Istwerte 0 bis 3 gesendet, im Istwert-Telegramm 2 werden die Istwerte 4 bis 7 gesendet. Lücken bei den Istwertnummern sind zulässig.

Eingang x_COPY_RD_ARRAY:

Am Eingang x_COPY_TO_RD_ARRAY wird mit TRUE angegeben daß die mitgelesenen Istwerte in das zweidimensionale Feld an a_RD_ARRAY eingetragen werden.

Wird an x_COPY_TO_RD_ARRAY FALSE angegeben, oder ist x_COPY_TO_RD_ARRAY nicht belegt, werden die mitgelesenen Istwerte nicht in das zweidimensionale Feld an a_RD_ARRAY eingetragen.

Ausgänge a_WR_VALUES_RECEIVED, si_WRC1_RECEIVED, si_WRC2_RECEIVED:

An dem Ausgang a_WR_VALUES_RECEIVED wird eine Variable vom Datentyp DINT_8_BMARRAY angeschlossen. Der Datentyp DINT_8_BMARRAY ist ein Feld mit 8 Einträgen des Datentyps Double Integer:

```
DINT_8_BMARRAY                  : ARRAY [0..7] OF DINT;
```

Beispiel:

```
a_Sollwerte : DINT_8_BMARRAY;
```

dabei ist:

`a_Sollwerte`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`DINT_8_BMARRAY`	der Datentyp

Die vom CANsync-Master empfangenen Sollwerte 0 bis 7 werden dann z. B. in die Feld-Elemente `a_Sollwerte[0]` bis `a_Sollwerte[7]` eingetragen. Die Sollwerte können vom Format Wort oder Doppelwort sein.

Am Ausgang si_WRC1_RECEIVED wird mit 2 angezeigt, ob das Sollwert-Telegramm 1 empfangen wurde. Wenn das Sollwert-Telegramm 1 nicht empfangen wurde, ist si_WRC1_RECEIVED = 0. Nur wenn das Sollwert-Telegramm 1 empfangen wurde, werden die neuen Sollwerte 0 bis 3 an a_WR_VALUES_RECEIVED ausgegeben.

Am Ausgang si_WRC2_RECEIVED wird mit 2 angezeigt, ob das Sollwert-Telegramm 2 empfangen wurde. Wenn das Sollwert-Telegramm 2 nicht empfangen wurde, ist si_WRC2_RECEIVED = 0. Nur wenn das Sollwert-Telegramm 2 empfangen wurde, werden die neuen Sollwerte 4 bis 7 an a_WR_VALUES_RECEIVED ausgegeben.

HINWEIS

Im Sollwert-Telegramm 1 werden die Sollwerte 0 bis 3 empfangen, im Sollwert-Telegramm 2 werden die Sollwerte 4 bis 7 empfangen. Lücken bei den Sollwertnummern sind zulässig.

Ausgänge si_RD_SL_NR1_RECEIVED, si_RD_SL_NR2_RECEIVED:

Am Ausgang si_RD_SL_NR1_RECEIVED wird die Slavenummer des CANsync-Slaves angezeigt von dem im letzten CANsync-Intervall das Istwert-Telegramm 1 mitgelesen wurde. Wenn in einem CANsync-Intervall kein Istwert-Telegramm 1 mitgelesen wurde, wird -128 an si_RD_SL_NR1_RECEIVED angezeigt.

Am Ausgang si_RD_SL_NR2_RECEIVED wird die Slavenummer des CANsync-Slaves angezeigt von dem im letzten CANsync-Intervall das Istwert-Telegramm 2 mitgelesen wurde. Wenn in einem CANsync-Intervall kein Istwert-Telegramm 2 mitgelesen wurde, wird -128 an si_RD_SL_NR2_RECEIVED angezeigt.

7.3.23 CANsync_SL_TYP_INIT

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um die Slave-Typen für die CANsync-Initialisierung anzugeben.

HINWEIS

Dieser FB wird zusammen mit dem FB CANsync_INIT eingesetzt.

Der FB CANsync_SL_TYP_INIT verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
us_SL_TYP0	USINT	Slave-Typ 0
us_SL_TYP1	USINT	Slave-Typ 1
us_SL_TYP2	USINT	Slave-Typ 2
...	...	...
us_SL_TYP31	USINT	Slave-Typ 31

Parameter Ausgang	Datentyp	Beschreibung
a_SL_TYP	BYTE_32_BMARRAY	Initialisierungsdaten "Slave-Typen"

Mit diesem FB werden die Slave-Typen der CANsync-Slaves am CANsync-Bus angegeben. Die Daten werden in ein Feld (Ausgang a_SL_TYP) eingetragen. Dieses Feld wird am Eingang a_SL_TYP des FB CANsync_INIT angeschlossen.

Eingänge us_SL_TYP0 bis us_SL_TYP31:

Für den CANsync-Slave mit der Slavenummer 0 wird der Slave-Typ am Eingang us_SL_TYP0 angegeben.

us_SL_TYP0 = 0 bedeutet kein CANsync-Slave mit der Slavenummer 0 vorhanden

us_SL_TYP0 = 1 bedeutet CANsync-Slave mit der Slavenummer 0 vorhanden

Für den CANsync-Slave mit der Slavenummer 1 wird der Slave-Typ am Eingang us_SL_TYP1 angegeben.

us_SL_TYP1 = 0 bedeutet kein CANsync-Slave mit der Slavenummer 1 vorhanden

us_SL_TYP1 = 1 bedeutet CANsync-Slave mit der Slavenummer 1 vorhanden

usw...

Für den CANsync-Slave mit der Slavenummer 31 wird der Slave-Typ am Eingang us_SL_TYP31 angegeben.

us_SL_TYP31 = 0 bedeutet kein CANsync-Slave mit der Slavenummer 31 vorhanden

us_SL_TYP31 = 1 bedeutet CANsync-Slave mit der Slavenummer 31 vorhanden

Bedeutung der Slave-Typen:

Slave-Typ	Bedeutung
0	kein CANsync-Slave mit der Slavenummer x
1	CANsync-Slave-Anschaltung eines Ωmega Drive-Line II mit der Slavenummer x, V-Regler mit CANsync-Interface mit der Slavenummer x vorhanden
2 - 255	reserviert

Ausgang a_SL_TYP:

Am Ausgang a_SL_TYP wird eine Variable vom Datentyp BYTE_32_BMARRAY angeschlossen. Der Datentyp BYTE_32_BMARRAY ist ein Feld von 32 Einträgen des Datentyps Byte:

```
BYTE_32_BMARRAY                 : ARRAY [0..31] OF BYTE;
```

Beispiel:

```
a_Slave_Typen : BYTE_32_BMARRAY;
```

dabei ist:

`a_Slave_Typen`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`BYTE_32_BMARRAY`	der Datentyp

In den einzelnen Einträgen des Feldes steht der Slave-Typ der CANsync-Slaves am CANsync-Bus. Im Eintrag [0] steht der Slave-Typ des CANsync-Slaves mit der Slavenummer 0, im Eintrag [1] steht der Slave-Typ des CANsync-Slaves mit der Slavenummer 1, usw...

Eine 0 im Eintrag [x] bedeutet, daß kein CANsync-Slave mit der Slavenummer x am CANsync-Bus vorhanden ist.

Ein Wert ≠ 0 im Eintrag [x] bedeutet, daß ein CANsync-Slave mit der Slavenummer x am CANsync-Bus vorhanden ist.

HINWEIS

Diese Variable wird am Eingang a_SL_TYP des FB CANsync_INIT angeschlossen.

7.3.24 CANsync_UPDOWNLOAD_MA

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um einen Upload oder einen Download im Block 1 - Bereich (siehe “Up/Download Blockbereich” auf Seite 166) durchzuführen. Er ist speziell für den Einsatz mit dem FB CANsync_UPDOWNLOAD_SL im CANsync-Slave konzipiert.

HINWEIS

Für den Einsatz dieses FBs ist es notwendig, für den Kommandokanal (CC) die Steuerwortkommandos freizugeben. Diese Freigabe erfolgt mit dem FB CANsync_COMM_CONTROL_MA.

Der FB CANsync_UPDOWNLOAD_MA verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
a_DOWNLOAD	CANsync_UPDOWN_BMARRAY	Downloadwerte
a_UPLOAD	CANsync_UPDOWN_BMARRAY	Uploadwerte
si_SL_NR	SINT 0 bis 31	Slavenummer des CANsync-Slaves, an den der Download- oder Upload-Auftrag gerichtet ist
x_UPDOWN	BOOL	Up- oder Download
u_LENGTH	UINT 1 bis 2048	Länge des Updownblocks
us_MAX_NR_OF_COMM	USINT	max. Anzahl der Kommunikations-versuche
t_TIME	TIME	Überwachungszeit
x_EN	BOOL	Freigabe
x_RESET	BOOL	Reset

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_MA_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
a_DOWNLOAD	CANsync_UPDOWN_BMARRAY	Downloadwerte
a_UPLOAD	CANsync_UPDOWN_BMARRAY	Uploadwerte
x_BUSY	BOOL	Kommunikation ist aktiv
w_ERR_SL	WORD	Fehlerwort (CANsync-Slave -> CANsync-Master)
x_ERR_SL	BOOL	Fehlerbit (Sammel-Fehlerbit von w_ERR_SL)
b_ERR	BYTE	Fehlerbyte

Parameter Ausgang	Datentyp	Beschreibung
x_ERR	BOOL	Fehlerbit (Sammel-Fehlerbit von b_ERR)
x_OK	BOOL	OK-Bit

Der FB CANsync_UPDOWNLOAD_MA führt einen Upload oder einen Download im Block 1 - Bereich durch. Der FB CANsync_UPDOWNLOAD_MA ist speziell für den Einsatz mit dem FB CANsync_UPDOWNLOAD_SL im CANsync-Slave konzipiert.

Der FB CANsync_UPDOWN_MA übergibt Download-Daten an den CANsync-Slave mit der Slavenummer si_SL_NR.

Der Download-Auftrag besteht aus u_LENGTH Werten aus dem Feld a_DOWNLOAD.

(Der FB CANsync_UPDOWN_MA führt einen Download-Auftrag an den CANsync-Slave in mehreren Download-Telegramm-Blöcken aus. Je Download-Telegramm-Block werden maximal 75 Werte aus dem Feld a_DOWNLOAD an den CANsync-Slave gesendet.)

Der CANsync-Slave bearbeitet den Download-Auftrag und gibt das Ergebnis der Kommunikation zurück.

Der FB CANsync_UPDOWN_MA fordert Upload-Daten von dem CANsync-Slave mit der Slavenummer si_SL_NR an.

Es werden u_LENGTH Werte vom CANsync-Slave angefordert.

(Der FB CANsync_UPDOWN_MA führt einen Upload-Auftrag an den CANsync-Slave in mehreren Upload-Telegramm-Blöcken aus. Je Upload-Telegramm-Block werden maximal 75 Werte vom CANsync-Slave empfangen.)

Der CANsync-Slave bearbeitet den Upload-Auftrag und gibt das Ergebnis der Kommunikation zurück. Die Werte werden an a_UPLOAD ausgegeben.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_MA_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 2 (Node 2) auf Ωmega Drive-Line II

```
_CANsync_CTRL_MA  AT  %MB3.200000  :  CANsync_MA_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_MA`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_MA_CTRL_BMSTRUCT`	der Datentyp
`%MB3.200000`	die Basisadresse der CANsync-Anschaltung 2 auf Ωmega Drive-Line II

Ein-/Ausgang a_DOWNLOAD:

An a_DOWNLOAD wird eine Variable vom Datentyp CANsync_UPDOWN_BMARRAY angeschlossen. Der Datentyp CANsync_UPDOWN_BMARRAY ist ein Feld von 2048 Einträgen des Datentyps Double Integer.

```
CANsync_UPDOWN_BMARRAY          : ARRAY [0..2047] OF DINT;
```

Beispiel:

```
a_Downloadwerte : CANsync_UPDOWN_BMARRAY;
```

dabei ist:

`a_Downloadwerte`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`CANsync_UPDOWN_BMARRAY`	der Datentyp

Die Daten für den Download werden dann z. B. in den Feld-Elementen `a_Downloadwerte[0]` bis `a_Downloadwerte[2047]` erwartet.

Ein-/Ausgang a_UPLOAD:

An a_DOWNLOAD wird eine Variable vom Datentyp CANsync_UPDOWN_BMARRAY angeschlossen. Der Datentyp CANsync_UPDOWN_BMARRAY ist ein Feld von 2048 Einträgen des Datentyps Double Integer.

```
CANsync_UPDOWN_BMARRAY          : ARRAY [0..2047] OF DINT;
```

Beispiel:

```
a_Uploadwerte : CANsync_UPDOWN_BMARRAY;
```

dabei ist:

`a_Uploadwerte`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`CANsync_UPDOWN_BMARRAY`	der Datentyp

Beim Upload werden die Daten dann z. B. in den Feld-Elementen `a_Uploadwerte[0]` bis `a_Uploadwerte[2047]` ausgegeben.

Eingang si_SL_NR:

Am Eingang si_SL_NR wird die Slavenummer des CANsync-Slaves am CANsync-Bus angegeben, von dem ein Upload oder zu dem ein Download durchgeführt werden soll.

HINWEIS

Der Up-/Download zu diesem CANsync-Slave muß über den FB CANsync_COMM_CONTROL_MA freigegeben werden (siehe Beschreibung FB CANsync_COMM_CONTROL_MA).

Eingang x_UPDOWN:

Am Eingang x_UPDOWN wird mit x_UPDOWN = FALSE ein Upload-Auftrag und mit x_UPDOWN = TRUE ein Download-Auftrag eingestellt.

Eingang u_LENGTH:

Am Eingang u_LENGTH wird die Länge des zu übertragenden Up-/Download-Bereiches angegeben. Es werden maximal 2048 Werte á 32 Bit übertragen. Wenn an u_LENTGH eine 0 angegeben ist, wird im Fehlerbyte b_ERR das Bit1 auf TRUE gesetzt.

Die Daten werden Blockweise in Telegramm-Blöcken übertragen. Ein Telegramm-Block ist 75 Werte á 32 Bit groß. D. h. bei einem Download- oder Upload-Auftrag über mehr als 75 Werte á 32 Bit werden entsprechend mehrere Telegramm-Blöcke gesendet.

Eingang us_MAX_NR_OF_COMM:

Am Eingang us_MAX_NR_OF_COMM kann angegeben werden, wie oft ein Block des Up-Download-Bereiches wiederholt werden soll, wenn a) die Überwachungszeit (des CANsync-Masters) abgelaufen und b) vom CANsync-Slave kein Fehler gemeldet wurde (Voreinstellung ist us_MAX_NR_OF_COMM = 1). Die Timeout-Meldung (siehe Eingang t_TIME) erfolgt dann erst, nachdem die Zeit us_MAX_NR_OF_COMM * t_TIME abgelaufen ist und bis dahin keine Antwort vom CANsync-Slave vorliegt.

Eingang t_TIME:

Die Überwachungszeit wird am Eingang t_TIME angegeben. Wird der Download-Auftrag oder der Upload-Auftrag nicht innerhalb der Überwachungszeit vollständig abgearbeitet, wird im Fehlerbyte das Bit 2 gesetzt (siehe auch Eingang us_MAX_NR_OF_COMM).

Ursache für das nicht vollständige Abarbeiten eines Download- oder Upload-Auftrags kann die Belegung des Kommandokanals mit höherprioren Nachrichten sein (Broadcast-Kommandos, Steuerwort-Senden-Aufträge, Parameter-Aufträge; siehe “Bedarfsdaten” auf Seite 130).

Eingang x_EN:

Mit x_EN = TRUE wird der FB CANsync_UPDOWNLOAD_MA freigegeben d. h. Download- oder Upload-Aufträge können an den CANsync-Slave gesendet werden.

Falls x_EN auf FALSE gesetzt wird, bevor der Download- oder Upload-Auftrag abgeschlossen ist, wird von einem bewußten Abbruch des Auftrags ausgegangen. Der FB CANsync_UPDOWNLOAD_MA muß anschließend mit x_RESET = TRUE zurückgesetzt werden, um einen neuen Download- oder Upload-Auftrag über die angegebene Länge (Eingang u_LENGTH) zu starten.

Eingang x_RESET:

Mit x_RESET glecih TRUE kann der FB CANsync_UPDOWNLOAD_MA zurückgesetzt werden. Dies ist z. B. nach einem Abbruch des Download- oder Upload-Auftrags (durch x_EN = FALSE) oder nach einer Fehlermeldung notwendig. Anschließend muß x_RESET wieder auf FALSE gesetzt werden.

Ausgang x_BUSY:

Der Ausgang x_BUSY zeigt mit TRUE an, daß der FB CANsync_UPDOWNLOAD_MA einen Auftrag abarbeitet.

Ausgang x_OK:

Der Ausgang x_OK zeigt mit TRUE die erfolgreiche Ausführung des Download- oder Upload-Auftrags an. Der Ausgang x_OK ist FALSE wenn kein Download- oder Upload-Auftrag ausgeführt oder ein Auftrag nicht korrekt ausgeführt wurde.

Ausgänge x_ERR_SL, w_ERR_SL:

Falls bei der Ausführung des Download- oder Upload-Auftrags im CANsync-Slave ein Fehler auftritt, wird das Fehlerbit x_ERR_SL auf TRUE gesetzt und das Fehlerwort w_ERR_SL ausgegeben (siehe unten). Der Ausgang x_OK bleibt dann FALSE.

Ausgänge x_ERR, b_ERR:

Falls bei der Ausführung des Download- oder Upload-Auftrags ein Fehler auftritt, wird das Fehlerbit x_ERR auf TRUE gesetzt und das Fehlerbyte b_ERR ausgegeben (siehe unten). Der Ausgang x_OK bleibt dann FALSE.

Fehlerbyte b_ERR:

Bit-Nr.	Bedeutung
0	Timeout
1	Länge des zu übertragenden Up-/Download-Bereiches gleich 0 (Eingang u_LENGTH)
2 bis 7	Reserviert

Fehlerwort w_ERR_SL:

w_ERR_SL	Bedeutung
16#0000	Reserviert
16#0001	CANsync-Slave quittiert falsche Blocknummer
16#0002	eingetragene Übertragungsblock-Länge > 300 Byte / 75 Doppelworte
16#0003 bis 16#00FF	Reserviert
16#0100	CANsync-Slave erwartet Übertragungsblock mit der Nummer, die im Zähler eingetragen ist
16#0101	CANsync-Slave erwartet Übertragungsblock-Ende
16#0102	CANsync-Slave erwartet noch nicht Übertragungsblock-Ende
16#0103	CANsync-Slave bricht Auftrag ab
16#0104	Auftrag nicht möglich
16#0105	Basisadresse nicht erlaubt
16#0106	Reserviert
16#0107	Up-/Download-Bereich CANsync-Master > Up-/Download-Bereich CANsync-Slave
16#0108	Telegramm-Modefehler (in dieser Phase nicht erlaubter Mode)
16#0109 bis 16#FFFF	Reserviert

7.3.25 CANsync_UPDOWNLOAD_SL

Beschreibung

Diesen Funktionsbaustein für CANsync können Sie verwenden, um je nach Auftrag vom CANsync-Master einen Upload oder einen Download im Block 1 - Bereich (siehe “Up/Download-Kommando” auf Seite 190) durchzuführen. Er ist speziell für den Einsatz mit dem FB CANsync_UPDOWNLOAD_MA im CANsync-Master konzipiert.

HINWEIS

Der FB CANsync_UPDOWNLOAD_SL verwendet die Bibliothek BM_TYPES_20bd00 oder höher.

Parameter Eingang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
a_DOWNLOAD	CANsync_UPDOWN_BMARRAY	Downloadwerte
a_UPLOAD	CANsync_UPDOWN_BMARRAY	Uploadwerte
t_TIME	TIME	Überwachungszeit
x_EN	BOOL	Freigabe
x_RESET	BOOL	Reset

Parameter Ausgang	Datentyp	Beschreibung
_BASE	CANsync_SL_CTRL_BMSTRUCT	Betriebsdaten für die CANsync-Anschaltung
a_DOWNLOAD	CANsync_UPDOWN_BMARRAY	Downloadwerte
a_UPLOAD	CANsync_UPDOWN_BMARRAY	Uploadwerte
x_UPDOWN	BOOL	Anzeige Up- oder Download
x_ORDER_ACTIV	BOOL	Anzeige Auftrag steht an
u_LENGTH	UINT	Länge des übertragenen Blocks
x_BUSY	BOOL	Anzeige FB ist aktiv
w_ERR	WORD	Fehlerwort
x_ERR	BOOL	Fehlerbit
x_OK	BOOL	OK-Bit

Der FB CANsync_UPDOWNLOAD_SL führt einen Download-Auftrag oder einen Upload-Auftrag des CANsync-Masters aus. Downloadwerte (a_DOWNLOAD) werden ausgegeben, Uploadwerte (a_UPLOAD) werden an den CANsync-Master gesendet.

Die Daten werden in mehreren Telegramm-Blöcken übertragen. Je Telegramm-Block werden maximal 75 Werte á 32 Bit übertragen.

Der FB CANsync_UPDOWNLOAD_SL ist speziell für den Einsatz mit dem FB CANsync_UPDOWNLOAD_MA im CANsync-Master konzipiert.

Ein-/Ausgang _BASE:

An _BASE muß eine globale Variable vom Datentyp CANsync_SL_CTRL_BMSTRUCT angeschlossen werden. Diese Variable muß über die Deklaration der globalen Variablen auf die Basisadresse der CANsync-Anschaltung gelegt werden.

Beispiel:

CANsync-Anschaltung 1 (Node 1) auf Ωmega Drive-Line II

```
_CANsync_CTRL_SL  AT  %MB3.100000  :  CANsync_SL_CTRL_BMSTRUCT;
```

dabei ist:

`_CANsync_CTRL_SL`	der Variablenname mit der Datentypkurzbezeichnung "_" für Struct
`CANsync_SL_CTRL_BMSTRUCT`	der Datentyp
`%MB3.100000`	die Basisadresse der CANsync-Anschaltung 1 auf Ωmega Drive-Line II

Ein-/Ausgang a_DOWNLOAD:

An a_DOWNLOAD wird eine Variable vom Datentyp CANsync_UPDOWN_BMARRAY angeschlossen. Der Datentyp CANsync_UPDOWN_BMARRAY ist ein Feld von 2048 Einträgen des Datentyps Double Integer.

```
CANsync_UPDOWN_BMARRAY          : ARRAY [0..2047] OF DINT;
```

Beispiel:

```
a_Downloadwerte : CANsync_UPDOWN_BMARRAY;
```

dabei ist:

`a_Downloadwerte`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`CANsync_UPDOWN_BMARRAY`	der Datentyp

Beim Download werden die Daten dann z. B. in den Feld-Elementen `a_Downloadwerte[0]` bis `a_Downloadwerte[2047]` ausgegeben.

Ein-/Ausgang a_UPLOAD:

An a_DOWNLOAD wird eine Variable vom Datentyp CANsync_UPDOWN_BMARRAY angeschlossen. Der Datentyp CANsync_UPDOWN_BMARRAY ist ein Feld von 2048 Einträgen des Datentyps Double Integer.

```
CANsync_UPDOWN_BMARRAY          : ARRAY [0..2047] OF DINT;
```

Beispiel:

```
a_Uploadwerte : CANsync_UPDOWN_BMARRAY;
```

dabei ist:

`a_Uploadwerte`	der Variablenname mit der Datentypkurzbezeichnung "a" für Array
`CANsync_UPDOWN_BMARRAY`	der Datentyp

Die Daten für den Upload werden dann z. B. in den Feld-Elementen `a_Uploadwerte[0]` bis `a_Uploadwerte[2047]` erwartet.

Eingang t_TIME:

Die Überwachungszeit wird am Eingang t_TIME angegeben. Wird der Download-Auftrag oder der Upload-Auftrag nicht innerhalb der Überwachungszeit vollständig abgearbeitet, wird im Fehlerwort das Bit 0 gesetzt. Wird der Eingang t_TIME nicht belegt, ergibt sich eine Voreinstellung der Überwachungszeit von 30 s.

Ursache für das nicht vollständige Abarbeiten eines Download- oder Upload-Auftrags kann die Belegung des Kommandokanals mit höherprioren Nachrichten sein (Broadcast-Kommandos, Steuerwort-Senden-Aufträge, Parameter-Aufträge; siehe “Bedarfsdaten” auf Seite 130).

Eingang x_EN:

Mit x_EN = TRUE wird der FB CANsync_UPDOWNLOAD_SL freigegeben, d. h. Download- oder Upload-Aufträge können bearbeitet werden.

Falls x_EN auf FALSE gesetzt wird, bevor der Download- oder Upload-Auftrag abgeschlossen ist, wird von einem bewußten Abbruch ausgegangen. Der FB CANsync_UPDOWNLOAD_SL muß anschließend mit x_RESET = TRUE zurückgesetzt werden, um einen neuen Download- oder Upload-Auftrag ausführen zu können.

Eingang x_RESET:

Mit x_RESET = TRUE kann der FB CANsync_UPDOWNLOAD_SL zurückgesetzt werden. Dies ist z. B. nach einem Abbruch des Download- oder Upload-Auftrags (durch x_EN = FALSE) oder nach einer Fehlermeldung notwendig. Anschließend muß x_RESET wieder auf FALSE gesetzt werden.

Ausgänge x_UPDOWN, x_ORDER_ACTIV:

Der Ausgang x_ORDER_ACTIV zeigt mit TRUE an, daß ein Auftrag vorliegt. Die Art des Auftrags wird durch den Ausgang x_UPDOWN angezeigt. Bei einem Upload-Auftrag ist x_UPDOWN = FALSE, bei einem Download-Auftrag ist x_UPDOWN = TRUE.

Ausgang u_LENGTH:

Der Ausgang u_LENGTH zeigt die Anzahl der Werte á 32 Bit des übertragenen Telegramm-Blocks an. Ein Telegramm-Block ist maximal 75 Werte á 32 Bit groß. Bei einem Download- oder Upload-Auftrag über mehr als 75 Werte á 32 Bit werden entsprechend mehrere Telegramm-Blöcke übertragen und an u_LENGTH die Anzahl der Werte á 32 Bit des aktuell übertragenen Telegramm-Blocks angezeigt.

Ausgang x_BUSY:

Der Ausgang x_BUSY zeigt mit TRUE an, daß der FB CANsync_UPDOWNLOAD_SL einen Download- oder Upload-Auftrag bearbeitet.

Ausgang x_OK:

Der Ausgang x_OK zeigt mit TRUE die erfolgreiche Ausführung des Download- oder Upload-Auftrags an. Der Ausgang x_OK ist FALSE wenn kein Download- oder Upload-Auftrag ausgeführt wurde oder ein Auftrag nicht korrekt ausgeführt wurde.

Der Ausgänge x_ERR, w_ERR:

Falls ein Fehler auftritt, wird das Fehlerbit x_ERR auf TRUE gesetzt und das Fehlerwort w_ERR ausgegeben. Der Ausgang x_OK bleibt dann FALSE.

Fehlerwort w_ERR:

Bit-Nr.	Bedeutung
0	Timeout
1 bis 15	reserviert

8 INDEX

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z

be in motion

Baumüller Nürnberg GmbH Ostendstraße 80-90 90482 Nürnberg T: +49(0)911-5432-0 F: +49(0)911-5432-130 **www.baumueller.de**