@PLC-C20x (EN / PB)

Betriebsanleitung

Ausgabe-/Rev.-Datum:	01.09.2005
Dokument-Rev.-Nr.:	01
Softstand Firmware:	Version 3.0.0
Softstand CoDeSys:	Version 2.3.4.7
Softstand CoDeSys RT:	Version 2.3
Dateiname:	@PLCC20xxx-TRS-V-BA-D-0000-01.doc
Artikel-Nr.:	TRS-DOC-000012
Verfasser:	KAO / HIE

TR-Systemtechnik GmbH
Eglishalde 16
D-78647 Trossingen

Tel. 07425 / 228-0
Fax 07425 / 228-34

Impressum

TR-Systemtechnik GmbH
D-78647 Trossingen
Eglishalde 16
Tel.: (+49) 07425/228-0
Fax: (+49) 07425/228-34
info@tr-systemtechnik.de
http://www.tr-systemtechnik.de/

Änderungsvorbehalt

Änderungen der in diesem Dokument enthaltenen Informationen, die aus unserem stetigen Bestreben zur Verbesserung unserer Produkte resultieren, behalten wir uns jederzeit vor.

Dieses Dokument beschreibt die angegebenen Produkte und Funktionen in Art und Eigenschaft zum Zeitpunkt der Erstellung. Es besteht keine Gewähr auf Vollständigkeit. Ein Rechtsanspruch aufgrund nicht aufgeführter oder anders lautender Informationen ist ausgeschlossen.

Druck

Dieses Handbuch wurde mit MS-WORD für Windows auf einem Personal-Computer erstellt. Der Text wurde in *Arial* gedruckt.

Schreibweisen

Kursive oder **fette** Schreibweise steht für den Titel eines Dokuments oder wird zur Hervorhebung benutzt.

`Courier`-Schrift zeigt Text an, der auf dem Bildschirm / Display sichtbar ist und Menü auswählen von Software.

″ < > ″ weist auf Tasten der Tastatur Ihres Computers hin (wie etwa <RETURN>).

Hinweis

Meldungen, die nach dem Symbol "HINWEIS" erscheinen, markieren wichtige Merkmale des verwendeten Produkts.

Hinweise zu Urheberrechten (Copyright ©)

MS-WORD ist ein eingetragenes Warenzeichen der Microsoft AG.

CoDeSys ist ein eingetragenes Warenzeichen der 3S – Smart Software Solutions GmbH

Literatur + Link

siehe Anhang F.

Ausschluß

Das vorliegende Dokument beschreibt die angegebenen Produkte und Funktionen in Art und Eigenschaft zum Zeitpunkt der Erstellung. Es besteht keine Gewähr auf Vollständigkeit. Ein Rechtsanspruch, aufgrund nicht aufgeführten oder anders lautenden Informationen, ist ausgeschlossen.

Änderungs-Index Dokument

Hinweis

Auf dem Deckblatt dieses Dokumentes ist der aktuelle Revisionsstand mit dem dazugehörigen Datum vermerkt. Da jedes einzelne Blatt in der Fusszeile mit einem eigenen Revisionsstand und Datum versehen ist, kann es vorkommen, dass sich unterschiedliche Revisionsstände innerhalb des Dokumentes ergeben.

Zeichnungen, die sich im Anhang befinden können, sind mit einem eigenen Änderungs-Index versehen.

Das Dokument wurde für die auf der Titelseite angegebene Kombination aus **Firmware**- und **CoDeSys-Version** erstellt. Alle Informationen beziehen sich auf diese Versionskombination.

Des Weiteren sei an dieser Stelle angemerkt, dass nur die auf der Titelseite angegebene CoDeSys-Version 2.3.4.7 für die Verwendung mit der @PLC-C20x freigegeben ist. Bei Verwendung einer anderen CoDeSys-Version, als die angegebene Version, kann keine Gewähr auf die korrekte Funktion der Firmware gegeben werden.

Dokumenterstellung: 01.09.2005

	Änderung	Datum
00	Beschreibung für Firmware-Version **2.2.0**	01.10.2004
01	Neue Firmware-Release **V3.0.0** Beschreibung Systemvariablen Kap. 3.4 und Bibliothek Kap. 4.5.1 hinzugefügt	01.09.2005

Änderungs-Index Firmware

Hinweis

Auf dem Deckblatt dieses Dokumentes ist der aktuelle Revisionsstand der Firmware mit dem dazugehörigen Datum vermerkt.

Die Firmware unterliegt einer regelmäßigen Pflege. Änderungen werden unter Benennung der neuen Release in nachfolgender Auflistung dokumentiert und bekannt gegeben.

Änderung	Datum
Firmware-Release V2.2.0	01.10.2004
Firmware-Release V3.0.0	01.09.2005

Inhaltsverzeichnis

1. Allgemeines

Modulare, dezentral verteilte Intelligenz ist der Schlüssel zu modularen, flexiblen Maschinenkonzepten, wie sie der Markt zunehmend fordert. *@PLC* ist unsere Antwort auf diese Herausforderungen. Äußerlich nicht mehr als ein Feldbusknoten leistet die *@PLC* bereits im Feld Schwerstarbeit, um Signale dezentral vorzuverarbeiten und Teilprozesse selbständig abzuwickeln. Dies entlastet die übergeordnete Steuerung, sowie den Feldbus. Anlagenteile können unabhängig voneinander aufgebaut und getestet werden.

Dank IEC-61131 geschieht dies mit einer standardisierten Programmiersprache, was weitere Durchgängigkeit in Ihrer Anlage bedeutet. Die @PLC verhält sich zur übergeordneten Steuerung wie ein Slave am Feldbus. Dadurch ist es praktisch unerheblich, mit welcher Programmiersprache Sie Ihre Steuerung programmieren. Und weil wir dieselbe Hardware einsetzen, wie für den traditionellen Feldbusknoten, stehen Ihnen auch alle E/A-Module zur Verfügung. Und das mit der vollen Modularität von *@ctiveIO.*

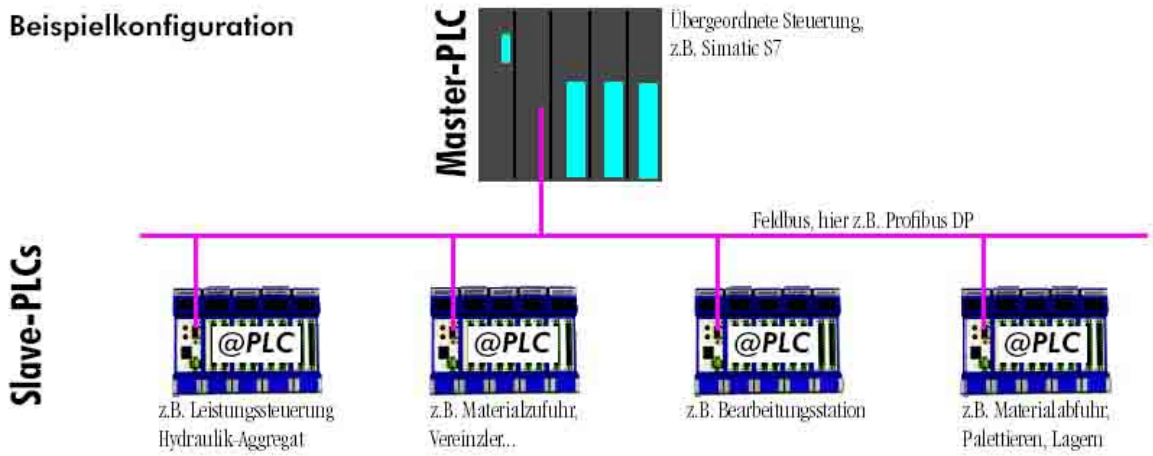

Abbildung 1: Beispielkonfiguration

Das Modul @PLC-C20x besitzt verschiedene Schnittstellen. Dies sind eine 10/100Mbit Ethernet-Schnittstelle, ein serieller Anschluss, die Spannungsversorgung und der @BUS zum Anschluss unterschiedlicher @Module.

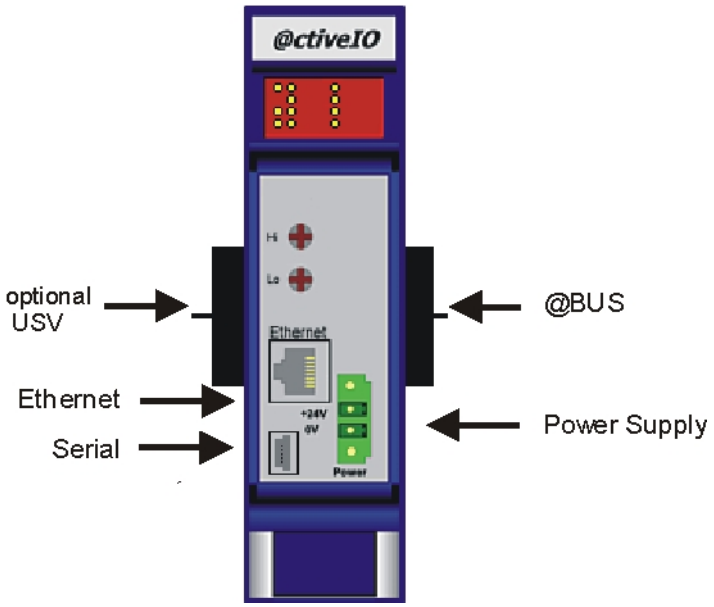

Abbildung 2: Modul @PLC-C20x-EN

@BUS

Der @BUS ist ein serieller Bus und ist auf der rechten Seite des Moduls herausgeführt. An der linken Seite des Moduls können verschieden Optionseinheiten wie z.B. eine USV angeschlossen werden. An der rechten Seite des Moduls werden die IO-Module angedockt.

Ethernet

Das Modul beinhaltet eine 10/100Mbit Fast-Ethernet Schnittstelle, die über den RJ45 Stecker realisiert ist. Die IP-Adresse kann über die @ctiveIO-Toolkit-Software eingestellt werden. Die Fast-Ethernet Schnittstelle ist die bevorzugte Schnittstelle zur Kommunikation zwischen Programmiergerät (CoDeSys) und dem IEC61131-Laufzeitsystem. Alternativ kann auch die serielle Schnittstelle verwendet werden.

Serielle Schnittstelle

Die serielle Schnittstelle (RS232, optional RS485) wird über eine Mini-USB-Kupplung geführt. Sie dient der Kommunikation zwischen Programmiergerät (CoDeSys) und dem IEC61131-Laufzeitsystem. Alternativ kann sie auch vom IEC61131-Programm bzw. zur Modbus-Kommunikation verwendet werden.

Spannungsversorgung

Für den Betrieb des Moduls wird eine 24V Gleichspannung benötigt. Diese wird an einem zweipoligen Federkraft-Stecker angeschlossen.

2. Erste Schritte

An dieser Stelle wird mit Hilfe eines Beispielprogramms die Erstellung eines IEC-Projektes für das **@PLC-C201-EN-Modul** beschrieben. Hierzu sollte der Rechner über COM-LINK2-Kabel seriell mit der @PLC-C201-EN verbunden sein.

Für das Beispielprogramm wird von der folgenden Konfiguration ausgegangen:

1. 1 St. @PLC-C201-EN (PLC-Controller C201)
2. 2 St. @P1800 (8Kanal, 8Bit digitales Eingangsmodul)
3. 2 St. @P2810 (8Kanal, 8Bit digitales Ausgangsmodul)

Abbildung 3: Aufbau

Sinn und Zweck dieses Beispielprogramms ist es dem Anwender einen ersten Überblick zu vermitteln, welche grundsätzlichen Aktionen zur Erstellung und Ausführung eines Programms durchzuführen sind.

Programmablauf:

Es wird ein Zähler implementiert, der kontinuierlich zählen soll (16Bit-Zähler). Der Zähler kann über einen Freigabe- und einen Reset-Eingang gesteuert werden. Die Ausgabe des Zählerstandes erfolgt über 2 @P2810-Prints. Der Freigabe- und der Reset-Eingang werden über 2 @P1800-Prints eingelesen.

2.1. Installation der CoDeSys Entwicklungsumgebung

Der erste Schritt ist die Installation der CoDeSys Entwicklungsumgebung. Benötigt wird die Version 2.3.4.7, wie sie auf der Produkt-CD im Verzeichnis Products\Software\CoDeSys zu finden ist. Die Verwendung einer anderen Version von CoDeSys wird nicht empfohlen, da nur die angegebene Version für die Verwendung mit der @PLC-C20x Baureihe freigegeben ist.

Die Installation gestaltet sich wie folgt: Starten Sie das Programm „setup.exe“ und folgen sie den Anweisungen des Installationsprogramms.

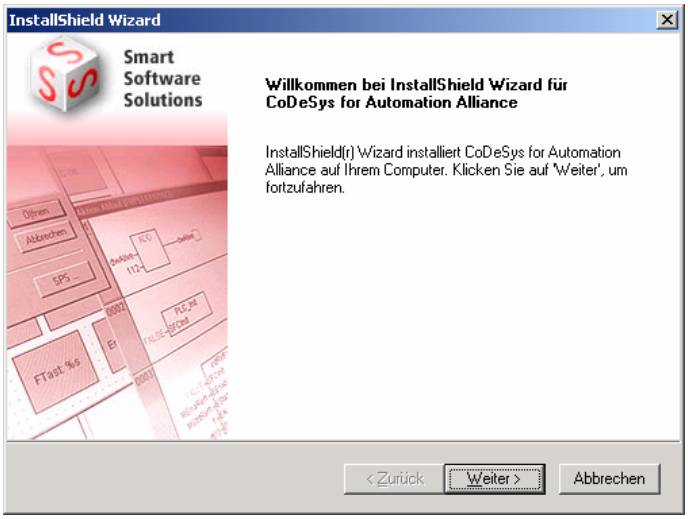

Abbildung 4: Installation CoDeSys Entwicklungsumgebung 1

Im folgenden Fenster (Abbildung 5) können Sie die zu Installierenten Komponenten auswählen. Zur Erstellung des Beispielprogramms genügt es, wenn Sie ***CoDeSys V2.3*** (IEC 61131 Programmiersystem) anwählen. Sollten Sie die anderen Anwendungen benötigen, so können Sie diese extra anwählen oder nachträglich installiert werden.

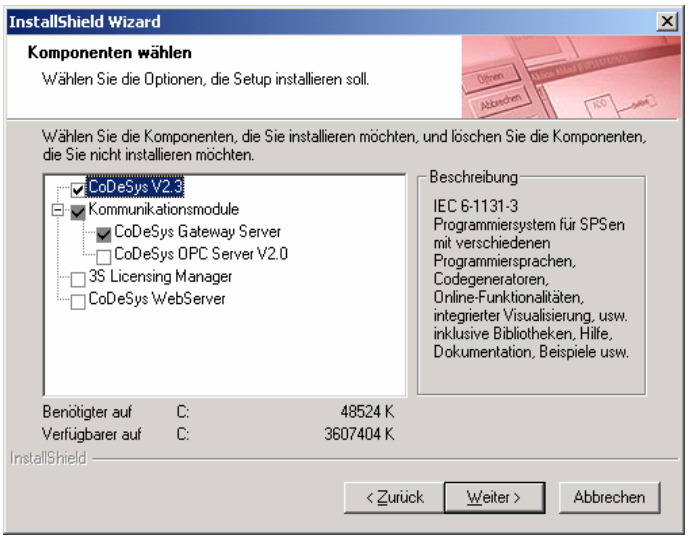

Abbildung 5: Installation CoDeSys Entwicklungsumgebung 2

Die folgenden angezeigten Fenster beinhalten Defaulteinstellungen, diese müssen nur im Einzelfall an die speziellen Bedürfnisse angepasst werden.

Somit wäre das CoDeSys-Programmiersystem V2.3 installiert und kann jetzt verwendet werden.

2.2. Inbetriebnahme der Hardware

Zunächst werden die Module und der Controller, wie in der unten stehenden Abbildung 6, zusammengefügt. Danach werden die Leitungen für die Versorgungsspannung angeschlossen (siehe Datenblätter der einzelnen Module). Vor dem Einschalten der Versorgungsspannung müssen die @Module angekoppelt sein. Nach Einschalten der Versorgungsspannung geht die RUN-LED an und die DIAG- und ERR-LED leuchten während der Initialisierungsphase auf.

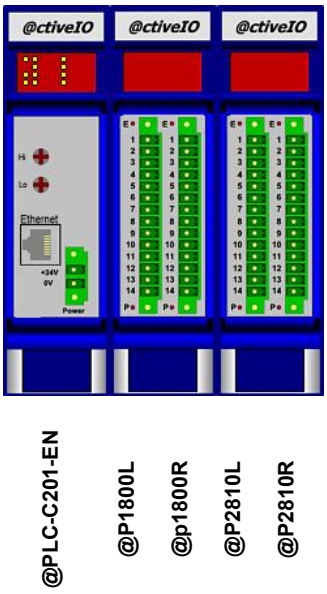

Abbildung 6 Aufbau @Controller und @Module

2.3. Installation des Target-File

Bevor mit CoDeSys ein IEC-Projekt für die @PLC-C201-EN erstellt werden kann, muss das sogenannte Target-File installiert werden. Aktuelle Target-Files finden Sie auf der TRS_Produkt-CD oder auf Anfrage:

info@tr-systemtechnik.de

Das Target-File, auch ***T***arget ***S***upport ***P***ackage (TSP) genannt, enthält die Konfigurationsdaten der anzusprechenden Steuerung. In diesem Paket sind unter anderem die Compilereinstellungen und das Speicherlayout der Steuerung zu finden, die für die Programmerstellung benötigt werden. Zusätzlich hierzu befinden sich die Konfigurationsdateien für die I/O-Konfiguration des Controllers und einige spezifische Bibliotheken in diesem Paket.

Bei der Installation werden diese Daten auf den Rechner kopiert und registriert. Nach dem Start von CoDeSys stehen sie zur Verfügung. An dieser Stelle sei angemerkt, dass einige korrespondierende Informationen zum Target-File auch im IEC-Projekt gespeichert werden. Daher sollte immer das für das verwendete Projekt benötigte Target-File installiert sein, da es sonst zu Fehlermeldungen kommen kann.

Die Installation des Target-Files kann wie folgt durchgeführt werden (Abbildung 7):

1. Beenden Sie das **CoDeSys** Programm (sofern aktiv)
2. Starten Sie das **InstallTarget** Programm
   (Start [Windows-Taskleiste] => Programme => 3S Software => CoDeSys V2.3 => InstallTarget)
3. Drücken Sie den `'Öffnen'` Button und wählen im Öffnen-Dialog-Fenster das gewünschte Target-File aus (in diesem Falle @PLC-C201-EN.tnf)
4. Markieren sie das gewünschte Zielsystems unter `'Mögliche Zielsysteme'`
5. Installieren Sie das Zielsystem durch Drücken des `'Installieren'` Button
6. Nun ist das Target-File installiert und das Programm kann mittels des `'Schließen'` Buttons beendet werden
7. Schließen und Neustart von CoDeSys stellt das neue Target zur Verfügung

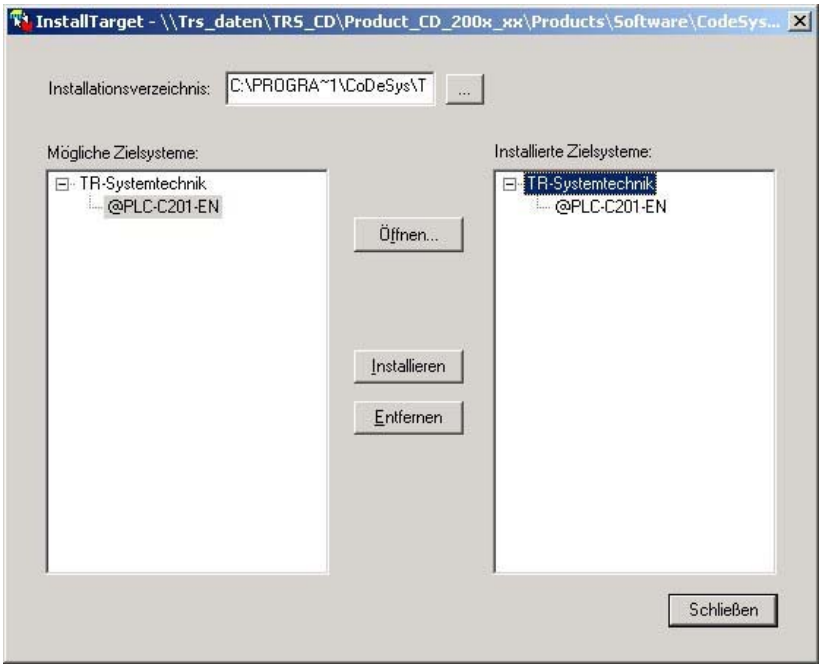

Abbildung 7: InstallTarget CoDeSys V2.3

2.4. Erstellung eines leeren Projekts

Nach dem Start von CoDeSys erscheint zunächst eine leere Arbeitsfläche. Um nun ein neues, leeres Projekt zu erstellen, wählen Sie unter dem Menüpunkt `'Datei'` den Eintrag `'Neu'` aus.

Im Anschluss daran erscheinen eine Reihe von Fenstern zur Einstellung eines Basisprojekts: Einstellung des Zielsystem und Erstellung eines Programmbausteins (siehe die nachfolgenden Kapitel).

2.5. Zielsystem-Einstellung

Es erscheint zunächst ein Fenster (Abbildung 8), in dem Sie die Konfiguration Ihres Zielsystem auswählen können --- in diesem Fall wählen Sie @PLC-C201-EN aus. Nach der Auswahl verändert sich das Fenster und es erscheinen weitere Einstellungsmöglichkeiten (siehe Abbildung). Für unser kleines Beispielprojekt sind an dieser Stelle jedoch keine Veränderungen an den Defaulteinstellungen notwendig. Die Defaulteinstellungen sollten für die meisten Fälle sinnvolle Werte enthalten. Auf spezielle Konfigurationen, in denen Änderungen notwendig sind, wird in einem späteren Kapitel eingegangen.

Ablauf:

- Öffnen des Menüs `'Datei'`,
- Auswahl des Menüeintrag `'Neu'`,
- Mit `'OK'` bestätigen

Einstellungen: „@PLC-C201-EN“

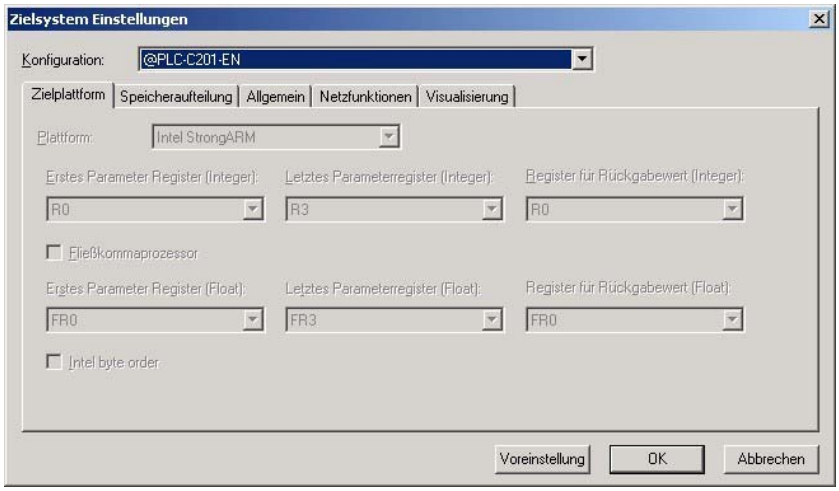

Abbildung 8: Zielsystemeinstellung CoDeSys V2.3

2.6. Erstellung eines Programmbausteins

Anschließend werden Sie aufgefordert, einen neuen Baustein anzulegen. Dieser Baustein stellt zunächst das „Hauptprogramm“ dar. Für unser kleines Beispielprojekt können Sie die Defaulteinstellungen (siehe Abbildung 9) unverändert lassen. Als Name des Bausteins wird defaultmäßig „PLC_PRG“ verwendet. Dieser Name kann verändert werden. In diesem Fall muss dann allerdings die Taskkonfiguration erstellt werden (siehe Abschnitt 2.9).

Dieser Baustein wird später den Programmcode unseres Beispielprojekts enthalten.

Mit der Erstellung eines Programmbausteins haben wir das „Pflicht-Programm“ hinter uns gebracht. Jetzt können wir den Feinschliff des Projekts vornehmen.

Abbildung 9: Neuer Baustein CoDeSys V2.3

2.7. Erstellung der Steuerungskonfiguration

Der nächste wichtige Schritt ist die Erstellung der Steuerungskonfiguration. Die dient dazu, das Prozessabbild mit der angeschlossenen Hardware zu verbinden. Die Hardware können wie in diesem Fall @ctiveIO-Module oder auch die unterstützten Feldbus-Komponenten sein.

Die Steuerungskonfiguration befindet sich im Objekt Organizer in der Registerkarte **Ressourcen**. Sie bietet einen Konfigurationseditor, mit dem die Ziel-Hardware beschrieben werden kann, auf der das geöffnete Projekt laufen soll. Wichtig ist hier, dass die eingestellte Hardware-Konfiguration exakt der angeschlossenen Konfiguration entspricht. Sind diese unterschiedlich, findet kein Austausch der Daten statt (kein Lesen von Eingängen bzw. Schreiben von Ausgängen). Der Anwender erhält in einem solchen Fall eine Fehlermeldung (Abbildung 10) angezeigt, sobald das Programm auf die Steuerung übertragen wird.

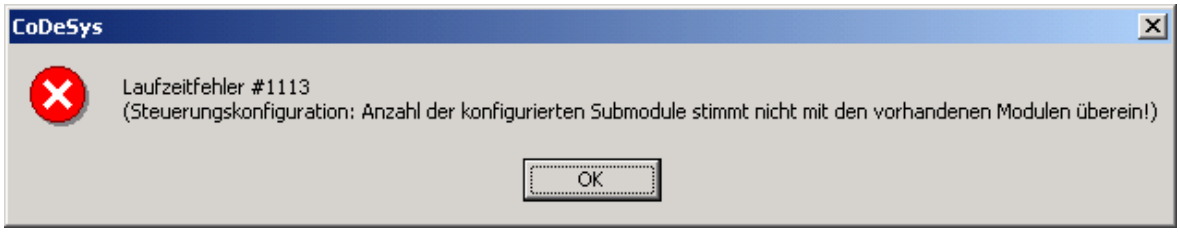

Abbildung 10: Fehlermeldung CoDeSys falsche Steuerungskonfiguration

Für das Erstellen des Beispielprogramms wird eine **Variable Konfiguration** (Abbildung 11) gewählt. Bei der Variablen Konfiguration werden die verwendeten Module einzeln entsprechend der angeschlossenen Hardware in die Konfiguration eingefügt.

Nun müssen die eingesetzten Module, in diesem Beispiel @P1800L, @P1800R, @P2810L und @P2810R, in der Steuerungskonfiguration eingetragen werden.

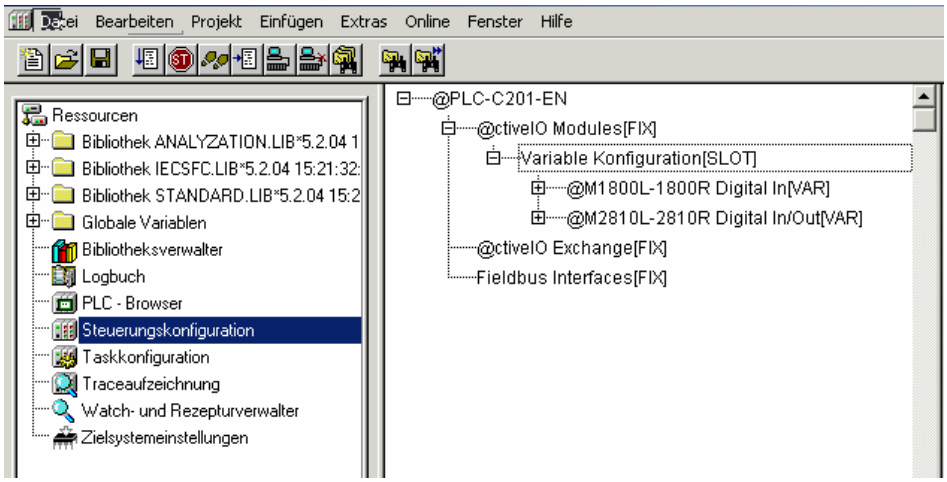

Abbildung 11: CoDeSys Steuerungskonfiguration

Hierzu markieren Sie ’Variable Konfiguration’ und öffnen mit der rechten Maustaste ein Kontextmenü. Mit der Auswahl ’Unterelemente anhängen’ können nun jeweiligen die Module der Steuerungskonfiguration zugewiesen werden. Zum Anhängen von neuer Module (Abbildung 12 links) muss das Element ’Variable Konfiguration[SLOT]’ markiert werden.

Abbildung 12: CoDeSys Steuerungskonfiguration

Soll ein Modul vor dem anderen eingefügt werden, so muss das Modul (Abbildung 12 rechts), vor dem das andere eingefügt werden soll, markiert werden.

Einstellungen: 1.Steckplatz: - @M1800L-1800R Digital In
2.Steckplatz: - @M2810L-2810R Digital In/Out

Standardmäßig (Abbildung 14) werden die Adressen für die Module in steigender Reihenfolge automatisch gesetzt und auf Adressüberschneidungen geprüft. Das hat den Vorteil, dass es zu keiner Adressüberschneidung innerhalb der Konfiguration kommt. Will der Benutzer die Adressen allerdings selber zuweisen (Abbildung 14), sollten diese Einstellungen deaktiviert werden.

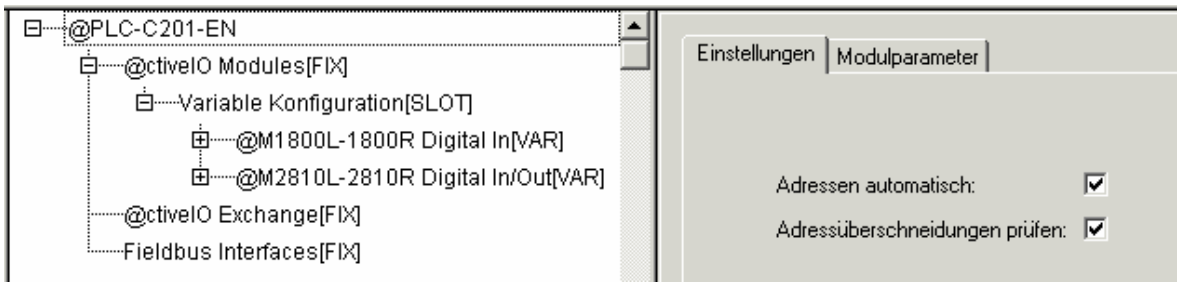

Abbildung 13: Einstellungen CoDeSys Steuerungskonfiguration

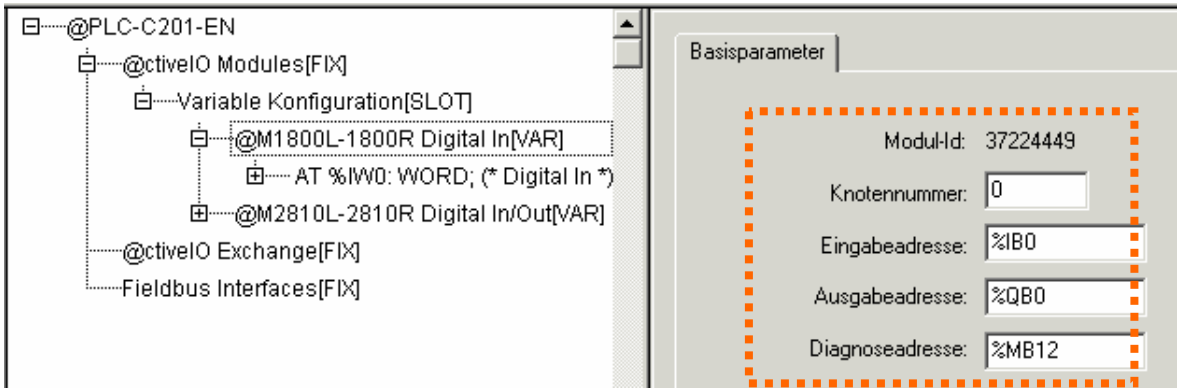

Abbildung 14: Adresszuweisung in der Steuerungskonfiguration

2.8. Erstellung des Beispielprogramms

Zum Erstellen des Beispielprogramms wechseln Sie in das noch offene Fenster des Programmeditors, das beim Erstellen des Programmbausteins erzeugt wurde.

Der Programmeditor (Abbildung 15) ist zweigeteilt: im oberen Teil werden die Variablen deklariert, der untere Teil enthält den Programmcode.

Zunächst sollten die Variablen deklariert werden. Hierfür werden „Counter“ als Output-Variable (Typ WORD) und „Enable“ und „Reset“ als Input-Variablen (Typ BOOL) deklariert. Nach der Deklaration der Variablen wird das Programm im Programmeditor geschrieben.

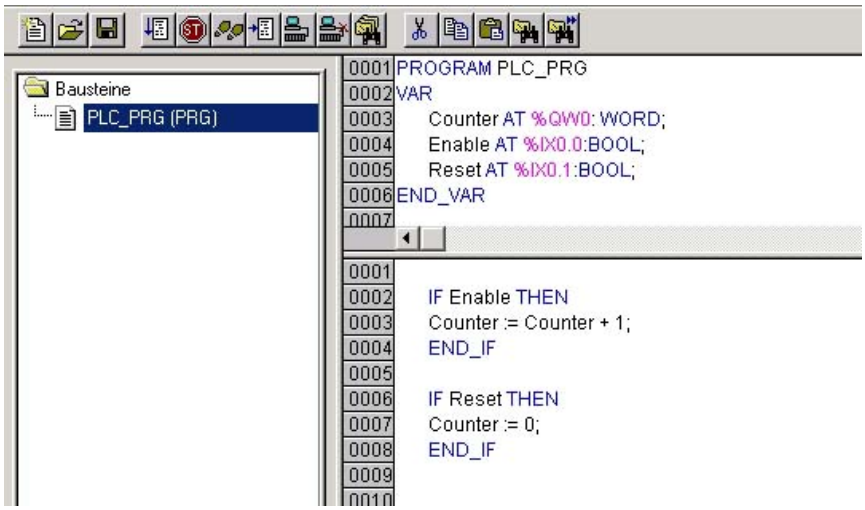

Abbildung 15: Beispielprogramm

Bevor das Projekt übersetzt werden kann, muss die Bibliothek **„Lib_C200_Utils.lib“** eingebunden werden. Diese Bibliothek enthält die Deklaration der von System verwendeten Systemvariablen. Die Einbindung ist im Kapitel 3.4 beschrieben.

Nachdem das Programm erstellt worden ist, sollte es abgespeichert werden um es anschließend unter ´Projekt´ ´Übersetzen´ oder mit der Funktionstaste **<F11>** übersetzten zu lassen. Enthält das Projekt noch Fehler, wird dies im Meldungsfenster angezeigt. Um ein Projekt auf den Controller zu laden bzw. um überhaupt eine Verbindung zum Controller herstellen zu können, muss das Projekt fehlerfrei sein.

2.9. Erstellen einer Taskkonfiguration

Bei kleineren Projekten, die nur aus einem zyklisch durchlaufenden Hauptprogramm bestehen, ist es nicht unbedingt notwendig, eine Taskkonfiguration durchzuführen. Mitunter kann es aber sinnvoll sein. In diesem Fall wollen wir den Zähler alle 100ms um 1 hochzählen lassen. Hierfür gibt es mehrere Möglichkeiten. Eine Möglichkeit wäre, einen Timer zu verwenden und immer wenn der Timer abgelaufen ist, den Zähler hochzählen zu lassen. Eine elegantere Möglichkeit bietet die Taskkonfiguration: Wir können eine Task definieren, deren Eigenschaft es ist, alle 100ms aufgerufen zu werden. Bei jedem Aufruf wird dann der Zähler um 1 hochgezählt.

Auf die Frage, was eine Task eigentlich ist und wie die Verwaltung und Parametrierung mehrerer Tasks funktioniert, wird zu einem späteren Zeitpunkt näher eingegangen.

Um nun eine Task anzulegen, wechseln sie in das Fenster der Taskkonfiguration (zu finden im Object Organizer unter **Ressourcen** => Taskkonfiguration). Das Fenster ist zweigeteilt. Links befindet sich eine Baumansicht, in der alle Tasks mit dazugehörigen Programmen dargestellt werden. In der rechten Hälfte befinden sich die Einstellungen zum links ausgewählten Objekt.

Zum Anlegen der Task bewegen Sie den Mauscursor auf den Baumeintrag „Taskkonfiguration“ und betätigen die rechte Maustaste. Wählen Sie im dann erscheinenden Menü den Eintrag „Task anhängen„ aus. In der linken Fensterseite erscheint nun unter „Taskkonfiguration“ ein weiterer Eintrag mit dem Namen „NeueTask“. Dieser Name kann geändert werden, indem Sie den Eintrag „NeueTask“ markieren und dann im rechten Fensterteil den Parameter „Name“ editieren, wie in diesem Beipiel „MainTask“. Damit die Task wie gewünscht alle 100ms aufgerufen wird, sind noch weitere Einstellungen notwendig. Unter „Typ“ sollte wie in diesem Fall „Zyklisch“ ausgewählt sein. Unter „Eigenschaften“ kann jetzt die Zeitspanne angegeben werden, wann die Task abgearbeitet werden

soll: „Intervall“ bekommt den Wert „T#100ms“. Achtung: Der Syntax für die Zeit-Eingabe muss eingehalten werden, da sonst die Task nicht ausgeführt wird.

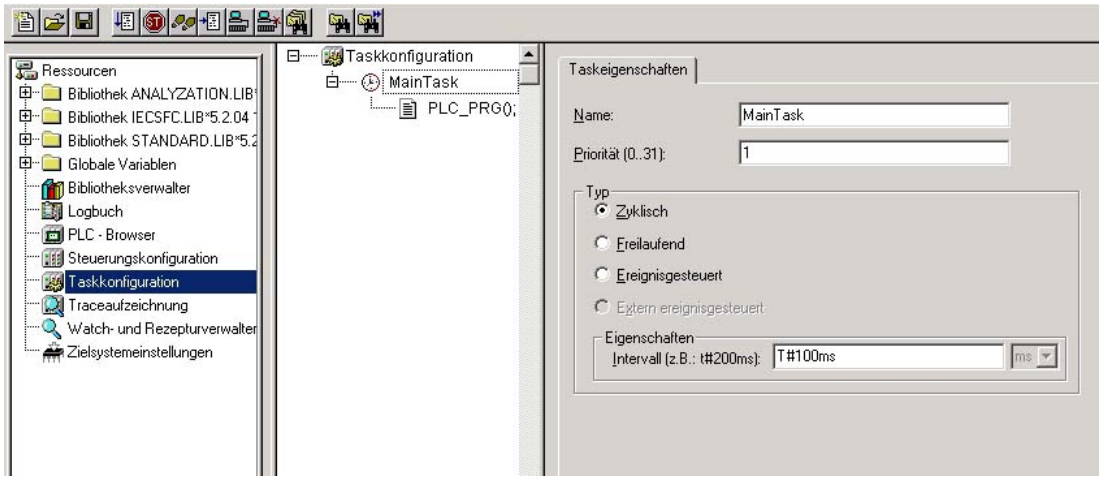

Abbildung 16: Taskkonfiguration

Im nächsten Schritt muss jetzt nur noch der Task mitgeteilt werden, welche Programmteile von ihr aufgerufen werden sollen. Hierzu bewegen Sie den Mauszeiger auf den Eintrag „MainTask“ (Abbildung 16) in der linken Fensterseite und klicken die rechte Maustaste. Wählen Sie „Programmaufruf anhängen“ in dem erscheinenden Menü aus. In der rechten Fensterseite haben Sie nun die Möglichkeit, ein Programm auszuwählen: Tragen die das gewünschte Programm, in unserem Fall das **PLC_PRG(PRG)**, von Hand ein oder wählen Sie es durch Drücken des „...“-Buttons aus einer Liste aus. Bei einem Handeintrag sollte auf die exakte Schreibweise geachtet werden, da sonst die Task nicht ausgeführt werden kann! Anschließend ist die Taskkonfiguration abgeschlossen.

2.10. Einstellung der Kommunikationsparameter

Bevor das Programm dem Controller gesendet werden kann, muss der PC und der Controller, mittels eines seriellen Übertragungskabels oder Netzwerkkabels verbunden sein und die korrekten Kommunikationsparameter eingestellt werden. Diese werden unter ’Online’ ’Kommunikationsparameter’ eingestellt. (Eine nähere Beschreibung der Kommunikationsparameter finden Sie im Abschnitt A.1).

Zur Erstellung einer neuen Verbindung klicken Sie auf den Button „Neu“ und wählen anschließend die Art der Verbindung aus: „Serial RS232“ bzw. „TCP/IP (Level2)“ und bestätigen Sie dann mit dem ’OK’ Button (Abbildung 17)

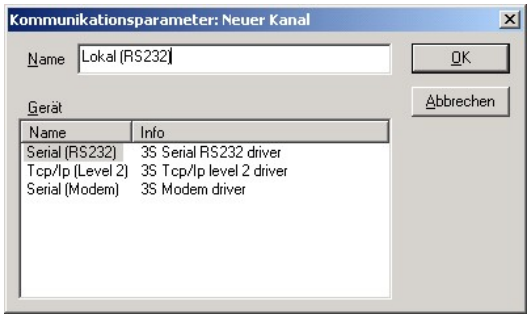

Abbildung 17: Kommunikationsparameter Neuer Kanal

Ändern sie dann die Parameter entsprechend Ihrer Konfiguration wie in den nachfolgenden Abbildung 18 bzw. Abbildung 19 dargestellt. Sollten Sie eine Verbindung über TCP/IP ausgewählt haben, stellen Sie unter **„Address“** die für Ihr Modul korrekte IP-Adresse ein.

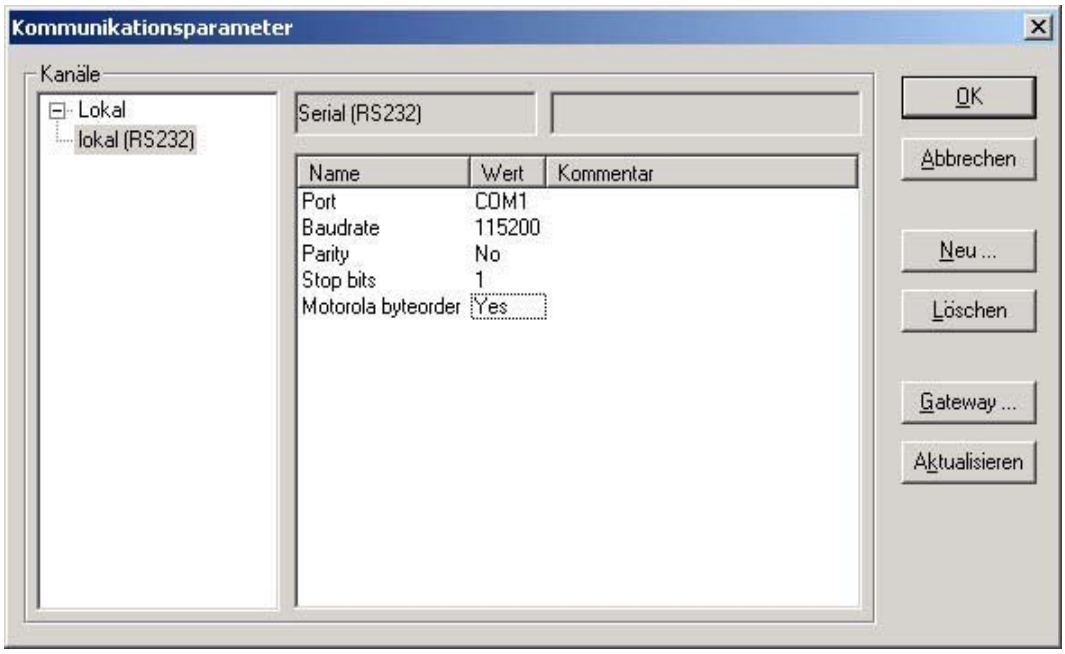

Abbildung 18: Kommunikationsparameter über serielle Schnittstelle (RS232)

Bei Netzwerkverbindungen ist darauf zu beachten, dass die entsprechenden Einstellungen zu Subnet, Gateway etc. sowohl an der @PLC als auch am PC korrekt vorgenommen werden und übereinstimmen.

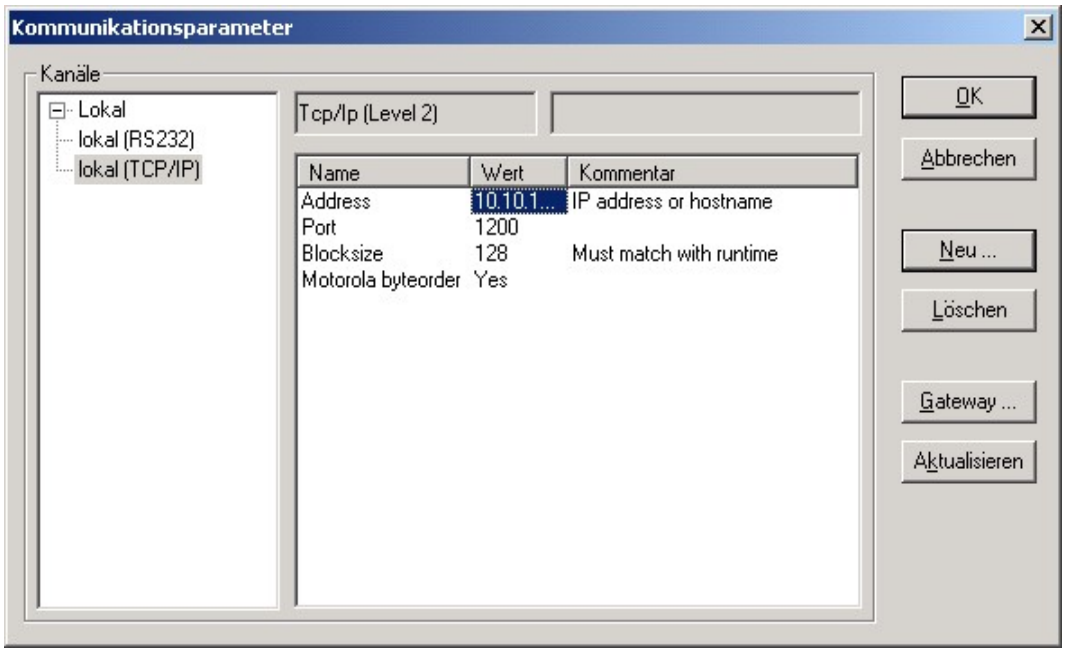

Abbildung 19: Kommunikationsparameter über Netzwerk (TCP/IP)

2.11. Laden und Starten des Beispielprogramms

Nachdem die Kommunikationsparameter eingestellt worden sind, kann das Programm unter ´Online´ ´Einloggen´ an den Controller gesendet werden. Gestartet wird das Programm unter ´Online´ ´Start´ oder mit **<F5>**.

Am rechten Rand der unteren Statusleiste (Abbildung 20) befinden sich Informationen wie z.B. über den Zustand der Verbindung und ob die PLC sich im Halt- bzw. Run-Zustand befindet.

Abbildung 20: Statusleiste CoDeSys Programm

2.11.1 Verbindung aktiv

Besteht eine Verbindung zur PLC, verändern sich die Programmfenster (Abbildung 21). Sie sind in diesem Zustand zweigeteilt und enthalten neben dem Programmcode nun auch den Zustand bzw. den Wert der verwendeten Variablen.

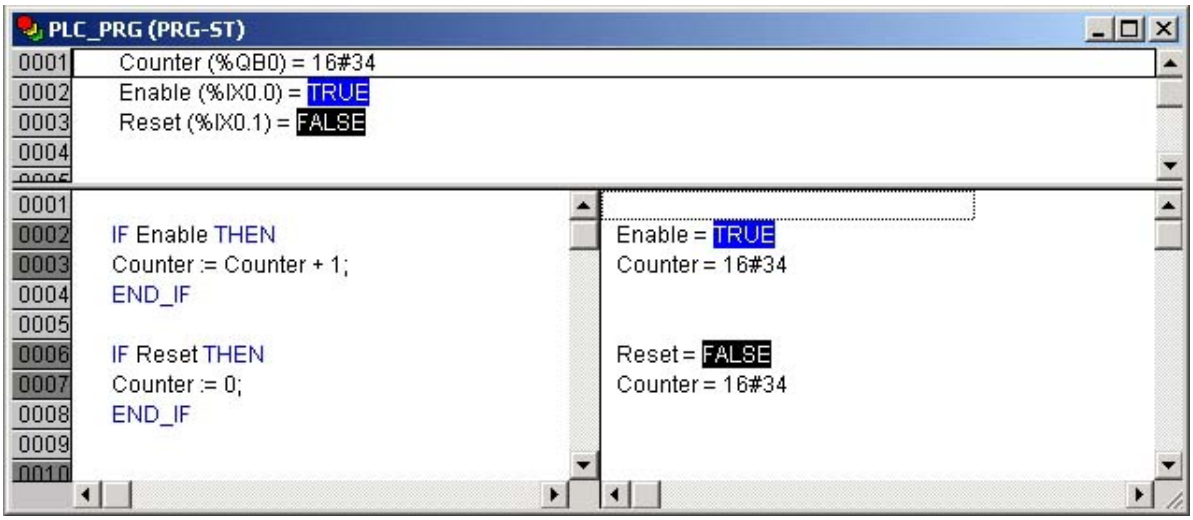

Abbildung 21: Programmfenster CoDeSys Programm

Nachdem Sie das Programm gestartet haben, wird aller Wahrscheinlichkeit nach der Zähler nicht hochzählen (alle Eingänge haben Low-Pegel). Sobald Sie aber den ersten Eingang mit 24V belegen, also die Freigabe für den Zähler setzen, beginnt er zu zählen. Über den zweiten Eingang lässt sich der Zähler nun wieder zurücksetzen.

2.12. Einfaches Debuggen

CoDeSys bietet eine Vielzahl von Möglichkeiten zum Debuggen eine Projekts: Setzen von Breakpoints, Einzelschritt und Einzelzyklus, Setzen und Forcen von Variablen, usw. Eine vollständige Übersicht und Beschreibung finden Sie in der CoDeSys-Dokumentation.

Zur Demonstration der Debug-Möglichkeiten soll das Beispielsprogramm mit einem Breakpoint versehen werden. Klicken Sie mit der linken Maustaste hierzu in der Zeile 6 des Beispielprogramms (Setzen des Counters auf 0) auf die dunkelgrau hinterlegte Zeilennummer im Programmierfenster. Nach dem Klicken ändert sich die Farbe von dunkelgrau nach turkis. Hiermit ist der Breakpoint aktiv.

Um nun den Programmablauf auf diesen Breakpoint zu lenken, habe wir zwei Möglichkeiten: Setzen des Reset-Eingangs auf 24V oder forcen des Eingangssignals. In diesem Fall soll der Eingang geforced werden. Um einen Wert zu forcen, muss zunächst der Wert der Variable verändert werden. Hierzu klicken Sie in der Variablenansicht (oberer Teil des Fenster bzw. rechts neben dem Programmtext) doppelt auf die Variable „Reset“. Bei Bool-Variablen wird direkt der neue Wert angezeigt, bei anderen Variablen erscheint zunächst ein weiteres Eingabefenster. Nun ist der Wert gesetzt und muss nur noch aktiviert werden. Das Forcen des Wertes kann mit Hilfe von Online -> Wert forcen bzw. **<F7>** geschehen. Sie haben damit den Wert der Variable temporär beeinflusst.

Nachdem die Variable geforced wurde, wechselt die Farbe des zuvor angelegten Breakpoint von turkis nach rot. Das bedeutet, dass der Breakpoint angesprungen wurde und das Programm an dieser Stelle stehen geblieben ist. Wir können nun das Forcen wieder aufheben (Online -> Forcen aufheben bzw <Umschalt+F7>), um beim nächsten Zyklus wieder den Originalwert einzulesen.

Um zu sehen, wie der Zähler zurückgesetzt wird, verwenden wir jetzt die Einzelschritt-Funktion: Online -> Einzelschritt über bzw. <F10>). Die rote Markierung springt dann zum nächsten Befehl und der Zählerwert wurde zurückgesetzt. Um das Programm weiter laufen zu lassen, verwenden wir jetzt einige Male die Einzelzyklusfunktion (Online -> Einzelzyklus bzw. <Strg+F5>), um zu sehen, wie der Zähler hochzählt.

Wenn Ihre Aktion erfolgreich war, können Sie das Programm wieder normal laufen lassen. Hierzu können Sie über `Online` -> `Start` bzw. <F5> das Programm wieder starten.

2.13. Erstellen des Bootprojekts

In der Praxis macht es wenig Sinn, wenn ein Programm nach jedem Einschalten der Betriebsspannung erneut mittels CoDeSys auf die PLC gespielt werden muss. Hierzu gibt es die Funktion „Bootprojekt erstellen“. Das Bootprojekt wird hierbei dauerhaft auf der PLC gespeichert und nach dem Einschalten der Betriebsspannung automatisch geladen und gestartet. Es enthält neben dem eigentlichen PLC-Programm auch dessen Konfiguration (Steuerungskonfiguration, Taskkonfiguration, usw.).

Das Erzeugen des Bootprojekts funktioniert ganz einfach. Im Online-Modus auf Online -> Bootprojekt erzeugen klicken. Anschließend wird das Bootprojekt erzeugt, zum Modul übertragen und überprüft. Sollte bereits ein Bootprojekt auf der PLC vorhanden sein, wird dieses zuvor gelöscht.

Nach einem Neustart der PLC wird direkt nach dem Hochlauf dieses Bootprojekt ausgeführt.

2.14. Viel Erfolg

Wenn Sie das **Kapitel 2**. auf die beschriebene Weise durchgeführt haben, haben Sie bereits Ihr erstes CoDeSys Projekt erstellt. Viel Erfolg bei der weiteren Erstellung Ihres eigenen Projektes.

3. Programmkonfiguration

3.1. Zielsystemeinstellungen

Die Zielsystemeinstellungen erlauben es, das Zielsystem in gewissen Grenzen an die Vorgaben durch das IEC-Projekt anzupassen. Es stehen die folgenden Einstellungsbereiche zur Verfügung:

Zielplattform:

In diesem Bereich können Einstellungen durchgeführt werden, die für die Programmerstellung von CoDeSys benötigt werden. Diese Einstellungen können durch den Anwender nicht verändert werden.

Speicheraufteilung:

In diesem Bereich können Einstellungen zum Adressbereich der einzelnen von CoDeSys verwendeten Segmente durchgeführt werden. Diese Einstellungen können durch den Anwender nicht verändert werden.

Allgemein:

Hier können allgemeinen Einstellungen durchgeführt werden, die das Verhalten des IEC-Programms während der Initialisierungsphase bestimmen. Des weiteren sind hier auch Einstellungen möglich, die das Verhalten von CoDeSys während des Programm-Downloads festlegen.

Netzfunktionen:

CoDeSys bietet einige Möglichkeiten zur Kommunikation zwischen PLCs an, die in diesem Bereich angegeben werden können. Hier sind es der „Parameter-Manager“ und die „Netzvariablen“. Netzvariablen werden über UDP (Ethernet) unterstützt.

Visualisierung:

In diesem Bereich können Angaben zur Visualisierung auf der PLC gemacht werden. Diese Funktion wird zur Zeit nicht unterstützt.

3.2. Steuerungskonfiguration

Die Steuerungskonfiguration dient dazu, das Prozessabbild der IEC mit der angeschlossenen Hardware (@ctiveIO-Module und Feldbuskomponenten) zu verbinden, um so einen Datenaustausch von und zur angeschlossenen Hardware durchzuführen.

Der Editor der Steuerungskonfiguration ist zweigeteilt: auf der linken Seite befindet sich ein Baum mit der (angeschlossenen) Hardwarekonfiguration und auf der rechten Seite befindet sich die Konfiguration zu dem ausgewählten Knoten der linken Seite. Bei einem neuen Projekt besteht die Steuerungskonfiguration zunächst aus einem leeren Konfigurationsbaum, der nur die wesentlichen Strukturelemente enthält. Diese Elemente sind mit den voreingestellten Standardwerte vorkonfiguriert. Ausgehend von dieser Standardkonfiguration kann der Anwender nun seine Konfiguration entsprechend der angeschlossenen Hardware vornehmen.

3.2.1 Basiskonfiguration

Mit der Basiskonfiguration können Parameter des zugrundeliegenden Laufzeitsystems eingestellt werden, die das Gesamtverhalten der PLC beeinflussen („Modulparameter“).

Max. Zykluszeit

Mit diesem Parameter kann die Zeitspanne in Millisekunden eingestellt werden, mit der der PLC-Zyklus überwacht wird. Benötigt ein PLC-Zyklus, z.B. aufgrund eines Fehlers oder einer überlangen Schleife, mehr als die hier eingestellte Zeitspanne, schlägt ein Watchdog zu und führt eine Neuinitialisierung der PLC durch: das Bootprojekt wird geladen, aber nicht gestartet. Tritt der Watchdog während des Laden des Bootprojekts auf (z.B. wegen eines fehlerhaften Bootprojekts), wird wiederum eine Neuinitialisierung der PLC durchgeführt. Diesmal wird allerdings keine Bootprojekt geladen.

Kommunikationszyklus

Kommunikation findet bei der PLC standardmäßig zwischen zwei PLC-Zyklen statt. Während des Zyklus ist die Kommunikation blockiert. Bei Programmen mit langen Zykluszeiten kann dieses

Verhalten dazu führen, dass die PLC gar nicht mehr ansprechbar erscheint – CoDeSys meldet beim Verbindungsaufbau eine Kommunikationsfehler.

Mit diesem Parameter kann vorgegeben werden, wann ein Kommunikationszyklus durchgeführt werden soll. Angegeben wird der Wert in Ticks (1 Tick = 10ms) nach dem Start des PLC-Zyklus. Ein Wert von 5 bedeutet beispielsweise, dass die PLC alle 50ms für einen Kommunikationszyklus unterbrochen wird. Ist die PLC-Zykluszeit geringer als diese 50ms, findet der Kommunikationszyklus wie gewohnt am Ende des PLC Zyklus statt. Der Default-Wert von 0 sorgt dafür, dass der Kommunikationszyklus zwischen zwei PLC Zyklen stattfindet.

Timeout Dateizugriff

Das verwendete Dateisystem ist dafür ausgelegt, dass z.B. das PLC-Programm und der ftp-Server gleichzeitig auf Dateien im Dateisystem zugreifen können. Damit es aber nicht zu einer Beschädigung der internen Kontrollstrukturen des Dateisystems kommt, ist der Zugriff verriegelt, so dass nur jeweils einer die Daten verändern kann. Der oder die Anderen müssen solange warten. Mit Hilfe dieses Parameters kann die maximale Wartezeit eingestellt werden. Konnte innerhalb diese Zeitspanne kein Zugriff auf die Kontrollstrukturen erfolgen, werden die entsprechenden Dateisystemfunktionen mit einem Fehler verlassen. Das IEC-Programm muss darauf in geeigneter Weise reagieren.

ActiveBus-Fehler ignorieren

Mit diesem Flag kann eingestellt werden, wie auf Fehler vom ActiveBus reagiert werden soll. Dieses Flag findet Anwendung, wenn die PLC ohne ActiveBus-Module betrieben wird (z.B. nur als PB-Slave). Ist dieses Flag aktiviert, werden sämtliche ActiveBus-Fehlermeldungen unterdrückt.

I/O-Update Zyklus

Mit diesem Parameter kann eingestellt werden, mit welcher Taktrate der I/O-Update Zyklus betrieben werden soll (1 Tick entspricht einer Millisekunde). Der I/O-Update Zyklus ist unabhängig vom PLC-Zyklus und dessen Austausch des Prozessabbildes. Beide sind allerdings synchronisiert, so dass ein PLC-Zyklus (mit PA-Austausch) erst nach dem nächsten I/O-Update Zyklus abgeschlossen werden kann. Wird der Wert dieses Parameters zu hoch eingestellt, kann das einen Einfluss auf die PLC-Zykluszeit haben.

3.2.2 @ctiveIO Module

In diesem Element werden die @ctiveIO-Module konfiguriert, die als Erweiterungs-Hardware an die PLC angeschlossen sind. Nach Anwendungsfall stehen drei verschiedene Konfigurationsarten zur Verfügung: Variable Konfiguration (Standard), automatische Konfiguration und automatische Konfiguration (sortiert). Die Konfigurationsart kann durch Rechtsklick auf den Konfigurationseintrag verändert werden (Menüeintrag „Element ersetzen“).

Konfigurationsart: „Variable Konfiguration“

Der Anwender wählt hier alle seine Module aus einer Liste aus. Es ist zu beachten, dass die so erstellte Modulliste exakt mit den vorhandenen und angeschlossenen Modulen übereinstimmen muss, ansonsten findet kein Datenaustausch zwischen dem PLC-Prozessabbild und den Modulen statt. Einige Module bzw. einzelne Kanäle dieser Module bieten die Möglichkeit, sich parametrieren zu lassen (über „Modulkonfiguration“ bzw. „Kanalkonfiguration“). Sollte die Steuerungskonfiguration nicht mit den angeschlossenen Modulen übereinstimmen, so wird beim einloggen eine Fehlermeldung ausgegeben.

Konfigurationsart: „Automatische Konfiguration“

Auf Anwenderseite ist hier keine weitere Konfiguration notwendig. Das Prozessabbild der @ctiveIO-Module wird unverändert ins Prozessabbild der PLC kopiert. Es ist zu beachten, dass das Prozessabbild der @ctiveIO-Module im Intel-Format (Little Endian Byte Order) vorliegt, die PLC aber mit Motorola-Format (Bit Endian Byte Order) arbeitet. Um einfach auf diese Daten zu zugreifen, existieren eine Reihe von Zugriffsfunktionen (siehe Lib_C200_Utils).

Konfigurationsart: „Automatische Konfiguration (sortiert)“

Auch hier ist von Anwenderseite aus keine weitere Konfiguration notwendig. Anders als bei der automatischen Konfiguration sind die Daten der @ctiveIO-Module sortiert im Prozessabbild der PLC abgelegt. Zunächst werden die „digitalen“ Module (siehe Übersicht) angeordnet, alle weiteren Module folgen dann entsprechend. Für diese Module kann ein Offset festgelegt werden, ab dem sie im Prozessabbild abgelegt werden. Dieser Offset bezieht sich auf den Begin des @ctiveIO Eingangs- bzw. Ausgangsprozessabbildes. Eine Besonderheit bei dieser Konfigurationsart im Vergleich zur

vorherigen automatischen Konfiguration liegt darin, dass eine Konvertierung ins Motorola-Format durchgeführt wird.

Die „digitalen“ Baugruppen sind: (Diese Liste wird erweitert)

- P1410 ..... 4 dig. Eingänge 8,2V Namur
- P1800 ..... 8 dig. Eingänge 24V
- P1801 ..... 8 dig. Eingänge 24V 0,2ms
- P1803 ..... 8 dig. Eingänge 12V
- P1804 ..... 8 dig. Eingänge 24V
- P2411 ..... 4 dig. Ausgänge 24V 1,5A
- P2412 ..... 4 dig. Ausgänge 24V 2,5A
- P2430 ..... 4 dig. Ausgänge Relais-Output 24V
- P2810 ..... 8 dig. Ausgänge 24V 0,8A
- P2813 ..... 8 dig. Ausgänge 12V 0,8A

3.2.3 @ctiveIO Exchange

Dieser Teil des Prozessabbildes dient dem Datenaustausch zwischen einem PC und der PLC. Es gibt einen Eingangs- und einen Ausgangsblock, dessen Format an dieser Stelle konfiguriert werden kann. Hierzu stehen verschiedene Datentypen jeweils im Intel- und Motorola-Format zur Verfügung. Der Zugriff von PC aus erfolgt über DLLs, die das @ctiveIO-Toolkit zur Verfügung stellt.

3.2.4 Fieldbus Interface

Neben den @ctiveIO-Modulen können auch weitere Feldbusse im Slave-Modus verwendet werden. Feldbus-Master werden an dieser Stelle nicht konfiguriert, sondern stehen als Funktionsblöcke zur Verfügung.

Abhängig von der eingesetzten PLC stehen verschiedene Feldbusmodule zur Verfügung. Dieser können in die Konfiguration eingehängt und anschließend konfiguriert werden. Zum Einhängen führen sie eine Rechtsklick auf den „Fieldbus Interfaces“ Eintrag aus und wählen im Menüpunkt „Unterelement anhängen“ den gewünschten Feldbus aus.

3.2.5 Profibus DP Slave

Für den Profibus stehen zwei Konfigurationsarten zur Verfügung: die **„Standardkonfiguration“** und die **„Variable Konfiguration“**. Die gewünschte Konfigurationsart kann durch Rechtsklick auf den Konfigurationseintrag geändert werden („Element ersetzen“ auswählen).

Die Bezeichnungen **„Eingangsdaten“** bzw. **„Ausgangsdaten“** beziehen sich immer aus der Sichtweise der PLC. **„Eingangsdaten“** enthalten Daten, die an die PLC vom Profibus gesendet werden und **„Ausgangsdaten“** werden von der PLC an den Profibus-Master weitergeleitet.

Die **Eingangs**- und **Ausgangsadresse** unter Basisparameter können frei gewählt werden, es muss aber darauf geachtet werden, dass es zu keiner Adressüberschneidung kommt. Sollte das der Fall sein so wird eine Fehlermeldung im unterem Meldungsfenster ausgegeben.

Für die jeweilige Konfigurationsart stehen noch weiter Parametriermöglichkeiten zur Verfügung. Diese sind unter „Modulparameter“ zu finden.

Parameter	Bedeutung
Schnittstelle	Auswahl der Profibus-Schnittstelle (z.Zt. steht nur eine Schnittstelle zur Verfügung: /pb/0)
Kennung	Profibuskennung – Über den eingestellten Wert kann auf die Eingangs- und Ausgangsdaten zugegriffen werden. Bei einem Wert von **„-1“** erfolgt die Einstellung der Profibus-Kennung über die Drehschalter der CPU.

Konfigurationsart: „Standardkonfiguration“

Bei der Standardkonfiguration stehen drei Datenblöcke zur Verfügung: **Eingangsdaten**, **Ausgangsdaten** und **Status/Control**. Die Eingangs- und die Ausgangsdaten können unabhängig voneinander in der Größe verändert werden. Es stehen 1 bis 10 Worte zur Auswahl, wobei nur jedes einzelne Wort konsistent ist. Der Status/Control Block kann nicht verändert werden. Die Einstellung der Blockgröße erfolgt durch Rechtsklick auf das Blockelement mit anschließender Auswahl der gewünschten Blockgröße („Element ersetzen“ auswählen).

Konfigurationsart: „Variable Konfiguration“

Im Gegensatz zu Standardkonfiguration kann das Prozessabbild bei der variablen Konfiguration feiner eingestellt werden. Das Prozessabbild kann hier aus diversen Datenblöcken zusammengebaut werden. Diese Datenblock stehen in Größen von **1**, **2**, **4**, **8**, **16** und **32 Worte** zur Verfügung. Des Weiteren kann zwischen konsistenten und nicht konsistenten Blöcken unterschieden werden. Diese Blöcke können bis zur maximal von Profibus zur Verfügung gestellten Größe kombiniert werden. Das Einfügen von Datenblöcken geschieht durch Rechtsklick auf das Konfigurationselement mit anschließender Auswahl über „Unterelement anhängen“.

3.2.6 Modbus RS232

Das Modbus RS232 Interface kann hochgradig konfiguriert und den jeweiligen Anforderungen entsprechend angepasst werden. Unterstützt werden die Übertragungsarten RTU und ASCII mit bis zu 115200 Baud. Die maximale Buffergröße beträgt bis zu 16 KB, d.h. es können bis zu 131072 digitale Ein- und Ausgänge bzw. 8192 Register angesprochen werden. Es stehen zwei Buffer-Modi zur Verfügung: 2-Buffer-Mode und 4-Buffer-Mode.

Im 2-Buffer-Mode verwenden „Input Register“ und „Output Register“ bzw. „Input Holding Register“ und „Output Holding Register“ auf Modbus-Seite jeweils den selben Buffer für Eingangs- und Ausgangsdaten. Um auch auf IEC-Seite die Ausgänge mit den Eingängen abzugleichen, muss die Option „Ausgänge mit Eingängen überschreiben“ ausgewählt werden. Im 4-Buffer-Mode stehen auf Modbus-Seite vier unabhängige Buffer zur Verfügung.

Für jeden Register-Bereich kann ein Offset festgelegt werden, der angibt, ab welcher Modbus-Adresse das erste Element angesprochen werden kann. Hierbei wird der Offset für lesenden und schreibenden Zugriff auf einen Ausgang gesondert behandelt.

Eine Sonderfunktion hat die Modbus-Funktion „Read Holding Register“. Mit ihr ist es möglich, alle Registerbereiche auszulesen. Über die 4 „HR-Offsets“ kann eingestellt werden, an welchem Offset auf den jeweiligen Registerbereich zugegriffen werden kann.

Die Bezeichnungen „Input Register“ bzw. „Output Register“ beziehen sich immer aus der Sichtweise der PLC. „Input Register“ enthalten Daten, die an die PLC gesendet werden, und „Output Register“ werden an den anfragenden Modbus-Master weitergeleitet.

Parameter	Beschreibung
Schnittstelle	Bezeichnung der seriellen Schnittstelle (z.Zt. steht nur /com/0 zur Verfügung)
Baudrate	Baudrate der Schnittstelle (75 – 115200 Baud)
Parität	Keine, gerade oder ungerade Parität
Stoppbits	1 oder 2 Stoppbits
Übertragungsmodus	Modbus Übertragungsmodus (RTU oder ASCII)
Adresse	Modbus Adresse
Watchdog Lesen	Max. zulässige Zeitspanne zwischen zwei Lesetelegrammen in Ticks (1 Tick = 10ms), startet mit dem ersten Lesetelegramm
Watchdog Schreiben	Max. zulässige Zeitspanne zwischen zwei Schreibtelegrammen in Ticks (1 Tick = 10ms), startet mit dem ersten Schreib-telegramm
Watchdog Zyklus	Zykluszeit der Watchdog-Überwachung in Ticks (1 Tick = 10ms)
Bit-Offset dig. Eingang (Lesen)	Adressoffset zum Lesen eines digitalen Eingangs
Bit-Offset dig. Ausgang (Lesen)	Adressoffset zum Lesen eines digitalen Ausgangs
Bit-Offset dig. Ausgang (Schreiben)	Adressoffset zum Schreiben eines digitalen Ausgangs
Offset Eingangsregister (Lesen)	Adressoffset zum Lesen eines Eingangsregisters
Offset Ausgangsregister (Lesen)	Adressoffset zum Lesen eines Ausgangsregisters

Offset Ausgangsregister (Schreiben)	Adressoffset zum Schreiben eines Ausgangsregisters
HR-Offset dig. Eingaenge	Read Holding Register: Offset der digitalen Eingänge
HR-Offset dig. Ausgaenge	Read Holding Register: Offset der digitalen Ausgänge
HR-Offset Eingangsregister	Read Holding Register: Offset der Eingangsregister
HR-Offset Ausgangsregister	Read Holding Register: Offset der Ausgangsregister
Modus I/O-Bereich	Auswahl zwischen 2- und 4-Buffer-Mode
Ausgänge mit Eingängen überschreiben	2-Buffer-Mode: IEC-Prozessabbild (Eingänge und Ausgänge) abgleichen
Buffergröße I/O-Bereich	Einzelgröße der Buffer (256B – 16KB)

Zur Inbetriebnahme des Modbus reicht die Konfiguration alleine nicht aus. Der Modbus ist standardmäßig nicht aktiviert. Erst durch den Einsatz der Bibliothek Lib_ModbusRS232 und des Funktionsbausteins „ModbusRS232_Init“ ist eine Aktivierung möglich (siehe dort).

3.2.7 Modbus TCP

Steht zur Zeit nicht zur Verfügung (in Vorbereitung).

3.3. Taskkonfiguration

Außer über das spezielle Programm „PLC_PRG“ kann die Abarbeitung eine Projekts auch über die Taskkonfiguration gesteuert werden.

Eine Task ist eine zeitliche Ablaufeinheit eines IEC-Programms. Sie ist definiert durch einen Namen, eine Priorität und einen Typ, der festlegt, welche Bedingungen ihren Start auslöst. Diese Bedingung kann entweder zeitlich definiert sein (Zyklusintervall, freilaufend) oder durch ein internes oder externes Ereignis, bei dessen Eintreten die Task ausgeführt werden soll; beispielsweise eine steigende Flanke einer globalen Projektvariablen oder ein Interrupt-Event der Steuerung.

Jeder Task kann eine Folge von Programmen zugeordnet werden, die bei Ausführen der Task abgearbeitet werden. Durch Zusammenwirken von Priorität und Bedingung wird festgelegt, in welcher zeitlichen Abfolge die Task abgearbeitet werden:

- Es wird die Task ausgeführt, deren Bedingung gilt, das heißt, wenn die bei Intervall angegebene Zeit abgelaufen ist, oder nach einer steigenden Flanke der bei Ereignis angegebenen Bedingungsvariable.
- Haben mehrere Tasks eine gültige Bedingung, dann wird die Task mit der höchsten Priorität ausgeführt.
- Haben mehrere Tasks eine gültige Bedingung und gleich hoch Priorität, dann wird die Task ausgeführt, die die längste Wartezeit hatte.
- Die Abarbeitung der Programmaufrufe pro Task im Online Modus erfolgt gemäß der Reihenfolge ihrer Anordnung im Taskeditor von oben nach unten.

Hinweis: Das Laufzeitsystem der @PLC-C20x Baureihe (32-Bit CoDeSys Version) ist nur bedingt multitaskingfähig: Tasks können nicht unterbrochen werden. Nach dem Start läuft die Task voll durch, bevor zur nächsten umgeschaltet werden kann. Aufgrund dieses Verhaltens kann es möglich sein, dass niederpriore Task eine höherpriore Task blockieren.

3.3.1 Zyklische Tasks

Bei zyklischen Tasks hat der Anwender die Möglichkeit, ein Intervall anzugeben, wann die Task laufen soll. Aufgrund interner Vorgänge kann es passieren, dass das Intervall zwischen zwei Aufrufen größer als die eingestellte Zeit wird. Das hat zur Folge, dass es dadurch zu einen Drift der Aufrufzeit kommt, die von der Taskverwaltung nicht kompensiert wird. Die Task wird weniger als erwartet aufgerufen.

3.3.2 Ereignisgesteuerte Tasks

Ereignisgesteuerte Tasks sind Tasks, die außerhalb der CoDeSys Taskverwaltung ablaufend und daher auch nicht ihren Einschränkungen unterliegen. Es stehen die folgenden Ereignisse zur Verfügung: System_10ms, TimerPAUpdate, TimerTick.

Bei der Verwendung von ereignisgesteuerten Tasks sind einige Punkte zu beachten. Diese Tasks laufen unabhängig und asynchron zum PLC-Zyklus. Bei gemeinsam genutzten Variablen sollte daher immer nur einer Schreibrechte erhalten, damit es nicht zu Datenverfälschungen aufgrund unterbrochener Schreibvorgänge kommt. Es sollte außerdem ein Handshake eingeführt werden, mit dem sichergestellt ist, dass die Daten gültig sind.

Auf Daten des Prozessabbildes sollte in ähnlicher Weise zugegriffen werden. Alternativ stehen in der Bibliothek Lib_C200_Utils Funktionen bereit, die den direkten Zugriff auf die I/O-Daten erlauben.

Warnung: Bei ereignisgesteuerten Tasks sollten Sie mit äußerster Vorsicht vorgehen, da sie das Verhalten der PLC stark beeinflussen können. Die Laufzeit sollte so kurz wie möglich sein, damit gerade die Timer-Events nicht die gesamte PLC zum Stillstand bringen und die dortigen Sicherheitsmechanismen (Watchdog) umgehen. Die Verwendung erfolgt auf eigenen Gefahr!

System_10ms

Dieses Event wird vom Betriebssystem generiert und ermöglicht eine Task, die alle 10ms aufgerufen wird. Aufgrund interner Vorgänge kann es zu einem Jitter kommen, der über die Zeit allerdings ausgeglichen wird.

TimerPAUpdate

Dieses Event wird nach jedem Update der I/O-Daten aufgerufen. Die Zeitspanne kann konfiguriert werden (siehe 3.2.1 Basiskonfiguration -> I/O-Update Zyklus). Der Aufruf der Funktion erfolgt in einer Interrupt Service Routine und sollte daher mit äußerster Vorsicht verwendet werden!

TimerTick

Dieses Event wird bei jedem Timer-Tick aufgerufen (1 Tick entspricht 1 Millisekunde). Der Aufruf der Funktion erfolgt in einer Interrupt Service Routine und sollte daher mit äußerster Vorsicht verwendet werden!

3.4. Systemvariablen

Die bei der @PLC verwendeten Systemvariablen sind globale Variablen, die Informationen zum Systemzustand liefern.

Die folgenden Systemvariablen stehen zur Verfügung:

- **C200_STARTUPSTATUS**
- **C200_SYSPARAMS**

Die Deklaration der von System verwendeten Systemvariablen befindet sich in der Bibliothek **„Lib_C200_Utils.lib“**.

3.4.1 C200_STARTUPSTATUS

Diese Systemvariable enthält gesammelte Informationen zum Start der CPU.

Variable	Datentyp	Beschreibung
ID	DWORD	ID der Datenstruktur
CRC	DWORD	Prüfsumme der Speicherstruktur
Params	D_C200_StartUpParams	Informationen zum Start der CPU
Regs	D_C200-StartUpRegs	Informationen zur zuletzt aufgetretener Exception
Reset	D_C200_StartUpReset	Informationen zur Startursache
ActiveThread	D_C200_StartUpThread	Zustand des zum Zeitpunkt der Exception gerade aktiven Threads
UserThread	D_C200_StartUpThread	Zustand eines applikationsspezifischen Threads zum Zeitpunkt der Exception

C200_StartUpParams (Enthält Informationen, wie die CPU gestartet werden soll)		
Variable	**Datentyp**	**Beschreibung**
StartUpMode	DWORD	Gibt an, welche Firmware beim Start verwendet werden soll
StartUpOptions	DWORD	Startoptionen - legen das Startverhalten fest

C200_StartUpRegs (Enthält Informationen über die zuletzt aufgetretene Exception (Register Dump)		
Variable	**Datentyp**	**Beschreibung**
Exception	DWORD	ID der aufgetretenen Exception
Reg_CPSR	DWORD	Registerinhalt „Current Program Status Register“
Reg SPSR	DWORD	Registerinhalt „Saved Program Status Register“
Reg_LR	DWORD	Registerinhalt „Link Register“
Reg_SP	DWORD	Registerinhalt „Stack Pointer“
Regs[8]	DWORD	Registerinhalt „R0 ... R7“
Stack[8]	DWORD	Stack-Ausschnitt

C200_StartUpReset (Liefert Informationen zur (Neu-) Startursache)		
Variable	**Datentyp**	**Beschreibung**
Counter	DWORD	Zähler unerwarteter Starts (bedingt z.B. durch Exceptios
SWTriggered	DWORD	Gibt an, ob der erfolgte Start durch einen erwarteten Software-Reset erfolgte
Flags	DWORD	Enthält Kopie der StartUp-Flags des FPGA-Bausteins (siehe StartUp-Flag Konstanten)

C200_StartUpThread (Liefert Informationen zum Zustand eines Threads vor dem letztem Neustart)		
Variable	**Datentyp**	**Beschreibung**
Name[32]	DWORD	Name des Threads
State	DWORD	Zustandskennung des Threads
RunCount	DWORD	Zähler wie oft der Thread vom Scheduler aufgerufen wurde
Stack[32]	DWORD	Stack-Ausschnitt

3.4.2 C200_SYSPARAMS

Diese Systemvariable enthält Daten zum Modul, die während der Startphase ermittelt werden.

Variable	Datentyp	Beschreibung
ID	DWORD	ID der Datenstruktur
BootString [C_SYSPARAMS_BootStringSize]	BYTE	Beim Start übergebener Bootstring („Kernel-Parameter“)
BootStringLen	WORD	Länge des übergebenen Bootstrings
ModeID	DWORD	Start-Modus der CPU – gibt an welche Firmware geladen werden soll (siehe Mode-ID Konstanten)
OptParams	BYTE	Zeiger auf zusätzliche Boot-Parameter (siehe BootString)
HWVersion	DWORD	Beim Start erkannte Hardwareversion der CPU (siehe HW-Variante Konstante)
Flash_ManufacturerID	DWORD	Hersteller des verwendeten Flashbausteins (siehe Flash-Hersteller Konstanten)
Flash_DeviceID	DWORD	Verwendeter Flash-Baustein (siehe Flash-Baustein Konstanten)
Size_SDRAM	DWORD	Grösse des SDRAMs
Size_NVRAM	DWORD	Grösse des NVRAMs
Addr_NVRAM	DWORD	Startadresse des NVRAMs
Size_SRAM	DWORD	Grösse des GoldCap-gepufferten SRAMs
Addr_SRAM	DWORD	Startadresse des GoldCap-gepufferten SRAMs
Size_SRAMBuffer	DWORD	Grösse des verwendbaren Speicherbereiches im SRAM
Addr_SRAMBuffer	DWORD	Startadresse des verwendbaren Speicherbereiches im SRAM

4. Bibliotheksverwaltung

4.1. Einfügen einer Bibliothek

Öffnen Sie das Register **Ressourcen** und betätigen Sie ein Doppelklick auf den **`Bibliotheksverwalter`**, somit wird die CoDeSys Bibliothekenverwaltung geöffnet. Das Fenster des Bibliotheksverwalter ist durch Bildschirmteiler in drei bzw. vier Bereiche aufgeteilt. Im linken oberen Bereich sind die dem Projekt angeschlossenen Bibliotheken aufgelistet. Bewegen Sie den Mauszeiger auf dieses Fenster und drücken die rechte Maustaste, es öffnet sich ein Kontextmenü (Abbildung 22). In diesem den Befehl **`'Weitere Bibliothek... Einfg'`** wählen und die jeweilige Bibliothek öffnen (Abbildung 23).

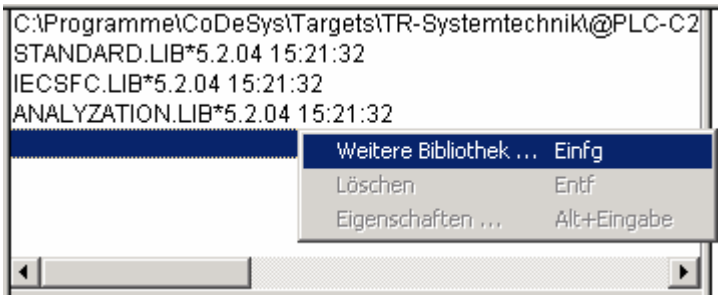

Abbildung 22: CoDeSys Bibliotheksverwalter 1

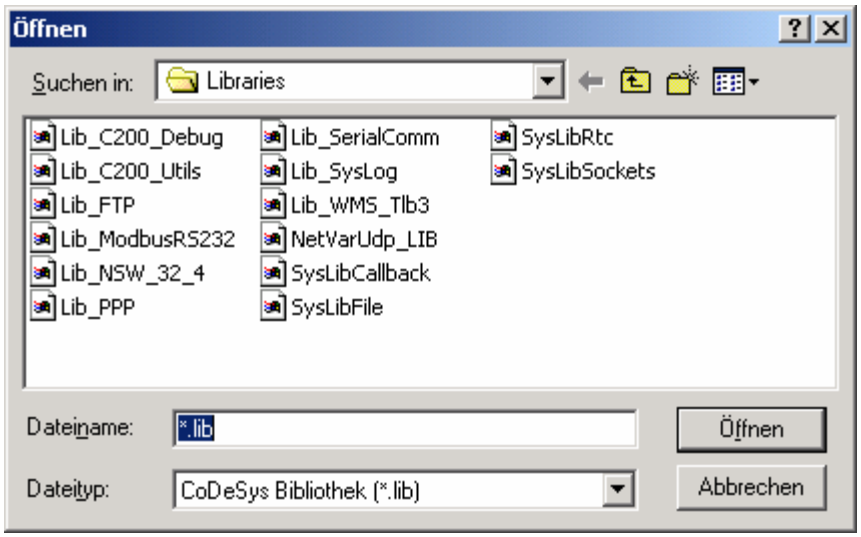

Abbildung 23: CoDeSys Bibliothekenverwalter 2

4.2. Standardbibliotheken

4.2.1 Die Bibliothek SysLibFile

Je nach Bauart der @PLC stehen unterschiedliche Speichermedien zur Verfügung: Alle Baugruppe verfügen über einen Flash-Baustein, der die Firmware und diverse Konfigurationsdaten enthält, und zur Speicherung eigener Daten verwendet werden kann. Optional erhältlich ist ein Gold-Cap gepufferter SRAM-Baustein, der ebenfalls zur Datenspeicherung herangezogen werden kann.

Diese beiden Bausteine besitzen besondere Charaktere, die den bevorzugten Verwendungszweck bestimmen. Flash-Bausteine haben eine begrenzte Anzahl an Schreibzugriffen, womit sie sich bevorzugt für Daten eignen, die eher selten geändert werden (wie z.B. Konfigurationsdaten). Diese Einschränkung besitzen SRAM-Bausteine nicht, daher eignen sie sich auch für sich ständig ändernde Daten (wie z.B. Log-Files). Aufgrund ihrer Gold-Cap Pufferung bleiben die Daten bei einem Spannungsausfall eine begrenzte Zeit erhalten.

Mit dieser Bibliothek kann auf das Dateisystem (Flash-FS und ggf. gepuffertes SRAM-FS) zugegriffen werden. Die Unterscheidung, auf welches Dateisystem zugegriffen wird, erfolgt über einen „Laufwerksbuchstaben“. Das Laufwerk A: bietet Zugriff auf das Flash-FS. Wird kein Laufwerksbuchstabe angegeben, wird standardmäßig auf Laufwerk A: zugegriffen. Der Zugriff auf des gepufferte SRAM-FS erfolgt über den Laufwerksbuchstaben **B:**.

Die nachfolgenden Funktionen stehen für den Umgang mit den Dateien zur Verfügung.

Hinweis: Das Dateisystem verfügt über mehrere Zugriffsebenen. Somit kann es möglich sein, dass auf bestimmte Dateien nicht zugegriffen werden kann, da mit zu geringen Zugriffsrechten gearbeitet wird (Die Bibliotheksroutinen arbeiten auf der niedrigsten Zugriffsebene).

Hinweis: Die Bibliotheksroutinen arbeiten zum Teil blockierend. Je nach Funktion und Umfang der verarbeiteten Daten kann der Zeitbedarf der einzelnen Funktionen unter Umständen bis auf mehrere Sekunden ansteigen (Löschung von Flash-Sektoren, usw.). Bei Verwendung dieser Bibliothek sollten die Zeiten der Zykluszeitüberwachung in der Steuerungskonfiguration entsprechend angepasst werden. Die Zykluszeit kann nicht aus dem Projekt heraus dynamisch angepasst werden.

Synchroner Zugriff (blockierend)

- SysFileOpen — Datei öffnen
- SysFileClose — Datei schließen
- SysFileFlush — Dateipuffer sofort schreiben
- SysFileWrite — in Datei schreiben
- SysFileRead — aus Datei lesen
- SysFileDelete — Datei löschen
- SysFileGetPos — aktuellen Offset in Datei ermitteln
- SysFileSetPos — Offset in Datei setzen
- SysFileEOF — Prüfen, ob Ende der Datei erreicht ist
- SysFileGetSize — Dateigröße ermitteln
- SysFileGetTime — Zeitangaben bzgl. Erstellung, Zugriff, Änderung
- SysFileCopy — Datei kopieren
- SysFileRename — Datei umbenennen
- SysFileGetFreeMem — liefert den freien Speicher des angegebenen Dateisystems

Asynchroner Zugriff (nicht-blockierend)

- SysFileOpenA — Datei öffnen
- SysFileCloseA — Datei schließen
- SysFileFlushA — Dateipuffer sofort schreiben
- SysFileWriteA — in Datei schreiben
- SysFileReadA — aus Datei lesen
- SysFileDeleteA — Datei löschen
- SysFileCopyA — Datei kopieren
- SysFileRenameA — Datei umbenennen

Die Funktionsblöcke zum asynchronen Zugriff arbeiten genau wie ihr Pendant des synchronen Zugriffs. Sie haben die selben Parameter. Der Unterschied liegt darin, dass die asynchronen Funktionen über zwei weitere Parameter verfügen (fEnable und fDone) und sie als Funktionsblock realisiert sind. Die beiden Funktionsarten können beliebig kombiniert werden.

Die Arbeit mit diesen Funktionsblöcken gestaltet sich folgendermaßen: zunächst werden die Funktionsblöcken mit den üblichen Parametern versorgt. Anschließend kann mittels fEnable die Ausführung gestartet werden. Die Parameter und besonders die Datenpuffer dürfen dann nicht mehr verändert werden, bis mit fDone das Ende der Ausführung signalisiert wird. Liegt fDone an, haben die Ergebniswerte Gültigkeit. Mit dem Abschließenden Zurücksetzen von fEnable werden fDone und die Ergebniswerte ebenfalls wieder zurückgesetzt und sind dann nicht mehr gültig.

Die Erläuterung der Parameter der einzelnen Funktionsblöcke findet sich bei ihrem Pendant des synchronen Zugriffs.

Allgemeine Erklärung:

FUN	→	Funktion
FB	→	Funktionsblock
PRG	→	Programm

SysFileOpen (FUN)

Diese Funktion vom Typ DWORD dient dem Öffnen einer bereits bestehenden oder neu zu generierenden Datei.

Der Rückgabewert ist eine 'Datei-Handle, die in den Funktionen SysFileWrite, SysFileRead, SysFileClose als Eingabe ('File') verwendet wird bzw. '0' für Fehler.

Input-Variable	Datentyp	Beschreibung
FileName	STRING	Datei-Name
Mode	STRING	Modus, in dem die Datei bearbeitet werden soll: w Schreiben (Öffnet Datei zum Schreiben, Datei wird überschrieben oder neu angelegt, Dateizeiger zeigt auf den Beginn der Datei) w+ Lesen / Schreiben (Öffnet Datei zum Lesen und Schreiben, Datei wird überschrieben oder neu angelegt, Dateizeiger zeigt auf den Beginn der Datei) r Lesen (Öffnet existierende Datei zum Lesen, Dateizeiger zeigt auf den Anfang der Datei) r+ Lesen / Schreiben (Öffnet existierende Daten zum Lesen und Schreiben, Dateizeiger zeigt auf den Anfang der Datei) a Anhängen (Öffnet Datei zum Schreiben, Dateizeiger zeigt auf das Ende der Datei) a+ Lesen / Anhängen (Öffnet Datei zum Lesen und Schreiben, Dateizeiger zeigt auf das Ende der Datei)

SysFileClose (FUN)

Diese Funktion vom Typ BOOL dient dem Schließen einer Datei, die mit SysFileOpen geöffnet worden war.

Der Rückgabewert ist 1 (OK) oder 0 (Fehler).

Variable	Datentyp	Beschreibung
File	DWORD	Datei-Handle (aus SysFileOpen)

SysFileFlush (FUN)

Diese Funktion vom Typ BOOL dient dem sofortigen Schreiben der Dateibuffer einer Datei, die mit SysFileOpen geöffnet worden war.

Der Rückgabewert ist 1 (OK) oder 0 (Fehler).

Variable	Datentyp	Beschreibung
File	DWORD	Datei-Handle (aus SysFileOpen)

SysFileWrite (FUN)

Diese Funktion vom Typ DWORD dient zum Schreiben von Daten in die Datei, die zuvor über die Funktion SysFileOpen geöffnet wurde.

Als Rückgabewert erhält man die Anzahl der erfolgreich geschriebenen Bytes.

Variable	Datentyp	Beschreibung
File	DWORD	Datei-Handle (siehe SysFileOpen)
Buffer	DWORD	Adresse des Buffers (ermittelbar mit Hilfe des Operators ADR) der zu schreibenden Daten
Size	DWORD	Anzahl der Bytes, die in die Datei geschrieben werden sollen (ermittelbar mit Hilfe des Operators SIZEOF)

SysFileRead (FUN)

Diese Funktion vom Typ DWORD dient zum Lesen einer Datei, die zuvor mit SysFileOpen geöffnet wurde.

Als Rückgabewert erhält man die Anzahl der erfolgreich gelesenen Bytes.

Variable	*Datentyp*	*Beschreibung*
File	DWORD	Datei-Handle (siehe SysFileOpen)
Buffer	DWORD	Adresse des Buffers der zu lesenden Daten (ermittelbar mit Hilfe des Operators ADR)
Size	DWORD	Anzahl der Bytes, die die aus dem Buffer gelesen werden sollen

SysFileDelete (FUN)

Diese Funktion vom Typ Bool dient dem Löschen einer Datei.

Der Rückgabewert ist 1 (OK) oder 0 (Fehler).

Variable	Datentyp	Beschreibung
FileName	STRING	Datei-Name

SysFileGetPos (FUN)

Diese Funktion vom Typ DINT gibt die aktuell gesetzte Offset-Position in der Datei, die mit der aus SysFileOpen erhaltenen Dateinummer angegeben wird.

Der Rückgabewert ist 1 (ok) oder 0 (Fehler).

Variable	Datentyp	Beschreibung
File	DWORD	Datei-Handle (aus SysFileOpen)

SysFileSetPos (FUN)

Diese Funktion vom Typ BOOL dient dazu, den aktuellen Offset (der über SysFileGetPos gelesen werden kann) für einen Dateizugriff zu verändern. Die Datei wird mit der aus SysFileOpen erhaltenen Dateinummer angegeben.

Der Rückgabewert ist 1 (ok) oder 0 (Fehler).

Variable	Datentyp	Beschreibung
File	DWORD	Datei-Handle (aus SysFileOpen)
Pos	DWORD	Offset innerhalb der Datei, der für Zugriffe gilt

SysFileEOF (FUN)

Diese Funktion vom Typ BOOL gibt an, ob das Dateiende erreicht ist.

Der Rückgabewert ist 1 (TRUE), wenn der aktuelle Offset am Ende der Datei steht; sie liefert 0 (FALSE), wenn das Dateiende noch nicht erreicht ist.

Variable	Datentyp	Beschreibung

FileName	DWORD	Datei-Handle (aus SysFileOpen)

SysFileGetSize (FUN)

Diese Funktion vom Typ DINT ermittelt die Dateigröße der angegebenen Datei.

Der Rückgabewert ist die Größe der mit FileName angegebenen Datei in Bytes.

Variable	Datentyp	Beschreibung
FileName	STRING	Datei-Name

SysFileGetTime (FUN)

Diese Funktion vom Typ BOOL liefert den Zeitpunkt der Erstellung der Datei (angegeben mit FileName) im Format DT. Die Angaben zum letzten Zugriff und der letzten Modifikation werden nicht unterstützt (Zeitstempel 0 bzw. 01.01.1970). Zu diesem Zweck kann auf die Elemente der Struktur FILETIME zugegriffen werden.

Der Rückgabewert ist 1 (ok) oder 0 (Fehler).

Variable	Datentyp	Beschreibung
FileName	STRING	Datei, in die kopiert werden soll
FileTime	POINTER TO FILETIME	Zeigt auf Struktur FILETIME; der Operator ADR kann hierbei zu Hilfe genommen werden

SysFileCopy (FUN)

Diese Funktion vom Typ UDINT dient dazu, den Dateiinhalt in eine Datei mit anderem Namen zu kopieren.

Der Rückgabewert ist die Anzahl der kopierten Bytes.

Variable	Datentyp	Beschreibung
FileDest	STRING	Datei, in die kopiert werden soll
FileSource	STRING	Datei, aus der kopiert werden soll

SysFileRename (FUN)

Diese Funktion vom Typ BOOL dient dem Umbenennen einer Datei.

Der Rückgabewert ist 1 (ok) oder 0 (Fehler).

Variable	Datentyp	Beschreibung
FileOldName	STRING	Bisheriger Datei-Name
FilenNewName	STRING	Neuer Datei-Name

SysFileGetFreeMem (FUN)

Diese Funktion vom Typ DINT ermittelt den freien Speicherplatz des angegebenen Dateisystems.

Der Rückgabewert ist der freien Speicherplatz des angegenbenen Dateisystems.

Variable	Datentyp	Beschreibung
Drive	D_SysFile_Drive	Dateisystem

Enumeration
D_SysFile_Drive:
C_SysFile_Drive_Flash = 0 (* Flash_Filesystem – Laufwerk A: *)
C_SysFile_Drive_RAM = 1 (* RAM_Filesystem – Laufwerk B: *)

Beispiel: Auslesen des freien Speicherplatzes vom Flash-Filesystem.

a) in „ST“

```
FreeMem := SysFileGetFreeMem(D_SysFile_Drive_Flash);
```

b) in “FUP”

4.2.2 Die Bibliothek SysLibRtc

Hinweis: Diese Funktion wird nur von der Hardware-Variante @PLC-C202 unterstützt.

Diese Bibliothek bietet Funktionen zum Zugriff auf die Echtzeituhr der PLC.

- SysRtcCheckBattery Batteriestatus prüfen
- SysRtcGetHourMode RTC im 12h bzw. 24h Modus
- SysRtcGetTime RTC auslesen
- SysRtcSetTime RTC setzen

SysRtcCheckBattery (FUN)

Diese Funktion vom Type BOOL prüft den Zustand der Rechnerbatterie (Gold Cap), der ja Einfluss auf die korrekte Uhrzeitangabe hat.

Der Rückgabewert ist 0, wenn die Batterie nicht in Ordnung ist, bzw. 1, wenn sie ok ist.

Variable	Datentyp	Beschreibung
bDummy	BOOL	TRUE startet die Funktion

Hinweis: Eine Überprüfung der Batterie ist bei der @PLC-C20x zur Zeit noch nicht möglich, die Funktion liefert immer **OK** bzw. **1** zurück.

SysRtcGetHourMode (FUN)

Diese Funktion vom Typ BOOL dient zum auslesen des Anzeigemodus der Echtzeituhr des Rechners.

Rückgabewert 0 bedeutet, die Anzeige arbeitet im 12-Stunden Modus; Rückgabewert 1 bedeutet, die Anzeige arbeitet im 24-Stunden Modus.

Variable	Datentyp	Beschreibung
bDummy	BOOL	TRUE startet die Funktion

SysRtcGetTime (FUN)

Diese Funktion vom Typ DATE_AND_TIME gibt die aktuelle Echtzeit, die von der Rechneruhr gelesen wird, zurück.

Variable	Datentyp	Beschreibung
bDummy	BOOL	TRUE startet diese Funktion

SysRtcSetTime (FUN)

Diese Funktion vom Typ DATE_AND_TIME dient dazu, die Echtzeituhr des Rechners zu setzen. Als Rückgabewert gibt mit 1 oder 0 Auskunft über Erfolg oder Misserfolg der Aktion.

Variable	Datentyp	Beschreibung
ActDateAndTime	DATE_AND_TIME	Uhrzeit, auf die die Rechnerechtzeituhr gesetzt werden soll.

4.3. Feldbusse

4.3.1 Die Bibliothek Lib_ModbusRS232

Diese Bibliothek dient dazu, eine seriellen Modbus-Verbindung (RS232) aufzubauen und Statusmeldungen abzufragen.

- ModbusRS232_Init Aktiviert bzw. Deaktiviert einen Modbus-Slave

ModbusRS232_Init (FB)

Dieser Funktionsbaustein steuert einen Modbus-Slave an der angegebenen Schnittstelle. Über den „Active“-Eingang kann der Slave aktiviert bzw. deaktiviert werden. Die Parameterierung des Modbus-Slaves kann wahlweise über die Steuerungskonfiguration bzw. direkt über diesen Baustein erfolgen.

Soll die Parametrierung über die Steuerungskonfiguration erfolgen, muss als Wert für die „Baudrate“ der Wert „MB_RS232_Baud_None“ (= 0) angegeben werden. Alle weiteren Parameter außer „Active“ und „Interface“ werden dann ignoriert und es werden die Einstellungen der Steuerungskonfiguration zu dieser Schnittstelle verwendet.

Erfolgt die Parametrierung jedoch über den Baustein selbst, müssen alle Parameter mit korrekten Werten versehen sein. Die Einstellungen der Steuerungskonfiguration werden dann ignoriert.

Die Rückgabewerte „Ready“, „Err“ und „ErrNr“ liefern Statusinformationen über den Zustand des Modbus-Slaves

Variable	Datentyp	Beschreibung
Active	BOOL	Aktiviert bzw. Deaktiviert den Modbus-Slave
Interface	… _Interface	Serielle Schnittstelle (z.Zt. kann nur COM0 verwendet werden)
Baudrate	… _Baudrate	Baudrate der seriellen Schnittstelle
Parity	… _Parity	Paritätseinstellung der seriellen Schnittstelle
Stoppbits	… _Stopbit	Anzahl der bei der Übertragung verwendeten Stoppbits
TransmissionMode	… _TransMode	Modbus-Übertragungsmodus (RTU oder ASCII)
Address	BYTE	Modbus-Adresse
Ready	BOOL	Statusausgang, liefert Zustand des Modbus-Slaves
Err	BOOL	Fehlerausgang
ErrNr	DWORD	Fehlernummer (nur gültig, wenn Fehlerausgang aktiv ist)

Konstanten der Bausteine

ModbusRS232_Interface		
Konstante	Wert	Beschreibung
MB_RS232_COM0	0	Schnittstelle COM0
MB_RS232_COM1	1	Schnittstelle COM1 (z.Zt. nicht verfügbar)
MB_RS232_COM2	2	Schnittstelle COM2 (z.Zt. nicht verfügbar)
MB_RS232_COM3	3	Schnittstelle COM3 (z.Zt. nicht verfügbar)
MB_RS232_COM4	4	Schnittstelle COM4 (z.Zt. nicht verfügbar)
MB_RS232_COM5	5	Schnittstelle COM5 (z.Zt. nicht verfügbar)
MB_RS232_COM6	6	Schnittstelle COM6 (z.Zt. nicht verfügbar)
MB_RS232_COM7	7	Schnittstelle COM7 (z.Zt. nicht verfügbar)
MB_RS232_COM8	8	Schnittstelle COM8 (z.Zt. nicht verfügbar)
MB_RS232_COM9	9	Schnittstelle COM9 (z.Zt. nicht verfügbar)
MB_RS232_COM10	10	Schnittstelle COM10 (z.Zt. nicht verfügbar)
MB_RS232_COM11	11	Schnittstelle COM11 (z.Zt. nicht verfügbar)
MB_RS232_COM12	12	Schnittstelle COM12 (z.Zt. nicht verfügbar)
MB_RS232_COM13	13	Schnittstelle COM13 (z.Zt. nicht verfügbar)
MB_RS232_COM14	14	Schnittstelle COM14 (z.Zt. nicht verfügbar)
MB_RS232_COM15	15	Schnittstelle COM15 (z.Zt. nicht verfügbar)

ModbusRS232_Baudrate		
Konstante	Wert	Beschreibung
MB_RS232_Baud_None	0	Steuerungskonfiguration verwenden
MB_RS232_Baud_75	1	Baudrate 75
MB_RS232_Baud_150	2	Baudrate 150
MB_RS232_Baud_300	3	Baudrate 300
MB_RS232_Baud_600	4	Baudrate 600
MB_RS232_Baud_1200	5	Baudrate 1200
MB_RS232_Baud_2400	6	Baudrate 2400
MB_RS232_Baud_4800	7	Baudrate 4800
MB_RS232_Baud_7200	8	Baudrate 7200
MB_RS232_Baud_9600	9	Baudrate 9600
MB_RS232_Baud_14400	10	Baudrate 14400
MB_RS232_Baud_19200	11	Baudrate 19200
MB_RS232_Baud_28800	12	Baudrate 28800
MB_RS232_Baud_38400	13	Baudrate 38400
MB_RS232_Baud_57600	14	Baudrate 57600
MB_RS232_Baud_115200	15	Baudrate 115200

ModbusRS232_Parity		
Konstante	Wert	Beschreibung
MB_RS232_Parity_None	0	Keine Parität
MB_RS232_Parity_Even	1	Gerade Parität
MB_RS232_Parity_Odd	2	Ungerade Parität

ModbusRS232_Stopbit		
Konstante	Wert	Beschreibung
MB_RS232_Stopbit_1	0	1 Stoppbit
MB_RS232_Stopbit_2	1	2 Stoppbits

ModbusRS232_TransMode		
Konstante	Wert	Beschreibung
MB_RS232_TRANSMODE_RTU	0	Übertragungsmodus RTU
MB_RS232_TRANSMODE_ASCII	1	Übertragungsmodus ASCII

Fehlercodes der Bausteine

Fehlercode	Gruppe	Beschreibung
16#0001	Watchdog	Schreib-Watchdog hat zugeschlagen
16#0002	Watchdog	Lese-Watchdog hat zugeschlagen

4.3.2 Die Bibliothek Lib_ModbusTCP

Steht zur Zeit nicht zur Verfügung.

4.4. Teleservice und Kommunikation

4.4.1 Die Bibliothek Lib_SerialComm

Diese Bibliothek dient dazu, die serielle Schnittstelle direkt an der PLC bzw. weitere über @ctiveIO-Module zur Verfügung gestellte serielle Schnittstellen vom PLC-Programm aus anzusprechen. (ab Firmware-Version 3.1.x!)

- SerialComm_Init Aktivierung bzw. Deaktivierung der seriellen Schnittstelle
- SerialComm_ReadData Lesen der eingehenden Daten
- SerialComm_WriteData Schreiben der ausgehenden Daten

SerialComm_Init (FB)

Dieser Funktionsbaustein steuert die Aktivierung bzw. Deaktivierung und Parametrierung der jeweiligen seriellen Schnittstelle. Eine steigende Flanke am „Active“-Eingang öffnet und initialisiert die serielle Schnittstelle mit den eingestellten Daten. Bei Erfolg wird der „Ready“-Ausgang gesetzt und „Handle“ mit einem korrekten Handle versehen. Dieses Handle wird von den nachfolgend beschriebenen Funktionen zum Senden und Empfangen von Daten benötigt. Im Fehlerfall wird der „Err“-Ausgang gesetzt und „ErrNr“ enthält die dazugehörige Fehlernummer.

Erhält der „Active“-Eingang eine fallende Flanke, wird die serielle Schnittstelle wieder freigegeben bzw. ein anliegender Fehler wird quittiert. Anschließend steht die Schnittstelle wieder für andere Anwendungen zur Verfügung.

Variable	Datentyp	Beschreibung
Active	BOOL	Aktiviert bzw. Deaktiviert die serielle Schnittstelle
Port	… _Port	Verwendete serielle Schnittstelle
Baudrate	… _Baudrate	Baudrateneinstellunge
Databits	… _DataBits	Anzahl der Datenbits
Stoppbits	… _StoppBits	Anzahl der Stoppbits
Parity	… _Partity	Parityeinstellung
Handshake	… _Handshake	Handshakeeinstellung
Ready	BOOL	Schnittstelle bereit
Err	BOOL	Fehler bei Schnittstellen-Aktivierung
ErrNr	DWORD	Fehlernummer
Handle	DWORD	Schnittstellen-Handle

SerialComm_ReadData (FUN)

Mit dieser Funktion können eingehende Daten ausgelesen werden. Stehen keine Daten zur Verfügung, wird die Funktion sofort wieder verlassen und das Funktionsergebnis ist null.

Rückgabewert: Im Fehlerfall wird –1 zurückgegeben, ansonsten die Anzahl der gelesenen Zeichen.

Variable	Datentyp	Beschreibung
Handle	DWORD	Schnitstellen-Handle vom Init-Baustein
Buffer	POINTER TO BYTE	Zeiger auf den Eingangsbuffer
BufferSize	DWORD	Größe des Eingangsbuffers
SerialComm_ReadData	DINT	Anzahl gelesener Daten

SerialComm_WriteData (FUN)

Mit dieser Funktion können Daten über die serielle Schnittstelle ausgegeben werden.

Rückgabewert: Im Fehlerfall wird –1 zurückgegeben, ansonsten die Anzahl der gelesenen Zeichen.

Variable	Datentyp	Beschreibung
Handle	DWORD	Schnitstellen-Handle vom Init-Baustein
Buffer	POINTER TO BYTE	Zeiger auf den Ausgangsbuffer
BufferSize	DWORD	Anzahl zu schreibender Daten
SerialComm_WriteData	DINT	Anzahl geschriebene Daten

Konstanten der Bausteine

SerialComm_Port		
Konstante	Wert	Beschreibung
SerialComm_Port_COM0	0	serielle Schnittstelle der CPU
SerialComm_Port_COM1	1	@ctiveIO Serialmodule (z.Zt. nicht unterstützt) *)
SerialComm_Port_COM2	2	@ctiveIO Serialmodule (z.Zt. nicht unterstützt) *)
SerialComm_Port_COM3	3	@ctiveIO Serialmodule (z.Zt. nicht unterstützt) *)
SerialComm_Port_COM4	4	@ctiveIO Serialmodule (z.Zt. nicht unterstützt) *)
SerialComm_Port_COM5	5	@ctiveIO Serialmodule (z.Zt. nicht unterstützt) *)
SerialComm_Port_COM6	6	@ctiveIO Serialmodule (z.Zt. nicht unterstützt) *)
SerialComm_Port_COM7	7	@ctiveIO Serialmodule (z.Zt. nicht unterstützt) *)

*) ab Firmware-Version 3.1.x

SerialComm_Baudrate		
Konstante	Wert	Beschreibung
SerialComm_Baud_75	0	75 Baud
SerialComm_Baud_150	1	150 Baud
SerialComm_Baud_300	2	300 Baud
SerialComm_Baud_600	3	600 Baud
SerialComm_Baud_1200	4	1200 Baud
SerialComm_Baud_2400	5	2400 Baud
SerialComm_Baud_4800	6	4800 Baud
SerialComm_Baud_7200	7	7200 Baud
SerialComm_Baud_9600	8	9600 Baud
SerialComm_Baud_14400	9	14400 Baud
SerialComm_Baud_19200	10	19200 Baud
SerialComm_Baud_28800	11	28800 Baud
SerialComm_Baud_38400	12	38400 Baud

SerialComm_Baud_57600	13	57600 Baud
SerialComm_Baud_115200	14	115200 Baud

SerialComm_DataBits		
Konstante	Wert	Beschreibung
SerialComm_DataBits_5	0	5 Datenbits
SerialComm_DataBits_6	1	6 Datenbits
SerialComm_DataBits_7	2	7 Datenbits
SerialComm_DataBits_8	3	8 Datenbits

SerialComm_StoppBits		
Konstante	Wert	Beschreibung
SerialComm_StoppBit_1	0	1 Stoppbit
SerialComm_StoppBit_2	1	2 Stoppbits

SerialComm_Parity		
Konstante	Wert	Beschreibung
SerialComm_Parity_None	0	Keine Parität
SerialComm_Parity_Even	1	Gerade Parität
SerialComm_Parity_Odd	2	Ungerade Parität

SerialComm_Handshake		
Konstante	Wert	Beschreibung
SerialComm_Handshake_None	0	Handshake deaktiviert
SerialComm_Handshake_RTSCTS	1	Hardware-Handshake (RTS / CTS)
SerialComm_Handshake_Software	2	Software-Handshake

Fehlercodes der Bausteine

Fehlercode	Gruppe	Beschreibung
16#0001	Init	Öffnen der Schnittstelle fehlgeschlagen
16#0002	Init	Setzen der Baudrate fehlgeschlagen
16#0003	Init	Setzen Kommunikationsparameter fehlgeschlagen
16#0004	Init	Allokieren der Schnittstelle fehlgeschlagen (Schnittstelle wird gerade verwendet)
16#0005	Init	Unbekannte Baudrateeinstellung
16#0006	Init	Nicht unterstützte Anzahl von Datenbits
16#0007	Init	Nicht unterstützte Anzahl von Stoppbits
16#0008	Init	Unbekannte Paritätseinstellung
16#0009	Init	Unbekannte Handshake-Einstellung

4.4.2 Die Bibliothek Lib_PPP

Diese Bibliothek dient dazu, eine PPP-Verbindung (Point to Point Protocol) über eine serielle Verbindung oder ein Modem aufzubauen. Der Verbindungsaufbau kann hierbei aktiv (PPP-Client) oder passiv (PPP-Server) erfolgen.

- PPPClient_Modem_Init — Initiiert eine Client-Verbindung über ein Modem auf
- PPPClient_Serial_Init — Initiiert eine Client-Verbindung über eine seriellen Verbindung auf
- PPPServer_Modem_Init — Initialisiert einen Modem-PPP-Server

- PPPServer_Serial_Init Initialisiert einen Serial-PPP-Server
- PPPServer_GetConnectionCounter Anzahl der Verbindungsauf- bzw abbauten

PPPClient_Modem_Init (FB)

Dieser Funktionsbaustein steuert den Auf- und Abbau einer PPP-Clientverbindung. Über den „Active"-Eingang kann die PPP-Verbindung mit den angegebenen Parametern aktiviert oder deaktiviert werden.

Hinweis: Die Parameter dürfen nicht verändert werden, solange der Baustein nicht im Status **Offline** ist.

Die Rückgabewerte „State" und „ErrNr" liefern Statusinformationen über den Zustand der Verbindung.

Variable	Datentyp	Beschreibung
Active	BOOL	Eine steigende Flanke startet den Verbindungs-aufbau und eine fallende Flanke beendet die Verbindung
COMPort	PPP_Port	Gibt an, welcher Serial-Port des C20x-Moduls verwendet werden soll *)
Baudrate	PPP_Baudrate	Baudrate des gewählten Serial-Ports
ModemInit	STRING(80)	Initialisierung-String des Modems (AT-Befehle)
ModemNumber	STRING(80)	Telefonnummer des PPP-Servers
Username	STRING(80)	Benutzername des PPP-Servers
Password	STRING(80)	Passwort des PPP-Servers
Auth	PPP_Authentication	Authentifizierungsmethode des PPP-Servers
GatewayAddr	STRING(80)	Interne Adresse der PPP-Verbindung
GatewaySubnet	STRING(80)	Adressmaske
RoutingDest	STRING(80)	Adressbereich, der über die PPP-Verbindung erreicht werden soll.
RoutingMask	STRING(80)	Adressmaske
State	PPP_Status	Status der PPP-Verbindung
ErrNr	DWORD	Im Fehlerfall: Fehlercode

*) Serial Module auf @ctiveIO-Bus ab Firmware-Version 3.1.x

Beispiel in ST (Strukturierter Text)**:**

```
PPPClient1:              PPPClient_Modem_Init;
PPPClient1_Active:       BOOL := FALSE;
PPPClient1_State:        PPP_Status;
PPPClient1_ErrNr:        DWORD;
```

Beispiel in FUP (Funktionsplan)**:**

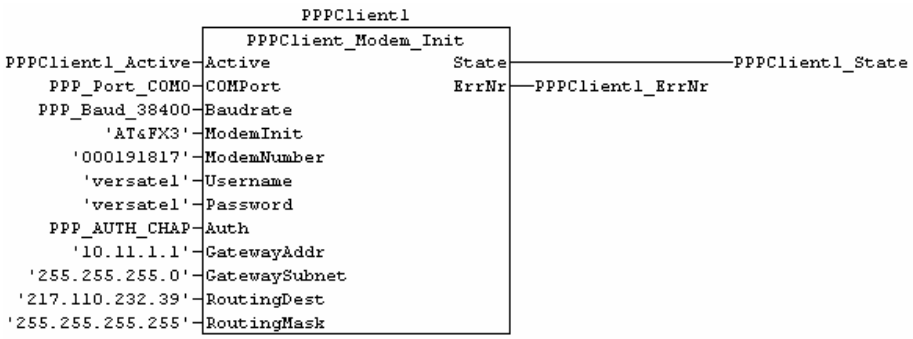

Abbildung 24: PPP Client Baustein

PPPClient_Serial_Init (FB)

Dieser Funktionsbaustein steuert den Auf- und Abbau einer PPP-Clientverbindung. Über den „Active“-Eingang kann die PPP-Verbindung mit den angegebenen Parametern aktiviert oder deaktiviert werden.

Hinweis: Die Parameter dürfen nicht verändert werden, solange der Baustein nicht im Status **Offline** ist.

Die Rückgabewerte „State“ und „ErrNr“ liefern Statusinformationen über den Zustand der Verbindung.

Variable	Datentyp	Beschreibung
Active	BOOL	Eine steigende Flanke startet den Verbindungs-aufbau und eine fallende Flanke beendet die Verbindung
COMPort	PPP_Port	Gibt an, welcher Serial-Port des C20x-Moduls verwendet werden soll
Baudrate	PPP_Baudrate	Baudrate des gewählten Serial-Ports
Username	STRING(80)	Benutzername des PPP-Servers
Password	STRING(80)	Passwort des PPP-Servers
Auth	PPP_Authentication	Authentifizierungsmethode des PPP-Servers
GatewayAddr	STRING(80)	Interne Adresse der PPP-Verbindung
GatewaySubnet	STRING(80)	Adressmaske
RoutingDest	STRING(80)	Adressbereich, der über die PPP-Verbindung erreicht werden soll.
RoutingMask	STRING(80)	Adressmaske
State	PPP_Status	Status der PPP-Verbindung
ErrNr	DWORD	Im Fehlerfall: Fehlercode

PPPServer_Modem_Init (FB)

Dieser Funktionsbaustein aktiviert oder deaktiviert einen PPP-Server an der angegebenen Serial-Schnittstelle. Über den „Active“-Eingang kann der PPP-Server aktiviert oder deaktiviert werden.

Hinweis: Die Parameter dürfen nicht verändert werden, solange der Baustein nicht im Status **Offline** ist.

Die Rückgabewerte „State“ und „ErrNr“ liefern Statusinformationen über den Zustand der Verbindung.

Variable	Datentyp	Beschreibung
Active	BOOL	Eine steigende Flanke aktiviert den PPP-Server und eine fallende Flanke deaktiviert ihn
COMPort	PPP_Port	Gibt an, welcher Serial-Port des C20x-Moduls verwendet werden soll
Baudrate	PPP_Baudrate	Baudrate des gewählten Serial-Ports
ModemInit	STRING(80)	Initialisierung-String des Modems (AT-Befehle)
RingCount	INT	Anzahl der „RING“ bis Modem abnimmt
Username	STRING(80)	Benutzername des PPP-Clients
Password	STRING(80)	Passwort des PPP-Clients
Auth	PPP_Authentication	Authentifizierungsmethode des PPP-Servers
GatewayAddr	STRING(80)	Interne Adresse der PPP-Verbindung
GatewaySubnet	STRING(80)	Adressmaske
ClientAddr	STRING(80)	Dem Client zuzuweisende IP-Adresse
RoutingDest	STRING(80)	Adressbereich, der über die PPP-Verbindung erreicht werden soll.
RoutingMask	STRING(80)	Adressmaske
State	PPP_Status	Status der PPP-Verbindung
ErrNr	DWORD	Im Fehlerfall: Fehlercode

Beispiel in ST:

```
PPPServer0:              PPPServer_Modem_Init;
PPPServer0_Active:      BOOL := FALSE;
PPPServer0_State:       PPP_Status := PPP_Status_Offline;
PPPServer0_ErrNr:       DWORD := 0;
```

Beispiel in FUP:

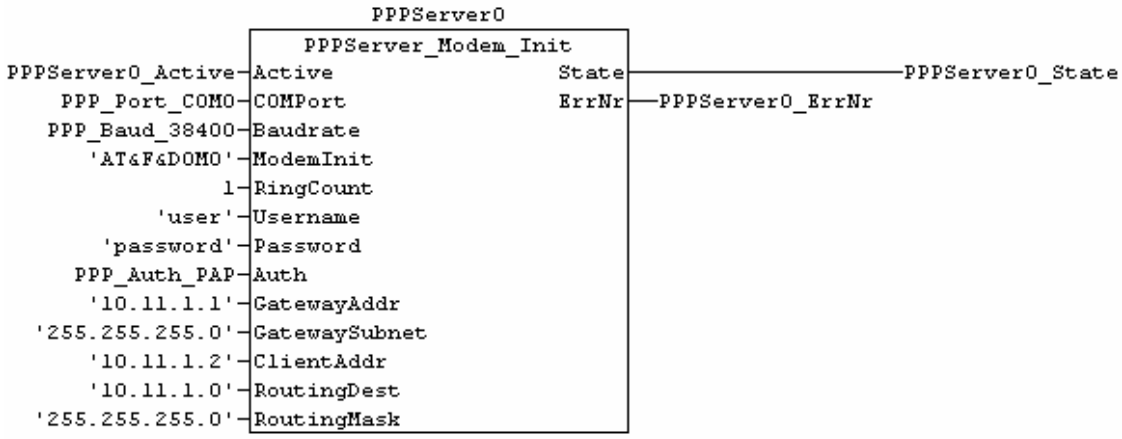

Abbildung 25: PPP Server 0 Baustein

PPPServer_Serial_Init (FB)

Dieser Funktionsbaustein aktiviert oder deaktiviert einen PPP-Server an der angegebenen Serial-Schnittstelle. Über den „Active“-Eingang kann der PPP-Server aktiviert oder deaktiviert werden.

Hinweis: Die Parameter dürfen nicht verändert werden, solange der Baustein nicht im Status **Offline** ist.

Die Rückgabewerte **„State“** und **„ErrNr“** liefern Statusinformationen über den Zustand der Verbindung.

Variable	Datentyp	Beschreibung
Active	BOOL	Eine steigende Flanke aktiviert den PPP-Server und eine fallende Flanke deaktiviert ihn
COMPort	PPP_Port	Gibt an, welcher Serial-Port des C20x-Moduls verwendet werden soll
Baudrate	PPP_Baudrate	Baudrate des gewählten Serial-Ports
Username	STRING(80)	Benutzername des PPP-Clients
Password	STRING(80)	Passwort des PPP-Clients
Auth	PPP_Authentication	Authentifizierungsmethode des PPP-Servers
GatewayAddr	STRING(80)	Interne Adresse der PPP-Verbindung
GatewaySubnet	STRING(80)	Adressmaske
ClientAddr	STRING(80)	Dem Client zuzuweisende IP-Adresse
RoutingDest	STRING(80)	Adressbereich, der über die PPP-Verbindung erreicht werden soll.
RoutingMask	STRING(80)	Adressmaske
State	PPP_Status	Status der PPP-Verbindung
ErrNr	DWORD	Im Fehlerfall: Fehlercode

PPPServer_GetConnectionCounter (FB)

Dieser Funktionsbaustein liefert Informationen darüber, wie oft eine (Modem-) Verbindung seit dem Öffnen der seriellen Schnittstelle auf- bzw. abgebaut wurde. Hiermit kann erkannt werden, ob Verbindungen hergestellt wurden, die nicht als PPP-Verbindungen identifiziert wurden (einfache Modemverbindungen mit Terminalprogramm, fehlerhafte PPP-Verbindungen, usw.). Nach Beendigung einer (erfolgreichen) PPP-Verbindung werden die Zähler zurückgesetzt.

Variable	Datentyp	Beschreibung
COMPort	PPP_Port	Gibt an, welcher Serial-Port des C20x-Moduls verwendet werden soll
clearCounter	BOOL	Setzt die Zähler zurück
counterConnect	DWORD	Anzahl der aufgebauten Verbindungen
counterDisconnect	DWORD	Anzahl der abgebauten Verbindungen

Konstanten der Bausteine

PPP_Port		
Konstante	Wert	Beschreibung
PPP_Port_COM0	0	Schnittstelle COM0
PPP_Port_COM1	1	Schnittstelle COM1 (z.Zt. nicht verfügbar)
PPP_Port_COM2	2	Schnittstelle COM2 (z.Zt. nicht verfügbar)
PPP_Port_COM3	3	Schnittstelle COM3 (z.Zt. nicht verfügbar)
PPP_Port_COM4	4	Schnittstelle COM4 (z.Zt. nicht verfügbar)
PPP_Port_COM5	5	Schnittstelle COM5 (z.Zt. nicht verfügbar)
PPP_Port_COM6	6	Schnittstelle COM6 (z.Zt. nicht verfügbar)
PPP_Port_COM7	7	Schnittstelle COM7 (z.Zt. nicht verfügbar)

PPP_Baudrate		
Konstante	Wert	Beschreibung
PPP_Baud_75	0	75 Baud
PPP_Baud_150	1	150 Baud
PPP_Baud_300	2	300 Baud
PPP_Baud_600	3	600 Baud
PPP_Baud_1200	4	1,2 kBaud
PPP_Baud_2400	5	2,4 kBaud
PPP_Baud_4800	6	4,8 kBaud
PPP_Baud_7200	7	7,2 kBaud
PPP_Baud_9600	8	9,6 kBaud
PPP_Baud_14400	9	14,4 kBaud
PPP_Baud_19200	10	19,2 kBaud
PPP_Baud_28800	11	28,8 kBaud
PPP_Baud_38400	12	38,4 kBaud
PPP_Baud_57600	13	57,6 kBaud
PPP_Baud_115200	14	115,2 kBaud

PPP_Authentication		
Konstante	Wert	Beschreibung
PPP_Auth_PAP	0	verwende PAP Authentifizierung
PPP_Auth_CHAP	1	verwende CHAP Authentifizierung

PPP_Status		
Konstante	Wert	Beschreibung
PPP_Status_Offline	0	Baustein befindet sich im Offline Mode, keine aktive Verbindung hergestellt
PPP_Status_Initializing	10	Baustein (und ggf. Modem) wird initialisiert
PPP_Status_Connecting	20	Baustein stellt Verbindung zur Gegenstelle her (Client)
PPP_Status_Waiting	30	Baustein wartet auf eingehende Verbindung (Server)
PPP_Status_Connected	40	Es besteht eine PPP-Verbindung zur Gegenstelle
PPP_Status_Disconnected	50	PPP-Verbindung wurde abgebrochen
PPP_Status_HangingUp	60	PPP-Verbindung wird beendet
PPP_Status_ShuttingDown	70	Baustein wird deaktiviert
PPP_Status_Error	99	Fehler (siehe Fehlercode)

Fehlercodes der Bausteine

Fehlercode	Gruppe	Beschreibung
16#0001	System	Speicher kann nicht zugewiesen werden
16#0002	System	Semaphore kann nicht erzeugt werden
16#0003	System	Thread kann nicht erzeugt werden
16#0004	System	Thread kann nicht beendet werden
16#0005	System	Unbekannte Baudrate
16#0010	AllocComm	Schnittstelle kann nicht verwendet werden
16#0011	AllocComm	Schnittstelle wurde in einem anderen Modi initialisiert (Serial-Mode, Modem-Mode)
16#0012	AllocComm	Gateway-Adresse hat sich geändert
16#0013	AllocComm	Gateway-Subnet-Mask hat sich geändert
16#0020	CreateDevice	PPP-Baustein konnte nicht erzeugt werden
16#0021	CreateDevice	Schnittstelle wird bereits von einem anderen PPPBaustein verwendet
16#0022	CreateDevice	Speicher kann nicht allokiert werden
16#0023	CreateDevice	Gateway-Adresse oder Subnet-Mask sind 0
16#0030	SetAuth	Authentifikation kann nicht eingestellt werden
16#0031	SetAuth	Schnittstelle ist nicht zur Verwendung als PPP-Port initialisiert
16#0032	SetAuth	Unbekannter Authentifikationsmethode
16#0033	SetAuth	Kein PAP-Passwort angegeben
16#0034	SetAuth	Kein PAP-Benutzername angegeben
16#0035	SetAuth	Kein CHAP-Name angegeben
16#0036	SetAuth	Kein CHAP-Secret angegeben
16#0037	SetAuth	User konnte nicht in die User-Datenbank aufgenommen warden
16#0040	VJCompress	Van Jacobsen Komprimierung konnte nicht eingestellt werden
16#0041	VJCompress	Schnittstelle ist nicht zur Verwendung als PPP-Port initialisiert
16#0050	ModemDial	Telefonnummer konnte nicht eingetragen werden
16#0051	ModemDial	Telefonnummer ist zu lang
16#0060	ModemInit	Modem konnte nicht initialisiert werden
16#0061	ModemInit	Schnittstelle ist nicht zur Verwendung als PPP-Port initialisiert
16#0062	ModemInit	Schnittstelle konnte nicht geöffnet werden
16#0063	ModemInit	Baudrate konnte nicht gesetzt werden
16#0064	ModemInit	Initialisierung des Modems ist fehlgeschlagen
16#0065	ModemInit	Init-String ist zu lang
16#0066	ModemInit	Thread konnte nicht gestartet werden
16#0067	ModemInit	Unbekannter PPP-Modus

16#0068	ModemInit	Verbindung konnte nicht hergestellt werden
16#0070	AddRoute	Route konnte nicht eingetragen warden
16#0071	AddRoute	IP-Stack Tabelle ist voll
16#0072	AddRoute	Schnittstelle ist nicht zur Verwendung als PPP-Port initialisiert
16#0080	CheckLink	PPP-Verbindung konnte nicht hergestellt werden
16#0081	CheckLink	Schnittstelle ist nicht zur Verwendung als PPP-Port initialisiert
16#0090	ModemClose	Modem-Verbindung konnte nicht beendet werden
16#0091	ModemClose	Schnittstelle ist nicht zur Verwendung als PPP-Port initialisiert
16#0092	ModemClose	Es wurde kein Modem erkannt (Modem nicht angeschlossen)
16#0093	ModemClose	Schnittstelle wurde nicht geöffnet
16#0094	ModemClose	Schnittstelle konnte nicht geschlossen warden
16#0095	ModemClose	Modem wurde nicht initialisiert
16#00A0	SetPeerAddr	Client-Adresse konnte nicht gesetzt werden
16#00A1	SetPeerAddr	Schnittstelle ist nicht zur Verwendung als PPP-Port initialisiert
16#00A2	SetPeerAddr	Route konnte nicht zur Routing Tabelle hinzugefügt warden
16#00B0	ModemRings	Anzahl der „RING“ konnte nicht eingestellt werden
16#00B1	ModemRings	Anzahl der „RING“ zu groß
16#00C0	SerialInit	Schnittstelle konnte nicht initialisiert werden
16#00C1	SerialInit	Schnittstelle ist nicht zur Verwendung als PPP-Port initialisiert
16#00C2	SerialInit	Schnittstelle konnte nicht geöffnet werden
16#00C3	SerialInit	Baudrate konnte nicht eingestellt werden
16#00C4	SerialInit	Thread konnte nicht gestartet werden
16#00C5	SerialInit	Unbekannter PPP-Modus
16#00D0	SerialClose	Serial-Verbindung konnte nicht beendet werden
16#00D1	SerialClose	Schnittstelle ist nicht zur Verwendung als PPP-Port initialisiert
16#00D2	SerialClose	Schnittstelle wurde nicht geöffnet
16#00D3	SerialClose	Schnittstelle konnte nicht geschlossen werden

4.4.3 Die Bibliothek Lib_FTP

Diese Bibliothek beinhaltet Funktionen zur Statusabfrage des FTP-Servers auf dem C20x-Modul und zur Parametrierung des externen Zugriffs (Username und Passwort).

- ftpServer_GetUserID Auslesen des eingestellten Usernamens
- ftpServer_GetUserPassword Auslesen des eingestellten Passwortes
- ftpServer_SetUserID Setzen des Usernamens
- ftpServer_SetUserPassword Setzen des Passwortes
- ftpServer_Status Statusabfrage des FTP-Servers
- ftpClient_Connection Aufbau einer Verbindung zum ftp-Server
- ftpClient_RetrieveFile Download einer Datei
- ftpClient_StoreFile Upload einer Datei
- ftpClient_DeleteFile Löschen einer Datei
- ftpClient_GetWorkingDir Auslesen des aktuellen Verzeichnisses
- ftpClient_SetWorkingDir Setzen des aktuellen Verzeichnisses

ftpServer_GetUserID (FUN)

Diese Funktion von Typ STRING dient zum Auslesen des eingestellten Benutzernamens (Zugriffsebene 3).

Der Rückgabewert ist der eingestellte Benutzername.

Variable	Datentyp	Beschreibung
bDummy	BOOL	Dummy-Wert

ftpServer_GetUserPassword (FUN)

Diese Funktion von Typ STRING dient zum Auslesen des eingestellten Password (Zugriffsebene 3).

Der Rückgabewert ist das eingestellte Passwort.

Variable	Datentyp	Beschreibung
bDummy	BOOL	Dummy-Wert

ftpServer_SetUserID (FUN)

Diese Funktion vom Typ BOOL dient dem Setzen des Benutzernamen (Zugriffsebene 3). Die Änderung wird sofort durchgeführt und kann zur nächsten Anmeldung verwendet werden. Der Benutzername kann maximal 15 Zeichen lang sein.

Der Rückgabewert ist 1 (OK) oder 0 (Fehler).

Variable	Datentyp	Beschreibung
ID	STRING(80)	Neuer Benutzername

ftpServer_SetUserPassword (FUN)

Diese Funktion vom Typ BOOL dient dem Setzen des Passwortes (Zugriffsebene 3). Die Änderung wird sofort durchgeführt und kann zur nächsten Anmeldung verwendet werden. Das Password kann maximal 15 Zeichen lang sein.

Der Rückgabewert ist 1 (ok) oder 0 (Fehler).

Variable	Datentyp	Beschreibung
Password	STRING(80)	Neues Passwort

ftpServer_Status (FB)

Dieser Funktionsbaustein liefert Informationen über den aktuellen Status des FTP-Servers.

Variable	Datentyp	Beschreibung
Active	BOOL	FTP-Server wurde gestartet
CtrlConn	BOOL	Es besteht eine Control-Verbindung (Benutzer ist eingeloggt und Verbindung wurde noch nicht wieder getrennt)
DataConn	BOOL	Es besteht eine Daten-Verbindung (Daten werden empfangen oder gesendet)
ActiveUserID	STRING(80)	Benutzername des eingeloggten Benutzers
LastAction	FTPSERVER_ACTION	Zuletzt durchgeführte Aktion (siehe unten)
FileName	STRING(80)	Dateiname, auf den die Aktion durchgeführt wurde
RcvBytes	DWORD	Empfangene Bytes seit Start des FTP-Servers
SndBytes	DWORD	Gesendete Bytes seit Start des FTP-Servers

ftpClient_Connection (FB)

Dieser Funktionsblock stellt die Verbindung zu einem ftp-Server her. Über den „Active“-Eingang kann der Verbindungsaufbau zum ftp-Server gesteuert werden. Die Serveradresse wird über die Variable „ServerAddr“ angegeben. Hierbei kann keine Name angegeben werden, sondern es ist erforderlich, die IP-Adresse in der Form „192.168.1.1“ anzugeben. Anschließend werden noch zwei Passwort/Benutzernamen Paare benötigt: „Username“ und „Password“ für den Zugang zum ftp-Server und „LocalUsername“ und „LocalPassword“ für den Schreibzugriff auf das Dateisystem des Moduls.

Variable	Datentyp	Beschreibung
Active	BOOL	Verbinden bzw. Trennen der Verbindung zum ftp-Server
ServerAddr	STRING(20)	IP-Adresse des Servers
Username	STRING(80)	Username (ftp-Server)

Password	STRING(80)	Passwort (ftp-Server)
LocalUsername	STRING(80)	Username (@PLC-C20x – Modul)
LocalPassword	STRING(80)	Passwort (@PLC-C20x – Modul)
State	FTPCLIENT_CONNSTATUS	Status der Verbindung (s.u.)
Handle	DWORD	Handle der Verbindung

Mit Hilfe der Konstanten „FTPCLIENT_CONNSTATUS“ kann der Status der Verbindung abgefragt. Beim Aktivieren der Verbindung (steigende Flanke „Active“-Eingang) wechselt der Verbindungsstatus von IDLE zu CONNECTING. Nach der Herstellung der Verbindung wird der Status auf CONNECTED gesetzt. Im Fehlerfalle wird der Status FAILURE zurückgeliefert.

Beim Deaktivieren der Verbindung (fallende Flanke „Active“-Eingang) wechselt der Status von CONNECTED zu DISCONNECTING. Nach einer erfolgreichen Trennung steht der Status auf DISCONNECTED. Im Fehlerfalle wird der Status FAILURE zurückgeliefert.

ftpClient_RetrieveFile (FB)

Dieser Funktionsblock lädt eine Datei von einem entfernten ftp-Server auf das Modul. Hierzu muß zuvor mit Hilfe von „ftpClient_Connection“ eine Verbindung hergestellt worden sein. Die Datei kann während der Übertragung umbenannt werden.

Variable	Datentyp	Beschreibung
Active	BOOL	Steigende Flanke aktiviert den Datei-Download
Handle	DWORD	Handle der ftp-Verbindung
SrcFilename	STRING(80)	Dateiname auf dem ftp-Server
DstFilename	STRING(80)	Dateiname im lokalen Dateisystem
State	FTPCLIENT_STATUS	Status der Verarbeitung (s.u.)

Die Konstante FTPCLIENT_STATUS liefert Informationen zum Status der Verarbeitung. Befindet sich der „Active“-Eingang im Ruhezustand (Low-Pegel), wird als Status IDLE zurückgeliefert. Wird der Funktionsblock aktiviert (steigende Flanke des „Active“-Eingangs), ändert sich der Status zu WAITING und anschließend PROCESSING. Wurde die Verarbeitung erfolgreich abgeschlossen, wird dies mit dem Status SUCCESS angezeigt. Im Fehlerfalle wird FAILURE zurückgeliefert. Wurde entweder der Status SUCCESS oder FAILURE erreicht, kann der „Active“-Eingang wieder zurückgesetzt werden.

ftpClient_StoreFile (FB)

Dieser Funktionsblock überträgt eine Datei von dem Modul auf einen entfernten ftp-Server. Hierzu muß zuvor mit Hilfe von „ftpClient_Connection“ eine Verbindung hergestellt worden sein. Die Datei kann während der Übertragung umbenannt werden.

Variable	Datentyp	Beschreibung
Active	BOOL	Steigende Flanke aktiviert den Datei-Upload
Handle	DWORD	Handle der ftp-Verbindung
SrcFilename	STRING(80)	Dateiname im lokalen Dateisystem
DstFilename	STRING(80)	Dateiname auf dem ftp-Server
State	FTPCLIENT_STATUS	Status der Verarbeitung (s.o.)

ftpClient_DeleteFile (FB)

Dieser Funktionsblock löscht eine Datei auf einem entfernten ftp-Server. Hierzu muß zuvor mit Hilfe von „ftpClient_Conenction“ eine Verbindung hergestellt worden sein.

Variable	Datentyp	Beschreibung
Active	BOOL	Steigende Flanke aktiviert die Datei-Löschung
Handle	DWORD	Handle der ftp-Verbindung
Filename	STRING(80)	Dateiname auf dem ftp-Server
State	FTPCLIENT_STATUS	Status der Verarbeitung (s.o.)

ftpClient_GetWorkingDir (FB)

Dieser Funktionsblock liefert den Bezeichner des aktuellen Arbeitsverzeichnisses auf dem ftp-Server für die bestehende Verbindung.

Variable	Datentyp	Beschreibung
Active	BOOL	Steigende Flanke startet die Abfrage
Handle	DWORD	Handle der ftp-Verbindung
State	FTPCLIENT_STATUS	Status der Verarbeitung (s.o.)
Dirname	STRING(80)	Aktuelles Arbeitsverzeichnis

ftpClient_SetWorkingDir (FB)

Dieser Funktionsblock setzt das aktuelle Arbeitsverzeichnis auf dem ftp-Server für die bestehende Verbindung.

Variable	Datentyp	Beschreibung
Active	BOOL	Steigende Flanke startet die Abfrage
Handle	DWORD	Handle der ftp-Verbindung
Dirname	STRING(80)	Neues Arbeitsverzeichnis
State	FTPCLIENT_STATUS	Status der Verarbeitung (s.o.)

Konstanten der Bausteine

FTPSERVER_ACTION		
Konstante	Wert	Beschreibung
FTPSERVER_ACTION_NoAction	0	Keine Aktion anliegend
FTPSERVER_ACTION_List	1	Ausgabe der Verzeichnisstruktur (DIR)
FTPSERVER_ACTION_RetrieveFile	2	Senden einer Datei (GET)
FTPSERVER_ACTION_StoreFile	3	Empfangen einer Datei (PUT)
FTPSERVER_ACTION_DeleteFile	4	Löschen einer Datei (DEL)

FTPCLIENT_CONNSTATUS		
Konstante	Wert	Beschreibung
FTPCLIENT_CONNSTATUS_IDLE	0	ftp-Client befindet sich im Ruhezustand (Keine Verbindung aktiv)
FTPCLIENT_CONNSTATUS_CONNECTING	1	Verbindungsaufbau zum ftp-Server
FTPCLIENT_CONNSTATUS_CONNECTED	2	Verbindung wurde hergestellt
FTPCLIENT_CONNSTATUS_DISCONNECTING	3	Verbindungsabbau
FTPCLIENT_CONNSTATUS_DISCONNECTED	4	Verbindung wurde abgebaut
FTPCLIENT_CONNSTATUS_FAILURE	5	Fehler bei Verbindungsaufbau bzw. -abbau

FTPCLIENT_STATUS		
Konstante	Wert	Beschreibung
FTPCLIENT_STATUS_IDLE	0	Datenübertragung ist nicht aktiv
FTPCLIENT_STATUS_WAITING	1	Warte auf Beginn der Datenübertragung
FTPCLIENT_STATUS_PROCESSING	2	Dateiübertragung wird durchgeführt
FTPCLIENT_STATUS_SUCCESS	3	Dateiübertragung wurde erfolgreich abgeschlossen
FTPCLIENT_STATUS_FAILURE	4	Dateiübertragung konnte nicht durchgeführt werden

4.5. Sonstige Bibliotheken

4.5.1 Die Bibliothek Lib_C200_Utils

Die Bibliothek Lib_C200_Utils stellt eine Reihe von Funktionen und Funktionsblöcken zur Verfügung, die sich in mehrere Untergruppen aufteilen lassen: Hilfsfunktionen der @ctive-Bus zum Zugriff auf die I/O-Daten und die Konfiguration und weiterer Routinen. Diese sind nachfolgend näher erläutert.

Zum @ctive-Bus befinden sich eine Reihe von Funktionen, die dem Lesen und Schreiben einzelnen Werte aus dem Prozessabild dienen. Der Grund hierzu liegt darin, dass das rohe Prozessabbild des @ctive-Bus im Intel-Format (Little Endian, 16Bit Worte) eingelesen wird. Mit Hilfe dieser Funktionen wird die Konvertierung der Daten vorgenommen.

4.5.1.1 Hilfsfunktionen @ctiveIO

In dieser Untergruppe befinden sich Funktionen, mit denen sich allgemeine Informationen über die @ctive-Bus Konfiguration auslesen lassen.

- C200_ActiveBus_GetModuleCount
- C200_ActiveBus_GetModuleID

C200_ActiveBus_GetModuleCount (FUN)

Diese Funktion liefert die Anzahl der angeschlossenen und erkannte Prints der @ctive-Bus Konfiguration.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
C200_ActiveBus_GetModuleCount	INT	Anzahl der angeschlossenen Prints

C200_ActiveBus_GetModuleID (FUN)

Diese Funktion liefert die Modul-ID des an der angegebenen Position befindlichen Prints. Ist die Positionsangabe außerhalb des zulässigen Bereichs, wird -1 zurückgegeben.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Position des Prints (0-basierend)
C200_ActiveBus_GetModuleID	INT	Modul-ID des Prints

4.5.1.2 Hilfsfunktionen @ctiveIO Automatische Konfiguration

In dieser Untergruppe befinden sich Funktionen, die Informationen zur automatischen Konfiguration bieten bzw. Zugriff auf die Daten der Module erlauben.

- C200_ActiveBus_GetInputPA
- C200_ActiveBus_GetOutputPA
- C200_ActiveBus_GetModuleInputOffset
- C200_ActiveBus_GetModuleInputSize
- C200_ActiveBus_GetModuleOutputOffset
- C200_ActiveBus_GetModuleOutputSize

- C200_ActiveBus_GetConfig
- C200_ActiveBus_SetConfig
- C200_ActiveBus_WriteConfig

- C200_ActiveBus_GetData_BOOL
- C200_ActiveBus_GetData_BYTE
- C200_ActiveBus_GetData_WORD

- C200_ActiveBus_GetData_DWORD
- C200_ActiveBus_SetData_BOOL
- C200_ActiveBus_SetData_BYTE

- C200_ActiveBus_SetData_WORD
- C200_ActiveBus_SetData_DWORD

- C200_ActiveBus_GetModuleData_BOOL
- C200_ActiveBus_GetModuleData_BYTE
- C200_ActiveBus_GetModuleData_WORD
- C200_ActiveBus_GetModuleData_DWORD
- C200_ActiveBus_SetModuleData_BOOL
- C200_ActiveBus_SetModuleData_BYTE
- C200_ActiveBus_SetModuleData_WORD
- C200_ActiveBus_SetModuleData_DWORD

C200_ActiveBus_GetInputPA (FUN)

Diese Funktion liefert einen Zeiger auf den Anfang des Eingangsprozessabbildes.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
… _GetInputPA	POINTER TO WORD	Zeiger auf das Prozessabbild

C200_ActiveBus_GetOutputPA (FUN)

Diese Funktion liefert einen Zeiger auf den Anfang des Ausgangsprozessabbildes.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
... _GetOutputPA	POINTER TO WORD	Zeiger auf das Prozessabbild

C200_ActiveBus_GetModuleInputOffset (FUN)

Diese Funktion liefert den Offset zum Anfang des Eingangsprozessabbild des an der angegebenen Position befindlichen Moduls zurück. Ist die Position außerhalb des zulässigen Bereichs, wird -1 zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Position des Moduls
… _GetModuleInputOffset	INT	Offset zum Eingangsprozessabbild

C200_ActiveBus_GetModuleInputSize (FUN)

Diese Funktion liefert die Eingangsdatengröße (in Bytes) des an der angegebenen Position befindlichen Moduls zurück. Ist die Position außerhalb des zulässigen Bereichs, wird -1 zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Position des Moduls
… _GetModuleInputSize	INT	Größe der Eingangsdaten (in Bytes)

C200_ActiveBus_GetModuleOutputOffset (FUN)

Diese Funktion liefert den Offset zum Anfang des Ausgangsprozessabbild des an der angegebenen Position befindlichen Moduls zurück. Ist die Position außerhalb des zulässigen Bereichs, wird -1 zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Position des Moduls
... _GetModuleOutputOffset	INT	Offset zum Ausgangsprozessabbild

C200_ActiveBus_GetModuleOutputSize (FUN)

Diese Funktion liefert die Ausgangsdatengröße (in Bytes) des an der angegebenen Position befindlichen Moduls zurück. Ist die Position außerhalb des zulässigen Bereichs, wird -1 zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Position des Moduls
... _GetModuleOutputSize	INT	Größe der Ausgangsdaten (in Bytes)

C200_ActiveBus_GetConfig (FUN)

Diese Funktion liest die Parameterdaten (Konfigurationsdaten) des angegebenen Moduls. Es werden die Daten der gerade aktiven @ctiveBus-Konfiguration gelesen. Konnten die Parameterdaten gelesen werden, wird TRUE zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Position des Moduls
ModulID	INT	ID die das Modul haben soll oder –1 wenn ID nicht überprüft werden soll
ParamData	POINTER TO BYTE	Zeiger auf die Parameterdaten
ParamSize	INT	Größe der Parameterdaten
... _GetConfig	BOOL	Leseaktion erfolgreich durchgeführt

C200_ActiveBus_SetConfig (FUN)

Diese Funktion setzt die Parameterdaten (Konfigurationsdaten) des angegebenen Moduls. Es werden die Daten der gespeicherten Konfiguration überschrieben, die mittels C200_ActiveBus_WriteConfig aktiviert werden können. Konnten die Parameterdaten geschrieben werden, wird TRUE zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Position des Moduls
ModulID	INT	ID die das Modul haben soll oder –1 wenn ID nicht überprüft werden soll
ParamData	POINTER TO BYTE	Zeiger auf die Parameterdaten
ParamSize	INT	Größe der Parameterdaten
... _GetConfig	BOOL	Schreibaktion erfolgreich durchgeführt

C200_ActiveBus_WriteConfig (FUN)

Aktiviert die zuvor mit C200_ActiveBus_SetConfig eingetragene Konfiguration und schreibt sie in die @ctiveBus-Module. Wurde die Konfiguration erfolgreich geschrieben, wird TRUE zurückgeliefert.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummywert
... _WriteConfig	BOOL	Aktivierung erfolgreich durchgeführt

C200_ActiveBus_GetData_BOOL (FUN)

Diese Funktion liefert den Wert des angegebenen Bits zurück. Hierzu wird ein Zeiger auf einen Speicherbereich (das Eingangsprozessabbild) und ein Bit-Offset angegeben.

Hinweis: Es wird davon ausgegangen, dass der Speicherbereich im Intel-Format (Little Endian) vorliegt.

Variable	Datentyp	Beschreibung
Addr	POINTER TO WORD	Zeiger auf den Speicherbereich
Offset	INT	Bit-Offset innerhalb des Bereichs
... _GetData_BOOL	BOOL	Wert des angegebenen Bits

C200_ActiveBus_GetData_BYTE (FUN)

Diese Funktion liefert den Wert des angegebenen Bytes zurück. Hierzu wird ein Zeiger auf einen Speicherbereich (das Eingangsprozessabbild) und ein Byte-Offset angegeben.

Hinweis: Es wird davon ausgegangen, dass der Speicherbereich im Intel-Format (Little Endian) vorliegt.

Variable	Datentyp	Beschreibung
Addr	POINTER TO WORD	Zeiger auf den Speicherbereich
Offset	INT	Byte-Offset innerhalb des Bereichs
... _GetData_BYTE	BYTE	Wert des angegebenen Bytes

C200_ActiveBus_GetData_WORD (FUN)

Diese Funktion liefert den Wert des angegebenen Wortes zurück. Hierzu wird ein Zeiger auf einen Speicherbereich (das Eingangsprozessabbild) und ein Byte-Offset angegeben.

Hinweis: Es wird davon ausgegangen, dass der Speicherbereich im Intel-Format (Little Endian) vorliegt.

Variable	Datentyp	Beschreibung
Addr	POINTER TO WORD	Zeiger auf den Speicherbereich
Offset	INT	Byte-Offset innerhalb des Bereichs
... _GetData_WORD	WORD	Wert des angegebenen Wortes

C200_ActiveBus_GetData_DWORD (FUN)

Diese Funktion liefert den Wert des angegebenen Doppelwortes zurück. Hierzu wird ein Zeiger auf einen Speicherbereich (das Eingangsprozessabbild) und ein Byte-Offset angegeben.

Hinweis: Es wird davon ausgegangen, dass der Speicherbereich im Intel-Format (Little Endian) vorliegt.

Variable	Datentyp	Beschreibung
Addr	POINTER TO WORD	Zeiger auf einen Speicherbereich
Offset	INT	Byte-Offset innerhalb des Bereichs
... _GetData_DWORD	DWORD	Wert des angegebenen Doppelwortes

C200_ActiveBus_SetData_BOOL (FUN)

Diese Funktion schreibt den angegeben Wert in den Speicherbereich. Hierzu wird ein Zeiger auf einen Speicherbereich (das Ausgangsprozessabbild) und ein Bit-Offset angegeben. Der vorherige Wert aus dem Speicherbereich wird als Rückgabewert zurückgeliefert.

Hinweis: Es wird davon ausgegangen, dass der Speicherbereich im Intel-Format (Little Endian) vorliegt.

Variable	Datentyp	Beschreibung
Addr	POINTER TO WORD	Zeiger auf einen Speicherbereich
Offset	INT	Bit-Offset innerhalb des Bereichs
Data	BOOL	Neuer Wert des Bits
… _SetData_BOOL	BOOL	Vorheriger Wert des Bits

C200_ActiveBus_SetData_BYTE (FUN)

Diese Funktion schreibt den angegebenen Wert in den Speicherbereich. Hierzu wird ein Zeiger auf einen Speicherbereich (das Ausgangsprozessabbild) und ein Byte-Offset angegeben. Der vorherige Wert aus dem Speicherbereich wird als Rückgabewert zurückgeliefert.

Hinweis: Es wird davon ausgegangen, dass der Speicherbereich im Intel-Format (Little Endian) vorliegt.

Variable	Datentyp	Beschreibung
Addr	POINTER TO WORD	Zeiger auf einen Speicherbereich
Offset	INT	Byte-Offset innerhalb des Bereichs
Data	BYTE	Neuer Wert des Bytes
… _SetData_BYTE	BYTE	Vorheriger Wert des Bytes

C200_ActiveBus_SetData_WORD (FUN)

Diese Funktion schreibt den angegebenen Wert in den Speicherbereich. Hierzu wird ein Zeiger auf einen Speicherbereich (das Ausgangsprozessabbild) und ein Byte-Offset angegeben. Der vorherige Wert aus dem Speicherbereich wird als Rückgabewert zurückgeliefert.

Hinweis: Es wird davon ausgegangen, dass der Speicherbereich im Intel-Format (Little Endian) vorliegt.

Variable	Datentyp	Beschreibung
Addr	POINTER TO WORD	Zeiger auf den Speicherbereich
Offset	INT	Byte-Offset innerhalb des Bereichs
Data	WORD	Neuer Wert des Wortes
… _SetData_WORD	WORD	Vorheriger Wert des Wortes

C200_ActiveBus_SetData_DWORD (FUN)

Diese Funktion schreibt den angegebenen Wert in den Speicherbereich. Hierzu wird ein Zeiger auf einen Speicherbereich (das Ausgangsprozessabbild) und ein Byte-Offset angegeben. Der vorherige Wert aus dem Speicherbereich wird als Rückgabewert zurückgeliefert.

Hinweis: Es wird davon ausgegangen, dass der Speicherbereich im Intel-Format (Little Endian) vorliegt.

Variable	Datentyp	Beschreibung
Addr	POINTER TO WORD	Zeiger auf den Speicherbereich
Offset	INT	Byte-Offset innerhalb des Bereichs
Data	DWORD	Neuer Wert des Doppelwortes
... _SetData_DWORD	DWORD	Vorheriger Wert des Doppelwortes

C200_ActiveBus_GetModuleData_BOOL (FUN)

Diese Funktion liefert den Wert eines Bits des angegebenen Moduls zurück. Hierzu wird der Modul-Index und ein Bit-Offset angegeben.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Bit-Offset innerhalb des ausgewählten Moduls
… _GetModuleData_BOOL	BOOL	Wert des angegebenen Bits

C200_ActiveBus_GetModuleData_BYTE (FUN)

Diese Funktion liefert den Wert eines Bytes des angegebenen Moduls zurück. Hierzu wird der Modul-Index und ein Byte-Offset angegeben.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
... _GetModuleData_BYTE	BYTE	Wert des angegebenen Bytes

C200_ActiveBus_GetModuleData_WORD (FUN)

Diese Funktion liefert den Wert eines Wortes des angegebenen Moduls zurück. Hierzu wird der Modul-Index und ein Byte-Offset angegeben.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
... _GetModulData_WORD	WORD	Wert des angegebenen Wortes

C200_ActiveBus_GetModuleData_DWORD (FUN)

Diese Funktion liefert den Wert eines Doppelwortes des angegebenen Moduls zurück. Hierzu wird der Modul-Index und ein Byte-Offset angegeben.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
... _GetModuleData_DWORD	DWORD	Wert des angegebenen Doppelwortes

C200_ActiveBus_SetModuleData_BOOL (FUN)

Diese Funktion schreibt den angegebenen Wert an die vorgegebene Stelle des Moduls. Hierzu wird der Modul-Index und ein Bit-Offset angegeben. Der vorherige Wert aus dem Modul wird als Rückgabewert zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Bit-Offset innerhalb des ausgewählten Moduls
Data	BOOL	Neuer Wert des Bits
... _SetModuleData_BOOL	BOOL	Vorheriger Wert des Bits

C200_ActiveBus_SetModuleData_BYTE (FUN)

Diese Funktion schreibt den angegebenen Wert an die vorgegebene Stelle des Moduls. Hierzu wird der Modul-Index und ein Byte-Offset angegeben. Der vorherige Wert aus dem Modul wird als Rückgabewert zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
Data	BYTE	Neuer Wert des Bytes
... _SetModuleData_BYTE	BYTE	Vorheriger Wert des Bytes

C200_ActiveBus_SetModuleData_WORD (FUN)

Diese Funktion schreibt den angegebenen Wert an die vorgegebene Stelle des Moduls. Hierzu wird der Modul-Index und ein Byte-Offset angegeben. Der vorherige Wert aus dem Modul wird als Rückgabewert zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
Data	WORD	Neuer Wert des Wortes
... _SetModuleData_WORD	WORD	Vorheriger Wert des Wortes

C200_ActiveBus_SetModuleData_DWORD (FUN)

Diese Funktion schreibt den angegebenen Wert an die vorgegebene Stelle des Moduls. Hierzu wird der Modul-Index und ein Byte-Offset angegeben. Der vorherige Wert aus dem Modul wird als Rückgabewert zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
Data	DWORD	Neuer Wert des Doppelwortes
... _SetModuleData_DWORD	DWORD	Vorheriger Wert des Doppelwortes

4.5.1.3 Hilfsfunktionen @ctiveIO Automatische Konfiguration (sortiert)

In dieser Untergruppe befinden sich weitere Funktionen, die Informationen zur automatischen Konfiguration bieten. Diese Informationen beziehen sich allerdings auf die zweite automatische Konfigurationsart: Automatische Konfiguration (sortiert).

- C200_ActiveBus_GetDigitalInputCount
- C200_ActiveBus_GetDigitalInputPA
- C200_ActiveBus_GetDigitalInputPAOffset
- C200_ActiveBus_GetDigitalOutputCount
- C200_ActiveBus_GetDigitalOutputPA
- C200_ActiveBus_GetDigitalOutputPAOffset
- C200_ActiveBus_GetAnalogInputCount
- C200_ActiveBus_GetAnalogInputPA
- C200_ActiveBus_GetAnalogInputPAOffset
- C200_ActiveBus_GetAnalogOutputCount
- C200_ActiveBus_GetAnalogOutputPA
- C200_ActiveBus_GetAnalogOutputPAOffset

C200_ActiveBus_GetDigitalInputCount (FUN)

Diese Funktion liefert die Anzahl der angeschlossenen digitalen Eingangsmodule zurück.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
... _GetDigitalInputCount	INT	Anzahl digitaler Eingangsmodule

C200_ActiveBus_GetDigitalInputPA (FUN)

Diese Funktion liefert einen Zeiger auf den Bereich des @ctiveBus Eingangsprozessabbildes zurück, der von den digitalen Eingangsmodulen verwendet wird.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert

... _GetDigitalInputPA	POINTER TO WORD	Zeiger auf Eingangsprozessabbild

C200_ActiveBus_GetDigitalInputPAOffset (FUN)

Diese Funktion liefert den Byte-Offset zum Anfang des @ctiveBus Eingangsprozessabbildes zurück, ab dem die Daten der digitalen Eingangsmodule zu finden sind.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
... _GetDigitalInputPAOffset	INT	Byte-Offset zum Eingangsprozessabbild

C200_ActiveBus_GetDigitalOutputCount (FUN)

Diese Funktion liefert die Anzahl der angeschlossenen digitalen Ausgangsmodule zurück.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
... _GetDigitalOutputCount	INT	Anzahl digitaler Ausgangsmodule

C200_ActiveBus_GetDigitalOutputPA (FUN)

Diese Funktion liefert einen Zeiger auf den Bereich des @ctiveBus Ausgangsprozessabbildes zurück, der von den digitalen Ausgangsmodulen verwendet wird.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
... _GetDigitalOutputPA	POINTER TO WORD	Zeiger auf Ausgangsprozessabbild

C200_ActiveBus_GetDigitalOutputPAOffset (FUN)

Diese Funktion liefert den Byte-Offset zum Anfang des @ctiveBus Ausgangsprozessabbildes zurück, ab dem die Daten der digitalen Ausgangsmodule zu finden sind.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
... _GetDigitalOutputPAOffset	INT	Byte-Offset zum Ausgangsprozessabbild

C200_ActiveBus_GetAnalogInputCount (FUN)

Diese Funktion liefert die Anzahl der angeschlossenen analogen Eingangsmodule zurück.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
... _GetAnalogInputCount	INT	Anzahl analoger Eingangsmodule

C200_ActiveBus_GetAnalogInputPA (FUN)

Diese Funktion liefert einen Zeiger auf den Bereich des @ctiveBus Eingangsprozessabbildes zurück, der von den analogen Eingangsmodulen verwendet wird.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
... _GetAnalogInputPA	POINTER TO WORD	Zeiger auf Eingangsprozessabbild

C200_ActiveBus_GetAnalogInputPAOffset (FUN)

Diese Funktion liefert den Byte-Offset zum Anfang des @ctiveBus Eingangsprozessabbildes zurück, ab dem die Daten der analogen Eingangsmodule zu finden sind.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert

... _GetAnalogInputPAOffset	INT	Byte-Offset zum Eingangsprozessabbild

C200_ActiveBus_GetAnalogOutputCount (FUN)

Diese Funktion liefert die Anzahl der angeschlossenen analogen Ausgangsmodule zurück.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
... _GetAnalogOutputCount	INT	Anzahl analoger Ausgangsmodule

C200_ActiveBus_GetAnalogOutputPA (FUN)

Diese Funktion liefert einen Zeiger auf den Bereich des @ctiveBus Ausgangsprozessabbildes zurück, der von den analogen Ausgangsmodulen verwendet wird.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
... _GetAnalogOutputPA	POINTER TO WORD	Zeiger auf Ausgangsprozessabbild

C200_ActiveBus_GetAnalogOutputPAOffset (FUN)

Diese Funktion liefert den Byte-Offset zum Anfang des @ctiveBus Ausgangsprozessabbildes zurück, ab dem die Daten der analogen Ausgangsmodule zu finden sind.

Variable	Datentyp	Beschreibung
Dummy	INT	Dummy-Wert
... _GetAnalogOutputPAOffset	INT	Byte-Offset zum Ausgangsprozessabbild

4.5.1.4 Hilfsfunktionen @ctiveIO Zugriff aus Timerfunktionen

In dieser Untergruppe sind Funktionen zusammengefaßt, die innerhalb einer Timer-Funktion den Zugriff auf die Rohdaten des @ctiveBus ermöglichen. Die Schreibroutinen schreiben ihre Daten in das IEC-Prozessabbild und parallel auf den @ctiveBus.

- C200_ActiveBus_GetDAInputPA
- C200_ActiveBus_GetDAOutputPA

- C200_ActiveBus_GetDAModuleOffset
- C200_ActiveBus_GetDAModuleSize

- C200_ActiveBus_GetDAModuleData_BOOL
- C200_ActiveBus_GetDAModuleData_BYTE
- C200_ActiveBus_GetDAModuleData_WORD
- C200_ActiveBus_GetDAModuleData_DWORD

- C200_ActiveBus_SetDAModuleData_BOOL
- C200_ActiveBus_SetDAModuleData_BYTE
- C200_ActiveBus_SetDAModuleData_WORD
- C200_ActiveBus_SetDAModuleData_DWORD

C200_ActiveBus_GetDAInputPA

Diese Funktion liefert einen Zeiger auf den Anfang des Eingangsprozessabbildes (Rohdaten).

Variable	Datentyp	Beschreibung
Dummy	INT	Dummywert
... _GetDAInputPA	POINTER TO WORD	Zeiger auf das Prozessabbild

C200_ActiveBus_GetDAOutputPA

Diese Funktion liefert einen Zeiger auf den Anfang des Ausgangsprozessabbildes (Rohdaten).

Variable	Datentyp	Beschreibung
Dummy	INT	Dummywert
... _GetDAOutputPA	POINTER TO WORD	Zeiger auf das Prozessabbild

C200_ActiveBus_GetDAModuleOffset

Diese Funktion liefert den Offset zum Anfang des Prozessabbildes des an der angegebenen Position befindlichen Moduls zurück. Ist die Position außerhalb des zulässigen Bereichs, wird –1 zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
... _GetDAModuleOffset	INT	Offset zum Prozessabbild

C200_ActiveBus_GetDAModuleSize

Diese Funktion liefert die Datengröße (für Eingangs- und Ausgangsdaten) in Bytes des an der angegebenen Position befindlichen Moduls zurück. Ist die Position außerhalb des zulässigen Bereichs, wird –1 zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
... _GetDAModuleSize	INT	Datengröße (in Bytes)

C200_ActiveBus_GetDAModuleData_BOOL

Diese Funktion liefert den Wert eines Bits des angegebenen Moduls zurück. Hierzu wird der Modul-Index und ein Byte-Offset angegeben.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
... _GetDAModuleData_BOOL	BOOL	Wert des angegebenen Bits

C200_ActiveBus_GetDAModuleData_BYTE

Diese Funktion liefert den Wert eines Bytes des angegebenen Moduls zurück. Hierzu wird der Modul-Index und ein Byte-Offset angegeben.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
... _GetDAModuleData_BYTE	BYTE	Wert des angegebenen Bytes

C200_ActiveBus_GetDAModuleData_WORD

Diese Funktion liefert den Wert eines Wortes des angegebenen Moduls zurück. Hierzu wird der Modul-Index und ein Byte-Offset angegeben.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
... _GetDAModuleData_WORD	WORD	Wert des angegebenen Words

C200_ActiveBus_GetDAModuleData_DWORD

Diese Funktion liefert den Wert eines Doppelwortes des angegebenen Moduls zurück. Hierzu wird der Modul-Index und ein Byte-Offset angegeben.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
... _GetDAModuleData_DWORD	DWORD	Wert des angegebenen Dwords

C200_ActiveBus_SetDAModuleData_BOOL

Diese Funktion setzt das angegebene Bit auf den angegebenen Wert. Hierzu wird der Modul-Index und ein Byte-Offset angegeben. Der vorherige Wert aus dem Modul wird als Rückgabewert zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
Data	BOOL	Neuer Wert des Bits
... _SetDAModuleData_BOOL	BOOL	Vorheriger Wert des Bits

C200_ActiveBus_SetDAModuleData_BYTE

Diese Funktion setzt das angegebene BYTE auf den angegebenen Wert. Hierzu wird der Modul-Index und ein Byte-Offset angegeben. Der vorherige Wert aus dem Modul wird als Rückgabewert zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
Data	BYTE	Neuer Wert des Bytes
... _SetDAModuleData_BYTE	BYTE	Vorheriger Wert des Bytes

C200_ActiveBus_SetDAModuleData_WORD

Diese Funktion setzt das angegebene WORD auf den angegebenen Wert. Hierzu wird der Modul-Index und ein Byte-Offset angegeben. Der vorherige Wert aus dem Modul wird als Rückgabewert zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
Data	WORD	Neuer Wert des Words
... _SetDAModuleData_WORD	WORD	Vorheriger Wert des Words

C200_ActiveBus_SetDAModuleData_DWORD

Diese Funktion setzt das angegebene DWORD auf den angegebenen Wert. Hierzu wird der Modul-Index und ein Byte-Offset angegeben. Der vorherige Wert aus dem Modul wird als Rückgabewert zurückgeliefert.

Variable	Datentyp	Beschreibung
ModulIdx	INT	Index des Moduls
Offset	INT	Byte-Offset innerhalb des ausgewählten Moduls
Data	DWORD	Neuer Wert des Dwords
... _SetDAModuleData_DWORD	DWORD	Vorheriger Wert des Dwords

4.5.1.5 Diverse Funktionen

In dieser Untergruppe sind Funktionen zusammengefasst, die nicht in die anderen Gruppen passen.

- C200_GetBufferedRAMAddress
- C200_GetBufferedRAMSize
- C200_GetTimestamp
- C200_RestartModule
- C200_GetIPConfiguration
- C200_Watchdog_GetState
- C200_Watchdog_SetState
- C200_Watchdog_Trigger

C200_GetBufferedRAMAddress (FUN)

Diese Funktion liefert einen Zeiger auf den Anfang des Gold-Cap gepufferten SRAM-Speichers zurück.

Variable	Datentyp	Beschreibung
bDummy	BOOL	Dummy-Wert
... _GetBufferedRAMAddress	POINTER TO DWORD	Startadresse des Buffers

C200_GetBufferedRAMSize (FUN)

Diese Funktion liefert die max. verfügbare Größe des Gold-Cap gepufferten SRAM-Speichers.

Variable	Datentyp	Beschreibung
bDummy	BOOL	Dummy-Wert
... _GetBufferedRAMSize	DWORD	Max. Größe des Buffers

C200_GetTimestamp (FUN)

Diese Funktion liefert einen Zeitstempel zurück. Der Zeitstempel ist mikrosekundenbasierend und wird mit Anlegen der Versorgungsspannung gestartet. Der Zeitstempel läuft nach ca. 71,58 Minuten über.

Variable	Datentyp	Beschreibung
Dummy	DWORD	Dummy-Wert
... _GetTimestamp	DWORD	Zeitstempel in Mikrosekunden

C200_RestartModule (FUN)

Diese Funktion löst einen Reset aus. Der Reset erfolgt nicht sofort, sondern wird um ca. 1,5 Sekunden verzögert ausgelöst. Bis zu diesem Zeitpunkt läuft die PLC normal weiter.

Variable	Datentyp	Beschreibung
bDummy	BOOL	Dummy-Wert
... _RestartModule	BOOL	Dummy-Wert

C200_GetIPConfiguration

Diese Funktion liefert die IP-Konfiguration des Moduls zurück. Über den Parameter fUseStoreConfig kann zwischen der gerade aktiven und der eingestellten (gespeicherten) Konfiguration unterschieden werden. Die IP-Adressen werden als 32 Bit-Werte im Rohformat zurückgegeben.

Variable	Datentyp	Beschreibung
fUseStoredConfig	BOOL	Verwende gespeicherte (TRUE) bzw. aktive Konfiguration (FALSE)
IP_Address	POINTER TO DWORD	IP-Adresse
IP_SubnetMask	POINTER TO DWORD	Subnetz-Maske
IP_Gateway	POINTER TO DWORD	Gateway-Adresse
DHCP_Active	POINTER TO BOOL	DHCP aktiv
... _GetIPConfiguration	BOOL	Ergebnis der Abfrage

C200_Watchdog_GetState (FUN)

Diese Funktion liefert den Zustand des Laufzeit-Watchdog (Zykluszeit-Überwachung) zurück. Der Rückgabewert gibt an, ob der Watchdog aktiviert ist oder nicht.

Variable	Datentyp	Beschreibung
Dummy	BOOL	Dummy-Wert
... _Watchdog_GetState	BOOL	Zustand des Watchdogs

C200_Watchdog_SetState (FUN)

Mit dieser Funktion kann der Laufzeit-Watchdog (Zykluszeit-Überwachung) aktiviert oder deaktiviert werden. Der Rückgabewert gibt an, ob die Aktion erfolgreich durchgeführt wurde (TRUE).

Variable	Datentyp	Beschreibung
Enable	BOOL	Zukünftiger Zustand des Watchdogs
... _Watchdog_SetState	BOOL	Ergebnis der Aktion

C200_Watchdog_Trigger (FUN)

Mit dieser Funktion kann der Laufzeit-Watchdog (Zykluszeit-Überwachung) bei lang laufenden Routinen zusätzlich getriggert werden.

Variable	Datentyp	Beschreibung
Dummy	BOOL	Dummy-Wert
... _Watchdog_Trigger	BOOL	Dummy-Wert

4.5.2 Die Bibliothek Lib_Syslog

Mit Hilfe der Bibliothek Lib_Syslog können Log-Meldungen auf einem speziellen Server übertragen werden. Dieser Syslog-Server steht für diverse Betriebssystem zur Verfügung.

Damit diese Bibliothek korrekt arbeiten kann, muss die C200-Schnittstelle korrekt konfiguriert werden. D.h. es muss eingestellt werden, an welchen Server die Daten versendet werden sollen. Die Übertragung der Daten zum Server erfolgt über TCP/IP, daher sollte auch die Netzwerkumgebung vor der Verwendung kontrolliert werden, um eine Übertragung zu gewährleisten.

- C200_SysLog_SendMsg

C200_SysLog_SendMsg

Mit Hilfe dieser Funktion können Log-Meldungen zu einem entfernter Syslog-Server übertragen werden.

Variable	Datentyp	Beschreibung
Facility	D_SysLog_Facility	Ursprung der Meldung
Severity	D_SysLog_Severity	Schwere der Meldung
Msg	STRING(255)	Zu sendende Meldung
... _SendMsg	BOOL	Dummy-Wert

A Kommunikation und Diagnose

A.1 Kommunikationsparameter

A.2 PLC-Browser

A.3 Fehlermeldungen

A.4 @PLC-C20x Modbus Interface

A.5 Modbus RS232 Slave

A.6 Modbus RS232 Master

A.7 Modbus TCP Slave

A.8 Modbus TCP Master

B Teleservice und Kommunikation

B.1 Serielle Kommunikation

B.2 TCP/IP – Wie funktioniert das?

B.3 Netzwerkkommunikation mit Sockets

B.4 Rechneranschaltung mit PPP (Point To Point Protocol)

B.5 Datentransfer mit FTP (File Transfer Protocol)

Die @PLC-C20x verfügt über einen integrierten ftp-Server, mit dem auf das interne Dateisystem (Flash-FS und ggf. gepuffertes SRAM-FS) passwortgeschützt zugegriffen werden kann. Es stehen 4 Zugriffsebenen zur Verfügung, die Schutz vor unberechtigten Schreib- / Lesezugriffen bieten. Die Benutzernamen und die dazugehörigen Passwörter lassen sich vom Anwender bei Bedarf ändern.

B.5.1 Zugriffsrechte

Wird eine Datei neu im Dateisystem angelegt, bekommt sie die Zugriffebene des angemeldeten Benutzers zugewiesen. Anschließend kann nur noch auf diese Datei zugegriffen werden, wenn der angemeldete Benutzer mindestens dieser Zugriffsebene (oder einer höherwertigen Ebene) angehört. Dieser Punkt ist besonders zu beachten, wenn von der IEC aus auf diese Dateien zugegriffen werden soll. Innerhalb der @PLC-C20x sind den einzelnen Zugriffsebenen feste Bedeutungen zugeordnet:

Zugriffsebene 0 (max. Zugriffsrechte)

Auf dieser Ebene ist der Zugriff auf wesentliche Systemkomponenten und Konfigurationsdateien möglich. Hierzu gehören das „BIOS“ (die Datei „boot.bin“) und die elementaren Konfigurationsdateien „system.cfg“ und „modul.cfg“.

Zugriffsebene 1 (Firmware-Update)

Auf dieser Ebene kann auf die Firmware-Dateien („ftpserv.bin“ und „app.bin“) und die applikationsspezifische Konfigurationsdatei „config.cfg“ zugegriffen werden.

Zugriffsebene 2 (IEC Projektdaten)

Auf dieser Ebene werden die IEC Projektdateien (Bootprojekt und Source- bzw. Projektdateien) abgelegt.

Zugriffsebene 3 (Anwenderdaten)

Auf dieser Ebene sind alle weiteren Dateien abgelegt. Hierzu zählen auch die Dateien, die mittels der Bibliothek „SysLibFile“ erzeugt und bearbeitet werden.

B.5.2 Zugriff auf das gepufferte SRAM-FS

Während innerhalb der IEC mittels Laufwerksbuchstaben auf Flash-FS und gepuffertem SRAM-FS (sofern vorhanden) zugegriffen werden kann, existieren bei Zugriff über den ftp-Server keine Laufwerksbuchstaben. Aus diesem Grund wird das gepufferte SRAM-FS ins Flash-FS gemappt. Der Mechanismus hierzu arbeitet wie folgt: Nach erfolgter Anmeldung ist zunächst das Flash-FS aktiv. Bei der Ausgabe der Dateiliste erscheint neben den Dateien des Flash-FS das virtuelle Verzeichnis „BufferedRAM“. Mit Hilfe eines Verzeichniswechsels („CWD BufferedRAM“) kann von Flash-FS ins gepufferte SRAM-FS gewechselt werden. Bei der Ausgabe der Dateiliste erscheint nun das virtuelle Verzeichnis nicht mehr. Soll nun wieder zum Flash-FS gewechselt werden, kann dies mit Hilfe eines Verzeichniswechsels erfolgen („CWD ..“ bzw. ein von „BufferedRAM“ unterschiedlicher Bezeichner).

Bei diesem Mechanismus stellt das virtuelle Verzeichnis „BufferedRAM“ nichts weiter als einen Schalter zwischen den Dateisystemen dar. Der direkte Zugriff auf eine Datei des gepufferten SRAM-FS in der Form „/BufferedRAM/Datei.txt“ ist daher nicht möglich!

B.5.3 Liste der unterstützten ftp-Befehle

Nachfolgend sind alle unterstützten ftp-Befehl aufgelistet, sowie Anmerkungen zu ihrer Anwendung, sollte diese anders sein, als gemeinhin erwartet wird.

USER <Benutzername>

➔ gebräuchlich

PASS <Passwort>

➔ gebräuchlich

SYST

➔ gebräuchlich

TYPE <Typ>

➔ gebräuchlich

PORT <Portangabe>

➔ gebräuchlich

LIST

Der optionale Parameter des LIST-Befehls wird in dieser Implementation des ftp-Server nicht weiter verarbeitet. Die Ausgabe der Dateiliste orientiert sich am unter UNIX-Systemen gebräuchlichen Format. Abhängig vom Erstellungsjahr und dem aktuellen Datum erfolgt die Ausgabe in zwei leicht verschiedenen Formaten.

1. Aktuelles Jahr gleich Erstellungsjahr

```
drw-------  1 user     users          0 Jan  1 00:00 BufferedRAM
-rw-------  1 root     root      131072 Jan  1 00:00 boot.bin
-rw-------  1 root     root          24 Jan  7 01:25 modul.cfg
-rw-------  1 root     root         192 Jan  7 01:25 system.cfg
-rw-------  1 update   users      819616 Jan  7 23:30 app.bin
-rw-------  1 update   users      539624 Jan  7 23:30 ftpserv.bin
```

2. Aktuelles Jahr ungleich Erstellungsjahr

```
drw-------  1 user     users          0 Jan  1  2003 BufferedRAM
-rw-------  1 root     root      131072 Jan  1  2003 boot.bin
-rw-------  1 root     root          24 Jan  7  2003 modul.cfg
-rw-------  1 root     root         192 Jan  7  2003 system.cfg
-rw-------  1 update   users      819616 Jan  7  2003 app.bin
-rw-------  1 update   users      539624 Jan  7  2003 ftpserv.bin
```

Formatbeschreibung														
Variante 1:	d	rw-------		1		user		users		0		Jan 1 2003		Buffered RAM
Variante 2:	-	rw-------		1		root		root		24		Jan 7 01:25		modul.cfg
Zeichenbreite:	1	9	2	1	1	9	1	9	1	8	1	12	1	variabel
Beschreibung:	**"d" = Verzeichnis / "-" = Datei**	**Zugriffsrechte (immer rw-------)**		**Fülldaten**		**Benutzername (eingetragener Name der jeweiligen Zugriffsebene)**		**Gruppenname (immer „users" bzw. „root")**		**Dateigröße**		**Erstellungsdatum**		**Dateiname**

RETR <Dateiname>

➔ gebräuchlich

STOR <Dateiname>

➔ gebräuchlich

DELE <Dateiname>

➔ gebräuchlich

CWD <Verzeichnis>

Der CWD-Befehl dient in dieser ftp-Server Implementierung nicht dem Verzeichniswechsel, sondern der Umschaltung zwischen Flash-FS und gepuffertem SRAM-FS. Ein „CWD BufferedRAM" schaltet immer zum gepuffertem SRAM-FS um, jeder andere „Verzeichnisname" wechselt wieder zurück zum Flash-FS (bevorzug sollte „CWD .." verwendet werden).

PWD

Dieser Befehl liefert unabhängig vom gewählten Verzeichnis (Flash-FS bzw. gepuffertem SRAM-FS) immer „/" als aktuelles Verzeichnis zurück.

QUIT

➔ gebräuchlich

C Fehlerbehandlung (Exception Handling)

Treten während des Programmablaufs schwerwiegende Fehler auf, bietet die @PLC-C20x-EN/PB eine Funktion, diese Fehler bei der Ausführung abzufangen, so dass der Programmierer diese im nachhinein korrigieren kann. Zu unterscheiden sind zwei Fehlergruppen: Zugriffsverletzungen (bei fehlerhafter Verwendung von Zeigervariablen) bzw. Divisionen durch Null.

Die Fehlerbehandlung kann **aktiviert bzw. deaktiviert** werden. Defaultmäßig ist sie deaktiviert und kann mit den folgenden Schritten aktiviert bzw. wieder deaktiviert werden:

Wählen Sie in der CoDeSys Entwicklungsumgebung im Objekt Organizer in der Registerkarte `Ressourcen` den „`PLC-Browser`“ aus. Dort können Sie die Fehlerbehandlung durch Eingabe von `enableexceptionhandling` aktivieren bzw. durch Eingabe von `disableexceptionhandling` deaktivieren. Alternativ finden Sie die beiden Befehle auch unter den Standardkommando.

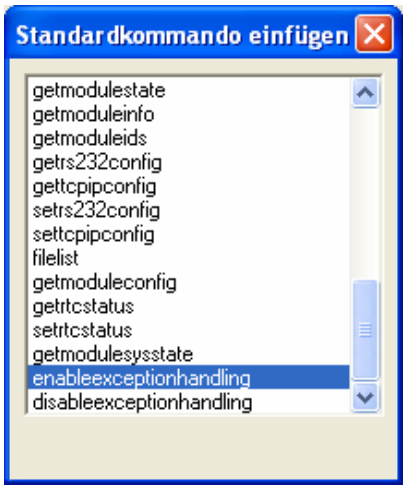

Abbildung 26: CoDeSys PLC Browser

C.1 Fehlerbehandlung deaktiviert

Zugriffsverletzung / Division durch “0”

Nach der Zugriffsverletzung bzw. Division durch 0 führt die PLC einen Reboot durch. Anschließend startet die PLC mit dem Laden und Starten des Bootprojektes (sofern vorhanden).

Hinweis: Dieses Verhalten kann dazu führen, dass die PLC einen Dauer-Reboot durchführt und, sofern die Zugriffsverletzung bzw. Division durch 0 am Programmanfang auftritt, nicht mehr ansprechbar ist. Dieser Zustand lässt sich mittels geeigneter Tools beheben (auf Anfrage erhältlich).

Ausnahme:

Bei der Division durch 0 mit einer **Fließkommazahl** wird die Programmausführung nicht unterbrochen. Am Modul (Diag.-LED leuchtet und Err.-LED blinkt) und im Onlinemodus wird eine Warnung ausgegeben. Dieser Fehlerzustand bleibt bis zur Behebung durch Starten des Programms aktiv.

C.2 Fehlerbehandlung aktiviert

Zugriffsverletzung

Wie schon bei deaktivierter Fehlerbehandlung führt die PLC einen Reboot durch. Anschließend startet die PLC. In diesem Fall wird allerdings das Bootprojekt nicht gestartet (im wiederholten Fall wird das Bootprojekt auch nicht mehr geladen). Der Anwender kann sich nun über CoDeSys auf die PLC einloggen und den Fehler korrigieren bzw. das Programm einfach wieder starten.

Division durch 0

Tritt eine Division durch 0 auf, geht die PLC in einen Fehlerzustand über (Diag.-LED leuchtet und Err.-LED blinkt) und die Programmausführung wird gestoppt. Im Onlinemodus wird eine Warnung am Bildschirm ausgegeben. Dieser Fehlerzustand bleibt bis zur Behebung durch Starten des Programms aktiv.

D Anwendungsbeispiele

D.1 Erstellen der Hardware Konfiguration mit einem Pofibus-Master

Folgendes Bild zeigt die Hardware-Konfigurationseinstellung unter **S7**© (Abbildung 24). Dieses Beispiel bezieht sich auf das Kapitel Profibus DP Slave 3.2.5.

Nachdem Sie das @PLC-C20x-PB Modul in Ihr Projekt eingefügt haben, können Sie mit einem Doppelklick auf das Modul, das Eigenschaften-Fenster (Abbildung 27) öffnen.

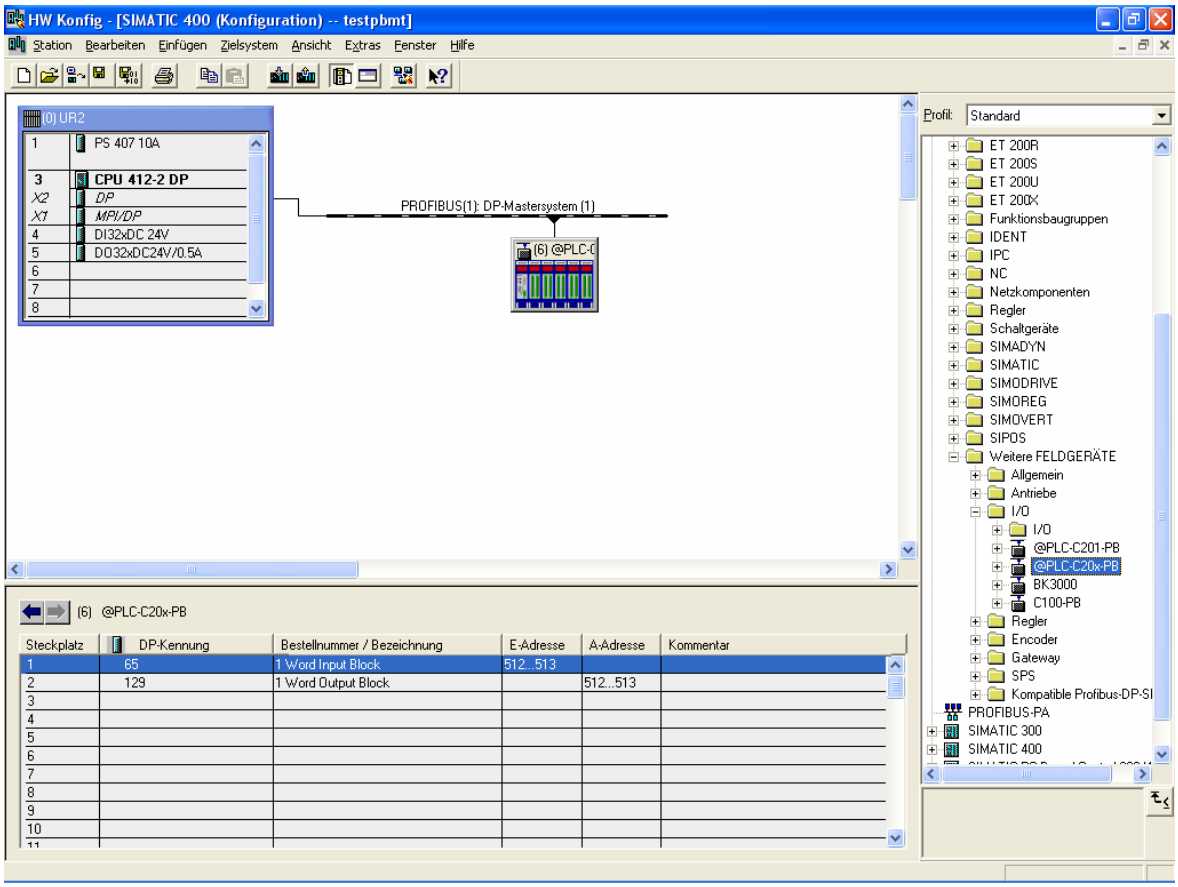

Abbildung 27: Profibus-Slave Hadware-Konfiguration unter **S7**©

Im Eigenschaften –Fenster des DP-Slave (Abbildung 28) können Sie unter: `Parametrierung > Gerätespezifische Parameter` das Datenformat (`Intel` oder `Motorola`) frei wählen. Das @PLC-C20x-PB Modul (Profibus-Slave) arbeitet intern im **Big Indian** (Motorola) Format.

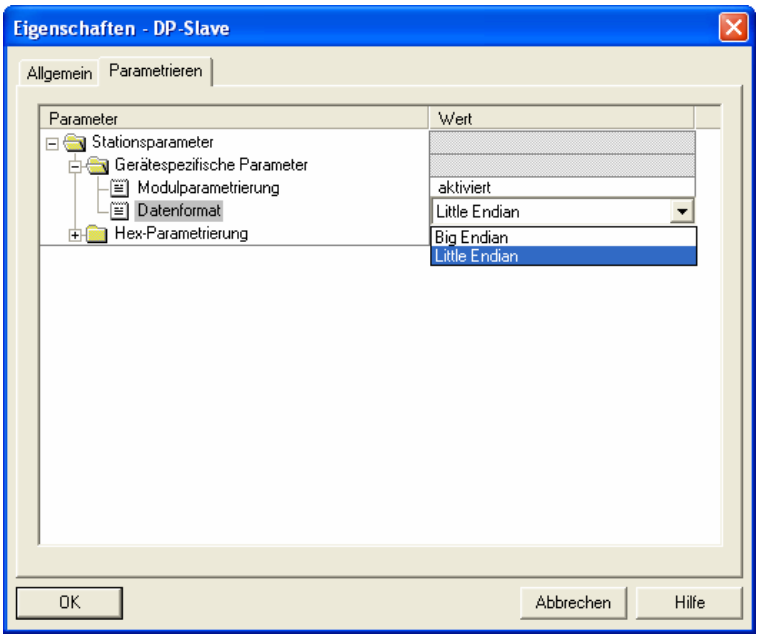

Abbildung 28: Eigenschaften-Fenster DP-Slave unter **S7**©

Einstellung der Eigenschaften im DP Mastersystem

Im folgendem Fenster (Abbildung 29) können Sie für den Datenaustausch zwischen Master und Slave die Netzeinstellungen wie Übertragungsgeschwindigkeit und Profil einstellen. Klicken Sie mit einem Doppelklick auf den Master und öffnen somit das Fenster `Eigenschaften-DP Mastersystem`. Nun befinden sich die Eigenschaften-Profibus unter den Botton `Eigenschaften`.

Vorgeschlagene Werte: Übertragungsgeschwindigkeit > 12Mbit/s
Profil > DP (Dezentrale Peripherie)

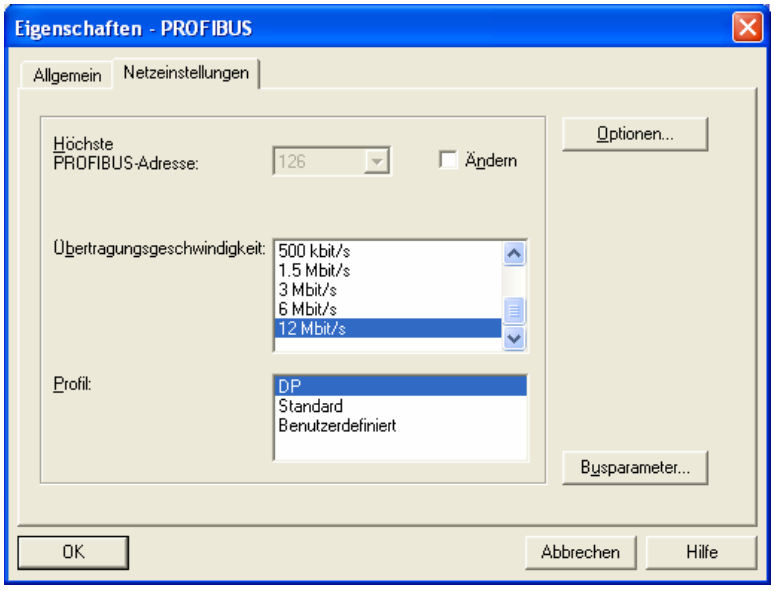

Abbildung 29: Eigenschaften Profibus

E Technische Daten

@PLC-C201	EN	PB
Spannungsversorgung	24V DC +/- 20%	
Verpolungsschutz	24V-Versorgung ist gegen Verpolung geschützt	
Stromaufnahme	95mA lastfrei, Netzteil mit Softstart-Eigenschaft	170mA
Feldbus	Ethernet/ Modbus(RS232)	+ Profibus-Slave
Serielle Schnittstelle	RS232 an MiniUSB, optional RS485	
Ethernet	10/100Mbit Netzwerkkarte	
Abmessungen	105 x 80mm	
Gewicht	280g incl. Steckverbinder	
Betriebstemperatur	0°C ...+55°C	
Lagertemperatur	-20°C ...+70°C	
CPU Spezifikationen		
Verarbeitungszeiten:		
Bit Operation	0,8 µs	
Word Operation	1,1 µs	
DWORD Operation	1,2 µs	
Festpunktarithmetik	1,6 µs	
Gleitpunktarithmetik	7,9 µs	
Hardware Funktionen		
CPU	NetARM 50	
CPU clock	44,2 MHz	
FLASH memory	2 MB x 16	
SDRAM	8 MB x 32	
NVRAM	64 KB x 16	
Diagn.-LED; Feldbus	X	
Watchdog	X	
@ctiveIO-Bus support	X	
zweiter Fieldbus	O	
Basis Firmware Funktionen		
Firmware download	ftp-Server	
@ctiveIO-Toolkit diagnostic	X	
PLC functions	IEC-61131-3	
Runtime system	CSP32E Version 2.3	
Memory für IEC Programm	768 KByte	
IEC Daten	3 MByte	
IEC Retain Daten	32 KByte	
Programmieren		
Serial	X	
TCP/IP	X	
Überwachung	X	
Breakpoints	X	
Online change	X	
Gesicherter Quell Code	X	
Bibliotheken		
Standardbibliothek / Timer	X	
Serielle Kommunikation	X	
Boot Projekt	X	
Retain Daten (NVRAM)	X	
PLC Browser	X	
Objektverzeichnis	X	
Netzwerkvariablen	X	
Syslib:RTC	N	

Konfiguration	
Watchdog	X
Multitasking	N
Exception handling	N
Modbus (RS232)	
Interface connection	Slave
Transmission protocols	RTU / ASCII
Memory size	(adjustable)
Digital output (coil)	max. 131072 Bits
Digital input (input)	max. 131072 Bits
Holding Register	max. 8192 Words
Input Register	max. 8192 Words
Function codes	1, 2, 3, 4, 5, 6, 8, 15, 16, 23

X: Standard, O: Option, N: nicht unterstützt vom @PLC-C201

@PLC-C202	**EN/PB**
Unterschiede zum @PLC-C201 Modul:	
Hardware Funktionen	
FLASH memory	4 MB x 16
SDRAM	8 MB x 32
NVRAM	32 KB x 8
Gepuffertes SRAM (für 24h)	1MBx16 Dateisystem / 1MBx16 direkter Zugriff
Gepufferte Real Time Calendar Clock	X
Bibliotheken	
Standardbibliothek / Timer	X
serielle Kommunikation	X
Boot ProjeKt	X
Retain Daten (NVRAM)	X
PLC Browser	X
Objektverzeichnis	X
Netzwerkvariablen	X
Exception handling	N
SyslibRTC	X

X: Standard, O: Option, N: nicht unterstützt vom @PLC-C202

F Literaturhinweise

- **TRS_Produkt_CD**

- **Handbuch zur SPS-Programmentwicklung mit „CoDeSys V2.3“ (Hilfedatei CoDeSys Programm)**

  Internet: http://www.3s-software.com/

- **OPC-Server**

  Internet: http://www.opcfoundation.org/

- **Profibus**

  Internet: http://www.profibus.com/

- **Modbus**

  Internet: http://www.modbus.org/

- **@ctiveIO-Toolkit: TR-Systemtechnik Produkt-CD**

  Internet: http://www.tr-systemtechnik.de/