

Automatisierungstechnik nach internationaler Norm programmieren (16)

Autor: Dr. Ulrich Becker

Fachzentrum Automatisierungstechnik und vernetzte Systeme im BTZ Rohr-Kloster

Mail: Ulrich.Becker@BTZ-Rohr.de

Einführung in die Vernetzung von Automatisierungskomponenten mit Ethernet Teil 2

Im ersten Teil zum Thema Ethernet in der Automatisierungstechnik (Folge 15) wurden grundlegende Fragen zu Protokollen rund um die Ethernet-Kommunikation erörtert. Ethernet / TCP / UDP mit Erweiterungen für Echtzeitanwendungen entwickelt sich zur tragenden Säule der Vernetzung von Automatisierungskomponenten. In dieser Folge werden zwei Methoden der Vernetzung von Ethernet-Controllern des Systems WAGO-IO-750 mit CoDeSys Software-Werkzeugen vorgestellt.

Ethernet-Kommunikation im Busklemmensystem WAGO-I/O-750

Die Ethernetkommunikation soll nun am Trainingsrack mit Automatisierungskomponenten des Systems WAGO-I/O-750 getestet werden. Als Stärke der programmierbaren Ethernet-Feldbus-Controller 750-841 und der Koppler 750-341 gilt die Anwendung des „normalen“ Ethernet – Protokolls TCP/IP/UDP ohne spezielle Echtzeit-Erweiterungen. Der Mischbetrieb von Automatisierungs- und Bürokomponenten verläuft problemfrei, und für eine Vielzahl von Aufgaben der Automatisierungstechnik sind die zu erwartenden Reaktionszeiten zwischen 10 und 100ms akzeptabel.

In Folge 4 hatten wir den Controller 750-841 in Betrieb genommen. Der Eintrag des Programms und dessen Online-Beobachtung basierten auf der Kommunikation zwischen PC und Controller über einen Kanal Ethernet TCP/IP. Wir hatten dem Controller eine gültige IP-Adresse gegeben und die Netzwerkkarte des PC über ein gekreuztes Patchkabel mit dem Controller verbunden. Mit wenigen grundlegenden Kenntnissen über Netzwerke gelang die Kommunikation problemlos.

Nunmehr wollen wir das Netzwerk erweitern und weitere Controller, Koppler oder auch PC einbinden. Ziel ist dabei der Datenaustausch zwischen den Automatisierungskomponenten (**Bild 87**).

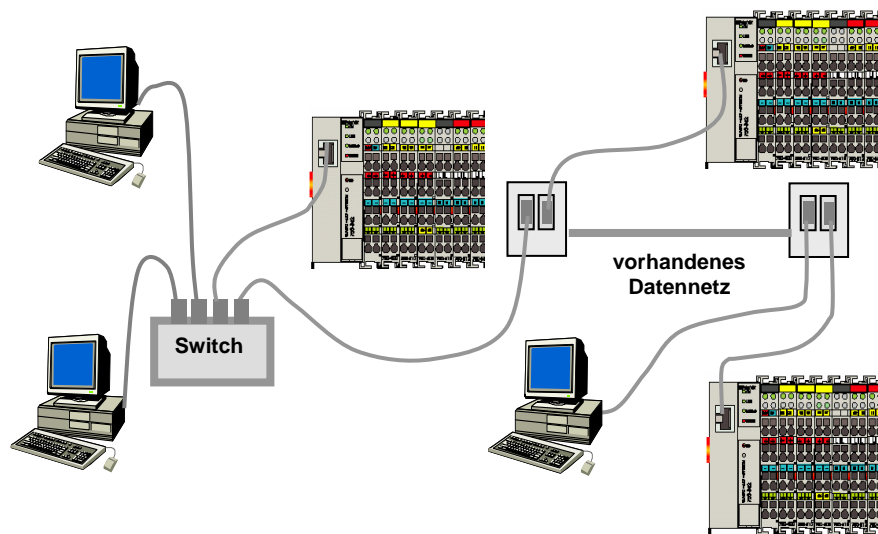


Bild 87: Programmiergeräte, Controller und Koppler im Ethernet-Netz

Wiederholend sollen hier nochmals die Begriffe Controller und Koppler erwähnt werden: Ein Feldbuskoppler wird in dezentralen Automatisierungssystemen eingesetzt, um Signale der Eingangs-Busklemmen auf das Bussystem zu legen oder Bussignale auf Ausgangsbusklemmen zu schalten. Ein programmierbarer Feldbuscontroller (PFC) leistet gleiches, kann aber darüberhinaus mit seinem Anwenderprogramm Ein- und Ausgangssignale verarbeiten. Äußerlich sind beide Komponenten - wenn sie in Busklemmensysteme eingebaut sind - kaum zu unterscheiden (**Bild 88**).

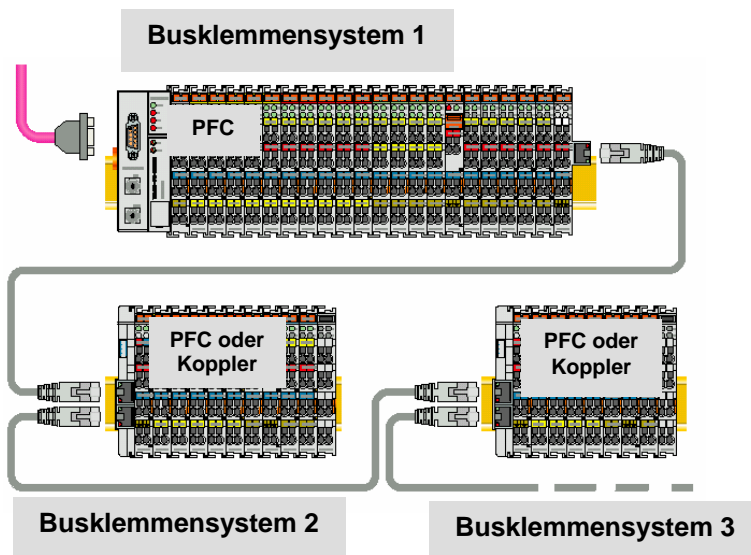


Bild 88: Der Einsatz von PFC oder Kopplern bestimmt den Grad der Dezentralisierung

Werden im Bild 88 in allen drei Systemen Controller eingesetzt, so kann jedes System mit den eigenen Busklemmen autark als programmierbares Automatisierungssystem arbeiten. Über das Netz werden nur die Signale geleitet, welches das Zusammenwirken des Gesamtsystems sicherstellen. Dies stellt die höchste Stufe der Dezentralisierung von Automatisierungssystemen dar. Werden dagegen in den Systemen 2 und / oder 3 Koppler eingesetzt, so können diese lediglich Signale an ihren Eingangsklemmen an den Controller des Systems 1 leiten. Dort werden sie gemäß Anwenderprogramm verarbeitet. Ausgangssignale sendet der

PFC über das Netz an die Koppler, welche diese auf ihre Ausgangsklemmen legen. Eine solche Architektur wäre die erste Stufe der Dezentralisierung, und zwischen beiden erläuterten Stufen sind beliebige Mischungen von Controllern und Kopplern möglich.

In der ersten Folge dieser Serie hatten wir das Trainingsrack für das Testen der Programme vorgestellt, wie im **Bild 89** nochmals gezeigt. Rüstet man diese Trainingsgeräte mit einfachsten Switchen aus, wie sie für eine „fliegende Bürokabelung“ Verwendung finden, so steht der Vernetzung zweier oder mehrerer PFC mittels ungekreuzter Patchkabel nichts im Wege. Selbstverständlich sollten wir vor Augen haben, dass industriemäßige Anwendungen höhere Schutzgrade wie der hier vorliegende Grad IP20 sowie sichere Stecker und Leitungen erfordern. Dafür bietet der Markt eine Vielzahl erprobter Lösungen.

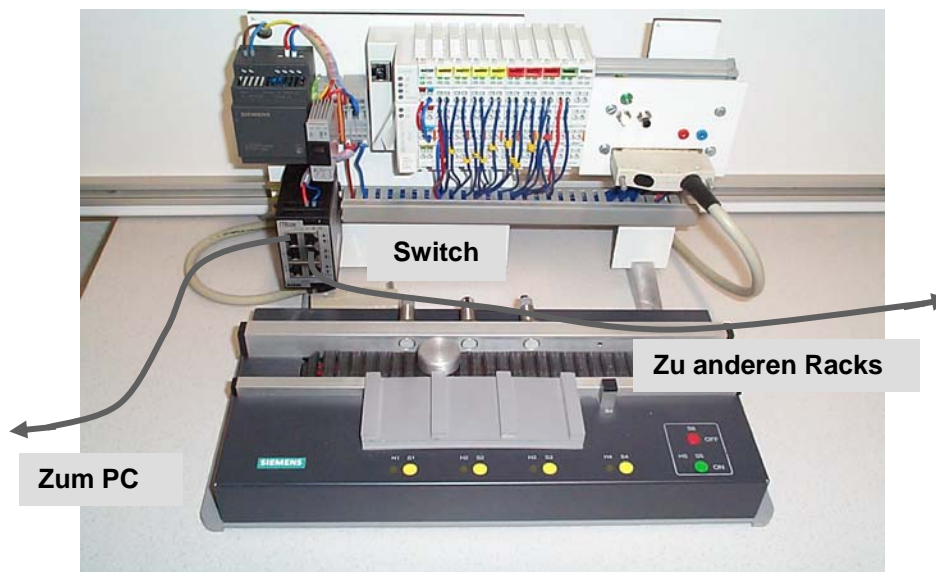


Bild 89: Vernetzung von Trainingsracks mittels einfachster Switches

Kommunikation mit Netzvariablen

Bild 88 zeigt, daß in dezentralen Automatisierungssystemen bestimmte Signale über das Netz bzw. das Feldbussystem geleitet werden müssen. Um Details der Verpackung in Datentelegramme braucht sich der Anwender zumeist nicht zu kümmern. Für ihn sichtbar erfolgt der Versand in Form von Variablen, welche durch ihren Namen und Datentyp festgelegt sind.

Eine sehr einfach zu parametrierende Kommunikation ist die mittels Netzvariablen. Hierbei werden in den beteiligten Automatisierungskomponenten Globale Variablen als Netzvariablen deklariert, die dann von den Komponenten entweder gelesen oder auch geschrieben werden können. Grundsätzlich ist keine Programmierung, sondern nur eine Parametrierung erforderlich. Netzvariablen werden nach dem Prinzip des Rundfunks (Broadcast) im gesamten Netz oder aber auch in Teilnetzen vertrieben. Das Automatisierungssystem CoDeSys / WAGO-IO-750 ermöglicht eine solche Kommunikation. Zunächst muß dazu in den Einstellungen des Zielsystems die Kommunikation mit Netzvariablen freigeschaltet werden (**Bild 90**).

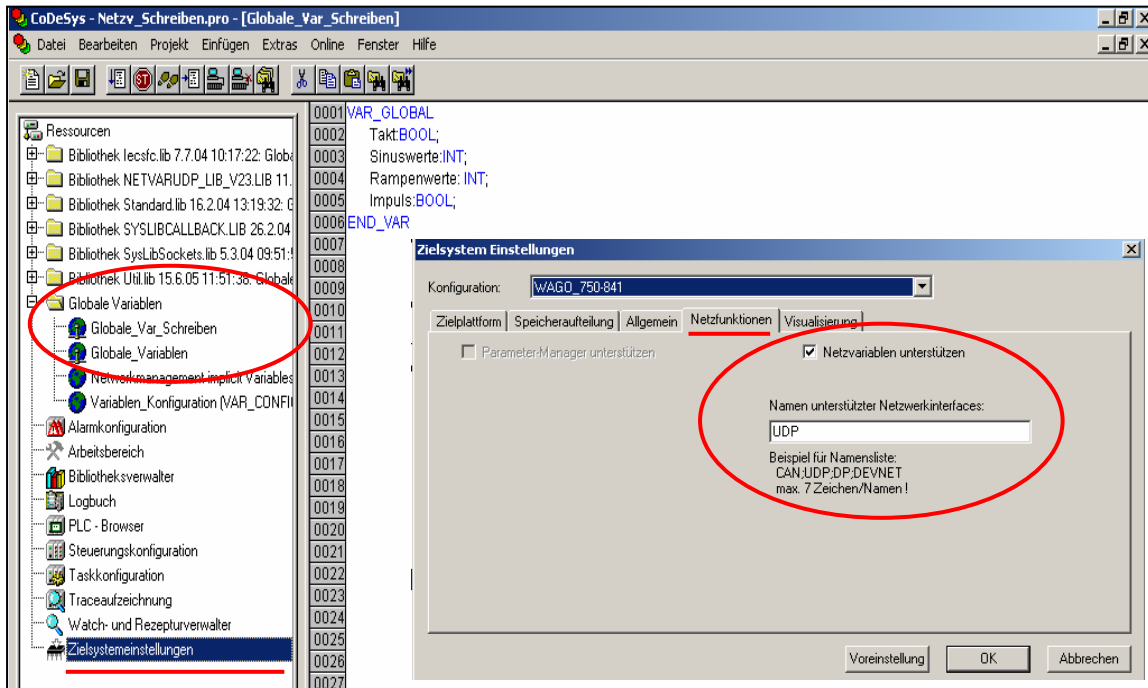


Bild 90: Freischalten und Deklaration von Netzvariablen für PFC 750-841

Im zweiten Schritt werden Listen globaler Variablen angelegt, Verbindungen zugeordnet und dabei die Attribute Lesen oder Schreiben deklariert. **Bild 91** zeigt einige Details dieser Arbeit. Hier wurde zunächst im Hauptordner Globale Variablen über -> *Einfügen* ein spezielles Verzeichnis mit Namen „Globale_Var_Schreiben“ angelegt. Für einen Test der Kommunikation zwischen den PFC zweier Trainingsracks eignen sich besonders gut periodische Signale. Im Beispiel wurden vier solche Variablen mit Signalformen Takt, Sinus und Dreieck (Rampe) sowie ein pulsbreitengesteuerter Impuls benutzt. Die ersteren Signale stellt der Funktionsbaustein „GEN“ der Bibliothek „util.lib“ zur Verfügung. Sie sind von Datentyp BOOL bzw. INT.

Mit rechter Maustaste gelangt man zu den im Bild gezeigten Objekteigenschaften des Verzeichnisses der Globalen Variablen. Sie gelten für eine erste Verbindung (Connection 1), weitere Verbindungen könnten angelegt werden. Wir erkennen die „Verpackung“ der Daten gemäß UDP-Protokoll. Über -> *Einstellungen* gelangen wir in diesem Fenster zu Möglichkeiten, die Variablen per Broadcast an alle Netzteilnehmer zu versenden oder aber bestimmte Teilnetze festzulegen. Wichtig ist die Deklaration „Schreiben“. Sie legt fest, dass die globalen Variablen dieses Verzeichnisse in das Netz geschrieben werden, und zwar im gezeigten Beispiel nur bei Änderungen ihrer Werte. Bei schnellen Änderungen werden die Daten im zeitlichen Mindestabstand von 50 ms versandt.

In Automatisierungskomponenten, die diese Variablen empfangen sollen, geht man gleichermaßen vor, allerdings werden dort die gleichen Variablen als gelesen deklariert.

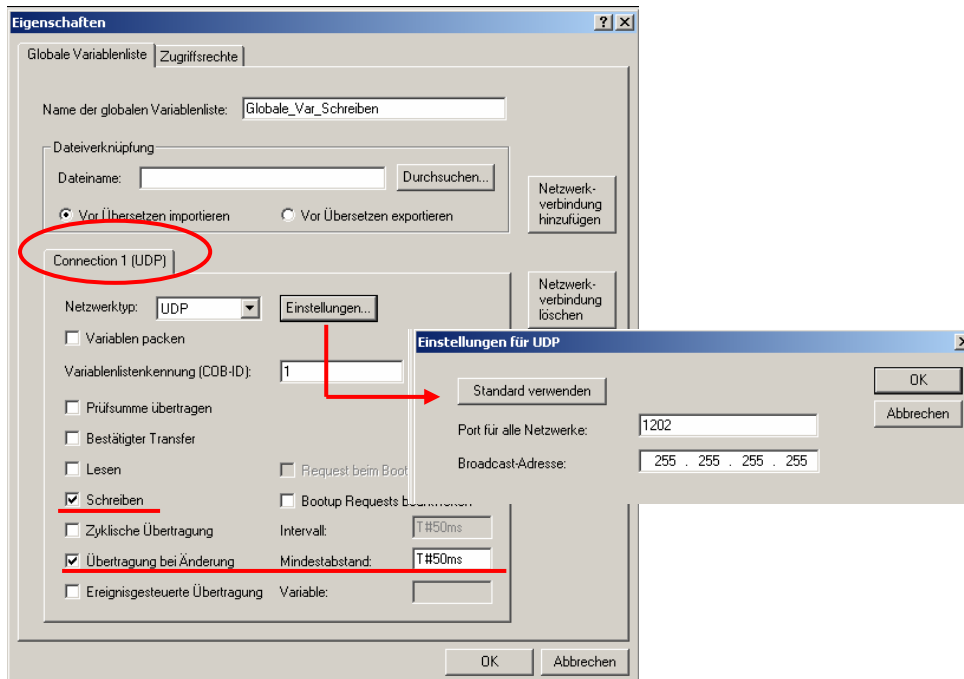


Bild 91: Festlegung der Eigenschaften der Netzvariablen

Werden nun die jeweiligen Parametrierungen der Netzvariablen zusammen mit den Programmen in die zugeordneten Controller geladen und diese gestartet, so ist die Übertragung der Variablen vom schreibenden zum lesenden Teilnehmer bereits gewährleistet. Selbstverständlich müssen die Variablen fehlerfrei gleichlautend deklariert werden. Es kommt vor, daß diese in manchen Fällen „nicht synchronisiert“ sind. Um Fehler zu vermeiden, empfiehlt es sich unbedingt, das Verzeichnis der Netzvariablen nur einmal anzulegen und dieses dann per Export / Import in die anderen Komponenten zu übertragen. **Bild 92** zeigt die Vorgehensweise. Mit dem Menu -> *Projekt -> Exportieren* erscheinen alle Komponenten des Projektes, aus denen die gewünschten Objekte ausgewählt werden können. Im Bild ist dies allein die spezielle Liste Globaler Netzvariablen. Nach Bestätigung eines vom System vorgeschlagenen Verzeichnisses werden die Komponenten in einer Datei xxx.EXP abgelegt. Sie können über -> *Projekt -> Importieren* von dort aus in ein anderes Projekt übernommen werden.

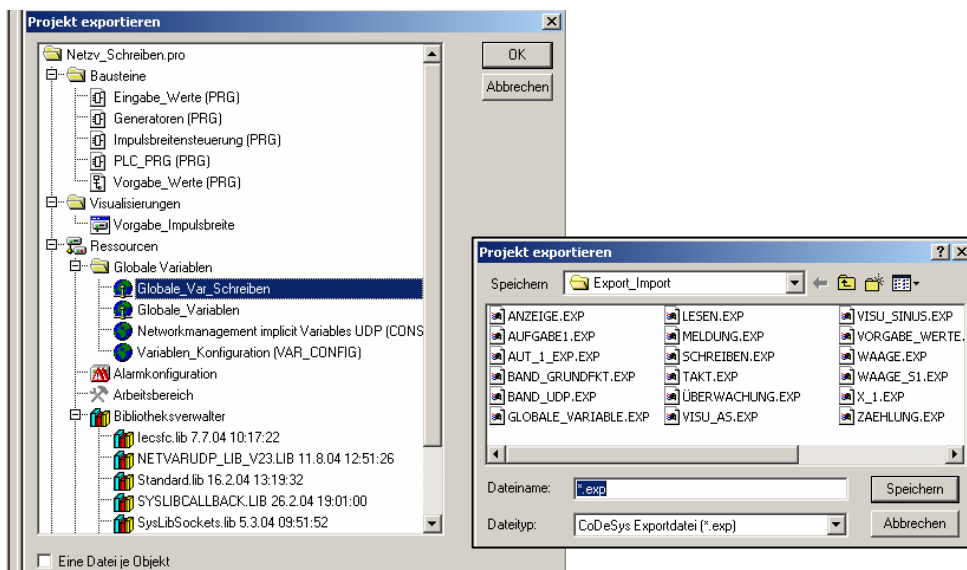


Bild 92: Export der Liste von Netzvariablen

Kommunikation mit dem Ethernet _Modbusmaster_UDP

Es ist verständlich, dass das „Rundfunk-Prinzip“ zu einer hohen Buslast führen kann, denn die so ins Netz geschriebenen Variablen werden grundsätzlich auch zu Teilnehmern im Netz geführt, die keinerlei Bezug zu den Netzvariablen haben. Als Alternative können wir mit CoDeSys auch eine gerichtete Kommunikation zwischen zwei Automatisierungskomponenten parametrieren. Dies erfolgt nach dem Master-Slave-Prinzip mit einem Funktionsbaustein „*Ethernet_Modbusmaster_UDP*“.

Nach den Erläuterungen zu Protokollen in Folge 15 erschließt uns bereits der Name dieses Bausteins seine grundsätzliche Funktion. Er ermöglicht eine zielgerichtete Kommunikation auf Basis Ethernet / UDP nach dem Master-Slave-Prinzip, wobei die Anwenderdaten Modbus-Format haben. Der Baustein steht als Bibliothek „*ModbusEthernet_x.lib*“ in verschiedenen Versionen zur Verfügung. Seine Anwendung erfordert weiter auch die Einbindung der Bibliotheken „*System.lib*“ und „*Ethernet.lib*“ in das Projekt.

Nachfolgend wollen wir die gleiche Kommunikationsaufgabe - wie bereits mit Netzvariablen ausgeführt - mit diesem Baustein lösen. Zusätzlich werden die erforderlichen Schritte gezeigt, um auch Daten aus dem Slave herauszulesen. Es sind zwei getrennte Programme zu schreiben, eines für den Master und eines für den Slave. Im Masterprogramm werden zyklischen Signale erzeugt und dem Baustein „*Ethernet_Modbusmaster_UDP*“ zum Versand zur Verfügung gestellt. Im Slaveprogramm könnten diese genutzt und verarbeitet werden. Für die Parametrierung des Masters ist die Funktionsbausteinsprache sehr anschaulich. **Bild 93** zeigt wesentliche Teile dieses Programms. Kernstück ist Netzwerk 5 mit der Instanzierung des Masterbausteins. Netzwerke 1 bis 4 zeigen den Eintrag der zu schreibenden Daten in den als „Schreibspeicher“ angelegten Datenspeicher des Masters, von dem aus sie in den Slave übertragen werden. Er besteht aus einem ARRAY von 3 Worten, von denen im dritten Wort allerdings nur zwei Bit genutzt werden. Netzwerk 6 beinhaltet die Erzeugung eines Taktes für die intervallgesteuerte Kommunikation des Masters mit dem Slave.

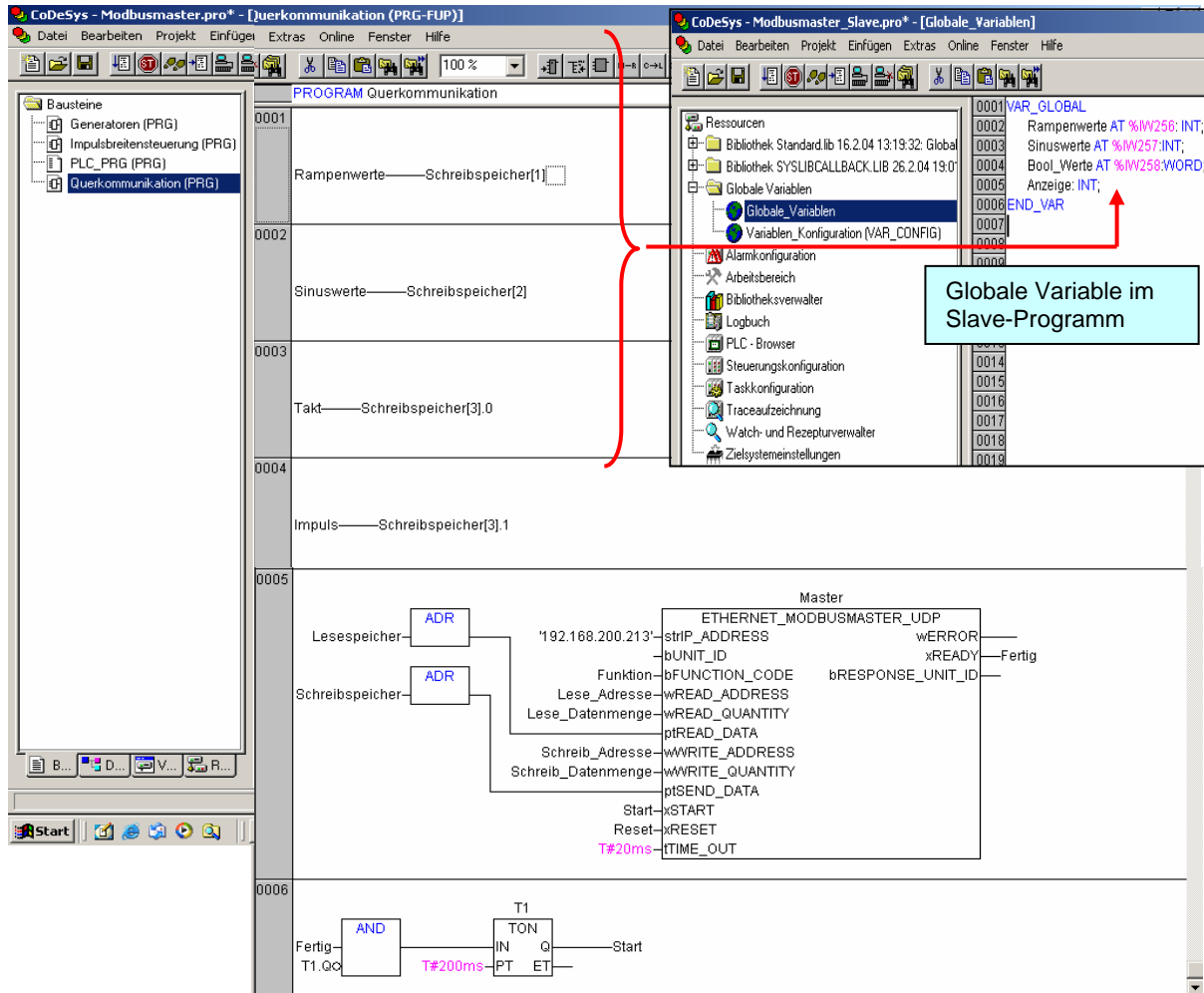


Bild 93: Auszüge des Programms für die Kommunikation mittels Ethernet-Modbusmaster in Funktionsbausteinsprache, rechts oben der Eintrag der Daten in Globale Variable des Slave

Die Details lassen sich in der ausführlich kommentierten Variablendeklaration zu dieser POE verfolgen (**Bild 94**). Es wurden dort bewußt Variablennamen gewählt, welche unmittelbar auf die Namen der Parameter des FB und weiter auch auf Aufgabe und Funktion der Variablen verweisen. Folgende Funktionen sind abzulesen:

- Der Master kommuniziert mit einem Slave mit der IP 192.168.200.213
- Es wird ein Wort aus der Adresse %QW256 des Slave ausgelesen.
- Es werden drei Worte in den Slave auf die Adressen %IW256 bis 258 geschrieben, wobei von den 16 Bit des dritten Wortes nur zwei Bit genutzt werden.
- Die Kommunikation zwischen Master und Slave erfolgt in Intervallen von 200ms.
- Im Slave erfolgt der Eintrag der geschriebenen Werte in dafür bereitgestellte globale Variable. Diese können in einem hier nicht weiter betrachteten Slaveprogramm genutzt werden.

Weitere Einzelheiten können dem Anwenderhinweis A 30002 „Querkommunikation mit Ethernet Controllern 750-842“ (www.wago.com) entnommen werden. Dort sind u.a. auch die Modbus-Funktionen sowie Hinweise zur Korrespondenz von Modbus- und IEC-Adressen angeführt.

PROGRAM Querkommunikation VAR Master:Ethernet_Modbusmaster_UDP; Funktion:BYTE:=16#17; Lese_Adresse:WORD:=16#0100; Lese_Datenmenge:WORD:=16#0001; Lesespeicher:WORD; Schreib_Adresse:WORD:=16#0100; Schreib_Datenmenge:WORD:=16#0003; Schreibspeicher:ARRAY[1..3] OF WORD; Start: BOOL; Fertig: BOOL; Reset: BOOL := FALSE; T1: TON;	Kommentar: Instanzierung des Bibliotheksbausteins Nach dem Schlüssel der Modbus-Funktionen wurde hier die Funktion 23 (Lesen und Schreiben) gewählt. Dieser Wert ist im Hexacode eingegeben. Adresse im Datenspeicher des Slave, aus dem heraus der Master Daten im Slave liest.Sie wurde auf 16#0100 eingestellt, was der Dezimal-Adresse 256 entspricht. Hier wird der Umfang der gelesenen Daten deklariert, im Beispiel wird ein Wort gelesen. Deklaration des Speichers für die gelesenen Daten im Master. Mit dem Operator ADR wird aber nicht auf den Inhalt, sondern auf die Adresse des Lesespeichers verwiesen. Adresse im Datenspeicher des Masters, aus dem heraus auf den Slave geschrieben wird. Sie wurde wie auch die des Lesespeichers im Slave auf das erste dafür verfügbare Wort 256 eingestellt. Deklaration des Umfangs der geschriebenen Daten, hier 3 Worte Deklaration des Speichers für die zu schreibenden Daten. Mit dem Operator ADR wird aber nicht auf den Inhalt, sondern auf die Adresse des Speichers verwiesen. Diese Variablen dienen der Steuerung des Zeitfensters, zu dem der Master schreibend und lesend mit dem Slave kommuniziert. Im Beispiel erfolgt dies in Intervallen von 200ms.
--	--

Bild 94: Deklaration der Variablen des Programms „*Querkommunikation*“ nach Bild 93

Der Eintrag der zu schreibenden Werte in den Schreibspeicher des Master ist noch genauer zu untersuchen: Im Masterprogramm werden die vier globalen Variablen Rampenwerte:INT; Sinuswerte:INT; Takt:BOOL und Impuls:BOOL deklariert. Deren aktuellen Werte werden Generatoren mit den entsprechenden Funktionen bzw. einer hier nicht dargelegten Pulsbreitensteuerung entnommen. In den Netzwerken 1 bis 4 des Bildes 93 erfolgt der Eintrag der Variablen in die Felder [1] bis [3] des Schreibspeichers. Bei den Booleschen Variablen „*Takt*“ und „*Impuls*“ erfolgt mit der Schreibweise [3].0 bzw.[3].1 der bitweise Zugriff auf ein Feld vom Typ WORD.

Sowohl für Master als auch Slave werden Adressen ab 256_{dez} gewählt, weil der Adressbereich 0...255_{dez} für die Prozessabbilder der Eingangs- und Ausgangs-Busklemmen reserviert sind. Dieser Bereich wird aber nur bei maximalem Ausbau der Systeme genutzt. Adressen ab 256_{dez} sind frei für externe, über das Netz geleitete Variablen.

Bild 95 zeigt schließlich die Online-Kontrolle der Funktion des Modbusmasters beim Schreiben der Daten in den Slave.

The screenshot displays the 'pro*' software interface for monitoring a Master's write functions. The interface is divided into three main sections:

- Parameter List (Left):** Shows configuration for the Master (Funktion = 16#17) and its write functions. Parameters include 'Lese_Adresse = 16#0100', 'Lese_Datenmenge = 16#0001', 'Lesespeicher = 16#0000', 'Schreib_Adresse = 16#0100', 'Schreib_Datenmenge = 16#0003', and three 'Schreibspeicher' entries (1, 2, 3) with addresses 16#0061, 16#FFE8, and 16#0003. The 'Fertig' status is highlighted as TRUE.
- Data Table (Right):** Shows real-time values for various signals: 'Rampenwerte (%IW256) = 16#0047', 'Sinuswerte (%IW257) = 16#FFC8', 'Bool_Werte (%IW258) = 16#0003', and 'Anzeige = 16#FFC8'.
- Diagram (Bottom):** Illustrates the connection between the Master and Slave. The Master block includes signals like 'strip_ADDRESS', 'bFUNCTION_CODE', 'wREAD_ADDRESS', 'wWRITE_ADDRESS', 'xSTART', and 'xRESET'. The Slave block includes 'wERROR', 'xREADY', and 'bRESPONSE_UNIT_ID'. The 'Fertig' signal is shown as a blue line connecting the two.

Bild 95: Komfortable Online-Beobachtung der Schreibfunktionen des Master

Fazit:

Nachdem Folge 15 in grundlegende Fragen zu Protokollen rund um die Ethernet-Kommunikation einführte, wurden in dieser Folge zwei Möglichkeiten der Kommunikation zwischen Automatisierungskomponenten über Ethernet / TCP / UDP praktisch ausgeführt. Es sind dies die Broadcast-Kommunikation mit Netzvariablen und die gerichtete Kommunikation mit einem Master-Slave-Prinzip. Mit den Werkzeugen des Programmiersystems CoDeSys gelingt die Kommunikation mit Netzvariablen ohne Programmieraufwand problemlos. Die Master-Slave-Kommunikation erfordert Kenntnisse über parametrierbaren Bibliotheksbausteine und Adressen, ist aber hinsichtlich der Buslast von Vorteil. Die nächste Folge stellt einige IT-Funktionen vor, denn deren Nutzung ist einer der Gründe, warum sich Ethernet zur tragenden Säule der Vernetzung von Automatisierungskomponenten entwickelt.

Glossar:

Broadcast	engl. Breitwurf, auch Rundfunk. Methode der Datenübertragung, bei der die Telegramme an alle Teilnehmer des Netzes und nicht an spezielle Empfänger gesendet werden, vergleichbar mit einer Rundfunksendung
Buslast	Zeitliche Dichte der Datenübertragung in einem Bussystem. Diese wird bestimmt von der Anzahl der zu „meldenden“ Ereignisse im System und kann unnötig hoch werden, wenn beispielsweise Sensoren unnötig oft abgefragt werden oder unnötiger Weise ihren „OK-Zustand“ melden.
Netzvariablen	Bezeichnung für globale Variablen, welche nach dem Rundfunk-Prinzip alle Teilnehmer eines Netz oder Teilnetzes erreichen.
Parametrierung	Bei der Arbeit mit Softwaretools unterscheidet man bewußt Parametrierung und Programmierung. Beim Parametrieren werden lediglich aktuelle Werte in vorgegebene zumeist grafische Masken eingegeben.