12. Vernetzung von Automatisierungskomponenten mit „Standard Ethernet“

12.1 Ethernet in der Automatisierungstechnik am Beispiel System WAGO-IO-750

Dezentralisierung und Vernetzung sind aktuelle Trends der Automatisierungstechnik. Der Streit um geeignete Bussysteme für die Vernetzung füllte über ein Jahrzehnt die Fachpresse und bleibt weiterhin aktuell. Gegenwärtig ist die Vernetzung mit Systemen, die auf Ethernet TCP/IP aufbauen, deutlich als Zukunftstechnologie der Automatisierungstechnik zu erkennen. Jedoch wird es ein einheitliches Bussystem auf dieser Basis nicht geben, weil an verschiedenen Protokoll-Erweiterungen für „harte Echtzeitfähigkeit“ gearbeitet wird (siehe Abschnitt 2.4).

In der Fachpresse wird von der stetig wachsenden Zahl von Vernetzungen mit dem „Standard“ Ethernet berichtet. Dabei werden drei entscheidende Vorzüge gegenüber anderen Feldbussen genannt:

- **Die Möglichkeit einer flachen Architektur der Automatisierungslösung**

Das Argument der flachen Struktur meint, dass die hierarchische Automatisierungsstruktur in Form der Pyramide mit Feld-, Automatisierungs- und Leitebene wie im **Bild 1-3** gezeigt immer weniger Bestand hat. In diesen Ebenen wurden überwiegend unterschiedliche Bussysteme eingesetzt. Ethernet aber kann als durchgängiges Kommunikationssystem diese Grenzen überwinden und macht darüber hinaus Internet-Technologien verfügbar.

- **Die durchgängige Kommunikation und Datenhaltung von der automatisierten Produktion bis hin in die Leit- und Managment-Ebene**

Dieses Argument verweist auf die vorteilhafte Verbindung von Produktions- und Officewelt ohne aufwändige Datenkonvertierungen. Die genannten Vorteile setzen aber voraus, dass Ethernet die Funktionen spezieller, sehr schneller Feldbusse übernehmen kann, ohne seine angestammten Eigenschaften und Aufgaben im IT-Bereich aufzugeben!

- **Die Nutzung von IT-Standards wie z.B. Maildienste für Diagnose und Service**

In Abschnitt 2 wurde bereits dargelegt, dass Ethernet kein Standard im eigentlichen Sinne ist, dass die verwendeten Protokolle näher geprüft werden müssen und dass es unterschiedliche Protokoll-Erweiterungen für das Paket Ethernet TCP/IP gibt. Sehr salopp ausgedrückt: Nicht überall, wo Ethernet „draufsteht“, ist das „gleiche Ethernet“ drin!

Das Busklemmensystem WAGO-IO-750 verwendet für die Vernetzung von Systemen mit Ethernet-Controllern 841 und Ethernet-Kopplern 341 gemäß **Bild 12-1** sogenanntes **Standard Ethernet TCP/IP**, wobei die Nutzdaten als Modbus-Daten „verpackt“ werden.

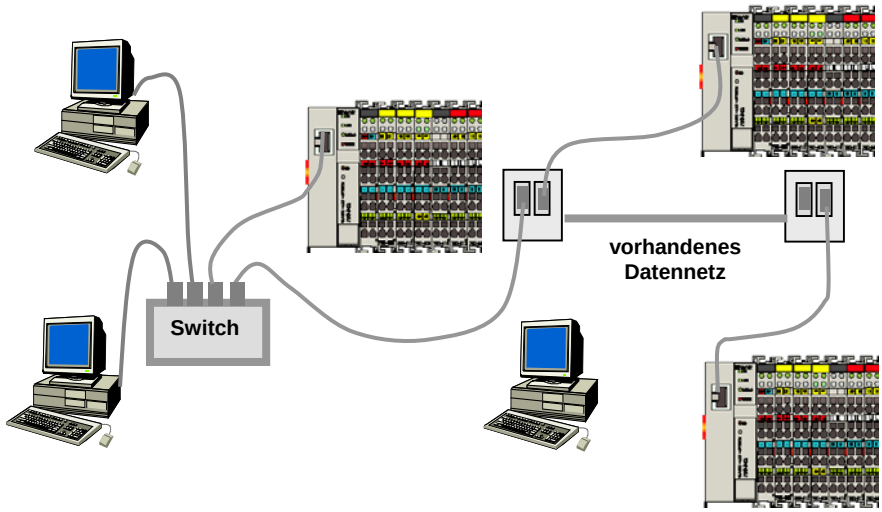


Bild 12-1: Vernetzung von Busklemmensystemen WAGO-IO-750 mit Controllern und Kopplern über Standard Ethernet TCP/IP

In Ergänzung zum Abschnitt 2.2 zeigt **Bild 12-2** die hierbei genutzten Schichten im OSI-Modell. Dargestellt ist die fortwährende Erweiterung der Header der Datentelegramme durch Port-Nr., IP-Adresse und Ethernet-Adresse beim Durchlaufen der Schichten. Nur die im Bild grau hinterlegte Bitübertragungsschicht 1 und die Sicherungsschicht 2 verkörpern Ethernet.
Die Anwendung des „normalen“ Ethernet – Protokolls TCP/IP ohne spezielle Echtzeit-Erweiterungen hat den Vorteil, dass der „Mischbetrieb“ von Automatisierungs- und Bürokomponenten ohne zusätzliche Aufwände problemfrei verläuft. Für eine Vielzahl von Aufgaben der Automatisierungstechnik sind die hierbei zu erwartenden Reaktionszeiten zwischen 10 und 100 ms akzeptabel.

Für die Abwicklung der Kommunikation sind in den Programmierbaren Feldbuscontrollern 841 (PFC) die im **Bild 12-3** aufgeführten Protokolle implementiert. Bei den Transportprotokollen (Schicht 4) ist **neben TCP auch UDP** erwähnt. Während TCP für die Dauer der Kommunikation zweier Komponenten eine gesicherte Verbindung im Netz herstellt, arbeitet UDP (User Datagramm Protokoll) ohne eine solche spezielle gesicherte Verbindung. Dadurch erlaubt UDP eine deutlich schnellere Kommunikation.

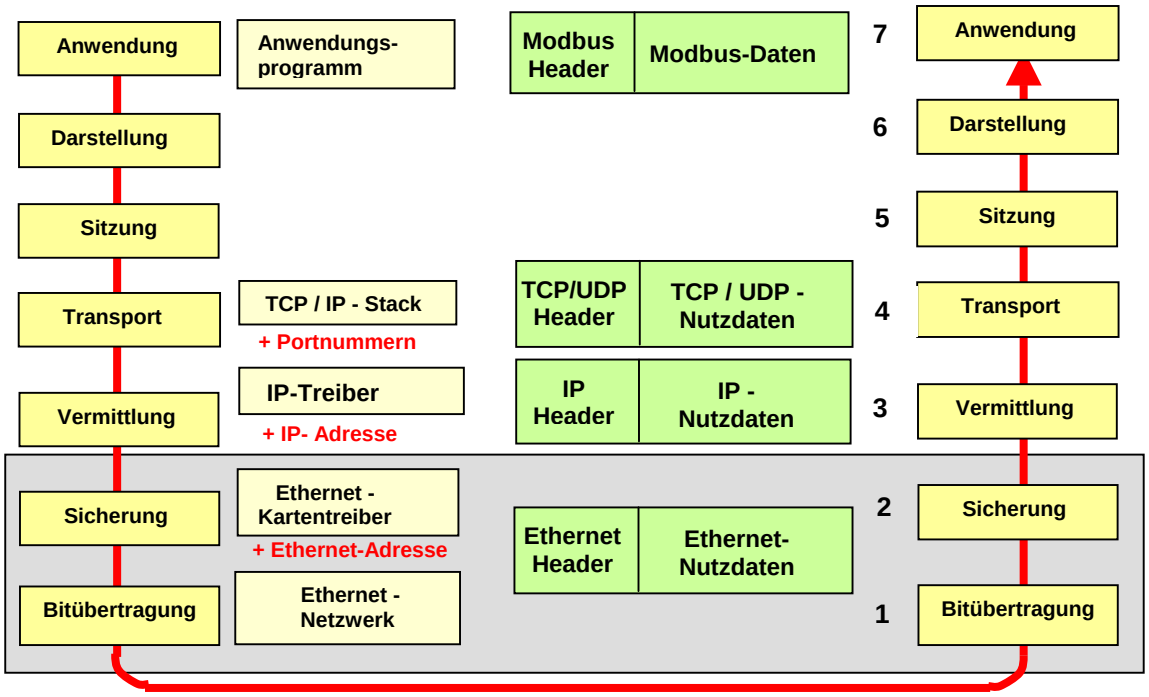
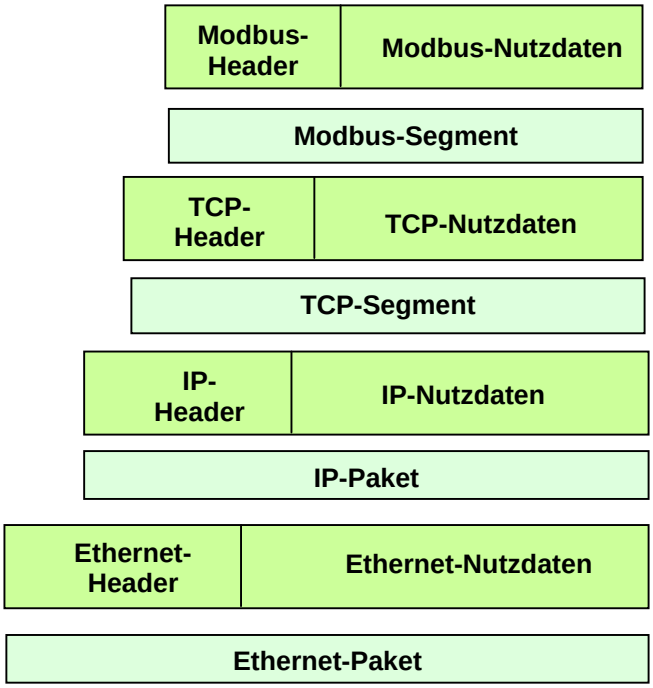


Bild 12-2: Schichten und Protokolle beim Versenden von Modbus-Anwenderdaten über Ethernet TCP / IP

Auf der Anwendungsebene (Schicht 7) stehen die Nutzdaten im „Modbus-Format“. **Modbus** wurde 1979 von Modicon (AEG, heute Schneider Elektrik) als Kommunikationsprotokoll entwickelt. Das Protokoll ist offengelegt und man darf Modbus im Verbund mit dem Transportprotokoll TCP inzwischen als eine Art „Industriestandard“ für die Handhabung von Anwenderdaten bezeichnen. Es arbeitet nach dem Client-Server-Prinzip: Für die Kommunikation bearbeitet eine Automatisierungskomponente mit Serverstatus die Anfragen von Clients mittels einer Reihe von Funktionscodes. Der Anwender hat dabei zumeist nur fertige Funktionsbausteine zu parametrieren.
Es gibt verschiedene Versionen für serielle Schnittstelle und für Ethernet:

- Modbus RTU (Remote Terminal Unit): Datenübertragung in binärer Form, hoher Datendurchsatz
- Modbus ASCII: Datenübertragung im ASCII-Code, dadurch lesbar, geringerer Datendurchsatz gegenüber RTU
- Modbus TCP:Datenübertragung mit TCP / IP – Datenpaketen, reservierter Port: 502

Jeder Teilnehmer muss eine eindeutige Adresse haben und darf Befehle über den Bus senden. In der Regel wird dies aber nur vom Master ausgeführt. Adresse 0 ist für Broadcast reserviert.
Details kann man auf der Seite **www.modbus.org** einsehen.



UDP: User Datagramm Protokoll, wie TCP
Sicherung des Datentransports im Netz, verbindungslos, hocheffizient
(siehe Handbuch WAGO Handbuch 750 - 841 S.130 ff)

TCP: aufgesetzt auf IP-Protokoll, für die Sicherung des Datentransports im Netz; stellt für Dauer der Übertragung eine Verbindung her
Nutzung einer Port-Nr.
Quittierung mit Acknowledgement-Nr.
(siehe WAGO Handbuch 750 - 841 S.129 ff)

IP Vers.4 für die Verbindungslose Übertragung von Ethernet-Paketen auf Übertragungsmedium, CSMA, IP-Adressierung, RAW- IP und IP- Multicast
(siehe WAGO Handbuch 750 - 841 S.123 ff)

ARP Protokoll für die Zuordnung von MAC- und IP-Adresse
Einsatz bei Datenübertragung im gleichen logischen Netz
(siehe WAGO Handbuch 750 - 841 S.130 ff)

Bild 12-3: Die implementierten Kommunikationsprotokolle des PFC 841
(„WAGO Handbuch 750-841“ steht für Handbuch WAGO-IO-System 750 Ethernet TCP/IP 841 -> www.wago.com)

Als **Anwendungsprotokolle** sind auf dem PFC 841 folgende Protokolle implementiert:
- Modbus mit Kommunikation über TCP / IP – Verbindungen über Port 502
- Modbus mit Kommunikation über UDP
- Ethernet Industrial Protocol als Erweiterung des Ethernet um Industrieforderungen, basierend auf TCP/ IP

Die Vielzahl unterschiedlicher Protokolle auf den verwendeten Schichten macht deutlich, dass für eine erfolgreiche Kommunikation zwischen zwei Anwenderprogrammen bzw. zwischen Automatisierungskomponenten alle Daten mit gleichem Protokoll verschlüsselt werden müssen.

Weiter stehen **für Verwaltung und Diagnose** –auch über Internet – auf dem PFC 841 zur Verfügung die Protokolle:
(siehe Handbuch 750 - 841 S.133 ff)
- BootP
- HTTP
- DNS
- SNTP
- FTP
- SNMP
- SMTP

12.2 Realisierung einer gerichteten Kommunikation mit einem Master-Slave-Prinzip

Anwenderhinweis A 30002 „Querkommunikation mit Ethernet Controllern 750-842“ (www.wago.com)

Bereits der Eintrag eines Programms in einen Controller über die Netzwerkkarte eines Programmierrechners und gekreuztes Patchkabel erfolgt über einen Kanal Ethernet TCP/IP (hierzu Teil 1 der Anleitung zum Praktikum). Auch die Online-Beobachtung eines Programms basiert auf dieser Kommunikation.

Nunmehr soll das Netzwerk durch Zuschalten weiterer Controller oder Koppler erweitert werden. Ziel ist der Datenaustausch zwischen den Automatisierungskomponenten (**Bild 12-4**).

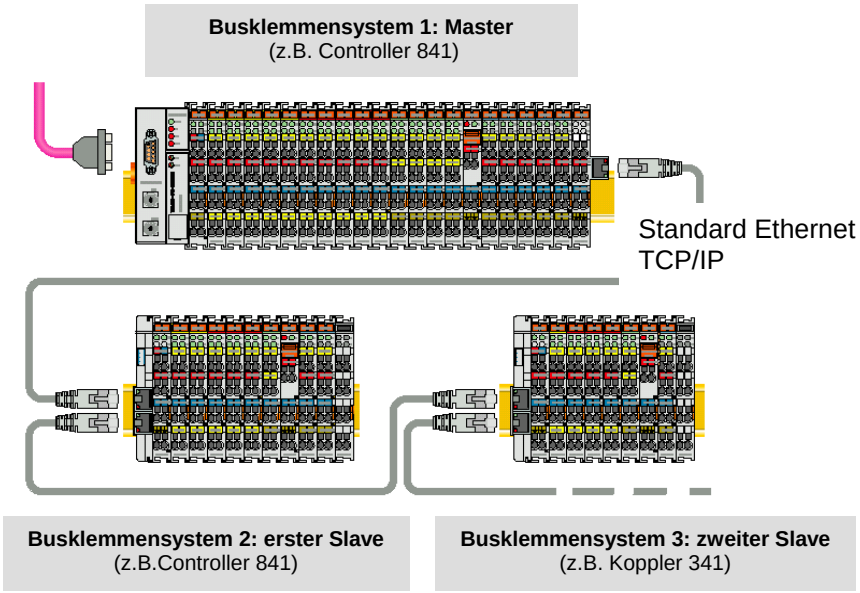


Bild 12-4: Datenausstausch zwischen Ethernet-Controllern und -Kopplern

Die Datenübertragung über Standard-EthernetTCP/IP bringt nochmals genannt folgende Vorteile:

- Zugang zum Automatisierungssystem über die Standard Netzwerkkarte des PC. Keine zusätzlichen Karten mit speziellem Treiber!
- Komponenten und vorhandene Netze des „Weltstandards Ethernet“ können genutzt werden.

- Als zusätzlicher Vorteil steht mit Ethernet_Modbusmaster_UDP eine übersichtliche und zeitgesteuerte Kommunikation zwischen Controllern und Kopplern zur Verfügung. Dafür wird die Datenübertragung mit ein **Master-Slave-Prinzip** vollzogen, obwohl dieses bewährte Prinzip nicht typisch für die Kommunikation zwischen Ethernet-Netzkomponenten ist! Vom Grundsatz her sind Ethernet-Komponenten gleichberechtigt. Master-Slave-Strukturen kennt der Automatisierungstechniker u.a. vom Bussystem Profibus-DP.

Grundsätzliche Eigenschaften des Master-Slave-Prinzips:

- Das Master-Slave-Prinzip realisiert eine **gerichtete** Verbindung zwischen Automatisierungskomponenten. Das erlaubt eine geringere Buslast gegenüber einem Broadcast-Prinzip.

- Allein der Master hat **Schreib- und Leserechte** und fragt der Reihe nach zyklisch in determinierten Zeitfenstern alle Slaves in seinem Netz ab.

- Slaves verhalten sich bei der Kommunikation passiv, empfangen Daten und stellen Daten zur Verfügung. Der Slave kann sowohl ein PFC als auch ein einfacher Koppler sein. Verfügt ein Slave über „Intelligenz“, so kann er Programme und Daten verarbeiten. Lese- und Schreibrechte hat er dennoch nicht.

- Der Master muss programmierbar sein. Hier werden die Parameter der Kommunikation hinterlegt. Die Programmierung ist so erstellen, dass möglichst wenig Datenverkehr auf dem Bus stattfindet!

Parametrierung des Master-Slave-Systems:

Der Name des speziellen Bausteins „*Ethernet_Modbusmaster_UDP*“ erklärt seine grundsätzliche Funktion: Aufbau einer gerichtete Kommunikation nach dem Master-Slave-Prinzip auf Basis Ethernet / UDP, wobei die Anwenderdaten Modbus-Format haben. Der Baustein steht als Bibliothek „*ModbusEthernet_x.lib*“ in verschiedenen Versionen zur Verfügung. Seine Anwendung erfordert zusätzlich auch die Einbindung der Bibliotheken „*System.lib*“ und „*Ethernet.lib*“ in das Projekt.

Der Bausteins wird im Master angelegt und dort parametriert. Sehr anschaulich ist hier die Funktionsbausteinsprache (**Bild 12-5**). Zu parametrieren sind
1. die Schreibfunktion für die Daten, welche zum Slave geschrieben werden
2. die Lesefunktion für Daten des Slaves.
Selbstverständlich ist der Funktionsbaustein zu instanzieren.

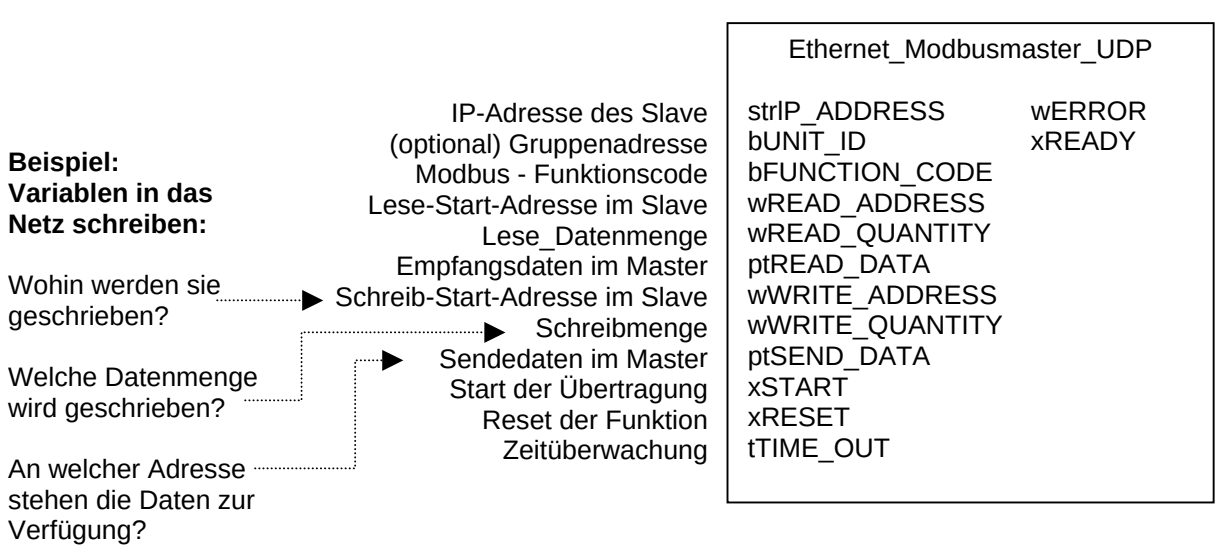


Bild 12-5: Die Parameter des Funktionsblocks „Ethernet_Modbusmaster_UDP“

Die Vorgehensweise beim Parametrieren einer Master-Slave-Kommunikation wird nachfolgend an einem Beispiel erklärt: Für den Test der Datenübertragung eignen sich besonders gut periodische Signale. Vier solcher Signale mit Signalformen Takt, Sinus und Dreieck (Rampe) sowie ein pulsbreitengesteuerter Impuls sollen vom Master zu einem Slave übertragen werden. Für Simulationszwecke stellt der Funktionsbaustein „*GEN*“ der Bibliothek „*util.lib*“ die ersteren Signale zur Verfügung. Sie sind von Datentyp BOOL bzw. INT.

Der untenstehende Kasten zeigt zunächst die Variablendeklaration eines Programms, in welchem eine Instanz des Masterbausteins eingefügt wurde. Zur Erklärung der einzelnen Variablen wurden diese ausführlich kommentiert. 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.

Bild 12-6 zeigt wesentliche Teile des Programms. Kernstück ist Netzwerk 5 mit der Instanzierung des Masterbausteins. In den Netzwerken 1 bis 4 erfolgt der 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.
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 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;	**Kommentar:** Instanz des Bausteins Ethernet_Modbusmaster_UDP
Funktion: BYTE:=16#17;	Nach dem Schlüssel der Modbus-Funktionen wurde hier die Funktion 23 (Lesen und Schreiben) gewählt. Dieser Wert ist im Hexacode eingegeben.
Lese_Adresse:WORD:=16#0100;	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.
Lese_Datenmenge:WORD:=16#0001;	Hier wird der Umfang der gelesenen Daten deklariert, im Beispiel wird ein Wort gelesen.
Lesespeicher:WORD;	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.
Schreib_Adresse:WORD:=16#0100;	Adresse im Datenspeicher des Slave, auf den geschrieben wird. Sie wurde wie auch die des Lesespeichers im Slave auf das erste dafür verfügbare Wort 256 eingestellt.
Schreib_Datenmenge:WORD:=16#0003;	Deklaration des Umfangs der geschriebenen Daten, hier 3 Worte
Schreibspeicher:ARRAY[1..3] OF WORD;	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.
Start: BOOL; Fertig: BOOL; Reset: BOOL := FALSE; T1: TON; END_VAR	Diese Variablen dienen der Steuerung des Zeitfensters, in welchem der Master schreibend und lesend mit dem Slave kommuniziert. Im Beispiel erfolgt dies in Intervallen von 200ms.

Für die Festlegung der Adressen im Datenspeicher, von dem aus das Schreiben bzw. Lesen von Daten erfolgt, wird der **Operator ADR** eingesetzt. Dieser IEC Operator liefert in einem Doppelwort die Anfangs-Adresse seines Argumentes. Ein solcher Operator ist erforderlich, weil das System den Datenspeicher selbständig verwaltet und die Adressen im einzelnen nicht bekannt sind.

Zur Festlegung der **MODBUS-Funktion** (im Beispiel FC 23 bzw. hexadezimal 16#23):
Im WAGO Feldbus-Controller ETHERNET TCP/IP sind eine Reihe von MODBUS-Funktionen aus der OPEN MODBUS / TCP SPECIFICATION (www.modbus.org) implementiert (siehe Handbuch WAGO-IO-System 750 Ethernet TCP/IP 841 Abschnitt Modbusfunktionen S.149 ff).

Nachfolgende Tabelle zeigt diese verfügbaren MODBUS-Funktionen. Einzelheiten sind dem Handbuch zu entnehmen.

Funktions-code	**Funktions-code hex**	**MODBUS-Funktion**	**Beschreibung**
FC1	01	read coils	Lesen mehrerer Eingangsbits
FC2	02	read input discretes	Lesen mehrerer Eingangsbits
FC3	03	read multiple registers	Lesen mehrerer Eingangsregister
FC4	04	read input registers	Lesen mehrerer Eingangsregister
FC5	05	write coil	Schreiben mehrerer Ausgangsbits
FC6	06	write single registers	Schreiben eines einzelnen Ausgangsregisters
FC7	07	read exception status	Lesen der ersten acht Eingangsbits
FC11	0B	get comm event counters	Kommunikationsereigniszähler
FC15	0F	force multiple coils	Schreiben mehrerer Ausgangsbits
FC16	0010	write multiple registers	Schreiben mehrerer Ausgangsregisters
FC23	**0017**	**read / write multiple registers**	**Lesen und Schreiben mehrerer Ausgangsregister**

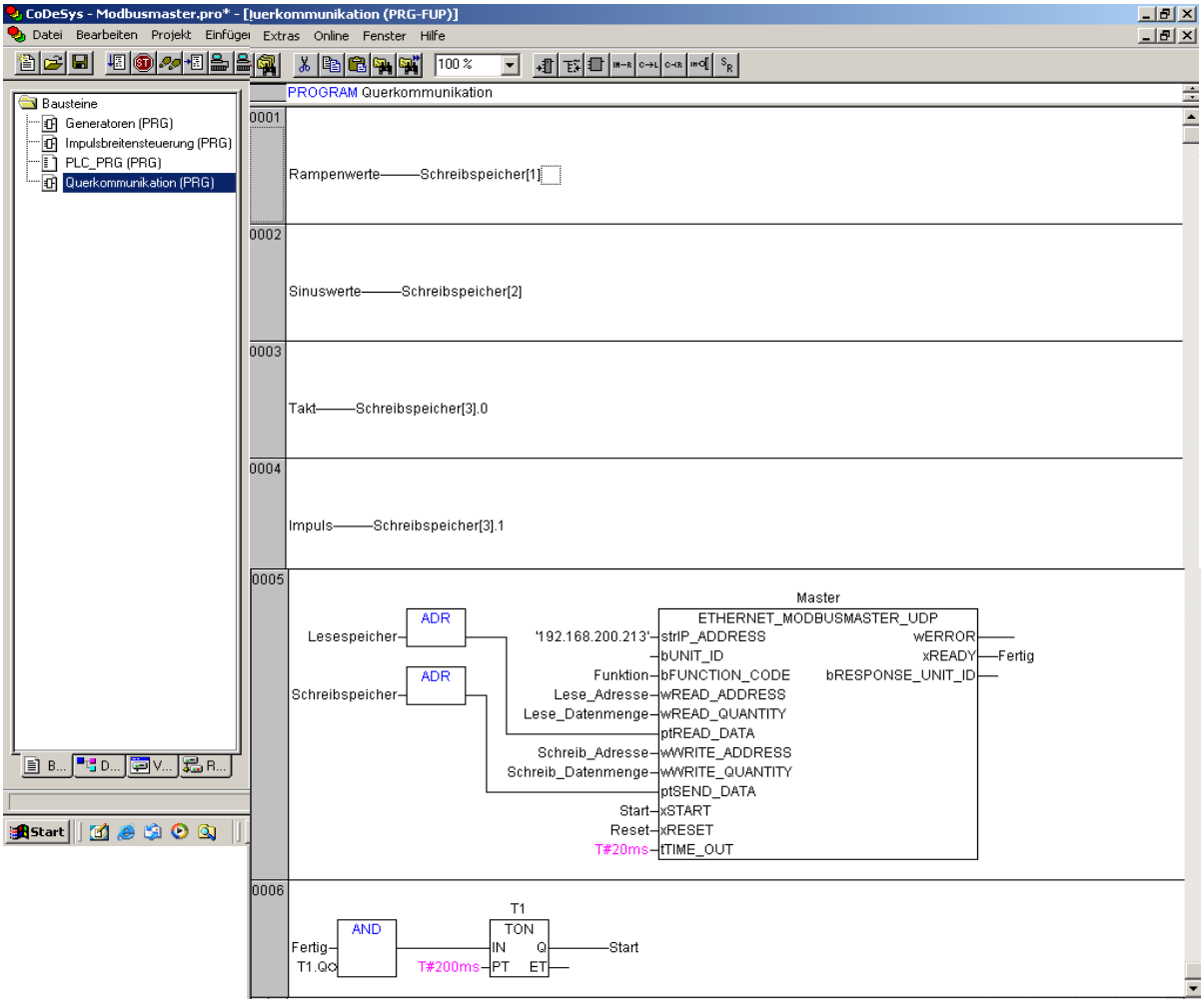


Bild 12-6: Auszug des Programms zur Übertragung periodischer Signale an den Slave mit IP 192.168.200.213 in FUP

Für weitere Teilnehmer im Netz könnte man für jeden Slave eine eigenständige weitere Instanz des Ethernet-Modbusmaster-UDP im Programm einbinden oder aber einen einzigen Master-Baustein **zeitgesteuert umparametrieren**.

Zu begründen sind die **Adressen im Speicherbereich des PFC 841**, die für Daten des Feldbus zur Verfügung stehen (**Bild 12-7**). Im o.a. Programm wird auf Slave-Adresse %IW256 geschrieben und auch von dortiger Adresse %QW256 gelesen.

Der Controller hat neben 512 kByte Programmspeicher einen Datenspeicher von 128 kByte (nebst 24 kByte Retainspeicher). Davon sind für die Prozessabbilder 256 Worte (0 ... 255) reserviert. Ab Wort 256 (16#0100) stehen Speicher für Daten zur Verfügung, die über den Ethernet-Feldbus geschrieben und gelesen werden können.

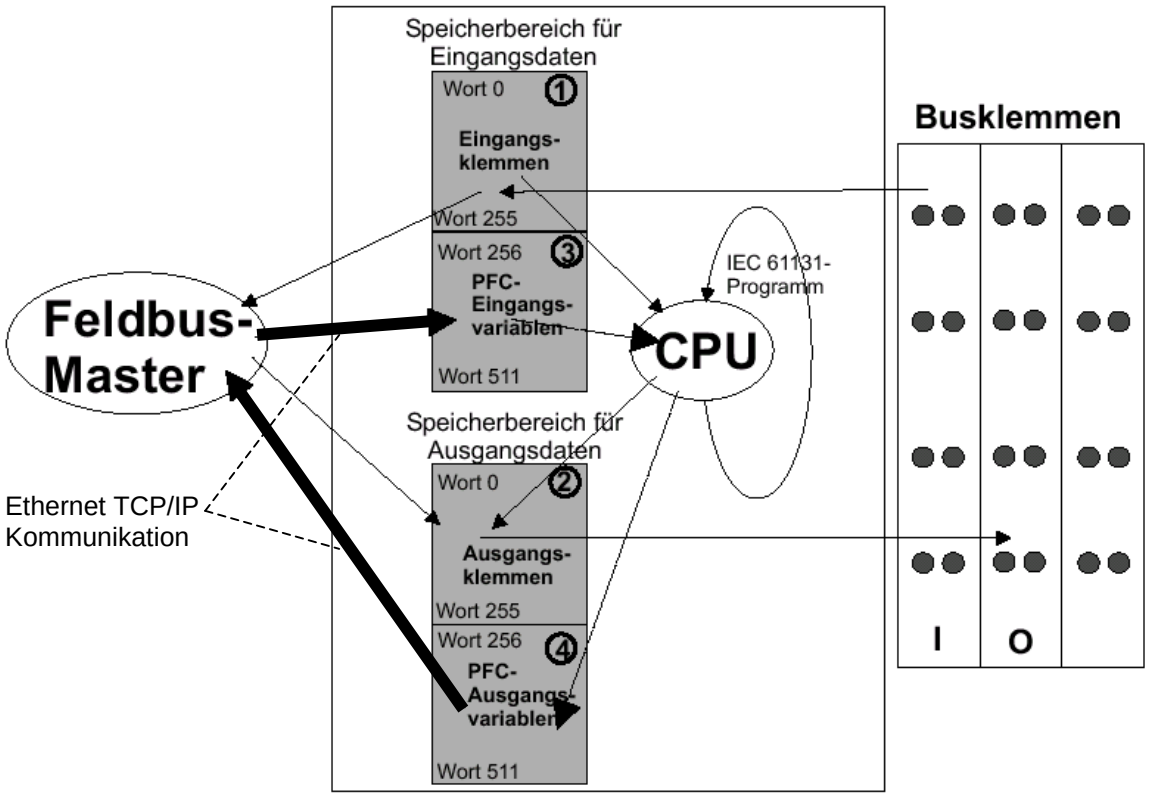


Bild 12-7: Auszug aus dem Handbuch WAGO-IO-System 750 Ethernet TCP/IP 841
Abschnitt Datenaustausch S.86:
Speicherbereiche für den Datenaustausch über Ethernet TCP/IP

Die zum Slave geschriebenen Daten müssen selbstverständlich dort weiterbearbeitet werden, entweder bei programmierbarem Slave mittels Programm oder aber bei Kopplern durch Weiterleitung auf Busklemmen. Nachfolgend ist ein Beispiel gezeigt. Der Empfang der geschriebenen Daten Rampe, Sinus, Takt und Impuls im Slave wird hier durch globale Variablen gesichert, die auf die entsprechenden Wort-Adressen gelegt wurden. Der Takt kann dann durch Bool_Werte.0, der Impuls durch Bool_werte.1 selektiert werden. Mit diesen Variablen ist danach jede Art Verarbeitung der Signale im Slave denkbar.

```
VAR_GLOBAL
Rampenwerte AT %IW256: INT;
Sinuswerte AT %IW257:INT;
Bool_Werte AT %IW258:WORD;
Anzeige: INT;
END_VAR
```

Vom Slave zu lesende Daten sind in gleicher Weise auf den Worten ab %QW256 bereitzustellen.

12.3 Realisierung einer „Broadcast – Kommunikation“ mit Netzwerkvariablen

Nachfolgend soll die Kommunikationsaufgabe nach Abschnitt 12.2 mit Netzvariablen gelöst werden. Hierzu 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. Vorteilhaft für den Anwender ist, dass allein durch Festlegung von Namen und Datentyp eine Kommunikation zwischen den Automatisierungskomponenten erfolgt.

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. Sie ist allerdings mit höherer **Buslast** verbunden.

Für die Parametrierung der Kommunikation muss zuerst in den Einstellungen des Zielsystems die Kommunikation mit **Netzvariablen freigeschaltet** werden (**Bild 12-8** rechts). Im Hauptordner Globale Variablen wurde über -> *Einfügen* ein spezielles Verzeichnis mit Namen „*Globale_Var_Schreiben“* angelegt. Es wurden die vier periodischen Variablen mit den Signalformen Takt, Sinus und Dreieck (Rampe) sowie ein pulsbreitengesteuerter Impuls gemäß Abschnitt 12.2 deklariert.

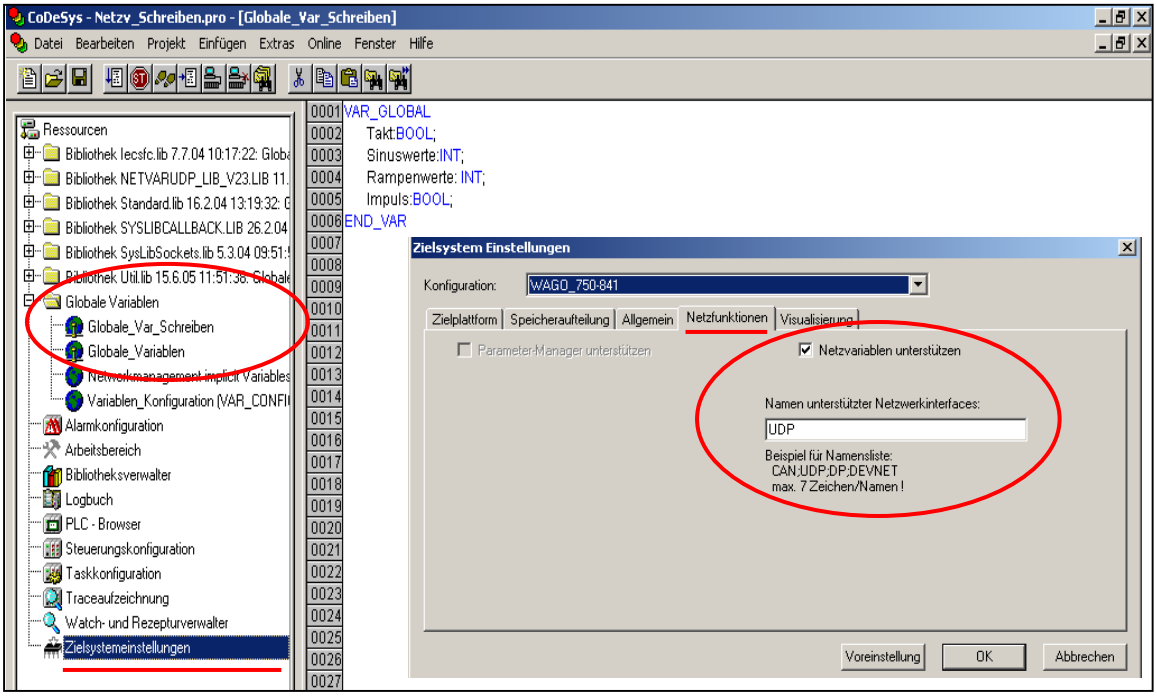


Bild 12-8: Freischalten (rechts) und Deklaration von Netzvariablen für den PFC 750-841

Im zweiten Schritt werden über die -> *Eigenschaften* dieser globalen Variablenlisten die Attribute Lesen oder Schreiben deklariert und Verbindungen zugeordnet. **Bild 12-9** zeigt einige Details dieser Arbeit.

Die globale Variablenliste nach Bild 12-9 gilt für eine erste Verbindung (Connection 1), weitere Verbindungen könnten angelegt werden. Erkennbar ist die „Verpackung“ der Daten gemäß UDP-Protokoll. Über -> *Einstellungen* gelangt man 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.

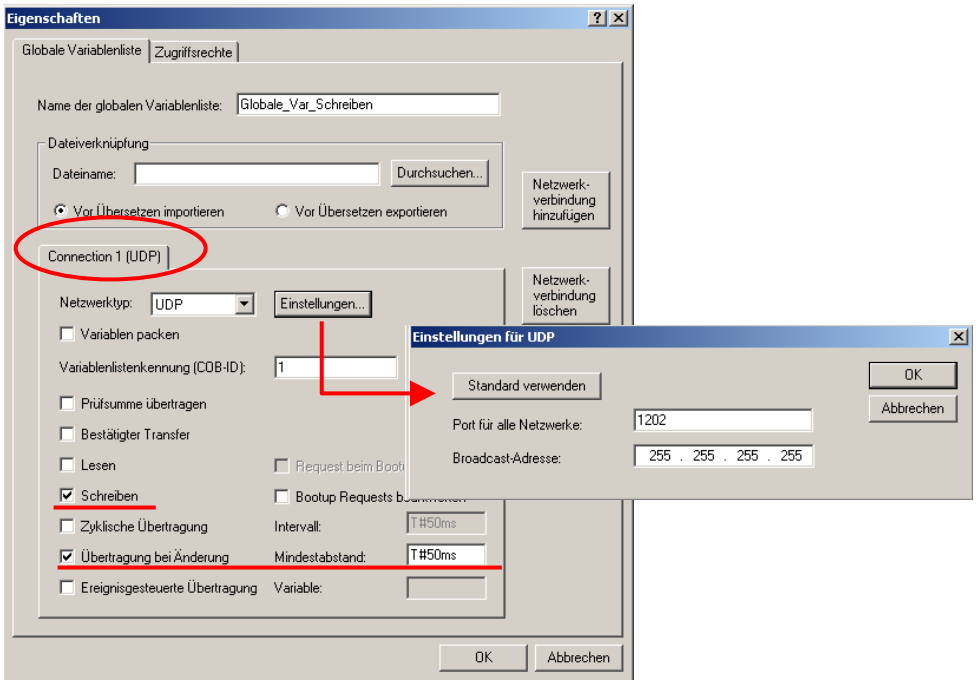


Bild 12-9: Festlegung der Eigenschaften der Netzvariablen

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

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 12-10** zeigt die Vorgehensweise. Mit dem Menu -> *Projekt* -> *Exportieren* erscheinen alle Komponenten des Projektes, aus denen die gewünschten Objekte für einen Export 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.

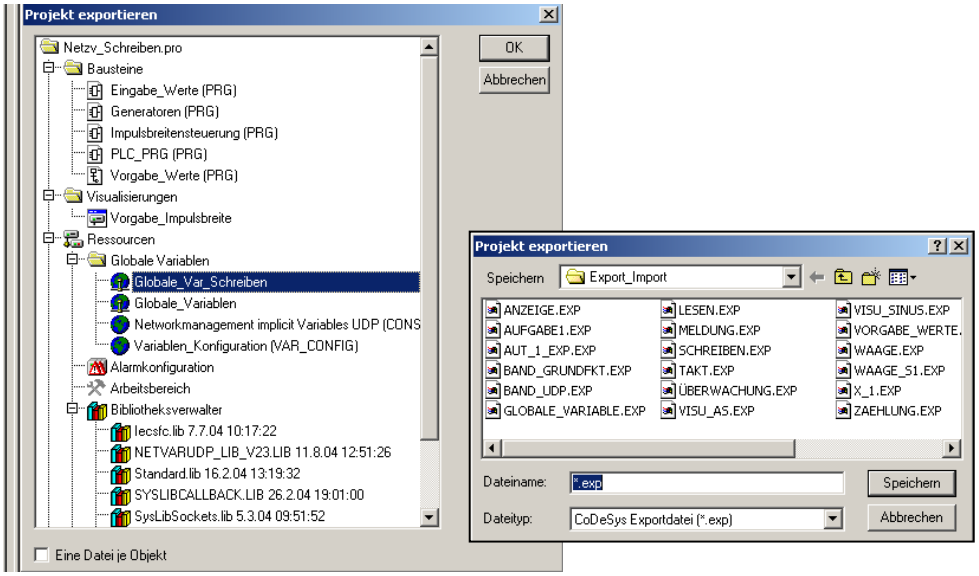


Bild 12-10: Export der Liste von Netzvariablen, um „synchrone“ Variablen zu erhalten