SIEMENS

SIMATIC

SIMATIC Modbus/TCP Redundante Kommunikation über die integrierte PN-Schnittstelle der H-CPUen

Handbuch

SIEMENS

SIMATIC S7

SIMATIC Modbus/TCP Redundante Kommunikation über die integrierte PN-Schnittstelle der H-CPUen

Handbuch

Ausgabe 1.1

Sicherheitstechnische Hinweise

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit, sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise sind durch ein Warndreieck hervorgehoben und je nach Gefährdungsgrad folgendermaßen dargestellt:

Gefahr

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

Warnung

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

Vorsicht

bedeutet, dass eine leichte Körperverletzung oder ein Sachschaden eintreten können, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Hinweis

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

Qualifiziertes Personal

Inbetriebsetzung und Betrieb eines Gerätes dürfen nur von **qualifiziertem Personal** vorgenommen werden. Qualifiziertes Personal im Sinne der sicherheitstechnischen Hinweise dieses Handbuchs sind Personen, welche die Berechtigung haben, Geräte, Systeme und Stromkreise gemäß den Standards der Sicherheitstechnik in Betrieb zu nehmen, zu erden und zu kennzeichnen.

Bestimmungsgemäßer Gebrauch

Beachten Sie folgendes:

Warnung

Das Gerät darf nur für die im Katalog und in der technischen Beschreibung vorgesehenen Einsatzfälle und nur in Verbindung mit von Siemens empfohlenen bzw. zugelassenen Fremdgeräten und -Komponenten verwendet werden.

Der einwandfreie und sichere Betrieb des Produktes setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung und Montage sowie sorgfältige Bedienung und Instandhaltung voraus.

Marken

SIMATIC® ist eingetragenes Warenzeichen der SIEMENS AG.

Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen können.

Siemens AG
Industry Sector
Industry Automation Division / Industrial Automation Systems
Factory Automation
I IA AS FA DH FTH 6
Postfach 23 55, D- 90713 Fürth

Haftungsausschluss

Wie haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, und notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Verbesserungsvorschläge sind wir dankbar.

Technische Änderungen bleiben vorbehalten.

Vorwort

Zweck des Handbuchs

Die Informationen dieses Handbuchs ermöglichen es Ihnen, eine Kopplung zwischen einer H-CPU mit integrierter PN-Schnittstelle und einem Gerät, welches das Protokoll Modbus/TCP unterstützt, aufzubauen und in Betrieb zu nehmen.

Inhalte des Handbuchs

Im vorliegenden Handbuch sind die Funktion des Modbus Funktionsbausteins und dessen Parametrierung beschrieben.

Das Handbuch beinhaltet folgende Themen:

- Produktbeschreibung
- Getting Started
- Inbetriebnahme
- Parametrieren der Modbus-Kommunikation
- Lizenzierung
- Funktionsbausteine
- Additional Blocks
- Diagnose
- Applikationsbeispiel

Gültigkeitsbereich des Handbuchs

Das vorliegende Handbuch ist gültig für folgende Software:

Produkt	Identifizierungsnummer	ab Version
Modbus/TCP PN CPU Redundant	6AV6676-6MB10-0AX0	1.0.1
FB 913 „TCP_COMM“		3.2
FB 914 „MOD_CLI“		1.6
FB 915 „MB_PNHCL“		1.0
FB 916 „MOD_SERV“		1.5
FB 917 „MB_PNHSV“		1.0

Hinweis

Das vorliegende Handbuch enthält die Beschreibung der FBs, wie sie zum Zeitpunkt der Herausgabe des Handbuchs gültig ist.

Weiterführende Informationsquellen

Alle weiteren Informationen bezüglich der PN-H-CPUen (Montage, Inbetriebnahme etc.) entnehmen Sie bitte dem Handbuch:

SIEMENS
SIMATIC Hochverfügbare Systeme S7-400H
Systemhandbuch
A5E00267693-11

SIEMENS
SIMATIC S7-400
Automatisierungssystem S7-400 CPU-Daten
Gerätehandbuch
A5E00850745-10

Weitere Informationen bezüglich STEP7 entnehmen Sie bitte den folgenden Handbüchern:

SIEMENS
SIMATIC Software
Basissoftware für S7 und M7
STEP7 Benutzerhandbuch
C79000-G7000-C502-..

SIEMENS
SIMATIC Software
Systemsoftware für S7-300/400
System- und Standardfunktionen
Referenzhandbuch
C79000-G7000-C503-02

Rückfragen

Bei Fragen zur Nutzung der in diesem Handbuch beschriebenen FBs, die Sie hier nicht beantwortet finden, wenden Sie sich bitte an Ihren Siemens-Ansprechpartner, von dem Sie diesen Funktionsbaustein erhalten haben.

Konventionen

In der vorliegenden Dokumentation wird im Folgenden die Bezeichnung CPU verwendet. Die Ausführungen sind für H-CPUen mit integrierter PN-Schnittstelle gültig.

Einsatzbereich

Die in diesem Handbuch beschriebenen Funktionsbausteine stellen eine Verbindung zwischen einer PN-H-CPU und Modbusgeräten anderer Hersteller dar.

Inhaltsverzeichnis

1 Produktbeschreibung

1.1 Einsatzmöglichkeiten

Einordnen in die Systemumgebung

Der vorliegende Funktionsbaustein stellt ein Software-Produkt für CPUen mit integrierter PN-Schnittstelle in einem SIMATIC S7-H-System dar.

Funktion der FBs

Mit diesen Funktionsbausteinen wird eine Kommunikation zwischen einer S7-H-CPU mit integrierter PN-Schnittstelle und einem Gerät, welches das Protokoll Modbus/TCP unterstützt, ermöglicht. Es werden die Funktionscodes 1, 2, 3, 4, 5, 6, 15 und 16 unterstützt.

Die Datenübertragung wird nach dem Client-Server-Prinzip abgewickelt.

Die SIMATIC S7 kann bei der Übertragung sowohl als Client als auch als Server betrieben werden.

Redundante Kommunikation wird unterstützt. Dabei ist die Verwendung sowohl in einem S7-400H-System möglich wie auch in einer S7-Single-PN-CPU.

Die Bausteine laufen im Hot-Standby-Betrieb. Hot Standby bezeichnet die parallele redundante Verarbeitung der Signale in redundanten Komponenten. Dadurch kann das Gesamtsystem stoßfrei auf die Standby-Komponenten umschalten.

Verwendung der Portnummer 502

Das Protokoll Modbus/TCP läuft üblicherweise über den Port 502. Diese Portnummer ist nur für PN-CPUen mit entsprechender Firmwareversion möglich. Die Information bezüglich der Freigabe der Portnummern finden Sie hier: http://support.automation.siemens.com/WW/view/de/34010717.

Bestimmte CPU-Typen können über den lokalen Port 502 Verbindungen zu mehreren Clients parallel halten und bedienen.
Im Kapitel „Mehrere Verbindungen auf Port 502“ sind die technischen Details zu diesem Thema erläutert.

1.2 Hard- und Softwarevoraussetzungen

Verwendbare Baugruppen für MB_PNHCL und MB_PNHSV

Die aktuellen Hardwarevoraussetzungen entnehmen Sie bitte hier: www.siemens.de/s7modbus.

Softwareausgabestände

Der Einsatz des FB MB_PNHCL bzw. MB_PNHSV ist ab **STEP 7-Version 5.5 SP2 HF1** möglich.

Speicherbedarf

Der FB MB_PNHCL benötigt 17 kByte Arbeitsspeicher und 20 kByte Ladespeicher.

Der FB MOD_CLI benötigt 10 kByte Arbeits- und Ladespeicher.

Der FB MB_PNHSV benötigt 14 kByte Arbeits- und 17 kByte Ladespeicher.

Der FB MOD_SERV benötigt 10 kByte Arbeits- und Ladespeicher.

Der FB TCP_COMM benötigt 2 kByte Arbeits- und Ladespeicher.

Die exakten Längen der Bausteine können Sie über deren Eigenschaften im SIMATIC Manager ermitteln.

2 Getting Started

Vorgehensweise

1. Installation von „SIMATIC Modbus/TCP PN CPU Redundant“ und Einfügen der Modbusbausteine in das Anwenderprojekt
   => Kapitel 3.1

2. Parametrierung des Parameter-DBs MODBUS_HPARAM_PN entsprechend den Anforderungen (Client/Server, Verbindungsaufbau bei Neustart, Modbusregister, DB-Bereiche etc.)
   => Kapitel 4

3. Für **Modbus Client**: Aufruf und Parametrierung des Modbusbausteins MB_PNHCL in den notwendigen OBs
   => Kapitel 7.1 und 7.2

   oder:

   Für **Modbus Server**: Aufruf und Parametrierung des Modbusbausteins MB_PNHSV in den notwendigen OBs
   => Kapitel 8.1 und 8.2

4. Laden des Anwenderprogramms in die CPU und Lizenzierung des Modbusbausteins für diese CPU
   => Kapitel 5

3 Inbetriebnahme

Allgemeines

Die im Folgenden verwendeten Angaben zu STEP 7 und zur Projektierung der Kommunikationsverbindungen beziehen sich auf die STEP7-Version 5.5 SP2 HF1.

Bei späteren Versionen können Abläufe, Namens- und Verzeichnisangaben geändert sein.

Voraussetzungen

STEP 7–Grundkenntnisse, AWL–Kenntnisse, SPS–Grundkenntnisse

3.1 Installieren der Bibliothek auf dem STEP7-PG/-PC

Lieferumfang

Die beiliegende CD enthält einen Setup, mit dem die Bibliothek „**Modbus_PN_CPU_Red**“, die Beispielprojekte und die Handbücher in Deutsch und Englisch in den entsprechenden STEP7-Verzeichnissen installiert werden.

Zusätzlich befinden sich auf der CD die Handbücher im PDF-Format.

Voraussetzungen

Um die Installation durchführen zu können, muss vorher die Projektiersoftware **STEP7 V5.5** installiert worden sein.

Installation

Legen Sie die Modbus-CD in das CD-ROM-Laufwerk Ihres PGs/PCs ein. Wenn das Setupprogramm nicht automatisch startet, erfolgt die Installation folgendermaßen:

1. Wählen Sie im Windows Explorer das CD-ROM-Laufwerk, öffnen Sie das Verzeichnis Setup und starten Sie das Setuppropramm.
2. Befolgen Sie Schritt für Schritt die Anweisungen, die Ihnen das Installationsprogramm anzeigt.

Sie finden nun

- die Bibliotheken in \Program Files\Siemens\Step7\S7libs,
- die Beispielprojekte in \Program Files\Siemens\Step7\Examples,
- das Handbuch in \Program Files\Siemens\Step7\S7manual\S7Comm.
- das Software Registration Form in \Program Files\Siemens\Step7\S7libs\ Modbus_PN_CPU_Red.

Beim ersten Aufruf der Modbus-Bibliothek verwenden Sie bitte die Funktion „Durchsuchen“ des Öffnen-Dialogs um auf die Bibliothek in S7libs zuzugreifen.

Das Handbuch kann auch über den Shortcut unter \Program Files\Siemens \Dokumentation geöffnet werden.

3.2 CPU – IP-Adresse zuweisen

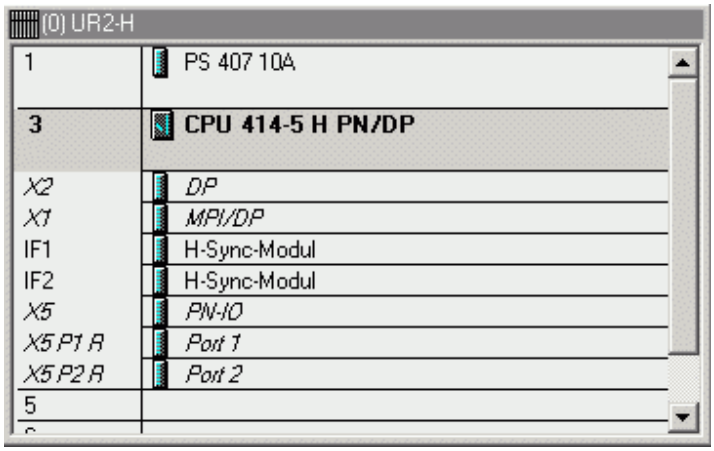
Einleitung

In diesem Beispiel der Vergabe der IP-Adresse wird eine CPU 414-5H PN/DP verwendet.

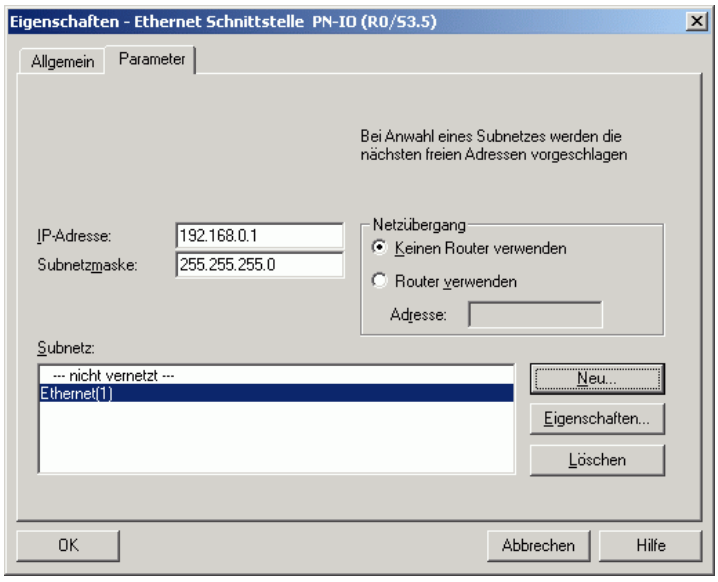
Vorgehensweise

Bevor Sie die Konfiguration vornehmen können, müssen Sie mit STEP7 ein **S7-Projekt** angelegt haben.

1. Öffnen Sie HWKonfig und fügen Sie die Racks und passende Stromversorgungen ein.
   Fügen Sie hinter der Stromversorgung eine CPU 414-5H PN/DP ein.



2. Das Eigenschaftsfenster der PN-IO-Schnittstelle X5 wird angezeigt.



3. Geben Sie die IP-Adresse und die Subnetzmaske ein.
   Wenn Sie eine Verbindung über einen Router aufbauen, müssen Sie zusätzlich noch die Adresse des Routers eingeben.

4. Klicken Sie auf die Schaltfläche "Neu..." und vergeben Sie einen Namen für ein neues Industrial Ethernet Subnetz. Klicken Sie dann auf die Schaltfläche "OK".
   Ergebnis: Sie haben ein neues Industrial Ethernet-Subnetz angelegt.

5. Klicken Sie auf die Schaltfläche "OK".
   Ergebnis: Das Eigenschaftsfenster der PN-IO-Schnittstelle der CPU 414-5H PN/DP wird geschlossen.

6. Fügen Sie in das 2. Rack ebenfalls eine CPU 414-5H PN/DP ein. Diese CPU erhält automatisch die nächste numerisch folgende IP-Adresse.

3.3 Einfügen der Funktionsbausteine in das Programm

Inhalt der Modbus-Bibliothek

Die Modbus-Bibliothek enthält die Ordner „S7 Client“ und „S7 Server“ mit den FBs für die redundante Kommunikation.

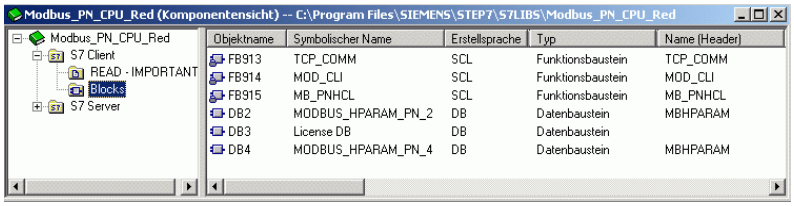
S7 Client

In dem Ordner „S7 Client“ sind u. a. die Bausteine

- FB915 (MB_PNHCL),
- FB914 (MOD_CLI) und
- FB913 (TCP_COMM) enthalten.

Für eine redundante Kommunikation werden immer alle 3 Bausteine benötigt. Der Baustein MB_PNHCL ruft intern mehrfach den Baustein MOD_CLI auf, der wiederum den TCP_COMM aufruft.

Weiterhin befinden sich ein Parameterdatenbaustein MODBUS_HPARAM_PN_2 für einseitige Redundanz, ein Parameterdatenbaustein MODBUS_HPARAM_PN_4 für beidseitige Redundanz sowie der Lizenz-DB als Vorlage in der Bibliothek. Diesen können Sie zur Arbeitserleichterung ebenfalls in Ihr Projekt kopieren.



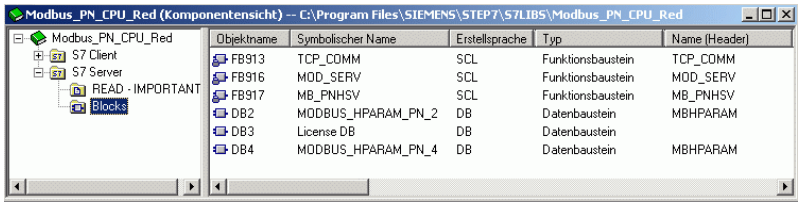
S7 Server

In dem Ordner „S7 Server“ sind u. a. die Bausteine

- FB917 (MB_PNHSV),
- FB916 (MOD_SERV) und
- FB913 (TCP_COMM) enthalten.

Für eine redundante Kommunikation werden immer alle 3 Bausteine benötigt. Der Baustein MB_PNHSV ruft intern mehrfach den Baustein MOD_SERV auf, der wiederum den TCP_COMM aufruft.

Weiterhin befinden sich ein Parameterdatenbaustein MODBUS_HPARAM_PN_2 für einseitige Redundanz, ein Parameterdatenbaustein MODBUS_HPARAM_PN_4 für beidseitige Redundanz sowie der Lizenz-DB als Vorlage in der Bibliothek. Diesen können Sie zur Arbeitserleichterung ebenfalls in Ihr Projekt kopieren.



Bausteine der Standardbibliothek

Folgende FBs werden für die Modbuskommunikation benötigt:

- TSEND (FB63)
- TRCV (FB64)
- TCON (FB65)
- TDISCON (FB66).

Diese Kommunikationsbausteine finden Sie in der Bibliothek „**Standard Library → Communication Blocks**“ und müssen ebenfalls in Ihr Projekt eingefügt werden.

Beachten Sie, dass folgende Versionen der FBs Voraussetzung für den einwandfreien Betrieb der FBs MB_PNHCL bzw. MB_PNHSV sind:

TSEND	V2.1
TRCV	V2.2
TCON	V2.4
TDISCON	V2.1

3.4 Mehrere Verbindungen auf Port 502

Allgemeines

Einige CPUs können TCP-Verbindungen multiplexen. Dabei können sich mehrere MODBUS Clients auf den Port 502 der CPU verbinden (Mulitport). Die CPU fungiert als MODBUS Server.

Hier finden Sie Informationen, welche CPU mit welchem Firmwarestand die mehrfache Nutzung des Ports 502 ermöglicht:
www.siemens.de/s7modbus

Voraussetzungen

Um diese Funktion nutzen zu können, müssen bei der Auswahl der Bausteine und der Parametrierung folgende Einstellungen vorgenommen werden:

- CPU ist Server
- Port 502 als lokaler Port
- unspezifizierte TCP-Verbindung
- passiver Verbindungsaufbau

Freigegebene Anzahl der Verbindungen

Die Anzahl der Verbindungen, die eine CPU auf dem Port 502 annehmen kann, ist geräteabhängig und kann den technischen Daten der CPU entnommen werden.

Projektierung

Für jeden Client, der sich auf den Port 502 des Servers verbinden will, ist je 1 eindeutige Verbindung im Parameter-DB notwendig.

4 Parametrieren der Modbus-Kommunikation

Allgemeines

Für die Kommunikation über die integrierte PN-Schnittstelle der CPU ist keine Verbindungsprojektierung in NetPro notwendig. Die Verbindungen werden mit Hilfe der Funktionsbausteine TCON (FB65) und TDISCON (FB66) auf- bzw. abgebaut.

Es können mehrere Verbindungen zu verschiedenen Kommunikationspartnern projektiert und gleichzeitig aufgebaut werden. Die Anzahl der gleichzeitig aufgebauten Verbindungen ist CPU-abhängig.

Verbindungs-Datenbaustein MODBUS_ HPARAM

Die für den Aufbau der Verbindungen und die Bearbeitung der Modbustelegramme notwendigen Daten werden in einem Datenbaustein – dem Parameterdatenbaustein MODBUS_HPARAM_PN – definiert. Dabei werden zuerst die verbindungsspezifischen Daten abgelegt. Nach den verbindungsspezifischen Daten folgen die Modbusparameter.

Für jede Verbindung 0A, 1A, 0B und 1B wird **je 1 Verbindungsblock** benötigt, in dem die Verbindungsparameter zwischen den Kommunikationspartnern definiert werden. Bei einseitiger Redundanz werden 2 Verbindungsblöcke angelegt, bei beidseitiger Redundanz werden 4 Verbindungsblöcke angelegt. Im Anschluss an die Verbindungsblöcke werden die Modbusparameter angegeben.

Je ein vorgefertigter Parameter-Datenbaustein für einseitige und beidseitige Redundanz ist als Beispiel in der Bibliothek „**Modbus_PN_CPU_Red**“ enthalten.

Aufbau des DB MODBUS_HPARAM_PN bei einseitiger Verbindung:

Adresse	Name
0.0	FALSE: Einseitige Verbindung
2.0	STRUCT
	Verbindung 0A: Verbindungsparameter
	END_STRUCT
66.0	STRUCT
	Verbindung 1A: Verbindungsparameter
	END_STRUCT
130.0	Modbusparameter

Aufbau des DB MODBUS_HPARAM_PN bei beidseitiger Verbindung:

Adresse	Name
0.0	TRUE: Beidseitige Verbindung
2.0	STRUCT
	Verbindung 0A: Verbindungsparameter
	END_STRUCT
66.0	STRUCT
	Verbindung 1A: Verbindungsparameter
	END_STRUCT
130.0	STRUCT
	Verbindung 0B: Verbindungsparameter
	END_STRUCT
194.0	STRUCT
	Verbindung 1B: Verbindungsparameter
	END_STRUCT
258.0	Modbusparameter

Verbindungs-parameter

In den Verbindungsblöcken werden die verbindungsspezifischen Parameter, wie z.B. die lokal verwendete Hardwareschnittstelle und die IP-Adresse des Kommunikationspartners, definiert. Mit Hilfe dieser Parameter können die Funktionen TCON und TDISCON eine Verbindung auf- bzw. abbauen. Den genauen Aufbau finden Sie in Kapitel 4.2.

Die Datenstruktur des Verbindungsparameterblocks muss zwingend eingehalten werden, da sonst keine Verbindung aufgebaut werden kann.

Modbusparameter

In den Modbusparametern werden die für die Betriebsart und Adressreferenz notwendigen Daten abgelegt, wie z.B. die Modbusbereiche, die in den Datenbausteinen abgebildet werden und die Betriebsart der S7 als Modbus Server oder Modbus Client. Die Datenstruktur der Modbusparameter muss eingehalten werden, da sonst keine fehlerfreie Bearbeitung möglich ist.

Projektierungs-möglichkeiten

Es gibt zwei Möglichkeiten, die Projektierung für die Verbindungs- und Modbusparameter vorzunehmen. Zum einen ist die Eingabe über einen Wizard möglich, mit dem sehr komfortabel die Parametrierung vorgenommen werden kann. Zum anderen können die Parameter durch Editieren der Struktur im Parameterdatenbaustein eingestellt werden.

Diese beiden Möglichkeiten werden in den folgenden Kapiteln 4.1 und 4.2 beschrieben.

In dem Parameter-Datenbaustein dürfen keine weiteren Parameter abgelegt werden.

4.1 Parametrieren mit dem Wizard

Allgemeines

Mit dem „**Modbus/TCP PN Red Wizard**“ ist eine komfortable Projektierung der Verbindungen und der Modbusparameter im Parameterdatenbaustein MODBUS_HPARAM_PN möglich. Dabei wird der komplette Datenbaustein (Verbindungsparameter und Modbusparameter) angelegt.

Es wird empfohlen für die Parametrierung des MODBUS_HPARAM_PN den Wizard einzusetzen.

Den Wizard finden Sie unter
http://support.automation.siemens.com/WW/view/de/2077896767.

4.2 Manuelle Parametrierung

Vorgehensweise

Kopieren Sie den DB2 für einseitige Redundanz bzw. den DB5 für beidseitige Redundanz für Client oder Server aus der Bibliothek „**Modbus_PN_CPU_Red**“ und fügen Sie diesen in Ihr Projekt ein. Wird die Nummer bereits anderweitig verwendet, kann der DB umbenannt werden. In diesem Beispiel wird der DB2 für einseitige Redundanz verwendet.

Die Parameter im Baustein MODBUS_HPARAM_PN dürfen während der Laufzeit nicht verändert werden. Nach einer Änderung der Parameter muss die CPU mit STOP -> RUN neu gestartet werden.

Aufbau und Anpassungen der Verbindungs-parameter

Pro Verbindung wird ein Block benötigt.

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	double_sided_red	BOOL	FALSE
+2.0	connection_0A	STRUCT	
+0.0	block_length	WORD	W#16#40
+2.0	id	WORD	W#16#1
+4.0	connection_type	BYTE	B#16#11
+5.0	active_est	BOOL	FALSE
+6.0	local_device_id	BYTE	B#16#5
+7.0	local_tsap_id_len	BYTE	B#16#2
+8.0	rem_subnet_id_len	BYTE	B#16#0
+9.0	rem_staddr_len	BYTE	B#16#0
+10.0	rem_tsap_id_len	BYTE	B#16#0
+11.0	next_staddr_len	BYTE	B#16#0
+12.0	local_tsap_id	ARRAY[1..16]	B#16#1, B#16#F6, B#16#0, B#16#0,
*1.0		BYTE	
+28.0	rem_subnet_id	ARRAY[1..6]	B#16#0, B#16#0, B#16#0, B#16#0,
*1.0		BYTE	
+34.0	rem_staddr	ARRAY[1..6]	B#16#0, B#16#0, B#16#0, B#16#0,
*1.0		BYTE	
+40.0	rem_tsap_id	ARRAY[1..16]	B#16#0, B#16#0, B#16#0, B#16#0,
*1.0		BYTE	
+56.0	next_staddr	ARRAY[1..6]	B#16#0, B#16#0, B#16#0, B#16#0,
*1.0		BYTE	
+62.0	spare	WORD	W#16#0
=64.0		END_STRUCT	

block_length

Dieser Parameter bezeichnet die Länge der Verbindungsparameter und darf nicht verändert werden.

Fester Wert: W#16#40

id

Für jede logische Verbindung wird eine Verbindungs-ID vergeben. Diese muss bei Verwendung der T-Kommunikation auf der gesamten CPU eindeutig sein. Die ID wird beim Aufruf des FB MB_PNHCL bzw. MB_PNHSV angegeben und bei den internen Aufrufen der T-Bausteine (TCON, TSEND, TRCV und TDISCON) verwendet.

Wertebereich: W#16#1 bis W#16#FFF

connection_type Hier wird der Verbindungstyp für den Aufbau der Verbindung durch die Funktion TCON definiert. Der einzustellende Wert ist CPU-abhängig.

TCP (Kompatibilitätsmode):	B#16#01 für CPU 315 bzw. 317 <= FW V2.3
TCP:	B#16#11 für CPU 315 bzw. 317 >= FW V2.4, IM 151-8 PN/DP CPU, CPU314C, CPU319, CPU412(H), CPU414(H), CPU416(H), CPU417(H)

Je nach verwendeter Firmware können diese Angaben variieren.
Weitere Angaben finden Sie unter:
http://support.automation.siemens.com/WW/view/de/24294554

active_est Dieser Parameter bezeichnet die Art des Verbindungsaufbaus, aktiv oder passiv. Der Modbus Client übernimmt den aktiven und der Modbus Server den passiven Verbindungsaufbau.

Aktiver Verbindungsaufbau:	TRUE
Passiver Verbindungsaufbau:	FALSE

local_device_id Die *local_device_id* definiert die IE-Schnittstelle der verwendeten PN-CPU. Je nach PN-CPU-Typ werden unterschiedliche Einstellungen benötigt.

IM 151-8 PN/DP CPU:	B#16#1
CPU 314C, 315 bzw. 317:	B#16#2
CPU 319:	B#16#3
CPU 412(H), 414(H), 416(H) sowie 417(H)	B#16#5
CPU im Rack1 der H-Station	B#16#15

In H-Stationen gilt:
Die S7400-PN-CPU in Rack 0 kommuniziert über local_device_id = 5, die CPU in Rack 1 kommuniziert über local_device_id = 15_{Hex}.

local_tsap_id_len Es wird die Länge des Parameters *local_tsap_id* (= lokale Portnummer) angegeben.

Aktiver Verbindungsaufbau:	0
Passiver Verbindungsaufbau:	2

rem_subnet_id_len Dieser Parameter wird derzeit nicht verwendet und muss mit B#16#0 belegt sein.

rem_staddr_len Es wird die Länge des Parameters *rem-staddr*, also der IP-Adresse des Kommunikationspartners, angegeben. Soll über eine unspezifizierte Verbindung kommuniziert werden, wird keine IP-Adresse für den Partner angegeben.

Unspezifizierte Verbindung:	B#16#0
Spezifizierte Verbindung:	B#16#4

rem_tsap_id_len Dieser Parameter bezeichnet die Länge des Parameters *rem_tsap_id*, der Portnummer des remoten Kommunikationspartners.

Aktiver Verbindungsaufbau:	2
Passiver Verbindungsaufbau:	0

next_staddr_len Hier wird die Länge des Parameters *next_staddr* festgelegt.
Bei PN-Schnittstelle: B#16#0

local_tsap_id Mit diesem Parameter wird die lokale Portnummer eingestellt. Die Art der Darstellung wird dabei abhängig des Parameters *connection_type* unterschieden. Der Wertebereich ist CPU-abhängig. Die Portnummer muss auf der CPU eindeutig sein.

Bei *connection_type* B#16#01

local_tsap_id[1]	low byte der Port-Nr. in Hex-Darstellung
local_tsap_id[2]	high byte der Port-Nr. in Hex-Darstellung
local_tsap_id[3-16]	B#16#00

Bei *connection_type* B#16#11

local_tsap_id[1]	high byte der Port-Nr. in Hex-Darstellung
local_tsap_id[2]	low byte der Port-Nr. in Hex-Darstellung
local_tsap_id[3-16]	B#16#00

rem_subnet_id Dieser Parameter wird derzeit nicht verwendet und muss mit 0 belegt werden.

rem_staddr In diesem Byte-Array wird die IP-Adresse des remoten Kommunikationspartners eingetragen. Im Falle einer unspezifizierten Verbindung wird keine IP-Adresse eingetragen. Die Art der Darstellung wird abhängig vom Parameter *connection_type* unterschieden.

Beispiel: IP-Adresse 192.168.0.1:

Bei *connection_type* B#16#01

rem_staddr[1] =	B#16#01 (1),
rem_staddr[2] =	B#16#00 (0),
rem_staddr[3] =	B#16#A8 (168),
rem_staddr[4] =	B#16#C0 (192),
rem_staddr[5-6]=	B#16#00 (reserviert)

Bei *connection_type* B#16#11

rem_staddr[1] =	B#16#C0 (192),
rem_staddr[2] =	B#16#A8 (168),
rem_staddr[3] =	B#16#00 (0),
rem_staddr[4] =	B#16#01 (1),
rem_staddr[5-6]=	B#16#00 (reserviert)

rem_tsap_id Mit diesem Parameter wird die remote Portnummer eingestellt. Die Art der Darstellung wird dabei abhängig vom Parameter *connection_type* unterschieden. Der Wertebereich ist CPU-abhängig.

Bei *connection_type* B#16#01

local_tsap_id[1]	low byte der Port-Nr. in Hex-Darstellung
local_tsap_id[2]	high byte der Port-Nr. in Hex-Darstellung
local_tsap_id[3-16]	B#16#00

Bei *connection_type* B#16#11

local_tsap_id[1]	high byte der Port-Nr. in Hex-Darstellung
local_tsap_id[2]	low byte der Port-Nr. in Hex-Darstellung
local_tsap_id[3-16]	B#16#00

next_staddr Dieser Parameter bezeichnet die Rack- und Steckplatznummer des verwendeten CPs. Bei Verwendung der integrierten PN-Schnittstelle der CPU muss dieser Parameter auf 0 gesetzt werden.

next_staddr[1-6]	B#16#00

spare Dieser Parameter wird nicht verwendet und muss mit 0 vorbelegt werden.

Anpassungen der Modbusparameter

Mit den Modbusparametern im Baustein MODBUS_HPARAM_PN werden die Betriebsart der Modbuskommunikation und die Adressabbildung von Modbusadressen auf SIMATIC-Adressen festgelegt.

+130.0	server_client	BOOL	TRUE
+130.1	single_write	BOOL	FALSE
+130.2	conn_at_startup	BOOL	FALSE
+131.0	reserved	BYTE	B#16#0
+132.0	data_type_1	BYTE	B#16#3
+134.0	db_1	WORD	W#16#B
+136.0	start_1	WORD	W#16#0
+138.0	end_1	WORD	W#16#1F3
+140.0	data_type_2	BYTE	B#16#3
+142.0	db_2	WORD	W#16#C
+144.0	start_2	WORD	W#16#2D0
+146.0	end_2	WORD	W#16#384
+148.0	data_type_3	BYTE	B#16#4
+150.0	db_3	WORD	W#16#D
+152.0	start_3	WORD	W#16#2D0
+154.0	end_3	WORD	W#16#3E8
+156.0	data_type_4	BYTE	B#16#0
+158.0	db_4	WORD	W#16#0
+160.0	start_4	WORD	W#16#0
+162.0	end_4	WORD	W#16#0
+164.0	data_type_5	BYTE	B#16#1
+166.0	db_5	WORD	W#16#E
+168.0	start_5	WORD	W#16#280
+170.0	end_5	WORD	W#16#4E2
+172.0	data_type_6	BYTE	B#16#2
+174.0	db_6	WORD	W#16#F
+176.0	start_6	WORD	W#16#6A4
+178.0	end_6	WORD	W#16#8FC
+180.0	data_type_7	BYTE	B#16#1
+182.0	db_7	WORD	W#16#10
+184.0	start_7	WORD	W#16#6A4
+186.0	end_7	WORD	W#16#8FC
+188.0	data_type_8	BYTE	B#16#0
+190.0	db_8	WORD	W#16#0
+192.0	start_8	WORD	W#16#0
+194.0	end_8	WORD	W#16#0
+196.0	conn_0A_send_buffer	ARRAY[1..260]	B#16#0
*1.0		BYTE	
+456.0	conn_0A_recv_buffer	ARRAY[1..260]	B#16#0
*1.0		BYTE	
+716.0	conn_1A_send_buffer	ARRAY[1..260]	B#16#0
*1.0		BYTE	
+976.0	conn_1A_recv_buffer	ARRAY[1..260]	B#16#0
*1.0		BYTE	
=1236.0		END_STRUCT	

server_client

TRUE: S7 ist Server, bei Verwendung von MB_PNHSV einzustellen
FALSE: S7 ist Client, bei Verwendung von MB_PNHCL einzustellen

single_write

Im Baustein MB_PNHCL wird mit dem Parameter single_write = TRUE bei schreibenden Aufträgen mit Länge 1 die Funktionscodes 5 und 6 verwendet. Ist single_write = FALSE, werden bei allen schreibenden Aufträgen die Funktionscodes 15 und 16 verwendet.

connect_at_ startup

Hiermit wird der Zeitpunkt des Verbindungsaufbaus festgelegt.
Ist *connect_at_startup* auf TRUE gesetzt, wird der Verbindungsaufbau – unabhängig von ENR - direkt nach dem CPU-Neustart ausgeführt. In diesem Fall darf erst ein Datenauftrag abgesetzt werden, wenn die Verbindungen korrekt aufgebaut werden konnten (ESTAB_x = TRUE) oder ein entsprechender Fehler an ERROR und STATUS_x angezeigt wird. Spätestens nach Ablauf von CONN_TIMEOUT werden die Status-Ausgänge aktualisiert.

FALSE: Verbindungsaufbau bei gesetztem ENQ bzw. ENR
TRUE: Verbindungsaufbau direkt nach Neustart

8 Datenbereiche

Es werden 8 Datenbereiche für die Abbildung der MODBUS Adressen im S7-Speicher angeboten. Es muss mindestens der erste Datenbereich definiert werden, die anderen 7 Datenbereiche sind optional. Aus den Datenbereichen wird abhängig vom Auftragstyp gelesen oder in sie geschrieben.

Mit einem Auftrag kann immer nur aus einem DB gelesen / in einen DB geschrieben werden. Zugriffe auf Register oder Bitwerte, die in mehreren DBs liegen, auch wenn die Nummern lückenlos hintereinander liegen, sind auf zwei Aufträge aufzuteilen. Dies ist bei der Projektierung zu berücksichtigen.
Es ist möglich in einen Datenbaustein mehr Modbusbereiche (Register oder Bitwerte) abzubilden als mit einem Telegramm bearbeitet werden können.

data_type_x

Mit dem Parameter *data_type_x* wird angegeben, welche MODBUS Datentypen in diesem Datenbaustein abgebildet werden.
Wird in *data_type_x* der Wert 0 eingetragen, wird der entsprechende Datenbereich nicht verwendet.

Kennung	Datentyp	Datenbreite
0	Bereich nicht verwendet	
1	Coils	Bit
2	Inputs	Bit
3	Holding Register	Word
4	Input Register	Word

db_x

Der Parameter *db_x* legt den Datenbaustein fest, in dem die nachfolgend definierten MODBUS Register oder Bitwerte abgebildet werden. Die DB-Nummer 0 ist nicht erlaubt, da diese für das System reserviert ist.

db_x
DB-Nummer 1 bis 65535 (W#16#0001 bis W#16#FFFF)

Der Datenbaustein muss 2 Byte länger angelegt werden als für die parametrierten Daten notwendig ist. Die beiden letzten Bytes werden für interne Zwecke verwendet.

start_x
end_x

Mit *start_x* wird die erste Modbusadresse, die im Datenwort 0 des DB abgebildet wird, angegeben. Der Parameter *end_x* definiert die Adresse der letzten MODBUS-Adresse.

Bei Registerzugriffen berechnet sich die Datenwortnummer im S7 DB, in die die letzte Modbusadresse eingetragen wird, nach folgender Formel:

DBW Nummer = (*end_x – start_x*) ∗ 2

Bei Bitzugriffen berechnet sich die Datenbytenummer im S7 DB, in die die letzte Modbusadresse eingetragen wird, nach folgender Formel:

DBB Nummer = (*end_x – start_x* + 7) / 8

Die definierten Datenbereiche dürfen sich nicht überlappen. Der Parameter *end_x* darf nicht kleiner als *start_x* sein. Im Fehlerfall wird der Anlauf des FBs mit Fehler beendet. Sind beide Werte gleich, wird 1 Modbusadresse (1 Register oder 1 Bitwert) zugeordnet.

In Kapitel 7.3 und 8.3 ist ein Beispiel für die Abbildung der MODBUS Adressen auf S7-Speicherbereiche dargestellt.

start_x, end_x

MODBUS Adresse	0 bis 65535 (W#16#0000 bis W#16#FFFF)

conn_0A_send_buffer
Dieses Array wird intern im FB für die Sendedaten von Verbindung 0A verwendet. Zugriffe oder Änderungen in diesem Bereich sind nicht zulässig.

conn_0A_recv_buffer
Dieses Array wird intern im FB für die Empfangsdaten von Verbindung 0A verwendet. Zugriffe oder Änderungen in diesem Bereich sind nicht zulässig.

conn_1A_send_buffer
Dieses Array wird intern im FB für die Sendedaten von Verbindung 1A verwendet. Zugriffe oder Änderungen in diesem Bereich sind nicht zulässig.

conn_1A_recv_buffer
Dieses Array wird intern im FB für die Empfangsdaten von Verbindung 1A verwendet. Zugriffe oder Änderungen in diesem Bereich sind nicht zulässig.

conn_0B_send_buffer
Dieses Array wird intern im FB für die Sendedaten von Verbindung 0B verwendet. Zugriffe oder Änderungen in diesem Bereich sind nicht zulässig.

conn_0B_recv_buffer
Dieses Array wird intern im FB für die Empfangsdaten von Verbindung 0B verwendet. Zugriffe oder Änderungen in diesem Bereich sind nicht zulässig.

conn_1B_send_buffer
Dieses Array wird intern im FB für die Sendedaten von Verbindung 1B verwendet. Zugriffe oder Änderungen in diesem Bereich sind nicht zulässig.

conn_1B_recv_buffer
Dieses Array wird intern im FB für die Empfangsdaten von Verbindung 1B verwendet. Zugriffe oder Änderungen in diesem Bereich sind nicht zulässig.

5 Lizenzierung

Allgemeines

Die Bausteine MB_PNHCL bzw. MB_PNHSV müssen auf jeder CPU lizenziert werden.
Die Lizenzierung erfolgt in 2 Schritten: dem Auslesen des IDENT_CODEs und der Eingabe des Freischaltcodes REG_KEY. Dabei muss der OB121 in der CPU vorhanden sein.

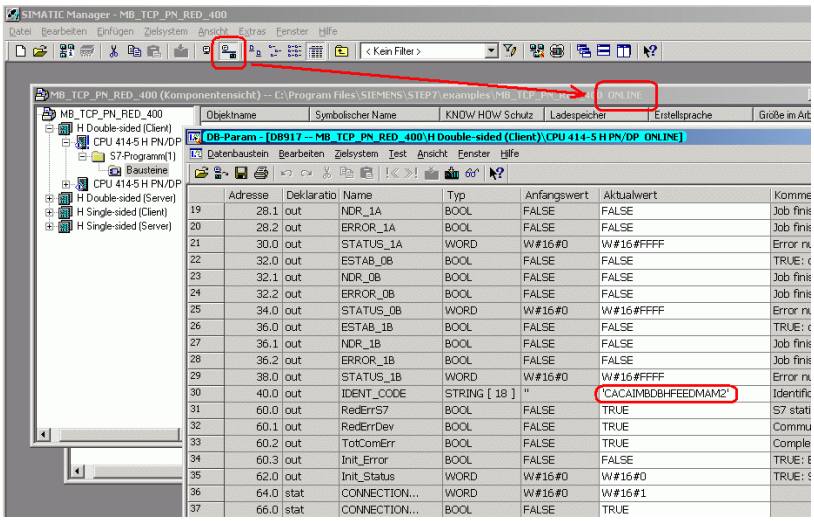
Bitte beachten Sie:
Bei einer S7-H-Station wird nur die CPU in Rack 0 lizenziert. Daher ist ein Tausch der CPU aus Rack 0 nach der Lizenzierung nicht mehr möglich.

Auslesen des IDENT_CODES

Für das Auslesen des IDENT_CODE gehen Sie wie folgt vor:

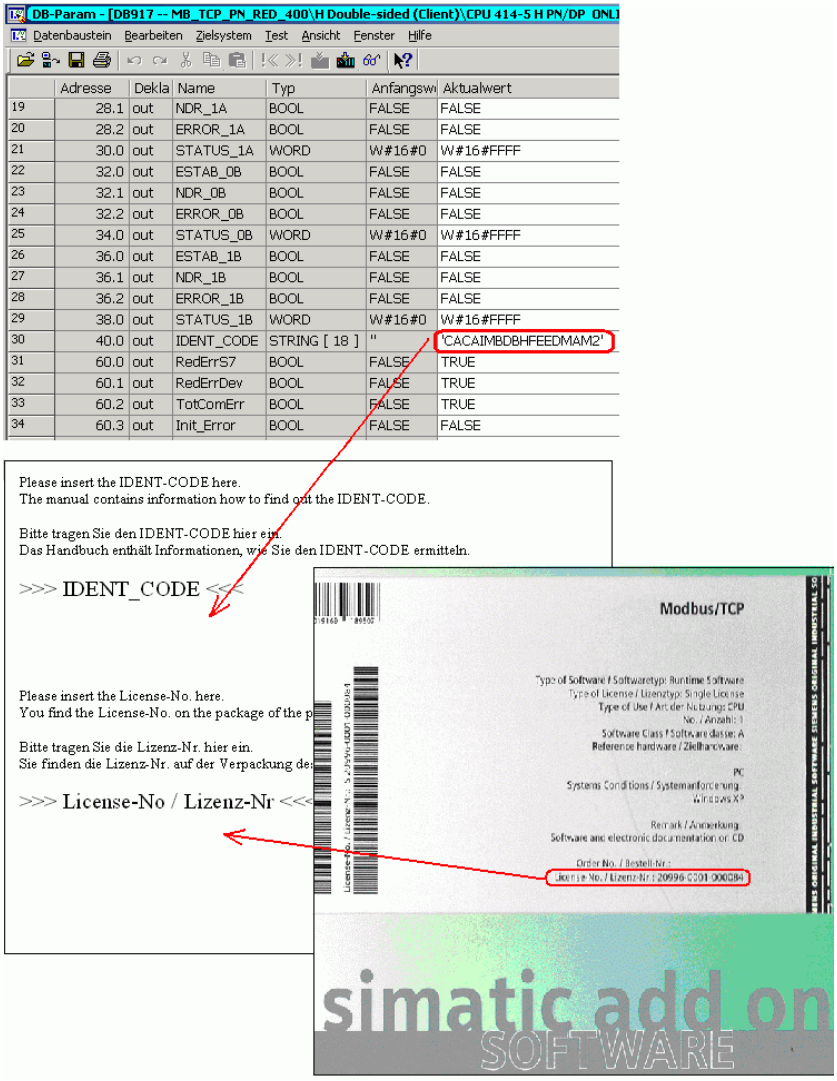
1. Parametrieren Sie den Baustein MB_PNHCL bzw. MB_PNHSV Ihren Anforderungen entsprechend in einem zyklischen OB (OB1 oder Weckalarm-OB) und im OB100.
   Laden Sie das Programm in die CPU und setzen Sie diese in RUN.

2. Öffnen Sie im SIMATIC Manager das Projekt im Online-Modbus. Öffnen Sie in diesem Online-Projekt den Instanz-DB des Modbusbausteins.



3. Am Ausgang IDENT_CODE wird eine 18stellige Zeichenfolge angezeigt.

   Kopieren Sie diesen String per Copy/Paste aus dem DB und fügen ihn in das Formular ***SOFTWARE REGISTRATION FORM*** ein. Dieses Formular wird bei der Installation im Bibliotheks-Pfad ..\Program Files\Siemens\Step7\S7LIBS\ Modbus_PN_CPU_Red abgelegt und liegt zusätzlich auch auf der Installations-CD.
   Tragen Sie die Lizenz-Nr. von der Produktverpackung in das Formular ein.



4. Senden Sie das Formular als Service Request (http://support.automation.siemens.com/WW/view/de/38718979) an den Customer Support. Sie erhalten daraufhin den Freischaltcode für ihre CPU.

5. Hinweis für die Verwendung in CFC: Der CFC-Editor kann online nur eine bestimmte Anzahl von Zeichen anzeigen. Der vollständige IDENT_CODE wird im Tooltipp des Ausgangsparameters bzw. im IDB anzeigt.

Eingabe des Freischaltcodes REG_KEYs

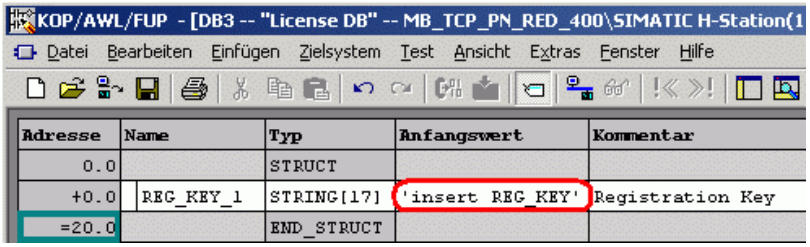
Die Angabe des Freischaltcodes REG_KEY muss an jedem Modbus-Bausteinaufruf erfolgen.

Der REG_KEY sollte in einem Global-DB gespeichert werden, über den alle Modbus-Bausteine den notwendigen Freischaltcode erhalten (siehe nachfolgendes Beispiel).

Für die Eingabe des Freischaltcodes REG_KEY gehen Sie wie folgt vor:

1. Kopieren Sie den vorgefertigten Lizenzierungsbaustein DB3 aus der Bibliothek „Modbus_PN_CPU_Red“ in ihr Projekt. Wird die DB-Nummer bereits im Projekt verwendet, kann der Lizenz-DB auch umbenannt werden.

2. Öffnen Sie den Lizenz-DB und kopieren Sie den übermittelten 17stelligen Freischaltcode per Copy/Paste in die Spalte ‚Anfangswert’. Die Eingabe von mehreren Keys als Liste ist möglich.



3. Damit der Freischaltcode nach dem Neuladen der CPU nicht erneut eingegeben werden muss, muss er im Datenbaustein fest eingetragen werden. Wechseln Sie dazu über den Menüpunkt „Ansicht“ -> „Datensicht“ in die Datensicht des DBs. Über den Menübefehl „Bearbeiten“ -> „Datenbaustein initialisieren" werden dann alle Werte der Spalte „Anfangswert" in die Spalte „Aktualwert" übernommen.

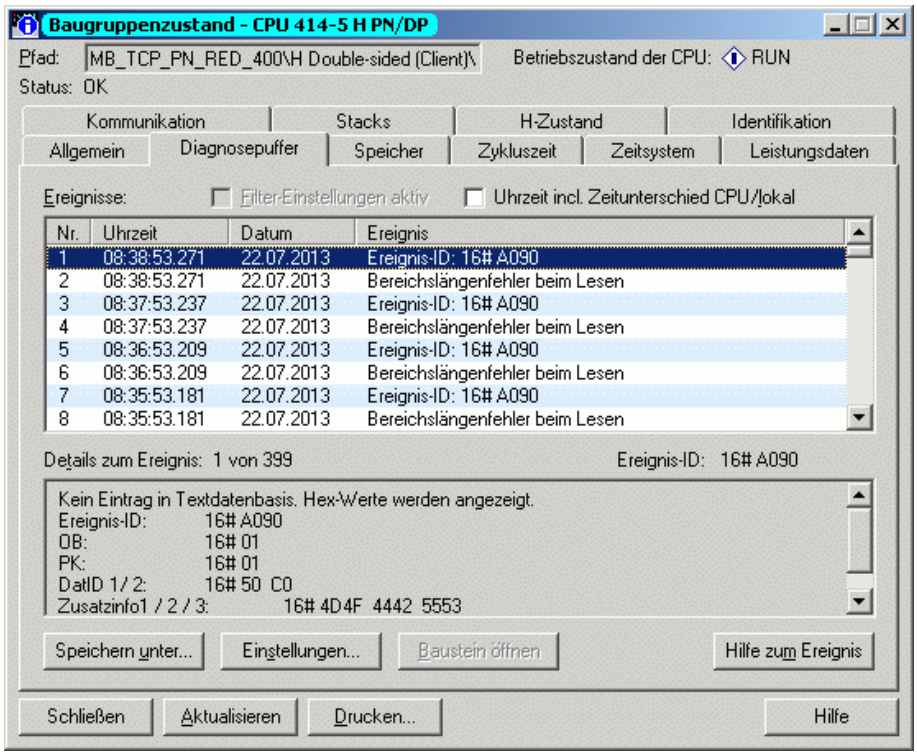
4. Geben Sie im zyklischen OB am Parameter REG_KEY des Modbus-Bausteins die Datenbausteinnummer des Lizenz-DBs an.

5. Laden Sie die geänderten Bausteine in die CPU. Die Eingabe des Freischaltcodes kann zur Laufzeit erfolgen, ein Wechsel von STOP -> RUN ist nicht erforderlich.

Der Baustein ist nun für diese CPU lizenziert.

Fehlende oder fehlerhafte Lizenzierung

Ist kein oder ein falscher Freischaltcode eingetragen, blinkt die INTF-LED der H-CPU 1 Mal pro Minute und es wird zyklisch ein Eintrag in den Diagnosepuffer bezüglich der fehlenden Lizenz vorgenommen. Die Fehlernummer für eine fehlende Lizenz ist W#16#A090.
Bei Verwendung einer Single-PN-CPU blinkt die LED alle 4 Sekunden und es wird ein entsprechender Eintrag in den Diagnosepuffer vorgenommen..



Warnung

Falls der OB121 in der Steuerung fehlt, wird die CPU in den STOP-Zustand gesetzt.

Bei einem fehlenden oder falschen Freischaltcode wird die Modbus-Kommunikation bearbeitet, allerdings wird an den Ausgängen STATUS_x stets W#16#A090 „Keine gültige Lizenz vorhanden“ angezeigt.

6 Redundanz

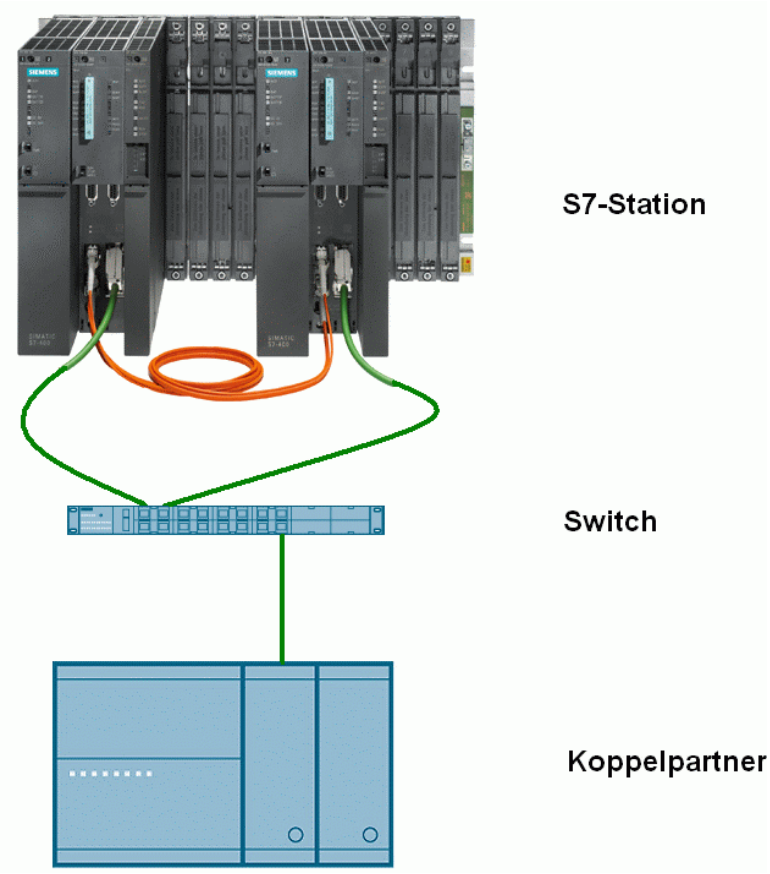
6.1 Projektierung der redundanten Kommunikation

Allgemeines

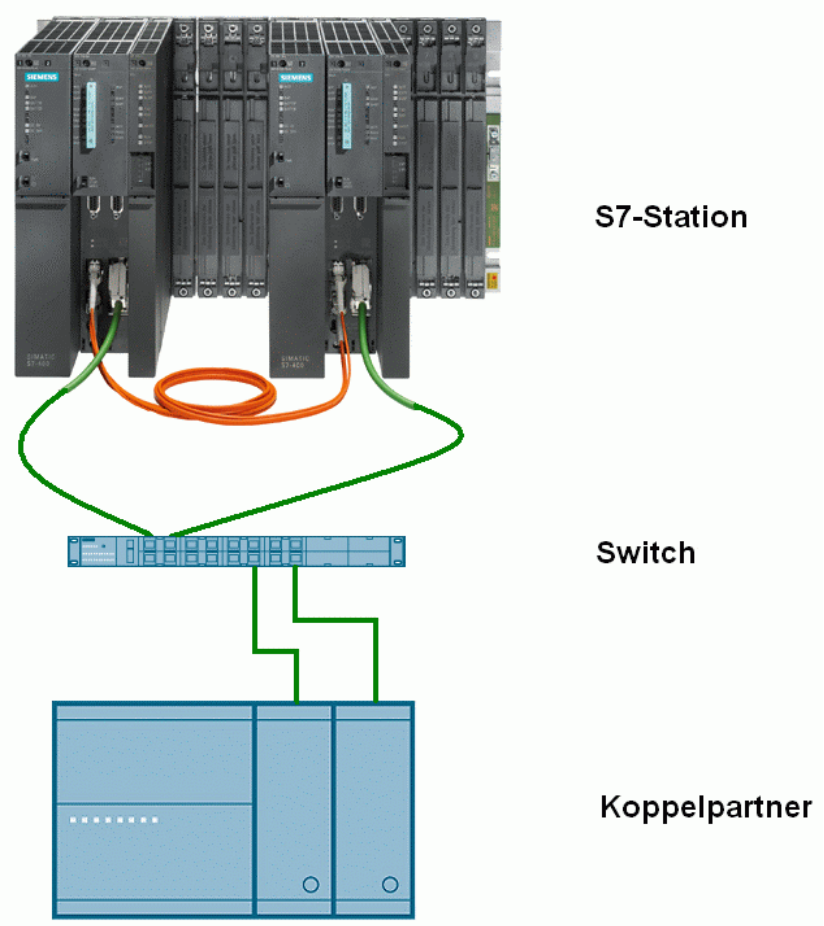
Die folgenden Seiten geben einen Überblick über die verschiedenen Möglichkeiten des Redundanz-Aufbaus.

Die Kommunikationsteilnehmer können standalone oder redundant aufgebaut werden. Ist einer der beiden Teilnehmer standalone aufgebaut, spricht man von einseitiger Redundanz. Sind beide Partner redundant aufgebaut, bezeichnet man es als beidseitige Redundanz.

Einseitige Redundanz:



Beidseitige Redundanz:



Portnummer für Client und Server

Der Modbus Client verwendet eine Portnummer ab 2000.

Der Modbus Server wird normalerweise über die Portnummer 502 angesprochen. Abhängig von der CPU ist es möglich, den Port 502 für mehrere Verbindungen zu projektieren (Mulitport). Wurde für 2 oder mehr Verbindungen der lokale Port 502 projektiert, werden die anfragenden Clients zufällig auf die vorhandenen Server-Verbindungen verteilt. Der 1. Client, der sich auf den MB_PNHSV-Baustein verbinden will, erhält nicht automatisch die Verbindung 0A zugeordnet. Ist die Zuordnung der Client-Anfragen zu den Server-Verbindungen erfolgt, bleibt die Zuordnung während des Telegrammverkehrs so lange bestehen bis die Verbindung abgebaut wird.

6.2 Einseitige Redundanz

Allgemein

Für jede Verbindung zwischen den Kommunikationspartnern muss je 1 Verbindung im Parameterdatenbaustein projektiert werden.

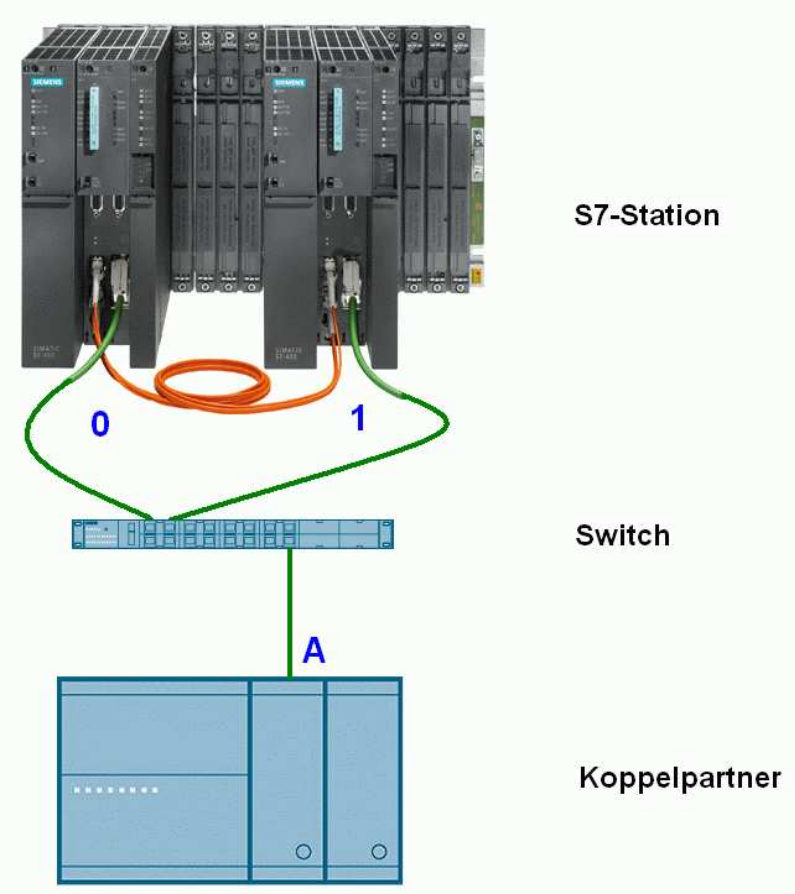
Die Verbindungspunkte der **S7** haben die Bezeichnungen **0 und 1**, die Verbindungspunkte des **Kommunikationspartners** haben die Bezeichnungen **A und B**.

Projektierung

Ist die S7 redundant aufgebaut, wird 1 Verbindung für CPU0 zum Knotenpunkt A des Koppelpartners und 1 Verbindung für CPU1 zum Knotenpunkt B des Koppelpartners angelegt.

- Verbindung von CPU0 zum Partner/Knoten A => **Verbindung 0A**
- Verbindung von CPU1 zum Partner/Knoten A => **Verbindung 1A**

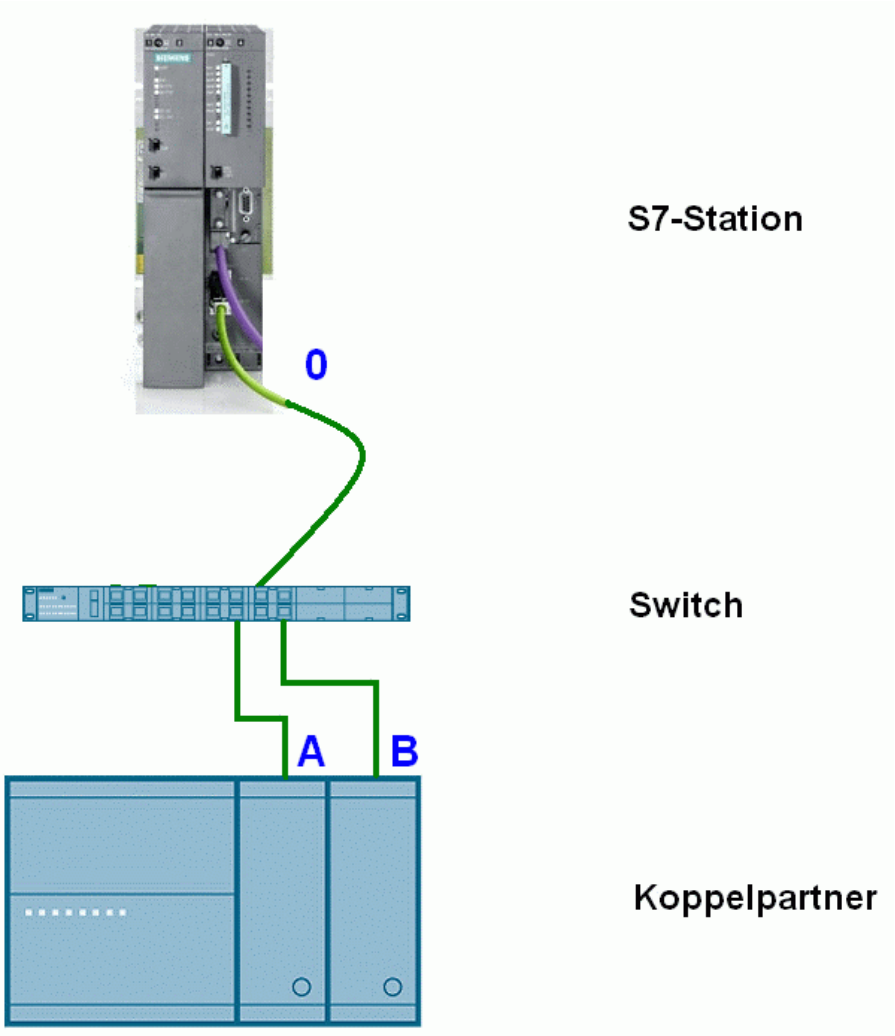
Das Bild veranschaulicht die Verbindungsbezeichnungen graphisch.



Ist die S7 standalone und der Koppelpartner redundant aufgebaut, wird eine Verbindung von CPU0 zum Knotenpunkt A des Koppelpartners und eine Verbindung von CPU0 zum Knotenpunkt B des Koppelpartners angelegt.

- Verbindung von CPU0 zum Partner/Knoten A => **Verbindung 0A**
- Verbindung von CPU0 zum Partner/Knoten B => **Verbindung 0B**

Das Bild veranschaulicht die Verbindungsbezeichnungen graphisch.



6.3 Beidseitige Redundanz

Allgemein

Für jede Verbindung zwischen den Kommunikationspartnern muss je 1 Verbindung im Parameterdatenbaustein projektiert werden.

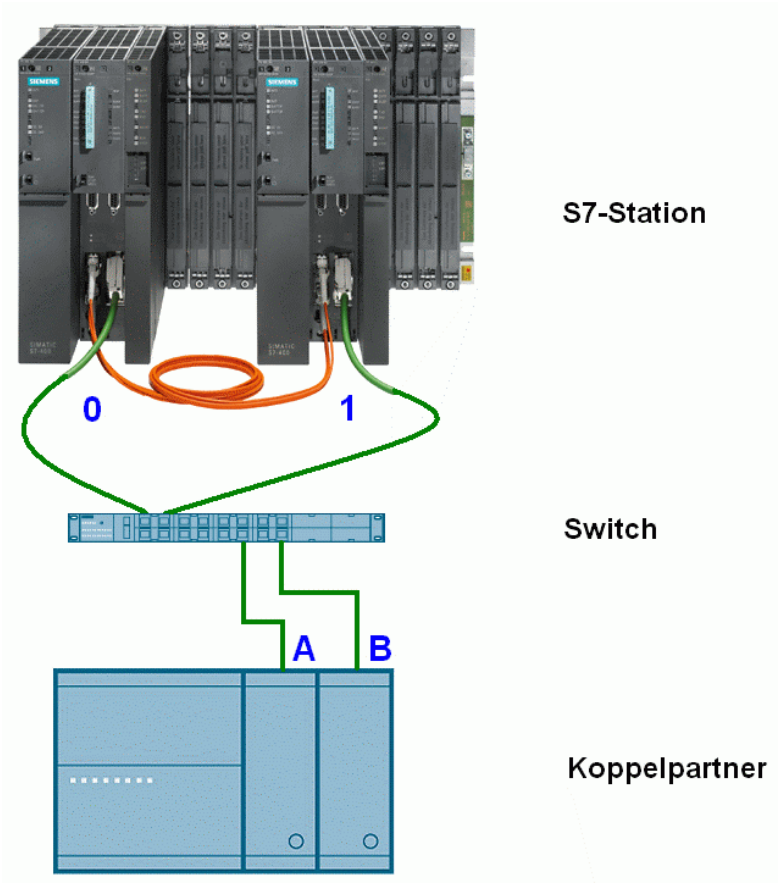
Die Verbindungspunkte der **S7** haben die Bezeichnungen **0 und 1**, die Verbindungspunkte des **Kommunikationspartners** haben die Bezeichnungen **A und B**.

Projektierung

Bei beidseitiger Redundanz werden 2 Verbindungen für CPU0 zum Koppelpartner und 2 Verbindungen für CPU1 zum Koppelpartner angelegt:

- Verbindung von CPU0 zum Partner/Knoten A => **Verbindung 0A**
- Verbindung von CPU1 zum Partner/Knoten A => **Verbindung 1A**
- Verbindung von CPU0 zum Partner/Knoten B => **Verbindung 0B**
- Verbindung von CPU1 zum Partner/Knoten B => **Verbindung 1B**

Das Bild veranschaulicht die Verbindungsbezeichnungen graphisch.



7 Funktionsbaustein MB_PNHCL – Modbus Client

7.1 Funktionsweise des FB MB_PNHCL

Allgemeines

Die CPU ist Client wenn die S7 das Lesen bzw. Schreiben der Daten aus bzw. in den remoten Partner initiiert.

Für die Clientfunktionalität werden die FBs MB_PNHCL, MOD_CLI und TCP_COMM benötigt.

Es können mehrere Instanzen des Bausteins MB_PNHCL im Programm aufgerufen werden. Es gibt seitens der Bibliothek keine Begrenzung der maximalen Anzahl von parallel laufenden Modbusbausteinen. Allerdings gibt es eine CPU-abhängige maximale Anzahl von gleichzeitig aufgebauten Verbindungen. Im Handbuch der CPUs befindet sich die Angabe, wie viele Verbindungen parallel aufgebaut werden können.
Bei mehreren Instanzen von MB_PNHCL ist zu beachten, dass jede Bausteininstanz einen eigenen Parameter-Datenbaustein erhält und die **Verbindungs-IDs CPU-weit eindeutig** sind.

Aufgaben des FB

Der Funktionsbaustein MB_PNHCL erfüllt folgende Aufgaben:

- Koordinierung über welche Verbindung(en) die Telegramme gesendet werden.
- Transaction Identifier TI führen
- Lizenzprüfung

Der Baustein MB_PNHCL ruft intern mehrfach den Baustein MOD_CLI auf.

Der Baustein MOD_CLI erfüllt folgende Aufgaben:

- MODBUS spezifischen Telegrammheader beim Senden generieren
- Prüfung des MODBUS spezifischen Telegrammheaders beim Empfang
- Prüfung ob die angesprochenen Datenbereiche vorhanden sind
- Datentransfer von/in den parametrierten DB

Der Baustein MOD_CLI ruft intern mehrfach den Baustein TCP_COMM auf.

Der TCP_COMM erfüllt folgende Aufgaben:

- Verbindungs- und Datenhandling unter Verwendung der T-Bausteine aus der Standard-Bibliothek
- Zeitliche Überwachung des Verbindungsaufbaus und –abbaus sowie des Empfangs von Daten

Online-Hilfe

Für den Funktionsbaustein MB_PNHCL steht im SIMATIC-Manager eine Baustein-Online-Hilfe zur Verfügung. Wenn der Baustein markiert und die Taste „F1“ gedrückt wird, wird die Online-Hilfe mit den wichtigsten Informationen zum Baustein geöffnet.

Aufruf des FBs

Der Funktionsbaustein **MB_PNHCL** muss für einen korrekten Programmablauf in 2 OBs eingebaut werden:

- im Anlauf-OB100 und
- in einem zyklischen OB (OB1 oder in einem zeitgesteuerten OB, z.B. OB35)

Dabei muss derselbe Instanz-Datenbaustein verwendet werden.
Die anderen in der Bibliothek enthaltenen FBs MOD_CLI und TCP_COMM werden unterlagert aufgerufen und dürfen nicht zusätzlich in einem OB aufgerufen werden.

Der gleichzeitige Aufruf des FB MB_PNHCL im OB1 und in einem zeitgesteuerten OB (z.B. OB35) ist nicht zulässig.

Der OB121 **muss** in der CPU vorhanden sein. Nähere Informationen dazu erhalten Sie im **Kapitel „Lizenzierung“**.

Anlauf des FBs

Der Funktionsbaustein MB_PNHCL wird im OB100 einmal unbedingt aufgerufen.

- Die Initialisierungsparameter müssen entsprechend der Anlagenkonfiguration belegt sein.
- Die Initialisierungsparameter werden in den Instanz-DB übernommen.
- Die Laufzeitparameter werden im Anlauf nicht ausgewertet.
- Die Daten aus MODBUS_HPARAM_PN werden auf Plausibilität überprüft.

Zyklischer Betrieb des FBs

Im zyklischen Betrieb wird der FB MB_PNHCL z.B. im OB35 aufgerufen.

- Anhand der Laufzeitparameter werden die Funktionen des Bausteins aktiviert.
- Während ein Auftrag läuft, werden Änderungen an den Laufzeitparametern nicht ausgewertet.
- Die Initialisierungsparameter werden nicht ausgewertet, solange keine manuelle Initialisierung durchgeführt wird.

Programmierfehler OB121

Ist der Modbus-Baustein für diese CPU noch nicht lizenziert, wird OB121 aufgerufen.

Warnung

Falls der OB121 in der Steuerung fehlt, wird die CPU in den STOP-Zustand gesetzt.

Verbindungs-bearbeitung

Den aktiven Verbindungsaufbau führt der Modbus Client aus. Die Daten hierfür werden aus den Verbindungsparametern im DB MODBUS_HPARAM_PN ausgelesen.

Über einen Parameter im Verbindungsparameterblock (*active_est*) wird festgelegt, ob die PN-CPU als aktiver oder als passiver Kommunikationspartner fungieren soll.
Zur Laufzeit wird bei beiden Verbindungstypen, aktiv und passiv, mit der Funktion TCON ein Kommunikationskanal zum Koppelpartner geöffnet.

Der Zeitpunkt des Verbindungsaufbaus wird mit einem Parameter im DB MODBUS_HPARAM_PN festgelegt (*connect_at_startup)*.

Der Verbindungsabbau wird mit dem Parameter DISCONNECT am FB MB_PNHCL geregelt.

Auftragsanstoß

Durch einen positiven Flankenwechsel am Triggereingang ENQ wird ein Auftrag initiiert. Abhängig von den Eingangsparametern UNIT, DATA_TYPE, START_ADDRESS, LENGTH und WRITE_READ wird ein MODBUS Anforderungstelegramm generiert und zur Partnerstation über die TCP/IP Verbindung gesendet. Der Baustein wartet die parametrierte Zeit RECV_TIMEOUT auf eine Antwort vom Server.

Handling einer fehlerhaften Verbindung

Der Baustein MB_PNHCL erkennt einen Verbindungsfehler, wenn bei einer Übertragung von Datentelegrammen die Kommunikationsfunktionen TSEND/TRCV einen Fehler melden. Nach Anzeige des Fehlercodes wird im Folgenden der Status A0FF angezeigt. Dies bedeutet, dass die Verbindung zwar projektiert, aber derzeit nicht aufgebaut ist.

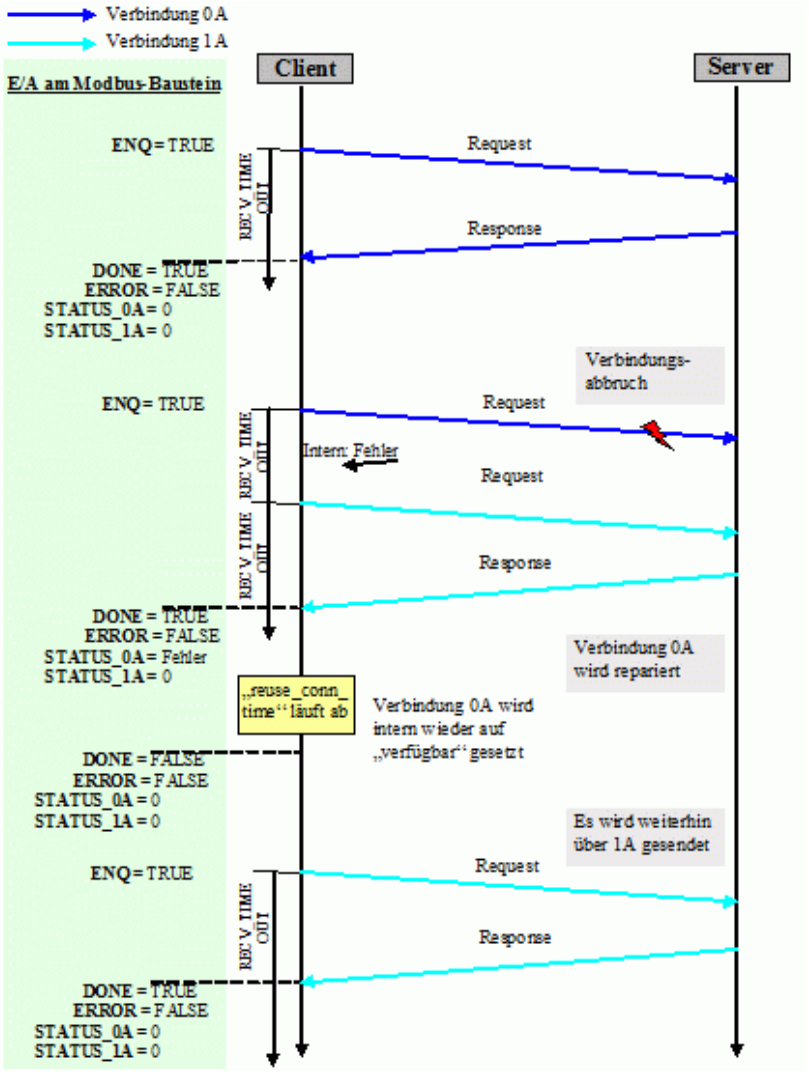
Wird ein Fehler auf einer Verbindung erkannt, wird die Zeit am Parameter **„reuse_conn_time“** gestartet. Solange der Timer „reuse_conn_time“ noch läuft, wird nicht versucht Modbustelegramme über diese defekte Verbindung zu senden.

Nach Ablauf dieser Zeit wird versucht, die fehlerhafte Verbindung wieder aufzubauen.

Telegramme über 1 Verbindung senden

Mit der Einstellung use_all_conn = FALSE wird das MODBUS Telegramm über 1 - die derzeit aktive - Verbindung gesendet. Im Falle einer Zeitüberschreitung (keine Antwort vom Server) oder eines Verbindungsfehlers wird versucht, das Telegramm nacheinander über die anderen (maximal 4) projektierten Verbindungen zu senden. Die Reihenfolge dabei ist 0A, 1A, 0B und 1B. Konnte ein Telegramm erfolgreich über eine Verbindung übertragen werden, wird diese Verbindung als „aktiv“ markiert und der weitere Telegrammverkehr wird über diese Verbindung abgewickelt. Im Falle eines Verbindungsfehlers der aktiven Verbindung wird erneut versucht, nacheinander über alle projektierten Verbindungen das Telegramm zu senden. Schlagen alle Sendeversuche fehl, werden ERROR und STATUS_x entsprechend gesetzt.

Wird ein Antworttelegramm empfangen, wird eine Plausibilitätsprüfung durchgeführt. Verläuft diese positiv, werden die erforderlichen Aktionen durchgeführt und der Auftrag wird ohne Fehler beendet, der Ausgang DONE wird gesetzt. Wurden bei der Prüfung Fehler erkannt, wird der Auftrag mit Fehler beendet, das Bit ERROR wird gesetzt und eine Fehlernummer an STATUS_x angezeigt. In diesem Fall erfolgt kein erneuter Sendeversuch des Telegramms auf der nächsten projektierten Verbindung. Eine Umschaltung auf die anderen projektierten Verbindungen findet nur statt, wenn ein Verbindungsfehler erkannt oder keine Antwort empfangen wurde.

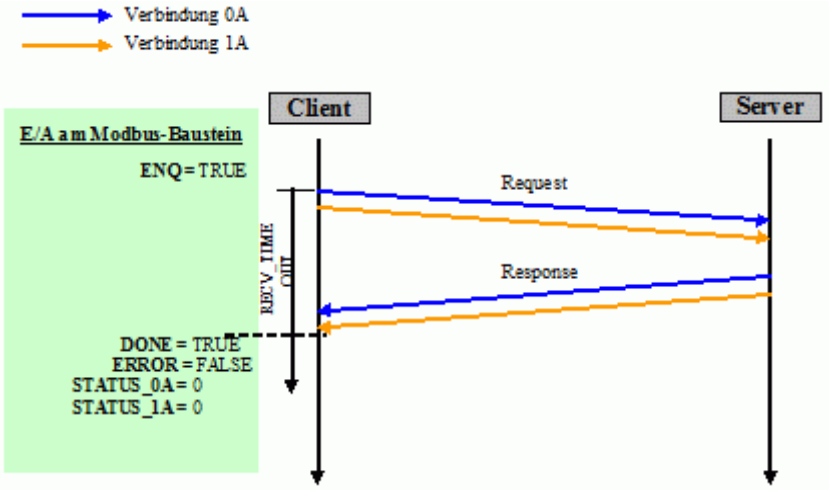


Telegramme über alle Verbindungen senden

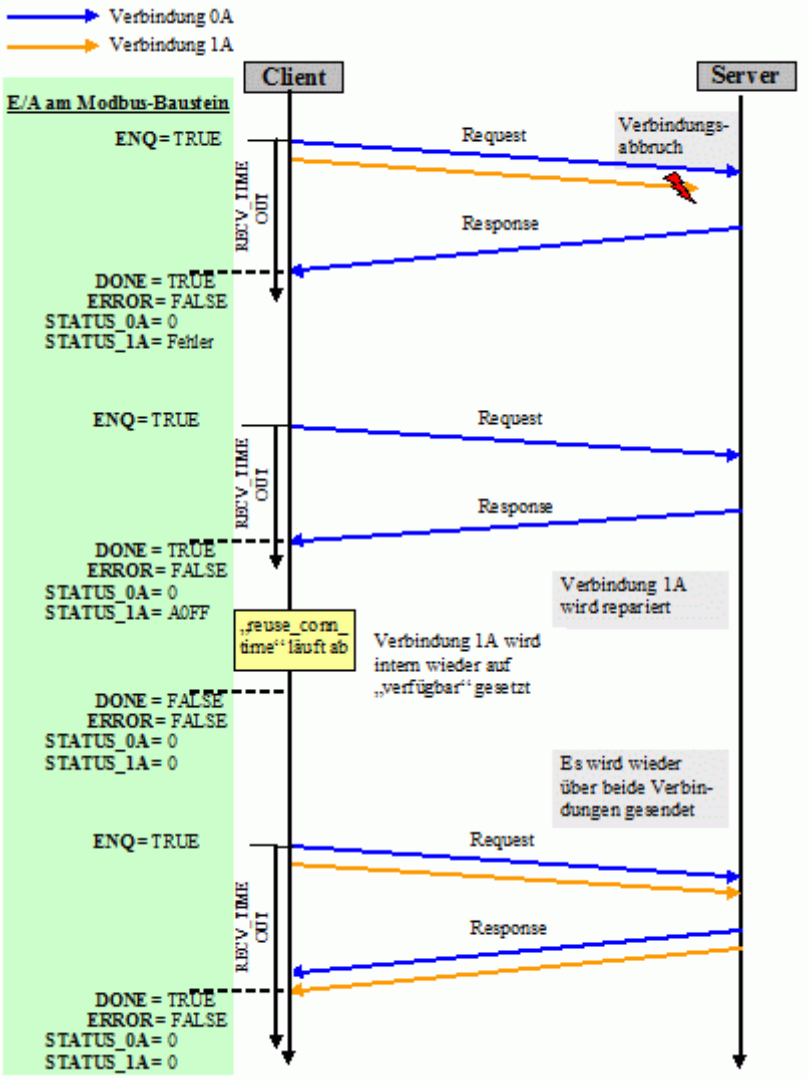
Mit der Einstellung use_all_conn = TRUE wird das MODBUS Telegramm über alle projektierten, aufgebauten Verbindungen gesendet. Nach dem Empfang eines Antworttelegramms auf einer der Verbindungen wird eine Plausibilitätsprüfung durchgeführt. Verläuft diese positiv, werden die erforderlichen Aktionen durchgeführt.
Die Ausgänge **DONE, ERROR und STATUS_x** werden erst dann **aktualisiert**, wenn **auf allen projektierten Verbindungen** der aktivierte Auftrag beendet wurde – entweder ein Antwortelegramm empfangen wurde oder die Überwachungszeit abläuft.
Wurde auf mindestens 1 Verbindung ein gültiges Antworttelegramm empfangen, wird der Ausgang DONE gesetzt.
Wurde auf allen Verbindungen ein Fehler erkannt, wird das Bit ERROR gesetzt und die Fehlernummern an STATUS_x angezeigt.

Ist eine der projektierten Verbindungen ausgefallen, werden die folgenden MODBUS Telegramme nicht mehr über die defekte Verbindung gesendet.

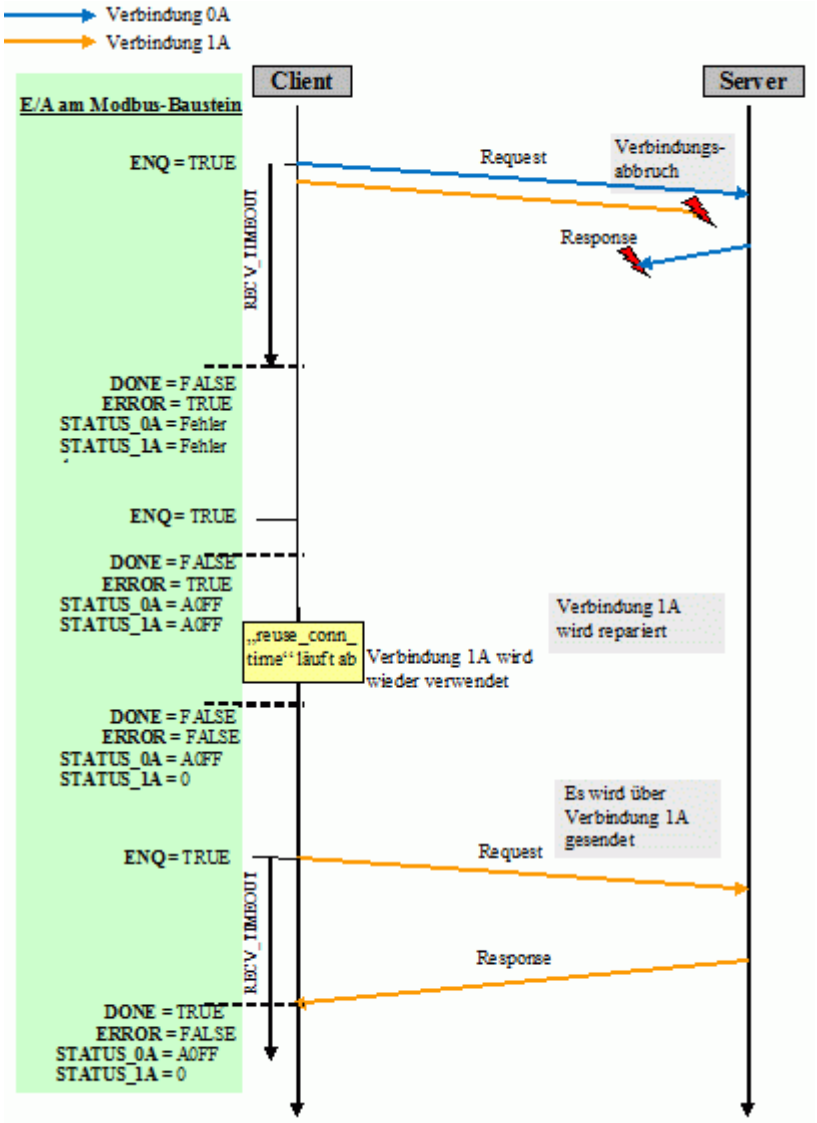
Fall a) Alle Antworttelegramme werden fehlerfrei empfangen.



Fall b) Mindestens 1 Antworttelegramm wird nicht empfangen.



Fall c) Es wird kein Antworttelegramm empfangen.



7.2 Parameter des Funktionsbausteins MB_PNHCL

Parameter	Dekl	Typ	Beschreibung	Wertebereich	Init
id_0_a	IN	WORD	Verbindungs-ID für CPU0 zum Koppelpartner (Knoten A), gemäß der Projektierung im Parameter-DB	1 bis 4095 W#16#1 bis W#16#FFF	ja
id_1_a	IN	WORD	Verbindungs-ID für CPU1 zum Koppelpartner (Knoten A), gemäß der Projektierung im Parameter-DB	1 bis 4095 W#16#1 bis W#16#FFF	ja
id_0_b	IN	WORD	Verbindungs-ID für CPU0 zum Koppelpartner (Knoten B), gemäß der Projektierung im Parameter-DB	1 bis 4095 W#16#1 bis W#16#FFF	ja
id_1_b	IN	WORD	Verbindungs-ID für CPU1 zum Koppelpartner (Knoten B), gemäß der Projektierung im Parameter-DB	1 bis 4095 W#16#1 bis W#16#FFF	ja
db_param	IN	BLOCK_DB	Parameter-DB, enthält alle Verbindungs- und Modbusdaten für diese Modbusbaustein-Instanz	CPU-abhängig	ja
reuse_conn_time	IN	TIME	Zeit, nach der versucht wird, die Verbindung wieder aufzubauen; mindesten 1s	T#1s bis T#+24d20h31m23s	ja
use_all_conn	IN	BOOL	Telegramm über 1 Verbindung senden Telegramm über alle projektierten Verbindungen senden	FALSE TRUE	ja
RECV_TIMEOUT	IN	TIME	Überwachungszeit für Datenempfang, mind. 20ms	T#20ms bis T#+24d20h31m23s	nein
CONN_TIMEOUT	IN	TIME	Überwachungszeit für den Verbindungsauf- und –abbau, mind. 100ms	T#100ms bis T#+24d20h31m23s	nein
DISCON-NECT	IN	BOOL	TRUE: Verbindungsabbau nach Empfang des Antworttelegramms	TRUE/FALSE	nein
REG_KEY_DB	IN	BLOCK_DB	Datenbaustein mit dem Registrierungsschlüssel für die Lizenzierung	CPU-abhängig	nein
Init	IN	BOOL	Manuelle Initialisierung bei positiver Flanke	TRUE/FALSE	nein
ENQ	IN	BOOL	Auftragsanstoß bei positiver Flanke	TRUE/FALSE	nein
DATA_TYPE	IN	BYTE	zu bearbeitender Datentyp Coils Inputs Holding Register Input Register	 1 2 3 4	nein
START_ADDRESS	IN	WORD	MODBUS Startadresse	0 bis 65535 W#16#0000 bis W#16#FFFF	nein

Parameter	Dekl	Typ	Beschreibung	Wertebereich	Init
LENGTH	IN	WORD	Anzahl der zu bearbeitenden Werte Coils Lesende Funktion Schreibende Funktion Inputs Lesende Funktion Holding Register Lesende Funktion Schreibende Funktion Input Register Lesende Funktion	 1 bis 2000 1 bis 1968 1 bis 2000 1 bis 125 1 bis 123 1 bis 125	nein
WRITE_ READ	IN	BOOL	Schreibzugriff Lesezugriff	TRUE FALSE	nein
UNIT	IN	BYTE	Unit Identifier	0 bis 255 B#16#0 bis B#16#FF	nein
LICENSED	OUT	BOOL	Lizenzzustand des Bausteins: Baustein ist lizenziert Baustein ist nicht lizenziert	 TRUE FALSE	nein
BUSY	OUT	BOOL	Bearbeitungszustand eines Modbustelegramms in Bearbeitung nicht in Bearbeitung	 TRUE FALSE	nein
DONE	OUT	BOOL	TRUE: aktivierter Auftrag wurde auf mindestens 1 Verbindung fehlerfrei beendet	TRUE/FALSE	nein
ERROR	OUT	BOOL	TRUE: Es ist auf allen Verbindungen ein Fehler aufgetreten.	TRUE/FALSE	nein
ESTAB_0A	OUT	BOOL	TRUE: Verbindung 0A ist aufgebaut	TRUE/FALSE	nein
STATUS_0A	OUT	WORD	Status für Verbindung 0A	0 bis FFFF	nein
ESTAB_1A	OUT	BOOL	TRUE: Verbindung 1A ist aufgebaut	TRUE/FALSE	nein
STATUS_1A	OUT	WORD	Status für Verbindung 1A	0 bis FFFF	nein
ESTAB_0B	OUT	BOOL	TRUE: Verbindung 0B ist aufgebaut	TRUE/FALSE	nein
STATUS_0B	OUT	WORD	Status für Verbindung 0B	0 bis FFFF	nein
ESTAB_1B	OUT	BOOL	TRUE: Verbindung 1B ist aufgebaut	TRUE/FALSE	nein
STATUS_1B	OUT	WORD	Status für Verbindung 1B	0 bis FFFF	nein
IDENT_ CODE	OUT	STRING [18]	Identifikation für die Lizenzierung. Fordern Sie mit diesem String die Lizenz an.	Character	nein
RedErrS7	OUT	BOOL	TRUE: Redundanzverlust auf der S7-Seite	TRUE/FALSE	nein
RedErrDev	OUT	BOOL	TRUE: Redundanzverlust auf der Seite des Kommunikationspartners	TRUE/FALSE	nein
TotComErr	OUT	BOOL	TRUE: Vollständiger Kommunikationsausfall	TRUE/FALSE	nein
Init_Error	OUT	BOOL	TRUE: Bei der manuellen Initialisierung ist ein Fehler aufgetreten.	TRUE/FALSE	nein
Init_Status	OUT	WORD	Status der manuellen Initialisierung	0 bis FFFF	nein

Allgemeines

Die Parameter des FB MB_PNHCL gliedern sich in zwei Gruppen:

- Initialisierungsparameter (klein geschrieben)
- Laufzeitparameter (groß geschrieben)

Die **Initialisierungsparameter** werden nur beim Aufruf im OB100 ausgewertet und in den Instanz-DB übernommen. Die Initialisierungsparameter sind in der obigen Tabelle in der Spalte „INIT“ mit „ja“ gekennzeichnet.
Eine Änderung der Initialisierungsparameter während des laufenden Betriebs hat keine Auswirkung. Nach einer Änderung dieser Parameter z.B. im Testbetrieb muss der Instanz-DB (I-DB) durch STOP/RUN der CPU neu initialisiert werden.
Die Initialisierung kann auch mit Hilfe des Parameters „Init“ durchgeführt werden.

Laufzeitparameter können im zyklischen Betrieb verändert werden. Es ist nicht sinnvoll die Eingangsparameter zu ändern während ein Auftrag läuft. Mit den Vorbereitungen für den nächsten Auftrag und den damit verbundenen Änderungen der Parameter sollte gewartet werden, bis der vorherige Auftrag mit DONE oder ERROR beendet wurde.

Die Ausgangsparameter sind **dynamische Anzeigen** und stehen somit nur **1 CPU-Zyklus** an. Sie müssen für eine eventuelle Weiterverarbeitung oder eine Anzeige in der Variablentabelle in andere Speicherbereiche kopiert werden.

Wertebereiche

Bei den Wertebereichen für die verschiedenen Parameter sind ggf. auch CPU-spezifische Einschränkungen zu beachten.

id_0_a, id_1_a
id_0_b, id_1_b

Für jede Verbindung von der PN-CPU zu einem Kommunikationspartner wird eine Verbindungs-ID benötigt. Für jede logische Verbindung muss eine andere Verbindungs-ID verwendet werden. Diese Verbindungs-ID wird im Verbindungsparameterblock projektiert, welcher im Parameterdatenbaustein MODBUS_HPARAM_PN enthalten ist. Die Verbindungs-ID beschreibt eindeutig die Verbindung von der CPU zum Koppelpartner und kann Werte von 1 bis 4095 annehmen.
Die Verbindungs-ID aus dem Verbindungsparameterblock ist hier einzutragen und muss CPU-weit eindeutig sein.

id_0_a bezeichnet die Verbindung von CPU0 zum Koppelpartner/Knoten A
id_1_a bezeichnet die Verbindung von CPU1 zum Koppelpartner/Knoten A
id_0_b bezeichnet die Verbindung von CPU0 zum Koppelpartner/Knoten B
id_1_b bezeichnet die Verbindung von CPU1 zum Koppelpartner/Knoten B

Die Verbindung 0A ist die Default-Verbindung und muss zwingend projektiert werden.

Ist der Koppelpartner standalone aufgebaut, dann werden nur die Parameter id_0_a und id_1_a benötigt. Ist die S7 standalone aufgebaut, dann werden nur die Parameter id_0_a und id_0_b benötigt.Sind beide Kommunikationspartner redundant aufgebaut, werden alle 4 Verbindungen projektiert.

db_param

Der Parameter db_param bezeichnet die Nummer des Datenbausteins MODBUS_HPARAM_PN. In diesem Parameterdatenbaustein sind die verbindungs- und modbusspezifischen Parameter hinterlegt, die für die Kommunikation zwischen der PN-CPU und dem Koppelpartner notwendig sind.

Der Wertebereich für diesen Parameter ist CPU-abhängig. Die DB-Nummer 0 ist nicht zulässig, da diese für das System reserviert ist.
Die Eingabe der DB-Nummer erfolgt im Klartext in der Form „DBxy“.

Für jede Modbusbaustein-Instanz ist ein eigener Parameter-Datenbaustein notwendig.

reuse_conn_time

Mit diesem Parameter wird festgelegt, in welchem Zeitintervall eine als defekt erkannte Verbindung wieder bei der Kommunikation berücksichtigt werden soll. Bei einem Verbindungsfehler auf 0A, 1A, 0B oder 1B wird ein Timer mit der an diesem Parameter angegebenen Zeit gestartet. Während dieser Timer läuft, wird nicht versucht, die Verbindung aufzubauen oder ein Modbustelegramm über diese Verbindung zu senden. Wenn der Timer aufgelaufen ist, aktiviert der Modbusbaustein automatisch einen neuen Verbindungsaufbau.

Konnte die Verbindung seit der Initialisierung noch nie aufgebaut werden, werden Verbindungsfehler mit ERROR = TRUE angezeigt, anderenfalls mit ERROR = FALSE.

Die minimal einstellbare Zeit ist 1 Sekunde.

use_all_conn

Dieser Parameter legt fest, über wie viele Verbindungen die Modbus-telegramme gesendet werden sollen. Bei FALSE werden die Modbus-telegramme nur über 1 Verbindung gesendet. Ist der Parameter TRUE, werden die Modbustelegramme über alle projektierten Verbindungen gesendet.

RECV_TIMEOUT

Die Überwachungszeit RECV_TIMEOUT überwacht den Empfang des Antworttelegramms vom Koppelpartner. Der Minimalwert beträgt 20ms.

Wenn die RECV_TIMEOUT auf < 20ms gesetzt wird, erscheint eine entsprechende Fehlermeldung und der aktivierte Auftrag wird abgewiesen. Bei Ablauf der Überwachungszeit wird der aktivierte Auftrag mit Fehler beendet.

CONN_TIMEOUT

Mit der Zeit CONN_TIMEOUT wird der Verbindungsaufbau bzw. -abbau überwacht. Der minimale Wert ist 100ms.

Konnte innerhalb der parametrierten Überwachungszeit die Verbindung nicht erfolgreich auf- bzw. abgebaut werden, erscheint eine entsprechende Fehlermeldung am Ausgang STATUS_x.

Bei *connect_at_startup* = TRUE wird eine zu gering parametrierte CONN_TIMEOUT auf 5 s gesetzt. Im zyklischen Betrieb wird bei einer zu kleinen CONN_TIMEOUT eine Fehlermeldung ausgegeben und der aktivierte Auftrag abgewiesen.

DISCONNECT

Mit DISCONNECT = TRUE wird festgelegt, dass die Verbindung nach dem Datentransfer abgebaut werden soll. Ist dieser Parameter auf TRUE gesetzt, wird die Zeit reuse_conn_time für einen Wiederaufbau der Verbindungen nicht gestartet.

Dieser Parameter ist ein Laufzeitparameter und kann entsprechend der Anforderung beliebig gesetzt bzw. zurückgesetzt werden.

REG_KEY_DB

Der Baustein muss auf jedem H-System lizenziert werden. Mit der korrekten Eingabe des Freischaltcodes wird der Baustein lizenziert und die Modbus-Kommunikation kann ohne Einschränkungen genutzt werden. Es wird die Datenbausteinnummer angegeben, der den Freischaltcode enthält. Es ist möglich, mehrere Freischaltcodes untereinander in den DB einzutragen. Der Modbusbaustein durchsucht den DB nach dem passenden Freischaltcode. Weitere Informationen entnehmen Sie bitte dem **Kapitel „Lizenzierung“**.

Init

Mit dem Parameter Init = TRUE ist eine manuelle Initialisierung des Modbusbausteins möglich. Die Initialisierung kann nur durchgeführt werden, wenn aktuell kein Auftrag läuft. Dies muss mit ENQ = FALSE und BUSY = FALSE programmtechnisch sichergestellt werden.
Bei der manuellen Initialisierung ist zu beachten, dass die Initialisierungsparameter im zyklischen OB parametriert werden müssen.

Warnung

Bei einer manuellen Initialisierung werden die projektierten Verbindungen abgebaut. Bei Änderung der id-Parameter, müssen die Verbindungen vor der manuellen Initialisierung mit DISCONNECT = TRUE manuell abgebaut werden.

ENQ

Mit einer positiven Flanke wird der Datentransfer initiiert. Mit den Werten der Eingangsparameter UNIT, WRITE_READ, DATA_TYPE, START_ADDRESS und LENGTH wird das Anforderungstelegramm generiert. Ein neuer Auftrag kann nur gesendet werden, wenn der vorherige mit DONE oder ERROR abgeschlossen wurde.
Wenn die Verbindung nicht aufgebaut ist (ESTAB_x = FALSE), wird zuerst die Verbindung aufgebaut und dann der Datentransfer ausgeführt.

DATA_TYPE

Der Parameter DATA_TYPE zeigt an, welcher Modbus-Datentyp mit dem aktuellen Telegramm bearbeitet wird. Es sind folgende Werte zulässig:

Coils	B#16#1
Inputs	B#16#2
Holding Register	B#16#3
Input Register	B#16#4

Die unterschiedlichen Datentypen haben einen direkten Zusammenhang mit den verwendeten Funktionscodes.

Datentyp	DATA_ TYPE	Funktion	Länge	single_ write	Funk-tions-code
Coils	1	lesen	beliebig	irrelevant	1
Coils	1	schreiben	1	TRUE	5
Coils	1	schreiben	1	FALSE	15
Coils	1	schreiben	>1	irrelevant	15
Inputs	2	lesen	beliebig	irrelevant	2
Holding Register	3	lesen	beliebig	irrelevant	3
Holding Register	3	schreiben	1	TRUE	6
Holding Register	3	schreiben	1	FALSE	16
Holding Register	3	schreiben	>1	irrelevant	16
Input Register	4	lesen	beliebig	irrelevant	4

START_ADDRESS

Der Parameter START_ADDRESS bestimmt die erste MODBUS-Adresse die geschrieben bzw. gelesen wird.

LENGTH

Der Parameter LENGTH bestimmt die Anzahl der MODBUS-Werte die geschrieben bzw. gelesen werden.

Bei lesenden Funktionen sind pro Telegramm für Holding und Input Register maximal 125 Register möglich. Für Coils und Inputs sind maximal 2000 Bits möglich. Bei schreibenden Funktionen beträgt bei Holding Register die maximale Anzahl 123 Register und bei Coils 1968 Bits.
Die mit einem Anforderungstelegramm bearbeiteten Register bzw. Bitwerte müssen innerhalb eines DBs liegen.

WRITE_READ

Dieser Parameter definiert ob eine lesende oder schreibende Funktion ausgeführt werden soll. Hat der Eingang den Wert FALSE, handelt es sich um eine lesende Funktion. Der Wert TRUE definiert eine schreibende Funktion.

Es können nur Holding Register und Coils beschrieben werden. Input Register und Inputs lassen sich lediglich lesen.

UNIT

Der Parameter UNIT, Unit Identifier, bezeichnet die eindeutige Zuordnung des Koppelpartners. Er ist vor allem notwendig, wenn sich hinter einem Konverter mehrere serielle Teilnehmer befinden, die mit unterschiedlichen UNIT Nummern angesprochen werden.

Dieser Eingang ist den Anforderungen entsprechend zu setzen. Der FB übernimmt diesen Wert in das Anforderungstelegramm und überprüft ihn beim Empfang der Antwort. Zu beachten ist, dass einige Koppelpartner eine bestimmte UNIT-Nummer erwarten.

LICENSED

Ist dieser Ausgang auf TRUE gesetzt, ist der Modbusbaustein auf dieser CPU lizenziert. Hat der Ausgang den Zustand FALSE, wurde kein oder ein fehlerhafter Lizenzstring eingetragen. Weitere Informationen entnehmen Sie bitte dem **Kapitel „Lizenzierung“**.

BUSY

Wenn dieser Ausgang gesetzt ist, wird gerade ein Modbustelegramm bearbeitet.

DONE

Der aktivierte Auftrag wurde auf mindestens 1 Verbindung fehlerfrei beendet. Bei einer lesenden Funktion wurden die Antwortdaten vom Server bereits im DB eingetragen, bei einer schreibenden Funktion wurde vom Server die Antwort auf das Anforderungstelegramm erhalten.

ERROR

Wenn dieser Ausgang gesetzt ist, wurde auf allen aktiven Verbindungen ein Fehler erkannt.

use_all_conn = FALSE:
Im Falle eines Protokollfehlers wird ERROR sofort gesetzt. Im Falle eines Verbindungsfehlers werden alle projektierten Verbindungen geprüft und ERROR erst dann gesetzt, wenn alle Verbindungen fehlerhaft sind.

use_all_conn = TRUE:
Ist dieser Ausgang gesetzt, wurde auf allen projektierten Verbindungen ein Fehler erkannt.

Die Fehlernummern werden an den STATUS-Ausgängen angezeigt.

ESTAB_0A, ESTAB_1A, ESTAB_0B, ESTAB_1B

Mit ESTAB_x = TRUE wird angezeigt, dass eine Verbindung zum Koppelpartner besteht und Daten übertragen werden können.

Ist ESTAB_x = FALSE besteht keine Verbindung zum Koppelpartner.

Bei Ausfall von mindestens 1 projektierten Verbindung werden diese Ausgänge spätestens nach Ablauf von reuse_conn_time aktualisiert.

STATUS_0A, STATUS_1A, STATUS_0B, STATUS_1B

Die Ausgänge STATUS_x zeigen bei gesetztem ERROR die Fehlernummer, bei nicht gesetztem ERROR Statusinformationen für die entsprechende Verbindung an.

Die Fehlernummern und Statusinformationen sind im Kapitel „Diagnose“ beschrieben.

IDENT_CODE

Nach dem Anlauf der CPU0 wird an diesem Parameter eine 18stellige Identifikationskennung angezeigt, mit dem der REG_KEY (Freischaltcode) für die Modbus-Kommunikation beantragt wird.

Weitere Informationen entnehmen Sie bitte dem **Kapitel „Lizenzierung“**.

RedErrS7

Der Ausgang RedErrS7 = TRUE zeigt einen Redundanzfehler auf der SIMATIC-Seite an.
Bei einseitiger Redundanz bedeutet dies, dass die Verbindung von CPU0 oder CPU1 ausgefallen ist. Bei beidseitiger Redundanz sind beide Verbindungen von CPU0 oder beide Verbindungen von CPU1 ausgefallen.

Weitere Informationen entnehmen Sie bitte dem Kapitel „Diagnosemeldungen über Alarm-Bits“.

RedErrDev

Der Ausgang RedErrDev = TRUE zeigt einen Redundanzfehler auf der Seite des Koppelpartners an.
Bei einseitiger Redundanz bedeutet dies, dass die Verbindung vom Knotenpunkt A zum CPU0 oder CPU1 ausgefallen ist. Bei beidseitiger Redundanz sind beide Verbindungen zum Knoten A oder beide Verbindungen zum Knoten B des Koppelpartners ausgefallen.

Weitere Informationen entnehmen Sie bitte dem Kapitel „Diagnosemeldungen über Alarm-Bits“.

TotComErr

Der Ausgang TotComErr zeigt mit TRUE einen vollständigen Kommunikationsverlust an, d.h. alle projektierten Verbindungen sind gestört.

Weitere Informationen entnehmen Sie bitte dem Kapitel „Diagnosemeldungen über Alarm-Bits“.

Init_Error

Trat bei der manuellen Initialisierung ein Fehler auf, wird dies mit Init_Error = TRUE angezeigt.

Init_Status

Der Ausgang Init_Status zeigt bei gesetztem Init_Error die Fehlernummer an. Die Fehlernummern sind im Kapitel „Diagnose“ beschrieben.

7.3 Beispiel für die Adressabbildung

Interpretation der Modbus Adressen

Das MODBUS Datenmodell basiert auf einer Reihe von Speicherbereichen die unterschiedliche Charakteristiken haben. Die Unterscheidung dieser Speicherbereiche erfolgt bei einigen Systemen, z.B. MODICON PLCs über die Register- bzw. Bitadresse. So wird z.B. das Holding Register mit Offset 0 als Register 40001 bezeichnet (Speichertyp 4xxxx, Reference 0001).

Es führt immer wieder zur Verwirrung, weil in manchen Handbüchern die Registeradresse des Application Layers und in anderen die tatsächlich im Protokoll übertragene Register-/Bitadresse beschrieben und gemeint ist.

Der FB MODBUS verwendet bei seinen Parametern *start_x*, *end_x* und START_ADDRESS die **tatsächlich übertragene Modbusadresse**. Es können also mit jedem Funktionscode Register-/Bitadressen von 0000_H bis $FFFF_H$ übertragen werden.

Beispiel: Parametrierung der Datenbereiche

data_type_1 *db_1* *start_1* *end_1*	B#16#3 W#16#B W#16#0 W#16#1F3	Holding Register DB 11 Anfangsadresse: 0 Endadresse: 499
data_type_2 *db _2* *start _2* *end _2*	B#16#3 W#16#C W#16#2D0 W#16#384	Holding Register DB 12 Anfangsadresse: 720 Endadresse: 900
data_type_3 *db _3* *start _3* *end _3*	B#16#4 W#16#D W#16#2D0 W#16#3E8	Input Register DB 13 Anfangsadresse: 720 Endadresse: 1000
data_type_4 *db _4* *start _4* *end _4*	B#16#0 0 0 0	Nicht verwendet 0 0 0
data_type_5 *db _5* *start _5* *end _5*	B#16#1 W#16#E W#16#280 W#16#4E2	Coils DB 14 Anfangsadresse: 640 Endadresse: 1250
data_type_6 *db _6* *start _6* *end _6*	B#16#2 W#16#F W#16#6A4 W#16#8FC	Inputs DB 15 Anfangsadresse:1700 Endadresse: 2300
data_type_7 *db _7* *start _7* *end _7*	B#16#1 W#16#10 W#16#6A4 W#16#8FC	Coils DB 16 Anfangsadresse: 1700 Endadresse: 2300
data_type_8 *db _8* *start _8* *end _8*	B#16#0 0 0 0	Nicht verwendet 0 0 0

Für dieses Beispiel hat:

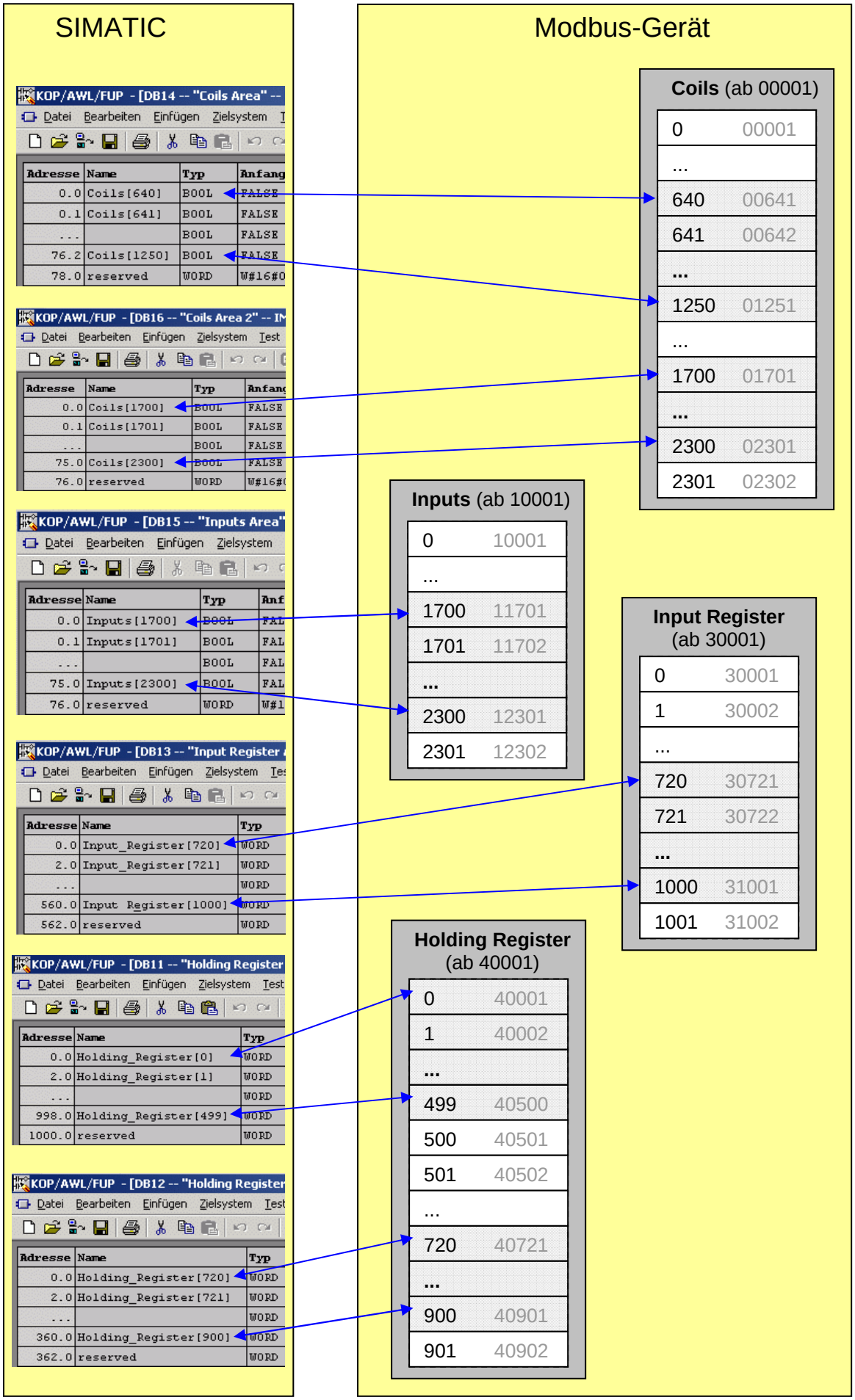
- DB11 eine Größe von 1002 Byte, es werden insgesamt 500 Register abgebildet (Register 0 – Register 499) + 2 reservierte Byte.
- DB12 eine Größe von 364 Byte, es werden insgesamt 181 Register abgebildet (Register 720 – Register 900) + 2 reservierte Byte
- DB13 eine Größe von 564 Byte, es werden insgesamt 281 Input Register abgebildet (Register 720 – Register 1000) + 2 reservierte Byte
- DB14 eine Größe von 80 Byte, es werden insgesamt 611 Coils (Bits) abgebildet (Coil 640 – Coil 1250) + 2 reservierte Byte
- DB15 eine Größe von 78 Byte, es werden insgesamt 601 Inputs (Bits) abgebildet (Input 1700 – Input 2300) + 2 reservierte Byte
- DB16 eine Größe von 78 Byte, es werden insgesamt 601 Coils (Bits) abgebildet (Coil 1700 – Coil 2300) + 2 reservierte Byte

Adressabbildung

Im folgenden Bild sehen Sie die Gegenüberstellung der Simatic-Speicherbereiche mit der register- und bitorientierten Speicheraufteilung der Modbus-Geräte. Dabei wird auf obige Parametrierung Bezug genommen.

Im Modbus-Gerät: Die Modbus-Adressen, die schwarz dargestellt sind, betreffen den Data Link Layer, die grau dargestellten den Applikation Layer.

In der SIMATIC: Die in der 1. Spalte dargestellten SIMATIC-Adressen sind der Offset im DB. In den eckigen Klammern sind die Modbus Registernummern eingetragen.


SIMATIC
Modbus-Gerät
KOP/AWL/FUP - [DB14 -- "Coils Area" --
Datei Bearbeiten Einfügen Zielsystem
Adresse Name Typ Anfang
0.0 Coils[640] BOOL FALSE
0.1 Coils[641] BOOL FALSE
... BOOL FALSE
76.2 Coils[1250] BOOL FALSE
78.0 reserved WORD W#16#0
KOP/AWL/FUP - [DB16 -- "Coils Area 2" -- IM
Datei Bearbeiten Einfügen Zielsystem Test
Adresse Name Typ Anfang
0.0 Coils[1700] BOOL FALSE
0.1 Coils[1701] BOOL FALSE
... BOOL FALSE
75.0 Coils[2300] BOOL FALSE
76.0 reserved WORD W#16#0
KOP/AWL/FUP - [DB15 -- "Inputs Area"
Datei Bearbeiten Einfügen Zielsystem
Adresse Name Typ Anf
0.0 Inputs[1700] BOOL FAL
0.1 Inputs[1701] BOOL FAL
... BOOL FAL
75.0 Inputs[2300] BOOL FAL
76.0 reserved WORD W#1
KOP/AWL/FUP - [DB13 -- "Input Register
Datei Bearbeiten Einfügen Zielsystem Tes
Adresse Name Typ
0.0 Input_Register[720] WORD
2.0 Input_Register[721] WORD
... WORD
560.0 Input Register[1000] WORD
562.0 reserved WORD
KOP/AWL/FUP - [DB11 -- "Holding Register
Datei Bearbeiten Einfügen Zielsystem Test
Adresse Name Typ
0.0 Holding_Register[0] WORD
2.0 Holding_Register[1] WORD
... WORD
998.0 Holding_Register[499] WORD
1000.0 reserved WORD
KOP/AWL/FUP - [DB12 -- "Holding Register
Datei Bearbeiten Einfügen Zielsystem Test
Adresse Name Typ
0.0 Holding_Register[720] WORD
2.0 Holding_Register[721] WORD
... WORD
360.0 Holding_Register[900] WORD
362.0 reserved WORD
Coils (ab 00001)
0 00001
...
640 00641
641 00642
...
1250 01251
...
1700 01701
...
2300 02301
2301 02302
Inputs (ab 10001)
0 10001
...
1700 11701
1701 11702
...
2300 12301
2301 12302
Input Register (ab 30001)
0 30001
1 30002
...
720 30721
721 30722
...
1000 31001
1001 31002
Holding Register (ab 40001)
0 40001
1 40002
...
499 40500
500 40501
501 40502
...
720 40721
...
900 40901
901 40902

7.4 Vom FB verwendete Daten und Standardfunktionen

Instanz DB

Der Funktionsbaustein MB_PNHCL speichert seine Daten in einem Instanz-DB. Dieser Instanz-DB wird beim ersten Aufruf des FB durch STEP 7 generiert.

Der Instanz-Datenbaustein enthält Parameterwerte vom Typ Input, Output sowie statische Variablen die er für seinen Ablauf benötigt. Diese Variablen sind remanent und behalten zwischen den FB-Aufrufen ihre Gültigkeit. Über die Variablen wird der interne Ablauf des FBs gesteuert.

Speicherbedarf des Instanz-DBs:

Instanz-DB	Arbeitsspeicher	Ladespeicher
MB_PNHCL	ca. 3 kByte	ca. 5 kByte

Lokale Variablen

In Summe werden maximal 186 Byte Lokaldaten für einen FB MB_PNHCL-Aufruf benötigt.

Parameter-DB

Die verbindungs- und modbusspezifischen Parameter werden in dem Parameter-DB MODBUS_HPARAM_PN gespeichert.

Zeiten

Der Funktionsbaustein verwendet keine Timer.

Merker

Der Funktionsbaustein verwendet keine Merker.

Standard-FBs zur Verbindungs-bearbeitung

Der im FB MB_PNHCL/MOD_CLI aufgerufene FB TCP_COMM verwendet die Bausteine TCON und TDISCON aus der Standard-Bibliothek für den Auf- und Abbau der Verbindungen zwischen der CPU und dem Kommunikationspartner.

Standard-FBs zur Datenübertragung

Der im FB MB_PNHCL/MOD_CLI aufgerufene FB TCP_COMM verwendet die Bausteine TSEND und TRCV aus der Standard-Bibliothek für die Datenübertragung zwischen der CPU und dem Kommunikationspartner.

MB_PNHCL: SFCs für sonstige Funktionen

Der FB MB_PNHCL verwendet folgende SFCs aus der Standardbibliothek:

- SFC6 „RD_SINFO“
- SFC20 „BLKMOV“
- SFC24 “TEST_DB”
- SFC51 “RDSYSST”
- SFC52 “WR_USMSG”

MOD_CLI: SFCs für sonstige Funktionen

Der FB MOD_CLI verwendet folgende SFCs aus der Standardbibliothek:

- SFC20 „BLKMOV“
- SFC24 “TEST_DB”

TCP_COMM: SFCs für sonstige Funktionen

Der FB TCP_COMM verwendet außer den T-Bausteinen den folgenden SFB aus der Standardbibliothek:

- SFB4 „TON“

Weitere Informationen

Der Parameter TI wird bei dem Baustein MB_PNHCL intern geführt und bei jedem neuen Auftrag um 1 inkrementiert.

Die Zeit, in welcher ein Verbindungsabbruch erkannt werden kann, kann mit dem Parameter Keep Alive-Time beeinflusst werden. Diesen Parameter finden Sie in den Eigenschaften der CPUs in der HW-Konfig.

7.5 Umbenennen / Umverdrahten von Funktionen und Funktionsbausteinen

Veranlassung

Falls in Ihrem Projekt die Nummern der Standardfunktionen bereits verwendet werden oder der Nummernbereich für andere Applikationen reserviert ist (z.B. in CFC), können Sie die intern aufgerufenen Funktionsbausteine FB63, FB64, FB65 und FB66 des FBs TCP_COMM oder die Bausteine MB_PNHCL, MOD_CLI, MOD_SERV und TCP_COMM umverdrahten.

Die Systemfunktionen SFC6, SFC20, SFC24, SFC51 und SFC52 sowie der Systemfunktionsbaustein SFB4 können nicht umbenannt/umverdrahtet werden.

Verhalten

Im SIMATIC Manager von STEP7 sind beim Umverdrahten einige Regeln bezüglich der Bausteinnummern zu beachten.

Wenn Sie die Bausteine aus der Modbusbibliothek umverdrahten wollen, gehen Sie in folgender Reihenfolge vor:

1. FB915 MB_PNHCL
2. FB914 MOD_CLI
3. FB913 TCP_COMM
4. FB63 TSEND
   FB64 TRCV
   FB65 TCON
   FB66 TDISCON

Es müssen nicht alle Funktionen bzw. Funktionsbausteine umverdrahtet werden. Auch wenn Sie nur einige von diesen umverdrahten wollen, muss diese Reihenfolge eingehalten werden.

Umverdrahten

Gehen Sie zum Verdrahten für die FBs folgendermaßen vor:

1. Holen Sie mittels „Extras > Referenzdaten > Anzeigen“ Informationen über die verwendeten Operanden ein.
2. Stellen Sie den Operandenvorrang in den Objekteigenschaften des Bausteinsordners auf „Absolutwert“.
3. Selektieren Sie im SIMATIC Manager den Baustein-Ordner und rufen Sie die Funktion „Extras > Umverdrahten“ auf, um die Operanden in freie Bereiche umzuverdrahten.
4. Um die Symbolik in Diagnosetools weiter nutzen zu können, ziehen Sie die Änderungen in der Symboltabelle nach.

Über „Extras > Referenzdaten > Anzeigen“ können Sie die Änderungen überprüfen.

8 Funktionsbaustein MB_PNHSV – Modbus Server

8.1 Funktionsweise des FB MB_PNHSV

Allgemeines

Die S7 ist Server wenn der remote Partner das Lesen bzw. Schreiben der Daten aus bzw. in die S7 initiiert.

Für die Serverfunktionalität werden die FBs MB_PNHSV, MOD_SERV und TCP_COMM benötigt.

Es können mehrere Instanzen des Bausteins MB_PNHSV im Programm aufgerufen werden. Es gibt seitens der Bibliothek keine Begrenzung der maximalen Anzahl von parallel laufenden Modbusbausteinen. Allerdings gibt es eine CPU-abhängige maximale Anzahl von gleichzeitig aufgebauten Verbindungen. Im Handbuch der CPUs befindet sich die Angabe, wie viele Verbindungen parallel aufgebaut werden können.
Bei mehreren Instanzen von MB_PNHSV ist zu beachten, dass jede Bausteininstanz einen eigenen Parameter-Datenbaustein erhält und die **Verbindungs-IDs CPU-weit eindeutig** sind.

Aufgaben des FB

Der Baustein MB_PNHSV ruft intern mehrfach den Baustein MOD_SERV auf und übernimmt die Lizenzierung sowie die Koordinierung der MOD_SERV-Aufrufe der verschiedenen Verbindungen.

Der Baustein MOD_SERV erfüllt folgende Aufgaben:

- MODBUS spezifischen Telegrammheader beim Senden generieren
- Prüfung des MODBUS spezifischen Telegrammheaders beim Empfang
- Prüfung ob die vom Client angesprochenen Datenbereiche vorhanden sind
- Datentransfer von/in den parametrierten DB
- Exception Telegramme generieren wenn ein Fehler aufgetreten ist

Exception Code	Bedeutung
1	Der gesendete Funktionscode wird nicht unterstützt.
2	Es erfolgte ein Zugriff auf eine nicht vorhandene bzw. nicht zulässige Adresse.
3	Es wurde eine ungültige Länge für diesen Funktionscode angegeben.

Der Baustein MOD_CLI ruft intern mehrfach den Baustein TCP_COMM auf.

Der TCP_COMM erfüllt folgende Aufgaben:

- Verbindungs- und Datenhandling unter Verwendung der T-Bausteine aus der Standard-Bibliothek
- Zeitliche Überwachung des Verbindungsaufbaus- und –abbaus sowie des Empfangs von Daten

Online-Hilfe

Für den Funktionsbaustein MB_PNHSV steht im SIMATIC-Manager eine Baustein-Online-Hilfe zur Verfügung. Wenn der Baustein markiert und die Taste „F1“ gedrückt wird, wird die Online-Hilfe mit den wichtigsten Informationen zum Baustein geöffnet.

Aufruf des FBs

Der Funktionsbaustein **MB_PNHSV** muss für einen korrekten Programmablauf in 2 OBs eingebaut werden:

- im Anlauf-OB100 und
- in einem zyklischen OB (OB1 oder in einem zeitgesteuerten OB, z.B. OB35)

Dabei muss derselbe Instanz-Datenbaustein verwendet werden.
Die anderen in der Bibliothek enthaltenen FBs MOD_SERV und TCP_COMM werden unterlagert aufgerufen und dürfen nicht zusätzlich in einem OB aufgerufen werden.

Der gleichzeitige Aufruf des FB MB_PNHSV im OB1 und in einem zeitgesteuerten OB (z.B. OB35) ist nicht zulässig.

Der OB121 **muss** in der CPU vorhanden sein. Nähere Informationen dazu erhalten Sie im **Kapitel „Lizenzierung“**.

Anlauf des FBs

Der Funktionsbaustein MB_PNHSV wird im OB100 einmal unbedingt aufgerufen.

- Die Initialisierungsparameter müssen entsprechend der Anlagenkonfiguration belegt sein.
- Die Initialisierungsparameter werden in den Instanz-DB übernommen.
- Die Laufzeitparameter werden im Anlauf nicht ausgewertet.
- Die Daten aus MODBUS_HPARAM_PN werden auf Plausibilität überprüft

Zyklischer Betrieb des FBs

Im zyklischen Betrieb wird der FB MB_PNHSV z.B. im OB35 aufgerufen.

- Anhand der Laufzeitparameter werden die Funktionen des Bausteins aktiviert.
- Während ein Auftrag läuft, werden Änderungen an den Laufzeitparametern nicht ausgewertet.
- Die Initialisierungsparameter werden nicht ausgewertet, solange keine manuelle Initialisierung durchgeführt wird.

Programmierfehler OB121

Ist der Modbus-Baustein für diese CPU noch nicht lizenziert, wird OB121 aufgerufen.

Warnung

Falls der OB121 in der Steuerung fehlt, wird die CPU in den STOP-Zustand gesetzt.

Instanz-DB: Informationen zum Request des Client

Bei jedem Request des Client werden die Werte zum ausgeführten Auftrag im I-DB des Servers in einem Informationsblock gespeichert. Sie können bei Bedarf im Anwenderprogramm ausgelesen werden. Folgende Werte werden im I-DB für jede Verbindung zwischengespeichert und sind bei NDR = TRUE gültig:

Adresse im I-DB für Verbindung 0A	Variablenname	Beschreibung
DBX 66.5	CONNECTION[1]. WRITE_READ	TRUE: Schreiben in S7 FALSE: Lesen aus S7
DBB 67	CONNECTION[1]. UNIT	Unit-Nummer
DBB 68	CONNECTION[1]. DATA_TYPE	Adressierter Datentyp 1: Coils 2: Inputs 3: Holding Register 4: Input Register
DBW 70	CONNECTION[1]. START_ADDRESS	Anfangsadresse
DBW 72	CONNECTION[1]. LENGTH	Anzahl bearbeiteter Register / Bits
DBW 74	CONNECTION[1]. TI	Transaction Identifier (Laufnummer)
DBD 88	CONNECTION[1]. Cnt_NDR	Zähler für fehlerfrei bearbeitete Aufträge
DBD 92	CONNECTION[1]. Cnt_ERROR	Zähler für erkannte Fehler

Für die Verbindung 1A (CONNECTION[2]) beginnt der Informationsblock ab Adresse DBX 96.0
Für die Verbindung 0B (CONNECTION[3]) beginnt der Informationsblock ab Adresse DBX 128.0
Für die Verbindung 1B (CONNECTION[4]) beginnt der Informationsblock ab Adresse DBX 160.0

Verbindungs-bearbeitung

Der Modbus Server führt den passiven Verbindungsaufbau aus. Die Daten hierfür werden aus den Verbindungsparametern im DB MODBUS_HPARAM_PN ausgelesen.

Über einen Parameter im Verbindungsparameterblock (*active_est*) wird festgelegt ob die PN-CPU als aktiver oder als passiver Kommunikations-partner fungieren soll.
Zur Laufzeit wird bei beiden Verbindungstypen, aktiv und passiv, mit der Funktion TCON ein Kommunikationskanal zum Koppelpartner geöffnet.

Der Zeitpunkt des Verbindungsaufbaus wird mit einem Parameter im DB MODBUS_HPARAM_PN festgelegt (*connect_at_startup)*.

Der Verbindungsabbau wird mit dem Parameter DISCONNECT am FB MB_PNHSV geregelt.

Aktivierung des FBs

Durch einen positiven Pegel am Triggereingang ENR ist der FB zum Empfang eines Anforderungstelegramms vom Client bereit. Der Server verhält sich dabei passiv.

Bei ENR = TRUE sind alle projektierten Verbindungen aktiv und empfangsbereit. Es findet keine Umschaltung zwischen den Verbindungen statt. Der Client kann wahlweise nur über 1 oder über alle Verbindungen senden.

Die empfangenen Telegramme werden überprüft. Verläuft die Prüfung positiv, wird das Antworttelegramm generiert und gesendet. Der beendete Telegrammverkehr wird dem Anwender mitgeteilt indem das Bit NDR_x für die entsprechende Verbindung gesetzt wird.

Ein fehlerhaftes Anforderungstelegramm bewirkt eine Fehlermeldung das Bit ERROR der jeweiligen Verbindung wird gesetzt und im Parameter STATUS_x wird die Fehlernummer angezeigt. Die Anforderung des Client wird - je nach Art des Fehlers - entweder mit einem Exception Telegramm beantwortet oder es wird kein Antworttelegramm zum Client gesendet.

Handling einer fehlerhaften Verbindung

Der Baustein MB_PNHSV erkennt einen Verbindungsfehler, wenn bei einer Übertragung von Datentelegrammen die Kommunikationsfunktionen TSEND/TRCV einen Fehler melden. Nach Anzeige des Fehlercodes wird im Folgenden der Status A0FF angezeigt. Dies bedeutet, dass die Verbindung zwar projektiert, aber derzeit nicht aufgebaut ist.

Wurde ein Fehler auf einer Verbindung erkannt, wird bei gesetztem ENR versucht, die Verbindung wieder aufzubauen.

8.2 Parameter des Funktionsbausteins MB_PNHSV

Parameter	Dekl	Typ	Beschreibung	Wertebereich	Init
id_0_a	IN	WORD	Verbindungs-ID für CPU0 zum Koppelpartner (Knoten A), gemäß der Projektierung im Parameter-DB	1 bis 4095 W#16#1 bis W#16#FFF	ja
id_1_a	IN	WORD	Verbindungs-ID für CPU1 zum Koppelpartner (Knoten A), gemäß der Projektierung im Parameter-DB	1 bis 4095 W#16#1 bis W#16#FFF	ja
id_0_b	IN	WORD	Verbindungs-ID für CPU0 zum Koppelpartner (Knoten B), gemäß der Projektierung im Parameter-DB	1 bis 4095 W#16#1 bis W#16#FFF	ja
id_1_b	IN	WORD	Verbindungs-ID für CPU1 zum Koppelpartner (Knoten B), gemäß der Projektierung im Parameter-DB	1 bis 4095 W#16#1 bis W#16#FFF	ja
db_param	IN	BLOCK_DB	Parameter-DB, enthält alle Verbindungs- und Modbusdaten für diese Modbusbaustein-Instanz	CPU-abhängig	ja
RECV_TIMEOUT	IN	TIME	Überwachungszeit für Datenempfang, mind. 20ms	T#20ms bis T#+24d20h31m23s	nein
CONN_TIMEOUT	IN	TIME	Überwachungszeit für den Verbindungsauf- und –abbau, mind. 100ms	T#100ms bis T#+24d20h31m23s	nein
DISCON-NECT	IN	BOOL	TRUE: Verbindungsabbau bei ENR = FALSE	TRUE/FALSE	nein
REG_KEY_DB	IN	BLOCK_DB	Datenbaustein mit dem Registrierungsschlüssel für die Lizenzierung	CPU-abhängig	nein
Init	IN	BOOL	Manuelle Initialisierung bei positiver Flanke	TRUE/FALSE	nein
ENR	IN	BOOL	Empfangsbereit bei positivem Pegel	TRUE/FALSE	nein
LICENSED	OUT	BOOL	Lizenzzustand des Bausteins Baustein ist lizenziert Baustein ist nicht lizenziert	 TRUE FALSE	nein
BUSY	OUT	BOOL	Bearbeitungszustand von TSEND/TRCV in Bearbeitung nicht in Bearbeitung	 TRUE FALSE	nein
NDR_0A	OUT	BOOL	TRUE: auf Verbindung 0A wurde die Anforderung des Clients ausgeführt und beantwortet	TRUE/FALSE	nein
ERROR_0A	OUT	BOOL	TRUE: Auf Verbindung 0A ist ein Fehler aufgetreten.	TRUE/FALSE	nein
STATUS_0A	OUT	WORD	Status für Verbindung 0A	0 bis FFFF	nein
NDR_1A	OUT	BOOL	TRUE: auf Verbindung 1A wurde die Anforderung des Clients ausgeführt und beantwortet	TRUE/FALSE	nein

Parameter	Dekl	Typ	Beschreibung	Wertebereich	Init
ERROR_1A	OUT	BOOL	TRUE: Auf Verbindung 1A ist ein Fehler aufgetreten.	TRUE/FALSE	nein
STATUS_1A	OUT	WORD	Status für Verbindung 1A	0 bis FFFF	nein
NDR_0B	OUT	BOOL	TRUE: auf Verbindung 0B wurde die Anforderung des Clients ausgeführt und beantwortet	TRUE/FALSE	nein
ERROR_0B	OUT	BOOL	TRUE: Auf Verbindung 0B ist ein Fehler aufgetreten.	TRUE/FALSE	nein
STATUS_0B	OUT	WORD	Status für Verbindung 0B	0 bis FFFF	nein
NDR_1B	OUT	BOOL	TRUE: auf Verbindung 1B wurde die Anforderung des Clients ausgeführt und beantwortet	TRUE/FALSE	nein
ERROR_1B	OUT	BOOL	TRUE: Auf Verbindung 1B ist ein Fehler aufgetreten.	TRUE/FALSE	nein
STATUS_1B	OUT	WORD	Status für Verbindung 1B	0 bis FFFF	nein
IDENT_ CODE	OUT	STRING [18]	Identifikation für die Lizenzierung. Fordern Sie mit diesem String die Lizenz an.	Character	nein
RedErrS7	OUT	BOOL	TRUE: Redundanzverlust auf der S7-Seite	TRUE/FALSE	nein
RedErrDev	OUT	BOOL	TRUE: Redundanzverlust auf der Seite des Kommunikationspartners	TRUE/FALSE	nein
TotComErr	OUT	BOOL	TRUE: Vollständiger Kommunikationsausfall	TRUE/FALSE	nein
Init_Error	OUT	BOOL	TRUE: Bei der manuellen Initialisierung ist ein Fehler aufgetreten.	TRUE/FALSE	nein
Init_Status	OUT	WORD	Status der manuellen Initialisierung	0 bis FFFF	nein

Allgemeines

Die Parameter des FB MB_PNHSV gliedern sich in zwei Gruppen:

- Initialisierungsparameter (klein geschrieben)
- Laufzeitparameter (groß geschrieben)

Die **Initialisierungsparameter** werden nur beim Aufruf im OB100 ausgewertet und in den Instanz-DB übernommen. Die Initialisierungsparameter sind in der obigen Tabelle in der Spalte „INIT“ mit „ja“ gekennzeichnet.
Eine Änderung der Initialisierungsparameter während des laufenden Betriebs hat keine Auswirkung. Nach einer Änderung dieser Parameter z.B. im Testbetrieb muss der Instanz-DB (I-DB) durch STOP/RUN der CPU neu initialisiert werden. Die Initialisierung kann auch mit Hilfe des Parameters „Init“ durchgeführt werden.

Laufzeitparameter können im zyklischen Betrieb verändert werden. Es ist nicht sinnvoll die Eingangsparameter zu ändern während ein Auftrag läuft.

Die Ausgangsparameter sind **dynamische Anzeigen** und stehen somit nur **1 CPU-Zyklus** an. Sie müssen für eine eventuelle Weiterverarbeitung oder eine Anzeige in der Variablentabelle in andere Speicherbereiche kopiert werden.

Wertebereiche

Bei den Wertebereichen für die verschiedenen Parameter sind ggf. auch CPU-spezifische Einschränkungen zu beachten.

id_0_a, id_1_a
id_0_b, id_1_b

Für jede Verbindung von der PN-CPU zu einem Kommunikationspartner wird eine Verbindungs-ID benötigt. Für jede logische Verbindung muss eine andere Verbindungs-ID verwendet werden. Diese Verbindungs-ID wird im Verbindungsparameterblock projektiert, welcher im Parameterdatenbaustein MODBUS_HPARAM_PN enthalten ist. Die Verbindungs-ID beschreibt eindeutig die Verbindung von der CPU zum Koppelpartner und kann Werte von 1 bis 4095 annehmen.
Die Verbindungs-ID aus dem Verbindungsparameterblock ist hier einzutragen und muss CPU-weit eindeutig sein.

id_0_a bezeichnet die Verbindung von CPU0 zum Koppelpartner/Knoten A
id_1_a bezeichnet die Verbindung von CPU1 zum Koppelpartner/Knoten A
id_0_b bezeichnet die Verbindung von CPU0 zum Koppelpartner/Knoten B
id_1_b bezeichnet die Verbindung von CPU1 zum Koppelpartner/Knoten B

Die Verbindung 0A ist die Default-Verbindung und muss zwingend projektiert werden.

Ist der Koppelpartner standalone aufgebaut, dann werden nur die Parameter id_0_a und id_1_a benötigt. Ist die S7 standalone aufgebaut, dann werden nur die Parameter id_0_a und id_0_b benötigt. Sind beide Kommunikationspartner redundant aufgebaut, werden alle 4 Verbindungen projektiert.

db_param

Der Parameter db_param bezeichnet die Nummer des Datenbausteins MODBUS_HPARAM_PN. In diesem Parameterdatenbaustein sind die verbindungs- und modbusspezifischen Parameter hinterlegt, die für die Kommunikation zwischen der PN-CPU und dem Koppelpartner notwendig sind.

Der Wertebereich für diesen Parameter ist CPU-abhängig. Die DB-Nummer 0 ist nicht zulässig, da diese für das System reserviert ist.
Die Eingabe der DB-Nummer erfolgt im Klartext in der Form „DBxy“.

Für jede Modbusbaustein-Instanz ist ein eigener Parameter-Datenbaustein notwendig.

RECV_TIMEOUT

Die Überwachungszeit RECV_TIMEOUT überwacht den Empfang der Daten vom Koppelpartner. Der Minimalwert beträgt 20ms. Es wird eine Überwachungszeit von ca. 1,5 Sekunden empfohlen.

Wenn die RECV_TIMEOUT auf < 20 ms gesetzt wird, wird der Defaultwert von 1,2 s verwendet. Bei Überschreitung der Überwachungszeit wird ein Fehler gemeldet. Die RECV_TIMEOUT überwacht die Laufzeit des Anforderungstelegramms. Die Pause zwischen einzelnen Request vom Client wird dabei nicht berücksichtigt.

CONN_TIMEOUT

Die CONN_TIMEOUT gibt die Zeit für die Überwachung des Verbindungsaufbaus bzw. -abbaus an. Der minimale Wert ist 100ms.

Konnte innerhalb der parametrierten Überwachungszeit die Verbindung nicht erfolgreich auf- bzw. abgebaut werden, erscheint eine entsprechende Fehlermeldung am Ausgang STATUS_x.

Wenn die CONN_TIMEOUT auf < 100 ms gesetzt wurde, wird der Defaultwert von 5 s verwendet.

DISCONNECT

Mit DISCONNECT = TRUE wird die Verbindung abgebaut, wenn der Parameter ENR auf FALSE gesetzt wird. Solange der Parameter TRUE ist, werden die Verbindungen nicht aufgebaut.

Dieser Parameter ist ein Laufzeitparameter und kann entsprechend der Anforderung beliebig gesetzt bzw. zurückgesetzt werden.

REG_KEY_DB

Der Baustein muss auf jedem H-System lizenziert werden. Mit der korrekten Eingabe des Freischaltcodes wird der Baustein lizenziert und die Modbus-Kommunikation kann ohne Einschränkungen genutzt werden. Es wird die Datenbausteinnummer angegeben, der den Freischaltcode enthält. Es ist möglich, mehrere Freischaltcodes untereinander in den DB einzutragen. Der Modbusbaustein durchsucht den DB nach dem passenden Freischaltcode.

Weitere Informationen entnehmen Sie bitte dem **Kapitel „Lizenzierung“**.

Init

Mit dem Parameter Init = TRUE ist eine manuelle Initialisierung des Modbusbausteins möglich. Die Initialisierung kann nur durchgeführt werden, wenn aktuell kein Auftrag läuft. Der Client darf zu dieser Zeit keinen Request schicken.
Bei der manuellen Initialisierung ist zu beachten, dass die Initialisierungsparameter im zyklischen OB parametriert werden müssen.

Warnung

Bei einer manuellen Initialisierung werden die projektierten Verbindungen abgebaut. Bei Änderung der id-Parameter, müssen die Verbindungen vor der manuellen Initialisierung mit DISCONNECT = TRUE und ENR = FALSE manuell abgebaut werden.

ENR

Mit einem positiven Pegel am Eingang wird der FB aktiviert. Es können Telegramme vom Client empfangen werden.
Falls die Verbindung bei gesetztem ENR nicht aufgebaut ist (ESTAB_x = FALSE), wird der Verbindungsaufbau aktiviert.
Wechselt während des laufenden Betriebs ENR von TRUE auf FALSE wird abhängig von der Einstellung am Parameter DISCONNECT ggf. die Verbindung abgebaut.
Bei einem nicht gesetzten Eingang ENR und einer bestehenden Verbindung werden die empfangenen Daten verworfen.

Es wird immer auf allen projektierten Verbindungen gehört und ankommende Anfragen beantwortet.

LICENSED

Ist dieser Ausgang auf TRUE gesetzt, ist der Modbusbaustein auf dieser CPU lizenziert. Hat der Ausgang den Zustand FALSE, wurde kein oder ein fehlerhafter Lizenzstring eingetragen. Weitere Informationen entnehmen Sie bitte dem **Kapitel „Lizenzierung“**.

BUSY

Wenn dieser Ausgang gesetzt ist, wird gerade ein Modbustelegramm bearbeitet.

ESTAB_0A, ESTAB_1A, ESTAB_0B, ESTAB_1B

Mit ESTAB_x = TRUE wird angezeigt, dass eine Verbindung zum Koppelpartner besteht und Daten übertragen werden können.

Ist ESTAB_x = FALSE besteht keine Verbindung zum Koppelpartner.

NDR_0A, NDR_1A, NDR_0B, NDR_1B	Der Ausgang zeigt einen fehlerfrei beendeten Telegrammverkehr mit dem Client auf dieser Verbindung an.
ERROR_0A, ERROR_1A, ERROR_0B, ERROR_1B	Wenn dieser Ausgang gesetzt ist, wurde bei einem Anforderungstelegramm des Client oder beim Senden des Antworttelegramms auf dieser Verbindung ein Fehler erkannt. Die zugehörige Fehlernummer wird im Ausgang STATUS_x angezeigt.
STATUS_0A, STATUS_1A, STATUS_0B, STATUS_1B	Die Ausgänge STATUS_x zeigen bei gesetztem ERROR_x die Fehlernummer, bei nicht gesetztem ERROR_x Statusinformationen für die entsprechende Verbindung an. Die Fehlernummern und Statusinformationen sind im Kapitel „Diagnose“ beschrieben.
IDENT_CODE	Nach dem Anlauf der CPU wird an diesem Parameter eine 18stellige Identifikationskennung angezeigt, mit dem der REG_KEY (Freischaltcode) für die Modbus-Kommunikation beantragt wird. Weitere Informationen entnehmen Sie bitte dem **Kapitel „Lizenzierung“**.
RedErrS7	Der Ausgang RedErrS7 = TRUE zeigt einen Redundanzfehler auf der SIMATIC-Seite an. Bei einseitiger Redundanz bedeutet dies, dass die Verbindung von CPU0 oder CPU1 ausgefallen ist. Bei beidseitiger Redundanz sind beide Verbindungen von CPU0 oder beide Verbindungen von CPU1 ausgefallen. Weitere Informationen entnehmen Sie bitte dem Kapitel „Diagnosemeldungen über Alarm-Bits“.
RedErrDev	Der Ausgang RedErrDev = TRUE zeigt einen Redundanzfehler auf der Seite des Koppelpartners an. Bei einseitiger Redundanz bedeutet dies, dass die Verbindung vom Knotenpunkt A zum CPU0 oder CPU1 ausgefallen ist. Bei beidseitiger Redundanz sind beide Verbindungen zum Knoten A oder beide Verbindungen zum Knoten B des Koppelpartners ausgefallen. Weitere Informationen entnehmen Sie bitte dem Kapitel „Diagnosemeldungen über Alarm-Bits“.
TotComErr	Der Ausgang TotComErr zeigt mit TRUE einen vollständigen Kommunikationsverlust an, d.h. alle projektierten Verbindungen sind gestört. Weitere Informationen entnehmen Sie bitte dem Kapitel „Diagnosemeldungen über Alarm-Bits“.
Init_Error	Trat bei der manuellen Initialisierung ein Fehler auf, wird dies mit Init_Error = TRUE angezeigt.
Init_Status	Der Ausgang Init_Status zeigt bei gesetztem Init_Error die Fehlernummer an. Die Fehlernummern sind im Kapitel „Diagnose“ beschrieben.

8.3 Beispiel für die Adressabbildung

Interpretation der Modbus Adressen

Das MODBUS Datenmodell basiert auf einer Reihe von Speicherbereichen die unterschiedliche Charakteristiken haben. Die Unterscheidung dieser Speicherbereiche erfolgt bei einigen Systemen, z.B. MODICON PLCs über die Register- bzw. Bitadresse. So wird z.B. das Holding Register mit Offset 0 als Register 40001 bezeichnet (Speichertyp 4xxxx, Reference 0001).

Es führt immer wieder zur Verwirrung, weil in manchen Handbüchern die Registeradresse des Application Layers und in anderen die tatsächlich im Protokoll übertragene Register-/Bitadresse beschrieben und gemeint ist.

Der FB MODBUS verwendet bei seinen Parametern *start_x*, *end_x* und START_ADDRESS die **tatsächlich übertragene Modbusadresse**. Es können also mit jedem Funktionscode Register-/Bitadressen von 0000_H bis $FFFF_H$ übertragen werden.

Beispiel: Parametrierung der Datenbereiche

data_type_1 *db_1* *start_1* *end_1*	B#16#3 W#16#B W#16#0 W#16#1F3	Holding Register DB 11 Anfangsadresse: 0 Endadresse: 499
data_type_2 *db _2* *start _2* *end _2*	B#16#3 W#16#C W#16#2D0 W#16#384	Holding Register DB 12 Anfangsadresse: 720 Endadresse: 900
data_type_3 *db _3* *start _3* *end _3*	B#16#4 W#16#D W#16#2D0 W#16#3E8	Input Register DB 13 Anfangsadresse: 720 Endadresse: 1000
data_type_4 *db _4* *start _4* *end _4*	B#16#0 0 0 0	Nicht verwendet 0 0 0
data_type_5 *db _5* *start _5* *end _5*	B#16#1 W#16#E W#16#280 W#16#4E2	Coils DB 14 Anfangsadresse: 640 Endadresse: 1250
data_type_6 *db _6* *start _6* *end _6*	B#16#2 W#16#F W#16#6A4 W#16#8FC	Inputs DB 15 Anfangsadresse:1700 Endadresse: 2300
data_type_7 *db _7* *start _7* *end _7*	B#16#1 W#16#10 W#16#6A4 W#16#8FC	Coils DB 16 Anfangsadresse: 1700 Endadresse: 2300
data_type_8 *db _8* *start _8* *end _8*	B#16#0 0 0 0	Nicht verwendet 0 0 0

Für dieses Beispiel hat:

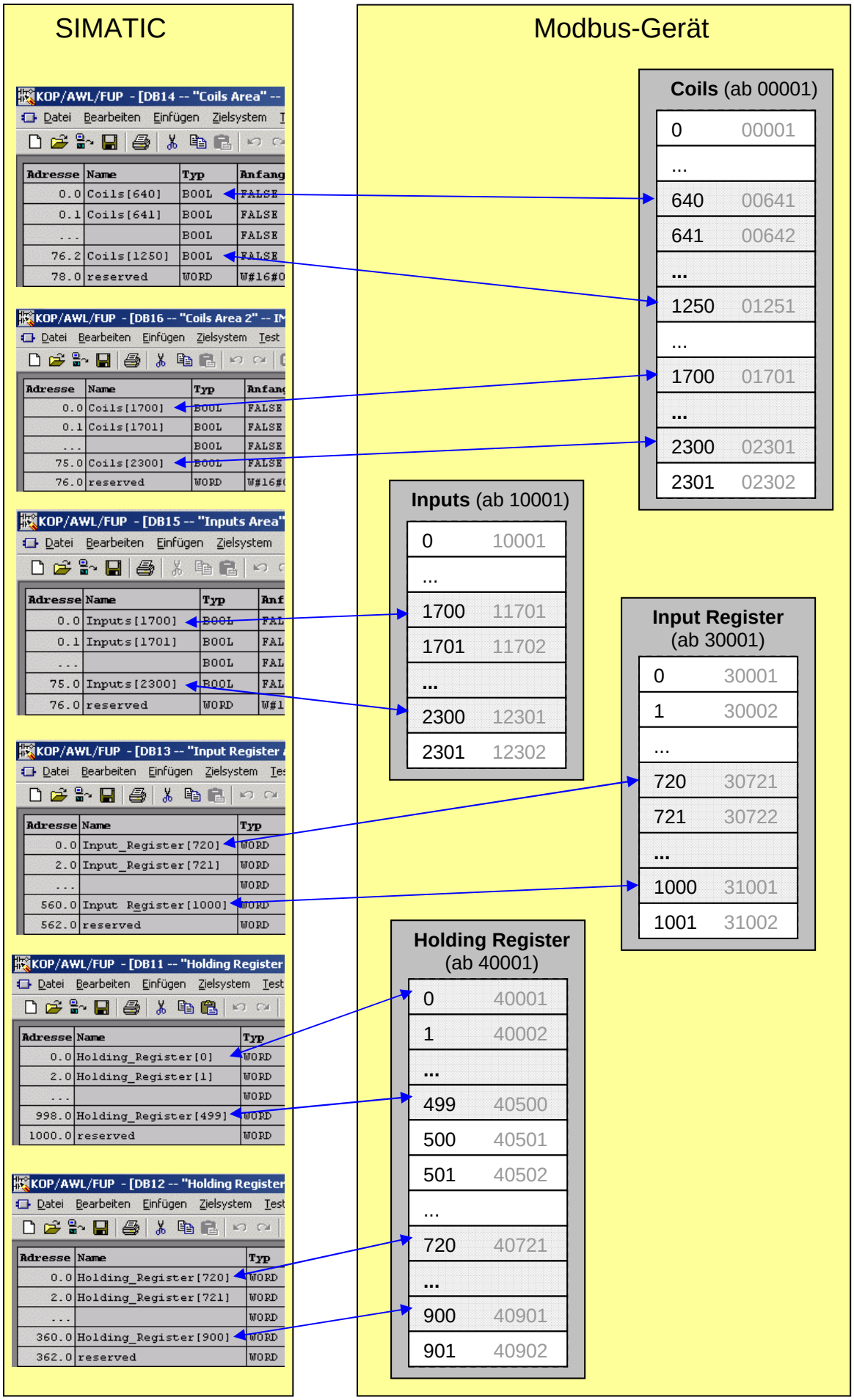
- DB11 eine Größe von 1002 Byte, es werden insgesamt 500 Register abgebildet (Register 0 – Register 499) + 2 reservierte Byte.
- DB12 eine Größe von 364 Byte, es werden insgesamt 181 Register abgebildet (Register 720 – Register 900) + 2 reservierte Byte
- DB13 eine Größe von 564 Byte, es werden insgesamt 281 Input Register abgebildet (Register 720 – Register 1000) + 2 reservierte Byte
- DB14 eine Größe von 80 Byte, es werden insgesamt 611 Coils (Bits) abgebildet (Coil 640 – Coil 1250) + 2 reservierte Byte
- DB15 eine Größe von 78 Byte, es werden insgesamt 601 Inputs (Bits) abgebildet (Input 1700 – Input 2300) + 2 reservierte Byte
- DB16 eine Größe von 78 Byte, es werden insgesamt 601 Coils (Bits) abgebildet (Coil 1700 – Coil 2300) + 2 reservierte Byte

Adressabbildung

Im folgenden Bild sehen Sie die Gegenüberstellung der Simatic-Speicherbereiche mit der register- und bitorientierten Speicheraufteilung der Modbus-Geräte. Dabei wird auf obige Parametrierung Bezug genommen.

Im Modbus-Gerät: Die Modbus-Adressen, die schwarz dargestellt sind, betreffen den Data Link Layer, die grau dargestellten den Applikation Layer.

In der SIMATIC: Die in der 1. Spalte dargestellten SIMATIC-Adressen sind der Offset im DB. In den eckigen Klammern sind die Modbus Registernummern eingetragen.


SIMATIC
Modbus-Gerät
KOP/AWL/FUP - [DB14 -- "Coils Area" --
Datei Bearbeiten Einfügen Zielsystem
Adresse Name Typ Anfang
0.0 Coils[640] BOOL FALSE
0.1 Coils[641] BOOL FALSE
... BOOL FALSE
76.2 Coils[1250] BOOL FALSE
78.0 reserved WORD W#16#0
KOP/AWL/FUP - [DB16 -- "Coils Area 2" -- IM
Datei Bearbeiten Einfügen Zielsystem Test
Adresse Name Typ Anfang
0.0 Coils[1700] BOOL FALSE
0.1 Coils[1701] BOOL FALSE
... BOOL FALSE
75.0 Coils[2300] BOOL FALSE
76.0 reserved WORD W#16#
KOP/AWL/FUP - [DB15 -- "Inputs Area"
Datei Bearbeiten Einfügen Zielsystem
Adresse Name Typ Anf
0.0 Inputs[1700] BOOL FAL
0.1 Inputs[1701] BOOL FAL
... BOOL FAL
75.0 Inputs[2300] BOOL FAL
76.0 reserved WORD W#1
KOP/AWL/FUP - [DB13 -- "Input Register
Datei Bearbeiten Einfügen Zielsystem Tes
Adresse Name Typ
0.0 Input_Register[720] WORD
2.0 Input_Register[721] WORD
... WORD
560.0 Input Register[1000] WORD
562.0 reserved WORD
KOP/AWL/FUP - [DB11 -- "Holding Register
Datei Bearbeiten Einfügen Zielsystem Test
Adresse Name Typ
0.0 Holding_Register[0] WORD
2.0 Holding_Register[1] WORD
... WORD
998.0 Holding_Register[499] WORD
1000.0 reserved WORD
KOP/AWL/FUP - [DB12 -- "Holding Register
Datei Bearbeiten Einfügen Zielsystem Test
Adresse Name Typ
0.0 Holding_Register[720] WORD
2.0 Holding_Register[721] WORD
... WORD
360.0 Holding_Register[900] WORD
362.0 reserved WORD
Coils (ab 00001)
0 00001
...
640 00641
641 00642
...
1250 01251
...
1700 01701
...
2300 02301
2301 02302
Inputs (ab 10001)
0 10001
...
1700 11701
1701 11702
...
2300 12301
2301 12302
Input Register (ab 30001)
0 30001
1 30002
...
720 30721
721 30722
...
1000 31001
1001 31002
Holding Register (ab 40001)
0 40001
1 40002
...
499 40500
500 40501
501 40502
...
720 40721
...
900 40901
901 40902

8.4 Vom FB verwendete Daten und Standardfunktionen

Instanz DB

Der Funktionsbaustein MB_PNHSV speichert die Daten in einem Instanz-DB. Dieser Instanz-DB wird beim ersten Aufruf des FB durch STEP 7 generiert.

Der Instanz-Datenbaustein enthält Parameterwerte vom Typ Input, Output sowie statische Variablen die er für seinen Ablauf benötigt. Diese Variablen sind remanent und behalten zwischen den FB-Aufrufen ihre Gültigkeit. Über die Variablen wird der interne Ablauf des FBs gesteuert.

Speicherbedarf des Instanz-DBs:

Instanz-DB	Arbeitsspeicher	Ladespeicher
MB_PNHSV	ca. 3 kByte	ca. 5 kByte

Lokale Variablen

In Summe werden maximal 186 Byte Lokaldaten für einen FB MB_PNHSV-Aufruf benötigt.

Parameter-DB

Die verbindungs- und modbusspezifischen Parameter werden in dem Parameter-DB MODBUS_HPARAM_PN gespeichert.

Zeiten

Der Funktionsbaustein verwendet keine Timer.

Merker

Der Funktionsbaustein verwendet keine Merker.

Standard-FBs zur Verbindungs-bearbeitung

Der im FB MB_PNHSV/MOD_SERV aufgerufene FB TCP_COMM verwendet die Bausteine TCON und TDISCON aus der Standard-Bibliothek für den Auf- und Abbau der Verbindungen zwischen der CPU und dem Kommunikationspartner.

Standard-FBs zur Datenübertragung

Der im FB MB_PNHSV/MOD_SERV aufgerufene FB TCP_COMM verwendet die Bausteine TSEND und TRCV aus der Standard-Bibliothek für die Datenübertragung zwischen der CPU und dem Kommunikationspartner.

MB_PNHSV: SFCs für sonstige Funktionen

Der FB MB_PNHSV verwendet folgende SFCs aus der Standardbibliothek:

- SFC6 „RD_SINFO“
- SFC20 „BLKMOV“
- SFC24 “TEST_DB”
- SFC51 “RDSYSST”
- SFC52 “WR_USMSG”

MOD_SERV: SFCs für sonstige Funktionen

Der FB MOD_SERV verwendet folgende SFCs aus der Standardbibliothek:

- SFC20 „BLKMOV“
- SFC24 “TEST_DB”

TCP_COMM: SFCs für sonstige Funktionen

Der FB TCP_COMM verwendet außer den T-Bausteinen den folgenden SFB aus der Standardbibliothek:

- SFB4 „TON“

8.5 Umbenennen / Umverdrahten von Funktionen und Funktionsbausteinen

Veranlassung

Falls in Ihrem Projekt die Nummern der Standardfunktionen bereits verwendet werden oder der Nummernbereich für andere Applikationen reserviert ist (z.B. in CFC), können Sie die intern aufgerufenen Funktionsbausteine FB63, FB64, FB65 und FB66 des FBs TCP_COMM oder die Bausteine MB_PNHSV, MOD_SERV und TCP_COMM umverdrahten.

Die Systemfunktionen SFC6, SFC20, SFC24, SFC51 und SFC52 sowie der Systemfunktionsbaustein SFB4 können nicht umbenannt/umverdrahtet werden.

Verhalten

Im SIMATIC Manager von STEP7 sind beim Umverdrahten einige Regeln bezüglich der Bausteinnummern zu beachten.

Wenn Sie die Bausteine aus der Modbusbibliothek umverdrahten wollen, gehen Sie in folgender Reihenfolge vor:

1. FB917 MB_PNHSV
2. FB916 MOD_SERV
3. FB913 TCP_COMM
4. FB63 TSEND
   FB64 TRCV
   FB65 TCON
   FB66 TDISCON

Es müssen nicht alle Funktionen bzw. Funktionsbausteine umverdrahtet werden. Auch wenn Sie nur einige von diesen umverdrahten wollen, muss diese Reihenfolge eingehalten werden.

Umverdrahten

Gehen Sie zum Verdrahten für die FBs folgendermaßen vor:

1. Holen Sie mittels „Extras > Referenzdaten > Anzeigen“ Informationen über die verwendeten Operanden ein.
2. Stellen Sie den Operandenvorrang in den Objekteigenschaften des Bausteinsordners auf „Absolutwert“.
3. Selektieren Sie im SIMATIC Manager den Baustein-Ordner und rufen Sie die Funktion „Extras > Umverdrahten“ auf, um die Operanden in freie Bereiche umzuverdrahten.
4. Um die Symbolik in Diagnosetools weiter nutzen zu können, ziehen Sie die Änderungen in der Symboltabelle nach.

Über „Extras > Referenzdaten > Anzeigen“ können Sie die Änderungen überprüfen.

9 Additional Blocks

9.1 Unterstützung in CFC

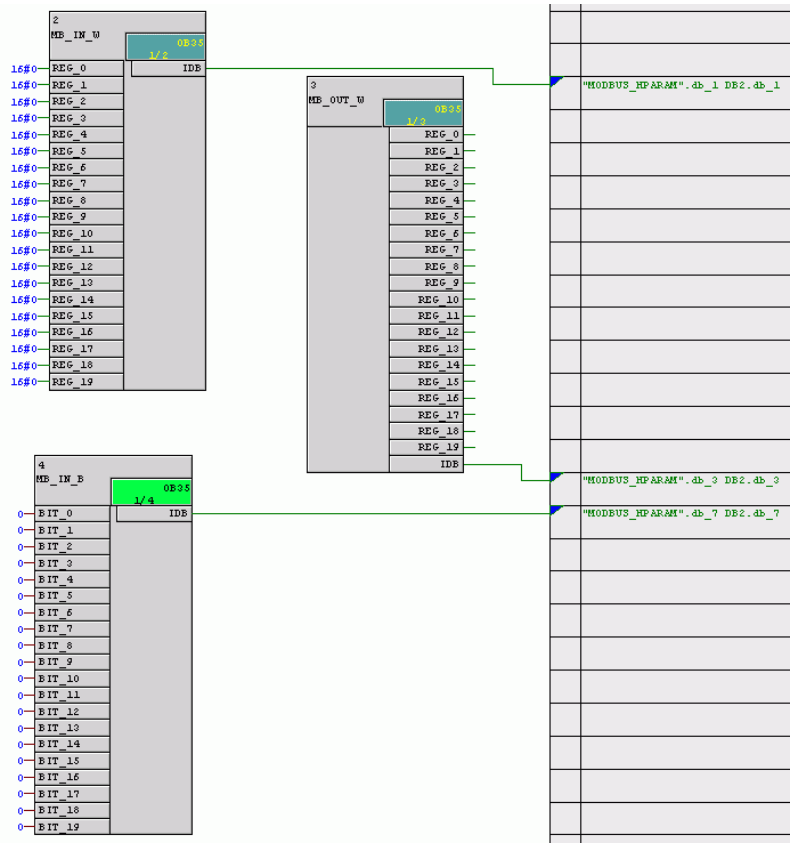
Allgemein

Um die Projektierung in CFC zu unterstützen, gibt es die Möglichkeit, die Modbuswerte nicht über Global-DBs sondern über „DataCollector -FBs“ zu projektieren. Dabei werden die Sende- und Empfangspuffer für die Werte per Drag & Drop in den CFC-Plan gezogen.

Verwendung - Beispiel

Die DataCollector -FBs werden im CFC-Plan platziert. Der Ausgang „IDB“ wird mit den DB-Parametern db_1 bis db_8 im Parameter-Datenbaustein verbunden.

Im weiteren Verlauf können die Modbuswerte direkt von den Channel-Bausteinen an den DataCollector-FB verschaltet werden.



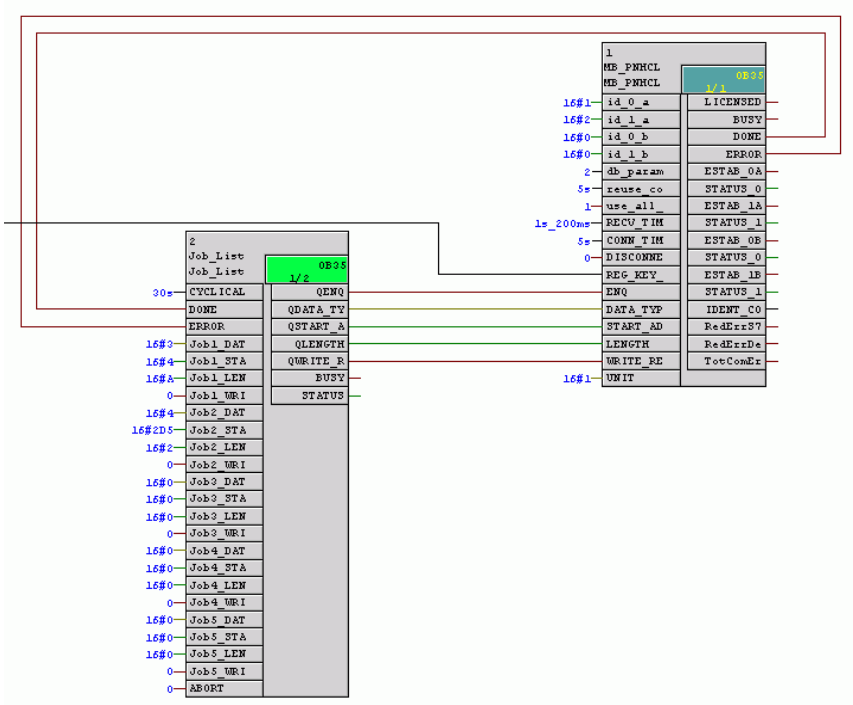
Die zusätzlichen Bausteine sowie eine detaillierte Beschreibung finden Sie hier: www.siemens.de/s7modbus bzw. beim Customer Support.

9.2 Auftragsliste für zyklischen Telegrammverkehr

Allgemein

Mit dem Baustein Job_List ist eine Liste von Aufträgen parametrierbar, die zyklisch abgearbeitet wird.

Verwendung - Beispiel



Den zusätzlichen Baustein sowie eine detaillierte Beschreibung finden Sie hier: www.siemens.de/s7modbus bzw. beim Customer Support.

10 Einsatz in einer Single-PN-CPU

Allgemein

Das Paket „Modbus/TCP PN CPU Redundant“ kann ebenfalls in einer S7-PN-Single-CPU (S7-400, S7-300, ET200S) betrieben werden.

Die Beschreibung der Funktionen und der Parameter in den vorhergehenden und nachfolgenden Kapiteln gelten analog für die Anwendung in einer Single-PN-CPU.

Verwendbare Baugruppen für MB_PNHCL und MB_PNHSV

Es können nur CPUen verwendet werden, die genügend Lokaldaten pro Prioritätsklasse zur Verfügung stellen (=> Kapitel 7.4 und Kapitel 8.4).

Ebenso sind die CPU-abhängigen Grenzwerte, wie z.B. die maximale FB-Nummer, zu beachten.

Modbus-Bausteine für Single-PN-CPUen

Bei Verwendung einer Single-PN-CPU werden die gleichen FBs verwendet wie für die hochverfügbaren PN-CPUen.

Bei der Projektierung ist darauf zu achten, dass die richtige local_device_id verwendet wird. Alle weiteren CPU-spezifischen Parameter müssen ebenfalls geprüft und gegebenenfalls manuell angepasst werden.

.

11 Diagnose

Diagnose-funktionen

Die Diagnosefunktionen der PN-CPU erlauben Ihnen ein schnelles Lokalisieren aufgetretener Fehler. Folgende Diagnosemöglichkeiten stehen zur Verfügung:

- Diagnose über die Anzeigeelemente der CPUs
- Diagnose über die STATUS-Ausgänge der Modbus-Funktionsbausteine.
- Diagnose über die Alarmbits der Modbus-Funktionsbausteine

Anzeigeelemente (LED)

Die Anzeigeelemente informieren Sie über den Betriebszustand bzw. über mögliche Fehlerzustände der CPUs. Die Anzeigeelemente geben Ihnen einen ersten Überblick über aufgetretene interne bzw. externe Fehler sowie schnittstellenspezifische Fehler.

STATUS-Ausgänge des MB_PNHCL und MB_PNHSV

Für eine Fehlerdiagnose besitzen die MB_PNHCL- bzw. MB_PNHSV-Funktionsbausteine STATUS-Ausgänge. Durch Lesen der STATUS-Ausgänge erhalten Sie allgemeine Aussagen zu Fehlern, die bei der Kommunikation aufgetreten sind. Die STATUS-Parameter können sie im Anwenderprogramm auswerten.

Alarmbits von MB_PNHCL und MB_PNHSV

Die Modbus-Funktionsbausteine bieten zusätzlich Ausgänge, die einen Redundanz- bzw. einen kompletten Kommunikationsverlust anzeigen. Die Alarmbits werden abhängig vom Zustand der ESTAB_x-Ausgänge der projektierten Verbindungen gesetzt.

11.1 Diagnose über die Anzeigeelemente der CPU

Anzeigefunktionen

Über die Anzeigeelemente der CPU erhalten Sie Auskunft über den Baugruppenzustand. Zu unterscheiden sind folgende Anzeigefunktionen:

- **Sammelstörungsanzeigen**

  PN-CPU 300 und IM 151-8 PN/DP CPU
  - SF Sammelfehler
  Blinkt diese LED, dann ist der Modbus-Baustein noch nicht lizenziert.
  Weitere Informationen entnehmen Sie dem **Kapitel „Lizenzierung“**.

  PN-(H)CPU 400
  - INTF Sammelfehler
  Blinkt diese LED, dann ist der Modbus-Baustein noch nicht lizenziert.
  Weitere Informationen entnehmen Sie dem **Kapitel „Lizenzierung“**.

- **Sonderanzeigen**

  PN-CPU 300, PN-(H)CPU 400 und IM 151-8 PN/DP CPU:
  - RX/TX ein Telegramm wird über die Schnittstelle übertragen

Eine detaillierte Beschreibung der Anzeigeelemente finden Sie im jeweiligen Gerätehandbuch der CPUs.

11.2 Diagnosemeldungen der FBs MB_PNHCL und MB_PNHSV

Meldungen am STATUS-Ausgang des FBs

An den Status-Ausgängen der FBs MB_PNHCL und MB_PNHSV werden die Fehlermeldungen angezeigt. Nachfolgend finden Sie eine Aufstellung der FB-spezifischen Fehlermeldungen.

Fehlermeldungen der aufgerufenen SFCs und FCs

Die Modbus-FBs benutzen die Standardbausteine SFC6, SFC20, SFC24, SFC51 und SFC52. Die Fehlermeldungen dieser Bausteine werden unverändert an STATUS_x weitergegeben.

Der in MOD_CLI bzw. MOD_SERV aufgerufene FB TCP_COMM benutzt die Standardbausteine SFB4, FB63, FB64, FB65 und FB66. Die Fehlermeldungen dieser Bausteine werden ebenfalls unverändert an STATUS_x weitergegeben.

Im Diagnosepuffer oder der Online-Hilfe zu den SFCs/FCs aus dem SIMATIC Manager finden Sie weitere Hinweise zu diesen Fehlermeldungen.

Fehlermeldungen der FBs MB_PNHCL und MB_PNHSV		
STATUS (Hex)	**Ereignistext**	**Abhilfe**
A001	Der Parameter-DB MODBUS_HPARAM_PN ist zu kurz oder zu lang.	Korrigieren Sie den DB MODBUS_HPARAM_PN.
A002	Der Parameter *end_x* ist kleiner als *start_x*.	Korrigieren Sie die Angaben im DB MODBUS_HPARAM_PN.
A003	Ein DB, auf den MODBUS-Adressen abgebildet werden sollen, ist zu kurz. Minimallänge: - bei Registern: (*end_x* - *start_x* + 1) ∗ 2 + 2 - bei Bitwerten: (*end_x* - *start_x*) / 8 + 1 + 2 weitere mögliche Ursachen: • Bei MB_PNHCL: falsche Aufrufparameter • Bei MB_PNHSV: falscher Adressbereich im Anforderungstelegramm des Clients. Die S7 antwortet mit einem Exceptiontelegramm.	Verlängern Sie den DB. Bei MB_PNHCL: Korrigieren Sie die Auftragsparameter START_ ADDRESS oder LENGTH. Bei MB_PNHSV: Ändern Sie die Anforderung des Client.
A004	Nur bei MB_PNHCL: Es wurde eine unzulässige Kombination von DATA_TYPE und WRITE_READ angegeben.	Korrigieren Sie die Aufrufparameter. Es können nur die Datentypen 1 und 3 geschrieben werden.
A005	Bei MB_PNHCL: Es wurde ein unzulässiger Wert am Parameter LENGTH angegeben. Bei MB_PNHSV: Die Register-/Bitanzahl im Anforderungstelegramm ist unzulässig. Die S7 antwortet mit einem Exceptiontelegramm. Wertebereiche: Coils/Inputs lesen: 1 bis 2000 Coils schreiben: 1 bis 1968 Register lesen: 1 bis 125 Holding Register schreiben: 1 bis 123	Bei MB_PNHCL: Korrigieren Sie den Parameter LENGTH. Bei MB_PNHSV: Verändern Sie im Anforderungstelegramm des Client die Anzahl.
A006	Der über DATA_TYPE, START_ADDRESS und LENGTH angegebene Bereich existiert nicht in *data_type_1* bis *data_type_8*. Bei MB_PNHSV: Die S7 antwortet mit einem Exceptiontelegramm.	Bei MB_PNHCL: Korrigieren Sie die Kombination DATA_TYPE, START_ ADDRESS und LENGTH. Bei MB_PNHSV: Ändern Sie die Anforderung des Client oder korrigieren Sie die Parametrierung an *data_type_x*.
A007	Bei MB_PNHCL: Es wurde eine ungültige Überwachungszeit an RECV_TIMEOUT oder CONN_TIMEOUT parametriert. Für RECV_TIMEOUT muss ein Wert >= 20 ms, für CONN_TIMEOUT >= 100 ms eingetragen werden.	Korrigieren Sie die Parametrierung.
A009	Bei MB_PNHCL: Der empfangene Transaction Identifier TI ist ungleich dem gesendeten. Die Verbindung wird abgebaut.	Überprüfen Sie mit Hilfe einer Telegrammaufzeichnung die Daten des Koppelpartners.
A00A	Bei MB_PNHCL: Die empfangene UNIT ist ungleich der gesendeten.	

Fehlermeldungen der FBs MB_PNHCL und MB_PNHSV		
STATUS (Hex)	**Ereignistext**	**Abhilfe**
A00B	Bei MB_PNHCL: Der empfangene Funktionscode ist ungleich dem gesendeten. Bei MB_PNHSV: Es wurde ein ungültiger Funktionscode empfangen. Der S7 antwortet mit einem Exceptiontelegramm.	Bei MB_PNHCL: Überprüfen Sie mit Hilfe einer Telegrammaufzeichnung die Daten des Koppelpartners. Bei MB_PNHSV: Ändern Sie die Client-Anforderung. Der Modbus-FB bearbeitet die Funktionscodes 1, 2, 3, 4, 5, 6, 15 und 16.
A00C	Der empfangene Bytecount passt nicht zur Registeranzahl. Die Verbindung wird abgebaut.	Überprüfen Sie mit Hilfe einer Telegrammaufzeichnung die Daten des Koppelpartners.
A00D	Bei MB_PNHCL: Die Register-/Bitadresse oder Register-/Bitanzahl im Antworttelegramm ist ungleich der im Anforderungstelegramm.	
A00E	Die Längenangabe im modbusspezifischen Telegrammheader passt nicht zu den Angaben der Register-/Bitanzahl oder des Bytecount im Telegramm. Der FB verwirft die Daten. Die Verbindung wird abgebaut.	
A00F	Es wurde ein Protocol Identifier ungleich 0 empfangen. Die Verbindung wird abgebaut.	Überprüfen Sie mit Hilfe einer Telegrammaufzeichnung die Daten des Koppelpartners.
A010	Bei den Parametern *db_1* bis *db_8* wurde eine DB-Nummer doppelt vergeben.	Korrigieren Sie die Parametrierung im DB MODBUS_HPARAM_PN.
A011	Am Eingangsparameter DATA_TYPE wurde ein unzulässiger Wert angegeben (zulässige Werte: 1 - 4).	Korrigieren Sie die Aufrufparameter.
A012	Die parametrierten Bereiche *data_type_1* und *data_type_2* überlappen.	Korrigieren Sie die Parametrierung. Die Datenbereiche dürfen keine gemeinsamen Register besitzen.
A013	Die parametrierten Bereiche *data_type_1* und *data_type_3* überlappen.	
A014	Die parametrierten Bereiche *data_type_1* und *data_type_4* überlappen.	
A015	Die parametrierten Bereiche *data_type_1* und *data_type_5* überlappen.	
A016	Die parametrierten Bereiche *data_type_1* und *data_type_6* überlappen.	
A017	Die parametrierten Bereiche *data_type_1* und *data_type_7* überlappen.	
A018	Die parametrierten Bereiche *data_type_1* und *data_type_8* überlappen.	
A019	Einer der Parameter *db_x* wurde auf 0 gesetzt, obwohl der zugehörige *data_type_x* mit > 0 parametriert ist. DB0 darf nicht verwendet werden, weil dieser für das System reserviert ist.	Korrigieren Sie die Parametrierung an *db_x* auf > 0.

Fehlermeldungen der FBs MB_PNHCL und MB_PNHSV		
STATUS (Hex)	**Ereignistext**	**Abhilfe**
A01A	Falsche Länge im Header: Es sind 3 bis 253 Byte zulässig. Die Verbindung wird abgebaut.	Überprüfen Sie mit Hilfe einer Telegrammaufzeichnung die Daten des Koppelpartners.
A01B	Bei MB_PNHSV und Funktionscode 5: Es wurde ein ungültiger Zustand für Coil empfangen. Die S7 antwortet mit einem Exceptiontelegramm.	Überprüfen Sie mit Hilfe einer Telegrammaufzeichnung die Daten des Koppelpartners.
A020	An reuse_conn_cycle ist keine oder eine zu kleine Zeit projektiert.	Korrigieren Sie die Parametrierung. Es muss eine Überwachungszeit > 1s parametriert werden.
A022	An den MB_PNHSV wurde ein Parameter-Datenbaustein für Clients parametriert bzw. an den MB_PNHCL wurde ein Parameter-Datenbaustein für Server parametriert.	Korrigieren Sie die Parametrierung des Parameter-Datenbausteins.
A023	Die parametrierten Bereiche data_type_2 und data_type_3 überlappen.	Korrigieren Sie die Parametrierung. Die Datenbereiche dürfen keine gemeinsamen Register besitzen.
A024	Die parametrierten Bereiche data_type_2 und data_type_4 überlappen.	
A025	Die parametrierten Bereiche *data_type*_2 und *data_type_5* überlappen.	
A026	Die parametrierten Bereiche *data_type*_2 und *data_type_6* überlappen.	
A027	Die parametrierten Bereiche *data_type*_2 und *data_type*_7 überlappen.	
A028	Die parametrierten Bereiche *data_type*_2 und *data_type*_8 überlappen.	
A034	Die parametrierten Bereiche *data_type_3* und *data_type*_4 überlappen.	
A035	Die parametrierten Bereiche *data_type_3* und *data_type*_5 überlappen.	
A036	Die parametrierten Bereiche *data_type_3* und *data_type_6* überlappen.	
A037	Die parametrierten Bereiche *data_type_3* und *data_type*_7 überlappen.	
A038	Die parametrierten Bereiche *data_type_3* und *data_type*_8 überlappen.	
A045	Die parametrierten Bereiche *data_type_4* und *data_type*_5 überlappen.	
A046	Die parametrierten Bereiche *data_type_4* und *data_type_6* überlappen.	
A047	Die parametrierten Bereiche *data_type_4* und *data_type*_7 überlappen.	
A048	Die parametrierten Bereiche *data_type_4* und *data_type*_8 überlappen.	
A056	Die parametrierten Bereiche *data_type_5* und *data_type_6* überlappen.	

Fehlermeldungen der FBs MB_PNHCL und MB_PNHSV		
STATUS (Hex)	**Ereignistext**	**Abhilfe**
A057	Die parametrierten Bereiche *data_type_5* und *data_type_7* überlappen.	
A058	Die parametrierten Bereiche *data_type_5* und *data_type_8* überlappen.	
A067	Die parametrierten Bereiche *data_type_6* und *data_type_7* überlappen.	
A068	Die parametrierten Bereiche *data_type_6* und *data_type_8* überlappen.	
A078	Die parametrierten Bereiche *data_type_7* und *data_type_8* überlappen.	
A079	Die am Parameter id_x angegebene Verbindungs-ID ist im Parameter-DB MODBUS_HPARAM_PN nicht enthalten.	Korrigieren Sie die Parametrierung am Eingang id_x.
A07A	An Parameter id_x des Bausteins wurde ein unzulässiger Wert (Wertebereich von 1 bis 4095) oder ein Wert doppelt vergeben.	
A07B	Die angegebene id_x ist im Parameter-DB doppelt enthalten.	Korrigieren Sie die Parametrierung im DB MODBUS_HPARAM_PN.
A07C	Im Parameter-DB wurde ein unzulässiger Wert am Parameter *data_type_x* angegeben (zulässige Werte sind 0 bis 4).	
A07D	Im Parameter-DB enthält der Parameter *data_type_1* keinen Eintrag. Der Parameterbereich „*_1*“ ist der Initialbereich und muss parametriert werden.	
A07E	An *db_x* wurde die Nummer des Instanz-DBs des Bausteins MB_PNHCL bzw. MB_PNHSV angegeben.	
A07F	Der an db_param angegebene DB ist kein Modbus-Parameter-DB.	Korrigieren Sie die Parametrierung am Eingang db_param.
A080	Für den Aufruf des Modbusbausteins wurden im OB100 und im zyklischen OB unterschiedliche Instanz-DBs verwendet oder der Instanz-DB wurde ohne Neustart auf die CPU geladen.	Der Modbus-Baustein muss im Anlauf-OB und im zyklischen OB mit demselben Instanz-DB aufgerufen werden. Nach dem Transferieren des IDBs in die CPU muss der Modbusbaustein initialisiert werden.
A081	Nur bei MB_PNHCL und Funktionscode 5: Die Daten des Antworttelegramms sind nicht das Echo der Anforderung.	Überprüfen Sie mit Hilfe einer Telegrammaufzeichnung die Daten des Koppelpartners.
A082	Nur bei MB_PNHCL und Funktionscode 6: Der empfangene Registerwert ist ungleich dem gesendeten.	
A083	Bei MB_PNHCL: Es ist ein Auftrag angestoßen worden, während der vorherige Auftrag noch läuft. Der Auftrag wird nicht ausgeführt. Dies ist eine Statusinformation. Das Bit ERROR ist nicht gesetzt. Es wurde versucht den Baustein manuell zu initialisieren während noch ein Auftrag läuft.	Stoßen Sie erst dann einen neuen Auftrag an, wenn der vorherige Auftrag mit DONE = TRUE oder ERROR = TRUE beendet wurde. Warten Sie mit der Initialisierung bis kein Auftrag mehr läuft.

Fehlermeldungen der FBs MB_PNHCL und MB_PNHSV		
STATUS (Hex)	**Ereignistext**	**Abhilfe**
A084	Es konnte keine Identifizierungskennung IDENT_CODE für die Lizenzierung ermittelt werden.	Wenden Sie sich an den Produkt Support.
A085	Es trat ein Fehler bei der Lizenzermittlung auf. Der Fehler wird beim 1. Auftreten mit ERROR = TRUE angezeigt. Im Folgenden wird es als Statusmeldung mit ERROR = FALSE angezeigt.	Prüfen Sie, dass keine unerlaubten Schreib-Zugriffe auf den Lizenz-DB vorhanden sind. Die Struktur des REG_KEYs darf nicht verändert werden. Wenden Sie sich gegebenenfalls an den Produkt Support.
A086	Es wurde versucht in einen schreibgeschützten Datenbaustein zu schreiben.	Entfernen Sie den Schreibschutz des Datenbausteins oder verwenden Sie einen anderen DB.
A090	Der Modbus-Baustein ist für diese CPU noch nicht lizenziert. Dies ist eine Statusinformation. Das Bit ERROR ist nicht gesetzt. Die Modbus-Kommunikation läuft auch ohne Lizenz.	Lesen Sie den Identifikationsstring IDENT_CODE für diese CPU aus und fordern Sie den Registrierungsschlüssel an.
A091	Nur bei MB_PNHCL: Als Antwort wurde ein Exceptiontelegramm mit Exception Code 1 empfangen.	Der Koppelpartner unterstützt die angeforderte Funktion nicht.
A092	Nur bei MB_PNHCL: Als Antwort wurde ein Exceptiontelegramm mit Exception Code 2 empfangen. Es erfolgte ein Zugriff auf eine nicht vorhandene/nicht zulässige Adresse beim Koppelpartner.	Korrigieren Sie LENGTH bzw. START_ADDRESS beim FB-Aufruf.
A093	Nur bei MB_PNHCL: Als Antwort wurde ein Exceptiontelegramm mit Exception Code 3 empfangen.	Der Koppelpartner kann das empfangene Telegramm nicht verarbeiten (z.B. er unterstützt die angeforderte Länge nicht).
A094	Nur bei MB_PNHCL: Als Antwort wurde ein Exceptiontelegramm mit Exception Code 4 empfangen.	Der Koppelpartner befindet sich in einem Zustand in dem er das empfangene Telegramm nicht bearbeiten kann.
A095	Nur bei MB_PNHCL: Als Antwort wurde ein Exceptiontelegramm mit einem unbekannten Exception Code empfangen.	Kontrollieren Sie die Fehlermeldungen des Koppelpartners und überprüfen Sie ggf. die Daten mit einer Telegrammaufzeichnung.
A0FF	Die Verbindung ist derzeit nicht aufgebaut. Dies ist eine Statusinformation.	Prüfen Sie die Verbindungen. Korrigieren Sie ggf. den Wert an reuse_conn_time.
A100	Für einen Auftrag ist die Überwachungszeit CONN_TIMEOUT oder RECV_TIMEOUT abgelaufen. Bei Ablauf der RECV_TIMEOUT wird die Verbindung abgebaut.	Prüfen Sie die Parametrierung der Verbindung.
A101	Die interne Überwachungszeit der Funktion TDISCON ist abgelaufen.	Kontaktieren Sie den Produkt Support.
FFFF	Die Verbindung ist nicht projektiert.	Soll diese Verbindung verwendet werden, muss sie im Anlauf an id_x projektiert werden.

11.3 Diagnosemeldungen der eingebundenen Bausteine

Fehlermeldungen der eingebundenen FCs/SFCs		
STATUS (Hex)	**Ereignistext**	**Abhilfe**
7xxx	Bitte entnehmen Sie die detaillierte Information der Online-Hilfe des SIMATIC Manager.	Siehe Online-Hilfe (SIMATIC Manager -> Baustein markieren -> Taste F1)
8xxx	Bitte entnehmen Sie die detaillierte Information der Online-Hilfe des SIMATIC Manager.	Siehe Online-Hilfe (SIMATIC Manager -> Baustein markieren -> Taste F1)
80C4	Das H-System befindet sich im Ankoppeln und Aufdaten.	Diese Fehlermeldung von TCON kann 1 Mal nach dem Neustart des H-Systems auftreten und kann ignoriert werden.

11.4 Diagnosemeldungen des SFC24

Fehlermeldungen des SFC24		
STATUS (Hex)	**Ereignistext**	**Abhilfe**
80A1	DB Nummer = 0 oder zu groß für die CPU	Wählen Sie eine zulässige DB Nummer.
80B1	Der DB existiert nicht auf der CPU.	Alle Datenbausteine, die an DB_x angegeben werden, müssen angelegt und auf die CPU übertragen werden.
80B2	DB UNLINKED	DB nicht als UNLINKED generieren.

11.5 Diagnosemeldungen über Alarm-Bits

Die Modbus-Bausteine bieten die Möglichkeit einen Redundanzverlust zu erkennen. Die Anzeige erfolgt über die Ausgänge RedErrS7, RedErrDev und TotComErr. Diese Status-Bits können jeweils an einen Alarmbaustein oder an andere Bausteine verschaltet und dort entsprechend ausgewertet werden.

Abhängig vom Zustand der projektierten Verbindungen an ESTAB_x werden die Alarmbits entsprechend gesetzt.

11.5.1 Client-Baustein

Abhängig von der Parametrierung werden die Alarmbits folgendermaßen gesetzt:

1) use_all_conn = FALSE

   Die Telegramme werden nur über 1 Verbindung gesendet und empfangen, die anderen projektierten Verbindungen sind auf Standby. Nach Ablauf der parametrierten Zeit „reuse_conn_time“ wird versucht, defekte Verbindungen wieder aufzubauen.

Anzahl der fehlerhaften Verbindungen	STATUS_0A	STATUS_0B	STATUS_1A	STATUS_1B	RedErrS7	RedErrDev	TotComErr
0	0	0	0	0	FALSE	FALSE	FALSE
1	0	0	0	<> 0	FALSE	FALSE	FALSE
	0	0	<> 0	0	FALSE	FALSE	FALSE
	0	<> 0	0	0	FALSE	FALSE	FALSE
	<> 0	0	0	0	FALSE	FALSE	FALSE
2	0	0	<> 0	<> 0	TRUE	FALSE	FALSE
	0	<> 0	0	<> 0	FALSE	TRUE	FALSE
	<> 0	0	0	<> 0	FALSE	FALSE	FALSE
	0	<> 0	<> 0	0	FALSE	FALSE	FALSE
	<> 0	0	<> 0	0	FALSE	TRUE	FALSE
	<> 0	<> 0	0	0	TRUE	FALSE	FALSE
3	<> 0	<> 0	<> 0	0	TRUE	TRUE	FALSE
	<> 0	<> 0	0	<> 0	TRUE	TRUE	FALSE
	<> 0	0	<> 0	<> 0	TRUE	TRUE	FALSE
	0	<> 0	<> 0	<> 0	TRUE	TRUE	FALSE
4	<> 0	<> 0	<> 0	<> 0	TRUE	TRUE	TRUE

2) use_all_conn = TRUE; S7 ist redundant, der Koppelpartner ist standalone aufgebaut

Die Telegramme werden über 2 projektierte Verbindungen gesendet und empfangen. Nach Ablauf der parametrierten Zeit „reuse_conn_time“ wird versucht, defekte Verbindungen wieder aufzubauen.

Anzahl der fehlerhaften Verbindungen	STATUS_0A	STATUS_0B	STATUS_1A	STATUS_1B	RedErrS7	RedErrDev	TotComErr
0	0	FFFF	0	FFFF	FALSE	FALSE	FALSE
1	0	FFFF	<> 0	FFFF	TRUE	TRUE	FALSE
	<> 0	FFFF	0	FFFF	TRUE	TRUE	FALSE
2	<> 0	FFFF	<> 0	FFFF	TRUE	TRUE	TRUE

3) use_all_conn = TRUE; S7 ist standalone, der Koppelpartner ist redundant aufgebaut

Die Telegramme werden über 2 projektierte Verbindungen gesendet und empfangen. Nach Ablauf der parametrierten Zeit „reuse_conn_time“ wird versucht, defekte Verbindungen wieder aufzubauen.

Anzahl der fehlerhaften Verbindungen	STATUS_0A	STATUS_0B	STATUS_1A	STATUS_1B	RedErrS7	RedErrDev	TotComErr
0	0	0	FFFF	FFFF	FALSE	FALSE	FALSE
1	0	<> 0	FFFF	FFFF	TRUE	TRUE	FALSE
	<> 0	0	FFFF	FFFF	TRUE	TRUE	FALSE
2	<> 0	<> 0	FFFF	FFFF	TRUE	TRUE	TRUE

4) use_all_conn = TRUE; 4 Verbindungen sind projektiert

Die Telegramme werden über 4 projektierte Verbindungen gesendet und empfangen. Nach Ablauf der parametrierten Zeit „reuse_conn_time“ wird versucht, defekte Verbindungen wieder aufzubauen.

Anzahl der fehlerhaften Verbindungen	STATUS_0A	STATUS_0B	STATUS_1A	STATUS_1B	RedErrS7	RedErrDev	TotComErr
0	0	0	0	0	FALSE	FALSE	FALSE
1	0	0	0	<> 0	FALSE	FALSE	FALSE
	0	0	<> 0	0	FALSE	FALSE	FALSE
	0	<> 0	0	0	FALSE	FALSE	FALSE
	<> 0	0	0	0	FALSE	FALSE	FALSE
2	0	0	<> 0	<> 0	TRUE	FALSE	FALSE
	0	<> 0	0	<> 0	FALSE	TRUE	FALSE
	<> 0	0	0	<> 0	FALSE	FALSE	FALSE
	0	<> 0	<> 0	0	FALSE	FALSE	FALSE
	<> 0	0	<> 0	0	FALSE	TRUE	FALSE
	<> 0	<> 0	0	0	TRUE	FALSE	FALSE
3	<> 0	<> 0	<> 0	0	TRUE	TRUE	FALSE
	<> 0	<> 0	0	<> 0	TRUE	TRUE	FALSE
	<> 0	0	<> 0	<> 0	TRUE	TRUE	FALSE
	0	<> 0	<> 0	<> 0	TRUE	TRUE	FALSE
4	<> 0	<> 0	<> 0	<> 0	TRUE	TRUE	TRUE

11.5.2 Server-Baustein

Der Serverbaustein versucht zyklisch, defekte Verbindungen wieder aufzubauen.

Anzahl der fehlerhaften Verbindungen	STATUS_0A	STATUS_0B	STATUS_1A	STATUS_1B	RedErrS7	RedErrDev	TotComErr
0	0	0	0	0	FALSE	FALSE	FALSE
1	0	0	0	<> 0	FALSE	FALSE	FALSE
	0	0	<> 0	0	FALSE	FALSE	FALSE
	0	<> 0	0	0	FALSE	FALSE	FALSE
	<> 0	0	0	0	FALSE	FALSE	FALSE
2	0	0	<> 0	<> 0	TRUE	FALSE	FALSE
	0	<> 0	0	<> 0	FALSE	TRUE	FALSE
	<> 0	0	0	<> 0	FALSE	FALSE	FALSE
	0	<> 0	<> 0	0	FALSE	FALSE	FALSE
	<> 0	0	<> 0	0	FALSE	TRUE	FALSE
	<> 0	<> 0	0	0	TRUE	FALSE	FALSE
3	<> 0	<> 0	<> 0	0	TRUE	TRUE	FALSE
	<> 0	<> 0	0	<> 0	TRUE	TRUE	FALSE
	<> 0	0	<> 0	<> 0	TRUE	TRUE	FALSE
	0	<> 0	<> 0	<> 0	TRUE	TRUE	FALSE
4	<> 0	<> 0	<> 0	<> 0	TRUE	TRUE	TRUE

12 Applikationsbeispiel

Allgemein

Mit der Installation werden 2 Beispielprojekte unter \Program Files\Siemens\Step7\Examples abgelegt:

- Beispielprojekt in AWL „MB_TCP_PN_RED_400“ und
- Beispielprojekt in CFC “MB_TCP_PN_RED_CFC“.

In den Beispielprojekten sind für alle Funktionsvarianten Simatic-Stationen angelegt:

- S7-H-Station ist Client oder Server
- Einseitige oder beidseitige Redundanz

Das S7-Programm soll als Informationsquelle dienen und kann nicht als verbindlicher Lösungsvorschlag kundenspezifischer Anlagenkonfigurationen betrachtet werden.

Simatic-Stationen im Beispielprojekt

Folgende Simatic-Stationen sind im Beispielprojekt angelegt:

Baustein / Stationsname	Einseitig	Beidseitig	Client	Server
H Double-sided (Client)		x	x	
H Double-sided (Server)		x		x
H Single-sided (Client)	x		x	
H Single-sided (Server)	x			x

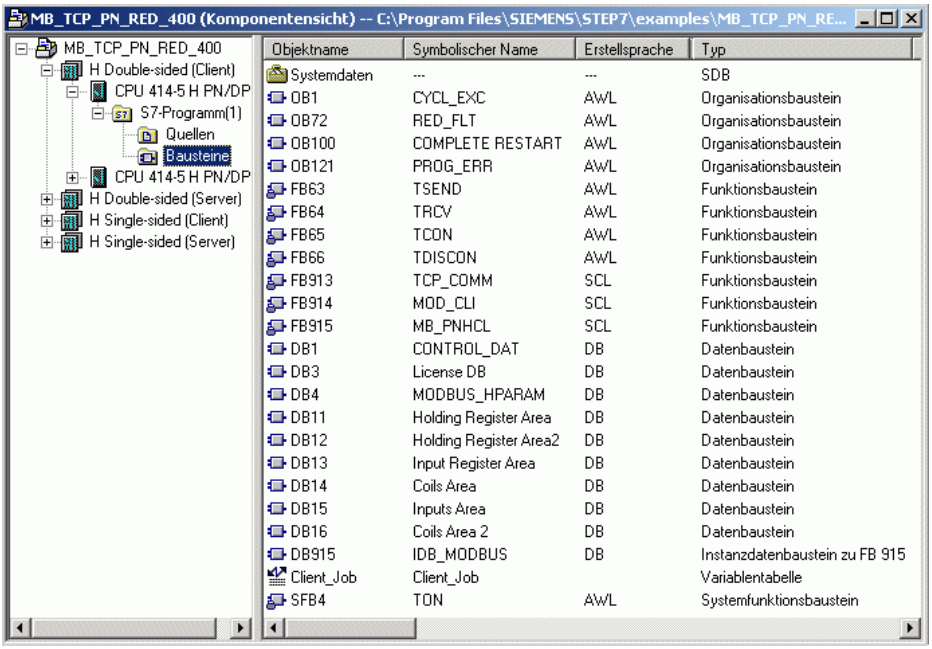
Programmbeispiel

Die Programmbeispiele bestehen aus den Bausteinen:

- Anlauf-Baustein OB100 mit Aufruf des FB915 bzw. FB917
- Programmierfehler-OB121
- Zyklischer Betrieb OB1 bzw. OB35 mit Aufruf des FB915 bzw. FB917
- Globale Datenbausteine zum Auftragsanstoß (z.B. mit Hilfe einer Variablentabelle) und für die Lizenzierung
- Datenbausteine für Register- und Bitwerte

12.1 Beispielprojekt in AWL – Modbus Client

Übersicht



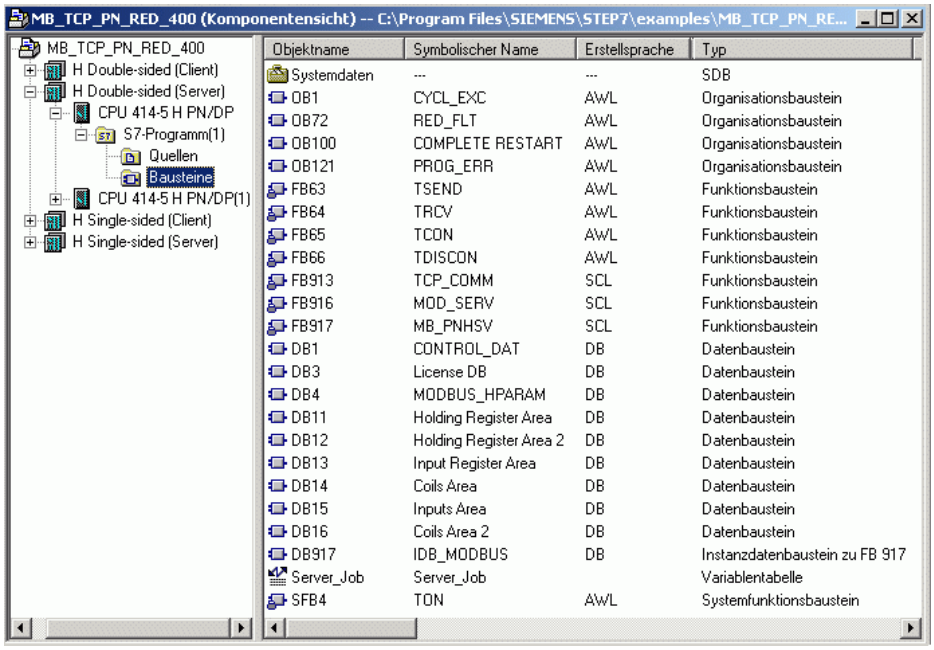
Verwendete Bausteine

Diese Bausteinnummern werden auch im mitgelieferten Beispielprojekt für S7-H-Stationen mit FB MB_PNHCL verwendet.

Baustein	Symbol	Kommentar
OB 1	CYCL_EXC	Zyklische Programmbearbeitung
OB 100	COMPLETE RESTART	Anlauf-OB für Neustart
OB 121	PROG_ERR	Programmierfehler-OB
FB 913	TCP_COMM	intern aufgerufener FB TCP_COMM
FB 914	MOD_CLI	intern aufgerufener FB MOD_CLI
FB 915	MB_PNHCL	Anwenderbaustein FB MB_PNHCL
DB 1	CONTROL_DAT	Arbeits-DB CONTROL DAT für FB MB_PNHCL
DB 3	LICENSE_DB	Lizenz-DB für FB MB_PNHCL
DB 4	MODBUS_HPARAM_PN	Parameter-DB für FB MB_PNHCL
DB 11	Holding Register Area	Werte-DB für Bereich 1
DB 12	Holding Register Area 2	Werte-DB für Bereich 2
DB 13	Input Register Area	Werte-DB für Bereich 3
DB 14	Coils Area	Werte-DB für Bereich 5
DB 15	Inputs Area	Werte-DB für Bereich 6
DB 16	Coils Area 2	Werte-DB für Bereich 7
DB 915	IDB_MODBUS	Instanz-DB für FB MB_PNHCL

12.2 Beispielprojekt in AWL – Modbus Server

Übersicht



Verwendete Bausteine

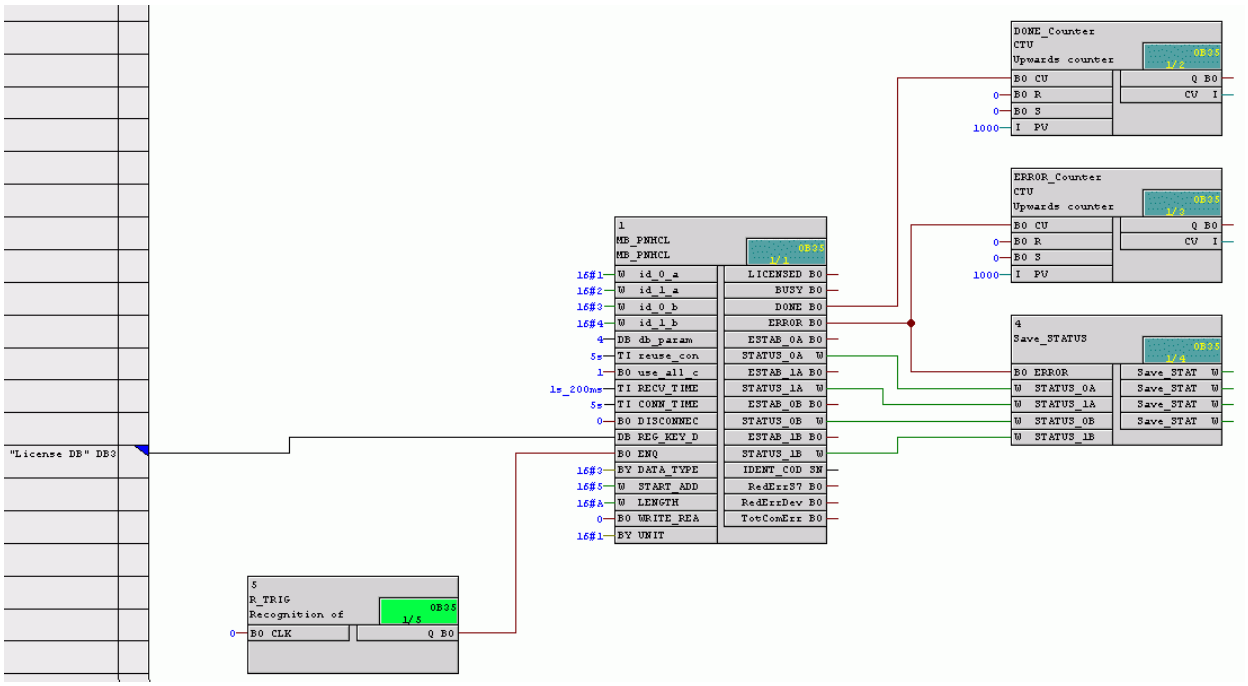
Diese Bausteinnummern werden auch im mitgelieferten Beispielprojekt für S7-H-Stationen mit FB MB_PNHSV verwendet.

Baustein	Symbol	Kommentar
OB 1	CYCL_EXC	Zyklische Programmbearbeitung
OB 100	COMPLETE RESTART	Anlauf-OB für Neustart
OB 121	PROG_ERR	Programmierfehler-OB
FB 913	TCP_COMM	intern aufgerufener FB TCP_COMM
FB 916	MOD_SERV	intern aufgerufener FB MOD_SERV
FB 917	MB_PNHSV	Anwenderbaustein FB MB_PNHSV
DB 1	CONTROL_DAT	Arbeits-DB CONTROL DAT für FB MB_PNHSV
DB 3	LICENSE_DB	Lizenz-DB für FB MB_PNHSV
DB 4	MODBUS_HPARAM_PN	Parameter-DB für FB MB_PNHCL
DB 11	Holding Register Area	Werte-DB für Bereich 1
DB 12	Holding Register Area 2	Werte-DB für Bereich 2
DB 13	Input Register Area	Werte-DB für Bereich 3
DB 14	Coils Area	Werte-DB für Bereich 5
DB 15	Inputs Area	Werte-DB für Bereich 6
DB 16	Coils Area 2	Werte-DB für Bereich 7
DB 917	IDB_MODBUS	Instanz-DB für FB MB_PNHSV

12.3 Beispielprojekt in CFC – Modbus Client

Übersicht

Das Beispielprojekt wurde mit CFC V8.0 Update 1 erstellt.



Verwendete Bausteine

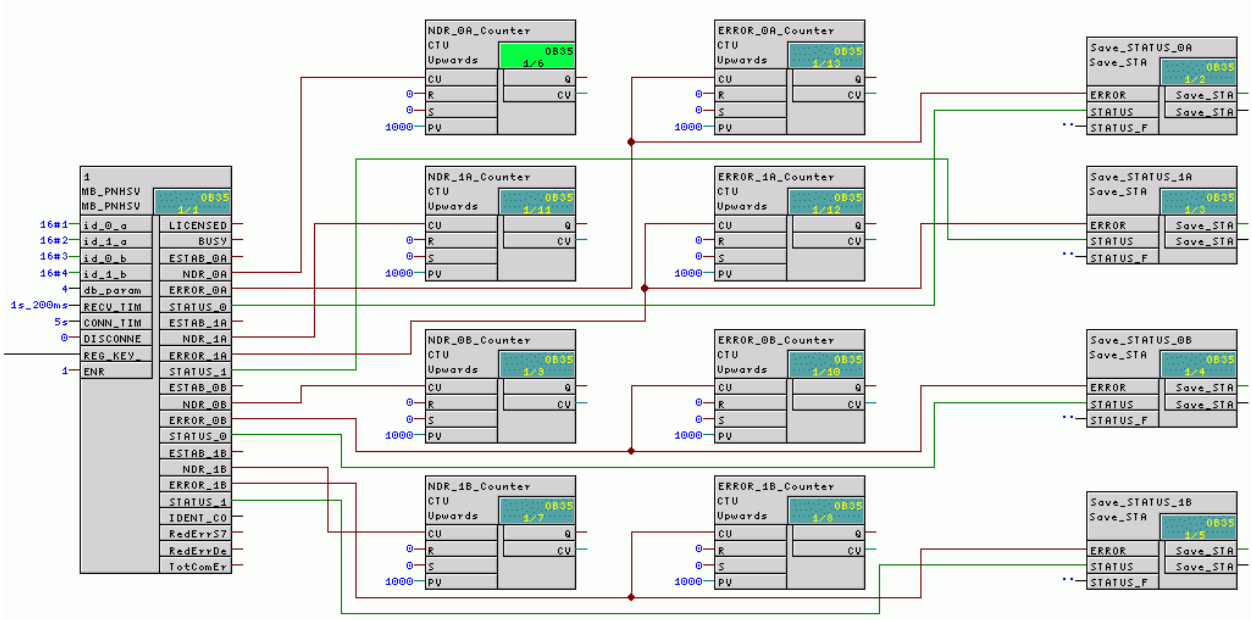
Diese Bausteinnummern werden auch im mitgelieferten Beispielprojekt für S7-H-Stationen mit FB MB_PNHCL verwendet.

Baustein	Symbol	Kommentar
OB 35	CYCL_EXC	Zyklische Programmbearbeitung
OB 100	COMPLETE RESTART	Anlauf-OB für Neustart
OB 121	PROG_ERR	Programmierfehler-OB
FB 99	Save_STATUS	Speicher-DB für Fehler
FB 913	TCP_COMM	intern aufgerufener FB TCP_COMM
FB 914	MOD_CLI	intern aufgerufener FB MOD_CLI
FB 915	MB_PNHCL	Anwenderbaustein FB MB_PNHCL
DB 3	LICENSE_DB	Lizenz-DB für FB MB_PNHCL
DB 4	MODBUS_HPARAM_PN	Parameter-DB für FB MB_PNHCL
DB 11	Holding Register Area	Werte-DB für Bereich 1
DB 12	Holding Register Area 2	Werte-DB für Bereich 2
DB 13	Input Register Area	Werte-DB für Bereich 3
DB 14	Coils Area	Werte-DB für Bereich 5
DB 15	Inputs Area	Werte-DB für Bereich 6
DB 16	Coils Area 2	Werte-DB für Bereich 7

12.4 Beispielprojekt in CFC – Modbus Server

Übersicht

Das Beispielprojekt wurde mit CFC V8.0 Update 1 erstellt.



Verwendete Bausteine

Diese Bausteinnummern werden auch im mitgelieferten Beispielprojekt für S7-H-Stationen mit FB MB_PNHSV verwendet.

Baustein	Symbol	Kommentar
OB 35	CYCL_EXC	Zyklische Programmbearbeitung
OB 100	COMPLETE RESTART	Anlauf-OB für Neustart
OB 121	PROG_ERR	Programmierfehler-OB
FB 99	Save_STATUS	Speicher-DB für Fehler
FB 913	TCP_COMM	intern aufgerufener FB TCP_COMM
FB 916	MOD_SERV	intern aufgerufener FB MOD_SERV
FB 917	MB_PNHSV	Anwenderbaustein FB MB_PNHSV
DB 3	LICENSE_DB	Lizenz-DB für FB MB_PNHSV
DB 4	MODBUS_HPARAM_PN	Parameter-DB für FB MB_PNHSV
DB 11	Holding Register Area	Werte-DB für Bereich 1
DB 12	Holding Register Area 2	Werte-DB für Bereich 2
DB 13	Input Register Area	Werte-DB für Bereich 3
DB 14	Coils Area	Werte-DB für Bereich 5
DB 15	Inputs Area	Werte-DB für Bereich 6
DB 16	Coils Area 2	Werte-DB für Bereich 7

A Literatur

MODBUS Organisation

MODBUS APPLICATION PROTOCOL SPECIFICATION
V1.1b3, April 26, 2012

http://www.modbus.org

Customer Support

Siemens AG
Industry Sector
Industry Automation Division / Industrial Automation Systems
Factory Automation
I IA AS FA

Tel: +49 (0)911 895 7 222
Customer Support

http://www.siemens.com/s7modbus

Siemens Aktiengesellschaft Subject to change without notice release: 08/2014