Master-Slave Kommunikation über einen CM PtP mit Hilfe des Modbus RTU Protokolls

S7-1500 CM PtP RS422/485 HF, ET 200SP CM PtP

Applikationsbeschreibung • März 2013

Applikationen & Tools

Answers for industry.

SIEMENS

Siemens Industry Online Support

Dieser Beitrag stammt aus dem Siemens Industry Online Support. Durch den folgenden Link gelangen Sie direkt zur Downloadseite dieses Dokuments:

http://support.automation.siemens.com/WW/view/de/68202723

Vorsicht:
Die in diesem Beitrag beschriebenen Funktionen und Lösungen beschränken sich überwiegend auf die Realisierung der Automatisierungsaufgabe. Bitte beachten Sie darüber hinaus, dass bei Vernetzung Ihrer Anlage mit anderen Anlagenteilen, dem Unternehmensnetz oder dem Internet entsprechende Schutzmaßnahmen im Rahmen von Industrial Security zu ergreifen sind. Weitere Informationen dazu finden Sie unter der Beitrags-ID 50203404.

http://support.automation.siemens.com/WW/view/de/50203404

SIEMENS

SIMATIC

Master-Slave Kommunikation mit Modbus RTU

Gewährleistung und Haftung

Hinweis

Die Applikationsbeispiele sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit hinsichtlich Konfiguration und Ausstattung sowie jeglicher Eventualitäten. Die Applikationsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern sollen lediglich Hilfestellung bieten bei typischen Aufgabenstellungen. Sie sind für den sachgemäßen Betrieb der beschriebenen Produkte selbst verantwortlich. Diese Applikationsbeispiele entheben Sie nicht der Verpflichtung zu sicherem Umgang bei Anwendung, Installation, Betrieb und Wartung. Durch Nutzung dieser Applikationsbeispiele erkennen Sie an, dass wir über die beschriebene Haftungsregelung hinaus nicht für etwaige Schäden haftbar gemacht werden können. Wir behalten uns das Recht vor, Änderungen an diesen Applikationsbeispielen jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in diesem Applikationsbeispiel und anderen Siemens Publikationen, wie z.B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.

Für die in diesem Dokument enthaltenen Informationen übernehmen wir keine Gewähr.

Unsere Haftung, gleich aus welchem Rechtsgrund, für durch die Verwendung der in diesem Applikationsbeispiel beschriebenen Beispiele, Hinweise, Programme, Projektierungs- und Leistungsdaten usw. verursachte Schäden ist ausgeschlossen, soweit nicht z.B. nach dem Produkthaftungsgesetz in Fällen des Vorsatzes, der groben Fahrlässigkeit, wegen der Verletzung des Lebens, des Körpers oder der Gesundheit, wegen einer Übernahme der Garantie für die Beschaffenheit einer Sache, wegen des arglistigen Verschweigens eines Mangels oder wegen Verletzung wesentlicher Vertragspflichten zwingend gehaftet wird. Der Schadensersatz wegen Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht Vorsatz oder grobe Fahrlässigkeit vorliegt oder wegen der Verletzung des Lebens, des Körpers oder der Gesundheit zwingend gehaftet wird. Eine Änderung der Beweislast zu Ihrem Nachteil ist hiermit nicht verbunden.

Inhaltsverzeichnis

1 Aufgabe

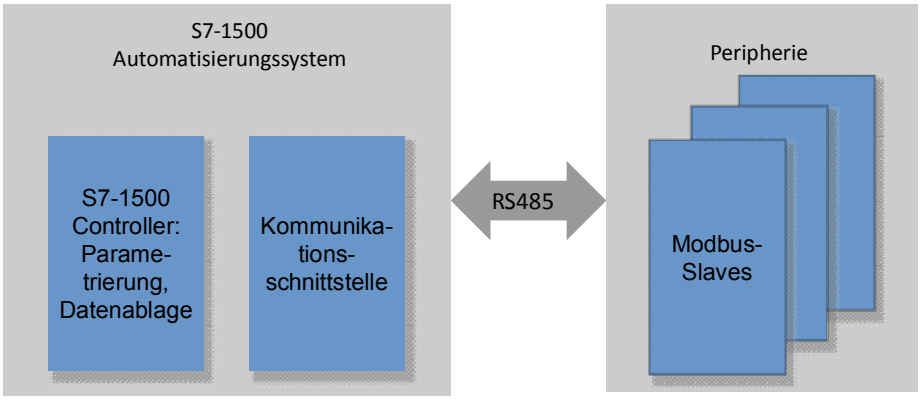
Einleitung

Diese Applikation zeigt Ihnen den Umgang mit dem Modbus RTU Protokoll der CM PtPs in der SIMATIC S7-1500 und dem dezentralen Peripheriesystem ET 200SP.

Überblick über die Automatisierungsaufgabe

Folgendes Bild gibt einen Überblick über die Automatisierungsaufgabe.

Abbildung 1-1



Beschreibung der Automatisierungsaufgabe

Die Applikation soll folgende Anforderungen abdecken:

- Demonstrieren des Umgangs mit dem CM PtP RS422/485 HF und dem CM PtP der ET 200SP an einem konkreten Anwendungsfall mit Modbus RTU.
- gekapselte, flexible Master/Slave Programmierung in einem Beispiel.

2 Lösung

2.1 Übersicht Gesamtlösung

Ziel der Applikation

Diese Applikation zeigt Ihnen

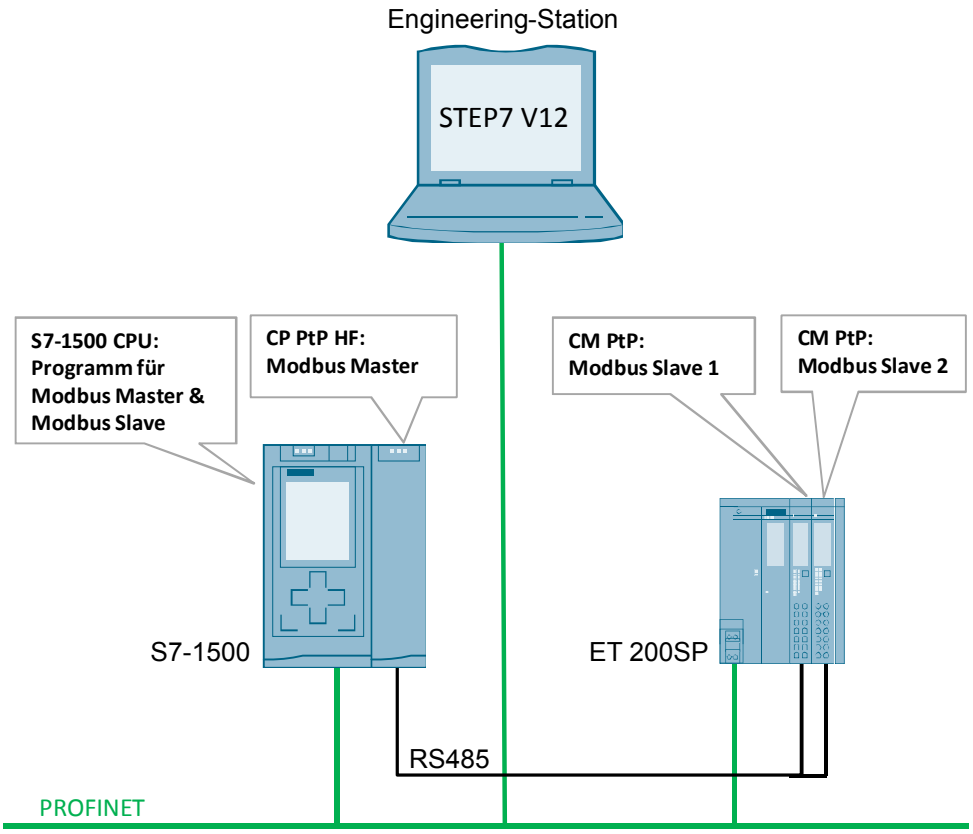
- die Parametrierung eines CM (Communication Modul) PtP für die Kommunikation mit Modbus RTU.
- die flexible Programmierung eines Modbus Masters zur Kommunikation mit mehreren Slaves.
- die Programmierung eines Modbus Slaves zur Kommunikation mit einem Master.

Die genaue Funktionsweise des Programms wird in Kapitel 4 beschrieben.

Schema

Die folgende Abbildung zeigt schematisch die wichtigsten Komponenten der Lösung:

Abbildung 2-1



Kerninhalte dieser Applikation

Folgende Punkte werden in dieser Applikation vermittelt:

- Grundlagen zum Modbus RTU-Protokoll
- Projektierung der Hardwareumgebung
- Parametrierung der seriellen Schnittstellen für das Modbus RTU Protokoll
- Programmierung des Daten-Lesens im Modbus RTU Master
- Programmierung der Modbus Slave Funktionalität in einer SIMATIC S7 CPU

Im Beispielprojekt liest der CM PtP RS422/485 HF als Modbus Master abwechselnd von den beiden Slaves (CM PtPs der ET 200SP) acht Wörter an Daten.

Das Anwenderprogramm des Masters und der Slaves liegt in der S7-1500 CPU.

Programmierung des Modbus Masters mit Hilfe des FB Master_Modbus (FB776) und

- einem Instanz-Datenbaustein (z.B. DB Master_Modbus_DB (DB775))
- dem DB Comm_Data (DB778)
  - mit der Struktur Param
  - mit der Struktur PublicParam
  - mit dem Array Master_comm
  - ohne das Array Slave
- für die Ausgangsparameter den DB Output_Data (DB779) mit der Struktur Master

Programmierung des Modbus Slaves mit Hilfe des FB Slave_Modbus (FB775) und

- einem Instanz-Datenbaustein (z.B. DB Slave_Modbus_DB_1 (DB776))
- dem DB Comm_Data (DB778)
  - mit der Struktur Param
  - mit der Struktur PublicParam
  - mit dem Array Slave
  - ohne das Array Master_Comm
- für die Ausgangsparameter den DB Output_Data (DB779) mit der Struktur SlaveX

Vorteile

Die vorliegende Applikation bietet Ihnen den Vorteil eines schnellen Einstiegs in das Thema Modbus RTU im SIMATIC S7-1500 Umfeld.

Sie erhalten gekapselte Funktionen zur Programmierung entweder eines Modbus-Slaves oder eines Modbus Masters.

Gültigkeit

- Softwareversionen ab TIA Portal V12
- SIMATIC S7-1500 CPUs
- CM PtP RS422/485 HF, CM PtP der ET 200SP

Abgrenzung

Diese Applikation enthält keine

- Einführung in das Thema SCL-Programmierung.
- Grundlagen zum TIA Portal V12.

Grundlegende Kenntnisse über diese Themen werden vorausgesetzt.

2.2 Verwendete Hard- und Software-Komponenten

Die Applikation wurde mit den nachfolgenden Komponenten erstellt:

Hardware-Komponenten

Tabelle 2-1

Komponente	Anz.	Bestellnummer	Hinweis
PM 70W 120/230 AVC	1	6EP1332-4BA00	
CPU 1516-3 PN/DP	1	6ES7516-3AN00-0AB0	Auch andere CPUs aus dem S7-1500 Spektrum sind einsetzbar
CM PtP RS422/485 HF	1	6ES7541-1AB00-0AB0	Die Basic-Baugruppe (BA) ist nicht Modbus RTU-fähig
IM 155-6PN ST	1	6ES7155-6AU00-0BN0	ET 200SP
CM PtP	2	6ES7137-6AA00-0BA0	
BaseUnit	2	6ES7193-6BP00-0xA0	
Servermodul	1	6ES7193-6PA00-0AA0	Bei der Bestellung der Kopfstation bereits mit enthalten.

Hinweis Wenn Sie andere Hardware als im Beispielprojekt verwenden, dann müssen Sie entsprechende Änderungen in der Hardwarekonfiguration vornehmen!

Standard Software-Komponenten

Tabelle 2-2

Komponente	Anz.	/Bestellnummer	Hinweis
STEP 7 V12 (TIA Portal V12)	1	6ES78221AE02-0YA5	

Beispieldateien und Projekte

Die folgende Liste enthält alle Dateien und Projekte, die in diesem Beispiel verwendet werden.

Tabelle 2-3

Komponente	Hinweis
68202723_ S7-1500_ModbusRTU_CODE_V1d0.zip	Diese gepackte Datei enthält das archivierte STEP 7 Projekt.
68202723_S7-1500_ModbusRTU_ DOKU_V1d0_d.pdf	Dieses Dokument.

Für weiterführende Dokumentationen etwa zur dezentralen Peripherie ET 200SP beachten Sie bitte das Kapitel 8 Literaturhinweise.

3 Beschreibung des Modbus RTU-Protokolls

3.1 Funktionsweise Modbus RTU

Übersicht

Modbus RTU (Remote Terminal Unit) ist ein Standardprotokoll für die serielle Kommunikation zwischen Master und Slave.

Andere Protokolle der Modbus-Spezifikation, wie Modbus ASCII werden von den seriellen SIMATIC S7-1500 CMs nicht unterstützt.

Master-Slave Beziehung

Modbus RTU nutzt eine Master/Slave-Beziehung, in der die gesamte Kommunikation von einem einzigen Master-Gerät ausgeht, während die Slaves lediglich auf die Anforderungen des Masters reagieren. Der Master sendet eine Anforderung an eine Slave-Adresse und nur der Slave mit dieser Slave-Adresse antwortet auf den Befehl.

Sonderfall: Bei Verwendung der Modbus-Slaveadresse 0 sendet der CM PtP ein Broadcast-Telegramm an alle Slaves (ohne eine Slave-Antwort zu erhalten).

Kommunikationsablauf

Die Kommunikation mit Modbus RTU läuft immer nach folgendem Schema ab:

1. Der Modbus-Master sendet eine Anforderung an einen Modbus-Slave in das Netz.
2. Der Slave antwortet mit einem Antworttelegramm, in dem die angeforderten Daten enthalten sind oder das den Empfang der Anforderung quittiert.
3. Wenn der Slave die Anforderung des Masters nicht verarbeiten kann, dann antwortet der Slave mit einem Fehlertelegramm.

Die folgende Tabelle zeigt als Beispiel den Aufbau des Telegramms, wenn Daten aus einem oder mehreren Halteregistern des Modbus Slaves gelesen werden sollen (Modbus Standard).

Tabelle 3-1

Telegramm	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	…
Anfrage	Slave-Adresse	Funktionscode	Anfangsadresse (ab welchem Halteregister gelesen werden soll)		Anzahl Register		
Gültige Antwort	Slave-Adresse	Funktionscode	Länge	Registerdaten			
Fehlermeldung	Slave-Adresse	0x83	Errorcode	---			

Der Funktionscode zeigt dem Slave an, welche Funktion er ausführen soll. Die Tabelle 3-2 listet die Funktionscodes auf, die mit den CM PtPs verwendet werden können:

Tabelle 3-2

Funktionscode	Funktion
01	Ausgangsbit lesen
02	Eingangsbit lesen

Funktionscode	Funktion
03	Halteregister lesen
04	Eingangswörter lesen
05	Ein Ausgangsbit schreiben
06	Ein Halteregister schreiben
15	Ein oder mehrere Ausgansbits schreiben
16	Ein oder mehrere Halteregister schreiben
11	Statuswort und Ereigniszähler der Slave-Kommunikation lesen
08	Slave Zustand über Daten-Diagnosecode prüfen/ Slave Ereigniszähler über Daten-Diagnosecode zurücksetzen

Leistungseckdaten

Anzahl Geräte am Bus

Tabelle 3-3

Modbus Norm	Anzahl Adressen
Modbus RTU-Standard	247
Modbus RTU-Extended Adressierung	65535

Bei Leitungslängen größer 50m müssen Sie für einen störungsfreien Datenverkehr einen Abschlusswiderstand von ca. 330 Ohm auf der Empfängerseite einlöten.

Datenlänge

Tabelle 3-4

Anweisungstyp	Funktionscodes	Maximale Anzahl pro Anforderung
Bit-Anweisung	1, 2, 5, 15	1992 Bits
Wort-Anweisung	3, 4, 6, 16	124 Register (Worte)

Die angegebenen Werte gelten für einen CM PtP RS422/485 HF und alle seriellen Kommunikationsprozessoren der SIMATIC S7-1500.

3.2 Projektierung in STEP 7 V12

Überblick

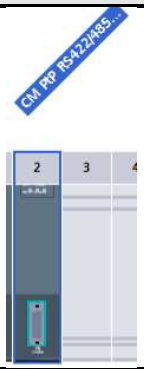
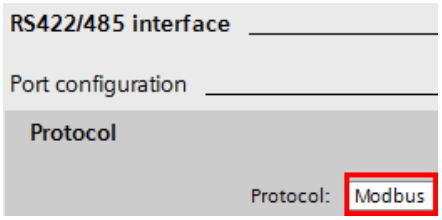
Das TIA-Portal ermöglicht die Projektierung einer Modbus-RTU Kommunikation. Dieses Kapitel zeigt Ihnen,

- welche Einstellungen Sie in der Hardware-Konfiguration vornehmen müssen.
- welche Eigenschaften die Anweisungen zur Modbus-RTU Kommunikation besitzen.

Hardware-Konfiguration

In der Hardware-Konfiguration stellen Sie für den verwendeten CM nur ein, dass er über das Modbus Protokoll kommunizieren soll. Tabelle 3-5 zeigt das Vorgehen:

Tabelle 3-5

Nr.	Vorgehen	Anmerkung
1.	Öffnen Sie Ihr Projekt. Wechseln Sie in die Gerätekonfiguration und wählen Sie ihr CM PtP aus.	
2.	Wechseln Sie in die „Eigenschaften> RS422/485-Schnittstelle> Anschlusskonfiguration“ („Properties> RS422/485 interface> Port configuration“) und wählen Sie als Protokoll „Modbus“. Alle weiteren Einstellungen werden im Anwenderprogramm vorgenommen.	

Kommunikationsbausteine (Anweisungen) für Modbus RTU

Die Einrichtung eines Kommunikationsmoduls für das Modbus-RTU Protokoll sowie dessen Betrieb als Master oder Slave, wird über die folgenden Anweisungen realisiert:

Tabelle 3-6

Anweisung	Beschreibung
Modbus_Comm_Load	Konfiguriert ein Kommunikationsmodul für die Kommunikation über das Modbus-RTU-Protokoll. Die Anweisung setzt Parameter wie • die Baudrate • Parität • Flusskontrolle • … Erst nach dem erfolgreichen Parametrieren des Kommunikationsmoduls ist ein Aufruf von **Modbus_Master** oder **Modbus_Slave** sinnvoll.
Modbus_Master	Kommuniziert als Modbus-Master über einen Port, der mit der Anweisung **Modbus_Comm_Load** konfiguriert wurde. Der Funktionscode des Modbus RTU-Protkolls (siehe Tabelle 3-2) wird über folgende Eingänge festgelegt: • MODE • DATA_ADDR • DATA_LEN Eine Übersicht welcher Funktionscode welchen Eingangsparametern entspricht erhalten Sie in der Hilfe zur Anweisung **Modbus_Master** im TIA Portal.
Modbus_Slave	Mit der Anweisung **Modbus_Slave** kann Ihr Programm über einen PtP-Port eines CM als Modbus-Slave kommunizieren. Die Anweisung **Modbus_Slave** realisiert die Kommunikation mit einem Modbus-Master. Die Belegung der Parameter entnehmen Sie bitte der Hilfe zur Anweisung **Modbus_Slave** im TIA Portal.

4 Beschreibung des STEP 7 Programms

4.1 Übersicht

Funktionen

Das S7-Programm realisiert die folgenden Funktionen

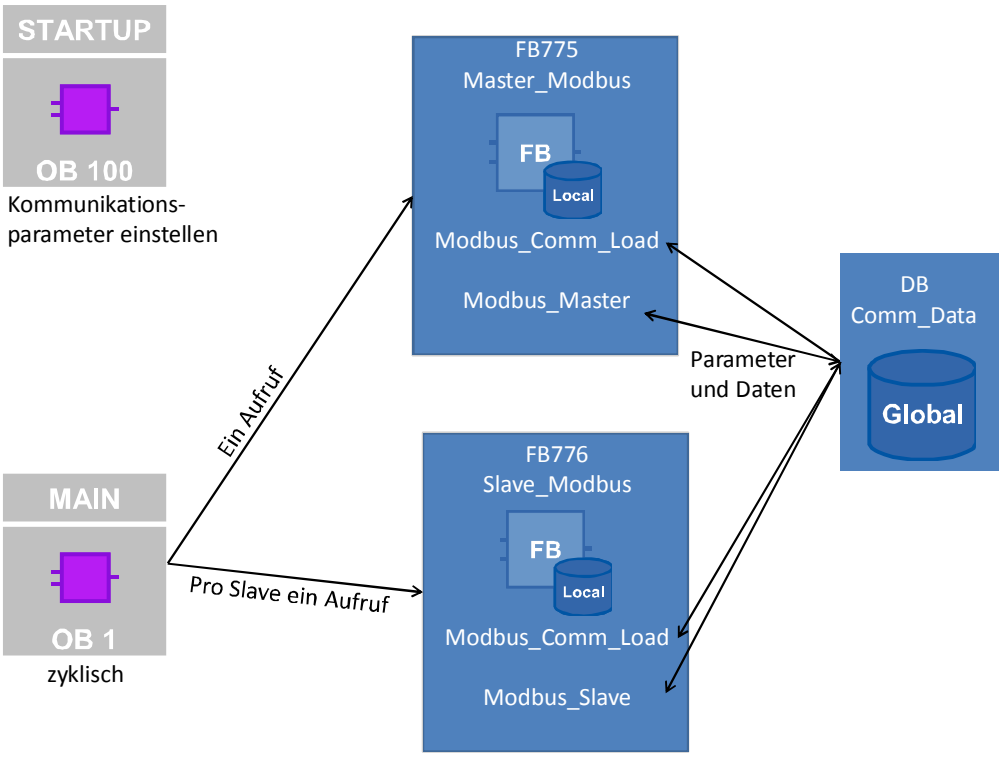
- Parametrieren der Kommunikationsmodule für die Kommunikation mit Modbus-RTU.
- Kommunikation der S7-CPU als Modbus-Master zum zyklischen Lesen von je acht Wörtern von zwei Modbus-Slaves.
- Kommunikation der S7-CPU über die dezentrale Peripherie (ET 200SP mit CM PtP-Modulen) als Modbus-Slave.

Sowohl das Kommunikationsprogramm für den Master als auch das für die Slaves ist in der SIMATIC S7-1500 CPU hinterlegt.

Sie können das Beispielprogramm an Ihre Anforderungen anpassen. Beachten Sie dazu Kapitel 5.

Programmübersicht

Abbildung 4-1



Bausteine und Anweisungen

Folgende Bausteine werden im STEP 7-V12 Projekt verwendet:

Tabelle 4-1

Element	symbolischer Name	Beschreibung	
OB1	Main	Beinhaltet das Hauptprogramm. • Ruft den FB **Master_Modbus** und den FB **Slave_Modbus** auf. • Liest zyklisch acht Wörter über den Modbus-Master abwechselnd von den Modbus-Slaves.	Programmaufruf
OB100	Startup	• Die Parameter zur Kommunikationseinstellung mit **Modbus_Comm_Load** werden vorbelegt. • Parameter für den Master zur Kommunikation mit den Slaves werden initialisiert. • Parameter für die Slaves werden initialisiert.	
FB775	Master_Modbus	Einrichten eines Kommunikationsmoduls als Modbus-Master zur Kommunikation mit zwei Modbus-Slaves.	Eigenentwicklung
FB776	Slave_Modbus	Pro Aufruf: Einrichten eines Kommunikationsmoduls als Modbus-Slave zur Kommunikation mit einem Modbus-Master.	
DB775	Master_Modbus_DB	Instanz-DB des FB **Master_Modbus**	
DB776	Slave_Modbus_DB_1	Instanz-DB des FB **Slave_Modbus_DB**	
DB777	Slave_Modbus_DB_2	Instanz-DB des FB **Slave_Modbus_DB**	
DB778	Comm_Data	Beinhaltet • die Parameter der Modbus-Kommunikationsverbindung. • ein Array vom Datentyp Data_for_Master, das dem Master die nötigen Daten zur Kommunikation mit den Slaves zur Verfügung stellt. • ein Array vom Datentyp Data_Slave, das den Slaves die nötigen Daten zur Kommunikation zur Verfügung stellt.	
DB779	Output_Data	Beinhaltet die Output-Parameter der in OB1 aufgerufenen Funktionsbausteine	
FB640	Modbus_Comm_Load	Konfiguration eines Kommunikationsmoduls zur Kommunikation mit dem Modbus-Protokoll.	Systembausteine
FB641	Modbus_Master	Kommunikation des Moduls als Modbus-Master über den mit **Modbus_Comm_Load** konfigurierten Port.	

Element	symbolischer Name	Beschreibung	
FB642	Modbus_Slave	Kommunikation des Moduls als Modbus-Slave über den mit **Modbus_Comm_Load** konfigurierten Port.	Systembausteine
weitere System-bausteine	zum Beispiel: Receive_Config	Werden von den genannten Systembausteinen FB640-FB642 aufgerufen.	

4.2 Funktionsweise des FB Master_Modbus (FB775)

4.2.1 Zustände und Aufruf des FB Master_Modbus

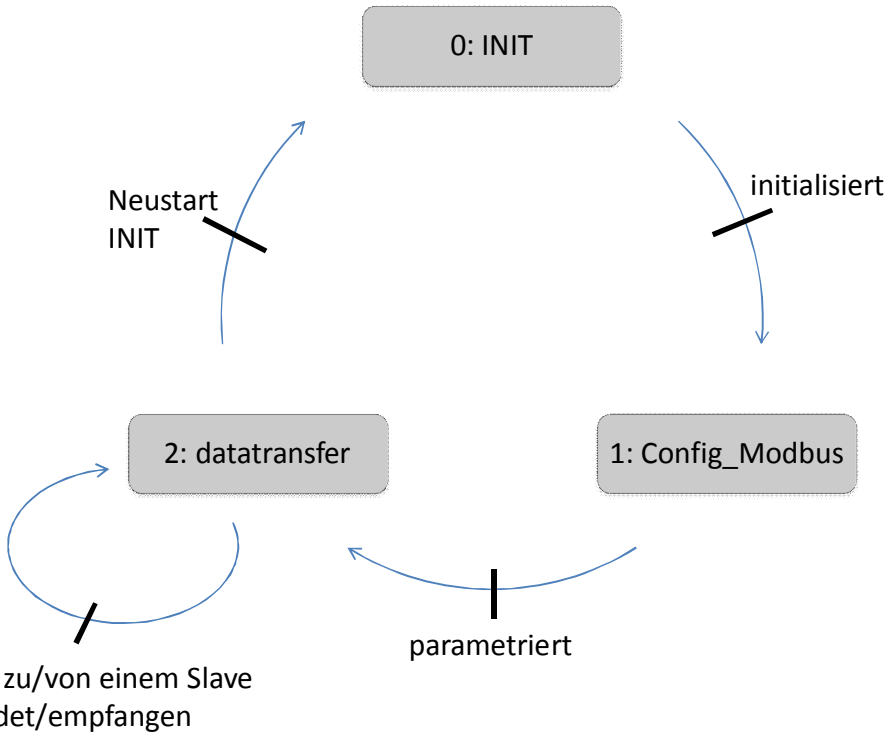
Zustände

Der FB **Master_Modbus** erfüllt die folgenden Aufgaben:

- Initialisieren der Kommunikationsparameter
- Parametrierung des Master-Kommunikationsmoduls
- Verwalten der Kommunikationsaufträge zu den Modbus-Slaves

Die Funktionalitäten sind in einer einfachen Schrittkette mit den folgenden Zuständen realisiert:

Abbildung 4-2

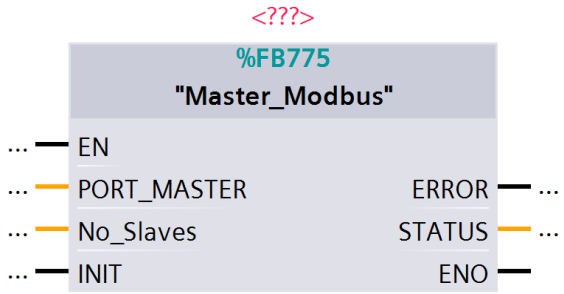


Eine genaue Beschreibung der einzelnen Zustände erhalten Sie ab Kapitel 4.2.2.

Aufruf und Parameter des FB Master_Modbus

Die Abbildung 4-3 zeigt die Aufrufschnittstelle des FB **Master_Modbus** (FB775). Die Parameter werden in Tabelle 4-2 beschrieben.

Abbildung 4-3



Der FB Master_Modbus besitzt folgende Ein-und Ausgangsparameter:

Tabelle 4-2

Parameter	Typ	Anmerkung
PORT_MASTER	IN: HW_SUBMODULE	HW-Kennung des Master-Kommunikationsmoduls
No_Slaves	IN: Int	Anzahl der aktiven, im DB **Comm_Data** (Array Master_comm) hinterlegten Slaves.
INIT	IN: Bool	Eine positive Flanke am Eingang INIT bewirkt, dass die Kommunikationsparameter aus dem DB **Comm_Data** neu übernommen werden.
ERROR	OUT: Bool	ERROR = TRUE, wenn ein Fehler im Baustein ansteht
STATUS	OUT: DWord	STATUS des Bausteins. Für nähere Informationen siehe unten.

Ausgangsparameter: STATUS

Tabelle 4-3

Status	Beschreibung
High Word	Zeigt an, an welcher Stationsadresse (an welchem Slave) der Status aufgetreten ist.
Low Word	Status des Bausteins, an dem der Fehler aufgetreten ist, oder • 16#FFFD: No_Slaves = 0. • 16#FFFE: übergebene MB_ADDR im DB **Comm_Data** = 0.

4.2.2 Zustand „INIT“

Übersicht

Der Zustand „INIT“ wird durch den Aufruf des FB **Master_Modbus** im OB1 im ersten Zyklus eingeleitet. Ebenfalls eingeleitet wird der Zustand „INIT“ durch eine positive Flanke am Eingang INIT.

In diesem Zustand werden für den Programmablauf benötigte Parameter initialisiert.

Beschreibung

Tabelle 4-4

Nr.	Vorgang	Anmerkung
1.	Zurücksetzen der REQ-Eingänge der verwendeten Anweisungen.	Es wird sichergestellt, dass an den Steuerungseingängen eine positive Flanke erzeugt wird.
2.	Zurücksetzen der im Funktionsbaustein verwendeten Zählvariablen, die beim Auftreten von ERROR=TRUE oder DONE=TRUE inkrementiert werden.	
3.	Sperren der Slaves mit der Modbus Stationsadresse=0.	Die Adresse 0 dient in der Modbus-Kommunikation als Broadcast.
4.	Festlegen mit welchem Slave die Kommunikation begonnen wird.	Die Kommunikation wird mit dem ersten Slave begonnen, dessen Modbus-Stationsadresse im Array Master_Comm ungleich null ist.

4.2.3 Zustand „Config_Modbus “

Übersicht

Nach der erfolgreichen Initialisierung der Parameter tritt der FB **Master_Modbus** in den Zustand „Config_Modbus“.

In diesem Zustand wird die Anweisung **Modbus_Comm_Load** zum Einstellen der Kommunikationsparameter aufgerufen.

Programmcode

Die Abbildung 4-4 zeigt den Aufruf der Anweisung **Modbus_Comm_Load**.

Abbildung 4-4

```
"Config_Modbus":
        (*****************************************************************************
        control.state = Config_modbus:
        Initialisation of the Port with the Parameters given in DB Comm_Data (DB777).
        *****************************************************************************
        #Modbus_Comm_Load_Instance.ICHAR_GAP:="Comm_Data".PublicParam.Ichar_gap;
        #Modbus_Comm_Load_Instance.MODE:="Comm_Data".PublicParam.Mode;
        #Modbus_Comm_Load_Instance.LINE_PRE:="Comm_Data".PublicParam.Line_pre;
        // call of the Modbus_Comm_Load: Configuration of the communication parameters
        #Modbus_Comm_Load_Instance(REQ :=#control.req_comm,
                                   MB_DB:=#Modbus_Master_Instance.MB_DB,
                                   "PORT":=#PORT_MASTER,
                                   BAUD := "Comm_Data".Param.Baud,
                                   FLOW_CTRL := "Comm_Data".Param.Flow_Ctrl,
                                   PARITY := "Comm_Data".Param.Parity,
                                   RTS_ON_DLY:="Comm_Data".Param.Rts_on_dly,
                                   RTS_OFF_DLY:= "Comm_Data".Param.Rts_off_dly,
                                   RESP_TO:="Comm_Data".Param.Resp_to,
                                   DONE=>#survey.done_comm,
                                   ERROR=> #survey.err_comm,
                                   STATUS=> #stat
      );
      //set the req input of the Modbus_Comm_Load instruction
      #control.req_comm := 1;
      (*****************************************************************************
        Analysis of the Output parameters and backup of the status if an error occurs.
        --> retry Modbus_Comm_Load if error occurs
        --> Go to control.step "datatransfer" if Modbus_Comm_Load is succesfull
      *****************************************************************************
```

1. 2. 3.

Beschreibung

Die folgende Schritttabelle beschreibt den Programmcode:

Tabelle 4-5

Nr.	Vorgang	Anmerkung
5.	Vorbelegen der öffentlichen Datenbausteinvariablen.	<table><tr><th>Variable</th><th>Typ</th><th>Beschreibung</th></tr><tr><td>ICHAR_GAP</td><td>Word</td><td>Verzögerung des Zeichenabstands, zusätzlich zum Modbus-Standardwert.</td></tr><tr><td>RETRIES</td><td>Word</td><td>Anzahl der Versuche, bevor der Fehlercode 0x80C8 ausgegeben wird.</td></tr><tr><td>EN_DIAG_ALARM</td><td>Word</td><td>Diagnosemeldung aktivieren.</td></tr><tr><td>MODE</td><td>USInt</td><td>Betriebsart (Voll- oder Halbduplex, RS232/RS485/RS422).</td></tr><tr><td>LINE_PRE</td><td>USInt</td><td>Vorbelegung der Empfangsleitung.</td></tr></table>
6.	Das Master-Kommunikationsmodul wird mit der Anweisung **Modbus_Comm_Load** für die Modbus RTU Kommunikation parametriert.	
7.	Auswerten des ERROR- und DONE-Ausgangs. Nach Abschluss des Zustands ist der Port des Kommunikationsmoduls bereit per Modbus RTU zu kommunizieren.	• Bei ERROR=TRUE wird der Fehlerzähler inkrementiert und der Status abgespeichert. • Bei DONE=TRUE wird der Done-Zähler inkrementiert und der nächste Zustand getriggert. Für weitere Details können Sie im Programmcode nachsehen.

Hinweis

Ein Kommunikationsmodul soll jeweils nur mit einem **Modbus_Comm_Load** initialisiert werden.

Pro **Modbus_Comm_Load** kann nur ein **Modbus_Master** oder ein **Modbus_Slave** aufgerufen werden.

4.2.4 Zustand „datatransfer“

Übersicht

Der Baustein befindet sich nach der erfolgreichen Parametrierung des Kommunikationsmoduls im Zustand „datatransfer“.

In diesem Zustand werden die Kommunikationsaufträge an die Modbus-Slaves abgesetzt und die Kommunikation verwaltet.

Programmcode

Abbildung 4-5

```
#Modbus_Master_Instance(REQ:=#control.req_master,
                        DATA_PTR:="Comm_Data".Master_comm[#control.number].buffer,
                        MB_ADDR := "Comm_Data".Master_comm[#control.number].MB_ADDR,
                        MODE:= "Comm_Data".Master_comm[#control.number].MODE,
                        DATA_ADDR:= "Comm_Data".Master_comm[#control.number].DATA_ADDR,
                        DATA_LEN:="Comm_Data".Master_comm[#control.number].DATA_LEN,
                        BUSY => #survey.busy_master,
                        DONE=>#survey.done_master,
                        ERROR=>#survey.err_master,
                        STATUS=> #stat
);
(*****************************************************************************************
  analysing of the output parameters busy, done and error of the instruction Modbus_Master
*****************************************************************************************)
    //after there is an error or a done on the Modbus_Master instruction:
    //change Modbus station address (MB_ADDR)
    IF #control.number < #No_Slaves THEN
      WHILE ("Comm_Data".Master_comm[#control.number].LOCK) AND (#control.number < #No_Slaves) DO
        #control.number:=#control.number +1;
      END_WHILE;
    END_IF;
    #control.number:=#control.number +1;
    IF #control.number >#No_Slaves THEN
        #control.number:=#survey.first_unlocked;
    END_IF;

  END_IF;
// sets the request input of the Modbus_Master instruction
IF NOT #survey.busy_master  THEN
  #control.req_master := 1;
END_IF;
```

Beschreibung

Tabelle 4-6

Nr.	Vorgang	Anmerkung
1.	Die Anweisung **Modbus_Master** wird mit den Parametern aus dem aktiven UDT Data_for_Master aufgerufen. Die Anweisung bewirkt mit den im Beispielprogramm gesetzten Parametern ein Lesen von acht Wörtern aus dem Slave und die Ablage der Daten in den Empfangspuffer des Slaves (im UDT Data_for_Master).	Wenn Sie die Aufträge an die Modbus-Slaves abändern möchten, dann nehmen Sie dazu Kapitel 0 zu Hilfe.
2.	Die Ausgänge BUSY, NDR und ERROR werden ausgewertet.	Für NDR=TRUE oder ERROR=TRUE werden die entsprechenden Fehler- bzw. Erfolgszähler hochgezählt. Für Details schauen Sie bitte im Programcode nach.
3.	Wenn ein Auftrag abgeschlossen ist, dann wird der nächste Slave im Array Master_comm als aktiv markiert.	Im nächsten OB1-Zyklus wird dadurch ein Telegramm an diesen Slave gesendet.
4.	Wenn der Baustein nicht beschäftigt ist (und damit der alte Auftrag abgeschlossen ist), dann wird ein neuer Auftrag angestoßen.	

4.2.5 Der UDT Data_for_Master

Übersicht

Der UDT (User Defined Datatype) Data_for_Master enthält die für den FB **Master_Modbus** relevanten Informationen zur Kommunikation mit einem Modbus Slave.

Aufbau

Abbildung 4-6

Data_for_Master

	Name	Datentyp	Kommentar
	MB_ADDR	UInt	station address of the slave
	MODE	USInt	modus: read or write
	DATA_ADDR	UDInt	specifies the data addresse where to read the data
	DATA_LEN	UInt	specifies the data length
	LOCK	Bool	slave locked -> no communication
	ERROR	Bool	error at communication with slave
	STATUS	Word	status of that error
▶	buffer	array[0..7] of Word	buffer for the data of or for the slave

Verwendung

Das Beispielprojekt enthält im DB **Comm_Data** ein Array aus zwei UDTs Data_for_Master. Ein UDT enthält Parameter für jeweils einen Kommunikationsauftrag des Modbus Masters mit einem Modbus Slave.

Die Parameter

- MODE
- DATA_ADDR
- DATA_LEN

spezifizieren den Auftrag des Masters an den Slave. Genaue Informationen zum, in Abhängigkeit von den Parametern, verwendeten Modbus-Funktionscode finden Sie in der Hilfe von STEP 7 V12 zur Anweisung **Modbus_Master**.

Der Bereich buffer dient als Ablage der Daten, die vom Slave gelesen werden.

Wenn Sie mit weiteren Slaves kommunizieren, oder andere Datenbereiche lesen/schreiben wollen, dann beachten Sie bitte das Kapitel 5.3.

4.3 Funktionsweise des FB Slave_Modbus (FB776)

4.3.1 Parameter

Überblick

Der FB Slave_Modbus

- initialisiert ein CM (Communication Modul)
- richtet die Kommunikation des CMs als Modbus-Slave ein.

Parameter des FB Slave_Modbus

Die Abbildung 4-7zeigt die Aufrufschnittstelle des FB **Slave_Modbus**. Die Parameter werden in Tabelle 4-7 beschrieben.

Abbildung 4-7

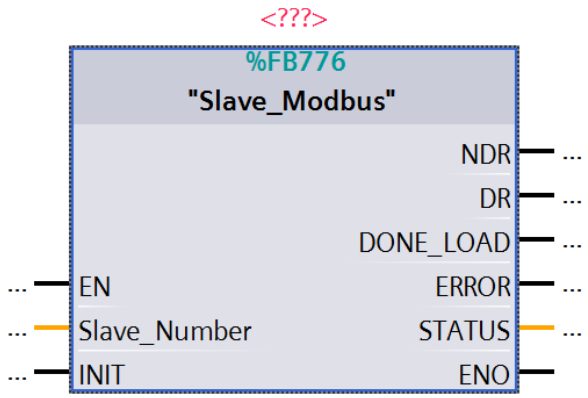


Tabelle 4-7

Parameter	Typ	Anmerkung
Data_Slave	IN: Int	Nummer des UDT Data_Slave im Array des DBs **Comm_Data** (siehe 4.3.3)
INIT	IN: Bool	Durch den Aufruf mit INIT = TRUE werden die Parameter aus **Comm_Data** neu übernommen. Muss zu Beginn des Programms einmalig mit INIT=TRUE augerufen werden.
NDR	OUT: Bool	Gibt für einen Zyklus NDR = TRUE aus, wenn der Slave Daten empfangen hat.
DR	OUT: Bool	Gibt für einen Zyklus DR = TRUE aus, wenn der Slave Daten gesendet hat.
DONE_LOAD	OUT: Bool	Gibt für einen Zyklus DONE_LOAD = TRUE zurück, wenn in das Slave-Kommunikationsmodul erfolgreich die Kommunikationseinstellungen geladen wurden.
ERROR	OUT: Bool	ERROR = TRUE, wenn ein Fehler im Baustein ansteht.
STATUS	OUT: DWord	STATUS des Bausteins. Für nähere Informationen siehe unten.

Ausgangsparameter: STATUS

Der Ausgangsparameter STATUS setzt sich aus zwei Wörtern zusammen:

Tabelle 4-8

Status	Beschreibung
High Word	Zeigt an, wo im FB **Slave_Modbus** der Fehler aufgetreten ist: • 16#0001: Im Aufruf **Modbus_Comm_Load** des Slaves • 16#0002: Im Aufruf **Modbus_Slave** des Slaves
Low Word	Nimmt den Wert des Status der Anweisung an, an der der Fehler aufgetreten ist. Wenn die Stationsadresse (MB_ADDR) oder der Port mit 0 angegeben ist, dann steht der Wert 16#FFFE an.

4.3.2 Bausteindetails

Übersicht

Der FB **Slave_Modbus** initialisiert ein Kommunikationsmodul als Modbus-Slave.

Programmcode

Abbildung 4-8

```
(*************************************************************
  Analyse the parameters PORT and MB_ADDR of the 'Slave'-Array
  lock functions if the PORT or the MB_ADDR = 0
*************************************************************
IF ("Comm_Data".Slave[#Slave_Number].MB_ADDR= 0) OR
  ("Comm_Data".Slave[#Slave_Number]."PORT"= 0) THEN
  #control.lock :=1;
ELSE
  #control.lock:=0;
END_IF;

                                        1.
                                        ⋮

// call of the Modbus_Comm_Load: Configuration of the communication parameters
#Modbus_Comm_Load_Instance(REQ :=#control.req_comm,
                           MB_DB:=#Modbus_Slave_Instance.MB_DB,
                           "PORT":="Comm Data".Slave[#Slave Number]."PORT",
                  2.       ⋮

//call of Modbus_Slave
#Modbus_Slave_Instance(MB_HOLD_REG:="Comm_Data".Slave[#Slave_Number].Slave_Data[0],
                       MB_ADDR:="Comm_Data".Slave[#Slave_Number].MB_ADDR,
                  3.       ⋮


ELSIF  #INIT = TRUE THEN
       (*************************************************************************
         Initialisation for the Instructions Modbus_Comm_Lo    d Modbus_Slave;
         Reset of the Counter;                                    4.
       *************************************************************************
```

Beschreibung

Durch den zyklischen Aufruf des FB **Slave_Modbus** wird der folgende Ablauf realisiert.

Abbildung 4-9

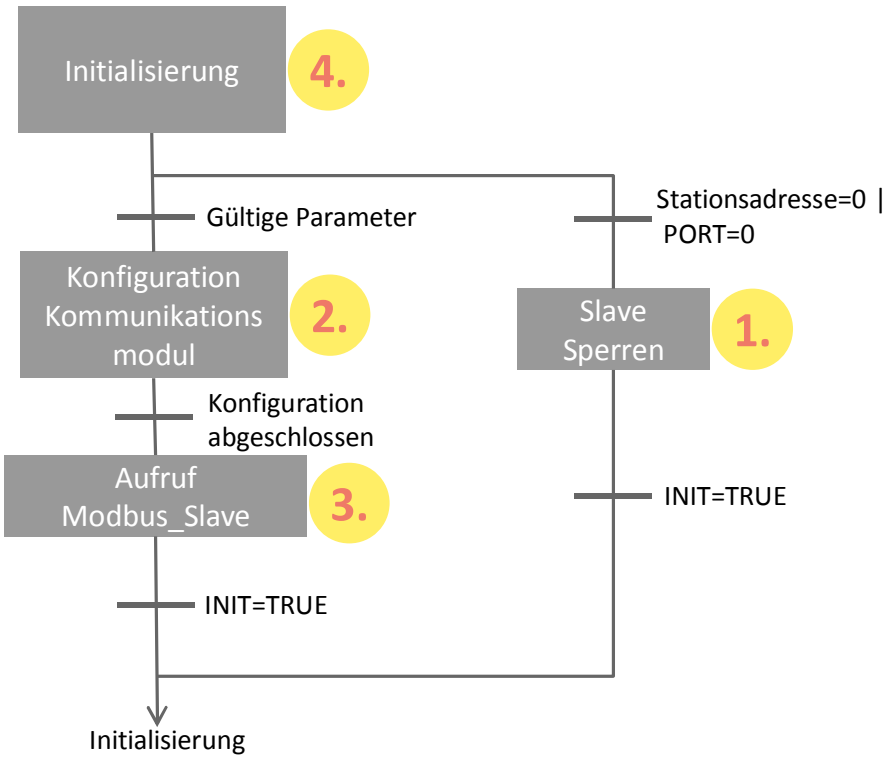


Tabelle 4-9

Nr.	Vorgang	Anmerkung
1.	Sperren des Slaves	Keine Konfiguration des Kommunikationsmoduls, weil PORT oder MB_ADDR=0.
2.	Konfiguration des Kommunikationsmoduls.	Aufruf von **Modbus_Comm_Load** mit den Parametern aus **Comm_Data**.
3.	Einrichten des Kommunikationsmoduls als Modbus-Slave und warten auf Telegramme des Modbus-Masters.	Aufruf von **Modbus_Slave**. Verweis auf den Slave-Puffer (Comm_Data.Slave[X].Slave_data).
4.	Initialisierung	• Zurücksetzen des REQ-Eingangs • Zurücksetzen der Fehler- und Erfolgszähler • Zurücksetzen des Status

Für Einzelheiten des Programmcodes schauen Sie bitte im Beispielprojekt nach.

Hinweis

Ein Kommunikationsmodul soll jeweils nur mit einem Modbus_Comm_Load initialisiert werden.

Pro Modbus_Comm_Load kann nur ein Modbus_Master oder ein Modbus_Slave aufgerufen werden.

4.3.3 Der UDT Data_Slave

Übersicht

Der UDT Data_Slave enthält die für den FB **Slave_Modbus** zur Einrichtung der Kommunikation mit einem Modbus-Master relevanten Informationen.

Am Eingang Slave_Number wird dem FB **Slave_Modbus** mitgeteilt auf welches Element des Arrays „Slave“ im DB **Comm_Data** der Baustein zugreifen soll.

Aufbau

Abbildung 4-10

Data_Slave

	Name	Datentyp	Kommentar
	PORT	HW_SUBMODULE	port of the communication module
	MB_ADDR	UInt	station address
▶	Slave_Data	Array[0..7] of Word	data buffer of the slave

Verwendung

Im Beispielprojekt DB **Comm_Data** ist ein Array „Slave“ aus zwei UDTs Data_Slave vorhanden, die unter anderem die Parameter PORT und MB_ADDR für den FB **Slave_Modbus** enthalten.

Wenn Sie weitere Aufrufe des FB **Slave_Modbus** programmieren wollen, dann können Sie an das Array im DB **Comm_Data** weitere Elemente anhängen. Ihrem Aufruf des FBs müssen Sie dann die Nummer des neuen Array-Elements übergeben.

Für weitere Informationen beachten Sie bitte Kapitel 5.2.

4.4 Der DB Comm_Data

Übersicht

Im DB **Comm_Data** sind für die FBs **Master_Modbus** und **Slave_Modbus** Daten abgelegt, die diese zur Modbus RTU-Kommunikation benötigen.

Aufbau

Abbildung 4-11

Comm_Data

	Name	Data type	Comment
	Static		
	Param	Struct	communication parameter, e.g. baudrate, etc.
	PublicParam	Struct	public datablock data for com_modbus_load
	Master_comm	array [1..2] of "Data_for_Master"	Array for FB Master_Modbus
	Slave	array[1..2] of "Data_Slave"	Array for call of FBs Slave_Modbus

Verwendung

Tabelle 4-10

Name	Datentyp	Verwendung	Anmerkung
Param	Struct	Parameter zur Einstellung der Kommunikation mit der Anweisung **Modbus_Comm_Load**	Die Parameter werden sowohl im FB **Master_Modbus** als auch im FB **Slave_Modbus** verschaltet.
PublicParam			
Master_comm	array[1..2] of „Data_for_Master“	Zur Verwendung des UDTs Data_for_Master siehe Kapitel 4.2.5 und Kapitel 0.	Parameter werden im FB **Master_Modbus** verwendet.
Slave	array [1..2] of „Data_Slave“	Zur Verwendung des UDTs Data_Slave siehe Kapitel 4.3.3. und Kapitel 0.	Parameter werden im FB **Slave_Modbus** verwendet.

5 Konfiguration und Projektierung

Überblick

Wenn Sie Änderungen am STEP 7-V12 Projekt vornehmen wollen, dann bietet Ihnen dieses Kapitel Unterstützung.

Die folgenden Anpassungsmöglichkeiten sind dokumentiert:

- Ändern von Kommunikationseinstellungen, wie zum Beispiel der Baudrate am Mobus-Master und an den beiden Modbus-Slaves
- Ändern der bestehenden Kommunikationsaufträge
- Hinzufügen weiterer Slaves in das Programm.
- Anpassen der Empfangspuffergröße, um Daten größer als acht Wörter zu senden oder zu empfangen.

5.1 Ändern der Kommunikationseinstellungen

Überblick

Im DB Comm_Data (DB778) sind die Daten für die Kommunikationseinstellungen abgelegt.

Diese Parameter können Sie abändern. Wenn Sie einen Modbus-Slave mit festen Kommunikationseinstellungen haben, dann müssen Sie die Einstellungen Ihres Modbus-Masters an diese Einstellungen anpassen.

Sowohl der FB **Master_Modbus** als auch der FB **Slave_Modbus** greifen auf diese Parameter zu.

Achten Sie darauf nur Parameter einzustellen, die von Ihren Geräten unterstützt werden.

Vorgehen

Tabelle 5-1

Nr.	Vorgehen	Anmerkung

1.	Passen Sie die Werte für den DB Comm_Data im OB100 Ihren Anforderungen an.
	Nutzen Sie zur Bedeutung der einzelnen Werte die Hilfefunktion des TIA Portals. (Hilfe zur Anweisung **Modbus_Comm_Load**)
		`//Set the communication parameter at startup`
		`"Comm_Data".Param.Baud := 16#2580;`
		`"Comm_Data".Param.Flow_Ctrl := 0;`
		`"Comm_Data".Param.Parity := 0;`
		`"Comm_Data".Param.Rts_on_dly := 0;`
		`"Comm_Data".Param.Rts_off_dly := 0;`
		`"Comm_Data".Param.Resp_to := 16#2710;`
		`"Comm_Data".Param.Startup := 1;`
		
		`"Comm_Data".PublicParam.Ichar_gap := 0;`
		`"Comm_Data".PublicParam.Mode := 4;`
		`"Comm_Data".PublicParam.Line_pre := 0;`

2.	Kompilieren Sie ihr Projekt und laden Sie es in die CPU.	

5.2 Ändern der bestehenden Kommunikationsaufträge

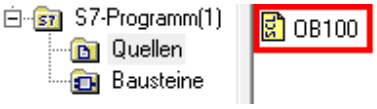
Überblick

Das Beispielprojekt enthält zwei Kommunikationsaufträge, auf Grund derer der Modbus Master abwechselnd von den beiden Modbus Slaves jeweils 8 Worte an Daten liest.

Das Kapitel beschreibt, wie sie die Parameter für die Kommunikationsaufträge ändern.

Vorgehen

Tabelle 5-2

<table>
<tr><th>Nr.</th><th>Vorgehen</th><th>Anmerkung</th></tr>
<tr><td>1.</td><td>Öffnen Sie den OB 100.</td><td> 
</td></tr>
<tr><td>2.</td><td>Passen Sie die Parameter des DB Comm_Data an Ihre Anforderungen an.</td><td>
<table>
<tr><th>Name</th><th>Bedeutung</th></tr>
<tr><td>MB_ADDR</td><td>Die Modbus Stationsadresse des Slaves.</td></tr>
<tr><td>MODE</td><td>Gibt die Art der Anforderung an.</td></tr>
<tr><td>DATA_ADDR</td><td rowspan="2">MODE, DATA_ADDR und DATA_LEN ergeben zusammen die genaue Anweisung, welche Daten der Modbus Master empfangen oder senden soll.</td></tr>
<tr><td>DATA_LEN</td></tr>
</table>
<pre>"Comm_Data".Master_comm[3].MB_ADDR   :=   2;
"Comm_Data".Master_comm[3].MODE   :=   0;
"Comm_Data".Master_comm[3].DATA_ADDR   :=   400001;
"Comm_Data"."Master_comm"[3]."DATA_LEN" :=   8;</pre>
</td></tr>
<tr><td>3.</td><td>Übersetzen Sie den DB, laden Sie ihn in Ihre CPU und starten Sie die CPU neu.</td><td>  </td></tr>
</table>

Hinweis Die Daten, die der Master vom Slave empfängt oder an den Slave sendet liegen im DB Comm_Data im Puffer des entsprechenden Kommunikationsauftrags (Array Master_comm).

5.3 Hinzufügen eines weiteren Slaves beziehungsweise Kommunikationsauftrags

Überblick

Wenn Sie mit mehr als den hier projektierten zwei Slaves kommunizieren möchten, dann sind Änderungen im Beispielprojekt vorzunehmen.

Beschreibung

Der UDT Data_for_Master enthält

- für den FB **Master_Modbus** relevante Informationen zur Kommunikation mit einem Modbus Slave.
- für den FB **Slave_Modbus** relevante Informationen zum Einrichten der Modbus RTU-Kommunikation als Slave.

Anhand des Eingangs No_Slaves wird dem FB **Master_Modbus** übermittelt, mit wie vielen Slaves er kommunizieren soll. Für die Kommunikation mit jedem Slave muss im DB **Comm_Data** ein UDT im Array Master_comm angelegt sein.

Anhand des Eingangs Slave_Number wird dem FB **Slave_Modbus** mitgeteilt, auf welches Element des Arrays „Slave“ er zugreifen muss, um die relevanten Daten zur Kommunikation als Modbus Slave zu erhalten.

Sollen von einem Slave Daten sowohl gesendet als auch empfangen werden, so empfiehlt es sich dafür den Array Master_comm um einen weiteren Auftrag zu erweitern.

In Tabelle 5-3 ist aufgelistet, welche Parameter Sie für die Kommunikation mit einem Slave einstellen müssen.

Abbildung 5-1

Data_for_Master

	Name	Datentyp	Kommentar
	MB_ADDR	UInt	station address of the slave
	MODE	USInt	modus: read or write
	DATA_ADDR	UDInt	specifies the data addresse where to read the data
	DATA_LEN	UInt	specifies the data length
	LOCK	Bool	slave locked -> no communication
	ERROR	Bool	error at communication with slave
	STATUS	Word	status of that error
▶	buffer	array[0..7] of Word	buffer for the data of or for the slave

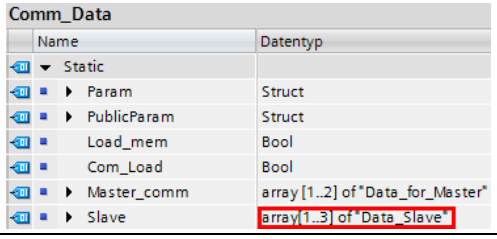
Tabelle 5-3

Variable	Funktion	Anmerkung
MB_ADDR	Die Modbus Stationsadresse des Slaves.	Der Slave muss zwingend dieselbe Modbus Stationsadresse besitzen.
MODE	Gibt die Art der Anforderung an.	MODE=0 entspricht dem Lesen von Daten
DATA_ADDR	MODE, DATA_ADDR und DATA_LEN ergeben zusammen die genaue Anweisung, welche Daten der Modbus Master empfangen oder senden soll.	Genauere Infos entnehmen Sie bitte der Hilfe zur Anweisung Modbus_Master im TIA Portal
DATA_LEN		

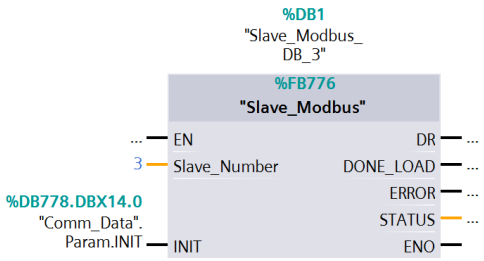
Hinweis Anhand der Parameter ERROR und STATUS können Sie den Zustand der Kommunikation zum jeweiligen Slave auslesen.

Vorgehen Slave

Tabelle 5-4

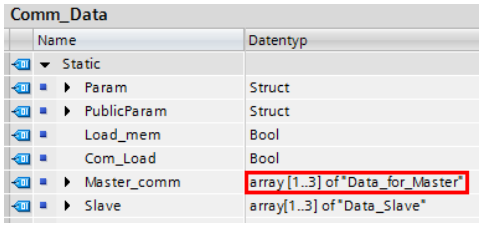
Nr.	Vorgehen	Anmerkung
1.	Fügen Sie im DB **Comm_Data** an das Array Slave ein weiteres Element an.	

2.	Fügen Sie im OB100 weitere Programmzeilen hinzu, in denen Sie die Parameter MB_ADDR und PORT des angefügten Array-Elements belegen.	Den Parameter PORT entnehmen Sie aus der Hardware-Konfiguration Ihres Slaves.
		`"Comm_Data".Slave[3].MB_ADDR := 2;`
		`"Comm_Data".Slave[3]."PORT" := "CM_PtP_1[AI]";`

3.	Rufen Sie in einem zyklischen OB den FB **Slave_Modbus** auf und übergeben Sie beim Aufruf am Eingang Slave_Number die Nummer des von Ihnen angefügen Array-Elements. Verschalten Sie am Eingang INIT den Parameter „Comm_Data“.Param.INIT. Jetzt ist nach dem Laden des Projektes in die CPU der Slave bereit zur Kommunikation mit dem Master.	

Vorgehen Master

Tabelle 5-5

Nr.	Vorgehen	Anmerkung
1.	Fügen Sie im DB **Comm_Data** an das Array Master_Comm ein weiteres Element an.	

2.	Fügen Sie im OB100 weitere Programmzeilen hinzu, in denen Sie die Parameter
	• MB_ADDR (identisch mit der Stationsadresse unter 2.)
	• MODE
	• DATA_ADDR
	• DAT_LEN
	des angefügten Array-Elements belegen.
		Genauere Informationen zur Bedeutung der einzelnen Parameter entnehmen Sie bitte der Hilfe zur Anweisung **Modbus_Master** im TIA Portals.
		
		`"Comm_Data".Master_comm[3].MB_ADDR := 2;`
		`"Comm_Data".Master_comm[3].MODE := 0;`
		`"Comm_Data".Master_comm[3].DATA_ADDR := 400001;`
		`"Comm_Data"."Master_comm"[3]."DATA_LEN" := 8;`

3.	Inkrementieren Sie im OB1 den Eingangsparameter No_Slaves um 1.	
4.	Kompilieren Sie ihr Projekt und laden Sie es in die CPU. Jetzt arbeitet Ihr Modbus Master einen weiteren Kommunikationsauftrag ab bzw. kommuniziert mit einem Slave mehr.	

5.4 Anpassen der Empfangspuffer

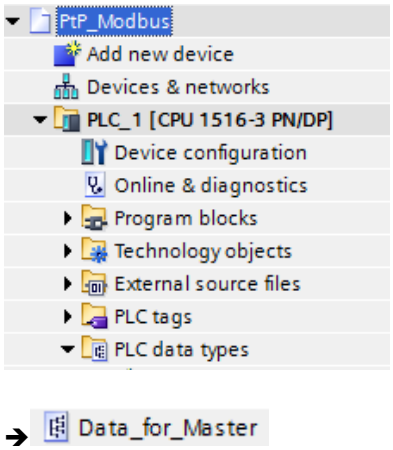
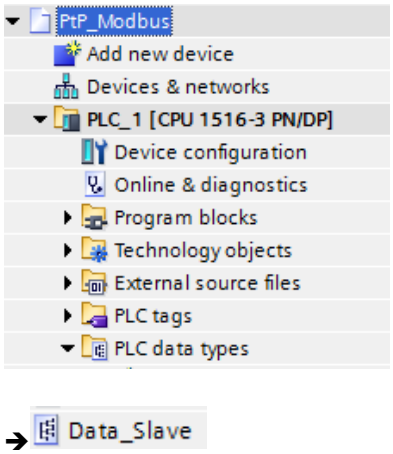
Überblick

Die Beispielapplikation liest mit einer Anforderung acht Wörter von einem Slave.

Wenn Sie größere Datenmengen lesen oder schreiben möchten, dann müssen Sie zum einen Änderungen, wie in Kapitel 5.2 beschrieben, vornehmen und zum anderen die verwendeten Puffer vergrößern

Vorgehen

Tabelle 5-6

Nr.	Aktion	Anmerkung
1.	Navigieren Sie in der Projektnavigation zu „PtP_Modbus> PLC_1> PLC-Datentypen> Data_for_Master“ („PtP_Modbus> PLC_1> PLC data types> Data_for_Master“) und öffnen Sie den Datentyp.	
2.	Passen Sie den Array „buffer“ auf die von Ihnen gewünschte Größe an.	
3.	Navigieren Sie in der Projektnavigation zu „PtP_Modbus> PLC_1> PLC-Datentypen> Data_Slave“ („PtP_Modbus> PLC_1> PLC data types> Data_Slave“) und öffnen Sie den Datentyp.	
4.	Vergrößern Sie den vorhanden Array Slave_Data auf denselben Wert wie die Arrays unter 2.	
5.	Kompilieren Sie das Projekt und laden Sie es neu in die SIMATIC S7-1500 Steuerung.	

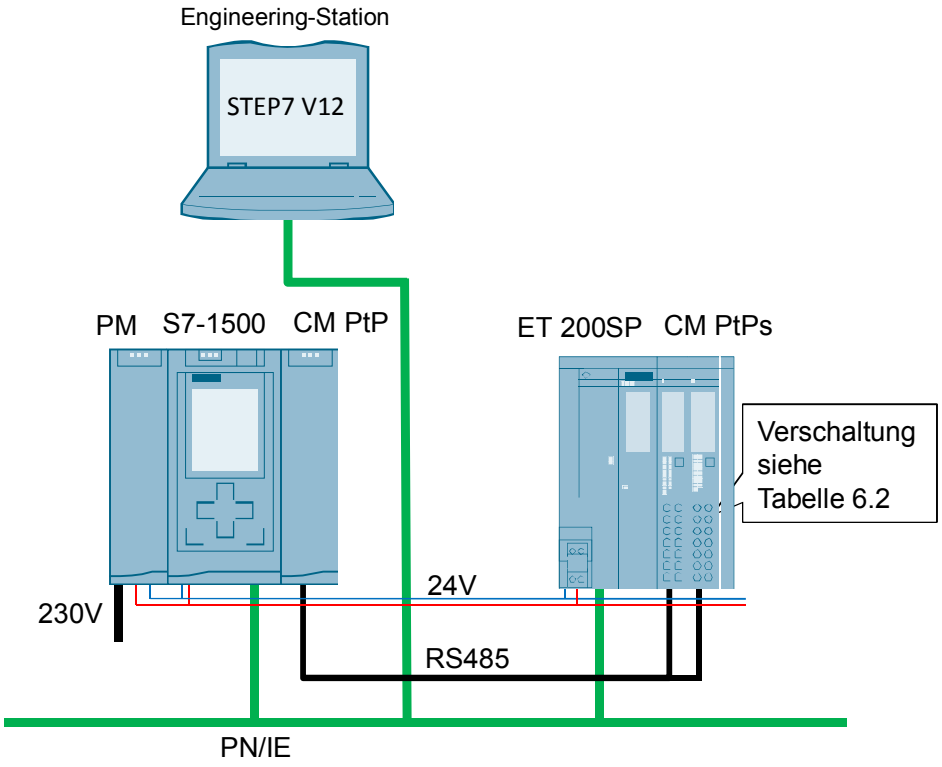
6 Inbetriebnahme der Applikation

6.1 Aufbau der Hardware

Übersicht

Nachfolgendes Bild zeigt den Hardwareaufbau des Beispiels.

Abbildung 6-1



Die folgenden Tabellen beschreiben das Vorgehen für den Hardwareaufbau des Projektes.

Beachten Sie dabei die Vorschriften für den Aufbau einer S7-Station.

Hardwareaufbau der SIMATIC S7-1500 Station

Tabelle 6-1

Nr.	Vorgehen	Anmerkung
1.	Stecken Sie das Power Supply und die CPU auf den entsprechenden Bauträger	
2.	Verdrahten Sie die CPU mit dem Power Supply.	Achten Sie auf die richtige Polung!
3.	Verbinden Sie Ihren Power Supply mit dem Stromnetz (230V Wechselstrom)	

Nr.	Vorgehen	Anmerkung
4.	Schließen Sie die CPU per Ethernet an Ihre Engineering-Station mit TIA-Portal V12 an.	
5.	Stellen Sie die IP-Adresse des Ports der S7-1500 über das Display auf die im Beispiel verwendete IP-Adresse ein (192.168.0.1). Die IP Adresse können Sie unter Einstellungen> Adressen> X1 (IE/PN) („Settings > Addresses >X1 (IE/PN) > IP address“) im Display einstellen.	Die Engineering Station sollte sich zum Laden in die Steuerung im selben Subnetz befinden.

Hardwareaufbau des ET 200SP

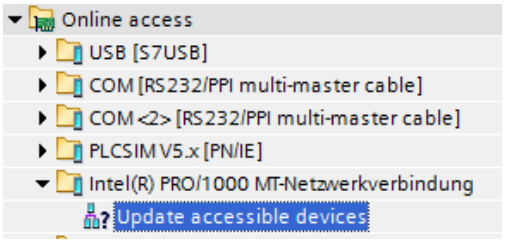
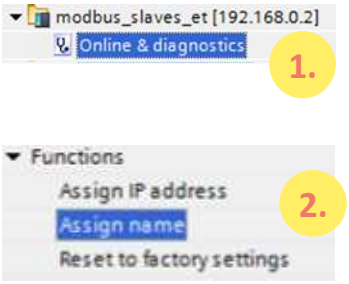
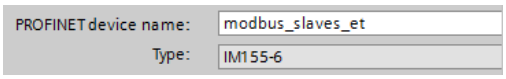
Tabelle 6-2

<table>
<tr><th>Nr.</th><th>Vorgehen</th><th>Anmerkung</th></tr>
<tr><td>1.</td><td>Stecken Sie das Kopfmodul sowie die CM PtPs mit Base Unit und als Abschluss das Server Modul auf eine Hutschiene.</td><td>Die Anweisungen aus dem Handbuch \6\ sind zu beachten!</td></tr>
<tr><td>2.</td><td>Verbinden Sie das Kopfmodul per Ethernet Kabel mit der SIMATIC S7-300.</td><td></td></tr>
<tr><td>3.</td><td>Verbinden Sie die CM PtPs des ET 200SP untereinander und mit dem CM PtP der SIMATIC S7-1500.
Die Belegung der Base Unit entnehmen Sie der Beschreibung auf der Front des CM PtP.</td><td>Anschlussbelegung Zweidraht-Betrieb:
<table>
<tr><th>Pin</th><th>Bezeichnung</th><th>Bedeutung</th></tr>
<tr><td>12</td><td>T(A)/R(A)</td><td>Empfangs-/Sendedaten</td></tr>
<tr><td>14</td><td>T(A)/R(A)</td><td>Empfangs-/Sendedaten</td></tr>
<tr><td>15+16</td><td>PE Ground</td><td>GND Betriebserde (potentialfrei)</td></tr>
</table>
Hinweis!
Ab einer Länge von 50 Metern benötigt Ihr Modbus-Bus zwei Abschlusswiderstände.</td></tr>
<tr><td>4.</td><td>Schließen Sie das ET 200SP an die Spannungsversorgung des Power Supply an.</td><td></td></tr>
</table>

6.2 Konfiguration der Hardware

Konfiguration der ET 200SP

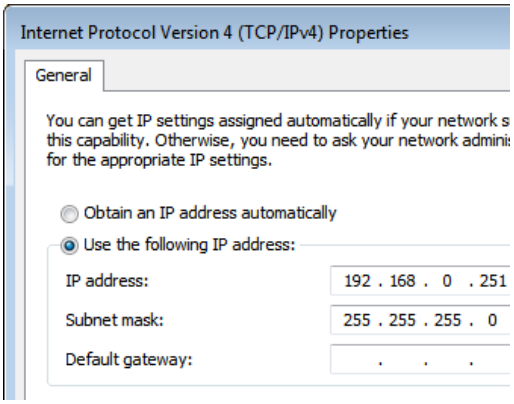
Tabelle 6-3

Nr.	Vorgehen	Anmerkung
1.	Öffnen Sie das TIA Portal V12 in der Projektsicht. Suchen Sie nach „Erreichbaren Teilnehmern“. Navigieren Sie dazu in „Projektnavigation> Online-Zugänge> [Ihr_Ethernet_Adapter]> Erreichbare Teilnehmer aktualisieren“ („Project Tree> Online Access> [Your_Ethernet_Adapter]> Update accessible devices“) Ihre ET 200SP-Station wird nun erkannt.	
2.	Navigieren Sie nun zu „[Ihre_ET200SP_Station]> Online&Diagnose“ („[Your_ET200SP_Station]> Online&Diagnostics“) Im grafischen Bereich von „Online&Diagnose“ wählen Sie nun „Funktionen> Name zuweisen“ („functions> assign name“)	
3.	Geben Sie den folgenden, im Projekt verwendeten, Namen in das Eingabefeld ein: modbus_slaves_et Bestätigen Sie die Aktion mit „Name zuweisen“ („Assign name“) Die ET 200SP Station erhält nun den PROFINET-Namen von Ihrer Engineering Station zugewiesen.	

6.3 Öffnen und Laden des STEP 7-Projekts

Die folgende Tabelle zeigt Ihnen, wie Sie das STEP 7-Projekt öffnen und in Ihre S7-Station laden.

Tabelle 6-4

Nr.	Vorgehen	Anmerkung
1.	Entzippen Sie die Datei „68202723_S7-1500_ModbusRTU_CODE_V1d0_d.zip“ in einen lokalen Ordner ihres PCs.	
2.	Navigieren Sie in den erstellten Ordner. Öffnen Sie das STEP 7-Projekt mit einem Doppelklick auf die Datei „PtP_Modbus.ap12“. Nun wird das TIA-Portal mit diesem Projekt geöffnet.	
3.	Stellen Sie sicher, dass sich Ihre Engineering Station im selben Subnetz wie die S7-1500 CPU befindet. Beispiel: IP-Adresse: 192.168.0.251 Subnetzmaske: 255.255.255.0	
4.	Übersetzen Sie das Projekt über "S7-1500 > Übersetzen" („S7-1500> Compile“) oder über das entsprechende Symbol. Im Inspektorfenster erscheint die Meldung, dass die Übersetzung erfolgreich durchgeführt wurde.	
5.	Nach der fehlerfreien Übersetzung laden Sie über den Button „Laden in Gerät“ die Projektierung in Ihre S7-1500 CPU. Nach dem Download erscheint die Meldung, dass der Ladevorgang erfolgreich beendet wurde.	

Laden des Projekts in die CPU

Die folgende Tabelle zeigt Ihnen, wie Sie das STEP 7-Projekt in die CPU laden.

Tabelle 6-5

Nr.	Vorgehen	Anmerkung

7 Bedienung der Applikation

7.1 Beobachten

Übersicht

Wenn Sie die das Beispielprojekt in Betrieb genommen haben, dann arbeitet Ihre CPU das Anwenderprogramm zyklisch ab.

Dabei werden Daten mit einer Länge von 8 Worten von den Slaves aus den Arrays „Comm_Data“.Slave[1].slave_data und „Comm_Data“.Slave[2].slave_data gelesen.

Abgelegt werden die vom Master gelesenen Daten im Array „Comm_Data“.Master_comm[1].buffer oder „Comm_Data“.Master_comm[2].buffer.

Um die Aktionen des Anwenderprogramms besser beobachten zu können, steht Ihnen die Beobachtungstabelle Modbus_Overview zur Verfügung.

Beobachtungstabelle Modbus_Overview

Die nachfolgende Tabelle zeigt Ihnen, welche Informationen Sie der Beobachtungstabelle entnehmen können.

Für Ihr eigenes Projekt können Sie die Beobachtungstabelle anpassen.

Tabelle 7-1

Nr.	Variable	Anmerkung
	Master_Modbus	
1	[…].stat_save_comm	Wenn ein Error an der Anweisung **Modbus_Comm_Load** auftritt, so wird der Wert des Status hier gespeichert.
2	[…].done_count_gen	Zählt die Anzahl der erfolgreichen Aufrufe von Anweisungen im FB.
3	[…].err_count_gen	Zählt die Anzahl der Errormeldungen der Anweisungen im FB.
4	[…].STATUS	Output Parameter STATUS.
5	[…].INIT	Input Parameter INIT. Mit „Comm_Data“.Param.INIT verschalten.
6	[…].step	Zeigt, in welchem Schritt der Schrittkette sich der FB befindet.
7	[…].number	Zeigt an, mit welchem Slave im Array von **Comm_Data** momentan kommuniziert wird/werden soll.
	Slave_Modbus	
8	[…].stat_save	Wenn ein Error an einer Anweisung des Slave1 auftritt, wird der Status gespeichert.
9	[…].stat_save	Wenn ein Error an einer Anweisung des Slave2 auftritt, wird der Status gespeichert.
10	[…].err_count_gen	Zählt die Anzahl der Errormeldungen der Anweisungen des Slave1.
11	[…].err_count_gen	Zählt die Anzahl der Errormeldungen der Anweisungen des Slave2.
12	**Comm_Data**	
13	[…].MB_ADDR	Array Master_Comm, Slave1: Modbus Stationsadresse
14	[…].STATUS	Array Master_Comm, Slave1: Bei Error gespeicherter Status
15	[…].buffer[0]	Array Master_Comm, Slave1: Gelesene Daten des Slaves werden hier abgelegt.
16	[…].MB_ADDR	Array Master_Comm, Slave2: Modbus Stationsadresse
17	[…].STATUS	Array Master_Comm, Slave2: Bei Error gespeicherter Status

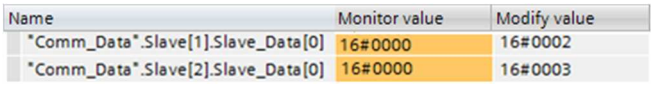
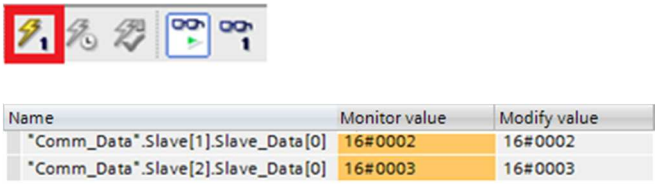
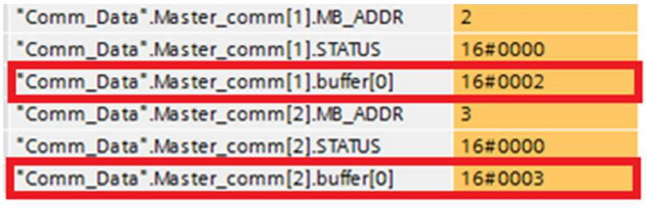
Nr.	Variable	Anmerkung
18	[…].buffer[0]	Array Master_Comm, Slave2: Gelesene Daten des Slaves werden hier abgelegt.
19	[…].INIT	Wenn INIT=TRUE, dann werden die FBs **Slave_Modbus** und **Master_Modbus** initialisiert. Der Baustein setzt nach Ende der Initialisiation die Variable zurück.
20	[…].Slave_data[0]	Erstes Wort des Puffers von Slave1.
21	[…].Slave_data[0]	Erstes Wort des Puffers von Slave2.

7.2 Datenlesen aus dem Modbus Slave zum Modbus Master

In diesem Kapitel wird beschrieben, wie Sie Daten von den Slaves zum Master transportieren können.

Das Beispielprogramm liest Daten aus den Modbus Slaves in den Modbus Master.

Tabelle 7-2

Nr.	Vorgehen	Anmerkung
1.	Nehmen Sie die Applikation, wie in Kapitel 6 beschrieben, in Betrieb.	
2.	Öffnen Sie die Beobachtungstabelle Modbus_Overview und wählen Sie die Option „Alle beobachten“ („Monitor all“).	 Nun sehen Sie die Aktualwerte der Beobachtungstabelle. Wenn Sie die Applikation erfolgreich in Betrieb genommen haben, dann wird die Variable „done_count_gen“ beständig inkrementiert.
3.	Tragen Sie für die Slaves einen beliebigen Wert in die Spalte Steuerwert („Modify value“).	 Die Werte entsprechen in der Abbildung den Stationsadressen der Slaves.
4.	Durch Klicken auf den Button „Steuert alle aktivierten Werte „einmalig und sofort““ (Modify all values once and now“) werden die Werte für die Slaves übernommen	
5.	Durch das Abarbeiten des Beispielprojektes liest nun der Master die eingetragenen Daten von den Slaves und legt Sie in den Puffer ab.	

8 Literaturhinweise

Internet-Link-Angaben

Diese Liste ist keinesfalls vollständig und spiegelt nur eine Auswahl an geeigneten Informationen wieder.

Tabelle 8-1

	Themengebiet	Titel
\1\	Referenz auf den Beitrag	http://support.automation.siemens.com/WW/view/de/68202723
\2\	Siemens Industry Online Support	http://support.automation.siemens.com
\3\	CM PtP-Konfigurationen für Punkt-zu-Punkt-Kopplungen	http://support.automation.siemens.com/WW/view/de/59057093
\4\	S7-1500 Kommunikations modul CM PtP RS422/485 HF	http://support.automation.siemens.com/WW/view/de/59061372
\5\	ET 200SP Kommunikations modul CM PtP	http://support.automation.siemens.com/WW/view/de/59061378
\6\	ET 200SP Dezentrales Peripheriesystem ET 200SP	http://support.automation.siemens.com/WW/view/de/58649293

9 Historie

Tabelle 9-1

Version	Datum	Änderung
V1.0	03/2013	Erste Ausgabe