Master-Slave Kommunikation mit Modbus RTU Protokoll für S7-300 und ET 200S Systeme

S7-300, CP 341, ET 200S 1SI

https://support.industry.siemens.com/cs/ww/de/view/109474714

Siemens Industry Online Support

Gewährleistung und Haftung

Hinweis

Die Anwendungsbeispiele sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit hinsichtlich Konfiguration und Ausstattung sowie jeglicher Eventualitäten. Die Anwendungsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern sollen lediglich Hilfestellung bei typischen Aufgabenstellungen bieten. Sie sind für den sachgemäßen Betrieb der beschriebenen Produkte selbst verantwortlich. Dieses Anwendungsbeispiel enthebt Sie nicht der Verpflichtung zu sicherem Umgang bei Anwendung, Installation, Betrieb und Wartung. Durch Nutzung dieses Anwendungsbeispiels 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 diesem Anwendungsbeispiel jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in diesem Anwendungsbeispiel 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 Anwendungsbeispiel 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.

Security-hinweise

Siemens bietet Produkte und Lösungen mit Industrial Security-Funktionen an, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen.

Um Anlagen, Systeme, Maschinen und Netzwerke gegen Cyber-Bedrohungen zu sichern, ist es erforderlich, ein ganzheitliches Industrial Security-Konzept zu implementieren (und kontinuierlich aufrechtzuerhalten), das dem aktuellen Stand der Technik entspricht. Die Produkte und Lösungen von Siemens formen nur einen Bestandteil eines solchen Konzepts.

Der Kunde ist dafür verantwortlich, unbefugten Zugriff auf seine Anlagen, Systeme, Maschinen und Netzwerke zu verhindern. Systeme, Maschinen und Komponenten sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn und soweit dies notwendig ist und entsprechende Schutzmaßnahmen (z.B. Nutzung von Firewalls und Netzwerksegmentierung) ergriffen wurden.

Zusätzlich sollten die Empfehlungen von Siemens zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Industrial Security finden Sie unter http://www.siemens.com/industrialsecurity.

Die Produkte und Lösungen von Siemens werden ständig weiterentwickelt, um sie noch sicherer zu machen. Siemens empfiehlt ausdrücklich, Aktualisierungen durchzuführen, sobald die entsprechenden Updates zur Verfügung stehen und immer nur die aktuellen Produktversionen zu verwenden. Die Verwendung veralteter oder nicht mehr unterstützter Versionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Produkt-Updates informiert zu sein, abonnieren Sie den Siemens Industrial Security RSS Feed unter http://www.siemens.com/industrialsecurity.

Inhaltsverzeichnis

Gewährleistung und Haftung ... 2

1 Aufgabe ... 4

2 Lösung ... 5

2.1 Übersicht Gesamtlösung ... 5
2.2 Verwendete Hard- und Software-Komponenten ... 6

3 Beschreibung des Modbus RTU-Protokolls ... 8

3.1 Funktionsweise Modbus RTU ... 8
3.2 Projektierung in STEP 7 (TIA Portal) ... 10
3.2.1 Projektierung des CP 341 als Modbus Master ... 12
3.2.2 Projektierung des ET 200S 1SI als Modbus Slave ... 14

4 Beschreibung des STEP 7 Programms ... 17

4.1 Übersicht ... 17
4.2 Funktionsweise des FB "Master_Modbus" ... 19
4.2.1 Aufruf des FB "Master_Modbus" ... 19
4.2.2 Initialisierung ... 20
4.2.3 Zyklische Abarbeitung der Kommunikationsaufträge ... 21
4.2.4 PLC-Datentyp "Data_for_Master" ... 23
4.3 Funktionsweise des "Modbus Slaves" ... 24
4.3.1 Aufruf des FB "S_MODB" (FB81) ... 24
4.3.2 Datenbausteine ... 25
4.4 DB "Master_Data" ... 27

5 Konfiguration und Projektierung ... 28

5.1 Ändern der Kommunikationseinstellungen ... 28
5.2 Ändern der bestehenden Kommunikationsaufträge ... 28
5.3 Hinzufügen eines weiteren Kommunikationsauftrags ... 29
5.3.1 Vorgehen Modbus Master ... 31
5.3.2 Vorgehen Modbus Slave ... 31
5.4 Anpassen der Empfangspuffer ... 34

6 Inbetriebnahme des Anwendungsbeispiels ... 35

6.1 Aufbau der Hardware ... 35
6.2 Konfiguration der Hardware ... 37
6.3 Öffnen und Laden des STEP 7-Projekts ... 39

7 Bedienung des Anwendungsbeispiels ... 40

7.1 Beobachten ... 40
7.2 Datenlesen vom Modbus Slave zum Modbus Master ... 41

8 Literaturhinweise ... 42

Internet-Link-Angaben ... 42

9 Historie ... 43

1 Aufgabe

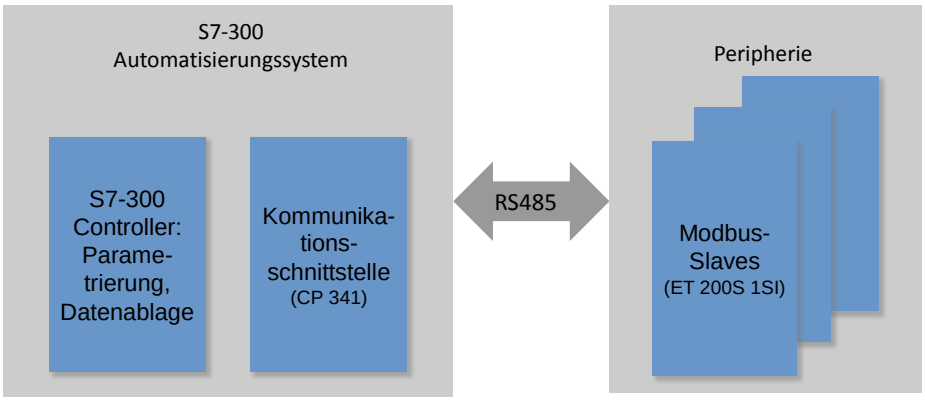
Einleitung

Dieses Anwendungsbeispiel zeigt Ihnen den Umgang mit dem Modbus RTU Protokoll im Automatisierungsumfeld einer S7-300 Station. Sie zeigt Ihnen die Programmierung sowohl eines Modbus Masters via CP 341, als auch eines Modbus Slaves via ET 200S 1SI in einer S7-300 CPU.

Überblick über die Automatisierungsaufgabe

Folgendes Bild gibt einen Überblick über die Automatisierungsaufgabe.

Abbildung 1-1



Beschreibung der Automatisierungsaufgabe

Dieses Anwendungsbeispiel soll folgendes zeigen:

- Projektierung des CP 341 und des ET 200S 1SI für Modbus RTU.
- Erstellung eines flexiblen Modbus-Master/Slave Programms in der S7-300.

2 Lösung

2.1 Übersicht Gesamtlösung

Ziel des Anwendungsbeispiels

Dieses Anwendungsbeispiel zeigt Ihnen:

- Grundlagen zum Modbus RTU-Protokoll
- die Parametrierung eines seriellen Kommunikationsprozessors (CP 341, ET 200S 1SI) für die Kommunikation mit Modbus RTU.
- die flexible Programmierung einer S7-300 CPU mit einem CP 341 als Modbus Master zur Kommunikation mit mehreren Slaves.
- die Programmierung der ET 200S 1SI-Module als Modbus Slave (dezentrale IO-Device vom IO-Controller S7-300) zur Kommunikation mit einem Master.

Im Beispielprojekt liest der CP 341 als Modbus Master abwechselnd von den beiden Slaves (ET 200S 1SI) zwei Halteregister (Datenworte).

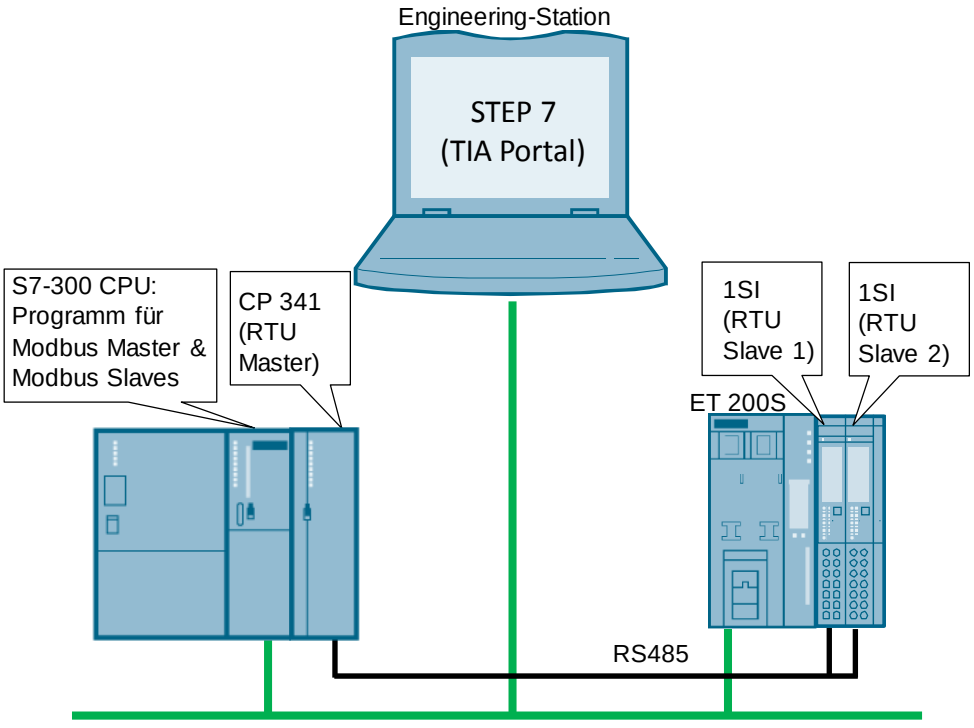
Das für die Verwendung der Funktionscodes 3 und 16 ausgelegte Anwenderprogramm des Masters und der Slaves läuft in der S7-300 CPU ab.

Die genaue Funktionsweise des Programms wird in Kapitel 4 beschrieben.

Schema

Die folgende Abbildung zeigt schematisch die wichtigsten Komponenten der Lösung:

Abbildung 2-1



Vorteile

Das vorliegende Anwendungsbeispiel bietet Ihnen folgende Vorteile:

- schneller Einstieg in das Thema Modbus RTU mit SIMATIC S7-300
- Sie erhalten gekapselte Funktionen zur Programmierung entweder eines Modbus Slaves oder eines Modbus Masters.

Gültigkeit

- Softwareversionen STEP 7 Professional ab V14
- SIMATIC S7-300 CPUs ab V3.2
- CP 341, ET 200S 1SI

Abgrenzung

Dieses Anwendungsbeispiel enthält keine Einführung in das Thema SCL-Programmierung.

Grundlegende Kenntnisse werden vorausgesetzt.

2.2 Verwendete Hard- und Software-Komponenten

Das Anwendungsbeispiel wurde mit den nachfolgenden Komponenten erstellt:

Hardware-Komponenten

Tabelle 2-1

Komponente	Anz.	Bestellnummer	Hinweis
PS 307 2A	1	6ES7 307-1BA01-0AA0	
CPU 315-2 PN/DP	1	6ES7 315-2EH14-0AB0	Firmware V3.2.14 (\9\) Auch andere CPUs aus dem S7-300 Spektrum sind einsetzbar
CP 341-RS422/485	1	6ES7 341-1CH01-0AE0	Firmware V2.1.6 (\12\) Die Baugruppe mit der RS232-Schnittstelle ist nicht zum Aufbau eines Buses geeignet.
IM151-3 PN HF Interfacemodul ET 200S	1	6ES7 151-3BA23-0AB0	Firmware V7.0.5 (\10\) inkl. Abschlussmodul
PM-E DC24V	1	6ES7 138-4CA01-0AA0	
ET 200S 1SI	2	6ES7 138-4DF11-0AB0	Firmware V1.4.0 (\11\)
BaseUnit (hell)	1	6ES7 193-4CC20-0AA0	
BaseUnit (dunkel)	2	6ES7 193-4CB20-0AA0	5 Stück je Verpackungseinheit

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
Modbus Master für CP341/CP441-2	1	6ES7870-1AA00-0YA0	
STEP 7 Professional V15.1 (TIA Portal V15.1)	1	**6ES7822-1..05-..**	

Beispieldateien und Projekte

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

Tabelle 2-3

Komponente	Hinweis
109474714_S7300_ModbusRTU_TIA_PROJ_v2d0_U	Diese Datei enthält das gepackte STEP 7 V15.1 Projekt.
109474714_S7300_ModbusRTU_TIA_DOC_v30_de.pdf	Dieses Dokument.

Für weiterführende Dokumentationen etwa zur dezentralen Peripherie ET 200S 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 S7-300 CPs nicht unterstützt.

Master-Slave Beziehung

Modbus RTU nutzt eine Master/Slave-Beziehung, in der die gesamte Kommunikation von einem einzigen Master-Gerät ausgeht. Die Slaves reagieren lediglich auf die Anforderungen des Masters. 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 das Master-Kommunikationsmodul ein Broadcast-Telegramm an alle Slaves (ohne eine Slave-Antwort zu erhalten).

Kommunikationsablauf

Die Kommunikation mit Modbus RTU läuft 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, wenn Daten angefordert wurden, die Daten bereits enthalten sind oder
3. Wenn der Slave die Anforderung des Masters nicht verarbeiten kann, dann antwortet er 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 soll gelesen werden)		Anzahl Register		---
Gültige Antwort	Slave-Adresse	Funktionscode	Länge	Registerdaten			
Fehlermeldung	Slave-Adresse	0x83	Errorcode	---			

Der Funktionscode bestimmt, welche Funktion ausgeführt werden soll.
Die Tabelle 3-2 listet die Funktionscodes auf, die mit dem CP 341 verwendet werden können:

Tabelle 3-2

Funktionscode	Funktion
01	Ausgangsbit lesen
02	Eingangsbit lesen
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
07	Ereignis-Bits lesen
08	Slave Zustand über Daten-Diagnosecode prüfen/ Slave Ereigniszähler über Daten-Diagnosecode zurücksetzen
11	Statuswort und Ereigniszähler der Slave-Kommunikation lesen
12	Statuswort, Ereigniszähler, Telegramzähler und Eventbytes der Slave-Kommunikation lesen.

Leistungseckdaten

Anzahl Geräte am Bus

Tabelle 3-3

Schnittstelle	Maximale Anzahl an Slaves
RS485*	32
RS422*	10
RS232	1

Jeder Modbus-Slaves muss eindeutig adressiert werden (1..247).

*) Bei Leitungslängen größer 50m müssen Sie für einen störungsfreien Datenverkehr das Bussegment an beiden Enden mit einem Abschlusswiderstand von ca. 330Ω abschließen (\4\).

Datenlänge

Tabelle 3-4

Anweisungstyp	Funktionscodes	Maximale Anzahl pro Anforderung (CP 341)
Bit-Anweisung	1, 2, 5, 15	2040 Bits
Register-Anweisung	3, 4, 6, 16	127 Register (Worte)

Es sind die jeweiligen Obergrenzen der Baugruppen zu beachten.

3.2 Projektierung in STEP 7 (TIA Portal)

Überblick

STEP 7 (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.

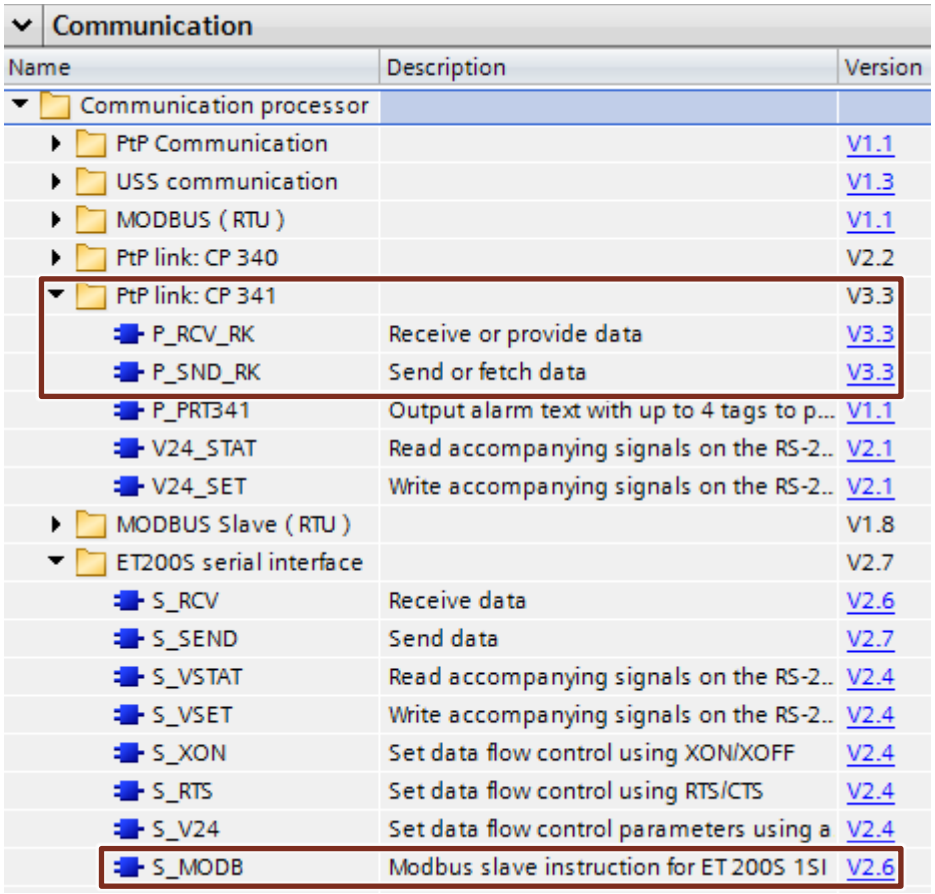
Lizenzierung

Für die Kommunikation als Modbus Master benötigen Sie eine Software Lizenz (Tabelle 2-2) und für Ihren CP 341 den entsprechenden Dongle (\8\).

Kommunikationsbausteine (Anweisungen) für Modbus RTU

Die benötigten Kommunikationsbausteine finden Sie in STEP 7 (TIA Portal) ab V14 in den Anweisungen unter "Kommunikation > Kommunikationsprozessor" ("Communication > Communication processor").

Abbildung 3-1



Folgende Bausteine werden in diesem Anwendungsbeispiel für die Modbus RTU – Kommunikation verwendet:

Tabelle 3-5

Anweisung	Version	Beschreibung
P_RCV_RK (FB7)	V3.3	CP 341 als Modbus Master: Daten empfangen oder Daten bereitstellen Bei lesenden Funktionscodes wird mit dem Funktionsbaustein "P_RCV_RK" (FB7) das Antworttelegramm des Slaves empfangen und die Daten in den an den Parametern "DB_NO" und "DBB_NO" angegebenen Empfangspuffer abgelegt. Der Funktionsbaustein übergibt an einen Slave einen Kommunikationsauftrag, mit den in einem Quell-Datenbereich abgelegten Daten.
P_SND_RK (FB8)	V3.3	CP 341 als Modbus Master: Daten senden oder holen Der Funktionsbaustein übergibt an einen Slave einen Kommunikationsauftrag, mit den in einem Quell-Datenbereich abgelegten Daten.
S_MODB (FB81)	V2.6	Modbus-Slave-Anweisung für ET 200S 1SI Durch die Initialisierung des Funktionsbaustein "S_MODB" wird das 1SI Modul als Modbus Slave eingerichtet und vom Master ankommende Telegramme werden vom FB81 "S_MODB" verarbeitet.
S_RCV (FB2)	V2.3	Realisieren die Kommunikation zwischen CPU und Modul, die FBs werden intern vom FB "S_MODB" verwendet.
S_SEND (FB3)	V2.4	

Die Projektierung der Modbus-Funktionen erfolgt in Abhängigkeit von der verwendeten Baugruppe.

Hinweis

Eine genaue Beschreibung zu den Anweisungen "P_RCV_RK" und "P_SND_RK" erhalten Sie im Handbuch "SIMATIC S7-300/S7-400 Ladbarer Treiber für Punkt-zu-Punkt-CPs: MODBUS-Protokoll, RTU-Format, S7 ist Master" (\15\).

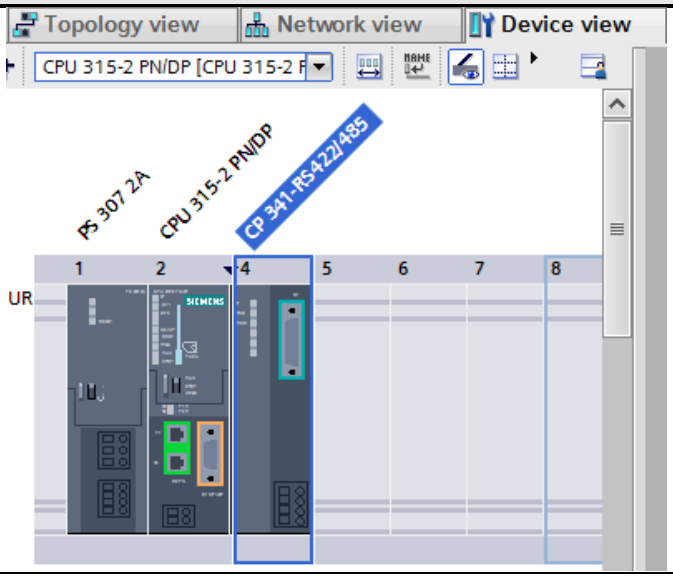
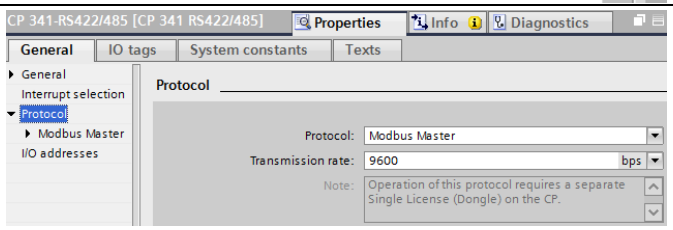
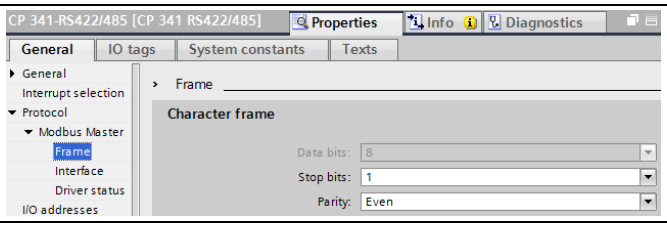
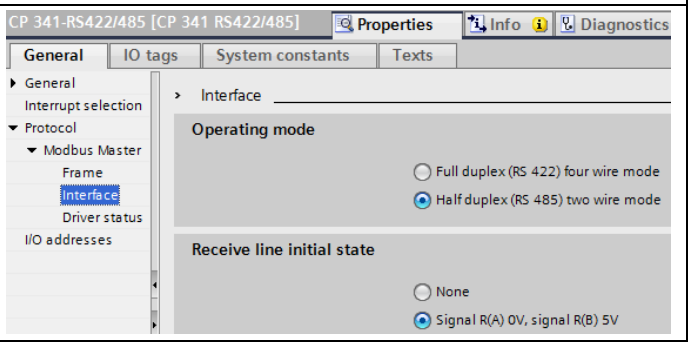
Eine genaue Beschreibung zur Anweisung "S_MODB" erhalten Sie im Handbuch "SIMATIC ET 200S Serielle Schnittstellenbaugruppen" (\7\).

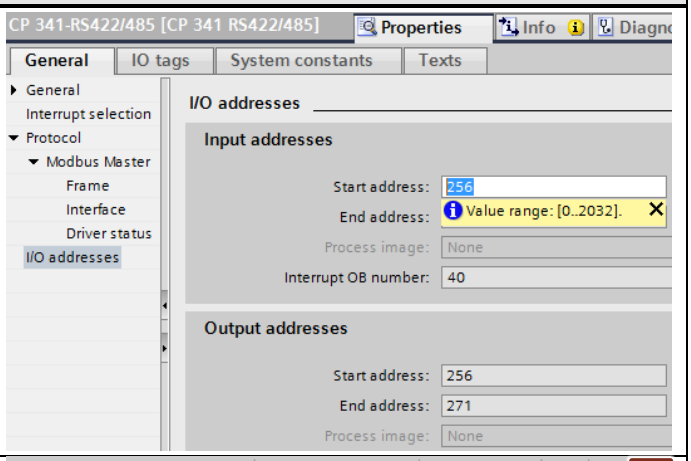
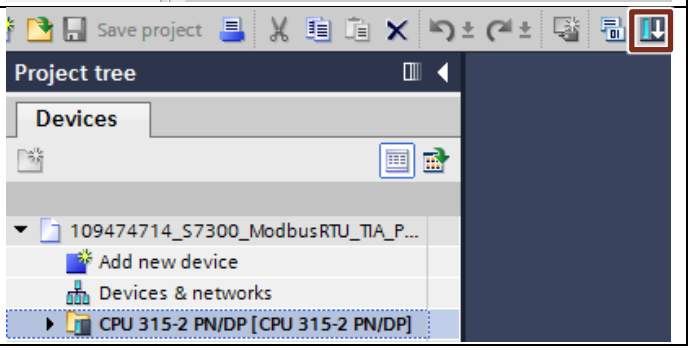
3.2.1 Projektierung des CP 341 als Modbus Master

Hardware-Konfiguration

Wenn Sie den CP 341 als Modbus Master betreiben wollen, dann müssen Sie in der Hardware-Konfiguration folgende Einstellungen vornehmen.

Tabelle 3-6

Nr.	Vorgehen	Anmerkung
1.	Öffnen Sie in Ihrem Projekt die Gerätesicht der CPU 315-2 PN/DP. Markieren Sie den CP 341-RS422/485, und öffnen Sie die Eigenschaften über die Tastenkombination "Alt+Enter".	
2.	Öffnen Sie den Menüpunkt "Protokoll" und wählen Sie das Protokoll "Modbus Master" und die Baudrate "9600" Bit/s.	
3.	Öffnen Sie den Menüpunkt "Telegramm" und wählen Sie "1" Stoppbit und "gerade" Parität.	
4.	Öffnen Sie den Menüpunkt "Schnittstelle" und wählen Sie als Betriebsart "Halbduplex (RS485) Zweidraht-Betrieb" und als Vorbelegung der Empfangsleitung "Signal R(A) 0V, Signal R(B) 5V".	

Nr.	Vorgehen	Anmerkung
5.	Öffnen Sie den Menüpunkt "E/A-Adressen". Die hier angegebene Startadresse der Eingänge wird auch als Startadresse der Ausgänge übernommen und identifiziert den CP im Anwenderprogramm.	
6.	Markieren Sie den Steuerungsordner in der Projektnavigation und laden Sie nun die geänderte Hardwareprojektierung über die Schaltfläche "Laden in Gerät" ("Download to device") in Ihre CPU.	

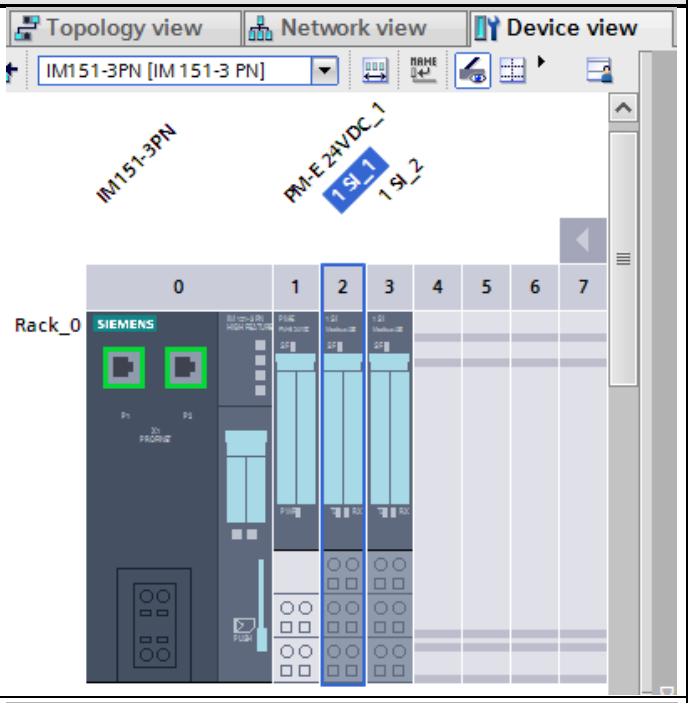
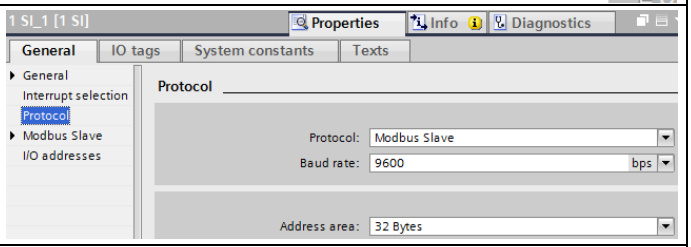
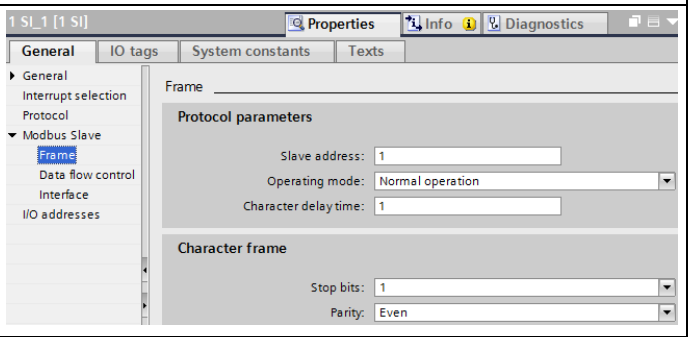
Achten Sie darauf, dass Sie dem Master dieselben Kommunikationseinstellungen wie dem Slave zuweisen!

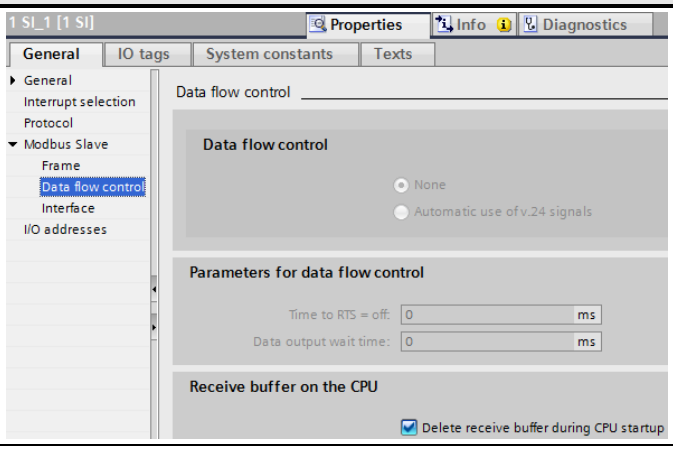
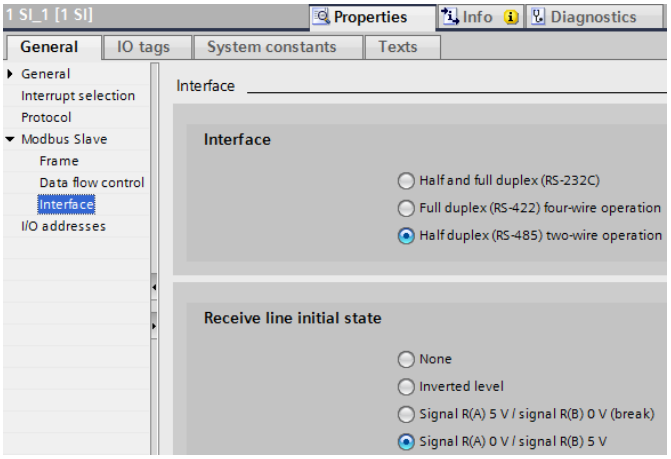
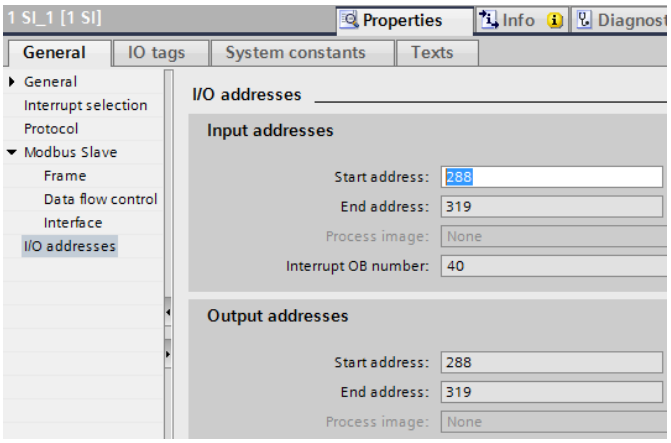
3.2.2 Projektierung des ET 200S 1SI als Modbus Slave

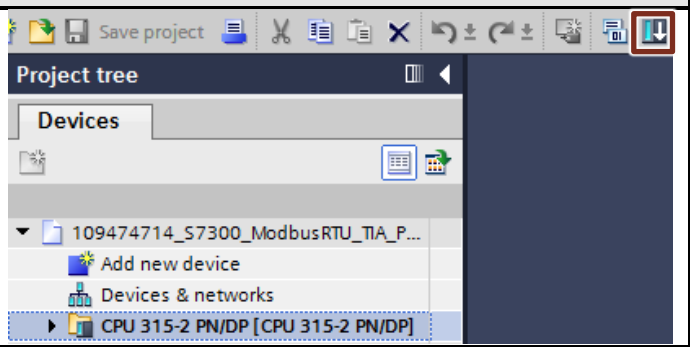
Hardwarekonfiguration

Wenn Sie das Modul ET 200S 1SI als Modbus Slave betreiben wollen, dann müssen Sie in der Hardware-Konfiguration folgende Einstellungen vornehmen.

Tabelle 3-7

Nr.	Vorgehen	Anmerkung
1.	Öffnen Sie in Ihrem Projekt die Gerätesicht des IM151-3PN (ET 200S). Markieren Sie das Modul "1 SI_1", und öffnen Sie die Eigenschaften über die Tastenkombination "Alt+Enter".	
2.	Öffnen Sie den Menüpunkt "Protokoll" und wählen Sie das Protokoll "Modbus Slave" und die Baudrate "9600" Bit/s, sowie den Adressbereich "32 Bytes" aus.	
3.	Öffnen Sie den Menüpunkt "Telegramm" und geben Sie unter "Protokollparameter" die Slave-Adresse dieses Moduls an (hier "1"). Belassen Sie die Voreinstellung: Betriebsart = "Normal-Betrieb" Zeichenverzugszeit = "1" Wählen unter "Zeichenrahmen" Sie "1" Stoppbit und "gerade" Parität aus.	

Nr.	Vorgehen	Anmerkung
4.	Öffnen Sie den Menüpunkt "Datenflusskontrolle" und wählen Sie die Option "Empfangspuffer löschen bei Anlauf der CPU" unter "Empfangspuffer auf CPU".	
5.	Öffnen Sie den Menüpunkt "Schnittstelle" und wählen Sie als Betriebsart "Halbduplex (RS485) Zweidraht-Betrieb" und als Vorbelegung der Empfangsleitung "Signal R(A) 0V, Signal R(B) 5V".	
6.	Öffnen Sie den Menüpunkt "E/A-Adressen". Die hier angegebene Startadresse der Eingänge wird auch als Startadresse der Ausgänge übernommen und identifiziert das Modul im Anwenderprogramm.	
7.	Wiederholen Sie die Schritte 1 bis 6 für das Modul "1 SI_2". Wählen Sie in Schritt 3 als Slave-Adresse "5" für das Modul "1 SI_2".	

Nr.	Vorgehen	Anmerkung
8.	Markieren Sie den Steuerungsordner in der Projektnavigation und laden Sie nun die geänderte Hardwareprojektierung über die Schaltfläche "Laden in Gerät" ("Download to device") in den IO-Controllers der ET 200S.	

4 Beschreibung des STEP 7 Programms

4.1 Übersicht

Funktionen

Das S7-Programm realisiert die folgenden Funktionen

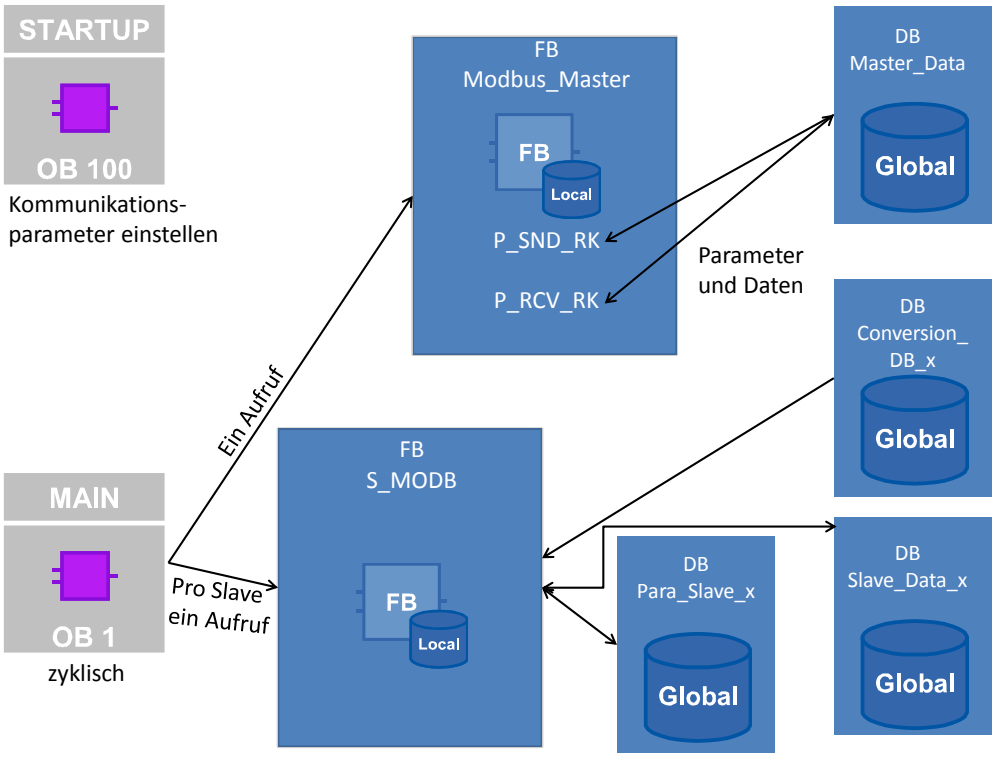
- Kommunikation der S7-CPU als Modbus Master (über den CP 341) zum zyklischen Lesen von je zwei Wörtern aus zwei Modbus Slaves.
- Kommunikation der S7-CPU über die dezentrale Peripherie (ET 200S mit 1SI-Modulen) als Modbus Slave.

Sowohl das Kommunikationsprogramm für den Master als auch das für die Slaves ist in der SIMATIC S7-300 CPU realisiert.

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-V15.1 Projekt verwendet:

Tabelle 4-1

Element	symbolischer Name	Beschreibung	
OB1	CYCLE	Beinhaltet das Hauptprogramm: Ruft den FB **Master_Modbus** (FB5) und pro Slave den FB **S_MODB** (FB81) auf.	Programmaufruf
OB100	RESTART	• Parameter für den Master zur Kommunikation mit den Slaves werden initialisiert. • Parameter für die Slaves werden initialisiert. • Der FB **Master_Modbus** (FB5) wird durch einen Aufruf initialisiert	
FB5	Master_Modbus	Einrichten eines Kommunikationsmoduls als Modbus Master. Der Funktionsbaustein verwaltet alle Kommunikationsaufträge zu einem oder mehreren Modbus Slaves.	Eigenentwicklung
DB4	Master_Data	Beinhaltet Parameter für den FB **Master_Modbus** (FB5) und die Puffer für Kommunikationsaufträge des Masters.	
DB5	I_Master_Modbus	Instanz-DB des FB **Master_Modbus** (FB5)	
DB100 DB200	CONVERSION_DB_x	Wird dem Aufruf des FB **S_MODB** (FB81) mitgegeben und gibt, abhängig vom Funktionscode, an, an welcher Stelle ein- und ausgehende Daten abgelegt werden sollen.	
DB101 DB201	Slave_Data_x	Datenablage für jeweils ein Kommunikationsmodul (1SI).	
DB102 DB202	I_S_MODBx	Instanz-DB des FB **S_MODB** (FB81)	
DB103 DB203	Para_Slave_x	Beinhaltet Parameter für den Aufruf des FB **S_MODB** (FB81)	
PLC-Datentyp	Data_for_Master	Beinhaltet Parameter zum Erstellen eines Modbus-Telegramms.	
	Slave_Para	Beinhaltet Parameter zum Aufrufen des FB **S_MODB** (FB81).	
FB2 FB3	S_RECV_SI S_SEND_SI	Zur Kommunikation zwischen CPU und 1SI Modul der dezentralen Peripherie. Die Funktionsbausteine werden vom FB **S_MODB** (FB81) intern aufgerufen.	Systembausteine
FB7	P_RCV_RK	Kommunikationsanweisung zum Empfangen von Daten als Modbus Master. Wird vom FB **Master_Modbus** (FB5) intern aufgerufen.	
FB8	P_SND_RK	Kommunikationsanweisung zur Kommunikation als Modbus Master. Wird vom FB **Master_Modbus** (FB5) intern aufgerufen.	
FB81	S_MODB	Pro Aufruf: Einrichten eines bereits parametrierten Kommunikationsmoduls als Modbus Slave zur Kommunikation mit einem Modbus Master.	

4.2 Funktionsweise des FB "Master_Modbus"

4.2.1 Aufruf des FB "Master_Modbus"

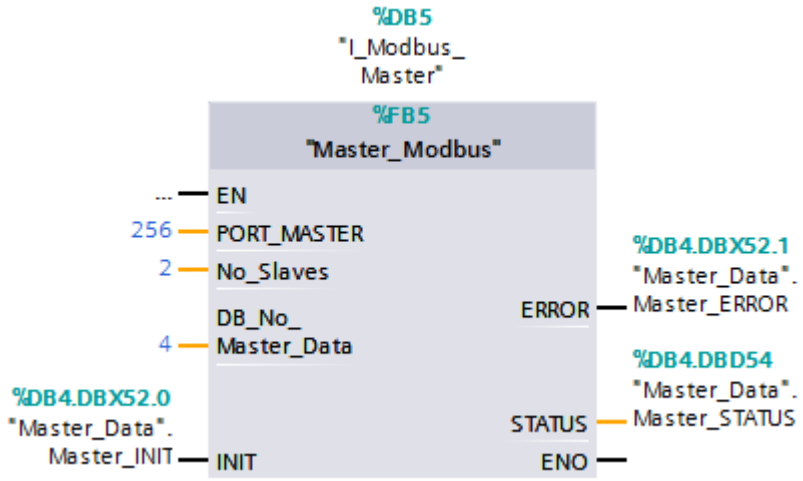
Aufgabe

Der FB "Master_Modbus" verwaltet die Kommunikationsaufträge des CP 341 zu den Modbus Slaves.

Aufruf und Parameter des FB "Master_Modbus"

Die Abbildung 4-2 zeigt die Aufrufschnittstelle des FB "Master_Modbus". Die Parameter werden in Tabelle 4-2 beschrieben.

Abbildung 4-2



Der FB "Master_Modbus" besitzt folgende Ein-und Ausgangsparameter:

Tabelle 4-2

Parameter	Typ	Anmerkung
PORT_MASTER	IN: INT	Anfangsadresse "LADDR" des Master-Kommunikationsmoduls. Kann aus der Hardwarekonfiguration entnommen werden (siehe Tabelle 3-6 Schritt 5).
No_Slaves	IN: INT	Anzahl der aktiven, im DB "Master_Data" (Array "Master_comm") hinterlegten Kommunikationsaufträge.
DB_No_Master_Data	IN: INT	Bausteinnummer des DB "Master_Data".
INIT	IN: BOOL	Für "INIT" = TRUE werden die Kommunikationsaufträge im Array "Master_comm" gesperrt, deren Stationsadressen den Wert Null besitzen. Für "INIT" = FALSE werden die Kommunikationsaufträge im DB "Master_Data" abgearbeitet.
ERROR	OUT: BOOL	"ERROR" = TRUE, wenn ein Fehler im Baustein ansteht.
STATUS	OUT: DWORD	STATUS des Bausteins. Für nähere Informationen siehe Tabelle 4-3.

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. Wenn alle Stationsadressen im DB "Master_Data" den Wert 0 besitzen, dann ist der Wert von "STATUS" = 16#FFFE.

4.2.2 Initialisierung

Übersicht

Der Zustand "INIT" wird durch den Aufruf des FB "Master_Modbus" im OB100 im ersten Zyklus mit "INIT" = TRUE eingeleitet. Im Anwenderprogramm selbst kann der Zustand durch "INIT" = TRUE weitere Male aufgerufen werden.

In diesem Zustand werden für den Programmablauf benötigte Parameter initialisiert.

Beschreibung

Tabelle 4-4

Nr.	Vorgang	Anmerkung
1.	Zurücksetzen des Eingangs "REQ" des FB "P_SND_RK" (FB8).	Es wird sichergestellt, dass am Steuerungseingang 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 0 ist.

4.2.3 Zyklische Abarbeitung der Kommunikationsaufträge

Übersicht

Nach der erfolgreichen Initialisierung der Parameter tritt der FB "Master_Modbus" in den Zustand "Kommunikationsaufträge zyklisch abarbeiten".

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

Programmcode: Aufruf der Kommunikationsbausteine

Abbildung 4-3

```
//calculation of the byte-address of the modbus telegram - information
#control.DBB_NO_S := (#control.number-1)*26+4;
//calculation of the length of the data of the modbus telegram
IF "Master_Data".Master_comm[#control.number].Comm_Param.functioncode = 16#3 THEN
#survey.length2:=6;
ELSIF "Master_Data".Master_comm[#control.number].Comm_Param.functioncode = 16#10 THEN
#survey.length2:=(("Master_Data".Master_comm[#control.number].Comm_Param.reg_number)*2) +6;
END_IF;
//call of the FB8 P_SND_RK
#Master_Instance_S (
                SF :=  'S'// IN: CHAR
               ,REQ := #control.req_master // IN: BOOL
               ,LADDR :=  #PORT_MASTER// IN: INT
               ,DB_NO :=  #DB_No_Master_Data// IN: INT
               ,DBB_NO:= #control.DBB_NO_S // IN: INT
               ,LEN :=  #survey.length2// IN: INT
);
#control.req_master := 1;
#survey.done_master :=     #Master_Instance_S.DONE; // OUT: BOOL
#survey.err_master:=     #Master_Instance_S.ERROR; // OUT: BOOL
#stat :=    #Master_Instance_S.STATUS; // OUT: WORD
//call the FB7 P_RCV_RK TO receive
// the answer of the modbus slave
#control.DBB_NO_R := (#control.number-1)*26+10;
#Master_Instance_R (
                EN_R := 1
               ,LADDR :=  #PORT_MASTER// IN: INT
               ,DB_NO := #DB_No_Master_Data
               ,DBB_NO:= #control.DBB_NO_R
);
#survey.length := #Master_Instance_R.LEN;
#survey.ndr := #Master_Instance_R.NDR;
#survey.errR:=#Master_Instance_R.ERROR;
#survey.statusR := #Master_Instance_R.STATUS;
```

1.

2.

3.

Beschreibung

Tabelle 4-5

Nr.	Vorgang	Anmerkung
1.	Die Eingangsparameter "DBB_NO" und "LEN" für den Aufruf des FB "P_SND_RK" (FB8) werden anhand der aktuellen "control.number" berechnet.	
2.	Der FB "P_SND_RK" (FB8) wird mit den Parametern aus dem aktiven PLC-Datentyp "Data_for_Master" aufgerufen. Als Reaktion auf das gesendete Telegramm sendet der Slave die geforderten Daten an den Master.	Wenn Sie die Aufträge an die Modbus Slaves abändern möchten, dann siehe Kapitel 5.2.
3.	Der FB "P_RCV_RK" (FB7) bewirkt, dass das Antworttelegramm des Slaves empfangen und die darin enthaltenen Daten in den Empfangspuffer für den Slave (im DB "Master_data", Array	Der FB "P_RCV_RK" (FB7) wird immer aufgerufen.

Nr.	Vorgang	Anmerkung
	"Master_comm") abgelegt werden.	

Programmcode: Auswertung der Parameter

Abbildung 4-4

```
//analysis of the output parameters
IF #survey.done_master OR #survey.err_master THEN

    IF #survey.err_master THEN
        //save error and count
        #control.save_number := #control.number;
        #survey.err_count_gen:= #survey.err_count_gen +1;
        #survey.stat_save_comm:=#stat;
        IF #survey.errR THEN
            #survey.status_saveR := #survey.statusR;
        END_IF;
        #control.req_master := 0; //reset the req input of the Modbus_Master instruction
        "Master_Data".Master_comm[#control.number].Diagnostic.ERROR:=1;
        "Master_Data".Master_comm[#control.number].Diagnostic.STATUS:= #stat;
    ELSE
        //save done and count
        //save number for data-transfer
        #survey.done_count_gen := #survey.done_count_gen +1;
        "Master_Data".Master_comm[#control.number].Diagnostic.ERROR:=0;
        "Master_Data".Master_comm[#control.number].Diagnostic.STATUS:= 0;
        #control.req_master:=0;  //reset the req input of the Modbus_Master instruction
    END_IF;
    //after there is an error or a done on the function blocks:
    //change Modbus station address
    IF #control.change = 1 THEN
        IF #control.number < #No_Slaves THEN
            WHILE ("Master_Data".Master_comm[#control.number+1].Diagnostic.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;
END_IF;
```

1.
2.
3.

Beschreibung

Tabelle 4-6

Nr.	Beschreibung
1.	Meldet der FB "P_SND_RK" (FB8) einen Fehler ("ERROR" = TRUE), dann wird der Status gespeichert, ein Error-Zähler inkrementiert und der Request-Eingang zurückgesetzt. Durch das Zurücksetzen des Request-Eingangs wird im nächsten Zyklus ein neuer Kommunikationsauftrag angestoßen.
2.	Meldet der FB "P_SND_RK" (FB8) die erfolgreiche Ausführung ("DONE" = TRUE), dann wird ein Done-Zähler inkrementiert und der Request Eingang zurückgesetzt. Durch das Zurücksetzen des Request-Eingangs wird im nächsten Zyklus ein neuer Kommunikationsauftrag angestoßen.
3.	Sowohl nach einer Fehlermeldung ("ERROR" = TRUE), als auch nach einer Fertigmeldung ("DONE" = TRUE), wird das nächste Element im Array "Master_comm" als aktiv markiert. Mit dem folgenden Zyklus des OB1 beginnt der nächste Kommunikationsauftrag.

4.2.4 PLC-Datentyp "Data_for_Master"

Übersicht

Der PLC-Datentyp "Data_for_Master" enthält die für den FB "Master_Modbus" relevanten Informationen zur Kommunikation mit einem Modbus Slave. Das Array "Master_comm" im DB "Master_Data" besteht aus PLC-Datentypen "Data_for_Master".

Aufbau

Die Tabelle 4-7 zeigt den Aufbau des PLC-Datentyps "Data_for_Master".

Tabelle 4-7

Name	Datentyp	Beschreibung
Diagnostic	Struct	Diagnosestruktur
• LOCK	Bool	Zugriff gesperrt (wenn „Comm_Param.Address“ = 0)
• ERROR	Bool	Fehler des FB8 "P_SND_RK"
• STATUS	Word	Fehler-Status des FB8 "P_SND_RK"
Comm_Param	Struct	Struktur der Kommunikationsparameter
• address	Byte	Modbus-Adresse des Slaves
• functioncode	Byte	Modbus-Funktionscode
• reg_startaddr	Word	Startadresse des Lese/Schreibzugriffs
• reg_number	Int	Anzahl der zu lesenden/schreibenden Register
buffer	Array[0..7] of Word	Empfangs/Sendepuffer

Verwendung

Im Beispielprojekt ist im DB "Master_Data" ein Array aus zwei PLC-Datentypen "Data_for_Master" vorhanden.

Die Parameter

- address
- functioncode
- reg_startaddr
- reg_number

spezifizieren den Auftrag des Masters an den Slave. Genaue Informationen zur Verwendung der Modbus-Funktionscodes finden Sie im Handbuch \3\.

Wenn Sie mit weiteren Slaves kommunizieren, oder andere Datenbereiche lesen/schreiben wollen, dann beachten Sie bitte das Kapitel 5.

4.3 Funktionsweise des "Modbus Slaves"

4.3.1 Aufruf des FB "S_MODB" (FB81)

Aufgabe

Der Funktionsbaustein "S_MODB" (FB81) empfängt das Modbus-Protokoll und setzt die Modbus-Adressen in SIMATIC-Speicherbereiche um.

Aufruf und Parameter

Abbildung 4-5

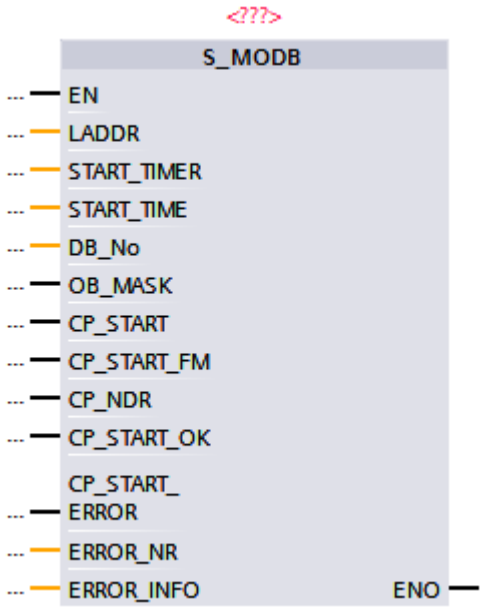


Tabelle 4-8

Parameter	Typ	Anmerkung
LADDR	IN: INT	HW-Kennung "LADDR" des Slave-Kommunikationsmoduls. Kann aus der Hardwarekonfiguration entnommen werden (siehe Tabelle 3-7 Schritt 6).
START_TIMER	IN: Timer	Timer um die Initialisierung des Funktionsbausteins nach einer festgelegten Zeitspanne abzubrechen.
START_TIME	IN: S5TIME	Maximale Zeit, die für die Initialisierung des Bausteins vergehen darf.
DB_No	IN: BLOCK_DB	Nummer des Datenbausteins, der die Konvertierungstabelle enthält ("CONVERSION_DB_x")
OB_MASK	IN: BOOL	Bei TRUE werden Zugriffsfehler im E/A-Bereich maskiert und bei einem Schreibzugriff auf nicht vorhandene Peripherie geht die CPU nicht in STOP.
CP_START	IN/OUT: BOOL	Startet die Initialisierung.
CP_START_FM	IN/OUT: BOOL	Wird vom FB "S_MODB" (FB81) selbst gesetzt. Muss bei der Initialisierung auf 0 gesetzt werden.
CP_NDR	IN/OUT: BOOL	Zeigt an, wenn Daten vom Master in den Slave-Datenbereich geschrieben wurden.

Parameter	Typ	Anmerkung
CP_START_OK	IN/OUT: BOOL	TRUE, wenn die Initialisierung des Funktionsbausteins erfolgreich war.
CP_START_ERROR	IN/OUT: BOOL	TRUE, wenn die Initialisierung des Funktionsbausteins nicht erfolgreich war.
ERROR_NR	IN/OUT: WORD	Fehlernummer.
ERROR_INFO	IN/OUT: WORD	zusätzliche Fehlerinformationen. Weitere Informationen zu "ERROR_NR" und "ERROR_INFO" entnehmen Sie bitte dem Handbuch \7\, Kapitel "Diagnosemeldungen der Funktionsbausteine".

4.3.2 Datenbausteine

Überblick

Im Beispielprojekt werden für den Aufruf des FB "S_MODB" (FB81) die folgenden Datenbausteine benötigt:

- CONVERSION_DB_x
- Slave_Data_x
- I_S_MODBx
- Para_Slave_x

Für den Slave 1 sind das die Datenbausteine 100-103, für den Slave 2 die Datenbausteine 200-203.

DB "I_S_MODBx"

Der Datenbaustein "I_S_MODBx" ist der Instanzdatenbaustein des FB "S_MODB" (FB81).

Der Datenbaustein wird automatisch generiert, sobald der FB "S_MODB" (FB81) aufgerufen wird.

DB "CONVERSION_DB_x"

Der Datenbaustein "CONVERSION_DB_x" teilt dem FB "S_MODB" (FB81) mit, an welche Stelle, abhängig vom verwendeten Funktionscode, die vom Master empfangenen Daten abgelegt werden sollen.

Der Abbildung 4-6 ist der Aufbau des DB "CONVERSION_DB_X" zu entnehmen. Der markierte Eintrag ist für die hier verwendeten Funktionscodes 3 und 16 gültig (ebenso für den hier nicht verwendeten Funktionscode 6).

Relevant sind ebenfalls die Parameter "DB_MIN" und "DB_MAX", da diese den Zugriff auf die Datenbausteine beschränken.

Abbildung 4-6

CONVERSION_DB_1

	Name	Data type	Offset	Start value
1	Static			
2	FC01_MOD_STRT_ADR_1	Word	0.0	16#0
3	FC01_MOD_END_ADR_1	Word	2.0	255
4	FC01_CNV_TO_FLAG_A	Word	4.0	16#0
5	FC01_MOD_STRT_ADR_2	Word	6.0	256
6	FC01_MOD_END_ADR_2	Word	8.0	511
7	FC01_CNV_TO_OUTPUT	Word	10.0	16#0
8	FC01_MOD_STRT_ADR_3	Word	12.0	512
9	FC01_MOD_END_ADR_3	Word	14.0	767
10	FC01_CNV_TO_TIMER	Word	16.0	16#0
11	FC01_MOD_STRT_ADR_4	Word	18.0	768
12	FC01_MOD_END_ADR_4	Word	20.0	1023
13	FC01_CNV_TO_COUNTER	Word	22.0	16#0
14	FC02_MOD_STRT_ADR_5	Word	24.0	16#0
15	FC02_MOD_END_ADR_5	Word	26.0	255
16	FC02_CNV_TO_FLAG_B	Word	28.0	16#0
17	FC02_MOD_STRT_ADR_6	Word	30.0	256
18	FC02_MOD_END_ADR_6	Word	32.0	767
19	FC02_CNV_TO_INPUT	Word	34.0	16#0
20	FC03_06_16_DB_NO	Word	36.0	101
21	FC04_DB_NO	Word	38.0	101
22	DB_MIN	Word	40.0	101
23	DB_MAX	Word	42.0	101
24	FLAG_MIN	Word	44.0	16#0
25	FLAG_MAX	Word	46.0	255
26	OUTPUT_MIN	Word	48.0	16#0
27	OUTPUT_MAX	Word	50.0	255

Im Beispielprojekt werden die Daten der Funktionscodes 3 und 16 in den Datenbaustein "Slave_Data_1" (DB101), nach Vorgabe im DB "CONVERSION_DB_1" oder entsprechend im DB "Slave_Data_2" (DB201), nach Vorgabe im DB "CONVERSION_DB_2" abgelegt.

DB "Slave_Data_x"

Im Datenbaustein "Slave_Data_x" legt der Master die vom Slave (mit Funktionscode 3) empfangenen Daten der Halteregister ab. Diese müssen nach dem Empfang entsprechend weiter verarbeitet werden.

Werden vom Master (mit Funktionsocde 16) Daten in die Halteregister des Slaves geschrieben, so werden diese ebenfalls aus dem DB "Slave_Data_x" übertragen (entsprechend der Vorgabe im DB "CONVERSION_DB_x"). Hierbei müssen Sie die entsprechenden Daten vor der Kommunikation in den DB "Slave_Data_x" eintragen.

DB "Para_Slave_x"

Im DB "Para_Slave_x" sind die Schnittstellenvariablen zur Verschaltung des FBs "S_MODB (FB81)" abgelegt.

Die relevanten Parameter werden beim Anlauf der CPU im OB "RESTART" (OB100) initialisiert.

4.4 DB "Master_Data"

Übersicht

Im DB "Master_Data" sind für den FB "Master_Modbus" (FB5) Daten abgelegt, die dieser zur Modbus RTU-Kommunikation benötigt.

Aufbau

Fehler! Verweisquelle konnte nicht gefunden werden.: Aufbau des DB " Master_Data"

Master_Data

		Name	Data type	Offset
1		Static		
2		Master_comm	Array[1..2] of "Data_for_Master"	0.0
3		Master_INIT	Bool	52.0
4		Master_ERROR	Bool	52.1
5		Master_STATUS	DWord	54.0

Verwendung

Tabelle 4-9

Name	Datentyp	Verwendung	Anmerkung
Master_comm	Array[1..2] of "Data_for_Master"	Zur Verwendung des PLC-Datentyps "Data_for_Master" siehe Kapitel 4.2.4 und Kapitel 5.2.	Parameter werden im FB "Master_Modbus" (FB5) verwendet.
Master_INIT Master_ERROR Master_STATUS	BOOL BOOL WORD	Werden mit den Ein- und Ausgangsparametern des FB "Master_Modbus" (FB5) verschalten.	

5 Konfiguration und Projektierung

Überblick

Wenn Sie Änderungen am STEP 7 (TIA Portal) Projekt vornehmen wollen, dann bietet Ihnen dieses Kapitel Unterstützung.

Die folgenden Anpassungsmöglichkeiten sind dokumentiert:

- Ändern von Kommunikationseinstellungen, wie etwa der Baudrate am Modbus Master und an den beiden Modbus Slaves
- Ändern der bestehenden Kommunikationsaufträge.
- Hinzufügen weiterer Kommunikationsaufträge in das Programm.
- Anpassen der Empfangspuffergröße um Daten größer als 8 Wörter zu senden oder zu empfangen.

5.1 Ändern der Kommunikationseinstellungen

Überblick

Ihr Modbus Master und Ihre Modbus Slaves benötigen identische Einstellungen der Kommunikationsparameter um miteinander kommunizieren zu können.

Die Kommunikationsparameter werden sowohl für den CP 341, als auch für die ET 200S-Module 1SI in den Eigenschaften der jeweiligen Gerätesicht eingestellt.

Vorgehen CP 341 und ET 200S 1SI

Um die Kommunikationsparameter Ihrer Kommunikationsmodule zu ändern, gehen Sie wie folgt vor:

- für den CP 341 siehe Kapitel 3.2.1
- für das Modul 1SI siehe Kapitel 3.2.2

5.2 Ändern der bestehenden Kommunikationsaufträge

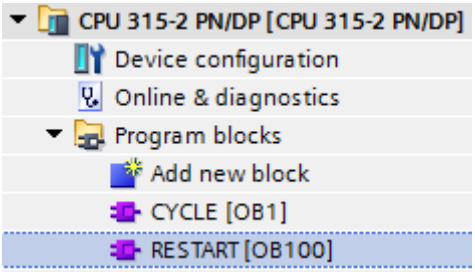
Überblick

Das Beispielprojekt enthält zwei Kommunikationsaufträge, aufgrund derer der Modbus Master abwechselnd von den beiden Modbus Slaves jeweils 2 Worte an Daten zyklisch liest.

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

Vorgehen

Tabelle 5-1

<table>
<tr><th>Nr.</th><th>Vorgehen</th><th>Anmerkung</th></tr>
<tr><td>1.</td><td>Öffnen Sie die den OB "RESTART" (OB100).</td><td> 
</td></tr>
<tr><td>2.</td><td>Passen Sie die Move-Befehle in den ersten beiden Netzwerken an Ihre Anforderungen an. Ändern Sie dadurch die Parameter des Arrays "Master_comm" im DB "Master_Data".

Wenn Sie die Größe der zu lesenden/schreibenden Register abändern möchten, dann beachten Sie Kapitel 5.4.</td><td>
<table>
<tr><th>Name</th><th>Bedeutung</th></tr>
<tr><td>address</td><td>Modbus-Slaveadresse des Slaves, mit dem kommuniziert werden soll</td></tr>
<tr><td>functioncode</td><td>Verwendeter Funktionscode. Das Beispielprojekt ist getestet mit
• 16#3: Daten in Halteregister schreiben.
• 16#10: Daten aus Halteregister lesen.</td></tr>
<tr><td>reg_startaddr</td><td>Startadresse des Lese/Schreibzugriffs</td></tr>
<tr><td>reg_number</td><td>Anzahl der zu lesenden/schreibenden Register</td></tr>
</table>
</td></tr>
<tr><td>3.</td><td>Laden Sie den OB 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 "Master_Data" im Puffer des entsprechenden Kommunikationsauftrags (ein Element des Arrays "Master_Comm").

5.3 Hinzufügen eines weiteren Kommunikationsauftrags

Überblick

Wenn Sie mehr als die zwei vorhandenen Kommunikationsaufträge vom Master abarbeiten lassen wollen, dann sind Änderungen im Beispielprojekt vorzunehmen.

Zu unterscheiden ist,

- ob nur ein weiterer Kommunikationsauftrag zu einem bestehenden Slave gesendet werden soll.
- oder ein zusätzlicher Slave programmiert werden soll, für den dann auch auf Master-Seite ein Kommunikationsauftrag eingerichtet werden muss.

Beschreibung

Anhand des Eingangs "No_Slaves" wird dem FB "Master_Modbus" übermittelt, wie viele Kommunikationsaufträge er abarbeiten soll. Für jeden

Kommunikationsauftrag muss im DB "Master_Data" ein PLC-Datentyp im Array "Master_comm" angelegt sein.

Wenn Sie beispielsweise zusätzlich Daten an einen bestehenden Slave übertragen oder mit einem weiteren Slave kommunizieren wollen, dann empfiehlt es sich dafür, das Array "Master_comm" im DB "Master_Data" um einen weiteren Auftrag zu erweitern.

In Tabelle 5-2 ist aufgelistet, welche Parameter Sie für die Kommunikation mit einem Slave einstellen müssen.

Abbildung 5-1

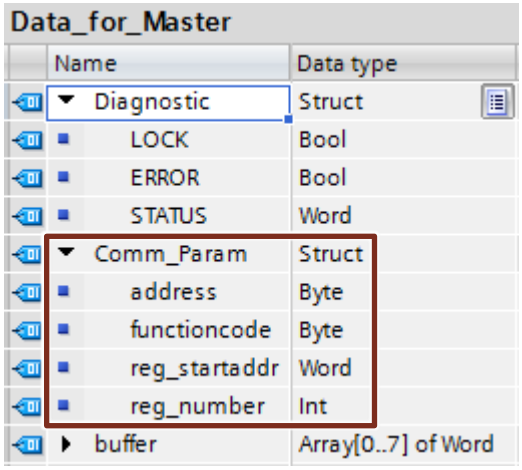


Tabelle 5-2

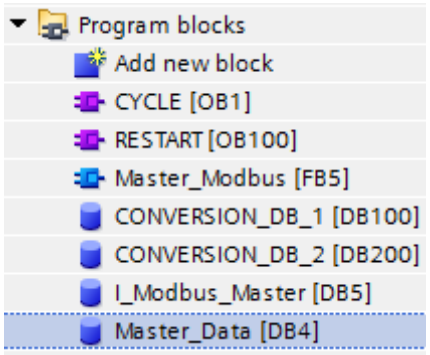
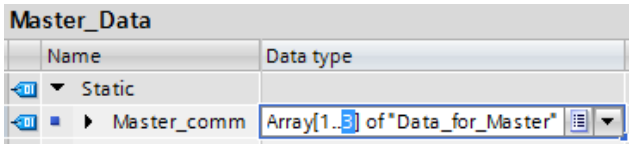
Variable	Funktion	Anmerkung
address	Die Modbus Stationsadresse des Slaves.	Tragen Sie die in der Gerätesicht projektierte Stationsadresse des Slaves ein (siehe Tabelle 3-7 Schritt 3).
functioncode	Gibt die Art der Anforderung an.	Das Beispielprojekt ist für die Verwendung der Funktionscodes 3 und 16 ausgelegt.
reg_startaddr	Gibt an, an welcher Registeradresse mit dem Lesen oder Schreiben begonnen werden soll.	
reg_number	Gibt die Anzahl der Register an, die gelesen oder geschrieben werden sollen.	

Hinweis Anhand der Parameter "ERROR" und "STATUS" der Struktur "Diagnostic" können Sie im Betrieb den Zustand der Kommunikation zum jeweiligen Slave auslesen.

5.3.1 Vorgehen Modbus Master

Die Tabelle 5-3 beschreibt das Vorgehen um einen Kommunikationsauftrag (z.B. zu einem weiteren Slave) an die bestehenden Kommunikationsaufträge anzufügen.

Tabelle 5-3

Nr.	Vorgehen	Anmerkung
1.	Öffnen Sie die den DB "Master_Data".	
2.	Fügen Sie im DB "Master_Data" an das Array "Master_Comm" ein weiteres Element an.	
3.	Fügen Sie im OB100 weitere Netzwerke hinzu, in denen Sie die Parameter • address (Die Stationsadresse Ihres Slaves entnehmen Sie der Hardware-Konfiguration) • functioncode • reg_startaddr • reg_number des angefügten Array-Elements belegen.	Genauere Informationen zur Bedeutung der einzelnen Parameter entnehmen Sie bitte Kapitel 5 oder dem Handbuch \3\.
4.	Inkrementieren Sie die Anzahl der Slaves am Eingangsparameter "No_Slaves" des FB "Master_Modbus" um 1 (im OB1 und im OB100).	
5.	Laden Sie Ihr Projekt in die CPU. Jetzt arbeitet ihr Modbus Master den angefügten Kommunikationsauftrag zusätzlich ab.	

5.3.2 Vorgehen Modbus Slave

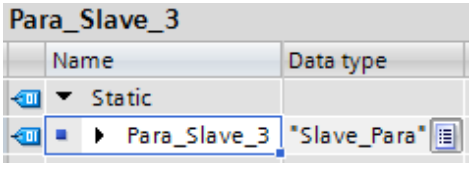
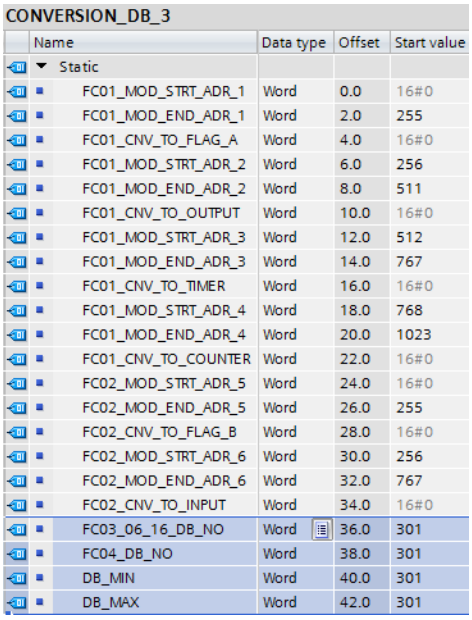
Wenn Sie einen Slave programmieren wollen, dann müssen Sie sowohl ihr Anwenderprogramm, als auch die Hardwarekonfiguration anpassen.

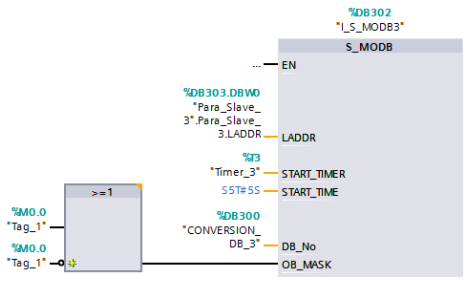
Hardware-Konfiguration

Fügen Sie in der Gerätesicht der ET 200S ein weiteres 1SI Modul hinzu. Zum Einstellen der Parameter des Moduls beachten Sie Kapitel 3.2.2.

Anwenderprogramm

Tabelle 5-4

Nr.	Vorgehen	Anmerkung
1.	• Erstellen Sie einen neuen DB "Para_Slave_x" (z.B. mit der Nummer 303) • Erstellen Sie im DB ein Element vom Datentyp "Slave_Para"	
2.	Fügen Sie im OB100 weitere Netzwerke hinzu, in denen Sie die Parameter • LADDR (die Startadresse der Eingangsparameter entnehmen Sie der Hardware-Konfiguration) • CP_START = 1 • CP_START_FM = 0 des neu erstellten DBs belegen.	
3.	Kopieren Sie einen der DBs "Slave_Data_x" und ändern Sie die Nummer des DBs in den Eigenschaften auf z.B. DB-Nr. 301.	Im Beispielprogramm ist ein Puffer von 2 Worten ausreichend, da keine Sende- oder Empfangsaufträge mit einer größeren Datenmenge abgesetzt werden. Für größere Aufträge siehe auch Kapitel 5.4.
4.	Kopieren Sie einen der DBs "CONVERSION_DB_x" und ändern Sie die Nummer des DBs in den Eigenschaften auf z.B. DB-Nr. 300. Nehmen Sie von Ihnen gewünschte Änderungen bei den Anfangswerten des DBs vor. Wenn Sie nur die Funktionscodes 3 und 16 verwenden, dann müssen Sie • die Variable "FC03_06_16_DB_NO" mit der Nummer ihres Puffer-DBs (siehe Schritt 4) belegen • die Variablen "DB_MIN" und "DB_MAX" so einstellen, dass die Nummer des Puffer-DBs in deren Spanne enthalten ist.	

Nr.	Vorgehen	Anmerkung
5.	Rufen Sie in einem zyklischen OB den FB "S_MODB" (FB81) auf und übergeben Sie beim Aufruf die gleichlautenden Parameter des im DB "Para_Slave_x" (unter Schritt 1) erstellten DBs. Folgende Parameter müssen individuell belegt werden: • START_TIMER Verwenden Sie eine freie Timer-Ressource. • START_TIME Eine Zeit von S5T#5S ist ausreichend. • DB_NO Geben Sie die Nummer des unter 4. erstellten "CONVERSION_DB_x" an. • OB_MASK Maskierung, ist im Beispiel mit TRUE belegt.	
6.	Laden Sie ihr Projekt in die CPU. Jetzt ist der Slave bereit zur Kommunikation mit dem Master.	

5.4 Anpassen der Empfangspuffer

Überblick

Das Anwendungsbeispiel liest mit einer Anforderung zwei Haltregister (Wörter) von einem Slave über die Vorgabe im OB100 „RESTART“ (siehe Tabelle 5-1):

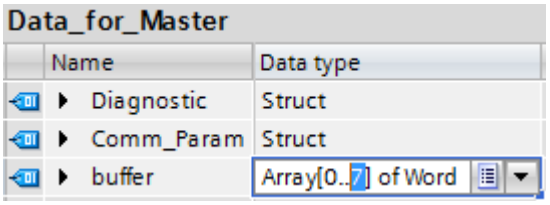
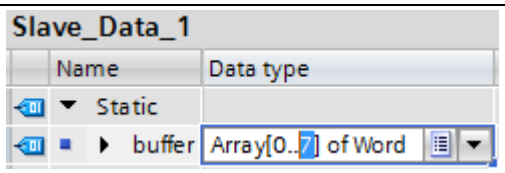
- *„Master_Data“.Master_comm[x].Comm_Param.functioncode* = 3
- *„Master_Data“.Master_comm[x].Comm_Param.reg_number* = 2

Die verwendeten Puffer im Programm besitzen eine Größe von 8 Wörtern.

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 wie folgt vergrößern.

Vorgehen

Tabelle 5-5

Nr.	Aktion	Anmerkung
1.	Öffnen Sie den PLC-Datentyp "Data_for_Master" und ändern Sie das Array "buffer" auf die von Ihnen gewünschte Größe.	
2.	Ändern Sie in den DBs "Slave_Data_x" die Arrays "buffer" auf die in Schritt 1 verwendete Größe.	 Das ET 200S 1SI kann maximal 110 Register pro Auftrag senden oder 109 Register empfangen. Informationen zur maximalen Größe eines Schreib-/ Leseauftrags finden Sie in den Handbüchern \3\, \4\ und \7\ in den Kapiteln zu den Modbus-Funktionscodes.
3.	Laden Sie alle Bausteine in Ihre CPU. Nun können Sie auch größere Sende- und Empfangsaufträge absetzen.	

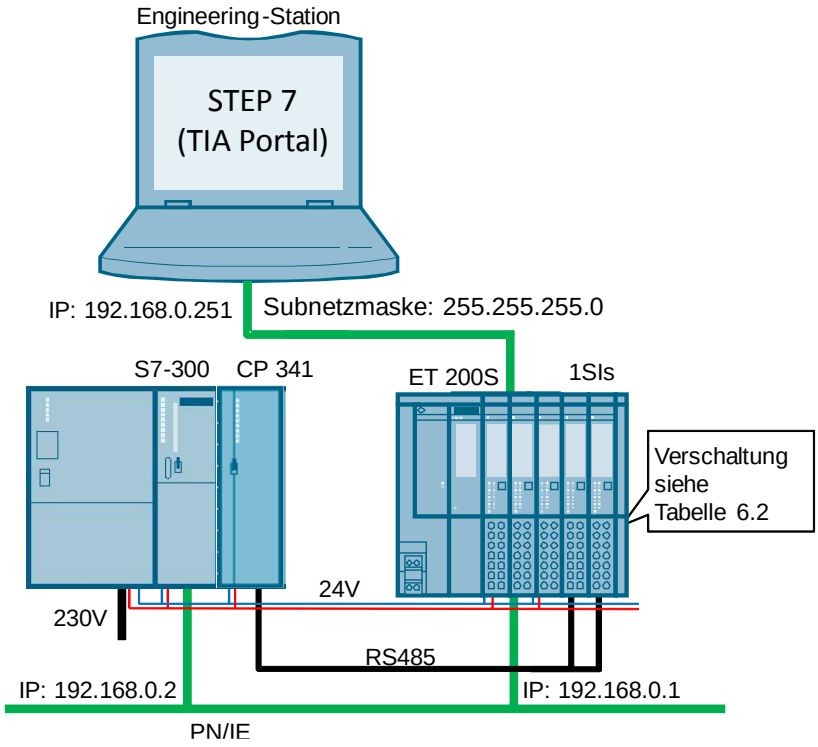
6 Inbetriebnahme des Anwendungsbeispiels

6.1 Aufbau der Hardware

Übersicht

Nachfolgendes Bild zeigt den Hardwareaufbau des Beispiels

Abbildung 6-1



Die Tabellen beschreiben das Vorgehen für den Hardwareaufbau des Projektes. Beachten Sie dabei die Vorschriften für den Aufbau einer S7-Station.

Hardwareaufbau der SIMATIC S7-300 Station

Tabelle 6-1

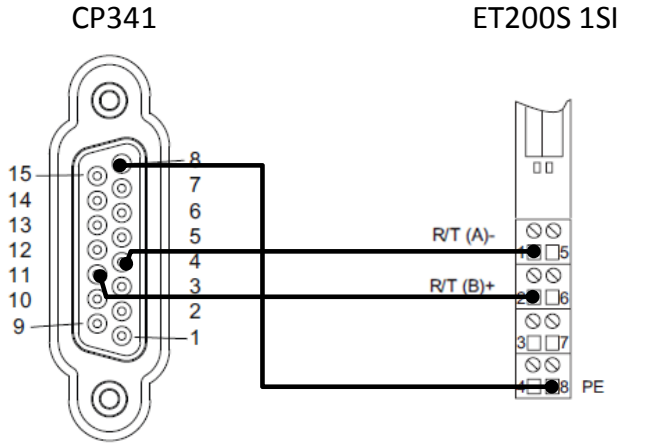
Nr.	Vorgehen	Anmerkung
1.	Stecken Sie das Power Supply, die CPU und den CP auf den entsprechenden Baugruppenträger.	Vergessen Sie nicht den Rückwandbus zwischen CP und CPU zu stecken.
2.	Verdrahten Sie die CPU und den CP mit dem Power Supply.	Achten Sie auf die richtige Polung!
3.	Verbinden Sie Ihren Power Supply mit dem Stromnetz (230V Wechselstrom).	
4.	Schließen Sie die CPU per Ethernet an Ihre Engineering-Station mit STEP 7 (TIA Portal) an.	Eine Verbindung über MPI oder PROFIBUS von Ihrem PG zur CPU ist ebenfalls möglich.

Hardwareaufbau des ET 200SP

Tabelle 6-2

Nr.	Vorgehen	Anmerkung
1.	Stecken Sie das Kopfmodul, das Powermodul sowie die 1SI-Module mit Base Unit und das Abschlussmodul auf eine Hutschiene.	Die Anweisungen aus dem Handbuch \6\ sind zu beachten!
2.	Verbinden Sie das Kopfmodul per Ethernet Kabel mit der SIMATIC S7-300.	
3.	Verbinden Sie die 1SI-Module des ET 200S untereinander und mit dem CP 341 der SIMATIC S7-300 (siehe Abbildung 6-2).	**Hinweis!** Ab einer Länge von 50 Metern benötigt Ihr Modbus-Bus zwei Abschlusswiderstände (\4\).
4.	Schließen Sie das ET 200S an die Spannungsversorgung des Power Supply an.	

Abbildung 6-2



4	R (A)/T (A) -	Eingang Ein-/Ausgang
5	-	-
6	-	-
7	-	-
8	GND	-
9	T (B) +	Ausgang
10	-	-
11	R (B)/T (B) +	Eingang Ein-/Ausgang

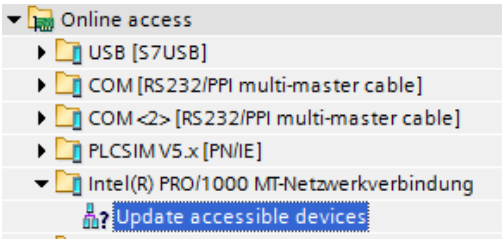
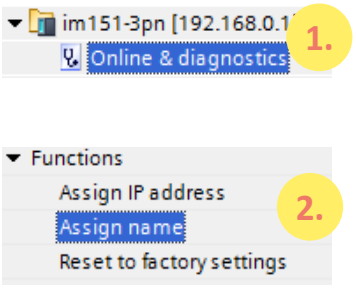
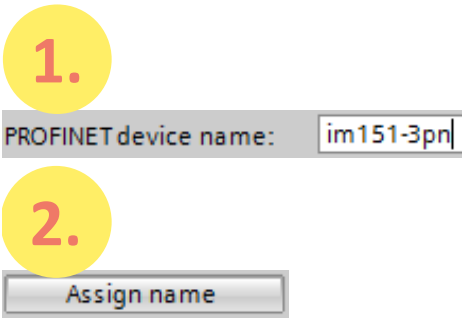
Modus: halbduplex
Klemmen

1	R/T (A)-
2	R/T (B)+
8	PE Erde

6.2 Konfiguration der Hardware

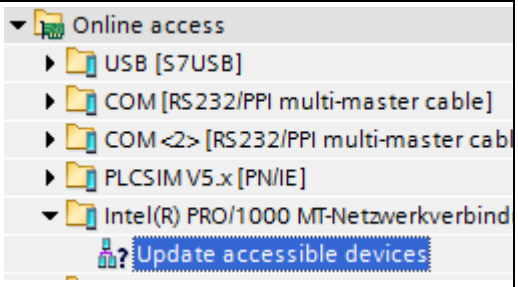
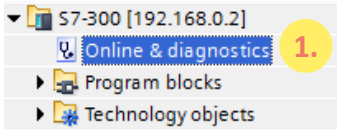
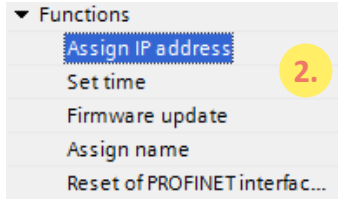
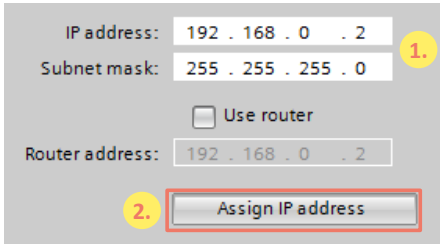
Konfiguration der ET 200S

Tabelle 6-3

1.	Öffnen Sie das TIA Portal V15.1 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_ET 200SP_Station] > Online&Diagnose" ("[Your_ET 200SP_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: *IM151-3PN* Bestätigen Sie die Aktion mit "Name zuweisen" ("Assign name") Die S7-300 Station erhält nun den PROFINET-Namen von Ihrer Engineering Station zugewiesen.	

Konfiguration der SIMATIC S7-300 CPU

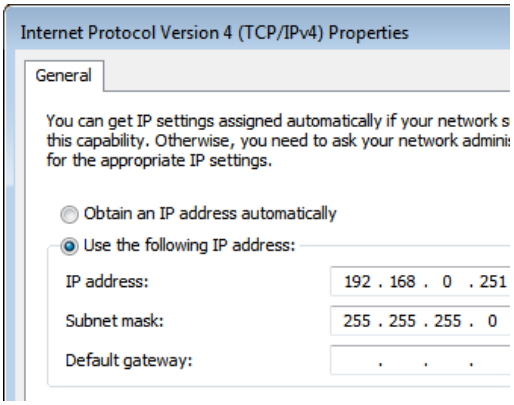
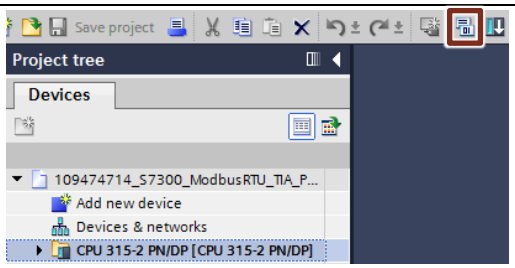
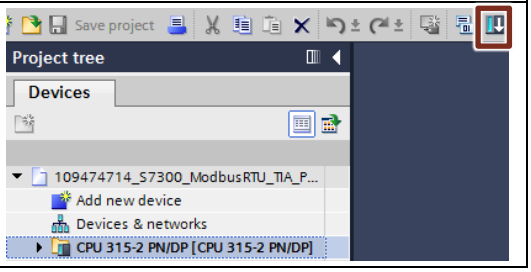
Tabelle 6-4

Nr.	Vorgehen	Anmerkung
1.	Öffnen Sie das TIA Portal V15.1 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 SIMATIC S7-Station wird nun erkannt.	
2.	Navigieren Sie zu "[Ihre_S7_Station] > Online&Diagnose" ("[Your_S7_Station] > Online & diagnostics") Im grafischen Bereich von "Online&Diagnose" wählen Sie nun "Funktionen > IP-Adresse zuweisen" ("Functions > Assign IP address")	 
3.	Geben Sie folgende IP-Adresse und Subnetzmaske in die Eingabefelder: *192.168.0.2.* *255.255.255.0* Bestätigen Sie die Aktion mit "IP-Adresse zuweisen" ("Assign IP address") Die S7-300 Station erhält nun die IP-Adresse 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 6-5

Nr.	Vorgehen	Anmerkung
1.	Entpacken Sie das archivierte Projekt (siehe Tabelle 2-3) für STEP 7 (TIA Portal) 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 mit der Endung "*.apxx" (xx = TIA Portal Version). Nun wird das TIA Portal mit diesem Projekt geöffnet.	
3.	Stellen Sie sicher, dass sich Ihre Engineering Station im selben Subnetz wie die S7-300 CPU befindet. Beispiel: IP-Adresse: 192.168.0.251 Subnetzmaske: 255.255.255.0	
4.	Markieren Sie den Steuerungsordner in der Projektnavigation und übersetzen Sie das Projekt über das entsprechende Symbol. Im Inspektorfenster erscheint die Meldung, dass die Übersetzung erfolgreich durchgeführt wurde.	
5.	Nach der fehlerfreien Übersetzung laden die Projektierung in Ihre S7-300 CPU über die Schaltfläche "Laden in Gerät" ("Download to device") in Ihre CPU. Nach dem Download erscheint die Meldung, dass der Ladevorgang erfolgreich beendet wurde.	

7 Bedienung des Anwendungsbeispiels

7.1 Beobachten

Übersicht

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

Dabei werden Daten mit einer Länge von 2 Worten von den Slaves aus den DBs "Data_Slave_1" und "Data_Slave_2" gelesen.

Abgelegt werden die vom Master gelesenen Daten im Array *"Master_Data".Master_comm[1].buffer* oder *"Master_Data".Master_comm[2].buffer*.

Um die Aktionen des Anwenderprogramms besser zu beobachten 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

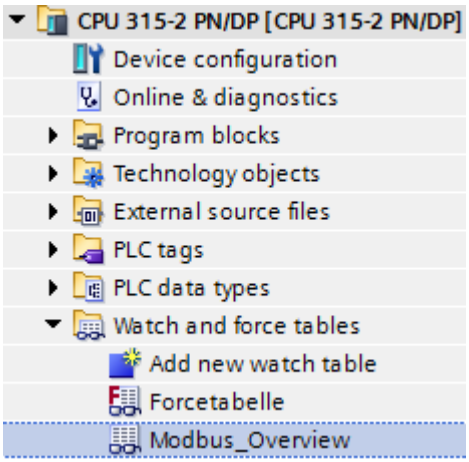
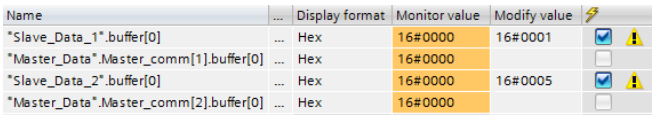
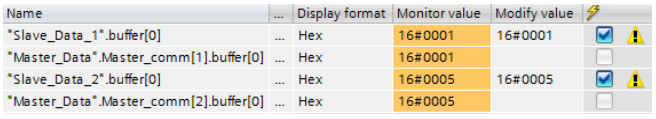
Variable	Anmerkung
Master_Modbus	
[…].stat_save_comm	Wenn ein Error auftritt, so wird der Wert des Status hier gespeichert.
[…].done_count_gen	Zählt die Anzahl der erfolgreichen Aufrufe von Anweisungen im FB "Master_Modbus" (FB5).
[…].err_count_gen	Zählt die Anzahl der Errormeldungen der Anweisungen im FB "Master_Modbus" (FB5).
"I_Master_Modbus".STATUS	Output Parameter "STATUS"
…[1].Diagnostic.STATUS	gespeicherter Parameter Status des 1. Kommunikationsauftrags
…[2].Diagnostic.STATUS	gespeicherter Parameter Status des 2. Kommunikationsauftrags
[…].INIT	Input Parameter "INIT"
[…].number	Zeigt an, mit welchem Slave im Array von "**Master_Data"** momentan kommuniziert wird/werden soll.
Slaves	
…_1.CP_START_OK	Initialisierung Slave1 erfolgreich
…_1.CP_START_ERROR	Initialisierung Slave1 nicht erfolgreich
…_1.ERROR_NR	Slave1: Bei Error gespeicherter Status
…_2.CP_START_OK	Initialisierung Slave2 erfolgreich
…_2.CP_START_ERROR	Initialisierung Slave2 nicht erfolgreich
…_2.ERROR_NR	Slave2: Bei Error gespeicherter Status
Data	
…_1.buffer[0]	Erstes Wort des Puffers für die Kommunikation mit Slave1
…[1].buffer[0]	Erstes Wort des Puffers von Slave1.
…_2.buffer[0]	Erstes Wort des Puffers für die Kommunikation mit Slave2.
…[2].buffer[0]	Erstes Wort des Puffers von Slave2.

7.2 Datenlesen vom 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 das Anwendungsbeispiel in Betrieb (siehe Kapitel 6).	
2.	Öffnen Sie die Beobachtungstabelle "Modbus_Overview" und wählen Sie die Option "Variable beobachten" ("Monitor all").	 Nun sehen Sie die Aktualwerte der Beobachtungstabelle. Wenn Sie das Anwendungsbeispiel 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 "Variable steuern" ("Modify all selected 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

Nr.	Thema
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link auf die Beitragsseite des Anwendungsbeispiels https://support.industry.siemens.com/cs/ww/de/view/109474714
\3\	Updates für STEP 7 V15.1 und WinCC V15.1 https://support.industry.siemens.com/cs/ww/de/view/109763893
\4\	FAQ "Wie schließen Sie RS485/RS422-Schnittstellen von SIMATIC und SIPLUS Baugruppen für die serielle Kommunikation an?" https://support.industry.siemens.com/cs/ww/de/view/109736665
\5\	SIMATIC S7-300 Punkt-zu-Punkt-Kopplung CP 341 Aufbauen und Parametrieren https://support.industry.siemens.com/cs/ww/de/view/1117397
\6\	SIMATIC Dezentrale Peripherie ET 200S Interfacemodul IM151-3 PN (6ES7151-3AA23-0AB0) https://support.industry.siemens.com/cs/ww/de/view/30598131
\7\	SIMATIC ET 200S Serielle Schnittstellenbaugruppen https://support.industry.siemens.com/cs/ww/de/view/9260793
\8\	MODBUS Software-Lizenz für CP 341 / CP 441-2 https://support.industry.siemens.com/cs/ww/de/view/13211560
\9\	Betriebssystem-Updates für CPU 315-2 PN/DP (6ES7315-2EH14-0AB0) https://support.industry.siemens.com/cs/ww/de/view/40360647
\10\	Betriebssystem-Updates für ET 200S IM151-3PN (6ES7151-3..23-0AB0) https://support.industry.siemens.com/cs/ww/de/view/35934244
\11\	Betriebssystem-Updates für ET 200S 1SI (ASCII, Modbus) https://support.industry.siemens.com/cs/ww/de/view/21363754
\12\	Basis-Firmware-Update für CP 341 https://support.industry.siemens.com/cs/ww/de/view/36037679
\13\	Funktionsbausteine, Beispiele und Handbücher der seriellen Schnittstelle ET 200S 1SI https://support.industry.siemens.com/cs/ww/de/view/25358470
\14\	Programmbeispiel ET 200S 1SI MODBUS zXX21_11_1SI_MODBUS.zip für STEP 7 (TIA Portal) https://support.industry.siemens.com/cs/ww/de/view/99742035
\15\	SIMATIC S7-300/S7-400 Ladbarer Treiber für Punkt-zu-Punkt-CPs: MODBUS-Protokoll, RTU-Format, S7 ist Master https://support.industry.siemens.com/cs/ww/de/view/1220184
\16\	SIMATIC S7-300/S7-400 Ladbarer Treiber für Punkt-zu-Punkt-CPs: MODBUS-Protokoll, RTU-Format, S7 ist Slave https://support.industry.siemens.com/cs/ww/de/view/1218007
\17\	CPU-CPU Kommunikation mit SIMATIC Controllern https://support.industry.siemens.com/cs/ww/de/view/78028908

9 Historie

Tabelle 9-1

Version	Datum	Änderung
V1.0	04/2013	Erste Ausgabe
V1.1	09/2017	Korrektur für STEP 7 V5.5 und STEP 7 V12 (TIA Portal)
V2.0	09/2017	Aktualisierung für STEP 7 V14 (TIA Portal)
V3.0	07/2019	Aktualisierung für STEP 7 V15.1 (TIA Portal)