Beispieldokumentation

Deutsche Beschreibung

NUTZUNGSBEDINGUNGEN

Die Verwendung der Beispielprogramme erfolgt ausschließlich unter Anerkennung folgender Bedingungen durch den Benutzer:

INSEVIS bietet kostenlose Beispielprogramme für die optimale Nutzung der S7-Programmierung und zur Zeitersparnis bei der Programmerstellung. Für direkte, indirekte oder Folgeschäden des Gebrauchs dieser Software schließt INSEVIS jegliche Gewährleistung genauso aus, wie die Haftung für alle Schäden, die aus die aus der Weitergabe der die Beispielinformationen beinhaltenden Software resultieren.

BEISPIELBESCHREIBUNG

Inhalt

Dieses Beispiel verbindet eine INSEVIS-SPS mit einem TB20 von Helmholz über Modbus-TCP.

Implementierung eines Modbus-TCP Clients in einer INSEVIS SPS

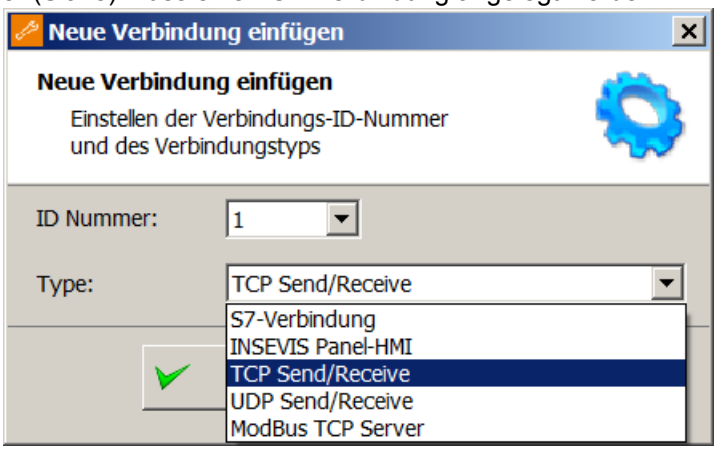
Bei Modbus-TCP werden die Stationen über die IP-Adresse identifiziert und die Nutzdaten in einfachen Telegrammen in die TCP-Datenpakete abgebildet.
Das Verhalten der Modbus-Geräte (Server, Slaves) ist genau beschrieben, der Ablauf im Master (Client) ist jedoch nicht spezifiziert. Eine feste Implementierung im Betriebssystem wäre damit nicht flexibel genug.

Das hier vorgestellte Modbus-TCP-Interface wurde deshalb in S7 realisiert. Um kundenspezifische Anpassungen zu ermöglichen ist es quelloffen.

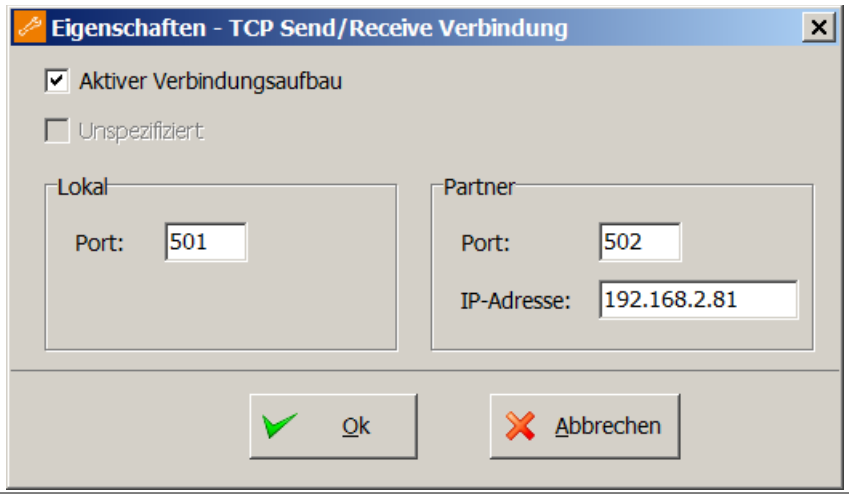
.

Konfiguration

Für jeden einzubindenden Modbus-TCP-Server (Slave) muss eine TCP-Verbindung angelegt werden.
Das erfolgt bei der INSEVIS-SPS im Konfigurationstool ConfigStage.



Unter „Ethernet“ wird eine neue Verbindung vom Typ TCP Send/Receive angelegt und bearbeitet.
Die automatisch vergebene ID-Nummer wird im S7-Programm zur Zuordnung benutzt.



Die erstellte TCP-Verbindung wird mit der IP-Adresse des Kommunikationspartner konfiguriert.
Die Partner-Port-Adresse (= die des Servers) ist bei Modbus per Standard 502; die lokale Portadresse kann weitgehend frei gewählt werden, sie sollte nicht 0 sein und darf nicht mit bestehenden Verbindungen kollidieren.

S7-Programm
FB1 dient als Modbus-TCP client Treiberbaustein und nutzt die Systembausteine zum Senden und Empfangen über TCP/IP und sollte **unverändert** bleiben.
Als Parameter werden VerbindungsID-Nummer, Knotennummer (UID), Modbus- Kommando (function code 1, 2, 3, 4, 6, 15 oder 16) und Nutzdatenpointer übergeben.

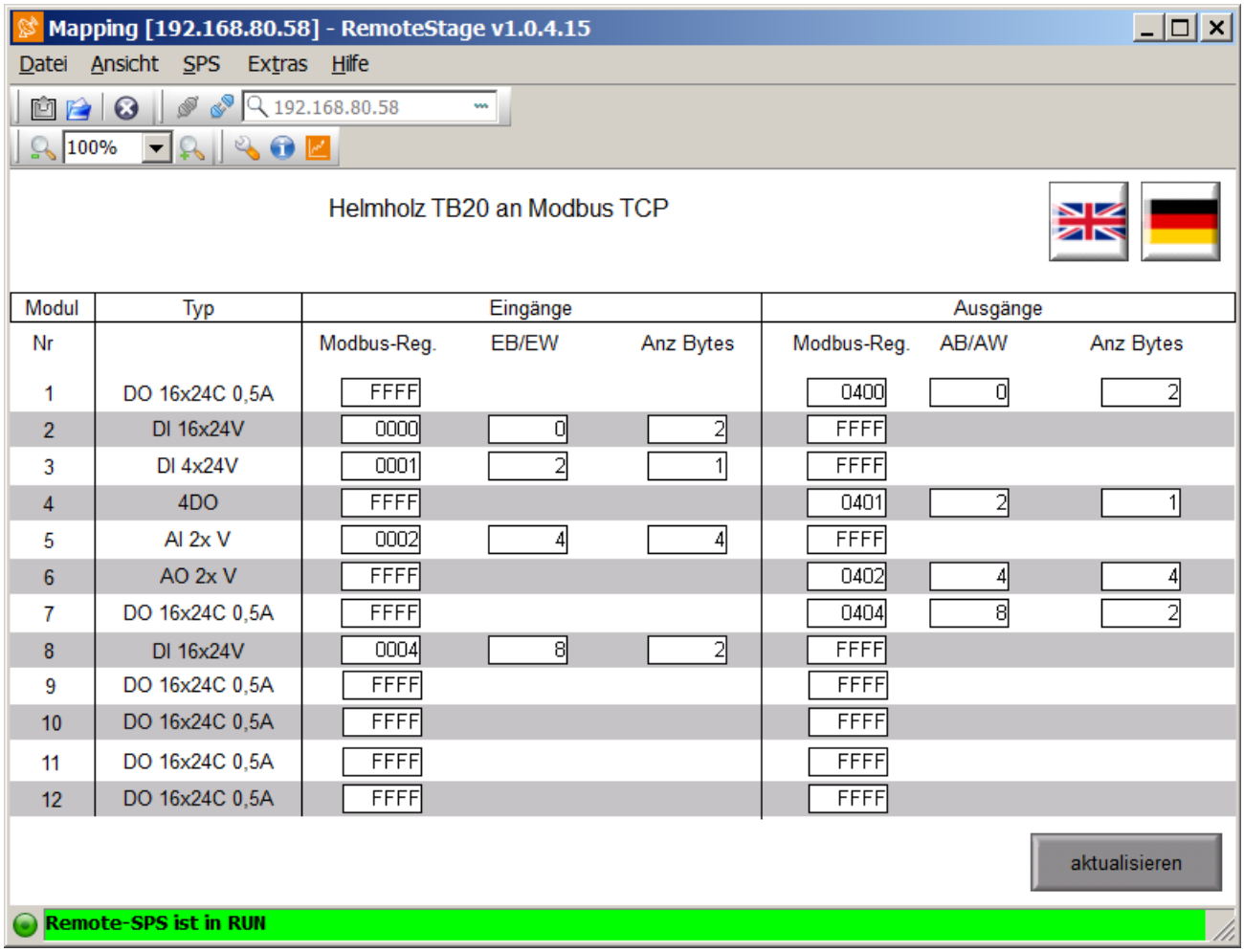
```
CALL  "ModbusTCP_Client" , "tcp"      // FB1 mit DB1 als Instanz-DB
  R         :=FALSE                   // Reset-Eingang, einmalig nach Hochlauf setzen
  ConnectID:=1                        //  VerbindungsID-Nummer aus ConfigStage
  UID       :=                        //  obsolete Knotennummer aus ModbusRTU
  CMD       :=B#16#2                  // Modbus Kommando (funktion code 1,2,3,4,6,15,16)
  Index     :=0                       // Register bzw. Bit-Adresse (0...65535)
  LEN       :=256          // Anzahl zu übertragender Register (1..125) bzw. Bits (1..2000)
  DATA      :="Daten_TCP".Inputs      // ANY-Pointer auf Nutzdatenbereich
  DONE      :="tcp1_done"             // TRUE wenn erfolgreich
  ERROR     :="tcp1_error"            // TRUE bei Fehler
  ErrSrc    :="tcp1_ErrorSrc"         // Fehlercode s. Tabelle
  ErrStatus:="tcp1_ErrCode"
```

Der Aufruf muss wiederholt werden, bis DONE oder ERROR zurückgegeben wird.
Die Längenangabe im Pointer DATA wird zum Kopieren der Nutzdaten benutzt und muss zu den Datenpaketen passen (Beim Senden wird sonst ggf. der Sendepuffer unvollständig gefüllt oder andere Werte überschrieben. Beim Empfang werden ggf. andere Nutzdaten überschrieben)
Alle lokalen Variablen des Kommunikationstreiberbausteins FB1 sowie Sende- und Empfangsdaten liegen in dem zugehörigen Instanzdatenbaustein.

FC1 stellt die Modbus-Kundenapplikation dar und **kann** an die jeweiligen Anforderungen **angepasst** werden. D.h. hier wird programmiert, zu welchen Teilnehmern wann mit welchen Daten kommuniziert wird.
Über einen Sprungverteiler ist eine state-machine realisiert, welche nach Initialisierung zyklisch nacheinander verschiedene Funktionen ausführt.
In diesem Beispiel werden 224 Byte Prozessdaten aktualisiert. Bei maximaler Belegung mit 32 Byte pro Modul entspricht das 7 Slots. In der Praxis werden weniger Prozessdaten vorhanden sein und der Datenrahmen könnte verkleinert werden. Da Modbus auf maximal 125 Register pro Auftrag limitiert ist, muss für weitere Prozessdaten ein weiterer State hinzugefügt werden.
Da die dezentralen Prozessdaten jetzt direkt in das S7-Prozessabbild abgebildet weden, kann der Bit-, Byte- und Wortzugriff mit S7-Befehlen erfolgen.
Am Ende des FC1 erfolgt eine Fehlerüberwachung. Die Reaktion auf Kommunikationsfehler liegt in der Verantwortung des S7-programmieres. In der Demo wird über Bit 6.7 (AutoRecover) definiert, ob beim 1. Fehler gestoppt wird (= gut zur Fehlersuche) oder die Kommunikation zurückgesetzt und damit neu gestartet wird.

Visualisierungsdemo

Das Mapping der Peripherie ist der Schlüssel um im S7-Prozessabbild die Signale der Module wiederzufinden. Im Programm "Toolbox" von Fa. Helmholz erfolgt die Parametrierung des Kopplers sowie der Module. Hier wird auch das Mapping definiert (Die Modbus-Wortregister müssen dann noch in Byteadressen umgerechnet werden). Da das Mapping auch über Modbus ausgelesen werden kann, demonstriert das S7-Beispiel zusammen mit der Visualisierung das Auslesen und Anzeigen der Mapping- und Modulinformationen, hier für 12 Module. Hier erfolgt auch das Umrechnen der Modbus-Wortregister in S7-Byteadressen. Die Anzeige dient nur zur Information und muss dann manuell auf das S7-Programm angewendet werden.
Das Lesen der Modulinformation kann ggf kundenspezifisch um Diagnose- und Parameterdaten erweitert werden.



Modul	Typ	Eingänge			Ausgänge		
Nr		Modbus-Reg.	EB/EW	Anz Bytes	Modbus-Reg.	AB/AW	Anz Bytes
1	DO 16x24C 0,5A	FFFF			0400	0	2
2	DI 16x24V	0000	0	2	FFFF		
3	DI 4x24V	0001	2	1	FFFF		
4	4DO	FFFF			0401	2	1
5	AI 2x V	0002	4	4	FFFF		
6	AO 2x V	FFFF			0402	4	4
7	DO 16x24C 0,5A	FFFF			0404	8	2
8	DI 16x24V	0004	8	2	FFFF		
9	DO 16x24C 0,5A	FFFF			FFFF		
10	DO 16x24C 0,5A	FFFF			FFFF		
11	DO 16x24C 0,5A	FFFF			FFFF		
12	DO 16x24C 0,5A	FFFF			FFFF		

Schritt-für-Schritt-Anleitung

- mit “Helmholz-Toolbox” die IP-Adresse des TB20 auf 192.168.80.160 einstellen oder
  in ConfigStage die IP-Adresse des TP20 (unter Verbindung – Partner) einstellen
- IP-Adresse der SPS auf 192.168.80.50 einstellen oder in ConfigStage anpassen
- Programmvorbereitung:
  SimaticManager:
  In der Hardwarekonfiguration eine S7-300-2PNDP anlegen und die
  IP-Adresse der SPS unter PN-IO einstellen,
  das S7-Programm aus dem Beispielprojekt einfügen
  TIA-Portal:
  In der Hardwarekonfiguration eine S7-300-2PNDP anlegen und die IP-Adresse der SPS einstellen,
  AWL-Quelldatei aus dem Beispielverzeichnis importieren
- Visualisierung in RemoteStage starten oder in (7"-Panel-) SPS laden
- S7-Programm und ConfigStage-Daten im STOP in die SPS laden, auf RUN schalten

- die belegten Adressen des S7-Prozessabbildes in Visualisierung ablesen und
  reale Ein- und Ausgänge mit S7-Variablentabelle "processdata" verifizieren
- OB1 Dummy-Anwendung nach Belieben modifizieren

Troubleshooting
- TB20 und SPS aus PC-Kommandozeile "pingen"
- Fehlercodes nach Tabelle auswerten
- TB20 ein- und ausschalten; SPS NICHT im RUN ausschalten und TB20 weiterlaufen lassen
- Variablentabellen "internal", "Mapping" und "ModuleList" benutzen um elementare Funktionen zu verifizieren
- Bei vorhandener Onboard-Peripherie muß ein Adresskonflikt zwischen Onboard- und ModbusTCP-Peripherie verhindert werden

a) Onboard-Peripherie per ConfigStage oberhalb EB/AB 225 adressieren und / oder
b) FC1, NW3 anpassen: Data-Pointer "P#E 0.0 BYTE 224" auf höhere Adresse
und / oder Anzahl der Register UND Länge des Any-Pointers verkleinern

Hinweise
Dieses Beispiel ist auf Übersichtlichkeit programmiert. Zur Übertragung von Ein- und Ausgängen werden mindestens je ein OB1-Zyklus benötigt. Bei kurzen OB1-Zykluszeiten kann eine hohe Ethernet-Buslast entstehen. Das Auslesen der Modulinformation benötigt weitere Zyklen und erfolgt daher nur auf Anforderung.

Achtung:
FC1 benutzt als state-Merker einen 0-initialisierten Merker. Dieser Merker darf nicht als remanent konfiguriert werden!

Fehlercodes
Die Rückgabewerte des FB1 sind in eine Fehlerquelle (ErrSrc) und einen StatusCode aufgeteilt.
ErrSrc entspricht dem state der state machine in FB1, in dem der Fehler aufgetreten ist. Davon abhängig sind die jeweiligen Fehlercodes:

ErrSrc	ErrStatus	Bedeutung
0,1	Rückgabewerte des SFB 124 TDISCON:	
	8001_{hex}	Parameter ID ist nicht korrekt
	8002_{hex}	Verbindung mit ID ist nicht konfiguriert oder inkorrekter Verbindungstyp.
4,5	Rückgabewerte des SFB 122 TSEND:	
	8001_{hex}	Parameter-ID ist nicht korrekt.
	8002_{hex}	Verbindung mit ID ist nicht konfiguriert oder inkorrekter Verbindungstyp.
	8003_{hex}	Parameter DATA ist nicht korrekt. Nur E, A, M, DB Bereiche erlaubt
	8004_{hex}	Parameter DATA ist nicht korrekt. z.B. DB nicht geladen
	8005_{hex}	Parameter LEN ist 0 oder groser als angegeben unter Parameter DATA
	8006_{hex}	Keine Verbindung zu Partner aufgebaut
	8007_{hex}	Auftrag fehlgeschlagen wegen Verbindungsproblem (Kabel abgezogen, Kommunikation durch Partner zuruckgewiesen).
3	Syntax check Parameter	
	8001_{hex}	UID > 127
	8002_{hex}	invalid CMD
	8003_{hex}	invalid len (Register > 250, Bits/Coils > 2000)
6	Rückgabewerte des SFB 123 TRECV:	
	8001_{hex}	Parameter ID ist nicht korrekt.
	8002_{hex}	Verbindung mit ID ist nicht konfiguriert oder inkorrekter Verbindungstyp.
	8003_{hex}, 8004_{hex}	Parameter DATA ist nicht korrekt.
	8006_{hex}	Keine Verbindung zu Partner aufgebaut
	8007_{hex}	Auftrag fehlgeschlagen wegen Verbindungsproblem (Kabel abgezogen, Kommunikation durch Partner zuruckgewiesen).
7	Syntax check Empfangsdaten	
	9000_{hex}	Ungültige Antwort, Request zurückgewiesen
	$8xxxx_{hex}$	Rückgabewert des SFC20
*	$CAFE_{hex}$	Timeout

RÜCKMELDUNGEN
Möchten Sie Erweiterungswünsche oder Fehler zu diesen Beispielen melden oder wollen Sie anderen eigene Bespielprogramme kostenlos zur Verfügung stellen? ***Bitte informieren Sie uns unter info@insevis.de***
Gern werden Ihre Programme -auf Wunsch mit Benennung des Autors- allen INSEVIS- Kunden zur Verfügung gestellt.

English description

TERMS OF USE

SAMPLE DESCRIPTION

Abstract

This example realizes a Modbus TCP client to demonstrate communication of INSEVIS PLC to Helmholz TB20.

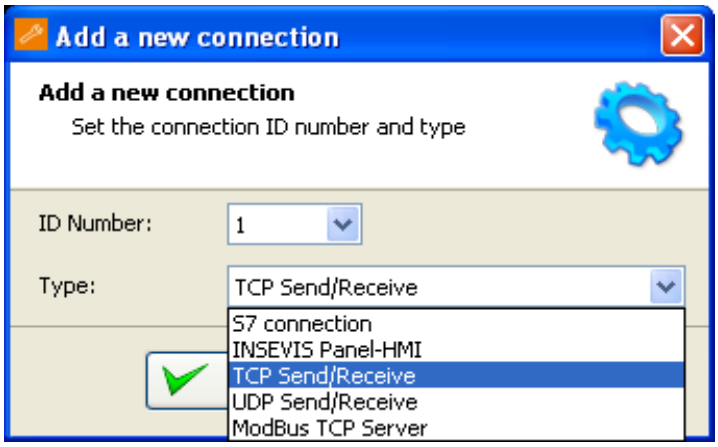
Implementation of a Modbus-TCP Client in an INSEVIS SPS

Modbus TCP identifies devices via a IP address, the payload data are transferred in simple TCP data packets. The behavior of Modbus-devices (Server, Slaves) is specified exactly, but the master (Client) is a "matter of application". A fixed implementation in the operating system would be not flexible.
That's why the introduced Modbus-TCP client application was written in S7. To allow customer adaptions it is open source.

Configuration

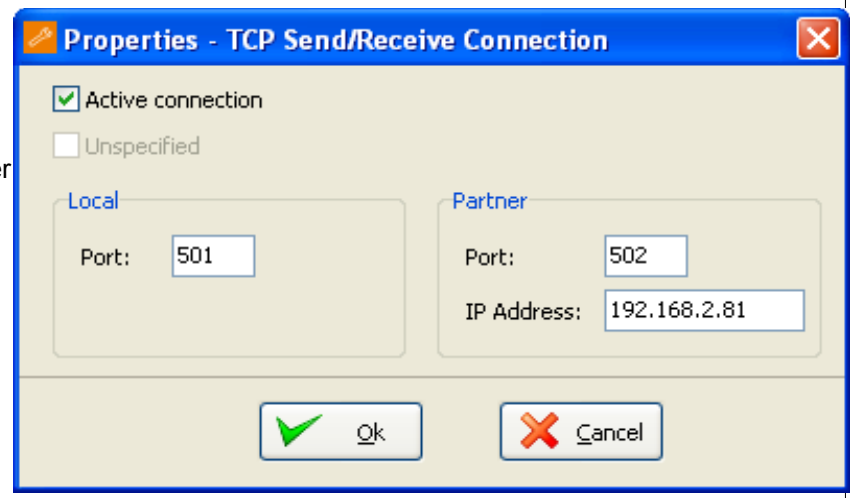
For every Modbus-TCP-Server (slave) an own TCP-connection must be generated.
This is done in the INSEVIS-PLC configuration tool ConfigStage.

Choose „Ethernet“ and create a new connection of type TCP Send/Receive
The automatic assigned ID-number will be used in the S7-program.



Configure the TCP connection with the IP-address of the communication partner

The partners port-address (= address of the server) is always Modbus default 502.
The local port address could be freely chosen. It should be not 0 and must not conflict to other existing connections.



S7-Program
FB1 works as Modbus-TCP client driver and handles the system calls for send und receive via TCP/IP and is **no to change**.
As parameter it uses connection ID-number, node-number (UID), Modbus-command (function code 1, 2, 3, 4, 6, 15 or 16) and payload data pointer.

```
CALL  "ModbusTCP_Client" , "tcp"      // FB1 + DB1 as Instance-DB
  R         :=FALSE                   // Reset-input, to set once while startup
  ConnectID:=1                        //  connection ID-number from ConfigStage
  UID       :=                        //  obsolete node number from ModbusRTU
  CMD       :=B#16#2                  // Modbus command (funktion code 1,2,3,4,6,15,16)
  Index     :=0                       // Register rsp. Bit-address (0...65535)
  LEN       :=256          // number of register (1..125) or bits (1..2000) to transfer
  DATA      :="Daten_TCP".Inputs      // ANY-Pointer payload dataa area
  DONE      :="tcp1_done"             // TRUE if well done
  ERROR     :="tcp1_error"            // TRUE in case of trouble
  ErrSrc    :="tcp1_ErrorSrc"         // errorcode s. table
  ErrStatus:="tcp1_ErrCode"
```

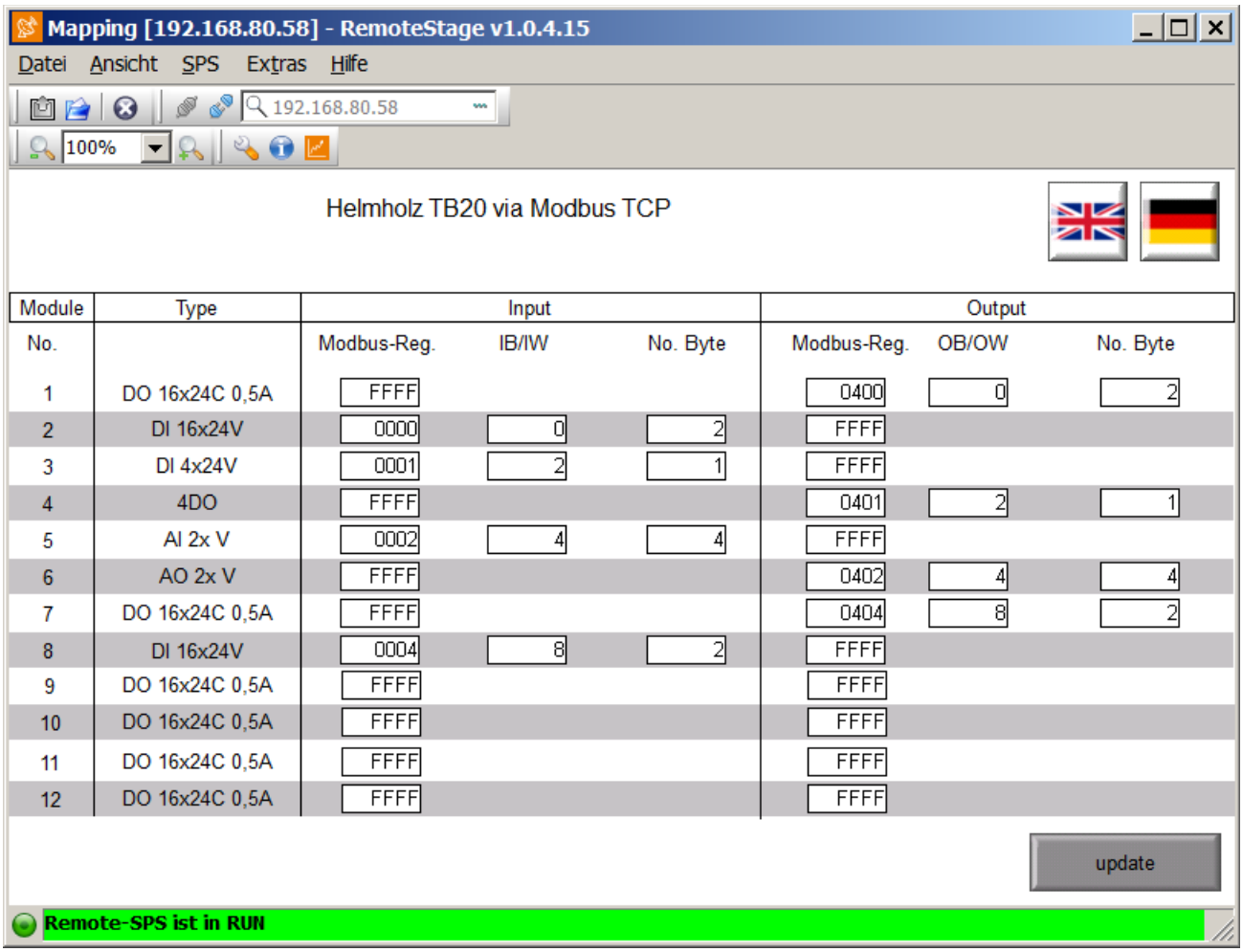
The call of FB1 must repeated until DONE or ERROR are returned.
The length information of Pointer DATA will be used to copy data and must match to the used data packets. (Otherwise sending the buffer could be filled incomplete or other data are overwritten.)
All local data are kept in the instance data block

FC1 represents the Modbus customers application and **must be adapted** i. e. here will be defined when and how to communicate.
A state-machine realizes the communication steps. After an initialization in this example 2 Modbus-commands (function codes) are called cyclically to read and write 225 byte of process data. In case of maximum occupancy each module needs 32 bytes, 7 modules are accessible. Usually the amount of data will be smaller and the number of data to read and write can be reduced. Increasing the amount of data additional transfere cycles must be implemented because Modbus transfers are limited to 125 register.
Due to process data are mapped into the S7 input and output area, user application can use S7's bit-, byte- and word-commands disregarding the Modbus register structure.

A simple error handling is implemented at the end of FC1. The S7 programmer is responsible for the system's error response. The demo uses two behaviours depending bit 6.7 (AutoRecover): stop at 1st error to ease error detection or automatic restart.

Visualization demo

The Mapping of peripheral I/O is the key to find your signals inside the S7-process data.
The program "Toolbox" from Helmholz does the parametrization of the TB20 head station as well as the modules. Here the mapping will be defined (Modbus counts 16 bit register and S7 counts bytes – do not forget to convert).
Fortunately the mapping can be read via Modbus as well. This demo shows, with some additional S7 blocks, how to read and visualize the mapping- and module information to ease your start in S7.
The demo works with maximum 12 modules. The S7 code also does the calculation from Modbus-register into S7-byte addresses. Unfortunately the Visualization is only for information; the S7-program must be written manually based on this information.
Of course this procedure can be expanded and customized for more diagnostic information or parametrization.



Module No.	Type	Input Modbus-Reg.	Input IB/IW	Input No. Byte	Output Modbus-Reg.	Output OB/OW	Output No. Byte
1	DO 16x24C 0,5A	FFFF			0400	0	2
2	DI 16x24V	0000	0	2	FFFF		
3	DI 4x24V	0001	2	1	FFFF		
4	4DO	FFFF			0401	2	1
5	AI 2x V	0002	4	4	FFFF		
6	AO 2x V	FFFF			0402	4	4
7	DO 16x24C 0,5A	FFFF			0404	8	2
8	DI 16x24V	0004	8	2	FFFF		
9	DO 16x24C 0,5A	FFFF			FFFF		
10	DO 16x24C 0,5A	FFFF			FFFF		
11	DO 16x24C 0,5A	FFFF			FFFF		
12	DO 16x24C 0,5A	FFFF			FFFF		

Step-by-Step-Manual

- set IP-address of TB20 to 192.168.80.160 by “Helmholz-Toolbox” or
  setup TB20's IP address in ConfigStage (Connection Partner)
- set IP-address of PLC to 192.168.80.50 or change in ConfigStage
- preparation before S7 program download:
  SimaticManager – hardware configuration:
  insert a S7-300-2PNDP and setup IP-address of PLC in PN-IO,
  copy and paste the S7-program of the example project
  TIA-Portal:
  hardware configuration: insert a S7-300-2PNDP and setup IP-address of PLC,
  import the AWL source file from example directory
- start visualizing in RemoteStage or download to a (7"-Panel-) PLC
- download S7-program and ConfigStage-data into PLC while in STOP, set to RUN
- notice occupied addresses of S7-processdata in visualizing and

verify real in- and output signals via S7-variable table "processdata"
- modify OB1 dummy-application ad libitum

Troubleshooting
- "ping" TB20 and PLC from PC-command shell
- check errorcodes regarding table
- switch TB20 off and on; DO NOT switch off PLC in RUN and keep TB20 running
- use variable tables "internal", "Mapping" and "ModuleList" to verify basic functions
- used onboard periphery addresses MUST NOT overlapp used ModbusTCP-peripherie addresses

a) configure Onboard-Periphery via ConfigStage above IB/OB 225 and / or
b) adapt FC1, NW3: move Data-Pointer "P#E 0.0 BYTE 224" to higher address and / or
reduce amount of register AND length of Any-Pointers

Hints:
This example was written to be easy to understand. It takes some OB1 cycles to update the process data and runs with maximum bus load. Adaption depended customer needs recommended.
Reading additional information needs additional OB1-cycles and is done only on request to keep I/O-transfer performance.

FC1 uses as state-variable a 0-initialized memory. Do not reconfigure this variable as remanent!

Errorcodes
Return values of FB1 are divided into error source (ErrSrc) and a status code (ErrStatus).
Error source accords with the last state of the state machine in FB1, as the error occurred. Related to the error source the status code contains information about the cause of error:

ErrSrc	ErrStatus	description
0,1	Status return of SFB 124 TDISCON:	
	8001_{hex}	parameter ID incorrect
	8002_{hex}	Verbindung mit ID ist nicht konfiguriert oder inkorrekter Verbindungstyp.
4,5	Status return of SFB 122 TSEND:	
	8001_{hex}	parameter ID incorrect
	8002_{hex}	Connection with specified ID is not configured or of incorrect connection type
	8003_{hex}	parameter DATA incorrect, only I, O, M or DB areas allowed
	8004_{hex}	parameter DATA incorrect e.g. DB not available
	8005_{hex}	parameter LEN is 0 or bigger than specified in parameter DATA
	8006_{hex}	no connection to partner established
	8007_{hex}	Job failed due to connection problem (e. g. cable unplugged,communication denied by partner)
3	Syntax check parameter	
	8001_{hex}	UID > 127
	8002_{hex}	invalid CMD
	8003_{hex}	invalid len (register > 250, bits/coils > 2000)
6	Status return of SFB 123 TRECV:	
	8001_{hex}	parameter ID incorrect.
	8002_{hex}	Connection with specified ID is not configured or of incorrect connection type.
	8003_{hex}, 8004_{hex}	parameter DATA incorrect
	8006_{hex}	no connection to partner established
	8007_{hex}	Job failed due to connection problem (e. g. cable unplugged,communication denied by partner)
7	Syntax check receive data	
	9000_{hex}	Invalid response or request denied
	$8xxxx_{hex}$	Status return of SFC20
*	$CAFE_{hex}$	Timeout

FEEDBACK
Do you want to inform us about necessary increments or errors or do you want to provide us with your sample programs to offer it for free to all customers? ***Please inform us at info@insevis.de***
Gladly we would provide your program -if you wish with the authors name- to all other customers of INSEVIS.