SIEMENS

Ingenuity for life

Modbus/TCP Communication of PCS 7 with External Systems

SIMATIC PCS 7, SIMATIC Modbus/TCP

https://support.industry.siemens.com/cs/ww/en/view/75867147

Siemens Industry Online Support

Legal information

Table of Contents

1 Description of the Task and the Solution

1.1 The Task

Modbus is a communication protocol that is used worldwide, is open to all users and supported by many manufacturers.

The request to couple third-party systems to SIMATIC PCS 7 via Modbus is particularly relevant for the:

- expansion or upgrading of existing systems
- coupling of controllers and systems of different manufacturers (also in the course of constructing new systems)

Modbus is based on a client/server architecture (also called master/slave architecture), which can be implemented within PCS 7 in various ways.

The choice depends on the one hand on the supported Modbus protocols (TCP or PtP) of the existing systems and controllers that are to communicate with each other. On the other hand, the connection type which communication partner is the client or server, results from the system interconnection.

1.2 Solution

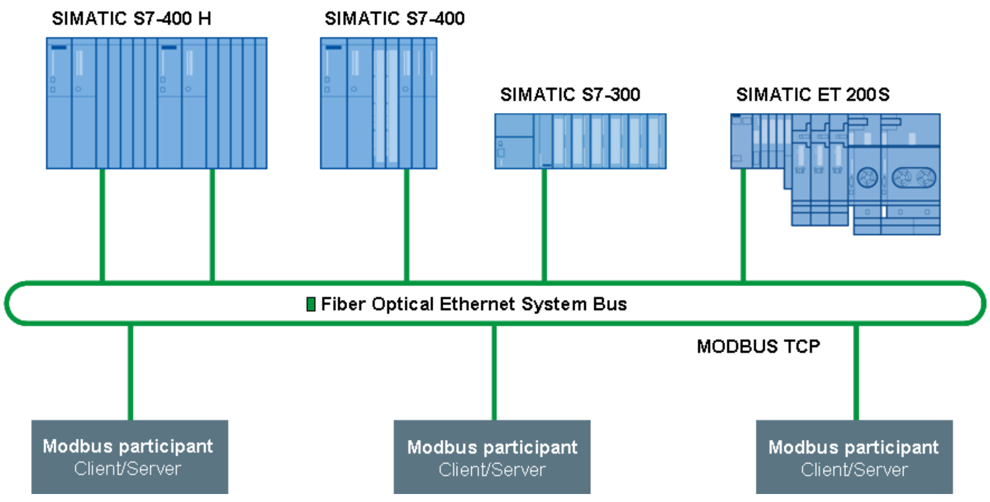
We present a Modbus/TCP library which is available in different variants and with which you can connect external systems to PCS 7 via Modbus/TCP.

This solution is suitable in particular for the expansion or upgrading of existing systems with SIMATIC automation systems and for the coupling of controllers and systems from different manufacturers.

The general description is followed by a configuration example that fully describes the Modbus/TCP client solution for the redundant connection of external systems to PCS 7 via Industrial Ethernet.

For this the Modbus/TCP library "Modbus/TCP CP RED" is used.

Figure 1-1


SIMATIC S7-400 H
SIMATIC S7-400
SIMATIC S7-300
SIMATIC ET 200S
Fiber Optical Ethernet System Bus
MODBUS TCP
Modbus participant Client/Server
Modbus participant Client/Server
Modbus participant Client/Server

Customer benefits

The following configuration example enables you to significantly reduce your engineering costs with PCS 7 standard resources, while improving performance. You can use this solution for re-engineering as well as for integration into existing projects.

The required hardware and software products are all made by Siemens and guarantee the accustomed compatibility, up-to-dateness and upgrade options. The PCS 7 and Modbus standards are consistently observed.

When using the Modbus library you also get the following benefits:

- Simple connection of systems from different manufacturers to PCS 7 via Industrial Ethernet
- Gradual and therefore low-cost expansion or modernization of existing systems over the entire life cycle
- Assured Siemens support
- No specific Modbus knowledge required
- Engineering in familiar PCS 7 environment
- Function block library with online help (F1) in German and English
- Setup in German and English

Core contents of the application description

- Advantages
- Hardware and software required
- Installation and setup
- Configuration
- Performance data

Applies to

The Modbus blocks described and used in this application description are approved for PCS 7 V9.x, PCS 7 V8.x, and PCS 7 V7.1.

2 General Library Information

2.1 Setup

Communication with the Modbus/TCP stations is performed via a communications processor (CP) or via the integrated PN interface of the SIMATIC CPU. A SIMATIC CPU can communicate simultaneously with several Modbus/TCP stations, depending on the number of connection resources of the CPU.

The Modbus/TCP libraries "SIMATIC Modbus/TCP CP Red" and "Modbus/TCP RED V2" support the following CP:

- S7 300: CP 343-1
- S7 400: CP 443-1

Note Use the current version, including the hotfixes for the **SIMATIC Modbus/TCP CP Red** library. The current version of the library can be found under the following link:
https://support.industry.siemens.com/cs/en/en/view/109763972

Note Use is possible only on CPs that support the AG_CNTRL function.

Modbus/TCP PN CPU, Modbus/TCP PN Red and Modbus/TCP SENTRON PAC support the following CPUs:

- ET 200 - IM 151-8 PN/DP CPU
  - IM 154-8 PN/DP CPU
- S7 300/400 - CPU 314C-2 PN/DP
  - CPU 416-3 PN/DP
  - CPU 414-3 PN/DP
  - CPU 412-2 PN
  - CPU 319-3 PN/DP
  - CPU 317-2 PN/DP
  - CPU 315-2 PN/DP
- S7-400 H
  (only SIMATIC Modbus/TCP PN CPU and SIMATIC Modbus/TCP PN Red)
  - CPU 410-5H
  - CPU 417-5H
  - CPU 416-5H
  - CPU 414-5H
  - CPU 412-5H
- Soft PLC
  (only SIMATIC Modbus/TCP PN CPU)
  - SIMATIC WinAC RTX

- SENTRON PAC
  (for SIMATIC Modbus/TCP SENTRON PAC)
  - PAC 4200, from FW V1.5.1
  - PAC 3200, from FW V2.2.1

The Modbus/TCP products are approved for standard CPUs, F CPUs and (PN)-(H)-CPUs.

2.2 Principle of Operation

The Modbus block operates on the Client–Server principle. The client is the active communication node and the server the passive communication station. Data is interchanged between the communication partners by means of various function codes. The automation system (AS) can be both the client and the server in the transfer process. In the initialization phase it is predefined onto which data blocks the Modbus registers and bit values will be mapped.

In cyclic mode a distinction is made between client and server functionality:

- If the automation system is working as a client, when a command is activated a Modbus telegram is generated from the specified actual parameters and sent to the link partner via the TCP/IP connection. After the response telegram has been received and the data has been successfully checked, the required action—such as reading or writing data—is performed. Any errors that have occurred during evaluation or processing are indicated on the Modbus block.
- If the automation system works as a server, the Modbus block waits for a request telegram from the client. If a telegram is received from the client, it is checked and evaluated. After successful checking, the response telegram is generated and the required action—such as reading or writing data—is performed. A processed request or any errors that have occurred during evaluation are indicated on the Modbus block.

TCP connections are set up for Modbus communication between the SIMATIC stations and other Modbus nodes. The standard functions of the SIMATIC NET library are used in that process:

- AG_(L)SEND and AG_(L)RECV for CP versions
- TSEND and TRECV for PN-CPU versions

All Modbus blocks are multi-instance capable.

2.3 Technical Specifications

Table 2-1

	2XV9450-1MB00	2XV9450-1MB02	2XV9450-1MB11
Description	SIMATIC Modbus/TCP CP for NCM_CP	SIMATIC Modbus/TCP PN CPU	SIMATIC Modbus/TCP Red V4 for S7-400 H System
License	Single license on CD	Single license on CD	Single license on CD
Client/server functionality	•/•	•/•	•/•
Function codes	1, 2, 3, 4, 5, 6, 15, 16	1, 2, 3, 4, 5, 6, 15, 16	1, 2, 3, 4, 5, 6, 15, 16
Modbus address area	0 – 65535	0 – 65535	0 – 65535
Read register	125	125	125
Write register	123	123	123
Read bits	2000	2000	2000
Write bits	1968	1968	1968
Capable of multi-instance	•	•	•
Max. number Parallel block calls	• CPU-dependent • **Client:** No block call command limit; max. number of simultaneously active blocks limited by the CPU (AG_SEND/AG_RECV) • **Server:** Limited by the max. number of AG_SEND/AG_RECV calls of the CPU	• Number of block calls unlimited • Number of simultaneously set up connections is CPU-dependent	• CPU-dependent • **Client:** No block call command limit; max. number of simultaneously active blocks limited by the CPU (AG_SEND/AG_RECV) • **Server:** Limited by the max. number of AG_SEND/AG_RECV calls of the CPU
Connection configuration	Static connections over NetPro	Dynamic connections via TCON and TDISCON	Static connections over NetPro
Communication	AG_(L)SEND/ AG_(L)RECV	TSEND/ TRCV	AG_(L)SEND/ AG_(L)RECV
RAM requirement FB (client/server) IDB	16 KB approx. 1 kByte	19 kByte approx. 1 kByte	20 kByte approx. 1 kByte
Application in CFC / PCS 7 possible	•	•	•
Use with older CPs that do not support AG_CNTRL	-	-	-
Multiplexing of TCP connections	CP-dependent	-	CP-dependent
Redundancy	-	-	Single-sided or double-

	2XV9450-1MB00	2XV9450-1MB02	2XV9450-1MB11
functionality			sided redundancy
Use of flag/timer	-	-	-

2.4 Modbus/TCP Versions

There are several versions of the Modbus/TCP library, these being modified for use on different CPU types and communication interfaces.

In addition, appropriate library variants are available for server and client operation as well as for redundant connection.

In addition to the Modbus/TCP library, which is fee-based, allocation blocks for the preproduced solution and a block for the job list are available for downloading. These blocks can be adapted to meet project requirements (open source).

Refer to the following table for the Modbus/TCP library variant that matches your operating range.

Table 2-2

Product	Description	License	Article number
Modbus/TCP CP	Modbus communication via CP	• Single license • Valid for 1 CPU • independent of the number of CPs plugged in and the connected devices	6AV6676-6MB00-6AX0 (Package V6.0)* 6AV6676-6MB00-6AD0 (Download V6.0)* 2XV9450-1MB00 (Announcement of product phase-out V5.0, 01.10.2018)
Modbus/TCP RED V2	Redundant Modbus communication via CPs for S7-400(H) and S7-300	• Single license • Valid for 1 pair of CPUs in an H System or 1 single CPU with CPs • Independent of the number of CPs plugged in or devices connected	2XV9459-1MB11
Modbus/TCP CP Red	Redundant Modbus communication via CPs for S7-400(H) and S7-300	• Single license • Valid for 1 pair of CPUs in an H System or 1 single CPU • Independent of the number of CPs plugged in or devices connected	6AV6676-6MB30-4AX0 (Package) 6AV6676-6MB30-4AD0 (Download)
Modbus/TCP PN-CPU	Modbus communication via the integrated	• Single license	6AV6676-6MB20-3AX0 (Package V3)***

Product	Description	License	Article number
	PROFINET interface	• Valid for 1 CPU • Independent of the number of connected devices	6AV6676-6MB20-3AD0 (Download V3)*** 2XV9450-1MB02 (Announcement of product phase-out V2.6: 01.10.2018)
Modbus/TCP PN RED	Redundant Modbus/TCP communication via the integrated PROFINET interface of S7-400(H) and S7-300 CPU	• Single license • Valid for 1 pair of CPUs in an H System or 1 single CPU • Independent of the number of connected devices	6AV6676-6MB10-0AX0
Modbus/TCP 20 SENTRON PAC	Communication via the integrated PROFINET interface for reading out values from SENTRON PAC 3200 devices and SENTRON PAC 4200 devices	• Single license • Valid for 1 CPU • For the connection of a maximum of 20 PAC devices	6AV6676-6MA30-0AX0
Modbus/TCP 100 SENTRON PAC	Communication via the integrated PROFINET interface for reading out values from SENTRON PAC 3200 devices and SENTRON PAC 4200 devices	• Single license • Valid for 1 CPU • For the connection of a maximum of 100 PAC devices	6AV6676-6MA30-1AX0
Modbus/TCP 512 SENTRON PAC	Communication via the integrated PROFINET interface for reading out values from SENTRON PAC 3200 devices and SENTRON PAC 4200 devices	• Single license • Valid for 1 CPU • For the connection of a maximum of 512 PAC devices	6AV6676-6MA30-2AX0

* If a S7-300 CPU FW < 3.2 or a S7-400 CPU with FW < 6.0 is used, the previous version 2XV9450-1MB00 must be used.

** If a S7-300 CPU < FW 3.2 or a S7-400 CPU < FW V6.0 is used, the previous version 6AV6676-6MB30-3AX0 (package) or 6AV6676-6MB30-3AD0 (Download) must be used.

*** If a S7-300 CPU < FW 3.2, a S7-400 CPU < FW V6.0 or WinAC RTX is used, the previous version 2XV9450-1MB02 must be used.

2.5 Ordering and Scope of Delivery

You can purchase the Modbus/TCP library variants (including licenses) from the Siemens Industry Mall.

The following components are included in the scope of delivery:

- Library with the corresponding Modbus driver blocks
- Online help (F1)
- Manual (PDF format) in German and English

You can download the above mentioned unlicensed Modbus/TCP library versions via the following link. In addition, the open source allocation blocks, the open source job list block and the relevant documentation are available for downloading. Furthermore, you will find there further information on the Modbus versions, ordering information and the scopes of delivery:

https://mall.industry.siemens.com/mall/en/WW/Catalog/Products/10165502?tree=CatalogTree

Note For Runtime operation the downloaded Modbus/TCP library variants require licenses that are separately obtainable from the Industry Mall.

2.6 Installation

Scope of delivery

The installation CD contains the library "Modbus_TCP_CP_Red300_400", sample projects and the manuals in German and English.

After running the setup program, you will find these components by following these paths:

- Library
  "\Program Files\Siemens\Step7\S7libs"
- Sample projects
  "\Program Files\Siemens\Step7\Examples"
- Manuals
  "\Program Files\Siemens\Step7\S7manual\S7Comm"

When you open the Modbus library for the first time, use the function "Browse" in the Open dialog to access the library "Modbus_TCP_CP_Red300_400" in the Siemens directory "S7libs".

There are five folders in the "Modbus_TCP_CP_Red300_400" library:

- Parameter data blocks
- S7 300 Client
- S7 300 Server
- S7 400 Client
- S7 400 Server

Make sure that you take the blocks from the corresponding folder, depending on your CPU type (series S7-300 or S7-400) and the architecture used (server or client).

CAUTION	**The blocks in the S7 folders for the CPU series S7-300 and S7-400 have the same FB numbers.** Make sure that the correct blocks are used in the correct CPU.

2.7 Licensing

The Modbus driver blocks MB_REDCL or MB_REDSV require a license. This licensing must be performed individually for each CPU.

Licensing is possible in two variants:

- Licensing Using the Wizard and the Industry Support App
- Reading out the data from the instance DB and the printed COL

Note With a S7-H station, only the CPU in rack 0 is licensed. The CPU in rack 0 therefore cannot be changed after licensing.

2.7.1 Licensing via the SIMATIC Modbus/TCP Wizard and the Industry Support App

1. Parameterize the block MB_REDCL or MB_REDSV according to your requirements in a cyclic OB (OB1 or cyclic interrupt OB)
2. Load the program into the CPU and set the CPU to RUN.
3. Scannen Sie die Daten nach der Anleitung im FAQ "Lizenz über Service & Support-App anfordern" ein und führen Sie anhand dessen die Registrierung durch.
   https://support.industry.siemens.com/cs/ww/en/view/109746433

Note The Wizard is used for licensing as well as for configuration of the Modbus/TCP communication.

You will find the SIMATIC Modbus/TCP Wizard in the following entry:

SIMATIC S7-300/S7-400: Wizard for creating the connection data for Modbus/TCP communication (Modbus/TCP Wizard)

2.7.2 Licensing by Reading out the IDENT_CODES

The licensing is performed by means of the following steps:

- Reading out the "IDENT_CODES"
- Insert into the "SOFTWARE REGISTRATION FORM"
- Transmission of the form by way of a support request to Customer Support www.siemens.com/industry/supportrequest
- Receipt of the registration key
- The entry of the registration key "REG_KEY"

Entry of the registration key requires the presence of OB121.

IDENT_CODE readout

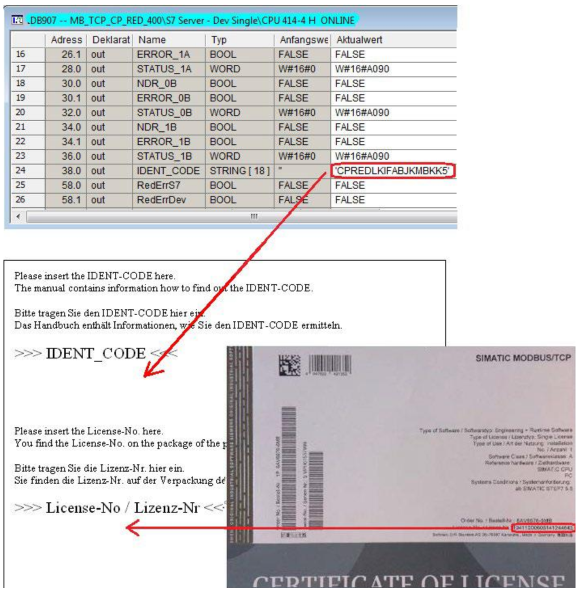
1. Parameterize the block MB_REDCL or MB_REDSV according to your requirements in a cyclic OB (OB1 or cyclic interrupt OB)
2. Load the program into the CPU and set the CPU to RUN.
3. Open the instance DB of the Modbus/TCP block.
4. Open the DB online via the menu item "Data block > Open online". A "Monitor block" via the button is not sufficient.
5. An 18-character string is displayed at the output IDENT_CODE.

CAUTION	When reading directly from the Modbus driver module, not all characters are displayed

6. Copy this string from the DB and paste it into the SOFTWARE REGISTRATION FORM. During installation, this form is stored in the library path "\Program Files\Siemens\Step 7\S7LIBS\Modbus_TCP_CP_Red300_400" and is also located on the installation CD.

7. Enter the license number on the product packaging into the form.

Figure 2-1



.DB907 -- MB_TCP_CP_RED_400\S7 Server - Dev Single\CPU 414-4 H ONLINE

	Adress	Deklarat	Name	Typ	Anfangswe	Aktualwert
16	26.1	out	ERROR_1A	BOOL	FALSE	FALSE
17	28.0	out	STATUS_1A	WORD	W#16#0	W#16#A090
18	30.0	out	NDR_0B	BOOL	FALSE	FALSE
19	30.1	out	ERROR_0B	BOOL	FALSE	FALSE
20	32.0	out	STATUS_0B	WORD	W#16#0	W#16#A090
21	34.0	out	NDR_1B	BOOL	FALSE	FALSE
22	34.1	out	ERROR_1B	BOOL	FALSE	FALSE
23	36.0	out	STATUS_1B	WORD	W#16#0	W#16#A090
24	38.0	out	IDENT_CODE	STRING [18]	''	'CPREDLKIFABJKMBKK5'
25	58.0	out	RedErrS7	BOOL	FALSE	FALSE
26	58.1	out	RedErrDev	BOOL	FALSE	FALSE

Please insert the IDENT-CODE here.
The manual contains information how to find out the IDENT-CODE.

Bitte tragen Sie den IDENT-CODE hier ein.
Das Handbuch enthält Informationen, wie Sie den IDENT-CODE ermitteln.

>>> IDENT_CODE <<<

Please insert the License-No. here.
You find the License-No. on the package of the p

Bitte tragen Sie die Lizenz-Nr. hier ein.
Sie finden die Lizenz-Nr. auf der Verpackung de

>>> License-No / Lizenz-Nr <<

8. Send the form by way of a support request to Customer Support (www.siemens.de/industry/supportrequest). You will then receive the registration key for your CPU.

Entering the registration key **REG_KEY**

The specification of the registration key "REG_KEY" must be made at each MB_REDCL or MB_REDSV block call. It is recommended to store the REG_KEY in a Global DB, via which all Modbus CP blocks receive the necessary registration key.

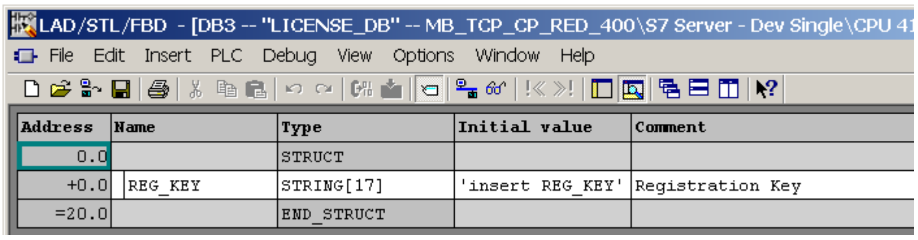
The following work operations are an example:

Copy the ready-made licensing module LICENSE_DB (DB3) from the library "Modbus_TCP_CP_Red300_400" into your project. If the DB number is already used in the project, the license DB can be renamed.

CAUTION	Make sure that you select DB3 from the corresponding library folder, for example, "S7 400 Client".

1. Open the license DB and copy and paste the transmitted 17-digit registration key into the column "Initial value". So that the registration key REG_KEY does not have to be entered again after reloading the CPU, it must be entered in the data block.

Figure 2-2


LAD/STL/FBD - [DB3 -- "LICENSE_DB" -- MB_TCP_CP_RED_400\S7 Server - Dev Single\CPU 41
File Edit Insert PLC Debug View Options Window Help

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	REG_KEY	STRING[17]	'insert REG_KEY'	Registration Key
=20.0		END_STRUCT		

2. Use the menu item "View > Data View" to switch to the data view of the DB.
3. Via the menu command "Edit > Initialize Data Block", all values of the "Initial value" column are transferred to the "Actual value" column.
4. Enter the number of the license DB in the cyclic OB at the parameter "REG_KEY_DB" of the MB_REDCL block.
5. Load the modified blocks into the CPU.
   Input of the REG_KEY registration key can take place at runtime. A switch from "STOP > RUN" is not required.

Result

The Modbus/TCP block is now licensed for this CPU, the output bit LICENSED is TRUE.

2.7.3 Missing or Erroneous Licensing

If no registration key or a wrong one is entered, the LED INTF on CPU S7-400 or the LED SF on CPU S7-300 flashes once per minute and an entry is written cyclically in the diagnostic buffer regarding the missing license.

CAUTION **The cyclic entries burden the diagnostic buffer**

The error number for a missing license is W #16 #A090.

In case of a missing or wrong registration key, the Modbus communication is processed, but W#16#A090 "No valid license available" is always displayed at the outputs STATUS_x. If this error code is displayed despite the registration key entered, it must be checked whether FC10 "EQ_STRING" has been inserted into the project.

CAUTION **If OB121 is missing in the controller, the CPU is set to STOP status.**

2.8 Requirements

2.8.1 Hardware Requirements

The "AG_CNTRL" block from the "SIMATIC_NET_CP" library is integrated in the Modbus driver block and is used to terminate and reestablish existing connections.

However, older CPs or older firmware versions do not support the block "AG_CNTRL" and therefore also not the Modbus driver block.

Refer to the following table for which CPs and firmware statuses support the Modbus driver block.

Table 2-3

Type	Article number	Firmware version
CP343-1 Lean	6GK7343-1CX10-0XE0	≥ 2.1
CP343-1	6GK7343-1EX21-0XE0	≥ 1.0.17
CP343-1	6GK7343-1EX30-0XE0	≥ 2.0.16
CP343-1 Advanced	6GK7343-1GX21-0XE0	≥ 1.0.24
CP343-1 Advanced	6GK7343-1GX30-0XE0	≥ 1.0.23
CP343-1 Advanced	6GK7343-1GX31-0XE0	≥ 3.0
CP443-1	6GK7443-1EX20-0XE0	≥ 1.0.26, but **not** V2.1.12
CP443-1	6GK7443-1EX30-0XE1	≥ 3.0, but **not** V3.2.9
CP443-1	6GK7443-1EX30-0XE0	≥ 3.0, but **not** V3.2.9
CP443-1 Advanced	6GK7443-1EX40-0XE0	≥ 2.2.35
CP443-1 Advanced	6GK7443-1EX41-0XE0	≥ 1.0.24
CP443-1 Advanced	6GK7443-1GX20-0XE0	≥ 2.0, but **not** V2.1.12
CP443-1 Advanced	6GK7443-1GX30-0XE0	≥ 3.0, but **not** V3.2.9

Note Further details about the hardware requirements can be found in the FAQ: Which technical data are valid for the SIMATIC Modbus/TCP blocks and for which CPUs and CPs are they released?

2.8.2 Software Requirements

The following software components are required for the redundant Modbus link presented in this section:

- Library "Modbus/TCP CP RED"
- SIMATIC PCS 7 V7.1 SP2 or higher
- STEP 7 V5.5 and higher
- Library "SIMATIC NET"

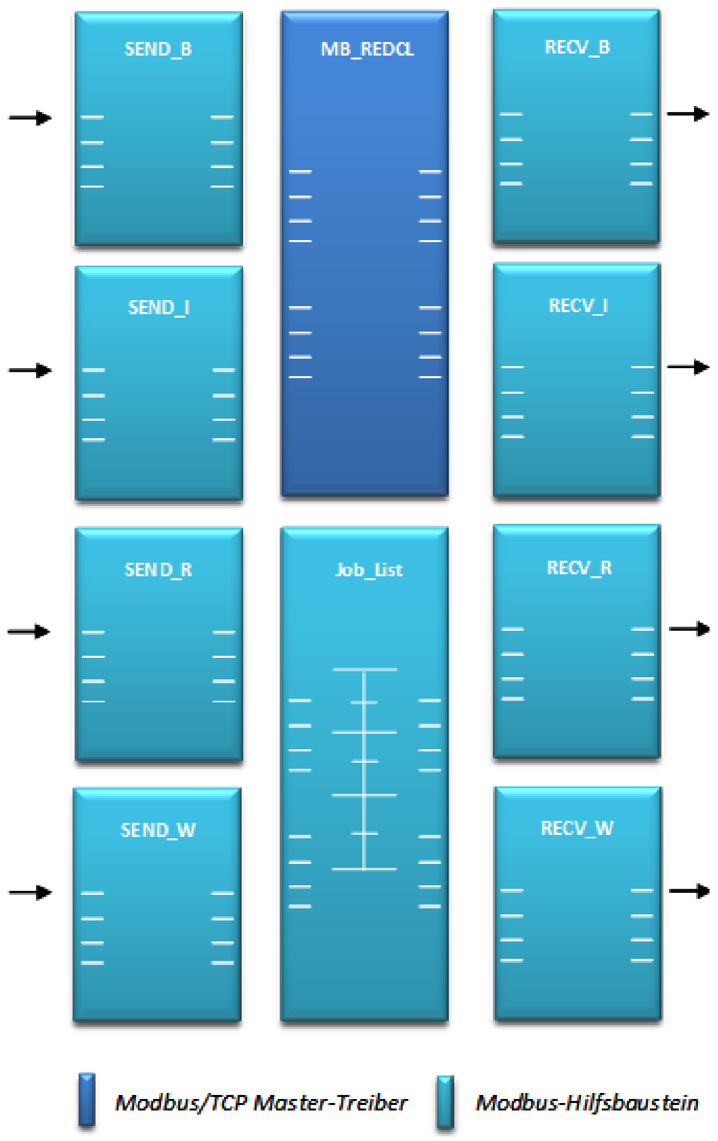
3 PCS 7 Modbus/TCP

The Modbus/TCP library, product blocks, interface blocks and the template presented are based on practical experience and analyses.

In addition to the product block MB_REDCL, utility blocks are at your disposal in the form of open source blocks. You can adapt these to your project requirements and supplement them.

By providing the Modbus library, the open source blocks and the present application, you save analysis effort, development work and engineering time.

Figure 3-1



A PCS 7 measurement point for a redundant Modbus/TCP client coupling consists of three block types:

- Interface blocks (SEND and RECEIVE)
- Modbus driver
- Job_list block

Interface blocks (add-on Modbus blocks)

The send blocks prepare the data provided by the CFC charts or data blocks for the communication data block.
The data stored in the data block can be represented in a convenient form on the CFC charts via the interface blocks.

Both the receive and send blocks are available for the data types bool, real, integer and word.

Modbus driver for redundant communication (MB_REDCL)

The function block MB_REDCL fulfills the following tasks:

- Coordination of the connection(s) over which the telegrams are sent
- Monitoring of all configured connections via AG_CNTRL
- Execution of the Transaction Identifier (TI)

The function block MB_REDCL allows communication between a CP443-1 and CP343-1 and a partner that supports the open Modbus/TCP protocol to be set up in a redundant system. Function codes 1, 2, 3, 4, 5, 6, 15 and 16 are supported.

In this case the block functions as a Modbus client, i.e. as an active partner, and can process a maximum of 4 telegram jobs.

You can find detailed block and function descriptions on the installation DVD and after installation via the following path:

\Program Files\Siemens\Step7\S7manual\S7Comm

Note The block MB_REDCL is usable only for the client application.

Job_list block

With the Job_List order list module, it is possible to process several Modbus/TCP telegrams one after the other with little programming effort.

The following functions are available:

- Status message for timeouts
- Job list can be processed time-controlled
- Job list can be cancelled at any time
- Jobs can be skipped temporarily

Note The block as well as detailed block and function descriptions can be found in the following SIOS entry:
https://support.industry.siemens.com/cs/ww/en/view/62830463

3.1 Address Mapping of the Function Block MB_REDCL

Modbus/TCP addresses

In the case of the present Modbus/TCP library the data are transferred in a series of memory areas that consist of different data types. The distinction between these memory areas is made via the register or bit addresses.

The following Modbus data types are supported:

- Coils
- Input register
- Holding register coil

For example, the holding register with offset 0 is called register 40001 (memory type 4xxxx, Reference 0001). The function block MB_REDCL uses the actually transmitted Modbus/TCP address for its parameters "start_x" and "START_ADDRESS". Transmission can be effected with any function code register/bit addresses from 0000H to FFFFH.

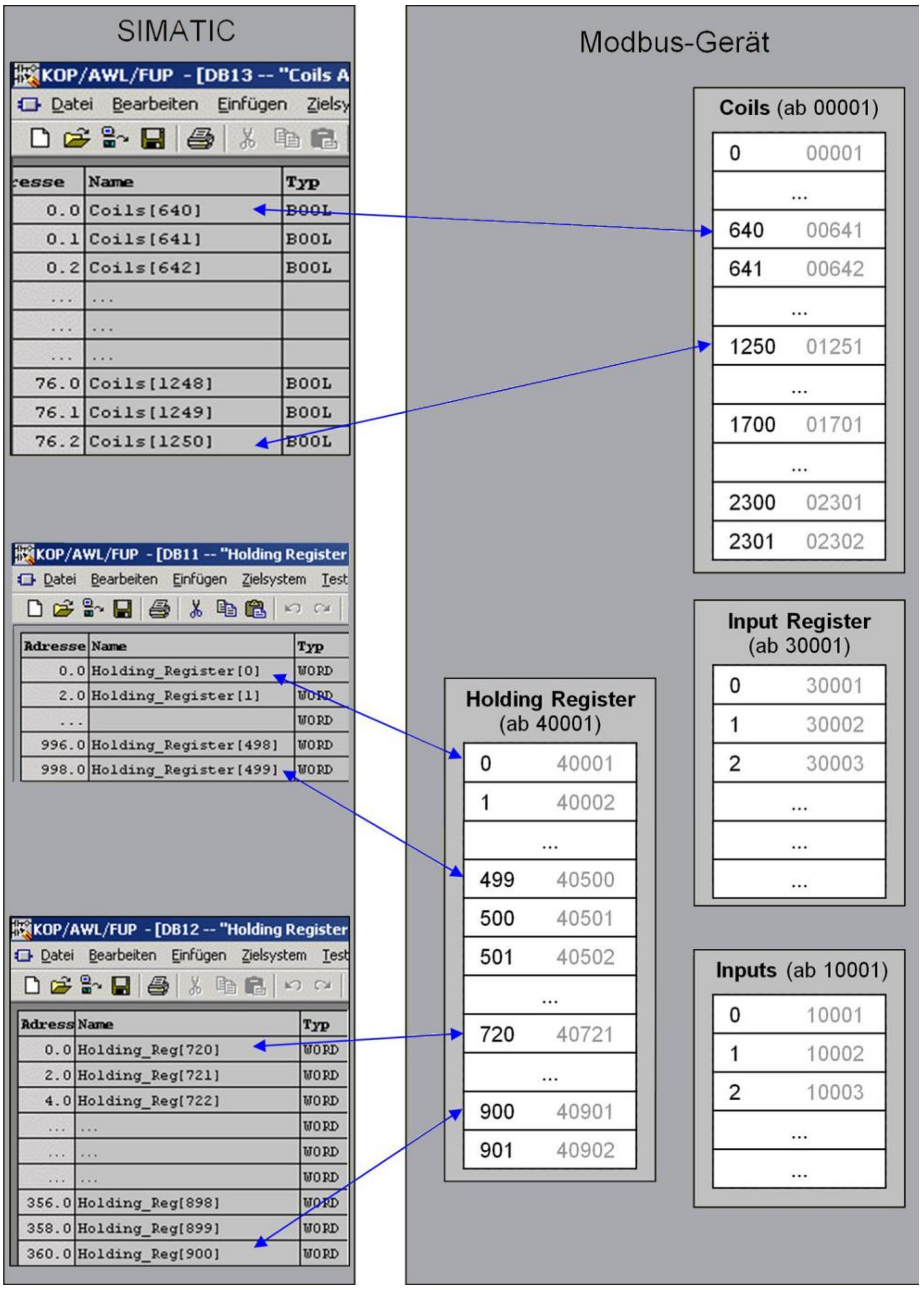
Example

Table 3-1

Address	Symbol name	Data type	Start value	Comment	
+16.0	data_type_1	BYTE	B#16#3	Write holding	
+18.0	db_1	WORD	W#16#B	DB 11	
+20.0	start_1	WORD	W#16#0	Start address:	0
+22.0	end_1	WORD	W#16#1F3	End address:	499
+24.0	data_type_2	BYTE	B#16#3	Write holding	
+26.0	db_2	WORD	W#16#C	DB 12	
+28.0	start_2	WORD	W#16#2D0	Start address:	720
+30.0	end_2	WORD	W#16#384	End address:	900
+32.0	data_type_3	BYTE	B#16#4	Input register	
+34.0	db_3	WORD	W#16#D	DB 13	
+36.0	start_3	WORD	W#16#2D0	Start address:	640
+38.0	end_3	WORD	W#16#3E8	End address:	1250
+40.0	data_type_4	BYTE	B#16#0	Not used	
+42.0	db_4	WORD	0	0	
+44.0	start_4	WORD	0	0	
+46.0	end_4	WORD	0	0	
+48.0	data_type_5	BYTE	B#16#0	Not used	
+50.0	db_5	WORD	0	0	
+52.0	start_5	WORD	0	0	
+54.0	end_5	WORD	0	0	
+56.0	data_type_6	BYTE	B#16#0	Not used	
+58.0	db_6	WORD	0	0	
+60.0	start_6	WORD	0	0	
+62.0	end_6	WORD	0	0	
+64.0	data_type_7	BYTE	B#16#0	Not used	

Address	Symbol name	Data type	Start value	Comment
+66.0	db_7	WORD	0	0
+68.0	start_7	WORD	0	0
+70.0	end_7	WORD	0	0
+72.0	data_type_8	BYTE	B#16#0	Not used
+74.0	db_8	WORD	W#16#0	0
+76.0	start_8	WORD	W#16#0	0
+78.0	end_8	WORD	W#16#0	0

Figure 3-2


SIMATIC
KOP/AWL/FUP - [DB13 -- "Coils A
Datei Bearbeiten Einfügen Zielsy
KOP/AWL/FUP - [DB11 -- "Holding Register
Datei Bearbeiten Einfügen Zielsystem Test
KOP/AWL/FUP - [DB12 -- "Holding Register
Datei Bearbeiten Einfügen Zielsystem Test
Modbus-Gerät
Coils (ab 00001)
Input Register (ab 30001)
Holding Register (ab 40001)
Inputs (ab 10001)

esse	Name	Typ
0.0	Coils[640]	BOOL
0.1	Coils[641]	BOOL
0.2	Coils[642]	BOOL
...	...	
...	...	
...	...	
76.0	Coils[1248]	BOOL
76.1	Coils[1249]	BOOL
76.2	Coils[1250]	BOOL

Adresse	Name	Typ
0.0	Holding_Register[0]	WORD
2.0	Holding_Register[1]	WORD
...		WORD
996.0	Holding_Register[498]	WORD
998.0	Holding_Register[499]	WORD

Adress	Name	Typ
0.0	Holding_Reg[720]	WORD
2.0	Holding_Reg[721]	WORD
4.0	Holding_Reg[722]	WORD
...	...	WORD
...	...	WORD
...	...	WORD
356.0	Holding_Reg[898]	WORD
358.0	Holding_Reg[899]	WORD
360.0	Holding_Reg[900]	WORD

Coils (ab 00001)

0	00001
...	
640	00641
641	00642
...	
1250	01251
...	
1700	01701
...	
2300	02301
2301	02302

Input Register (ab 30001)

0	30001
1	30002
2	30003
...	
...	
...	

Holding Register (ab 40001)

0	40001
1	40002
...	
499	40500
500	40501
501	40502
...	
720	40721
...	
900	40901
901	40902

Inputs (ab 10001)

0	10001
1	10002
2	10003
...	
...	

3.2 Installation of the Modbus library

Requirement

SIMATIC PCS 7 must be installed.

Procedure

1. Insert the Modbus/TCP CD into the CD drive of your PG/PC.
   If the setup program does not start automatically, install as follows:
   - In Windows Explorer, select the CD drive
   - Open the directory "Setup"
   - Start the file "SIMATIC ModbusTCP CP Redundant.exe"
2. Follow the instructions displayed by the installation program.

The following components are installed:

- Library "Modbus_TCP_CP_Red300_400"
  "\Program Files\Siemens\Step 7\S7LIBS"
- Two sample projects
  "\Program Files\Siemens\Step 7\EXAMPLES"
- The Programming Manual
  "\Program Files\Siemens\Step 7\S7MANUAL\S7Comm"
- Software registration form "SOFTWARE REGISTRATION FORM"
  "\Program Files\Siemens\Step 7\S7LIBS\Modbus_TCP_CP_Red300_400"

Sample project

The supplied sample project "MB_TCP_CP_RED_CFC" has six SIMATIC stations that cover different function variants:

- S7 Client – S7 Single
- S7 Servers – S7 Single
- S7 Client – Dev Single
- S7 Client – Double-sided
- S7 Servers – Dev Single
- S7 Server – Double-sided

The stations already have preconfigured connections and Modbus typicals (CFC plans).

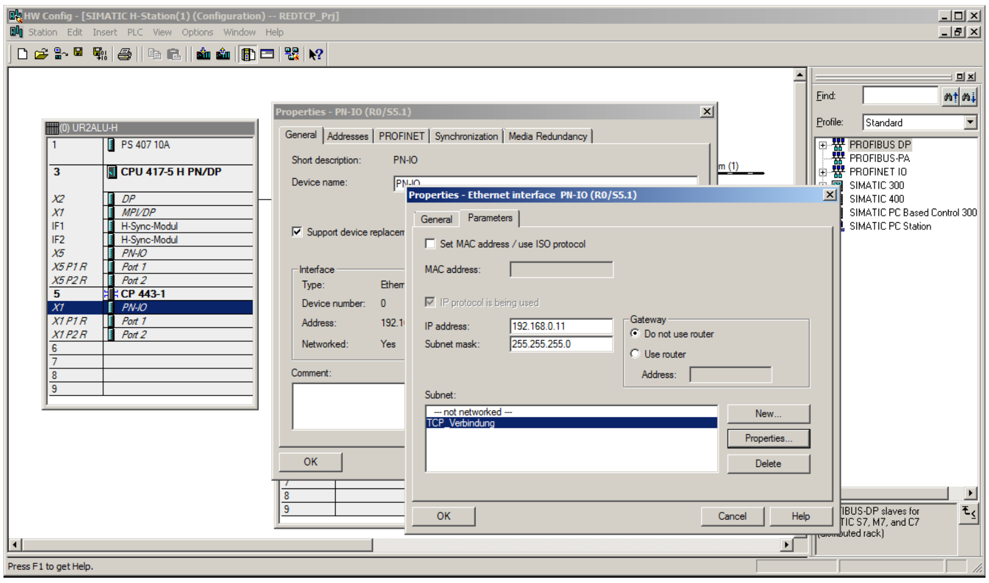
4 Configuration

4.1 Parameterizing the CP

Switch to parameterization of the CP in HW Config and perform the following steps:

1. In HW Config open the properties of the first CP.
2. Press the "Properties" button.
3. Switch to the "Parameters" tab.
4. Select the following subnet and IP address:
   - IP Address: 192.168.0.11
   - Subnet mask: 255.255.255.0
5. Repeat steps 1 through 4 and select the following subnet and IP address:
   - IP Address: 192.168.0.12
   - Subnet mask: 255.255.255.0

Figure 4-1


HW Config - [SIMATIC H-Station(1) (Configuration) -- REDTCP_Prj]

Properties - PN-IO (R0/S5.1)

Short description: PN-IO

Device name: PN-IO

Support device replacement: ✓

Interface – Type: Ethernet; Device number: 0; Address: 192.1…; Networked: Yes

Properties - Ethernet interface PN-IO (R0/S5.1) – Parameters

Set MAC address / use ISO protocol: ☐

MAC address:

IP protocol is being used: ✓

IP address: 192.168.0.11

Subnet mask: 255.255.255.0

Gateway: [● Do not use router] ○ Use router

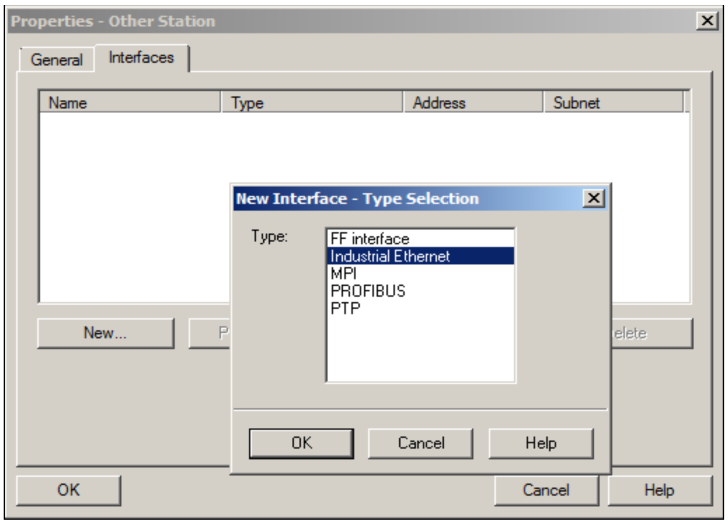
Subnet: --- not networked --- [✓ TCP_Verbindung]

4.2 Parameterization of the Link Partner

Switch to parameterization of the link partner, switch to NetPro and perform the following steps:

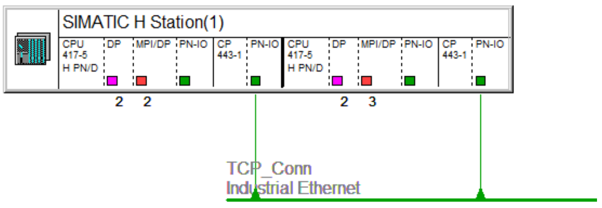
1. Select "Other Station" in the catalog under "Stations" and drag it to the NetPro Configuration window.
2. Open the properties of the "Other Stations" and switch to the "Interfaces" tab.
3. Press the "New..." button and select "Industrial Ethernet" in the window that appears. Confirm with "Ok".

Figure 4-2



4. Assign an IP address, in the example 192.168.0.11 for the primary interface, which is located in the same subnet as the link partner station. The subnet mask must match the subnet mask of the partner station.
5. Select the corresponding subnet that makes the connection between the CP interface and the link partner interface.
6. For a redundant connection repeat steps 3 through 5 and assign an IP address for the redundant interface, in this example 192.168.0.12, which is located in the same subnet as the link partner station. The subnet mask must match the subnet mask of the partner station.

Figure 4-3



4.3 Configuration of the CPU and of the Link Partners

This section describes the project planning of a redundant CPU with redundant coupling partners (section 4.3.1 "Configuration of a red. CPU and red. Coupling Partner"), as well as the project planning of a red. CPU with a single coupling partner (section 4.3.2 "Configuration of a red. CPU and a Single Coupling Partner").

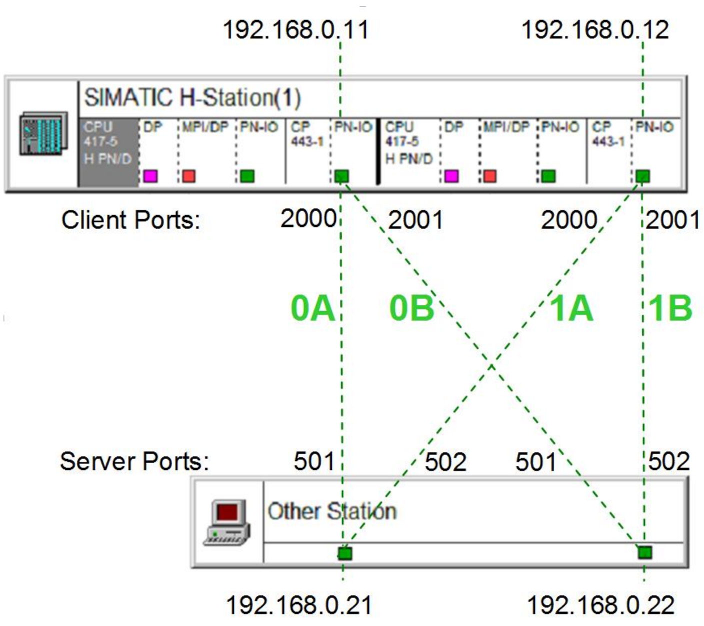
4.3.1 Configuration of a red. CPU and red. Coupling Partner

Note The configuration of a redundant CPU with a coupling partner that does not have a redundant communication processor is described in Section 3.7 "Configuration of a red. CPU and coupling partner with only one communication processor"

Configuration of the communication connections

The CPs represent the links for a redundancy between an H System and a communication partner connected over Industrial Ethernet. For the redundant connection on both sides, connection project planning must be carried out for the respective interfaces to the coupling partner.

Figure 4-4



IP addresses 192.168.0.11 and 192.168.0.12 are assigned to the H System in this example.

For the access to node A of the coupling partner the port number 2000 can be used for CP0 as well as for CP1, because the IP addresses of the two CPs are different (Connection **0A** and **1A**).

The same port number can also be used for accessing node B of the communication partner for both CPs: 2001 (connection **0B** and **1B**).

In this example, the partner station has the IP addresses 192.168.0.21 and 192.168.0.22.

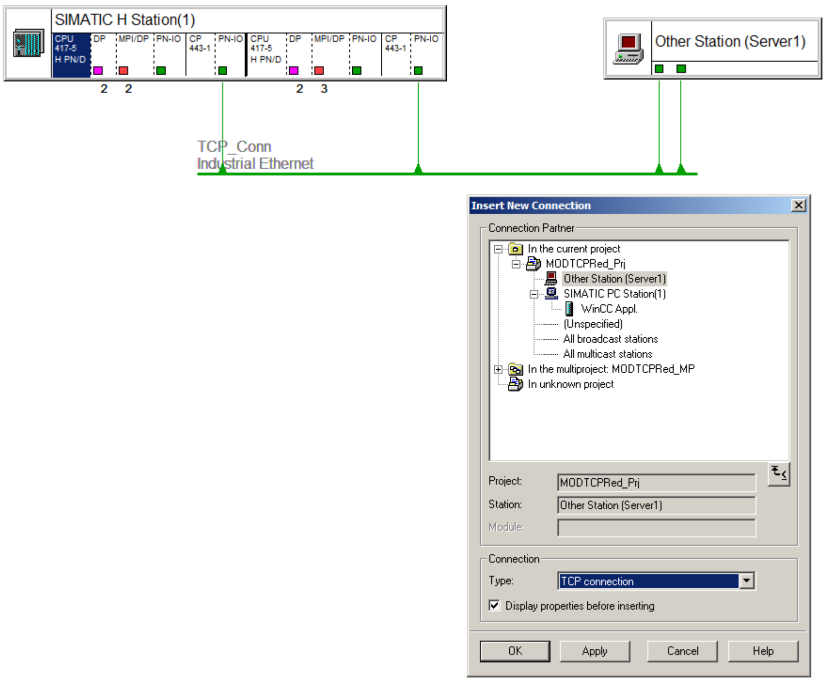
The same port number can be used for node A and node B to access CP0 of the H System: 501 (connection **0A** and **0B**).

You can also use the same port number to access CP1 of the H System: 502 (connection **1A** and **1B**).

Create 0A connection

1. Select the first CPU of the H System in NetPro.
2. Add a new connection to the NetPro configuration table.
3. Select "Other Station (Server 1)" as the connection partner.
4. Select the connection "TCP connection".

Figure 4-5



Project: MODTCPRed_Prj

Station: Other Station (Server1)

Module:

Connection

Type: TCP connection

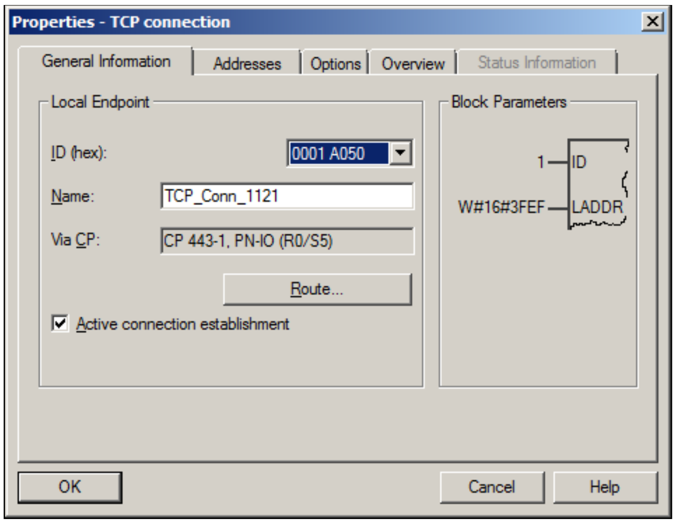
☑ Display properties before inserting

5. Click on the "OK" button. The connection properties window opens.

Parameterize connection 0A

1. Assign an ID in the tab "General Information. The ID is freely selectable.
2. Under "Names:" to be able to identify your TCP connection in the network

Figure 4-6



Properties - TCP connection

Local Endpoint

ID (hex): 0001 A050

Name: TCP_Conn_1121

Via CP: CP 443-1, PN-IO (R0/S5)

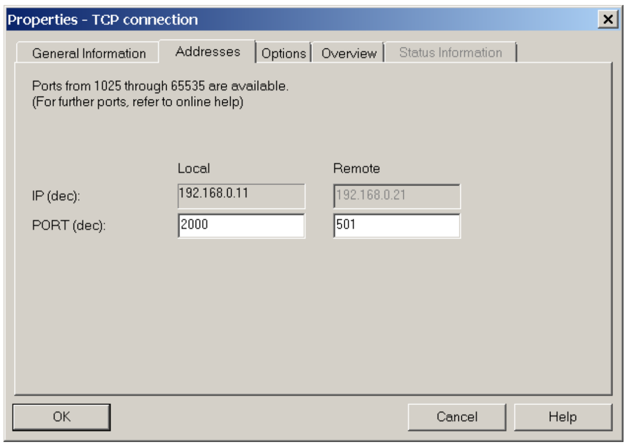
☑ Active connection establishment

Block Parameters

Note The block parameters "Connection ID" (Local ID) and "Load address" are required later for the parameterization of the Modbus driver.

3. The checkbox "Active connection establishment" must be enabled.
4. Actuate the "Route..." button and under "Remote" select the first port of the "Other Station (Server)".
5. Switch to the "Addresses" tab.
6. Assign the following port numbers and make sure that the corresponding IP addresses are selected.

Figure 4-7



Properties - TCP connection

Ports from 1025 through 65535 are available.
(For further ports, refer to online help)

	Local	Remote
IP (dec):	192.168.0.11	192.168.0.21
PORT (dec):	2000	501

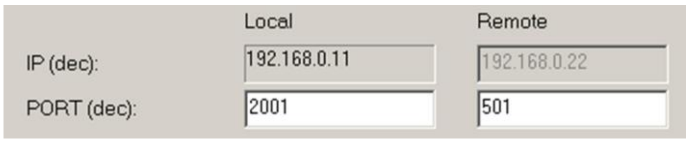
7. Click on "OK" to exit the dialog.

Note The IP addresses in this dialog are not editable. If an incorrect IP address is displayed, exit the dialog by clicking the "OK" button and reassign the ports of the foreign station.

Create 0B connection

1. Insert a second connection into the same CPU.
2. Activate the "Active connection establishment" checkbox in the "General Information" tab.
3. On the tab "General" click on the "Route..." button.
4. Select the <u>second</u> port of the third-party station.
5. Switch to the "Addresses" tab.
   Assign the following port numbers and make sure that the corresponding IP addresses are selected:

Figure 4-8



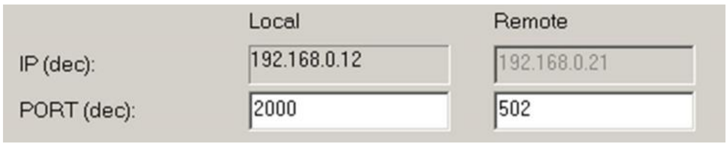
	Local	Remote
IP (dec):	192.168.0.11	192.168.0.22
PORT (dec):	2001	501

6. Click on "OK" to exit the dialog.

Create 1A connection

1. Highlight the second CPU and add a new connection (steps 2 through 8).
2. Activate the "Active connection establishment" checkbox in the "General Information" tab.
3. Click on the "Route..." button in the "General Information" tab.
4. Select the <u>first</u> port of the third-party station.
5. Switch to the "Addresses" tab.
   Assign the following port numbers and make sure that the corresponding IP addresses are selected.

Figure 4-9



	Local	Remote
IP (dec):	192.168.0.12	192.168.0.21
PORT (dec):	2000	502

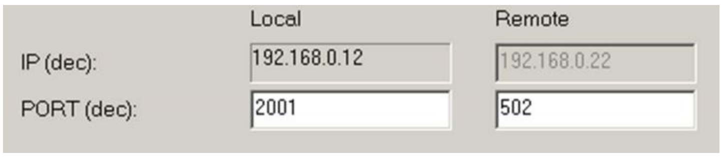
6. Click on "OK" to exit the dialog.

Create 1B connection

1. Select the second CPU again and insert a second new connection.
2. Activate the "Active connection establishment" checkbox in the "General Information" tab.
3. On the "General Information" click on the "Route selection..." button.
4. Select the second port of the third-party station.

Switch to the "Addresses" tab.
Assign the following port numbers and make sure that the corresponding IP addresses are selected.

Figure 4-10



	Local	Remote
IP (dec):	192.168.0.12	192.168.0.22
PORT (dec):	2001	502

5. Click on "OK" to exit the dialog.

Note In the "Addresses" tab, via the button ("Route..."), you can adjust the port numbers as required.

Network setup in NetPro

The following application description assumes a PCS 7 project with a redundant automation system and one CP 443-1 each.

For better understanding the CPs 443-1 have the following IP numbers and subnets:

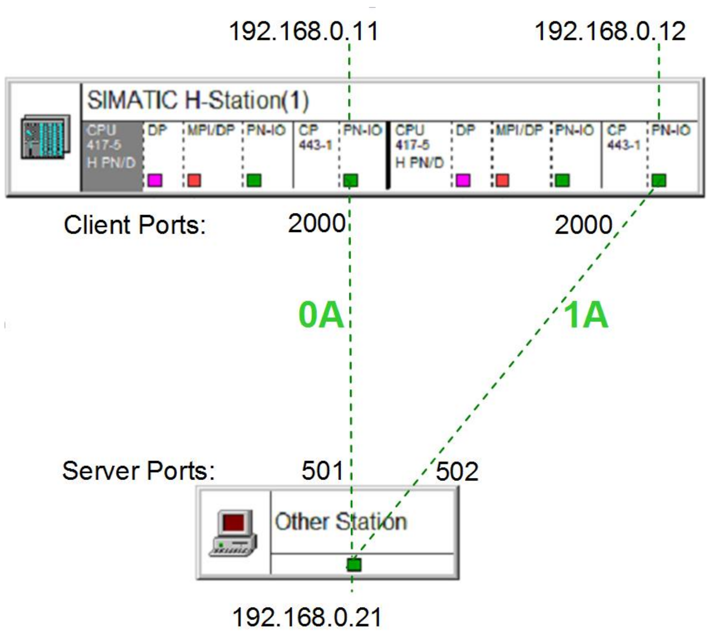
- First CP 443-1
  IP Address: 192.168.0.11
  Subnet mask: 255.255.255.0
- Second CP 443-1
  IP Address: 192.168.0.12
  Subnet mask: 255.255.255.0

4.3.2 Configuration of a red. CPU and a Single Coupling Partner

Configuration of the communication connections

The CPs represent the links for a redundancy between an H System and a communication partner connected over Industrial Ethernet. For the one-sided redundant connection, connection configurations must be made for the respective interfaces with the link partner.

Figure 4-11

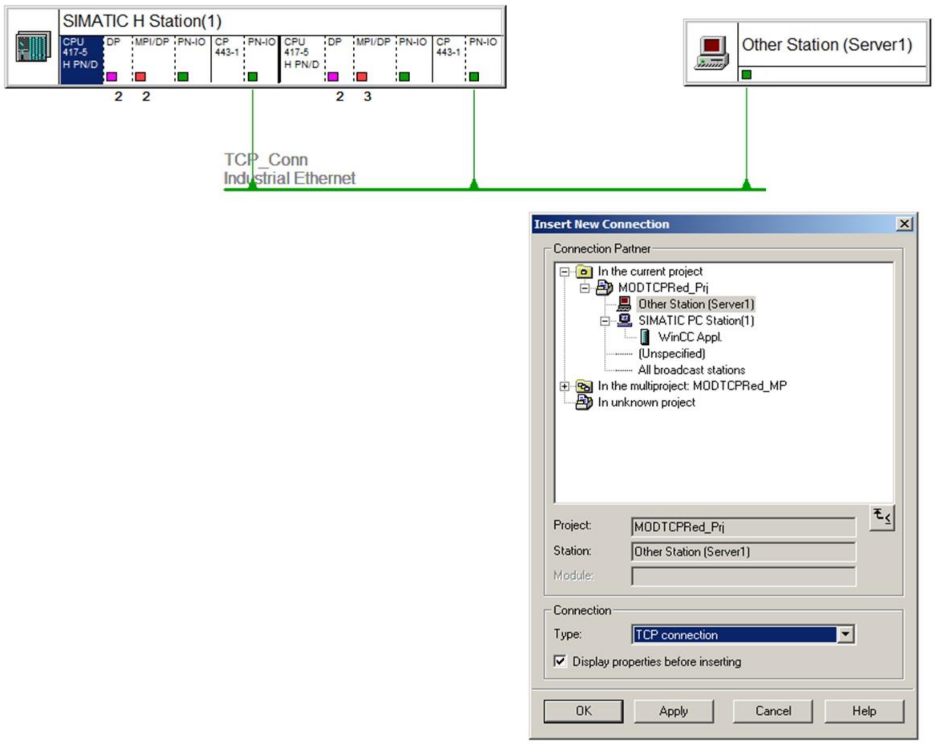


IP addresses 192.168.0.11 and 192.168.0.12 are assigned to the H System in this example.
For the access to the coupling partner the port number 2000 can be used for CP0 as well as for CP1, because the IP addresses of the two CPs are different (Connection **0A** and **1A**).
In this example the partner station has the IP address 192.168.0.21.

Create 0A connection

1. Select the first CPU of the H System in NetPro.
2. Add a new connection to the NetPro configuration table.
3. Select "Other Station" as the connection partner.
4. Select the connection "TCP connection".

Figure 4-12



Insert New Connection

Connection Partner
- In the current project
  - MODTCPRed_Prj
    - Other Station (Server1)
    - SIMATIC PC Station(1)
      - WinCC Appl.
    - (Unspecified)
    - All broadcast stations
- In the multiproject: MODTCPRed_MP
- In unknown project

Project: MODTCPRed_Prj
Station: Other Station (Server1)
Module:

Connection
Type: TCP connection
☑ Display properties before inserting

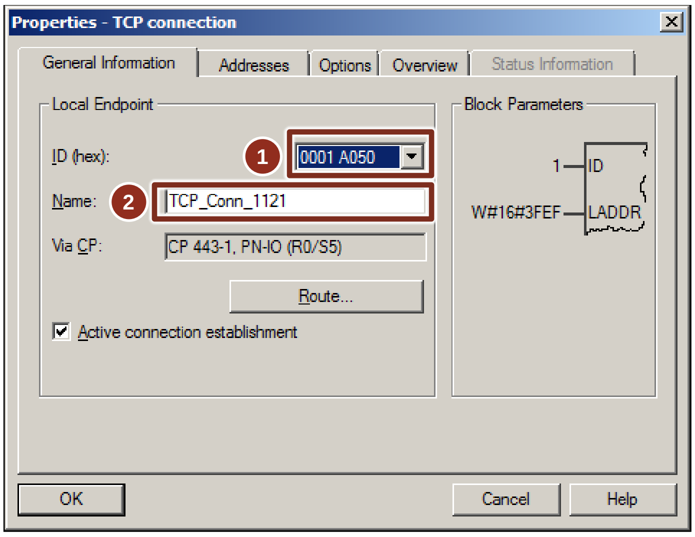
OK | Apply | Cancel | Help

5. Click the "OK" button. The properties window of the connection opens.

Parameterize connection 0A

1. Assign an ID (1) in the "General Information" tab. The ID is freely selectable.

Figure 4-13



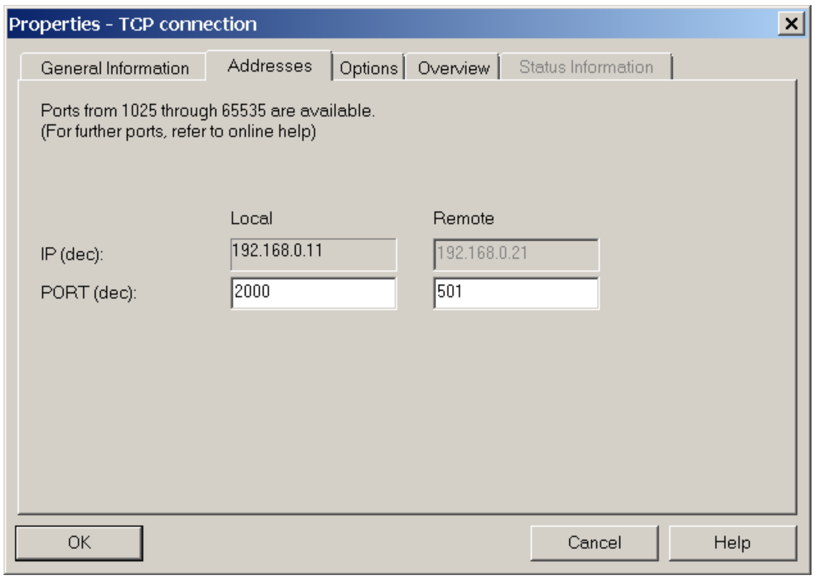
2. Under "Name:" (2) Enter a descriptive name to identify your TCP connection in the network.
3. Activate the checkbox "Active connection establishment".

Note The block parameters "Connection ID" (Local ID) and "Load address" are required later for the parameterization of the Modbus driver.

4. Click the "Route selection..." button ("Route...") and select the port of the foreign station (server) under "Partner" ("Remote").

5. Switch to the "Addresses" tab.
6. Assign the following port numbers and make sure that the corresponding IP addresses are selected:

Figure 4-14



Properties - TCP connection

Ports from 1025 through 65535 are available.
(For further ports, refer to online help)

	Local	Remote
IP (dec):	192.168.0.11	192.168.0.21
PORT (dec):	2000	501

7. Click "OK" to close the dialog.

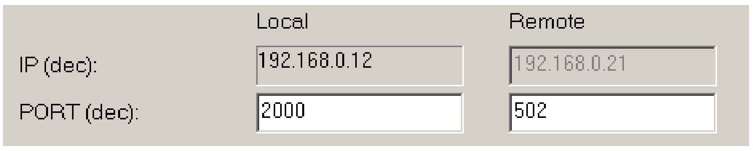
Note

The IP addresses in this dialog are not editable. If an incorrect IP address is displayed, exit the dialog by pressing the "OK" button and re-assign the ports of the external station
(steps 7 through 9).

Create 1A connection

1. Select the second CPU and add a new connection.
2. Activate the "Active connection establishment" checkbox in the "General Information" tab.
3. On the tab "General" click on the "Route..." button.
4. Select the port of the third-party station.
5. Switch to the "Addresses" tab
   Assign the following port numbers and make sure that the corresponding IP addresses are selected:

Figure 4-15



	Local	Remote
IP (dec):	192.168.0.12	192.168.0.21
PORT (dec):	2000	502

6. Click on "OK" to exit the dialog.

Note If necessary, you can adjust the port numbers in the "Addresses" tab.

Network setup in NetPro

The following application description assumes a PCS 7 project with a redundant automation system and one CP 443-1 each.

For a better understanding, the CPs of type 443-1 use the following IP numbers and subnets:

- First CP 443-1
  IP Address: 192.168.0.11
  Subnet: 255.255.255.0
- Second CP 443-1
  IP Address: 192.168.0.12
  Subnet: 255.255.255.0

4.4 Entry of the Blocks into the Master Data Library

Entry of the Modbus blocks

CAUTION

Check whether the following function block numbers (FB no.) and data block numbers (FC no.) are populated in your project and in the master data library:

- FB909
- FB908
- DB3

Once the block and DB numbers have been assigned, rewire the master data library (see section 7 "Renaming and Rewiring of Function Blocks").

If the numbers are not used, proceed further

Open the library "Modbus_TCP_CP_Red300_400" and copy the following blocks from the folder "S7 400 Client" into your master data library:

- FB909 (MB_REDCL)
- FB908 (MB_CPCLI)
- DB3 (LICENSE_DB)

Entry of the SIMATIC_NET_CP blocks

The Modbus blocks use the functions "AG_LSEND" and "AG_LRECV".

CAUTION

Check whether the following function numbers (FC no.) are populated in your project:

- FC50
- FC60

If the function numbers are assigned, perform rewiring in the master data library (see section 7 Renaming and Rewiring of Function Blocks")

If the numbers are not used, proceed with the engineering.

CAUTION

The use of Open Modbus/TCP Redundant V2.1 requires the blocks AG_LSEND and AG_LRECV from the SIMATIC NET library as of version V3.1.

4.5 Parameterize Modbus/TCP Communication

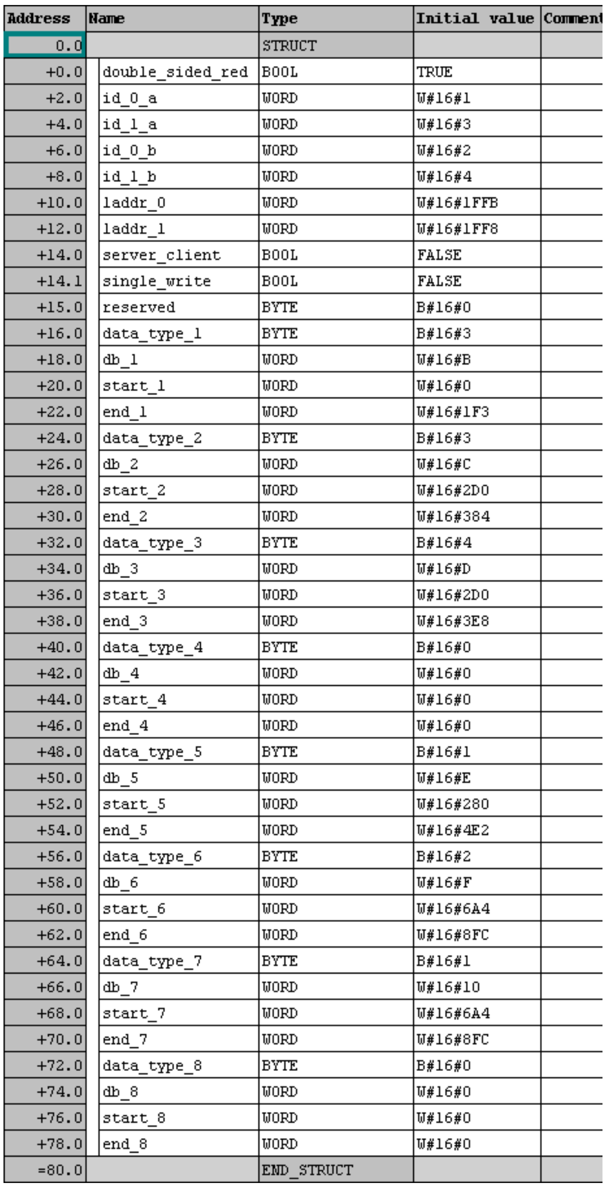
A connection configuration in NetPro is required for communication.

The assignment of the connections and the data for processing the Modbus/TCP telegrams are defined in a separate data block - the MODBUS_HPARAM_CP. First the connection-specific data and then the Modbus/TCP data are parameterized.

Structure MODBUS_HPARAM_CP

The following parameter data block shows an example of a DB for a double-sided redundant connection group. The data block MB_HPARAM_CP_DoubleRed and two others can be found in the folder "Parameter data blocks", in the library "Modbus_TCP_CP_Red300_400".

Figure 4-16



Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	double_sided_red	BOOL	TRUE	
+2.0	id_0_a	WORD	W#16#1	
+4.0	id_1_a	WORD	W#16#3	
+6.0	id_0_b	WORD	W#16#2	
+8.0	id_1_b	WORD	W#16#4	
+10.0	laddr_0	WORD	W#16#1FFB	
+12.0	laddr_1	WORD	W#16#1FF8	
+14.0	server_client	BOOL	FALSE	
+14.1	single_write	BOOL	FALSE	
+15.0	reserved	BYTE	B#16#0	
+16.0	data_type_1	BYTE	B#16#3	
+18.0	db_1	WORD	W#16#B	
+20.0	start_1	WORD	W#16#0	
+22.0	end_1	WORD	W#16#1F3	
+24.0	data_type_2	BYTE	B#16#3	
+26.0	db_2	WORD	W#16#C	
+28.0	start_2	WORD	W#16#2D0	
+30.0	end_2	WORD	W#16#384	
+32.0	data_type_3	BYTE	B#16#4	
+34.0	db_3	WORD	W#16#D	
+36.0	start_3	WORD	W#16#2D0	
+38.0	end_3	WORD	W#16#3E8	
+40.0	data_type_4	BYTE	B#16#0	
+42.0	db_4	WORD	W#16#0	
+44.0	start_4	WORD	W#16#0	
+46.0	end_4	WORD	W#16#0	
+48.0	data_type_5	BYTE	B#16#1	
+50.0	db_5	WORD	W#16#E	
+52.0	start_5	WORD	W#16#280	
+54.0	end_5	WORD	W#16#4E2	
+56.0	data_type_6	BYTE	B#16#2	
+58.0	db_6	WORD	W#16#F	
+60.0	start_6	WORD	W#16#6A4	
+62.0	end_6	WORD	W#16#8FC	
+64.0	data_type_7	BYTE	B#16#1	
+66.0	db_7	WORD	W#16#10	
+68.0	start_7	WORD	W#16#6A4	
+70.0	end_7	WORD	W#16#8FC	
+72.0	data_type_8	BYTE	B#16#0	
+74.0	db_8	WORD	W#16#0	
+76.0	start_8	WORD	W#16#0	
+78.0	end_8	WORD	W#16#0	
=80.0		END_STRUCT		

An additional parameter data block is required for each additional connection group. The parameters are divided into two areas.

The first area contains the connection parameters:

Table 4-1

Parameters	Description
double_sided_red	Specifies whether the redundancy is one-sided or two-sided.
id	Each connection configured in NetPro receives a unique connection ID, which must be entered here.
laddr	The parameter "laddr" is the base address of the CP from the HW config (E address). The configured value must be entered here. The value range for this parameter depends on the CPU. The parameters "id" and "laddr" can also be taken from the "Properties" screen of the TCP connection.
server_client	This parameter distinguishes between client and server operation. If the parameter is true, the operating mode "CP is server" is used. Otherwise the operating mode is "CP is Client".
single_write	In the operating mode "CP is Client" the function codes 5 and 6 are used with the parameter "single_write" = TRUE for write jobs with length 1. If "single_write" = FALSE, function codes 15 and 16 are used for all write jobs.

In the second area the parameters for the operating mode of the Modbus/TCP communication and the address mapping of the Modbus/TCP addresses to the SIMATIC addresses are defined.

Table 4-2

Parameters	Description
data_type_x	The parameter "data_type_x" specifies which Modbus/TCP data type is mapped in this DB. If the value 0 is entered in data_type_x, the corresponding data area will not be used. ID / Data type / Data width: 0 / Area not used / Bit 1 / Coils / Bit 2 / Inputs / Word 3 / Write holding / Word 4 / Input register / Word
db_x	The parameter "db_x" defines the data block in which the Modbus/TCP registers or bit values defined below are mapped. When using a Global DB, the DB number 0 is not allowed, since it is reserved for the system. DB number 1 to 65535 (W#16#0001 to W#16#FFFF) If a DataCollector block is used in CFC, the DB number 0 must be specified.
start_x, end_x	With "start_x" the first Modbus/TCP address, which is mapped in data word 0 of the DB, is specified. The parameter "end_x" defines the address of the last Modbus/TCP address. For register accesses, the data word number in the S7 DB in which the last Modbus/TCP address is entered is calculated according to the following formula: DBW number = (end_x – start_x) * 2 For bit accesses, the data byte number in the S7 DB in which the last Modbus/TCP address is entered is calculated according to the following formula: DBW number = (end_x – start_x) / 8 The defined data areas are not allowed to overlap. The parameter "end_x" must not be smaller than "start_x". In the event of an error, the initialization of the FB is terminated with an error. If both values are equal, 1 Modbus/TCP address (1 register or 1 bit value) is assigned. The data block must be created 2 bytes longer. These 2 reserve bytes are used for internal purposes.

The parameter DBs can be configured in two different ways:

- Parameterize Modbus/TCP communication with the Modbus/TCP Wizard.
- Parameterize Modbus/TCP communication manually

Parameterizing Modbus/TCP communication with the Modbus/TCP Wizard

The Modbus/TCP Wizard is a free tool that is available for the simple and convenient configuration of the MODBUS_HPARAM_CP parameter block. The Wizard creates a complete parameter block with connection parameters and Modbus/TCP parameters.

A detailed application description and the download can be found in the following entry:

SIMATIC S7-300/S7-400: Wizard for creating the connection data for Modbus/TCP communication (Modbus/TCP Wizard)

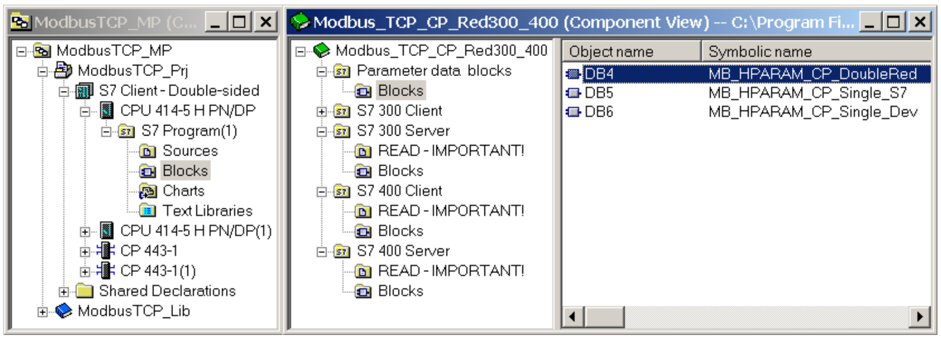
Note The predecessor product SIMATIC Modbus/TCP Red. V2 (article number 2XV9450-1MB11) the parameters are directly connected to the block.

The Modbus/TCP Wizard is an offline configuration aid. The telegram traffic is controlled at runtime via the Runtime parameters of the client.

Parameterize Modbus/TCP communication manually

1. Copy one of the parameter data blocks DB4 - DB6 from the folder "Parameter data blocks" in the "Modbus_TCP_CP_Red300_400" library. If the number of the DB is already assigned, the numbering of the DB can be adjusted.

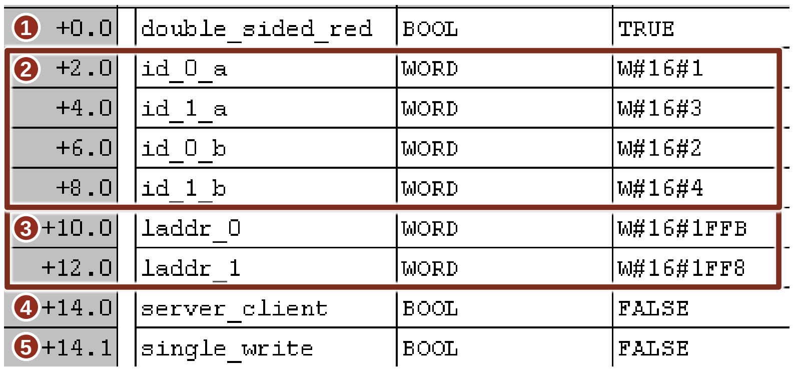
Figure 4-17



Object name	Symbolic name
DB4	MB_HPARAM_CP_DoubleRed
DB5	MB_HPARAM_CP_Single_S7
DB6	MB_HPARAM_CP_Single_Dev

2. Adjust the connection parameters in the DB MB_HPARAM_CP according to your configuration.

Figure 4-18

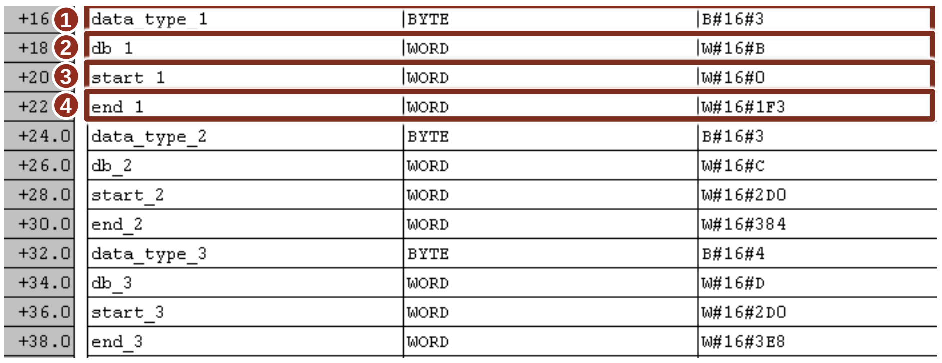


1 +0.0	double_sided_red	BOOL	TRUE
2 +2.0	id_0_a	WORD	W#16#1
+4.0	id_1_a	WORD	W#16#3
+6.0	id_0_b	WORD	W#16#2
+8.0	id_1_b	WORD	W#16#4
3 +10.0	laddr_0	WORD	W#16#1FFB
+12.0	laddr_1	WORD	W#16#1FF8
4 +14.0	server_client	BOOL	FALSE
5 +14.1	single_write	BOOL	FALSE

- Enter whether double-sided redundancy is involved (1)
- Enter the ID of the connections from NetPro (2).
- Enter the LADDR (3).
- Specify whether it is a client or server (4).
- Specify which function code is to be used for writing jobs with a length of one.

3. Adapt the Modbus/TCP parameters for the address mapping

Figure 4-19



+16 1	data type 1	BYTE	B#16#3
+18 2	db 1	WORD	W#16#B
+20 3	start 1	WORD	W#16#0
+22 4	end 1	WORD	W#16#1F3
+24.0	data_type_2	BYTE	B#16#3
+26.0	db_2	WORD	W#16#C
+28.0	start_2	WORD	W#16#2D0
+30.0	end_2	WORD	W#16#384
+32.0	data_type_3	BYTE	B#16#4
+34.0	db_3	WORD	W#16#D
+36.0	start_3	WORD	W#16#2D0
+38.0	end_3	WORD	W#16#3E8

- In the parameter "data_type_1", enter the data type to be transferred with the DB. (B#16#3 corresponds to the Holding Register) (1)
- Enter the number of the data block in the parameter "db_1", this can be in the value range from 1 to 65535 (2).
  If a data collector block is used in the program (variant B), the value 0 must be entered here.
- Enter the start address in the parameter "start_1" (3)
- Enter the end address in the parameter "end1" (4).

If the value B#16#0 is entered in the parameter "data_type_x", this data area is not used.

Note If changes are made to the DB "MODBUS_HPARAM_CP", the changes only become effective when the block MB_REDCL is reinitialized. The new initialization can be started via the input parameter "Init".

4.6 CFC Engineering

This application example describes a finished Modbus/TCP solution that is subdivided on a CFC chart into three chart partitions:

Figure 4-20

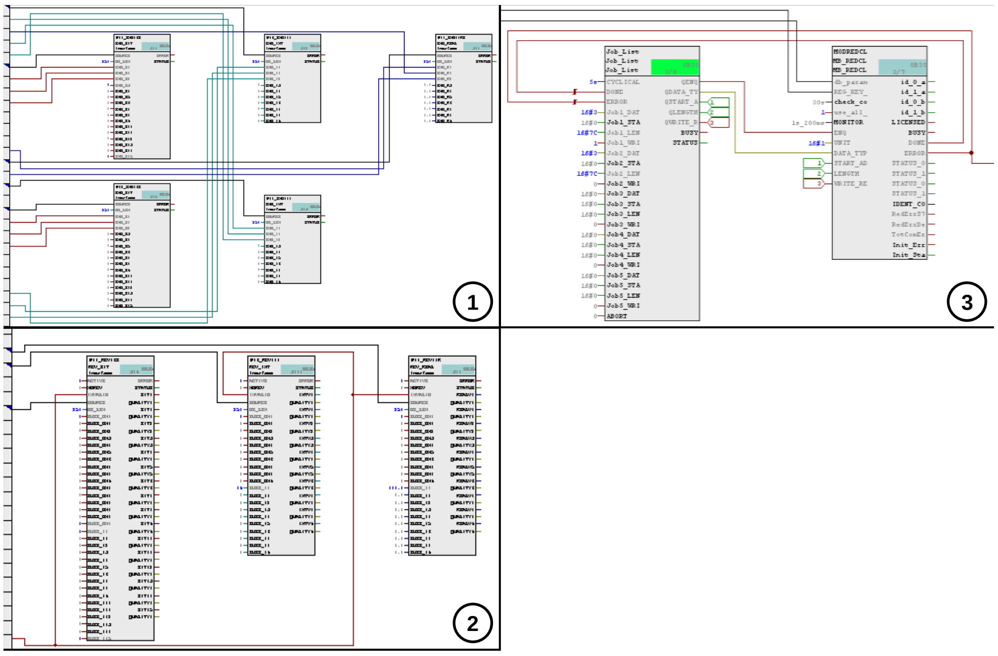


Table 4-3

CFC chart partition	Description
1	Transmit data
2	Received data
3	Modbus telegram management • Modbus driver • Job list

In this section, the Modbus/TCP solution is described in detail by means of a step-by-step guide.

Depending on the project requirements, two access options to the send and receive buffers (CFC chart partitions 1 and 2) are available. In the case of version A the accesses are via global DBs, while in the case of variant B they are via instance DBs.

The Modbus telegram management (CFC chart partition 3) is analogous for both versions in terms of function and differs only in terms of the preparation of the user data.

Version A: Allocation blocks for global data blocks

In version A the data to be sent and received are transmitted via global DBs. Only Modbus data type 3 (holding register; word) is supported. Sending of the data types bool, integer and real is also possible. These are converted into words before sending. Version A is recommended especially for mixed data formats in a DB and for large data volumes. Receive shunting modules contain the PCS 7 quality codes and in the event of an error or connection fault, substitute values can be switched on. Creation of the global data blocks is performed manually in accordance with project requirements.

Variant B: Data collector blocks with instance data blocks

In version B the data to be sent and received are transmitted via the instance DBs. The sending and receiving of data in the formats bool, integer, real, and word are supported (Modbus data types 1 (coils; bool), 2 (inputs; bool), 3 (holding register;) (word) and 4 (input register)). In this process it is not possible to mix the data in an instance DB. Only data of the type Word or Real or only Integer or only Bool can be sent and received per instance DB. The send and receive engineering takes place exclusively in the CFC environment.

Table 4-4

	Version A	Version B
Data blocks	Global DB	Instance DB
Data volume	Large data volumes	Small data volumes
Modbus data types	3 (holding register; word)	1 (coils; bool) 2 (inputs; bool) 3 (holding register; word) 4 (input register)
Formats	Bool, real and int These are converted into words before sending. Send/receive: 16 bool each Send/receive: 10 int each Send/receive: 10 real each	Bool, int, real, and word Remain in the original format for sending. Send/receive: 80 bool each Send/receive: 125 int each Send/receive: 63 real each Send/receive: 124 word received, 123 word sent
Mixed mode	The data formats in a DB can be mixed.	Mixed mode is not possible. Only bool or only real or only int or only word is possible for each DB.
Substitute values	Only for the receive blocks	Not available
PCS 7 Quality Code	PCS 7-Standard	Not available
Send and receive engineering	• CFC engineering • Manual creation of the data blocks.	CFC engineering

Note The flow sequence in the CFC charts must be adapted manually.

1. Send allocation blocks
2. Job_list
3. MB_REDCL
4. Receive allocation blocks

4.6.1 Version A: Allocation Blocks for Global Data Blocks

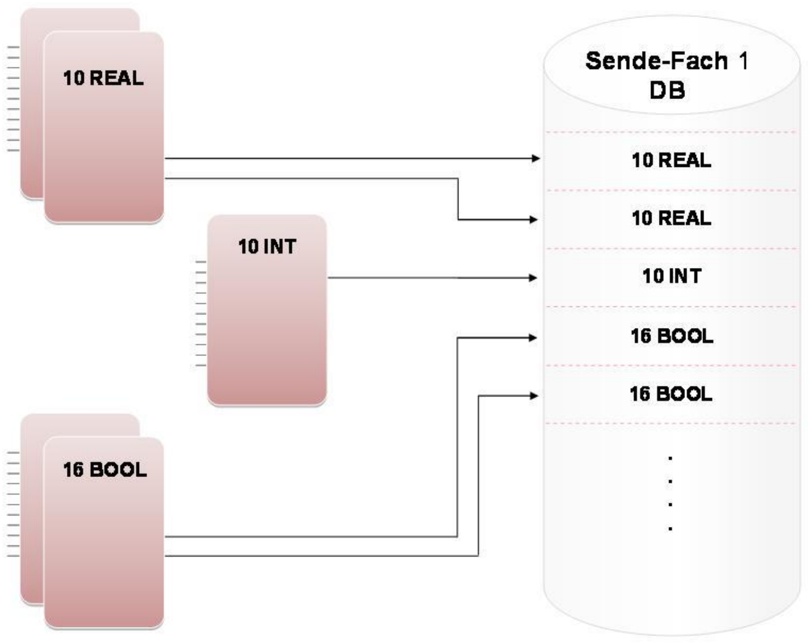
Allocation blocks

The allocation blocks are the interfaces for the user program. They enable access to the data to be sent or received of the send or receive memories of global data blocks.

Send allocation blocks

Blocks are available for routing 10 real, 10 integer and 16 boolean values into the transmission compartment.

Figure 4-21

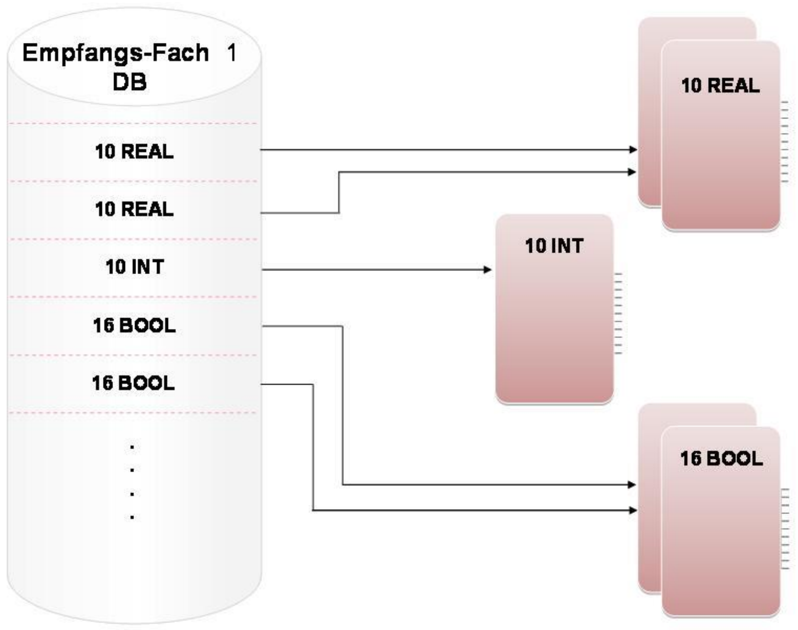


Receive allocation blocks

The receive jumper modules operate in a controlled manner, i.e. only when the receive function block sets the output "Receive new data" to 1, do the jumper modules output the data. In addition, there is a possibility of offering substitute values in the event of an error (for example, if there is a disconnection). The receiving jumper modules contain PCS 7 Quality Codes.

Blocks are available for the output of 10 real, 10 int and 16 bool values from the receive buffer.

Figure 4-22



Creation of the global data block

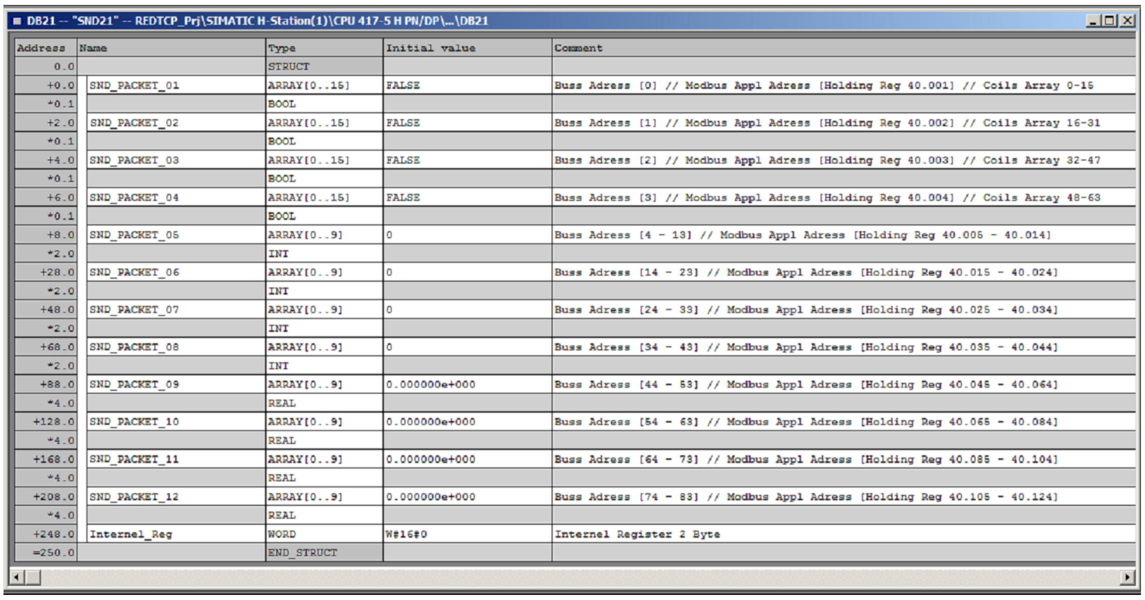
In version A the data (bool, real, int) are converted to word format.
In this example:

- 16 bool → into a coils array (1 word)
- 10 integers → into an int array (10 words)
- 10 real → into a real array (10 words)

CAUTION	In addition to the user data, the global DB requires two bytes (1 word) for internal processing. This word must not be changed **Maximum user data per job** (in this example): Receive: 248 bytes → DB length = 250 bytes Send: 246 bytes → DB length = 248 bytes

1. Configure the global send data block in accordance with the following Figure. In the Comment column you can see the referencing between the bus and the Modbus application addresses (holding register).

Figure 4-23 Transmission assignment between bus and Modbus application addresses

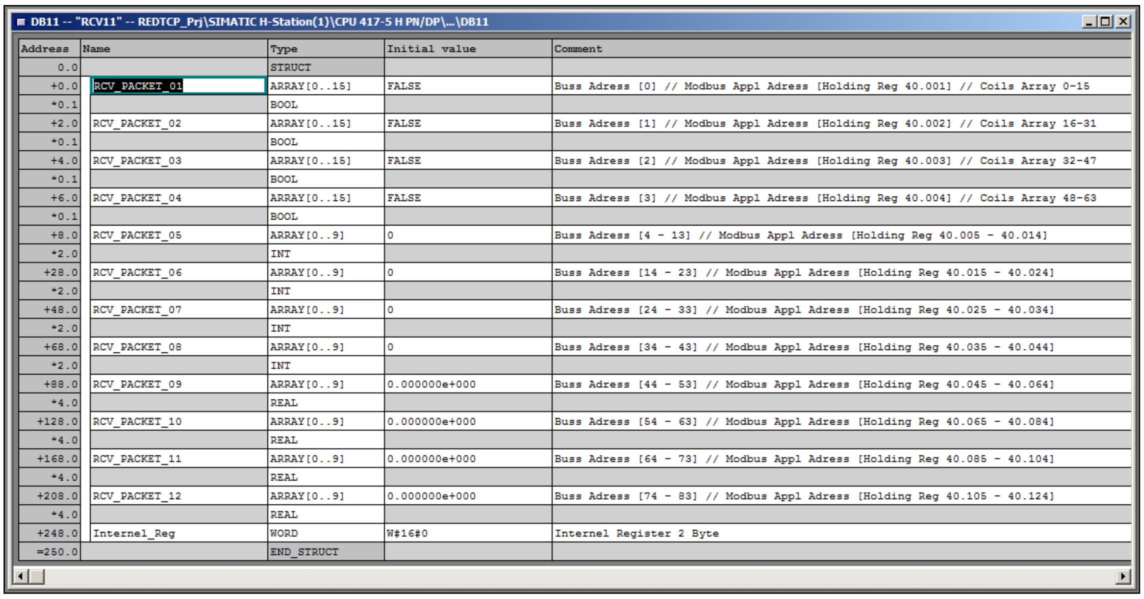


DB21 -- "SND21" -- REDTCP_Prj\SIMATIC H-Station(1)\CPU 417-5 H PN/DP\...\DB21

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	SND_PACKET_01	ARRAY[0..15]	FALSE	Buss Adress [0] // Modbus Appl Adress [Holding Reg 40.001] // Coils Array 0-15
*0.1		BOOL		
+2.0	SND_PACKET_02	ARRAY[0..15]	FALSE	Buss Adress [1] // Modbus Appl Adress [Holding Reg 40.002] // Coils Array 16-31
*0.1		BOOL		
+4.0	SND_PACKET_03	ARRAY[0..15]	FALSE	Buss Adress [2] // Modbus Appl Adress [Holding Reg 40.003] // Coils Array 32-47
*0.1		BOOL		
+6.0	SND_PACKET_04	ARRAY[0..15]	FALSE	Buss Adress [3] // Modbus Appl Adress [Holding Reg 40.004] // Coils Array 48-63
*0.1		BOOL		
+8.0	SND_PACKET_05	ARRAY[0..9]	0	Buss Adress [4 - 13] // Modbus Appl Adress [Holding Reg 40.005 - 40.014]
*2.0		INT		
+28.0	SND_PACKET_06	ARRAY[0..9]	0	Buss Adress [14 - 23] // Modbus Appl Adress [Holding Reg 40.015 - 40.024]
*2.0		INT		
+48.0	SND_PACKET_07	ARRAY[0..9]	0	Buss Adress [24 - 33] // Modbus Appl Adress [Holding Reg 40.025 - 40.034]
*2.0		INT		
+68.0	SND_PACKET_08	ARRAY[0..9]	0	Buss Adress [34 - 43] // Modbus Appl Adress [Holding Reg 40.035 - 40.044]
*2.0		INT		
+88.0	SND_PACKET_09	ARRAY[0..9]	0.000000e+000	Buss Adress [44 - 53] // Modbus Appl Adress [Holding Reg 40.045 - 40.064]
*4.0		REAL		
+128.0	SND_PACKET_10	ARRAY[0..9]	0.000000e+000	Buss Adress [54 - 63] // Modbus Appl Adress [Holding Reg 40.065 - 40.084]
*4.0		REAL		
+168.0	SND_PACKET_11	ARRAY[0..9]	0.000000e+000	Buss Adress [64 - 73] // Modbus Appl Adress [Holding Reg 40.085 - 40.104]
*4.0		REAL		
+208.0	SND_PACKET_12	ARRAY[0..9]	0.000000e+000	Buss Adress [74 - 83] // Modbus Appl Adress [Holding Reg 40.105 - 40.124]
*4.0		REAL		
+248.0	Internel_Reg	WORD	W#16#0	Internel Register 2 Byte
=250.0		END_STRUCT		

2. Configure the global receive data block according to the following figure. In the Comment column you can see the referencing between the bus and the Modbus application addresses (holding register).

Table 4-5: Send referencing between the bus and Modbus application addresses



DB11 -- "RCV11" -- REDTCP_Prj\SIMATIC H-Station(1)\CPU 417-5 H PN/DP\...\DB11

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	RCV_PACKET_01	ARRAY[0..15]	FALSE	Buss Adress [0] // Modbus Appl Adress [Holding Reg 40.001] // Coils Array 0-15
*0.1		BOOL		
+2.0	RCV_PACKET_02	ARRAY[0..15]	FALSE	Buss Adress [1] // Modbus Appl Adress [Holding Reg 40.002] // Coils Array 16-31
*0.1		BOOL		
+4.0	RCV_PACKET_03	ARRAY[0..15]	FALSE	Buss Adress [2] // Modbus Appl Adress [Holding Reg 40.003] // Coils Array 32-47
*0.1		BOOL		
+6.0	RCV_PACKET_04	ARRAY[0..15]	FALSE	Buss Adress [3] // Modbus Appl Adress [Holding Reg 40.004] // Coils Array 48-63
*0.1		BOOL		
+8.0	RCV_PACKET_05	ARRAY[0..9]	0	Buss Adress [4 - 13] // Modbus Appl Adress [Holding Reg 40.005 - 40.014]
*2.0		INT		
+28.0	RCV_PACKET_06	ARRAY[0..9]	0	Buss Adress [14 - 23] // Modbus Appl Adress [Holding Reg 40.015 - 40.024]
*2.0		INT		
+48.0	RCV_PACKET_07	ARRAY[0..9]	0	Buss Adress [24 - 33] // Modbus Appl Adress [Holding Reg 40.025 - 40.034]
*2.0		INT		
+68.0	RCV_PACKET_08	ARRAY[0..9]	0	Buss Adress [34 - 43] // Modbus Appl Adress [Holding Reg 40.035 - 40.044]
*2.0		INT		
+88.0	RCV_PACKET_09	ARRAY[0..9]	0.000000e+000	Buss Adress [44 - 53] // Modbus Appl Adress [Holding Reg 40.045 - 40.064]
*4.0		REAL		
+128.0	RCV_PACKET_10	ARRAY[0..9]	0.000000e+000	Buss Adress [54 - 63] // Modbus Appl Adress [Holding Reg 40.065 - 40.084]
*4.0		REAL		
+168.0	RCV_PACKET_11	ARRAY[0..9]	0.000000e+000	Buss Adress [64 - 73] // Modbus Appl Adress [Holding Reg 40.085 - 40.104]
*4.0		REAL		
+208.0	RCV_PACKET_12	ARRAY[0..9]	0.000000e+000	Buss Adress [74 - 83] // Modbus Appl Adress [Holding Reg 40.105 - 40.124]
*4.0		REAL		
+248.0	Internel_Reg	WORD	W#16#0	Internel Register 2 Byte
=250.0		END_STRUCT		

Insertion of the send blocks and connection

1. After creation of the global data blocks, open an CFC chart.
2. Drag the following blocks into CFC chart partition 1:

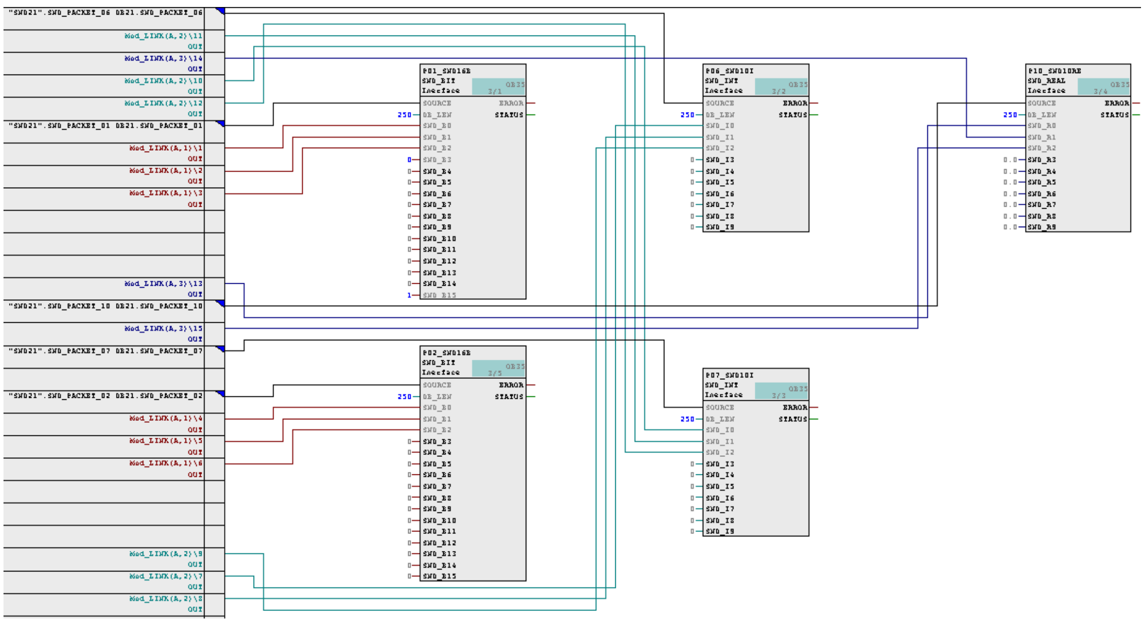
Table 4-6

Quantity	Block name	Rename as	Description
4	SND_BIT	P01_SND16B P02_SND16B P03_SND16B P04_SND16B	Send blocks for bool
4	SND_INT	P05_SND10I P06_SND10I P07_SND10I P08_SND10I	Send blocks for integer
4	SND_REAL	P09_SND10R P10_SND10R P11_SND10R P12_SND10R	Send blocks for Real

3. Link the SOURCE inputs of the blocks with the corresponding arrays of the global send data block.
4. Adjust the length of the global data block at the DB_LEN inputs. As all blocks access the same global data block in this example, the value for each block is the same:
   248 bytes of user data + 2 bytes for internal processing = 250 bytes.

Note No substitute values can be connected to the transmitter module.

Figure 4-24



Insertion of the receive blocks and connection

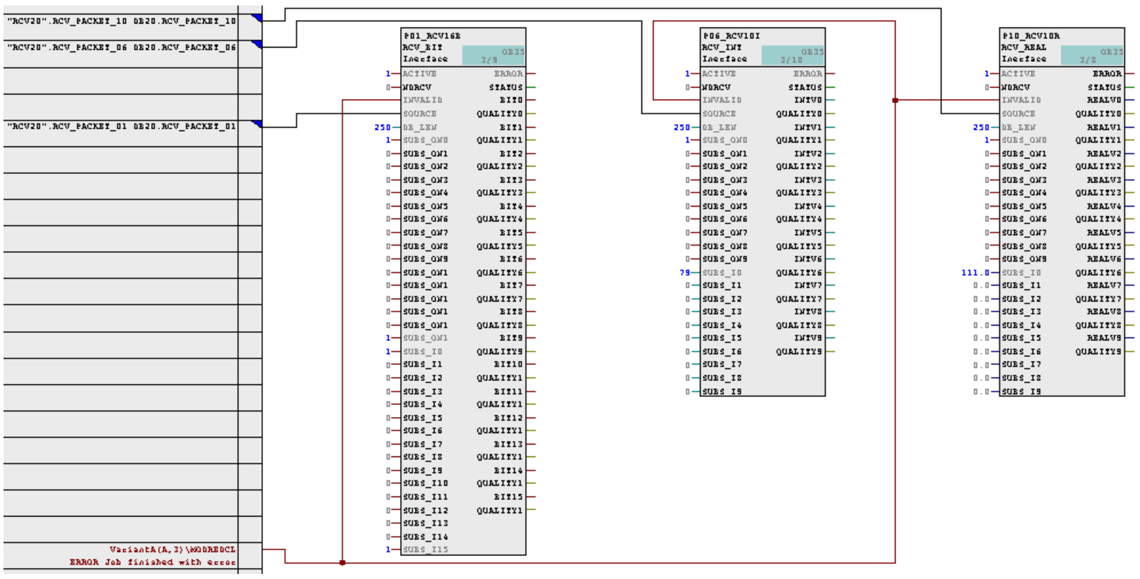
1. Drag the following blocks into CFC chart partition 2:

Table 4-7

Quantity	Block name	Rename as	Description
4	RCV_BIT	P01_RCV16B P02_RCV16B P03_RCV16B P04_RCV16B	Receive blocks for bool
4	RCV_INT	P05_RCV10I P06_RCV10I P07_RCV10I P08_RCV10I	Receive blocks for integer
4	RCV_REAL	P09_RCV10R P10_RCV10R P11_RCV10R P12_RCV10R	Receive blocks for Real

2. Link the SOURCE inputs of the blocks to the corresponding arrays of the global receive data block.
3. Set the ACTIVE inputs to 1.
4. Adjust the length of the global data block at the DB_LEN inputs. As all blocks access the same global data block in this example, the value for each block is the same:
   248 bytes of user data + 2 bytes for internal processing = 250 bytes.
5. Via the inputs SUBS_ON0 through SUBSONx you can enable for each individual value substitute offering in the event of an error.

Figure 4-25


"RCV20".RCV_PACKET_10 DB20.RCV_PACKET_10
"RCV20".RCV_PACKET_06 DB20.RCV_PACKET_06
"RCV20".RCV_PACKET_01 DB20.RCV_PACKET_01
P01_RCV16B
RCV_BIT
P06_RCV10I
RCV_INT
P10_RCV10R
RCV_REAL
ERROR Job finished with error

4.6.2 Variant B: Data Collector Blocks with Instance Data Blocksn

Insertion of the receive blocks and connection

1. Open a CFC chart.
2. Drag the following blocks into your CFC chart:

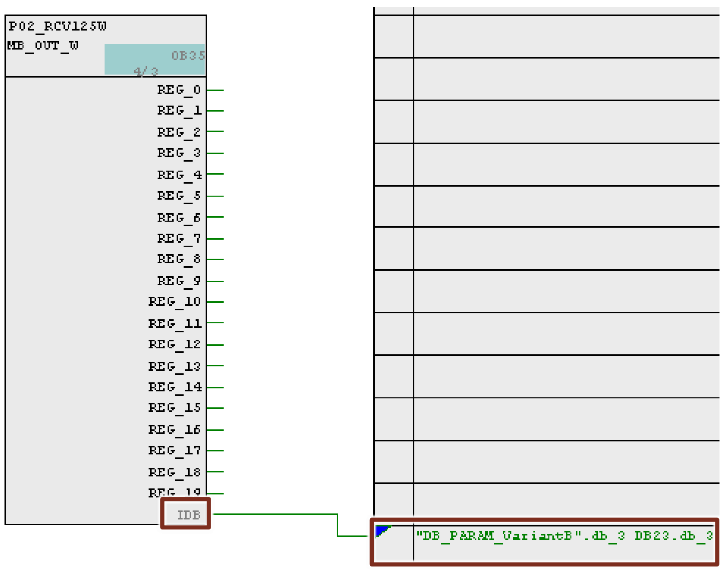
Table 4-8

Quantity	Block name	Rename as	Max. values	Description
0	MB_OUT_B	Block not being used	80 bool	Receive blocks for bool
0	MB_OUT_I	Block not being used	125 Integer	Receive blocks for integer
0	MB_OUT_R	Block not being used	63 real	Receive blocks for Real
1	MB_OUT_W	P02_RCV125W	125 words	Receive blocks for Words

Note The OUT blocks read the data from the data block and are therefore the receive blocks.

3. Connect the outputs IDB of the blocks with the input "db_1" to "db_8" in the parameter data block.

Figure 4-26



Insertion of the send blocks and connection

1. Drag the following blocks into your CFC chart:

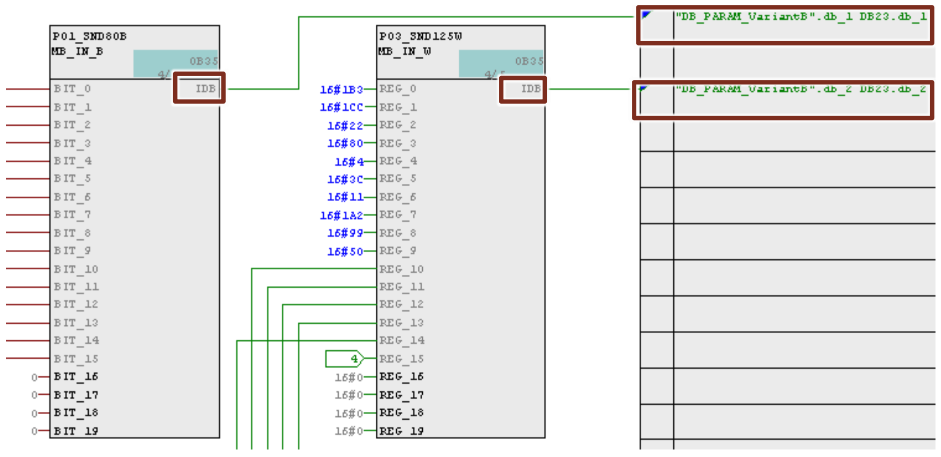
Table 4-9

Quantity	Block name	Rename as	Max. values	Description
1	MB_IN_B	P01_SND80B	80 bool	Send blocks for bool
0	MB_IN_I	Block not being used	125 Integer	Send blocks for integer
0	MB_IN_R	Block not being used	63 real	Send blocks for Real
1	MB_IN_W	P02_SND125W	125 words	Send blocks for word

Note The IN blocks forward the data to the data block and are therefore the send blocks.

2. Connect the outputs "IDB" of the blocks with the parameters "db_1" to "db_8" in the parameter data block.

Figure 4-27

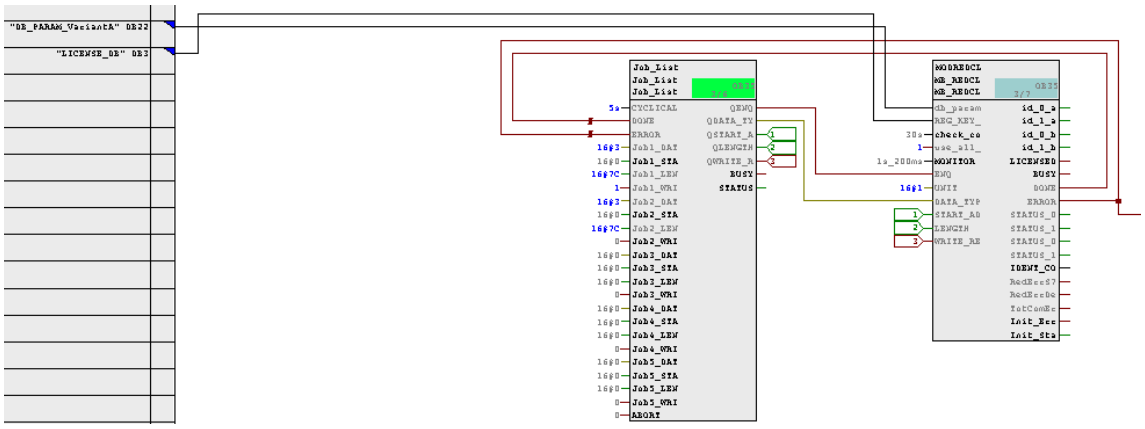


4.6.3 Modbus Telegram Management

In CFC segment plan 3 the two blocks Job_List and MB_REDCL and the Modbus telegram management are used for configuration.

The concept of Modbus communication is explained diagrammatically below and the block module parameterizations are described in detail.

Figure 4-28



Block parameters from MB_REDCL

The inputs and outputs of the Modbus driver are described in detail in the following table.

The following table is a component part of the manual:
"SIMATIC Modbus/TCP Redundant communication via CP443-1 in H Systems".

Table 4-10

Inputs/outputs	Description
db_param	The parameter db_param denotes the number of the data block MODBUS_HPARAM_CP. In this parameter data block the connection and ModbusModbus specific parameters are stored which are necessary for the communication between the CPU and the coupling partner. The value range for this parameter is CPU-dependent. 0 cannot be used as a DB number because it is reserved for the system. The DB number is entered in plain text in the form "DBxy". Each Modbus block instance requires its own parameter data block.
REG_KEY_DB	The block must be licensed on each CPU. With the correct input of the registration key to this parameter the block is licensed and the Modbus communication can be used without restrictions. The data block number of the block containing the registration key is specified. Multiple registration key can be entered one after another in the DB. The Modbus block browses the DB for the matching registration key.
check_conn_cycle	This parameter defines the time interval at which the configured connections are checked with "AG_CNTRL". The time can be set in the

Inputs/outputs	Description
	seconds raster. The default value is 30 seconds. The results of the connection test are displayed at the STATUS outputs. With "use_all_conn = FALSE" the monitoring can be switched off with the setting 0 seconds. If check_conn_cycle = 0 is parameterized, it is not possible to detect a connection error via this connection without any telegram traffic. It is recommended to set this parameter to > 0 ms.
use_all_conn	This parameter specifies via how many connections the Modbus telegrams are to be sent. With FALSE, the Modbus telegrams are only sent via one connection. If the parameter is TRUE, the Modbus telegrams are sent via all configured connections.
MONITOR	The monitoring time MONITOR monitors the receipt of data from the link partner on the currently active connections. The shortest settable time is 20 ms. A monitoring time of approx. 1.5 seconds is recommended. The receipt of the complete response telegram is monitored by the time MONITOR. If the monitoring time is exceeded, the activated job is terminated with errors. The time is started after completed sending of the request telegram and after receipt of the complete data.
ENQ	Data transfer is initiated with a positive edge. The request telegram is generated with the values of the input parameters UNIT, DATA_TYPE, START_ADDRESS, LENGTH and WRITE_READ. A new job can only be started if the previous one was completed with DONE or ERROR.
UNIT	The input is to be set in accordance with requirements. The FB accepts this value in the request telegram and checks it when it receives the response. Please note that some coupling partners expect a specific UNIT number. The UNIT number can also be used to address different stations behind a ModbusGateway.
DATA_TYPE	The DATA_TYPE parameter indicates which Modbus data type is being processed with the current telegram. The following values are allowed: • Coils B#16#1 • Inputs B#16#2 • Holding Register B#16#3 • Input Register B#16#4 The various data types are directly related to the function codes used.

Datentyp	DATA_TYPE	Funktion	Länge	Single_write	Funktionscode
Coils	1	lesen	beliebig	irrelevant	1
Coils	1	schreiben	1	TRUE	5
Coils	1	schreiben	1	FALSE	15
Coils	1	schreiben	>1	irrelevant	15
Inputs	2	lesen	beliebig	irrelevant	2
Holding Register	3	lesen	beliebig	irrelevant	3
Holding Register	3	schreiben	1	TRUE	6
Holding Register	3	schreiben	1	FALSE	16
Holding Register	3	schreiben	>1	irrelevant	16
Input Register	4	lesen	beliebig	irrelevant	4

Inputs/outputs	Description
START_ADDRESS	The parameter START_ADDRESS specifies the first MODBUS address that will be written or read.
LENGTH	The parameter LENGTH determines the number of MODBUS values that are written or read. With reading functions, a maximum of 125 registers are possible per telegram for holding and input registers. A maximum of 2000 bits is possible for coils and inputs. For write functions, there is a maximum of 123 registers for holding registers and a maximum of 1968 bits for coils. The registers or bit values processed with a request telegram must lie within a DB.
WRITE_READ	This parameter defines whether a read or write function is to be executed. If the input/output has the value FALSE, it is a reading function. The value TRUE defines a write function.
id_0_A id_1_A id_0_B id_1_B	A connection ID is assigned for each configured connection in STEP 7 (NetPro). The connection ID describes precisely the connection of the CPU via the CP to the link partner. The numbers from the connection project planning in the parameter data block are displayed here. id_0_A denotes the connection from CP0 to the coupling partner/node A id_1_A denotes the connection from CP1 to the coupling partner/node A id_0_B denotes the connection from CP0 to the coupling partner/node B id_1_B denotes the connection from CP1 to the coupling partner/node B Connection 0A is the default connection and it is imperative that it is configured. If the coupling partner is set up standalone, then only the parameters id_0_A and id_1_A are required.
LICENSED	If this output is set to TRUE, the MODBUS block is licensed on this CPU. If the output has the state FALSE, no or an incorrect license string has been entered.
BUSY	If this output is set, AG_LSEND or AG_LRECV is active.
DONE	The activated job was completed without errors on at least 1 connection. In the case of a read function, the response data from the server was already entered in the DB, in the case of a write function, the response to the request telegram was received from the server.
ERROR	If this output is set, an error has been detected on all active connections. use_all_conn = FALSE: In the event of a protocol error, ERROR is set immediately. In the event of a connection error, all configured connections are checked and ERROR is not set until all connections have errors. use_all_conn = TRUE: If this output is set, an error has been detected on all configured connections. The error numbers are displayed at the STATUS outputs.
STATUS_0A STATUS_1A STATUS_0B STATUS_1B	The outputs STATUS_x show the error number if ERROR is set, status information for the corresponding connection if ERROR is not set.
IDENT_CODE	After the start-up of the CPU, an 18-digit identification code is displayed at this parameter, with which the license code (REG_KEY) for the MODBUS communication is requested.
RedErrS7	The output RedErrS7 = TRUE indicates a redundancy error on the SIMATIC side.

Inputs/outputs	Description
	In the case of single-sided redundancy, this means that the CP0 or CP1 connection has failed. With redundancy on both sides, both connections of CP0 or both connections of CP1 have failed.
RedErrDev	The output RedErrDev = TRUE indicates a redundancy error on the side of the coupling partner. In the case of single-sided redundancy, this means that the connection from junction A to the CP0 or CP1 connection has failed. With redundancy on both sides, both connections to node A or both connections to node B of the coupling partner have failed.
TotComErr	The output TotComErr indicates a complete communication loss with TRUE, i.e. all configured connections are disturbed.
Init_Error	If an error occurred during initialization, this is indicated with Init_Error = TRUE.
Init_Status	The output Init_Status indicates the error number when Init_Error is set.
Init	The Modbus block is initialized with a positive edge at parameter Init. Initialization can be performed only if no job is in progress. This must be ensured in the program with ENQ = FALSE and BUSY = FALSE. The Init bit must not be set if CPU0 is in STOP state. Otherwise, 8085 is displayed by RDSYSST, since no serial number can be read out. **Note** During initialization, the configured connections are terminated and re-established.

Block of the parameter Job_List

The inputs, outputs, status displays and error displays are described in detail in the following tables.

These tables are a component part of the manual:
Job_List, Data Collector and routing blocks for convenient use of the Modbus/TCP blocks.

Note

A Modbus job with more than 240 bytes length is processed in 2 jobs (PUT/GET) via the backplane bus with the CP443-1.

Based on this, it is recommended to use only a maximum of 118 (word) registers (=236 bytes; results in 239 bytes incl. 3 bytes for header) instead of 124 registers per job (16#7C).

Table 4-11: Input parameters

Parameters	Data type	Description
CYCLICAL	TIME	> 0 ms: Cyclic processing of the job list
DONE	BOOL	Response message of the Modbus/TCP block that the last job was able to be executed error-free
ERROR	BOOL	Response message of the Modbus/TCP block that the last job was completed with errors
Job1_DATA_TYPE	BYTE	1. Job: Data type, 0 = no processing
Job1_START_ADDRESS	WORD	1. Job: Start address
Job1_LENGTH	WORD	1. Job: Length
Job1_WRITE_READ	BOOL	1. Job: Write/read
Job2_DATA_TYPE	BYTE	2. Job: Data type, 0 = no processing
Job2_START_ADDRESS	WORD	2. Job: Start address
Job2_LENGTH	WORD	2. Job: Length
Job2_WRITE_READ	BOOL	2. Job: Write/read
Job3_DATA_TYPE	BYTE	3. Job: Data type, 0 = no processing
Job3_START_ADDRESS	WORD	3. Job: Start address
Job3_LENGTH	WORD	3. Job: Length
Job3_WRITE_READ	BOOL	3. Job: Write/read
Job4_DATA_TYPE	BYTE	4. Job: Data type, 0 = no processing
Job4_START_ADDRESS	WORD	4. Job: Start address
Job4_LENGTH	WORD	4. Job: Length
Job4_WRITE_READ	BOOL	4. Job: Write/read
Job5_DATA_TYPE	BYTE	5. Job: Data type, 0 = no processing
Job5_START_ADDRESS	WORD	5. Job: Start address
Job5_LENGTH	WORD	5. Job: Length
Job5_WRITE_READ	BOOL	5. Job: Write/read
ABORT	BOOL	TRUE: Cancellation of the current processing of the job list

Table 4-12: Output parameters

Parameters	Data type	Description
QENQ	BOOL	Job start for the Modbus/TCP block
QDATA_TYPE	BYTE	DATA_TYPE of the current job
QSTART_ADDRESS	WORD	START_ADDRESS of the current job
QLENGTH	WORD	LENGTH of the current job
QWRITE_READ	BOOL	WRITE_READ of the current job
BUSY	BOOL	Job list is being executed
STATUS	WORD	Block status information

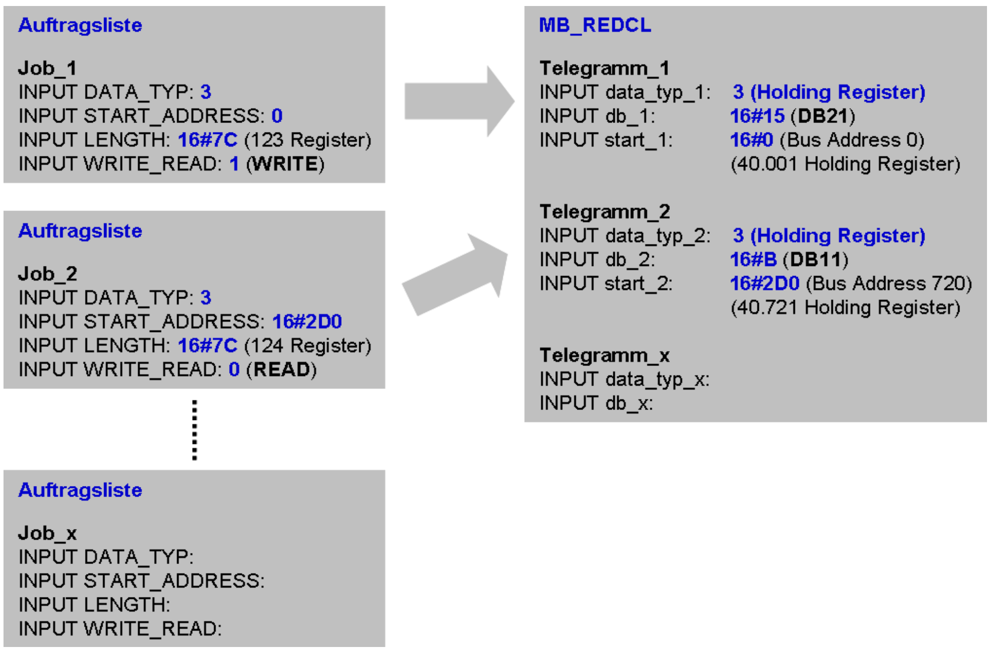
Table 4-13: Status and error display

Status	Meaning	Notes
16#A089	The set cycle time has expired, while the processing list is still running.	The job list will be reprocessed immediately after completion of the last job.

Modbus configuration principle

The diagrammatic representation shows the principle of the Modbus configuration:

Figure 4-29



Auftragsliste

Job_1
INPUT DATA_TYP: **3**
INPUT START_ADDRESS: **0**
INPUT LENGTH: **16#7C** (123 Register)
INPUT WRITE_READ: **1** (**WRITE**)

Auftragsliste

Job_2
INPUT DATA_TYP: **3**
INPUT START_ADDRESS: **16#2D0**
INPUT LENGTH: **16#7C** (124 Register)
INPUT WRITE_READ: **0** (**READ**)

Auftragsliste

Job_x
INPUT DATA_TYP:
INPUT START_ADDRESS:
INPUT LENGTH:
INPUT WRITE_READ:

MB_REDCL

Telegramm_1
INPUT data_typ_1: **3 (Holding Register)**
INPUT db_1: **16#15** (**DB21**)
INPUT start_1: **16#0** (Bus Address 0)
(40.001 Holding Register)

Telegramm_2
INPUT data_typ_2: **3 (Holding Register)**
INPUT db_2: **16#B** (**DB11**)
INPUT start_2: **16#2D0** (Bus Address 720)
(40.721 Holding Register)

Telegramm_x
INPUT data_typ_x:
INPUT db_x:

In its standard form the Modbus driver MB_REDCL is able to process up to four telegrams.

The data type is predefined and the start address of the communications registers and the data buffer used (DB) are specified.

The Job_List block is used to parameterize the accesses to the Modbus offsets of the registers. In addition, the length and the number of written and read registers are defined.

The command to read or write is specified via WRITE_READ.

Wiring of the blocks

1. Drag the following blocks into your CFC chart partition 3:

Table 4-14

Quantity	Block name	Rename as…	Description
1	Job_list	Order list	Job management
1	MB_REDCL	MODREDCL	Modbus/TCP driver block

Note

You can obtain the Job_List block and a detailed description on the following page:
https://support.industry.siemens.com/cs/en/en/view/62830463

2. Link the blocks in accordance with the following table:

Table 4-15

Job_list		MB_REDCL
Outputs		**Inputs**
QENQ	→	ENQ
QDATA_TYPE	→	DATA_TYPE
QSTART_ADDRESS	→	START_ADDRESS
QLENGTH	→	LENGTH
QWRITE_READ	→	WRITE_READ
Inputs		**Outputs**
DONE	←	DONE
ERROR	←	ERROR

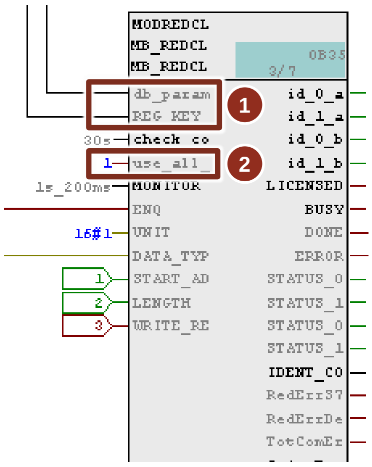
Table 4-16

MB_REDCL		RCV_BIT, RCV_INT, RCV_REAL
Outputs		**Inputs**
ERROR	→	INVALID

Parameterizing the block MB_REDCL

3. Connect the input "db_param" (1) of the MB_REDCL with the parameter DB, which contains the connection information and the address mapping ("MODBUS_HPARAM_CP").

Figure 4-30



4. Connect the input "REG_KEY" (1) to the "LICENSE-DB" (DB3).
5. Enter 1 at input "use_all_conn" (2) for sending the telegram via all configured connections.

Note

When configuring the parameter "use_all_conn = true", the telegrams are sent via all configured connections. This results in a higher communication load than if only one connection is used.

Therefore, ensure that no AS communication overload occurs. Further information can be found in Section 8: Performance.

Note

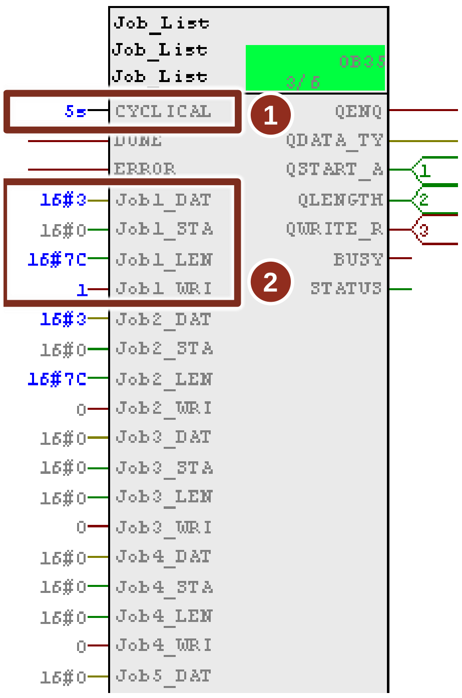
When configuring the parameter "use_all_conn = false" and a redundancy switch-over, a temporary loss of connection may occur. To prevent the loss of the communication link, the parameters restoration time "check_conn_cycle" and monitoring time "Monitor" should be appropriately selected. Further information can be found in Section 8: Performance.

In addition, the time in which a disconnection is detected should be adjusted accordingly with the Keep Alive-Time parameter. This parameter can be found in the properties of the CP443-1 in the HW Config.

Parameterizing the Job_List block

6. At input "CYCLICAL", enter the time (1) in which the job list is to be processed.

Figure 4-31



```
          Job_List
          Job_List
          Job_List
     5s — CYCLICAL           QENQ
          DONE               QDATA_TY
          ERROR              QSTART_A  1
   16#3 — Job1_DAT           QLENGTH   2
   16#0 — Job1_STA           QWRITE_R  3
  16#7C — Job1_LEN           BUSY
      1 — Job1_WRI           STATUS
   16#3 — Job2_DAT
   16#0 — Job2_STA
  16#7C — Job2_LEN
      0 — Job2_WRI
   16#0 — Job3_DAT
   16#0 — Job3_STA
   16#0 — Job3_LEN
      0 — Job3_WRI
   16#0 — Job4_DAT
   16#0 — Job4_STA
   16#0 — Job4_LEN
      0 — Job4_WRI
   16#0 — Job5_DAT
```

7. Enter a three for the data type 3 (Word) (2) at the input "Job1_DATA_TYPE".
8. At input "Job1_START_ADDRESS" (2), enter the job start address for the predefined area in the data block.
9. Enter the job length of the range at input "Job1_LENGH" (2).

10. At input "Job1_WRITE_READ" (2), select whether to write or read.

Orders 3 to 5 are not used in this example.

Note For unused orders ("Jobs"), enter a 0 at the "Job_Data_Type" input.

5 Diagnosis

The Modbus driver MB_REDCL monitors the four possible communication connections. The error messages are displayed in hexadecimal format at the four status outputs "STATUS_0A" to "STATUS_1B".

Error messages

The Modbus FBs use the standard blocks SFC20, SFC24, SFC51, SFC52 FC50 and FC60. The messages of these blocks are passed on unchanged to the Status_X outputs.

Table 5-1

Error messages from MB_REDCL and MB_REDSV		
Status [hex]	**Event text**	**Remedy**
A002	The check of "start_x" was faulty.	Contact Product Support.
A003	A DB onto which the Modbus addresses are to be mapped is too short. Minimum length in bytes: - for registers: (end-x – start_x + 1) * 2 + 2 for bit values: (end_x – start_x) / 8 + 1 + 2	Extend the DB.
	CP actual client: Incorrect call parameters.	CP actual client: Correct the job parameter START_ ADDRESS or LENGTH.
	CP actual server: Incorrect address area in the client's request telegram. The client accesses an area that was not parameterized in S7. The CP responds with an exception log	CP actual server: Change the client request or adjust the data areas in the DB that is connected with the input parameter db_param of the MB_REDCL or MB_REDSV.
A004	Only CP is client: An invalid combination of DATA_TYPE and WRITE_READ has been specified.	Correct the call parameters. Only data types 1 and 3 can be written.

Error messages from MB_REDCL and MB_REDSV		
Status [hex]	**Event text**	**Remedy**
A005	CP actual client: A non-permitted value has been entered at the parameter LENGTH. CP actual server: The number of registers/bits in the request telegram is not permitted. The CP responds with an exception telegram. Range of values: Read coils/inputs: 1 to 2000 Write coils: 1 to 1968 Read registers: 1 to 125 Write holding register: 1 to 123	CP actual client: Correct the parameter LENGTH. CP actual server: Change the quantity in the request telegram.
A006	The area specified using DATA_TYPE, START_ADDRESS and LENGTH does not exist in data_type_1 to data_type_8. CP actual server: The CP responds with an exception telegram.	CP actual client: Correct the parameterization combination DATA_TYPE, START_ADDRESS and LENGTH. CP actual server: Change the client request or correct the parameterization at data_type_x.
A007	CP actual client: An invalid monitoring time on the MONITOR has been parameterized. A value ≥ 20 ms must be entered.	Correct the parameterization.
A008	Within the parameterized monitoring time MONITOR the activated AG_RECV reports no reception, e.g. partner not ready. The connection will be broken and re-established.	Check the settings and, where applicable, the error messages of the link partner. Check whether the communication partner might possibly require a specific UNIT identifier exhaust air.
A009	CP actual client: The transaction identifier received (TI) is not the same as the one sent. The connection will be broken and re-established.	Check the data of the link partner with the aid of a telegram recording.
A00A	CP actual client: The UNIT received is not the same as the one sent. The connection will be broken and re-established.	Check the data of the link partner with the aid of a telegram recording.

Error messages from MB_REDCL and MB_REDSV		
Status [hex]	**Event text**	**Remedy**
A00B	CP actual client: The function code received is not the same as the one sent. CP actual server: An invalid function code has been received. The CP responds with an exception telegram.	CP actual client: Check the data of the link partner with the aid of a telegram recording. CP actual server: Change the request of the client. The FB MB_REDSV processes the function codes 1, 2, 3, 4, 5, 6, 15 and 16.
A00C	The number of bytes received does not match the number of registers. The CP responds with an exception telegram. The connection will be broken and re-established.	Check the data of the link partner with the aid of a telegram recording.
A00D	Only if CP is client: The register/bit address or the number of registers/bits in the response telegram is not the same as that in the request telegram.	Check the data of the link partner with the aid of a telegram recording.
A00E	The length specification in the Modbus-specific telegram does not match the specifications of the number of registers/bits or of the number of bytes in the telegram. The FB discards the data. The connection will be broken and re-established.	Check the data of the link partner with the aid of a telegram recording.
A00F	A protocol identifier not equal to 0 has been received. The connection will be broken and re-established.	Check the data of the link partner with the aid of a telegram recording.
A010	In parameters db_1 to db_8, a DB number was assigned twice.	Correct the parameterization at db_x.
A011	An invalid value has been specified at the input parameter DATA_TYPE (permitted values: 1–4).	Correct the call parameters.
A012	The parameterized areas data_type_1 and data_type_2 overlap.	Correct the parameterization. The data areas are not allowed to have common registers.
A013	The parameterized areas data_type_1 and data_type_3 overlap.	Correct the parameterization. The data areas are not allowed to have common registers.

Error messages from MB_REDCL and MB_REDSV		
Status [hex]	**Event text**	**Remedy**
A014	The parameterized areas data_type_1 and data_type_4 overlap.	Correct the parameterization. The data areas are not allowed to have common registers.
A015	The parameterized ranges data_type_1 and data_type_5 overlap.	Correct the parameterization. The data areas are not allowed to have common registers.
A016	The parameterized ranges data_type_1 and data_type_6 overlap.	Correct the parameterization. The data areas are not allowed to have common registers.
A017	The parameterized areas data_type_1 and data_type_7 overlap.	Correct the parameterization. The data areas are not allowed to have common registers.
A018	The parameterized ranges data_type_1 and data_type_8 overlap.	Correct the parameterization. The data areas are not allowed to have common registers.
A019	One of the parameters db_x was set to 0, although the corresponding data_ type_x is parameterized with > 0. DB0 must not be used because it is reserved for the system.	Correct the parameterization at db_x to > 0. If you are using a data collector FB, check the parameterization in CFC.
A01A	Wrong length in the header of the Modbus telegram: 3 to 253 bytes are permitted. The connection is broken and re-established	Check the data of the link partner with the aid of a telegram recording.
A01B	CP actual server and function code 5: Invalid status received for coil. An exception telegram is sent.	Check the data of the link partner with the aid of a telegram recording.
A01E	Invalid data have been received that could not be assigned. The connection will be broken and re-established.	Check the error messages of the link partner. Where applicable, check the data by means of a telegram recording.
A01F	The FB MB_REDCL or MB_REDSV is in an operating state that is not permitted.	Contact Product Support.
A020	No monitoring time or one that is too short is configured for AG_CNTRL at check_conn_cycle.	Correct the parameterization. A monitoring time > 1 s must be parameterized.

Error messages from MB_REDCL and MB_REDSV		
Status [hex]	**Event text**	**Remedy**
A022	A parameter data block for clients was parameterized at the MB_REDSV or a parameter data block for servers was parameterized at the MB_REDCL.	Correct the parameterization
A023	The parameterized areas data_type_2 and data_type_3 overlap each other.	Correct the parameterization. The data areas are not allowed to have common registers.
A024	A024 The parameterized data_type_2 and data_type_4 areas overlap.	
A025	A025 The parameterized data_type_2 and data_type_5 areas overlap.	
A026	A026 The parameterized data_type_2 and data_type_6 areas overlap.	
A027	A027 The parameterized data_type_2 and data_type_7 areas overlap.	
A028	A028 The parameterized data_type_2 and data_type_8 areas overlap.	
A034.	A034 The parameterized data_type_3 and data_type_4 areas overlap.	
A035	A035 The parameterized data_type_3 and data_type_5 areas overlap.	
A036	A036 The parameterized data_type_3 and data_type_6 areas overlap.	
A037	A037 The parameterized data_type_3 and data_type_7 areas overlap.	
A038	A038 The parameterized data_type_3 and data_type_8 areas overlap.	
A045	A045 The parameterized data_type_4 and data_type_5 areas overlap.	
A046	A046 The parameterized data_type_4 and data_type_6 areas overlap.	
A047	A047 The parameterized data_type_4 and data_type_7 areas overlap.	

Error messages from MB_REDCL and MB_REDSV		
Status [hex]	**Event text**	**Remedy**
A048	A048 The parameterized data_type_4 and data_type_8 areas overlap.	
A056	A056 The parameterized data_type_5 and data_type_6 areas overlap.	
A057	A057 The parameterized data_type_5 and data_type_7 areas overlap.	
A058	A058 The parameterized data_type_5 and data_type_8 areas overlap.	
A067	A067 The parameterized data_type_6 and data_type_7 areas overlap.	
A068	A068 The parameterized data_type_6 and data_type_8 areas overlap.	
A078	A078 The parameterized data_type_7 and data_type_8 areas overlap.	
A07A	A non-permitted value has been specified at parameter id_x (value range from 1 to 64).	Correct the parameterization in the DB that is connected to the input parameter db_param of MB_REDCL or MB_REDSV.
A07C	An invalid value was specified for the parameter data_type_x (valid values are 0 to 4).	
A07D	The parameter data_type_1 does not contain an entry. The parameter area _1 is the initial area and must be parameterized.	
A07E	The number of the instance DB of the block MB_REDCL or MB_REDSV has been specified at db_x.	
A07F	The DB specified at db_param is not a Modbus parameter DB.	Correct the parameterization at input db_param
A080	The Modbus block has not yet been initialized.	After the transfer of the IDB into the CPU the Modbus block must be initialized with Init = TRUE.
A081	Only if CP is client and function code 5: The response telegram data do not echo the request.	Check the data of the link partner with the aid of a telegram recording.
A082	Only if CP is client and function code 6: The register value received is not the same as the one sent.	Check the data of the link partner with the aid of a telegram recording.

Error messages from MB_REDCL and MB_REDSV		
Status [hex]	**Event text**	**Remedy**
A083	CP actual client: A new job has been triggered while the previous job is still running. The job will not be executed. This is status information. The ERROR bit is not set. An attempt was made to initialize the block manually while a job is still running. CP actual server: An attempt was made to initialize the block while ENR = TRUE was set.	CP actual client: Do not start a new job until the previous job has been completed with DONE = TRUE, NDR = TRUE or ERROR = TRUE. Wait with the initialization until there is no longer a job in progress. CP actual server: Set ENR = FALSE
A085	An internal error has occurred in the licensing check on account of an unauthorized write access.	Check whether there are no unauthorized write accesses to the license DB in the S7 program. The structure of the REG_KEY must not be changed. Contact product support. if necessary.
A086	An attempt was made to write to a write-protected data block.	Remove the write protection of the DB or use another DB.
A090	The block MB_REDCL or MB_REDSV is not yet licensed for this CPU. This is status information. The ERROR bit is not set. Modbus communication will also run unlicensed.	Read out the IDENT_CODE identification string for this CPU and request the registration key.
A091	Only if CP is client: An exception telegram with exception code 1 has been received in response.	The link partner does not support the requested function.
A092	Only if CP is client: An exception telegram with exception code 2 has been received in response. Access to a non-existent or invalid address has been performed by the link partner.	Correct the LENGTH or START_ADDRESS for the FB call.
A093	Only if CP is client: An exception telegram with exception code 3 has been received in response.	The coupling partner cannot process the received telegram (e.g. it does not support the requested length).
A094	Only if CP is client: An exception telegram with exception code 4 has been received in response.	In its current status the link partner cannot process the telegram received.

Error messages from MB_REDCL and MB_REDSV		
Status [hex]	**Event text**	**Remedy**
A095	Only if CP is client: An exception telegram with an unknown exception code has been received in response.	Check the link partner's error messages and, where applicable, check the data by means of a telegram recording.
A0FF	At present the connection is not ready for communication. The status message is the result of a previous connection error.	Check to see which error occurred and Determine its cause. Check the connections. Where applicable, correct the value at check_conn_cycle. The error can also occur if a CP is used that does not support the AG_CNTRL.
FFFF	The connection is not configured.	If this connection is to be used, it must be configured in the parameter DB, which is connected to the input parameter db_param of the MB_REDCL or MB_REDSV.

6 Hardware Changes of an H System During Operation

The functionality "Hardware changes during operation" is available in the H-System without additional settings.

There are some systems that must not be shut down during operation. Reasons for this are, for example, the complexity of the automated process or high start-up costs. Nevertheless, expansion or modification may be required.

Using H-CiR (Configuration in RUN), it is possible to make certain configuration changes to the Hardware Configuration in the operating state RUN.

H-CiR cannot be used until the function is enabled in HW Config and has been loaded once by way of an AS STOP.

A detailed description of the possible hardware changes for an H system can be found in the manual "SIMATIC Process Control System PCS 7 Engineering System":

https://support.industry.siemens.com/cs/ww/en/view/109754984

Communication delays

A communication delay can occur, if hardware changes are loaded into an H System during operation.

The communication delay with H-CiR depends on the current process status, the communication load, and the system parameters.

The maximum permissible communication delay can be set in the properties of the CPU under "H parameters". The delay times can be calculated on the basis of the system parameters.

Note Perform H-CiR only in non-critical processes statuses.

The necessity for H-CiR can be reduced by configuring reserve I/O modules and installing resistances at the inputs and outputs in order to prevent diagnostic messages.

Note It is recommended that all configured connections are used for communication ("use_all_conn = true"). In the event that only one connection is used, a temporary loss of connection may occur during HW change loading.

Further information can be found in Section 8.4: Scenario: HW Config. change loading.

Further information on hardware changes during operation can be found in the system manual "S7-400H Highly Available Systems":

https://support.industry.siemens.com/cs/ww/en/view/82478488

7 Renaming and Rewiring of Function Blocks

If the numbers of the standard functions are already used in a project or the number range is reserved for other applications, the internally called functions FC50, FC60 or FC10 and the function blocks MB_REDCL and MB_REDSV can be rewired. The blocks MB_CPCLI and MB_CPSRV cannot be rewired because they are called in the block Privacy protected blocks MB_REDCL and MB_REDSV.

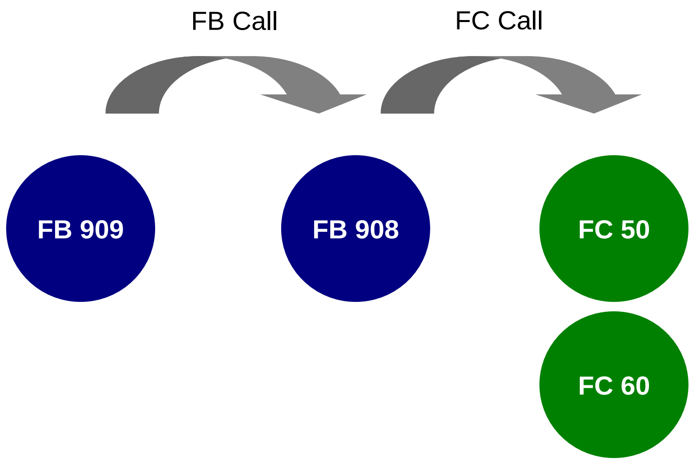
Rewired means that not only are the function block numbers changed, but also the internal calls that call these blocks or functions.

MB_REDCL / MB_REDSV

The function blocks MB_REDCL (FB909) and MB_REDSV (FB907) of the Modbus library internally call the function blocks MB_CPCLI (FB908) or MB_CPSRV (FB906), which are also a component part of the Modbus library.
The blocks MB_CPCLI and MB_CPSRV additionally call the blocks "AG_LSEND" (FC50) and "AG_LRECV" (FC60), which are part of the SIMATIC NET library.

The function code names FC50 and FC60 are frequently populated by other functions (for example, I/O modules), so that in the event of careless copying of the Modbus library blocks the latter are overwritten and malfunctions can occur as a result.

Figure 7-1



7.1 Renaming

You can rename if the add-on blocks do not call any additional function blocks (FBs) or functions (FCs) that are supplied and have the same numbers that are already assigned to other FBs/FCs in your project. In the current application, it is only possible to rename the MB_REDCL or MB_REDSV blocks, since they are not called internally in another block.

1. Create a new PCS 7-program
2. Copy the following Modbus/TCP and SIMATIC NET blocks into your new S7 program:
   - MB_REDCL
   - MB_REDSV
   - MB_CPCLI
   - MB_CPSRV
   - AG_LSEND
   - AG_LRECV
3. Assign a new block number for MB_REDCL, for example.
4. Adapt these changes in the symbol table.
5. Copy the blocks into your actual PCS 7 program.

7.2 Rewiring

The rewiring ensures that the internal fixed code calls are also adapted by certain FBs or FCs. A rewiring of the blocks MB_CPCLI (FB908) and MB_CPSRV (FB906) is unfortunately not possible, because they are called in the block Privacy protected blocks MB_REDCL (FB909) and MB_REDSV (FB907).

CAUTION

Not all functions or function blocks have to be rewired.

Even if only some of these blocks are rewired, this sequence must be complied with.

In this example, new numbers are assigned for the following FBs and FCs:

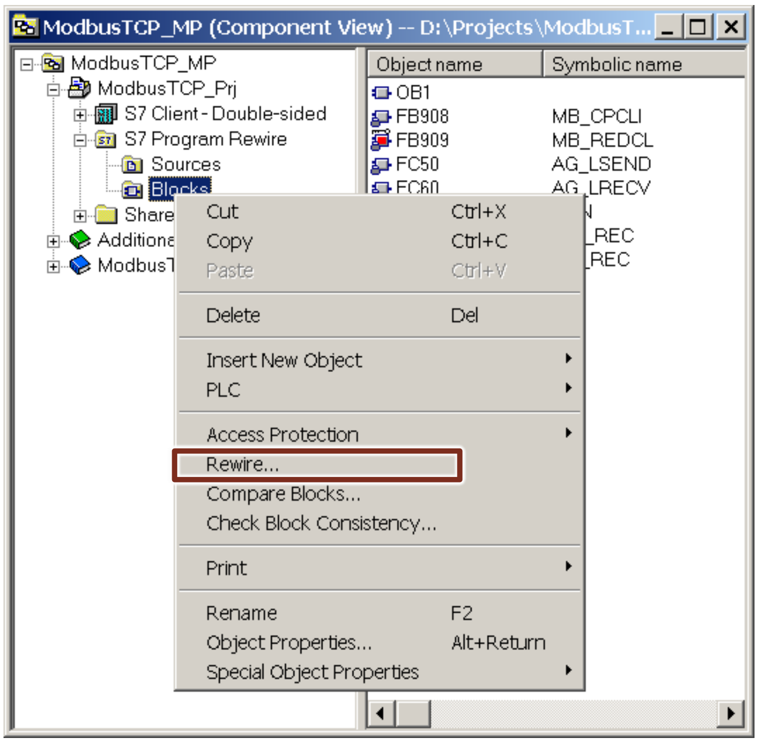
Table 7-1

Block name	from library	from		to
MB_REDCL	Modbus/TCP	FB909	→	FB1000
AG_LSEND	SIMATIC NET	FC50	→	FC550
AG_LRECV	SIMATIC NET	FC60	→	FC660

Procedure

1. Create a new PCS 7 program.
2. Copy the following Modbus/TCP and SIMATIC NET blocks into your new PCS 7 program:
   - MB_REDCL
   - MB_REDSV
   - MB_CPCLI
   - MB_CPSRV
   - AG_LSEND
   - AG_LRECV
3. Select the "Blocks" folder and choose "Rewire..." from the context menu.

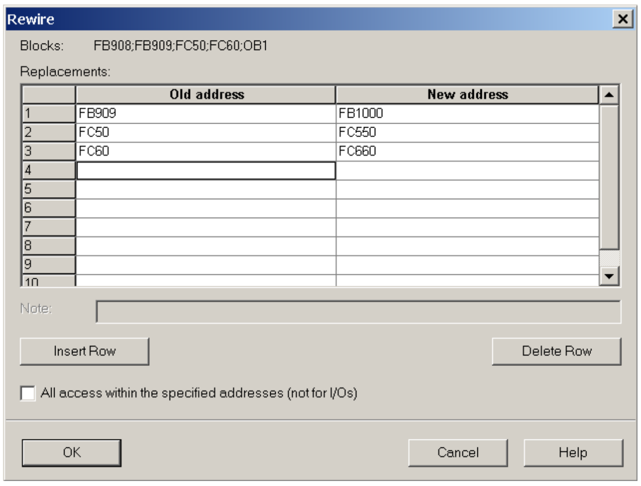
Figure 7-2



Object name	Symbolic name
OB1	
FB908	MB_CPCLI
FB909	MB_REDCL
FC50	AG_LSEND
FC60	AG_LRECV
	N
	_REC
	_REC

4. In the "Rewire" dialog, enter the FB and FC numbers you want to rewire in the first column. Enter the new FB and FC numbers in the second column.

Figure 7-3



Rewire

Blocks: FB908;FB909;FC50;FC60;OB1

Replacements:

	Old address	New address
1	FB909	FB1000
2	FC50	FC550
3	FC60	FC660
4		
5		
6		
7		
8		
9		
10		

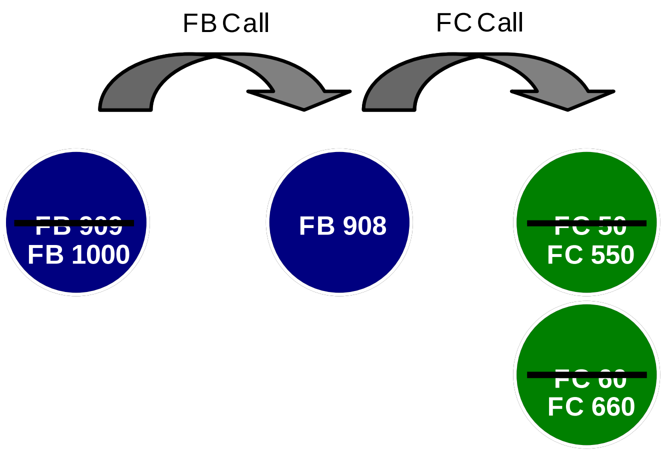
Note:

☐ All access within the specified addresses (not for I/Os)

5. Click the "OK" button to confirm the changes.
6. Adapt these changes in the symbol table.
7. Check the result.
   To do so, highlight the changed block and open its properties. Switch to the "Calls" tab. In this register the called blocks are listed.
8. Copy the blocks into your project.

After rewiring the blocks, the blocks are called as follows:

Figure 7-4



Note With this method all other add-on libraries can be implemented in the PCS 7 project independently of their FB/FC numbers.

8 Performance

The performance of the standardized communication with Modbus/TCP is shown in three scenarios.

Below is a list of the hardware and software used to perform the performance measurements:

- S7 412-5H PN/DP
- CPU 412-5HK06-0AB0 (firmware V6.0.1)
- CP 443-1EX20-0EX0 (firmware V2.0.57)
- PCS 7 V8.0 Upd1

CAUTION

The performance of the Modbus TCP communication is directly dependent upon the AS communication load. For a Modbus TCP communication, therefore, sensible quantity structures must be used for the number of connections and the number of packets to be transmitted.

If too many configured MB_REDCL block instances are called too quickly in an AS, there is a risk of overloading the AS communication.

Excel tool for estimating the AS communication load

Using Excel, you can generate an individual estimate of the AS communication load and use the results to ensure the communication behavior quality or to implement optimization measures. Excel can be downloaded from the following link:
https://support.industry.siemens.com/cs/en/en/view/109778857

Note

Diagnostics: S7-400-CPUs Connection statistics

The current connection statistics of the AS can be displayed in the module status of the CPU. The data flow rate of the communication connections can be monitored here.

8.1 Scenario 1

In the first scenario, two blocks MB_REDCL are configured for the test case. A block for sending and a block for receiving the data. One job is processed per send and receive block (up to 4 jobs are possible). Each job uses a separate data block. The jobs are processed via two Job_List blocks.

For the first measurement 124 words are received and 123 words are sent.

For the second measurement the two MB_REDCL blocks and the two Job_List blocks are copied and pasted. As a result, double the number of words are received and sent (248 words/246 words).

Two further MB_REDCL blocks and two Job_List blocks are copied at a time for each further measurement.

For communication, all configured connections are used ("use_all_conn = TRUE").

Schematic measurement setup

1. Measuring

2 * MB_REDCL + 2 * Job_List
2 * data blocks (1 * for send; 1 * for receive)
Receive 124 words and send 123 words

2. Measurement

4 * MB_REDCL + 4 * Job_List
4 * data blocks (2 * for send; 2 * for receive)
Receive 248 words and send 246 words

3. Measurement

6 * MB_REDCL + 6 * Job_List
6 * data blocks (3 * for send; 3 * for receive)
Receive 372 words and send 369 words

4. Measurement

8 * MB_REDCL + 8 * Job_List
8 * data blocks (4 * for send; 4 * for receive)
Receive 496 words and send 492 words

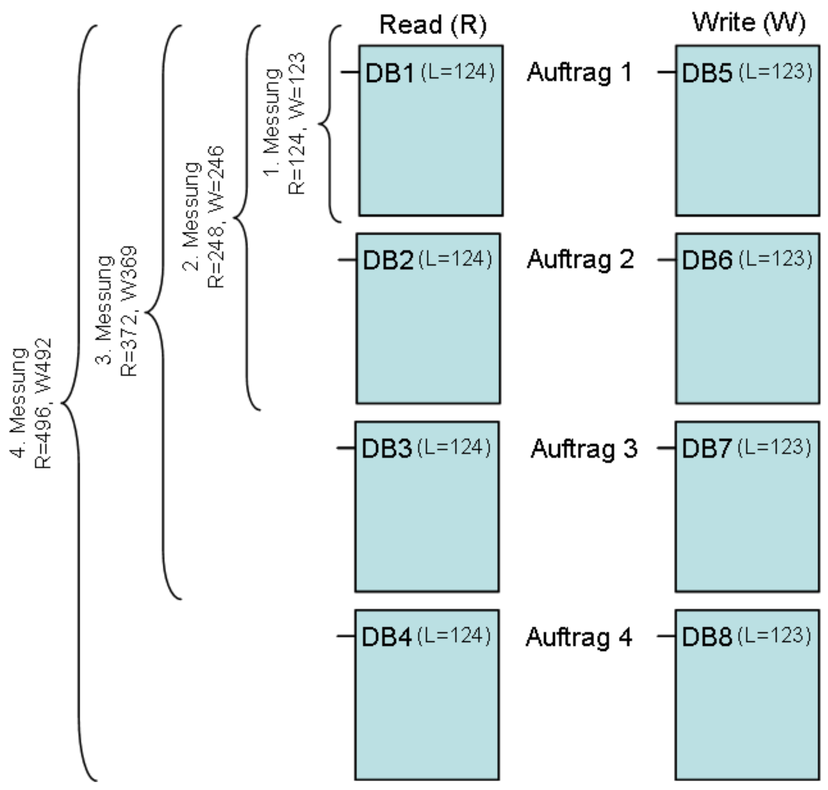


Figure 8-1

Measurement results

Table 8-1

OB35	Cyclical	Monitor	check_conn	READ words	WRITE words	Register amount	Comm. Speed
100 ms	350 ms	1200 ms	5000 ms	124	123	257	599 ms
100 ms	450 ms	1200 ms	5000 ms	248	246	494	1398 ms
100 ms	550 ms	1200 ms	5000 ms	372	369	741	2098 ms
100 ms	650 ms	1200 ms	5000 ms	496	492	988	3200 ms

8.2 Scenario 2A

In scenario 2A, two MB_REDCL blocks are configured; one block for sending and one block for receiving. Each block can process up to 4 jobs. Each job requires a data block. The jobs are processed via two Job_List blocks.

In the first measurement 124 words are read and 123 words are sent via one job.

In the second measurement, within the MB_REDCL block a second job is parameterized and via the latter 124 words are also read and 123 words sent, so that in total the read and sent words are doubled. The number of MB_REDCL and number of Job_List blocks are maintained, however, the number of data blocks is increased to 2.

Parameterized again in the third measurement is an additional job that also reads 124 words and sends 123 words.

Parameterized again in the fourth measurement is an additional job that reads 124 words and sends 123 words.

For communication, all configured connections are used ("use_all_conn = TRUE").

Schematic measurement setup

1. Measuring

2 * MB_REDCL + 2 * Job_List
2 * data blocks (1 * for send; 1 * for receive)
Receive 124 words and send 123 words

2. Measurement

2 * MB_REDCL + 2 * Job_List
4 * data blocks (2 * for send; 2 * for receive)
Receive 248 words and send 246 words

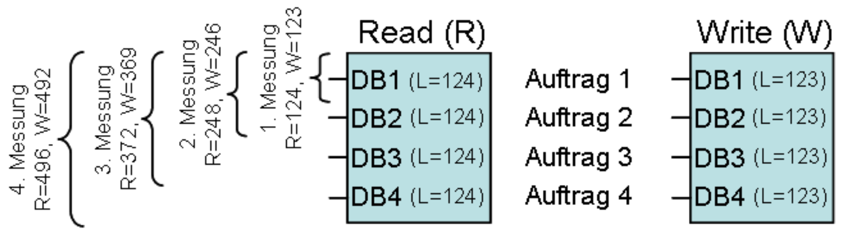
3. Measurement

2 * MB_REDCL + 2 * Job_List
6 * data blocks (3 * for send; 3 * for receive)
Receive 372 words and send 369 words

4. Measurement

2 * MB_REDCL + 2 * Job_List
8 * data blocks (4 * for send; 4 * for receive)
Receive 469 words and send 492 words

Figure 8-2



Measurement results

Table 8-2

OB35	Cyclical	Monitor	check_conn	READ words	WRITE words	Register amount	Comm. Speed
100 ms	350 ms	1200 ms	5000 ms	124	123	257	599 ms
100 ms	850 ms	1200 ms	5000 ms	248	246	494	1098 ms
100 ms	1500 ms	1200 ms	5000 ms	372	369	741	1700 ms
100 ms	1900 ms	1200 ms	5000 ms	496	492	988	2099 ms

8.3 Scenario 2B

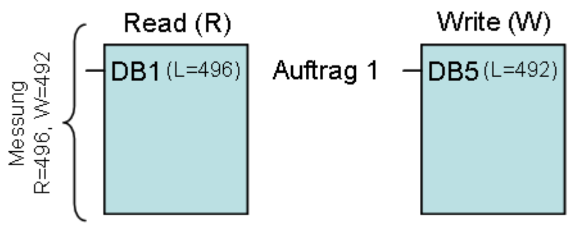
Scenario 2B is analogous to the first measurement from scenario 2A, but the data size received and sent via an job (1 DB) is increased to 496 / 492 words.

For communication, all configured connections are used ("use_all_conn = TRUE").

Schematic measurement setup

2 * MB_REDCL + 2 * Job_List
2 * data blocks (1 * for send; 1 * for receive)
Receive 496 words and send 492 words

Figure 8-3



Measurement results

Table 8-3

OB35	Cyclical	Monitor	Check_conn	READ words	WRITE words	Register amount	Comm. Speed
100 ms	1900 ms	1200 ms	5000 ms	496	492	988	2009

8.4 Scenario: HW Config. Change Loading

This scenario considers the redundant communication of an H System with a communication partner connected via Industrial Ethernet. There are two communication paths from the CPU to the linked external device. The structure corresponds to the configuration structure in Section 4.3.2.

In order to reduce the communication load, the parameter "use_all_conn" has been deactivated. Accordingly, only one configured communication path is used.

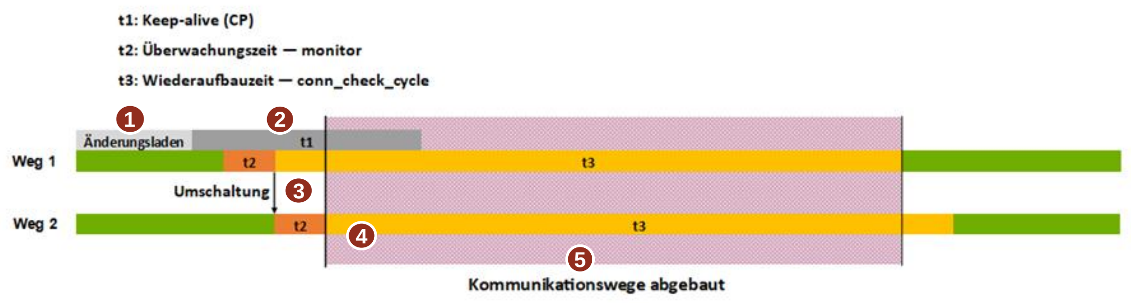
Note It is recommended to use all configured communication paths ("use_all_conn" = TRUE). This parameterization should only be used if the communication load cannot be reduced in any other way.

Modbus configuration

Table 8-4

Name	Configuration
Restoration time: "check_conn_cycle"	30 s
Monitoring time: "Monitor"	2.5 s
Connections used: "use_all_conn"	FALSE

Figure 8-4


t1: Keep-alive (CP)
t2: Überwachungszeit — monitor
t3: Wiederaufbauzeit — conn_check_cycle
1
2
Änderungsladen
t1
Weg 1
t2
t3
Umschaltung
3
Weg 2
t2
4
t3
5
Kommunikationswege abgebaut

Timing for HW Config. change loading

1. HW Config. change loading is executed
2. Response time after change loading exceeds the monitoring time
3. For communication, all configured connections are used
4. After the monitoring time, the restoration time also starts here
5. Both communication paths can be broken for up to 27.5 s

With this configuration, it cannot be guaranteed that a communication path is always established in the redundant system. This can lead to the triggering of communication monitoring and a corresponding reaction in the PCS. This could be, for example, the startup of a system safe state.

Measure to optimize communication

Restoration time and monitoring time can be reduced to optimize communication. In addition, a downstream TimerP block can be configured as Off delay.

Table 8-5

Name	Configuration
Restoration time: "check_conn_cycle"	10 s
Monitoring time: "Monitor"	2.5 s
Off delay: "TimerP"	20 s
Connections used: "use_all_conn"	FALSE

Figure 8-5 Configuration example TimerP block

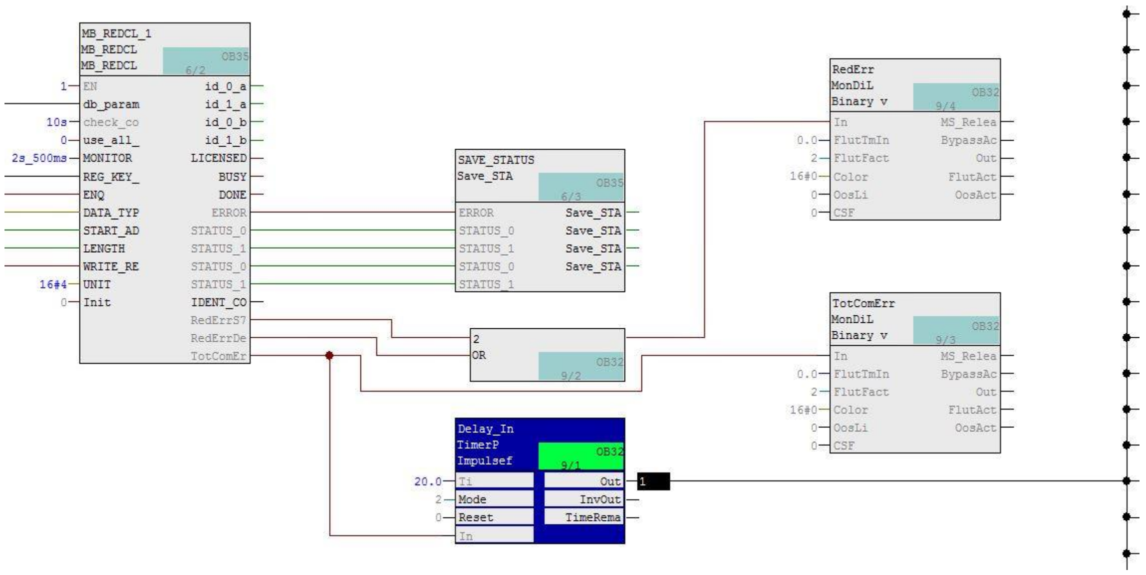
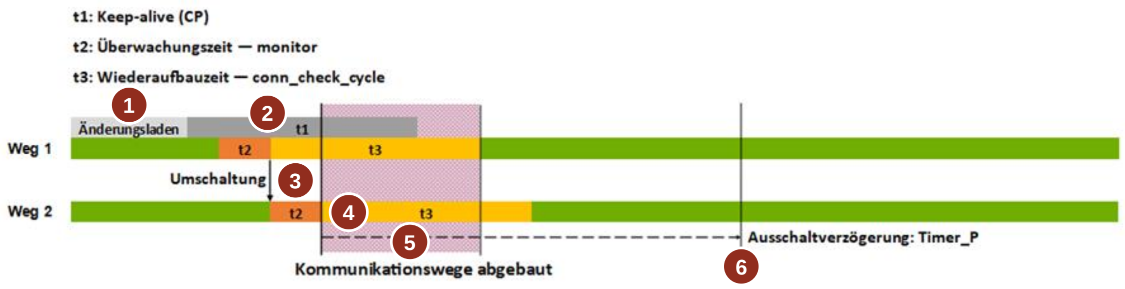


Figure 8-6


t1: Keep-alive (CP)
t2: Überwachungszeit — monitor
t3: Wiederaufbauzeit — conn_check_cycle
1
2
Änderungsladen
t1
Weg 1
t2
t3
Umschaltung
3
Weg 2
t2
4
t3
5
Ausschaltverzögerung: Timer_P
Kommunikationswege abgebaut
6

Timing for change loading:

1. Change loading is performed
2. Response time after change loading exceeds the monitoring time
3. For communication, all configured connections are used
4. After the monitoring time, the restoration time also starts here
5. Both communication paths can be broken for up to 7.5 s
6. TimerP is not activated
7. No reaction from change loading

With this change of restoration time and monitoring time, as well as the addition of the Off delay by the TimerP block, the reaction in the PCS can be prevented. Switching and loss of both communication paths is not prevented in this configuration.

9 Appendix

9.1 Service and support

Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:

support.industry.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts.

Please send queries to Technical Support via Web form:

support.industry.siemens.com/cs/my/src

SITRAIN – Training for Industry

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page:

siemens.com/sitrain

Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog web page:

support.industry.siemens.com/cs/sc

Industry Online Support app

You will receive optimum support wherever you are with the "Siemens Industry Online Support" app. The app is available for iOS and Android:

Fehler! Linkreferenz ungültig.

9.2 Industry Mall

The Siemens Industry Mall is the platform on which the entire siemens Industry product portfolio is accessible. From the selection of products to the order and the delivery tracking, the Industry Mall enables the complete purchasing processing – directly and independently of time and location:
mall.industry.siemens.com

9.3 Links and literature

Table 9-1

No.	Subject
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to the entry page of the application example https://support.industry.siemens.com/cs/ww/en/view/75867147
\3\	

9.4 Change documentation

Table 9-2

Version	Date	Change
V1.0	09/2013	First edition
V1.1	09/2015	Update and supplements
V1.2	02/2016	Update and supplements
V1.3	01/2017	Update
V1.4	11/2018	Update to PCS 7 V9.0 SP1 and revision due to a new licensing model and interface changes of the blocks.
V1.5	01/2021	Update to PCS 7 V9.0 SP2 and supplements