

SIMATIC Modbus/TCP

Redundant communication via CP443-1 in H systems

Programming manual · 03/2018

SIMATIC

Answers for industry.

SIEMENS

SIEMENS

SIMATIC

SIMATIC Modbus/TCP Redundant communication via the CP 443-1 in H-systems

Programming Manual

| | |
|--|----|
| Preface | |
| Product description | 1 |
| Getting Started | 2 |
| Commissioning | 3 |
| Parameter assignment for Modbus/TCP communication | 4 |
| Licensing | 5 |
| Function block MB_REDCL - Modbus client | 6 |
| Function block MB_REDSV - Modbus server | 7 |
| Additional blocks | 8 |
| Use in an S7-300 station | 9 |
| Diagnostics | 10 |
| Examples of applications | 11 |
| References | A |

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

CAUTION

indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

WARNING

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Preface

Purpose of this manual

With the information in this manual, you can establish and commission a link between a CP 443-1 in an S7 H system and a device that supports the MODBUS/TCP protocol.

Contents of the manual

This manual describes the function of the MODBUS function block and its integration in the hardware and software of the CP 443-1 communications processor.

The manual covers the following topics:

- Product description
- Getting Started
- Commissioning / installing / parameter assignment
- Licensing
- Function blocks
- Additional blocks
- Diagnostics
- Examples of applications

Validity of the manual

This manual is valid for the following software:

| Product | Identification number | as of version |
|-------------------------|--|---------------|
| MODBUS/TCP CP Redundant | 6AV6676-6MB30-4AX0 6AV6676-6MB30-4AD0 | 4.0 |
| FB 909 "MB_REDCL" | | 4.0 |
| FB 908 "MB_CPCLI" | | 4.0 |
| FB 907 "MB_REDSV" | | 4.0 |
| FB 906 "MB_CPSRV" | | 4.0 |

Note

This manual contains the description of the FB valid at the time the manual was published.

Aids for accessing the manual

To provide you with fast access to special information, the manual contains the following aids for access:

- On the following pages you will find a complete table of contents

Additional sources of information

You will find all other information relating to the CP 443 (installation, commissioning etc.) in the manual:

SIEMENS
SIMATIC
Fault-tolerant Systems
S7-400H
system manual
A5E00267693-03

SIEMENS
SIMATIC NET
S7 CPs for Industrial Ethernet
manual
C79000-G8976-C155

SIEMENS
SIMATIC NET
S7 CPs for Industrial Ethernet
manual, Part B4
CP 443-1
C79000-G8976-C152

SIEMENS
SIMATIC NET
NCM S7 for Industrial Ethernet
manual
C79000-G8976-C129

You will find further information relating to STEP 7 in the following manuals:

SIEMENS
SIMATIC Software
Basic Software for S7 and M7
STEP 7 user manual
C79000-G7076-C502-..

SIEMENS
SIMATIC Software
System Software for S7-300/400
System and Standard Functions
reference manual
C79000-G7076-C503-02

Queries

Your Siemens contact from whom you received this function block will be pleased to provide answers to any open issue relating to the use of the FBs described in this manual:

Conventions

The term CP or CP 443 is used in this documentation. The descriptions are valid for the CP443-1 and CP343-1 communication processors.

Area of application

The function block described in this manual establishes a connection between the CP443-1/CP343-1 and MODBUS devices of other manufacturers.

Table of contents

| | | |
|----------|---|-----------|
| | Preface | 3 |
| 1 | Product description | 9 |
| 1.1 | Possible applications | 9 |
| 1.2 | Hardware and software requirements..... | 10 |
| 2 | Getting Started | 11 |
| 3 | Commissioning | 13 |
| 3.1 | Installing the library on the STEP 7 PG/PC | 13 |
| 3.2 | Assigning CP parameters | 14 |
| 3.3 | Configuring the communications connection | 17 |
| 3.3.1 | Configuring the connection in "CP is Client" | 17 |
| 3.3.2 | Configuring an unspecified connection with "CP is server" | 20 |
| 3.4 | Inserting the function blocks in the program | 23 |
| 3.5 | Startup behavior of the CP 443 | 24 |
| 4 | Parameter assignment for Modbus/TCP communication | 25 |
| 4.1 | Parameter assignment for Modbus/TCP communication with the Wizard | 26 |
| 4.2 | Manual parameter assignment for Modbus/TCP communication | 27 |
| 4.2.1 | Structure of the connection parameters..... | 27 |
| 4.2.2 | Parameter assignment for Modbus/TCP communication | 30 |
| 5 | Licensing | 31 |
| 5.1 | Licensing via the wizard and the "Industry Support" app | 31 |
| 5.2 | Licensing by reading out the IDENT_CODES | 31 |
| 5.3 | Missing or incorrect licensing..... | 35 |
| 6 | Function block MB_REDCL - Modbus client | 37 |
| 6.1 | Configuring the redundant communication | 37 |
| 6.2 | How FB MB_REDCL works | 40 |
| 6.3 | Connection monitoring with AG_CNTRL | 47 |
| 6.4 | Parameters of the MB_REDCL function block..... | 48 |
| 6.5 | Example of the address mapping | 54 |
| 6.6 | Data and standard functions used by the FB | 57 |
| 6.7 | Renaming / rewiring functions and function blocks | 58 |

| | | |
|-----------|---|-----------|
| 7 | Function block MB_REDSV – Modbus server | 59 |
| 7.1 | Configuring the redundant communication | 59 |
| 7.2 | Using connections on port 502 | 61 |
| 7.3 | How FB MB_REDSV works | 63 |
| 7.4 | Connection monitoring with AG_CNTRL | 66 |
| 7.5 | Parameters of the MB_REDSV function block..... | 66 |
| 7.6 | Example of the address mapping | 71 |
| 7.7 | Data and standard functions used by the FB..... | 74 |
| 7.8 | Renaming / rewiring functions and function blocks..... | 75 |
| 8 | Additional blocks | 77 |
| 8.1 | Support in CFC | 77 |
| 8.2 | Job list for cyclic data exchange | 78 |
| 9 | Use in an S7-300 station | 79 |
| 10 | Diagnostics | 81 |
| 10.1 | Diagnostics using the display elements of the CP..... | 81 |
| 10.2 | Diagnostics messages of the FBs MB_REDCL and MB_REDSV | 82 |
| 10.3 | Diagnostics messages of the linked in blocks | 86 |
| 10.4 | Diagnostics messages of SFC24..... | 86 |
| 10.5 | Diagnostics messages via alarm bits..... | 86 |
| 10.5.1 | Client block..... | 87 |
| 10.5.2 | Server block | 88 |
| 11 | Examples of applications | 89 |
| 11.1 | Sample project in STL - Modbus client | 91 |
| 11.2 | Sample project in STL - Modbus server | 92 |
| 11.3 | Sample project in CFC - Modbus client | 93 |
| 11.4 | Sample project in CFC - Modbus server..... | 94 |
| A | References | 97 |

Product description

1.1 Possible applications

Integration in the system environment

The following driver represents a software product for the communications processor CP 443-1 in a SIMATIC S7 H system.

The CP 443-1 can be used in the S7-400 programmable controllers and can establish communications connections to partner systems.

Function of the FBs

With these function blocks, a communications connection is possible between the communications module CP 443-1 and a device that supports the MODBUS/TCP protocol. The function codes 1, 2, 3, 4, 5, 6, 15 and 16 are supported.

Data transmission takes place according to the client-server principle.

The SIMATIC S7 can be operated as client as well as server during the transfer.

Redundant communication is supported. The blocks can be used both in an S7-400 H system as well as in an S7-400 single CPU with two CPs.

The blocks run in hot-standby mode. Hot standby describes the parallel redundant processing of the signals in redundant components. This allows bumpless switchover of the entire system to the standby components

TCP/IP with CP 443-1

TCP/IP with CP 443-1 runs via static connections. The TCP connection is not terminated during error-free operation.

With the TCP stack used on the CP, the STEP 7 connection configuration only allows one use of a particular port number.

Certain CP types can maintain and operate connections to multiple clients simultaneously via the local port 502.

The technical details on this topic are explained in section 7.2 "Using connections on port 502 (Page 61)".

1.2 Hardware and software requirements

Usable modules for MB_REDCL or MB_REDSV

With the AG_CNTRL block from the SIMATIC_NET library, it is possible to terminate and re-establish an existing connection. To be able to use the resources of the CPU/CP more effectively, this block was also included in the Modbus blocks. The older CPs or older firmware versions, however, do not support this AG_CNTRL.

You will find information about which CP supports AG_CNTRL with which firmware version here: Ethernet CP and AG_CNTRL

(<https://support.industry.siemens.com/cs/ww/en/view/33414377>).

You can find technical specifications for the SIMATIC Modbus/TCP blocks and learn which CPUs and CPs have been released here: Technical specifications of the Modbus/TCP blocks (<http://support.automation.siemens.com/WWW/view/en/104946406>).

The MB_REDCL and MB_REDSV blocks are Block-Privacy-protected and can be used in an S7-400 CPU as of firmware V6.0 and in an S7-300 CPU as of firmware V3.2.

Software versions

The use of FB MB_REDCL or MB_REDSV is possible as of STEP 7 version 5.5. The AG_LSEND/AG_LRECV V3.1 blocks must be used for this.

Memory requirements

FB MB_REDCL requires 13 KB of work memory and 16 KB of load memory.

FB MB_CPCLI requires 9 KB of work memory and load memory.

FB MB_REDSV requires 12 KB of work memory and 14 KB of load memory.

FB MB_CPSRV requires 8 KB of work memory and load memory.

You will find the precise lengths of the blocks in their properties in the SIMATIC Manager.

Getting Started

Procedure

1. Installation of "SIMATIC Modbus/TCP CP Redundant" and inclusion of the MODBUS blocks in the user project
=> section 3.1 (Page 13)
2. Setting the parameter DB MODBUS_HPAMM_CP according to the requirements (client/server, Modbus register, DB areas, etc.)
=> section 4 (Page 25)
3. For Modbus client: Call the MODBUS block MB_REDCL in cyclic OB
=> section 6.2 (Page 40) and 6.3 (Page 47)
or:
For Modbus server: Call the MODBUS block MB_REDSV in cyclic OB
=> section 7.3 (Page 63) and 7.4 (Page 66)
4. Downloading the user program to the CPU and licensing of the Modbus block for this CPU
=> section 5 (Page 31)

See also

- Inserting the function blocks in the program (Page 23)
- Configuring the redundant communication (Page 37)
- Assigning CP parameters (Page 14)
- Configuring the communications connection (Page 17)
- Licensing by reading out the IDENT_CODES (Page 31)

Commissioning

General information

The CP 443-1 can be configured via MPI or LAN/Industrial Ethernet. STEP 7 with NCM S7 for Industrial Ethernet (simply called "NCM IE" below) is required.

The information on STEP 7 and configuration of the communications connection used below relates to STEP 7 version 5.5.

In later versions, sequences, name and directory information may have changed.

Requirements

STEP 7 basic knowledge, STL knowledge, PLC basic knowledge

3.1 Installing the library on the STEP 7 PG/PC

Scope of delivery

The supplied CD or ISO image contains a setup with which the library "Modbus_TCP_CP_Red300_400", the example projects and the manuals in German and English can be installed in the relevant STEP 7 directories.

The CD also contains the manuals in PDF format.

Requirements

To be able to perform the installation, the STEP 7 configuration software must first be installed.

Installation

Insert the MODBUS CD in the CD-ROM drive of your PG/PC. If the setup program does not start automatically, install as follows:

1. Select the CD-ROM drive in the Windows Explorer, open the Setup directory and start the setup program.
2. Follow the instructions displayed by the installation program step by step.

With the download version, as an alternative you can unpack the ISO image with a corresponding program and start the setup program in the "Setup" directory.

You will now find

- The library in \Program Files\Siemens\Step 7\S7LIBS,
- 2 Example projects in \Program Files\Siemens\Step 7\EXAMPLES,
- The manual in \Program Files\Siemens\Step 7\S7MANUAL\S7Comm,
- The form SOFTWARE REGISTRATION FORM in \Program Files\Siemens\Step 7\S7LIBS\Modbus_TCP_CP_Red300_400.

The manual can also be opened using the shortcut in \Program Files\Siemens\Documentation.

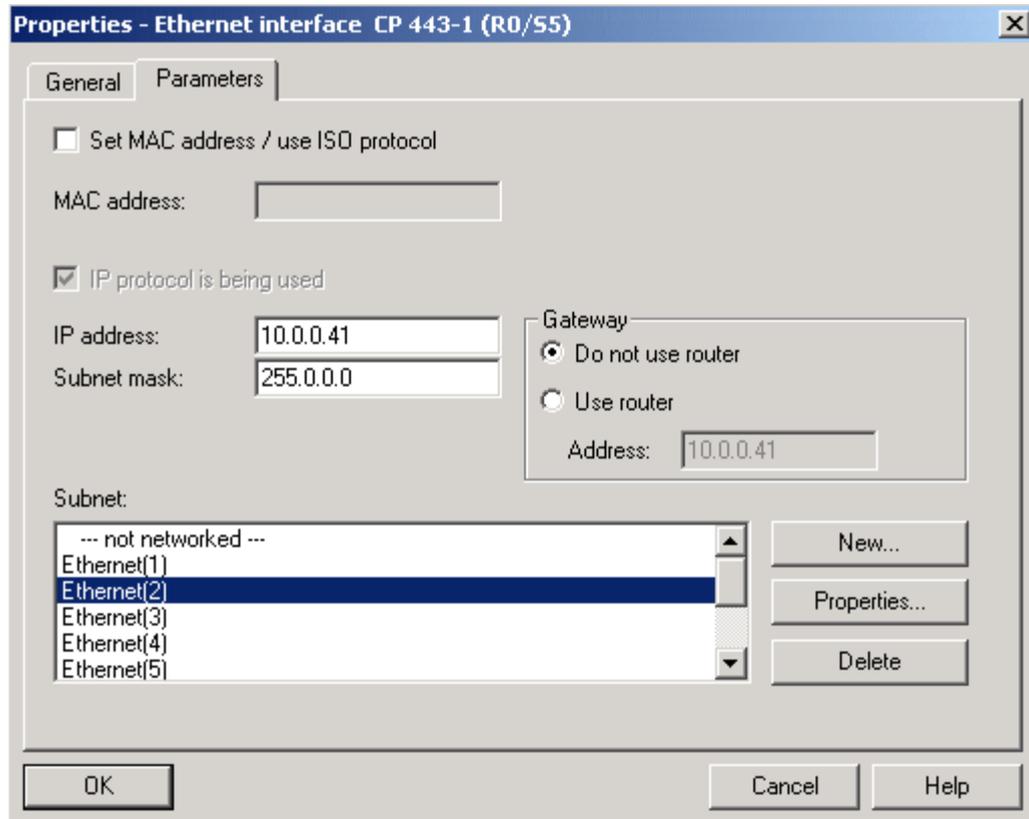
3.2 Assigning CP parameters

The CP is connected to Industrial Ethernet in the "Subnet" list. If you have interconnected your stations without routers, they must be located in the same subnet.

Procedure

1. Select the list entry with the name of your network.

For a new network, this is normally "Ethernet(1)".



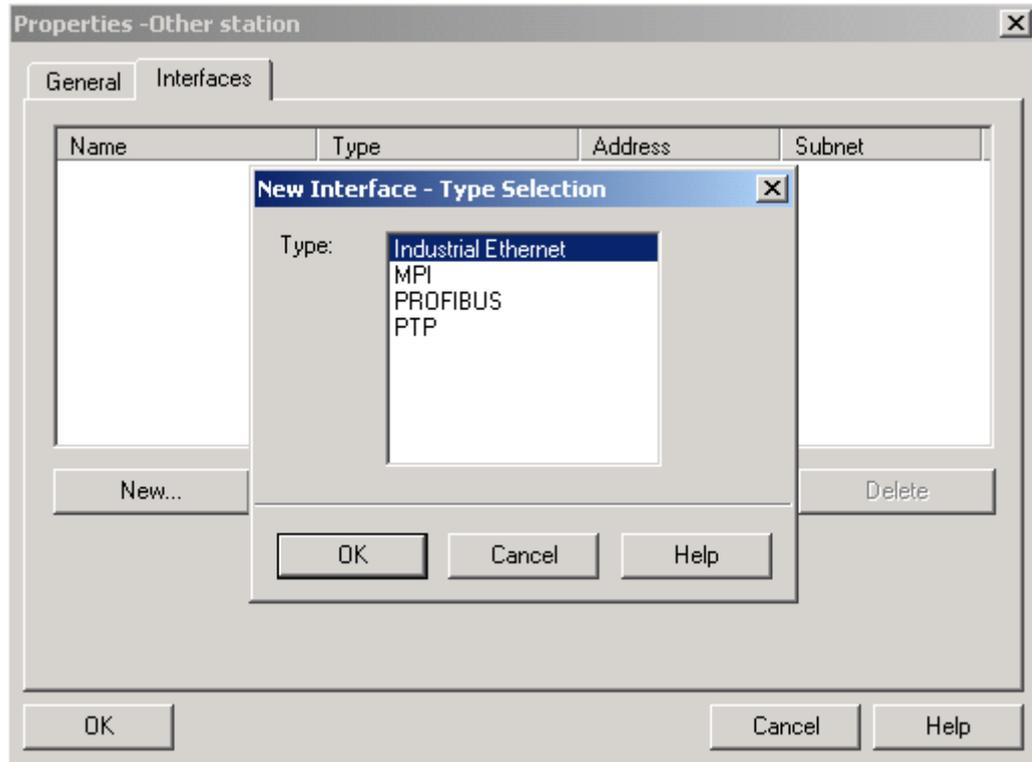
2. Confirm with "OK".
3. The parameter assignment is compiled and saved.

Assigning link partner parameters

In the "CP is Client" mode, an "Other station" is necessary for the connection configuration. After you have inserted the station of the link partner in your STEP 7 project, you need to specify the object properties of this station.

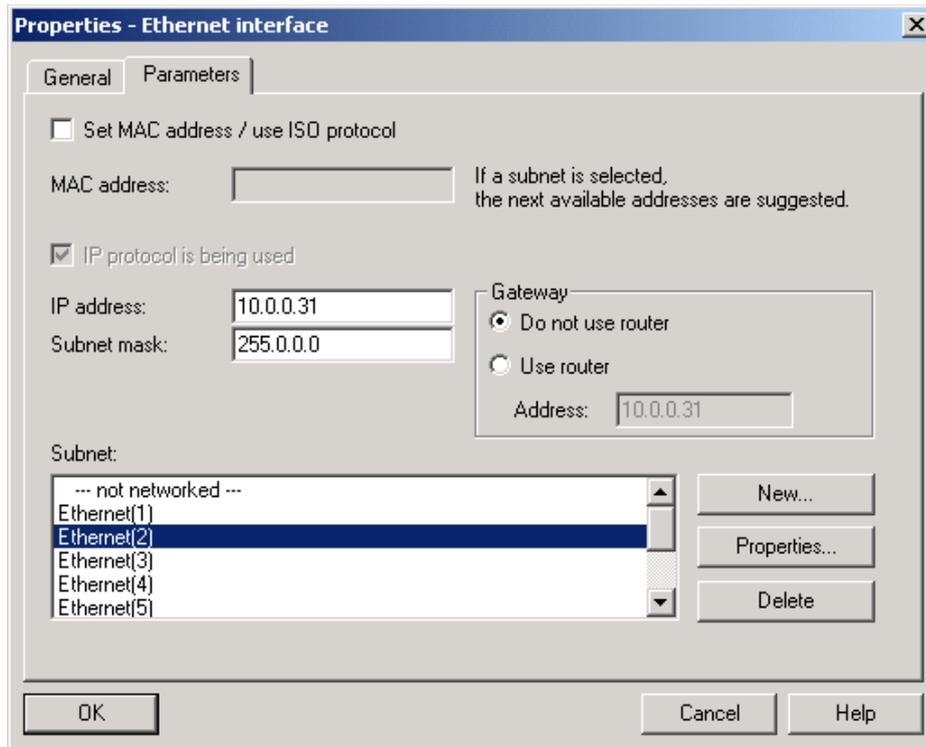
Procedure

1. Select "Properties > Other station > Interfaces".

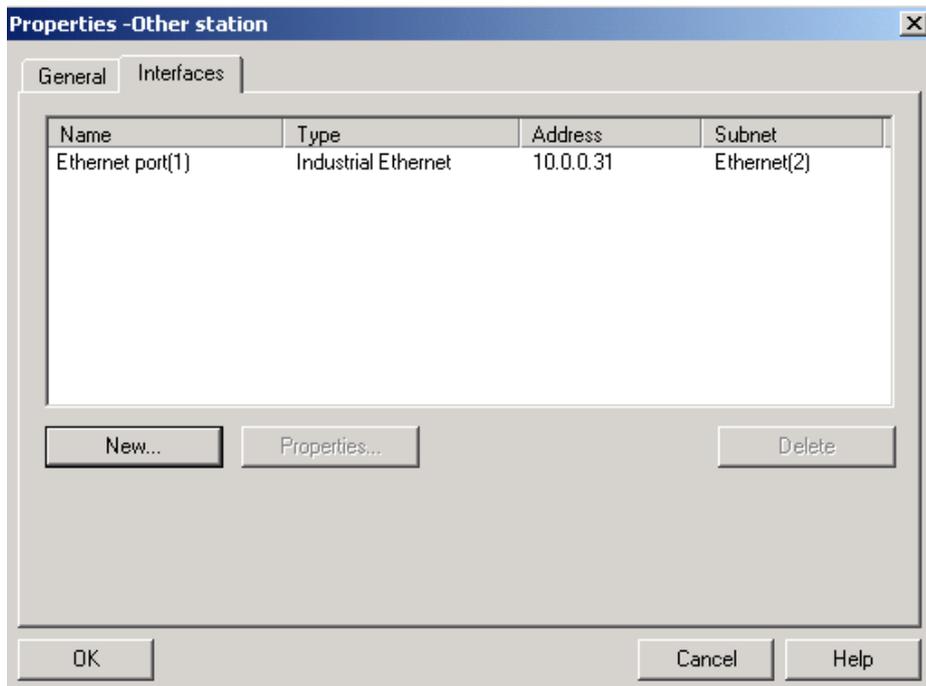


2. Press the "New" button.
3. In the type selection box that then appears, select "Ind. Ethernet".

4. Press OK. The following dialog opens.



5. Enter an IP address located in the same subnet as the link partner station.
6. Select the subnet that represents the connection between CP interface and the link partner interface. By pressing the "OK" button, you return to the "Interfaces" tab.



7. "Properties > Other station > General".
No settings are required in the "General" tab.

3.3 Configuring the communications connection

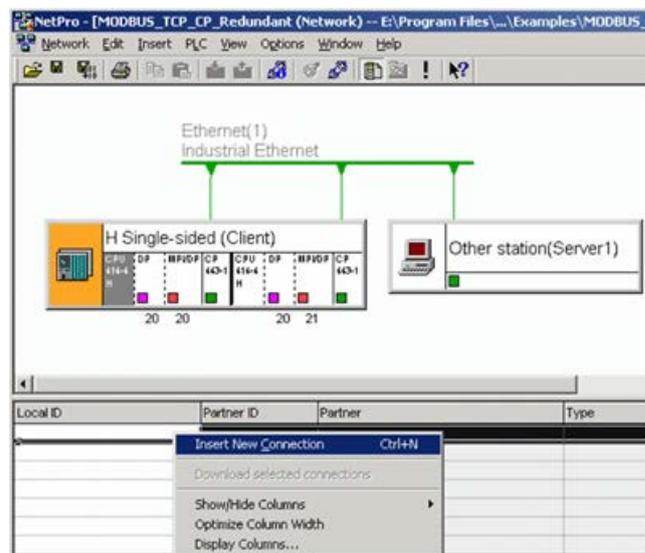
For the connection between an S7 CPU and a communications partner/bus connected via Industrial Ethernet, the CP represents the link. To connect the relevant interface to the link partner/bus, the connection must be configured.

3.3.1 Configuring the connection in "CP is Client"

Procedure

1. In the STEP 7 project, select the CPU in your open S7-400H station.
2. By double-clicking on "Connections" you open the connection configuration.
The "NetPro" program opens with which you can configure your connections.
3. Select "Insert > New Connection...".

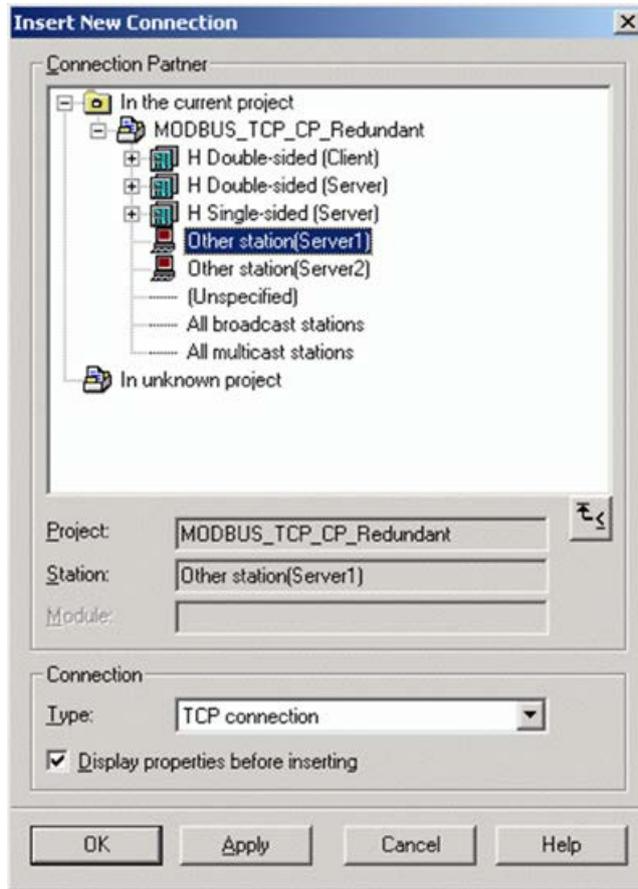
The following dialog is displayed.



3.3 Configuring the communications connection

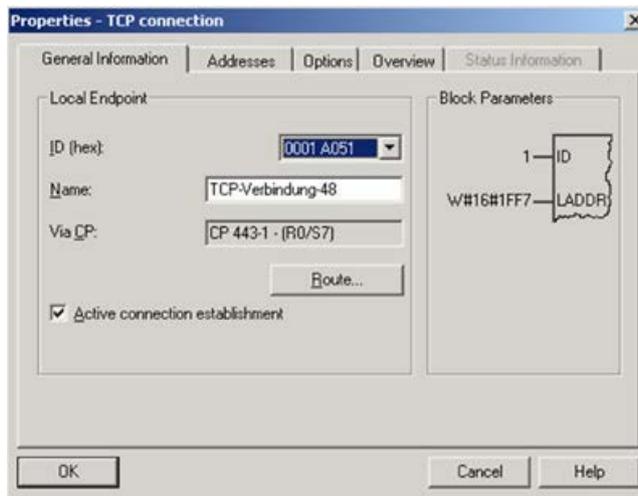
- 4. For the new connection, select the connection partner "Other station" and "TCP connection" as the connection.

The following dialog is displayed.



- 5. Select the "Display properties before inserting" check box.
- 6. Confirm by clicking "OK".

The following dialog is displayed.



In the dialog, you can set the properties for the connection.

Note

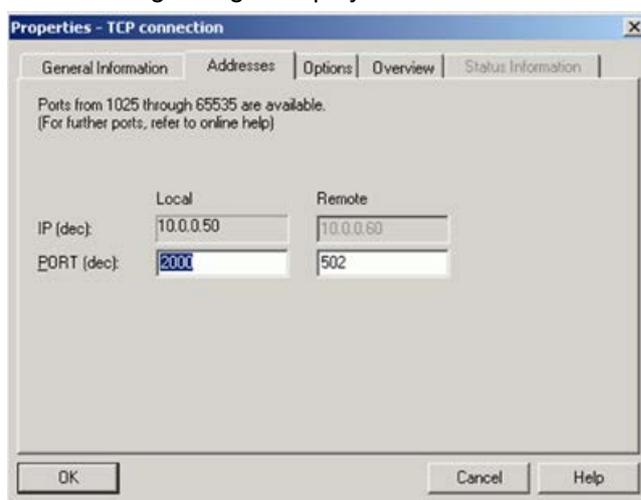
Note that the connection ID (local ID) must be used in the parameter DB for the Modbus block.

7. Set the ID based on the project requirements.
8. Enter a name for the connection.

The "Via CP" box displays the communications processor used to establish the connection. The assigned connection path is displayed with the "Route..." button. If you want to use a different plugged-in CP, this can be selected.

9. Select the "Active connection establishment" check box.
10. Change to the "Addresses" tab.

The following dialog is displayed.



In this dialog, the port numbers for both communications partners are specified.

11. Specify the port numbers in the "Addresses" tab.

Note

Selecting the port number:

- In Modbus/TCP communication, the Modbus/TCP servers are normally addressed via port 502.
 - With Modbus/TCP clients, port numbers higher than 2000 are used.
-

12. Confirm by clicking "OK".

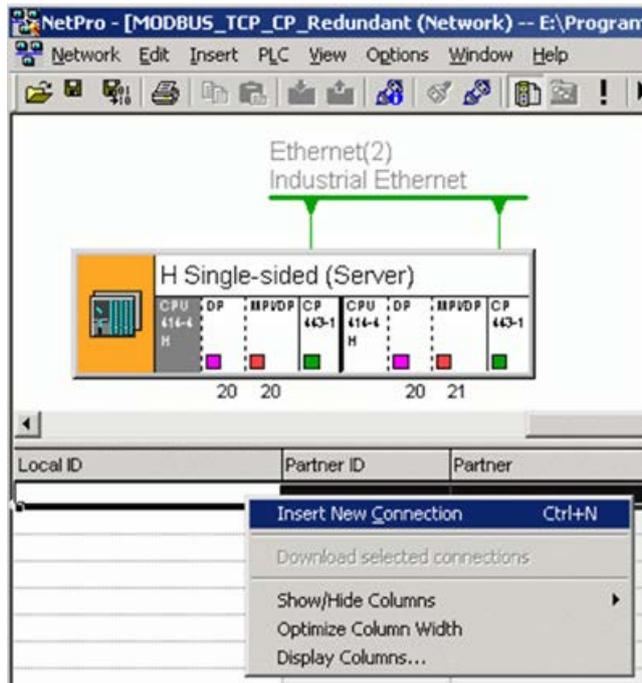
The entries are applied.

13. Save the connection configuration and close the "NetPro" program.

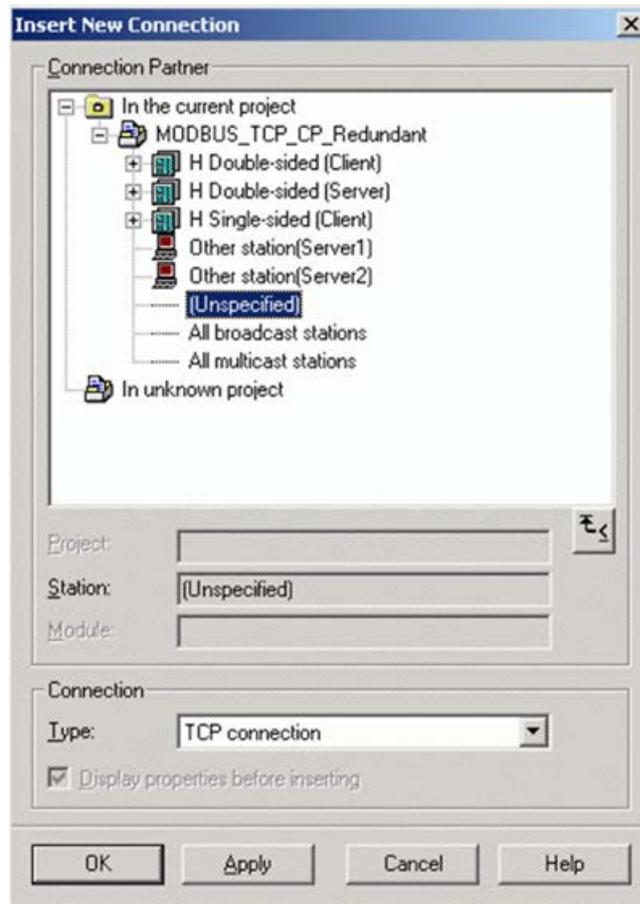
3.3.2 Configuring an unspecified connection with "CP is server"

Procedure

1. If the CP operates as a server on a link, communication is activated via an unspecified connection.
The client needs to handle active connection establishment.
2. After selecting Insert > New connection..., you change to the "Insert New Connection" dialog.

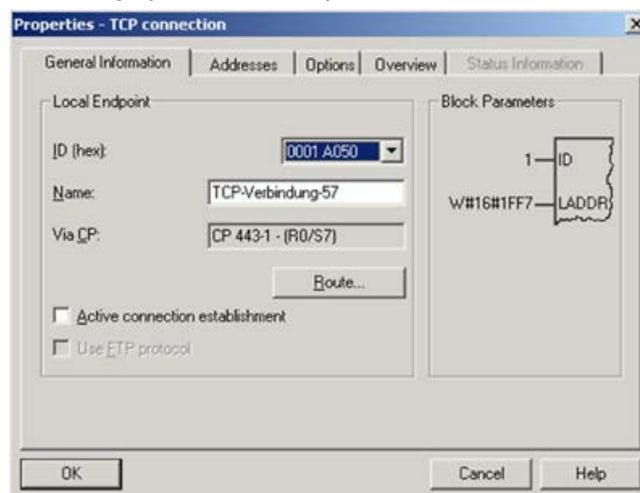


- For the new connection, instead of the connection partner, select "Unspecified" and "TCP connection" as the connection.



- Select the "Display properties before inserting" check box.
- Confirm by clicking "OK".

This brings you to the "Properties - TCP connection" dialog.



Note

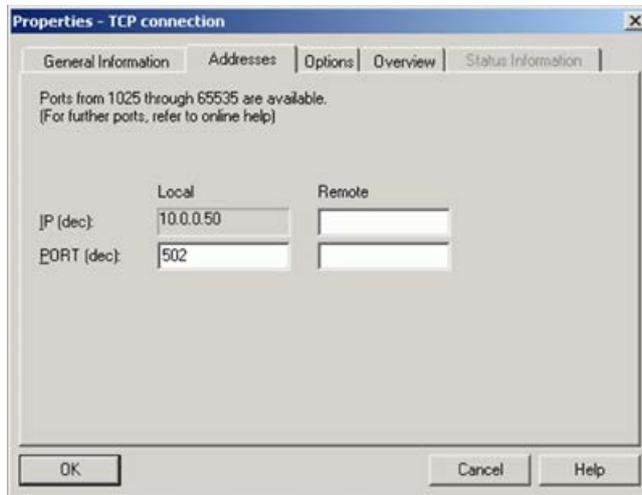
Note that the connection ID (local ID) must be used in the parameter data block for the Modbus block.

- 6. Set the ID based on the project requirements.
- 7. Enter a name for the connection.

The "Via CP" box displays the communications processor used to establish the connection. The assigned connection path is displayed with the "Route..." button. If you want to use a different plugged-in CP, this can be selected.

- 8. Disable the "Active connection establishment" check box.
- 9. Change to the "Addresses" tab.

In the "Addresses" dialog, no settings are made for the "Partner". "IP" and "PORT" do not have an entry.



- 10. Confirm by clicking "OK".
- The entries are applied.

3.4 Inserting the function blocks in the program

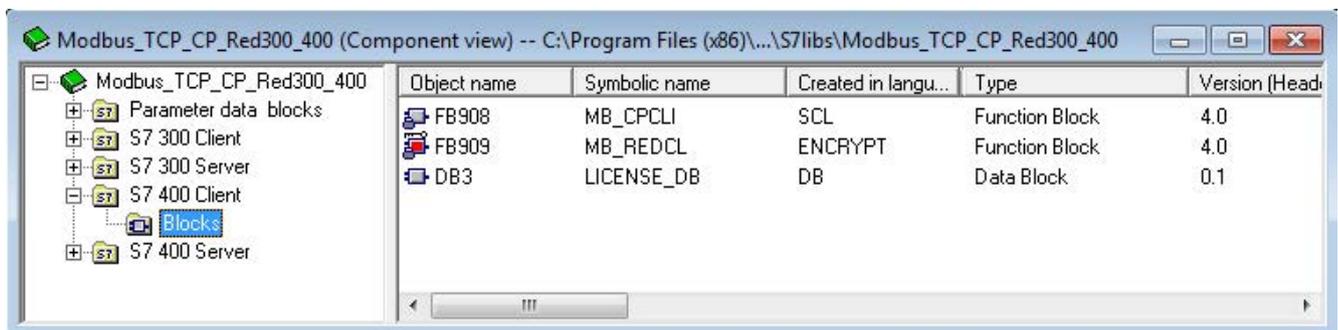
Content of the Modbus library:

The Modbus library contains the folders "S7 300 Client", "S7 300 Server", "S7 400 Client", "S7 400 Server" and "Parameter data blocks" with the FBs for redundant communication.

S7 400 Client

Among other things, the "S7 400 Client" folder contains the blocks

- FB909 (MB_REDCL) and
- FB908 (MB_CPCLI).

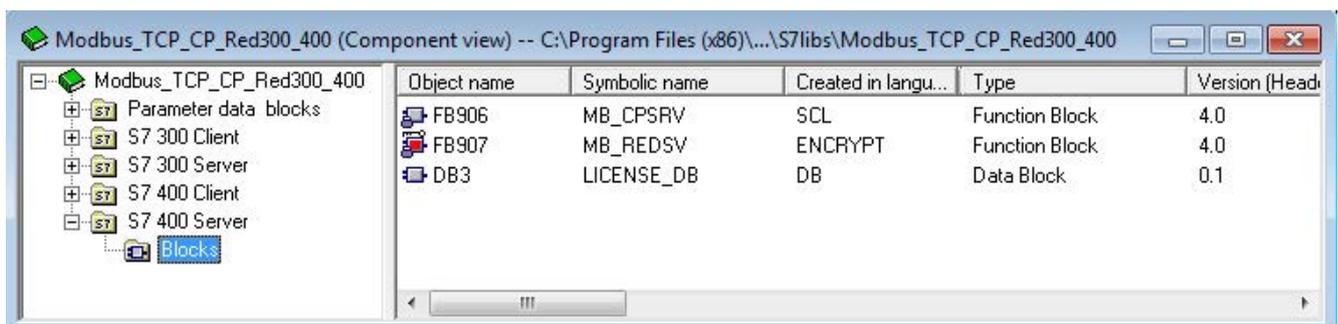


For redundant communication, both blocks are always required. The MB_REDCL calls the MB_CPCLI block several times internally.

S7 400 Server

Among other things, the "S7 400 Server" folder contains the blocks

- FB907 (MB_REDSV) and
- FB906 (MB_CPSRV).



For redundant communication both blocks are always required. The MB_REDSV block calls the MB_CPSRV block several times internally.

Parameter data blocks

The "Parameter data blocks" folder contains parameter data blocks as examples.

Inserting the blocks

Open the "Modbus_TCP_CP_Red300_400" library and copy the required blocks. Insert the blocks in your program.

The Modbus blocks use the functions AG_LSEND and AG_LRECV. Copy these blocks from the "SIMATIC_NET_CP" -> "CP 400" library and insert them in your program.

Also copy AG_CNTRL from this library folder to your project.

Note

Note that the following versions of the FCs are required for error-free operation of FBs the MB_REDCL / MB_REDSV:

- AG_LSEND V3.1 or higher
 - AG_LRECV V3.1 or higher
 - AG_CNTRL V1.0 or higher
-

3.5 Startup behavior of the CP 443

Introduction

The startup of the CP is divided into the following phases:

- Initialization (CP network on)
- Parameter assignment

Initialization

As soon as power is applied to the CP, the firmware on the CPU is prepared for operation after running through a hardware test program.

Parameter assignment

During the parameter assignment, the CP receives the module parameters assigned to the current slot.

The CP is now ready for operation.

Parameter assignment for Modbus/TCP communication

4

For communication via a CP 443, the connection must be configured in NetPro.

Several connections to different communications partners can be configured and established at the same time. The number of simultaneously established connections depends on the CPU and the CP being used.

Parameter data block MODBUS_HPPARAM_CP

The data required to assign the connections and process the Modbus/TCP requests are defined in a data block (the parameter data block MODBUS_HPPARAM_CP). The connection-specific data is first saved. The connection-specific data is followed by the Modbus/TCP parameters.

A parameter block in which the connection parameters of the NetPro configuration and the Modbus/TCP parameters are defined is required for each connection. For each further connection, a new block needs to be created and the matching parameter block with the connection and Modbus/TCP parameters has to be inserted.

A completed parameter data block is included in the "Modbus_TCP_CP_Red300_400" library as an example.

Structure of DB MODBUS_HPPARAM_CP

| Address | Name |
|---------|-----------------------|
| +0.0 | Connection parameters |
| +8.0 | |
| +10.0 | Modbus parameters |
| +74.0 | |

- Connection parameters

In the first block, the connection-specific parameters "id" and "laddr" are defined. Based on these parameters, the connection configured in NetPro can be assigned.

- Modbus/TCP parameters

The data required for the mode and address reference is saved in the Modbus/TCP parameters, for example the mode of the S7 as Modbus/TCP server or Modbus/TCP client, the Modbus/TCP register addresses and the numbers of the DBs in which the data is mapped. You must keep to the data structure of the Modbus/TCP parameters because they cannot be processed correctly otherwise.

Configuration options

You have two options for configuring the connection and Modbus/TCP parameters. On the one hand, the entries can be made with a Wizard making the parameter assignment extremely convenient. On the other hand, the parameters can be set by editing the parameter data block.

These two options are described in the sections "Parameter assignment for Modbus/TCP communication with the Wizard (Page 26)" and "Structure of the connection parameters (Page 27)".

4.1 Parameter assignment for Modbus/TCP communication with the Wizard

The "SIMATIC Modbus/TCP Wizard" allows convenient configuration of the connection ID, the CP address and the Modbus/TCP parameters in the parameter data block MODBUS_HP_PARAM_CP. The entire parameter block is created with connection parameters and Modbus/TCP parameters.

You will find the Wizard in: SIMATIC Modbus/TCP Wizard
(<http://support.automation.siemens.com/WW/view/en/60735352>)

4.2 Manual parameter assignment for Modbus/TCP communication

4.2.1 Structure of the connection parameters

Structure of the connection parameters

The figure below shows the connection parameters based on the example of DB6.

| 0.0 | | STRUCT | |
|-------|------------------|------------|----------|
| +0.0 | double_sided_red | BOOL | FALSE |
| +2.0 | id_0_a | WORD | W#16#0 |
| +4.0 | id_1_a | WORD | W#16#0 |
| +6.0 | laddr_cp0 | WORD | W#16#0 |
| +8.0 | laddr_cp1 | WORD | W#16#0 |
| +10.0 | server_client | BOOL | FALSE |
| +10.1 | single_write | BOOL | FALSE |
| +11.0 | reserved | BYTE | B#16#0 |
| +12.0 | data_type_1 | BYTE | B#16#3 |
| +14.0 | db_1 | WORD | W#16#B |
| +16.0 | start_1 | WORD | W#16#0 |
| +18.0 | end_1 | WORD | W#16#1F3 |
| +20.0 | data_type_2 | BYTE | B#16#3 |
| +22.0 | db_2 | WORD | W#16#C |
| +24.0 | start_2 | WORD | W#16#2D0 |
| +26.0 | end_2 | WORD | W#16#384 |
| +28.0 | data_type_3 | BYTE | B#16#4 |
| +30.0 | db_3 | WORD | W#16#D |
| +32.0 | start_3 | WORD | W#16#2D0 |
| +34.0 | end_3 | WORD | W#16#3E8 |
| +36.0 | data_type_4 | BYTE | B#16#0 |
| +38.0 | db_4 | WORD | W#16#0 |
| +40.0 | start_4 | WORD | W#16#0 |
| +42.0 | end_4 | WORD | W#16#0 |
| +44.0 | data_type_5 | BYTE | B#16#1 |
| +46.0 | db_5 | WORD | W#16#E |
| +48.0 | start_5 | WORD | W#16#280 |
| +50.0 | end_5 | WORD | W#16#4E2 |
| +52.0 | data_type_6 | BYTE | B#16#2 |
| +54.0 | db_6 | WORD | W#16#F |
| +56.0 | start_6 | WORD | W#16#6A4 |
| +58.0 | end_6 | WORD | W#16#8FC |
| +60.0 | data_type_7 | BYTE | B#16#1 |
| +62.0 | db_7 | WORD | W#16#10 |
| +64.0 | start_7 | WORD | W#16#6A4 |
| +66.0 | end_7 | WORD | W#16#8FC |
| +68.0 | data_type_8 | BYTE | B#16#0 |
| +70.0 | db_8 | WORD | W#16#0 |
| +72.0 | start_8 | WORD | W#16#0 |
| +74.0 | end_8 | WORD | W#16#0 |
| =76.0 | | END_STRUCT | |

4.2 Manual parameter assignment for Modbus/TCP communication

The first parameter specifies whether the redundancy is single-sided or double-sided. The parameters 2-5 are passed on to the AG_LSEND/AG_LRECV calls.

The remaining parameters specify the mode of the Modbus/TCP communication and the mapping of the Modbus/TCP addresses to the SIMATIC addresses. Up to 8 data areas can be set. At least the first of these data areas must be defined; the other 7 are optional.

The parameters are explained below.

id

A connection ID is assigned for each configured connection in STEP 7 (NetPro). The connection ID uniquely describes the connection from the CPU via the CP to the link partner. The number from the connection configuration is entered here. The range of values for this parameter is 1 to 64.

laddr

The "laddr" parameter is the base address of the CP from HW Config (I address). The configured value is entered here. The range of values for this parameter depends on the CPU. The "id" and "laddr" parameters can also be taken from the "Properties" dialog of the TCP connection.

server_client

This parameter distinguishes client and server mode. If the input is TRUE, the "CP is server" mode is used. If the setting is FALSE, the "CP is client" mode is used.

single_write

In the "CP is client" mode, if the "single_write" parameter = TRUE, the function codes 5 and 6 are used for write jobs with length 1. If "single_write" = FALSE, function codes 15 and 16 are used for all write jobs.

data_type_x

The "data_type_x" parameter 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 is not used.

| Identifier | Data type | Data width |
|------------|------------------|------------|
| 0 | Area not used | - |
| 1 | Coils | Bit |
| 2 | Inputs | Bit |
| 3 | Holding register | Word |
| 4 | Input register | Word |

db_x

The "db_x" parameter specifies the data block in which Modbus/TCP registers or bit values defined below are mapped. When using a global DB, the DB number 0 is not permitted because it is reserved for the system.

DB number 1 to 65535 (W#16#0001 to W#16#FFFF)

If a data collector block is used in CFC, the DB number 0 must be specified. The DataCollector blocks are explained in section 8, "Additional Blocks".

start_x, end_x

"start_x" specifies the first Modbus/TCP address that is mapped in data word 0 of the DB. The "end_x" parameter defines the address of the last Modbus/TCP address.

- When accessing registers, the data word number in the S7 DB in which the last Modbus/TCP address is entered is calculated as follows:

$$\text{DBW number} = (\text{end_x} - \text{start_x}) * 2$$

- With bit access, the data byte number in the S7 DB in which the last Modbus/TCP address is entered is calculated as follows:

$$\text{DBB number} = (\text{end_x} - \text{start_x}) / 8$$

The defined data areas must not overlap. The "end_x" parameter must not be lower than "start_x". In case of an error, the initialization of the FB is aborted with an error. If both values are identical, one Modbus/TCP address (1 register or 1 bit value) is assigned.

The data block must be created 2 bytes longer. These two reserve bytes are used for internal purposes.

In the sections below, there are examples of the mapping of the Modbus/TCP addresses to S7 memory areas.

- Example of the address mapping MB_REDCL (Page 54)
- Example of the address mapping MB_REDSV (Page 71)

4.2.2 Parameter assignment for Modbus/TCP communication

Procedure

1. Copy one of the blocks DB4-DB6 from the "Modbus_TCP_CP_Red300_400" library and insert it in your project.

If the number is already being used elsewhere, the DB can be renamed.

2. Adapt the parameters to meet your requirements.
3. Initialize the block with the Init bit.

Note

Parameter changes in the MODBUS_HPPARAM_CP block only take effect once the module is re-initialized.

A parameter block is required for each connection group.

Licensing

The MB_REDCL or MB_REDSV blocks must be licensed individually on each CPU. Licensing takes place in two steps: reading out the IDENT_CODE and entering the registration key REG_KEY. OB121 must exist on the CPU.

Note

With an S7 H station only the CPU in rack 0 is licensed. For this reason, replacing the CPU from rack 0 after the licensing is no longer possible.

There are 2 options for licensing available:

- Licensing via the wizard and the "Industry Support" app
- Reading the data from the instance DB and the printed COL

5.1 Licensing via the wizard and the "Industry Support" app

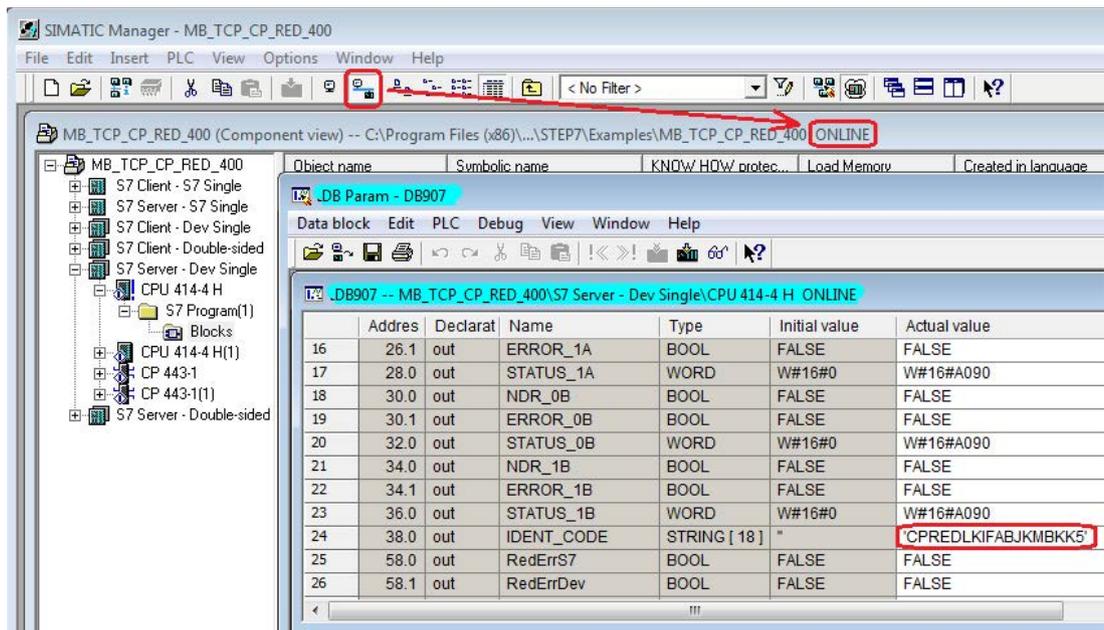
1. Set the parameters for the MB_REDCL or MB_REDSV block according to your requirements in a cyclic OB (OB1 or a cyclic interrupt OB).
2. Download the program to the CPU and set it to RUN.
3. Scan the data in according to the instructions in the FAQ Request a license via the Service&Support app (<https://support.industry.siemens.com/cs/de/en/view/109746433>) and register with this information.

5.2 Licensing by reading out the IDENT_CODES

Reading out the IDENT_CODE

To read out the IDENT_CODE, follow the steps below:

1. Set the parameters for the MB_REDCL or MB_REDSV block according to your requirements in a cyclic OB (OB1 or a cyclic interrupt OB).
Download the program to the CPU and set it to RUN.
2. Open the project in online mode in the SIMATIC Manager.
3. Open the instance DB of the Modbus block in this online project.



- An 18-character string is displayed at the IDENT_CODE output.

Copy this string from the DB and paste it in the form SOFTWARE REGISTRATION FORM. This form is saved in the library path ..\Program Files\Siemens\Step7\S7LIBS\Modbus_TCP_CP_Red300_400 during installation and is also available on the installation CD.
Enter the license number from the product packaging in the form.

Note

Use in CFC

Online, the CFC editor can only display a certain number of characters. The full IDENT_CODE is displayed in the tooltip of the output parameter.

| | Address | Declaration | Name | Type | Initial value | Actual value |
|----|---------|-------------|------------|---------------|---------------|----------------------|
| 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.

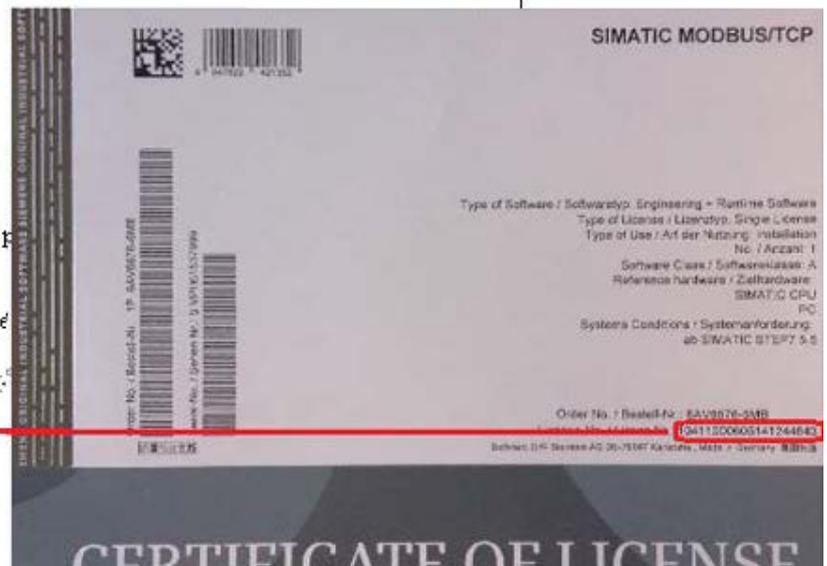
Bitte tragen Sie den IDENT-CODE hier ein.
Das Handbuch enthält Informationen, wie Sie den IDENT-CODE ermitteln.

>>> IDENT_CODE <<<

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

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

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



- Send the form to Customer Support (<http://www.siemens.com/automation/support-request>) using a service request. You will then receive the registration key for your CPU.

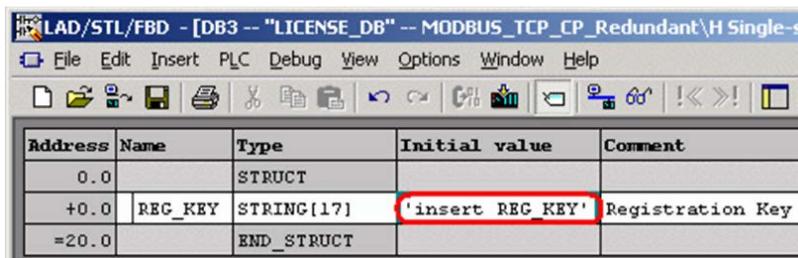
Entering the registration key REG_KEY.

The registration key REG_KEY must be specified with each MB_REDCL or MB_REDSV block call.

You should save the REG_KEY in a global data block via which all MB_REDCL or MB_REDSV blocks receive the necessary registration key (see the following example).

Proceed as follows to enter the registration key REG_KEY:

1. Copy the ready-made licensing block DB3 from the "Modbus_TCP_CP_Red300_400" library to your project. If the DB number is already being used in the project, the license DB can also be renamed.
2. Open the license DB and copy the 17-character registration key and paste it into the "Initial value" column.

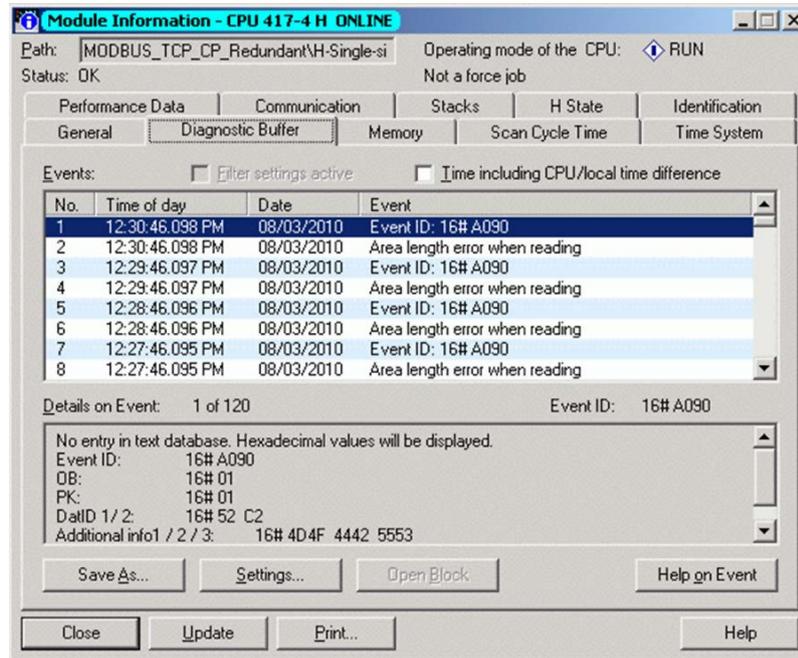


3. To avoid having to enter the registration key again after reloading the CPU, it needs to be entered permanently in the data block. Change to the data view of the DB with the menu item "View > Data View". With the menu command "Edit > Initialize Data Block", all the values from the "Initial value" column are transferred to the "Actual value" column.
4. In the cyclic OB, enter the number of the license DB at the parameter REG_KEY_DB of MB_REDCL or MB_REDSV.
5. Download the modified blocks to the CPU. The registration key can be entered during runtime; a change from STOP -> RUN is not necessary.

The block is now licensed for this CPU.

5.3 Missing or incorrect licensing

If you enter an incorrect registration key or no registration key at all, the INTF LED of the CPU flashes **once** a minute and an entry is made cyclically in the diagnostics buffer regarding the missing license. The error number for a missing license is W#16#A090.



WARNING

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

If a registration key is missing or incorrect, the MODBUS communication is processed. If a registration key is missing or incorrect, W#16#A090 "No valid license available" is displayed at the STATUS_x output. The output is ERROR=FALSE in this case.

Function block MB_REDCL - Modbus client

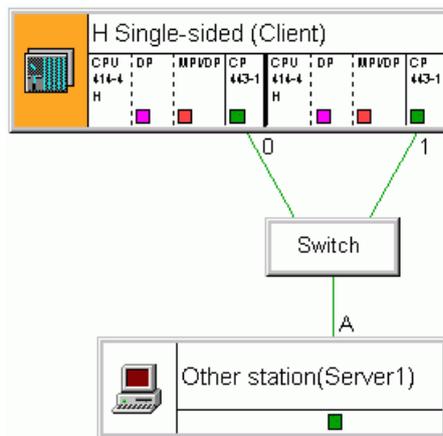
6.1 Configuring the redundant communication

General information

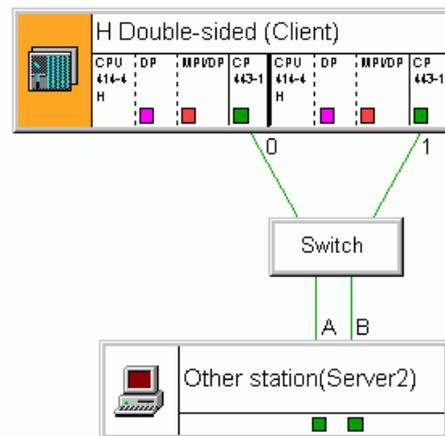
The CP is the client when the S7 initiates the reading or writing of the data from or to the remote partner.

The communications partner of the H system can be stand-alone or also redundantly structured (redundancy single-sided or double-sided).

Single-sided redundancy



Double-sided redundancy



Configuration in HW Config

When configuring the hardware in HW Config, CP0 and CP1 receive different input addresses and different IP addresses so that they can be addressed uniquely in the S7 program or by communications partners.

Configuration in NetPro

For every possible connection between the communications partners, 1 connection must be configured in NetPro. With single-sided redundancy, one connection is created for CPU0/CP0 and one connection for CPU1/CP1:

- Connection from CPU0/CP0 to partner => **connection 0A**
- Connection from CPU1/CP1 to partner => **connection 1A**

With double-sided redundancy, 2 connections are created for CPU0/CP0 and 2 connections for CPU1/CP1:

- Connection from CPU0/CP0 to partner/node A => **connection 0A**
- Connection from CPU1/CP1 to partner/node A => **connection 1A**
- Connection from CPU0/CP0 to partner/node B => **connection 0B**
- Connection from CPU1/CP1 to partner/node B => **connection 1B**

6.1 Configuring the redundant communication

The figures in the following examples illustrate the connection identifiers graphically.

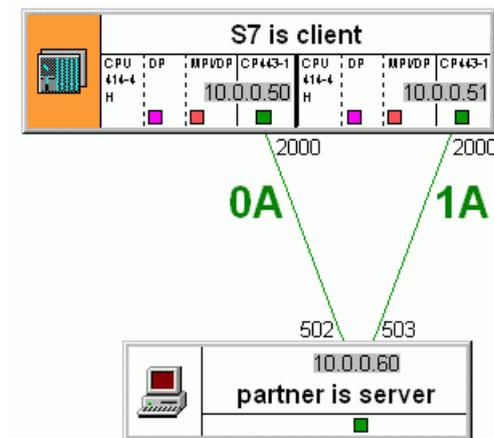
During connection configuration care must be taken that for the connection endpoints (S7: CP0 and CP1, partner: node A and node B) there must be at least 1 distinguishing characteristic for the addressing: either the IP address or the port number. CP0 and CP1 always receive different IP addresses during configuration. For this reason, during connection configuration the same port numbers can be used for both CPs. If the communications partner only has 1 IP address, a different port number must be used for each connection.

The Modbus server is usually addressed via port number 502; the Modbus client uses a port number starting at 2000.

Incorrect entry of the port numbers (e.g. identical entry of a port number) is recognized by NetPro when you close the configuration dialog.

Example: Single-sided redundancy

An example of configuration in NetPro for single-sided redundancy is shown in the following figure:

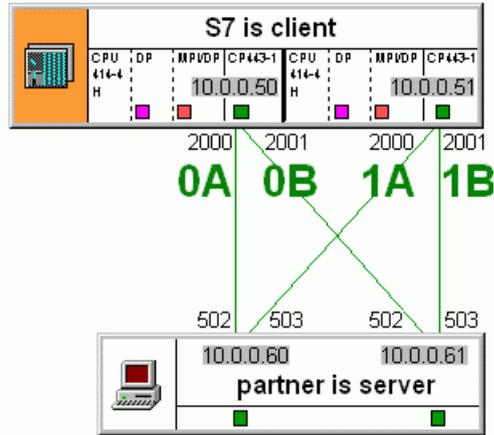


The S7 station has the IP addresses 10.0.0.50 and 10.0.0.51 and can use the port number 2000 for both connections.

The partner station has the IP address 10.0.0.60 and must be addressed via 2 different port numbers: 502 and 503.

Example: Double-sided redundancy

An example of configuration in NetPro for double-sided redundancy is shown in the following figure:



The S7 station has the IP addresses 10.0.0.50 and 10.0.0.51. For the access to node A of the link partner port number 2000 can be used both for CP0 and CP1 because the IP addresses of the two CPs are different (connection 0A and 1A).

For access to node B of the link partner, the same port number can also be used for both CPs: 2001 (connection 0B and 1B).

The partner station has the IP addresses 10.0.0.60 and 10.0.0.61. For access to the CP0 of the S7, node A and node B can use the same port number: 502 (connection 0A and 0B). For access to the CP1 of the S7, they can also use the same port number: 503 (connection 1A and 1B).

Special case - multiple use of port 502

NetPro does not allow a multiple connection with the same IP address and port number via a specified connection. If the link partner with 1 IP address only supports port 502, instead of specified connections unspecified connections must be configured in NetPro.

When using unspecified connections, a remote port 502 can be used more than once.

6.2 How FB MB_REDCL works

General information

It is also possible to operate an H station as a Modbus client and Modbus server at the same time. To do this, the relevant blocks for the client and server must be called and the required connections created in NetPro. For every redundant connection group - consisting of 2 connections (single-sided redundancy) or 4 connections (double-sided redundancy) - the Modbus block may only be called **once**.

In other words, if the H station is intended to operate as client and server in single-sided redundancy, 2 connections for the server block and 2 connections for the client block must be created in NetPro.

If the H station is intended to operate in double-sided redundancy, 4 connections must be created for the client block and 4 connections for the server block.

In terms of the library, there is no limitation regarding the maximum number of Modbus blocks running at the same time. The number of available connection resources is limited for a CPU and CP. You can find the required information on the Internet at Number of devices on an S7-300/S7-400 with Modbus/TCP (<https://support.industry.siemens.com/cs/ww/en/view/103709646>).

Tasks of the FB

The function block MB_REDCL performs the following tasks:

- Coordination of the connection(s) over which message frames are sent.
- Maintaining the transaction identifier TI

The MB_REDCL block calls the MB_CPCLI block several times internally and handles the coordination of the MB_CPCLI calls of the different connections.

The MB_CPCLI block performs the following tasks:

- Monitoring of all configured connections with AG_CNTRL
- Calling the standard functions for data transfer between CPU and CP
- Generating Modbus-specific telegram header when sending
- Checking the MODBUS-specific telegram header when receiving
- Data transfer from/to the set DB
- Time monitoring of the receipt of data

Calling the FB

The function block must be called in a cyclic OB.

Per connection group the FB MB_REDCL can be called **once**.

Online Help

For the MB_REDCL function block, there is a block online help available in the SIMATIC Manager. By selecting the block and pressing the "F1" key, the online help with the most important information on the module opens.

Initialization

The MB_REDCL function block is initialized with a positive edge at the "Init" input. The initialization parameters must be assigned according to the plant configuration. They are checked for plausibility and entered in the instance DB. During the initialization, the parameter DB is evaluated and the parameter assignment is applied in the instance DB. No initialization can be performed while a job is running (BUSY = TRUE).

If a positive edge is detected at the "Init" parameter, the actions described above are carried out. If it was possible to complete the check without error, "Init_Error" and "Init_Status" display 0.

If errors occurred during the check, this is displayed at the "Init_Error" and "Init_Status" outputs. As long as an Init error is pending, no Modbus/TCP communication is possible via the block. The Init error must be corrected first. The values at "Init_Error" and "Init_Status" are reset at the next positive edge and are assigned the latest values after a new check.

The "Init" input parameter is reset once the check is completed.

The runtime parameters are not evaluated during the initialization.

Cyclic operation of the FB

In cyclic operation, FB MB_REDCL is called e.g. in OB35. The block functions are activated based on the runtime parameters. Changes to the runtime parameters are not evaluated while a job is in progress.

In cyclic mode, the initialization parameters are not evaluated.

Programming error OB121

If the Modbus block is not yet licensed for this CPU, OB121 is called.

| |
|--|
|  WARNING |
| If OB121 is missing in the controller, the CPU is set to STOP. |

Job initialization

A positive edge at the trigger input ENQ initiates a job. Depending on the input parameters UNIT, DATA_TYPE, START_ADDRESS, LENGTH and WRITE_READ, a MODBUS request frame is generated and sent to the partner station via the TCP/IP connection. The block waits for the set time MONITOR for a response from the server.

If after restarting the H system a job is initiated by all connections during the first run-through after initialization, the error code 80B2 can be displayed **once** for the connection of the reserve CPU. This is a system property of the H system.

Sending requests via 1 connection

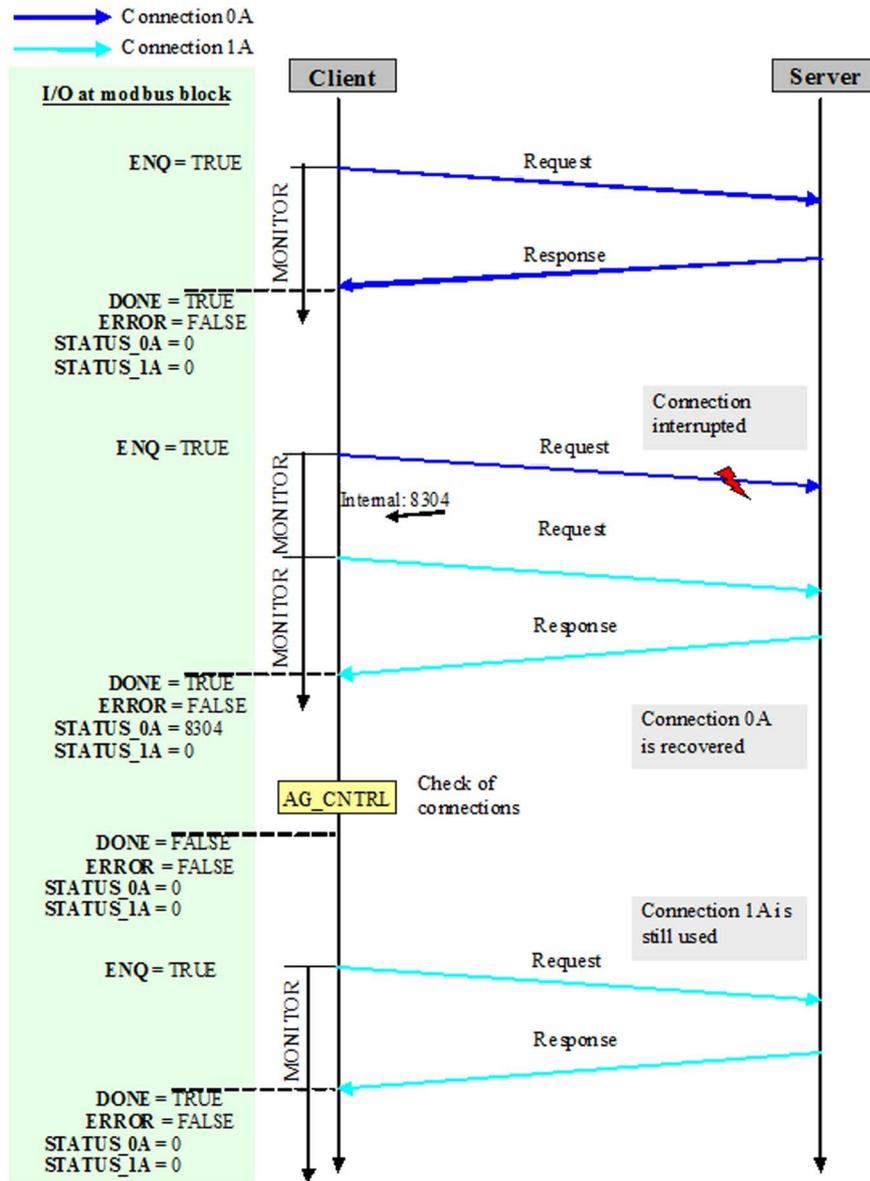
With the setting use_all_conn = FALSE, the MODBUS request is sent via 1 connection - the currently active connection. If there is a timeout (no response from the server) or a connection error, there is an attempt to send the request via the other (maximum 4) configured connections one after the other. The order for this is 0A, 1A, 0B and 1B. If a request could be transferred successfully via a connection, this connection is marked as "active" and further requests are sent via this connection. If there is a connection error on the active connection, there is once again an attempt to send the request via all configured connections one after the other.

If all attempts to send fail, ERROR and STATUS_x are set accordingly.

When a response is received, a plausibility check is performed. If this check is successful, the required actions are performed and the job is executed without errors; the output DONE is set. If errors were detected during the check, the job is ended with error, the ERROR bit is set and an error number is displayed at STATUS_x. In this case, there is no renewed attempt to send the request on the next configured connection.

There is only a switchover to the other configured connections when a connection error is detected or no response was received.

Unavailable connections are indicated by the status message A0FF in the STATUS output.



Sending requests via all connections

With the setting `use_all_conn = TRUE`, the MODBUS request is sent via all configured connections. After receiving a response on one of the connections, a plausibility check is performed. If this check is successful, the required actions are performed.

The outputs `DONE`, `ERROR` and `STATUS` are only updated when a response has been received on all configured connections.

If a valid response was received on at least 1 connection, the output `DONE` is set.

If an error was detected on all connections, the `ERROR` bit is set and the error numbers are displayed at `STATUS_x`.

6.2 How FB MB_REDCL works

If one of the configured connections has failed, the following MODBUS requests are no longer sent via the defective connection until the connection check (=> section 6.3 (Page 47)) has detected connection re-establishment.

Unavailable connections are indicated by the status message A0FF in the STATUS output.

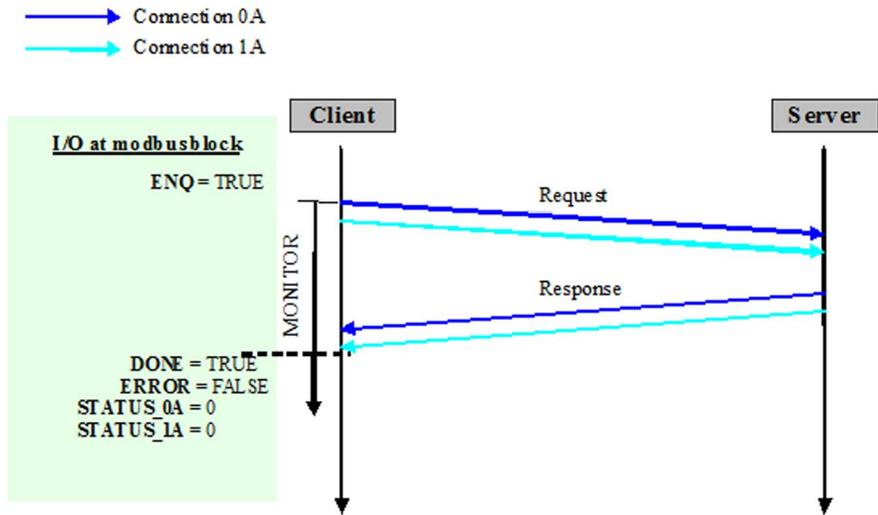


Figure 6-1 Case a) All responses are received free of errors.

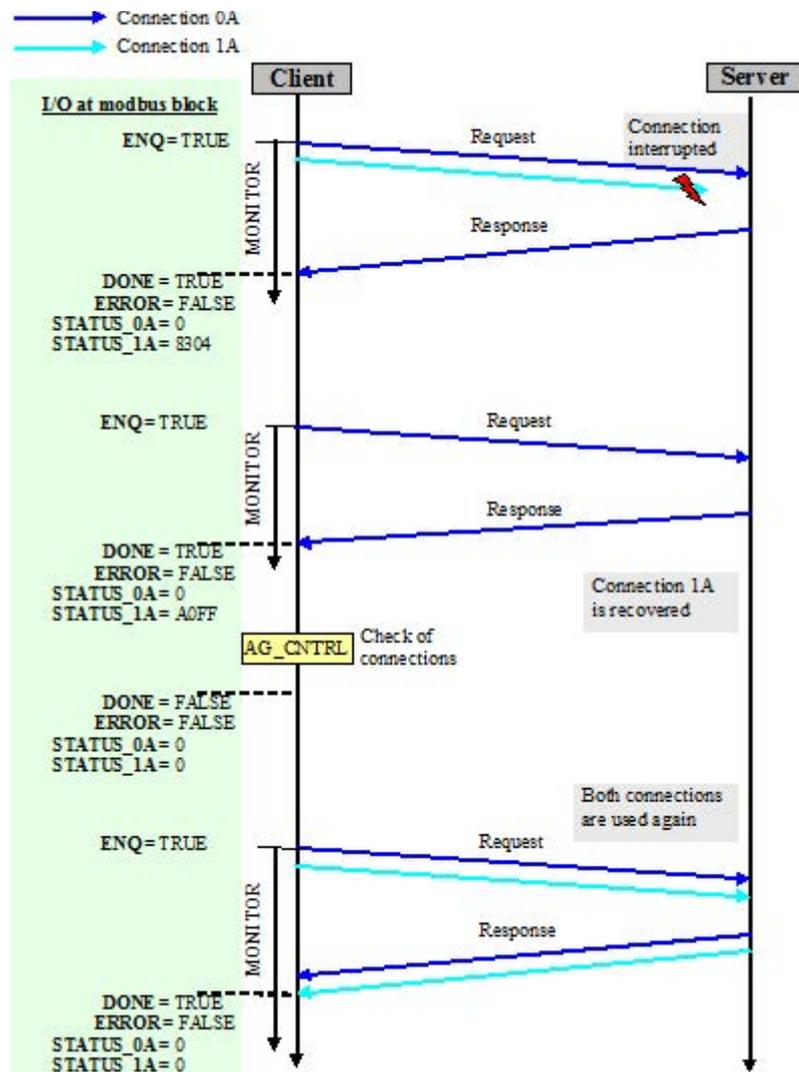


Figure 6-2 Case b) At least 1 response is not received.

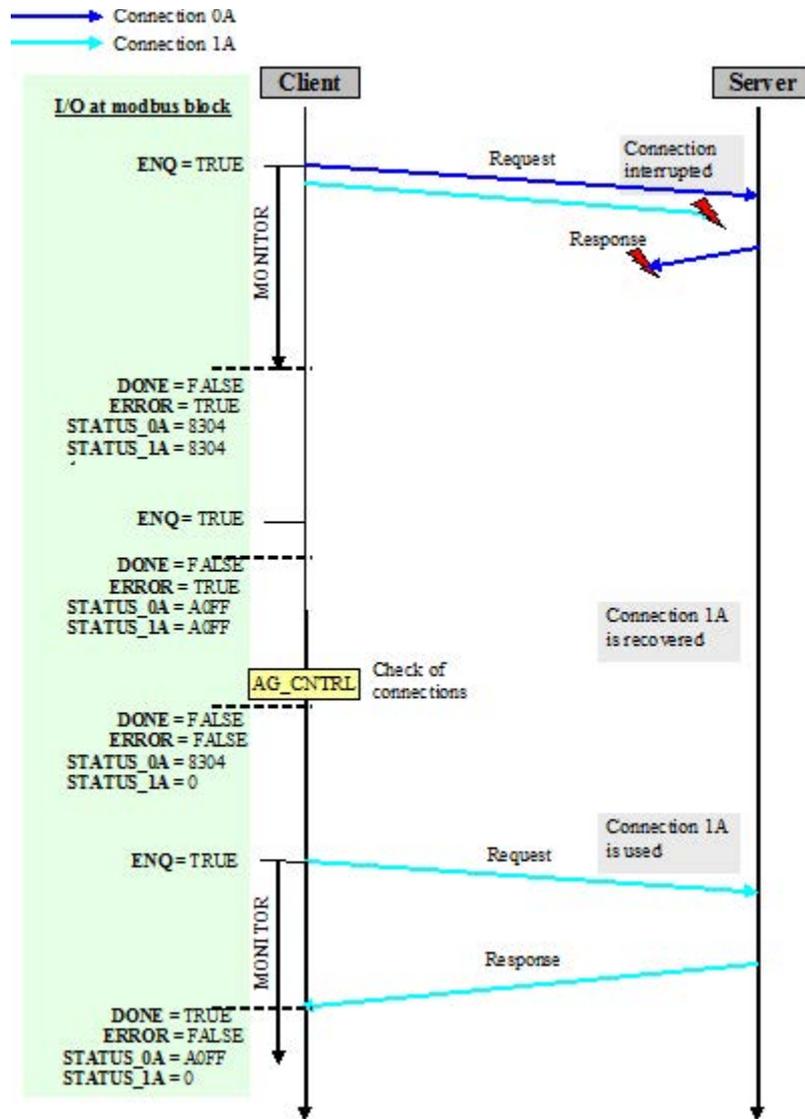


Figure 6-3 Case c) No response is received.

Data transfer CPU - CP

Data is transferred between the CP and CPU using the standard blocks AG_LSEND and AG_LRECV.

When a MODBUS job is activated by the user, the FB calls the standard blocks for the CP in the required sequence and number.

TCP/IP with CP 443-1

TCP/IP with CP 443-1 runs via static connections. The TCP connection is not terminated during error-free operation.

Due to this system property in unfavorable situations with certain error situations synchronization may be lost. If a loss of synchronization is detected, the connection is terminated and re-established with AG_CNTRL.

Connection termination by the communications partner

When the TCP connection is terminated by the communications partner, data can only be sent again after 150 ms due to system properties. This delay time is implemented by the function block.

Programming error OB121

If the Modbus block is not yet licensed for this CPU, OB121 is called.

| |
|--|
|  WARNING |
| If OB121 is missing in the controller, the CPU is set to STOP. |

6.3 Connection monitoring with AG_CNTRL

Connection check

The MB_REDCL block detects a connection error when the communications functions AG_LSEND/AG_LRECV report an error in the transfer of data frames. This means that a connection error can only be detected with long pauses between job initiations or when using use_all_conn=FALSE.

So that any connection problems occurring are detected more quickly and can be eliminated, there is the option of checking the configured connections cyclically using the "check_conn_cycle" parameter. This connection check is performed by the AG_CNTRL function.

Setting a time at "check_conn_cycle" specifies the interval at which the configured connections are checked for errors. The status of the connection check is output at STATUS_x and the alarm bits are set accordingly. As long as a defect is detected on the connection, there is no attempt to send Modbus requests.

If the connection check detects that a defective connection could be re-established, the future Modbus requests will also be sent via this connection again.

6.4 Parameters of the MB_REDCL function block

| Parameter | Decl. | Type | Description | Range of values | Init |
|------------------|-------|----------|---|---|------|
| db_param | IN | BLOCK_DB | Parameter DB, contains all connection and Modbus data for this Modbus block instance | Depends on CPU | Yes |
| REG_KEY_DB | IN | BLOCK_DB | Data block with the registration key for licensing | Depends on CPU | No |
| check_conn_cycle | IN | TIME | Cycle time for the connection check with AG_CNTRL | T#1s to T#24d20h31 m23s | Yes |
| use_all_conn | IN | BOOL | Send frame via all configured connections Send frame via one connection | TRUE FALSE | Yes |
| MONITOR | IN | TIME | Monitoring time for data reception from link partner | T#20ms to T#24d20h31 m23s647ms | No |
| ENQ | IN | BOOL | Job triggered on positive edge | TRUE/FALSE | No |
| UNIT | IN | BYTE | Unit identifier | 0 to 255 B#16#0 to B#16#FF | No |
| DATA_TYPE | IN | BYTE | Data type to be processed Coils Inputs Holding register Input register | 1 2 3 4 | No |
| START_ADDRESS | IN | WORD | MODBUS start address | 0 to 65535 W#16#0000 to W#16#FFFF | No |
| LENGTH | IN | WORD | Number of values to be processed Coils Read function Write function Inputs Read function Holding register Read function Write function Input register Read function | 1 to 2000 1 to 1968 1 to 2000 1 to 125 1 to 123 1 to 125 | No |
| WRITE_READ | IN | BOOL | Write access Read access | TRUE FALSE | No |
| id_0_A | OUT | WORD | Connection ID for CPU/CP0 according to the configuration in NetPro, read from the parameter DB | 1 to 64 W#16#1 to W#16#40 | No |
| id_1_A | OUT | WORD | Connection ID for CPU/CP1 according to the configuration in NetPro, read from the parameter DB | 0 to 64 W#16#0 to W#16#40 | No |
| id_0_B | OUT | WORD | Connection ID for CPU/CP0 according to the configuration in NetPro, read from the parameter DB Only relevant with double-sided redundancy one-sided redundancy of the link partner | 0 to 64 W#16#0 to W#16#40 | No |

| Parameter | Decl. | Type | Description | Range of values | Init |
|-------------|------------|-------------|---|---------------------------------|------|
| id_1_B | OUT | WORD | Connection ID for CPU/CP1 according to the configuration in NetPro, read from the parameter DB only relevant with double-sided redundancy | 0 to 64 W#16#0 to W#16#40 | No |
| LICENSED | OUT | BOOL | License status of the block: Block is licensed Block is not licensed | TRUE FALSE | No |
| BUSY | OUT | BOOL | Processing status of the functions AG_LSEND or AG_LRECV being processed not being processed | TRUE FALSE | No |
| DONE | OUT | BOOL | TRUE: Activated job completed without errors on at least 1 connection | TRUE/FALSE | No |
| ERROR | OUT | BOOL | TRUE: An error has occurred on all connections | TRUE/FALSE | No |
| STATUS_0A | OUT | WORD | Status for connection 0A | 0 to FFFF | No |
| STATUS_1A | OUT | WORD | Status for connection 1A | 0 to FFFF | No |
| STATUS_0B | OUT | WORD | Status for connection 0B | 0 to FFFF | No |
| STATUS_1B | OUT | WORD | Status for connection 1B | 0 to FFFF | No |
| IDENT_CODE | OUT | STRING [18] | Identification for licensing. Request the license with this identification string. | Character | No |
| RedErrS7 | OUT | BOOL | TRUE: Loss of redundancy at the S7 end | TRUE/FALSE | No |
| RedErrDev | OUT | BOOL | TRUE: Loss of redundancy at the communications partner end | TRUE/FALSE | No |
| TotComErr | OUT | BOOL | TRUE: Complete communication failure | TRUE/FALSE | No |
| Init_Error | OUT | BOOL | TRUE: An error occurred during initialization. | TRUE/FALSE | No |
| Init_Status | OUT | WORD | Status of initialization | 0 to FFFF | No |
| Init | IN/ OUT | BOOL | Initialization on a positive edge | TRUE/FALSE | No |

General information

The parameters of FB MB_REDCL are divided into two groups:

- Initialization parameters (written in lowercase)
- Runtime parameters (written in uppercase)

The initialization parameters are only evaluated and entered in the instance DB if there is a positive edge at the "Init" parameter. The initialization parameters are identified with "Yes" in the "INIT" column in the above table.

A change to the initialization parameters during operation has no effect. After changing these parameters, e.g. in test mode, the instance DB (I-DB) needs to be re-initialized with Init = TRUE.

Runtime parameters can be changed during cyclic operation. It does not make any sense to change input parameters while a job is running. Preparations for the next job and the associated changes to the parameters should only start after the previous job was ended with DONE or ERROR.

The output parameters are **displayed dynamically** and are therefore only pending for **1 CPU cycle**. This means they have to be copied to other memory areas for further processing or display in the variable table.

Ranges of values

With the ranges of values of the various parameters, CPU-specific restrictions may need to be taken into account.

db_param

The db_param parameter contains the number of the MODBUS_HPARAM_CP data block. The connection-specific and Modbus-specific parameters required for communication between the CPU and the link partner are stored in this parameter data block.

The range of values for this parameter depends on the CPU. The DB number 0 is not permitted because it is reserved for the system. The DB number is entered in plain text as "DBxy". A separate parameter data block is necessary for each Modbus block instance.

check_conn_cycle

This parameter specifies the interval at which the configured connections are checked with AG_CNTRL. The time can be set in seconds, the default value is 30 seconds. The results of the connection check are indicated at the STATUS outputs.

use_all_conn

This parameter specifies via how many connections the Modbus requests will be sent. If FALSE is set, the Modbus requests will only be sent via 1 connection. If the parameter is set to TRUE, the Modbus requests will be sent via all configured connections.

MONITOR

The monitoring time MONITOR monitors the incoming data from the link partner **on the currently active connections**. The minimum time that can be set is 20 ms. A monitoring time of approximately 1.5 seconds is recommended.

The MONITOR time monitors the arrival of the complete response from the server. If the monitoring time is exceeded, the activated job is ended with an error. The time is started after the request has been sent and stopped after the response has been entirely received.

REG_KEY_DB

The block must be licensed on every CPU. With correct entry of the registration key in this parameter, the block is licensed and MODBUS communication can be used without restrictions. The number of the data block which contains the registration key is specified. It is possible to enter several registration keys one under the other in the DB. The Modbus block searches the DB for the suitable registration key. You can find additional information in the section "Licensing (Page 31)".

ENQ

The data transfer is initiated on a positive edge. The request is generated with the input parameters UNIT, DATA_TYPE, START_ADDRESS, LENGTH and WRITE_READ. A new job can only be started if the previous job has been completed with DONE or ERROR.

UNIT

This input needs to be set according to the requirements. The FB enters this value in the request and checks the value when it receives the response.

It must be noted that some links partners expect a certain UNIT number.

With the UNIT number, different nodes downstream from a Modbus gateway can be addressed.

DATA_TYPE

The DATA_TYPE parameter indicates which MODBUS data type is processed with the current request. The following values are permitted:

| | |
|------------------|--------|
| Coils | B#16#1 |
| Inputs | B#16#2 |
| Holding register | B#16#3 |
| Input register | B#16#4 |

The data types are directly related to the used function codes.

| Data type | DATA_TYPE | Function | Length | single_write | Function code |
|------------------|-----------|----------|--------|--------------|---------------|
| Coils | 1 | read | any | irrelevant | 1 |
| Coils | 1 | write | 1 | TRUE | 5 |
| Coils | 1 | write | 1 | FALSE | 15 |
| Coils | 1 | write | >1 | irrelevant | 15 |
| Inputs | 2 | read | any | irrelevant | 2 |
| Holding register | 3 | read | any | irrelevant | 3 |
| Holding register | 3 | write | 1 | TRUE | 6 |
| Holding register | 3 | write | 1 | FALSE | 16 |
| Holding register | 3 | write | >1 | irrelevant | 16 |
| Input register | 4 | read | any | irrelevant | 4 |

START_ADDRESS

The START_ADDRESS parameter determines the first MODBUS address that is written or read.

LENGTH

The LENGTH parameter determines the number of MODBUS values that are written or read.

With read functions, a maximum of 125 registers are possible per request for holding and input registers. For coils and inputs, a maximum of 2000 bits are possible. With write functions, the maximum number of registers is 123 for holding registers and 1968 bits for coils.

The registers or bit values processed with a request must be located within one DB.

WRITE_READ

This parameter defines whether a reading or writing function is to be performed. If the input/output has the value FALSE, it is a read 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 uniquely describes the connection from the CPU via the CP to the link partner.

The number from the connection configuration in the parameter data block is displayed here.

id_0_A identifies the connection from CP0 to the link partner/node A

id_1_A identifies the connection from CP1 to the link partner/node A

id_0_B identifies the connection from CP0 to the link partner/node B

id_1_B identifies the connection from CP1 to the link partner/node B

The connection 0A is the default connection and must be configured.

If the link partner is structured as a stand-alone station, 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 status FALSE, no or an incorrect license string was entered. You will find further information in the section "Licensing (Page 31)".

BUSY

If this output is set, AG_LSEND or AG_LRECV is active.

DONE

The activated job completed without errors on at least 1 connection. With a read function, the response data from the server has already been entered in the DB; with a write function, the response to the request was received from the server.

ERROR

When this output is set, an error was detected on all active connections.

use_all_conn = FALSE:

In the case of a protocol error, ERROR is set immediately. In the case of a connection error, all configured connections are checked and ERROR is only set when all connections have problems.

use_all_conn = TRUE:

When this output is set, an error was detected on all configured connections.

The error numbers are indicated at the STATUS outputs.

STATUS_0A, STATUS_1A, STATUS_0B, STATUS_1B

When ERROR is set, the STATUS_x outputs show the error number, if ERROR is not set they show status information for the connection in question.

The error numbers and status information are described in the following section: "Diagnostics (Page 81)".

IDENT_CODE

After the CPU has started up, an 18-character identification code with which the registration key (REG_KEY) for MODBUS communication is requested is indicated at this parameter.

You can find additional information in the section "Licensing (Page 31)".

RedErrS7

The output RedErrS7 = TRUE shows a redundancy error at the SIMATIC end. With single-sided redundancy, this means that the connection from CP0 or CP1 has failed. With double-sided redundancy, both connections from CP0 or both connections from CP1 have failed.

You will find further information in the section: Diagnostics messages via alarm bits (Page 86).

RedErrDev

The output RedErrDev = TRUE shows a redundancy error at the link partner end. With single-sided redundancy this means that the connection from node point A to CP0 or CP1 has failed. With double-sided redundancy, both connections to node A or both connections to node B of the link partner have failed.

You will find further information in the section: Diagnostics messages via alarm bits (Page 86).

TotComErr

With TRUE, the output TotComErr shows complete loss of communication; in other words, all configured connections are disrupted.

You will find further information in the section: Diagnostics messages via alarm bits (Page 86).

Init_Error

If an error occurred during initialization, this is indicated by Init_Error = TRUE.

Init_Status

If Init_Error is set, the Init_Status output indicates the error number. The error numbers are described in the following section: "Diagnostics (Page 81)".

Init

The Modbus block is initialized on a positive edge at the Init parameter. The initialization can only be performed when no job is currently running. This must be ensured with ENQ = FALSE and BUSY = FALSE in the program.

The Init bit must not be set when CPU0 is in the STOP status. Otherwise error 8085 appears from RDSYSST, since no serial number can be read out.

Note

At initialization, the configured connections are terminated and re-established.

6.5 Example of the address mapping

Interpretation of the MODBUS addresses

The MODBUS data model is based on a series of memory areas that have different characteristics. These memory areas are distinguished in some systems, for example, MODICON PLCs, by means of the register address or bit address. The holding register with offset 0, for example, is referred to as register 40001 (memory type 4xxxx, reference 0001).

This often causes confusion because some manuals describe and mean the register address of the application layer and others the register/bit address actually transferred in the protocol.

In its parameters start_x and START_ADDRESS, FB MB_REDCL uses the actually transferred MODBUS address. This means that register/bit addresses from 0000H to FFFFH can be transferred with each function code.

Example: Parameter assignment for the data areas

| | | | |
|-------------|-------------|----------|--------------------|
| Data area 1 | data_type_1 | B#16#3 | Holding register |
| | db_1 | W#16#B | DB 11 |
| | start_1 | W#16#0 | Start address: 0 |
| | end_1 | W#16#1F3 | End address: 499 |
| Data area 2 | data_type_2 | B#16#3 | Holding register |
| | db_2 | W#16#C | DB 12 |
| | start_2 | W#16#2D0 | Start address: 720 |
| | end_2 | W#16#384 | End address: 900 |
| Data area 3 | data_type_3 | B#16#1 | Coils |
| | db_3 | W#16#D | DB 13 |
| | start_3 | W#16#280 | Start address: 640 |
| | end_3 | W#16#4E2 | End address: 1250 |
| Data area 4 | data_type_4 | B#16#0 | Not used |
| | db_4 | 0 | 0 |
| | start_4 | 0 | 0 |
| | end_4 | 0 | 0 |
| Data area 5 | data_type_5 | B#16#0 | Not used |
| | db_5 | 0 | 0 |
| | start_5 | 0 | 0 |
| | end_5 | 0 | 0 |
| Data area 6 | data_type_6 | B#16#0 | Not used |
| | db_6 | 0 | 0 |
| | start_6 | 0 | 0 |
| | end_6 | 0 | 0 |
| Data area 7 | data_type_7 | B#16#0 | Not used |
| | db_7 | 0 | 0 |
| | start_7 | 0 | 0 |
| | end_7 | 0 | 0 |
| Data area 8 | data_type_8 | B#16#0 | Not used |
| | db_8 | 0 | 0 |
| | start_8 | 0 | 0 |
| | end_8 | 0 | 0 |

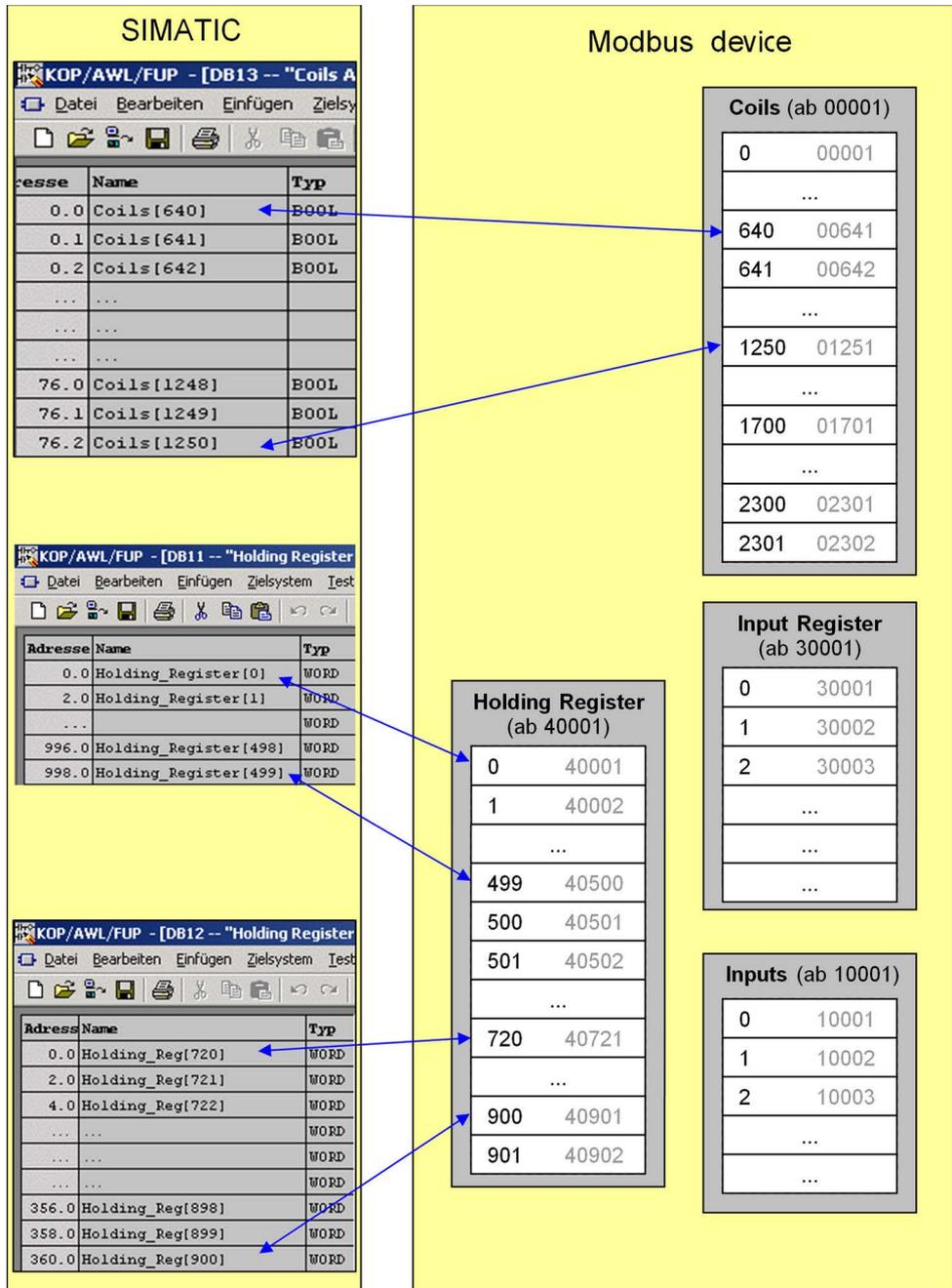
Address mapping

The figure below shows a comparison of the SIMATIC memory areas with the register-oriented and bit-oriented memory allocation of the MODBUS devices. The figure references the parameter assignment described above.

In the MODBUS device: The MODBUS addresses shown in black relate to the data link layer, those shown in gray the application layer.

In the SIMATIC: The SIMATIC addresses shown in black are the offset in the DB. In addition to this, the Modbus register numbers are shown in gray.

6.5 Example of the address mapping



6.6 Data and standard functions used by the FB

Instance DB

The MB_REDCL function block saves its data in an instance DB. This instance DB is generated by STEP 7 the first time the FB is called.

The instance data block contains parameters of the type Input, Output, Input/Output and static variables it requires to run. These variables are retentive and remain valid between FB calls. The internal execution of the FB is controlled by the variables.

Memory requirements of the instance DB:

| Instance DB | Work memory | Load memory |
|-------------|--------------|--------------|
| MB_REDCL | approx. 4 KB | approx. 6 KB |

Local variables

120 bytes of local data are required for the FB. In addition to this, there is the local data of the lower-level FB MB_CPCLI (28 bytes) and AG_CNTRL (178 bytes). This results in 326 bytes of local data for an FB MB_REDCL call.

Timers

The function block does not use any timers

Memory bits

The function block does not use any memory bits.

Standard FCs for data transfer

The function block uses the blocks AG_LSEND and AG_LRECV from the SIMATIC NET library (CP 400) for the data transfer between CPU and CP.

In addition to this, AG_CNTRL from this library is used to reset and restart the connection if an error occurs.

FB MB_REDCL has been tested and released with the following versions of the FCs:

- FC50 "AG_LSEND" version 3.1
- FC60 "AG_LRECV" version 3.1
- FC10 "AG_CNTRL" Version 1.0

MB_REDCL: SFCs for other functions

FB MB_REDCL uses the following SFCs from the standard library:

- SFB4 "TON"
- SFC20 "BLKMOV"
- SFC24 "TEST_DB"
- SFC51 "RDSYSST"
- SFC52 "WR_USMSG"

MB_CPCLI: SFCs for other functions

FB MB_CPCLI uses the following SFCs from the standard library:

- SFC20 "BLKMOV"
- SFC24 "TEST_DB"
- SFB4 "TON"

Additional information

The parameter TI is kept by the MB_REDCL block internally and incremented by 1 with each new job.

Error 8304: When the connection from the CP to the link partner is interrupted, this is detected by the CP and saved as error code 8304.

If a communications job is started, the error code 8304 is output first since this is still stored - even if the connection is re-established. This is a system property of the CP.

When the MODBUS block outputs ERROR = TRUE and STATUS = 8304, the communications job should be restarted.

The time in which a loss of connection can be detected is determined by the Keep Alive Time parameter. You will find this parameter in the properties of the CP in HW Config.

6.7 Renaming / rewiring functions and function blocks

Objective

If the numbers of the standard functions are already being used in your project or the number range is reserved for other applications (e.g. in CFC), you can rewire the internally called functions FC50, FC60 or FC10. The MB_REDCL block is BlockPrivacy protected. This prevents the internally called block, MB_CPCLI, from being rewired.

The system functions SFC20, SFC24, SFC51 and SFC52 and the system function block SFB4 cannot be renamed/rewired.

Rewiring

It is not possible to rewire the blocks in the library itself. To rewire, copy the blocks from the Modbus/TCP library and all lower-level blocks from the SIMATIC NET library into a new project.

Now proceed as follows:

1. Call up information about the addresses used with "Options > Reference Data > Display".
2. Set the address priority in the object properties of the block folder to "Absolute value".
3. In the SIMATIC Manager, select the "Options > Rewire" function to rewire the addresses into the unused range.
4. To be able to continue using symbols in diagnostics tools, update the symbol table with the changes.

If you want to check the changes, select "Options >Reference Data >Display".

Function block MB_REDSV – Modbus server

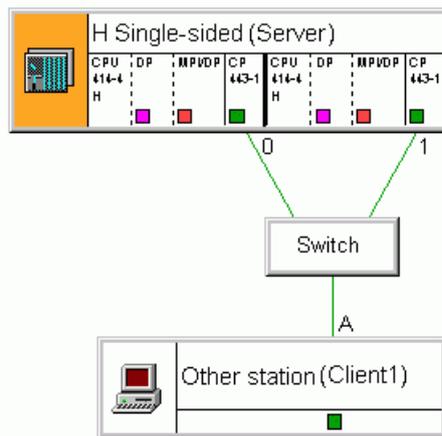
7.1 Configuring the redundant communication

General information

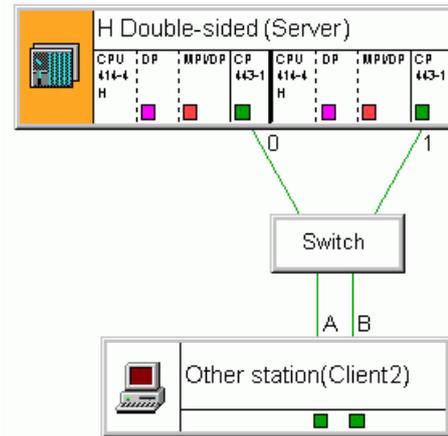
The CP is the server when the remote partner initiates the reading or writing of the data from or to the S7.

The communications partner of the H system can be stand-alone or also redundantly structured (redundancy single-sided or double-sided).

Single-sided redundancy



Double-sided redundancy



Configuration in HW Config

When configuring the hardware in HW Config, CP0 and CP1 receive different input addresses and different IP addresses so that they can be addressed uniquely in the S7 program or by communications partners.

Configuration in NetPro

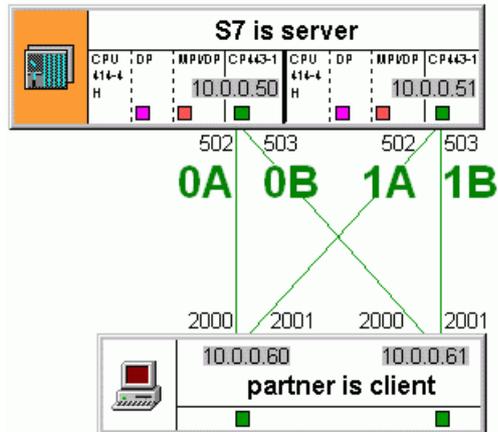
For every possible connection between the communications partners, 1 connection must be configured in NetPro.

With single-sided redundancy, 1 connection is created for CPU0/CP0 and 1 connection for CPU1/CP1:

- Connection from CPU0/CP0 to partner => **connection 0A**
- Connection from CPU1/CP1 to partner => **connection 1A**

Example: Double-sided redundancy

An example of configuration in NetPro for double-sided redundancy is shown in the following figure:



The S7 station has the IP addresses 10.0.0.50 and 10.0.0.51. For the access from node A of the link partner port number 502 can be used both for CP0 and CP1 because the IP addresses of the two CPs are different (connection **0A** and **1A**).

For access from node B of the link partner, the same port number can also be used for both CPs: 503 (connection **0B** and **1B**).

The partner station has the IP addresses 10.0.0.60 and 10.0.0.61. For access to the CP0 of the S7, node A and node B can use the same port number: 2000 (connection **0A** and **0B**). For access to the CP1 of the S7, they can also use the same port number: 2001 (connection **1A** and **1B**).

7.2 Using connections on port 502

Some CPs can multiplex TCP connections. This means that several Modbus/TCP clients can connect to port 502 of the CP. The CP acts as a Modbus/TCP server.

This is where you can find information about the technical specifications for the SIMATIC Modbus/TCP blocks and learn which CPUs and CPs have been released: Technical specifications of the Modbus/TCP blocks

(<http://support.automation.siemens.com/WW/view/en/104946406>)

Requirements

To be able to use this function, the following settings need to be made in the connection parameter assignment in NetPro:

- CP is server
- Port 502 as local port
- Passive connection establishment

Note

Note that only 1 connection is configured in NetPro regardless of how many clients access the CP as a server.

Number of connections

Via port 502, the CP can communicate with a maximum of 4 clients at any one time.

Special case "Use of multiport 502 with double-sided redundancy".

With redundancy on both sides, 2 connections per CP are normally created in NetPro: 0A and 0B as well as 1A and 1B. When configuration is performed in the wizard, the parameter DB is also created with redundancy on both sides.

When using the multiport, i.e. when both clients access the CP via port 502, only one connection per CP is created in NetPro: 0(A/B) and 1(A/B). For this reason, set one-sided redundancy in the wizard during configuration by selecting the "S7 is redundant" check box.

Displaying the status of the connection

Both in NetPro online and in the special diagnostics of the CP, the status of the connection can be displayed.

Since only 1 connection is configured in NetPro, the display represents the status of all TCP connections to the various clients.

If no client has yet established a connection, "Passive connection establishment in progress" is displayed. As soon as a client has established a connection, "Connection established" is displayed. It is not possible to determine how many clients are currently connected to the CP.

Response to errors

In certain error situations, the CP needs to terminate and then re-establish the connection to be able to return to the initial state. This action is handled by the Modbus block. All existing connections via port 502 are then terminated.

Tips for the user program

If there are multiple connections via port 502, it is not possible to differentiate in the user program which client sent the current Modbus/TCP request. If the clients use different UNIT numbers, a distinction can be made by evaluating these in the program.

7.3 How FB MB_REDSV works

General information

It is also possible to operate an H station as a Modbus client and Modbus server at the same time. To do this, the relevant blocks for the client and server must be called and the required connections created in NetPro. For every redundant connection group - consisting of 2 connections (single-sided redundancy) or 4 connections (double-sided redundancy) - the Modbus block may only be called **once**.

In other words, if the H station is intended to operate as client and server in single-sided redundancy, 2 connections for the server block and 2 connections for the client block must be created in NetPro. If the H station is intended to operate in double-sided redundancy, 4 connections must be created for the client block and 4 connections for the server block.

In terms of the library, there is no limitation regarding the maximum number of Modbus blocks running at the same time. The number of available connection resources is limited for a CPU and CP. You can find the required information on the Internet at Number of devices on an S7-300/S7-400 with Modbus/TCP

(<https://support.industry.siemens.com/cs/ww/en/view/103709646>).

Tasks of the FB

The MB_REDSV block calls the MB_CPSRV block several times internally and handles the coordination of the MB_CPSRV calls of the different connections.

The MB_CPSRV block performs the following tasks:

- Monitoring of all configured connections with AG_CNTRL
- Calling the standard functions for data transfer between CPU and CP
- Generating Modbus-specific telegram header when sending
- Checking the MODBUS-specific telegram header when receiving
- Checking if the data areas addressed by the client exist
- Generating exception telegrams if an error has occurred

| Exception code | Meaning |
|----------------|--|
| 1 | The sent function code is not supported. |
| 2 | There was access to an address that is non-existent or is not permitted. |
| 3 | An invalid length was specified for this function code. |

- Data transfer from/to the set DB
- Time monitoring of the receipt of data

Calling the FB

The function block must be called in a cyclic OB.

Online Help

For the MB_REDSV function block, there is a block online help available in the SIMATIC Manager. By selecting the block and pressing the "F1" key, the online help with the most important information on the module opens.

Initialization

The MB_REDSV function block is initialized with a positive edge at the "Init" input. The initialization parameters must be assigned according to the plant configuration. They are checked for plausibility and entered in the instance DB. During the initialization, the parameter DB is evaluated and the parameter assignment is applied in the instance DB.

If a positive edge is detected at the "Init" parameter, the actions described above are carried out. If it was possible to complete the check without error, "Init_Error" and "Init_Status" display 0.

If errors occurred during the check, this is displayed at the "Init_Error" and "Init_Status" outputs. As long as an Init error is pending, no Modbus/TCP communication is possible via the block. The Init error must be corrected first. The values at "Init_Error" and "Init_Status" are reset at the next positive edge and are assigned the latest values after a new check.

The "Init" input parameter is reset once the check is completed.

The runtime parameters are not evaluated during the initialization.

Cyclic operation of the FB

In cyclic operation, FB MB_REDSV is called e.g. in OB 35. The block functions are activated based on the runtime parameters.

In cyclic mode, the initialization parameters are not evaluated.

After restarting the H system the error code 80B2 can be displayed **once** for the connection of the reserve CPU. This is a system property of the H system.

Activating the FB

With a positive level at the ENR trigger input, the FB is ready to receive a request from the client. The server is passive at this stage.

If ENR = TRUE, all configured connections are active and ready to receive. There is no switchover between the connections. The client can optionally send only via 1 or all connections. The received requests are checked. If the check is successful, the response is generated and sent. The user is informed about the completed data transfer by setting the NDR_x bit for the relevant connection.

A request with an error results in an error message. The ERROR bit of the connection is set and the error number is indicated in the STATUS_x parameter. Depending on the type of error, the request from the client is replied to either with an exception telegram or no response is sent to the client.

Instance DB: Information on the request of the client

With each request of the client, the values of the executed job are saved in the I-DB of the server in an information block. When necessary they can be read out in the user program. The following values are stored temporarily in the I-DB for every connection and are valid if NDR = TRUE:

| Address in the I-DB for connection 0A | Tag name | Description |
|---------------------------------------|-----------------------------|--|
| DBX 80.0 | CONNECTION[1].WRITE_READ | TRUE: Writing in S7 FALSE: Reading from S7 |
| DBB 81 | CONNECTION[1].UNIT | Unit number |
| DBB 82 | CONNECTION[1].DATA_TYPE | Addressed data type 1: Coils 2: Inputs 3: Holding register 4: Input register |
| DBW 84 | CONNECTION[1].START_ADDRESS | Start address |
| DBW 86 | CONNECTION[1].LENGTH | Number of processed registers / bits |
| DBW 88 | CONNECTION[1].TI | Transaction identifier (sequential number) |
| DBD 90 | CONNECTION[1].Cnt_NDR | Counter for jobs processed error-free |
| DBD 94 | CONNECTION[1].Cnt_ERROR | Counter for detected errors |

For the connection 1A (CONNECTION[2]), the information block begins at address DBX 114.0.

For the connection 0B (CONNECTION[3]), the information block begins at address DBX 148.0.

For the connection 1B (CONNECTION[4]), the information block begins at address DBX 182.0.

Data transfer CPU - CP

Data is transferred between the CP and CPU using the standard blocks AG_LSEND and AG_LRECV.

When a request is received from the client, the FB calls the standard blocks for the CP in the required order and number.

TCP/IP with CP443-1

TCP/IP with CP 443-1 runs via static connections. The TCP connection is not terminated during error-free operation.

Due to this system property in unfavorable situations with certain error situations synchronization may be lost. If a loss of synchronization is detected, the connection is terminated and re-established with AG_CNTRL.

Connection termination by the communications partner

When the TCP connection is terminated by the communication partner, the next requests can only be received after 1 second due to system properties. This delay time is implemented by the function block.

Programming error OB121

If the Modbus block is not yet licensed for this CPU, OB121 is called.

| |
|--|
|  WARNING |
| If OB121 is missing in the controller, the CPU is set to STOP. |

7.4 Connection monitoring with AG_CNTRL

Connection check

The MB_REDSV block detects a connection error when the communications functions AG_LSEND/AG_LRECV report an error in the data transfer. Disrupted connections are marked internally by the FB accordingly and are not used as long as the fault is present.

The connection check detects the AG_CNTRL function when a connection is available again. The "check_conn_cycle" parameter defines the time interval at which a check of the configured connections is performed.

The status of the connection check is output at STATUS_x and the alarm bits are updated accordingly.

7.5 Parameters of the MB_REDSV function block

| Parameter | Decl. | Type | Description | Range of values | Init |
|------------------|-------|----------|---|--------------------------------|------|
| db_param | IN | BLOCK_DB | Parameter DB, contains all connection and Modbus data for this Modbus block instance | Depends on CPU | Yes |
| REG_KEY_DB | IN | BLOCK_DB | Data block with the registration key for licensing | Depends on CPU | No |
| check_conn_cycle | IN | TIME | Cycle time for the connection check with AG_CNTRL | T#1s to T#24d20h31m23s | Yes |
| MONITOR | IN | TIME | Monitoring time for data reception from link partner | T#20ms to T#+24d20h31m23s647ms | No |
| ENR | IN | BOOL | Ready to receive if positive level | TRUE/FALSE | No |
| id_0_A | OUT | WORD | Connection ID for CPU/CP0 according to the configuration in NetPro, read from the parameter DB | 1 to 64 W#16#1 to W#16#40 | No |
| id_1_A | OUT | WORD | Connection ID for CPU/CP1 according to the configuration in NetPro, read from the parameter DB | 0 to 64 W#16#0 to W#16#40 | No |
| id_0_B | OUT | WORD | Connection ID for CPU/CP0 according to the configuration in NetPro, read from the parameter DB Only relevant with double-sided redundancy one-sided redundancy of the link partner | 0 to 64 W#16#0 to W#16#40 | No |

| Parameter | Decl. | Type | Description | Range of values | Init |
|-------------|------------|-------------|---|------------------------------|------|
| id_1_B | OUT | WORD | Connection ID for CPU/CP1 according to the configuration in NetPro, read from the parameter DB only relevant with double-sided redundancy | 0 to 64 W#16#0 to W#16#40 | No |
| LICENSED | OUT | BOOL | License status of the block: Block is licensed Block is not licensed | TRUE FALSE | No |
| BUSY | OUT | BOOL | Processing status of the functions AG_LSEND or AG_LRECV being processed not being processed | TRUE FALSE | No |
| NDR_0A | OUT | BOOL | TRUE: The request of the client was executed on connection 0A and responded to | TRUE/FALSE | No |
| ERROR_0A | OUT | BOOL | TRUE: An error has occurred on connection 0A. | TRUE/FALSE | No |
| STATUS_0A | OUT | WORD | Status for connection 0A | 0 to FFFF | No |
| NDR_1A | OUT | BOOL | TRUE: the request of the client was executed on connection 1A and responded to | TRUE/FALSE | No |
| ERROR_1A | OUT | BOOL | TRUE: An error has occurred on connection 1A. | TRUE/FALSE | No |
| STATUS_1A | OUT | WORD | Status for connection 1A | 0 to FFFF | No |
| NDR_0B | OUT | BOOL | TRUE: the request of the client was executed on connection 0B and responded to | TRUE/FALSE | No |
| ERROR_0B | OUT | BOOL | TRUE: An error has occurred on connection 0B. | TRUE/FALSE | No |
| STATUS_0B | OUT | WORD | Status for connection 0B | 0 to FFFF | No |
| NDR_1B | OUT | BOOL | TRUE: the request of the client was executed on connection 1B and responded to | TRUE/FALSE | No |
| ERROR_1B | OUT | BOOL | TRUE: An error has occurred on connection 1B. | TRUE/FALSE | No |
| STATUS_1B | OUT | WORD | Status for connection 1B | 0 to FFFF | No |
| IDENT_CODE | OUT | STRING [18] | Identification for licensing. Request the license with this identification string. | Character | No |
| RedErrS7 | OUT | BOOL | TRUE: Loss of redundancy at the S7 end | TRUE/FALSE | No |
| RedErrDev | OUT | BOOL | TRUE: Loss of redundancy at the communications partner end | TRUE/FALSE | No |
| TotComErr | OUT | BOOL | TRUE: Complete communication failure | TRUE/FALSE | No |
| Init_Error | OUT | BOOL | TRUE: An error occurred during initialization. | TRUE/FALSE | No |
| Init_Status | OUT | WORD | Status of initialization | 0 to FFFF | No |
| Init | IN/ OUT | BOOL | Initialization on a positive edge | TRUE/FALSE | No |

General information

The parameters of FB MB_REDSV are divided into two groups:

- Initialization parameters (written in lowercase)
- Runtime parameters (written in uppercase)

The initialization parameters are only evaluated and entered in the instance DB if there is a positive edge at the "Init". The initialization parameters are identified with "Yes" in the "INIT" column in the above table.

A change to the initialization parameters during operation has no effect. After changing these parameters, e.g. in test mode, the instance DB (I-DB) needs to be re-initialized with Init = TRUE.

Runtime parameters can be changed during cyclic operation. It does not make any sense to change input parameters while a job is running.

The output parameters are **displayed dynamically** and are therefore only pending for **1 CPU cycle**. This means they have to be copied to other memory areas for further processing or display in the variable table.

Ranges of values

With the ranges of values of the various parameters, CPU-specific restrictions may need to be taken into account.

db_param

The db_param parameter contains the number of the MODBUS_HPARAM_CP data block. The connection-specific and Modbus-specific parameters required for communication between the CPU and the link partner are stored in this parameter data block.

The range of values for this parameter depends on the CPU. The DB number 0 is not permitted because it is reserved for the system. The DB number is entered in plain text as "DBxy". A separate parameter data block is necessary for each Modbus block instance.

REG_KEY_DB

The block must be licensed on every CPU. With correct entry of the registration key in this parameter, the block is licensed and MODBUS communication can be used without restrictions. The number of the data block which contains the registration key is specified. The DB number is entered in plain text as "DBxy". It is possible to enter several registration keys one under the other in the DB. The Modbus block searches the DB for the suitable registration key.

You will find further information in the section "Licensing"

check_conn_cycle

This parameter specifies the interval at which the configured connections are checked with AG_CNTRL. The time can be set in seconds, the default value is 30 seconds. The results of the connection check are indicated at the STATUS outputs.

MONITOR

The monitoring time MONITOR monitors the incoming data from the link partner on the currently active connections. The minimum time that can be set is 20 ms. A monitoring time of approximately 1.5 seconds is recommended.

The input of the second part of the frame is monitored with the MONITOR time. If the monitoring time is exceeded, an error is reported. The time is started after receipt of the MODBUS-specific telegram header and stopped after the request has been entirely received.

ENR

With a positive level at the input, the FB is activated. Requests from the client can be received. If the input is not set, the data is fetched from the CP and discarded. All configured connections are listened to and incoming requests are processed.

When configured as a server, ENR = TRUE must not be initialized, otherwise an error message is displayed at init_error and init_status.

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 uniquely describes the connection from the CPU via the CP to the link partner.

The number from the connection configuration is displayed here.

id_0_A identifies the connection from CP0 to the link partner/node A

id_1_A identifies the connection from CP1 to the link partner/node A

id_0_B identifies the connection from CP0 to the link partner/node B

id_1_B identifies the connection from CP1 to the link partner/node B

The connection 0A is the default connection and must be configured.

If the link partner is structured as a stand-alone station, 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 status FALSE, no or an incorrect license string was entered. You can find additional information in the section "Licensing (Page 31)".

BUSY

If this output is set, AG_LSEND or AG_LRECV is active.

NDR_0A, NDR_1A, NDR_0B, NDR_1B

The output indicates error-free completed data exchange with the client on this connection.

ERROR_0A, ERROR_1A, ERROR_0B, ERROR_1B

When this output is set, an error was detected in a request from the client or when sending the response on this connection. The corresponding error number is indicated at the STATUS_x output.

STATUS_0A, STATUS_1A, STATUS_0B, STATUS_1B

When ERROR is set, the STATUS_x outputs show the error number, if ERROR is not set they show status information for the connection in question.

The error numbers and status information are described in the following section: Diagnostics (Page 81).

IDENT_CODE

After the CPU has started up, an 18-character identification code with which the registration key (REG_KEY) for MODBUS communication is requested is indicated at this parameter.

You will find further information in the following section: Licensing (Page 31).

RedErrS7

The output RedErrS7 = TRUE shows a redundancy error at the SIMATIC end. With single-sided redundancy, this means that the connection from CP0 or CP1 has failed. With double-sided redundancy, both connections from CP0 or both connections from CP1 have failed.

You will find further information in the following section: Diagnostics messages via alarm bits (Page 86).

RedErrDev

The output RedErrDev = TRUE shows a redundancy error at the link partner end. With single-sided redundancy this means that the connection from node point A to CP0 or CP1 has failed. With double-sided redundancy, both connections to node A or both connections to node B of the link partner have failed.

You will find further information in the section: Diagnostics messages via alarm bits (Page 86).

TotComErr

With TRUE, the output TotComErr shows complete loss of communication; in other words, all configured connections are disrupted.

You will find further information in the section: Diagnostics messages via alarm bits (Page 86).

Init_Error

If an error occurred during initialization, this is indicated by Init_Error = TRUE.

Init_Status

If Init_Error is set, the Init_Status output indicates the error number. The error numbers are described in the following section: Diagnostics (Page 81).

Init

The Modbus block is initialized on a positive edge at the Init parameter. The initialization can only be performed when no job is currently running. The client must not send a request during this time. ENR must be set to FALSE.

The Init bit must not be set when CPU0 is in the STOP status. Otherwise error 8085 appears from RDSYSST, since no serial number can be read out.

Note

At initialization, the configured connections are terminated and re-established.

7.6 Example of the address mapping

Interpretation of the MODBUS addresses

The MODBUS data model is based on a series of memory areas that have different characteristics. These memory areas are distinguished in some systems, for example, MODICON PLCs, by means of the register address or bit address. The holding register with offset 0, for example, is referred to as register 40001 (memory type 4xxxx, reference 0001).

This often causes confusion because some manuals describe and mean the register address of the application layer and others the register/bit address actually transferred in the protocol.

In its parameters start_x and START_ADDRESS, FB MB_REDSV uses the actually transferred MODBUS address. This means that register/bit addresses from 0000H to FFFFH can be transferred with each function code.

7.6 Example of the address mapping

Example: Parameter assignment for the data areas

| | | | |
|-------------|-------------|----------|--------------------|
| Data area 1 | data_type_1 | B#16#3 | Holding register |
| | db_1 | W#16#B | DB 11 |
| | start_1 | W#16#0 | Start address: 0 |
| | end_1 | W#16#1F3 | End address: 499 |
| Data area 2 | data_type_2 | B#16#3 | Holding register |
| | db_2 | W#16#C | DB 12 |
| | start_2 | W#16#2D0 | Start address: 720 |
| | end_2 | W#16#384 | End address: 900 |
| Data area 3 | data_type_3 | B#16#1 | Coils |
| | db_3 | W#16#D | DB 13 |
| | start_3 | W#16#280 | Start address: 640 |
| | end_3 | W#16#4E2 | End address: 1250 |
| Data area 4 | data_type_4 | B#16#0 | Not used |
| | db_4 | 0 | 0 |
| | start_4 | 0 | 0 |
| | end_4 | 0 | 0 |
| Data area 5 | data_type_5 | B#16#0 | Not used |
| | db_5 | 0 | 0 |
| | start_5 | 0 | 0 |
| | end_5 | 0 | 0 |
| Data area 6 | data_type_6 | B#16#0 | Not used |
| | db_6 | 0 | 0 |
| | start_6 | 0 | 0 |
| | end_6 | 0 | 0 |
| Data area 7 | data_type_7 | B#16#0 | Not used |
| | db_7 | 0 | 0 |
| | start_7 | 0 | 0 |
| | end_7 | 0 | 0 |
| Data area 8 | data_type_8 | B#16#0 | Not used |
| | db_8 | 0 | 0 |
| | start_8 | 0 | 0 |
| | end_8 | 0 | 0 |

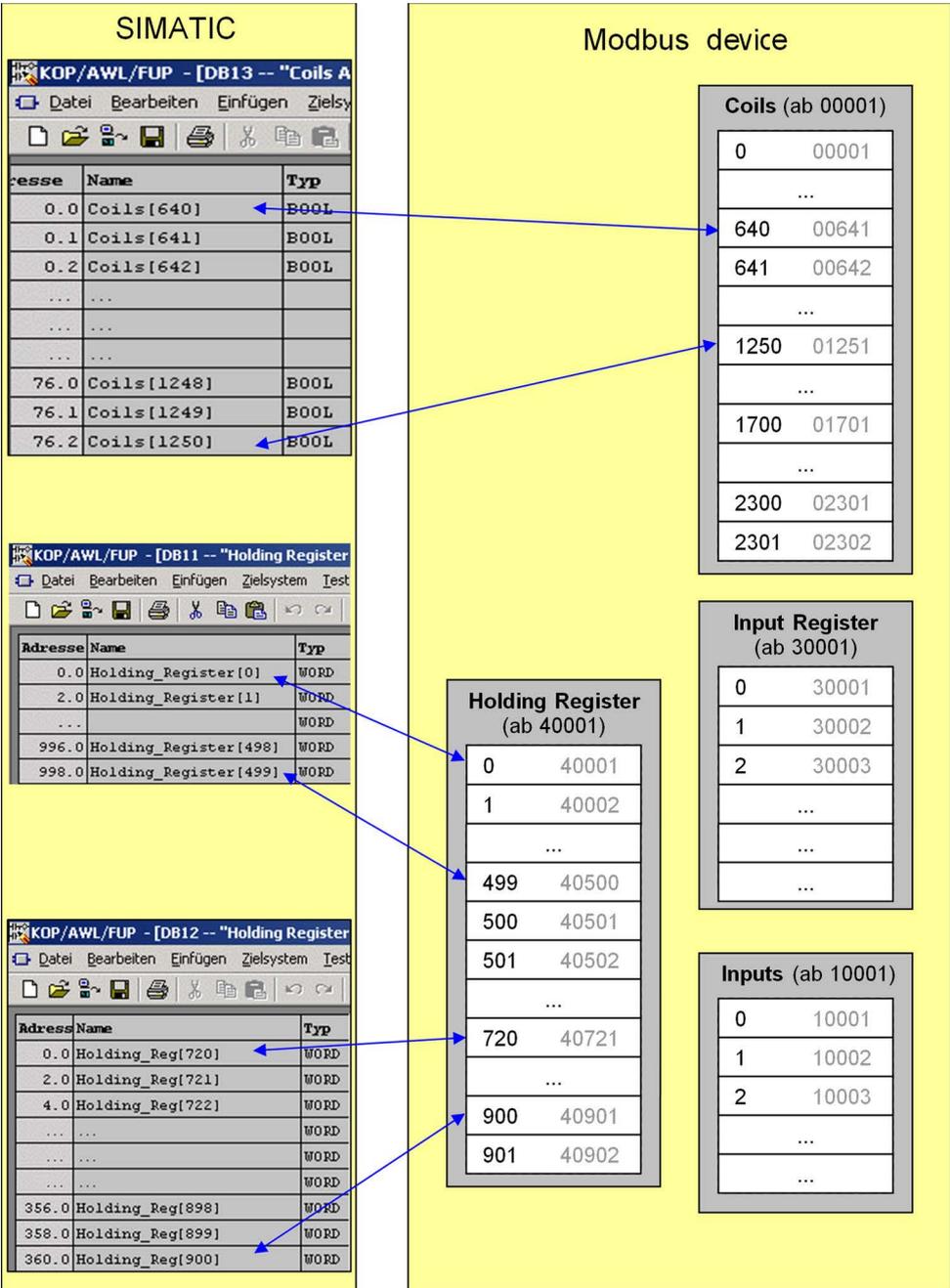
Address mapping

The figure below shows a comparison of the SIMATIC memory areas with the register-oriented and bit-oriented memory allocation of the MODBUS devices. The figure references the parameter assignment described above.

In the MODBUS device: The MODBUS addresses shown in black relate to the data link layer, those shown in gray the application layer.

In the SIMATIC: The SIMATIC addresses shown in black are the offset in the DB. In addition to this, the Modus register numbers are shown in gray.

7.6 Example of the address mapping



7.7 Data and standard functions used by the FB

Instance DB

The MB_REDSV function block saves its data in an instance DB. This instance DB is generated by STEP 7 the first time the FB is called.

The instance data block contains parameters of the type Input, Output, Input/Output and static variables it requires to run. These variables are retentive and remain valid between FB calls. The internal execution of the FB is controlled by the variables.

Memory requirements of the instance DB:

| Instance DB | Work memory | Load memory |
|-------------|--------------|--------------|
| MB_REDSV | approx. 4 KB | approx. 6 KB |

Local variables

136 bytes of local data are required for the FB. In addition to this, there is the local data of the lower-level FB MB_CPSRV (30 bytes) and AG_CNTRL (178 bytes). This results in 344 bytes of local data for an FB MB_REDSV call.

Timers

The function block does not use any timers

Memory bits

The function block does not use any memory bits.

Standard FCs for data transfer

The function block uses the blocks AG_LSEND and AG_LRECV from the SIMATIC NET library -> CP 400 for the data transfer between CPU and CP.

In addition to this, AG_CNTRL from the same library is used to reset and restart the connection if an error occurs.

FB MB_REDSV has been tested and released with the following versions of the FCs:

- FC50 "AG_LSEND" version 3.1
- FC60 "AG_LRECV" version 3.1
- FC10 "AG_CNTRL" Version 1.0

SFCs for other functions

FB MB_REDSV uses the following SFCs from the standard library:

- SFB4 "TON"
- SFC20 "BLKMOV"
- SFC24 "TEST_DB"
- SFC51 "RDSYSST"
- SFC52 "WR_USMSG"

MB_CPSRV: SFCs for other functions

FB MB_CPSRV uses the following SFCs from the standard library:

- SFC20 "BLKMOV"
- SFC24 "TEST_DB"
- SFB4 "TON"

Additional information

The time in which a loss of connection can be detected is determined by the Keep Alive Time parameter. You will find this parameter in the properties of the CP in HW Config.

7.8 Renaming / rewiring functions and function blocks

Objective

If the numbers of the standard functions are already being used in your project or the number range is reserved for other applications (e.g. in CFC), you can rewire the internally called functions FC50, FC60 or FC10. The MB_REDSV block is BlockPrivacy protected. This prevents the internally called block, MB_CPSRV, from being rewired.

The system functions SFC20, SFC24, SFC51 and SFC52 and the system function block SFB4 cannot be renamed/rewired.

Rewiring

It is not possible to rewire the blocks in the library itself. To rewire, copy the blocks from the Modbus/TCP library and all lower-level blocks from the SIMATIC NET library into a new project.

Now proceed as follows:

1. Call up information about the addresses used with "Options > Reference Data > Display".
2. Set the address priority in the object properties of the block folder to "Absolute value".
3. In the SIMATIC Manager, select the "Options > Rewire" function to rewire the addresses into the unused range.
4. To be able to continue using symbols in diagnostics tools, update the symbol table with the changes.

If you want to check the changes, select "Options >Reference Data >Display".

Additional blocks

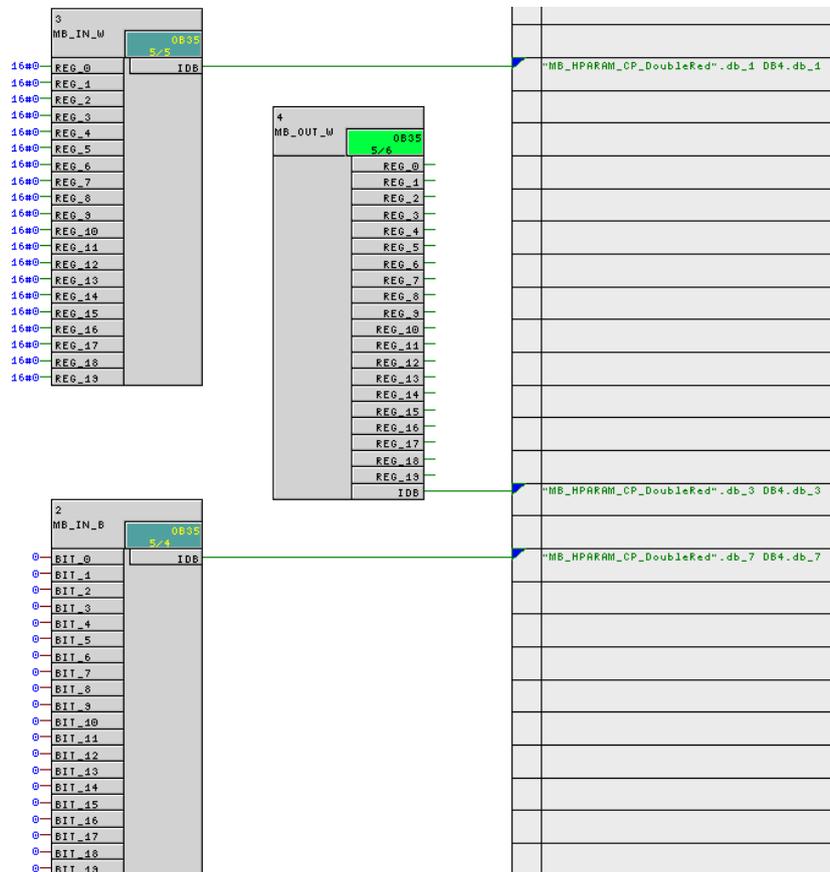
8.1 Support in CFC

General

To support configuration in CFC, it is possible to configure the Modbus values not using global DBs but using "data collector FBs". In this case, the send and receive buffers for the values are dragged to the CFC chart.

Application- example

The data collector FBs are placed in the CFC chart. The "IDB" output is connected to the DB parameters db_1 to db_8 in the parameter data block. The Modbus values can subsequently be interconnected directly from the channel blocks with the data collector FB.



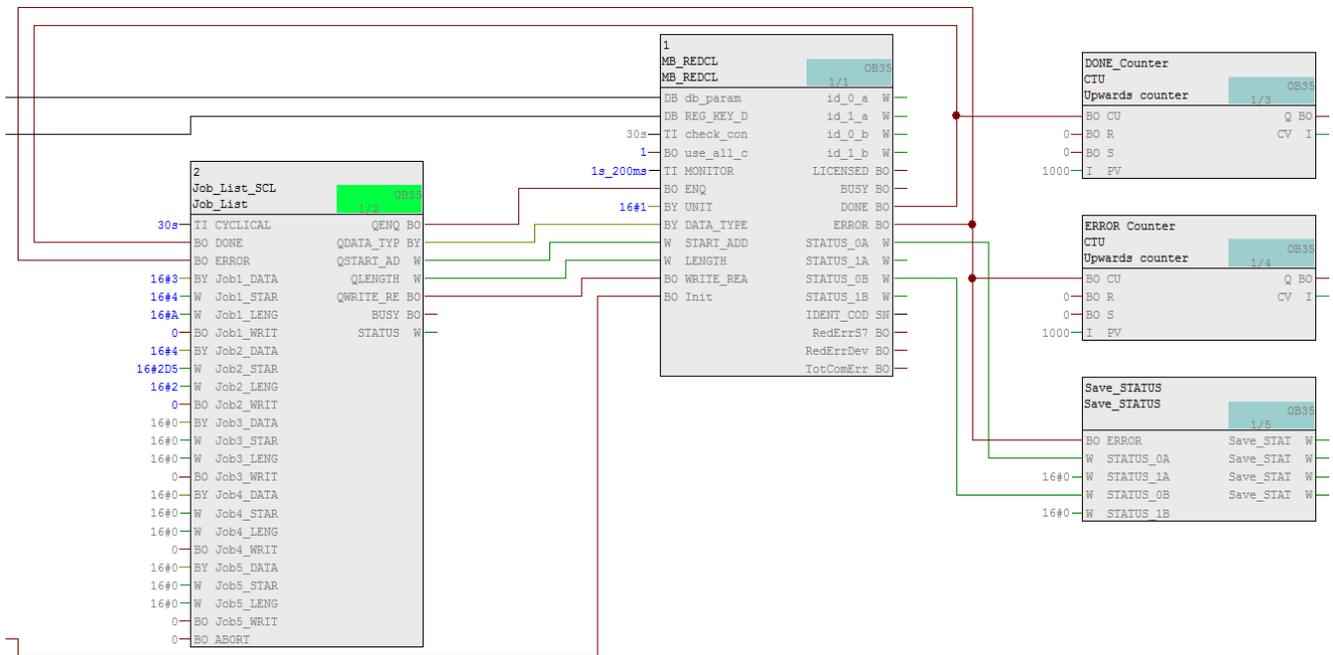
You can find the additional blocks and a detailed description on the Internet (<http://support.automation.siemens.com/WW/view/en/62830463>).

8.2 Job list for cyclic data exchange

General

With the Job_List block parameters can be set for a list of jobs that is worked through cyclically.

Application example

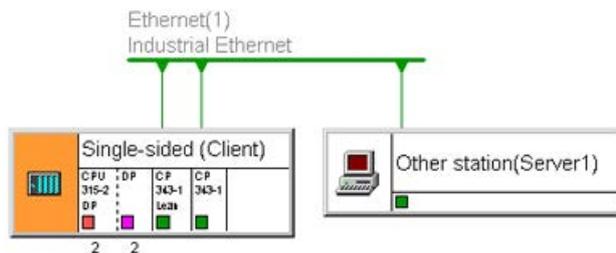


You can find the additional block and a detailed description on the Internet (<https://support.industry.siemens.com/cs/ww/en/view/62830463>).

Use in an S7-300 station

General

The "Modbus/TCP CP Redundant" package can also be operated in an S7-300 station with 2 CPs. The description of the functions and parameters in the preceding and following sections apply analogously for the use in an S7-300.



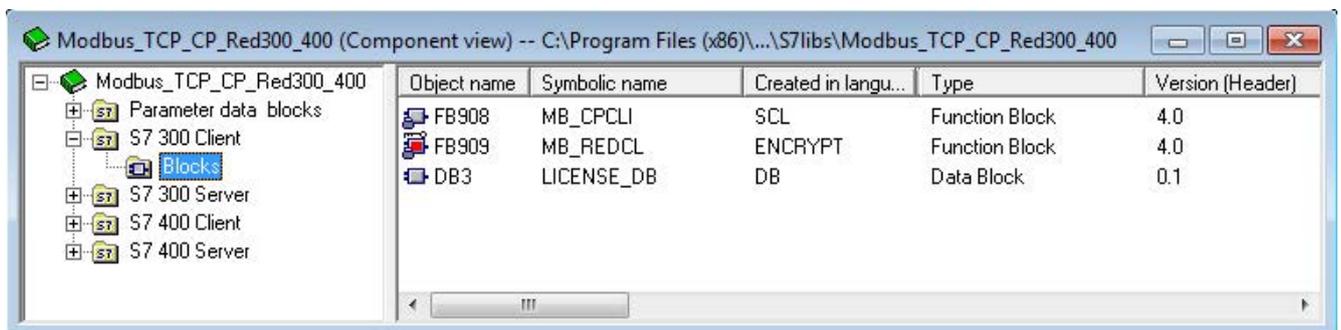
Usable modules for MB_REDCL and MB_REDSV

Only 300 CPUs can be used that provide enough local data per priority class (=> section 6.6 (Page 57) and section 7.7 (Page 74)).

With the AG_CNTRL block from the SIMATIC_NET library, it is possible to terminate and re-establish an existing connection. To be able to use the resources of the CPU/CP more effectively, this block was also included in the Modbus blocks. The older CPs or older firmware versions, however, do not support this AG_CNTRL. You will find information about which CP supports AG_CNTRL with which firmware version here: Ethernet CP and AG_CNTRL (<https://support.industry.siemens.com/cs/ww/en/view/33414377>).

Modbus blocks for S7-300

The installed library "Modbus_TCP_CP_Red300_400" contains the folders "S7 300 Client" and "S7 300 Server" with the blocks for the S7-300.



Copy the blocks to your project and assign parameters to the blocks as described in this manual. Use the blocks AG_CNTRL V1.0, AG_SEND V4.2 and AG_RECV V4.7 from the SIMATIC NET Library -> CP300.

Diagnostics

Diagnostics functions

The diagnostics functions of the CP 443 allow you to localize errors quickly. The following diagnostics options are available:

- Diagnostics using the display elements of the CP
- Diagnostics via the STATUS output of the MODBUS function blocks

Display elements (LEDs)

The display elements indicate the operating status or possible error states of the CP. The display elements give you an initial overview of any internal or external errors that have occurred as well as interface-specific errors.

STATUS output of MB_REDCL and MB_REDSV

For the error diagnostics, the MB_REDCL or MB_REDSV function blocks have STATUS outputs. By reading the STATUS outputs, you can obtain general information about errors that have occurred in the communication. You can evaluate the STATUS parameters in the user program.

10.1 Diagnostics using the display elements of the CP

Display functions

The display elements of the CP provide information about the module. The following display functions need to be distinguished:

Group error displays:

| | |
|------|-----------------|
| INTF | Internal errors |
| EXTF | External errors |

Special displays:

| | |
|-----------|---------------------------------------|
| CP 443-1: | |
| TXD | a frame is sent via the interface |
| RXD | a frame is received via the interface |

You will find a detailed description of the display elements in the manual of the CP.

10.2 Diagnostics messages of the FBs MB_REDCL and MB_REDSV

Messages at the STATUS outputs of the FB

Error messages are indicated at the status outputs of the FBs MB_REDCL and MB_REDSV. Below is a listing of the error messages of specific to the FBs.

Error messages from the called SFCs and FBs

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

You will find information on these error messages in the diagnostics buffer or the online help for the SFCs/FCs in the SIMATIC Manager and in the SIMATIC STEP 7 NCM S7 Industrial Ethernet manual.

Table 10- 1 Error messages from FB MB_REDCL or MB_REDSV

| Status (hex) | Event text | Remedy | |
|--------------|--|--|-----------|
| A001 | The parameter DB MODBUS_HPPARAM_CP has the wrong length. | Correct DB MODBUS_HPPARAM_CP. | |
| A002 | The end_x parameter is lower than start_x. | Correct the information at the start_x and end_x in the DB MODBUS_HPPARAM_CP. | |
| A003 | A DB to which Modbus addresses are to be mapped is too short. Minimum length in bytes: - with registers: $(end_x - start_x + 1) * 2 + 2$ - with bit values: $(end_x - start_x) / 8 + 1 + 2$ | Lengthen the DB. | |
| | CP is client: Incorrect call parameters. | CP is client: Correct the job parameter START_ADDRESS or LENGTH. | |
| | CP is server: Wrong address area in the request frame of the client; the client accesses an area that was not assigned in the S7 parameters. The CP replies with an exception frame. | CP is server: Change the client request or adapt the data areas in the DB MODBUS_HPPARAM_CP. | |
| A004 | Only CP is client: An invalid combination of DATA_TYPE and WRITE_READ was specified. | Correct the call parameters. Only data types 1 and 3 can be written. | |
| A005 | CP is client: An invalid value was specified in the LENGTH parameter. | CP is client: Correct the LENGTH parameter. CP is server: Change the number in the request. | |
| | CP is server: The number of registers/bits in the request is invalid. The CP replies with an exception telegram. | | |
| | Ranges of values: | | |
| | Read coils/inputs | | 1 to 2000 |
| | Write coils | | 1 to 1968 |
| A006 | Read registers | 1 to 125 | |
| | Write holding registers | 1 to 123 | |
| | The area specified with DATA_TYPE, START_ADDRESS and LENGTH does not exist in data_type_1 to data_type_8. | CP is client: Correct the parameter combination DATA_TYPE, START_ADDRESS and LENGTH. CP is server: Change the client request or correct the parameter assignment for data_type_x. | |
| A007 | CP is client: An invalid monitoring time was set for MONITOR. A value ≥ 20 ms must be entered. | Correct the parameter assignment. | |

| Status (hex) | Event text | Remedy |
|--------------|--|--|
| A008 | Within the set monitoring time MONITOR, the activated AG_RECV signals no reception, e.g. partner not ready. The connection is terminated and re-established. | Check the settings and if applicable error messages of the link partner. Check whether or not the link partner requires a specific UNIT identifier. |
| A009 | CP is client: The received transaction identifier TI is not the same as the one sent. The connection is terminated and re-established. | Record frames to check the data of the link partner. |
| A00A | CP is client: The received UNIT is not the same as the one sent. The connection is terminated and re-established. | Record frames to check the data of the link partner. |
| A00B | CP is client: The received function code is not the same as the one sent. CP is server: An invalid function code was received. The CP replies with an exception frame. | CP is client: Record frames to check the data of the link partner. CP is server: Change the client request. The FB MB_REDSV processes the function codes 1, 2, 3, 4, 5, 6, 15 and 16. |
| A00C | The received bytecount does not match the number of registers/bits. CP is server: The CP sends an exception frame. The connection will be terminated and established again. | Record frames to check the data of the link partner. |
| A00D | Only when CP is client: The register address/bit address or the number of registers/bits in the response is not the same as in the request. | |
| A00E | The length information in the Modbus-specific telegram header does not match the specified number of registers/bits or the bytecount in the request. The FB discards the data. The connection is terminated and re-established. | |
| A00F | A protocol identifier other than 0 was received. The connection will be terminated and re-established. | |
| A010 | A DB number was assigned twice in the parameters db_1 to db_8 . | Correct the parameter assignment for db_x. |
| A011 | An invalid value was specified for the DATA_TYPE input parameter (valid values are 1 - 4). | Correct the call parameters. |
| A012 | Configured data_type_1 and data_type_2 areas overlap. | Correct the parameter assignment, The data areas must not have a common register address area. |
| A013 | Configured data_type_1 and data_type_3 areas overlap. | |
| A014 | Configured data_type_1 and data_type_4 areas overlap. | |
| A015 | Configured data_type_1 and data_type_5 areas overlap. | |
| A016 | Configured data_type_1 and data_type_6 areas overlap. | |
| A017 | Configured data_type_1 and data_type_7 areas overlap. | |
| A018 | Configured data_type_1 and data_type_8 areas overlap. | |
| A019 | One of the db_x parameters was set to 0 even though the associated data_type_x is set to > 0. DB0 must not be used because this is reserved for the system. | Correct the parameter assignment for db_x to >0. If you use a data collector FB, check the parameter assignment in CFC. |
| A01A | Incorrect length in the header of the Modbus frame: 3 to 253 bytes are permitted. The connection is terminated and re-established. | Record frames to check the data of the link partner. |
| A01B | CP is server and function code 5: An invalid status was received for coil. An exception frame is sent. | Record frames to check the data of the link partner. |
| A01E | Invalid data was received that could not be assigned. The connection is terminated and re-established. | Check the error messages of the link partner and if necessary record the frames to check the data. |

| Status (hex) | Event text | Remedy |
|--------------|---|--|
| A01F | FB MB_REDCL or MB_REDSV is in an illegal operating status. | Contact Product Support. |
| A020 | No or too short a monitoring time for AG_CNTRL is configured with check_conn_cycle. | Correct the parameter assignment. A monitoring time > 1s must be configured. |
| A022 | For MB_REDSV a parameter data block for clients was set in the parameters or for MB_REDCL a parameter data block for servers was set in the parameters. | Correct the parameter assignment. |
| A023 | Configured data_type_2 and data_type_3 areas overlap. | Correct the parameter assignment. The data areas must not contain any overlapping register areas. |
| A024 | Configured data_type_2 and data_type_4 areas overlap. | |
| A025 | Configured data_type_2 and data_type_5 areas overlap. | |
| A026 | Configured data_type_2 and data_type_6 areas overlap. | |
| A027 | Configured data_type_2 and data_type_7 areas overlap. | |
| A028 | Configured data_type_2 and data_type_8 areas overlap. | |
| A034 | Configured data_type_3 and data_type_4 areas overlap. | |
| A035 | Configured data_type_3 and data_type_5 areas overlap. | |
| A036 | Configured data_type_3 and data_type_6 areas overlap. | |
| A037 | Configured data_type_3 and data_type_7 areas overlap. | |
| A038 | Configured data_type_3 and data_type_8 areas overlap. | |
| A045 | Configured data_type_4 and data_type_5 areas overlap. | |
| A046 | Configured data_type_4 and data_type_6 areas overlap. | |
| A047 | Configured data_type_4 and data_type_7 areas overlap. | |
| A048 | Configured data_type_4 and data_type_8 areas overlap. | |
| A056 | Configured data_type_5 and data_type_6 areas overlap. | |
| A057 | Configured data_type_5 and data_type_7 areas overlap. | |
| A058 | Configured data_type_5 and data_type_8 areas overlap. | |
| A067 | Configured data_type_6 and data_type_7 areas overlap. | |
| A068 | Configured data_type_6 and data_type_8 areas overlap. | |
| A078 | Configured data_type_7 and data_type_8 areas overlap. | |
| A07A | An invalid value was specified for the id_x parameter (value range from 1 to 64). | Correct the parameter assignment in the DB MODBUS_HPARAM_CP. |
| A07B | The specified id_x is defined twice in the parameter DB. | |
| A07C | An invalid value was specified for the data_type_x parameter (valid values are 0 to 4). | |
| A07D | The data_type_1 parameter does not contain an entry. The parameter area "_1" is the initial area and must be set. | |
| A07E | The number of the instance DB of the MB_REDCL or MB_REDSV block or the parameter DB was specified in db_x. | |
| A07F | The DB specified at db_param is not a Modbus parameter DB. | Correct the parameter assignment at the db_param input. |
| A080 | The Modbus block has not yet been initialized. | The Modbus block must be initialized with Init = TRUE after the transfer of the IDB to the CPU. |
| A081 | Only for CP is client and function code 5: The data of the response is not the echo of the request. | Record frames to check the data of the link partner. |

| Status (hex) | Event text | Remedy |
|--------------|---|---|
| A082 | Only for CP is client and function code 6: The received register value is not the same as the one sent. | Record frames to check the data of the link partner. |
| A083 | CP is client: A job was triggered while the previous job is still in progress. The job will not be executed. This is status information. The ERROR bit is not set. There was an attempt to initialize the block while a job was still running. CP is server: There was an attempt to initialize the block while ENR = TRUE was set. | CP is client: Only start a new job when the previous job ended with DONE = TRUE, NDR = TRUE or ERROR = TRUE. Wait with the initialization until no job is running any more. CP is server: Set ENR = FALSE. |
| A085 | Due to an illegal write access an internal error has occurred during the license check. | Check that there is no illegal write access to the license DB in the S7 program. The structure of REG_KEY must not be modified. If necessary, contact Product Support. |
| A086 | An attempt was made to write to a write-protected data block. | Remove the write protection of the data block or use a different DB. |
| A090 | The MB_REDCL or MB_REDSV block has not yet been licensed for this CPU. This is status information. The ERROR bit is not set. MODBUS communication is running even without a license. | Read the IDENT_CODE identification string for this CPU and use it to request the registration key. See "Licensing by reading out the IDENT_CODES (Page 31)". |
| A091 | Only when CP is client: An exception telegram with exception code 1 was received as the reply. | The link partner does not support the requested function. |
| A092 | Only when CP is client: An exception telegram with exception code 2 was received as the reply. There was access to a non-existent or illegal address on the link partner. | Correct LENGTH or START_ADDRESS in the FB call. |
| A093 | Only when CP is client: An exception telegram with exception code 3 was received as the reply. | The link partner cannot process the received request (for example, it does not support the requested length). |
| A094 | Only when CP is client: An exception telegram with exception code 4 was received as the reply. | The link partner is in a status in which it cannot process the request. |
| A095 | Only when CP is client: An exception telegram with an unknown exception code was received as the reply. | Check the error messages of the link partner and if necessary record the frames to check the data. |
| A0FF | The connection is currently not ready for communication. This is a status message and the result of a previous connection error. | Check which error occurred before and eliminate the cause. Check the connections. If applicable correct the value at check_conn_cycle. The error can also occur when a CP is used that does not support AG_CNTRL. |
| FFFF | The connection is not configured. | If this connection is to be used, It must be configured in the parameter DB MODBUS_HPARAM_CP. |

10.3 Diagnostics messages of the linked in blocks

Diagnostics message

Table 10- 2 Error messages of the linked in FCs/SFCs

| STATUS (hex) | Event text | Remedy |
|--------------|--|--|
| 7xxx | You will find more detailed information in the online help of the SIMATIC Manager. | Refer to the online help (SIMATIC Manager > select block > F1 key > Ethernet > see also > Condition codes) |
| 8xxx | You will find more detailed information in the online help of the SIMATIC Manager. | Refer to the online help (SIMATIC Manager > select block > F1 key > Ethernet > see also > Condition codes) |
| 80B2 | K bus connection between CPU and CP not established. | This error message can occur once after restarting the H system and can be ignored. |
| 8186 | Parameter ID is invalid | Correct the parameter assignment. Use the ID from NetPro. |

10.4 Diagnostics messages of SFC24

Diagnostics message

Table 10- 3 Error messages of SFC24

| STATUS (hex) | Event text | Remedy |
|--------------|--|---|
| 80A1 | DB number = 0 or too high for the CPU. | Select a permitted DB number. |
| 80B1 | The DB does not exist on the CPU. | All data blocks specified in db_x must be created and transferred to the CPU. |
| 80B2 | DB UNLINKED | Do not generate DB as UNLINKED. |

10.5 Diagnostics messages via alarm bits

The Modbus blocks provide the option of detecting a loss of redundancy. This is displayed via the outputs RedErrS7, RedErrDev and TotComErr. These status bits can be interconnected with an alarm block or another block and evaluated there.

10.5.1 Client block

Depending on the parameter assignment, the alarm bits are set as follows:

1. use_all_conn = FALSE

The requests are sent and received via 1 connection, the other configured connections are on standby. There is a cyclic check of all configured connections after the set time "check_conn_cycle".

| Number of faulty connections | STATUS_0A | STATUS_0B | STATUS_1A | STATUS_1B | RedErrS7 | RedErrDev | TotComErr |
|------------------------------|-----------|-----------|-----------|-----------|----------|-----------|-----------|
| 0 | 0 | 0 | 0 | 0 | FALSE | FALSE | FALSE |
| 1 | 0 | 0 | 0 | <> 0 | FALSE | FALSE | FALSE |
| | 0 | 0 | <> 0 | 0 | FALSE | FALSE | FALSE |
| | 0 | <> 0 | 0 | 0 | FALSE | FALSE | FALSE |
| | <> 0 | 0 | 0 | 0 | FALSE | FALSE | FALSE |
| 2 | 0 | 0 | <> 0 | <> 0 | TRUE | FALSE | FALSE |
| | 0 | <> 0 | 0 | <> 0 | FALSE | TRUE | FALSE |
| | <> 0 | 0 | 0 | <> 0 | FALSE | FALSE | FALSE |
| | 0 | <> 0 | <> 0 | 0 | FALSE | FALSE | FALSE |
| | <> 0 | 0 | <> 0 | 0 | FALSE | TRUE | FALSE |
| | <> 0 | <> 0 | 0 | 0 | TRUE | FALSE | FALSE |
| 3 | <> 0 | <> 0 | <> 0 | 0 | TRUE | TRUE | FALSE |
| | <> 0 | <> 0 | 0 | <> 0 | TRUE | TRUE | FALSE |
| | <> 0 | 0 | <> 0 | <> 0 | TRUE | TRUE | FALSE |
| | 0 | <> 0 | <> 0 | <> 0 | TRUE | TRUE | FALSE |
| 4 | <> 0 | <> 0 | <> 0 | <> 0 | TRUE | TRUE | TRUE |

2. use_all_conn = TRUE, 2 connections are configured

The frames are sent and received via 2 configured connections. When the set time "check_conn_cycle" elapses, there is a cyclic check of the connections.

| Number of faulty connections | STATUS_0A | STATUS_0B | STATUS_1A | STATUS_1B | RedErrS7 | RedErrDev | TotComErr |
|------------------------------|-----------|-----------|-----------|-----------|----------|-----------|-----------|
| 0 | 0 | FFFF | 0 | FFFF | FALSE | FALSE | FALSE |
| 1 | 0 | FFFF | <> 0 | FFFF | TRUE | TRUE | FALSE |
| | <> 0 | FFFF | 0 | FFFF | TRUE | TRUE | FALSE |
| 2 | <> 0 | FFFF | <> 0 | FFFF | TRUE | TRUE | TRUE |

3. use_all_conn = TRUE, 4 connections are configured

The frames are sent and received via 4 configured connections. When the set time "check_conn_cycle" elapses, there is a cyclic check of the connections.

| Number of faulty connections | STATUS_0A | STATUS_0B | STATUS_1A | STATUS_1B | RedErrS7 | RedErrDev | TotComErr |
|------------------------------|-----------|-----------|-----------|-----------|----------|-----------|-----------|
| 0 | 0 | 0 | 0 | 0 | FALSE | FALSE | FALSE |
| 1 | 0 | 0 | 0 | <> 0 | FALSE | FALSE | FALSE |
| | 0 | 0 | <> 0 | 0 | FALSE | FALSE | FALSE |
| | 0 | <> 0 | 0 | 0 | FALSE | FALSE | FALSE |
| | <> 0 | 0 | 0 | 0 | FALSE | FALSE | FALSE |
| 2 | 0 | 0 | <> 0 | <> 0 | TRUE | FALSE | FALSE |
| | 0 | <> 0 | 0 | <> 0 | FALSE | TRUE | FALSE |
| | <> 0 | 0 | 0 | <> 0 | FALSE | FALSE | FALSE |
| | 0 | <> 0 | <> 0 | 0 | FALSE | FALSE | FALSE |
| | <> 0 | 0 | <> 0 | 0 | FALSE | TRUE | FALSE |
| <> 0 | <> 0 | 0 | 0 | TRUE | FALSE | FALSE | |
| 3 | <> 0 | <> 0 | <> 0 | 0 | TRUE | TRUE | FALSE |
| | <> 0 | <> 0 | 0 | <> 0 | TRUE | TRUE | FALSE |
| | <> 0 | 0 | <> 0 | <> 0 | TRUE | TRUE | FALSE |
| | 0 | <> 0 | <> 0 | <> 0 | TRUE | TRUE | FALSE |
| 4 | <> 0 | <> 0 | <> 0 | <> 0 | TRUE | TRUE | TRUE |

10.5.2 Server block

Data reception is monitored on all configured connections.

When the set time "check_conn_cycle" elapses, there is a cyclic check of the connections.

| Number of faulty connections | STATUS_0A | STATUS_0B | STATUS_1A | STATUS_1B | RedErrS7 | RedErrDev | TotComErr |
|------------------------------|-----------|-----------|-----------|-----------|----------|-----------|-----------|
| 0 | 0 | 0 | 0 | 0 | FALSE | FALSE | FALSE |
| 1 | 0 | 0 | 0 | <> 0 | FALSE | FALSE | FALSE |
| | 0 | 0 | <> 0 | 0 | FALSE | FALSE | FALSE |
| | 0 | <> 0 | 0 | 0 | FALSE | FALSE | FALSE |
| | <> 0 | 0 | 0 | 0 | FALSE | FALSE | FALSE |
| 2 | 0 | 0 | <> 0 | <> 0 | TRUE | FALSE | FALSE |
| | 0 | <> 0 | 0 | <> 0 | FALSE | TRUE | FALSE |
| | <> 0 | 0 | 0 | <> 0 | FALSE | FALSE | FALSE |
| | 0 | <> 0 | <> 0 | 0 | FALSE | FALSE | FALSE |
| | <> 0 | 0 | <> 0 | 0 | FALSE | TRUE | FALSE |
| <> 0 | <> 0 | 0 | 0 | TRUE | FALSE | FALSE | |
| 3 | <> 0 | <> 0 | <> 0 | 0 | TRUE | TRUE | FALSE |
| | <> 0 | <> 0 | 0 | <> 0 | TRUE | TRUE | FALSE |
| | <> 0 | 0 | <> 0 | <> 0 | TRUE | TRUE | FALSE |
| | 0 | <> 0 | <> 0 | <> 0 | TRUE | TRUE | FALSE |
| 4 | <> 0 | <> 0 | <> 0 | <> 0 | TRUE | TRUE | TRUE |

Examples of applications

General

With the installation, 2 sample projects are stored in \Program Files\Siemens\Step 7\Examples:

- 1 sample project in STL "MB_TCP_CP_RED_400" and
- 1 sample project in CFC "MB_TCP_CP_RED_CFC".

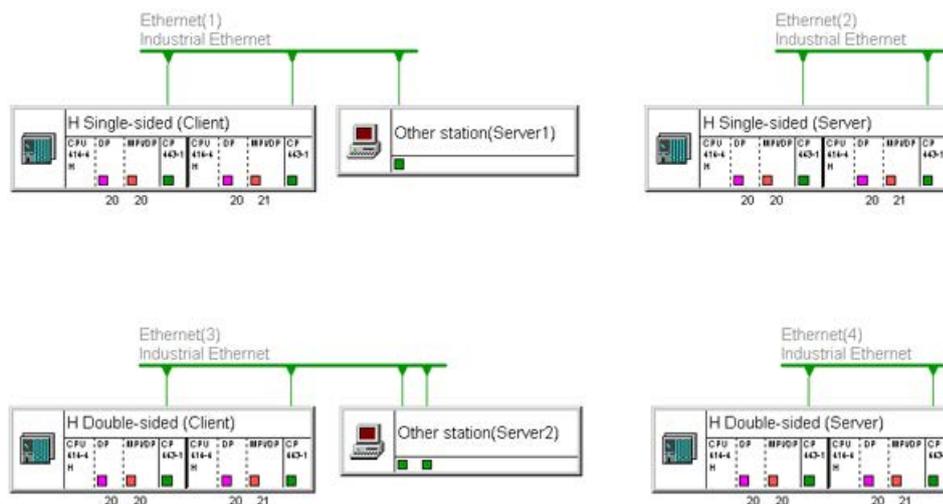
Note

The S7 program is intended as a source of information and should not be considered as a binding solution proposal for a customer-specific plant configuration.

Sample project on the CD

On the CD or ISO image, you can find an extensive example project in which SIMATIC stations have been created for all function variants.

- S7 H station is client or server
- Single-sided or double-sided redundancy



SIMATIC stations in the sample project

The following SIMATIC stations have been created in the sample project:

| Block / station name | Single-sided - S7 is redundant | Single-sided - device is redundant | Double-sided | Client | Server |
|--------------------------|--------------------------------|------------------------------------|--------------|--------|--------|
| S7 client - double-sided | | | x | x | |
| S7 server double-sided | | | x | | x |
| S7 client - dev single | x | | | x | |
| S7 server dev single | x | | | | x |
| S7 Client - S7 Single | | x | | x | |
| S7 Server - S7 Single | | x | | | x |

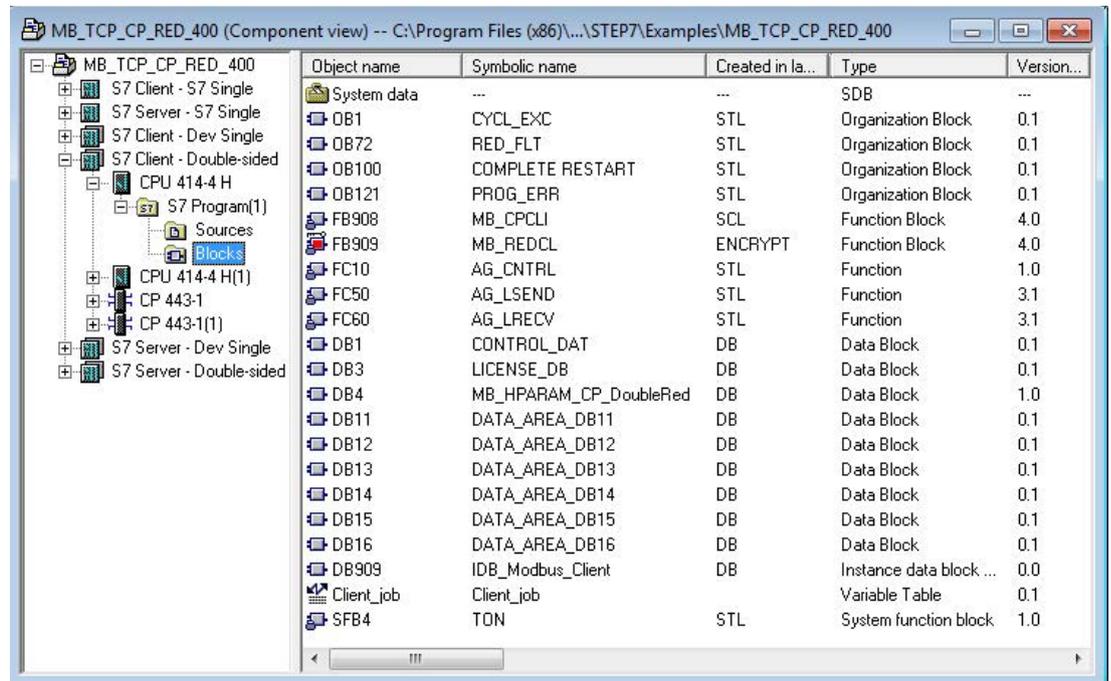
Program example

The program examples consist of the following blocks:

- Startup block OB100 with setting of the Init bit
- Programming error OB121
- Cyclic operation OB1 or OB35 with an FB909 or FB907 call
- Global data blocks for starting a job (e.g. with the aid of a variables table) and for licensing
- Data blocks for register and bit values

11.1 Sample project in STL - Modbus client

Overview



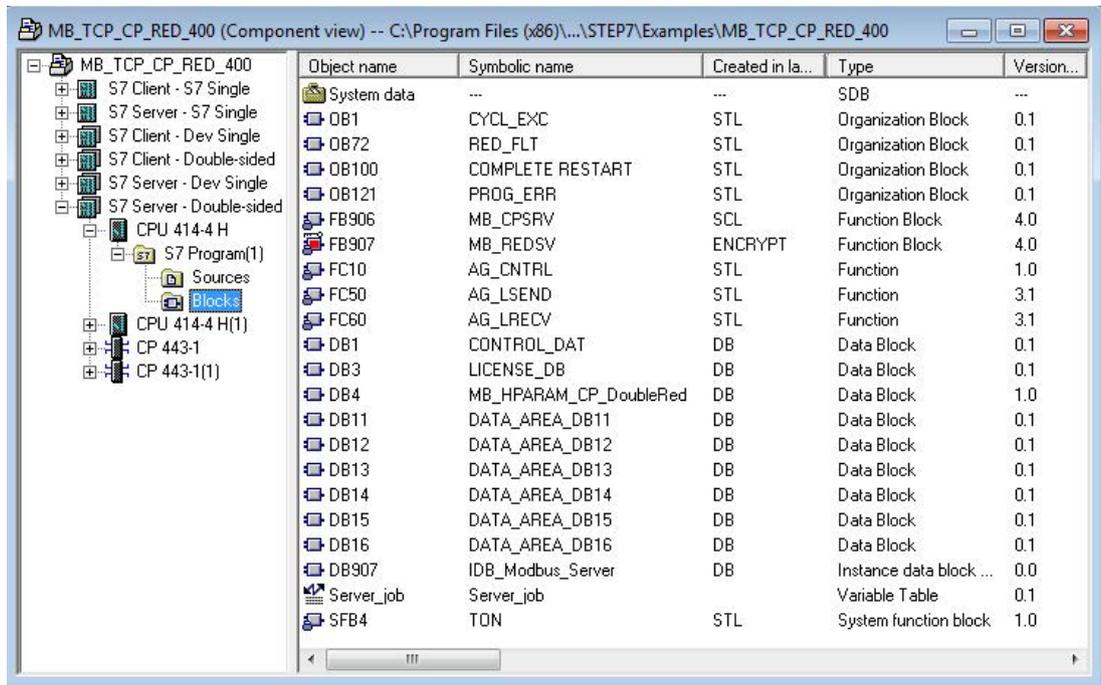
Blocks used

These block numbers are also used in the supplied sample project for S7 H stations with FB MB_REDCL.

| Block | Symbol | Comment |
|--------|------------------|---|
| OB 1 | CYCL_EXC | Cyclic program execution |
| OB 100 | COMPLETE RESTART | Startup OB for cold restart |
| OB 121 | PROG_ERR | Programming error OB |
| FB 908 | MB_CPCLI | FB MB_CPCLI called internally |
| FB 909 | MB_REDCL | User block FB MB_REDCL |
| FC10 | AG_CNTRL | FC for connection handling |
| FC50 | AG_LSEND | FC for sending |
| FC60 | AG_LRECV | FC for receiving |
| DB 1 | CONTROL_DAT | Work DB CONTROL DAT for FB MB_REDCL |
| DB 3 | LICENSE_DB | License DB for FB MB_REDCL |
| DB 4 | MODBUS_HPARAM_CP | Parameter DB for client (double-sided redundancy) |
| DB 11 | DATA_AREA_DB11 | Values DB for area 1 |
| DB 12 | DATA_AREA_DB12 | Values DB for area 2 |
| DB 13 | DATA_AREA_DB13 | Values DB for area 3 |
| DB 14 | DATA_AREA_DB14 | Values DB for area 5 |
| DB 15 | DATA_AREA_DB15 | Values DB for area 6 |
| DB 16 | DATA_AREA_DB16 | Values DB for area 7 |
| DB 909 | IDB_MODBUS | Instance DB for FB MB_REDCL |

11.2 Sample project in STL - Modbus server

Overview



Blocks used

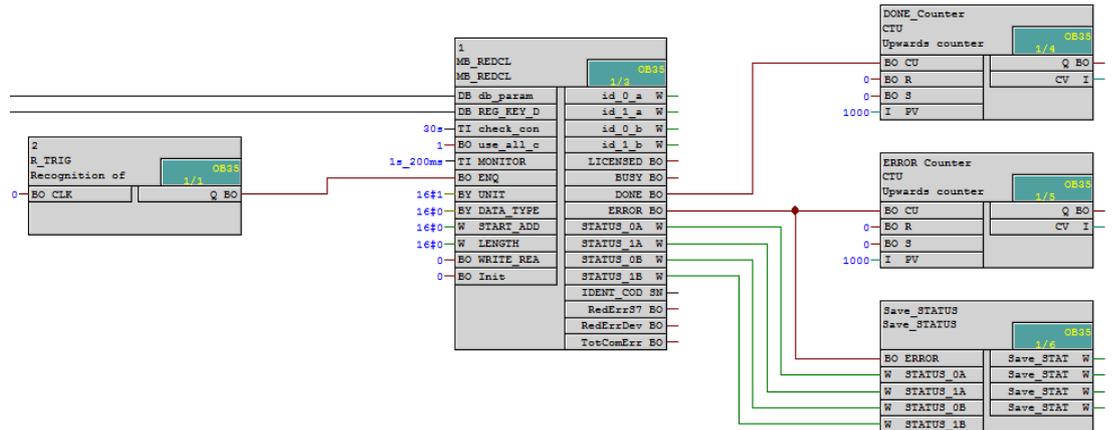
These block numbers are also used in the supplied sample project for S7 H stations with FB MB_REDSV.

| Block | Symbol | Comment |
|--------|--------------------|---|
| OB 1 | CYCL_EXC | Cyclic program execution |
| OB 100 | COMPLETE RESTART | Startup OB for cold restart |
| OB 121 | PROG_ERR | Programming error OB |
| FB 906 | MB_CPSRV | FB MB_CPSRV called internally |
| FB 907 | MB_REDSV | User block FB MB_REDSV |
| FC10 | AG_CNTRL | FC for connection handling |
| FC50 | AG_LSEND | FC for sending |
| FC60 | AG_LRECV | FC for receiving |
| DB 1 | CONTROL_DAT | Work DB CONTROL DAT for FB MB_REDSV |
| DB 3 | LICENSE_DB | License DB for FB MB_REDSV |
| DB 4 | MODBUS_HPAPARAM_CP | Parameter DB for server (double-sided redundancy) |
| DB 11 | DATA_AREA_DB11 | Values DB for area 1 |
| DB 12 | DATA_AREA_DB12 | Values DB for area 2 |
| DB 13 | DATA_AREA_DB13 | Values DB for area 3 |
| DB 14 | DATA_AREA_DB14 | Values DB for area 5 |
| DB 15 | DATA_AREA_DB15 | Values DB for area 6 |
| DB 16 | DATA_AREA_DB16 | Values DB for area 7 |
| DB 907 | IDB_MODBUS | Instance DB for FB MB_REDSV |

11.3 Sample project in CFC - Modbus client

Overview

The sample project was created with CFC V8.0 Update 1.



Blocks used

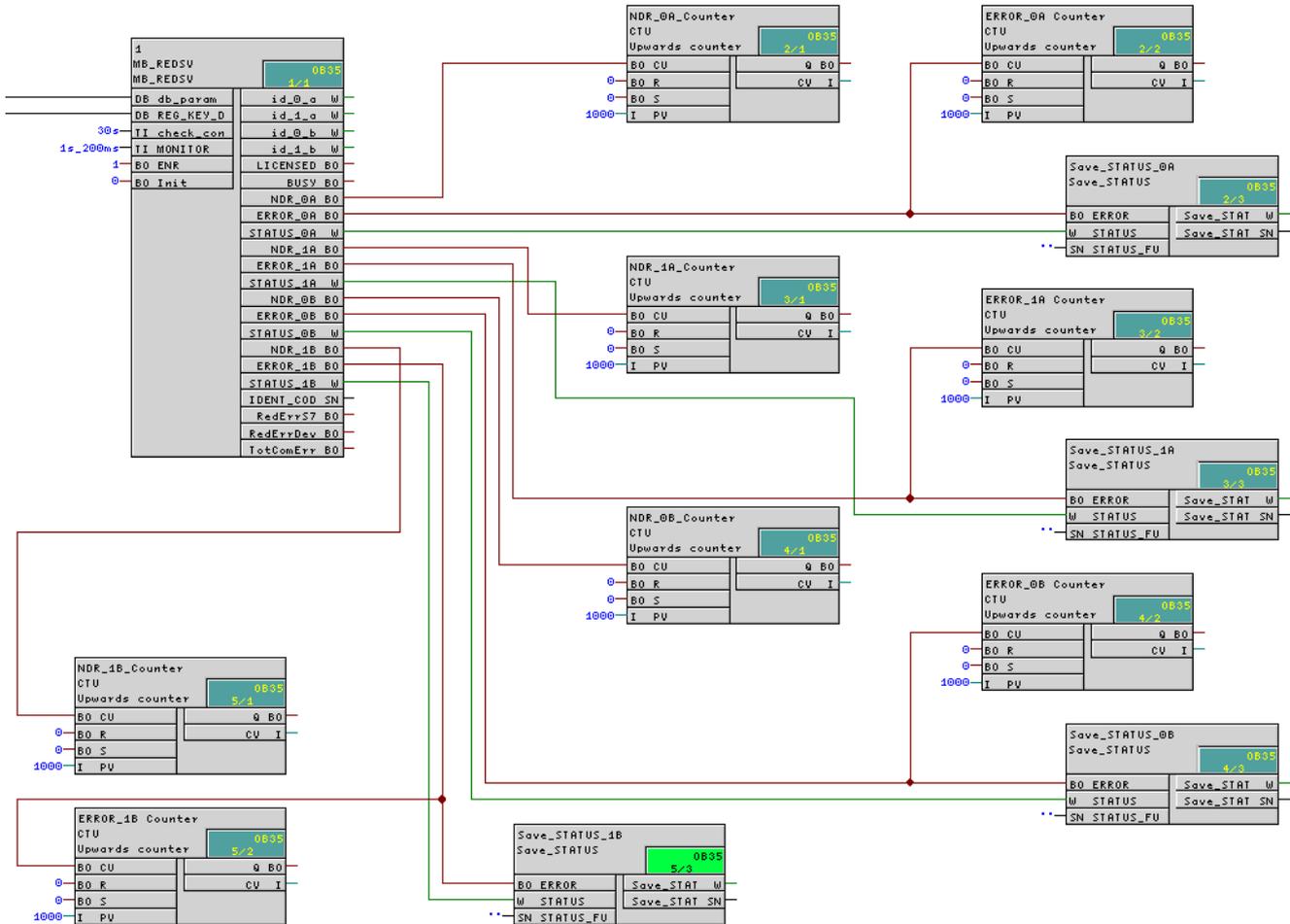
These block numbers are also used in the supplied sample project for S7 H stations with FB MB_REDCL.

| Block | Symbol | Comment |
|--------|------------------|---|
| OB 35 | CYCL_EXC | Cyclic program execution |
| OB 100 | COMPLETE_RESTART | Startup OB for cold restart |
| OB 121 | PROG_ERR | Programming error OB |
| FB 908 | MB_CPCLI | FB MB_CPCLI called internally |
| FB 909 | MB_REDCL | User block FB MB_REDCL |
| FC100 | AG_CNTRL | FC for connection handling |
| FC500 | AG_LSEND | FC for sending |
| FC600 | AG_LRECV | FC for receiving |
| DB 4 | MODBUS_HPARAM_CP | Parameter DB for client (double-sided redundancy) |
| DB 11 | DATA_AREA_DB11 | Values DB for area 1 |
| DB 12 | DATA_AREA_DB12 | Values DB for area 2 |
| DB 13 | DATA_AREA_DB13 | Values DB for area 3 |
| DB 14 | DATA_AREA_DB14 | Values DB for area 5 |
| DB 15 | DATA_AREA_DB15 | Values DB for area 6 |
| DB 16 | DATA_AREA_DB16 | Values DB for area 7 |
| DB xy | | DBs generated by CFC |

11.4 Sample project in CFC - Modbus server

Overview

The sample project was created with CFC V8.0 Update 1.



Blocks used

These block numbers are also used in the supplied sample project for S7 H stations with FB MB_REDSV.

| Block | Symbol | Comment |
|--------|------------------|---|
| OB 35 | CYCL_EXC | Cyclic program execution |
| OB 100 | COMPLETE RESTART | Startup OB for cold restart |
| OB 121 | PROG_ERR | Programming error OB |
| FB 906 | MB_CPSRV | FB MB_CPSRV called internally |
| FB 907 | MB_REDSV | User block FB MB_REDSV |
| FC100 | AG_CNTRL | FC for connection handling |
| FC500 | AG_LSEND | FC for sending |
| FC600 | AG_LRECV | FC for receiving |
| DB 4 | MODBUS_HPARAM_CP | Parameter DB for server (double-sided redundancy) |
| DB 11 | DATA_AREA_DB11 | Values DB for area 1 |
| DB 12 | DATA_AREA_DB12 | Values DB for area 2 |
| DB 13 | DATA_AREA_DB13 | Values DB for area 3 |
| DB 14 | DATA_AREA_DB14 | Values DB for area 5 |
| DB 15 | DATA_AREA_DB15 | Values DB for area 6 |
| DB 16 | DATA_AREA_DB16 | Values DB for area 7 |
| DB xy | | DBs generated by CFC |

References

The MODBUS Organization

MODBUS APPLICATION PROTOCOL SPECIFICATION
V1.1b, December 28, 2006

Modbus home page (<http://www.modbus.org>)

Siemens AG
Digital Factory Division
Postfach 48 48
90026 NÜRNBERG
GERMANY

Subject to change
A5E36465068-AC
© Siemens AG 2018

www.siemens.com/automation