

SIEMENS

SIMATIC

SIMATIC Modbus/TCP Redundant Communication via CP443-1 in H-Systems

Manual



SIMATIC S7

**SIMATIC Modbus/TCP
Redundant Communication
via CP443-1 in H-Systems**

Manual

Preface, Table of Contents

Product Description	1
Getting Started	2
Commissioning	3
Licensing	4
FB MB_REDCL	5
FB MB_REDSV	6
Add-On for CFC	7
Usage in S7-300	8
Diagnostics	9
Application Sample	10
Appendices	
Literature	
Glossary	

Safety Precautions and Warnings

This manual contains warnings, which you should note for your own safety as well as for the prevention of damage to property. These warnings are indicated by means of a triangle and displayed as follows in accordance with the level of danger:



Danger

indicates that loss of life, severe personal injury or substantial damage **will** result if proper precautions are not taken.



Warning

indicates that loss of life, severe personal injury or substantial damage **can** result if proper precautions are not taken.



Caution

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

Notes

call attention to information that is especially significant to the product, handling of the product or a specific part of this documentation.

Qualified Personnel

The equipment may be commissioned and put into operation by **qualified personnel** only. For the purpose of safety relevant warnings of this manual a qualified person is one who is authorized to commission, ground and tag devices, systems and circuits.

Use as prescribed

Please note the following:



Warning

This equipment must only be used in applications as prescribed in the catalogue and the technical description and in conjunction with equipment and components recommended and authorized by Siemens.

Successful and safe operation of this equipment is dependent upon proper transport, and storage, erection and installation as well as careful operation and maintenance.

SIMATIC[®] and SIMATIC NET[®] are registered trademarks of SIEMENS AG.

Trademarks

The other brand names in this manual may be trademarks use of which by third parties for their purposes may infringe the proprietors' rights.

Copyright © Siemens AG 2008-20012 All Rights Reserved

The reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders will be liable for damages. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

Siemens AG
Industry Sector
Industry Automation Division / Industrial Automation Systems
Factory Automation
I IA AS FA WF FTH 1
Postfach 23 55, D- 90713 Fürth

Exclusion from Liability

We have checked the contents of this document with regard to conformity to the described hardware and software. Deviations, however, cannot be excluded; therefore we cannot accept prejudice for its complete conformity. The information in this document is checked regularly and necessary corrections are contained in subsequent issues. Any suggestions for improvement are gratefully received.

We reserve the right to make technical changes.

Preface

Purpose of the Manual

The information in this manual allows you to set up and put in operation the connection between the CP 443-1 in an S7 redundant system and a device that supports the Open MODBUS/TCP protocol.

Contents of the Manual

This manual describes the function of the Modbus function block and their integration into the hardware and software of the communication processors CP 443-1.

The manual contains the following topics:

- Production description
- Getting Started
- Commissioning / Installation / Parameterization
- Licensing
- Function blocks
- Diagnostics
- Application sample

Scope of this Manual

This manual is valid for the following software:

Product	Identification number	From version
MODBUS/TCP Redundant	2XV9 450-1MB11	2.1.1
FB 909 „MB_REDCL“		2.4
FB 908 „MB_CPCLI“		2.3
FB 907 „MB_REDSV“		2.3
FB 906 „MB_CPSRV“		2.2

Note

This manual contains the FB description, as it is valid at the time of publication.

How to Access the Information in this Manual

To enable you a quick access to selected information, the manual provides the following access aids:

- The next pages contain a complete table of contents.

Additional Sources of Information	<p>All additional information concerning CP 443 (mounting, commissioning etc.) can be found in the manual</p> <p>SIEMENS SIMATIC Fault-Tolerant Systems S7-400H System Manual A5E00267693-03</p> <p>SIEMENS SIMATIC NET S7-CPs for Industrial Ethernet device manual C79000-G8900-C155</p> <p>SIEMENS SIMATIC NET S7-CPs for Industrial Ethernet device manual part B4 CP 443-1 C79000-G8900-C152</p> <p>SIEMENS SIMATIC NET NCM S7 for Industrial Ethernet manual C79000-G8900-C129</p>
	<p>Additional information concerning STEP7 can be found in the following manuals:</p>
	<p>SIEMENS SIMATIC Software Base software for S7 and M7 STEP7 user manual C79000-G7000-C502-..</p>
	<p>SIEMENS SIMATIC Software System software for S7-300/400 System and standard functions Reference manual C79000-G7000-C503-02</p>
Additional Questions	<p>If you have further questions regarding the use of the FBs described in this manual, which are not answered in this document, please contact your Siemens partner who supplied you with this function block.</p>
Terminology	<p>This document uses the term CP or CP 443. The descriptions only apply to communications processor CP 443-1.</p>
Scope of Application	<p>The function block described in this manual establishes a connection between the CP 443-1 and third party MODBUS devices.</p>

Table of Contents

1	Product Description.....	1-1
1.1	Field of Applications.....	1-1
1.2	Hardware and Software Prerequisites.....	1-2
2	Getting Started.....	2-1
3	Commissioning.....	3-1
3.1	Installing the Library on the STEP7 PG/-PC	3-1
3.2	Parameterization of the CP	3-2
3.3	Network Configuration.....	3-4
3.4	Insertion of the Function Blocks into the Program.....	3-9
3.5	Multiple Connections via Port 502	3-11
3.6	Start_up Characteristics of CP443.....	3-12
4	Licensing.....	4-1
5	Function Block MB_REDCL – Modbus Client	5-1
5.1	Configuration of the Redundant Communication	5-1
5.2	Functionality of FB MB_REDCL	5-4
5.3	Connection Testing by means of AG_CNTRL.....	5-10
5.4	Parameters of the Function Block MB_REDCL.....	5-11
5.5	Example for Address Mapping.....	5-19
5.6	Data and Standard Function used by the FB	5-21
5.7	Renaming / Rewiring of Standard Functions and Function Blocks	5-23
6	Function Block MB_REDSV – Modbus Server	6-1
6.1	Configuration of the Redundant Communication	6-1
6.2	Functionality of the FB MB_REDSV	6-4
6.3	Connection Testing by means of AG_CNTRL.....	6-7
6.4	Parameters of the Function Block MB_REDSV	6-8

6.5	Example for Address Mapping.....	6-14
6.6	Data and Standard Function used by the FB	6-16
6.7	Renaming / Rewiring of Standard Functions and Function Blocks	6-18
7	Additional Blocks.....	7-1
7.1	Add-On for CFC	7-1
7.2	Job List for cyclical telegram transfer	7-2
8	Application with S7-300.....	8-1
9	Diagnostics	9-1
9.1	Diagnostics via the Display Elements of the CP	9-1
9.2	Diagnostic Messages of the FBs MB_REDCL and MB_REDSV.....	9-2
9.3	Diagnostic Messages of included FCs/SFCs.....	9-6
9.4	Diagnostic Messages of SFC24.....	9-6
9.5	Diagnostics with Alarm Bits	9-7
9.5.1	Client Block.....	9-7
9.5.2	Server Block	9-9
10	Application Sample.....	10-1
10.1	Example project in AWL – Modbus Client	10-2
10.2	Example project in AWL – Modbus Server	10-3
10.3	Example project in CFC – Modbus Client.....	10-4
10.4	Example project in CFC – Modbus Server.....	10-5
A	Literature	1

1 Product Description

1.1 Field of Applications

Placement in the System Environment

The driver described here is a software product for the Communications Processor CP443-1 in a SIMATIC S7 redundant system.

CP 443-1 can be used in the SIMATIC S7-400 automation systems and can establish communication links to partner systems.

Function of the FBs

These function blocks enable a communication link between CP 443-1 and a device that supports the Open MODBUS/TCP protocol. The function codes 1, 2, 3, 4, 5, 6, 15 and 16 are provided.

Data transmission is carried out following the Client-Server principle.

The SIMATIC S7 can act as both client and server during the data transmission.

Redundant communication is supported. A S7-400H system as well as a S7-400 single CPU with 2 CPs can be used.

Hot standby means the parallel redundant processing of signals in redundant components. This allows a bumpless failover of the entire system to the standby components.

TCP/IP with CP443-1

TCP/IP with CP443-1 uses static connections. The TCP connection is not disconnected during operation.

Network configuration of STEP7 enables only a **unique use of a specific port number**, when using TCP native stack of the CP.

However, with specific CP modules it is possible to use multiple connection via port 502 to different clients simultaneously.

In section 3.6 "Multiple connections via port 502" you can find technical details regarding this matter.

1.2 Hardware and Software Prerequisites

Usable Modules for MB_REDCL and MB_REDSV

The block AG_CNTRL of the SIMATIC_NET library permits to terminate and reestablish an established connection. This block was also implemented in the Modbus blocks for a more effective use of the resources of CPU and CP. However, previous CPs or previous firmware releases do not support the use of AG_CNTRL.

Here you can find up-to-date information which CPs and which firmware releases support AG_CNTRL: [Ethernet CPs and AG_CNTRL](#).

You can find further hardware prerequisites on the internet:
www.siemens.com/s7modbus

Software Versions

The usage of the FB MB_REDCL and MB_REDSV is possible with **STEP7 Version 5.4** or higher. Withal the use of the blocks AG_LSEND/AG_LRCV V3.1 of the update of SIMATIC NET library (<http://support.automation.siemens.com/WW/view/de/22172239>) is required.

Memory requirements

The FB MB_REDCL requires 14 kByte work memory and 17 kByte load memory.

The FB MB_CPCLI requires 10 kByte work memory and load memory.

The FB MB_REDSV requires 12 kByte work memory and 14 kByte load memory.

The FB MB_CPSRV requires 10 kByte work memory and load memory.

You can find precise length information of the blocks in their properties in SIMATIC Manager.

2 Getting Started

Procedure

1. Install "SIMATIC Modbus/TCP Redundant".
=> Section 3.1
2. Parameterize the connection parameters regarding your requirements (IP-address, port number, etc.).
=> Section 3.2 and 3.3
3. Insert the Modbus function blocks into your SIMATIC project.
=> Section 3.4
4. **Modbus Client:** Call and parameterize the Modbus block MB_REDCL in the required OBs.
=> Section 5.2 and 5.3

or:

Modbus Server: Call and parameterize the Modbus block MB_REDSV in the required OBs.
=> Section 6.2 and 6.3
5. Load the user program into the PLC and license the Modbus block for this CPU.
=> Section 4

3 Commissioning

General Information

The configuration of the CP 443-1 is possible via MPI or LAN/Industrial Ethernet. Required software is STEP7 with NCM S7 for Industrial Ethernet (shortly named "NCM IE").

All statements in the following sections referring to STEP7 or NCM IE are related to the STEP7-Version 5.4 SP3 and NCM S7 Industrial Ethernet Version 5.4.

Operation flows, names and directory names might be different in other STEP7 versions.

Requirements

Knowledge of AWL and basic knowledge of STEP7 and PLC.

3.1 Installing the Library on the STEP7 PG/-PC

What We Provide You

The attached CD contains a setup, which installs the library "**Modbus_TCP_CP_Red300_400**", the example projects and the manuals in English and German in the corresponding STEP7 directories.

The manuals can also be found on the CD as PDF file.

Requirements

To install, **STEP7** must be installed.

Installation

Insert your Modbus CD into the CD-ROM drive and follow step-by-step the instructions that are automatically displayed by the installation program. If the installation program fails to automatically run, perform these steps:

1. Using Windows Explorer, navigate to the CD-ROM drive and go to the directory setup and double-click to the setup file to start the installation procedure.
2. Follow step-by-step the instructions that are displayed by the installation program.

Now you can find

- the libraries in \Program Files\Siemens\Step7\S7libs,
- the example projects in \Program Files\Siemens\Step7\Examples,
- the manual in \Program Files\Siemens\Step7\S7manual\S7Comm.
- the Software Registration Form in \Program Files\Siemens\Step7\S7libs\ Modbus_TCP_CP_Red300_400.

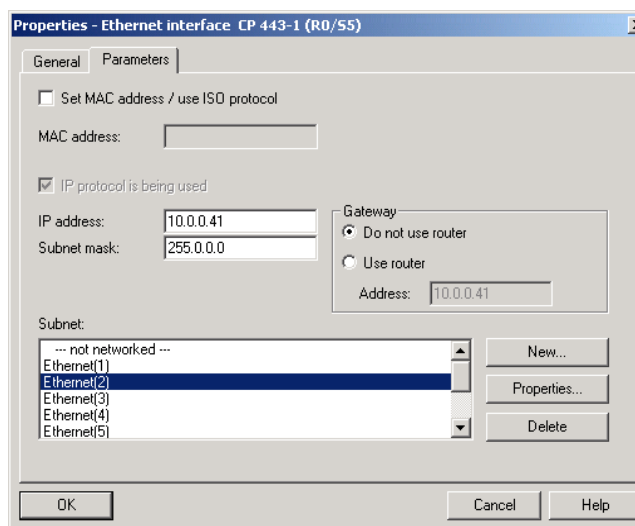
To access the Modbus library the first time, use the browse function of the open dialog for libraries.

The manual can be accessed via short cut in \Program Files \Siemens \Documentation as well.

3.2 Parameterization of the CP

Parameterization of the CP

If you have your stations connected with each other without a router, then they have to be within the same subnet. In the field *Subnet* connect the CP with the Industrial Ethernet. In order to do that, select the entry with the name of your network. For newly created networks this is normally **"Ethernet(1)"**.



Save and Compile the parameterization.

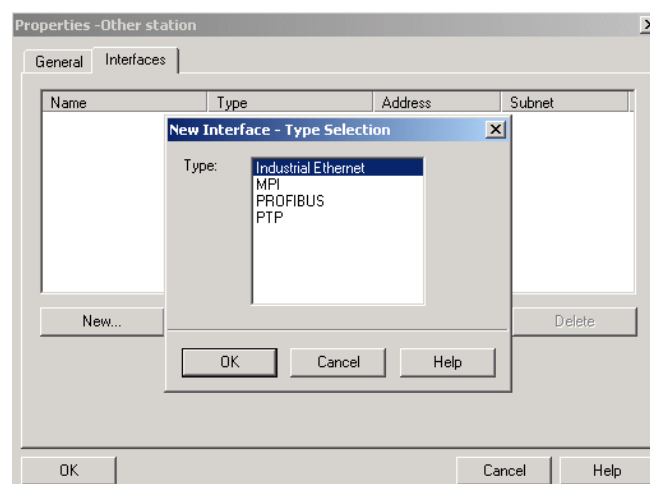
Parameterization of the Communication Partner

In the **mode "CP is client"** an "other station" is required for network configuration.

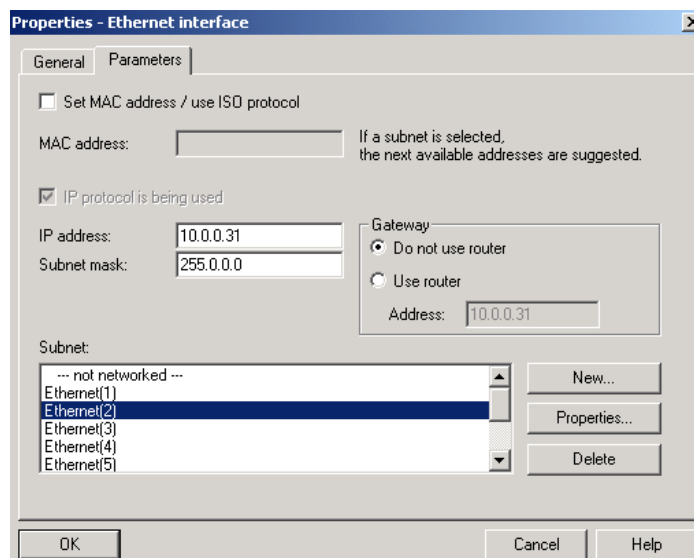
After you have inserted the communication partner's station into your STEP7 project you have to specify the object properties of the external station.

1) Properties – Other Station ➤ Interfaces

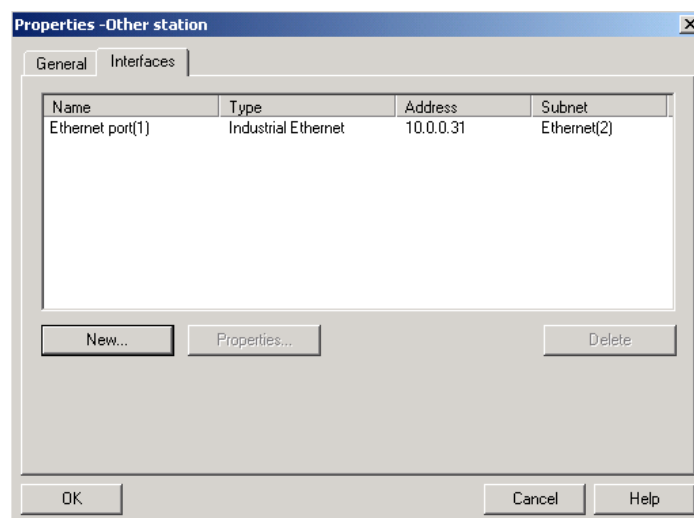
On the tab **"Interfaces"** click on "New". In the upcoming selection, select **"Industrial Ethernet"** and click on "OK".



This opens a dialog box “Properties – Ethernet Interface“. Enter an IP Address that is in the same subnet as the communication partner’s station. The subnet mask should be the same as the one of the partner’s station. Select the associated **subnet** that connects the CP interface with the communication partner’s interface.



Click on the “OK“ button. This will bring you back to the tab “Interfaces“.



2) Properties – Other Station ➤ General

In tab “**General**” you do not have to make any settings.

3.3 Network Configuration

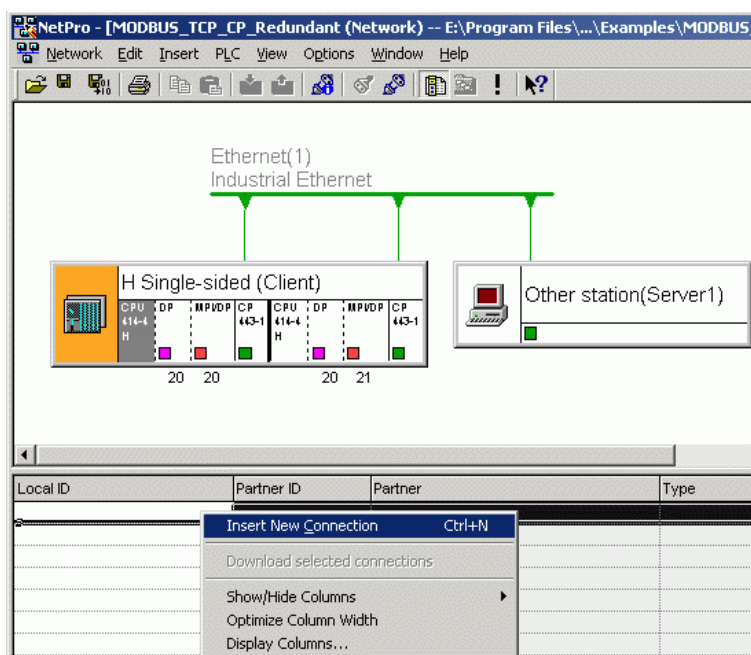
Communications Connection

The CP is the link for the Industrial Ethernet connection between the S7-CPU and the communication partner / bus. A connection configuration must be made for the connection of the interfaces to the communication partner / bus.

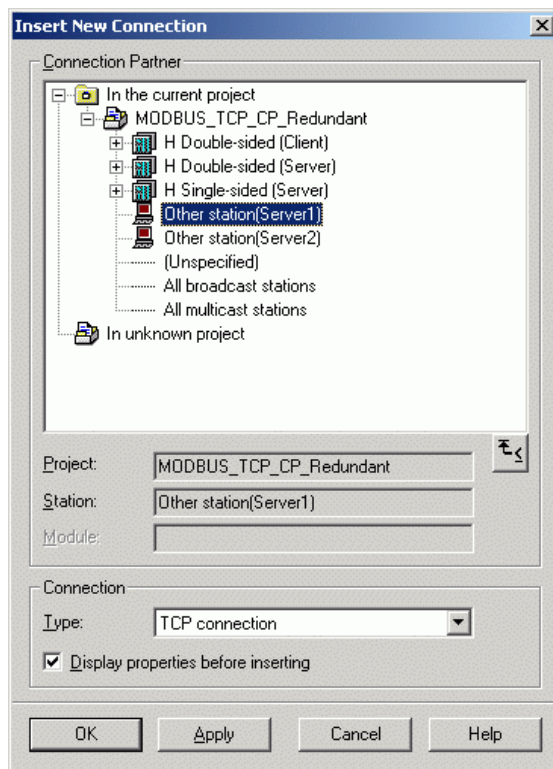
Configure Network in mode "CP is Client"

In the STEP7 project, select the CPU in your S7400H-Station and open "Network configuration" by double clicking "**Connections**". This opens the program "**NetPro**" with which your connections can be configured.

After selecting **Insert** ➤ **New Connection...** the dialog box "Insert new connection" will come up.



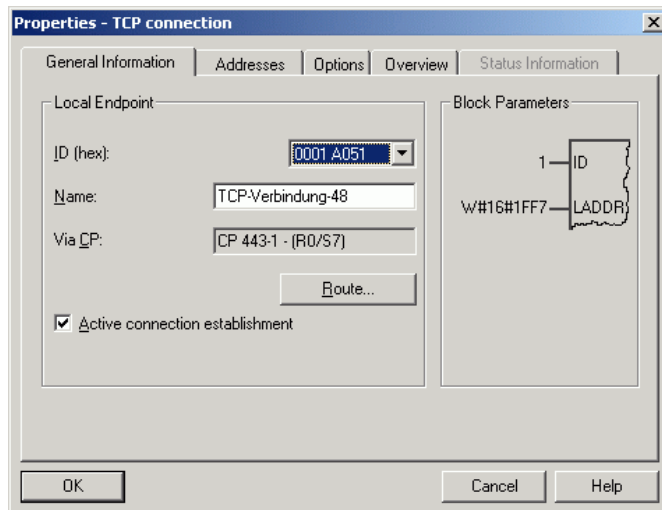
Select the connection partner (Other Station) for the new connection and use "TCP Connection" for the connection. Put a check mark on "Show properties dialog".



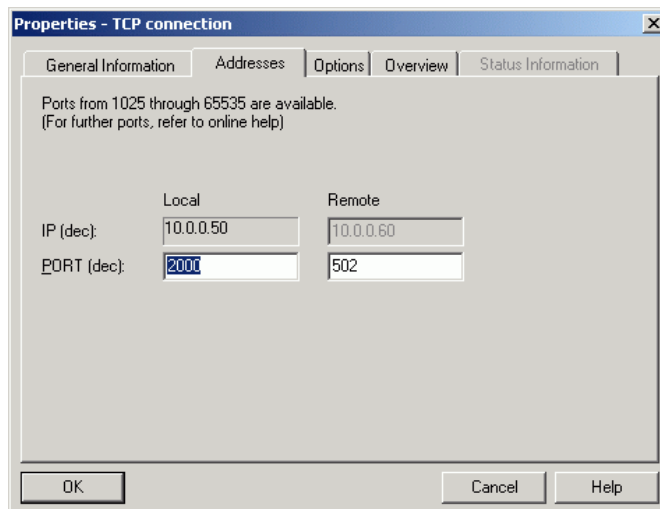
Click "OK". This will take you back to the dialog box "Properties TCP connection".

Object Properties of the Connection

- An ID is provided. You can change the ID if needed.
- Click on the button "Routing" and the configured connection will be shown.
- The MODBUS client does "Active connection establishment".



- In the register “Addresses” the port numbers are defined.



Click on “OK” and the inputs are accepted.

Save the network configuration and close the program “NetPro”.

Please note that the connection ID (Local ID) has to be used when the FB is called in the user program.

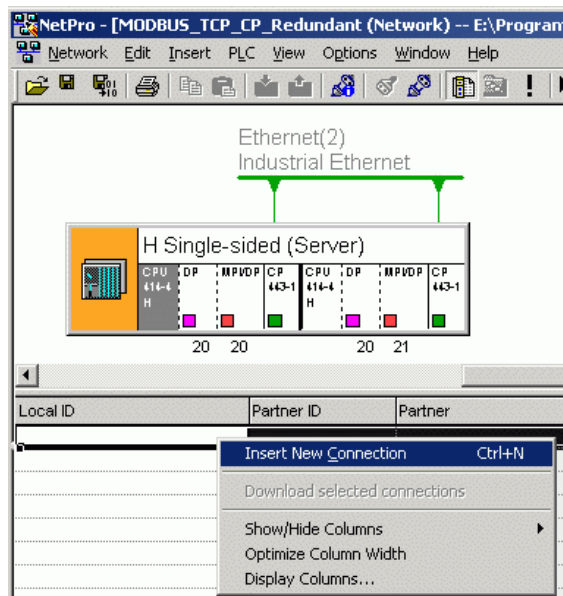
Selection of the Port Number

In a MODBUS communication a MODBUS server are normally addressed via port 502, whereas a MODBUS client uses a port from 2000.

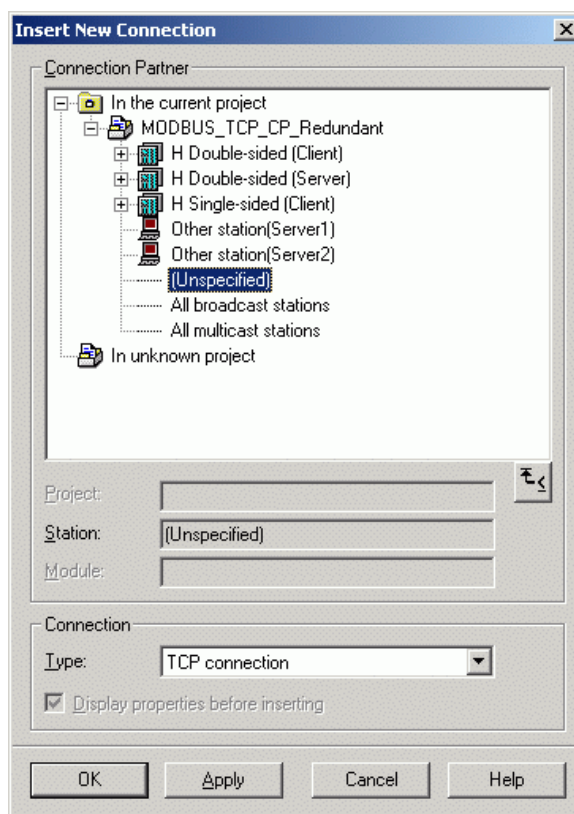
Unspecified Connection with "CP is server"

If you have got a communication with CP as MODBUS server, the communication is set up as "unspecified connection". The client has to meet the requirement of active connection establishment.

After selecting **Insert** ➤ **New Connection...** the dialog box "Insert new connection" will come up.

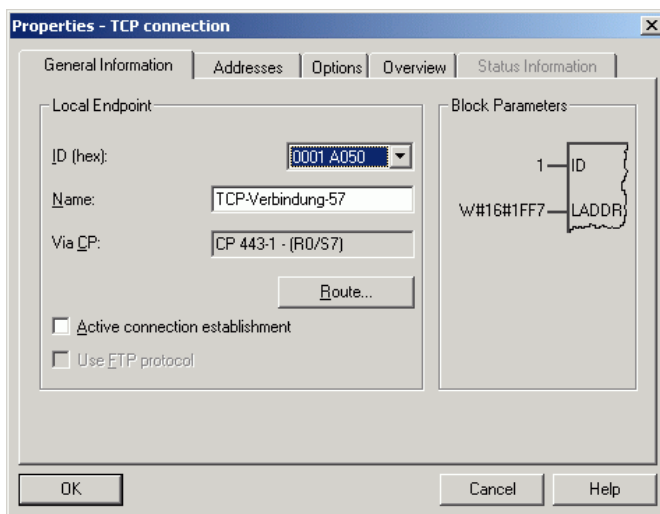


Select here "**unspecified**" instead of the communication partner and use "TCP Connection" for the connection. Put a check mark on "Show properties dialog"

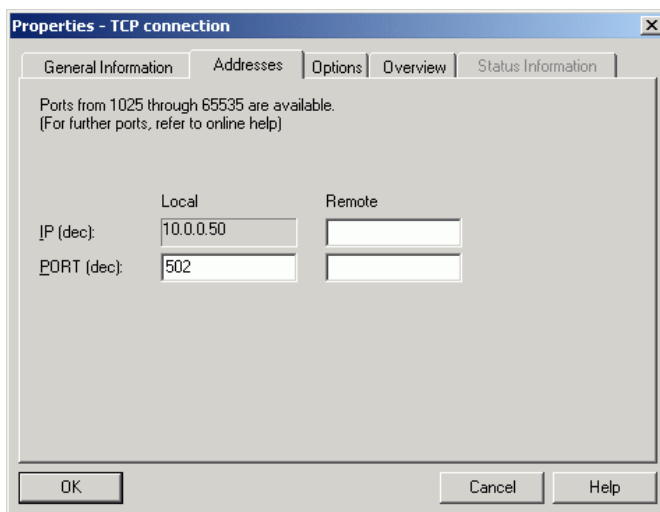


Click “OK”. This will take you back to the dialog box “Properties TCP connection”.

The check box “active connection establishment” must not be activated.



In the register “addresses” all information regarding the partner, “IP” and “PORT” are left blank.



Click on “OK” and the inputs are accepted.

3.4 Insertion of the Function Blocks into the Program

Content of MODBUS library

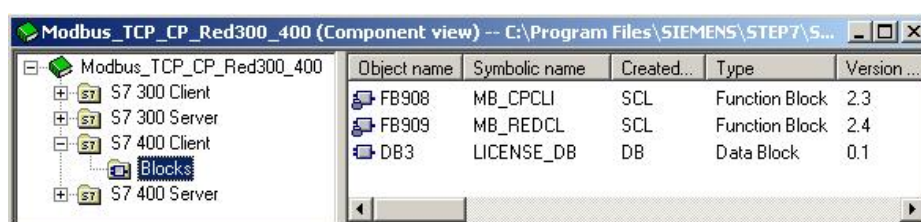
The Modbus library contains the folders „S7 300 Client“, „S7 300 Server“, „S7 400 Client“ and „S7 400 Server“ with the FBs for the redundant communication.

S7 400 Client

The folder “S7 400 Client” contains the blocks

- FB909 (MB_REDCL) and
- FB908 (MB_CPCLI)

amongst others.



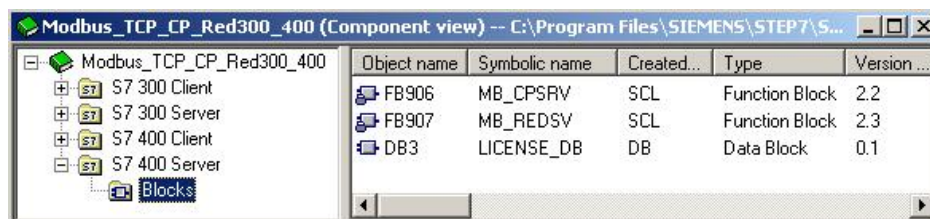
Both blocks are required for redundant communication. The function block MB_REDCL executes a multiple call of MB_CPCLI.

S7 400 Server

The folder “S7 400 Server” contains the blocks

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

amongst others.



Both blocks are required for redundant communication. The function block MB_REDSV executes a multiple call of MB_CPSRV.

Insertion of the blocks

Open the library “**Modbus_TCP_CP_Red300_400**” and copy the necessary blocks. Insert the blocks into your program.

The Modbus function block uses the function blocks AG_LSEND and AG_LRECV. Copy these functions from the library “**SIMATIC_NET_CP**” -> “**CP 400**” and insert them into your program.

Copy also the function AG_CNTRL from the library “**SIMATIC_NET_CP**” into your program.

Please note, that the following versions of the FCs are a prerequisite for the faultless function of the FBs MB_REDCL/MB_REDSV:

AG_LSEND	V3.1 or higher
AG_LRECV	V3.1 or higher

3.5 Multiple Connections via Port 502

General

Some CP modules are able to multiplex TCP connections. In doing so several Modbus clients can establish a connection via port 502. The CP acts as Modbus server.

Here you can find information which CP and which firmware release support multiple connections via port 502:

www.siemens.com/s7modbus

Pre-requisites

To use the functionality, the parameterization must be carried out as follows:

- CP is server
- port 502 as local port
- unspecified TCP connection in NETPRO
- passive connection establishment

Please note that only 1 connection is parameterized in NETPRO irrespective of the number of clients which address the CP as server.

Number of released Connections

The CP is capable to keep up to **4 connections** to different clients.

Displaying the Status of the Connection

The status of the connection is displayed even in NETPRO online as well as in special diagnosis of the CP.

As only 1 connection is parameterized in NETPRO, the display shows the status of all TCP connections to the several clients.

As long as no client has established a connection "Passive connection establishment in progress" is displayed.

As soon as one client has established a connection "established" is displayed. It is not possible to check how many clients have actually established a connection to the CP.

Characteristics of the Error Handling

The FB MB_REDSV and the CP respectively must terminate and reestablish the connection in certain error situations. This action is carried out by the FB MB_REDSV. Thereby all existing connections via port 502 are terminated.

Tip for your User Program

When several connections are established via port 502, it is not possible to identify the client which has sent the recent request.

If the clients use different UNIT numbers, the verification of the same in the user program allows a determination.

3.6 Start_up Characteristics of CP443

Introduction	<p>The start up of the CP is divided into the following phases:</p> <ul style="list-style-type: none">• Initialization (Power on of the CP)• Parameterization
Initialization	<p>As soon as the CP is connected to power, the hardware self test runs. The firmware of the CP is set up for operation.</p>
Parameterization	<p>During parameterization the CP receives the device parameters that are assigned to its slot.</p> <p>The CP is now ready for operation.</p>

4 Licensing

General

The blocks MB_REDCL and MB_REDSV must be licensed for each CPU individually.

The licensing takes place in two steps: reading the IDENT_CODE and declaring the registration key REG_KEY. The OB121 must be available in the CPU.


Please note:

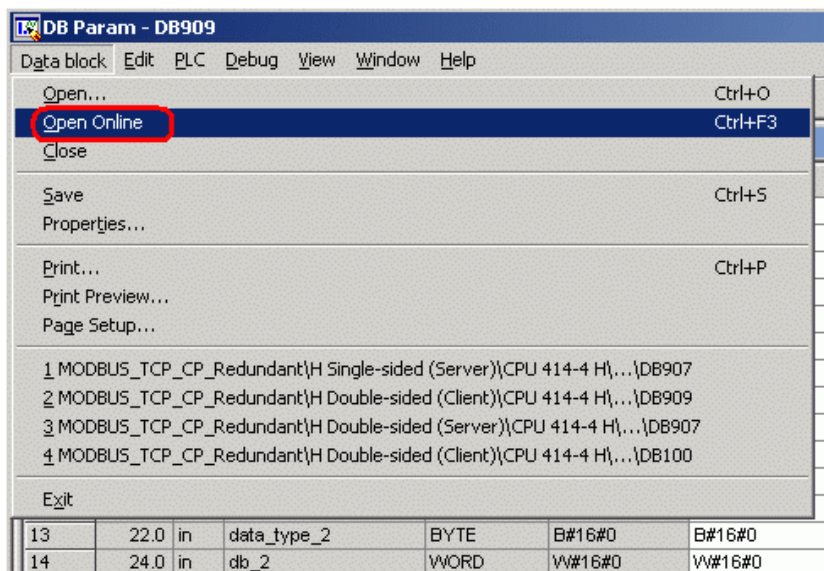
With a S7-H station only the CPU in rack 0 is licensed. Hence the replacement of the CPU in rack 0 after licensing is no longer possible.

Read the IDENT_CODE

To read the IDENT_CODE please proceed as follows:

1. Parameterize the block MB_REDCL and MB_REDSV respectively in the cyclic OB (OB1 or cyclic interrupt OB) and in OB100 according to your requirements. Transfer the program to the PLC and turn it to RUN mode.
2. Open the instance DB of the Modbus block. „Data block“ -> „Open Online“ to open the DB.

Monitoring the block via the button  is insufficient.



- The output IDENT_CODE shows an 18 character string

Copy this string per copy/paste from the DB und and insert it in the form **SOFTWARE REGISTRATION FORM**. This form is stored in the library path `..\Program Files\Siemens\Step7\S7LIBS\Modbus_TCP_CP_Red300_400` during installation and is also available on the installation CD.
Insert the License-No. of the product package into the form.

DB Param - [DB909 - MODBUS_TCP_CP_Redundant\H Single-sided (Client)\CPU 414-4 H...]

Address	Decl	Name	Type	Initial value	Actual value
24	63.0	in	ENQ	BOOL	FALSE
25	64.0	in	DATA_TYPE	BYTE	B#16#0
26	66.0	in	START_ADDRESS	WORD	W#16#0
27	68.0	in	LENGTH	WORD	W#16#0
28	70.0	in	WRITE_READ	BOOL	FALSE
29	71.0	in	UNIT	BYTE	B#16#0
30	72.0	out	LICENSED	BOOL	FALSE
31	72.1	out	BUSY	BOOL	FALSE
32	72.2	out	DONE	BOOL	FALSE
33	72.3	out	ERROR	BOOL	FALSE
34	74.0	out	STATUS_OA	WORD	W#16#A090
35	76.0	out	STATUS_IA	WORD	W#16#A090
36	78.0	out	STATUS_OB	WORD	W#16#FFFF
37	80.0	out	STATUS_IB	WORD	W#16#FFFF
38	82.0	out	IDENT_CODE	STRING [18]	EMCEKBAHJAMJMEKK2
39	102.0	stat	SEND_BUFFER[1]	BYTE	B#16#0
40	103.0	stat	SEND_BUFFER[2]	BYTE	B#16#0

Press F1 for help. RUN

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 Produkts.

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

S7-OpenModbus/TCP

Type of Software / Softwaretype: Runtime Software
Type of License / Lizenztype: Single License
Type of the File for the Addressing CPU
No. / Anzahl: 1
Software Class / Softwareklasse: A
Reference hardware / Zielhardware: SIMATIC 300 CPU
PC
System Conditions / Systemanforderung: Windows XP
Remark / Anmerkung: Software and electronic documentation on CD
Order No. / Bestell-Nr.: 2009450-1MB11
License No. / Lizenz-Nr.: 2009450-1MB11-00000000

simatic add on SOFTWARE

- Send this form as [Service Request](#) to Customer Support.
Hereupon you will receive the registration key for your PLC.

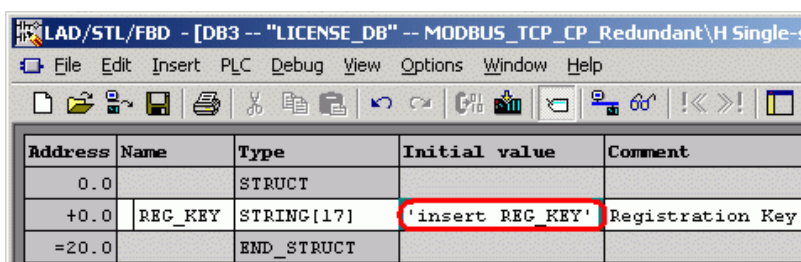
Declaration of the Registration Key REG_KEY

The registration key REG_KEY must be declared for each call of the Modbus block.

The registration key REG_KEY should be stored in a global DB. Via this global DB all Modbus blocks can receive the registration key (See also the following example).

Please proceed as follows to declare the registration key REG_KEY:

1. Copy the prepared license block DB3 of the library "Modbus_TCP_CP_Red300_400" into your project. If the DB number is already used in your project, rename the license DB.
2. Open the license DB and copy the 17 digit registration key you received to the column "Initial value".



The screenshot shows the SIMATIC Manager interface with the title bar "LAD/STL/FBD - [DB3 -- 'LICENSE_DB' -- MODBUS_TCP_CP_Redundant\H Single-s...". The menu bar includes File, Edit, Insert, PLC, Debug, View, Options, Window, and Help. Below the menu bar is a toolbar with various icons. The main window displays a table with the following data:

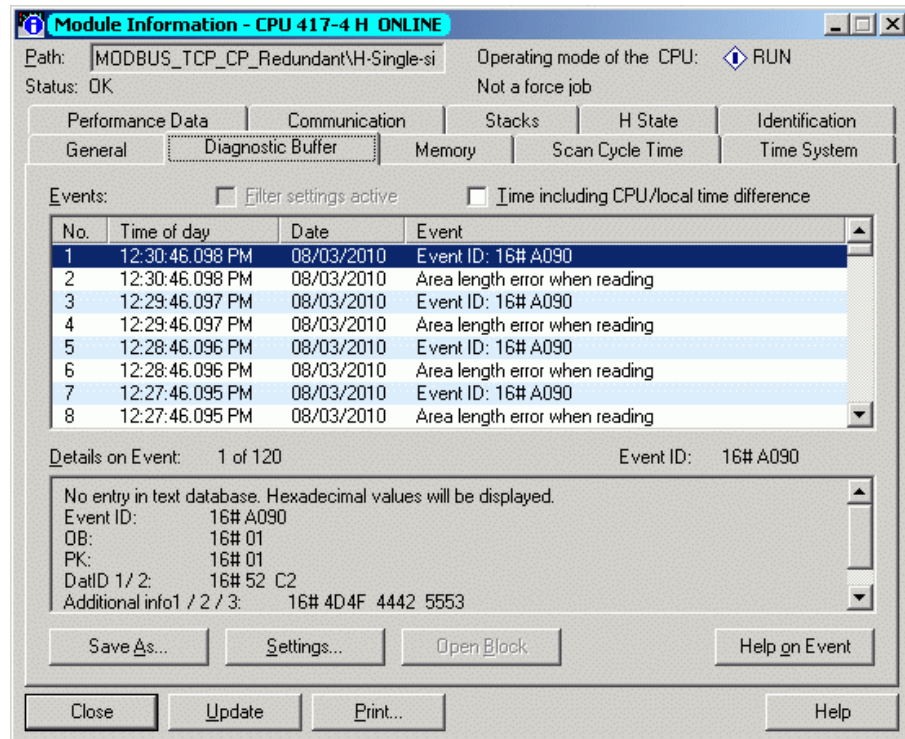
Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	REG_KEY	STRING[17]	'insert REG_KEY'	Registration Key
=20.0		END_STRUCT		

3. Declare the registration key in the data block as "initial value" to avoid a repeated insertion after reloading the PLC. Open the data block in the SIMATIC manager with the editor in the declaration view. Change over to the data view via the menu "View" -> "Data View". Choose in the menu "Edit > Initialize Data Block" – all values of the column "initial value" are copied to "actual values".
4. Assign the value "DB3.REG_KEY" to the parameter REG_KEY of the Modbus block.
5. Transfer the changed blocks to the PLC. The registration key can be set at run time. A STOP -> RUN transition is not necessary.

The block is now licensed for this CPU.

Missing or Wrong Licensing

When the registration key is missing or a wrong one is detected, the INTF LED of the CPU is flashing once per minute. A cyclic error message regarding the missing license is displayed in the diagnostic buffer, too. The error number of the missing license is W#16#A090.



Warning

The CPU will turn to STOP mode, if the OB121 is not available.

Modbus communication is carried out even with a missing or wrong registration key, but the outputs STATUS_x is set to W#16#A090 "no valid license".

5 Function Block MB_REDCL – Modbus Client

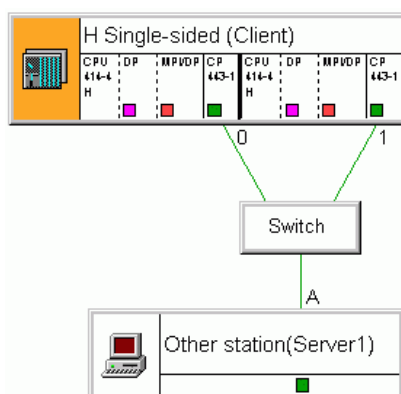
5.1 Configuration of the Redundant Communication

General Information

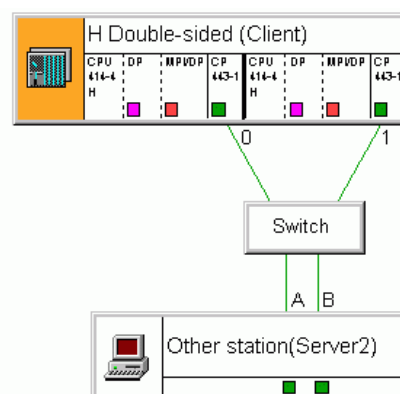
The CP is client if the S7 takes the initiative to read data from or to write data to the remote partner.

The communication partner of the H-system can be mounted stand-alone or redundant, too (single sided or double sided redundancy).

Single sided redundancy



Double sided redundancy



Configuration in HW Config

While configuring the hardware in HW Config, both CP0 and CP1 get different input and IP addresses, so that they can be addressed unique in the S7-program respectively from the communication partner.

Configuration in NetPro

You have to define one connection in NetPro for each possible connection between the communication partners.

With single sided redundancy, there is one connection for CPU0/CP0 and one for CPU1/CP1:

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

With double-sided redundancy, there are two connections for CPU0/CP0 and two 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**

The figures in the following example illustrate the denotation of the connections.

Please note when configuring network connections, that the end points of the connection (S7: CP0 und CP1, partner: node A and node B) must have at least one distinctive feature for addressing: either the IP address or the port number. In HW Config CP0 und CP 1 get always different IP-addresses. Therefore in the network configuration each CP can use the same port number.

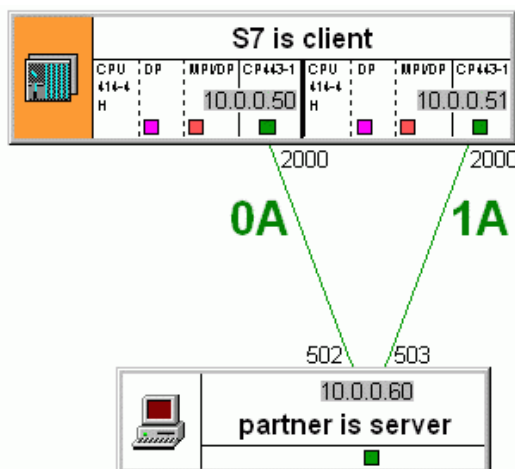
If the communication partner has got only one IP address, then each connection has to use a different port number.

Usually the Modbus server is addressed via the port number 502; the Modbus client uses a port number from 2000.

A wrong port number (identical port numbers) will be recognized by NetPro while entering the value and/or closing the configuration window.

Example: Single Sided Redundancy

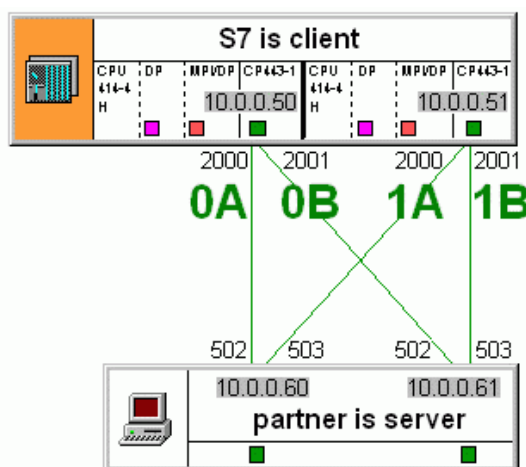
The following figure illustrates a configuration example in NetPro for single sided redundancy.



The S7-station has got the IP-addresses 10.0.0.50 and 10.0.0.51 and can use the port number 2000 for both connections. The communication partner has got the IP-Address 10.0.0.60 and must be addressed with 2 different port numbers: 502 and 503.

Example: Double Sided Redundancy

The following figure illustrates a configuration example in NetPro for double sided redundancy.



The S7-station has got the IP-addresses 10.0.0.50 and 10.0.0.51. For the access to node A of the communication partner, both CPs, CP0 and CP1, can use port number 2000, because both of them are having different IP-Addresses (connection **0A** and **1A**).

For the access to node B of the communication partner, it is also possible that both CPs, CP0 and CP1, use the same Port number 2001 (Connection **0B** and **1B**).

The communication partner has got the IP-Addresses 10.0.0.60 and 10.0.0.61. For the access to CP0 of S7, node A and node B can use the same port number: 502 (Connection **0A** and **0B**). For the access to CP1 of S7 it is also possible to use the same port number: 503 (Connection **1A** and **1B**).

5.2 Functionality of FB MB_REDCL

General

It is possible to run an H-station as Modbus client and Modbus server simultaneously. Therefore the adequate blocks for client and server communication must be called and the necessary connections in NetPro must be parameterized. For each group of redundant connections – consisting of 2 connections for single sided redundancy or 4 connections for double sided redundancy – the Modbus block must be called once. That means, when the H-station acts as client and as server with single sided redundancy, 2 connections for the server call and 2 connections for the client call are required in NetPro. When the H-station acts as client and as server with double sided redundancy, 4 connections for the server call and 4 connections for the client call are required in NetPro.

There is no limitation of the maximum number of parallel called Modbus blocks on the part of the library. Though it depends on the CPU how many AG functions can run simultaneously. The maximum number of AG calls can be taken from the manual of the CPU: "Technical Data" > "Communication". In the manual of the CP it's detailed, how many connections can be processed by this CP simultaneously.

Performed Functions

The function block MB_REDCL performs the following functions:

- Coordination via which connection(s) the telegrams are sent
- Monitoring of all parameterized connections with AG_CNTRL
- Control the Transaction identifier TI

The function block MB_REDCL executes a multiple call of MB_CPCLI and coordinates these calls for the different connections.

The function block MB_CPCLI performs the following functions:

- Calls the standard functions for the data transfer between the CPU and the CP
- Generates MODBUS specific telegram header before send
- Verification of the MODBUS specific telegram header after receive
- Verification if the memory areas exist which are requested by the client
- Data transfer to and from the parameterized DB
- Monitoring the data reception with a time-out

Call of the FB

The function block has to be called both in the start up OB100 as well as in a cyclic OB.

For each connection of NetPro, the FB MB_REDCL can be called **once**.

Online-Help

In SIMATIC Manager an online help for function block MB_REDCL is provided. Mark the FB and press the key "F1". The online help is displayed; it contains the main information regarding the FB.

Start up of the FB

The function block MB_REDCL should be unconditionally called once in OB100. The initialization parameters must be set according to the station configuration. They will be copied into the instance DB.

The runtime parameters will not be evaluated during the start up.

Cyclical Operation of the FB

In cyclical operation the MB_REDCL is called e.g. in OB35. According to the runtime parameters, the functions of the function block are activated. While a request is running, changes to the runtime parameters are ignored.

During cyclical operation changes to the initialization parameters are ignored.

**OB121
"Programming
Error"**

If the block has not been licensed yet, the OB121 is called.

Warning

The CPU will turn to STOP mode, if the OB121 is not available.

**Initiate Request
CP is Client**

A rising edge at the trigger input ENQ initiates a request via the active connection. Depending on the inputs parameters UNIT, DATA_TYPE, START_ADDRESS, LENGTH and WRITE_READ a MODBUS request is generated and sent to the partner station via TCP/IP connection. The client waits for the parameterized monitoring time for a response from the server.

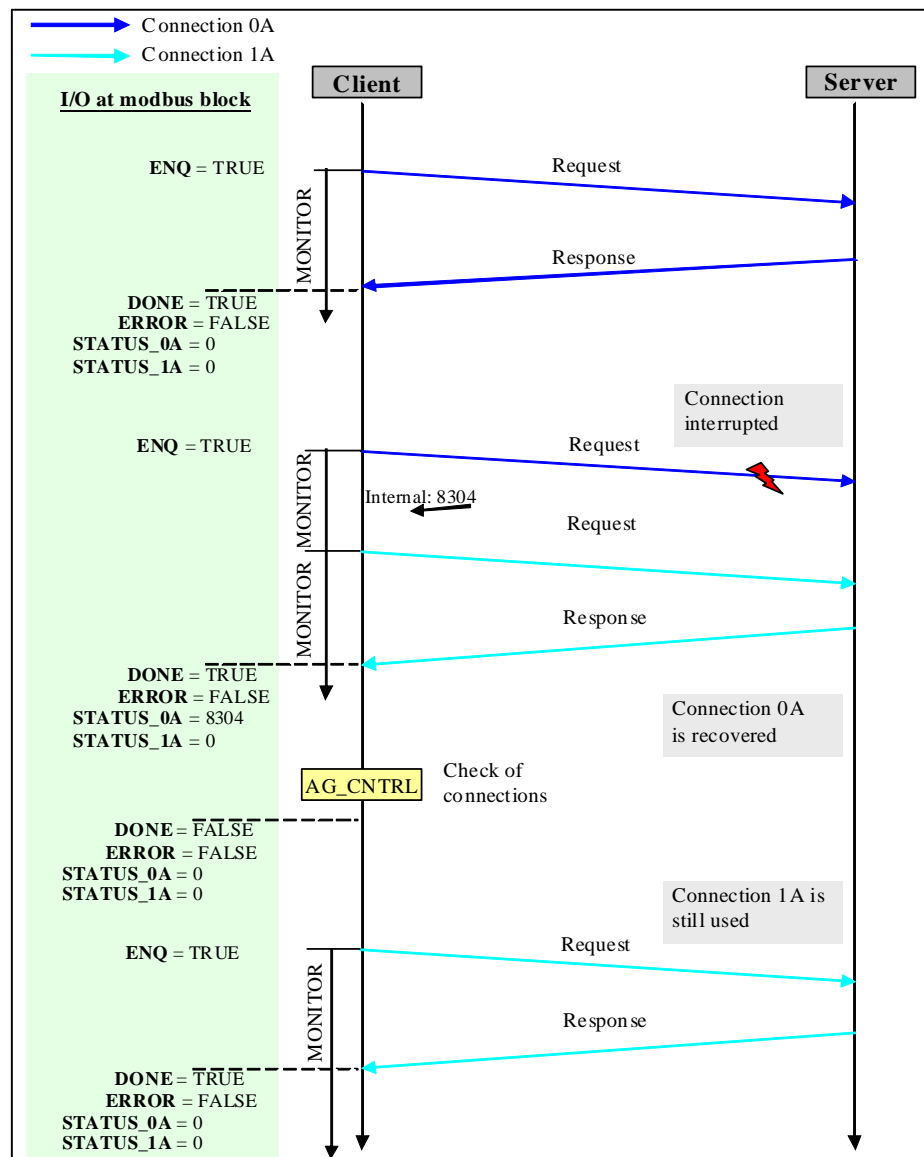
When in the first execution of a cyclic FB after restart of the H-system a job is triggered via all connections, the standby CPU may report error 80B2 once. This happens due to system characteristics of the H-system.

Sending Telegrams via a Single Connection

The Modbus telegram is sent via a **single** - the current active - connection, when use_all_conn = FALSE. In case the monitoring time elapses (no answer from the server) or a faulty connection, the telegram will be sent successively via the other configured connections (maximum 4). The sequence in doing so is 0A, 1A, 0B und 1B. When a telegram was transferred successfully via one connection, this one is marked "active" and the following telegrams are transferred via this active connection. In case of a connection error of the active connection, a transmission retry is carried out via the other parameterized connections. If all transmission attempts fail, ERROR and STATUS_x are set accordingly.

When the client receives a respond, a validity check is carried out. If the result is positive, necessary actions will be taken and the request will be terminated without error. The output DONE is set. When an error is recognized during verification, the request is terminated with an error, the ERROR bit is set and an error number is returned in STATUS_x. In this case the request telegram will not be sent again via the next configured connection.

The switch-over to the next configured connection only takes place if a connection break-down occurs or no answer was received.



Sending Telegrams via all Connections

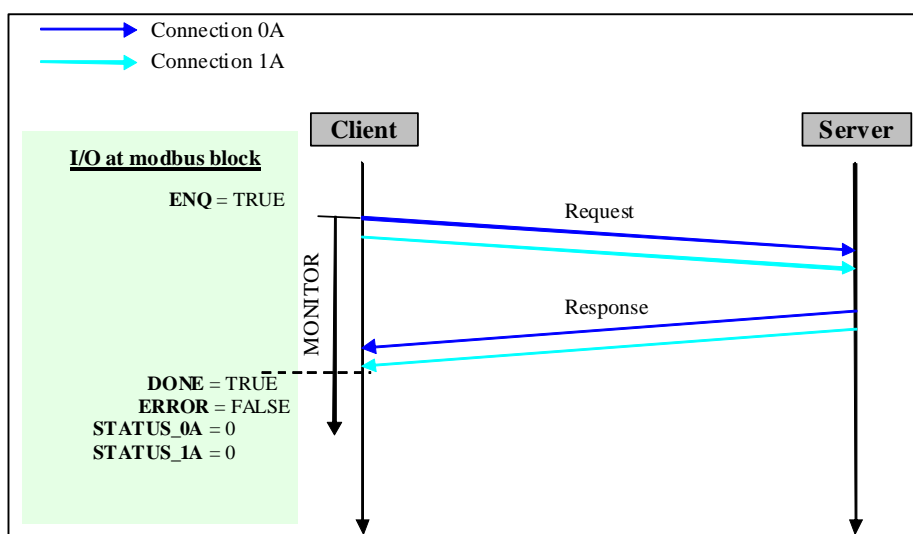
The Modbus telegram is sent via **all** parameterized connection, when use_all_conn = TRUE. When the client receives a respond, a validity check is carried out. If the result is positive, necessary actions will be taken. The outputs **DONE**, **ERROR** and **STATUS** are set **not until** the reception of **responds via all parameterized connections**.

While at least a response is received via one connection, the output DONE is set.

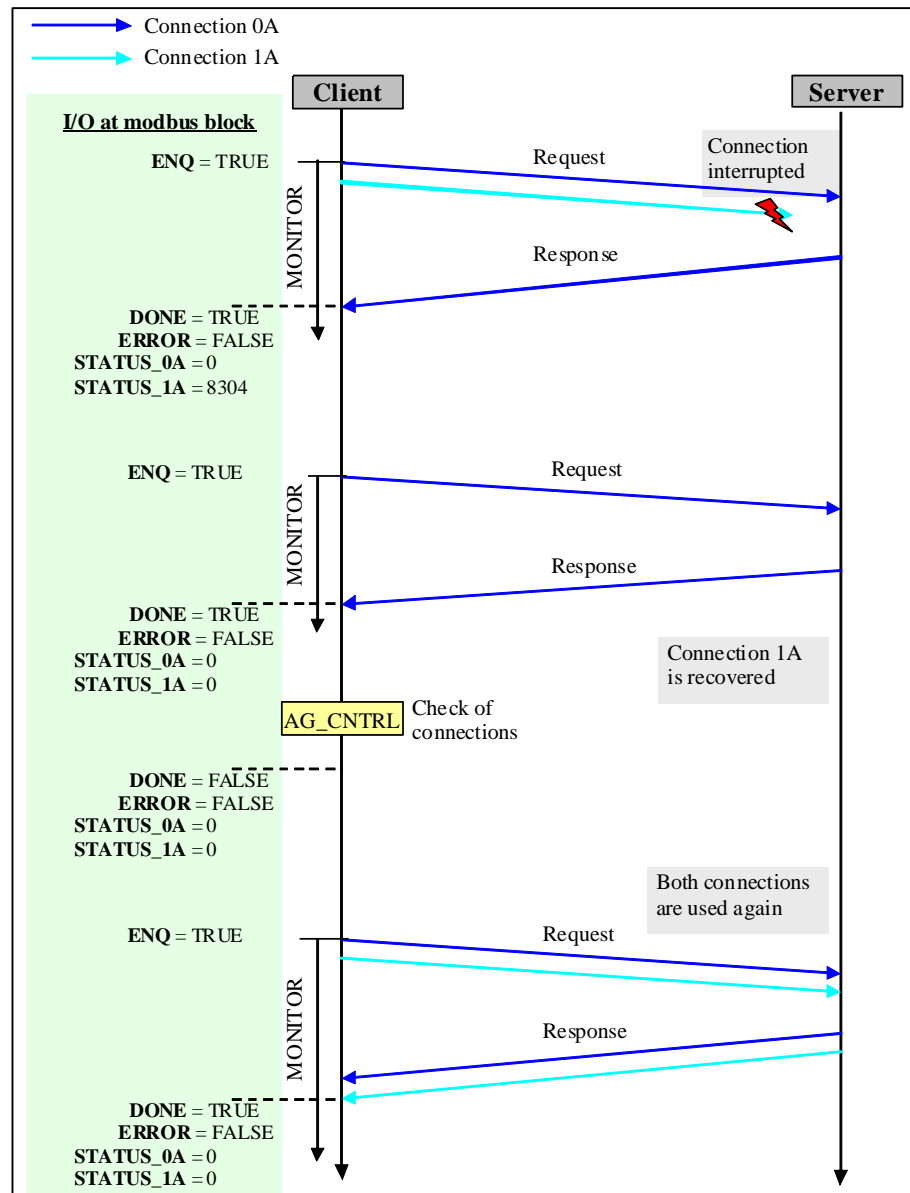
If errors are detected at all connections, the output ERROR is set and the error codes are displayed in STATUS_x.

When a parameterized connection fails, the following Modbus telegrams are not transferred via the faulty connection as long as the connection testing (=> section 5.3) recognizes the reestablishment of this connection.

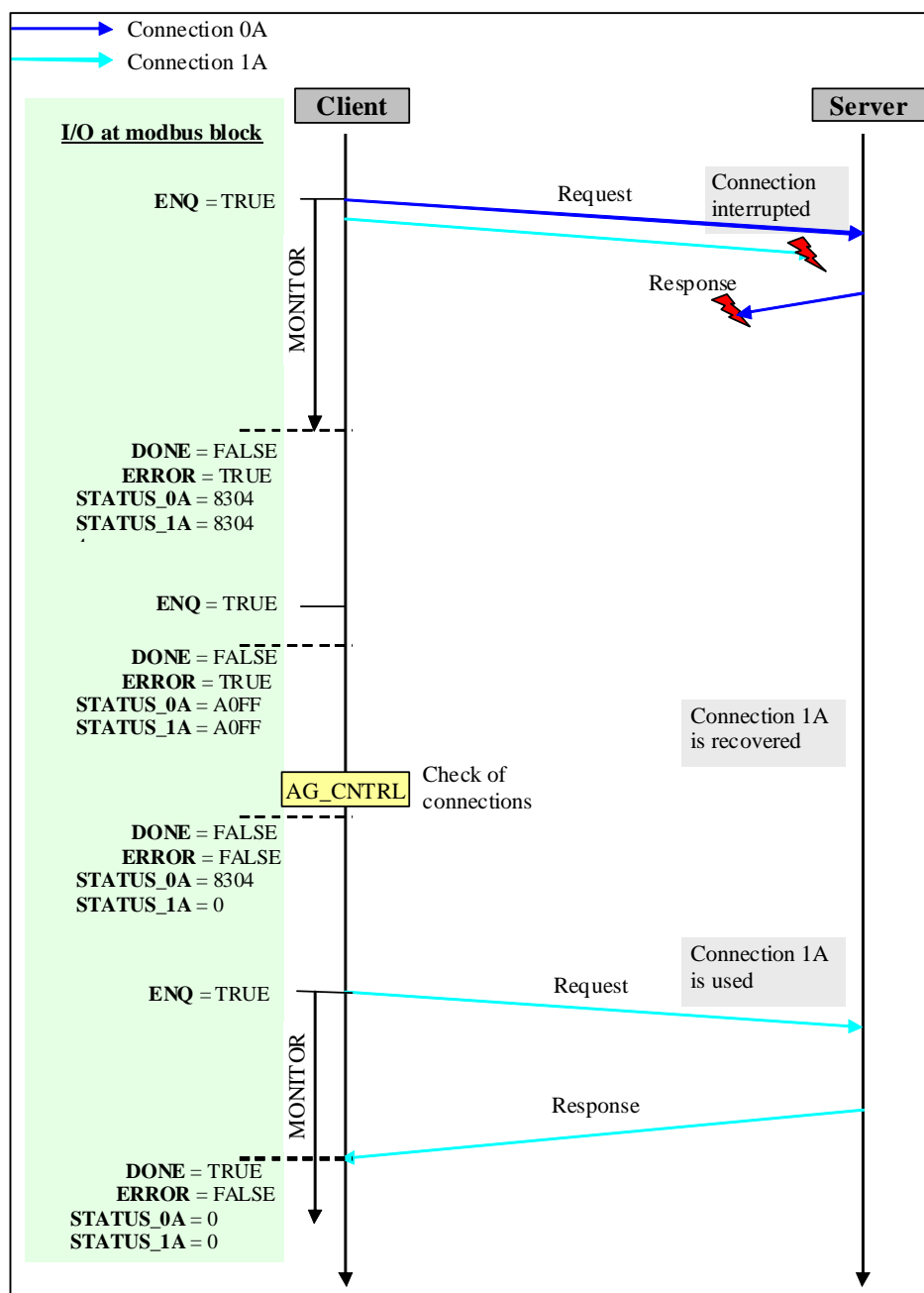
Case a) All responds were received without an error.



Case b) At least one respond was not received successfully.



Case c) Not any response was received.



**Data Transfer
CPU – CP**

The data transfer between CP and CPU are done with the standard function blocks AG_LSEND and AG_LRECV.

At the activation of a MODBUS request by the user the standard blocks necessary for the CP are called by the FB in the right order and number.

**TCP/IP with
CP443-1**

TCP/IP with CP443-1 uses statistic connections. The TCP connection cannot be disconnected while in run mode. Given this system characteristic, telegrams might be lost under unfavorable conditions when the synchronization had been lost after an error. When lost synchronization is recognized, the connection is terminated and reestablished by means of AG_CNTRL.

**Termination of the
TCP connection
by the Communi-
cation Partner**

If the communication partner terminates the TCP connection, the next send can be started not until an idle time of 150 milliseconds has elapsed. This delay, due to system characteristics, is executed by the function block.

**OB121
“Programming
Error”**

If the block has not been licensed yet, the OB121 is called.

Warning

The CPU will turn to STOP mode, if the OB121 is not available.

5.3 Connection Testing by means of AG_CNTRL

**Connection
Testing**

The block MB_REDCL recognizes a faulty connection, if during the transmission of telegrams the communication functions AG_LSEND/ AG_LRECV report an error.

To detect and solve actual connection problems immediately, you can activate a feature to check the configured connections cyclically by the means of AG_CNTRL. The parameter “**check_conn_cycle**” defines the interval of the testing. The result of the connection testing is displayed at the output STATUS_x.

As long as an error of the connection is detected, the FB Modbus doesn't attempt to transfer telegrams via this connection.

When the connection testing recognizes that a faulty connection was reestablished, all further telegrams are transmitted via this connection, too.

When a response was received via all configured connections, the timer of “check_conn_cycle” is restarted.

5.4 Parameters of the Function Block MB_REDCL

Parameter	Decl	Type	Description	Value range	Init
id_0_a	IN	WORD	Connection-ID for CPU/CP0 according to the configuration in NetPro	1 to 64 W#16#1 to W#16#40	yes
id_1_a	IN	WORD	Connection-ID for CPU/CP1 according to the configuration in NetPro	1 to 64 W#16#1 to W#16#40	yes
id_0_b	IN	WORD	Connection-ID for CPU/CP0 according to the configuration in NetPro Only required with double-sided redundancy	1 to 64 W#16#1 to W#16#40	yes
id_1_b	IN	WORD	Connection-ID for CPU/CP1 according to the configuration in NetPro , Only required with double-sided redundancy	1 to 64 W#16#1 to W#16#40	yes
laddr_cp0	IN	WORD	Input address of the CP0 from HW Config	CPU dependent	yes
laddr_cp1	IN	WORD	Input address of the CP1 from HW Config	CPU dependent	yes
check_conn_cycle	IN	TIME	Cycle time for connection testing by the means of AG_CNTRL No connection testing	T#1s to T#+24d20h31m23s 0	yes
use_all_conn	IN	BOOL	Telegrams are sent via one connection Telegrams are sent via all configured connections	FALSE TRUE	yes
single_write	IN	BOOL	Write 1 Coil/Register: Function code 5 and 6 are used respectively Function code 15 and 16 are used respectively	TRUE FALSE	yes
Init	IN	BOOL	Manual initialization with positive edge	TRUE/FALSE	no
data_type_1	IN	BYTE	1st data area: data type Coils Inputs Holding Register Input Register	1 2 3 4	yes
db_1	IN	WORD	1st data area: data block number	1 to 65535 W#16#1 to W#16#FFFF	yes
start_1	IN	WORD	1st data area: first Modbus address in this DB	0 to 65535 W#16#0000 to W#16#FFFF	yes

Parameter	Decl	Type	Description	Value range	Init
data_type_2	IN	BYTE	2nd data area: data type (Coils, Inputs, Holding Register, Input Register), NULL if not used	0 to 4	yes
db_2	IN	WORD	2nd data area: data block number	1 to 65535 W#16#1 to W#16#FFFF	yes
start_2	IN	WORD	2nd data area: first Modbus address in this DB	0 to 65535 W#16#0000 to W#16#FFFF	yes
data_type_3	IN	BYTE	3rd data area: data type (Coils, Inputs, Holding Register, Input Register), NULL if not used	0 to 4	yes
db_3	IN	WORD	3rd data area: data block number	1 to 65535 W#16#1 to W#16#FFFF	yes
start_3	IN	WORD	3rd data area: first Modbus address in this DB	0 to 65535 W#16#0000 to W#16#FFFF	yes
data_type_4	IN	BYTE	4th data area: data type (Coils, Inputs, Holding Register, Input Register), NULL if not used	0 to 4	yes
db_4	IN	WORD	4th data area: data block number	1 to 65535 W#16#1 to W#16#FFFF	yes
start_4	IN	WORD	4th data area: first Modbus address in this DB	0 to 65535 W#16#0000 to W#16#FFFF	yes
MONITOR	IN	TIME	Monitoring Time: Wait for data from communication partner	T#20ms to T#+24d20h31m23s647ms	no
REG_KEY	IN	STRING [17]	Registration key to activate the license	Character	no
ENQ	IN	BOOL	Initiate request at positive edge	TRUE/FALSE	no
DATA_TYPE	IN	BYTE	Data type to be accessed: Coils Inputs Holding registers Input registers	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 <u>Coils</u> : reading function writing function <u>Inputs</u> : reading function <u>Holding Register</u> : reading function writing function <u>Input Register</u> : reading function	1 to 2000 1 to 1968 1 to 2000 1 to 125 1 to 123 1 to 125	no

Parameter	Decl	Type	Description	Value range	Init
WRITE_ READ	IN	BOOL	Write access Read access	TRUE FALSE	no
UNIT	IN	BYTE	Unit Identifier	0 to 255 B#16#0 to B#16#FF	no
LICENSED	OUT	BOOL	License state of the function block: Block is licensed Block is not licensed	TRUE FALSE	no
BUSY	OUT	BOOL	Operating state of the functions AG_LSEND and AG_LRECV Job processing No job processing active	TRUE FALSE	no
DONE	OUT	BOOL	TRUE: Active request finished without errors via one connection at least.	TRUE FALSE	no
ERROR	OUT	BOOL	TRUE: Errors have occurred at all configured 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 Please order your license with this identification string.	Character	no
RedErrS7	OUT	BOOL	TRUE: S7 lost redundancy	TRUE FALSE	no
RedErrDev	OUT	BOOL	TRUE: 3rd party device lost redundancy	TRUE FALSE	no
TotComErr	OUT	BOOL	TRUE: Complete communication failure	TRUE FALSE	no
Init_Error	OUT	BOOL	TRUE: Error occurred during manual initialization	TRUE FALSE	no
Init_Status	OUT	WORD	Status of manual initialization	0 to FFFF	no

General information

The parameters of the FB MB_REDCL can be divided into two groups:

- Initialization parameter
- Runtime parameter

The **initialization parameters** are evaluated only at the first initial execution of the function block MB_REDCL and taken over into the instance DB. They are marked in the above table in the column „INIT“ with „yes“.
A modification of the initialization parameters during the runtime has no impact. After the modification of these parameters (e.g. at the test mode), the instance DB must be initialized via a STOP/RUN transition of the CPU.

Runtime parameters can be used in cyclical operations. It is not advised to change the input parameters while a request is active. Wait with the preparation of the next request and the change of the parameter until the previous request ends with DONE or ERROR.

The output parameters are **displayed dynamically**, i.e. they are only available for **one PLC cycle**. They have to be copied to an additional memory area if you need to process them or to display the values in a VAT (STEP7 variable table).

Range of Values

For the range of values of the different parameters, CPU specific restrictions must be taken in consideration.

id_0_a, id_1_a id_0_b, id_1_b

For each configured connection in STEP7/NetPro, a connection ID is assigned. The connection ID is the distinct description of the connection from the CPU via the CP to the communication partner.

The number of the configured connection has to be entered here. The value range for this parameter depends on the CPU.

id_0_a	represents the connection from the CP0 to node A
id_1_a	represents the connection from the CP1 to node A
id_0_b	represents the connection from the CP0 to node B
id_1_b	represents the connection from the CP1 to node B

The connection 0A is the default connection, its configuration is mandatory.

If the communication partner is built up standalone, then the parameters id_0_a and id_1_a are used.

laddr_cp0, laddr_cp1

The parameters laddr_cp0 and laddr_cp1 represent the input address of the CP0 and the CP1 from HW Config (I-Address). The configured value has to be entered here.

The range of values for these parameters depends on the CPU.

check_conn_cycle

This parameter defines the time interval for connection testing by means of AG_CNTRL. The time can be parameterized in 1 second steps; the default value is 30 seconds. The results of the connection testing are displayed at the outputs STATUS_x.

When use_all_conn = FALSE, the connection testing can be skipped by setting the time interval to 0 seconds.

If check_conn_cycle is parameterized with 0 ms, it's not possible to recognize a connection error without a telegram traffic via this connection. It is recommended to set this parameter > 0 ms.

use_all_conn

This parameter defines the number of connections Modbus telegrams are transferred. With the setting FALSE the Modbus telegrams are sent via 1 connection. This parameter set to TRUE causes the Modbus telegrams to be transferred via all configured connections.

single_write With single_write = TRUE write requests with length 1 are carried out with the function codes 5 and 6.
With single_write = FALSE all write requests use the function codes 15 and 16.

Init The parameter INIT = TRUE enables a manual initialization. The initialization can be executed, if currently no order is running. This must be ensured in the user program with ENQ = FALSE and BUSY = FALSE.
If manual initialization, it is important, that the initialization parameters are parameterized in the cyclical OB.

During a manual initialization the parameterized connections are terminated and reestablished again.

data_type_x The parameter data_type_x defines the MODBUS data type which is to be mapped with this definition.
If data_type_x set to 0, this data area is not used.

Identifier	Data type	Size
0	Area not used	
1	Coils	Bit
2	Inputs	Bit
3	Holding Register	Word
4	Input Register	Word

db_x The parameter db_x defines the number of the data block in which the consecutively defined MODBUS registers or bits are mapped.
0 cannot be used as DB number since it is reserved for system functions.

The size of the data blocks must be 2 byte larger than it is necessary for the Modbus values. The last word is used for internal purposes and it is not allowed to change it.

start_x Start_x specifies the first register or bit address, which is stored in the data element 0 of the DB.

The maximum number of Modbus addresses which can be stored in the DB is defined by its length. The DB length is calculated during start-up of the CPU.

When accessing registers, the last register address which is mapped in the S7 DB can be calculated with the following formula:

$$\text{Last register address in DB} = (\text{DB-length (in byte)} - 2) / 2 + \text{start_x} - 1$$

When accessing coils or inputs, the last register address which is mapped in the S7 DB can be calculated with the following formula:

$$\text{Last coil address in DB} = (\text{DB-length (in byte)} - 2) * 8 + \text{start_x} - 1$$

Please note:

The defined data areas must not overlap.

In section 5.5 you can find an example of the mapping of the MODBUS addresses to S7 memory areas.

MONITOR

The monitoring time observes the entry of the data input from the communication partner over the **active** connection. The minimum time that can be set is 20ms. A monitoring time of 1,5 seconds is recommended.

MONITOR specifies the timeout for the receipt of the complete response telegram from the server. When the monitoring time elapses, the active request is cancelled with an error. The time is started after sending of the request telegram is finished and is stopped after the receipt of the complete data.

REG_KEY

The Modbus block must be licensed for each CPU individually to permit a correct program sequence. With the registration key REG_KEY the Modbus block is licensed and the Modbus communication runs without any restraint.

You can find further information in **section “Licensing”**.

ENQ

The data transfer is initiated with a TRUE signal at the input. The request telegram is generated with the values of input parameters UNIT, WRITE_READ, DATA_TYPE, START_ADDRESS and LENGTH. A new request can only be sent when the previous one is finished with DONE or ERROR.

DATA_TYPE

The parameter DATA_TYPE defines which Modbus data type is to be accessed with the current job. The following data types are available:

Coils	B#16#1
Inputs	B#16#2
Holding Register	B#16#3
Input Register	B#16#4

The different data types are related directly 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 parameter START_ADDRESS specifies the first MODBUS address that is read or written.

LENGTH	<p>The parameter LENGTH specifies the number of MODBUS values that are read or written.</p> <p>For read functions, a maximum of 125 Holding Registers and Input Registers is possible per telegram. For Coils and Inputs a maximum of 2000 bits is possible. For write functions a maximum of 123 Holding Registers or 1968 Coils is possible.</p> <p>All registers or bits have to be in the same DB per telegram.</p>
WRITE_READ	<p>This parameter defines if a read or write function should be carried out. If the value of the input/output is FALSE, it specifies the read mode. The value TRUE specifies the write mode.</p>
UNIT	<p>This input has to be set according to the requirements. The FB copies this value to the request telegram and verifies when the respond telegram is received.</p> <p>Please note, that some communication partners expect a certain unit number.</p>
LICENSED	<p>If this output is TRUE, then the Modbus block is licensed on this CPU. If the output is FALSE, none or a faulty license string was typed in. You can find further information in section 4 “Licensing”.</p>
BUSY	<p>If this output is TRUE, one of the functions AG_LSEND or AG_LRECV is running.</p>
DONE	<p>The activated request was executed without error via one connection at least. For a read function the response data from the server has already been entered into the DB. For a write function the response to the request telegram has been received from the server.</p>
ERROR	<p>When this output is set, errors occurred at all active connections.</p> <p>use_all_conn = FALSE: In case of a protocol error, ERROR is immediately set. In case of a connection error, all the configured connections are verified and ERROR is set not until all connections report an error.</p> <p>use_all_conn = TRUE: When this output is set, errors occurred at all configured connections.</p> <p>The error number is displayed in the STATUS_x outputs.</p>
STATUS_0A, STATUS_1A, STATUS_0B, STATUS_1B	<p>The STATUS_x outputs display the error number when ERROR is TRUE. As long as ERROR is FALSE, STATUS_x displays status information for the corresponding connection.</p> <p>The error numbers and the status information are described in chapter “Diagnostics”.</p>
IDENT_CODE	<p>After start-up of the PLC this parameter displays the identification code, an 18 character string. With this IDENT_CODE you can order the registration key.</p> <p>You can find further information in section “Licensing”.</p>

RedErrS7	<p>If this output is true, the SIMATIC lost redundancy. In single-sided redundancy mode the connection of CP0 or CP1 is disturbed. In double-sided redundancy mode both connections of CP0 or both connections of CP1 are disturbed.</p> <p>You can find further information in section 9.5 “Diagnostics with Alarm Bits”.</p>
RedErrDev	<p>If this output is true, the 3rd party device lost redundancy. In single-sided redundancy mode the connection from node A to CP0 or to CP1 is disturbed. In double-sided redundancy mode both connections to node A or both connections to node B are disturbed.</p> <p>You can find further information in section 9.5 “Diagnostics with Alarm Bits”.</p>
TotComErr	<p>The output TotComErr = TRUE shows a complete communication failure. All connections are disturbed.</p> <p>You can find further information in section 9.5 “Diagnostics with Alarm Bits”.</p>
Init_Error	<p>An error during the manual initialization is shown with Init_Error = TRUE.</p>
Init_Status	<p>The output Init_Status displays the error number when Init_Error is TRUE. The error numbers are described in section 9 “Diagnostics”.</p>

5.5 Example for Address Mapping

Interpretation of Modbus Addresses

MODBUS bases its data model on a series of tables, which have distinguishing characteristics. The distinction between these memory areas is done via the register address by some systems, e.g. MODICON PLCs. So a MODBUS message requesting the read of a holding register at offset 0 would return the value known to the application programmer as found in register 40001 (memory type 4xxxx, reference 0001).

One potential source of confusion is the varying interpretation of the register address in different manuals. Sometimes the register address means the address of the application layer, sometimes the address transferred.

The FB MB_REDCL uses the **Modbus address transferred** at its parameters start_x und START_ADDRESS. So it is possible to use register addresses from von 0000_H to FFFF_H with each function code.

Example: Parameterization of the Memory Areas

data_type_1 db_1 start_1	B#16#3 W#16#B W#16#1	Holding Register DB 11 start address: 0
data_type_2 db_2 start_2	B#16#3 W#16#C W#16#2D0	Holding Register DB 12 start address: 720
data_type_3 db_3 start_3	B#16#0 0 0	not used 0 0
data_type_4 db_4 start_4	B#16#1 W#16#D W#16#2D0	Coils DB 13 start address: 640

For this example applies:

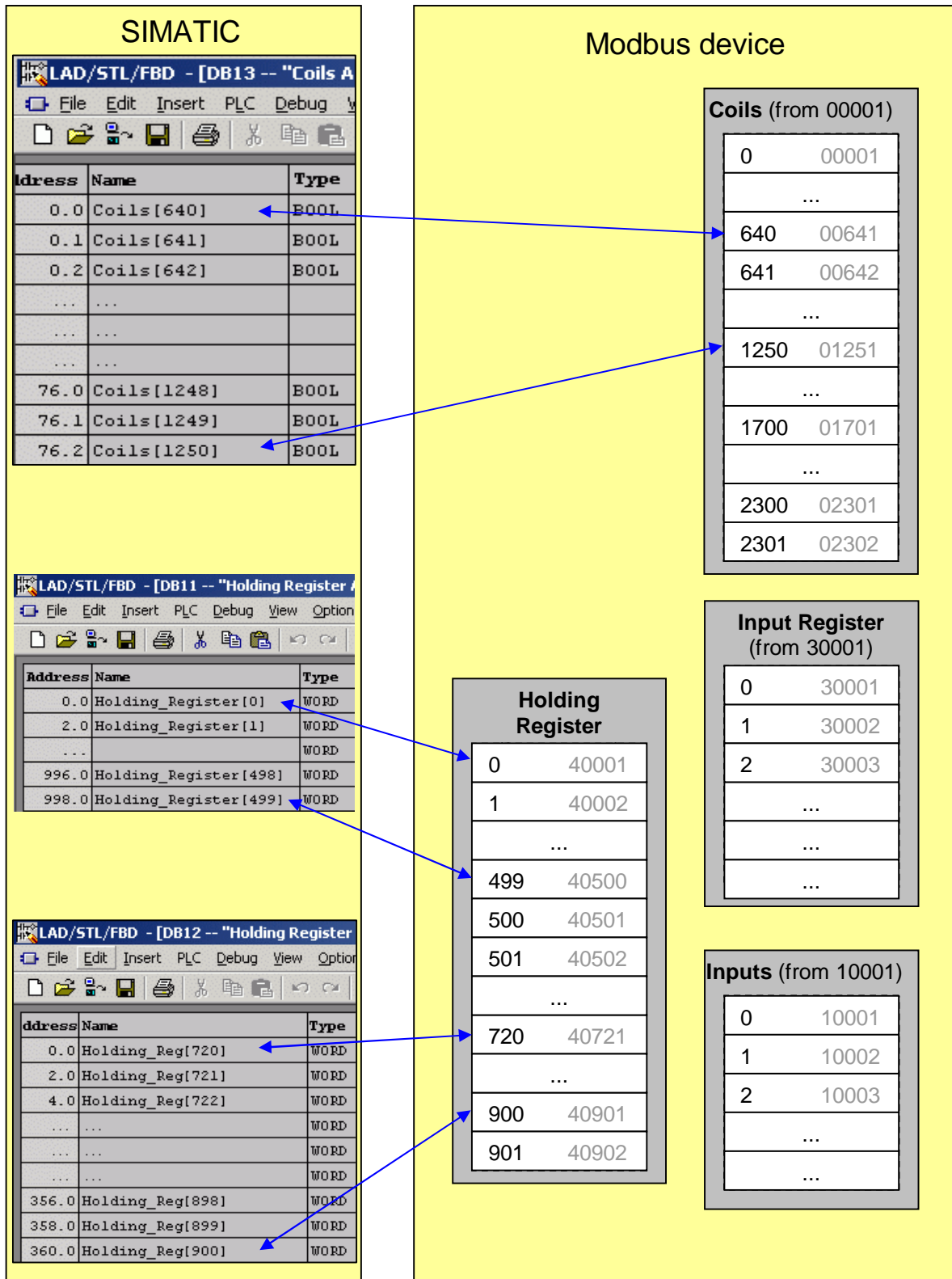
- DB11 consists of 1002 byte; 500 registers are mapped (register 0 – register 499) + 2 reserved byte
- DB12 consists of 364 byte, 181 register are mapped (register 720 – register 900) + 2 reserved byte
- DB13 consists of 79 byte, 611 bits are mapped (Coil 640 – Coil 1250) + 2 reserved byte

Address Mapping

The following diagram shows the comparison of the SIMATIC memory area and the Modbus typical register oriented storage arrangement. The diagram is based on the above parameterization.

Box Modbus device: The Modbus addresses printed in black refer to the Data Link Layer, the ones printed in grey refer to the Application Layer.

Box SIMATIC: The SIMATIC addresses printed in black are the offset in the DB. You can find additionally the Modbus register numbers printed in grey.



5.6 Data and Standard Function used by the FB

Instance DB

The function block MB_REDCL stores its data in an instance DB. This instance DB is created by STEP7 at the first call of the FB.

The instance data block contains parameters of type Input, Output, Input/Output as well as static variables needed for its execution. These variables are non-volatile and keep its validity between FB calls. The variables control the internal process flow of the FB.

Required memory of the instance-DBs:

Instance-DB	work memory	load memory
MB_REDCL	ca. 4 kByte	ca. 5 kByte

Local Variables

The FB requires 96 Bytes of local variables. Additionally the subordinate FBs require local variables: FBs MB_CPCLI (28 byte) or AG_CNTRL (178 byte). That gives a maximum of 302 Bytes of local data for a FB MB_REDCL-call.

Timers

The function block does not use any timer.

Flags

The function block does not use any flags.

Standard-FCs for Data Transfer

The function block uses the blocks AG_LSEND and AG_LRECV from the SIMATIC_NET library for the data transfer between CPU and CP.

AG_CNTRL is used additionally to reset and restart a connection in case of error.

The following versions of the FCs are tested with the FB MB_REDCL and released for the communication:

- FC50 „AG_LSEND“ Version 3.1
- FC60 „AG_LRECV“ Version 3.1
- „AG_CNTRL“ (CP 400) Version 1.0

**SFCs and FCs for
Miscellaneous
Functions**

The FB MB_REDCL uses the following SFCs from the standard library:

- SFB4 „TON“
- SFC6 „RD_SINFO“
- SFC20 „BLKMOV“
- SFC51 „RDSYST“
- SFC52 „WR_USMSG“

The block MB_CPCLI uses the following system functions:

- SFC20 „BLKMOV“
- SFC24 „TEST_DB“
- SFB4 „TON“

**Additional
Information**

The parameter TI is operated internally in MB_REDCL and is incremented with each new request.

Error 8304: When the connection from the CP to the communication partner is interrupted, the interruption is recognized by the CP and error number 8304 stored.

When a communication request is activated, first error code 8304 is returned even if the connection is available again, because this error code is still stored. This is a given system characteristics of the CP.

When the MODBUS function block returns ERROR=TRUE and STATUS=8304, the communication request should be activated once again.

The time delay to recognize the termination of a connection can be modified with the parameter “Send Keepalives for Connections” of the CP’s properties in HW Config.

5.7 Renaming / Rewiring of Standard Functions and Function Blocks

Inducement

Whether you have already used the numbers of standard functions in your project or the block number is reserved for a different application (e.g. in CFC), you may rename/rewire the internal called function blocks AG_LSEND/AG_LRECV or the blocks MB_REDCL and MB_CPCLI.

It is not possible to rename/rewire the system functions SFC6, SFC20, SFC24, SFC51 and SFC52 as well as the system function block SFB4.

Sequence

You must follow some rules regarding the block numbers when rewiring blocks in the SIMATIC Manager of STEP7.

If you want to rewire the blocks of the Modbus library, the following sequence is required:

1. FC50 AG_LSEND
FC60 AG_LRECV
2. FB908 MB_CPCLI
FB906 MB_CPSRV
3. FB909 MB_REDCL
FB907 MB_REDSV

It is not necessary to rewire all blocks. Even if you want to rewire only some of the blocks, you must follow the mentioned sequence.

Renaming

To rewire proceed as described subsequently:

1. Get the information about the used operand by clicking "Extras > Reference data > Display".
2. Set the operand priority in the object properties of the block folder to "Absolute value".
3. Call the function "Extras > Re-wire" in the SIMATIC Manager, in order to re-wire the operands into free areas.
4. To be able to keep on using the symbolism diagnostics tools, add the modifications in the symbolism table supplementary.

The modifications can be verified by clicking "Extras > Reference data > Display".

6 Function Block MB_REDSV – Modbus Server

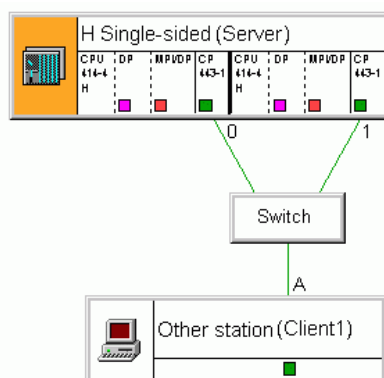
6.1 Configuration of the Redundant Communication

General Information

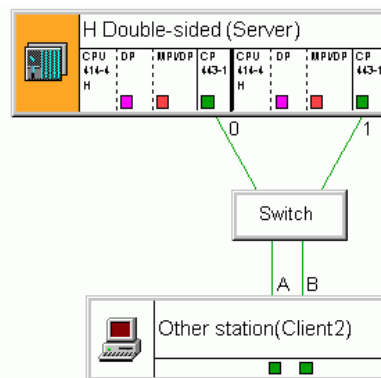
The CP is server if the remote partner takes the initiative to read data from or to write data to the S7.

The communication partner of the H-system can be mounted stand-alone or redundant, too (single sided or double sided redundancy).

Single sided redundancy



Double sided redundancy



Configuration in HW Config

While configuring the hardware in HW Config, both CP0 and CP1 get different input and IP addresses, so that they can be addressed unique in the S7-program respectively from the communication partner.

Configuration in NetPro

You have to define one connection in NetPro for each possible connection between the communication partners.

With single sided redundancy, there is one connection for CPU0/CP0 and one for CPU1/CP1:

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

With double-sided redundancy, there are two connections for CPU0/CP0 and two 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**

The figures in the following example illustrate the denotation of the connections.

Please note when configuring network connections, that the end points of the connection (S7: CP0 und CP1, partner: node A and node B) must have at least one distinctive feature for addressing: either the IP address or the port number. In HW Config CP0 und CP 1 get always different IP-addresses. Therefore in the network configuration each CP can use the same port number.

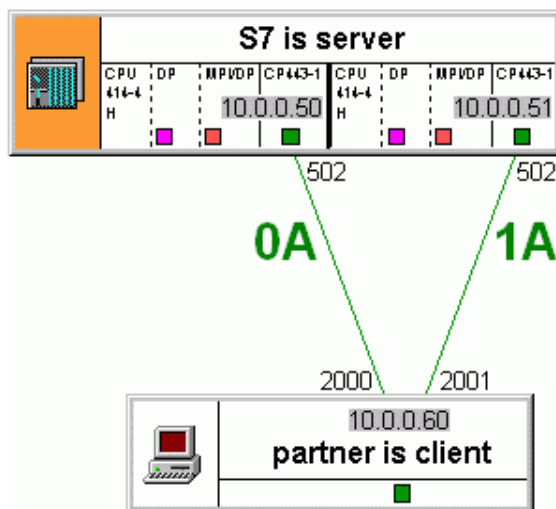
If the communication partner has got only one IP address, then each connection has to use a different port number.

Usually the Modbus server is addressed via the port number 502; the Modbus client uses a port number different from 502.

A wrong port number (identical port numbers) will be recognized by NetPro while entering the value and/or closing the configuration window.

Example: Single Sided Redundancy

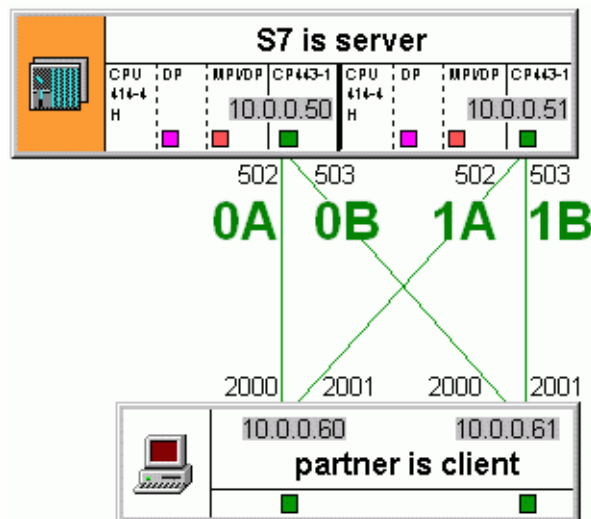
The following figure illustrates a configuration example in NetPro for single sided redundancy.



The S7-station has got the IP-addresses 10.0.0.50 und 10.0.0.51 and can be addressed via both connections with the port number 502. The communication partner has got the IP-Address 10.0.0.60 and has to use 2 different port numbers for the two connections: 2000 und 2001

Example: Double Sided Redundancy

The following figure illustrates a configuration example in NetPro for double sided redundancy.



The S7-station has got the IP addresses 10.0.0.50 and 10.0.0.51 for the access from node A of the communication partner. Both CPs, CP0 and CP1, can use port number 502, because both of them are having different IP-addresses (connection **0A** and **1A**).

For the access from node B of the communication partner, it is also possible that both CPs, CP0 and CP1 use the same port number 503 (Connection **0B** and **1B**).

The communication partner has got the IP-Addresses 10.0.0.60 and 10.0.0.61 for the access to CP0 of S7, node A and node B can use the same port number: 2000 (connection **0A** and **0B**). For the access to CP1 of S7 it is also possible to use the same port number: 2001 (connection **1A** and **1B**).

6.2 Functionality of the FB MB_REDSV

General

It is possible to run an H-station as Modbus client and Modbus server simultaneously. Therefore the adequate blocks for client and server communication must be called and the necessary connections in NetPro must be parameterized. For each group of redundant connections – consisting of 2 connections for single sided redundancy or 4 connections for double sided redundancy – the Modbus block must be called once. That means, when the H-station acts as client and as server with single sided redundancy, 2 connections for the server call and 2 connections for the client call are required in NetPro. When the H-station acts as client and as server with double sided redundancy, 4 connections for the server call and 4 connections for the client call are required in NetPro.

There is no limitation of the maximum number of parallel called Modbus blocks on the part of the library. Though it depends on the CPU how many AG functions can run simultaneously. The maximum number of AG calls can be taken from the manual of the CPU: "Technical Data" > "Communication". In the manual of the CP it's detailed, how many connections can be processed by this CP simultaneously.

Performed Functions

The function block MB_REDSV executes a multiple call of MB_CPSRV and coordinates these calls for the different connections.

The function block MB_CPSRV performs the following functions:

- Calls the standard functions for the data transfer between the CPU and the CP
- Generates MODBUS specific telegram header before send
- Verification of the MODBUS specific telegram header after receive
- Verification if the memory areas exist which are requested by the client
- Generate exception telegrams when failures occur

Exception code	Meaning
1	The requested function code is not supported.
2	An access to a not existing or not permitted address was carried out.
3	An invalid length was indicated for this function code.

- Data transfer to and from the parameterized DB
- Monitoring the data reception with a time-out

Call of the FB

The function block has to be called both in the start up OB100 as well as in a cyclic OB.

Online-Help	<p>In SIMATIC Manager an online help for function block MB_REDSV is provided. Mark the FB and press the key "F1". The online help is displayed; it contains the main information regarding the FB.</p>
Start up of the FB	<p>The function block MB_REDSV should be unconditionally called once in OB100. The initialization parameters must be set according to the station configuration. They will be copied into the instance DB.</p> <p>The runtime parameters will not be evaluated during the start up.</p>
Cyclical Operation of the FB	<p>In cyclical operation the MB_REDSV is called e.g. in OB35. According to the runtime parameters, the functions of the function block are activated.</p> <p>During cyclical operation changes to the initialization parameters are ignored.</p> <p>The connection of the standby CPU may report error 80B2 once after restart of the H-system. This happens due to system characteristics of the H-system.</p>
Activation of the Function Block	<p>With a true signal at the input ENR the FB is ready to receive a request telegram from the client. The server remains passive.</p> <p>With ENR = TRUE all configured connections are active and the CP listens to them. No switch over from one connection to the redundant one takes place. The client can send the request via one or via all connections alternatively.</p> <p>The received telegrams are verified. If a telegram is verified positive, the response telegram is generated and sent. The completed transmission is reported to the user by setting the output NDR_x.</p> <p>A faulty request telegram causes an error message and the output ERROR_x of the associated connection is set. The error number is returned in STATUS_x. Depending on the kind of error, the request of the client is either answered with an exception telegram or no response is sent.</p>

**Instance-DB:
Information
Regarding the
Request of the
Client**

The basic data of the executed Modbus order is stored in an information block in the IDB of the server with each request of the client. This information can be readout in the application program if required. The following values are buffered in the IDB for each connection and are valid when NDR = TRUE:

Address in I-DB for Connection 0A	Variable	Description
DBX 120.0	CONNECTION[1].WRITE_READ	TRUE: Write to S7 FALSE: Read from S7
DBB 121	CONNECTION[1].UNIT	Unit number
DBB 122	CONNECTION[1].DATA_TYPE	Accessed data type 1: Coils 2: Inputs 3: Holding register 4: Input register
DBW 124	CONNECTION[1].START_ADDRESSES	First address accessed
DBW 126	CONNECTION[1].LENGTH	Number of executed registers / bits
DBW 128	CONNECTION[1].TI	Transaction Identifier (Sequence number)
DBD 130	CONNECTION[1].Cnt_NDR	Counter of correct executed requests
DBD 134	CONNECTION[1].Cnt_ERROR	Counter of recognized errors

Two counters for NDR and ERROR are available for each connection as well.

The information block for connection 1A (CONNECTION[2]) starts at DBX 154.0 of the IDB.

The information block for connection 0B (CONNECTION[3]) starts at DBX 188.0 of the IDB.

The information block for connection 1B (CONNECTION[4]) starts at DBX 222.0 of the IDB.

**Data Transfer
CPU – CP**

The data transfer between CP and CPU are done with the standard function blocks AG_LSEND and AG_LRECV.

At the receipt of a telegram from the client the standard blocks necessary for the CP are called by the FB in the right order and number.

**TCP/IP with
CP443-1**

TCP/IP with CP443-1 uses statistic connections. The TCP connection cannot be disconnected while in run mode. Given this system characteristic, telegrams might be lost under unfavorable conditions when the synchronization had been lost after an error. When the FB detects lost synchronization, the connection is terminated and reestablished by means of AG_CNTRL.

**Termination of the
TCP connection
by the Communi-
cation Partner**

If the communication partner terminates the TCP connection, the next receive can be started not until an idle time of 1 second has elapsed. This delay, due to system characteristics, is executed by the function block.

**OB121
“Programming
Error”**

If the block has not been licensed yet, the OB121 is called.

Warning

The CPU will turn to STOP mode, if the OB121 is not available.

6.3 Connection Testing by means of AG_CNTRL

**Connection
Testing**

The block MB_REDSV recognizes a faulty connection, if during the transmission of telegrams the communication functions AG_LSEND/AG_LRECV report an error.

To detect and solve actual connection problems immediately, the configured connections are checked cyclically by the means of AG_CNTRL.

The parameter “**check_conn_cycle**” defines the interval of the testing. The result of the connection testing is displayed at the output STATUS_x.

6.4 Parameters of the Function Block MB_REDSV

Parameter	Decl	Type	Description	Value range	Init
id_0_a	IN	WORD	Connection-ID for CPU/CP0 according to the configuration in NetPro	1 to 64 W#16#1 to W#16#40	yes
id_1_a	IN	WORD	Connection-ID for CPU/CP1 according to the configuration in NetPro	1 to 64 W#16#1 to W#16#40	yes
id_0_b	IN	WORD	Connection-ID for CPU/CP0 according to the configuration in NetPro Only required with double-sided redundancy	1 to 64 W#16#1 to W#16#40	yes
id_1_b	IN	WORD	Connection-ID for CPU/CP1 according to the configuration in NetPro , Only required with double-sided redundancy	1 to 64 W#16#1 to W#16#40	yes
laddr_cp0	IN	WORD	Input address of the CP0 from HW Config	CPU dependent	yes
laddr_cp1	IN	WORD	Input address of the CP1 from HW Config	CPU dependent	yes
check_conn_cycle	IN	TIME	Cycle time for connection testing by the means of AG_CNTRL	T#1s to T#+24d20h31m23s	yes
data_type_1	IN	BYTE	1st data area: data type Coils Inputs Holding Register Input Register	1 2 3 4	yes
db_1	IN	WORD	1st data area: data block number	1 to 65535 W#16#1 to W#16#FFFF	yes
start_1	IN	WORD	1st data area: first Modbus address in this DB	0 to 65535 W#16#0000 to W#16#FFFF	yes
data_type_2	IN	BYTE	2nd data area: data type (Coils, Inputs, Holding Register, Input Register), NULL if not used	0 to 4	yes
db_2	IN	WORD	2nd data area: data block number	1 to 65535 W#16#1 to W#16#FFFF	yes
start_2	IN	WORD	2nd data area: first Modbus address in this DB	0 to 65535 W#16#0000 to W#16#FFFF	yes
data_type_3	IN	BYTE	3rd data area: data type (Coils, Inputs, Holding Register, Input Register), NULL if not used	0 to 4	yes

Parameter	Decl	Type	Description	Value range	Init
db_3	IN	WORD	3rd data area: data block number	1 to 65535 W#16#1 to W#16#FFFF	yes
start_3	IN	WORD	3rd data area: first Modbus address in this DB	0 to 65535 W#16#0000 to W#16#FFFF	yes
data_type_4	IN	BYTE	4th data area: data type (Coils, Inputs, Holding Register, Input Register), NULL if not used	0 to 4	yes
db_4	IN	WORD	4th data area: data block number	1 to 65535 W#16#1 to W#16#FFFF	yes
start_4	IN	WORD	4th data area: first Modbus address in this DB	0 to 65535 W#16#0000 to W#16#FFFF	yes
MONITOR	IN	TIME	Monitoring Time: Wait for data from communication partner	T#20ms to T#+24d20h31m23s647ms	no
REG_KEY	IN	STRING [17]	Registration key to activate the license	Character	no
ENR	IN	BOOL	Ready for receive at true signal	TRUE/FALSE	no
Init	IN	BOOL	Manual initialization with positive edge	TRUE/FALSE	no
LICENSED	OUT	BOOL	License state of the function block: Block is licensed Block is not licensed	TRUE FALSE	no
BUSY	OUT	BOOL	Operating state of the functions AG_LSEND and AG_LRECV Job processing No job processing active	TRUE FALSE	no
NDR_0A	OUT	BOOL	TRUE: The request of the client via connection 0A was executed and responded to.	TRUE FALSE	no
ERROR_0A	OUT	BOOL	TRUE: An error has occurred – 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 via connection 1A was executed and responded to.	TRUE FALSE	no
ERROR_1A	OUT	BOOL	TRUE: An error has occurred – 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 via connection 0B was executed and responded to.	TRUE FALSE	no
ERROR_0B	OUT	BOOL	TRUE: An error has occurred – connection 0B	TRUE FALSE	no

Parameter	Decl	Type	Description	Value range	Init
STATUS_0B	OUT	WORD	Status for connection 0B	0 to FFFF	no
NDR_1B	OUT	BOOL	TRUE: The request of the client via connection 1B was executed and responded to.	TRUE FALSE	no
ERROR_1B	OUT	BOOL	TRUE: An error has occurred – 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. Please order your license with this identification string.	Character	no
RedErrS7	OUT	BOOL	TRUE: S7 lost redundancy	TRUE/FALSE	no
RedErrDev	OUT	BOOL	TRUE: 3rd party device lost redundancy	TRUE/FALSE	no
TotComErr	OUT	BOOL	TRUE: Complete communication failure	TRUE/FALSE	no
Init_Error	OUT	BOOL	TRUE: Error occurred during manual initialization.	TRUE/FALSE	no
Init_Status	OUT	WORD	Status of manual initialization	0 to FFFF	no

General information

The parameters of the FB MB_REDSV can be divided into two groups:

- Initialization parameter
- Runtime parameter

The **initialization parameters** are evaluated only at the first initial execution of the function block MB_REDSV and taken over into the instance DB. They are marked in the above table in the column „INIT“ with „yes“.

A modification of the initialization parameters during the runtime has no impact. After the modification of these parameters (e.g. at the test mode), the instance DB must be initialized via a STOP/RUN transition of the CPU.

Runtime parameters can be used in cyclical operations. It is not advised to change the input parameters while a request is active.

The output parameters are **displayed dynamically**, i.e. they are only available for **one PLC cycle**. They have to be copied to an additional memory area if you need to process them or to display the values in a VAT (STEP7 variable table).

Range of Values

For the range of values of the different parameters, CPU specific restrictions must be taken in consideration.

id_0_a, id_1_a
id_0_b, id_1_b

For each configured connection in STEP7/NetPro, a connection ID is assigned. The connection ID is the distinct description of the connection from the CPU via the CP to the communication partner.

The number of the configured connection has to be entered here. The value range for this parameter depends on the CPU.

id_0_a represents the connection from the CP0 to node A.
 id_1_a represents the connection from the CP1 to node A
 id_0_b represents the connection from the CP0 to node B
 id_1_b represents the connection from the CP1 to node B

The connection 0A is the default connection, its configuration is mandatory.

If the communication partner is built up standalone, then the parameters id_0_a and id_1_a are used.

laddr_cp0,
laddr_cp1

The parameters laddr_cp0 and laddr_cp1 represent the input address of the CP0 and the CP1 from HW Config (I-Address). The configured value has to be entered here.

The range of values for these parameters depends on the CPU.

check_conn_cycle

This parameter defines the time interval for connection testing by the means of AG_CNTRL. The time can be parameterized in 1 second steps; the default value is 30 seconds. The results of the connection testing are displayed at the outputs STATUS_x.

data_type_x

The parameter data_type_x defines the MODBUS data type which is to be mapped with this definition.
 If data_type_x set to 0, this data area is not used.

Identifier	Data type	Size
0	Area not used	
1	Coils	Bit
2	Inputs	Bit
3	Holding Register	Word
4	Input Register	Word

db_x

The parameter db_x defines the number of the data block in which the consecutively defined MODBUS registers or bits are mapped.
 0 cannot be used as DB number since it is reserved for system functions.

The size of the data blocks must be 2 byte larger than it is necessary for the Modbus values. The last word is used for internal purposes and it is not allowed to change it.

start_x	<p>Start_x specifies the first register or bit address, which is stored in the data element 0 of the DB.</p> <p>The maximum number of Modbus addresses which can be stored in the DB is defined by its length. The DB length is calculated during start-up of the CPU.</p> <p>When accessing registers, the last register address which is mapped in the S7 DB can be calculated with the following formula:</p> $\text{Last register address in DB} = (\text{DB-length (in byte)} - 2) / 2 + \text{start_x} - 1$ <p>When accessing coils or inputs, the last register address which is mapped in the S7 DB can be calculated with the following formula:</p> $\text{Last coil address in DB} = (\text{DB-length (in byte)} - 2) * 8 + \text{start_x} - 1$ <hr/> <p>Please note: The defined data areas must not overlap.</p> <p>In section 6.5 you can find an example of the mapping of the MODBUS addresses to S7 memory areas.</p> <hr/>
MONITOR	<p>The monitoring time observes the entry of the data input from the communication partner over the active connection. The minimum time that can be set is 20ms. A monitoring time of 1,5 seconds is recommended.</p> <p>MONITOR specifies the timeout for the receipt of the complete response telegram from the server. When the monitoring time elapses, the active request is cancelled with an error. The time is started after sending of the request telegram is finished and is stopped after the receipt of the complete data.</p>
REG_KEY	<p>The Modbus block must be licensed for each CPU individually to permit a correct program sequence. With the registration key REG_KEY the Modbus block is licensed and the Modbus communication runs without any restraint.</p> <p>You can find further information in section “Licensing”.</p>
ENR	<p>The FB is activated with a TRUE signal at this input. Telegrams from the client can be received. With a FALSE signal at this input data is received from the CP and discarded.</p> <p>The FB listens to all configured connections. Requests via all connections are executed and replied to.</p>
Init	<p>The parameter INIT = TRUE enables a manual initialization. The initialization can be executed, if currently no order is running. The client must not send a request during initialisation.</p> <p>If manual initialization, it is important, that the initialization parameters are parameterized in the cyclical OB.</p> <p>During a manual initialization the parameterized connections are terminated and reestablished again.</p>
LICENSED	<p>If this output is TRUE, then the Modbus block is licensed on this CPU. If the output is FALSE, none or a faulty license string was typed in. You can find further information in section 4 “Licensing”.</p>

BUSY	If this output is TRUE, one of the functions AG_LSEND or AG_LRECV is running.
NDR_0A, NDR_1A, NDR_0B, NDR_1B	The output is set for each correct executed request of the client via the corresponding connection.
ERROR_0A, ERROR_1A, ERROR_0B, ERROR_1B	If this output is set, an error occurred at the corresponding connection when receiving a request or sending a response. The error number is displayed in the STATUS_x outputs.
STATUS_0A, STATUS_1A, STATUS_0B, STATUS_1B	<p>The STATUS_x outputs display the error number when ERROR is TRUE. As long as ERROR is FALSE, STATUS_x displays status information for the corresponding connection.</p> <p>The error numbers and status information are described in chapter “Diagnostics”.</p>
IDENT_CODE	<p>After start-up of the PLC this parameter displays the identification code, an 18 character string. With this IDENT_CODE you can order the registration key.</p> <p>You can find further information in section “Licensing”.</p>
RedErrS7	<p>If this output is true, SIMATIC side lost redundancy. In single-sided redundancy mode the connection of CP0 or CP1 is disturbed. In double-sided redundancy mode both connections of CP0 or both connections of CP1 are disturbed.</p> <p>You can find further information in section 9.5 “Diagnostics with Alarm Bits”.</p>
RedErrDev	<p>If this output is true, the 3rd party device lost redundancy. In single-sided redundancy mode the connection from node A to CP0 or to CP1 is disturbed. In double-sided redundancy mode both connections to node A or both connections to node B are disturbed.</p> <p>You can find further information in section 9.5 “Diagnostics with Alarm Bits”.</p>
TotComErr	<p>The output TotComErr = TRUE shows a complete communication failure. All connections are disturbed.</p> <p>You can find further information in section 9.5 “Diagnostics with Alarm Bits”.</p>
Init_Error	An error during the manual initialization is shown with Init_Error = TRUE.
Init_Status	The output Init_Status displays the error number when Init_Error is TRUE. The error numbers are described in section 9 “Diagnostics”.

6.5 Example for Address Mapping

Interpretation of Modbus Addresses

MODBUS bases its data model on a series of tables, which have distinguishing characteristics. The distinction between these memory areas is done via the register address by some systems, e.g. MODICON PLCs. So a MODBUS message requesting the read of a holding register at offset 0 would return the value known to the application programmer as found in register 40001 (memory type 4xxxx, reference 0001).

One potential source of confusion is the varying interpretation of the register address in different manuals. Sometimes the register address means the address of the application layer, sometimes the address transferred.

The FB MB_REDCL uses the **Modbus address transferred** at its parameters start_x und START_ADDRESS. So it is possible to use register addresses from 0000_H to FFFF_H with each function code.

Example: Parameterization of the Memory Areas

data_type_1 db_1 start_1	B#16#3 W#16#B W#16#1	Holding Register DB 11 start address: 0
data_type_2 db_2 start_2	B#16#3 W#16#C W#16#2D0	Holding Register DB 12 start address: 720
data_type_3 db_3 start_3	B#16#0 0 0	not used 0 0
data_type_4 db_4 start_4	B#16#1 W#16#D W#16#2D0	Coils DB 13 start address: 640

For this example applies:

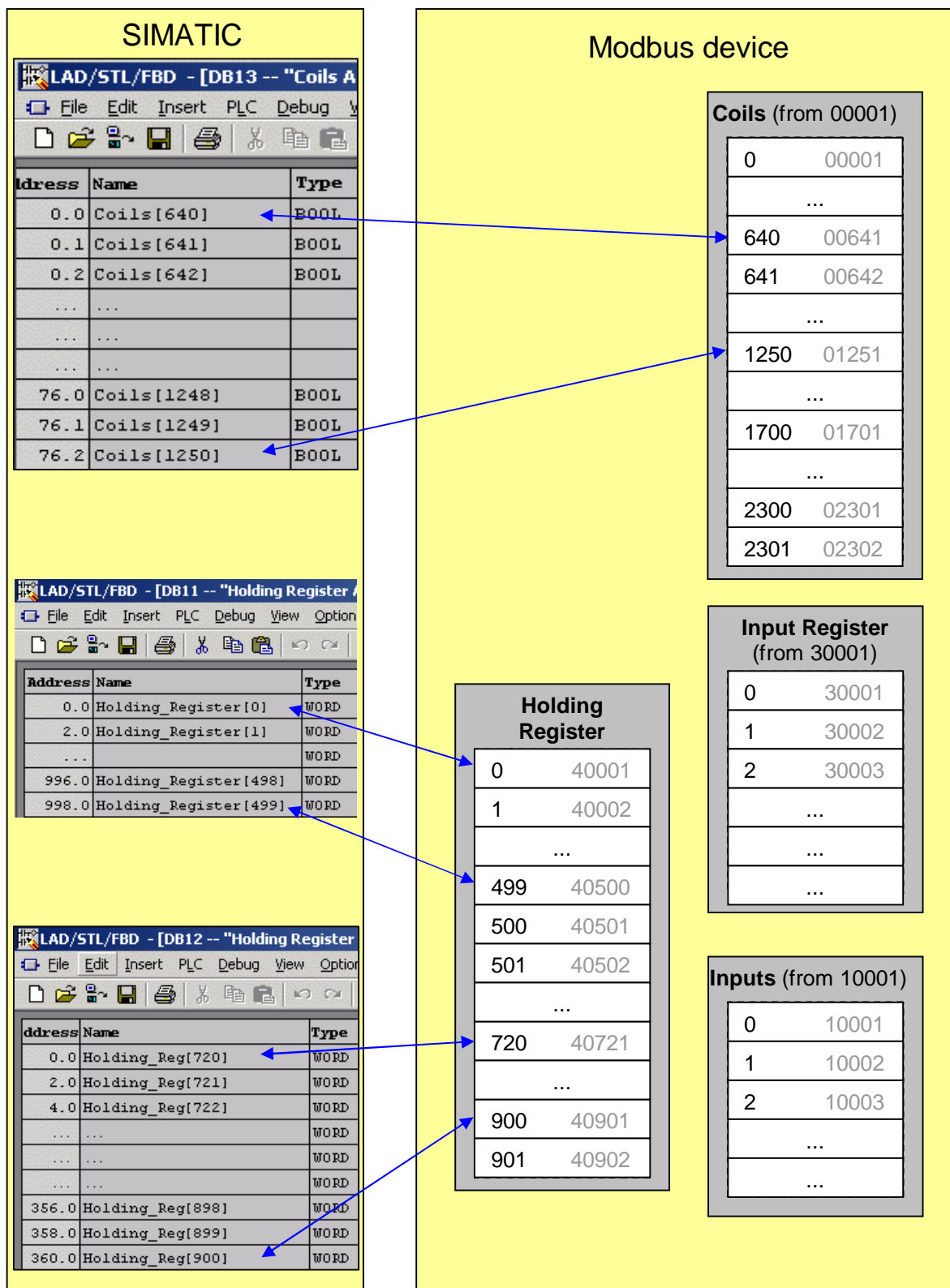
- DB11 consists of 1002 byte; 500 registers are mapped (register 0 – register 499) + 2 reserved byte
- DB12 consists of 364 byte, 181 register are mapped (register 720 – register 900) + 2 reserved byte
- DB13 consists of 79 byte, 611 bits are mapped (Coil 640 – Coil 1250) + 2 reserved byte

Address Mapping

The following diagram shows the comparison of the SIMATIC memory area and the Modbus typical register oriented storage arrangement. The diagram is based on the above parameterization.

Box Modbus device: The Modbus addresses printed in black refer to the Data Link Layer, the ones printed in grey refer to the Application Layer.

Box SIMATIC: The SIMATIC addresses printed in black are the offset in the DB. You can find additionally the Modbus register numbers printed in grey.



6.6 Data and Standard Function used by the FB

Instance DB

The function block MB_REDSV stores its data in an instance DB. This instance DB is created by STEP7 at the first call of the FB.

The instance data block contains parameters of type Input, Output, Input/Output as well as static variables needed for its execution. These variables are non-volatile and keep its validity between FB calls. The variables control the internal process flow of the FB.

Required memory of the instance-DBs:

Instance-DB	work memory	load memory
MB_REDSV	ca. 4 kByte	ca. 5 kByte

Local Variables

The FB requires 96 Bytes of local variables. Additionally the subordinate FBs require local variables: FBs MB_CPSRV (30 byte) or AG_CNTRL (178 byte). That gives a maximum of 304 Bytes of local data for a FB MB_REDSV-call.

Timers

The function block does not use any timer.

Flags

The function block does not use any flags.

Standard-FCs for Data Transfer

The function block uses the blocks AG_LSEND and AG_LRECV from the SIMATIC_NET library for the data transfer between CPU and CP.

The following versions of the FCs are tested with the FB and released for the communication:

- FC50 „AG_LSEND“ Version 3.1
- FC60 „AG_LRECV“ Version 3.1
- „AG_CNTRL“ (CP 400) Version 1.0

SFCs and FCs for Miscellaneous Functions

The FB MB_REDSV uses the following SFCs from the standard library:

- SFB4 „TON“
- SFC6 „RD_SINFO“
- SFC20 „BLKMOV“
- SFC24 „TEST_DB“
- SFC51 „RDSYST“
- SFC52 „WR_USMSG“

The block MB_CPSRV uses the following system functions:

- SFB4 „TON“
- SFC20 „BLKMOV“
- SFC24 „TEST_DB“

**Additional
Information**

Error 8304: When the connection from the CP to the communication partner is interrupted, the interruption is recognized by the CP and error number 8304 stored.

When a communication request is activated, first error code 8304 is returned even if the connection is available again, because this error code is still stored. This is a given system characteristics of the CP.

When the MODBUS function block returns ERROR=TRUE and STATUS=8304, the communication request should be activated once again.

The time delay to recognize the termination of a connection, can be modified with the parameter "Send Keepalives for Connections" of the CP's properties in HW Config.

6.7 Renaming / Rewiring of Standard Functions and Function Blocks

Inducement

Whether you have already used the numbers of standard functions in your project or the block number is reserved for a different application (e.g. in CFC), you may rename/rewire the internal called function blocks AG_LSEND/AG_LRECV or the blocks MB_REDSV and MB_CPSRV.

It is not possible to rename/rewire the system functions SFC6, SFC20, SFC24, SFC51 and SFC52 as well as the system function block SFB4.

Sequence

You must follow some rules regarding the block numbers when rewiring blocks in the SIMATIC Manager of STEP7.

If you want to rewire the blocks of the Modbus library, the following sequence is required:

1. FC50 AG_LSEND
FC60 AG_LRECV
2. FB908 MB_CPCLI
FB906 MB_CPSRV
3. FB909 MB_REDCL
FB907 MB_REDSV

It is not necessary to rewire all blocks. Even if you want to rewire only some of the blocks, you must follow the mentioned sequence.

Renaming

To rewire proceed as described subsequently:

5. Get the information about the used operand by clicking "Extras > Reference data > Display".
6. Set the operand priority in the object properties of the block folder to "Absolute value".
7. Call the function "Extras > Re-wire" in the SIMATIC Manager, in order to re-wire the operands into free areas.
8. To be able to keep on using the symbolism diagnostics tools, add the modifications in the symbolism table supplementary.

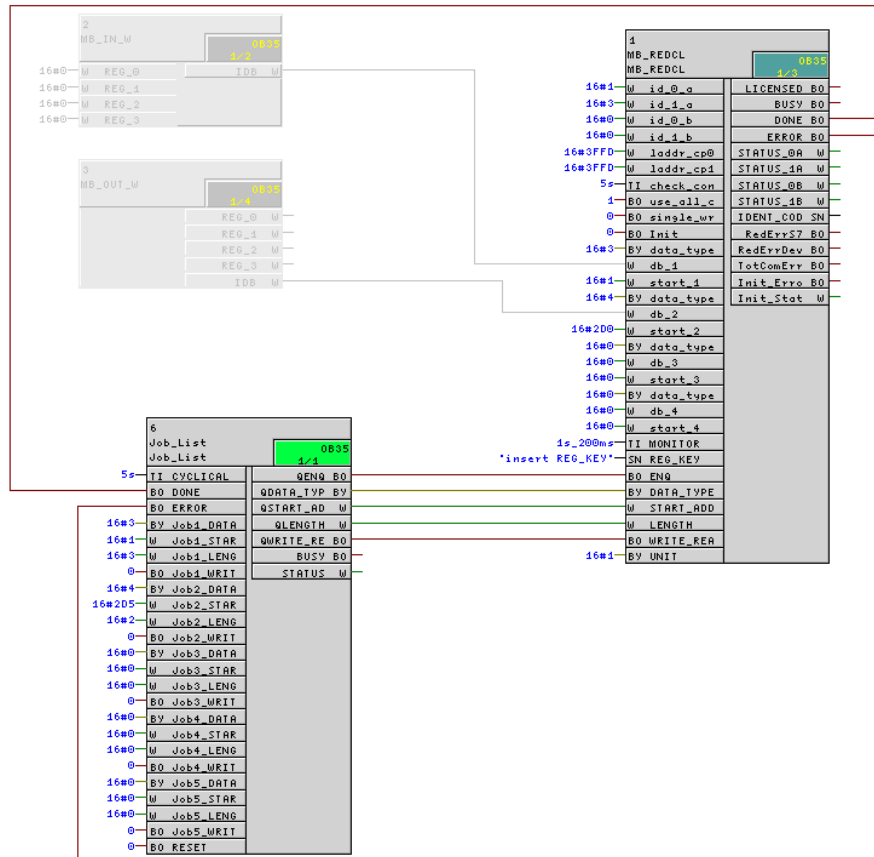
The modifications can be verified by clicking "Extras > Reference data > Display".

7.2 Job List for cyclical telegram transfer

General

By means of the block Job_List it is possible to generate a list of jobs, which are executed cyclically.

Application - Sample



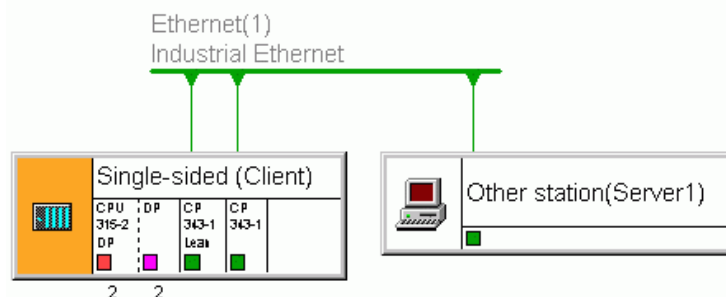
You can find the additional block and a detailed description here:
www.siemens.com/s7modbus or contact the Customer Support.

8 Application with S7-300

General

The library “Modbus/TCP Redundant” can be used with a S7-300 station as well.

The description of functions and parameters in the previous and following sections apply accordingly with S7-300.



Usable Modules for MB_REDCL und MB_REDSV

You can only use s7-300 CPUs which provide enough local data per priority class (=> section 5.6 and section 6.6).

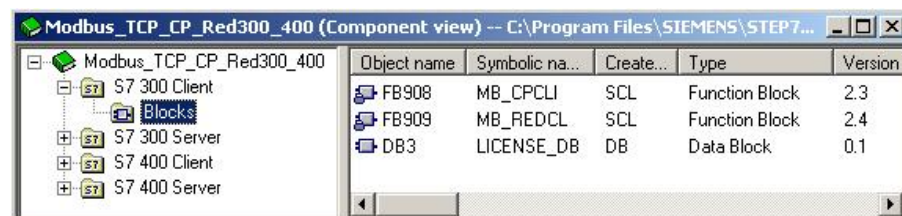
The block AG_CNTRL of the SIMATIC_NET library permits to terminate and reestablish an established connection. This block was also implemented in the Modbus blocks for a more effective use of the resources of CPU and CP. However, previous CPs or previous firmware releases do not support the use of AG_CNTRL.

Here you can find up-to-date information which CPs and which firmware releases support AG_CNTRL: [Ethernet CPs and AG_CNTRL](#).

You can find further hardware prerequisites on the internet: www.siemens.com/s7modbus

Modbus Blocks for S7-300

The installed library “Modbus_TCP_CP_Red300_400” includes the folders “S7 300 client” and “S7 300 server” with the blocks for S7-300.



Copy the blocks into your project and parameterize the blocks as described in this manual.

Please make sure to use the blocks AG_CNTRL, AG_SEND and AG_RECV of the S7-300 library.

9 Diagnostics

Diagnostic Function	<p>The diagnostic functions of the CP443 allow you fast failure localization. The following diagnostic features are available:</p> <ul style="list-style-type: none"> • Diagnostics via the display elements of the CP • Diagnostics via the STATUS output of the modbus function block.
Display Elements (LED)	<p>The display elements inform you about the operating mode or about the failure conditions of the CP. The display elements give you an overview of internal failures, external failures and interface specific failures.</p>
STATUS Output of the MODBUS FB	<p>For error diagnostics, the FBs MB_REDCL and MB_REDSV respectively have got STATUS outputs. By reading the STATUS outputs you get a general indication of failures that have occurred during the communication. The STATUS parameters can be evaluated in the user program.</p>

9.1 Diagnostics via the Display Elements of the CP

Display Functions	<p>The display elements of the CP give you information on the module status. There are two types of display functions:</p> <ul style="list-style-type: none"> • Group Error Displays <ul style="list-style-type: none"> - INTF Internal failure - EXTF External failure • Special Displays <p>CP 443-1:</p> <ul style="list-style-type: none"> - TXD A telegram is being sent via the interface. - RXD A telegram is being received via the interface. <p>A detailed description of the display elements can be found in the device manual of the CP.</p>
--------------------------	--

9.2 Diagnostic Messages of the FBs MB_REDCL and MB_REDSV

Messages at the STATUS Output of the FB

The error messages are displayed at the status outputs of the FBs MB_REDCL and MB_REDSV. Below you will find a list of FB specific error messages.

Error Messages of the called SFCs and FCs

The Modbus FBs use the standard functions SFC6, SFC20, SFC24, SFC51, SFC52, FC50 and FC60. The error messages of these blocks are passed on to STATUS_x without any changes.

In the diagnostics buffer or in the online help of SIMATIC Manager you will find further details on these error messages, as well as in the SIMATIC STEP7 NCM S7 Industrial Ethernet Manual.

Error messages of FBs MB_REDCL and MB_REDSV		
STATUS (Hex)	Event text	Remedy
A002	An error was detected when checking start_x.	Please contact the product support.
A003	The DB, to which MODBUS addresses shall be mapped, is too short. Minimum length in byte: - registers: $(\text{START_ADDRESS} - \text{start_x} + \text{LENGTH}) * 2 + 2$ - bits $(\text{START_ADDRESS} - \text{start_x} + \text{LENGTH}) / 8 + 2$ Other possible reasons: CP is client: Wrong initialization parameter CP is server: Wrong address area in the request telegram of the client	Extend the DB. CP is client: Correct the parameters START_ADDRESS or LENGTH. CP is server: Modify the request of the client.
A004	Applies only with CP is client: An invalid combination of DATA_TYPE and WRITE_READ is given.	Correct the parameters. Only data type 1 or 3 can be written.
A005	CP is client: An invalid value for the parameter LENGTH is given. CP is server: The number of registers or bits in the request telegram is invalid. The CP sends an exception telegram. Range of values: Read coils/inputs: 1 to 2000 Write coils: 1 to 1968 Read registers: 1 to 125 Write holding registers: 1 to 123	CP is client: Correct the parameter LENGTH. CP is server: Modify the number of registers/bits in the request telegram.
A006	The given range of data defined with DATA_TYPE, START_ADDRESS and LENGTH does not exist in data_type_1 to data_type_4. CP is server: The CP sends an exception telegram.	CP is client: Correct the parameter's combination DATA_TYPE, START_ADDRESS, LENGTH. CP is server: Modify the request of the client or correct the parameterization of data_type_x.

STATUS (Hex)	Event text	Remedy
A007	CP is client: An invalid monitoring time MONITOR is parameterized. A value > 20ms is required.	Correct the parameterization.
A008	Monitoring time MONITOR elapsed when AG_RECV waits for receipt. E.g. connection is not established. Partner is not ready. The connection is terminated and reestablished.	Verify error messages at the communication partner. Check if the communication partner needs a special unit identifier.
A009	CP is client: The received transaction identifier TI is not equal to the sent one. The connection is terminated and reestablished.	Verify the data of the communication partner with the help of a telegram trace.
A00A	CP is client: The received UNIT is not equal to the sent one. The connection is terminated and reestablished.	Verify the data of the communication partner with the help of a telegram trace.
A00B	CP is client: Received function code is not equal to the sent one. CP is server: An invalid function code was received. The CP sends an exception telegram. The connection is terminated and reestablished.	CP is client: Verify the data of the communication partner with the help of a telegram trace. CP is server: The Modbus FB supports the function codes 1, 2, 3, 4, 5, 6, 15 and 16
A00C	The received byte count does not match the number of registers/bits. CP is server: The CP sends an exception telegram. The connection is terminated and reestablished.	Verify the data of the communication partner with the help of a telegram trace.
A00D	Only when CP is client: The register/bit address or the number of registers/bits in the response telegram is not equal to the one in the request telegram.	Verify the data of the communication partner with the help of a telegram trace.
A00E	The length indicated in the MODBUS specific telegram header does not match the number of registers/bits or the byte count in the telegram. The FB receives all data and ignores them. The connection is terminated and reestablished.	Verify the data of the communication partner with the help of a telegram trace.
A00F	A protocol identifier <> 0 was received. The connection is terminated and reestablished.	Verify the data of the communication partner with the help of a telegram trace.
A010	In the parameterized area db_1 to db_4 a DB number is used twice.	Correct the parameterization.
A011	An invalid value for DATA_TYPE is given (Value range: 1 to 4).	Correct the parameters.
A012	The parameterized areas data_type_1 and data_type_2 overlap.	Correct the parameterization. The data areas must not contain any overlapping register areas.
A013	The parameterized areas data_type_1 and data_type_3 overlap.	„
A014	The parameterized areas data_type_1 and data_type_4 overlap.	„

STATUS (Hex)	Event text	Remedy
A019	0 is assigned to one of the parameters db_x while the according data_type_x is \neq 0. DB 0 can't be used; it is reserved for system functions.	Correct the parameterization of db_x to > 0 .
A01A	Wrong length in header: Range of values: 3 to 253 bytes. The connection is terminated and reestablished.	Verify the data of the communication partner with the help of a telegram trace.
A01B	CP is server and function code 5: An invalid value for coils was received. CP sends an exception telegram.	Verify the data of the communication partner with the help of a telegram trace.
A01E	The CP has received invalid data which could not be assigned. The connection is terminated and reestablished.	Check the error message of the communication partner and verify the data with a telegram trace if needed.
A01F	The FB MB_REDCL and MB_REDSV respectively have turned to an invalid state.	Please contact the product support.
A020	Check_conn_cycle is set to < 1 s, so no or a too small cycle time is parameterized for AG_CNTRL.	Correct the parameterization. Client: With use_all_conn = TRUE a cycle time > 1 s is required. Server: A cycle time > 1 s is required.
A023	The parameterized areas data_type_2 and data_type_3 overlap.	Correct the parameterization. The data areas must not contain any overlapping register areas.
A024	The parameterized areas data_type_2 and data_type_4 overlap.	"
A034	The parameterized areas data_type_3 and data_type_4 overlap.	"
A07A	An invalid value id is parameterized. Range of values is 1 to 64.	Correct the parameterization of id.
A07C	An invalid value data_type_x was given. The value range is 0 to 4.	Correct the parameterization of data_type_x.
A07D	Parameter data_type_1 is not defined. The parameter area _1 is the default area and must be defined.	Correct the parameterization of data_type_1.
A07E	The DB number of db_x is identical to the number of the instance DB.	Correct the parameterization of db_x.
A080	Different instance DBs were used for the call of Modbus block in OB100 and the cyclic OB.	Modbus block must be called with the identical instance DB in OB100 and the cyclic OB.
A081	Only if CP is client and function code 5: The received coil status is not equal to the sent one.	Verify the data of the communication partner with the help of a telegram trace.
A082	Only if CP is client and function code 6: The received register value is not equal to the sent one.	Verify the data of the communication partner with the help of a telegram trace.

STATUS (Hex)	Event text	Remedy
A083	<p>Only if CP is client: A request was initiated prior to the completion of the previous one. The request is not executed.</p> <p>The manual initialization was started during processing a telegram.</p>	<p>Wait with the initiation of a new request until the previous one was finished either with DONE = TRUE or ERROR = TRUE.</p> <p>Wait for the end of the running job before you start the manual initialization.</p>
A085	An error occurred during the license handling due to an invalid write access.	Verify the project if there is any invalid write access to the license DB. The structure of REG_KEY must not be changed. Please contact the Product Support if necessary.
A090	The Modbus block is not licensed for this CPU. This is a status information. The bit ERROR is not set. The Modbus communication runs without a license as well.	Read the identification string IDENT_CODE for this CPU and order the registration key.
A091	An exception telegram with exception code 1 was received (only if CP is client). The connection is terminated and reestablished.	The communication partner does not support the requested function.
A092	An exception telegram with exception code 2 was received (only if CP is client). An attempt to an invalid or non existing address at the communication partner was made.	Correct LENGTH or START_ADDRESS at the call of the FB.
A093	An exception telegram with exception code 3 was received (only if CP is client).	The communication partner is not able to execute the received telegram (e.g. the requested length is not supported)
A094	An exception telegram with exception code 4 was received (only if CP is client).	The communication partner is in a state in which it is not able to execute a received telegram.
A095	An exception telegram with an unknown exception code was received (only if CP is client).	Check the error message of the communication partner and verify the data with the help of a telegram trace.
A0FF	The connection is not ready for communication.	Check the connections. Correct the value of check_conn_cycle if applicable. This error code is also reported, when a CP is used, which doesn't support AG_CNTRL.
FFFF	The connection is not parameterized.	If the connection should be used, id_x and laddr_x must be parameterized at start-up.

9.3 Diagnostic Messages of included FCs/SFCs

Error messages of included FCs/SFCs		
STATUS (Hex)	Event text	Remedy
7xxx	For detailed information please refer to the online help of SIMATIC Manager.	See online help (SIMATIC manager -> mark block -> key F1 -> Ethernet -> see also -> code evaluation)
8xxx	For detailed information please refer to the online help of SIMATIC Manager.	See online help (SIMATIC manager -> mark block -> key F1 -> Ethernet -> see also -> code evaluation)
80B2	The communication bus connection between the CPU and CP is not established.	This error may occur once after restart of the H-system and can be ignored.
8186	ID parameter invalid This error code is also reported, when MB_REDCL/MB_REDSV is called with different instance DBs in cyclical OB and OB100.	Correct the parameterization. Use the ID of NetPro and work with only 1 instance DB.

9.4 Diagnostic Messages of SFC24

Error messages of SFC24		
STATUS (Hex)	Event text	Remedy
80A1	DB Number = 0 or too large for the CPU	Choose a valid DB number.
80B1	The DB does not exist in the CPU.	All data blocks that are specified in DB_x must be created and copied into the CPU.
80B2	DB UNLINKED	DB must not be created as UNLINKED.

9.5 Diagnostics with Alarm Bits

The Modbus block provides the possibility to detect lost redundancy. This information is displayed in the output parameters RedErrS7, RedErrDev and TotComErr. It is possible to connect these status bits to an alarm block or to another block for further evaluation.

9.5.1 Client Block

Depending on the parameterizations the alarm bits are set as follows:

1. use_all_conn = FALSE, check_conn_cycle = 0ms

The telegrams are transferred via 1 connection. The other projected connections are on standby. There is no cyclical check of the connections.

With these parameter settings a check of the connections and the updating of the alarm bits are only executed, if a telegram is transferred via this connection.
Disturbed or re-established connections on standby can't be recognized.

2. use_all_conn = FALSE, check_conn_cycle > 0s

The telegrams are transferred via 1 connection. The other projected connections are on standby. When the parameterized time "check_conn_cycle" elapses, a cyclical check of all parameterized connections is executed.

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

3. use_all_conn = TRUE, check_conn_cycle > 0ms, 2 projected connections

The telegrams are transferred via 2 projected connections. When the parameterized time "check_conn_cycle" elapses, a cyclical check of the connections is executed.

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

4. use_all_conn = TRUE, check_conn_cycle > 0ms, 4 projected connections

The telegrams are transferred via 4 projected connections. When the parameterized time "check_conn_cycle" elapses, a cyclical check of the connections is executed.

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

9.5.2 Server Block

The time “check_conn_cycle” must be parameterized when using the server block.

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 Application Sample

Sample General Information

With the installation 2 sample projects are stored in \Program Files\Siemens\Step7\Examples.

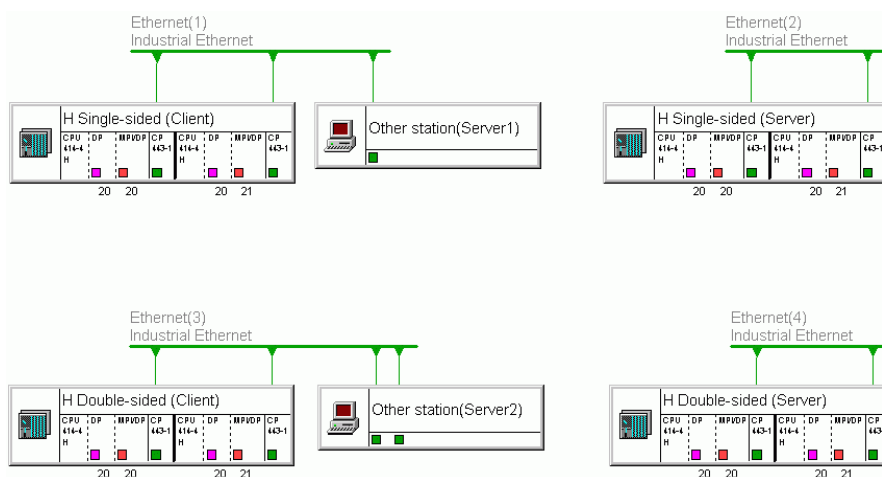
- sample project “MB_TCP_CP_RED_400” written in STL and
- sample project “MB_TCP_CP_RED_CFC” written in CFC.

The S7 programs are for information purposes only and are not to be understood as a solution for a customer specific installation configuration.

Example Project on the CD

On the CD you can find an extensive example project, which offers all varieties of parameterization possibilities for the Simatic stations.

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



Simatic Stations in the Example Project

The example project consists of the following Simatic stations:

Block / Station Name	Single-sided	Double-sided	Client	Server
H Double-sided (Client)		x	x	
H Double-sided (Server)		x		x
H Single-sided (Client)	x		x	
H Single-sided (Server)	x			x

Programming Example

The programming example consists of the blocks:

- Start-Up OB100 with call of FB909 and FB907 respectively
- OB121 programming error
- Cyclic program processing OB1 with call of FB909 and FB907 respectively
- Global DBs for job trigger e.g. with variable table and for licensing
- Data blocks for register values and bit values

10.1 Example project in AWL – Modbus Client

Overview

Object name	Symbolic name	Create...	Size i...	Type	Version
System data	---	---	---	SDB	---
OB1	CYCL_EXC	STL	714	Organization Block	0.1
OB72	RED_FLT	STL	38	Organization Block	0.1
OB100	COMPLETE RESTART	STL	364	Organization Block	0.1
OB121	PROG_ERR	STL	38	Organization Block	0.1
FB908	MB_CPCLI	SCL	9828	Function Block	2.3
FB909	MB_REDCL	SCL	14594	Function Block	2.4
FC10	AG_CNTRL	STL	1610	Function	1.0
FC50	AG_LSEND	STL	846	Function	3.1
FC60	AG_LRECV	STL	992	Function	3.1
DB1	CONTROL_DAT	DB	102	Data Block	0.1
DB3	LICENSE_DB	DB	56	Data Block	0.1
DB11	Holding Register Area 1	DB	1036	Data Block	0.1
DB12	Holding Register Area 2	DB	398	Data Block	0.1
DB13	Coils Area 1	DB	114	Data Block	0.1
DB909	IDB_Modbus_Client	DB	2084	Instance data block ...	0.0
Client_job	Client_job	---	---	Variable Table	0.1
SFB4	TON	STL	---	System function block	1.0

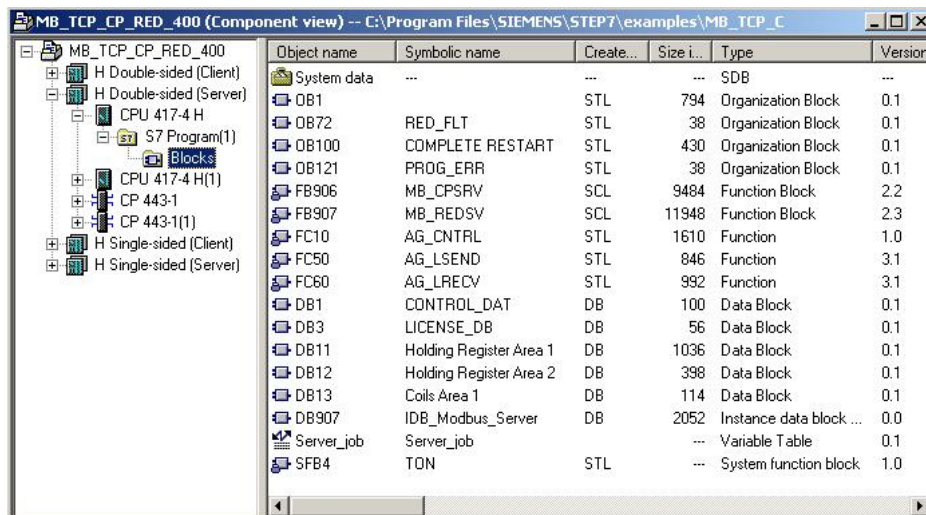
Used Blocks

This block numbers are also used in the provided example project for S7 stations with FB MB_REDCL.

Block	Symbol	Comment
OB 1	CYCL_EXC	Cyclic program processing
OB 100	COMPLETE RESTART	Start-Up OB for Re-Start
OB 121	PROG_ERR	Programming error OB
FB 909	MB_REDCL	FB MB_REDCL
FB 908	MB_CPCLI	Internal called FB MB_CPCLI
DB 1	CONTROL_DAT	Work-DB CONTROL_DAT for FB MB_REDCL
DB 3	LICENSE_DB	License-DB for FB MB_REDCL
DB 11	DATA_AREA_1	DB for memory area 1
DB 12	DATA_AREA_2	DB for memory area 2
DB 13	DATA_AREA_3	DB for memory area 3
DB 909	IDB_MODBUS	Instance-DB for FB MB_REDCL

10.2 Example project in AWL – Modbus Server

Overview



Used Blocks

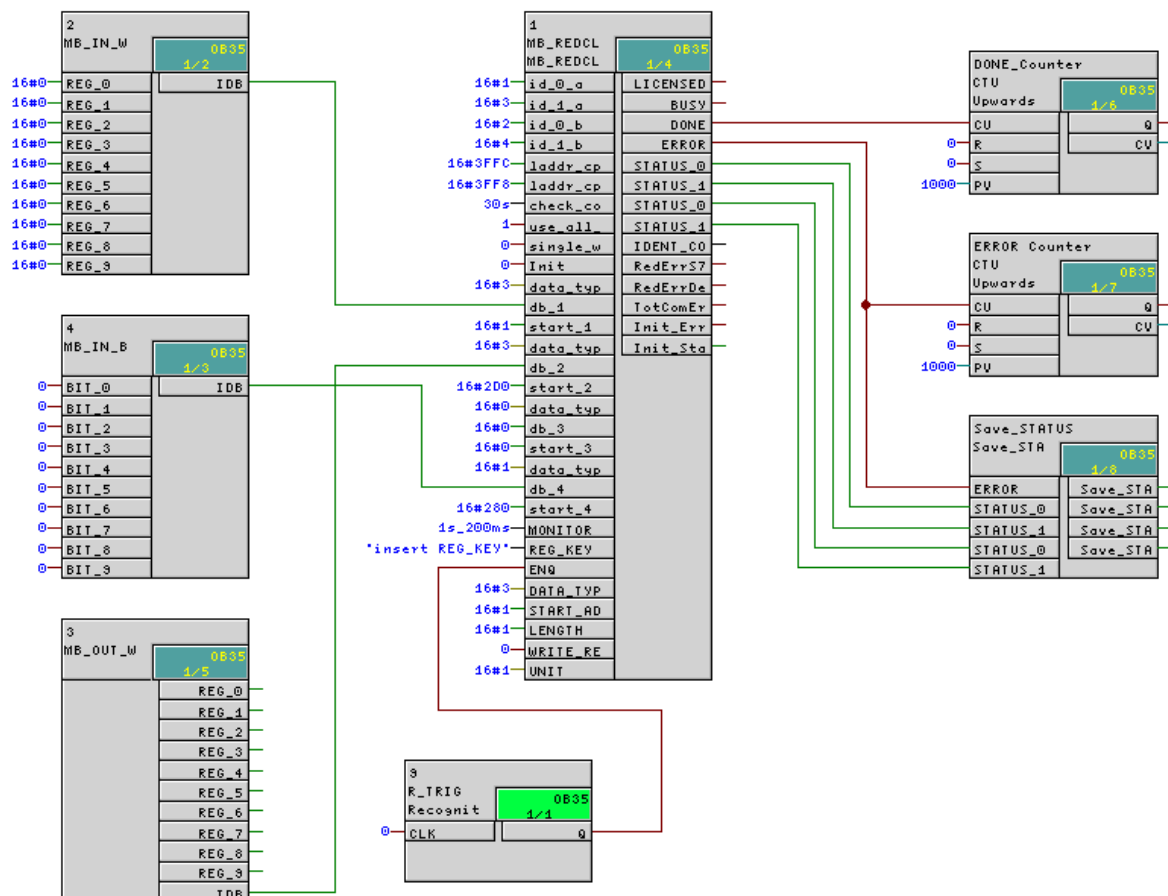
This block numbers are also used in the provided example project for S7 stations with FB MB_REDCL

Block	Symbol	Comment
OB 1	CYCL_EXC	Cyclic program processing
OB 100	COMPLETE RESTART	Start-Up OB for Re-Start
OB 121	PROG_ERR	Programming error OB
FB 907	MB_REDSV	FB MB_REDSV
FB 906	MB_CPSRV	Internal called FB MB_CPSRV
DB 1	CONTROL_DAT	Work-DB CONTROL_DAT for FB MB_REDSV
DB 3	LICENSE_DB	License-DB for FB MB_REDSV
DB 11	DATA_AREA_1	Register -DB for memory area 1
DB 12	DATA_AREA_2	Register -DB for memory area 2
DB 13	DATA_AREA_3	Register -DB for memory area 3
DB 907	IDB_MODBUS	Instance-DB for FB MB_REDSV

10.3 Example project in CFC – Modbus Client

Overview

The example was created with CFC V7.1 SP2.



Used Blocks

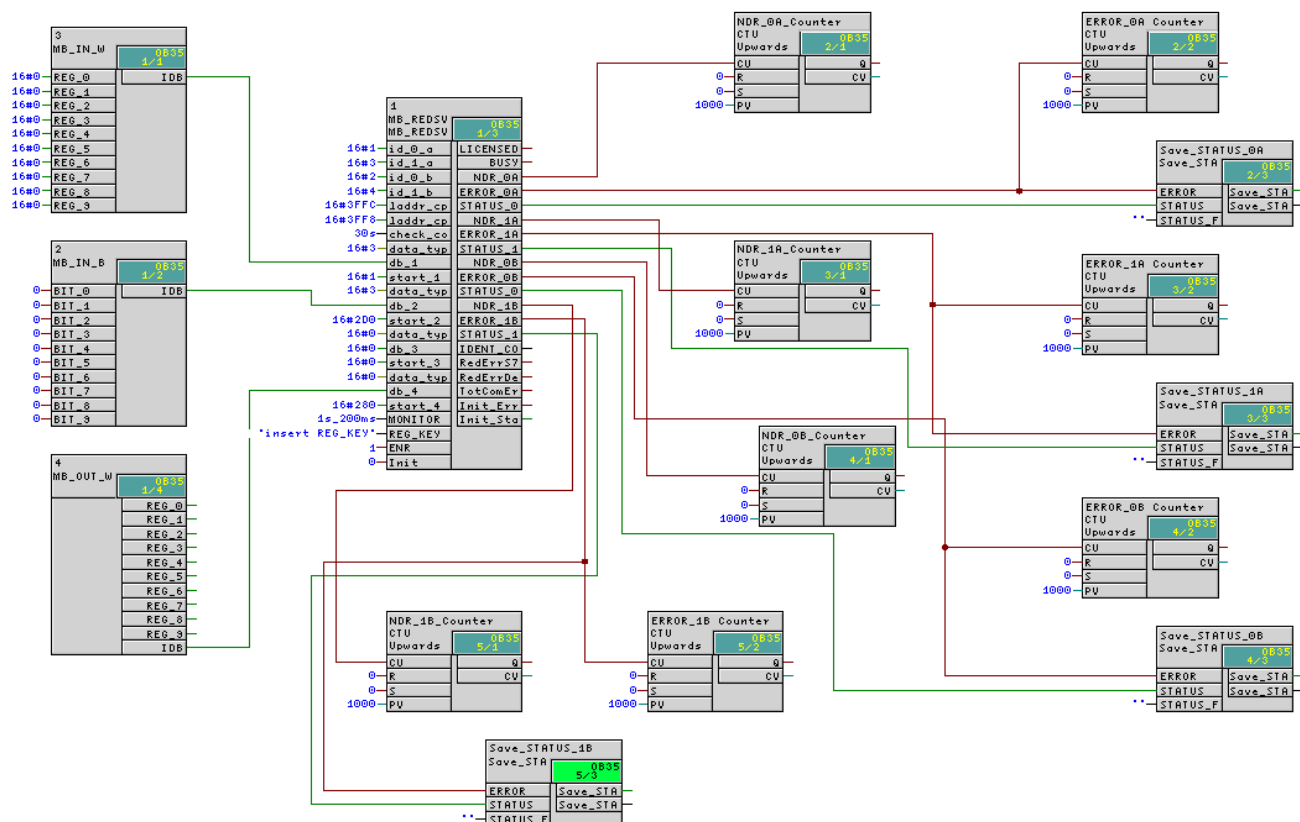
This block numbers are also used in the provided example project for S7 stations with FB MB_REDCL.

Block	Symbol	Comment
OB 35	CYCL_EXC	Cyclic program processing
OB 100	COMPLETE RESTART	Start-Up OB for Re-Start
OB 121	PROG_ERR	Programming error OB
FB 909	MB_REDCL	FB MB_REDCL
FB 908	MB_CPCLI	Internal called FB MB_CPCLI
FB 900	MB_IN_R	Data collector-FB for inputs of real
FB 903	MB_OUT_W	Data collector-FB for outputs of word
FB 904	MB_IN_B	Data collector-FB for inputs of bool
DB xy		DBs, generated by CFC

10.4 Example project in CFC – Modbus Server

Overview

The example was created with CFC V7.1 SP2.



Used Blocks

This block numbers are also used in the provided example project for S7 stations with FB MB_REDSV.

Block	Symbol	Comment
OB 35	CYCL_EXC	Cyclic program processing
OB 100	COMPLETE RESTART	Start-Up OB for Re-Start
OB 121	PROG_ERR	Programming error OB
FB 907	MB_REDSV	FB MB_REDSV
FB 906	MB_CPSRV	Internal called FB MB_CPSRV
FB 900	MB_IN_R	Data collector-FB for inputs of real
FB 903	MB_OUT_W	Data collector-FB for outputs of word
FB 904	MB_IN_B	Data collector-FB for inputs of bool
DB xy		DBs, generated by CFC

A Literature

**The MODBUS
Organization**

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

<http://www.modbus.org>

Customer Support

Siemens AG
Industry Sector
Industry Automation Division / Industrial Automation Systems
Factory Automation
I IA AS FA

Phone: +49 (0)911 895 7 222

[Customer Support](#)

<http://www.siemens.com/s7modbus>

Siemens Aktiengesellschaft

Subject to change without notice

release: 08/2014