

SIEMENS

SIMATIC

SIMATIC Modbus/TCP redundant communication via the integrated PN interface of H-CPU

Manual

SIMATIC S7

SIMATIC Modbus/TCP redundant communication via the integrated PN interface of H-CPUs

Manual

Preface, Table of Contents

Product description	1
Getting started	2
Commissioning	3
Parameter assignment	4
Licensing	5
Redundancy	6
FB MB_PNHCL	7
FB MB_PNHSV	8
Additional blocks	9
Use in a single PN CPU	10
Diagnostics	11
Application example	12
Appendices	
References	
Glossary	

Safety instructions

This manual contains information which you must observe in order to ensure your own personal safety and avoid material damage. This information is highlighted in the manual by a warning triangle and marked as follows according to the level of danger:



Danger

indicates that death, serious personal injury or substantial property damage **will** result if proper precautions are not taken.



Warning

indicates that death, serious personal injury or substantial property damage **may** result if proper precautions are not taken.



Caution

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

Note

draws your attention to particularly important information on the product or on handling the product, or to a particular part of the documentation.

Qualified personnel

A device may only be commissioned and operated by **qualified personnel**. Qualified personnel for the purposes of the safety instructions contained in this manual are persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Intended use

Please note the following:



Warning

The device may only be used for the applications specified in the catalog and technical description, and only in conjunction with non-Siemens equipment and components if these have been specifically recommended or approved by Siemens.

Trouble-free and safe operation requires proper transport, storage, installation and assembly, as well as careful use and maintenance.

Trademarks

SIMATIC® is a registered trademark of SIEMENS AG.

Any other names appearing in this document may be trademarks, the use of which by third parties for their own purposes may breach owners' rights.

Copyright © Siemens AG 2013 All Rights Reserved

The reproduction, transmission, or use of this document or its contents is not permitted without express written authorization. Parties breaching this provision shall be liable for damages. All rights reserved, in particular for patent and utility model registration.

Siemens AG
Industry Sector
Industry Automation Division / Industrial Automation Systems
Factory Automation
I IA AS FA DH FTH 6
P.O. Box 23 55, D- 90713 Fürth, Germany

Disclaimer

We have checked that the contents of this document correspond to the hardware and software described. Nevertheless, we cannot assume responsibility for any deviations that may arise. The information in this publication is checked at regular intervals and necessary corrections included in subsequent editions. Suggestions for improvement are welcome.

This document is subject to technical changes without prior notice.

Preface

Purpose of this manual

The information in this manual will enable you to set up and put in operation the connection between an H-CPU with an integrated PN interface and a device supporting Modbus/TCP protocol.

Contents of this manual

This manual details the function of the Modbus function block and its parameter assignment.

The manual covers the following:

- Product description
- Getting started
- Commissioning
- Assigning Modbus communication parameters
- Licensing
- Function blocks
- Additional blocks
- Diagnostics
- Application example

Scope of the manual

This manual applies to the following software:

Product	Identification number	From version
Modbus/TCP PN CPU redundant	6AV6676-6MB10-0AX0	1.0.1
FB 913 "TCP_COMM"		3.2
FB 914 "MOD_CLI"		1.6
FB 915 "MB_PNHCL"		1.0
FB 916 "MOD_SERV"		1.5
FB 917 "MB_PNHSV"		1.0

Note

This manual describes the FB version valid at the time the manual was issued.

Additional information	<p>For all other information on PN H-CPU's (installation, commissioning, etc.), refer to the following manual:</p> <p>SIEMENS SIMATIC High-availability Systems S7-400H System manual A5E00267693-11</p> <p>SIEMENS SIMATIC S7-400 Automation System S7-400 CPU Data Manual A5E00850745-10</p> <p>For additional information on STEP7, refer to the following manuals:</p> <p>SIEMENS SIMATIC Software Basic 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>
Questions	<p>If you have any questions on the use of the FBs described in this manual and do not find the answers in this document, contact the local Siemens representative who provided you with this function block.</p>
Conventions	<p>This documentation uses the generic term CPU. The information applies to H-CPU's with an integrated PN interface.</p>
Area of application	<p>The function blocks detailed in this manual connect PN H-CPU's to non-Siemens Modbus devices.</p>

Table of contents

1	Product description	1-1
1.1	Applications.....	1-1
1.2	Hardware and software requirements.....	1-2
2	Getting started.....	2-1
3	Commissioning	3-1
3.1	Installing the library in the STEP7 PG/PC.....	3-1
3.2	Assigning a CPU IP address.....	3-2
3.3	Inserting the function blocks into the program	3-4
3.4	Multiple connections to port 502	3-6
4	Assigning Modbus communication parameters	4-1
4.1	Parameter assignment with the wizard.....	4-3
4.2	Manual parameter assignment	4-4
5	Licensing.....	5-1
6	Redundancy.....	6-1
6.1	Configuration of redundant communication.....	6-1
6.2	Single-sided redundancy	6-3
6.3	Double-sided redundancy	6-5
7	MB_PNHCL function block – Modbus client	7-1
7.1	How the MB_PNHCL FB works	7-1
7.2	Parameters of the function block MB_PNHCL	7-8
7.3	Example of address mapping	7-16
7.4	Data and standard functions used by the FB.....	7-19
7.5	Renaming / rewiring functions and function blocks.....	7-21
8	MB_PNHSV function block – Modbus server.....	8-1
8.1	How the MB_PNHSV FB works	8-1
8.2	Parameters of the MB_PNHSV function block	8-5

8.3	Example of address mapping	8-10
8.4	Data and standard functions used by the FB.....	8-13
8.5	Renaming / rewiring functions and function blocks.....	8-14
9	Additional blocks	9-1
9.1	Support in CFC	9-1
9.2	Job list for cyclic telegrams	9-2
10	Use in a single PN CPU.....	10-1
11	Diagnostics	11-1
11.1	Diagnostics via the display elements of the CPU	11-1
11.2	MB_PNHCL and MB_PNHSV FB diagnostics messages	11-2
11.3	Diagnostics messages of integrated blocks	11-8
11.4	SFC24 diagnostics messages	11-8
11.5	Diagnostic messages with alarm bits.....	11-9
11.5.1	Client block	11-9
11.5.2	Server block.....	11-12
12	Application example	12-1
12.1	Sample project in STL – Modbus client	12-2
12.2	Sample project in STL – Modbus server.....	12-3
12.3	Sample project in CFC – Modbus client	12-4
12.4	Sample project in CFC – Modbus server.....	12-5
A	References	A

1 Product description

1.1 Applications

Classification in the system environment

This function block is a software product for CPUs with an integrated PN interface in a SIMATIC S7 H system.

FB function

The function blocks enable communication between an S7 H-CPU with an integrated PN interface and a device supporting Modbus/TCP. Function codes 1, 2, 3, 4, 5, 6, 15 and 16 are supported.

Data transmission is carried out in accordance with the client-server principle.

The SIMATIC S7 can be operated as a client or as a server during data transmission.

Redundant communication is supported. Use in both an S7-400H system and in an S7 Single PN CPU is possible.

The blocks operate in hot standby mode. Hot standby is the term for parallel redundant signal processing in redundant components. This enables the system as a whole to switch bumpless to the standby components.

Using port number 502

Modbus/TCP usually runs via port 502. This port number is only an option for PN CPUs with the corresponding firmware version. Information on port number release can be found here:
<http://support.automation.siemens.com/WW/view/en/34010717>.

Certain types of CPU can maintain and operate connections to multiple clients via local port 502.

The technical details are set out in section "Multiple connections to Port 502".

1.2 Hardware and software requirements

Modules suitable for MB_PNHCL and MB_PNHSV

The current hardware requirements can be found here: www.siemens.com/s7modbus.

Software versions

The MB_PNHCL or MB_PNHSV FB can be used as of **STEP 7 Version 5.5 SP2 HF1**.

Memory space required

The MB_PNHCL FB requires 17 KB of working memory and 20 KB of load memory.

The MOD_CLI FB requires 10 KB of working and load memory.

The MB_PNHSV FB requires 14 KB of working memory and 17 KB of load memory.

The MOD_SERV FB requires 10 KB of working and load memory.

The MOD_COMM FB requires 2 KB of working and load memory.

You can calculate the exact block lengths using the block properties in SIMATIC Manager.

2 Getting started

Procedure

1. Install "SIMATIC Modbus/TCP PN CPU redundant" and add Modbus blocks to the user project
=> Section 3.1
2. Assign MODBUS_HPAM_PN parameter DB parameters in accordance with requirements (client/server, connection establishment upon restart, Modbus registers, DB areas, etc.)
=> Section 4
3. For **Modbus client**: Call and configure Modbus block MB_PNHCL in the necessary OBs
=> Sections 7.1 and 7.2

or:

For **Modbus server**: Call and configure Modbus block MB_PNHSV in the necessary OBs
=> Sections 8.1 and 8.2
4. Load the user program to the CPU and license the Modbus block for the CPU
=> Section 5

3 Commissioning

General information	The information below on STEP 7 and configuring communication connections relates to STEP 7 Version 5.5 SP2 HF1. Procedures, names and directory information can differ in later versions.
Requirements	Basic knowledge of STEP 7, knowledge of STL, basic knowledge of PLC

3.1 Installing the library in the STEP7 PG/PC

Product package	The enclosed CD contains a setup program for installing the " Modbus_PN_CPU_Red " library, the example projects, and the German and English manuals in the corresponding STEP 7 directories. The CD also contains the manuals in PDF format.
------------------------	--

Requirements	Before installation, STEP7 V5.5 configuration software must first have been installed.
---------------------	---

Installation	<p>Insert the Modbus CD into the CD-ROM drive of your programming device / PC. If the setup program does not start automatically, install as follows:</p> <ol style="list-style-type: none"> 1. In Windows Explorer, select the CD-ROM drive, open the Setup directory and launch the setup program. 2. Follow the on-screen step-by-step instructions of the installation program. <p>You can now find:</p> <ul style="list-style-type: none"> • The libraries under \Program Files\Siemens\Step7\S7libs, • The example projects under \Program Files\Siemens\Step7\Examples, • The manual under \Program Files\Siemens\Step7\S7manual\S7Comm. • The software registration form under \Program Files\Siemens\Step7\S7libs\ Modbus_PN_CPU_Red.
---------------------	--

The first time you call the Modbus library, use the "Browse" function in the Open dialog to access the library in S7libs.

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

3.2 Assigning a CPU IP address

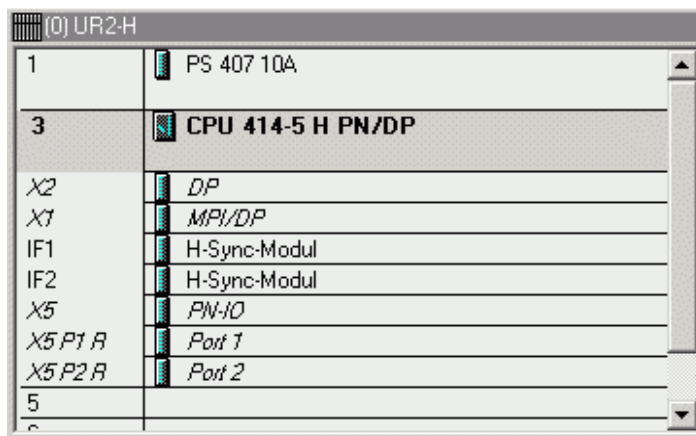
Introduction

This example of IP address assignment uses a CPU 414-5H PN/DP.

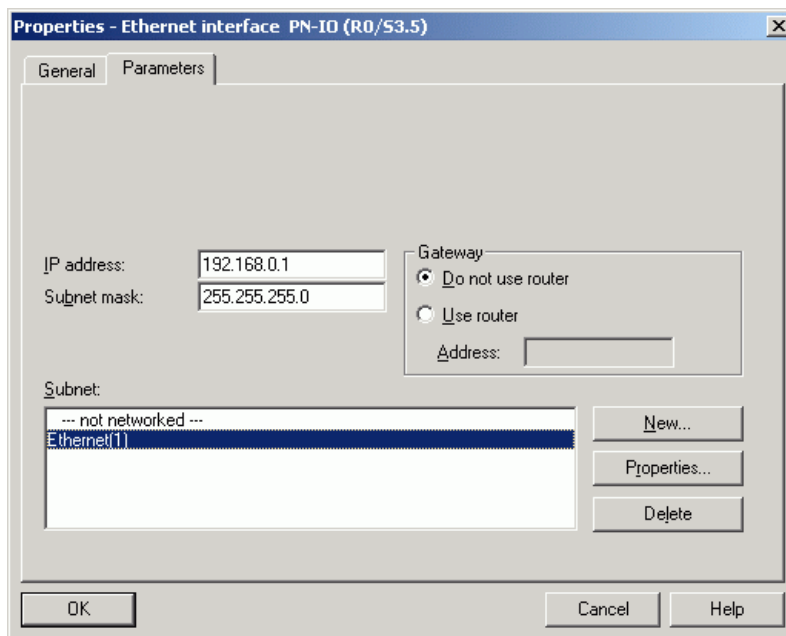
Procedure

Before configuration, you must first generate an **S7 project** with STEP7.

1. Open HW Config and insert the racks and power supplies. Add a CPU 414-5H PN/DP after the power supply.



2. The properties dialog box of the PN-IO interface X5 is displayed.



3. Enter the IP address and the subnet mask.
If you are establishing a connection via a router, you must also enter the address of the router.

4. Click the "New..." button and assign a name for a new Industrial Ethernet subnet. Confirm with "OK".
Result: You have now created a new Industrial Ethernet subnet.
5. Click the "OK" button.
Result: The properties dialog box of the CPU 414-5H PN/DP PN-IO interface closes.
6. Insert a CPU 414-5H PN/DP into the second rack as well. This CPU will automatically be assigned the next consecutive IP address.

3.3 Inserting the function blocks into the program

Modbus library content

The Modbus library contains the "S7 Client" and "S7 Server" folders with the FBs for redundant communication.

S7 client

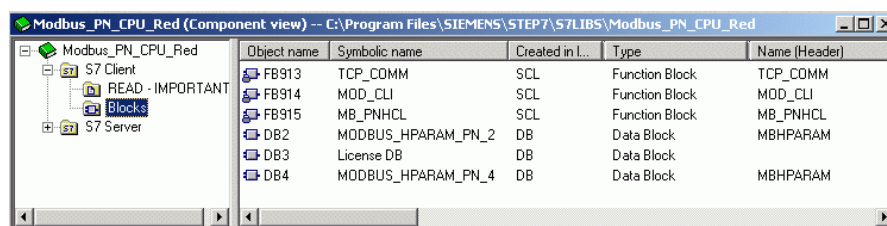
The "S7 Client" folder includes the blocks

- FB915 (MB_PNHCL),
- FB914 (MOD_CLI) and
- FB913 (TCP_COMM).

All 3 blocks are always needed for redundant communication.

The MB_PNHCL block calls the MOD_CLI block internally multiple times, and this calls TCP_COMM.

The library also contains a MODBUS_HPARAM_PN_2 parameter data block for single-sided redundancy, a MODBUS_HPARAM_PN_4 parameter data block for double-sided redundancy and the license DB as a template. You can also copy this to your project to facilitate processing.



Object name	Symbolic name	Created in l...	Type	Name (Header)
FB913	TCP_COMM	SCL	Function Block	TCP_COMM
FB914	MOD_CLI	SCL	Function Block	MOD_CLI
FB915	MB_PNHCL	SCL	Function Block	MB_PNHCL
DB2	MODBUS_HPARAM_PN_2	DB	Data Block	MBHPARAM
DB3	License DB	DB	Data Block	MBHPARAM
DB4	MODBUS_HPARAM_PN_4	DB	Data Block	MBHPARAM

S7 server

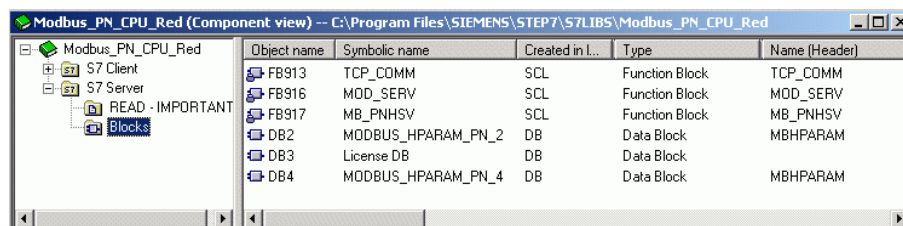
The "S7 Server" folder includes the blocks

- FB917 (MB_PNHVS),
- FB916 (MOD_SERV) and
- FB913 (TCP_COMM).

All 3 blocks are always needed for redundant communication. The

MB_PNHVS block calls the MOD_SERV block internally multiple times, and it calls TCP_COMM.

The library also contains a MODBUS_HPARAM_PN_2 parameter data block for single-sided redundancy, a MODBUS_HPARAM_PN_4 parameter data block for double-sided redundancy, and the license DB as a template. You can also copy this to your project to facilitate processing.



Object name	Symbolic name	Created in l...	Type	Name (Header)
FB913	TCP_COMM	SCL	Function Block	TCP_COMM
FB916	MOD_SERV	SCL	Function Block	MOD_SERV
FB917	MB_PNHVS	SCL	Function Block	MB_PNHVS
DB2	MODBUS_HPARAM_PN_2	DB	Data Block	MBHPARAM
DB3	License DB	DB	Data Block	MBHPARAM
DB4	MODBUS_HPARAM_PN_4	DB	Data Block	MBHPARAM

Blocks in the standard library

The following FBs are required for Modbus communication:

- TSEND (FB63)
- TRCV (FB64)
- TCON (FB65)
- TDISCON (FB66).

These communication blocks can be found in the "**Standard Library → Communication Blocks**" library and must also be inserted into your project.

Please note that the following versions of the FBs are required for the smooth operation of the MB_PNHCL and MB_PNHSV FBs:

TSEND	V2.1
TRCV	V2.2
TCON	V2.4
TDISCON	V2.1

3.4 Multiple connections to port 502

General information

Some CPUs can multiplex TCP connections. In these cases, multiple MODBUS clients can connect to port 502 of the CPU (multiport). The CPU acts as the MODBUS server.

Information on which CPUs with which firmware versions allow multiple port 502 use is available here:

www.siemens.com/s7modbus

Requirements

For this function to be available, the following settings must be made in block selection and parameter assignment:

- CPU is the server
- Port 502 is the local port
- Unspecified TCP connection
- Passive connection establishment

Number of connections enabled

The number of connections that a CPU can accept at port 502 depends on the module. Please see the technical data of the CPU for details.

Configuration

One unique connection in the parameter DB is required for each client that is to connect to port 502 of the server.

4 Assigning Modbus communication parameters

General information

Communication via the integrated PN interface of the CPU does not require connection configuration in NetPro. The connections are established and terminated using the TCON (FB65) and TDISCON (FB66) function blocks.

Multiple connections to different communication partners can be configured and established at the same time. The number of connections established simultaneously depends on the CPU.

The MODBUS_HP PARAM connection data block

The data required for establishing connections and processing Modbus telegrams is defined in a data block: the parameter data block MODBUS_HP_PARAM_PN. At first the connection parameters are defined, subsequently, the Modbus parameters are defined

Each connection, 0A, 1A, 0B and 1B, requires **1 connection block** in which the connection parameters between the communication partners are defined. Two connection blocks are created for single-sided redundancy and four connection blocks for double-sided redundancy. Once the connection blocks are defined, the Modbus parameters are specified.

One predefined parameter data block each for single-sided and double-sided redundancy can be found as an example in the "**Modbus_PN_CPU_Red**" library.

Setup of DB MODBUS_HP_PARAM_PN with single-sided connection:

Address	Name
0.0	FALSE: Single-sided connection
2.0	STRUCT
	Connection 0A: Connection parameters
	END_STRUCT
66.0	STRUCT
	Connection 1A: Connection parameters
	END_STRUCT
130.0	Modbus parameters

Setup of DB MODBUS_HPAM_PN with double-sided connection:

Address	Name
0.0	TRUE: Double-sided connection
2.0	STRUCT
	Connection 0A: Connection parameters
	END_STRUCT
66.0	STRUCT
	Connection 1A: Connection parameters
	END_STRUCT
130.0	STRUCT
	Connection 0B: Connection parameters
	END_STRUCT
194.0	STRUCT
	Connection 1B: Connection parameters
	END_STRUCT
258.0	Modbus parameters

Connection parameters

The connection-specific parameters such as the hardware interface used locally and the IP address of the communication partner are defined in the connection blocks. These parameters enable the TCON and TDISCON functions to establish and terminate a connection. See section 4.2 for details.

You must comply with the data structure of the connection parameter block or no connection will be established.

Modbus parameters

The data required for the mode and address reference such as the Modbus areas mapped in the data blocks, and the S7 mode - Modbus server or Modbus client - is stored in the Modbus parameters. You must comply with the data structure of the Modbus parameters or error-free processing will not be possible.

Configuration options

There are two possible ways to configure the connection and Modbus parameters. You can enter the information using a wizard for quick and easy configuration. Alternatively, you can set the parameters by editing the structure in the parameter data block.

These two options are detailed in sections 4.1 and 4.2 below.

No further parameters must be saved in the parameter datablock.

4.1 Parameter assignment with the wizard

General information

The "**Modbus/TCP PN Red Wizard**" offers a simple and easy way to configure the connections and the Modbus parameters in the MODBUS_HPAM_PN parameter data block. This method creates the complete data block (connection parameters and Modbus parameters).

We recommend using the wizard for MODBUS_HPAM_PN parameter assignment.

You will find the wizard at

<http://support.automation.siemens.com/WW/view/en/2077896767>.

4.2 Manual parameter assignment

Procedure

Copy DB2 for single-sided redundancy or DB5 for double-sided redundancy for the client or server from the "Modbus_PN_CPU_Red" library and add it to your project. If the number is already in use elsewhere, the DB can be renamed. In this example, DB2 for single-sided redundancy is used.

The parameters in the MODBUS_HPARAM_PN block must not be changed during runtime. If the parameters are changed, the CPU must be restarted with STOP -> RUN.

Connection parameter setup and modifications

One block is required for each connection.

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	double_sided_red	BOOL	FALSE
+2.0	connection_0A	STRUCT	
+0.0	block_length	WORD	W#16#40
+2.0	id	WORD	W#16#1
+4.0	connection_type	BYTE	B#16#11
+5.0	active_est	BOOL	FALSE
+6.0	local_device_id	BYTE	B#16#5
+7.0	local_tsap_id_len	BYTE	B#16#2
+8.0	rem_subnet_id_len	BYTE	B#16#0
+9.0	rem_staddr_len	BYTE	B#16#0
+10.0	rem_tsap_id_len	BYTE	B#16#0
+11.0	next_staddr_len	BYTE	B#16#0
+12.0	local_tsap_id	ARRAY[1..16]	B#16#1, B#16#F6, B#16#0, B#16#0
*1.0		BYTE	
+28.0	rem_subnet_id	ARRAY[1..6]	B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0
*1.0		BYTE	
+34.0	rem_staddr	ARRAY[1..6]	B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0
*1.0		BYTE	
+40.0	rem_tsap_id	ARRAY[1..16]	B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0
*1.0		BYTE	
+56.0	next_staddr	ARRAY[1..6]	B#16#0, B#16#0, B#16#0, B#16#0, B#16#0, B#16#0
*1.0		BYTE	
+62.0	spare	WORD	W#16#0
=64.0		END_STRUCT	

block_length

This parameter defines the length of the connection parameters and must not be changed.

Fixed value: W#16#40

id

A connection ID is issued for each logical connection. This ID must be unique across the CPU when T-communication is used. The ID is specified when the FB MB_PNHCL or MB_PNHSV is called and is used for internal calls of the T-blocks (TCON, TSEND, TRCV and TDISCON).

Value range: W#16#1 to W#16#FFF

connection_type This is where you specify the connection type for connection establishment by the TCON function. The value to set depends on the CPU.

TCP (compatibility mode):	B#16#01 for CPU 315 or 317 <= FW V2.3
TCP:	B#16#11 for CPU 315 or 317 >= FW V2.4, IM 151-8 PN/DP CPU, CPU314C, CPU319, CPU412(H), CPU414(H), CPU416(H), CPU417(H)

These specifications data may vary depending on the firmware used.

For additional information, refer to:

<http://support.automation.siemens.com/WW/view/en/24294554>

active_est This parameter specifies the type of connection establishment, active or passive. Active connection establishment is done by the Modbus client, and passive connection establishment by the Modbus server.

Active connection establishment:	TRUE
Passive connection establishment:	FALSE

local_device_id The *local_device_id* defines the IE interface of the PN CPU used. Different settings are required depending on the PN CPU type.

IM 151-8 PN/DP CPU:	B#16#1
CPU 314C, 315 or 317:	B#16#2
CPU 319:	B#16#3
CPU 412(H), 414(H), 416(H) and 417(H)	B#16#5
CPU in rack 1 of the H station	B#16#15

In H stations:

The S7400 PN CPU in rack 0 communicates via *local_device_id* = 5 and the CPU in rack 1 communicates via *local_device_id* = 15_{Hex}.

local_tsap_id_len The length of the *local_tsap_id* parameter (= local port number) is specified here.

Active connection establishment:	0
Passive connection establishment:	2

rem_subnet_id_len This parameter is not used currently and must be assigned B#16#0.

rem_staddr_len The length of the *rem-staddr* parameter. i.e. the IP address of the communication partner, is specified here. No IP address for the partner is specified if communication is to take place with an unspecified connection.

Unspecified connection:	B#16#0
Specified connection:	B#16#4

rem_tsap_id_len This parameter defines the length of the *rem_tsap_id* parameter, the port number of the remote communication partner.

Active connection establishment:	2
Passive connection establishment:	0

next_staddr_len The length of the *next_staddr* parameter is specified here.

For PN interfaces: B#16#0

local_tsap_id This parameter is used to set the local port number. The representation depends on the parameter *connection_type*. The value range depends on the CPU. The port number must be unique within the CPU.

For connection_type B#16#01

local_tsap_id[1] low byte of port no. in hex display
 local_tsap_id[2] high byte of port no. in hex display
 local_tsap_id[3-16] B#16#00

For connection_type B#16#11

local_tsap_id[1] high byte of port no. in hex display
 local_tsap_id[2] low byte of the port no. in hex display
 local_tsap_id[3-16] B#16#00

rem_subnet_id This parameter is not used currently and must be assigned 0.

rem_staddr This byte array is where the IP address of the remote communication partner is entered. No IP address is entered for unspecified connections. The representation depends on the parameter *connection_type*.

Example: IP address 192.168.0.1:

For connection_type B#16#01

rem_staddr[1] = B#16#01 (1),
 rem_staddr[2] = B#16#00 (0),
 rem_staddr[3] = B#16#A8 (168),
 rem_staddr[4] = B#16#C0 (192),
 rem_staddr[5-6]= B#16#00 (reserved)

For connection_type B#16#11

rem_staddr[1] = B#16#C0 (192),
 rem_staddr[2] = B#16#A8 (168),
 rem_staddr[3] = B#16#00 (0),
 rem_staddr[4] = B#16#01 (1),
 rem_staddr[5-6]= B#16#00 (reserved)

rem_tsap_id This parameter is used to set the remote port number. The type of representation depends on the parameter *connection_type*. The value range depends on the CPU.

For connection_type B#16#01

local_tsap_id[1] low byte of port no. in hex display
 local_tsap_id[2] high byte of port no. in hex display
 local_tsap_id[3-16] B#16#00

For connection_type B#16#11

local_tsap_id[1] high byte of port no. in hex display
 local_tsap_id[2] low byte of port no. in hex display
 local_tsap_id[3-16] B#16#00

next_staddr This parameter defines the rack and slot number of the CP used. If the CPU's integrated PN interface is being used, this parameter must be set to 0.

next_staddr[1-6] B#16#00

spare This parameter is not used and must be preset to 0.

Adapting Modbus parameters

The Modbus parameters in the MODBUS_HPARAM_PN block are used to define the mode of Modbus communication and how Modbus addresses are mapped to SIMATIC addresses.

+130.0	server_client	BOOL	TRUE
+130.1	single_write	BOOL	FALSE
+130.2	conn_at_startup	BOOL	FALSE
+131.0	reserved	BYTE	B#16#0
+132.0	data_type_1	BYTE	B#16#3
+134.0	db_1	WORD	W#16#B
+136.0	start_1	WORD	W#16#0
+138.0	end_1	WORD	W#16#1F3
+140.0	data_type_2	BYTE	B#16#3
+142.0	db_2	WORD	W#16#C
+144.0	start_2	WORD	W#16#2D0
+146.0	end_2	WORD	W#16#384
+148.0	data_type_3	BYTE	B#16#4
+150.0	db_3	WORD	W#16#D
+152.0	start_3	WORD	W#16#2D0
+154.0	end_3	WORD	W#16#3E8
+156.0	data_type_4	BYTE	B#16#0
+158.0	db_4	WORD	W#16#0
+160.0	start_4	WORD	W#16#0
+162.0	end_4	WORD	W#16#0
+164.0	data_type_5	BYTE	B#16#1
+166.0	db_5	WORD	W#16#E
+168.0	start_5	WORD	W#16#280
+170.0	end_5	WORD	W#16#4E2
+172.0	data_type_6	BYTE	B#16#2
+174.0	db_6	WORD	W#16#F
+176.0	start_6	WORD	W#16#6A4
+178.0	end_6	WORD	W#16#8FC
+180.0	data_type_7	BYTE	B#16#1
+182.0	db_7	WORD	W#16#10
+184.0	start_7	WORD	W#16#6A4
+186.0	end_7	WORD	W#16#8FC
+188.0	data_type_8	BYTE	B#16#0
+190.0	db_8	WORD	W#16#0
+192.0	start_8	WORD	W#16#0
+194.0	end_8	WORD	W#16#0
+196.0	conn_0A_send_buffer	ARRAY[1..260]	B#16#0
*1.0		BYTE	
+456.0	conn_0A_recv_buffer	ARRAY[1..260]	B#16#0
*1.0		BYTE	
+716.0	conn_1A_send_buffer	ARRAY[1..260]	B#16#0
*1.0		BYTE	
+976.0	conn_1A_recv_buffer	ARRAY[1..260]	B#16#0
*1.0		BYTE	
=1236.0		END_STRUCT	

server_client TRUE: S7 is the server, to set when using MB_PNHSV
 FALSE: S7 is the client, to set when using MB_PNHCL

single_write Function codes 5 and 6 are used for write jobs with a length of 1 in the MB_PNHCL block when the parameter single_write = TRUE.
 If single_write = FALSE, function codes 15 and 16 are used for all write jobs.

connect_at_startup This specifies the time when the connection is established.
 If connect_at_startup is set to TRUE, the connection will be – independent of ENR - established as soon as the CPU is restarted. In this case, a job may not be triggered until the connections have been correctly established (ESTAB_x = TRUE) or an error has been displayed at ERROR and STATUS_x. At the latest when CONN_TIMEOUT has elapsed the status outputs are updated.

FALSE: Connection established when ENQ or ENR set
 TRUE: Connection established immediately after restart

8 data areas 8 data areas are available for mapping MODBUS addresses in the S7 memory. At least the first data area must be defined; the other 7 data areas are optional. The system either reads from or writes to the data areas, depending on the triggered job. With one request, only one DB can be accessed. Even if consecutive register numbers or coils are located in two different DBs, two requests are necessary to access them both. This must be taken into account during the parameterization.

It is possible to map more Modbus areas (registers or bit values) to a data block than can be processed with one telegram.

data_type_x The data_type_x parameter specifies which MODBUS data types are mapped in this data block.
 If a value of 0 is entered in data_type_x, the data area will not be 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 db_x parameter defines the data block in which the MODBUS registers or bit values subsequently defined are to be mapped.
 0 cannot be used as a DB number because it is reserved for system functions.

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

The data block must be 2 bytes longer than required for the configured data. The last two bytes are required for internal purposes.

start_x specifies the first Modbus address mapped in data word 0 of the DB.
end_x The *end_x* parameter defines the address of the last MODBUS address.

For register access, the data word number in the S7 DB in which the last Modbus address is entered is calculated according to the following formula:

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

For bit access, the data byte number in the S7 DB in which the last Modbus address is entered is calculated according to the following formula:

$$\text{DBB number} = (\text{end_x} - \text{start_x} + 7) / 8$$

The defined data areas must not overlap. The *end_x* parameter must not be smaller than *start_x*. In the event of an error, FB startup will finished with an error. When *start_x* is equal to *end_x*, 1 Modbus address (1 register or 1 bit value) is assigned.

Sections 7.3 and 8.3 give an example of mapping MODBUS addresses to S7 memory areas.

start_x, end_x

MODBUS address 0 to 65535 (W#16#0000 to W#16#FFFF)

conn_0A_send_buffer This array is used internally for message data within the FB. Accessing or changing the array is inadmissible.

conn_0A_recv_buffer This array is used internally for message data within the FB. Accessing or changing the array is inadmissible.

conn_1A_send_buffer This array is used internally for message data within the FB. Accessing or changing the array is inadmissible.

conn_1A_recv_buffer This array is used internally for message data within the FB. Accessing or changing the array is inadmissible.

conn_0B_send_buffer This array is used internally for message data within the FB. Accessing or changing the array is inadmissible.

conn_0B_recv_buffer This array is used internally for message data within the FB. Accessing or changing the array is inadmissible.

conn_1B_send_buffer This array is used internally for message data within the FB. Accessing or changing the array is inadmissible.

conn_1B_recv_buffer This array is used internally for message data within the FB. Accessing or changing the array is inadmissible.

5 Licensing

General information

The MB_PNHCL / MB_PNHSV blocks must be licensed on each CPU individually.
Licensing is performed in two steps: the IDENT_CODE is read out and the REG_KEY activation code entered. The OB121 must be available in the CPU.

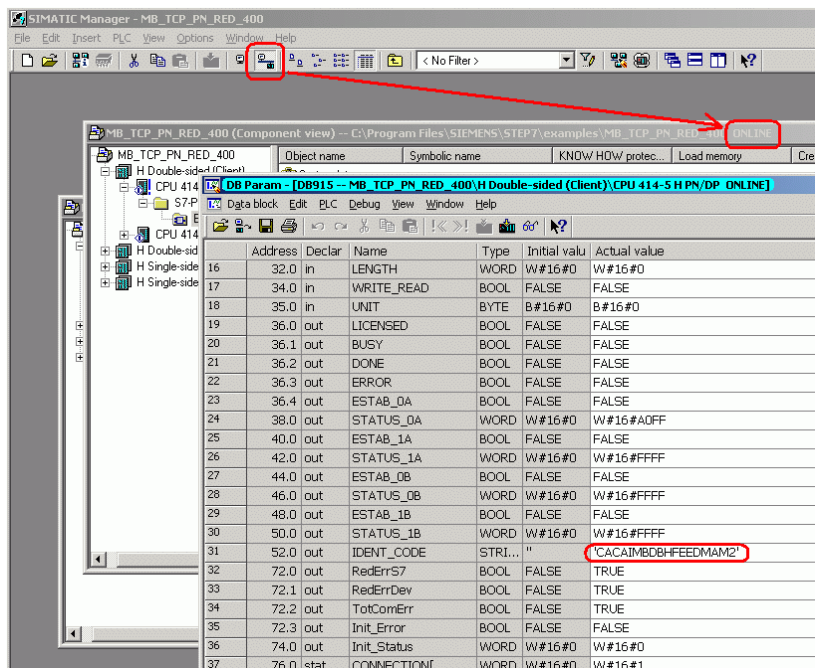
Please note:

In S7 H stations, only the CPU in rack 0 is licensed. The CPU in rack 0 therefore cannot be changed after licensing.

Reading out the IDENT_CODE

Proceed as follows to read out the IDENT_CODE:

1. Assign the parameters for block MB_PNHCL or MB_PNHSV in accordance with your requirements in a cyclic OB (OB1 or cyclic interrupt OB) and in OB100.
Load the program to the CPU and switch the CPU to RUN.
2. Open the project in online mode in SIMATIC Manager. In this online mode, open the instance DB of the Modbus block.



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

Copy this string from the DB to the **SOFTWARE REGISTRATION FORM** using copy and paste. This form is saved during installation at the library path ..\Program Files\Siemens\Step7\S7LIBS\Modbus_PN_CPU_Red and is also available on the installation CD. Enter the license no. of the product packaging on the form.

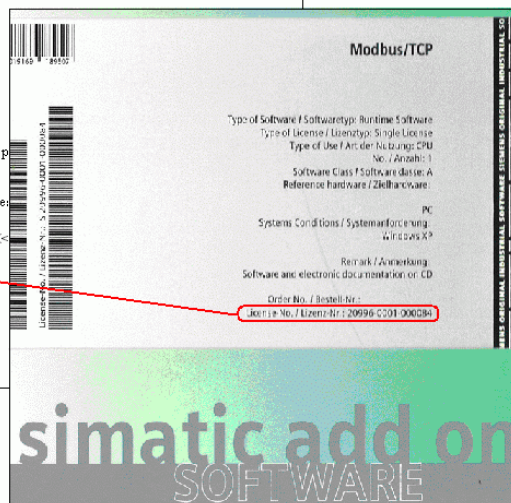
Address	Declar	Name	Type	Initial valu	Actual value	
20	36.1	out	BUSY	BOOL	FALSE	FALSE
21	36.2	out	DONE	BOOL	FALSE	FALSE
22	36.3	out	ERROR	BOOL	FALSE	FALSE
23	36.4	out	ESTAB_OA	BOOL	FALSE	FALSE
24	38.0	out	STATUS_OA	WORD	W#16#0	W#16#ADFF
25	40.0	out	ESTAB_1A	BOOL	FALSE	FALSE
26	42.0	out	STATUS_1A	WORD	W#16#0	W#16#FFFF
27	44.0	out	ESTAB_0B	BOOL	FALSE	FALSE
28	46.0	out	STATUS_0B	WORD	W#16#0	W#16#FFFF
29	48.0	out	ESTAB_1B	BOOL	FALSE	FALSE
30	50.0	out	STATUS_1B	WORD	W#16#0	W#16#FFFF
31	52.0	out	IDENT_CODE	STRI...	"	'CACAIMBDBHFEEDMAM2'
32	72.0	out	RedErrS7	BOOL	FALSE	TRUE
33	72.1	out	RedErrDev	BOOL	FALSE	TRUE
34	72.2	out	TotComErr	BOOL	FALSE	TRUE
35	72.3	out	Init_Error	BOOL	FALSE	FALSE

Please insert the IDENT-CODE here.
The manual contains information how to find out the IDENT-CODE.
Bitte tragen Sie den IDENT-CODE hier ein.
Das Handbuch enthält Informationen, wie Sie den IDENT-CODE ermitteln.

>>> IDENT_CODE <<<

Please insert the License-No. here.
You find the License-No. on the package of the product.
Bitte tragen Sie die Lizenz-Nr. hier ein.
Sie finden die Lizenz-Nr. auf der Verpackung des Produktes.

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



1. Send this form as Service Request (<http://support.automation.siemens.com/WW/view/en/38718979>) to Customer Support.
Hereupon you will receive the registration key for your PLC.
2. Information on use in CFC: The CFC editor can only display a set number of characters online. The complete IDENT_CODE is displayed in the tooltip of the output parameter or in the IDB.

Entering the REG_KEY activation code

The REG_KEY activation code must be declared for each call of the Modbus block.

The REG_KEY should be saved in a global DB. Via this global DB all Modbus blocks can receive the activation code (see also example below).

Process as follows to enter the REG_KEY activation code:

1. Copy the predefined licensing block DB3 from the "Modbus_PN_CPU_Red" library to your project. If the DB number is already in use in the project, the license DB can be renamed.
2. Open the license DB and copy the 17-digit activation code provided to the "Initial value" column using copy and paste. Multiple keys can be entered as a list.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	REG_KEY_1	STRING(17)	'insert REG_KEY'	Registration Key
=20.0		END_STRUCT		

3. The activation code must be permanently saved in the data block so that it does not need to be entered each time the CPU is re-loaded. Switch to the DB data view by selecting "View" -> "Data view". The menu command "Edit" -> "Initialize data block" applies all values from the "Initial value" column to the "Actual value" column.
4. In the cyclic OB, enter the data block number of the license DB at the REG_KEY parameter of the Modbus block.
5. Load the modified blocks to the CPU. The activation code can be entered during runtime; a STOP -> RUN transition is not necessary.

The block is now licensed for this CPU.

No or incorrect licensing

If no activation code is entered or the activation code is wrong, the INTF LED of the H-CPU flash once a minute and an entry is written cyclically to the diagnostics buffer indicating the lack of a license. The error number for no license is W#16#A090.

If you are using a single PN CPU, the LED will flash every 4 seconds and an entry will be made in the diagnostics buffer.

Path: MB_TCP_PN_RED_400\H Double-sided (Cli) Operating mode of the CPU: RUN
Status: OK

No.	Time of day	Date	Event
1	07:56:08.467 AM	09/16/2013	Event ID: 16# A090
2	07:56:08.467 AM	09/16/2013	Area length error when reading
3	07:55:08.438 AM	09/16/2013	Event ID: 16# A090
4	07:55:08.438 AM	09/16/2013	Area length error when reading
5	07:54:08.411 AM	09/16/2013	Event ID: 16# A090
6	07:54:08.411 AM	09/16/2013	Area length error when reading
7	07:53:08.383 AM	09/16/2013	Event ID: 16# A090
8	07:53:08.383 AM	09/16/2013	Area length error when reading

Details on Event: 1 of 79 Event ID: 16# A090

No entry in text database. Hexadecimal values will be displayed.
 Event ID: 16# A090
 OB: 16# 01
 PK: 16# 01
 DatID 1 / 2: 16# 50 C0
 Additional info 1 / 2 / 3: 16# 4D4F 4442 5553



Warning

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

If no or the wrong activation code is entered, Modbus communication will be processed but W#16#A090 "No valid license" will always be displayed at the STATUS_x outputs.

6 Redundancy

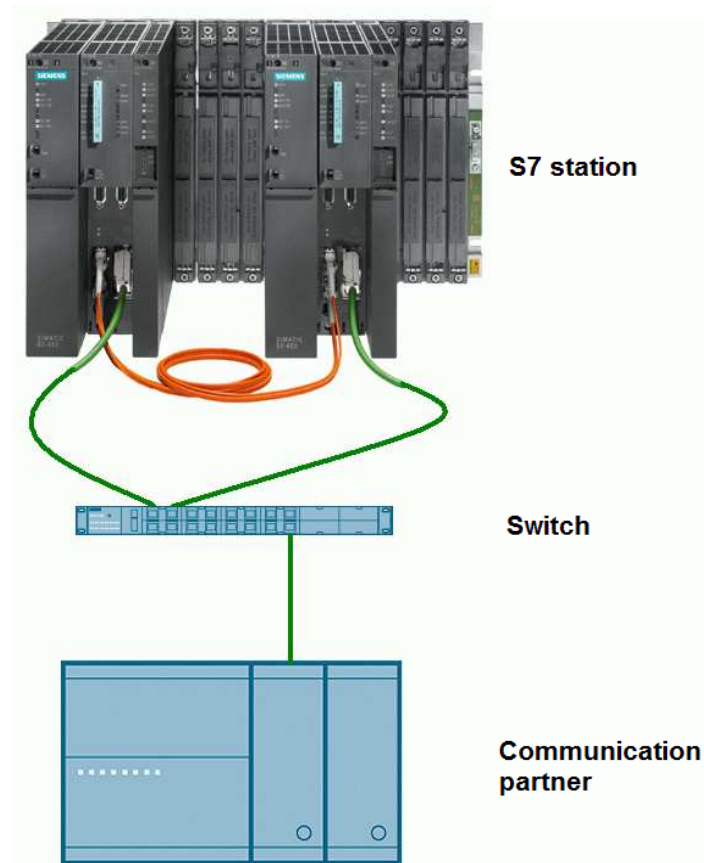
6.1 Configuration of redundant communication

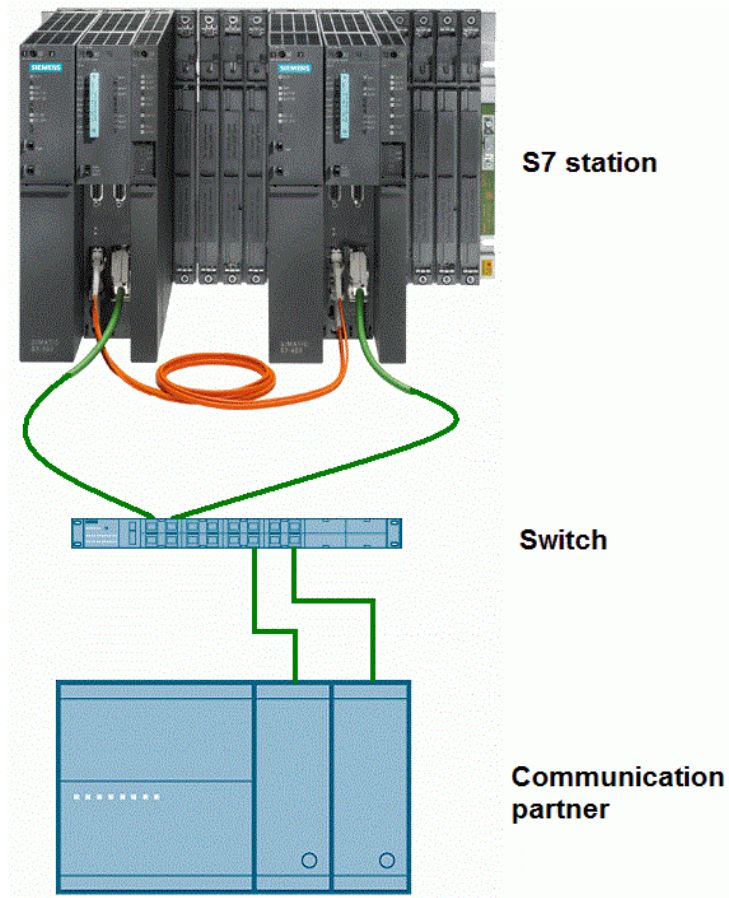
General information

The following pages provide an overview of the various options for configuring redundancy.

The communication stations can be standalone or redundant. If one of the two stations is standalone, the term single-sided redundancy is used. If both partners are redundant, this is known as double-sided redundancy.

Single-sided redundancy:



Double-sided redundancy:**Port number for client and server**

The Modbus client uses a port number of 2000 or higher.

The Modbus server is usually addressed with port number 502. Depending on the CPU, it may be possible to configure port 502 for multiple connections (multiport). If local port 502 has been configured for two or more connections, the requesting clients are randomly assigned to the available server connections. The first client that tries to connect to the MB_PNHSV block is not automatically assigned connection 0A. Once the client requests have been assigned to the server connections, this assignment remains in place for the duration of telegram traffic until the connection is terminated.

6.2 Single-sided redundancy

General information

One connection must be configured in the parameter data block for each connection between the communication partners.

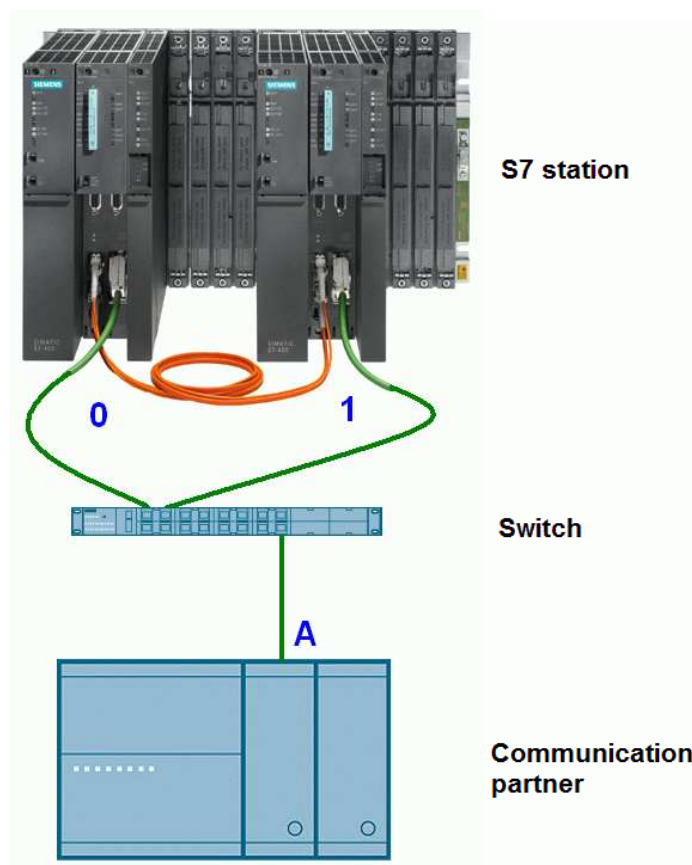
The connection points of the **S7** are labeled **0 and 1**, and the connection points of the **communication partner A and B**.

Configuration

If the S7 is set up as redundant, one connection is created for CPU0 to node A of the communication partner, and one connection for CPU1 to node B of the communication partner.

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

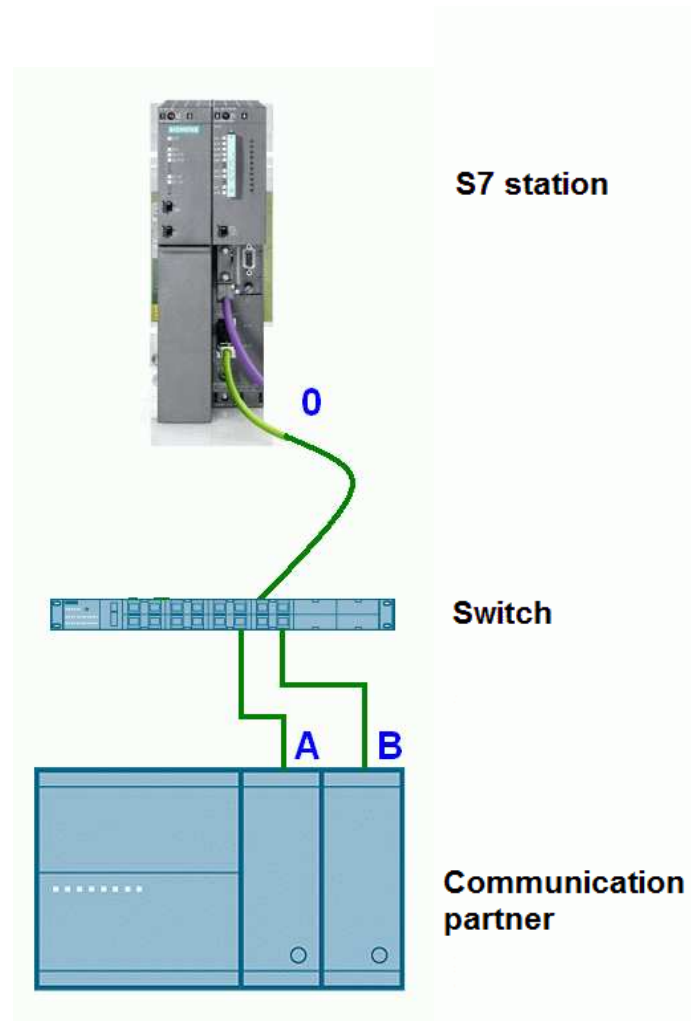
The figure below illustrates the connection names.



If the S7 is set up as standalone and the communication partner as redundant, one connection is created from CPU0 to node A of the communication partner, and one connection from CPU0 to node B of the communication partner.

- CPU0 connection to partner/node A => **Connection 0A**
- CPU0 connection to partner/node B => **Connection 0B**

The figure below illustrates the connection names.



6.3 Double-sided redundancy

General information

One connection must be configured in the parameter data block for each connection between the communication partners.

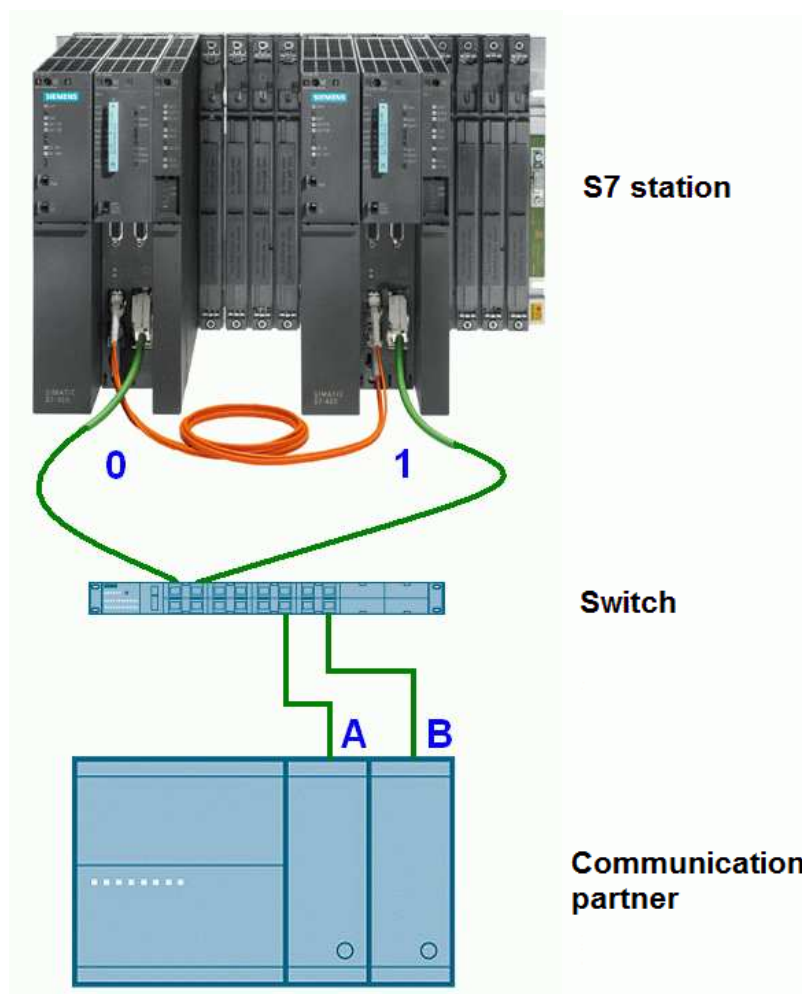
The connection points of the **S7** are labeled **0 and 1**, and the connection points of the **communication partner A and B**.

Configuration

With double-sided redundancy, two connections are created for CPU0 to the communication partner and two connections for CPU1 to the communication partner:

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

The figure below illustrates the connection names.



7 MB_PNHCL function block – Modbus client

7.1 How the MB_PNHCL FB works

General information

The CPU is the client if the S7 initiates reading data from or writing data to the remote partner.

The FBs MB_PNHCL, MOD_CLI and TCP_COMM are required for client operation. Multiple instances of the MB_PNHCL block can be called in the program. There is no limitation of the maximum number of parallel called Modbus blocks on the part of the library. However, the CPU may have a limit on the number of connections that can be established simultaneously. In the CPU manual it is detailed how many connections can be processed simultaneously.

If there are multiple instances of MB_PNHCL, you must make sure that each block instance has allocated its own parameter data block and that the **connection IDs are unique across the CPU**.

FB tasks

The MB_PNHCL function block performs the following tasks:

- Coordinating the connection(s) via which the telegrams are sent.
- Managing Transaction Identifiers TI
- License check

The MB_PNHCL block calls the MOD_CLI block internally multiple times.

The MOD_CLI block performs the following tasks:

- Generating MODBUS-specific telegram headers during sending
- Checking the MODBUS-specific telegram headers upon receipt
- Checking whether the data areas addressed exist
- Data transfer from/to the DB configured

The MOD_CLI block calls the TCP_COMM block internally multiple times.

TCP_COMM performs the following tasks:

- Handling connections and data using the T-blocks of the standard library
- Time monitoring of connection establishment and termination and data receipt

Online help Block online help is available for the MB_PNHCL function block in SIMATIC Manager. Selecting the block and pressing the key "F1" opens the online help with the most important information about the block.

FB call The **MB_PNHCL** function block must be integrated into two OBs to ensure the program runs correctly:

- in the startup OB100 and
- in a cyclic OB (OB1 or in a cyclic interrupt OB, e.g. OB35)

The same instance data block must be used.
The other FBs in the library, MOD_CLI and TCP_COMM, are called subordinately and must not also be called in an OB.

The MB_PNHCL FB must not be called simultaneously in OB1 and in a cyclic interrupt OB (e.g. OB35).

There **must** be an OB121 in the CPU. Additional information on this can be found in the section "**Licensing**".

FB startup The MB_PNHCL function block is called unconditionally once in OB100.

- The initialization parameters must be set according to the station configuration.
- The initialization parameters are applied to the instance DB.
- The runtime parameters are not evaluated during startup.
- The data from MODBUS_HPARAM_PN is checked for plausibility.

Cyclic operation of the FB In cyclic operation, the MB_PNHCL FB is called in OB35, for example.

- The block functions are activated according to the runtime parameters.
- When a job is running, changes to the runtime parameters are ignored.
- The initialization parameters are not evaluated unless manual initialization is executed.

Programming error OB121 If the Modbus block has not been licensed for this CPU yet, OB121 is called.



Warning

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

Connection processing

Active connection establishment is done by the Modbus client. The required information is read out of the connection parameters in the MODBUS_HPAM_PN DB.

A parameter in the connection parameter block (*active_est*) is used to define whether the PN CPU is to act as active or passive communication partner. With both connection types, active and passive, the TCON function opens a communication channel to the communication partner during runtime.

The time when the connection establishment starts is defined with a parameter in the DB MODBUS_HPAM_PN (*connect_at_startup*).

Connection termination is controlled with the DISCONNECT parameter at the MB_PNHCL FB.

Job trigger

A rising edge at the trigger input ENQ initiates a job. In accordance with the UNIT, DATA_TYPE, START_ADDRESS, LENGTH and WRITE_READ input parameters, a MODBUS request telegram is generated and sent to the partner station via the TCP/IP connection. The block waits for the configured delay, RECV_TIMEOUT, for a response from the server.

Handling for a faulty connection

The MB_PNHCL block detects a connection error if the TSEND/TRCV communication functions report an error during data telegram transfer. Once the error code has been displayed, the status A0FF is then indicated. This means that the connection has been configured but is not currently established.

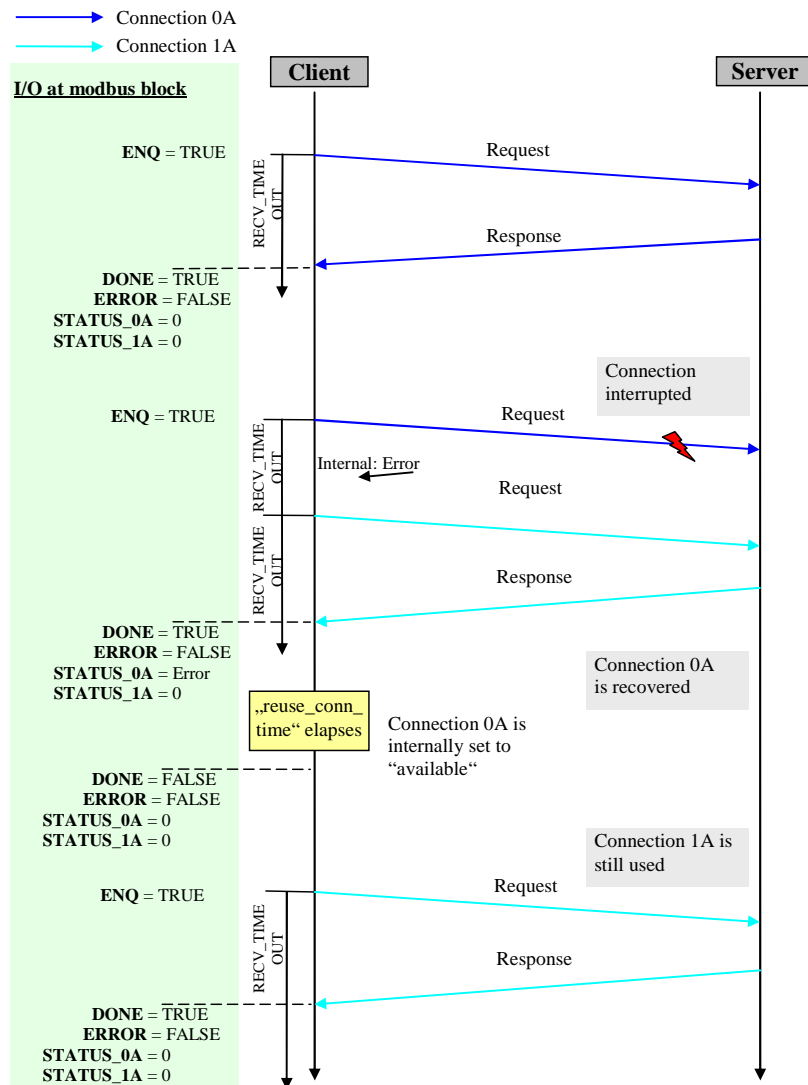
If a connection error is detected, the time "**reuse_conn_time**" is started. As long as the "reuse_conn_time" timer is still running, the system does not try to send Modbus telegrams via the faulty connection.

Once the time has elapsed, the system attempts to re-establish the connection.

Send telegrams via one connection

With the setting use_all_conn = FALSE the MODBUS telegram is sent via one - the currently active - connection. In case of a timeout (no response from the server) or a connection error, the system attempts to send the configured telegram via the other (a maximum of 4) configured connections. The sequence in doing so is 0A, 1A, 0B and 1B. When a telegram is successfully transferred via a connection, this connection is marked "active" and is used for subsequent telegram transfer. In case of a connection error of the active connection, a transmission retry is carried out via the other configured connections. If all send attempts fail, ERROR and STATUS_x are set accordingly.

When a response telegram is received, a validity check is carried out. If the result is positive, the required actions will be taken and the job will be completed without error. The DONE output is set. If an error is recognized during verification, the job is finished with an error, the ERROR bit is set and an error number is displayed at STATUS_x. In this case, the system does not launch another send attempt for the telegram via the next configured connection. The system only switches to the other configured connections if a connection error is detected or no response was received.



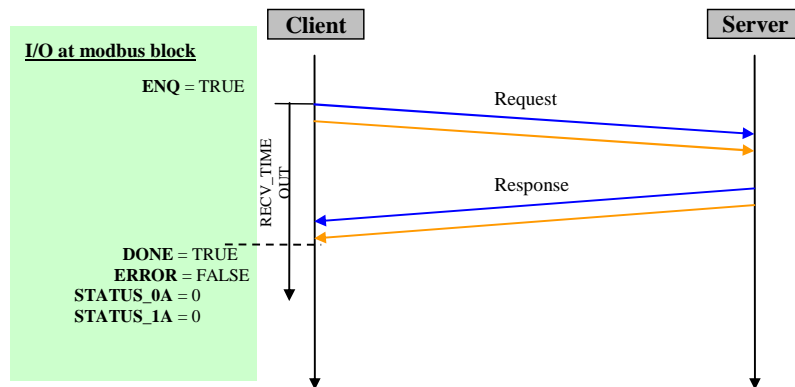
Send telegrams via all connections

With the setting use_all_conn = TRUE the MODBUS telegram is sent via all configured and established connections. A plausibility check is carried out once a response telegram is received via one of the connections. If the plausibility check is positive, the required actions are executed. The **DONE, ERROR and STATUS_x** outputs are only **updated** once the activated job has been completed **on all configured connections** – either a response telegram has been received or the monitoring time elapsed. If a valid response telegram has been received on at least one connection, the DONE output is set. If errors have been detected on all connections, the ERROR bit is set and the error numbers are displayed at STATUS_x.

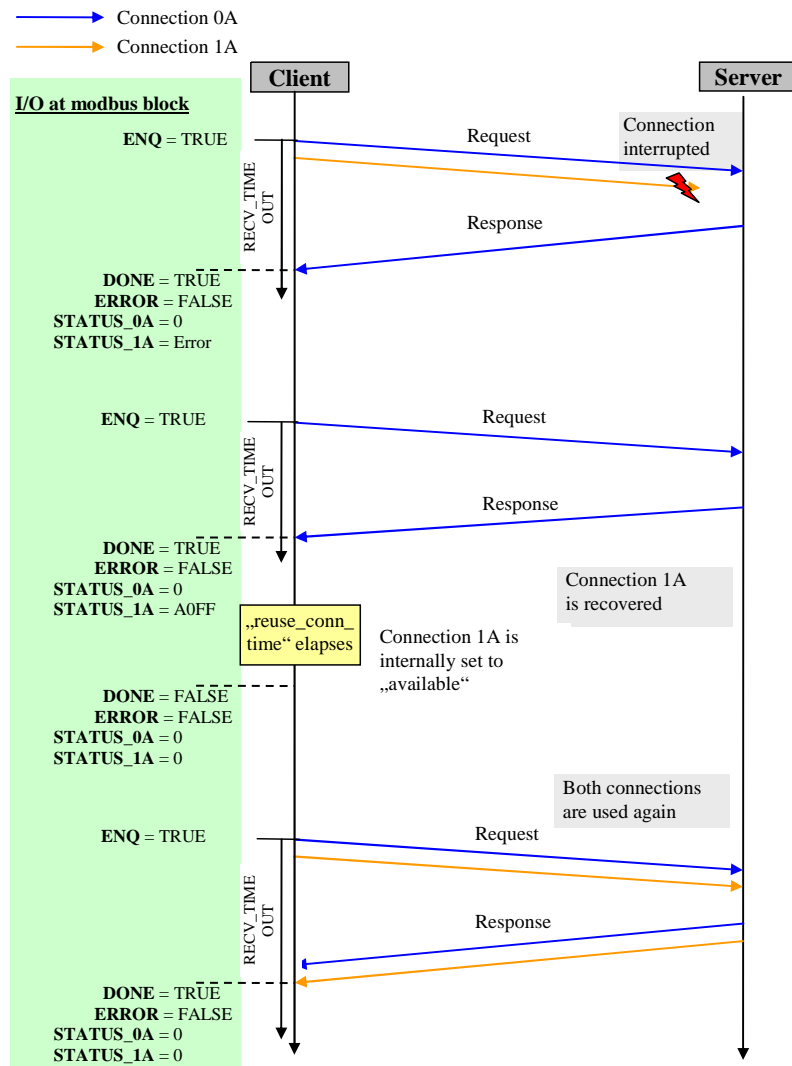
If one of the configured connections has failed, subsequent MODBUS telegrams are not sent via the faulty connection.

Scenario a) All response telegrams are received without errors.

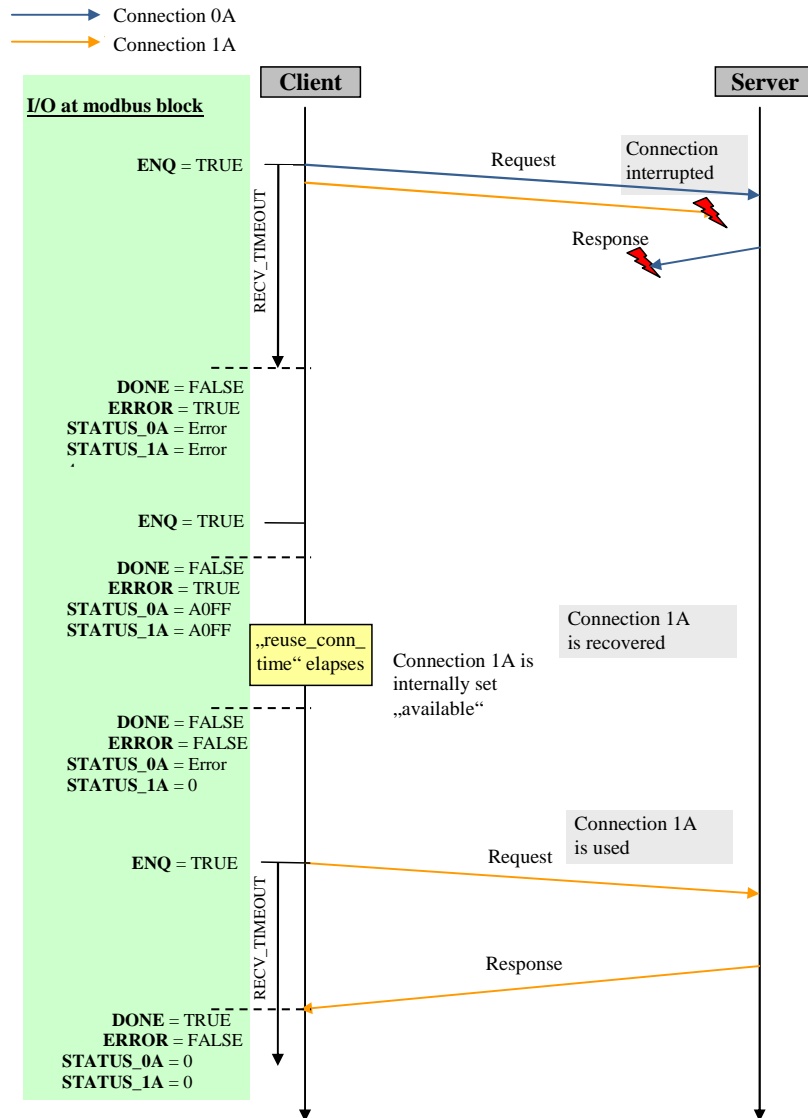
- Connection 0A
- Connection 1A



Scenario b) At least one response telegram is not received.



Scenario c) No response telegrams are received.



7.2 Parameters of the function block MB_PNHCL

Parameter	Decl	Type	Description	Value range	Init
id_0_a	IN	WORD	Connection ID for CPU0 to communication partner (node A) in accordance with configuration in the parameter DB	1 to 4095 W#16#1 to W#16#FFF	Yes
id_1_a	IN	WORD	Connection ID for CPU1 to communication partner (node A) in accordance with configuration in the parameter DB	1 to 4095 W#16#1 to W#16#FFF	Yes
id_0_b	IN	WORD	Connection ID for CPU0 to communication partner (node B) in accordance with configuration in the parameter DB	1 to 4095 W#16#1 to W#16#FFF	Yes
id_1_b	IN	WORD	Connection ID for CPU1 to communication partner (node B) in accordance with configuration in the parameter DB	1 to 4095 W#16#1 to W#16#FFF	Yes
db_param	IN	BLOCK_DB	Parameter DB, contains all connection and Modbus parameters for this Modbus block instance	Depends on CPU	Yes
reuse_conn_time	IN	TIME	Time after which it will attempt to re-establish the connection; at least 1s	T#1s to T#+24d20h31 m23s	Yes
use_all_conn	IN	BOOL	Send telegram via one connection Send telegram via all configured connections	FALSE TRUE	Yes
RECV_TIME_OUT	IN	TIME	Monitoring time for data receipt, at least 20ms	T#20ms to T#+24d20h31 m23s	No
CONN_TIME_OUT	IN	TIME	Monitoring time for connection establishment and termination, at least 100ms	T#100ms to T#+24d20h31 m23s	No
DISCONNECT	IN	BOOL	TRUE: Connection termination after receipt of response telegram	TRUE/FALSE	No
REG_KEY_DB	IN	BLOCK_DB	Data block with registration key for licensing	Depends on CPU	No
Init	IN	BOOL	Manual initialization on rising edge	TRUE/FALSE	No
ENQ	IN	BOOL	Job trigger on rising edge	TRUE/FALSE	No
DATA_TYPE	IN	BYTE	Data type to be processed Coils Inputs Holding register Input register	1 2 3 4	No
START_ADDRESS	IN	WORD	MODBUS start address	0 to 65535 W#16#0000 to W#16#FFFF	No

Parameter	Decl	Type	Description	Value range	Init
LENGTH	IN	WORD	Number of values to be processed <u>Coils</u> Read function Write function <u>Inputs</u> Read function <u>Holding register</u> Read function Write function <u>Input register</u> Read function	1 to 2000 11 to 1968 1 to 2000 1 to 125 1 to 123 1 to 125	No
WRITE_READ	IN	BOOL	Write access Read access	TRUE FALSE	No
UNIT	IN	BYTE	Unit Identifier	0 to 255 B#16#0 to B#16#FF	No
LICENSED	OUT	BOOL	Licensing status of the Modbus block: Block is licensed Block is not licensed	TRUE FALSE	No
BUSY	OUT	BOOL	Processing status of a Modbus telegram In progress Not in progress	TRUE FALSE	No
DONE	OUT	BOOL	TRUE: Activated job completed without errors on at least one connection	TRUE/FALSE	No
ERROR	OUT	BOOL	TRUE: Errors have occurred on all connections.	TRUE/FALSE	No
ESTAB_0A	OUT	BOOL	TRUE: Connection 0A has been established	TRUE/FALSE	No
STATUS_0A	OUT	WORD	Status of connection 0A	0 to FFFF	No
ESTAB_1A	OUT	BOOL	TRUE: Connection 1A has been established	TRUE/FALSE	No
STATUS_1A	OUT	WORD	Status of connection 1A	0 to FFFF	No
ESTAB_0B	OUT	BOOL	TRUE: Connection 0B has been established	TRUE/FALSE	No
STATUS_0B	OUT	WORD	Status of connection 0B	0 to FFFF	No
ESTAB_1B	OUT	BOOL	TRUE: Connection 1B has been established	TRUE/FALSE	No
STATUS_1B	OUT	WORD	Status of connection 1B	0 to FFFF	No
IDENT_CODE	OUT	STRING [18]	Identification for licensing. Use this string to request the REG_KEY.	Character	No
RedErrS7	OUT	BOOL	TRUE: S7 lost redundancy	TRUE/FALSE	No
RedErrDev	OUT	BOOL	TRUE: communication partner lost redundancy	TRUE/FALSE	No
TotComErr	OUT	BOOL	TRUE: Complete failure of communication	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 MB_PNHCL FB are divided into two groups:

- Initialization parameters (lower case)
- Runtime parameters (upper case)

The **initialization parameters** are only evaluated and applied to the instance DB when called in OB100. The initialization parameters are marked "Yes" in the "INIT" column of the table above.

Changes to the initialization parameters during normal operation have no effect. Following a change to these parameters, for example in test operation, the instance DB (I-DB) must be re-initialized with a CPU STOP/RUN. Initialization can also be carried out using the "Init" parameter.

Runtime parameters can be changed during cyclic operation. You should not change the input parameters while a job is in progress. You should wait until one job has completed with DONE or ERROR before preparing for the next job and making the necessary parameter changes.

The output parameters are **dynamic displays** and are therefore only pending for **1 CPU cycle**. They must be copied to other memory areas for additional processing or for display in the variable table.

Value ranges

There may also be CPU-specific restrictions on the value ranges for the various parameters.

**id_0_a, id_1_a
id_0_b, id_1_b**

A connection ID is required for each PN CPU connection to a communication partner. A different connection ID must be used for each logical connection. This connection ID is configured in the connection parameter block in the MODBUS_HPARAM_PN parameter data block. The connection ID is a unique number for the connection from the CPU to the communication partner and can have a value between 1 and 4095. The connection ID from the connection parameter block is entered here and must be unique across the CPU.

id_0_a Connection from CPU0 to communication partner/node A
id_1_a Connection from CPU1 to communication partner/node A
id_0_b Connection from CPU0 to communication partner/node B
id_1_b Connection from CPU1 to communication partner/node B

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

If the communication partner is set up as standalone, you only need parameters id_0_a and id_1_a. If the S7 is set up as standalone, you only need parameters id_0_a and id_0_b. If both communication partners are set up as redundant, all 4 connections are configured.

db_param

The parameter db_param is the number of data block MODBUS_HPARAM_PN. This parameter data block contains the connection-specific and Modbus-specific parameters required for communication between the PN CPU and the communication partner.

The value range for this parameter depends on the CPU. 0 cannot be used as a DB number because it is reserved for the system. The DB number is input in plain text in the following format: "DBxy".

Each Modbus block instance requires its own parameter data block.

reuse_conn_time

This parameter defines the intervals at which a connection recognized as faulty is to be included in communication again. In the event of a connection error on 0A, 1A, 0B or 1B, a timer is started with the time specified at this parameter. As long as this timer is running, the system does not attempt to establish the connection or to send Modbus telegrams via the connection. When the timer stops, the Modbus block automatically activates connection establishment again.

If the connection was not established since the initialization, connection errors are shown with ERROR = TRUE, otherwise with ERROR = FALSE.

The minimum time that can be set is one second.

use_all_conn

This parameter defines the number of connections via which the Modbus telegrams are to be sent. If set to FALSE, the Modbus telegrams are only sent via one connection. If the parameter is TRUE, the Modbus telegrams are sent via all configured connections.

RECV_TIMEOUT The monitoring time RECV_TIMEOUT monitors the receipt of the response telegram from the communication partner. The minimum value is 20ms.

If RECV_TIMEOUT is set to < 20ms, an error message appears and the activated job is rejected. When the monitoring time elapses without receiving a telegram, the activated job finishes with an error.

CONN_TIMEOUT The CONN_TIMEOUT time monitors connection establishment and termination. The minimum value is 100ms.

If the connection is not successfully established or terminated within the configured monitoring time, a corresponding error message appears at the STATUS_x output.

When *connect_at_startup* = TRUE, a too low configured CONN_TIMEOUT is set to 5s. In cyclic operation, a too short CONN_TIMEOUT results in an error message and the rejection of the activated job.

DISCONNECT DISCONNECT = TRUE specifies that the connection is to be terminated after data transfer. If this parameter is TRUE, the time reuse_conn_time for the re-establishment of the connections is not started.

This parameter is a runtime parameter and can be set and reset as required.

REG_KEY_DB The block must be licensed on each H system. The block is licensed and Modbus communication can be used without restrictions once the activation code has been entered correctly. The data block number containing the activation code is entered here. Multiple activation codes can be entered one after another in the DB. The Modbus block browses the DB for the right activation code.

For additional information, see the section "**Licensing**".

Init The parameter Init = TRUE enables manual initialization of the Modbus block. Initialization can only be performed if there is no job in progress. This must be ensured in the program with ENQ = FALSE and BUSY = FALSE. Please note with manual initialization that the initialization parameters need to be configured in the cyclic OB.

Warning



Manual initialization terminates the configured connections. If the ID parameters are changed, the connections must be terminated manually with DISCONNECT = TRUE before manual initialization.

ENQ Data transfer is initiated with a rising edge. The request telegram is generated with the values of the UNIT, WRITE_READ, DATA_TYPE, START_ADDRESS and LENGTH input parameters. A new job cannot be sent until the previous job has completed with DONE or ERROR. If the connection has not been established (ESTAB_x = FALSE), this is done before data transfer is carried out.

DATA_TYPE

The DATA_TYPE parameter indicates which Modbus data type is being processed with the current telegram. The following values are permitted:

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

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

Data type	DATA_TYPE	Function	Length	single_write	Function code
Coils	1	Read	Any	Irrelevant	1
Coils	1	Write	1	TRUE	5
Coils	1	Write	1	FALSE	15
Coils	1	Write	>1	Irrelevant	15
Inputs	2	Read	Any	Irrelevant	2
Holding register	3	Read	Any	Irrelevant	3
Holding register	3	Write	1	TRUE	6
Holding register	3	Write	1	FALSE	16
Holding register	3	Write	>1	Irrelevant	16
Input register	4	Read	Any	Irrelevant	4

START_ADDRESS

The START_ADDRESS parameter specifies the first MODBUS address to be written or read.

LENGTH

The LENGTH parameter specifies the number of MODBUS values to be written or read.

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

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

WRITE_READ

This parameter defines whether a read or a write function is to be executed. If the input is FALSE, the function is a read function. TRUE indicates a write function.

Write access is only possible to holding registers and coils. Input register and inputs can only be written.

UNIT	<p>The UNIT parameter, Unit Identifier, uniquely identifies the communication partner. It is most important when one converter has multiple serial nodes to be addressed with different UNIT numbers.</p> <p>The input is to be set in accordance with requirements. The FB applies this value to the request telegram and checks it when the response is received. Please note that some communication partners expect a specific UNIT number.</p>
LICENSED	<p>If this output is set to TRUE, the Modbus block is licensed on this CPU. If the output is FALSE, no license string or the wrong license string has been entered. For additional information, see the section "Licensing".</p>
BUSY	<p>If this output is set, a Modbus telegram is currently being processed.</p>
DONE	<p>The activated job has completed without errors on at least one connection. Read function: the response data from the server has already been entered in the DB. Write function: the server response to the request telegram has been received.</p>
ERROR	<p>If this output is set, errors have been detected on all active connections.</p> <p>use_all_conn = FALSE: ERROR is set immediately in the event of a protocol error. In the event of a connection error, all configured connections are checked and ERROR is only set if all connections are faulty.</p> <p>use_all_conn = TRUE: If this output is set, errors have been detected on all configured connections.</p> <p>The error numbers are displayed at the STATUS outputs.</p>
ESTAB_0A, ESTAB_1A, ESTAB_0B, ESTAB_1B	<p>ESTAB_x = TRUE indicates that a connection to the communication partner is established and that data can be transferred.</p> <p>ESTAB_x = FALSE indicates that there is no connection to the communication partner.</p> <p>If at least 1 projected connection fails, these outputs are updated after reuse_conn_time is elapsed.</p>
STATUS_0A, STATUS_1A, STATUS_0B, STATUS_1B	<p>The STATUS_x outputs show the error number when ERROR is set and the status information for the corresponding connection when ERROR is not set.</p> <p>The error numbers and status information are described in "Diagnostics".</p>
IDENT_CODE	<p>Following CPU0 startup, this parameter displays an 18-digit identifier that is used to request the REG_KEY (activation code) for Modbus communication.</p> <p>For additional information, see the section "Licensing".</p>

RedErrS7	<p>Output RedErrS7 = TRUE indicates a redundancy error at the SIMATIC. With single-sided redundancy, this means that the CPU0 or CPU1 connection has failed. With double-sided redundancy, it means that both CPU0 connections or both CPU1 connections have failed.</p> <p>For additional information, see the section "Diagnostic messages with alarm bits".</p>
RedErrDev	<p>Output RedErrDev = TRUE indicates a redundancy error at the communication partner. With single-sided redundancy, this means that the connection from node A to CPU0 or CPU1 has failed. With double-sided redundancy, it means that both connections to node A or both connections to node B of the communication partner have failed.</p> <p>For additional information, see the section "Diagnostic messages with alarm bits".</p>
TotComErr	<p>The TotComErr output value TRUE indicates a complete loss of communication, i.e. all configured connections have been disrupted.</p> <p>For additional information, see the section "Diagnostic messages with alarm bits".</p>
Init_Error	<p>If an error has occurred in manual initialization, this is indicated with Init_Error = TRUE.</p>
Init_Status	<p>The Init_Status output displays the error number when Init_Error is set. The error numbers are described in "Diagnostics".</p>

7.3 Example of address mapping

Interpretation of Modbus addresses

The MODBUS data model is based on a range of memory areas with varying characteristics. Some systems, such as MODICON PLCs, distinguish between these memory areas using the register or bit address. For example, the holding register is defined as register 40001 with offset 0 (memory type 4xxxx, reference 0001).

This issue is often a source of confusion, as some manuals describe and refer to the register address of the application layer, and others use the register or bit address actually transferred in the protocol.

For its *start_x*, *end_x* and START_ADDRESS parameters, the MODBUS FB uses the **Modbus address actually transferred**. Each function code can therefore transfer register/bit addresses of 0000_H to FFFF_H.

**Example:
Data area
parameter
assignment**

<i>data_type_1</i> <i>db_1</i> <i>start_1</i> <i>end_1</i>	B#16#3 W#16#B W#16#0 W#16#1F3	Holding register DB 11 Start address: 0 End address: 499
<i>data_type_2</i> <i>db_2</i> <i>start_2</i> <i>end_2</i>	B#16#3 W#16#C W#16#2D0 W#16#384	Holding register DB 12 Start address: 720 End address: 900
<i>data_type_3</i> <i>db_3</i> <i>start_3</i> <i>end_3</i>	B#16#4 W#16#D W#16#2D0 W#16#3E8	Input register DB 13 Start address: 720 End address: 1000
<i>data_type_4</i> <i>db_4</i> <i>start_4</i> <i>end_4</i>	B#16#0 0 0 0	Not used 0 0 0
<i>data_type_5</i> <i>db_5</i> <i>start_5</i> <i>end_5</i>	B#16#1 W#16#E W#16#280 W#16#4E2	Coils DB 14 Start address: 640 End address: 1250
<i>data_type_6</i> <i>db_6</i> <i>start_6</i> <i>end_6</i>	B#16#2 W#16#F W#16#6A4 W#16#8FC	Inputs DB 15 Start address: 1700 End address: 2300
<i>data_type_7</i> <i>db_7</i> <i>start_7</i> <i>end_7</i>	B#16#1 W#16#10 W#16#6A4 W#16#8FC	Coils DB 16 Start address: 1700 End address: 2300
<i>data_type_8</i> <i>db_8</i> <i>start_8</i> <i>end_8</i>	B#16#0 0 0 0	Not used 0 0 0

In this example:

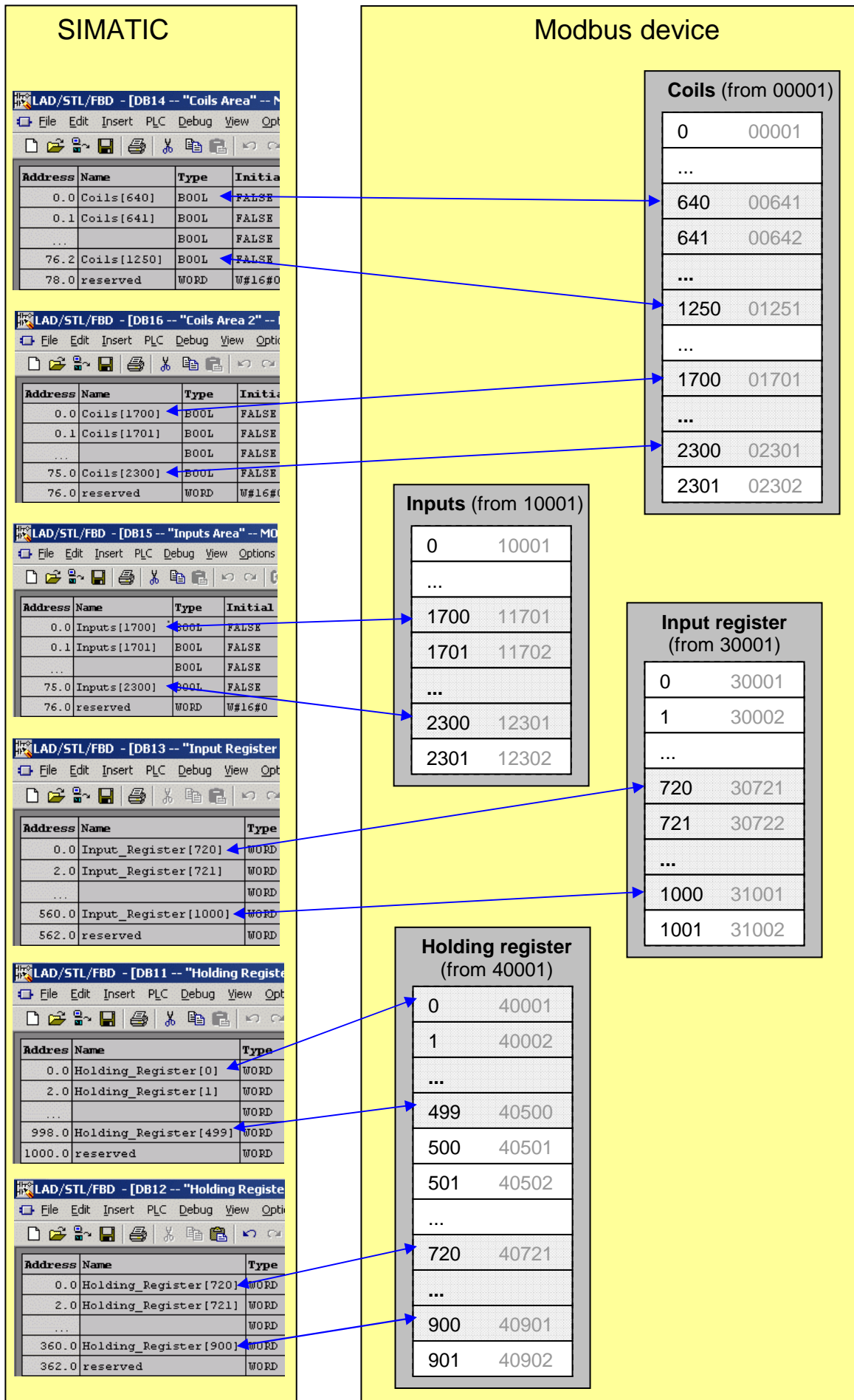
- DB11 is 1002 bytes; a total of 500 registers are mapped (register 0 – register 499) + 2 reserved bytes.
- DB12 is 364 bytes; a total of 181 registers are mapped (register 720 – register 900) + 2 reserved bytes
- DB13 is 564 bytes; a total of 281 input registers are mapped (register 720 – register 1000) + 2 reserved bytes
- DB14 is 80 bytes; a total of 611 coils (bits) are mapped (coil 640 – coil 1250) + 2 reserved bytes
- DB15 is 78 bytes; a total of 601 inputs (bits) are mapped (input 1700 – input 2300) + 2 reserved bytes
- DB16 is 78 bytes; a total of 601 coils (bits) are mapped (coil 1700 – coil 2300) + 2 reserved bytes

Address mapping

The figure below shows the Simatic memory areas and the register- and bit-based definition of memory in the Modbus devices. The figure is based on the parameter assignment above.

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

In SIMATIC: The SIMATIC addresses in the first column are the offset in the DB. The Modbus register numbers are in the square brackets.



7.4 Data and standard functions used by the FB

Instance DB	<p>The MB_PNHCL function block saves its data in an instance DB. This instance DB is generated by STEP 7 the first time the FB is called.</p> <p>The instance data block contains parameters of the type input and output type as well as static variables required for its execution. These variables are remanent and remain valid between FB calls. The variables control the internal process of the FB.</p> <p>Memory requirement of the instance DB:</p> <table border="1" data-bbox="475 674 1104 770"> <thead> <tr> <th>Instance DB</th> <th>Work memory</th> <th>Load memory</th> </tr> </thead> <tbody> <tr> <td>MB_PNHCL</td> <td>approx. 3 KB</td> <td>approx. 5 KB</td> </tr> </tbody> </table>	Instance DB	Work memory	Load memory	MB_PNHCL	approx. 3 KB	approx. 5 KB
Instance DB	Work memory	Load memory					
MB_PNHCL	approx. 3 KB	approx. 5 KB					
Local variables	A maximum total of 186 bytes of local data is required for an MB_PNHCL FB call.						
Parameter DB	The connection-specific and Modbus-specific parameters are saved in the MODBUS_HPAMM_PN parameter DB.						
Timers	The function block does not use any timers.						
Flags	The function block does not use any flags.						
Standard FBs for processing connections	The TCP_COMM FB called in the MB_PNHCL/MOD_CLI FB uses the TCON and TDISCON blocks from the standard library to establish and terminate connections between the CPU and the communication partner.						
Standard FBs for data transfer	The TCP_COMM FB called in the MB_PNHCL/MOD_CLI FB uses the TSEND and TRCV blocks from the standard library for data transfer between the CPU and the communication partner.						
MB_PNHCL: SFCs for other functions	<p>The MB_PNHCL FB uses the following SFCs from the standard library:</p> <ul style="list-style-type: none"> • SFC6 "RD_SINFO" • SFC20 "BLKMOV" • SFC24 "TEST_DB" • SFC51 "RDSYSST" • SFC52 "WR_USMSG" 						

**MOD_CLI:
SFCs for other
functions**

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

- SFC20 "BLKMOV"
- SFC24 "TEST_DB"

**TCP_COMM:
SFCs for other
functions**

The TCP_COMM FB uses the following SFB from the standard library as well as the T blocks:

- SFB4 "TON"

**Additional
information**

The TI parameter is updated internally by the MB_PNHCL block and incremented by one with each new job.

The time within which connection termination can be detected can be adjusted with the Keep Alive Time parameter. You will find this parameter in the CPU properties in HW Config.

7.5 Renaming / rewiring functions and function blocks

Motive

If the numbers of the standard functions are already being used in your project, or the number range is reserved for other applications (e.g. in CFC), you can rewire the internally called FB63, FB64, FB65 and FB66 function blocks of the TCP_COMM FB, or the MB_PNHCL, MOD_CLI, MOD_SERV and TCP_COMM blocks.

The system functions SFC6, SFC20, SFC24, SFC51 and SFC52 and the system function block SFB4 cannot be renamed/rewired.

Reaction

A set of rules concerning the function block numbering have to be considered when rewiring function blocks in SIMATIC STEP 7 Manager:

To rewire blocks from the Modbus library, proceed in this order:

1. FB915 MB_PNHCL
2. FB914 MOD_CLI
3. FB913 TCP_COMM
4. FB63 TSEND
FB64 TRCV
FB65 TCON
FB66 TDISCON

You do not need to rewire all functions or all function blocks. However, you must work in this order even if you are only rewiring some of them.

Rewiring

Proceed as follows to rewire FBs:

1. Information on the addresses used is found under "Options > Reference data > Display".
2. Set the address priority to "Absolute value" in the object properties of the block folder.
3. In SIMATIC Manager, select the block folder and open the "Options > Rewire" function to rewire the addresses to free areas.
4. To continue using the symbols in diagnostic tools, apply the changes to the symbol table.

The modifications can be verified by clicking "**Options > References data > Display**".

8 MB_PNHSV function block – Modbus server

8.1 How the MB_PNHSV FB works

General information

The S7 is the server if the remote partner initiates reading data from or writing data to the S7.

The FBs MB_PNHSV, MOD_SERV and TCP_COMM are required for server functionality.

Multiple instances of the MB_PNHSV block can be called in the program. The library has no limit to the number of Modbus blocks that can run simultaneously. However, the CPU may have a limit on the number of connections that can be established simultaneously. The CPU manuals set out how many connections can be established simultaneously.

If there are multiple instances of MB_PNHSV, you must make sure that each block instance has allocated its own parameter data block and that the **connection IDs are unique across the CPU**.

FB tasks

The MB_PNHSV block calls the MOD_SERV block internally multiple times, and implements licensing and coordinates MOD_SERV calls of the various connections.

The MOD_SERV block performs the following tasks:

- Generating MODBUS-specific telegram headers during sending
- Checking the MODBUS-specific telegram headers upon receipt
- Checking whether the data areas addressed by the client exist
- Data transfer from/to the DB configured
- Generating exception telegrams when errors occur

Exception code	Meaning
1	The function code sent is not supported.
2	Access to a non-existent or invalid address.
3	Invalid length entered for this function code.

The MOD_CLI block calls the TCP_COMM block internally multiple times.
TCP_COMM performs the following tasks:

- Handling connections and data using the T-blocks in the standard library
- Time monitoring of connection establishment and termination and data receipt

Online help

Block online help is available for the MB_PNHSV function block in SIMATIC Manager. Selecting the block and pressing "F1" opens the online help with the most important information about the block.

FB call

The **MB_PNHSV** function block must be integrated into two OBs to ensure the program runs correctly:

- in the startup OB100 and
- in a cyclic OB (OB1 or in a cyclic interrupt OB, e.g. OB35)

The same instance data block must be used.
The other FBs in the library, MOD_SERV and TCP_COMM, are called subordinately and must not also be called in an OB.

The MB_PNHSV FB must not be called simultaneously in OB1 and in a cyclic interrupt OB (e.g. OB35).

There **must** be an OB121 in the CPU. Additional information on this can be found in the section "**Licensing**".

FB startup

The MB_PNHSV function block is called unconditionally once in OB100.

- The initialization parameters must be assigned in line according to the station configuration.
- The initialization parameters are applied to the instance DB.
- The runtime parameters are not evaluated during startup.
- The data from MODBUS_HPARAM_PN is checked for plausibility

Cyclic operation of the FB

In cyclic operation, the MB_PNHSV FB is called in OB35, for example.

- The block functions are activated on the basis of the runtime parameters.
- If a job is running, changes to the runtime parameters are not evaluated.
- The initialization parameters are not evaluated unless manual initialization is executed.

**Programming error
OB121**



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

Warning

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

**Instance DB:
Information on the
client request**

At each client request, the values for the job executed are saved in the I-DB of the server in an information block. If necessary, they can be read in the user program. The following values are buffered in the I-DB for each connection and are valid when NDR = TRUE:

Address in the IDB for connection 0A	Variable name	Description
DBX 66.5	CONNECTION[1]. WRITE_READ	TRUE: Writes to S7 FALSE: Reads from S7
DBB 67	CONNECTION[1]. UNIT	Unit number
DBB 68	CONNECTION[1]. DATA_TYPE	Addressed data type 1: Coils 2: Inputs 3: Holding register 4: Input register
DBW 70	CONNECTION[1]. START_ADDRESS	Start address
DBW 72	CONNECTION[1]. LENGTH	Number of registers / bits processed
DBW 74	CONNECTION[1]. TI	Transaction Identifier (sequential number)
DBD 88	CONNECTION[1]. Cnt_NDR	Counter for jobs processed without errors
DBD 92	CONNECTION[1]. Cnt_ERROR	Counter for errors detected

For connection 1A (CONNECTION[2]), the information block starts at address DBX 96.0

For connection 0B (CONNECTION[3]), the information block starts at address DBX 128.0

For connection 1B (CONNECTION[4]) the information block starts at address DBX 160.0

Connection processing

Passive connection establishment is done by the Modbus server. The required data is read out of the connection parameters in the MODBUS_HPAM_PN DB.

A parameter in the connection parameter block (*active_est*) is used to define whether the PN CPU is to act as active or passive communication partner. With both connection types, active and passive, the TCON function opens a communication channel to the communication partner during runtime.

The time of connection establishment is defined with a parameter in the DB MODBUS_HPAM_PN (*connect_at_startup*).

Connection termination is controlled with the DISCONNECT parameter at the MB_PNHSV FB.

Activate the FB

A positive level at the ENR trigger input prepares the FB to receive a request telegram from the client. The server is passive.

When ENR = TRUE, all configured connections are active and ready to receive. The system does not switch between connections. The client can send either via one connection only or via all connections.

The telegrams received are checked. If the result is positive, the response telegram is generated and sent. The NDR_x bit for the relevant connection is set to inform the user of the completed telegram traffic.

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

Handling for a faulty connection

The MB_PNHSV block detects a connection error if the TSEND/TRCV communication functions report an error during data telegram transfer. Once the error code has been displayed, the status A0FF is then indicated. This means that the connection has been configured but is not currently established.

If an error is detected in a connection and ENR is set, the system tries to establish the connection again.

8.2 Parameters of the MB_PNHSV function block

Parameter	Decl	Type	Description	Value range	Init
id_0_a	IN	WORD	Connection ID for CPU0 to communication partner (node A) in accordance with configuration in the parameter DB	1 to 4095 W#16#1 to W#16#FFF	Yes
id_1_a	IN	WORD	Connection ID for CPU1 to communication partner (node A) in accordance with configuration in the parameter DB	1 to 4095 W#16#1 to W#16#FFF	Yes
id_0_b	IN	WORD	Connection ID for CPU0 to communication partner (node B) in accordance with configuration in the parameter DB	1 to 4095 W#16#1 to W#16#FFF	Yes
id_1_b	IN	WORD	Connection ID for CPU1 to communication partner (node B) in accordance with configuration in the parameter DB	1 to 4095 W#16#1 to W#16#FFF	Yes
db_param	IN	BLOCK_DB	Parameter DB, contains all connection and Modbus parameters for this Modbus block instance	Depends on CPU	Yes
RECV_TIME OUT	IN	TIME	Monitoring time for data receipt, at least 20ms	T#20ms to T#+24d20h31 m23s	No
CONN_TIME OUT	IN	TIME	Monitoring time for connection establishment and termination, at least 100ms	T#100ms to T#+24d20h31 m23s	No
DISCON- NECT	IN	BOOL	TRUE: Connection termination when ENR = FALSE	TRUE/FALSE	No
REG_KEY_ DB	IN	BLOCK_DB	Data block with registration key for licensing	Depends on CPU	No
Init	IN	BOOL	Manual initialization on rising edge	TRUE/FALSE	No
ENR	IN	BOOL	Ready to receive with positive level	TRUE/FALSE	No
LICENSED	OUT	BOOL	Licensing status of the Modbus block Block is licensed Block is not licensed	TRUE FALSE	No
BUSY	OUT	BOOL	Processing status of a Modbus telegram In progress Not in progress	TRUE FALSE	No
NDR_0A	OUT	BOOL	TRUE: Client request has been executed and answered on connection 0A	TRUE/FALSE	No
ERROR_0A	OUT	BOOL	TRUE: An error has occurred on connection 0A.	TRUE/FALSE	No
STATUS_0A	OUT	WORD	Status of connection 0A	0 to FFFF	No
NDR_1A	OUT	BOOL	TRUE: Client request has been executed and answered on connection 1A	TRUE/FALSE	No
ERROR_1A	OUT	BOOL	TRUE: An error has occurred on connection 1A.	TRUE/FALSE	No

Parameter	Decl	Type	Description	Value range	Init
STATUS_1A	OUT	WORD	Status of connection 1A	0 to FFFF	No
NDR_0B	OUT	BOOL	TRUE: Client request has been executed and answered on connection 0B	TRUE/FALSE	No
ERROR_0B	OUT	BOOL	TRUE: An error has occurred on connection 0B.	TRUE/FALSE	No
STATUS_0B	OUT	WORD	Status of connection 0B	0 to FFFF	No
NDR_1B	OUT	BOOL	TRUE: Client request has been executed and answered on connection 1B	TRUE/FALSE	No
ERROR_1B	OUT	BOOL	TRUE: An error has occurred on connection 1B.	TRUE/FALSE	No
STATUS_1B	OUT	WORD	Status of connection 1B	0 to FFFF	No
IDENT_CODE	OUT	STRING [18]	Identification for licensing. Use this identification string to request the license.	Character	No
RedErrS7	OUT	BOOL	TRUE: S7 lost redundancy	TRUE/FALSE	No
RedErrDev	OUT	BOOL	TRUE: Communication partner lost redundancy	TRUE/FALSE	No
TotComErr	OUT	BOOL	TRUE: Complete failure of communication	TRUE/FALSE	No
Init_Error	OUT	BOOL	TRUE: An 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 MB_PNHSV FB are divided into two groups:

- Initialization parameters (lower case)
- Runtime parameters (upper case)

The **initialization parameters** are only evaluated and applied to the instance DB when called in OB100. The initialization parameters are marked "Yes" in the "INIT" column of the table above.

Changes to the initialization parameters during normal operation have no effect. Following a change to these parameters, for example in test operation, the instance DB (I-DB) must be re-initialized with a CPU STOP/RUN.

Initialization can also be carried out using the "Init" parameter.

Runtime parameters can be changed during cyclic operation. You should not change the input parameters while a job is in progress.

The output parameters are **dynamic displays** and are therefore only pending for **1 CPU cycle**. They must be copied to other memory areas for additional processing or for display in the variable table.

Value ranges	There may also be CPU-specific restrictions on the value ranges for the various parameters.
id_0_a, id_1_a id_0_b, id_1_b	<p>A connection ID is required for each PN CPU connection to a communication partner. A different connection ID must be used for each logical connection. This connection ID is configured in the connection parameter block in the MODBUS_HPPARAM_PN parameter data block. The connection ID is a unique number for the connection from the CPU to the communication partner and can have a value between 1 and 4095.</p> <p>The connection ID from the connection parameter block is entered here and must be unique across the CPU.</p> <p>id_0_a Connection from CPU0 to communication partner/node A id_1_a Connection from CPU1 to communication partner/node A id_0_b Connection from CPU0 to communication partner/node B id_1_b Connection from CPU1 to communication partner/node B</p> <p>Connection 0A is the default connection and must be configured.</p> <p>If the communication partner is set up as standalone, you only need parameters id_0_a and id_1_a. If the S7 is set up as standalone, you only need parameters id_0_a and id_0_b. If both communication partners are set up as redundant, all 4 connections are configured.</p>
db_param	<p>The parameter db_param is the number of data block MODBUS_HPPARAM_PN. This parameter data block contains the connection-specific and Modbus-specific parameters required for communication between the PN CPU and the communication partner.</p> <p>The value range for this parameter depends on the CPU. 0 cannot be used as a DB number because it is reserved for the system. The DB number is input in plain text in the following format: "DBxy".</p> <p>Each Modbus block instance requires its own parameter data block.</p>
RECV_TIMEOUT	<p>The monitoring time RECV_TIMEOUT monitors the receipt of the response from the communication partner. The minimum value is 20ms. We recommend a monitoring time of approx. 1.5 seconds.</p> <p>If RECV_TIMEOUT is set to < 20ms, a default value of 1.2 s is used. An error is reported if this monitoring time elapses. RECV_TIMEOUT monitors the runtime of the request telegram. The time gap between individual requests from the client is not included.</p>
CONN_TIMEOUT	<p>CONN_TIMEOUT specifies the time for monitoring connection establishment/termination. The minimum value is 100ms.</p> <p>If the connection is not successfully established or terminated within the configured monitoring time, a corresponding error message appears at the STATUS_x output.</p> <p>If CONN_TIMEOUT has been set to < 100ms, a default value of 5s is used.</p>

DISCONNECT DISCONNECT = TRUE terminates the connection if the ENR parameter is set to FALSE. If the parameter is TRUE, the connections are not re-established.

This parameter is a runtime parameter and can be set and reset as required.

REG_KEY_DB The block must be licensed on each H system. The block is licensed and Modbus communication can be used without restrictions once the activation code has been entered correctly. The data block number containing the activation code is entered here. Multiple activation codes can be entered one after another in the DB. The Modbus block browses the DB for the right activation code.

For additional information, see the section "**Licensing**".

Init The parameter Init = TRUE enables manual initialization of the Modbus block. Initialization can only be performed if there is no job in progress. The client may not send a request during this time. Please note with manual initialization that the initialization parameters need to be configured in the cyclic OB.

Warning



Manual initialization terminates the configured connections. If the ID parameters are changed, the connections must be terminated manually with DISCONNECT = TRUE and ENR = FALSE before manual initialization.

ENR The FB is activated by a positive level at the input. Telegrams from the client can be received. If the connection is not established and ENR is set (ESTAB_x = FALSE), connection establishment is activated. If ENR switches from TRUE to FALSE during normal operation, the connection may be terminated. This depends on the setting at the DISCONNECT parameter. If the ENR input is not set and there is a connection in place, the data received are rejected.

All configured connections are always monitored and incoming requests answered.

LICENSED If this output is set to TRUE, the Modbus block is licensed on this CPU. If the output is FALSE, no license string or the wrong license string has been entered. For additional information, see the section "**Licensing**".

BUSY If this output is set, a Modbus telegram is currently being processed.

ESTAB_0A, ESTAB_1A, ESTAB_0B, ESTAB_1B ESTAB_x = TRUE indicates that a connection to the communication partner is in place and that data can be transferred.

ESTAB_x = FALSE indicates that there is no connection to the communication partner.

NDR_0A, NDR_1A, NDR_0B, NDR_1B	The output indicates that telegram traffic has completed without errors on this connection.
ERROR_0A, ERROR_1A, ERROR_0B, ERROR_1B	If this output is set, errors have been detected on this connection during a request telegram from the client or during the sending of the response telegram. The corresponding error number is displayed at the STATUS_x output.
STATUS_0A, STATUS_1A, STATUS_0B, STATUS_1B	<p>The STATUS_x outputs show the error number when ERROR_x is set and the status information for the corresponding connection when ERROR_x is not set.</p> <p>The error numbers and status information are described in section "Diagnostics".</p>
IDENT_CODE	<p>Following CPU startup, this parameter displays an 18-digit identifier that is used to request the REG_KEY (activation code) for Modbus communication.</p> <p>For additional information, see the section "Licensing".</p>
RedErrS7	<p>Output RedErrS7 = TRUE indicates a redundancy error at the SIMATIC. With single-sided redundancy, this means that the CPU0 or CPU1 connection has failed. With double-sided redundancy, it means that both CPU0 connections or both CPU1 connections have failed.</p> <p>For additional information, see the section "Diagnostic messages with alarm bits".</p>
RedErrDev	<p>Output RedErrDev = TRUE indicates a redundancy error at the communication partner. With single-sided redundancy, this means that the connection from node A to CPU0 or CPU1 has failed. With double-sided redundancy, it means that both connections to node A or both connections to node B of the communication partner have failed.</p> <p>For additional information, see the section "Diagnostic messages with alarm bits".</p>
TotComErr	<p>The TotComErr output value TRUE indicates a complete loss of communication, i.e. all configured connections have been disrupted.</p> <p>For additional information, see the section "Diagnostic messages with alarm bits".</p>
Init_Error	If an error has occurred in manual initialization, this is indicated with Init_Error = TRUE.
Init_Status	The Init_Status output displays the error number when Init_Error is set. The error numbers are described in section "Diagnostics".

8.3 Example of address mapping

Interpretation of Modbus addresses

The MODBUS data model is based on a range of memory areas with varying characteristics. Some systems, such as MODICON PLCs, distinguish between these memory areas using the register or bit address. For example, the holding register is defined as register 40001 with offset 0 (memory type 4xxxx, reference 0001).

This issue is often a source of confusion, as some manuals describe and refer to the register address of the application layer, and others use the register or bit address actually transferred in the protocol.

For its *start_x*, *end_x* and START_ADDRESS parameters, the MODBUS FB uses the **Modbus address actually transferred**. Each function code can therefore transfer register/bit addresses of 0000_H to FFFF_H.

Example: Data area parameter assignment

<i>data_type_1</i> <i>db_1</i> <i>start_1</i> <i>end_1</i>	B#16#3 W#16#B W#16#0 W#16#1F3	Holding register DB 11 Start address: 0 End address: 499
<i>data_type_2</i> <i>db_2</i> <i>start_2</i> <i>end_2</i>	B#16#3 W#16#C W#16#2D0 W#16#384	Holding register DB 12 Start address: 720 End address: 900
<i>data_type_3</i> <i>db_3</i> <i>start_3</i> <i>end_3</i>	B#16#4 W#16#D W#16#2D0 W#16#3E8	Input register DB 13 Start address: 720 End address: 1000
<i>data_type_4</i> <i>db_4</i> <i>start_4</i> <i>end_4</i>	B#16#0 0 0 0	Not used 0 0 0
<i>data_type_5</i> <i>db_5</i> <i>start_5</i> <i>end_5</i>	B#16#1 W#16#E W#16#280 W#16#4E2	Coils DB 14 Start address: 640 End address: 1250
<i>data_type_6</i> <i>db_6</i> <i>start_6</i> <i>end_6</i>	B#16#2 W#16#F W#16#6A4 W#16#8FC	Inputs DB 15 Start address: 1700 End address: 2300
<i>data_type_7</i> <i>db_7</i> <i>start_7</i> <i>end_7</i>	B#16#1 W#16#10 W#16#6A4 W#16#8FC	Coils DB 16 Start address: 1700 End address: 2300
<i>data_type_8</i> <i>db_8</i> <i>start_8</i> <i>end_8</i>	B#16#0 0 0 0	Not used 0 0 0

In this example:

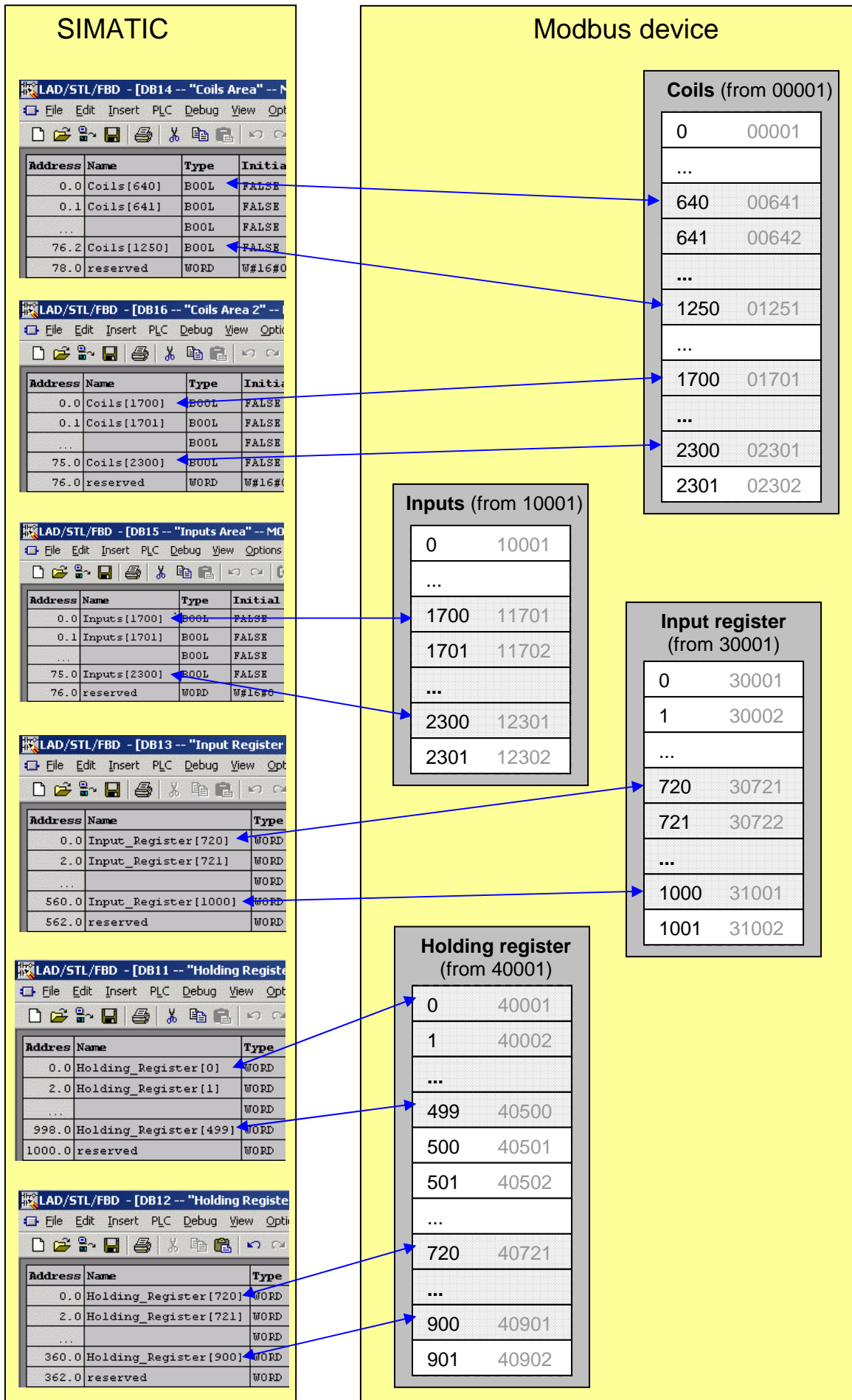
- DB11 is 1002 bytes; a total of 500 registers are mapped (register 0 – register 499) + 2 reserved bytes.
- DB12 is 364 bytes; a total of 181 registers are mapped (register 720 – register 900) + 2 reserved bytes
- DB13 is 564 bytes; a total of 281 input registers are mapped (register 720 – register 1000) + 2 reserved bytes
- DB14 is 80 bytes; a total of 611 coils (bits) are mapped (coil 640 – coil 1250) + 2 reserved bytes
- DB15 is 78 bytes; a total of 601 inputs (bits) are mapped (input 1700 – input 2300) + 2 reserved bytes
- DB16 is 78 bytes; a total of 601 coils (bits) are mapped (coil 1700 – coil 2300) + 2 reserved bytes

Address mapping

The figure below shows the Simatic memory areas and the register- and bit-based definition of memory in the Modbus devices. The figure is based on the parameter assignment above.

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

In SIMATIC: The SIMATIC addresses in the first column are the offset in the DB. The Modbus register numbers are in the square brackets.



8.4 Data and standard functions used by the FB

Instance DB	<p>The MB_PNHSV function block saves the data in an instance DB. This instance DB is generated by STEP 7 the first time the FB is called.</p> <p>The instance data block contains parameters of the type input and output as well as static variables required for its execution. These variables are remanent and remain valid between FB calls. The variables control the internal process of the FB.</p> <p>Memory requirement of the instance DB:</p> <table border="1"> <thead> <tr> <th>Instance DB</th> <th>Work memory</th> <th>Load memory</th> </tr> </thead> <tbody> <tr> <td>MB_PNHSV</td> <td>approx. 3 KB</td> <td>approx. 5 KB</td> </tr> </tbody> </table>	Instance DB	Work memory	Load memory	MB_PNHSV	approx. 3 KB	approx. 5 KB
Instance DB	Work memory	Load memory					
MB_PNHSV	approx. 3 KB	approx. 5 KB					
Local variables	A maximum total of 186 bytes of local data is required for an MB_PNHSV FB call.						
Parameter DB	The connection-specific and Modbus-specific parameters are saved in the MODBUS_HPAMM_PN parameter DB.						
Timers	The function block does not use any timers.						
Flags	The function block does not use any flags.						
Standard FBs for processing connections	The TCP_COMM FB called in the MB_PNHSV/MOD_SERV FB uses the TCON and TDISCON blocks from the standard library to establish and terminate connections between the CPU and the communication partner.						
Standard FBs for data transfer	The TCP_COMM FB called in the MB_PNHSV/MOD_SERV FB uses the TSEND and TRCV blocks from the standard library for data transfer between the CPU and the communication partner.						
MB_PNHSV: SFCs for other functions	<p>The MB_PNHSV FB uses the following SFCs from the standard library:</p> <ul style="list-style-type: none"> • SFC6 "RD_SINFO" • SFC20 "BLKMOV" • SFC24 "TEST_DB" • SFC51 "RDSYSST" • SFC52 "WR_USMSG" 						
MOD_SERV: SFCs for other functions	<p>The MOD_SERV FB uses the following SFCs from the standard library:</p> <ul style="list-style-type: none"> • SFC20 "BLKMOV" • SFC24 "TEST_DB" 						
TCP_COMM: SFCs for other functions	<p>The TCP_COMM FB uses the following SFB from the standard library as well as the T blocks:</p> <ul style="list-style-type: none"> • SFB4 "TON" 						

8.5 Renaming / rewiring functions and function blocks

Motive

If the numbers of the standard functions are already being used in your project, or the number range is reserved for other applications (e.g. in CFC), you can rewire the internally called FB63, FB64, FB65 and FB66 function blocks of the TCP_COMM FB, or the MB_PNHSV, MOD_SERV and TCP_COMM blocks.

The system functions SFC6, SFC20, SFC24, SFC51 and SFC52 and the system function block SFB4 cannot be renamed/rewired.

Reaction

A set of rules concerning the function block numbering have to be considered when rewiring function blocks in SIMATIC STEP 7 Manager:

To rewire blocks from the Modbus library, proceed in this order:

1. FB917 MB_PNHSV
2. FB916 MOD_SERV
3. FB913 TCP_COMM
4. FB63 TSEND
FB64 TRCV
FB65 TCON
FB66 TDISCON

You do not need to rewire all functions or all function blocks. However, you must work in this order even if you are only rewiring some of them.

Rewiring

Proceed as follows to rewire FBs:

1. Information on the addresses used is found under "Options > Reference data > Display".
2. Set the address priority to "Absolute value" in the object properties of the block folder.
3. In SIMATIC Manager, select the block folder and open the "Options > Rewire" function to rewire the addresses to free areas.
4. To continue using the symbols in diagnostic tools, apply the changes to the symbol table.

The modifications can be verified by "Options > References data > Display".

9 Additional blocks

9.1 Support in CFC

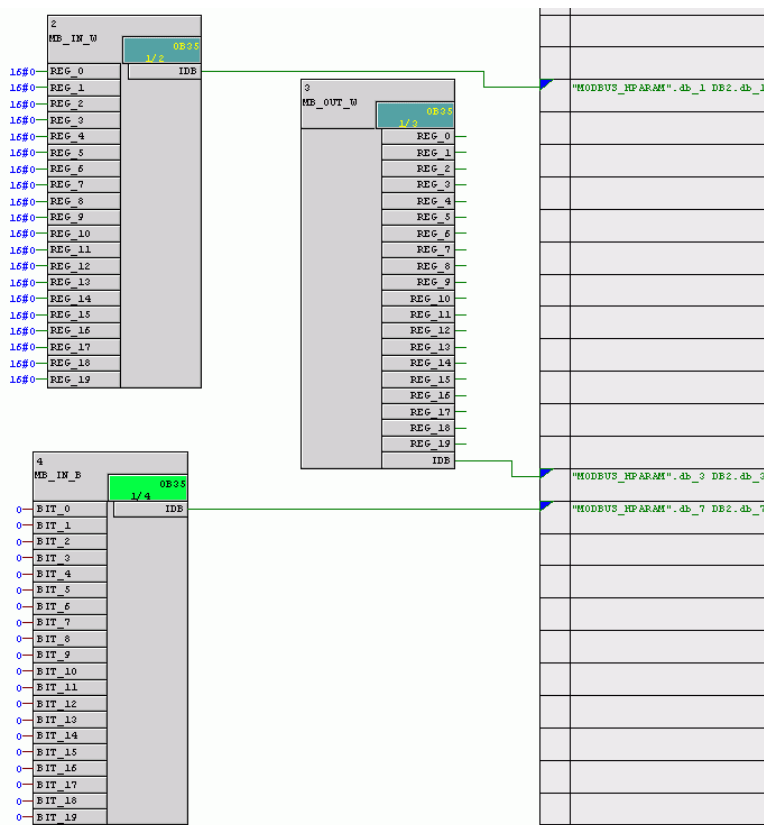
General information

To facilitate configuration in CFC, there is the option of configuring the Modbus values using "DataCollector FBs" instead of global FBs. The send and receive buffers are shifted to the CFC chart using drag & drop.

Application example

The DataCollector FBs are placed in the CFC chart. The "IDB" output is connected to DB parameters db_1 to db_8 in the parameter data block.

The Modbus values can then subsequently be connected straight from the channel blocks to the DataCollector FB.



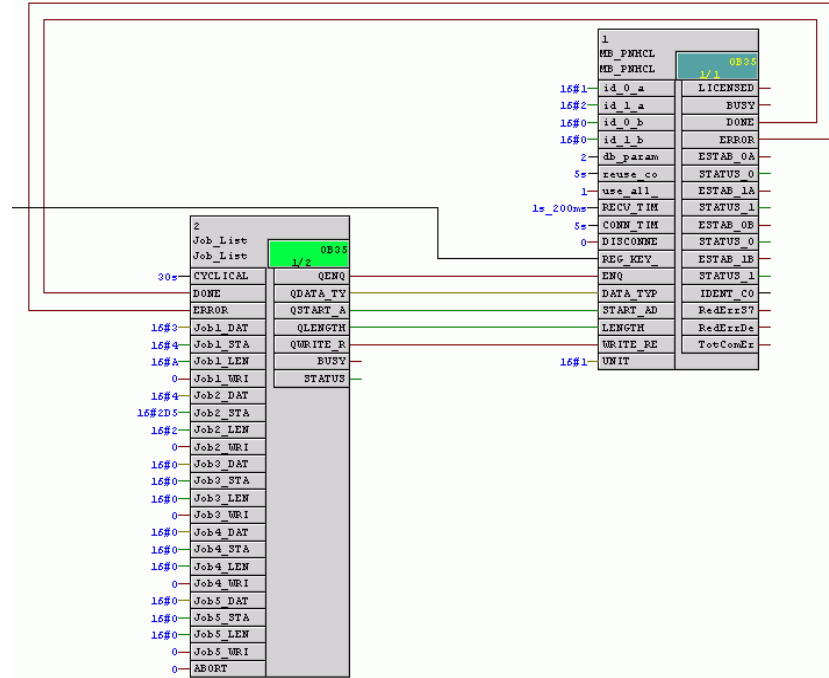
You will find the supplementary blocks and a detailed description here: www.siemens.com/s7modbus or from Customer Support.

9.2 Job list for cyclic telegrams

General information

The Job_List block allows you to configure a list of jobs to be processed cyclically.

Application example



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

10 Use in a single PN CPU

General information

The "Modbus/TCP PN CPU Redundant" package can also be operated in an S7 PN Single CPU (S7-400, S7-300, ET200S).

The description of the functions and parameters in the sections above and below also apply to application in a single PN CPU.

Modules suitable for MB_PNHCL and MB_PNHSV

You must use CPUs that can provide sufficient local data for each priority class (=> Section 7.4 and Section 8.4).

You also need to check the CPU-specific limits such as the maximum FB number.

Modbus blocks for single PN CPUs

The same FBs are used with a single PN CPU as with high-availability PN CPUs.

Make sure that you use the correct `local_device_id` for configuration. All other CPU-specific parameters must also be checked, and modified manually if necessary.

11 Diagnostics

Diagnostics functions

The diagnostics functions of the PN CPU enable you to quickly locate any errors that occur. The following diagnostic functions are available:

- Diagnostics via the display elements of the CPUs
- Diagnostics via the STATUS outputs of the Modbus function blocks.
- Diagnostics via the alarm bits of the Modbus function blocks

Display elements (LED)

The display elements provide information on the operating status and/or any errors of the CPUs. The display elements give a broad overview of internal, external and interface-specific errors.

MB_PNHCL and MB_PNHSV STATUS outputs

The MB_PNHCL and MB_PNHSV function blocks have STATUS outputs for error diagnostics. Reading the STATUS outputs gives you general information on errors that occurred during communication. You can evaluate the STATUS parameters in the user program.

MB_PNHCL and MB_PNHSV alarm bits

The Modbus function blocks also have outputs to display the loss of redundancy and complete loss of communication. The alarm bits are set in accordance with the status of the ESTAB_x outputs of the configured connections.

11.1 Diagnostics via the display elements of the CPU

Display functions

The CPU display elements provide information on the module status. The following display functions are used:

- **Group error displays**

PN CPU 300 and IM 151-8 PN/DP CPU

- SF Group error

If this LED flashes, the Modbus block has not yet been licensed. For additional information, see the section "**Licensing**".

PN-(H)CPU 400

- INTF Group error

If this LED flashes, the Modbus block has not yet been licensed. For additional information, see the section "**Licensing**".

- **Special displays**

PN CPU 300, PN-(H)CPU 400 and IM 151-8 PN/DP CPU:

- RX/TX A telegram is being transferred via the interface

A detailed description of the display elements can be found in the corresponding CPU manuals.

11.2 MB_PNHCL and MB_PNHSV FB diagnostics messages

Messages at the STATUS output of the FBs

The error messages are displayed at the status outputs of the MB_PNHCL and MB_PNHSV. Below you can find a list of the FB-specific error messages.

Error messages of the SFCs and FCs called

The Modbus FBs use the standard blocks SFC6, SFC20, SFC24, SFC51 and SFC52. The error messages of these blocks are forwarded unchanged to STATUS_x.

The TCP_COMM FB called in MOD_CLI or MOD_SERV uses the standard blocks SFB4, FB63, FB64, FB65 and FB66. The error messages of these blocks are also forwarded unchanged to STATUS_x.

You will find more information on these error messages in the diagnostics buffer or in the online help for the SFCs/FCs in SIMATIC Manager.

Error messages for the MB_PNHCL and MB_PNHSV FBs		
STATUS (Hex)	Event text	Remedy
A001	The parameter DB MODBUS_HPPARAM_PN is too short or too long.	Correct the MODBUS_HPPARAM_PN DB.
A002	The <i>end_x</i> parameter is less than <i>start_x</i> .	Correct the information in the MODBUS_HPPARAM_PN DB.
A003	A DB to which the MODBUS addresses are to be mapped is too short. Minimum length: - register values: $(end_x - start_x + 1) * 2 + 2$ - bit values: $(end_x - start_x) / 8 + 1 + 2$ Other possible causes: <ul style="list-style-type: none"> With MB_PNHCL: Incorrect call parameters With MB_PNHSV: Incorrect address area in the client's request telegram. The S7 responds with an exception telegram. 	Extend the DB. With MB_PNHCL: Correct the START_ADDRESS or LENGTH job parameters. With MB_PNHSV: Change the client request.
A004	With MB_PNHCL only: An invalid combination of DATA_TYPE and WRITE_READ has been specified.	Correct the call parameters. Only data types 1 and 3 can be written.
A005	With MB_PNHCL: Invalid value entered at the LENGTH parameter. With MB_PNHSV: Invalid number of registers/bits in the request telegram. The S7 responds with an exception telegram. <u>Ranges:</u> Read coils/inputs: 1 to 2000 Write coils: 1 to 1968 Read registers: 1 to 125 Write holding registers: 1 to 123	With MB_PNHCL: Correct the LENGTH parameter. With MB_PNHSV: Change the number in the client request telegram.
A006	The range specified with DATA_TYPE, START_ADDRESS and LENGTH does not exist in <i>data_type_1</i> to <i>data_type_8</i> . With MB_PNHSV: The S7 responds with an exception telegram.	With MB_PNHCL: Correct the DATA_TYPE, START_ADDRESS and LENGTH combination. With MB_PNHSV: Change the client request or correct the parameter assignment at <i>data_type_x</i> .
A007	With MB_PNHCL: Invalid monitoring time configured at RECV_TIMEOUT or CONN_TIMEOUT. A value ≥ 20 ms must be entered for RECV_TIMEOUT and a value ≥ 100 ms for CONN_TIMEOUT.	Correct the parameter assignment.
A009	With MB_PNHCL: Transaction Identifier TI received does not correspond to TI sent. The connection is terminated.	Verify the data of the communication partner with the help of a telegram trace.
A00A	With MB_PNHCL: UNIT received does not correspond to UNIT sent.	

Error messages for the MB_PNHCL and MB_PNHSV FBs		
STATUS (Hex)	Event text	Remedy
A00B	With MB_PNHCL: Function code received does not correspond to function code sent. With MB_PNHSV: Invalid function code received. The S7 responds with an exception telegram.	With MB_PNHCL: Check the communication partner using telegram recording. With MB_PNHSV: Change the client request. The Modbus FB processes function codes 1, 2, 3, 4, 5, 6, 15 and 16.
A00C	The byte count received does not match the number of registers. The connection is terminated.	Verify the data of the communication partner with the help of a telegram trace.
A00D	With MB_PNHCL: Register/bit address or number of registers/bits in the response telegram does not correspond to that in the request telegram.	
A00E	Length in the Modbus-specific telegram header does not correspond to the specified number of registers/bits or specified byte count in the telegram. The FB rejects the data. The connection is terminated.	
A00F	Protocol Identifier not equal to 0 received. The connection is terminated.	
A010	DB number assigned twice in parameters <i>db_1</i> to <i>db_8</i> .	Correct the parameter assignment in the MODBUS_HPARAM_PN DB.
A011	Invalid value given at the DATA_TYPE input parameter (valid values: 1 - 4).	Correct the call parameters.
A012	The configured areas <i>data_type_1</i> and <i>data_type_2</i> overlap.	Correct the parameter assignment. The data areas must not share registers.
A013	The configured areas <i>data_type_1</i> and <i>data_type_3</i> overlap.	
A014	The configured areas <i>data_type_1</i> and <i>data_type_4</i> overlap.	
A015	The configured areas <i>data_type_1</i> and <i>data_type_5</i> overlap.	
A016	The configured areas <i>data_type_1</i> and <i>data_type_6</i> overlap.	
A017	The configured areas <i>data_type_1</i> and <i>data_type_7</i> overlap.	
A018	The configured areas <i>data_type_1</i> and <i>data_type_8</i> overlap.	Correct the parameter assignment at <i>db_x</i> to > 0.
A019	One of the <i>db_x</i> parameters was set to 0 although the corresponding <i>data_type_x</i> is configured as > 0. DB0 may not be used because it is reserved for the system.	
A01A	Incorrect length in header: Values 3 to 253 bytes are possible. The connection is terminated.	Verify the data of the communication partner with the help of a telegram trace.

Error messages for the MB_PNHCL and MB_PNHSV FBs		
STATUS (Hex)	Event text	Remedy
A01B	With MB_PNHSV and function code 5: Invalid status received for coil. The S7 responds with an exception telegram.	Verify the data of the communication partner with the help of a telegram trace.
A020	An invalid value is configured at reuse_conn_cycle.	Correct the parameter assignment. A monitoring time of > 1s must be configured.
A022	A parameter data block for clients has been configured at the MB_PNHSV or a parameter data block for servers has been configured at the MB_PNHCL.	Correct the parameter assignment of the parameter data block.
A023	The configured areas data_type_2 and data_type_3 overlap.	Correct the parameter assignment. The data areas must not share registers.
A024	The configured areas data_type_2 and data_type_4 overlap.	
A025	The configured areas data_type_2 and data_type_5 overlap.	
A026	The configured areas data_type_2 and data_type_6 overlap.	
A027	The configured areas data_type_2 and data_type7 overlap.	
A028	The configured areas data_type_2 and data_type8 overlap.	
A034	The configured areas data_type_3 and data_type_4 overlap.	
A035	The configured areas data_type_3 and data_type_5 overlap.	
A036	The configured areas data_type_3 and data_type_6 overlap.	
A037	The configured areas data_type_3 and data_type_7 overlap.	
A038	The configured areas data_type_3 and data_type_8 overlap.	
A045	The configured areas data_type_4 and data_type_5 overlap.	
A046	The configured areas data_type_4 and data_type_6 overlap.	
A047	The configured areas data_type_4 and data_type_7 overlap.	
A048	The configured areas data_type_4 and data_type_8 overlap.	
A056	The configured areas data_type_5 and data_type_6 overlap.	
A057	The configured areas data_type_5 and data_type__7 overlap.	
A058	The configured areas data_type_5 and data_type__8 overlap.	

Error messages for the MB_PNHCL and MB_PNHSV FBs		
STATUS (Hex)	Event text	Remedy
A067	The configured areas <i>data_type_6</i> and <i>data_type_7</i> overlap.	
A068	The configured areas <i>data_type_6</i> and <i>data_type_8</i> overlap.	
A078	The configured areas <i>data_type_7</i> and <i>data_type_8</i> overlap.	
A079	The connection ID specified at the parameter <i>id_x</i> does not exist in the MODBUS_HPAM_PN parameter DB.	Correct the parameter assignment at input <i>id_x</i> .
A07A	Invalid value assigned (value range of 1 to 4095) or value assigned twice at parameter <i>id_x</i> of the block.	Correct the parameter assignment in the MODBUS_HPAM_PN DB.
A07B	The <i>id_x</i> specified exists twice in the parameter DB.	
A07C	Invalid value specified at parameter <i>data_type_x</i> in the parameter DB (valid values are 0 to 4).	
A07D	No entry in the <i>data_type_1</i> parameter in the parameter DB. Parameter area "_1" is the initial area and must be configured.	
A07E	The number of the instance DB of block MB_PNHCL or MB_PNHSV was specified at <i>db_x</i> .	
A07F	The DB specified at <i>db_param</i> is not a Modbus parameter DB.	Correct the parameter assignment at input <i>db_param</i> .
A080	Different instance DBs are used to call the Modbus block in cyclic OB and in OB100. Or the instance DB was transferred to the CPU without restart.	The Modbus block must be called with the same instance DB in the startup OB and the cyclic OB. After the transfer of the IDB to the CPU an initialization of the Modbus block is needed.
A081	Only with MB_PNHCL and function code 5: Response telegram data does not echo request.	Verify the data of the communication partner with the help of a telegram trace.
A082	Only with MB_PNHCL and function code 6: Register value received does not correspond to register value sent.	
A083	With MB_PNHCL: A new job was triggered before the current one was completed. The job is not executed. This is a status information. The ERROR bit is not set. Manual block initialization was started while a job was in progress.	Do not trigger a new job until the previous job has completed with DONE = TRUE or ERROR = TRUE. Do not initialize until all running jobs have completed.
A084	No IDENT_CODE identifier could be calculated for licensing.	Please contact the Product Support.
A085	An error occurred during license detection. The error is displayed the first time it occurs with ERROR = TRUE. Thereafter, it appears as a status message with ERROR = FALSE.	Check for unauthorized write access to the license DB. The REG_KEY structure must not be changed. Contact the Product Support in case of problems.

Error messages for the MB_PNHCL and MB_PNHSV FBs		
STATUS (Hex)	Event text	Remedy
A086	An attempt was made to write to a write-protected data block.	Remove the data block write protection or use another DB.
A090	The Modbus block has not yet been licensed for this CPU. This is status information. The ERROR bit is not set. Modbus communication will also run without a license.	Read the IDENT_CODE identification string for this CPU and request the registration key.
A091	With MB_PNHCL only: An exception telegram with exception code 1 was received in response.	The communication partner does not support the requested function.
A092	With MB_PNHCL only: An exception telegram with exception code 2 was received in response. Access to a non-existent/invalid address at the communication partner.	Correct the LENGTH or START_ADDRESS for the FB call.
A093	With MB_PNHCL only: An exception telegram with exception code 3 was received in response.	The communication partner cannot process the telegram received (for example, it does not support the requested length).
A094	With MB_PNHCL only: An exception telegram with exception code 4 was received in response.	The communication partner cannot process the telegram received in its current status.
A095	With MB_PNHCL only: An exception telegram with an unknown exception code was received in response.	Check the communication partner error messages and if necessary verify the data of the communication partner with the help of a telegram trace.
A0FF	The connection is not currently established. This is status information.	Check the connections. Correct the value at reuse_conn_time if necessary.
A100	The CONN_TIMEOUT or RECV_TIMEOUT monitoring time for a job has expired. The connection is terminated when RECV_TIMEOUT elapsed.	Check the parameter assignment of the connection.
A101	The internal monitoring time of the TDISCON function has expired.	Contact the Product Support.
FFFF	The connection has not been configured.	If this connection is to be used, it must be configured in startup at id_x.

11.3 Diagnostics messages of integrated blocks

Error messages of integrated FCs/SFCs		
STATUS (Hex)	Event text	Remedy
7xxx	Please see the SIMATIC Manager online help for detailed information.	See online help (SIMATIC Manager -> Select block -> Press F1)
8xxx	Please see the SIMATIC Manager online help for detailed information.	See online help (SIMATIC Manager -> Select block -> Press F1)
80C4	The H system is in the process of connecting and updating.	This error message from TCON may occur once after a warm restart of the H system and can be ignored.

11.4 SFC24 diagnostics messages

Error messages of the SFC24		
STATUS (Hex)	Event text	Remedy
80A1	DB number = 0 or too large for the CPU	Select a valid DB number.
80B1	The DB does not exist in the CPU.	All data blocks specified at DB_x must be created and transferred to the CPU.
80B2	DB UNLINKED	Do not generate DB as UNLINKED.

11.5 Diagnostic messages with alarm bits

The Modbus blocks allow you to detect a loss of redundancy. This is displayed at RedErrS7, RedErrDev and TotComErr outputs. These status bits can be connected to an alarm block or to other blocks, where they can be evaluated.

The alarm bits are set in accordance with the status of the configured connections at ESTAB_x.

11.5.1 Client block

The alarm bits are set as follows in line with the parameter assignment:

- 1) use_all_conn = FALSE

The telegrams are only sent and received via one connection; the other configured connections are on standby. Once the configured "reuse_conn_time" has elapsed, the system attempts to establish the faulty connections again.

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

2) use_all_conn = TRUE; S7 is redundant; the communication partner is set up as standalone

The telegrams are sent and received via 2 configured connections. Once the configured "reuse_conn_time" has elapsed, the system attempts to establish the faulty connections again.

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

3) use_all_conn = TRUE; S7 is standalone; the communication partner is set up as redundant

The telegrams are sent and received via 2 configured connections. Once the configured "reuse_conn_time" has elapsed, the system attempts to establish the faulty connections again.

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

4) use_all_conn = TRUE; 4 connections are configured

The telegrams are sent and received via 4 configured connections. Once the configured "reuse_conn_time" has elapsed, the system attempts to establish the faulty connections again.

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

11.5.2 Server block

The server block tries to establish faulty connections again on a cyclic basis.

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

12 Application example

General information

During installation, two example projects are stored in \Program Files\Siemens\Step7\Examples:

- Sample project in STL, "MB_TCP_PN_RED_400" and
- Sample project in CFC "MB_TCP_PN_RED_CFC".

The example projects contain Simatic stations for all function variants:

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

The S7 program is provided for information only and should not be considered as a recommended solution for customer-specific plant configurations.

Simatic stations in the example project

The example project contains 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

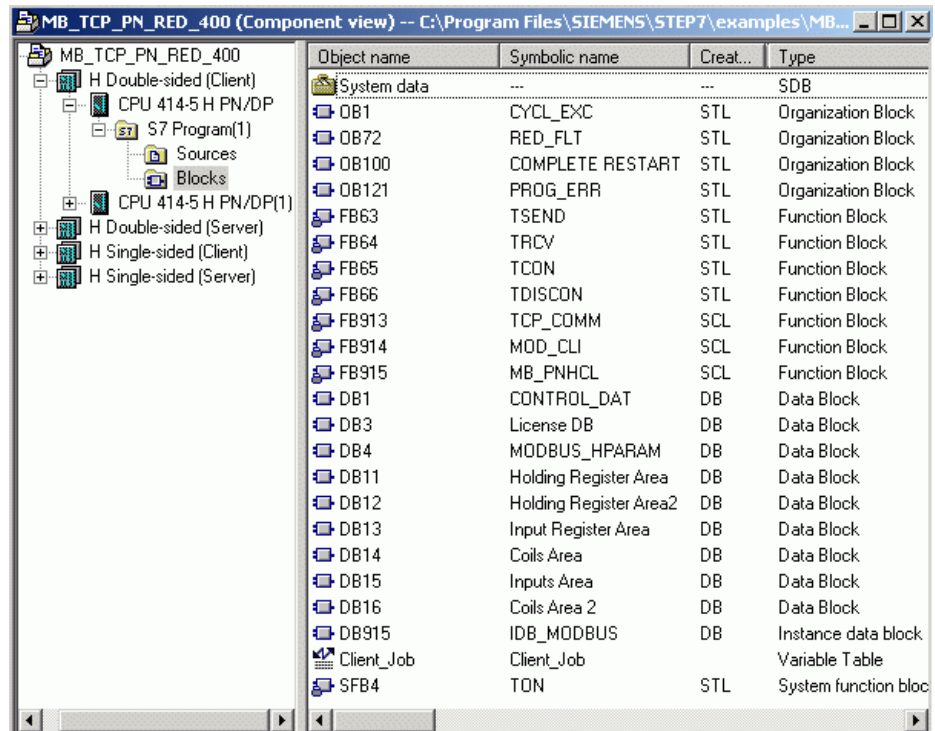
Program example

The program examples consist of the following blocks:

- Startup block OB100 with call of FB915 or FB917
- Programming error OB121
- Cyclic operation OB1 or OB35 with call of FB915 or FB917
- Global data blocks for triggering jobs (e.g. with a variable table) and for licensing
- Data blocks for register and bit values

12.1 Sample project in STL – Modbus client

Overview



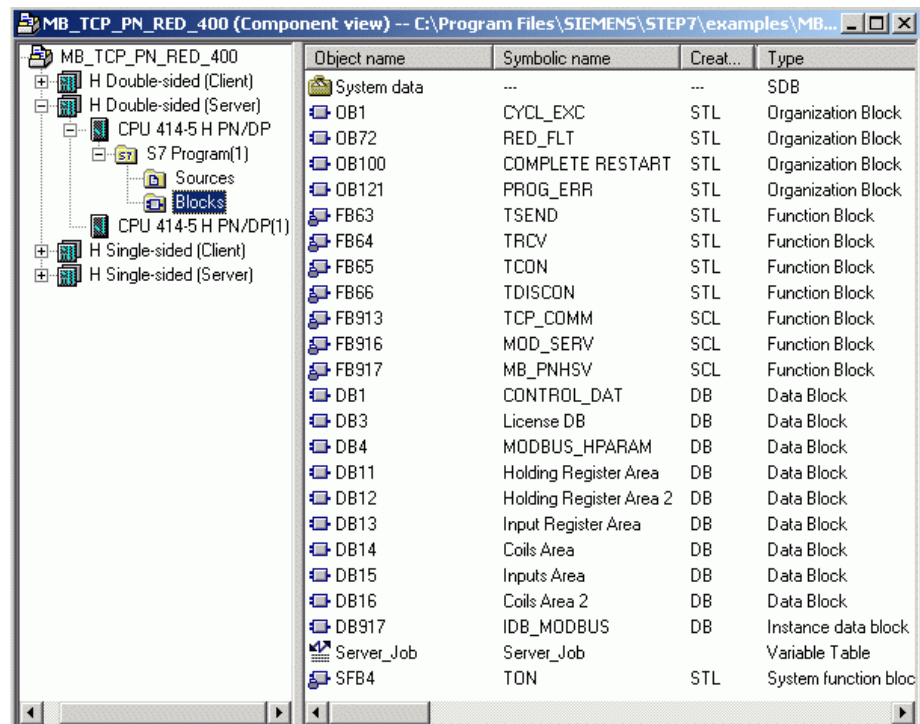
Blocks used

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

Block	Symbol	Comment
OB 1	CYCL_EXC	Cyclic program processing
OB 100	COMPLETE RESTART	Startup OB for restart
OB 121	PROG_ERR	Programming error OB
FB 913	TCP_COMM	Internally called FB TCP_COMM
FB 914	MOD_CLI	Internally called FB MOD_CLI
FB 915	MB_PNHCL	User block FB MB_PNHCL
DB 1	CONTROL_DAT	Work DB CONTROL DAT for FB MB_PNHCL
DB 3	LICENSE_DB	License DB for FB MB_PNHCL
DB 4	MODBUS_HPARAM_P N	Parameter DB for FB MB_PNHCL
DB 11	Holding register area	Value DB for area 1
DB 12	Holding register area 2	Value DB for area 2
DB 13	Input register area	Value DB for area 3
DB 14	Coils area	Value DB for area 5
DB 15	Inputs area	Value DB for area 6
DB 16	Coils area 2	Value DB for area 7
DB 915	IDB_MODBUS	Instance DB for FB MB_PNHCL

12.2 Sample project in STL – Modbus server

Overview



Blocks used

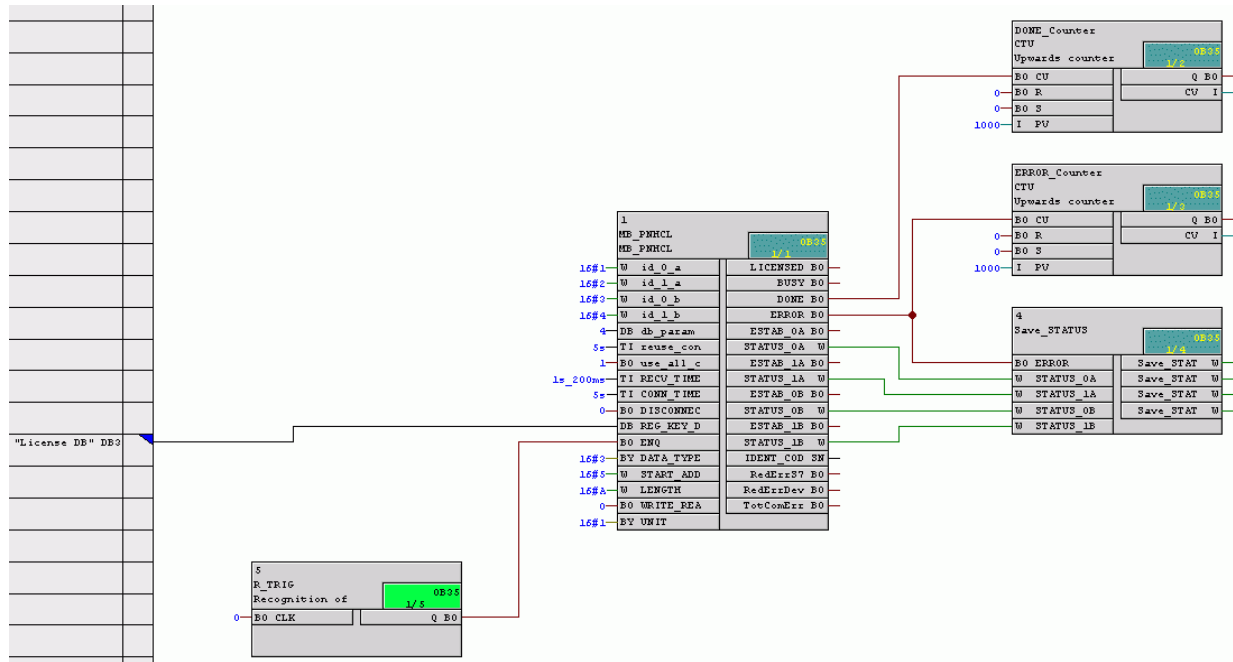
These block numbers are also used in the example project supplied for S7 H stations with FB MB_PNHSV.

Block	Symbol	Comment
OB 1	CYCL_EXC	Cyclic program processing
OB 100	COMPLETE RESTART	Startup OB for restart
OB 121	PROG_ERR	Programming error OB
FB 913	TCP_COMM	Internally called FB TCP_COMM
FB 916	MOD_SERV	Internally called FB MOD_SERV
FB 917	MB_PNHSV	User block FB MB_PNHSV
DB 1	CONTROL_DAT	Work DB CONTROL DAT for FB MB_PNHSV
DB 3	LICENSE_DB	License DB for FB MB_PNHSV
DB 4	MODBUS_HPAM_P N	Parameter DB for FB MB_PNHSV
DB 11	Holding register area	Value DB for area 1
DB 12	Holding register area 2	Value DB for area 2
DB 13	Input register area	Value DB for area 3
DB 14	Coils area	Value DB for area 5
DB 15	Inputs area	Value DB for area 6
DB 16	Coils area 2	Value DB for area 7
DB 917	IDB_MODBUS	Instance DB for FB MB_PNHSV

12.3 Sample project in CFC – Modbus client

Overview

The example project was generated with CFC V8.0 Update 1.



Blocks used

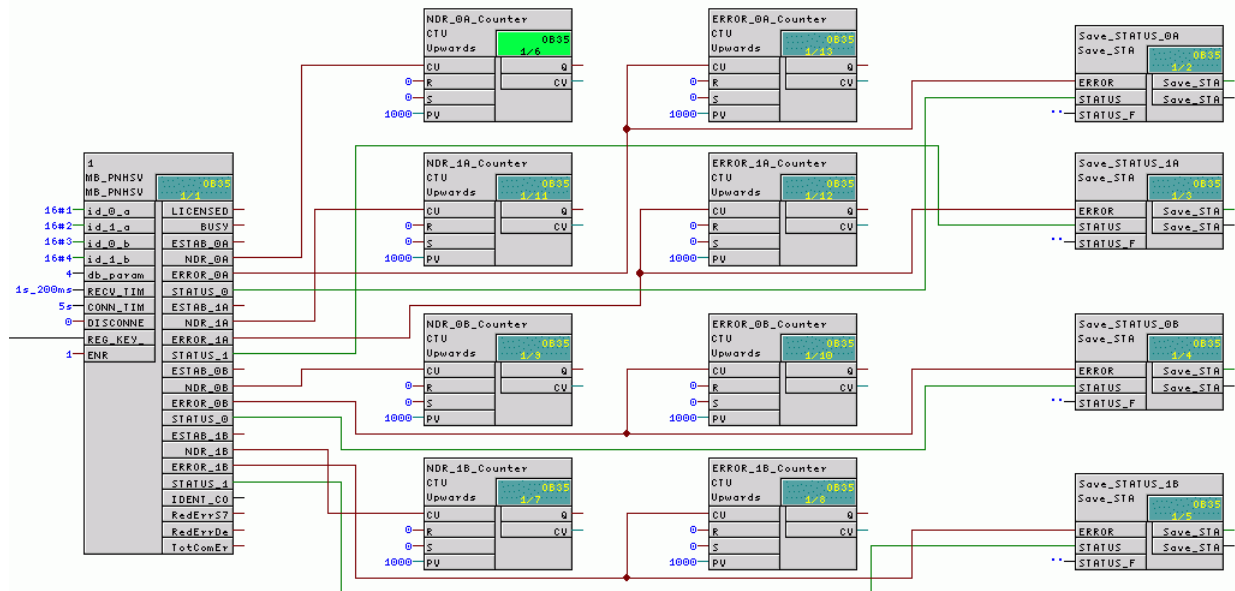
These block numbers are also used in the example project supplied for S7 H stations with FB MB_PNHCL.

Block	Symbol	Comment
OB 35	CYCL_EXC	Cyclic program processing
OB 100	COMPLETE RESTART	Startup OB for restart
OB 121	PROG_ERR	Programming error OB
FB 99	Save_STATUS	Memory-DB for errors
FB 913	TCP_COMM	Internally called FB TCP_COMM
FB 914	MOD_CLI	Internally called FB MOD_CLI
FB 915	MB_PNHCL	User block FB MB_PNHCL
DB 3	LICENSE_DB	License DB for FB MB_PNHCL
DB 4	MODBUS_HPARAM_P N	Parameter DB for FB MB_PNHCL
DB 11	Holding register area	Value DB for area 1
DB 12	Holding register area 2	Value DB for area 2
DB 13	Input register area	Value DB for area 3
DB 14	Coils area	Value DB for area 5
DB 15	Inputs area	Value DB for area 6
DB 16	Coils area 2	Value DB for area 7

12.4 Sample project in CFC – Modbus server

Overview

The example project was generated with CFC V8.0 Update 1.



Blocks used

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

Block	Symbol	Comment
OB 35	CYCL_EXC	Cyclic program processing
OB 100	COMPLETE RESTART	Startup OB for restart
OB 121	PROG_ERR	Programming error OB
FB 99	Save_STATUS	Memory-DB for errors
FB 913	TCP_COMM	Internally called FB TCP_COMM
FB 916	MOD_SERV	Internally called FB MOD_SERV
FB 917	MB_PNHSV	User block FB MB_PNHSV
DB 3	LICENSE_DB	License DB for FB MB_PNHSV
DB 4	MODBUS_HPARAM_P N	Parameter DB for FB MB_PNHSV
DB 11	Holding register area	Value DB for area 1
DB 12	Holding register area 2	Value DB for area 2
DB 13	Input register area	Value DB for area 3
DB 14	Coils area	Value DB for area 5
DB 15	Inputs area	Value DB for area 6
DB 16	Coils area 2	Value DB for area 7

A References

**MODBUS
organization**

MODBUS APPLICATION PROTOCOL SPECIFICATION
V1.1b3, April 26, 2012

<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