SIEMENS

MODBUSPN: Communicate as Modbus/TCP client or Modbus/TCP-Server 1

SIMATIC

MODBUS (TCP)

Programming and Operating Manual

06/2014

Legal information

Warning notice system

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

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

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

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

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

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

Qualified Personnel

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

Proper use of Siemens products

Note the following:

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Ⓟ 04/2017 Subject to change

Table of contents

MODBUSPN: Communicate as Modbus/TCP client or Modbus/TCP-Server

1

1.1 General Information

General

This instruction MODBUSPN represents a software product for CPUs with integrated PN interface of the SIMATIC S7-300, S7-400 and IM 151-8 PN/DP CPU.

This instruction enables communication between a SIMATIC CPU with integrated PN interface and a device which supports the Modbus/TCP protocol.

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

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

Step-by-step instructions

1. Assign the CPU IP address
2. Call the MODBUSPN instruction in the necessary OBs - see Commissioning (Page 5)
3. Assign parameters of the parameter DB according to the requirements (ID, port number, client/server, connection established at restart, Modbus registers, DB areas, etc.) - see Parameter data block (Page 7)
4. Assign parameters of Modbus block for initialization and for runtime - see Operating principle of the instruction (Page 12)
5. Download the user program to the CPU and licensing of the Modbus block for this CPU - see Licensing (Page 16)

1.2 Commissioning

Requirement and basics

The MODBUSPN instruction can be used as of **STEP7 V13 (TIA Portal)** .

The MODBUSPN instruction is based on the Modbus Application Protocol Specification V1.1b3, April 26, 2012 - see Modbus home page (http://www.modbus.org).

Call of the instruction

The MODBUSPN instruction must be installed in two OBs for correct program execution:

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

The same instance data block must be used in both. It is not permitted to call the MODBUSPN instruction in OB1 and in a time-controlled OB (e.g., OB35) at the same time. The OB121 must be present in the CPU. For more detailed information see Licensing (Page 16).

Inserting the Modbus block

Open the "COMPLETE RESTART [OB100]" block. If this block does not exist in the program blocks, insert it with "Insert new block > Organization block > Startup > COMPLETE RESTART [OB 100]".
The MODBUSPN instruction exists in the Task Card, palette and "Instructions > Communication > Other" folder. In it, open the "MODBUS TCP" folder and drag the MODBUSPN instruction in the OB100 block.

Under "System blocks > Program resources", the lower-level instructions MOD_CLI (FB72), MOD_SERV (FB73) and TCP_COMM (FB71) are displayed in addition to MODBUSPN (FB70) . These may not also be called in an OB. In addition, the internally called communication instructions TSEND (FB63), TRCV (FB64), TCON (FB65) and TDISCON (FB66) are displayed.

Note

Note that the following versions are required for trouble-free operation of the MODBUSPN instruction:

TSEND	V3.0
TRCV	V3.0
TCON	V3.0
TDISCON	V2.1

Open the "Main [OB1]" block or a cyclic block and drag the MODBUSPN [FB70] instruction in the OB. Select the MODBUSPN_DB data block from the OB100 call as instance data block. Do not create a new instance data block.

Multiple client and server connections

An S7 CPU can support several TCP connections whereby the maximum number of connections depends on the CPU being used. The total number of connections of one CPU including Modbus/TCP connections must not exceed the maximum number of supported connections.

Use of the port number 502

The Modbus/TCP protocol usually runs via port 502. This port number is only available for PN CPUs with the corresponding firmware version. Information regarding enabling of port numbers is available here: "Which ports are released for Modbus/TCP communication and how many Modbus clients can communicate with a SIMATIC S7 CPU as Modbus server? (http://support.automation.siemens.com/WW/view/en/34010717)".

Specific CPU types can maintain and operate connections to multiple clients (multiport) simultaneously via the local port 502. The following settings must be made during parameter assignment:

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

The number of connections that a CPU can accept on port 502 depends on the device and is available in the technical specifications of the CPU. One unique connection in the parameter DB and one Modbus block instance each in OB100 or cyclic OB is required for each client that wants to connect to port 502 of the server.

1.3 Parameter data block

Parameter assignment of the Modbus communication

You do not need to configure a connection in the network editor for communication using the integrated PN interface of the CPU. The connections are established and terminated with the help of the TCON and TDISCON instructions.

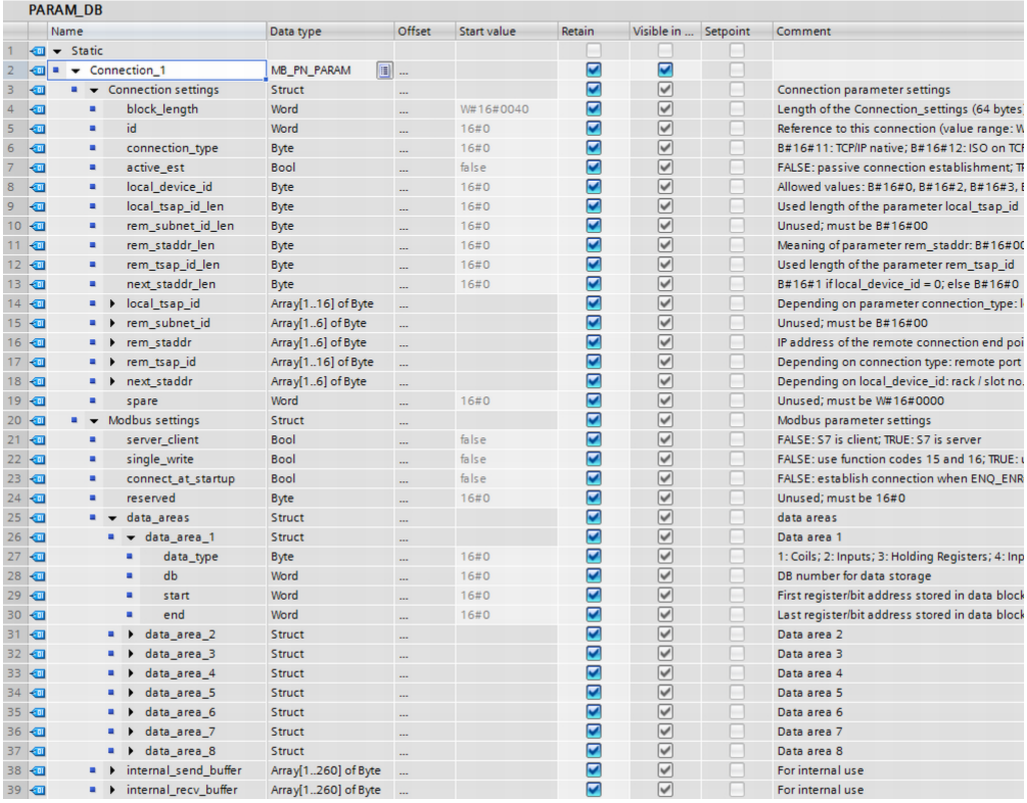
Parameter data block

The data required for establishing the connections and processing the Modbus messages are defined in the PLC data type MB_PN_PARAM. This PLC data type includes a structure for the connection-specific data and a structure for the Modbus parameters.

One instance of the PLC data type is required in a data block for each connection to a communication partner; in it you define the connection parameters and the Modbus parameters. The data block can be expanded for each additional connection or you can create a new data block.

This data block or these data blocks are intended for connection and Modbus parameters only; do not use it/them to save any other parameters.

Each instance of the "MODBUSPN" instruction requires a unique connection. Create a separate structure of the connection description for each instance of the instruction.



PARAM_DB

	Name	Data type	Offset	Start value	Retain	Visible in ...	Setpoint	Comment
1	Static				☐	☐	☐	
2	Connection_1	MB_PN_PARAM	...		☑	☑	☐	
3	Connection settings	Struct	...		☑	☑	☐	Connection parameter settings
4	block_length	Word	...	W#16#0040	☑	☑	☐	Length of the Connection_settings (64 bytes
5	id	Word	...	16#0	☑	☑	☐	Reference to this connection (value range: W
6	connection_type	Byte	...	16#0	☑	☑	☐	B#16#11: TCP/IP native; B#16#12: ISO on TC
7	active_est	Bool	...	false	☑	☑	☐	FALSE: passive connection establishment; T
8	local_device_id	Byte	...	16#0	☑	☑	☐	Allowed values: B#16#0, B#16#2, B#16#3,
9	local_tsap_id_len	Byte	...	16#0	☑	☑	☐	Used length of the parameter local_tsap_id
10	rem_subnet_id_len	Byte	...	16#0	☑	☑	☐	Unused; must be B#16#00
11	rem_staddr_len	Byte	...	16#0	☑	☑	☐	Meaning of parameter rem_staddr: B#16#0
12	rem_tsap_id_len	Byte	...	16#0	☑	☑	☐	Used length of the parameter rem_tsap_id
13	next_staddr_len	Byte	...	16#0	☑	☑	☐	B#16#1 if local_device_id = 0; else B#16#0
14	local_tsap_id	Array[1..16] of Byte	...		☑	☑	☐	Depending on parameter connection_type: l
15	rem_subnet_id	Array[1..6] of Byte	...		☑	☑	☐	Unused; must be B#16#00
16	rem_staddr	Array[1..6] of Byte	...		☑	☑	☐	IP address of the remote connection end poi
17	rem_tsap_id	Array[1..16] of Byte	...		☑	☑	☐	Depending on connection type: remote port
18	next_staddr	Array[1..6] of Byte	...		☑	☑	☐	Depending on local_device_id: rack / slot no.
19	spare	Word	...	16#0	☑	☑	☐	Unused; must be W#16#0000
20	Modbus settings	Struct	...		☑	☑	☐	Modbus parameter settings
21	server_client	Bool	...	false	☑	☑	☐	FALSE: S7 is client; TRUE: S7 is server
22	single_write	Bool	...	false	☑	☑	☐	FALSE: use function codes 15 and 16; TRUE:
23	connect_at_startup	Bool	...	false	☑	☑	☐	FALSE: establish connection when ENQ_ENR
24	reserved	Byte	...	16#0	☑	☑	☐	Unused; must be 16#0
25	data_areas	Struct	...		☑	☑	☐	data areas
26	data_area_1	Struct	...		☑	☑	☐	Data area 1
27	data_type	Byte	...	16#0	☑	☑	☐	1: Coils; 2: Inputs; 3: Holding Registers; 4: Inp
28	db	Word	...	16#0	☑	☑	☐	DB number for data storage
29	start	Word	...	16#0	☑	☑	☐	First register/bit address stored in data block
30	end	Word	...	16#0	☑	☑	☐	Last register/bit address stored in data block
31	data_area_2	Struct	...		☑	☑	☐	Data area 2
32	data_area_3	Struct	...		☑	☑	☐	Data area 3
33	data_area_4	Struct	...		☑	☑	☐	Data area 4
34	data_area_5	Struct	...		☑	☑	☐	Data area 5
35	data_area_6	Struct	...		☑	☑	☐	Data area 6
36	data_area_7	Struct	...		☑	☑	☐	Data area 7
37	data_area_8	Struct	...		☑	☑	☐	Data area 8
38	internal_send_buffer	Array[1..260] of Byte	...		☑	☑	☐	For internal use
39	internal_recv_buffer	Array[1..260] of Byte	...		☑	☑	☐	For internal use

Figure 1-1 Param_DB structure

Connection parameters in the "Connection settings" structure

The connection-specific parameters, such as the local hardware interface and the IP address of the communication partner, are defined in the first structure "Connection settings". The instructions TCON and TDISCON use these parameters to establish or terminate a connection.

You have to strictly adhere to the data structure of the connection parameter block because the connection cannot be established otherwise.

Modbus parameters in the "Modbus settings" structure

The data required for operating mode and address reference are stored in the second structure, the "Modbus settings"; these include, for example, the Modbus areas which are mapped in the data blocks and the operating mode of the S7 as Modbus server or Modbus client. You must adhere to the data structure of the Modbus parameters because they cannot be processed correctly otherwise.

Configuration

You have two options for configuring the connection and Modbus parameters.

1. Option:
   Create a new global data block and open it. Add a parameter and select the data type **MB_PN_PARAM** for this parameter. If this data type is not displayed in the drop-down list, enter it manually.
   You can insert several instances in one data block with this option.
2. Option:
   Create a new data block with "Add new block" and select **MB_PN_PARAM** as "Type". The new data block with the inserted connection and Modbus structure opens.
   This block is read-only. You cannot add any additional parameters. Existing parameters can be edited.

Changing the values

You may not change the values in the parameter data block during runtime. The CPU must be restarted with STOP -> RUN after the parameters have been changed.

"Connection settings" connection parameters

Parameter	Description	
block_length	This parameter defines the length of the connection parameters and may not be altered. Fixed value: W#16#40	
id	A new connection ID is assigned for each logical connection. It must be unique throughout the entire parameter data block. The ID is specified when the MODBUSPN instruction is called; it 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	The connection type for establishing the connection is defined by the TCON instruction. The CPU determines which value has to be set. TCP (compatibility mode): B#16#01 forr 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, CPU414, CPU416 and WinAC RTX This information can vary depending on the firmware.	
active_est	This parameter refers to the type of connection establishment, active or passive. The Modbus client is responsible for the active connection establishment and the Modbus server for passive connection establishment. Active connection establishment: TRUE Passive connection establishment: FALSE	
local_device_id	The local_device_id defines the IE interface of the PN CPU in use. Different settings are required depending on the PN CPU type.	
	IM 151-8 PN/DP CPU, WinAC RTX, IF 1: CPU 314C, 315 or 317: CPU 319: CPU 412, 414 or CPU 416 WinAC RTX, IF 2 WinAC RTX, IF 3 WinAC RTX, IF 4	B#16#1 B#16#2 B#16#3 B#16#5 B#16#6 B#16#B B#16#F

Parameter	Description	
local_tsap_id_len	The length of the local_tsap_id parameter (= local port number) is specified. Active connection establishment: 0 Passive connection establishment: 2	
rem_subnet_id_len	This parameter is currently not in use and must be allocated with B#16#0.	
rem_staddr_len	The length of the rem_staddr parameter, which is the IP address of the communication partner, is specified. An IP address is not specified for the partner if communication is to take place via 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 defined here. For PN interface: B#16#0	
local_tsap_id	You use this parameter to set the local port number. The type of representation differs depending on the connection_type parameter. The CPU determines the value range. The port number must be unique on the CPU.	
	For connection_type B#16#01: local_tsap_id[1] local_tsap_id[2] local_tsap_id[3-16] For connection_type B#16#11: local_tsap_id[1] local_tsap_id[2] local_tsap_id[3-16]	low byte of the port number in hex format high byte of the port number in hex format B#16#00 high byte of the port number in hex format low byte of the port number in hex format B#16#00
rem_subnet_id	This parameter is currently not in use and must be allocated with 0.	
rem_staddr	The IP address of the remote communication partner is entered in this byte array. No IP address is entered in case of an unspecified connection. The type of representation depends on the connection_type parameter. Example: IP address 192.168.0.1:	
	For connection_type B#16#01: rem_staddr[1] = rem_staddr[2] = rem_staddr[3] = rem_staddr[4] = rem_staddr[5-6]= For connection_type B#16#11: rem_staddr[1] = rem_staddr[2] = rem_staddr[3] = rem_staddr[4] = rem_staddr[5-6]=	B#16#01 (1) B#16#00 (0) B#16#A8 (168) B#16#C0 (192) B#16#00 (reserved) B#16#C0 (192) B#16#A8 (168) B#16#00 (0) B#16#01 (1) B#16#00 (reserved)

Parameter	Description	
rem_tsap_id	You use this parameter to set the remote port number. The type of representation differs depending on the connection_type parameter. The CPU determines the value range.	
	For connection_type B#16#01: rem_tsap_id[1] rem_tsap_id[2] rem_tsap_id[3-16]	 low byte of the port number in hex format low byte of the port number in hex format B#16#00
	For connection_type B#16#11: rem_tsap_id[1] rem_tsap_id[2] rem_tsap_id[3-16]	 high byte of the port number in hex format low byte of the port number in hex format B#16#00
next_staddr	This parameter defines the rack and slot number of the CP in use. This parameter must be set to 0 when you use the integrated PN interface of the CPU. next_staddr[1-6] B#16#00	
spare	This parameter is not in use and must have the default value 0.	

"Modbus settings" Modbus parameters

Parameter	Description		
server_client	TRUE: S7 is server FALSE: S7 is client		
single_write	In operating mode "S7 is client", the function codes 5 and 6 are used with the single_write = TRUE parameter for write jobs with length 1. If single_write = FALSE, the function codes 15 and 16 are used for all write jobs.		
connect_at_startup	Specifies the time of connection establishment. If connect_at_startup is set to TRUE, the connection is established immediately after CPU restart. In this case, a data job may only be transmitted if the connection was established correctly (CONN_ESTABLISHED = TRUE) or if a corresponding error is displayed at ERROR and STATUS_CONN. FALSE: Connection establishment with set ENQ_ENR TRUE: Connection establishment immediately after restart		
8 Datenbereiche	There are eight data areas available for mapping the MODBUS addresses in the S7 memory. At least the first data area must be defined; the remaining seven data areas are optional. Depending on the type of job, data is either read from or written to the data areas. You can only read from one DB or write to one DB with any job. Access to registers or bit values that are located in several DBs, even if the numbers are in a sequence without gaps, are to be divided into two jobs. Keep this in mind during configuration. It is possible to map more Modbus areas (registers or bit values) in one data block than can be processed with one message frame.		
data_type	The data_type parameter specifies which MODBUS data types are mapped in this data block. If the value 0 is entered in data_type, the corresponding data area is not used.		
	Identifier	Data type	Data width
	0	Area is not used	
	1	Coils	Bit
	2	Inputs	Bit
	3	Holding Register	Word
	4	Input Register	Word

Parameter	Description
db	The db parameter specifies the data block which maps the MODBUS registers or bit values defined below. The DB number 0 is not permitted because it is reserved for the system. DB number 1 to 65535 (W#16#0001 to W#16#FFFF) The data block must be 2 bytes longer than necessary for the configured data. The last two bytes are used for internal purposes.
start end	start specifies the first Modbus address which is mapped in data word 0 of the DB. The end parameter defines the address of the last MODBUS address. The data word number in the S7 DB with the last Modbus address input is calculated as follows for register access: DBW number = (end – start) * 2 The data byte number in the S7 DB with the last Modbus address input is calculated as follows for bit access: DBB number = (end – start + 7) / 8 The defined data areas must not overlap. The end parameter must not be less than start. In case of an error, startup of the instruction in aborted with error. If both values are identical, one Modbus address (1 register or 1 bit value) is assigned. Example for mapping of the MODBUS addresses in S7 memory areas. MODBUS address 0 to 65535 (W#16#0000 to W#16#FFFF)
internal_send_buffer	This array is used in the instruction for the send data. Access or changes to this area are not permitted.
internal_recv_buffer	This array is used in the instruction for the received data. Access or changes to this area are not permitted.

1.4 Description of MODBUSPN

Description

This MODBUSPN instruction enables communication between a CPU with integrated PN interface and a partner which supports the Modbus/TCP protocol. The function codes 1, 2, 3, 4, 5, 6, 15 and 16 are supported. Depending on the parameter assignment, the instruction can be operated as client (S7 is client) as well as server (S7 is server). You use the MODBUSPN instruction to establish a connection between the communication partners, to transmit the data and to control termination of the connection.

The following actions are executed during data transmission:

- Generating the MODBUS-specific message frame header when sending
- Checking the MODBUS-specific message frame header when receiving
- Checking to see if the data areas addressed by the client exist
- Generating exception message frames when an error has occurred (only if S7 is server)
- Data transfer from/to configured data block

The time it takes to establish the connection and to terminate it as well as the data reception are monitored as well.

The MODBUSPN instruction V1.0 can be used for the S7-300 as well as for the S7-400. The connection takes place by means of the local CPU interface. To use the instruction, you do not require an additional hardware module.

Operating principle of the instruction

Startup

The MODBUSPN instruction is called once in OB100.

- The initialization parameters must be assigned according to the plant configuration.
- The initialization parameters are applied to the instance DB.
- The runtime parameters are not evaluated during startup.
- The data from the parameter data block are checked for validity.

Cyclic mode

In cyclic mode the MODBUSPN instruction is called in OB1 or in a cyclic interrupt OB.

- The block functions are activated based on the runtime parameters.
- Changes to the runtime parameters are not evaluated while a job is in progress.
- Initialization parameters records are not evaluated.

Restart during commissioning

A repeated CPU restart after changing the initialization parameters can be rather time-consuming during commissioning. The restart program part of the Modbus block can be run through by manually setting the "Init_Start" parameter in the static area. A job may not be in progress during manual initialization. All initialization parameters must be configured in the cyclic OB for correct initialization.

Handling the connection

The Modbus client actively establishes the connection. The necessary data are read from the connection parameters in the parameter data block.

A parameter in the connection parameter block (active_est) specifies if the PN CPU is to serve as active or passive communication partner.
A communication channel to the link partner is opened during runtime for both connection types, active and passive, with the TCON instruction.

The time of the connection establishment is specified with the connect_at_startup parameter in the parameter data block.

The connection is terminated with the DISCONNECT parameter at the MODBUSPN instruction.

Job initialization for "S7 is client" or activation of the instruction for "S7 is server"

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

S7 is client: Job initialization

A job is activated by a positive edge change at the trigger input ENQ_ENR. Depending on the input parameters UNIT, DATA_TYPE, START_ADDRESS, LENGTH, TI and WRITE_READ, a MODBUS request message frame is generated and sent to the partner station via the TCP/IP connection. The client waits for a response from the server for the configured time RECV_TIME.
If this time is exceeded (no response from the server), the active job is canceled with an error. A new job can be initiated.

A validity check is performed after the response message has been received. If this check is successful, the required actions are performed and the job is executed without errors; the output DONE_NDR is set. If errors were detected during the check, the job is canceled with errors, the ERROR bit is set and an error number is displayed in STATUS_MODBUS.

S7 is server: Activation of the instruction

The instruction is ready to receive a request message from the client with a positive level at the trigger input ENQ_ENR. The server is passive in this case and waits for a message frame from the client. The received message frame is checked. If the check is successful, the request message is answered. The user is informed about the completed message traffic when the DONE_NDR bit is set. The executed function is now displayed at the outputs UNIT, DATA_TYPE, START_ADDRESS, LENGTH, TI and WRITE_READ.

A faulty request message results in an error message. The ERROR bit is set, the error number is displayed in the STATUS_MODBUS and the request of the client is not processed. An exception message frame is sent to the client, depending on the error.

Parameter

The following table shows the parameters of the "MODBUSPN" instruction:

Parameter	Declaration	Data type	Description	Value range
ID (Page 23)	Input	WORD	Connection ID must be identical to the associated id parameter in the parameter DB	1 to 4095 W#16#1 to W#16#FFF
DB_PARAM (Page 23)	Input	BLOCK_DB	Number of the parameter DB	Depends on CPU
RECV_TIME (Page 23)	Input	TIME	Monitoring time for data reception by link partner The minimum time that can be set is 20 ms.	T#20ms to T#+24d20h31m23s647ms
CONN_TIME (Page 23)	Input	TIME	Monitoring time for establishing or terminating the connection The minimum time that can be set is 100 ms.	T#100ms to T#+24d20h31m23s647ms
KEEP_ALIVE	Input	TIME	not used	

Parameter	Declaration	Data type	Description	Value range
ENQ_ENR (Page 24)	Input	BOOL	S7 is client: Job initialization with positive edge • Changes to the input parameters will not become effective until the server has responded or an error message has been output. • If the ENQ_ENR parameter is set again during an ongoing Modbus request, no additional transmission takes place afterwards. S7 is server: Ready to receive with positive level	TRUE FALSE
DISCONNECT (Page 24)	Input	BOOL	With this parameter, you control the establishment and termination of the connection to the Modbus server: S7 is client: TRUE: The connection is established once the response message has been received. S7 is server: TRUE: The connection is terminated if ENQ_ENR = FALSE.	TRUE FALSE
REG_KEY (Page 16)	Input	STRING [17]	Registration key (registration key) for licensing	Character
LICENSED (Page 16)	Output	BOOL	License status of block Block is licensed Block is not licensed	 TRUE FALSE
BUSY	Output	BOOL	Processing status of the T-functions (TCON, TDISCON, TSEND or TRCV) a T-function is in progress a T-function is not in progress	 TRUE FALSE
CONN_ESTABLISHED	Output	BOOL	Connection to link partner established Connection to link partner terminated	TRUE FALSE
DONE_NDR	Output	BOOL	S7 is client: TRUE: Activated job completed without errors S7 is server: TRUE: Request from client was executed and reply was sent	TRUE FALSE
ERROR	Output	BOOL	FALSE: No error TRUE: Error occurred. The cause of error is indicated by the STATUS_MODBUS and STATUS_CONN parameters.	FALSE TRUE
STATUS_MODBUS (Page 27)	Output	WORD	Error number for protocol error during processing of the Modbus message frames	0 to FFFF
STATUS_CONN (Page 27)	Output	WORD	Error number for connection errors during processing of the T-functions (TCON, TSEND, TRCV, TDISON)	0 to FFFF
STATUS_FUNC (Page 27)	Output	STRING [8]	Name of the instruction which has caused the error at STATUS_MODBUS or STATUS_CONN	Character
IDENT_CODE (Page 16)	Output	STRING [18]	Identification number for licensing You can use this code to request the registration key REG_KEY for your license.	Character
UNIT (Page 25)	InOut	BYTE	Unit Identifier (INPUT for client function, OUTPUT for server function)	0 to 255 B#16#0 to B#16#FF

Parameter	**Declaration**	**Data type**	**Description**	**Value range**
DATA_TYPE (Page 25)	InOut	BYTE	Data type to be processed: (INPUT for client function, OUTPUT for server function) Coils Inputs Holding Register Input Register	1 2 3 4
START_ADDRESS (Page 25)	InOut	WORD	MODBUS start address (INPUT for client function, OUTPUT for server function)	0 to 65535 W#16#0000 to W#16#FFFF
LENGTH (Page 25)	InOut	WORD	Number of values to be processed (INPUT for client function, OUTPUT for server function) Coils Reading function Writing function Inputs Reading function Holding Register Reading function Writing function Input Register Reading function	1 to 2000 1 to 1968 1 to 2000 1 to 125 1 to 123 1 to 125
TI (Page 25)	InOut	WORD	Transaction Identifier (INPUT for client function, OUTPUT for server function)	0 to 65535 W#16#0 to W#16#FFFF
WRITE_READ (Page 25)	InOut	BOOL	Write access or read access (INPUT for client function, OUTPUT for server function)	TRUE FALSE

See also

Commissioning (Page 5)

1.5 Licensing with the IDENT_CODE and REG_KEY parameters

Description

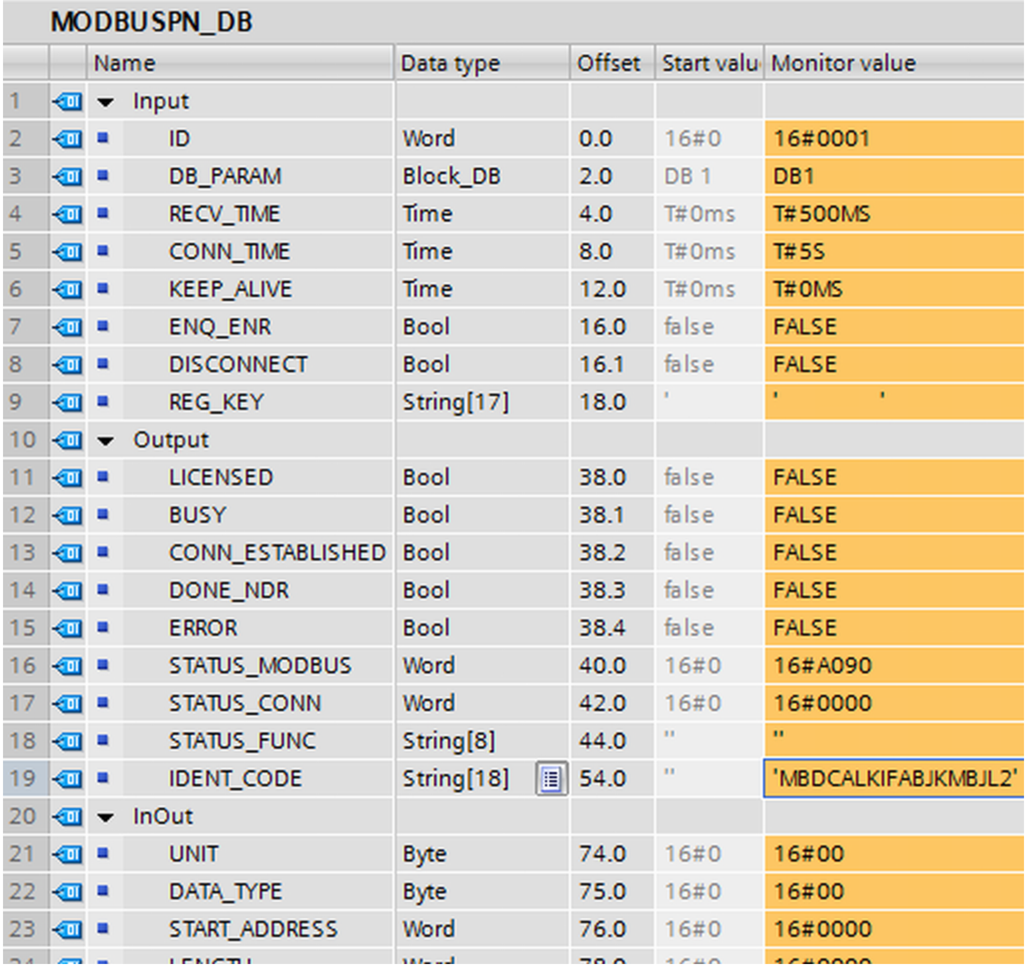
The MODBUSPN instruction must be licensed on each CPU individually. Licensing takes place in two steps:

- Reading of the IDENT_CODE and
- input of the registration key REG_KEY.

The OB121 must be present in the CPU.

Proceed as follows to read the IDENT_CODE:

1. Assign the parameters of the MODBUSPN instruction according to your requirements in a cyclic OB and in OB100. Download the program to the CPU and set it to RUN.
2. Open the instance DB of the Modbus instruction and click the "Monitor all" button.
3. An 18-digit character string is displayed at the IDENT_CODE output.

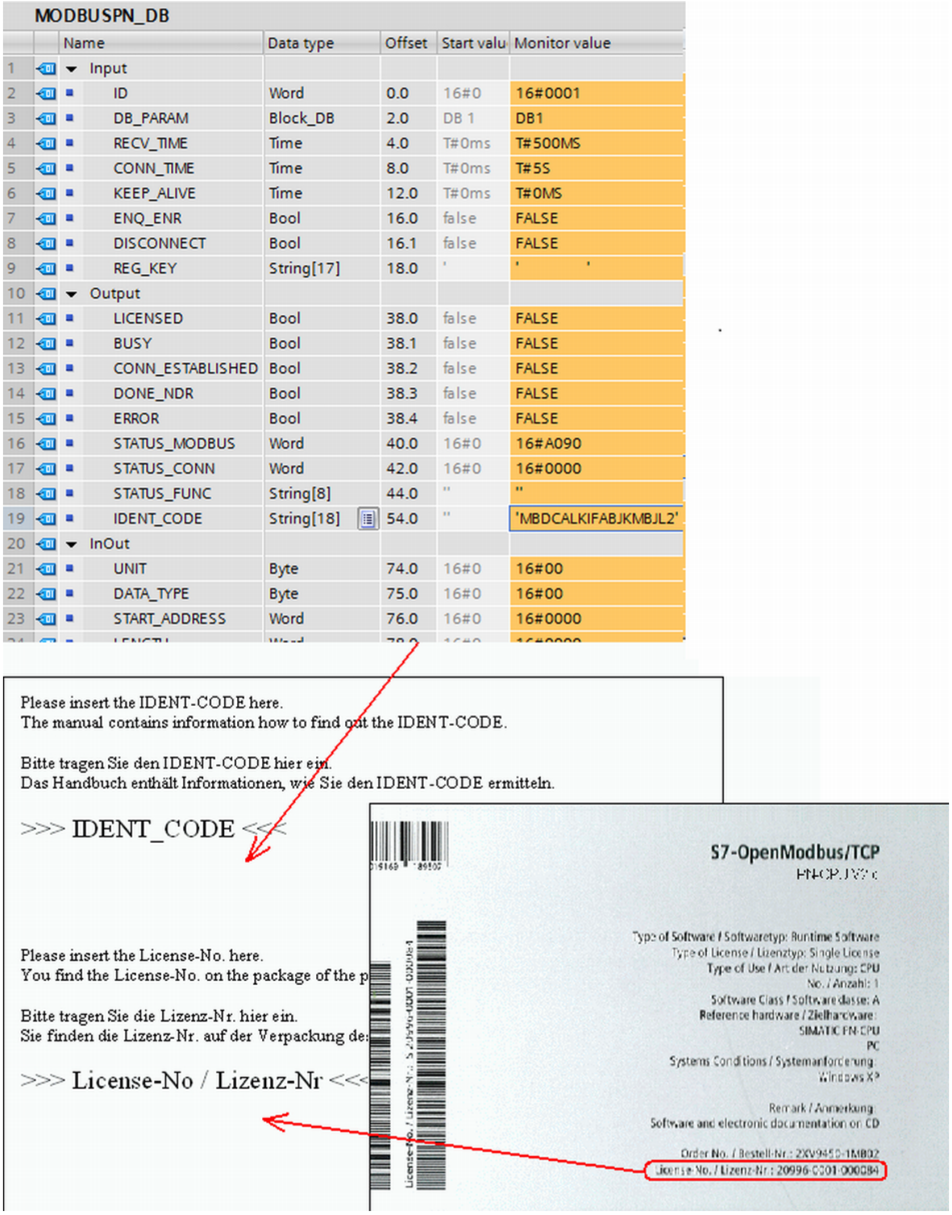


MODBUSPN_DB

	Name	Data type	Offset	Start valu	Monitor value
1	▼ Input				
2	ID	Word	0.0	16#0	16#0001
3	DB_PARAM	Block_DB	2.0	DB 1	DB1
4	RECV_TIME	Time	4.0	T#0ms	T#500MS
5	CONN_TIME	Time	8.0	T#0ms	T#5S
6	KEEP_ALIVE	Time	12.0	T#0ms	T#0MS
7	ENQ_ENR	Bool	16.0	false	FALSE
8	DISCONNECT	Bool	16.1	false	FALSE
9	REG_KEY	String[17]	18.0	'	' '
10	▼ Output				
11	LICENSED	Bool	38.0	false	FALSE
12	BUSY	Bool	38.1	false	FALSE
13	CONN_ESTABLISHED	Bool	38.2	false	FALSE
14	DONE_NDR	Bool	38.3	false	FALSE
15	ERROR	Bool	38.4	false	FALSE
16	STATUS_MODBUS	Word	40.0	16#0	16#A090
17	STATUS_CONN	Word	42.0	16#0	16#0000
18	STATUS_FUNC	String[8]	44.0	''	''
19	IDENT_CODE	String[18]	54.0	''	'MBDCALKIFABJKMBJL2'
20	▼ InOut				
21	UNIT	Byte	74.0	16#0	16#00
22	DATA_TYPE	Byte	75.0	16#0	16#00
23	START_ADDRESS	Word	76.0	16#0	16#0000
24	LENGTH	Word	78.0	16#0	16#0000

Figure 1-2 IDENT_CODE in the DB

4. Copy this string using copy/paste from the data block and enter it in the form SOFTWARE REGISTRATION FORM . This form is included on the installation CD.
   Enter the license number from the product packaging in the form.



MODBUSPN_DB

	Name	Data type	Offset	Start valu	Monitor value
1	▼ Input				
2	ID	Word	0.0	16#0	16#0001
3	DB_PARAM	Block_DB	2.0	DB 1	DB1
4	RECV_TIME	Time	4.0	T#0ms	T#500MS
5	CONN_TIME	Time	8.0	T#0ms	T#5S
6	KEEP_ALIVE	Time	12.0	T#0ms	T#0MS
7	ENQ_ENR	Bool	16.0	false	FALSE
8	DISCONNECT	Bool	16.1	false	FALSE
9	REG_KEY	String[17]	18.0	'	' '
10	▼ Output				
11	LICENSED	Bool	38.0	false	FALSE
12	BUSY	Bool	38.1	false	FALSE
13	CONN_ESTABLISHED	Bool	38.2	false	FALSE
14	DONE_NDR	Bool	38.3	false	FALSE
15	ERROR	Bool	38.4	false	FALSE
16	STATUS_MODBUS	Word	40.0	16#0	16#A090
17	STATUS_CONN	Word	42.0	16#0	16#0000
18	STATUS_FUNC	String[8]	44.0	''	''
19	IDENT_CODE	String[18]	54.0	''	'MBDCALKIFABJKMBJL2'
20	▼ InOut				
21	UNIT	Byte	74.0	16#0	16#00
22	DATA_TYPE	Byte	75.0	16#0	16#00
23	START_ADDRESS	Word	76.0	16#0	16#0000

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

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

>>> IDENT_CODE <<<

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

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

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

S7-OpenModbus/TCP
PN-CPU V? c

Type of Software / Softwaretyp: Runtime Software
Type of License / Lizenztyp: Single License
Type of Use / Art der Nutzung: CPU
No. / Anzahl: 1
Software Class / Softwareklasse: A
Reference hardware / Zielhardware:
SIMATIC PN-CPU
PC
Systems Conditions / Systemanforderung:
Windows XP

Remark / Anmerkung:
Software and electronic documentation on CD

Order No. / Bestell-Nr.: 2XV9450-1MB02
License-No. / Lizenz-Nr.: 20996-0301-000084

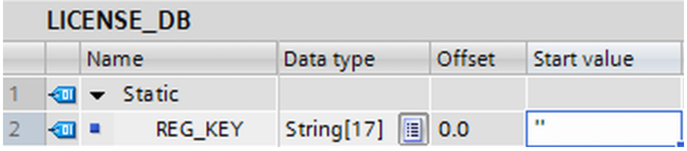
Figure 1-3 IDENT_CODE and license form

5. Send the form to Customer Support (https://support.automation.siemens.com/WW/view/en/38718979) using a service request . You will then receive the activation code for your CPU.

The registration key REG_KEY must be specified at each MODBUSPN instruction. You should save the REG_KEY in a global data block by which all MODBUSPN instructions receive the necessary activation code.

Proceed as follows to enter the registration key REG_KEY:

- Insert a new global data block with "Add new block…" with a unique symbolic name, for example, "License_DB".
- Create a REG_KEY parameter in this block with the data type STRING[17].



LICENSE_DB

	Name	Data type	Offset	Start value
1	▼ Static			
2	■ REG_KEY	String[17]	0.0	''

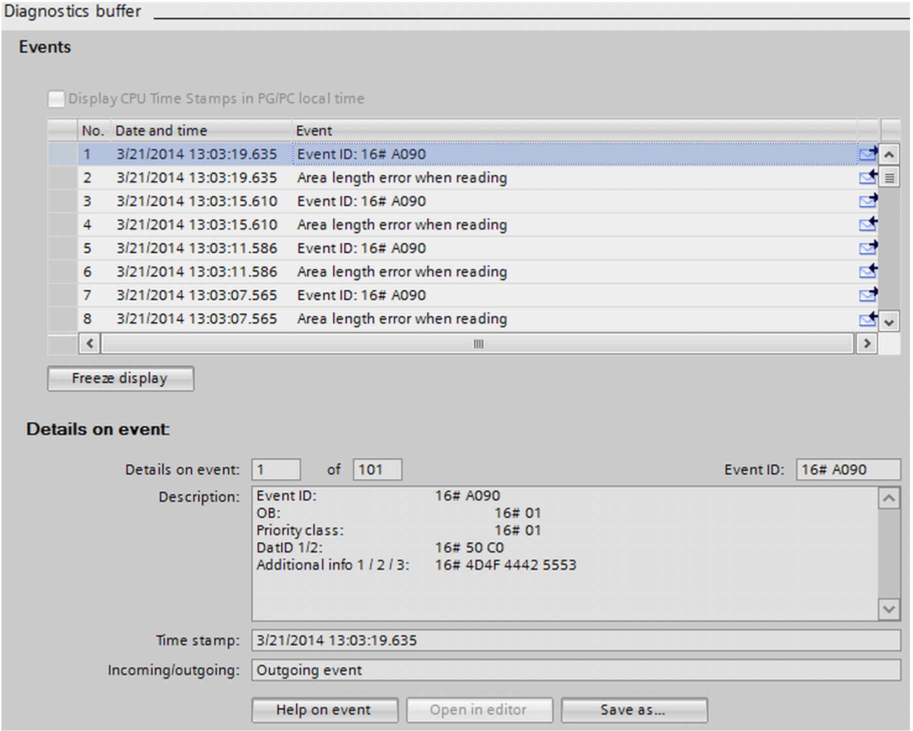
Figure 1-4 REG_KEY in DB

- Copy the transmitted 17-digit registration key using copy/paste to the "Start value" column.
- Enter the value "License_DB.REG_KEY" in the cyclic OB at the REG_KEY parameter of the MODBUSPN instruction.
- Download the modified blocks to the CPU. The registration key can be entered during runtime; a change from STOP -> RUN is not necessary.

The Modbus/TCP communication using the MODBUSPN instruction is now licensed for this CPU, the output bit LICENSED is TRUE.

Missing or incorrect licensing

If you enter an incorrect registration key or no activation code at all, the SF-LED (for S7-300 and IM151-8) or INTF-LED (for S7-400) of the CPU flashes and a cyclic entry is made in the diagnostics buffer regarding the missing license. The error number for a missing license is W#16#A090.



Diagnostics buffer

Events

□ Display CPU Time Stamps in PG/PC local time

No.	Date and time	Event
1	3/21/2014 13:03:19.635	Event ID: 16# A090
2	3/21/2014 13:03:19.635	Area length error when reading
3	3/21/2014 13:03:15.610	Event ID: 16# A090
4	3/21/2014 13:03:15.610	Area length error when reading
5	3/21/2014 13:03:11.586	Event ID: 16# A090
6	3/21/2014 13:03:11.586	Area length error when reading
7	3/21/2014 13:03:07.565	Event ID: 16# A090
8	3/21/2014 13:03:07.565	Area length error when reading

Details on event:

Details on event: 1 of 101   Event ID: 16# A090

Description:

```
Event ID:                16# A090
OB:                           16# 01
Priority class:               16# 01
DatID 1/2:               16# 50 C0
Additional info 1 / 2 / 3:   16# 4D4F 4442 5553
```

Time stamp: 3/21/2014 13:03:19.635

Incoming/outgoing: Outgoing event

Figure 1-5 Diag buffer with A090

⚠ **WARNING**

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

In case of a missing or incorrect registration key, the Modbus/TCP communication is processed but W#16#A090 "No valid license available" is always displayed at the STATUS_MODBUS output. The output bit LICENSED is FALSE.

See also

Description of MODBUSPN (Page 12)

1.6 Address mapping

Interpretation of the Modbus addresses

The MODBUS data model includes the following areas:

- Coils
- Inputs
- Holding Register
- Input Register

These memory areas are distinguished in some systems, for example, MODICON PLCs, by means of the register address or bit address. The Holding Register with offset 0, for example, is referred to as register 40001 (memory type 4xxxx, Reference 0001).

This often leads to some confusion because some manuals describe and refer to the register address of the Application Layers while other manuals use the register address/bit address that is actually transferred in the protocol.

The MODBUSPN instruction uses in its start, end and START_ADDRESS parameters the **actually transferred Modbus address**. This means you can transfer register addresses/bit addresses from 0000H to FFFFH with each function code.

Example

The Modbus addresses can be specified in decimal or hexadecimal format in the parameter DB.

Parameter	Decimal notation	Hexadecimal notation	Meaning
data_type	3	B#16#3	Holding Register
db	11	W#16#B	DB 11
start	0	W#16#0	Start address: 0

end	499	W#16#1F3	End address: 499
data_type	3	B#16#3	Holding Register
db	12	W#16#C	DB 12
start	720	W#16#2D0	Start address: 720
end	900	W#16#384	End address: 900

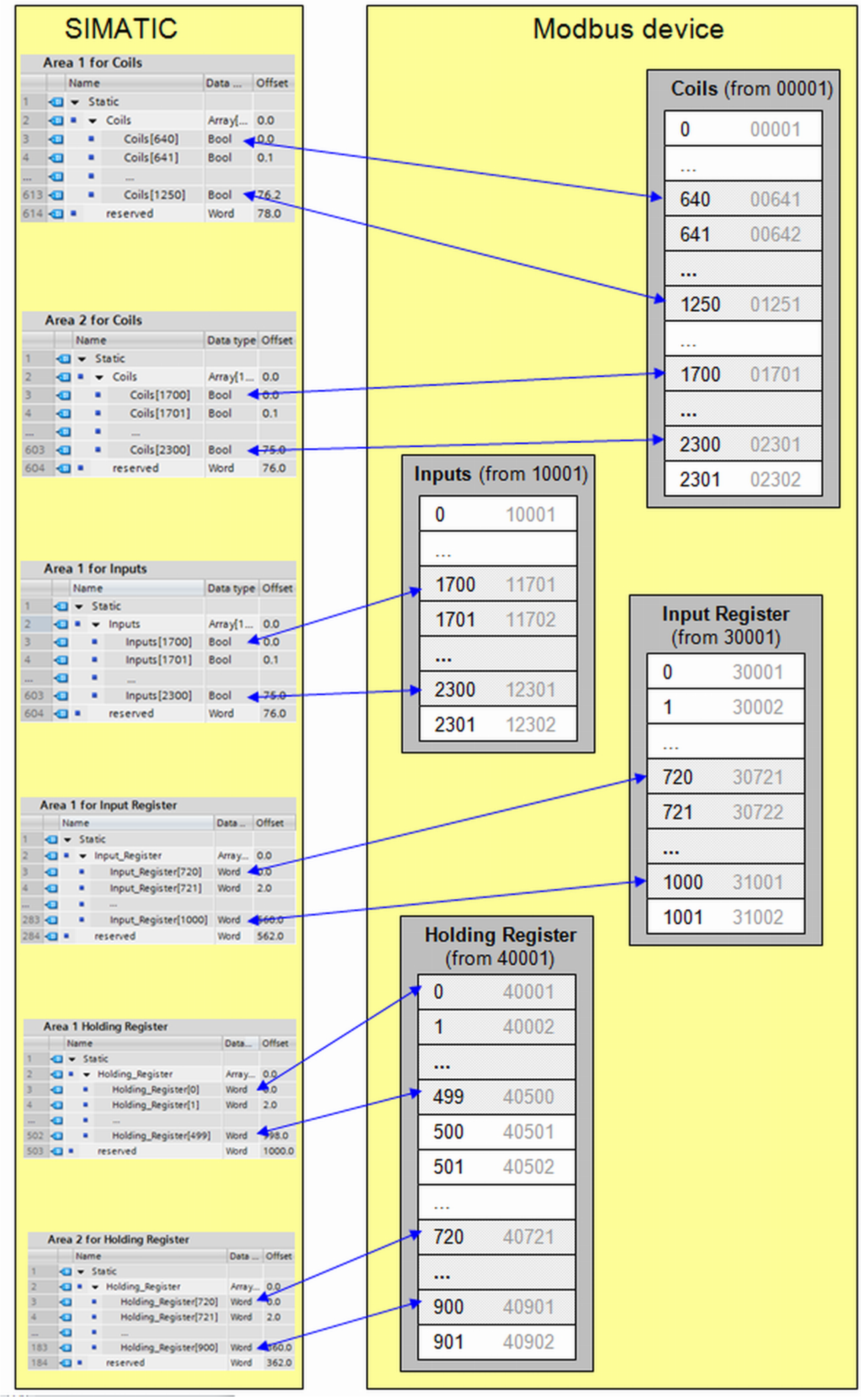
The figure below shows a comparison of the SIMATIC memory areas with the register-oriented and bit-oriented memory allocation of the Modbus devices. The allocation is based on the parameter assignment described above.

In the Modbus device:

The Modbus addresses shown in black refer to the Data Link Layer; the ones shown in gray refer to the Applikation Layer.

In SIMATIC:

The SIMATIC addresses in the first column are the offset in the DB. The Modbus register numbers are shown in brackets.



SIMATIC

Area 1 for Coils

	Name	Data ...	Offset
1	Static		
2	Coils	Array[...	0.0
3	Coils[640]	Bool	0.0
4	Coils[641]	Bool	0.1
...	...		
613	Coils[1250]	Bool	76.2
614	reserved	Word	78.0

Area 2 for Coils

	Name	Data type	Offset
1	Static		
2	Coils	Array[1...	0.0
3	Coils[1700]	Bool	0.0
4	Coils[1701]	Bool	0.1
...	...		
603	Coils[2300]	Bool	75.0
604	reserved	Word	76.0

Area 1 for Inputs

	Name	Data type	Offset
1	Static		
2	Inputs	Array[1...	0.0
3	Inputs[1700]	Bool	0.0
4	Inputs[1701]	Bool	0.1
...	...		
603	Inputs[2300]	Bool	75.0
604	reserved	Word	76.0

Area 1 for Input Register

	Name	Data ...	Offset
1	Static		
2	Input_Register	Array...	0.0
3	Input_Register[720]	Word	0.0
4	Input_Register[721]	Word	2.0
...	...		
283	Input_Register[1000]	Word	560.0
284	reserved	Word	562.0

Area 1 Holding Register

	Name	Data...	Offset
1	Static		
2	Holding_Register	Array...	0.0
3	Holding_Register[0]	Word	0.0
4	Holding_Register[1]	Word	2.0
...	...		
502	Holding_Register[499]	Word	998.0
503	reserved	Word	1000.0

Area 2 for Holding Register

	Name	Data ...	Offset
1	Static		
2	Holding_Register	Array...	0.0
3	Holding_Register[720]	Word	0.0
4	Holding_Register[721]	Word	2.0
...	...		
183	Holding_Register[900]	Word	360.0
184	reserved	Word	362.0

Modbus device

Coils (from 00001)

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

Inputs (from 10001)

0	10001
...	
1700	11701
1701	11702
...	
2300	12301
2301	12302

Input Register (from 30001)

0	30001
1	30002
...	
720	30721
721	30722
...	
1000	31001
1001	31002

Holding Register (from 40001)

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

Figure 1-6 Address mapping

See also

Description of MODBUSPN (Page 12)

1.7 ID and DB_PARAM parameters

Description

ID

A connection ID is required for each connection from the PN CPU to a communication partner. A different connection ID is to be used for each logical connection in case of multiple communication partners.
This connection ID is configured in the connection parameter block included in the parameter data block. The connection ID uniquely describes the connection from the CPU to the link partner and can have the values 1 to 4095.
The connection ID from the connection parameter block must be entered here; it must be unique throughout the entire CPU.

DB_PARAM

The DB_PARAM parameter refers to the number of the parameter data block. The connection-specific and Modbus-specific parameters which are required for communication between the PN CPU and the link partner are stored in this parameter data block.

The CPU determines the value range for this parameter. The DB number 0 is not permitted because it is reserved for the system.
The DB number is entered in plain text as "DBxy".

If you want to implement several connections, the parameter data block can include the necessary parameters of all connections in sequence. You can also create a separate parameter data block for each connection.

See also

Description of MODBUSPN (Page 12)

Parameter data block (Page 7)

1.8 RECV_TIME and CONN_TIME parameters

Description

RECV_TIME

The monitoring time RECV_TIME monitors the data received from the link partner. An error is signaled and the connection terminated when the monitoring time is exceeded.

The minimum value is 20 ms.

If the RECV_TIME is set to < 20 ms in "**S7 is client**" operating mode, a corresponding error message is displayed and the active job is rejected.
If the RECV_TIME is set to < 20 ms in "**S7 is server**" operating mode, the default value 1.2 s is used. The RECV_TIME monitors the runtime of the TCP stream. The break in between individual client requests is not taken into consideration.

CONN_TIME

The CONN_TIME specifies the time for monitoring establishment and termination of the connection. If the connection could not be established or terminated within the configured monitoring time, a corresponding error message is displayed at the STATUS_CONN output.

The minimum value is 100 ms.

In "**S7 is client**" operating mode, a CONN_TIME that is too short is set to the default value of 5 s for connect_at_startup = TRUE. An error message is output and the activated job rejected in cyclic operation if the CONN_TIME is too short.

The default value of 5 s is also used if the CONN_TIME was set to < 100 ms in "**S7 is server**" operating mode.

See also

Description of MODBUSPN (Page 12)

1.9 ENQ_ENR and DISCONNECT parameters

Description

"S7 is client" operating mode

The data transfer is initiated with a positive edge at ENQ_ENR. The request message is generated with the values of the input parameters UNIT, DATA_TYPE, START_ADDRESS, LENGTH, TI and WRITE_READ. A new message frame can only be sent if the previous message frame was completed with DONE_NDR or ERROR.
If the connection is not established (CONN_ESTABLISHED = FALSE), the connection is established first with subsequent data transfer.

If the DISCONNECT = TRUE parameter is set, the connection is terminated after the data transfer.

"S7 is server" operating mode

The instruction is activated with a positive level at the ENQ_ENR input. Client requests are evaluated and a reply is sent. If the connection is not established with set ENQ_ENR (CONN_ESTABLISHED = FALSE), the connection establishment is activated.

If ENQ_ENR changes from TRUE to FALSE during operation, the connection is terminated when DISCONNECT = TRUE.

In case of an existing connection and ENQ_ENR = FALSE , received requests are discarded.

See also

Description of MODBUSPN (Page 12)

1.10 DATA_TYPE, START_ADDRESS, LENGTH, TI, WRITE_READ and UNIT parameters

Description

In "**S7 is client**"operating mode, these parameters are input parameters; in "**S7 is server**", these parameters are output parameters.

DATA_TYPE

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

Modbus data type	DATA_TYPE
Coils	B#16#1
Inputs	B#16#2
Holding Register	B#16#3
Input Register	B#16#4

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

Modbus data type	DATA_TYPE	Function	Length	single_write	Function code
Coils	1	reading	any	irrelevant	1
Coils	1	writing	1	TRUE	5
Coils	1	writing	1	FALSE	15
Coils	1	writing	> 1	irrelevant	15
Inputs	2	reading	any	irrelevant	2
Holding Register	3	reading	any	irrelevant	3
Holding Register	3	writing	1	TRUE	6
Holding Register	3	writing	1	FALSE	16
Holding Register	3	writing	> 1	irrelevant	16
Input Register	4	reading	any	irrelevant	4

START_ADDRESS

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

LENGTH

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

A maximum of 125 registers is possible for reading functions per message frame for Holding and Input Register. A maximum of 2000 bits is possible for Coils and Inputs.

For writing functions the maximum number is 123 registers for Holding Register and 1968 bits for Coils.

The registers or bit values processed with a request message must be located within one data block.

TI

The TI, Transaction Identifier, parameter is copied by the server from the request message to the response message according to MODBUS specification.

In "**S7 is client**" operating mode, it is an input parameter. The instruction applies this value to the request message and checks the value when it receives the response.

In "**S7 is server**" operating mode, it is an output parameter. The instruction applies the values from the request message to the response.

The Transaction Identifier is used for message frame identification or unique assignment of requests to the response. The MODBUSPN instruction can only make this assignment, if the TI is changed for each message frame. Only then will the instruction work properly.

We therefore recommend that you increase the TI by 1 for each request.

WRITE_READ

This parameter defines if a reading or writing function is to be performed. If the input/output has the value FALSE, it is a reading function. The value TRUE defines a writing function.

Only Holding Register and Coils can be written. Input Register and Inputs can only be read.

UNIT

The UNIT, Unit Identifier, parameter refers to the unique assignment of the link partner. It is mainly necessary if there are several serial devices downstream of a converter which are addressed with different UNIT numbers.

In the "**S7 is client**" function, the UNIT parameter is an input parameter. This input has to be set according to the requirements. The instruction applies this value to the request message and checks the value when it receives the response.

In the "**S7 is server**" function, the UNIT parameter is an output parameter. The instruction applies the value from the request message to the response message and displays it when the job is complete.

DONE_NDR

In "**S7 is client**" operating mode, the active job was completed without errors. In case of a reading function, the response data from the server have already been entered in the DB; for a writing function, the response to the request message was received by the server.

In "**S7 is server**" operating mode, the output displays a message traffic completed without errors with the client. The job parameters of the client are displayed in the UNIT, DATA_TYPE, START_ADDRESS, LENGTH, TI and WRITE_READ parameters. These output are only valid as long as DONE_NDR is set.

See also

Description of MODBUSPN (Page 12)

Address mapping (Page 20)

1.11 ERROR, STATUS_MODBUS, STATUS_CONN and STATUS_FUNC parameters

Error evaluation

The MODBUSPN instruction has three status outputs for error diagnostics: STATUS_MODBUS, STATUS_CONN and STATUS_FUNC.

ERROR

An error is detected when this output is set.

In "**S7 is client**" operating mode, the active job was completed with errors. The associated error number is displayed in the STATUS_MODBUS or STATUS_CONN output.

In "**S7 is server**" operating mode, an error was detected in the request message of the client or when sending the response message. The associated error number is displayed in the STATUS_MODBUS or STATUS_CONN output.

STATUS_MODBUS

STATUS_MODBUS displays the error messages and status information regarding processing of Modbus-specific message frames. The MODBUSPN instruction uses different system blocks. The error messages of these blocks are forwarded without changes to STATUS_MODBUS.

STATUS_CONN

STATUS_CONN displays the error messages regarding processing of the connection. STATUS_CONN also displays messages such as "Job in progress". In this case the ERROR bit is not set. In addition, the internal instructions TCON, TSEND, TRCV and TDISCON are called. The error messages of these blocks are forwarded without changes to STATUS_CONN.

STATUS_FUNC

STATUS_FUNC displays the name of the function which has caused the error.

Below is a listing of the error messages for specific instructions.

STATUS_MODBUS parameter with STATUS_FUNC = 'MODBUSPN'

STATUS* (W#16#)	Description	Solution
A001	The parameter data block is too short.	Correct the length of the parameter DB.
A002	The end parameter is smaller than start.	Correct the information in the parameter DB.
A003	A DB on which MODBUS addresses are to be mapped is too short. Minimum length: • For registers: (end - start + 1) * 2 + 2 • For bit values: (end - start) / 8 + 1 + 2 Additional possible causes: • S7 is client: Incorrect call parameters • S7 is server: Incorrect address range in the client request message. The S7 responds with an exception message frame.	Lengthen the DB. S7 is client: Correct the START_ADDRESS or LENGTH call parameters. S7 is server: Change the client request.
A004	Only S7 is client: An invalid combination of DATA_TYPE and WRITE_READ was specified.	Correct the call parameters. Only data types 1 and 3 can be written.
A005	S7 is client: An invalid value was specified at the LENGTH parameter. S7 is server: The number of registers/bits is invalid in the request message. The S7 responds with an exception message frame. Value ranges: Coils/Inputs reading: 1 to 2000 Coils writing: 1 to 1968 Register reading: 1 to 125 Holding Register writing: 1 to 123	S7 is client: Correct the LENGTH parameter. S7 is server: Change the number in the client request message.
A006	The area specified with DATA_TYPE, START_ADDRESS and LENGTH does not exist in data_type from data_area_1 to data_area_8. S7 is server: The S7 responds with an exception message frame.	S7 is client: Correct the combination DATA_TYPE, START_ADDRESS and LENGTH. S7 is server: Change the client request or correct the parameter assignment in the parameter DB.
A007	S7 is client: An invalid monitoring time was configured at RECV_TIME or CONN_TIME. A value >= 20 ms must be entered for RECV_TIME and a value >= 100 ms for CONN_TIME.	Correct the parameter assignment.
A009	S7 is client: The received Transaction Identifier TI does not match the one sent. The communication connection is terminated.	Use a message log to check the data of the link partner.
A00A	S7 is client: The received Unit Identifier UNIT does not match the one sent.	Use a message log to check the data of the link partner.
A00B	S7 is client: The received function code does not match the one sent. S7 is server: An invalid function code was received. The S7 responds with an exception message frame.	S7 is client: Use a message log to check the data of the link partner. S7 is server: Change the client request. The MODBUSPN instruction processes the function codes 1, 2, 3, 4, 5, 6, 15 and 16.
A00C	The received bytecount does not match the number of registers. The communication connection is terminated.	Use a message log to check the data of the link partner.

STATUS* (W#16#)	Description	Solution
A00D	Only for S7 is client: The register address/bit address or the number of registers/bits in the response message is not the same as in the request message.	
A00E	The length information in the Modbus-specific message frame header does not match the information regarding number of registers/bits or of the bytecount in the message. The instructions discards all data. The communication connection is terminated.	
A00F	A Protocol Identifier unequal 0 has been received. The communication connection is terminated.	Use a message log to check the data of the link partner.
A010	A DB number was assigned twice in the db parameter from data_area_1 to data_area_8.	Correct the parameter assignment in the parameter DB.
A011	An invalid value was specified at the DATA_TYPE input parameter (valid values are 1, 2, 3 and 4).	Correct the call parameters.
A012	The configured areas data_area_1 and data_area_2 overlap.	Correct the parameter assignment. The data areas may not have a shared register address area.
A013	The configured areas data_area_1 and data_area_3 overlap.	
A014	The configured areas data_area_1 and data_area_4 overlap.	
A015	The configured areas data_area_1 and data_area_5 overlap.	
A016	The configured areas data_area_1 and data_area_6 overlap.	
A017	The configured areas data_area_1 and data_area_7 overlap.	
A018	The configured areas data_area_1 and data_area_8 overlap.	
A019	One of the db parameters has been set to 0 even though the associated data_type is configured with > 0.	Correct the parameter assignment in the db parameter to > 0.
A01A	Incorrect length in the header: 1 to 253 bytes are permitted. The communication connection is terminated.	Use a message log to check the data of the link partner.
A01B	S7 is server and function code 5: An invalid status was received for coil. The S7 responds with an exception message frame.	Use a message log to check the data of the link partner.

STATUS* (W#16#)	Description	Solution
A023	The configured areas data_area_2 and data_area_3 overlap.	Correct the parameter assignment in the parameter DB. The data areas may not have a shared register address area.
A024	The configured areas data_area_2 and data_area_4 overlap.	
A025	The configured areas data_area_2 and data_area_5 overlap.	
A026	The configured areas data_area_2 and data_area_6 overlap.	
A027	The configured areas data_area_2 and data_area_7 overlap.	
A028	The configured areas data_area_2 and data_area_8 overlap.	
A034	The configured areas data_area_3 and data_area_4 overlap.	
A035	The configured areas data_area_3 and data_area_5 overlap.	
A036	The configured areas data_area_3 and data_area_6 overlap.	
A037	The configured areas data_area_3 and data_area_7 overlap.	
A038	The configured areas data_area_3 and data_area_8 overlap.	
A045	The configured areas data_area_4 and data_area_5 overlap.	
A046	The configured areas data_area_4 and data_area_6 overlap.	
A047	The configured areas data_area_4 and data_area_7 overlap.	
A048	The configured areas data_area_4 and data_area_8 overlap.	
A056	The configured areas data_area_5 and data_area_6 overlap.	
A057	The configured areas data_area_5 and data_area_7 overlap.	
A058	The configured areas data_area_5 and data_area_8 overlap.	
A067	The configured areas data_area_6 and data_area_7 overlap.	
A068	The configured areas data_area_6 and data_area_8 overlap.	
A078	The configured areas data_area_7 and data_area_8 overlap.	
A079	The connection ID specified at the ID parameter is not included in the parameter DB.	Correct the parameter assignment at the id input.
A07A	An invalid value was specified at the ID parameter (value range from 1 to 4095).	
A07B	The specified ID is included twice in the parameter DB.	Correct the parameter assignment in the parameter DB.
A07C	An invalid value was specified at the data_type parameter in the parameter DB (value range from 0 to 4).	
A07D	The data_type parameter of the data_area_1 has no entry in the parameter DB. The parameter area "„_1" is the initial area and must be configured.	
A07E	The number of the parameter DB or the number of the instance DB of the MODBUSPN instruction was specified at db.	
A07F	The DB specified at DB_PARAM is not a Modbus parameter DB. The length information in DBW0 was changed or an incorrect DB was specified.	Correct the parameter assignment at the DB_PARAM input.
A080	This error message occurs if different instance DBs are used for the call of the MODBUSPN instruction in OB1 or the cyclic interrupt OB and OB100.	Use the same IDB in OB100 and in the cyclic OB.
A081	Only for S7 is client and function code 5: The data of the response message are not the echo of the request.	Use a message log to check the data of the link partner.
A082	Only for S7 is client and function code 6: The received register value is not the same as the one sent.	Use a message log to check the data of the link partner.

STATUS* (W#16#)	Description	Solution
A083	S7 is client: A job has been triggered while the previous job is still in progress. The job is not executed. This is a status information. The ERROR bit is not set.	Do not trigger a new job until the previous job has been completed with DONE _NDR = TRUE or ERROR = TRUE.
A084	Unable to determine an identification code IDENT_CODE for licensing.	Please contact Product Support.
A085	An error occurred while trying to find the license.	Check for unauthorized write access to the license DB. Please contact Product Support, if necessary.
A086	An attempt was made to write in a write-protected data block.	Remove the write protection or use a different DB.
A090	The block has not been licensed for this CPU yet. This is a status information. The ERROR bit is not set. Modbus communication is possible even without license.	Read the IDENT_CODE for this CPU and use it to request the registration key. See "Licensing (Page 16)".
A091	An exception message frame with exception code 1 was received as response (only for S7 is client).	The link partner does not support the requested function.
A092	An exception message frame with exception code 2 was received as response (only for S7 is client). You tried to access an address that does not exist or is not permitted at the link partner.	Correct LENGTH or START_ADDRESS when calling the instruction.
A093	An exception message frame with exception code 3 was received as response (only for S7 is client).	The link partner cannot process the received message (for example, it does not support the requested length).
A094	An exception message frame with exception code 4 was received as response (only for S7 is client).	The link partner is in a state in which it cannot process the received message frame.
A095	An exception message frame with an unknown exception code was received as response (only for S7 is client).	Check the error messages of the link partner and check the data with a message log, if necessary.

STATUS_MODBUS parameter with STATUS_FUNC = 'RD_SINFO', 'BLKMOV', 'TEST_DB', 'RDSYSST' or 'WR_USMSG'

STATUS* (W#16#)	Description	Solution
7xxx	Consult the online help for detailed information.	See online help (TIA Portal -> Select block -> F1 key)
80B1	STATUS_FUNC = 'TEST_DB': The DB does not exist on the CPU.	All data blocks specified at db must be created and transferred to the CPU.
80B2	STATUS_FUNC = 'TEST_DB': DB UNLINKED	Do not generate DB as UNLINKED.
8xxx	Consult the online help for detailed information.	See online help (TIA Portal -> Select block -> F1 key)

STATUS_CONN parameter with STATUS_FUNC = 'MODBUSPN'

STATUS* (W#16#)	Description	Solution
A100	The monitoring time CONN_TIME or RECV_TIME has expired for a job. The connection is terminated when RECV_TIME has expired.	Check the parameter assignment of the connection.
A101	The internal monitoring time of the TDISCON function has expired.	Please contact Product Support.

STATUS_CONN parameter with STATUS_FUNC = 'TCON', 'TSEND', 'TRCV' or 'TDISCON'

STATUS* (W#16#)	Description	Solution
7xxx	See online help.	See online help (TIA Portal -> Select block -> F1 key)
8xxx	See online help.	See online help (TIA Portal -> Select block -> F1 key)

See also

Description of MODBUSPN (Page 12)

Index