OMRON

CP-series Function Block Practices Guide

Modbus TCP Server

■ Introduction

This guide describes examples of using function blocks.
Omron does NOT warrant that the function blocks work properly at all times in actual programs and machines.
Please obtain the user's manuals of the used devices and be sure to understand the important precautions and reminders described on the manuals before attempting to start operation.

■ Intended Audience

This guide is intended for the following personnel, who must also have knowledge of electrical systems (an electrical engineer or the equivalent).

- Personnel in charge of installing FA systems
- Personnel in charge of designing FA systems
- Personnel in charge of managing FA systems and facilities

■ Related Manuals

Cat. No.	Model	Manual name
W613 (CP2E)	CP2E-E□□D□-□ CP2E-S□□D□-□ CP2E-N□□D□-□	CP Series CP2E CPU Unit Hardware User's Manual
W614 (CP2E)	CP2E-E□□D□-□ CP2E-S□□D□-□ CP2E-N□□D□-□	CP Series CP2E CPU Unit Software User's Manual
W483 (CP1E/CP2E)	CP1E-E□□D□-□ CP1E-N□□D□-□ CP2E-E□□D□-□ CP2E-S□□D□-□ CP2E-N□□D□-□	CP Series CP1E/CP2E CPU Unit Instructions Reference Manual
W446	CXONE-AL□□D-V4	CX-Programmer Ver.9.□ Operation Manual

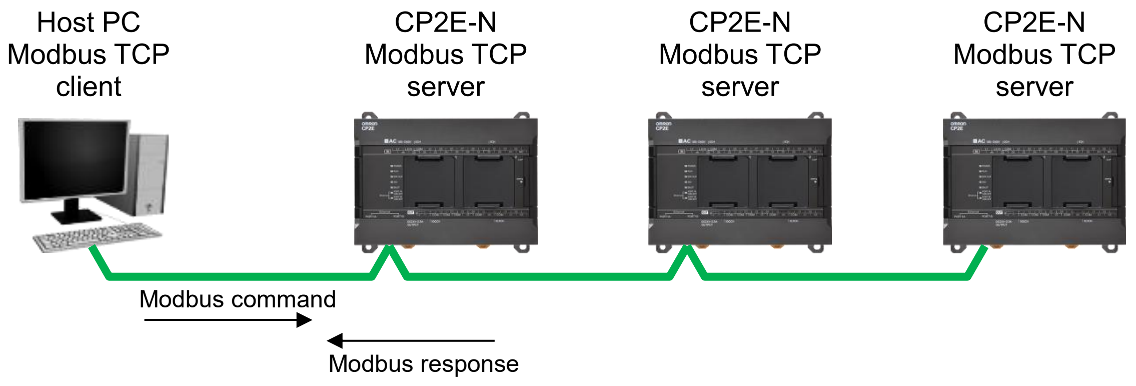
Practices Guide

1. Modbus TCP Server Function Block

The Modbus TCP Server Function Block executes the Modbus TCP Server function using the built-in Ethernet port.

1.1 Overview of Function Block

The CP2E-N-type CPU Unit with the Ethernet Socket Service function can be used as a Modbus TCP server. The Modbus TCP Server function automatically responds to access to the Work Area or Data Memory Area of the CP2E CPU Unit from a Modbus TCP client on the host PC or PLC. The function block eliminates the need for programming data exchange, making it easy to execute the Modbus TCP Server function.

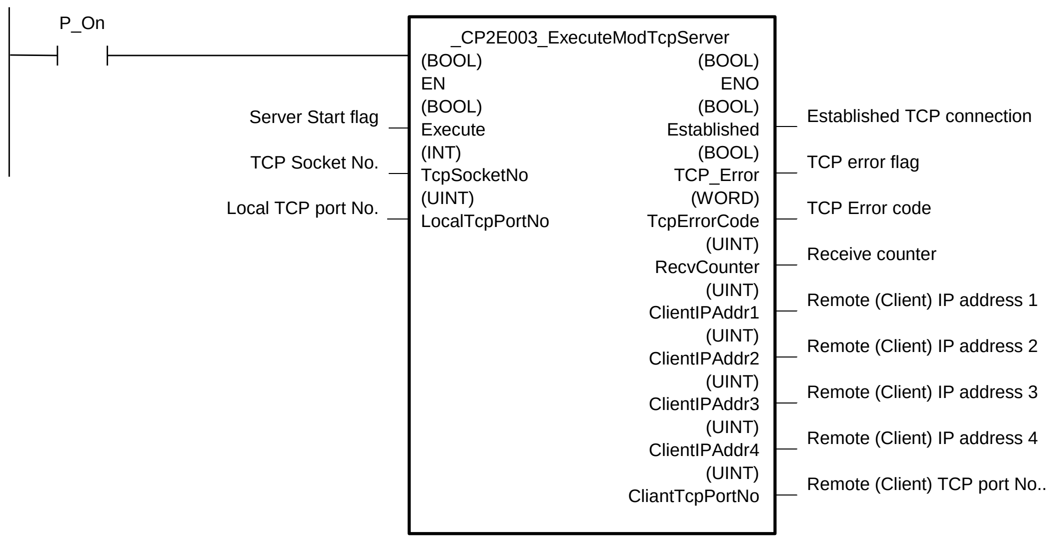


1.2 Function Block to Use

The Modbus TCP Server Function Block automatically responds to a Modbus command sent to the built-in Ethernet port.

For details on the function block, such as functions and memory area allocation, refer to *Description of Functions*.

Modbus TCP server: _CP2E003_ExecuteModTcpServer



The supported Modbus functions are listed below.

Function code	Modbus name	Function
01 Hex	Read Coils	Reads multiple bits from Work Area (W)
03 Hex	Read Holding Registers	Reads multiple words from Data Memory Area (D)
06 Hex	Write Single Register	Writes a word to Data Memory Area (D)
0F Hex	Write Multiple Coils	Writes multiple bits to Work Area (W)
10 Hex	Write Multiple Registers	Writes multiple words to Data Memory Area (D)

Precautions for Correct Use of Function Blocks

- This function block uses the Socket Service function on the built-in Ethernet port.
  This function block uses one TCP/IP connection. Up to thee function blocks can be used because the CP2E CPU Unit supports up to three TCP/IP connections.
- Do not use the same socket number, memory and auxiliary area allocation used for this function block in a ladder program.

Socket No.	DM area	AR area	Work area
TCP Socket No.1	D16000, D16004, D16008 to D16017	A567, A571.00 to 07	D15000 to D15149
TCP Socket No.2	D16001, D16005, D16018 to D16027	A568, A571.08 to 15	D15150 to D15299
TCP Socket No.3	D16002, D16006, D16028 to D16037	A569, A572.00 to 07	D15300 to D15449

- TCP passive open request and TCP close request are executed in this function block.
- The remote IP address cannot be specified. All TCP connection requests specified on the local TCP port number are accepted.

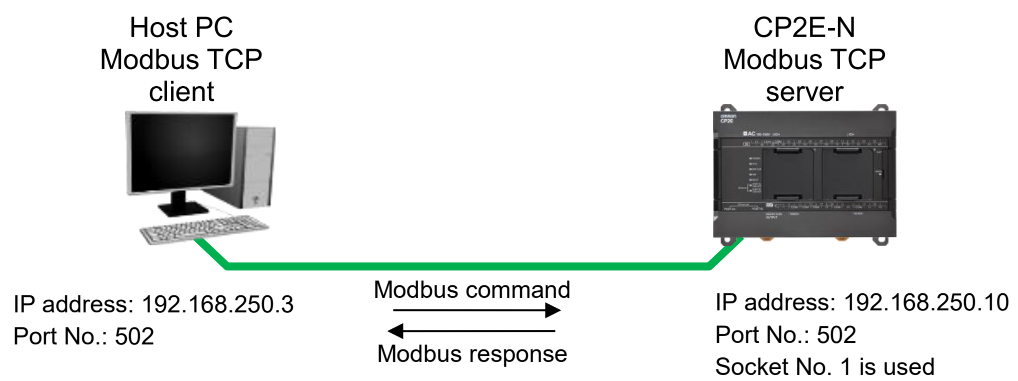
2. Operating Procedure

Connect a PC to the CP2E CPU Unit via the Ethernet port to exchange data using the Modbus TCP protocol.

Wiring	Connect the CP2E CPU Unit and host PC using an Ethernet cable.
↓	
PLC Settings	Make built-in Ethernet settings at PLC Settings in the CX-Programmer and transfer the settings to the CP2E CPU Unit. The settings are enabled when the power supply is turned ON.
↓	
Ladder programming	Insert the Modbus TCP Server Function Block to create a program.

3. Programming Example

The CP2E CPU Unit responds to a Modbus command sent from the host PC.



3.1 Wiring Example

Connect the CP2E CPU Unit and the host PC using an Ethernet cable.

3.2 PLC Settings Example

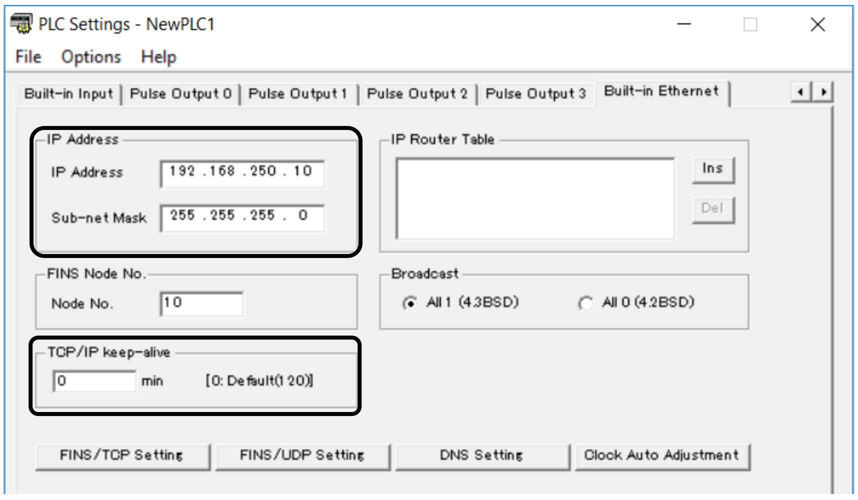
(1) Ethernet Setting

Start the CX-Programmer.

Built-in Ethernet Tab

Select the Built-in Ethernet Tab in the PLC Settings.

Set the IP address, subnet mask, and TCP/IP keep-alive of the CP2E CPU Unit.



IP Address

IP Address: 192 . 168 . 250 . 10

Sub-net Mask: 255 . 255 . 255 . 0

FINS Node No.

Node No.: 10

Broadcast: [✓ All 1 (4.3BSD)] All 0 (4.2BSD)

TCP/IP keep-alive: 0 min [0: Default(120)]

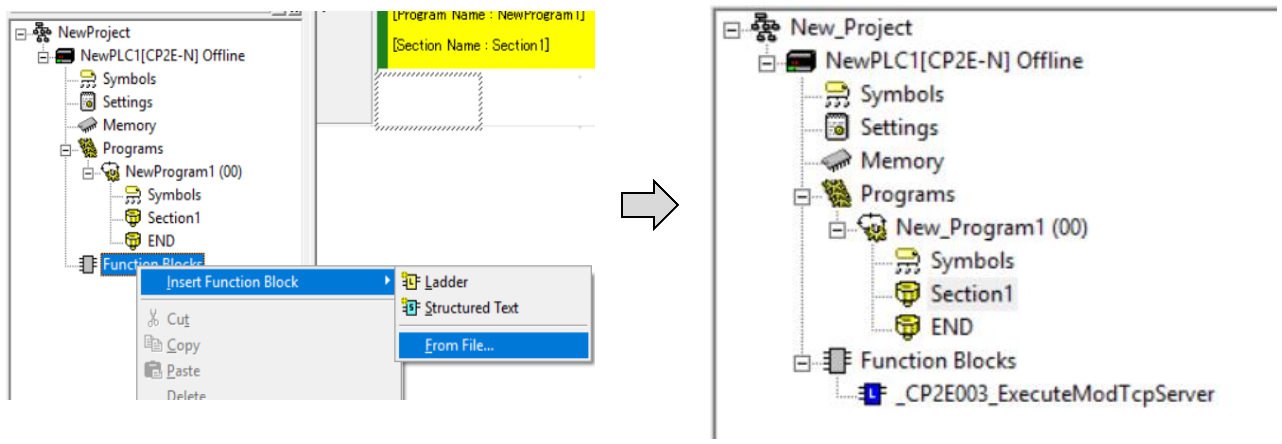
Details of settings

Item	Description
IP Address	Set the local IP address.
Sub-net Mask	Set the subnet mask.
TCP/IP keep-alive	Set the liveness-checking interval. Use the default value of 0 (120 min).

3.3 Ladder Programming Example

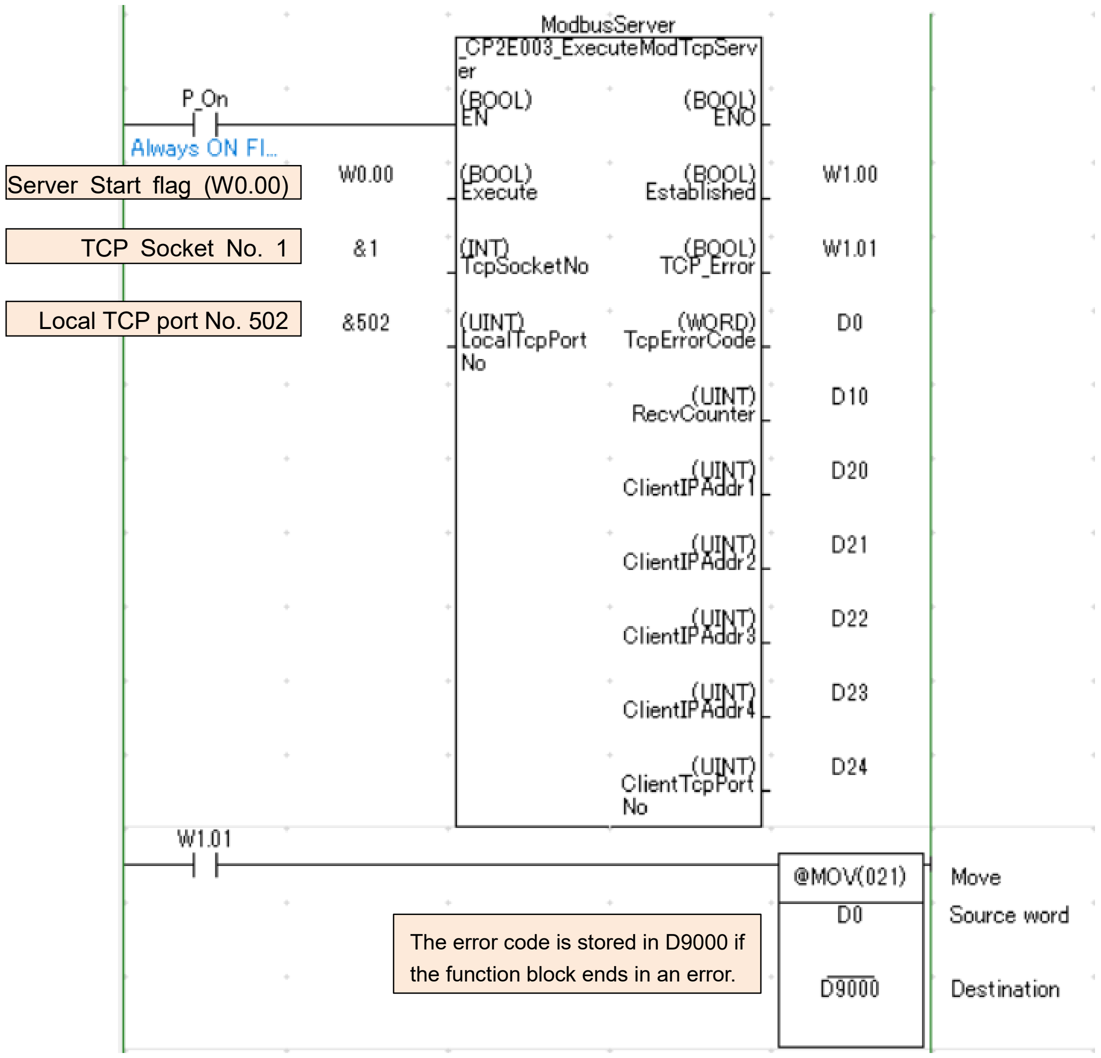
(1) Inserting the Function Block

Save the Modbus TCP server: _CP2E003_ExecuteModTcpServer Function Block file to your PC beforehand. Right-click ***Function Blocks*** at the project workspace in the CX-Programmer and select ***Insert Function Blocks*** - ***From File*** from the pop-up menu to load the _CP2E003_ExecuteModTcpServer.cxf file.



(2) Ladder Programming Example

Create a ladder program for the Modbus TCP server.



- The function block starts the Modbus TCP Server function with the TCP socket No. 1.
- When *Execute* (W0.00) is turned ON, a passive TCP socket is opened with the local TCP port No. 502. Open an active TCP socket from the host PC.
  Do not turn ON EN and Server Start flag (Execute) at the same time. If EN and Server Start flag (Execute) are turned ON at the same time, Modbus server function will not be executed.
- When communication between the host PC and CP2E CPU Unit is established, *Established* (W1.00) is turned ON and the CP2E Unit waits for a Modbus command.
- The IP address of the client is stored in D20, D21, D22, and D23, and the client TCP port number in D24.
- When a socket communication error occurs, the error code is stored in D0. When sending the data successfully next time, the function block clears D0 and stores the error code in D9000.

Additional information

After S*erver Start flag (Execute)* is turned OFF, it should be waited 120 second until next socket open. If Server Start flag will be turned ON within 120s, *TCP error flag (TCP_Error)* is turned ON and *TCP Error code (TcpErrorCode)* will be stored #2211.

■Practices Guide　ーRevision History

Version	**Date**	**Revised content**
A	October 2019	Original production

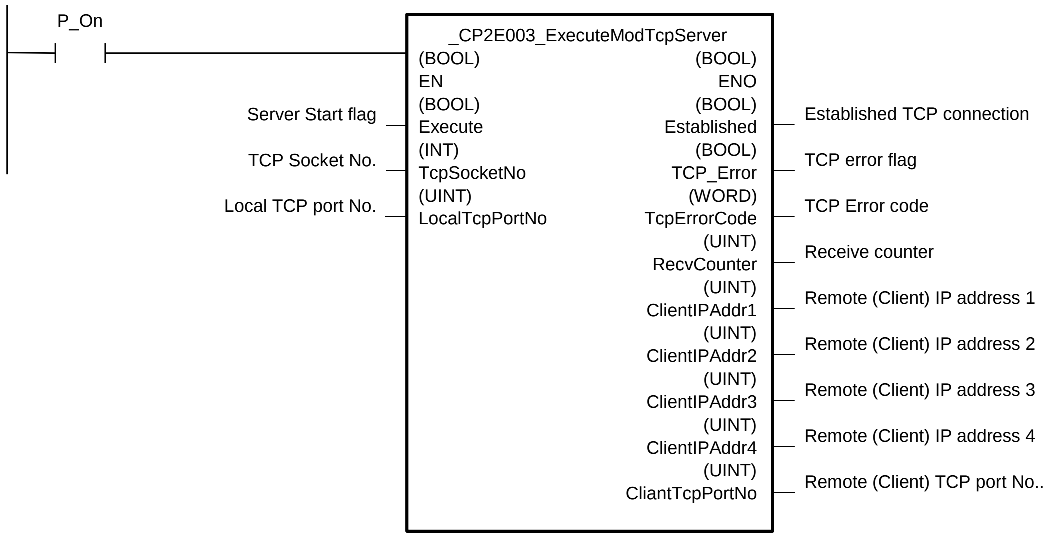
Description of Functions

CP2E 003	Modbus TCP server: _CP2E003_ExecuteModTcpServer

Basic function

Execute Modbus TCP server function with Built-in Ethernet Port Socket function.

Symbol



File name

_CP2E003_ExecuteModTcpServer.cxf

Applicable models

CPU Unit	CP2E
CX-Programmer	Version 9.72 or higher

Conditions for usage

Shared Resources

- Data Memory Area allocation for Socket service
- Auxiliary Area allocation for Socket service
- Work Area for Socket send/receive data

Socket No.	DM area	AR area	Work area
TCP Socket No.1	D16000, D16004 D16008 to D16017	A567 A571.00 to 07	D15000 to D15149
TCP Socket No.2	D16001, D16005 D16018 to D16027	A568 A571.08 to 15	D15150 to D15299
TCP Socket No.3	D16002, D16006 D16028 to D16037	A569 A572.00 to 07	D15300 to D15449

Function description

- When *Server Start flag (Execute)* is turned ON, passive TCP socket is opened on the specified *TCP Socket No. (TcpSocketNo) and Local TCP port No. (LocalTcpPortNo)*.
- After connection is established, Modbus TCP server is executed and waiting Modbus command.
- When *Server Start flag (Execute)* is turned OFF, passive TCP socket is closed.
- During connection is established, *Established TCP connection (Established)* is ON
  And client IP address and client TCP port No. is stored.
- If socket processing produces an error, ON is set to *TCP error flag (TCP_Error)* and socket error response code is output the *TCP error code (TcpErrorCode)*.
- If Modbus command is received, *Receive counter (RecvCounter)* is incremented.
- The supported Modbus command on this FB is following table.
  The details are refered to *Command and Response Details*.

Function code	Command	Function
01 Hex	Read Coils	Reads multiple bits from work area (W)
03 Hex	Read Holding Registers	Reads multiple words from data memory area (D)
06 Hex	Write Single Register	Writes a word to data memory area (D)
0F Hex	Write Multiple Coils	Writes multiple bits to work area (W)
10 Hex	Write Multiple Registers	Writes multiple words to data memory area (D)

Kind of FB definition

Connect Always ON type

Connect the *EN* input to the *Always ON Flag (P_On)*

- The same instance can not be used in two or more places.

FB precautions

- The FB is used Socket service function on built-in Ethernet port. So, do not use data memory and auxiliary area allocation for socket service specified on TCP socket No. in ladder.
- Status of TCP Socket and TCP connection is confirmed the following area.

Socket No.	TCP socket status	TCP connection status
TCP socket No.1	A567	D16004
TCP socket No.2	A568	D16005
TCP socket No.3	A569	D16006

- TCP passive open request and TCP close request is executed in this FB.
- Remote IP address and TCP port No. are not specified. All TCP connection requests specified on local TCP port No. is accepted.
- ENO is ON during TCP connection status is LISTEN and ESTABLISHED
- After *Server Start flag (Execute)* is turned OFF, it should be waited 120 second until next socket open. If *Server Start flag* will be turned ON within 120s, *TCP error flag (TCP_Error)* is turned ON and *TCP Error code (TcpErrorCode)* will be stored #2211.

EN input condition

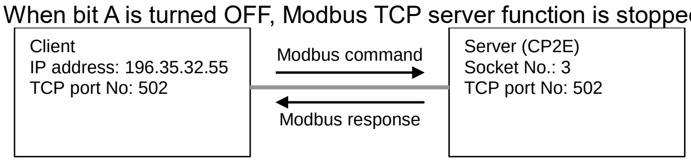
- Connect the EN input to the A*lways ON Flag (P_On)*.
- If a different type of bit is connected to *EN*, the FB outputs will be maintained when the connected bit is turned OFF.

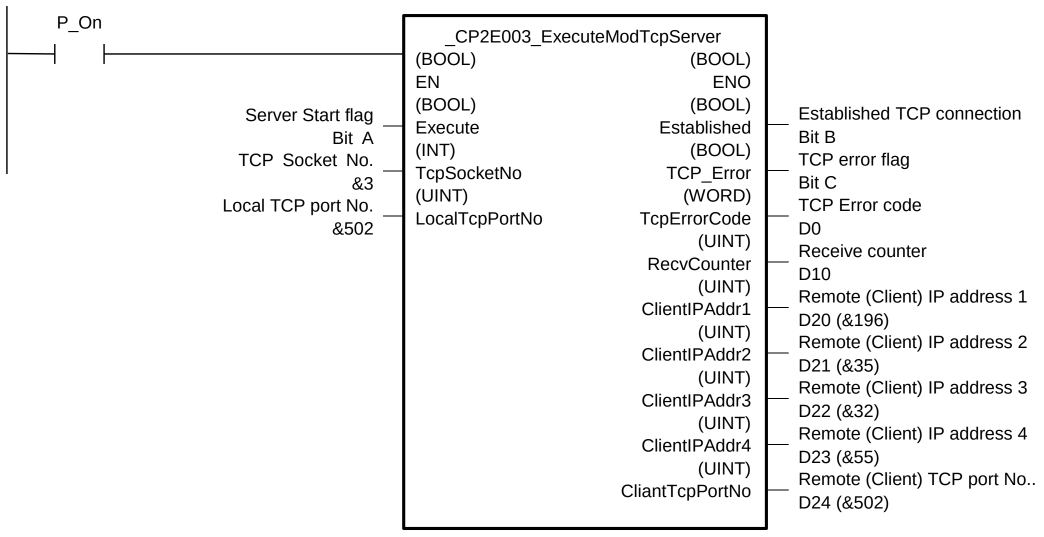
Restrictions Input variables

- Use the *Always ON Flag (P_On)* for *EN*.
- Do not turn ON *EN* and *Server Start flag (Execute)* at the same time. If *EN* and *Server Start flag (Execute)* are turned ON at the same time, Modbus server function will not be executed.
- If the input variables are out of range, the *ENO* will turn OFF and the FB will not be processed.

Application example

When bit A turned ON, passive TCP socked is opened with socket No.3. After TCP connection is established, Modbus TCP server function is started
When bit A is turned OFF, Modbus TCP server function is stopped and closed socket service.





Related manuals

CP2E CPU Unit Software User’s manual (W614)

■ **Variable Tables**

Input Variables

Name	Variable name	Data type	Default	Range	Description
EN	EN	BOOL			1 (ON): FB started. 0 (OFF): FB not started.
Server Start flag	Execute	BOOL			1 (ON): Start Modbus TCP server 0 (OFF): Stop Modbus TCP server
TCP socket No.	TcpSocketNo	INT	&1	&1 to &3	&1: Use socket No.1 &2: Use socket No.2 &3: Use socket No.3
Local TCP port No.	LocalTcpPortNo	UINT	&502	&1 to &65535	

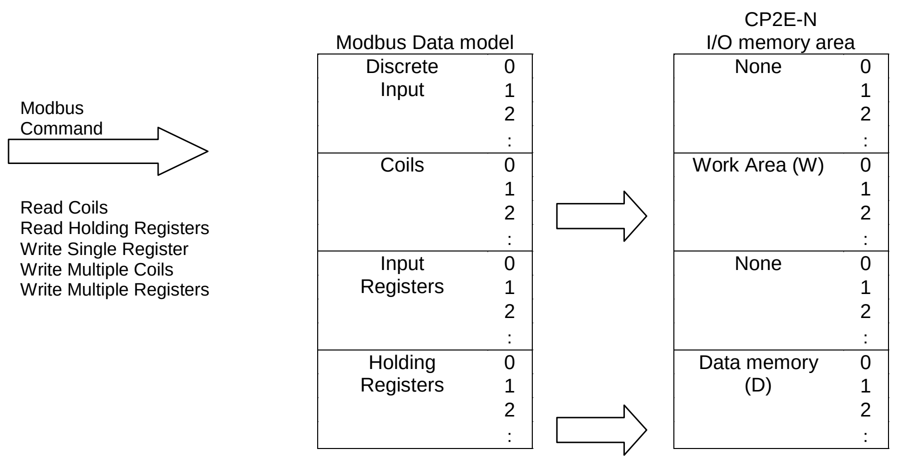
Output Variables

Name	Variable name	Data type	Range	Description
ENO	ENO	BOOL		1 (ON): connection is LISTEN or ESTABLISHED. 0 (OFF): connection is other status.
Establised TCP connection	Established	BOOL		1 (ON): TCP connection is established. 0 (OFF) : not established.
TCP error flag	TCP_Error	BOOL		Turns ON when TCP processing ends in an error.
TCP error code	TcpErrorCode	WORD		Outputs the error code when TCP execution ends in an error. Refer to the *CP2E CPU Unit Software User's manual (W614) 15-5 Socket Service* for details on the error codes. When new Modbus TCP command is received, TCP error code is updated.
Receive counter	RecvCounter	UINT		When Modbus command is received, receive counter is incremented.
Remote (Client) IP address 1	ClientIPAddr1	UINT		Outputs Modbus TCP client IP address "IP address1. IP address2. IP address3. IP address4"
Remote (Client) IP address 2	ClientIPAddr2	UINT		
Remote (Client) IP address 3	ClientIPAddr3	UINT		
Remote (Client) IP address 4	ClientIPAddr4	UINT		
Remote (Client) TCP port No.	ClientTcpPortNo	UINT		Output Modbus TCP Client port No.

■ Operation specifications of Modbus

Modbus has the following four data models.
CP2E allocates each area of these data models to an I/O memory are.

Data model	Date type	Read/Write	CP2E-N CPU unit I/O memory allocation
Discrete input	Bit	Read	None
Coils	Bit	Read/Write	Work Area (W)
Input Registers	Word (16bit)	Read	None
Holding Registers	Word (16bit)	Read/Write	Data Memory (D)



CP2E fixed allocations
The following table gives the relationship between Modbus data model and CP2E I/O memory.

Modbus data model	Modbus address	Address specified in Modbus command	Corresponding CP2E I/O memory address
Discrete input	-	-	-
Coils	1～2048	0～2047	W0.00～W127.15
Input Registers	-	-	-
Holding Registers	1～15000	0～14999	D0～D14999

* : Addresses in Modbus data models start from 1, but addresses specified in Modbus commands and addresses in the CP2E CPU Unit start from 0. Refer to the above table when specifying addresses in applications.

Supported Modbus command

Function code	Command	Function
01 Hex	Read Coils	Reads multiple bits from work area (W)
03 Hex	Read Holding Registers	Reads multiple words from data memory area (D)
06 Hex	Write Single Register	Writes a word to data memory area (D)
0F Hex	Write Multiple Coils	Writes multiple bits to work area (W)
10 Hex	Write Multiple Registers	Writes multiple words to data memory area (D)

■ Command and Response Detail

- Function 01 Read Coils : Reads multiple bits from work area (W)

Request

Field name	Data length	Data
Function code	1 byte	01 Hex
Coil starting address	2 bytes	0 to 07FF Hex (0 to 2047 : W0.00 to W127.15)
Quantity of coils	2 bytes	1 to 07D0 Hex (1 to 2000)

* : The maximum quantity of coils depends on the assigned starting address.
Coil starting address + Quantity of coils < 2048

Response

Field name	Data length	Data
Function code	1 byte	01 Hex
Byte count	1 byte	N *
Coil status	n bytes	n=N or N+1

* : N = Quantity of coils/8, if the remainder is different of 0, N = N+1.

Example : Read 24 bit from W1.00 to W2.07

Request (Modbus client)

Field name	Data
Function code	01 Hex
Coil starting address (High)	00 Hex
Coil starting address (Low)	14 Hex (from 1bit : from W1.04)
Quantity of coils (High)	00 Hex
Quantity of coils (Low)	14 Hex (14 bit) (from W1.04 to W2.07)

Response (CP2E-N)

Field name	Data
Function code	01 Hex
Byte count	02 Hex
Coil status 16 to 23	C0 Hex (from W1.04 to W1.11)
Coil status 24 to 31	34 Hex (from W1.12 to W2.03)
Coil status 32 to 39	0D Hex (from W2.04 to W2.07)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W0CH	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W1CH	31 (0)	30 (1)	29 (0)	28 (0)	27 (1)	26 (1)	25 (0)	24 (0)	23 (0)	22 (0)	21 (0)	20 (0)	19	18	17	16
W2CH	47	46	45	44	43	42	41	40	39 (1)	38 (1)	37 (0)	36 (1)	35 (0)	34 (0)	33 (1)	32 (1)
W3CH	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48

* : The subscript numbers in the shaded boxes indicate the ON/OFF (1/0) status of the bits that are read.

- Function 03 Read Holding Registers : Reads multiple words from data memory area (D)

Request

Field name	Data length	Data
Function code	1 byte	03 Hex
Register starting address	2 bytes	0 to 3A97 Hex (D0 to D14999)
Quantity of registers	2 bytes	1 to 007D Hex (1 to 125)

* : The maximum quantity of registers depends on the assigned starting address.
Register starting address + Quantity of registers < 15000

Response

Field name	Data length	Data
Function code	1 byte	01 Hex
Byte count	1 byte	2 x N (N: Quantity of registers)
Register value	2 x N byte	

Example : Read 3 words from D1000 to D1002

Request (Modbus client)

Field name	Data
Function code	03 Hex
Register starting address (High)	03 Hex
Register starting address (Low)	E8 Hex (D1000～)
Quantity of register (High)	00 Hex
Quantity of register (Low)	03 Hex (3CH) (D1000～D1002)

Response (CP2E-N)

Field name	Data
Function code	03 Hex
Byte count	06 Hex
Register value (High)	AB Hex (D1000 High)
Register value (Low)	12 Hex (D1000 Low)
Register value (High)	56 Hex (D1001 High)
Register value (Low)	78 Hex (D1001 Low)
Register value (High)	97 Hex (D1002 High)
Register value (Low)	13 Hex (D1002 Low)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D1000	A				B				1				2			
D1001	5				6				7				8			
D1002	9				7				1				3			

- Function 06 Write Single Register : Writes a word to data memory area (D)

Request

Field name	Data length	Data
Function code	1 byte	06 Hex
Register address	2 bytes	0 to 3A97 Hex (D0 to D14999)
Register value	2 bytes	0000 to FFFF Hex

Response

Field name	Data length	Data
Function code	1 byte	06 Hex
Register address	2 bytes	0 to 3A97 Hex (D0 to D14999)
Register value	2 bytes	0000～FFFF Hex

Example : Write 3AC5 Hex to D2000

Request (Modbus client)

Field name	Data
Function code	06 Hex
Register address (High)	07 Hex
Register value (Low)	D0 Hex (D2000)
Register value (High)	3A Hex
Register value (Lowh)	C5 Hex

Response (CP2E-N)

Field name	Data
Function code	06 Hex
Register address (High)	07 Hex
Register value (Low)	D0 Hex (D2000)
Register value (High)	3A Hex (D2000 High)
Register value (Lowh)	C5 Hex (D2000 Low)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D2000		3				A				C				5		
D2001																

- Function 0F Write Multiple Coils : Writes multiple bits to work area (W)

Request

Field name	Data length	Data
Function code	1 byte	0F Hex
Coil starting address	2 bytes	0 to 07FF Hex (0 to 2047 : W0.00 to W127.15)
Quantity of coils	2 bytes	1 to 07B0 Hex (1 to 1968)
Byte count	1 byte	N *
Coil value	N or N+1 bytes	Coil value

* : N = Quantity of coils/8, if the remainder is different of 0, N = N+1.
* : The maximum quantity of coils depends on the assigned starting address.
Coil starting address + Quantity of coils < 2048

Response

Field name	Data length	Data
Function code	1 byte	0F Hex
Coil starting address	2 bytes	0 to 7FF Hex (0 to 2047 : W0.00 to W127.15)
Quantity of coils	2 bytes	1 to 07B0 Hex (1 to 1968)

Example : Write 12bit from W1.00 to W1.11.

Request (Modbus client)

Field name	Data
Function code	06 Hex
Coil starting address (High)	00 Hex
Coil starting address (Low)	10 Hex (16 bit : from W1.00)
Quantity of coils (High)	00 Hex
Quantity of coils (Low)	0C Hex (20 bit) (from W1.00 to W1.11)
Byte count	2 Hex
Coil value 16 to 23	A2 Hex
Coil value 24 to 27	0C Hex

Response (CP2E-N)

Field name	Data
Function code	06 Hex
Coil starting address (High)	00 Hex
Coil starting address (Low)	10 Hex
Quantity of coils (High)	00 Hex
Quantity of coils (Low)	0C Hex

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W0CH	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W1CH	31	30	29	28	27 (1)	26 (1)	25 (0)	24 (0)	23 (1)	22 (0)	21 (1)	20 (0)	19 (0)	18 (0)	17 (1)	16 (0)
W2CH	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
W3CH	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48

* : The subscript numbers in the shaded boxes indicate the ON/OFF (1/0) status of the bits that are wrote.

- Function 10 Write Multiple Registers : Write Multiple Words in the Data Memory (D)

Request

Field name	Data length	Data
Function code	1 byte	10 Hex
Register starting address	2 bytes	0 to 3A97 Hex (D0 to D14999)
Quantity of register	2 bytes	1 to 007B Hex (1 to 123)
Byte count	1 byte	2 x N *
Register value	2 x N bytes	Register value

* : The maximum quantity of registers depends on the assigned starting address.
Register starting address + Quantity of registers < 15000

Response

Field name	Data length	Data
Function code	1 byte	10 Hex
Register starting address	2 bytes	0 to 3A97 Hex (D0 to D14999)
Quantity of register	2 bytes	1 to 007B Hex (1 to 123)

Example : Write 2 words (3AC5 Hex and 9713 Hex) from D1000 to D1001.

Request (Modbus client)

Field name	Data
Function code	06 Hex
Register starting address (high)	03 Hex
Register starting address (Low)	E8 Hex (D1000)
Quantity of register (high)	00 Hex
Quantity of register (Low)	02 Hex (2 words)
Byte count	04 Hex
Register value (high)	3A Hex
Register value (Low)	C5 Hex
Register value (high)	97 Hex
Register value (Low)	13 Hex

Response (CP2E-N)

Field name	Data
Function code	06 Hex
Register starting address (high)	03 Hex
Register starting address (Low)	E8 Hex
Quantity of register (high)	00 Hex
Quantity of register (Low)	02 Hex
Byte count	

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D1000	3				A				C				5			
D1001	9				7				1				3			

・Exception response at receiving illegal request

Response

Field name	Data length	Data
Function code	1 byte	Function code + 80　Hex Example 86 Hex for Write Single Register (06 Hex)
Exception code	1 byte	refer to following table

Exception codes

Exception code	Name	Meaning
01 Hex	Illegal function code	・ An unsupported function code is specified.
02 Hex	Illegal data address	・ There is an error in the specified start address ・ The specified start address and Data length exceed the valid range.
03 Hex	Illegal data value	・ Data number does not match data length

■ Version History

Version	Date	Contents
1.00	2019.11	Original production

Note
This manual is a reference that explains the function block functions.
It does not explain the operational limitations of Units, components, or combinations of Units and components. Always read and understand the Operation Manuals for the system's Units and other components before using them.