

Function Block



OMRON ELECTRONICS S.A.S.
14 Rue de Lisbonne
93561 Rosny-sous-Bois cedex

N° Indigo 0 825 825 679
0.15€ TTC/mn

Reference	MTCP_NJ_Server
Revision	1.9
Author	JP Viskovic
Date	16/08/2016
+ Support	http://support-omron.fr/

Modbus TCP Server for NJ Controller

Function	Modbus TCP server for Built-in Ethernet Port of NJ Controller
Server	
File	MTCP_NJ.zip

Conditions of use	<p>The FB Modbus TCP Server provides some read/write features in accordance with the specifications defined by the Modbus organization.</p> <p>The Modbus TCP Server function block is offered 'as is' and may serve as a basis for development. Users should previously test its adequacy to the final application. Omron could not be held responsible in case of malfunction.</p>																
Principe	<p>The function block MTCP_NJ_Server switch to LISTEN mode waiting for a Modbus TCP Client connection when Start input is activated. We recommend to use the FB in a periodical task to not overload the primary task.</p> <p>List of functions supported by MTCP_NJ_Server:</p> <table border="1"> <thead> <tr> <th>Code</th> <th>Modbus Function</th> </tr> </thead> <tbody> <tr> <td>0x01</td> <td>Read Coils</td> </tr> <tr> <td>0x02</td> <td>Read Discret Inputs</td> </tr> <tr> <td>0x03</td> <td>Read Holding Registers</td> </tr> <tr> <td>0x04</td> <td>Read Input Registers</td> </tr> <tr> <td>0x05</td> <td>Write Single Coil</td> </tr> <tr> <td>0x06</td> <td>Write Single Register</td> </tr> <tr> <td>0x10</td> <td>Write Multiple Registers</td> </tr> </tbody> </table>	Code	Modbus Function	0x01	Read Coils	0x02	Read Discret Inputs	0x03	Read Holding Registers	0x04	Read Input Registers	0x05	Write Single Coil	0x06	Write Single Register	0x10	Write Multiple Registers
Code	Modbus Function																
0x01	Read Coils																
0x02	Read Discret Inputs																
0x03	Read Holding Registers																
0x04	Read Input Registers																
0x05	Write Single Coil																
0x06	Write Single Register																
0x10	Write Multiple Registers																

1- I/O variable of MTCP_NJ_Server

Input variable

Name	type	range	Description
Start	Bool	OFF, ON	Server Activation

Input/output Variables

Name	type	range	Description
Registers	Array of 1024 Words	0-FFFF	Register area
Coils	Array of 1024 bool	OFF, ON	Coil area

Output Variables

Name	type	Range	Description
Connected	Bool	OFF, ON	ON : a client is connected
Error	Bool	OFF, ON	Error flag
ErrorID	UINT	0 - FFFF	Error Code returned by the socket or Modbus TCP server (see error code list below).
TCP_Status	_eCONNECTION_STATE	Enum	_CLOSED _LISTEN _SYN SENT _SYN RECEIVED _ESTABLISHED _CLOSE_WAIT _FIN WAIT1 _CLOSING _LAST ACK _FIN WAIT2 _TIME WAIT
IP_Client	String[24]	w.x.y.z	IP address of connected client
Port_Client	UINT	0-65535	Port number of connected client

Error Code returned in ErrorID

Code		Description
0001	Modbus Exception	ILLEGAL FUNCTION
0002		ILLEGAL DATA ADDRESS
0003		ILLEGAL DATA VALUE
2000	Socket error	Local IP Address Setting Error
2001		TCP/UDP Port Already in Use
2002		Address Resolution Failed
2003		Status Error
2004		Local IP Address Not Set
2006		Socket Timeout
2007		Socket Handle Out of Range
2008		Socket Communications Resource Overflow

Precautions in Using Socket Services

Precautions for UDP and TCP Socket Services

- Communications processing are sometimes delayed when multiple functions of the built-in EtherNet/IP port are used simultaneously or due to the contents of the user program.
- Communications efficiency is sometimes reduced by high communications traffic on the network line.
- The close processing for a close request instruction discards all of the buffered send and receive data for the socket. For example, send data from a send request instruction immediately before the close processing is sometimes not sent.
- After a socket is open, the built-in EtherNet/IP port provides a receive buffer of 9,000 bytes per TCP socket and 9,000 bytes per UDP socket to enable data to be received at any time. If the receive buffer is full, data received by that socket is discarded. Make sure that the user application always executes receive requests to prevent the internal buffer from becoming full.

Precautions for UDP Socket Services

- The destination IP address can be set to a broadcast address for a UDP socket to broadcast data to all nodes on the network. However, in this case, the maximum length of send data is 1,472 bytes. Data lengths broken into multiple fragments (1,473 bytes or more in UDP) cannot be sent.
- For UDP socket, controls to confirm the reliability of communications, such as the confirmation of send data, are not performed. To improve the reliability of communications when you use UDP sockets, make sure the user program confirms that data is sent and resends data when necessary.

Precautions for TCP Socket Services

- If the TCP socket is closed on the remote node without warning during communications (i.e., if the connection is closed), the socket at the local node must also be closed. You can use the Read TCP Socket Status instruction (SktGetTCPstatus) to see if the connection is closed. Immediately close the socket at the local node if the TCP socket at the remote node is closed.
- If the remote node's TCP socket closes without warning, the data to send may remain in the buffer at the local node. The remaining data is discarded in the local node's TCP close processing. The steps that are required in applications to avoid this include sending data from the sending node that permits closing and closing the socket only after checking the remote node.
- While open processing is performed for a TCP socket, a port that was closed first cannot be opened again for 60 seconds from the time the close processing is performed for the remote socket. However, this is not true if you specified 0 (automatic assignment by the Unit) as the port for the SktTCPConnect instruction.
- You can use *Connect* from another socket to open a connection to a socket that was opened with *Accept*. A connection is not opened if you try to use *Connect* from another socket to open a connection to a socket that was opened with *Connect*. Also, a connection is not opened if you attempt to use *Accept* from another socket to open a socket that was opened with *Accept*. Furthermore, you cannot use *Connect* from more than one other node to establish multiple connections with a single TCP socket that was opened with *Accept* on the built-in EtherNet/IP port.
- You can use the keep-alive function for TCP sockets at the built-in EtherNet/IP port. The keep alive function checks whether a connection is normally established when no data is sent or received for a certain period on the communications line where the connection was established. The built-in EtherNet/IP port responds to checks from other nodes even if keep alive is not specified.