Littelfuse
STARTCO

3714 Kinnear Place Saskatoon, SK Canada S7P 0A6 Ph: (306) 373-5505 Fx: (306) 374-2245 www.littelfuse.com/relayscontrols

MPU-32 AND FPU-32 MODBUS-TCP INTERFACE

REVISION 0-A-051414

Document Number:PM-1117-EN
Printed in Canada.

This page intentionally left blank.

TABLE OF CONTENTS

LIST OF TABLES

DISCLAIMER

This page intentionally left blank.

1. GENERAL

The Ethernet network communications interface supports the Modbus-RTU protocol over TCP/IP using an encapsulating protocol called Modbus TCP. Each MPU-32 or FPU-32 is a slave on the network and each slave can support up to five (5) masters. Up to 254 slaves can be connected on a network. Standard off-the-shelf Ethernet hardware is all that is required for the communications network.

2. MODBUS PROTOCOL

The MPU-32 and FPU-32 implement the Modbus®-TCP protocol on port 502. Only the master can initiate a transaction. Messages can be addressed to individual slaves or they can be broadcast messages (to slave address 255). Broadcast messages are executed on the slaves but unlike individually addressed messages, the slaves do not generate a reply message.

Modicon Modbus® is a registered trademark of Schneider Electric.

2.1 PROTOCOL SETUP

Configuration options are available in the *Setup* | *Hardware* | *Network Comms* menu. Select *Network ID, Ethernet IP, Ethernet Mask* and *Gateway*; then select *Modbus TCP* from the *Network Type* menu. When network parameters need to be changed, first set the *Network Type* to *None*, make the parameter adjustments and then select *Modbus TCP* as the network type. A *Network ID* of 0 disables communications. Valid slave addresses are 1 to 254.

2.2 ERROR CHECKING

Modbus TCP uses the TCP/IP checksum and error correction techniques to ensure reliable communications.

If the checksum is correct but the internal data in the message is not correct, the MPU-32 or FPU-32 will respond with an exception code.

2.3 FUNCTION CODES SUPPORTED

The MPU-32 and FPU-32 Modbus Protocol supports the following function codes:

- Read Holding Registers (Function Code 3)
- Read Input Registers (Function Code 4)
- Read Device Identification (Function Code 43)
- Write Single Register (Function Code 6)
- Write Multiple Registers (Function Code 16)
- Command Instruction (Function Code 5)

Function Codes 3 and 4 perform the same function in the slave.

Registers in Modbus start at 40001 decimal and the register address generated for this register is 0. See Appendix E in the MPU-32 or FPU-32 manual for the Modbus Register Table.

NOTE: For hexadecimal numbers, 0x precedes the value.

2.3.1 MODBUS-TCP HEADER

All Modbus-TCP messages are essentially Modbus-RTU messages encapsulated with a Modbus-TCP header, both of which are encapsulated in a TCP and an IP header. The TCP/IP-header information is beyond the scope of this document.

The first two bytes of the Modbus-TCP header are the transaction identifier. The slave simply returns the transaction identifier specified by the master. The next two bytes are the protocol identifier and should be zero. Following the protocol identifier are two more bytes that specify the length of the encapsulated Modbus-RTU packet. The high order length byte should always be zero because Modbus-RTU packets cannot exceed 256 bytes. The Modbus-RTU packet immediately follows the length.

TABLE 1. MODBUS-TCP HEADER

HEX BYTE	DESCRIPTION
Byte 1	MSB Transaction Identifier
Byte 2	LSB Transaction Identifier
Byte 3	MSB Protocol Identifier
Byte 4	LSB Protocol Identifier
Byte 5	MSB Length
Byte 6	LSB Length

2.3.2 READ INPUT/HOLDING REGISTERS (CODE 4/3)

Starting after the Modbus-TCP header, the first byte of the read message is the slave address. The second byte is the function code. Bytes three and four indicate the starting register. The next two bytes specify the number of 16-bit registers to read. For those familiar with Modbus RTU, note the lack of CRC is because error handling is taken care of by TCP/IP. Table 2 shows the packet starting after the Modbus-TCP header.

TABLE 2. READ REGISTERS (CODE 4/3)

HEX BYTE	DESCRIPTION
Byte 1	Slave Address
Byte 2	Function Code
Byte 3	MSB Register Address
Byte 4	LSB Register Address
Byte 5	MSB Number of Registers
Byte 6	LSB Number of Registers

The two-byte values of starting register and number of registers to read are transmitted with the high-order byte followed by the low-order byte.

The following message will obtain the value of register 1 (Modbus 40002) from slave 1. Note that Modbus registers are numbered from zero (40001 = zero, 40002 = one, etc.):

0x00 /0x00 /0x00 /0x00 /0x00 /0x06 /0x01 /0x03 /0x00 /0x01 /0x00 /0x01

The addressed slave responds with the Modbus-TCP header, its address, Function Code 3, followed by the information field. The information field contains an 8-bit byte count and the 16-bit data from the slave. The byte count specifies the number of bytes of data in the information field. The data in the information field consists of 16-bit data arranged so that the MSB is followed by the LSB.

2.3.3 WRITE SINGLE REGISTER (CODE 6)

The function code format for writing a single register is shown in Table 3.

The message consists of the Modbus-TCP header, then the slave address followed by the Function Code 6 and two 16-bit values. The first 16-bit value specifies the register to be modified and the second value is the 16-bit data.

Provided no errors occurred, the slave will resend the original message to the master. The response message is returned only after the command has been executed by the slave.

The following message will set register 3 to 300 in slave 5:

0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x06 | 0x05 | 0x06 | 0x00 | 0x03 | 0x01 | 0x2C

TABLE 3. WRITE SINGLE REGISTER (CODE 6)

HEX BYTE	DESCRIPTION
Byte 1	Slave Address
Byte 2	Function Code
Byte 3	MSB Register Address
Byte 4	LSB Register Address
Byte 5	MSB of Data
Byte 6	LSB of Data

2.3.4 WRITE MULTIPLE REGISTERS (CODE 16)

The function-code format in Table 4 can be used for writing single or multiple registers.

TABLE 4. WRITE MULTIPLE REGISTERS (CODE 16)

BYTE #	DESCRIPTION
Byte 1	Slave Address
Byte 2	Function Code
Byte 3	MSB Register Address
Byte 4	LSB Register Address
Byte 5	MSB of Quantity
Byte 6	LSB of Quantity
Byte 7	Byte Count
Byte 8	MSB of First Data Word
Byte 9	LSB of First Data Word
Byte n	MSB of Last Data Word
Byte n+1	LSB of Last Data Word

The slave will reply with the slave address, function code, register address, and the quantity of registers written.

2.3.5 COMMAND INSTRUCTION (CODE 5)

Modbus Function Code 5 (Force Single Coil) is used to issue commands to the MPU-32 or FPU-32 slave. The format for the message is listed in Table 5 and the command code actions and corresponding coil number are listed in Table 6.

TABLE 5. COMMAND FORMAT (CODE 5)

HEX BYTE	DESCRIPTION
Byte 1	Slave Address
Byte 2	Function Code
Byte 3	MSB of Command Code
Byte 4	LSB of Command Code
Byte 5	Fixed at 0xff
Byte 6	Fixed at 0x00

TABLE 6. SUPPORTED COMMANDS

COMMAND CODE	COIL NUMBER	ACTION
0x0003	4	Reset Trips
0x0004	5	Set Real-Time Clock
0x0005	6	Clear Data-Logging Records
0x0006	7	Clear Trip Counters
0x0008	9	Clear Running Hours
0x0009	10	Emergency I^2t and Trip Reset
0x000C	13	Re-enable Temperature Protection

Except for a broadcast address, the slave will return the original packet to the master.

2.3.6 COMMAND INSTRUCTIONS USING REGISTER WRITES

For PLCs not supporting Function Code 5, commands can be issued using Write Single Register (Code 6) and Write Multiple Register (Code 16).

Commands are written to MPU-32 or FPU-32 register 6 (Modbus register 40007). Supported commands are listed in the COMMAND CODE column in Table 6.

When using the Write Multiple Registers function code, the write should be to the single Register 6. If multiple registers are written starting at Register 6, the first data element will be interpreted as the command code but no other registers will be written. If the command is successful, the slave will return a valid response message.

2.3.7 READ DEVICE IDENTIFICATION (CODE 43)

Modbus Function Code 43 (Read Device Identification) allows remote clients to read the identification and additional information describing the slave. It is a new function only supported by Modbus TCP and uses a different scheme for requests than those based upon Modbus RTU. In addition to the Modbus-TCP header, there is a byte for the function code, Modbus Encapsulated Interface (MEI) type, a Read Device Identification Code, and finally the initial Object ID.

The MEI type is fixed at 0x0E. The Read Device Identification Code can be:

1 – Stream Basic Device Objects
2 – Stream Regular Device Objects
4 – Access Individual Object.

Table 7 shows the format of the Read Device Identification request.

TABLE 7. READ DEVICE IDENTIFICATION (CODE 43)

BYTE #	DESCRIPTION
Byte 1	Slave Address
Byte 2	Function Code
Byte 3	MEI Type
Byte 4	Read Device ID Code
Byte 5	(Initial) Object ID

There are three classes of objects: Basic, Regular, and Extended. All Basic objects and several Regular objects are supported at this time. Extended objects are not supported. Basic objects are a subset of Regular objects and can be streamed as either. The boundaries for Basic objects are 0-2 and the boundaries for Regular objects are 0-4. The supported objects are listed in Table 8. Note that all objects are ASCII strings.

TABLE 8. DEVICE IDENTIFICATION OBJECTS

OBJECT ID	DESCRIPTION	CATEGORY
0x00	Vendor Name	**Basic**
0x01	Product Code	
0x02	Major/Minor Revision	
0x03	Vendor URL	**Regular**
0x04	Product Name	

The slave responds with the function code, MEI Type, and Read Device Identification Code as submitted by the master. It will also respond with the Conformity Level, the More Follows flag, Next Object ID, Number of Objects, and finally the Object(s).

The Conformity Level is fixed at 82, which indicates that Regular and Basic Objects are supported and can be streamed or individually accessed. More Follows is either 0 if there are no more objects to follow in subsequent packets or 0xFF if there are more objects to follow. For the MPU-32 and FPU-32, all objects can be streamed in a single packet; therefore More Follows will be 0. The Next Object ID similarly exists for streaming across multiple packets, and would contain the next object to be requested to continue the stream, but since this is not an issue in the MPU-32 and FPU-32, it will always contain 0. Finally the Number of Objects is the amount contained in this packet. The Response Packet is shown in Table 9.

TABLE 9. READ DEVICE IDENTIFICATION RESPONSE

BYTE #	DESCRIPTION
Byte 1	Slave Address
Byte 2	Function Code
Byte 3	MEI Type
Byte 4	Read Device ID Code
Byte 5	Conformity Level
Byte 6	More Follows
Byte 7	Next Object ID
Byte 8	Number of Objects
Byte 9	First Object ID
Byte 10	First Object Length
Byte 11+	First Object Data
Byte n	Last Object ID
Byte n+1	Last Object Length
Byte n+	Last Object Data

Each Object contains an Object ID, Object Length, and Object Value. Below are several examples of reading Device Identification Objects.

The following message will obtain all basic objects from slave 1, using Read Device Identification Code 1 to stream Basic Objects, and starting at object 0:

0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x05 | 0x01 | 0x2B | 0x0E | 0x01 | 0x00

The response from the slave might look as follows:

0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x1D | 0x01 | 0x2B | 0x0E | 0x01 | 0x52 | 0x00 | 0x00 | 0x03 | 0x00 | 0x07 | “S” | “t” | “a” | “r” | “t” | “c” | “o” | 0x01 | 0x04 | “P” | “3” | “0” | “1” | 0x02 | 0x04 | “1” | “.” | “4” | “0”

Note that the transmission automatically ends at the last basic object since that was what was requested. If the same request had been made, starting at Object 0, with Read Device Identification Code 2 (read Regular objects), all Basic and Regular Objects would be returned.

A single object, for example, the Product Name, can also be requested as follows, using Read Device Identification Code 4:

0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x05 | 0x01 | 0x2B | 0x0E | 0x04 | 0x04

The response from the slave might look as follows:

0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x10 | 0x01 | 0x2B | 0x0E | 0x04 | 0x52 | 0x00 | 0x00 | 0x01 | 0x04 | 0x06 | "M" | "P" | "U" | "-" | "3" | "2"

2.4 Error Responses

The MPU-32 and FPU-32 support the following exception responses:

- 01: Illegal Function – The function code (Byte 2 of the Modbus-RTU packet or Byte 8 of the entire Modbus-TCP message) is not supported
- 02: Illegal Data Address – All accesses to communication registers must be within the specified address range.
- 03: Illegal Data Value – This error code is returned if there is a data value outside the allowable value for the slave.

The exception message consists of the Modbus-TCP header, and the slave address followed by a retransmission of the original function code. The function code will have the most-significant bit set to indicate an error. The 8-bit byte following the function code is the exception response code.

In the case of the Read Device Identification function (code 43), the exception message consists of the Modbus-TCP header, the slave address, function code, then the MEI type, followed by the Exception Code.

2.5 Register Database

Appendix E in the MPU-32 and FPU-32 manual contains the Modbus Register Table. The table starts at register 0 (Modbus 40001) and each register is 16-bits wide. Types "long" and "float" are 32-bit values. For both long and float types, the low-order word is transmitted first followed by the high-order word. Word values have the high byte followed by the low byte. Float types are per the IEEE 754 Floating-Point Standard. All bytes of long and float types must be written using one message or an error will result. This does not apply for read commands.

2.6 Network Status and Indication

Network status is viewed in the *Metering* | *Network Status* menu. "Type" should indicate Modbus TCP and "Link" indicates either ACTIVE if there is at least one connection or TIMED OUT if there are no active connections. The fourth line contains the most recent error and is not necessarily an active or current error. The last error will continue to be displayed *even after it has been resolved*.

Communication status LED's are located on the rear panel of the slave. When Modbus TCP is selected, the green Module Status (MS) LED will be ON if it is correctly configured. A flashing green or red MS LED indicates a configuration error. The Network Status (NS) LED will indicate solid green when there is link activity, flashing green when there are no active connections and will flash red when any link has timed out. Both LED's are OFF when the *Network Type* is set to *None*.

3. DATA RECORDS

Event record information is located starting at MPU-32 or FPU-32 register 973.

Only one event record can be read at a time. Record data is for the record indicated by the Record Selector. To select a record, write the record number to Record Selector with the first message and then read the values in the record with a second message. Record Head points to the next available record. The last event record captured is at Record Head minus one.

The Record Selector must be in the range of 0 to 99. Values outside this range will select record 0.

4. NETWORK TIMEOUT

The MPU-32 or FPU-32 slave can be configured to trip or alarm on a network timeout using the *Setup | Hardware | Network Comms* menu. The *Net Trip Action* and *Net Alarm Action* set points set the actions to be taken when a timeout occurs. To prevent a timeout, a valid message, addressed to the slave, must be received at time intervals less than 5 seconds. If frequent communication is not required the *Net Trip Action* and *Net Alarm Action* should be disabled and connections closed after each transaction. The Network Status window will display that the link is timed out but this timeout indication can be ignored.

5. USER DEFINED REGISTERS

User-Defined Registers are used to assemble data in groups in order to minimize the amount of message requests. User-Defined Register values are entered using the *Setup | Hardware |Network Comms | User Register* menu, by using SE-Comm-RIS, or by using network communication messages.

The values entered are the MPU-32 or FPU-32 register numbers corresponding to the required parameter. The entered values are accessible from the menu or via communications starting at MPU-32 or FPU-32 register 1400 (Modbus 41401).

The data corresponding to these register values is retrieved by reading the values starting at registers 1432 (Modbus 41433). The format of the data is a function of the associated MPU-32 register type.

Typically, for PLC communications it is desirable to define data assemblies that are grouped by data type (float or integer). A single read can then access all required float values while another read can access the integer values.

For example, to access the three phase currents, enter 860, 861, 862, 863, 864, and 865, in User Register 0 to 5. The values are read by reading three float values starting at Modbus 41433. In a similar manner, the status and trip bits 0 to 32 can be read by entering 1096, 1097, 1104, 1105 in the next available register locations starting at User Register 6. These can be read starting at MPU-32 or FPU-32 Register 1438 (Modbus 41439).

6. SPECIFICATIONS

Interface	10Base-T, 100Base-T, Cat. 3, 4, 5, UTP, STP
Protocol	Modbus TCP
Baud Rate	10/100 Mbps.
Number of MPU-32's Connected	Up to 254 units
Number of Connections/Unit	Up to five
Bus length	100 m (328') per segment

APPENDIX A
MPU-32 & FPU-32 MODBUS TCP INTERFACE REVISION HISTORY

MANUAL RELEASE DATE	MANUAL REVISION
May 14, 2014	0-A-051414

MANUAL REVISION HISTORY

REVISION 0-A-051414

SECTION 2

Gateway added to configuration options setup procedure.