MODBUS/TCP INTEGRATION KIT FOR PAC PROJECT

Form 1676-110223—February 2011

OPTO 22

43044 Business Park Drive • Temecula • CA 92590-3614
Phone: 800-321-OPTO (6786) or 951-695-3000
Fax: 800-832-OPTO (6786) or 951-695-2712
www.opto22.com

Product Support Services
800-TEK-OPTO (835-6786) or 951-695-3080
Fax: 951-695-3017
Email: support@opto22.com
Web: support.opto22.com

Modbus/TCP Integration Kit for PAC Project
Form 1676-110223—February 2011

Table of Contents

1: Getting Started

OPTO 22

The Modbus/TCP Integration Kit for PAC Project™ (Part # PAC-INT-MBTCP) allows Opto 22 SNAP PAC controllers, using PAC Control™ to connect via Ethernet to Modbus/TCP devices.

The Integration Kit contains:

- A set of PAC Control *master* subroutines that are added to a strategy to enable an Opto 22 SNAP PAC controller to communicate as a Modbus/TCP master
- An example Modbus/TCP *slave* strategy containing the slave chart MB_Slave_TCP that is imported into a strategy to enable an Opto 22 controller to communicate as a Modbus/TCP slave

Both the master subroutines and the slave strategy transmit message strings as specified in the *Modbus Application Protocol Specification v1.1a* and *Modbus Messaging on TCP/IP Implementation Guide v1.0a*. Both guides are available on the web at http://Modbus-IDA.org.

The master subroutines and slave strategy transmit and receive messages using Modbus standard register, input and coil numbers. The desired information is stored or retrieved using PAC Control numeric tables.

This manual assumes the user fully understands how to use PAC Control, Modbus/TCP, and the Modbus device to be used.

This chapter includes the following topics:

Understanding Modbus Protocol

This toolkit assumes that you are knowledgeable about Modbus protocol addressing, and register, coil, and input numbering. Even for those who are experienced, we highly recommend reading the *Modicon Modbus Protocol Reference Guide*, which is available at this link:

http://www.modbus.org/docs/PI_MBUS_300.pdf

We especially recommend the following two sections of the Modbus guide:

- *Chapter 2: Data and Control Functions* of the Modicon guide explains the key subtleties of register, coil, and input numbering/naming as opposed to register, coil, and input addressing. This helps eliminate a common point of confusion, even for those who are experienced with Modbus protocol.
- *Appendix A: Exception Responses* of the Modicon guide discusses the possible exception codes a Modbus device can reply with and what the codes mean. This helps when diagnosing communication problems.

What is Required

Before including the subroutines in your strategy, you will need a PC with PAC Project 8.1a (Basic or Pro) or newer.

Modbus/TCP Functions Supported

The following Modbus/TCP function codes are supported by the master subroutines. These function codes are also supported by the slave chart.

Modbus/TCP Function Code	Name	PAC Control Subroutine
01	Read Coils	MB_01_Read_Coils
02	Read Discrete Inputs	MB_02_Read_Discrete_Inputs
03	Read Holding Registers	MB_03_Read_Holding Registers
04	Read Input Registers	MB_04_Read_Input_Registers
05	Write Single Coil	MB_05_Write_Single_Coil
06	Write Single Register	MB_06_Write_Single_Register
08	Diagnostics	MB_08_Diagnostics
15	Write Multiple Coils	MB_15_Write_Multiple_Coils
16	Write Multiple Registers	MB_16_Write_Multiple_Registers
22	Mask Write Register	MB_22 Mask_Write_Registers
23	Read Write Multiple Registers	MB_23 Read_Write_Holding _Registers

Data Types Supported in the Input and Holding Registers

The following data types are supported for input and holding registers. These data types are set by the Data Type subroutine parameter in the master subroutines and the nMB_Data_Type variable used in the slave chart:

Value	Data Type
0	16-bit unsigned (Modbus standard and default)
1	16-bit signed
2	Floating point (Uses two registers)
3	Floating point (Swapped. Uses two registers.)
4	32-bit signed (Uses 2 registers)
5	32-bit signed (Swapped. Uses 2 registers.)

NOTE: Most Modbus devices store 32-bit data values in two consecutive 16-bit registers. However, Opto 22 SNAP PAC controllers store 32-bit data values in individual table elements because tables support full 32-bit data values.

When accessing 32-bit data in most Modbus protocol devices, the data is stored in two consecutive 16-bit registers. Data that is 16 bits is sometimes referred to as a *word*, just as 8-bit data is referred to as a *byte*.

Modbus protocol messages treat each 16-bit register, or word, as 2 bytes with the high order byte coming before the low order byte. However, when device manufactures started supporting 32-bit data in standard Modbus messages, there was not a standard regarding the order of 16-bit registers (words) in the message when it comes to 32-bit data. Because of this, some Modbus protocol devices put the bytes of the high order register (word) first, and the low order register (word) second. Other devices do just the opposite.

In order to provide flexibility when communicating with both types of devices, the PAC Modbus/TCP toolkit supports *word swapping* when using 32-bit data types. We normally send the high-order word before the low order word in the message. If the data is not correct, it may be because the word order is backwards compared to your Modbus device. If this is the case, you can simply change the word order by selecting the appropriate swapped data type.

For example, if you are dealing with floating point data using Data Type 2, you could try Data Type 3 in order to swap the order of the 16-bit words in the message.

Important Note for Users Upgrading from Version 8.1 (or Earlier) of this Toolkit

In prior versions of the toolkit (versions 8.1 and earlier), all Register data was handled with either a float table (in the case of Read Holding Registers, Read Input Registers, Preset Multiple Registers, and Read Write Holding Registers) or a float variable (in the case of Preset Single Register) regardless of the value of the Data Type parameter.

Starting with version R8.2a of the toolkit, the table and variable data types now correctly match the Data Type parameter. This impacts how the subroutines are called by the strategy and how the strategy interacts with the data tables.

When calling the subroutine, you will now need to pass both float and integer tables (or float and integer variables). The subroutine will know which to use based on the value of the Data Type parameter. For example, if using integer data, you still have to pass the float table (or variable) even though it won't be used (and vice versa). The simplest thing to do is to just configure the extra table as having a length of 1 so it does not take up too much room in the controller.

You will also need to make sure that your strategy interacts with the correct data tables. When using Data Types 2 or 3, which are both float data types, your strategy will need to interact with the appropriate float table. When using Data Types 0, 1, 4, or 5, which are all integer data types, your strategy will need to interact with the appropriate integer table.

Installing the Integration Kit

To install the integration kit on your computer, unzip the zip file to your C: drive. The expanded files will be placed automatically in C:\ModbusTCPPAC.

Changing the Modbus Slave TCP Port

An Opto 22 SNAP PAC controller has a built-in Modbus/TCP slave capability, which only provides access to the I/O portions of the memory map (see Note below). The slave toolkit, on the other hand, provides access to both the strategy, portions of the memory map, and the I/O. Therefore, in order for the slave toolkit to work, you must disable the built-in Modbus slave functionality by changing the Modbus slave TCP port to a value of 0 (zero) as follows.

NOTE: I/O access only applies to rack-mounted controllers, such as the SNAP-PAC-R1 or R2.

1. Open PAC Manager.
2. In the PAC Manager main window, click the Inspect button.
3. In the Device Name field, type the IP address for the SNAP PAC controller (or choose it from the drop-down list).
4. Click Communications and choose Network Security.
5. Under PORTS, click the Value field for Modbus.
6. Change the value to 0 (zero). Click Apply.
7. Click Status Write.
8. Under Operation Commands, choose Send configuration to flash. Click Send Command.
9. Under Operation Commands again, choose Restart Device from powerup. Click Send Command.

For more details on using PAC Manager, see the *PAC Manager User's Guide*, form 1704.

Running the Example Strategies

The toolkit includes example strategies to demonstrate how to use the master subroutines and the slave chart in a PAC Control strategy. Before including the subroutines in your own strategy, we recommend that you first run the example strategy and pay special attention to the strategy logic and the configuration of variables.

To run the example master strategy, start PAC Control, and then open the strategy file named MBMasterTCP.idb. To run the example slave strategy, open the strategy named MBSlaveTCP.idb.

Using Communication Handles

Be sure to use a separate TCP communication handle for each chart that uses the Modbus/TCP subroutines.

In PAC Control, if two charts were to run simultaneously while sharing an open communication handle, each chart would be able to read and write data from the communication handle as if the other running chart didn't exist. Because these reads and writes are not synchronized between the charts, it is possible for one chart to read the other chart's data.

Using Data Addresses in Modbus

When used as Modbus terminology, the term *address* can be confusing. In Modbus, *addresses* are always zero-based, which means that the first address is 0, not 1.

For example:

- The coil known as *coil 1* in a progammable controller is addressed as coil 0000 in the data address field of a Modbus message.
- Holding register 40001 is addressed as register 0000 in the data address field of the message. The function code field already specifies a *holding register* operation. Therefore the *4XXXX* reference is implicit.

PAC Display Examples

Two PAC Display™ example projects are included in the ZIP file to show what can be done using PAC Display. While not necessary components of the toolkit, you can use the ModbusTCPMaster and ModbusTCPSlave projects to check PAC Control master and slave connections. For more information on using PAC Display, see the *PAC Display User's Guide*, form 1702.

2: Using the Master Subroutines

OPTO 22

This chapter includes the following topics:

Adding Master Subroutines

The Modbus master subroutines allow an Opto 22 controller to function as a Modbus master device. Each master subroutine in the integration kit supports one Modbus function code and can function independently of the other subroutines. Therefore, you need only use the subroutines for the Modbus functions that you require. For more information about subroutines, see the *PAC Control User's Guide*.

When you decide which subroutines you need, include them in your strategy as follows:

1. Start PAC Control in Configure Mode and open the strategy that you intend to use with the integration kit.

2. Select Configure⟶Subroutines Included to open the Subroutine Files dialog.

3. Click the Add button and use the browser to select each subroutine file (.ISB extension) you wish to include in your strategy from the folder C:\ModbusTCPPAC\Subs.

4. Click OK.

The subroutines appear in the Subroutines Included folder and are ready to be used in your strategy.

Data Tables Used By Master Subroutines

These are the names used for Modbus data tables in the example Master strategy:

- ntMB_Coils_0X (integer 32 table)
- ntMB_Inputs_1X (integer 32 table)
- ftMB_Holding_Registers_4X_Float (float table)
- ftMB_Input_Registers_3X_Float (float table)
- ntMB_Holding_Registers_4X_Int (integer 32 table)
- ntMB_Input_Register_3X_Int (integer 32 table)

You can name the tables however you want because the names of the tables are passed to the subroutines.

You may need to adjust the lengths of these tables to accommodate the amount of Modbus data and the register, coil, and input numbers expected to be accessed by the Modbus master device.

Use strategy logic to populate data in or retrieve data from these tables from the most recent Modbus message received.

NOTE: An integer table is used for data types 0, 1, 4, and 5. A float table is used for data types 2 and 3.

See "Important Note for Users Upgrading from Version 8.1 (or Earlier) of this Toolkit" on page 3.

Operation Mode Details for Master Subroutines

Version 8.1d of the Modbus/TCP PAC Control toolkit added a new feature called the Operation Mode. In the Master subroutines, this is implemented by an additional passed parameter called Master Register.

If you want to use the same register, coil, or input numbers in the master and slave, use a value of -1 for the Master Register parameter. This is how the original versions of this toolkit worked prior to adding the Operation Mode feature.

When the Master Register parameter is greater than or equal to 0, it designates the starting table index used in the Opto 22 SNAP PAC controller (the Master).

All subroutines that are passed a table to read or write Modbus data use the new parameter. The following subroutines do not support the new Operation Mode feature:

- MB_05_Write Single Coil
- MB_06_Write_ Single Register
- MB_22_ Mask Write _Registers

If you want to use different register, coil, or input numbers in the master and slave, then use the specific starting table index in the master as the Master Register parameter and specify the slave starting register, coil, or input number in the Slave Register parameter.

This method is useful when the data in the slave is offset to a high register, coil, or input number. In this case, it allows using much smaller tables in the master. This method can also be used when accessing data in multiple slaves and consolidating it into one set of tables in the master.

NOTE: The parameter formerly named Starting_Address has been renamed to Slave_Register.

Examples Using the Master Register Parameter

Case 1

Subroutines Related to Coils and Inputs

For Coils and Inputs, there is always a one-to-one correlation between the number of coils or inputs in the slave and the number of table elements used for the data in the master.

The Master Register parameter only affects the starting coil (or input) number used in the master data tables. This corresponds to the data table index number.

Example 1A: When the Master Register parameter is -1, the subroutine uses the value of the Start Coil (or Start Input) parameter as the starting Coil (or Input) number in the slave and also as the starting table index in the master.

Parameter	Value
Master Register	-1
Slave Register	19
Qty of Coils	3

Master Table Index		Slave Coil Number
19	←	19
20	←	20
21	←	21

Example 1B: When the Master Register parameter is greater than or equal to 0, the subroutine uses the value of the Start Coil (or Start Input) parameter as the starting Coil (or Input) number in the Slave and it uses the value of the Master Register parameter as the starting table index in the master.

Parameter	Value
Master Register	1
Slave Register	101
Qty of Inputs	3

Master Table Index		Slave input Number
1	←	101
2	←	102
3	←	103

Case 2

Subroutines related to Input Registers and Holding Registers when the Data Type Parameter is 0 or 1 (both 16-bit data types)

For Input Registers and Holding Registers, when using 16-bit data types, there is always a one-to-one correlation between the number or registers in the slave and the number of table elements used for the data in the master.

The Master Register parameter only affects the starting register number used in the master data tables. This corresponds to the data table index number.

Example 2A: When the Master Register parameter is -1, the subroutine uses the value of the Start Register parameter as the starting register number in the slave and also as the starting table index in the master.

Parameter	Value	Comment
Master Register	-1	
Slave Register	7001	
Qty of H Registers	3	
Data Type	0 or 1	data type is 16-bit

Master Table Index		Slave Register Number
7001	→	7001
7002	→	7002
7003	→	7003

Example 2B: When the Master Register parameter is greater than or equal to 0, the subroutine uses the value of the Start Register parameter as the starting register number in the Slave and it uses the value of the Master Register parameter as the starting table index in the master.

Parameter	Value	Comment
Master Register	99	
Slave Register	7001	
Qty of I Registers	3	
Data Type	0 or 1	data type is 16-bit

Master Table Index		Slave Register Number
99	←	7001
100	←	7002
101	←	7003

Case 3

Subroutines related to Input Registers and Holding Registers when the Data Type parameter is 2, 3, 4, or 5 (all 32-bit data types)

NOTE: Opto 22 SNAP- PAC tables support 32-bit data; tables start with index 0. A table with length of 10 has indexes 0 through 9.

For Input Registers and Holding Registers, when using 32-bit data types, there is always a *two*-to-one correlation between the number of (16-bit) registers in the slave and the number of (32-bit) table elements used for the data in the master. This is because most slave devices store data in 16-bit registers. Consequently, 32-bit data in these devices are stored in two consecutive 16-bit registers.

For 32-bit data, the Master Register parameter affects not only the starting register number used in the master data tables, but also the quantity of registers accessed in the slave.

Example 3A: When the Master Register parameter is -1, the subroutine uses the value of the Start Register parameter as the starting register number in the slave and also as the starting table index in the master.

In addition, the Qty of H Registers (or Qty of I Registers) parameter determines the quantity of 16-bit registers read from or written to the slave. The number of 32-bit registers in the master will be half the quantity of this parameter.

The Qty of H Registers (or Qty of I Registers) parameter *must be an even number.*

The Start Register parameter *must be an odd number.*

The Start Register value will determine the first table index used for the 32-bit data in the master. Additional 32-bit values will be put into subsequent *odd indexes* of the table so that the table indexes in the master will match the first register number for each set of two consecutive 16-bit registers in the slave. All even table indexes are unused.

Parameter	Value	Comment
Master Register	-1	
Slave Register	7001	must be an odd number
Qty of I Registers	4	must be an even number
Data Type	2,3,4 or 5	data type is 32-bit

Master Table Index		Slave Register Number
7001	←	7001
		7002
7003	←	7003
		7004

Example 3B: When the Master Register parameter is greater than or equal to 0, the subroutine uses the value of the Start Register parameter as the starting register number in the Slave and it uses the value of the Master Register parameter as the starting table index in the master.

In addition, the Qty of H Registers (or Qty of I Registers) parameter determines the quantity of 32-bit data values you want to read from or write to the slave and the number of 32-bit table elements used in the master. However, the Modbus protocol has no mechanism for identifying 32-bit data in the messages, so the subroutine requests twice the quantity of registers because the data in the slave is assumed to be 16-bit.

The Master Register value will determine the first table index used for the 32-bit data in the master. Additional 32-bit values will be put into consecutive indexes of the table so that there will not be any gaps in the table.

Parameter	Value	Comment
Master Register	8	
Slave Register	7001	must be an odd number
Qty of H Registers	3	Qty of 32-bit values
Data Type	2,3,4 or 5	data type is 32-bit

Master Table Index		Slave Register Number
8	→	7001
		7002
9	→	7003
		7004
10	→	7005
		7006

Configuration of Subroutines

The following tables list the parameters for each function code and describe the type of data for each parameter:

Function Code Subroutine	Page
MB 01 Read Coils	page 15
MB 02 Read Discrete Inputs	page 16
MB 03 Read Holding Registers	page 17
MB 04 Read Input Registers	page 18
MB 05 Write Single Coil	page 19
MB 06 Write Single Register	page 20
MB 08 Diagnostics	page 21
MB 15 Write Multiple Coils	page 23
MB 16 Write Multiple Registers	page 24
MB 22 Mask Write Register	page 25
MB 23 Read Write Holding Registers	page 26

MB 01 Read Coils

Name	Description
Slave Address	Integer 32 Variable (1–255)
Start Coil	Integer 32 Variable (1–65536)
Quantity of Coils	Integer 32 Variable (1–2000)
Identifier	Integer 32 Variable (Used for transaction pairing)
Com Handle	Communication Handle
Wait Time (s)	Float Variable (Wait time in seconds for slave to respond)
MB Coils 0X	Integer 32 Table (The subroutine will support coils 1–65535)
Master Register	Integer 32 Variable < 0 = default operation mode -1 = When Master Register is -1, it uses the value of Start Coil parameter as the starting Coil number in the slave and also as the starting table index in the PAC (master). >=0 = When Master Register is greater than or equal to 0, it uses the value of Start Coil as the starting Coil number in the Slave and it uses the value of Master Register as the starting table index in the PAC (master).
Return Status	String No Session = Not able to open session Timeout = No response within time limit Too Many Characters = More then 260 characters Identifier Mismatch = Send and receive identifier do not match Function and Exception code = Error from PDU Invalid Table Length = Used an index greater than the number of elements in the table. OK = Success
Put Status In	Integer 32 Variable 0 = Success -67 = Out of memory -69 = Null object error

MB 02 Read Discrete Inputs

Name	Description
Slave Address	Integer 32 Variable (1–255)
Start Input	Integer 32 Variable (1–65536)
Quantity of Inputs	Integer 32 Variable (1–2000)
Identifier	Integer 32 Variable (Used for transaction pairing)
Com Handle	Communication Handle
Wait Time(s)	Float Variable (Wait time in seconds for slave to respond)
MB Inputs 1X	Integer 32 Table (The subroutine will support Inputs 1–65535)
Master Register	Integer 32 Variable < 0 = default operation mode -1 = When Master Register is -1, it uses the value of Start Input parameter as the starting Input number in the slave and also as the starting table index in the PAC (master). >=0 = When Master Register is greater than or equal to 0, it uses the value of Start Input as the starting Input number in the Slave and it uses the value of Master Register as the starting table index in the PAC (master).
Return Status	String No Session = Not able to open session Timeout = No response within time limit Too Many Characters = More then 260 characters Identifier Mismatch = Send and receive identifier do not match Function and Exception code = Error from PDU Invalid Table Length = Used an index greater than the number of elements in the table. OK = Success
Put Status In	Integer 32 Variable 0 = Success -67 = Out of memory -69 = Null object error

MB 03 Read Holding Registers

Name	Description
Slave Address	Integer 32 Variable (1–255)
Start Register	Integer 32 Variable (1–65536)
Qty of H Registers	Integer 32 Variable (1–125)
Identifier	Integer 32 Variable (Used for transaction pairing)
Com Handle	Communication Handle
Wait Time(s)	Float Variable (Wait time in seconds for slave to respond)
MB H Reg4X Int	Integer 32 Table PAC Control strategy table used by the master to store the Holding Register data when using integer data (data types 0, 1, 4, 5). MB H Register 4X - MB H Reg4X FloatFloat tablePAC Control strategy table used by the master to store the Holding Register data when using float data (data types 2, 3)
MB H Register 4X	Float Table (The subroutine will support registers 1-65535)
Data Type	Integer 32 Variable 0 = 16-bit unsigned (Modbus standard and default) 1 = 16-bit signed 2 = Floating Pt. 3 = Floating Pt. (swapped) 4 = 32-bit signed 5 = 32-bit signed (swapped)
Master Register	Integer 32 Variable < 0 = default operation mode -1 = When the Master Register parameter is -1 and the Data Type parameter is 0 or 1 (16-bit data), the subroutine uses the value of the Start Register parameter as the starting register number in the slave and also as the starting table index in the master. >=0 = When the Master Register parameter is greater than or equal to 0 and the Data Type parameter is 0 or 1 (16-bit data), the subroutine uses the value of the Start Register parameter as the starting register number in the Slave and it uses the value of the Master Register parameter as the starting table index in the master.
Return Status	String No Session = Not able to open session Timeout = No response within time limit Too Many Characters = More then 260 characters Identifier Mismatch = Send and receive identifier do not match Function and Exception code = Error from PDU Invalid Table Length = Used an index greater than the number of elements in the table. OK = Success
Put Status In	Integer 32 Variable 0 = Success -67 = Out of memory -69 = Null object error

MB 04 Read Input Registers

Name	Description
Slave Address	Integer 32 Variable (1–255)
Start Register	Integer 32 Variable (1–65536)
Qty of I Registers	Integer 32 Variable (1–125)
Identifier	Integer 32 Variable (Used for transaction pairing)
Com Handle	Communication Handle
Wait Time(s)	Float Variable (Wait time in seconds for slave to respond)
MB I Reg3X Int	Integer 32 Table
MB I Reg3X Float	Float Table (The subroutine will support registers 1-65535) PAC Control strategy table used by the master to store the Input Register data when using float data (data types 2, 3)Integer 32 Table
Data Type	Integer 32 Variable 0 = 16-bit unsigned (Modbus standard and default) 1 = 16-bit signed 2 = Floating Pt. 3 = Floating Pt. (swapped) 4 = 32-bit signed 5 = 32-bit signed (swapped)
Master Register	Integer 32 Variable < 0 = default operation mode -1 = When the Master Register parameter is -1 and the Data Type parameter is 0 or 1 (16-bit data), the subroutine uses the value of the Start Register parameter as the starting register number in the slave and also as the starting table index in the master. >=0 = When the Master Register parameter is greater than or equal to 0 and the Data Type parameter is 0 or 1 (16-bit data), the subroutine uses the value of the Start Register parameter as the starting register number in the Slave and it uses the value of the Master Register parameter as the starting table index in the master.
Return Status	String No Session = Not able to open session Timeout = No response within time limit Too Many Characters = More then 260 characters Identifier Mismatch = Send and receive identifier do not match Function and Exception code = Error from PDU Invalid Table Length = Used an index greater than the number of elements in the table. OK = Success
Put Status In	Integer 32 Variable 0 = Success -67 = Out of memory -69 = Null object error

MB 05 Write Single Coil

Name	Description
Slave Address	Integer 32 Variable (1–255)
Coil	Integer 32 Variable (1–65536)
Coil State	Integer 32 Variable (0 = OFF 1 = ON)
Identifier	Integer 32 Variable (Used for transaction pairing)
Com Handle	Communication Handle
Wait Time(s)	Float Variable (Wait time in seconds for slave to respond)
Return Status	String No Session = Not able to open session Timeout = No response within time limit Too Many Characters = More then 260 characters Identifier Mismatch = Send and receive identifier do not match Function and Exception code = Error from PDU OK = Success
Put Status In	Integer 32 Variable 0 = Success -67 = Out of memory -69 = Null object error

MB 06 Write Single Register

Name	Description
Slave Address	Integer 32 Variable (1–255)
Register	Integer 32 Variable (1–65536)
Reg Value Float	Float Variable Value to write to the register when using float data (data types 2, 3)
Reg Value Int	Integer 32 Variable Value to write to the register when using integer data (data types 0, 1, 4, 5)
Identifier	Integer 32 Variable (Used for transaction pairing)
Com Handle	Communication Handle
Wait Time(s)	Float Variable (Wait time in seconds for slave to respond)
Data Type	Integer 32 Variable 0 = 16-bit unsigned (Modbus standard and default) 1 = 16-bit signed 2 = Floating Pt. 3 = Floating Pt. (swapped) 4 = 32-bit signed 5 = 32-bit signed (swapped)
Return Status	String No Session = Not able to open session Timeout = No response within time limit Too Many Characters = More then 260 characters Identifier Mismatch = Send and receive identifier do not match Function and Exception code = Error from PDU OK = Success
Put Status In	Integer 32 Variable 0 = Success -67 = Out of memory -69 = Null object error

MB 08 Diagnostics

Name	Description
Slave Address	Integer 32 Variable (1–255)
Sub-Function	Integer 32 Variable (0–65535)
Data(Send)	Integer 32 Variable (0–65535)
Data(Rec)	Integer 32 Variable (see "Function 08 Diagnostics Sub-function Codes" on page 22)
Identifier	Integer 32 Variable (Used for transaction pairing)
Com Handle	Communication Handle
Wait Time (s)	Float Variable (Wait time in seconds for slave to respond)
Return Status	String No Session = Not able to open session Timeout = No response within time limit Too Many Characters = More then 260 characters Identifier Mismatch = Send and receive identifier do not match Sub-function Reserved = Not supported code Response Mismatch = Send and Receive packet mismatch Function and Exception code = Error from PDU OK = Success
Put Status In	Integer 32 Variable 0 = Success -67 = Out of memory -69 = Null object error

Function 08 Diagnostics Sub-function Codes

Sub-Function	Name	Data (Send)	Data(Rec)
0	Return Query Data	Any	0
1	Restart Communication Option	0 or 1 = Clear Log	0
2	Return Diagnostic Register	0	Register data
3	Change ASCII Input Delimiter	Decimal Value of Character	0
4	Force Listen Only Mode	0	0
5	Reserved		
6	Reserved		
7	Reserved		
8	Reserved		
9	Reserved		
10	Clear Counters and Diagnostic Register	0	0
11	Return Bus Message Count	0	Message Ct.
12	Return Bus Communication Error Count	0	Error Ct.
13	Return Bus Exception Error Count	0	Error Ct.
14	Return Slave Message Count	0	Message Ct.
15	Return Slave No Response Count	0	No Response Ct.
16	Return Slave NAK Count	0	NAK Ct.
17	Return Slave Busy Count	0	Busy Ct.
18	Return Bus Character Overrun Count	0	Overrun Ct.
19	Reserved		
20	Clear Overrun Counter And Flag	0	0
21	Reserved	0	

MB 15 Write Multiple Coils

Name	Description
Slave Address	Integer 32 Variable (1–255)
Start Coil	Integer 32 Variable (1–65536)
Quantity of Coils	Integer 32 Variable (1–1968)
MB Coil 0X	Integer 32 Table (The subroutine will support coils 1–65535)
Identifier	Integer 32 Variable (Used for transaction pairing)
Com Handle	Communication Handle
Wait Time(s)	Float Variable (Wait time in seconds for slave to respond)
Master Register	Integer 32 Variable < 0 = default operation mode -1 =– When Master Register is -1, it uses the value of Start Coil parameter as the starting Coil number in the slave and also as the starting table index in the PAC (master). >=0 = When Master Register is greater than or equal to 0, it uses the value of Start Coil as the starting Coil number in the Slave and it uses the value of Master Register as the starting table index in the PAC (master).
Return Status	String No Session = Not able to open session Timeout = No response within time limit Too Many Characters = More then 260 characters Identifier Mismatch = Send and receive identifier do not match Function and Exception code = Error from PDU Invalid Table Length = Used an index greater than the number of elements in the table. OK = Success
Put Status In	Integer 32 Variable 0 = Success -67 = Out of memory -69 = Null object error

MB 16 Write Multiple Registers

Name	Description
Slave Address	Integer 32 Variable (1–255)
Start Register	Integer 32 Variable (1–65536)
Qty of Registers	Integer 32 Variable (1–120)
MB H Reg4X Int	Integer 32 Table PAC Control strategy table used by the master to store the Holding Register data when using integer data (data types 0, 1, 4, 5).
MB H Reg4X Float	Float table PAC Control strategy table used by the master to store the Holding Register data when using float data (data types 2, 3)
Identifier	Integer 32 Variable (Used for transaction pairing)
Com Handle	Communication Handle
Wait Time(s)	Float Variable (Wait time in seconds for slave to respond)
Data Type	Integer 32 Variable 0 = 16-bit unsigned (Modbus standard and default) 1 = 16-bit signed 2 = Floating Pt. 3 = Floating Pt. (swapped) 4 = 32-bit signed 5 = 32-bit signed (swapped)
Master Register	Integer 32 Variable < 0 = default operation mode -1 = When the Master Register parameter is -1 and the Data Type parameter is 0 or 1 (16-bit data), the subroutine uses the value of the Start Register parameter as the starting register number in the slave and also as the starting table index in the master. >=0 = When the Master Register parameter is greater than or equal to 0 and the Data Type parameter is 0 or 1 (16-bit data), the subroutine uses the value of the Start Register parameter as the starting register number in the Slave and it uses the value of the Master Register parameter as the starting table index in the master.
Return Status	String No Session = Not able to open session Timeout = No response within time limit Too Many Characters = More then 260 characters Identifier Mismatch = Send and receive identifier do not match Function and Exception code = Error from PDU Invalid Table Length = Used an index greater than the number of elements in the table. OK = Success
Put Status In	Integer 32 Variable 0 = Success -67 = Out of memory -69 = Null object error

MB 22 Mask Write Register

Name	Description
Slave Address	Integer 32 Variable (1–255)
Ref Address	Integer 32 Variable (1–65536)
AND Mask	Integer 32 Variable (0 - 65535)
OR Mask	Integer 32 Variable (0 - 65535)
Identifier Integer 32 Variable	(Used for transaction pairing)
Com Handle	Communication Handle
Wait Time(s)	Float Variable (Wait time in seconds for slave to respond)
Return Status	String No Session = Not able to open session Timeout = No response within time limit Too Many Characters = More then 260 characters Identifier Mismatch = Send and receive identifier do not match Function and Exception code = Error from PDU OK = Success
Put Status In	Integer 32 Variable 0 = Success -67 = Out of memory -69 = Null object error

MB 23 Read Write Holding Registers

NOTE: This subroutine no longer has the Wait Time(s) passed parameter. Wait time is set in block 1 Init of the subroutine. Default = 4 seconds.

Name	Description
Slave Address	Integer 32 Variable (1 - 255)
R Start Register	Integer 32 Variable (1 - 65536)
R Qt H Registers	Integer 32 Variable (1 - 125)
W Start Register	Integer 32 Variable (1 - 65536)
W Qt H Registers	Integer 32 Variable (1 - 125)
Identifier	Integer 32 Variable (Used for transaction pairing)
Com Handle	Communication Handle
MB H Reg4X Int	Integer 32 Table PAC Control strategy table used by the master to store the Holding Register data when using integer data (data types 0, 1, 4, 5).
MB H Reg4X Float	Float Table (The subroutine will support registers 1 - 65535) PAC Control strategy table used by the master to store the Holding Register data when using float data (data types 2, 3)
Data Type	Integer 32 Variable 0 = 16bit unsigned (Modbus standard and default) 1 = 16bit signed 2 = Floating Pt. 3 = Floating Pt. (swapped) 4 = 32bit signed 5 = 32bit signed (swapped)
Master Register	Integer 32 Variable < 0 = default operation mode -1 = When the Master Register parameter is -1 and the Data Type parameter is 0 or 1 (16-bit data), the subroutine uses the value of the Start Register parameter as the starting register number in the slave and also as the starting table index in the master. >=0 = When the Master Register parameter is greater than or equal to 0 and the Data Type parameter is 0 or 1(16-bit data), the subroutine uses the value of the Start Register parameter as the starting register number in the Slave and it uses the value of the Master Register parameter as the starting table index in the master.
Return Status	String No Session = Not able to open session Timeout = No response within time limit Too Many Characters = More then 260 characters Identifier Mismatch = Send and receive identifier do not match Function and Exception code = Error from PDU Invalid Table Length = Used an index greater than the number of elements in the table. OK = Success
Put Status In	Integer 32 Variable 0 = Success -67 = Out of memory -69 = Null object error

3: Using the Slave Chart

OPTO 22

This chapter includes the following topics:

Initialization

Before starting the Slave chart, there are a number of parameters that must be initialized. These include:

nMB_Number_of_Masters_Supported

See "Number of Masters Supported" on page 28.

nMB_Data_Type

See "Data Types Supported in the Input and Holding Registers" on page 3.

nSlave_Register_Mode

See "Operation Mode Details for Slave Chart" on page 28.

nMB_Slave_Address

Modbus Slave Address. The default is set for address 1.

Number of Masters Supported

This Slave chart will support 1 to 4 Modbus masters. Set the variable (nMB_Number_of_Masters_Supported) to the number of Modbus masters this slave chart will support. The default is set to 1.

Data Tables Used By Slave Chart

- ntMB_Coils_0X (integer 32 table)
- ntMB_Inputs_1X (integer 32 table)
- ftMB_Holding_Registers_4X_Float (float table)
- ftMB_Input_Registers_3X_Float (float table)
- ntMB_Holding_Register_4X_Int (integer 32 table)
- ntMB_Input_Register_3X_Int (integer 32 table)

NOTE: An integer table is used for data types 0, 1, 4, and 5. A float table is used for data types 2 and 3.

You may need to adjust the lengths of these tables to accommodate the amount of Modbus data and the register, coil, and input numbers expected to be accessed by the Modbus master device.

Use strategy logic to populate data in or retrieve data from these tables.

Operation Mode Details for Slave Chart

Version 8.1d of the Modbus TCP PAC Control toolkit added a new feature called the Operation Mode. In the Slave chart, this is implemented by a new variable named Slave_Register_Mode.

The Slave_Register_Mode variable only applies when using 32-bit data types (DataType is 2, 3, 4, or 5) with Input and Holding Registers. When accessing coils or inputs, or when using 16-bit data types, the Slave_Register_Mode variable has no effect; data is accessed in consecutive indexes.

The Slave_Register_Mode variable is treated as a flag that can either be true or false. In PAC controllers, False is defined as the value 0, and True is defined as any non-zero value.

If the value of Slave_Register_Mode is True when accessing Input or Holding Registers using 32-bit data types (DataType is 2, 3, 4, or 5), data will be accessed in consecutive table indexes. However, if it is False, data will be in every other odd index of the data table, which requires that the Master use a starting register number that is odd.

If you want to maintain compatibility with previous versions of the toolkit, before the Operation Mode feature is added, set the Slave_Register_Mode variable to a value of False (0).

Examples

Coils & Inputs.

Example 1A:

Master		Slave Parameter	Value
		Slave_Register_Mode	(any value)
Starting Coil	19		
Qty of Coils	3		

Master Coil Number		Slave Table Index
19	→	19
20	→	20
21	→	21

Example 1B:

Master		Slave Parameter	Value
		Slave_Register_Mode	(any value)
Starting Input	102		
Qty of Inputs	3		

Master Input Number		Slave Table Index
102	←	102
103	←	103
104	←	104

Input & Holding Registers with 16-bit Data Types

Example 2A:

Master		Slave Parameter	Value
		Slave_Register_Mode	(any value)
Starting Holding Register	7001		
Qty of Registers	3		
		Data Type	0 or 1

Master Register Number		Slave Table Index
7001	→	7001
7002	→	7002
7003	→	7003

Example 2B:

Master		Slave Parameter	Value
		Slave_Register_Mode	(any value)
Starting Holding Register	7001		
Qty of Registers	3		
		Data Type	0 or 1

Example 2B (continued)

Master Register Number		Slave Table Index
7001	←	7001
7002	←	7002
7003	←	7003

Input & Holding Registers with 32-bit Data Types

Example 3A:

Master		Slave Parameter	Value
		Slave_Register_Mode	False (0)
Starting Input Register	7001	*must be an odd number*	
Qty of Registers	4	*must be an even number*	
		Data Type	2, 3, 4, or 5

Master Register Number		Slave Table Index
7001	←	7001
7002		
7003	←	7003
7004		

Example 3B:

Master		Slave Parameter	Value
		Slave_Register_Mode	True (1)
Starting Holding Register	8		
Qty of Registers	6		
		Data Type	2, 3, 4, or 5

Master Register Number		Slave Table Index
8	→	8
9		
10	→	9
11		
12	→	10
13		

Importing the Slave Chart

The MBSlaveTCP slave chart allows an Opto 22 controller to function as a Modbus slave device. Unlike the subroutines used in master strategies, which are run as needed, the MBSlaveTCP chart is started in the Powerup chart and must run all the time. After the chart is started, it continuously monitors port 502 for Modbus traffic.

To copy the Modbus Slave chart to your strategy, you must export the chart MBSlaveTCP as a PAC Control chart export file (.cxf file) and then import it into your strategy. For more information, see Chapter 8 of form 1700, the *PAC Control User's Guide*.

Start the Modbus Slave chart in the Powerup chart of your strategy.

4: Troubleshooting

OPTO 22

This chapter includes the following topics:

Addressing Problems

Controller and Modbus device will not connect

If the Master Subroutines are being used:

Verify the status of the Open Outgoing Communication command by inspecting the nModbus_Port_Status variable located in the Modbus command subroutine being executed. If it was successful, the value should be 0. Any other value indicates a problem. Please see form 1701, the *PAC Control Command Reference* for the definitions of error codes for the Open Outgoing Communication command.

If the Slave Chart is being used:

Verify the status of the Accept Incoming Communication Command by inspecting the ntModbus_Port_Status table. If it was successful the value should be >= 0. Any value < 0 indicates a problem. If the variable, nMB_Number_of_Masters_Supported, is set to 1 then index 1 of the table is the status. If the variable, nMB_Number_of_Masters_Supported, is set to 4 the status of the connection will be in index 1, 2, 3 or 4. When multiple masters are enabled the Slave will try to accept incoming communication on each Communication Handle. The first to connect depends on which Communication Handle is accepting incoming communication at that time. Please see form 1701, the *PAC Control Command Reference* for the definitions of error codes for the command.

The controller is receiving data from the Modbus device but the data is not correct

- Verify that the Data Type configured in the controller matches the Modbus device.
- If the Data Type for the controller and Modbus device match but you are still getting incorrect data, try assigning other Data Types on the controller to resolve the issue. For example, when using 32-bit data types, it may be necessary to use the appropriate “swapped” version of that data type.

Communication Problems When Using the Slave Chart

Use the following variables to help diagnose problems with Modbus communication when using the slave chart:

nChart_Status

This is the status of starting the slave chart.

nException_Code

This is the exception code to the master device returned by the slave. A value of 0 indicates no exception code.

sLast_Return_Status

This is the status of the last transmitted Modbus packet from the slave.

It will return OK if the transmit was successful.

ntReceive_Table

This table contains the characters that were received by the slave chart from the most recent Modbus message received.

sReturn_Status

This indicates the status of the received and transmitted Modbus packets.

It will return OK or several error messages. Examples of the error messages are CRC Mismatch and Wrong Slave Address.

Problems When Using the Master Subroutines

Use the following guidelines to help diagnose problems when using the master subroutines:

- Review the status of the communication handle variable to verify that it is open.
- Review the value of the variable used for the Return Status parameter. See “Configuration of Subroutines” on page 13.
- Review the value of the variable used for the Put Status In parameter.

Modbus Exception Codes

For a list of the Modbus exception codes and their meanings, see *Modicon Modbus Protocol Reference Guide*, which is available at this link:

http://www.modbus.org/docs/PI_MBUS_300.pdf