USING MODBUS DEVICES WITH OPTO 22 PRODUCTS

This document introduces the Modbus® protocol and gives you basic information for using it to communicate between Opto 22 products and other devices. Use this document together with the Modicon Modbus Reference Guide, your Opto 22 product manual, and other manufacturers' product manuals to integrate devices and systems.

About Modbus

Modbus is an open protocol widely used in automation and other industries. Published by Modicon in 1979, the protocol provides a messaging structure for master/slave communications between intelligent devices.

Originally developed for Modicon's PLCs, Modbus can now be found in valves, energy meters, variable frequency drives (VFDs), temperature probes, and all kinds of other devices. The Modbus Organization, a trade organization, provides protocol information and documentation.

Although it was designed for a serial network, Modbus has since been adapted to work over Ethernet as well. Over Ethernet the protocol is called Modbus/TCP.

Modbus Support in Opto 22 Products

Native Support

Many Opto 22 devices can communicate using Modbus/TCP and/or Modbus, often in addition to other protocols. For example:

- *groov* View in ***groov* EPIC**, the ***groov* Box,** and ***groov* Server for Windows** acts as a Modbus master over Ethernet to slave devices, providing data and control from a mobile operator interface for Modbus/TCP PLCs and devices.
- All Opto 22 **controllers and brains based on Ethernet and the OptoMMP protocol**—including all *groov* EPIC processors, SNAP PAC controllers, and Ethernet brains—have built-in Modbus/TCP slave capability.
- **OptoEMU Sensor** energy monitoring units can act as a Modbus master to slave devices using Modbus/TCP or Modbus serial (RTU or ASCII). OptoEMU Sensors can also act as Modbus/TCP slave devices.
- The **SNAP-B3000-MODBUS** analog/digital brain talks only Modbus serial.

Integration Kits

In addition to these products that natively communicate using Modbus, you can use free integration kits to communicate with other Opto 22 products:

- **PAC Project controllers**—One integration kit handles both Modbus/TCP and Modbus serial (ASCII or RTU) for PAC Control. This integration kit works with all controllers that use PAC Project, including groov EPIC and SNAP PAC controllers.
- **ioProject controllers**—Two kits are available for legacy ioControl, Modbus/TCP or Modbus serial.
- **FactoryFloor controllers**—The integration kit supports Modbus ASCII or RTU (no Modbus/TCP).

Each integration kit includes a single subroutine for your PAC Control, ioControl, or OptoControl strategy to make the controller a Modbus slave, plus several subroutines to make it a Modbus master (one for each supported Modbus master function code). To use a kit, you add code in your control strategy to call the subroutine and pass in the Modbus registers you want. The subroutine builds the appropriate Modbus messages to send and receive data.

If you're using the master subroutines, you'll notice that these subroutines are named by the Modbus command name and number that they execute. You call the command by setting a variable of that name to true. For example, to manipulate a digital output you would use the Coil Modbus command 01. So you would run the subroutine MB_01_Read_Coils.

Each subroutine has a table or tables associated with it. Which table is used depends on the data type you've set in the subroutine command parameter. The subroutine builds the Modbus message, sends it, receives a response, parses the response, and places the data in the proper table for you to use in your strategy.

All integration kits include documentation with more details for the specific kit. For the PAC Control kit, documentation is in the text blocks inside each subroutine and example strategy flowchart.

NOTE: Most Modbus devices store 32-bit data values in two consecutive 16-bit registers. However, PAC Project controllers store 32-bit data values in individual table elements, because these tables support full 32-bit data values. Don't be confused by the difference between reading two consecutive registers in the slave device and using a single table element in PAC Control.

Setting up Ports

To use a Modbus/TCP integration kit, you must change the Modbus port as follows:

***groov* EPIC processors:** EPIC uses ports 502 and 8502 for the built-in slave functionality, so do not use either of these port numbers.

- To use an EPIC as a Modbus master, change the Modbus port in the integration kit. Then use *groov* Manage to add a rule to open that port on the EPIC (Home > Security > Firewall > Add Rule).
- To use an EPIC as a slave, change the Modbus port both in the integration kit and on the Modbus master.

SNAP PAC controllers: Using PAC Manager, change the Modbus port on the controller. By default SNAP PAC controllers and brains open port 502 for the built-in Modbus/TCP slave functionality. However, the Modbus/TCP integration kit won't work unless you change the built-in Modbus slave port from 502 to 0. See the PAC Manager User's Guide (form 1704) for instructions.

Getting Started with Modbus/TCP

A little preparation now will save you a lot of time and trouble later.

1. First, download the Modicon Modbus Protocol Reference Guide, a short guide with key information on the protocol and its commands:
   http://www.modbus.org/docs/PI_MBUS_300.pdf
   If you're using Modbus/TCP, also download this guide:
   http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf
2. Skim these guides to familiarize yourself with the protocol and (for Modbus/TCP) how it's implemented over Ethernet. Pay attention to command numbers and exception response codes. (See "Common Commands" on page 5 for the commands used with most Opto 22 products.)
3. Read "Modbus Integration Issues" on page 3 so you'll understand the integration issues to watch out for while using Modbus.
4. Download Modscan32 utility software from Win-TECH: http://www.win-tech.com/html/modscan32.htm

You may also find the Technical Resources section on the Modbus Organization website useful:
http://www.modbus.org/tech.php

MODBUS INTEGRATION ISSUES

Because it was designed for Modicon's PLCs in the late 1970s, the Modbus protocol as published has limitations based on capabilities at that time. Manufacturers have worked around those limitations in various ways.

For example, the protocol was based on 16-bit unsigned integers, but now 32 bits (or 64) are often required to handle data as floats or signed integers. The way manufacturers deal with this situation varies, even though they're using the same protocol.

Three areas typically can cause trouble when integrating Modbus devices:

- Serial network wiring and setup (If you're using Modbus/TCP, skip this section.)
- Data type and word order
- Command and register numbering

Serial Network Wiring and Setup

If you're using Modbus/TCP, skip this section.

If you're using Modbus over a serial network, all devices must be set up with the same wiring and communication parameters in order to communicate.

Look in each device's user guide to see the choices specified for that device and how to change them.

Wiring choices include:

- RS-232, with or without flow control
- RS-485/422 four-wire
- RS-485 two-wire

An RS-485/422 serial network must also be properly terminated and biased. If a device is physically the first or last one on the serial network, it should be set as terminated. Any one device in the network (typically the host or master device) should be set as biased. Check your device's user guide to determine how to set termination and biasing for that device.

Parameters include:

- Baud rate
- Parity (Odd, Even, or None)
- Number of data bits
- Number of stop bits

Example

Suppose you have five devices on the serial network. You decide that network wiring will be RS-485 two-wire.

- Set termination on the first and last device in the network. Set biasing on any one of the devices.
- Set communication parameters all the same, for example:
  - 9600 baud
  - Parity: none
  - 8 data bits
  - 1 stop bit

Data Type and Word Order

Values are interpreted based on the data type—for example, Boolean, float, 64-bit signed integer—and on word order.

As mentioned earlier, the original Modbus spec used registers defined as 16-bit unsigned integers (two bytes, also called one *word*). But 32-bit floats and 32- or 64-bit integers have become common. To accommodate the extra bits for integers or floats, device manufacturers had to use two or four contiguous 16-bit registers (two or four words).

Within a register Modbus specifies byte order as Big Endian: the most significant byte is at the lower address (which is more human-readable). But multi-register data is not specified. Some manufacturers put the most significant word in higher-numbered registers, and some put it in lower-numbered registers.

To get accurate data when using Opto 22 products with Modbus devices, refer to the documentation for your device to see how the manufacturer approaches data types that span more than one 16-bit register/address. Look for terms like high or low order, high or low registers, MSB and LSB (most significant byte and least significant byte).

Obviously values will be very different depending on how you read multi-register data. For example, a 32-bit integer has two words. Suppose the following words are sent (in hex):

- Register 1 = `1234`
- Register 2 = `5678`

Depending on word order, these two words could represent two very different values:

- `12345678` (hex), which would be 305,419,896 (decimal)
- `56781234` (hex), which would be 1,450,709,556 (decimal)

For 64-bit values, which span four registers, there are four possible ways to represent a value. If register values are (in hex):

- Register 1 = `1234`
- Register 2 = `5678`
- Register 3 = `90AB`
- Register 4 = `CDEF`

The resulting data could be:

Possible word order		Resulting data
Register containing LSB	32-bit value containing LSB	64-bit value (hex) from registers 1-4
Higher	Higher	1234 5678 90AB CDEF
Lower	Higher	5678 1234 CDEF 90AB
Higher	Lower	90AB CDEF 1234 5678
Lower	Lower	CDEF 90AB 5678 1234

The problem of word order is often tricky enough that the best way to determine it is to check a known value. If the known value is very different from what it should be, try changing the order and see if that fixes the problem.

Commands and Registers

Common Commands

Here are typical actions you'll want to take with Opto 22 products and the Modbus commands to use. Commands are shown in both decimal and hex:

Action	Command		Command definition
	Dec.	Hex	
Read digital output	01	01	Read Coil Status
Read digital input	02	02	Read Input Status
Read analog output	03	03	Read Holding Registers
Read analog input	04	04	Read Input Registers
Turn on/off one digital output	05	05	Force Single Coil
Write to one analog output	06	06	Preset Single Register
Turn on/off multiple digital outputs	15	0F	Force multiple coils
Write multiple analog outputs	16	10	Preset multiple registers
Report hardware and firmware revision levels	17	11	Report slave ID

For details on each command, see the *Modicon Modbus Protocol Reference Guide*.

Register Addressing

Remember that all numbers in Modbus messages are in hex. Each register is numbered. The Modbus protocol provides for four types of registers:

Register type	Register address (hex)	Typical use
Coils	00000–0FFFF	Write to digital points
Inputs	10000–1FFFF	Read digital points
Input registers	30000–3FFFF	Read analog points
Holding registers	40000–4FFFF	Write to analog points

When you build a Modbus message, **leave off the first digit** of the register, because the command itself indicates the group of addresses it applies to. For example, Modbus command 03 is Read Holding Registers, which are by definition in addresses 4xxxx. Therefore the initial 4 is left off the address.

Confusingly, Modbus register *addresses* are zero-based, but register *numbers* are one-based. That means that, for example, register number 1 is at register address 0, and register number 48F is at register address 48E.

Many manufacturers of Modbus products use the zero-based register *addresses:* for example, when you write to coil #1, you use address 0000. However, some manufacturers use register *numbers* based on 1, where coil #1 would be address 0001. If you are getting the wrong data at a register, check to see if the correct data appears in the address just before or just after, and adjust your address accordingly.

MODBUS TROUBLESHOOTING

If you can't get your Modbus device to communicate with your Opto 22 device, the easiest way to determine the source of the problem is to start with a known good Modbus program. Modscan32 is an inexpensive, well-known Modbus master used for testing. It talks both Modbus/TCP and Modbus serial.

Because Modscan32 is a known quantity, using it to test communication with the Modbus slave is a key step in making sure that your setup is correct and the data you're getting is accurate. If you haven't already, download Modscan32 utility software from Win-TECH: http://www.win-tech.com/html/modscan32.htm

Native Support: Troubleshooting Steps

If you're taking advantage of the native Modbus/TCP slave support in a *groov* product, OptoEMU Sensors, or Ethernet-based OptoMMP controllers or brains, follow these steps. For *groov* View, also see the *groov View User's Guide* (form 2027).

1. If the Opto 22 product is the Modbus slave, use Modscan32 as the Modbus master to connect to the Opto 22 device. If it works, check the inputs and outputs and familiarize yourself with how Opto channels (points) map to Modbus coils, inputs, holding registers, etc. Refer to the *Modbus/TCP Protocol Guide* (Opto 22 form 1678) for help.
2. If the Opto 22 product is the Modbus master, start by using Modscan32 as the master to connect to your slave device (instead of the Opto 22 master). Make sure you understand how the slave device uses data types and word order. Test known values to make sure they are correct.
3. If the slave device (Opto or other manufacturer) does not work properly with Modscan32, contact the device's manufacturer for help. (See page 8 for Opto 22 Product Support contact information.)
4. If the slave device works fine with Modscan32, then connect the real master and slave and test them together, again checking known values. If you have problems, review "Modbus Integration Issues" on page 3. If communication still doesn't work, contact Opto 22 Product Support (contact information is on page 8).

Integration Kit: Troubleshooting Steps

If you're using an integration kit with PAC Project, ioProject, or FactoryFloor controllers, follow these troubleshooting steps.

First, make sure you have changed the Modbus/TCP port as described in "Setting up Ports" on page 2.

Master Subroutine

1. If you're using a Modbus master subroutine (for any Opto product), connect Modscan32 to the slave device instead of connecting the Opto controller to the slave.
2. If you cannot establish basic communication, review "Modbus Integration Issues" on page 3 and check the guide that came with the integration kit. Contact Opto 22 Product Support if it still doesn't work. (Contact information is on page 8.)
3. Once basic communication is established, connect the slave device to the Opto 22 controller. Download the example master strategy (comes with the kit) to the controller, run it, and test each subroutine that the slave supports and you will need.
4. If all subroutines are working and known data values are appearing correctly, then talk to the slave using your own custom strategy to call the Modbus subroutines.

Slave Subroutine

1. If you're using a Modbus slave subroutine (for any Opto product), download the example slave strategy (comes with the kit) to the controller, run the strategy, and attempt to connect with Modscan32.

2. If it does not work, review "Modbus Integration Issues" on page 3 and check the guide that came with the integration kit. If it still doesn't work, contact Opto 22 Product Support (page 8).
3. When communication is established, try to connect the Modbus master you intend to use to the Opto product, still running the example strategy.
4. After working out any problems here and making sure that known data values are correct, finally integrate the slave strategy with your own custom code.

Troubleshooting Q&A

Q: I am talking, and I can see my digital stuff, but analog data (holding registers) look crazy. What's wrong?

A: The data type might be wrong, the word order might be wrong, or the startng register number might be wrong. Modbus registers are only 16-bit. So when we try to read 32-bit floats or 32- or 64-bit integers, the data needs to be arranged correctly (see "Data Type and Word Order" on page 4). You can check the slave device's user guide to see how the data is going to be returned. You can also use Modscan32 to just quickly read the data as each type until things look normal.

Q: What else can go wrong?

A: If it is Modbus/TCP, the same things that always go wrong with TCP communications: wrong IP address, bad cable, bad network, etc. Make sure you have changed the Modbus/TCP port as described in "Setting up Ports" on page 2.

If it is Modbus serial, the same things that always go wrong with serial communications: cable too long, cable mis-wired, baud rate, data bit, parity or stop bit mismatch, and so on. Use the TX/RX LEDs to troubleshoot the problem.

If you're using an integration kit with PAC Control and you've checked all the basics, you can go into Debug mode and check the specific queries that are being sent out and responses received (or in the case of the slave subroutine, the queries received and responses sent out). Compare these to the *Modbus Protocol Reference Guide* to determine the problem. This is an advanced step. It generally does not get to this point unless there is a bug in software or the device you're talking to.

Q: My slave device sent me an exception response. What's that?

A: It means the slave device received a query addressed to it, but for some reason it could not respond with the data requested. Look it up in Modicon's *Modbus Protocol Reference Guide*. There are six exception codes, but you'll usually only see 01 ILLEGAL FUNCTION or 02 ILLEGAL DATA ADDRESS. Exceptions are usually caused by using a function code that is not supported by the slave device for the register numbers being accessed. Verify the correct starting register number and the function codes supported for that register number. Try reading only one value instead of a group of values. Using ModScan32 can be very helpful in this situation.

Q: My *groov* EPIC or SNAP PAC processor just needs to be a Modbus/TCP slave. Do I need the toolkit?

A: No, you can use the built-in Modbus/TCP slave functionality. All of the points are easily available. Some of the other memory map addresses can be directly accessed by determining where they are with the Modbus calculator in PAC Manager. Strategy variables can be accessed by building a chart to move them to or from the Scratch Pad, which is also available via Modbus/TCP. For more information, see the *Modbus/TCP Protocol Guide*, form 1678.

MADE IN THE USA

FOR HELP

Other Documents

You may find these other Opto 22 documents useful, too:

To do this	See this document	Form
Adding Modbus/TCP devices in *groov* View	groov View User's Guide	2027
Changing Modbus port on a PAC (not on EPIC) or calculating Modbus addresses (other than I/O points)	PAC Manager User's Guide	1704
Using Modbus/TCP with Opto 22 Ethernet controllers and brains	Modbus/TCP Protocol Guide	1678
Using subroutines in Modbus integration kits	PAC Control User's Guide	1700
Using Modbus with OptoEMU Sensors	OptoEMU Sensor Communication Guide	1958
Using Modbus integration kits:		
Modbus Integration Kit, PAC Control	Modbus Integration Kit for PAC Control Technical Note	2164
Modbus/Serial Integration Kit, ioControl	Modbus/Serial Integration Kit for ioProject	1660
Modbus/TCP Integration Kit, ioControl	Modbus/TCP Integration Kit for ioProject	1644
Modbus Integration Kit, OptoControl (Serial)	Modbus 3.1 Integration Kit for OptoControl	1128

Opto 22 Product Support

If you have problems using Modbus devices with Opto 22 products and cannot find the help you need in this guide or the guides listed above, contact Opto 22 Product Support.

NOTE: Email messages and phone calls to Opto 22 Product Support are grouped together and answered in the order received.

Phone:	800-TEK-OPTO (800-835-6786 toll-free in the U.S. and Canada) 951-695-3080 Monday through Friday, 7 a.m. to 5 p.m. Pacific Time
Fax:	951-695-3017
Email:	support@opto22.com
Opto 22 website:	www.opto22.com