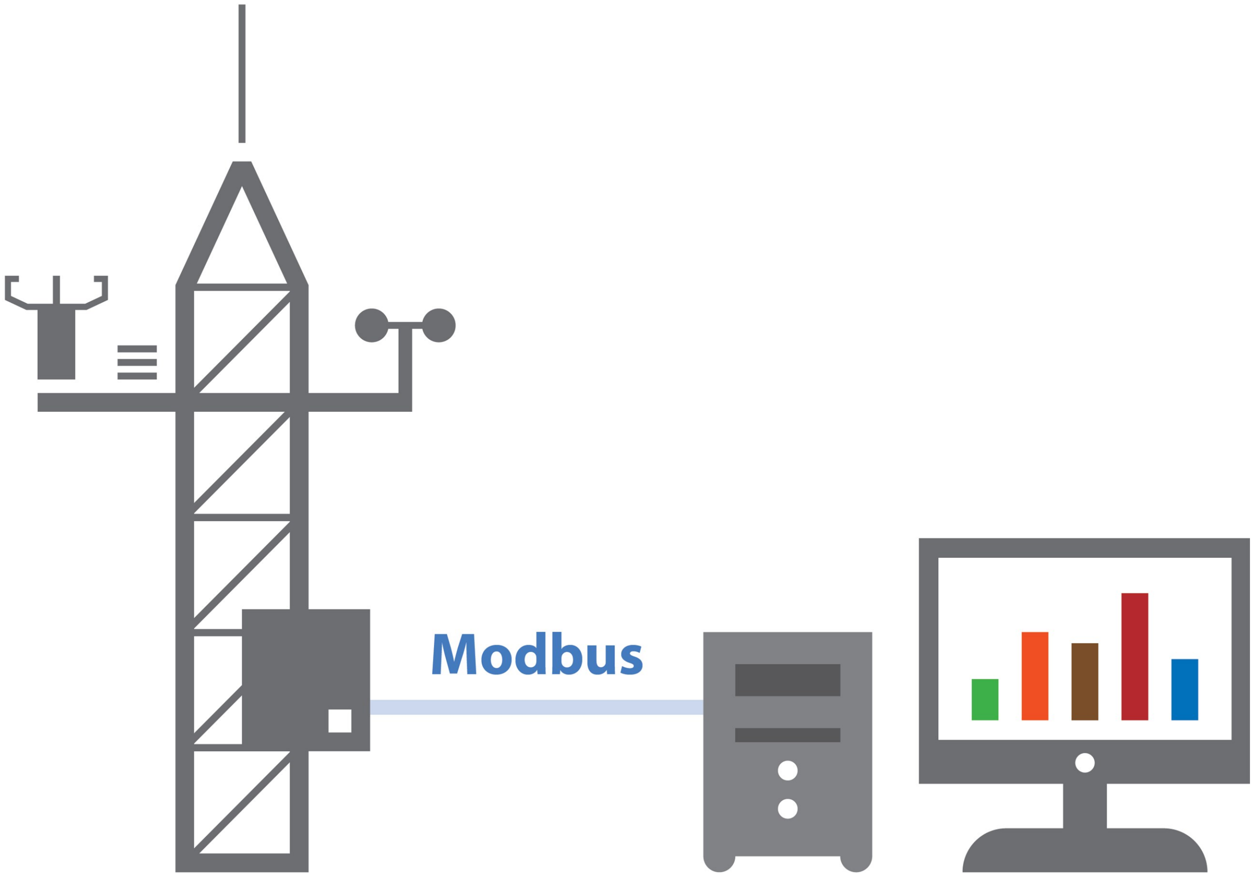
APPLICATION NOTE

Communications Device

Modbus

Troubleshooting Guide



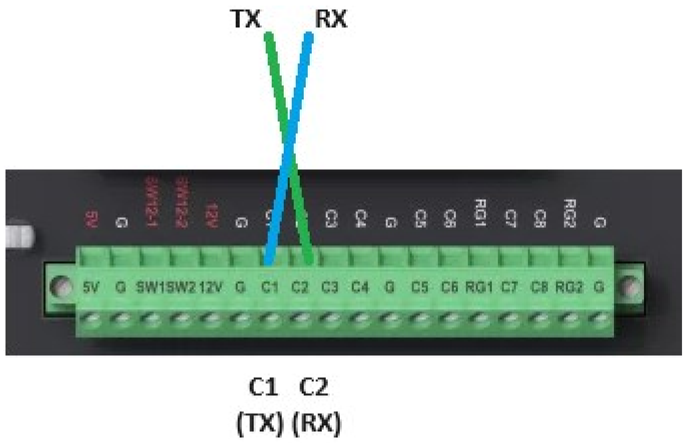
10/2024

Table of contents

1. RS-232, RS-485, and Ethernet wiring

Modbus can run on multiple interfaces on Campbell devices, including RS-232, RS-485, and Ethernet. Ensure that the correct cable is connected to the appropriate port on both the data logger and the Modbus device by tracing the cable and confirming that the leads are properly connected to the required port on the data logger.

1. When connecting a Modbus device to your data logger using RS-232 over **C** or **U** ports, verify that the TX wire coming from the other Modbus device is connected to the **RX** port on your data logger and that the RX wire from the Modbus device connects to the **TX** port on your data logger.



2. When wiring for RS-485 communications, ensure that the ports you are using on the data logger support RS-485. For example, **C1**-**C4** on a CR1000X do not support RS-485. However, **C5**-**C8** all support RS-485.

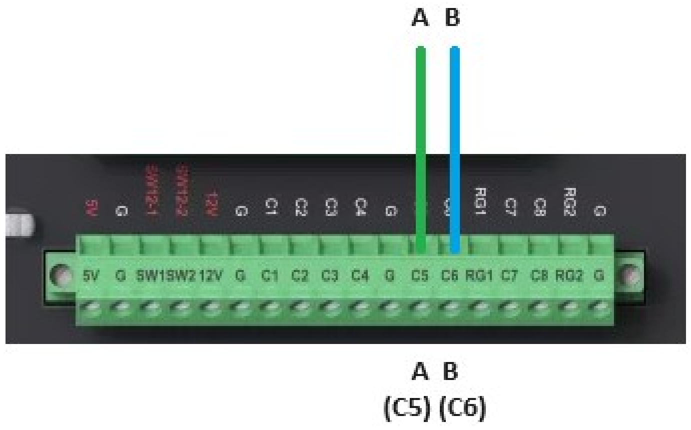
> **NOTE:**
> RS-485 is supported on all comport pairs on a CR1000Xe.

   In RS-485 communications, data is transmitted using either two wires for half-duplex or four wires for full-duplex. These wires are labeled as: A (or A–) and B (or B+). In full-duplex setups, TD A– specifically refers to the inverted terminal or wire used for transmitting data signals from a device, while TD B+ refers to the non-inverting wire used for transmit. The

RD A– wire refers to the inverted receive wire, and the RD B+ refers to the non-inverted receive wire.

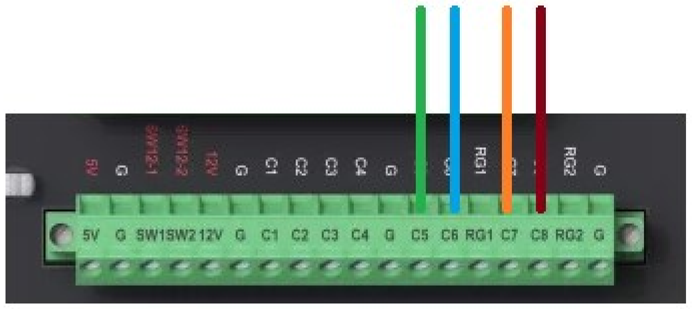
3. When wiring your logger for RS-485 in half-duplex mode, ensure that the A wires on your Modbus device connect to the designated **A** port on the data logger, and the B wires connect to the designated **B** port on the data logger.

   NOTE:
   Some manufacturers don't adhere to the RS-485 standard and may label their A and B wires backwards. If you find your setup isn't operating, you might try swapping the wires on the third party manufacturer's device. A way to identify the correct wiring is if the manufacturer labels one port as **–**, you can expect it to be the **A** port. If they label the wiring as **+**, then you can expect it to be the **B** port.



4. If wiring your data logger for RS-485 full-duplex mode in **Server** configuration, ensure the following connections:
   - Connect the TD A– wire from the third-party Modbus device to the **RD A–** port on the data logger.
   - Connect the TD B+ wire from the third-party Modbus device to the **RD B+** port on the data logger.
   - Connect the RD A– wire from the third-party Modbus device to the **TD A–** port on the data logger.
   - Connect the RD B+ wire from the third-party Modbus device to the **TD B+** port on the data logger.

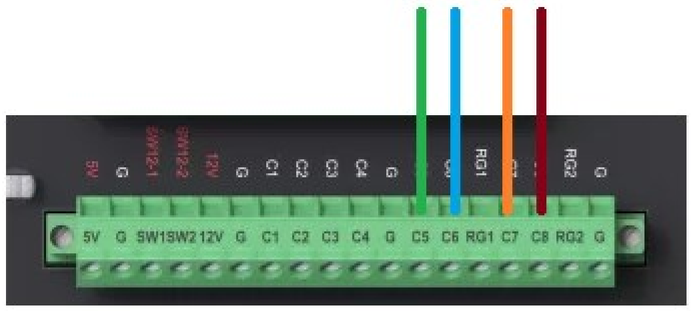
Client/Master TD A- TD B+ RD A- RD B+



Server/Slave RD A- RD B+ TD A- TD B+
C5 C6 C7 C8

5. If wiring your logger for RS-485 in full duplex mode in **Client** configuration, ensure the following connections:
   - Connect the RD A– wire from the third-party Modbus device to the **TD A–** port on the data logger.
   - Connect the RD B+ wire from the third-party Modbus device to the **TD B+** port on the data logger.
   - Connect the TD A- wire from the third-party Modbus device to the **RD A–** port on the data logger.
   - Connect the TD B+ wire from the third-party Modbus device to the **RD B+** port on the data logger.

Server/Slave RD A- RD B+ TD A- TD B+



Client/Master TD A- TD B+ RD A- RD B+
C5 C6 C7 C8

6. When connecting your data logger directly to a Modbus device over Ethernet or to an Ethernet network, verify the Ethernet cable is connected to the **RJ45 Ethernet** port, not the **RJ45 CPI** port.

2. Power considerations for data logger and Modbus device

You can verify the data logger has sufficient power by connecting to it and checking the battery voltage in the **Status** table. A healthy voltage typically reads slightly over 12 V. For example, a reading of 13.5 V is acceptable. Lower readings, such as 11 or 10.5 V may impair or completely prevent optimal communications. The battery voltage may be checked using ***Device Configuration Utility*** > **Data Monitor [tab]** > **Status** > **Battery**.

If voltage is low, check the 12 V power input to the data logger. Start by disconnecting the cable from the data logger side, then proceed to check the charging regulator, solar panel, or other power input, as well as the battery, to identify the source of the power issue.

See video on **Measuring Data Logger Output Voltage With a Multimeter** here: www.campbellsci.com/videos/basic-troubleshooting-01 ▶.

3. Choosing the correct Modbus instruction (`ModbusServer()` or `ModbusClient()`)

The terminology for Modbus instructions has changed. If you are accustomed to the former language of "Modbus Slave" and "Modbus Master" here's a quick reference to the updated language and a general description of their functions. It is essential to use the correct instruction for your application. If you are communicating with a device that is operating as a Modbus Server then you must be set up as a Modbus Client. The inverse is true, if you are communicating with a device that is set up as a Modbus Client then you must be set up as a Modbus Server.

NOTE:
The old instruction names are still supported in the ***CRBasic Editor*** and in the data logger operating systems, though they won't highlight blue in the ***CRBasic Editor***.

`ModbusServer()` = `ModbusSlave()`	Typically, it holds data and shares it with the Client or Master. (It primarily provides data.)
`ModbusClient()` = `ModbusMaster()`	Generally, requests data and receives it from the Server or Slave. (It generally receives data.)

NOTE:
Your data logger can simultaneously operate as both a `ModbusServer()` with one device to offer or share data and as a `ModbusClient()` with another device over a separate connection.

4. ModbusServer() parameters details

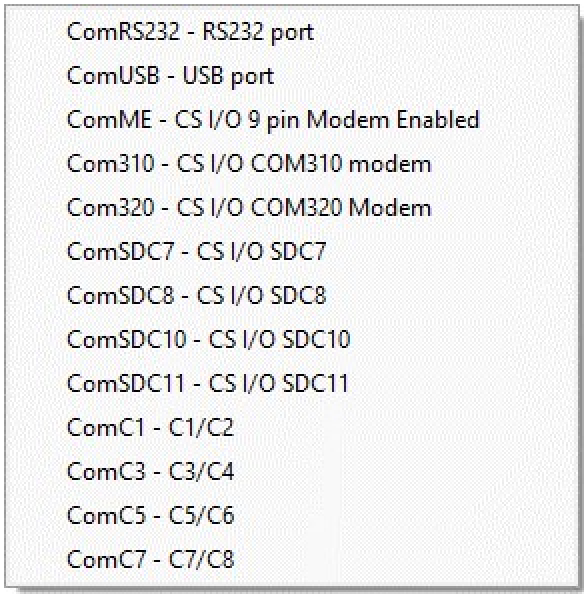
The `ModbusServer()` instruction enables a data logger configured as a Modbus Server to communicate with a device configured as a Modbus client. In the data logger program, the `ModbusServer()` instruction should be placed after the `BeginProg` instruction and before the `Scan()` instruction.

The `ModbusServer()` instruction has the following syntax:

```
ModbustServer (COMPortMB, BaudRate, MBAddr, ModbusVariable, ModbusBooleanVar,
ModbusOption)
```

The following are steps to ensure each parameter is configured correctly:

1. **ComPortMB:** Ensure that the `COMPortMB` setting matches the physical port that the Modbus device is wired into. Valid settings depend on the data logger model. Some example `COMPortMB` settings include COMRS232, ComC1, ComC3, and ComC5. The full list of options available to you can be viewed by right clicking the blue highlighted words **ModbusServer** in the *CRBasic Editor*.



```
ModbusServer (ComRS232,9600,1,ModIn(),0,0)
```

When using Modbus over TCP/IP, setting the `COMPort` parameter to 502 or higher designates the Modbus TCP/IP service port that your data logger will monitor for incoming Modbus connections over TCP/IP.

2. **BaudRate:** If using RS-232 or RS-485 for Modbus, check that the Modbus Server baud rate matches the Modbus client baud rate. If they do not match, they will not be able to communicate with each other. This parameter is not relevant when using Modbus over a TCP/IP connection, and therefore will be ignored.

```
ModbusServer (ComRS232,9600,1,ModIn(),0,0)
```

3. **MBAddress:** Verify the Modbus address for the data logger is correct. Each device in a Modbus network must have a unique Modbus address. Additionally, make sure to record this address so that other devices can be configured to connect to the data logger using the correct address.

```
ModbusServer (ComRS232,9600,1,ModIn(),0,0)
```

4. **Modbus Variable:** Ensure that the Modbus variable is set and dimensioned to the correct size. The dimension size depends on the Modbus command. For example, if you are offering three values, ensure that the array is defined here and has also been declared as a `Public` variable at the beginning of your program.

NOTE:
When these values are pulled or sent to the Modbus Server, the data logger does not differentiate between holding and input registers. The only difference is the address offset. The specified array will be used for requests of input (address offset of 30000) or holding (address offset of 40000) registers. Making the first address for input 30001 and the first address for output 40001.

NOTE:
Floating point variables take two Modbus registers.

```
ModbusServer (ComRS232,9600,1,ModIn(),0,0)
```

`Public` Modbus variable declaration example assuming 3 Modbus values are being made available via `ModbusServer()`:

```
Public ModIn(3)
```

The array values need to be set to a value in the `Scan` of the program somewhere between the `Scan` and `NextScan` statements. The following is an example using three values:

```
ModIn(1) = Batt_volt
ModIn(2) = PTemp
ModIn(3) = ModResult
```

5. **Modbus Boolean Variable:** This variable is used to hold the value of any discreet on/off commands sent to your Modbus Server. This parameter can be a variable or a variable array. If you're not interested in using this function, set this value to 0. You may see a compile warning when using a 0; you can safely ignore the warning.

```
ModbusServer (ComRS232,9600,1,ModIn(),0,0)
```

6. **ModbusOption:** Verify the correct `ModbusOption` has been selected. Specifying a value for this parameter is optional. If you don't specify a value, `ModbusOption` will default to a 32-bit float or long with the reversed CDAB byte order.

```
ModbusServer (ComRS232,9600,1,ModIn(),0,0)
```

The options for this parameter in CRBasic are:

32-bit float or Long/reversal of byte order (CDAB)

16-bit signed integer

32-bit float or Long/no reversal of byte order (ABCD)

16-bit unsigned integer

Modbus ASCII 4 bytes (CDAB)

Modbus ASCII 2 bytes

Modbus ASCII 4 bytes (ABCD)

> NOTE:
> Record the parameters used in the `ModbusServer()` instruction for future reference when setting up a Modbus client.

5. Modbus client troubleshooting

Refer to the following when troubleshooting a data logger that is set up as a Modbus Client:

5.1 Modbus client credentials checklist for third party Modbus server device

You need to gather key information from the Modbus (Server) device the data logger will communicate with. This information can be found in the manufacturer's documentation, the device configuration, and/or from the person who programmed the device. Use the following checklist to ensure you have all the necessary details:

☐ Physical interface the Modbus (Server) device used to connect to the data logger.

Interface: ______________________

☐ Baud rate of the other Modbus (Server) device (unless using an IP connection).

Baud rate: ______________________

☐ Modbus address of the Modbus Server your data logger will be communicating with.

Server address: ______________________ Client address: ______________________

☐ Starting register number for the Modbus data being retrieved from the Modbus Server.

Starting register: ______________________

☐ Number of values you are receiving from the Modbus Server and the order that they are received. Each value will have an address that you will need to know. In your CRBasic program, multiple `ModbusClient()` instructions are needed to retrieve data from non-sequential addresses.

Number of values: ______________________

☐ DataType of the Modbus data provided by the Modbus Server device. This includes understanding both the bit length and the byte order. The default is a 32-bit float or long with a reversed byte order of CDAB. The DataType will be one of the following:

32-bit float or Long/reversal of byte order (CDAB)
16-bit signed integer
32-bit float or Long/no reversal of byte order (ABCD)
16-bit unsigned integer
Modbus ASCII 4 bytes (CDAB)
Modbus ASCII 2 bytes
Modbus ASCII 4 bytes (ABCD)
Modbus Data Type: ______________________

☐ If making a connection over TCP/IP (Ethernet, Wi-Fi, or Cellular), you will need the IP address of the server.

Modbus Server IP address: ______________________

☐ When connecting over TCP/IP (Ethernet, Wi-Fi, or Cellular), you will need to know the port number the Modbus Server uses to provide the data. The default is 502, but you should confirm correct port number.

Modbus Server port number: ______________________

Once you have the necessary information, you can program your data logger using the `ModbusClient()` instruction to communicate with the Modbus Server device. This information can work as a guide for programming your Modbus device and is particularly useful for verifying your parameters during troubleshooting. Details on these parameters can be found in the *CRBasic Editor* help.

5.2 Setting up communication port for Modbus communications

NOTE:
Skip this section if using Modbus over an IP connection.

When setting up the data logger as a Modbus Server, the communication port needs to be set up for Modbus communications. The most common method for doing this is to use the `SerialOpen()` CRBasic instruction.

1. The `COMPort` parameter in your `SerialOpen()` instruction must match the physical port you have wired your Modbus device to. The example uses COMRS232. Valid values include but are not limited to: COMRS232, COMC1, COMC3, and COMC5. The COMPort options will vary depending on the data logger model you are working with. The full list of options available to you can be viewed by right clicking the `COMPort` parameter in the `SerialOpen()` instruction in the *CRBasic Editor*.

   ```
   SerialOpen (COMRS232,-9600,0,0,50)
   ```

2. The baud rate in your `SerialOpen()` instruction must match the other Modbus devices baud rate. The example uses –9600. A negative sign in front of the baud rate value indicates that the interfaces will autobaud if the specified baud rate doesn't match. This option is only compatible with the **RS-232** port. Right clicking the value in the ***CRBasic Editor*** will show you all the available options.

   ```
   SerialOpen (COMRS232,-9600,0,0,50)
   ```

3. The `SerialOpenFormat` parameter is only used for an RS-232 connection. The example uses the default value of zero, which sets the format to **RS232 No Parity/one stop bit/8 data bits/no error checking with concurrent Pakbus over the port**. This format is used for most RS-232 connections. However, check the manufacturer's documentation for your Modbus device to verify that you are using the correct format.

   ```
   SerialOpen (COMRS232,-9600,0,0,50)
   ```

4. The **CommsMode** of the interface must match the serial protocol you intend to use. The example uses code 0, which configures the port as RS232. Available modes include RS-232, TTL, RS-485 Half-Duplex Pakbus, RS-485 Half-Duplex transparent, and RS-485 Full-Duplex transparent. Note that not all modes are applicable to every physical interface. For instance, an RS-232 interface typically cannot support RS-485 protocols. However, the C5/C6 pair on a CR1000X can be configured for either RS-232 or RS-485 communications options.

```
SerialOpen (COMRS232,9600,0,0,50,0)
```

5.3 ModbusClient() parameter details

The `ModbusClient()` instruction should be placed in your program scan loop, after the `Scan` statement and before the `NextScan` statement.

1. Declare and reference a variable for storing the `ModbusResultCode`. Result is the variable used in the example. The ModbusResultCode indicates if the instruction failed and provides the reason for the failure (see Result code descriptions [p. 16]).

   ```
   ModbusClient (Result,COMRS232,115200,3,3,ModbusData(),1,10,3,100)
   ```

2. The `COMPort` is the port that the Modbus device is wired into. The following example uses RS232. Other valid ports include, but are not limited to, ComSDC7, ComC1, ComC2, ComC3, and ComC4.

   ```
   ModbusClient (Result,COMRS232,115200,3,3,ModbusData(),1,10,3,100)
   ```

   An IP connection can also be used for Modbus communications by specifying a variable containing an opened IP socket to the remote Modbus device as the COMPort. See the following example:

   Declare a variable named `Socket` at the top of your program in the `Public` variables.

   ```
   Pubic Socket
   ```

   In your program, before the `ModbusClient()` instruction, use the `TCPOpen()` instruction with a `Socket=` before the instruction to open the TCP/IP connection to the remote Modbus device as seen in the following. For example:

   ```
   Socket=TCPOpen ("192.168.1.2",502,30)
   ```

NOTE:
The first parameter in the `TCPOpen()` instruction is the IP address of the remote Modbus device. The second parameter is the port number the device is using to listen for Modbus traffic, typically port 502, as shown in the preceding example. The final parameter is the size of the memory buffer for the TCP/IP socket used for Modbus communications. This value should never be set to 0 when performing Modbus communications.

The `ModbusClient()` instruction should specify the `Socket` as shown in the following example:

```
ModbusClient (Result,Socket,115200,3,3,ModbusData(),1,10,3,100)
```

NOTE:
When using an IP Socket, the baud rate parameter is ignored.

If doing Modbus over an IP connection, refer to CR1000X example program for ModbusClient() over TCP/IP (Modbus TCP) (p. 30) for more information.

3. The baud rate must match the baud rate used on the other Modbus device. In the example, the baud rate is 115200 bps. If doing Modbus over a TCP/IP connection, this parameter is ignored.

   ```
   ModbusClient (Result,COMRS232,115200,3,3,ModbusData(),1,10,3,100)
   ```

4. The Modbus address must match the Modbus address of the other Modbus (Server) device. If unsure of the address of the other device, check the documentation, configuration, or programming of the Modbus device.

   ```
   ModbusClient (Result,COMRS232,115200,3,3,ModbusData(),1,10,3,100)
   ```

5. For the `ModBusFunction`, the example uses Option 3, which reads the holding Registers from the Modbus Server. Option 3 is the most common option. Descriptions of all the options are provided in Table 5-1 (p. 13).

   ```
   ModbusClient (Result,COMRS232,115200,3,3,ModbusData(),1,10,3,100)
   ```

Table 5-1: Modbus function options

Option	Name	Description
1	Read Coil/Port Status	Reads the On/Off status of discrete output(s) in the ModbusServer
2	Read Input Status	Reads the On/Off status of discrete input(s) in the ModbusServer
3	Read Holding Registers	Reads the binary contents of holding register(s) in the ModbusServer
4	Read Input Registers	Reads the binary contents of input register(s) in the ModbusServer
5	Force Single Coil/Port	Forces a single Coil/Port in the ModbusServer to either On or Off
6	Write Single Register	Writes a single register value (16-bit long) to a ModbusServer (ModbusOption must be set to 1)
15	Force Multiple Coils/Ports	Forces multiple Coils/Ports in the ModbusServer to either On or Off
16	Write Multiple Registers	Writes values into a series of holding registers in the ModbusServer

6. If your Modbus option is set to read data, the `ModbusVariable` stores the data received from the Modbus Server. If you are writing to the Modbus Server, this variable is the source of the data being sent.

   Declare the variable at the top of your CRBasic program along with the other `Public` variable declarations. The following is an example of the `Public` Modbus variable declaration, assuming 20 floating point Modbus values are being read from the `ModbusServer()` or held to be written to the Modbus Server:

   ```
   Public ModbusData(20)
   ```

   The type of variable you declare as the `Public` variable will affect its function. Be sure to declare the variable as the correct type based on the `ModbusFunction` and `ModbusOption` you are using. Refer to the `ModbusFunction`, which is the last parameter in the `ModbusClient()` instruction shown in the following example. If the values you are reading or writing are declared as four-byte floating-point (Float) data types (default in the preceding example), and a register function code of 3, 4, or 16 is used, the data points will be mapped to a Modbus Holding or Input register as floating-point data.

This parameter can also be declared as Boolean data type, which is used to represent conditions or hardware that have only two states (true or false), such as flags and control ports. A register function code (coil) of 1, 2, 5, or 15 will be mapped to a Modbus coil in the other device.

```
Public ModbusData(2) As Boolean
```

If this parameter is declared as a Long data type (used for integer variables), and a register function code of 3, 4, or 16 is used, it will be mapped to Modbus Holding or Input registers as an integer. Longs are treated as signed integers, meaning the values will range from –32,768 to +32,767.

```
Public ModbusData(2) As Long
```

This is the parameter as it appears in the `ModbusClient()` instruction in your data logger program.

```
ModbusClient (Result,COMRS232,115200,3,3,ModbusData(),1,10,3,100)
```

7. Ensure that you have specified the correct Modbus start register. This is the 16-bit address of the first register that will be requested (or acted upon). The start parameter is entered as a number in the range of 1 to 65535. The start parameter in the example is 1. The data address, or offset, in the Modbus frame sent to the server will be equal to "Start-1". For example, if you are reading the first starting register of a Modbus device, you should specify 1 for this parameter. For a Modbus function code 3, your manufacturer’s documentation might indicate the starting address is 40001, but you should specify 1 in your data logger. Setting the value to 40001 will tell your data logger to start reading 40,001 records into the holding register instead of the first one. This is a common mistake when using the `ModbusClient()` instruction.

> NOTE:
> Some manufacturers may have addressing errors in their documentation or a difference in approach. Campbell Scientific data loggers use "Start-1" addressing to determine the address being used. Some manufacturers might start their addressing at 0 or 1, which can create confusion. If you specify the starting address but do not receive correct values, it’s a good practice to attempt reading a fixed value from the Modbus

> registers. Then, adjust the address incrementally until you receive the correct data from the other Modbus device.

```
ModbusClient (Result,COMRS232,115200,3,3,ModbusData(),1,10,3,100)
```

8. The `ModbusLength` parameter is correct. This is the number of CRBasic variables to act on with this instruction. When reading coils or when using a 16-bit related `ModbusOption` parameter of 1, 3, 11, or 13, the `ModbusLength` parameter will match the number of 16-bit register values requested or acted upon. When using a 32-bit or 4-byte `ModbusOption` parameter (Modbus Options 0,2,10, or 12), the data logger will automatically double the Length value specified so that it requests 32-bits of data for each CRBasic variable. In short, specify the number of values you need and the data logger will handle the rest automatically, as long as you have selected the correct `ModbusOption` parameter. See item 11 to verify the correct `ModbusOption` parameter.

   ```
   ModbusClient (Result,COMRS232,115200,3,3,ModbusData(),1,10,3,100)
   ```

9. Verify the `ModbusTries` parameter. This value is the number of times the data logger will attempt to communicate with the Modbus Server before moving on to the next instruction in the program.

   ```
   ModbusClient (Result,COMRS232,115200,3,3,ModbusData(),1,10,3,100)
   ```

10. The `ModbusTimeOut` parameter is the amount of time the data logger program will wait for a response before considering the attempt a failure. This value is indicated in units of .01 seconds. Therefore, a value of 100 is equals one second. If you are using Modbus over an IP connection with some latency, you may want to increase this value to a few seconds. For close serial connections, one second is typically sufficient.

    ```
    ModbusClient (Result,COMRS232,115200,3,3,ModbusData(),1,10,3,100)
    ```

11. Verify the correct `ModbusOption` has been selected. If you don't apply a value for this parameter, it will default to 32-bit float or Long/reversal of byte order (CDAB). This parameter needs to match the other Modbus device your data logger is communicating with.

    ```
    ModbusClient (Result,COMRS232,115200,3,3,ModbusData(),1,10,3,100,0)
    ```

    Check the other manufacturer's manual for the other Modbus device or consult the person who programmed it to verify the correct format. Manufacturers will not always use clear language to spell out the format they are using. Some will only specify the bit size, like 16 or 32-bit. Others might use the phrase “reverse byte order,” or say “reverse endian,” or say just “CDAB.” Some might not list it at all, and you might have to contact the person who programmed the system to find out which byte order they used. The alternative is to keep trying different ones until it works. See Serial formatting (p. 32) for more information on big-endian and little-endian when it comes to Modbus communications.

The options for this parameter in CRBasic are:

32-bit float or Long/reversal of byte order (CDAB)

16-bit signed integer

32-bit float or Long/no reversal of byte order (ABCD)

16-bit unsigned integer

Modbus ASCII 4 bytes (CDAB)

Modbus ASCII 2 bytes

Modbus ASCII 4 bytes (ABCD)

5.4 Result code descriptions

Table 5-2 (p. 16) describes the result codes provided by the `ModbusClient()` instruction. When Modbus communications fail, the result code often provides an indication of what is wrong with your Modbus communications. A similar table is available in the ***CRBasic Editor*** help.

Table 5-2: Result code descriptions

Code	Descriptions
–1	Illegal function. The function code received in the query is not an allowable action for the device. The device may not support the function, or it may not be in a state to process the request. Check your `ModbusFunction` in the Modbus client troubleshooting (p. 8)
–2	Illegal data address. The data address received in the query is not an allowable address for the device. The combination of the reference number and transfer length may be invalid. Check the manufacturer's documentation of your Modbus Server and match the setting in the Modbus client troubleshooting (p. 8) section, item 7.
–3	Illegal data value. The value contained in the query data field is not an allowable value for the device.
–4	Server device failure: an unrecoverable error occurred while the device was attempting to perform the requested action.

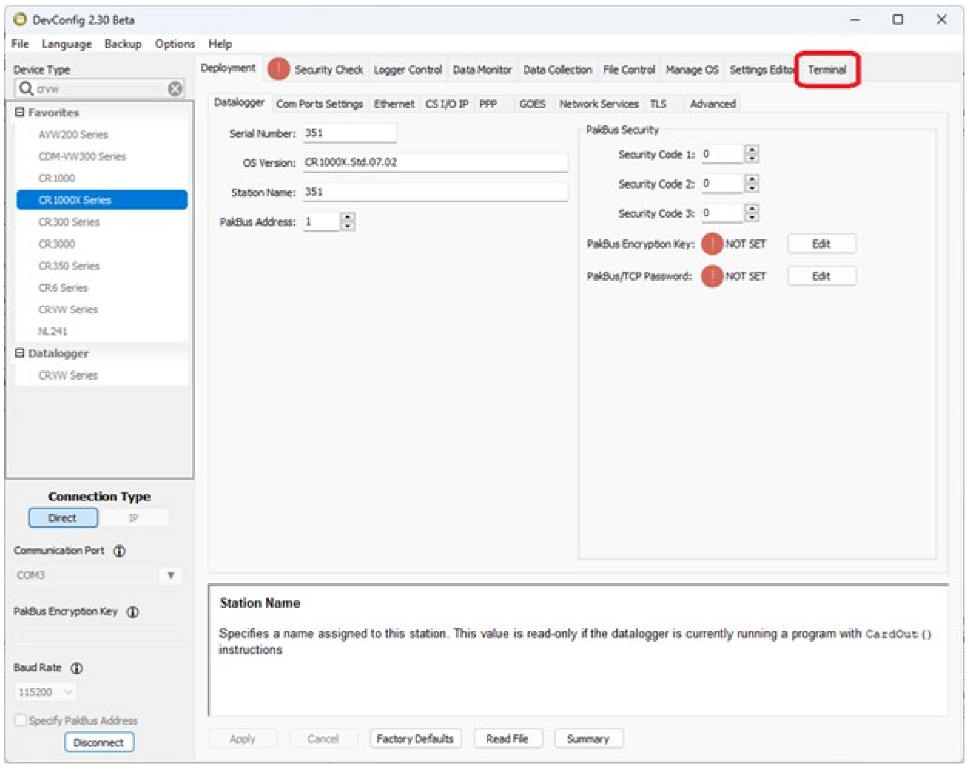
Table 5-2: Result code descriptions

Code	Descriptions
–5	Acknowledge. The device has accepted the request and is processing it, but a long duration of time will be required to do so. This is a specialized function used in conjunction with programming commands.
–6	Server device busy. The device is engaged in processing a long-duration program command.
–8	Memory parity error. The device attempted to read a record file but detected a parity error in the memory. Used in conjunction with function codes 20 and 21.
–9	Gateway path unavailable. Indicates that the gateway was unable to allocate an internal communications path from the input port to the output port for processing the request.
–10	ModbusClient error. Received an unexpected function code response from server.
–11	The specified ComPort (or TCP socket) is not opened. There is no connection to the Modbus Server. Verify the correct parameter is specified in the Modbus client troubleshooting (p. 8) section, item 2.
–16	ModbusClient error, out of comms memory.
–20	ModbusClient error. Variable not dimensioned large enough to store results from server.
1..2..n	Communications with the Modbus Server has failed. This can be due to incorrect serial communications settings or incompatible data types. See Setting up communication port for Modbus communications (p. 10) and Modbus client troubleshooting (p. 8).

6. Watch Modbus RTU traffic from the data logger terminal mode

You can ensure that your data logger and the other Modbus device are sending Modbus traffic to one another. This procedure shows using the data logger terminal mode to watch the outgoing and incoming Modbus traffic from a Campbell Scientific data logger. Although the example shows a Modbus Client, the steps are the same for a Modbus Server.

1. Open ***Device Configuration Utility*** and connect to your data logger.
2. Click the **Terminal** tab.



Serial Number: 351

OS Version: CR1000X.Std.07.02

Station Name: 351

PakBus Address: 1

PakBus Security

Security Code 1: 0

Security Code 2: 0

Security Code 3: 0

PakBus Encryption Key: NOT SET

PakBus/TCP Password: NOT SET

Communication Port: COM3

PakBus Encryption Key

Baud Rate: 115200

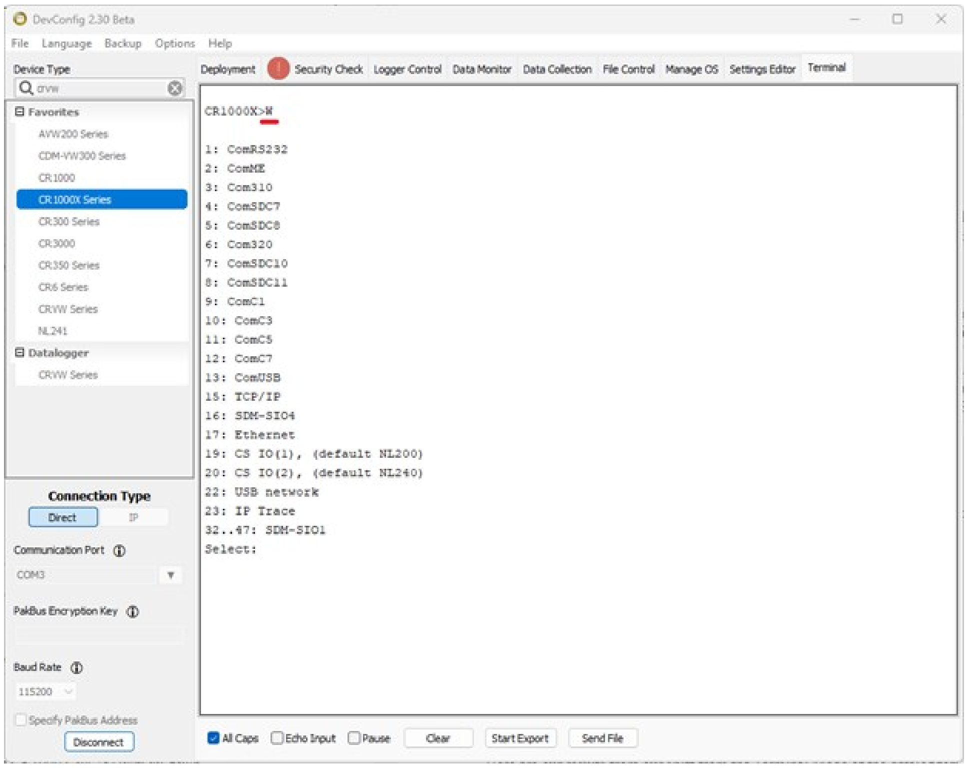
Specify PakBus Address

Station Name

Specifies a name assigned to this station. This value is read-only if the datalogger is currently running a program with `CardOut()` instructions

3. Press **Enter** until you see the **CR1000X>** prompt (your data logger model may be different).

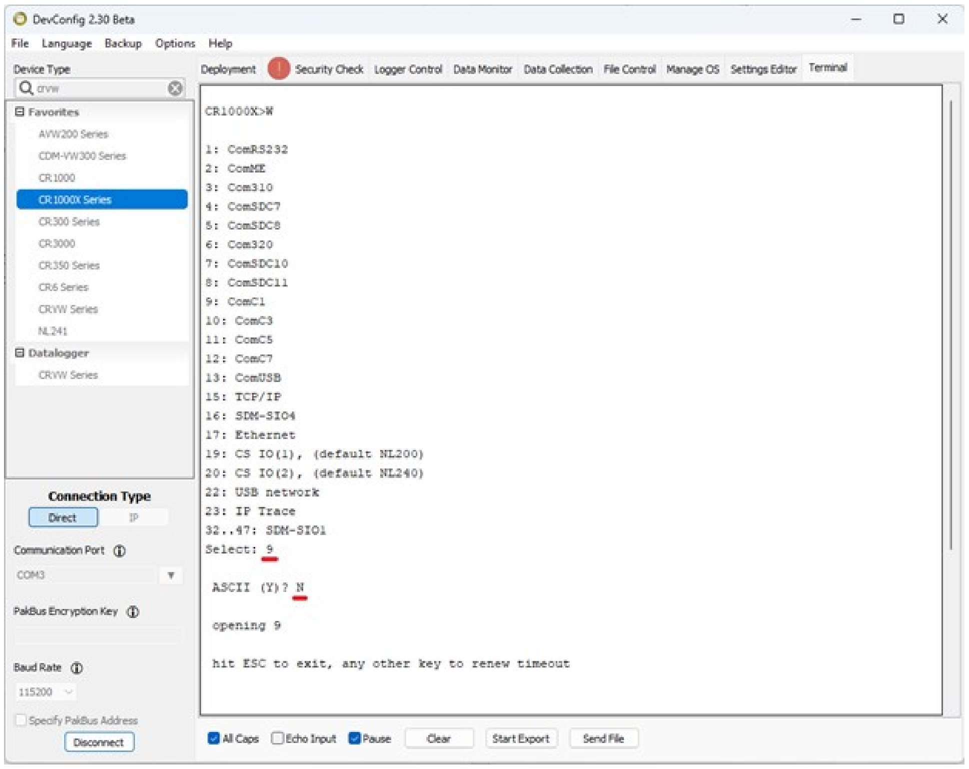
4. Type the capital letter `W` and press **Enter**.



```
CR1000X>W

1: ComRS232
2: ComME
3: Com310
4: ComSDC7
5: ComSDC8
6: Com320
7: ComSDC10
8: ComSDC11
9: ComC1
10: ComC3
11: ComC5
12: ComC7
13: ComUSB
15: TCP/IP
16: SDM-SIO4
17: Ethernet
19: CS IO(1), (default NL200)
20: CS IO(2), (default NL240)
22: USB network
23: IP Trace
32..47: SDM-SIO1
Select:
```

5. Enter the number that corresponds to the COM port you are using and then press **Enter**. If you are using IP for Modbus communications, go to the next section.

6. You will be prompted for `ASCII (Y)?`; answer with a capital letter **N** and press **Enter**.

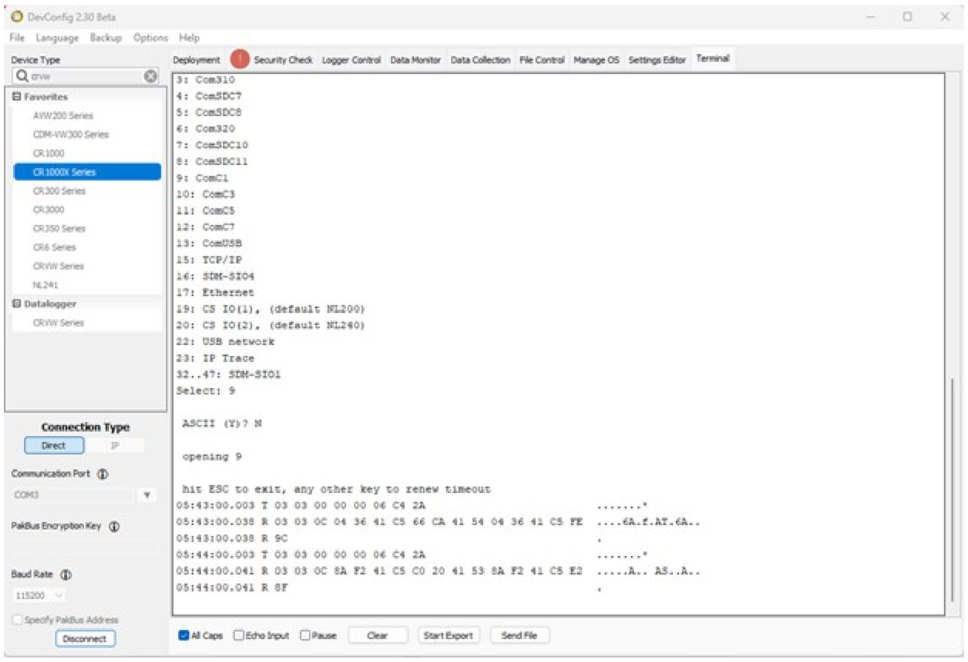


```
CR1000X>W

1: ComRS232
2: ComME
3: Com310
4: ComSDC7
5: ComSDC8
6: Com320
7: ComSDC10
8: ComSDC11
9: ComC1
10: ComC3
11: ComC5
12: ComC7
13: ComUSB
15: TCP/IP
16: SDM-SIO4
17: Ethernet
19: CS IO(1), (default NL200)
20: CS IO(2), (default NL240)
22: USB network
23: IP Trace
32..47: SDM-SIO1
Select: 9

 ASCII (Y)? N

 opening 9

 hit ESC to exit, any other key to renew timeout
```

7. You should now see a message that says `opening` followed by the port you specified. After that you will see the `hit ESC to exit, any other key to renew timeout` message. Now wait for your Modbus instruction to trigger or manually trigger it in your program. If the results begin to fill up the screen quickly, clicking the **Pause** check box at the bottom of the ***Device Configuration Utility*** window will pause the display, allowing you to take your time reading what's already been displayed.

   To export the results for later analysis, click the **Start Export** button. In the **Choose an export file** window, select a location to save the results of the sniff, then enter a file name (e.g., *myModbussniff.txt*) in the **File name** field at the bottom of the window. Finally, click the **Save** button.

Here is a screenshot of a sniff from the **Terminal** mode of the data logger:



```
3: Com310
4: ComSDC7
5: ComSDC8
6: Com320
7: ComSDC10
8: ComSDC11
9: ComC1
10: ComC3
11: ComC5
12: ComC7
13: ComUSB
15: TCP/IP
16: SDM-SIO4
17: Ethernet
19: CS IO(1), (default NL200)
20: CS IO(2), (default NL240)
22: USB network
23: IP Trace
32..47: SDM-SIO1
Select: 9

 ASCII (Y)? N

 opening 9

 hit ESC to exit, any other key to renew timeout
05:43:00.003 T 03 03 00 00 00 06 C4 2A                               .......*
05:43:00.038 R 03 03 0C 04 36 41 C5 66 CA 41 54 04 36 41 C5 FE     ....6A.f.AT.6A..
05:43:00.038 R 9C                                                    .
05:44:00.003 T 03 03 00 00 00 06 C4 2A                               .......*
05:44:00.041 R 03 03 0C 8A F2 41 C5 C0 20 41 53 8A F2 41 C5 E2     .....A.. AS..A..
05:44:00.041 R 8F                                                    .
```

This sniff shows that the data logger we are monitoring (a Modbus Client) has sent out a transmission, as indicated by the T in the results.

```
  hit ESC to exit, any other key to renew timeout
05:43:00.003 T 03 03 00 00 00 06 C4 2A                               .......*
05:43:00.038 R 03 03 0C 04 36 41 C5 66 CA 41 54 04 36 41 C5 FE     ....6A.f.AT.6A..
05:43:00.038 R 9C                                                    .
05:44:00.003 T 03 03 00 00 00 06 C4 2A                               .......*
05:44:00.041 R 03 03 0C 8A F2 41 C5 C0 20 41 53 8A F2 41 C5 E2     .....A.. AS..A..
05:44:00.041 R 8F                                                    .
```

In the next screenshot, we can see that a serial device has responded to the transmission request sent by the data logger running Modbus. This response is indicated by the R or received serial traffic. In this case, it represents the Modbus Server response to the Modbus Client request.

```
  hit ESC to exit, any other key to renew timeout
05:43:00.003 T 03 03 00 00 00 06 C4 2A                               .......*
05:43:00.038 R 03 03 0C 04 36 41 C5 66 CA 41 54 04 36 41 C5 FE     ....6A.f.AT.6A..
05:43:00.038 R 9C                                                    .
05:44:00.003 T 03 03 00 00 00 06 C4 2A                               .......*
05:44:00.041 R 03 03 0C 8A F2 41 C5 C0 20 41 53 8A F2 41 C5 E2     .....A.. AS..A..
05:44:00.041 R 8F                                                    .
```

Now that we have confirmed that traffic is being sent and received from our data logger operating as a Modbus Client, and that we are receiving responses, how can we verify that this traffic is indeed Modbus traffic?

The easiest way to recognize Modbus serial traffic (Modbus RTU) is to look for transmissions that start with the Modbus Server address and function code. Similarly, the server's response will also begin with its address and function code. Since we are monitoring the Modbus Client, notice that immediately after the initial T, the Modbus address of the server (03 or "3") is followed by the function code (03 or "3"). This confirms that the traffic we are observing is Modbus traffic.

```
 hit ESC to exit, any other key to renew timeout
05:43:00.003 T 03 03 00 00 00 06 C4 2A                            .......*
05:43:00.038 R 03 03 0C 04 36 41 C5 66 CA 41 54 04 36 41 C5 FE  .... 6A.f.AT.6A..
05:43:00.038 R 9C                                                 .
05:44:00.003 T 03 03 00 00 00 06 C4 2A                            .......*
05:44:00.041 R 03 03 0C 8A F2 41 C5 C0 20 41 53 8A F2 41 C5 E2  .....A.. AS..A..
05:44:00.041 R 8F
```

With Modbus Client, if you are not seeing responses from the Modbus Server, verify your configuration using the first seven steps of this document and ensure that the Modbus Server is configured correctly. This process can be performed for an IP connection as well.

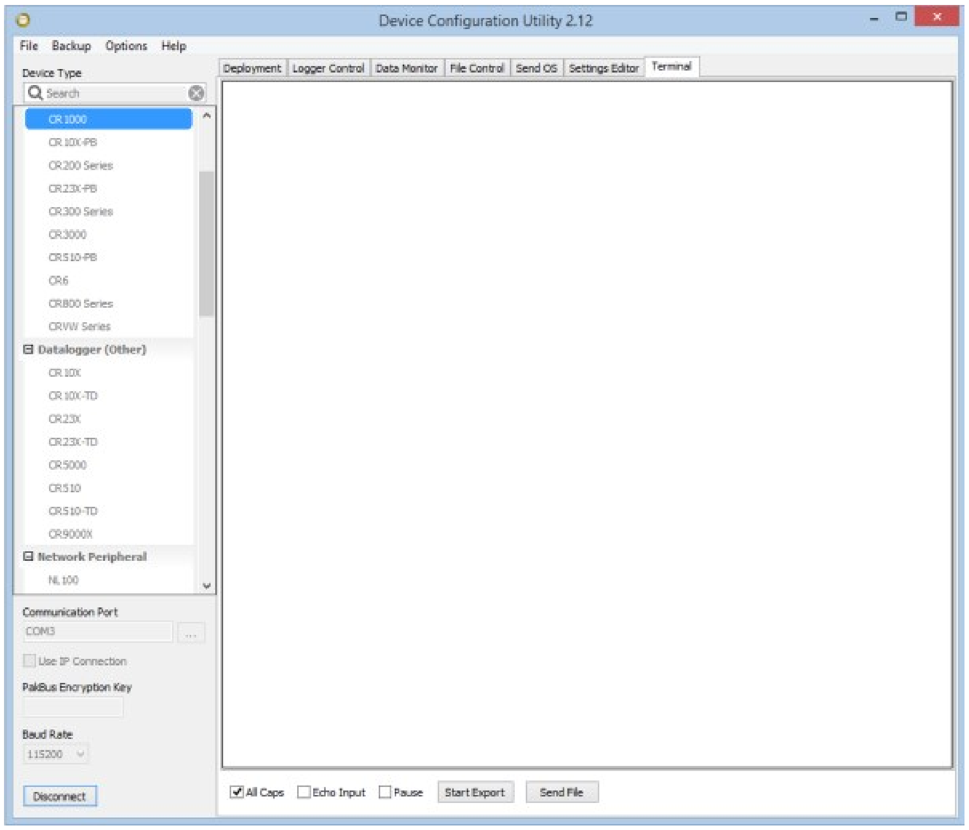
7. Watch Modbus TCP traffic from the data logger terminal mode

This section is derived from a blog post written by Paul Smart available on the Campbell Scientific website here: www.campbellsci.com/blog/diagnose-modbus-communication.

This example uses a CR1000 data logger operating as a Modbus Server. The process is the same for a Modbus Client operating on an IP connection.

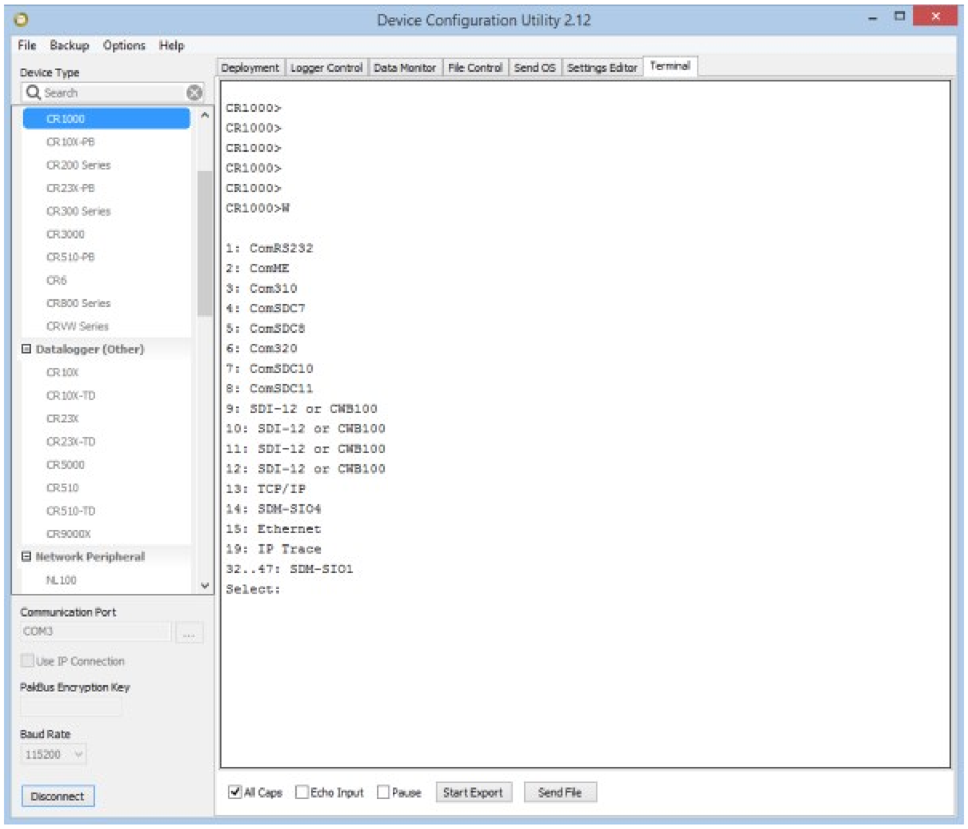
Follow these steps to use ***Device Configuration Utility*** to see the Modbus polls over your IP connection:

1. Open ***Device Configuration Utility*** and connect to your data logger.
2. Click the **Terminal** tab.



3. Press **Enter** on your keyboard until you see a prompt on your screen.

4. Type the capital letter W and press **Enter**.

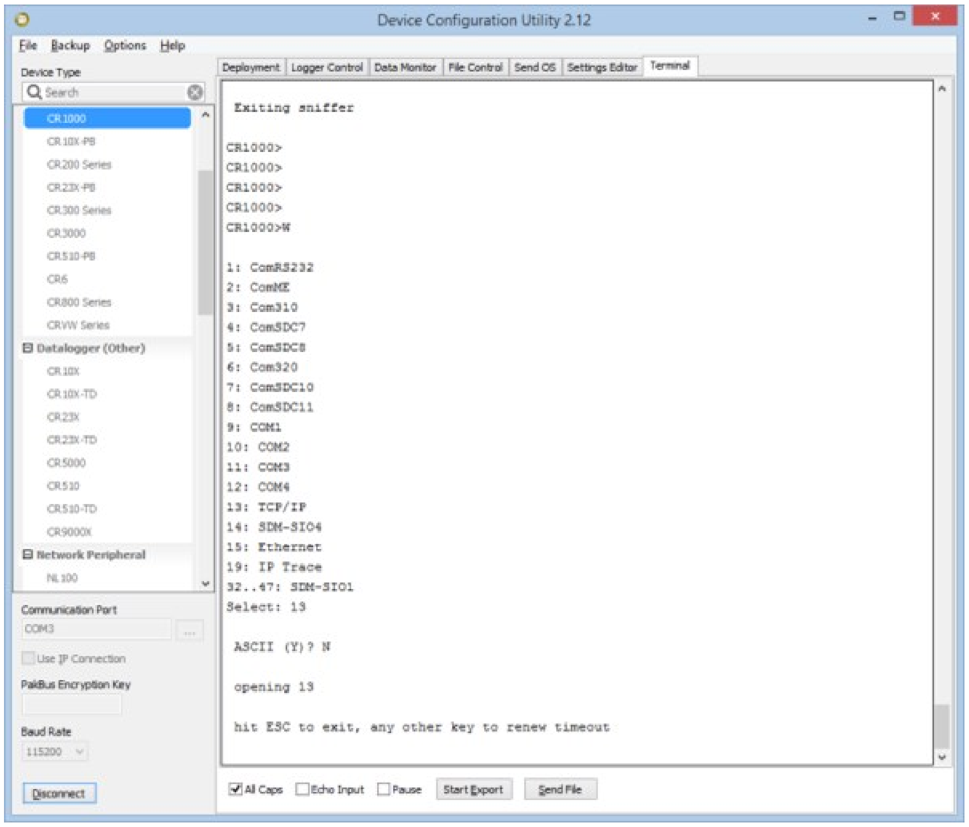


```
CR1000>
CR1000>
CR1000>
CR1000>
CR1000>
CR1000>W

1: ComRS232
2: ComME
3: Com310
4: ComSDC7
5: ComSDC8
6: Com320
7: ComSDC10
8: ComSDC11
9: SDI-12 or CWB100
10: SDI-12 or CWB100
11: SDI-12 or CWB100
12: SDI-12 or CWB100
13: TCP/IP
14: SDM-SIO4
15: Ethernet
19: IP Trace
32..47: SDM-SIO1
Select:
```

5. Select **TCP/IP** and press the **Enter** key.

6. You will be prompted for `ASCII (Y)?`; answer with a capital letter N and press **Enter**.

7. If your data logger is not receiving any Modbus polls, your screen will likely look something like this:



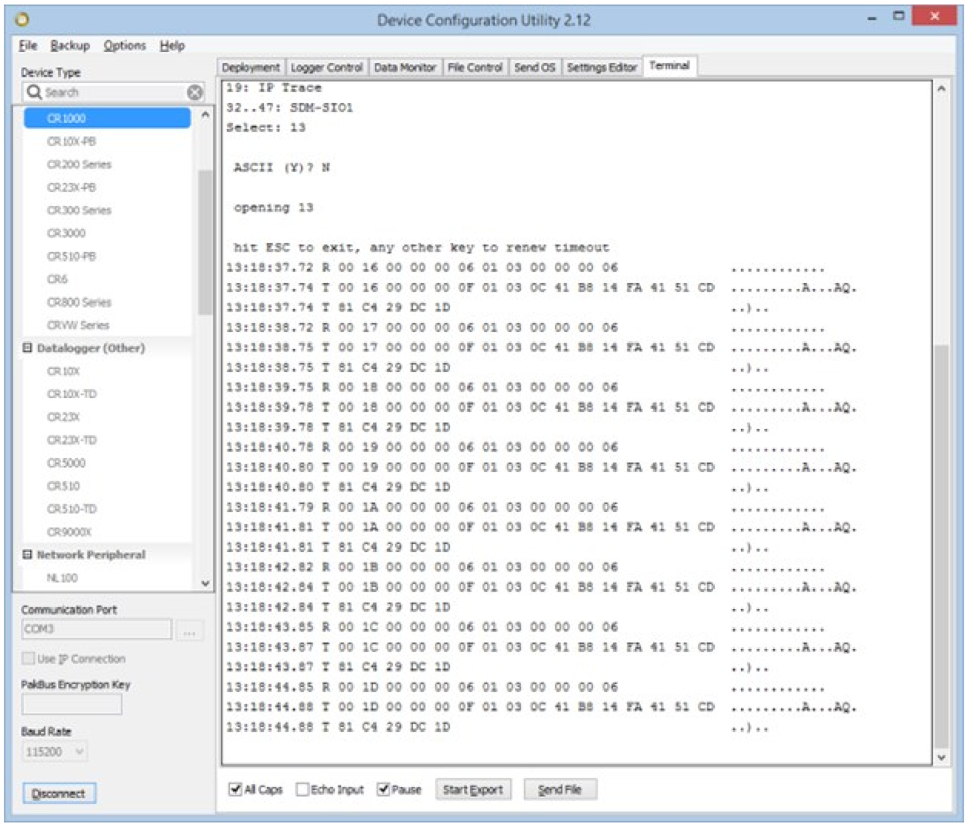
Device Configuration Utility 2.12
File Backup Options Help
Device Type
Deployment Logger Control Data Monitor File Control Send OS Settings Editor Terminal
CR1000
Datalogger (Other)
Network Peripheral
Communication Port
COM3
Use IP Connection
PakBus Encryption Key
Baud Rate
115200
Disconnect
All Caps Echo Input Pause Start Export Send File

```
 Exiting sniffer

CR1000>
CR1000>
CR1000>
CR1000>
CR1000>W

1: ComRS232
2: ComME
3: Com310
4: ComSDC7
5: ComSDC8
6: Com320
7: ComSDC10
8: ComSDC11
9: COM1
10: COM2
11: COM3
12: COM4
13: TCP/IP
14: SDM-SIO4
15: Ethernet
19: IP Trace
32..47: SDM-SIO1
Select: 13

 ASCII (Y)? N

 opening 13

 hit ESC to exit, any other key to renew timeout
```

Note there is no Modbus traffic detected over TCP/IP. The only message on the screen is `hit ESC to exit, any other key to renew timeout`. This scenario could indicate one or more of the following conditions:

- The SCADA system or other Modbus device (Modbus Client) is not polling the data logger (Modbus Server) or the data logger (Modbus Server) is not responding to the Modbus Client's requests.
- The SCADA system or other Modbus device is polling a different IP address.
- The data logger has been assigned an incorrect IP address.
- A cable is not plugged in.
- Modbus traffic is being blocked by the network.

Your data logger may be set up and programmed correctly, but if it is not receiving polling requests from the SCADA client, the data will not be transmitted as expected. In this case, direct your troubleshooting efforts toward the SCADA network, client configuration, and other factors external to the data logger. Note that In such cases, you might observe TCP/IP traffic that is not related to Modbus. For example, you may see PakBus traffic from a ***LoggerNet*** Server if there is a ***LoggerNet***-to-data-logger connection on the network. After network or SCADA system problems are resolved, a successful trace looks like this:



```
19: IP Trace
32..47: SDM-SIO1
Select: 13

 ASCII (Y)? N

 opening 13

 hit ESC to exit, any other key to renew timeout
13:18:37.72 R 00 16 00 00 00 06 01 03 00 00 00 06             ............
13:18:37.74 T 00 16 00 00 00 0F 01 03 0C 41 B8 14 FA 41 51 CD .........A...AQ.
13:18:37.74 T 81 C4 29 DC 1D                                   ..)..
13:18:38.72 R 00 17 00 00 00 06 01 03 00 00 00 06             ............
13:18:38.75 T 00 17 00 00 00 0F 01 03 0C 41 B8 14 FA 41 51 CD .........A...AQ.
13:18:38.75 T 81 C4 29 DC 1D                                   ..)..
13:18:39.75 R 00 18 00 00 00 06 01 03 00 00 00 06             ............
13:18:39.78 T 00 18 00 00 00 0F 01 03 0C 41 B8 14 FA 41 51 CD .........A...AQ.
13:18:39.78 T 81 C4 29 DC 1D                                   ..)..
13:18:40.78 R 00 19 00 00 00 06 01 03 00 00 00 06             ............
13:18:40.80 T 00 19 00 00 00 0F 01 03 0C 41 B8 14 FA 41 51 CD .........A...AQ.
13:18:40.80 T 81 C4 29 DC 1D                                   ..)..
13:18:41.79 R 00 1A 00 00 00 06 01 03 00 00 00 06             ............
13:18:41.81 T 00 1A 00 00 00 0F 01 03 0C 41 B8 14 FA 41 51 CD .........A...AQ.
13:18:41.81 T 81 C4 29 DC 1D                                   ..)..
13:18:42.82 R 00 1B 00 00 00 06 01 03 00 00 00 06             ............
13:18:42.84 T 00 1B 00 00 00 0F 01 03 0C 41 B8 14 FA 41 51 CD .........A...AQ.
13:18:42.84 T 81 C4 29 DC 1D                                   ..)..
13:18:43.85 R 00 1C 00 00 00 06 01 03 00 00 00 06             ............
13:18:43.87 T 00 1C 00 00 00 0F 01 03 0C 41 B8 14 FA 41 51 CD .........A...AQ.
13:18:43.87 T 81 C4 29 DC 1D                                   ..)..
13:18:44.85 R 00 1D 00 00 00 06 01 03 00 00 00 06             ............
13:18:44.88 T 00 1D 00 00 00 0F 01 03 0C 41 B8 14 FA 41 51 CD .........A...AQ.
13:18:44.88 T 81 C4 29 DC 1D                                   ..)..
```

The easiest way to recognize the Modbus TCP traffic and distinguish it from other protocols is that the transmission from the client always starts with an identifier in the first two bytes. In our example, the first poll that was recognized in the trace started with `00 16`. The data logger, in turn, responded with this same unique identifier (`00 16`). The next time the client polled, it used an identifier of `00 17`, and the data logger responded with `00 17`.

NOTE:
There is a difference between Modbus TCP traffic and Modbus RTU traffic. The easiest way to recognize Modbus RTU traffic is to look for a transmission from the client that starts with the Modbus Server address and function code. The server response will also start with its address and function code.

If you see Modbus polls coming from the client (T), but no responses from the data logger (R), it is time to check the configuration and programming of the data logger. You may have an error in your setup such as:

- The data logger is not programmed as a Modbus server.
- The data logger has been given a Modbus server ID that does not match what the client is polling.

At this point, you'll need to examine your data logger setup for further troubleshooting.

8. Troubleshooting basic IP connectivity for Modbus TCP communications

Work with your IT department to confirm that your Modbus devices, including your data logger, are configured correctly and communicating on the network. The following is a list of common troubleshooting steps to help determine and fix network connectivity issues.

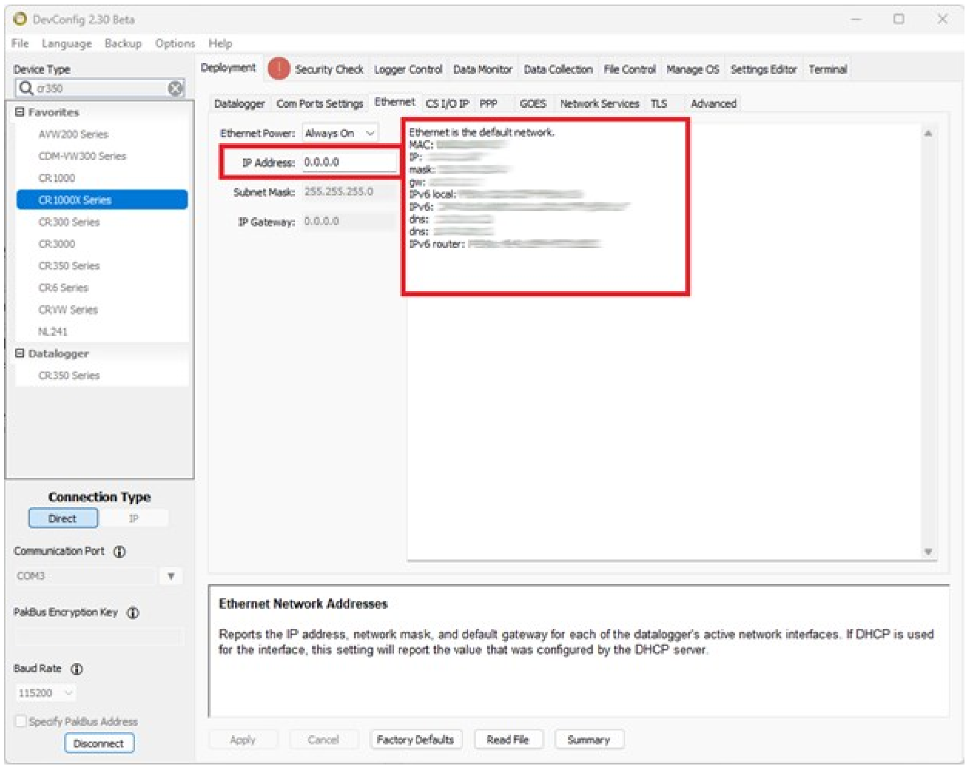
1. Ensure that the data logger IP address, Subnet Mask, and Gateway Address are correct.
2. If the data logger is polling a Modbus Server, verify the Modbus Server address matches the Modbus address in your data logger program.
3. Check the IP address, Subnet Mask, and Gateway Address configured on other Modbus devices.
4. Examine the Ethernet cable to ensure it is plugged in or ensure that the WiFi on both devices is connected properly.
5. Look at the Ethernet LEDs near the data logger Ethernet port. A green light and a flashing amber light indicate traffic is passing between the device and the network.
6. If the data logger is directly connected to the other Modbus device using Ethernet, set the Ethernet **Power** setting to **Always on** to keep the interface from going to sleep. This setting is found here: ***Device Configuration Utility*** > **Deployment** > **Ethernet** > **Ethernet Power**.

7. To check if the data logger and the other Modbus device are communicating, connect a computer to the same network subnet and try to ping the IP address of both devices from the Command Prompt of your computer.

   > **NOTE:**
   > For security, the data logger ping function is turned off by default. This setting is found here: ***Device Configuration Utility*** > **Deployment** > **Network Services** > **Ping (ICMP) Enabled**.

8. If the data logger is connected to the network, set its IP address in the **Ethernet** tab (or releveant IP interface) to **0.0.0.0** and apply the change. Setting 0.0.0.0 as the IP address enables DHCP on the data logger. If the network is assigning addresses using DHCP, the data logger will receive an IP address, which will appear in the status area of the **Ethernet** tab (or the corresponding IP interface) in the ***Device Configuration Utility***. When the data logger receives an address from the network, compare it with your existing address, subnet mask, and gateway address. The subnet mask and gateway address should match exactly, and the IP address should be very similar.

   > **NOTE:**
   > If after a few minutes, the data logger receives an IP address that starts with 169.254, this indicates that the network is not assigning addresses using DHCP. This may also mean that the data logger and computer are directly connected to each other.



Ethernet Power: Always On

IP Address: 0.0.0.0

Subnet Mask: 255.255.255.0

IP Gateway: 0.0.0.0

```
Ethernet is the default network.
MAC: [redacted]
IP: [redacted]
mask: [redacted]
gw: [redacted]
IPv6 local: [redacted]
IPv6: [redacted]
dns: [redacted]
dns: [redacted]
IPv6 router: [redacted]
```

Ethernet Network Addresses

Reports the IP address, network mask, and default gateway for each of the datalogger's active network interfaces. If DHCP is used for the interface, this setting will report the value that was configured by the DHCP server.

Example results follow:

Expected results:

IP Address: 172.91.23.45

Subnet Mask: 255.255.0.0

Gateway Address: 172.91.25.100

Results from **Status** area of **Ethernet** tab:

IP: 192.168.1.2

Mask: 255.255.255.0

GW: 192.168.1.1

The addresses 192.168.1.2 and 172.91.23.45 are from different networks and cannot communicate with each other. If your data logger receives an address like this, contact your IT department to request a new IP address and inform them that the network configuration is incorrect.

Appendix A. CR1000X example program for ModbusClient() over TCP/IP (Modbus TCP)

This program reads in two floating point values from the Modbus Server at IP Address 192.168.1.1 every 60 seconds and writes them to the ModbusData floating point variable. The ModbusClient() instruction is contained in a SlowSequence so as to allow measurement instructions to operate in the scan without being slowed down.

```
Public Temp, batt_volt, ModResult
Public ModbusData(2)
Public TCPSocket

'Main Program
BeginProg
  Scan (2,Sec,0,0)
    Battery(batt_volt)  'Simple diagnostic instructions to record
    PanelTemp(Temp)

  NextScan
  SlowSequence
  Scan (60,Sec,3,0)
    If TCPSocket = 0 Then
      TCPOpen ("192.168.1.1",502,1,,TCPSocket)
      'Check every time if the socket is still open. If not, reopen the socket and
      'assign the handle to a variable called TCPSocket.
    EndIf
    'Utilize the handle from the TCPOpen instruction as the COM port and read in
    'values from the Modbus server. These values are stored in the ModbusData
    'variable.
    ModbusClient (ModResult,TCPSocket,-9600,3,3,ModbusData(),1,2,3,200,0)
  NextScan
EndProg
```

Appendix B. CR1000X example program for ModbusClient() over serial (Modbus RTU)

This program is a modified version of the ModbusClient() example code in CRBasic. The code uses SerialOpen to specify certain serial configurations and ModbusClient() to query the Modbus Server over RS232.

```
'Declare Public Variables
Public PTemp, batt_volt,ModbusData(20),Result

'Define Data Tables
DataTable (Test,1,-1)
  DataInterval (0,15,Sec,10)
  Minimum (1,batt_volt,FP2,0,False)
  Sample (1,PTemp,FP2)
  Sample (1,ModbusData(),FP2)
EndTable

'Main Program
BeginProg

  SerialOpen (COMRS232,115200,1,0,50)

  Scan (1,Sec,0,0)
    PanelTemp (PTemp,15000)
    Battery (Batt_volt)

    'Retrieve Modbus Data
    ModbusClient (Result,COMRS232,115200,3,3,ModbusData(),1,10,3,100)

    'Enter other measurement instructions
    'Call Output Tables
    CallTable Test
  NextScan
EndProg
```

Appendix C. Serial formatting

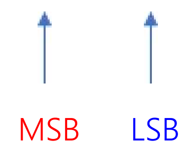
Communications port and Modbus configuration can be diverse among manufacturers. This section describes what settings you should look out for while reading a third party sensor using `ModbusClient()`.

1. **Big-endian vs. little-endian**

   The two types of byte order, little-endian and big-endian, originates from Gulliver's travels. The two factions of Lilliput were divided by whether they broke the shell of an egg on the big side or the little side. The concept for byte-order is similar, where little-endian would list the most significant byte on the right side and big-endian lists the most significant byte on the left side. For example, the decimal number of 39,534 using the traditional English reading and writing, is written in big-endian binary with the byte that changes the decimal value more severely on the left side.

   Big-endian

   1001101001101110

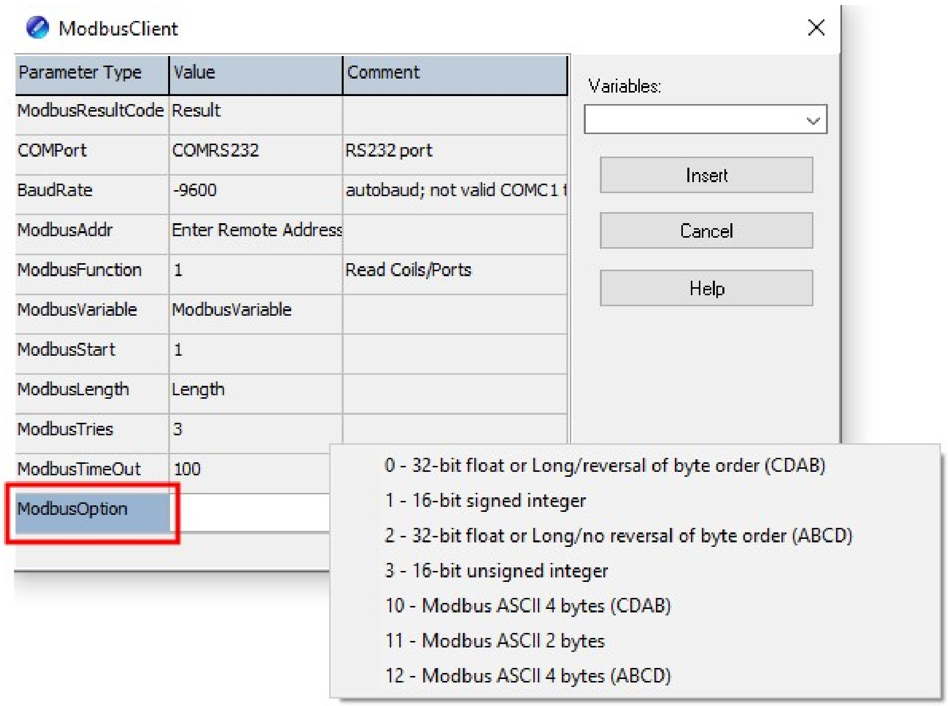


   Little-endian flips this order with the leftmost byte being the least significant and the rightmost byte being the most significant.

   Little-endian

   1001101001101110

   How does this all relate to Modbus? Modbus utilizes the same concept of endianness when it comes to the format in which the data is transmitted. This can be seen in the `ModbusOption` parameter within the `ModbusClient()` instruction.



ModbusClient

Parameter Type	Value	Comment
ModbusResultCode	Result	
COMPort	COMRS232	RS232 port
BaudRate	-9600	autobaud; not valid COMC1 t
ModbusAddr	Enter Remote Address	
ModbusFunction	1	Read Coils/Ports
ModbusVariable	ModbusVariable	
ModbusStart	1	
ModbusLength	Length	
ModbusTries	3	
ModbusTimeOut	100	
ModbusOption		

0 - 32-bit float or Long/reversal of byte order (CDAB)
1 - 16-bit signed integer
2 - 32-bit float or Long/no reversal of byte order (ABCD)
3 - 16-bit unsigned integer
10 - Modbus ASCII 4 bytes (CDAB)
11 - Modbus ASCII 2 bytes
12 - Modbus ASCII 4 bytes (ABCD)

Modbus register data are transmitted in hexadecimal, which means each hex digit represents a 4-bit pattern. Therefore, two hex digits are 1 byte, hence little-endian being represented as CDAB and big-endian as ABCD. The default for most Modbus servers is little-endian.

2. **Logic, parity, data bits, and stop bits**

   Modbus has layers of communications parameters that must be configured correctly. Logic, parity, data bits, and stop bits are all on the serial communications level.

   a. **Logic**

      The logic level determines whether a binary 0 is represented as a low or high voltage. This is determined with either a pull-up or pull-down resistor of some kind. This resistor ensures that, when there is no signal, a floating value is either connected to a low or high voltage source at all points in time to reduce errors. In simple terms, logic 1 low means that a 0 is formed from a low voltage level and logic 1 high means that 0 is formed from a high voltage level. True RS-232 typically uses logic 1 low, TTL typically uses logic 1 high.

   b. **Parity**

      Parity is a form of error checking for frames that come in. Most devices do not use parity; however, some utilize either odd or even parity. If parity is used, both the transmitter and receiver count the amount of binary 1s in the frame. Odd parity will contain an odd number of 1s in the frame. Even parity will contain an even number of 1s in the frame. An additional bit will be added to ensure the "even" or "odd" parameter is met.

Example:

Parity	Original byte	Modified byte
Odd	00101101 (four 1s)	001011011 (five 1s)
Even	00101101 (four 1s)	00101101 (four 1s)

c. **Data bits**

Data bits determine how many bits there are in a singular frame without parity and without the start and stop bits. Usually, this value is 8.

d. **Stop bit**

Stop bits are utilized to signal the end of a frame. This can be important to be able to differentiate one packet from the next by having a start bit as "low" and a stop bit as "high". This means that the change of logic levels indicates the beginning and end of a frame respectively. Usually, 1 stop bit is utilized in serial communications.

3. **Register types**

Register types are determined by their address within the register map, this is denoted as the Modicon notation.

Coil = 00001 to 09999

Discrete input = 10001 to 19999

Input register = 30001 to 39999

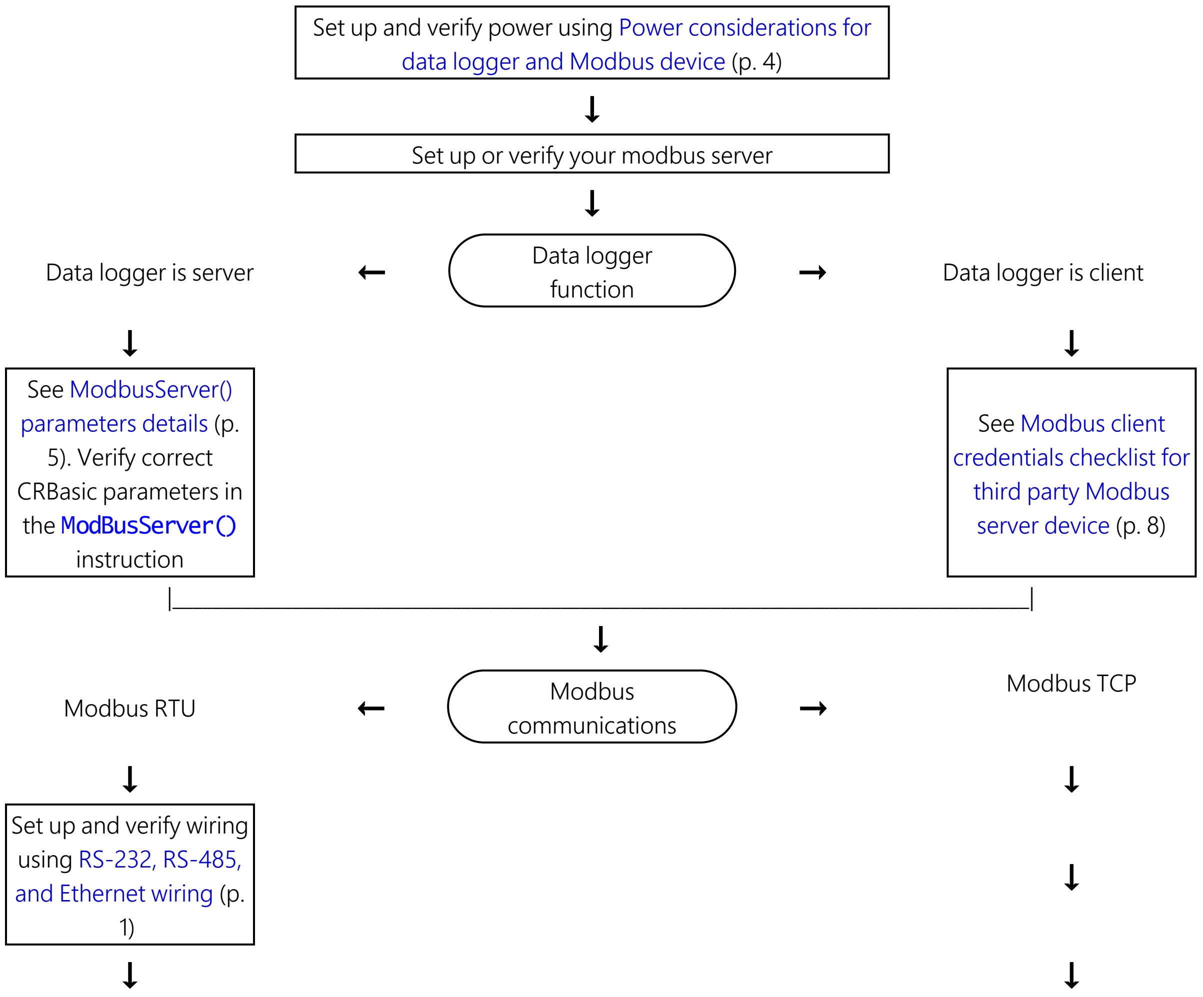
Holding register = 40001 to 49999

Holding registers are the most common if you are looking to read values from the server.

4. **Modbus address of the Modbus device**

Modbus addresses can range from 0 to 65,535 and should be listed on your Modbus device documentation. This must be entered in the forth parameter of the `ModbusClient()` instruction.

Appendix D. Modbus setup flowchart



Set up and verify power using Power considerations for data logger and Modbus device (p. 4)

Set up or verify your modbus server

Data logger function

Data logger is server

See ModbusServer() parameters details (p. 5). Verify correct CRBasic parameters in the `ModBusServer()` instruction

Data logger is client

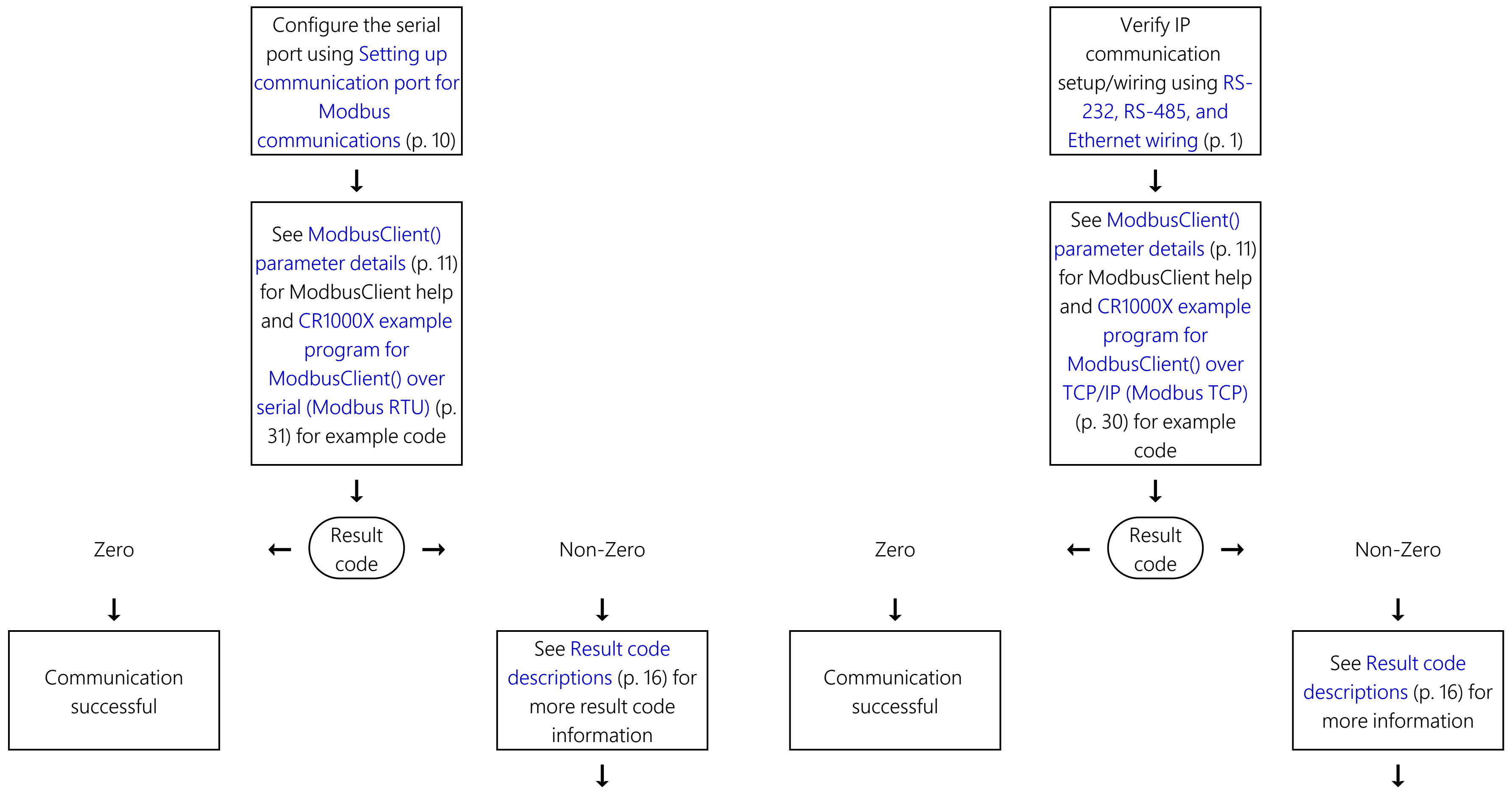
See Modbus client credentials checklist for third party Modbus server device (p. 8)

Modbus communications

Modbus RTU

Set up and verify wiring using RS-232, RS-485, and Ethernet wiring (p. 1)

Modbus TCP



Configure the serial port using Setting up communication port for Modbus communications (p. 10)

See ModbusClient() parameter details (p. 11) for ModbusClient help and CR1000X example program for ModbusClient() over serial (Modbus RTU) (p. 31) for example code

Communication successful

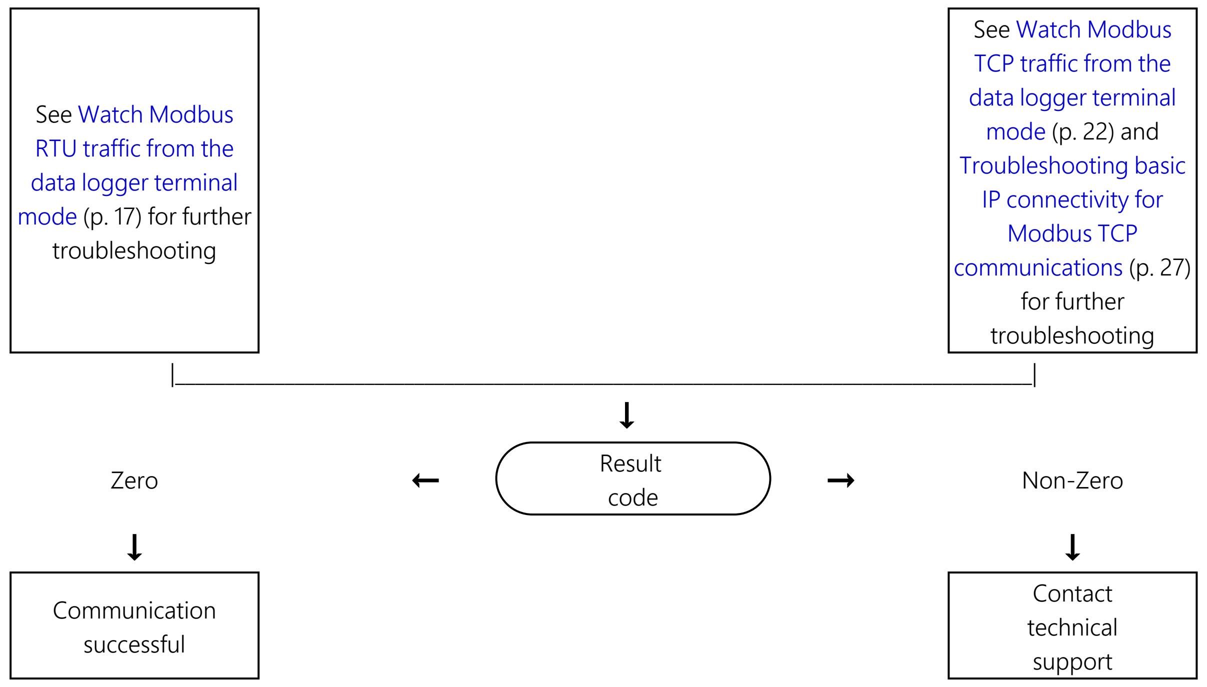
See Result code descriptions (p. 16) for more result code information

Verify IP communication setup/wiring using RS-232, RS-485, and Ethernet wiring (p. 1)

See ModbusClient() parameter details (p. 11) for ModbusClient help and CR1000X example program for ModbusClient() over TCP/IP (Modbus TCP) (p. 30) for example code

Communication successful

See Result code descriptions (p. 16) for more information


See Watch Modbus RTU traffic from the data logger terminal mode (p. 17) for further troubleshooting
See Watch Modbus TCP traffic from the data logger terminal mode (p. 22) and Troubleshooting basic IP connectivity for Modbus TCP communications (p. 27) for further troubleshooting
Zero
Result code
Non-Zero
Communication successful
Contact technical support

Global Sales and Support Network

A worldwide network to help meet your needs

Campbell Scientific Regional Offices

Australia

Location:	Garbutt, QLD Australia
Phone:	61.7.4401.7700
Email:	info@campbellsci.com.au
Website:	www.campbellsci.com.au

Brazil

Location:	São Paulo, SP Brazil
Phone:	11.3732.3399
Email:	vendas@campbellsci.com.br
Website:	www.campbellsci.com.br

Canada

Location:	Edmonton, AB Canada
Phone:	780.454.2505
Email:	dataloggers@campbellsci.ca
Website:	www.campbellsci.ca

China

Location:	Beijing, P. R. China
Phone:	86.10.6561.0080
Email:	info@campbellsci.com.cn
Website:	www.campbellsci.com.cn

Costa Rica

Location:	San Pedro, Costa Rica
Phone:	506.2280.1564
Email:	info@campbellsci.cc
Website:	www.campbellsci.cc

France

Location:	Montrouge, France
Phone:	0033.0.1.56.45.15.20
Email:	info@campbellsci.fr
Website:	www.campbellsci.fr

Germany

Location:	Bremen, Germany
Phone:	49.0.421.460974.0
Email:	info@campbellsci.de
Website:	www.campbellsci.de

India

Location:	New Delhi, DL India
Phone:	91.11.46500481.482
Email:	info@campbellsci.in
Website:	www.campbellsci.in

Japan

Location:	Kawagishi, Toda City, Japan
Phone:	048.400.5001
Email:	jp-info@campbellsci.com
Website:	www.campbellsci.co.jp

South Africa

Location:	Stellenbosch, South Africa
Phone:	27.21.8809960
Email:	sales@campbellsci.co.za
Website:	www.campbellsci.co.za

Spain

Location:	Barcelona, Spain
Phone:	34.93.2323938
Email:	info@campbellsci.es
Website:	www.campbellsci.es

Thailand

Location:	Bangkok, Thailand
Phone:	66.2.719.3399
Email:	info@campbellsci.asia
Website:	www.campbellsci.asia

UK

Location:	Shepshed, Loughborough, UK
Phone:	44.0.1509.601141
Email:	sales@campbellsci.co.uk
Website:	www.campbellsci.co.uk

USA

Location:	Logan, UT USA
Phone:	435.227.9120
Email:	info@campbellsci.com
Website:	www.campbellsci.com