CO2Meter.com
131 Business Center Dr. Unit A-3, Ormond Beach, FL 32174
TEL: 877.678.4259 | FAX: 866.422.2356
EMAIL: sales@co2meter.com | www.co2meter.com

AN117 – MODBUS Implementation Guidelines

Several CO2Meter products support RS-485 Modbus protocol. Modbus allows a single controller to support multiple sensor devices attached to a multi-point network of devices. Modbus support is complimentary to CO2Meter's proprietary command protocol which functions solely in point-to-point mode.

This document describes the CO2Meter implementation of a slave device together with examples.

Deviations from Official Protocol

The official protocol can be found on modbus.org. In order to work around hardware limitations minor deviations were required from the official standard. When connecting to the Modbus device the following communication settings must be used:

Baud Rate	115200bps
Flow Control	None
Data Format	8 data bits, 1 start bit NO Parity, 1 stop bit

Modbus Connections are made over the two wire RS-485, via the MCRX and MCTX pins. A USB to RS-485 cable adapter is offered on our website for purchase. It is also possible to use the protocol in point to point fashion over the built in USB port on platforms that have this feature. Note that the built-in USB can function only with point to point communication of short distances of a few meters. RS-485 will allow for extremely long runs of cable and integration with other RS-485 devices.

Implementation

Modbus RTU implementation is utilized with CO2Meter's products, using the compact byte-level formatting:

	MODBUS Serial PDU		
Adddress field (1 byte)	Function Code (1 byte)	Data	CRC (Hi and Low)

The device supports a configurable address, with address 0xFE reserved as an "Any Sensor". The sensor requires a CRC-16 checksum to be calculated for requests and it is expected that host devices will validate the checksum of responses.

Address 0xCC is also reserved for proprietary communication protocols. If this device is attached to a MODBUS network care must be taken to ensure no device on the network is using address 0xCC, for it will interfere with the ability of the CO2Meter device to properly function. The 0xFE address should only be used when testing with a single slave device. Since any sensor will respond to this address, multiple sensors will interfere with each other preventing successful communications.

Selection of RS-485 vs. UART Communication

Devices that support both built-in USB/UART and RS-485 communication will dynamically chose which hardware communication method to use upon startup. This will result in the potential loss of the first few communication packets. Ensure that any host communication software has sufficient retry logic built into it. Notice that USB/UART and RS-485 may not be used at the same time. The device must be power cycled to switch hardware communication methods.

Note that the USB/UART DTR (Data Terminal Ready) line functions as a reset to the board. Accordingly, if the software that opens the port asserts DTR, the board will be held in reset.

Timing

The characters that comprise a Modbus PDU must be sent in a continuous fashion. Any pause beyond a few character times is interpreted as the end of the PDU. It is not possible to use a terminal program (e.g. Hyperterminal or PuTTY) to generate Modbus packets.

131 Business Center Dr. Unit A-3, Ormond Beach, FL 32174
TEL: 877.678.4259 | FAX: 866.422.2356
EMAIL: sales@co2meter.com | www.co2meter.com

When the slave device receives a valid PDU addressed to it with valid CRC it will respond within 100ms. Any short or invalid PDU will be silently ignored.

Supported Commands

The CO2Meter MODBUS implementation supports three of the official commands, to allow for reading of sensor values, and reading/writing of command bytes.

Error Codes

Command not recognized or malformed

If the command is not recognized the device will not return any exception message. The command will silently be ignored.

Out of range register access

If the command tries to read inaccessible registers an exception response PDU will be returned.

Function Code	1 byte	Function Code + 0x80
Exception Code = Illegal Data Address	1 byte	0x02

Read Holding Register (0x03)

This command is to read configuration and settings registers from the device.

Quantity of registers is limited to 8. In practice the number of holding registers is usually smaller.

Request PDU

Function Code	1 byte	0x03
Starting Address	2 bytes	1-based Address
Quantity of Registers	2 bytes	Quantity of Registers

Reponse PDU

Function Code	1 byte	0x03
Byte Count	1 byte	2 * N
Quantity of Registers	2 bytes	Quantity of Registers

Read Input Register (0x04)

This command is to read sensor values from the device.

Quantity of registers is limited to 8. In practice the number of holding registers is usually smaller. The register numbers correspond to the sensor configuration register in configuration utilities.

Request PDU

Function Code	1 byte	0x04
Starting Address	2 bytes	1-based Address
Quantity of Registers	2 bytes	Quantity of Registers

Reponse PDU

Function Code	1 byte	0x04
Byte Count	1 byte	2 * N
Quantity of Registers	2 bytes	Quantity of Registers

131 Business Center Dr. Unit A-3, Ormond Beach, FL 32174
TEL: 877.678.4259 | FAX: 866.422.2356
EMAIL: sales@co2meter.com | www.co2meter.com

Write Single Holding Register (0x06)

This command is to read sensor values from the device.

Quantity of registers is limited to 8. In practice the number of holding registers is usually smaller. The register numbers correspond to the sensor configuration register in configuration utilities.

Request PDU

Function Code	1 byte	0x06
Address	2 bytes	1-based Address
Value	2 bytes	16-bit Register Value

Reponse PDU (echo of request)

Function Code	1 byte	0x06
Address	2 bytes	1-based Address
Value	2 bytes	16-bit Register Value

Example Command Sequences

These examples assume you have a single slave device attached to the MODBUS network. We will be using the "Any Sensor" MODBUS address for communication. A RS-485 communication cable is available on the CO2 Meter website for purchase.

Read Sensor ID 1 Sequence:

In this example an AQ500 board is used with a K30 sensor attached and configured in the first sensor configuration register. The sensor reports CO2 as a signed 16-bit integer with unity scaling.

Master Transmit:
0xFE 0x04 0x00 0x01 0x00 0x01 0x74 0x05

Slave Reply:
0xFE 0x04 0x02 0x01 0x90 0xAC 0xD8

In this reply we notice that the slave returned 2 bytes, 0x01 and 0x90. These bytes combine to form a single signed integer of 0x190, equivalent in decimal notation to 400. Because the sensor utilizes unity scaling we know this to be 400ppm.

Testing

Modbus testing can be done with a free utility called modpoll http://www.modbusdriver.com/modpoll.html.

Modpoll will work correctly only with an RS485 connection. This is because the program asserts the DTR signal which will reset the device.

The following is an example command:

```
c:\modpoll.3.1\win32>modpoll -b 115200 -p none -a 254
-t 3 -r 1 -c 3 -1   com5
modpoll 3.1 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright (c) 2002-2011 proconX Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: Modbus RTU
Slave configuration...: address = 1, start reference = 1, count = 3
```

131 Business Center Dr. Unit A-3, Ormond Beach, FL 32174
TEL: 877.678.4259 | FAX: 866.422.2356
EMAIL: sales@co2meter.com | www.co2meter.com

Communication.....: com5, 115200, 8, 1, none, t/o 1.00 s, poll rate 1000 ms
Data type.....: 16-bit register, input register table

-- Polling slave...
[1]: 0
[2]: 6
[3]: 0

Note that above example uses com5 and the modbus address of 0xfe (254). You will have to replace this with the correct comport on your system.