MODBUS-RTU Applied to the XR10CX Control

WATER HEATER DIGITAL OPERATING CONTROLLER

PV500-67 06/2014

Table of Contents

1. THIS MANUAL

This is a supplemental manual to cover the communication interface capabilities of the XR10CX thermostatic module using Modbus RTU. This manual will provide the particulars for the XR10CX, how the Modbus RTU protocol is implemented on the XR10CX, and how to interface with the XR10CX. For details on how to operate or configure the XR10CX, please refer to the XR10CX Manual.

This manual is not a complete or definitive guide to Modbus RTU communication. For detailed MODBUS and Modbus RTU information, consult other sources such as (www.modbus.org).

2. MODBUS RTU:

2.1 OVERVIEW:

Modbus RTU This is a communication over twisted pair from a Master device (also called a client) to multiple slave devices (also called servers). The master will send command to a particular slave. The slave will in turn process this command and respond to the Master. All communication is initiated by the Master. The XR10CX is a slave device and will not “talk” until a Master device talks to it first. Modbus RTU is widely used within Building Management Systems (BMS) and Industrial Automation Systems (IAS). This wide acceptance is due in large part to Modbus RTU’s ease of use. Modbus RTU is a low level communications that contain no unit type, data scaling, or data description. Simply put, it is a list of addresses that can be read and/or written to. The type of data will have to be known by the receiving device in order for the data to be useful. Data types and available registers are usually provided as device documentation register list.

2.1.1 DEFINITIONS:

MODBUS Protocol: A messaging structure used for communication between devices, machines, sensors, and/or computers.

Modbus-RTU: (Remote Terminal Unit) Implementation of the Modbus protocol on top of a serial line with an RS-232, RS-485 or similar physical layer. The XR10CX uses 2-wire RS-485 Physical Layer and implements Modbus RTU.

Master Device: Also known as Client, this device initiates all communication on the RS-485 network. The Master will send commands to Slave Devices.

Slave Device: Also known as Server, this device will respond only when addressed by the Master device. When the Master sends the slave a command, it will perform the command and respond back to the Master with the data requested, or if no data is required, then it will simply echo the command.

Slave Address: Each slave device in a network is assigned a unique address from 1 to 247. When the Master requests data, the first byte it sends is the Slave address. This way each slave knows after the first byte whether or not to ignore the message.

RS-485 (EIA-485): A 2 wire (twisted pair) multi drop network. Each device can send data by holding positive and negative voltage on the line and reversing polarity on the 2 wires. When no devices are transmitting, the line will be tri-state. The recommended arrangement of the wires is as a connected series of point-to-point (multi-dropped) nodes, i.e. a line or bus, not a star, ring, or multiply connected network. The number of devices that can be connected to a single line is defined in the RS-485 standard by the input impedance of 32 UNIT LOADs. The wire and circuits interfacing on this 2 wire connection is considered the PHYSICAL LAYER. (RS-485 is the same physical layer as used with BACNET MSTP.)

Line Termination: (LT) On RS-485, ideally the two ends of the cable will have a termination resistor connected across the two wires. This helps with reducing noise and interference on high speed and long line lengths. In practice, it is not needed in low speed and short line lengths. When needed, adding a 150ohm resistor across the line at the end devices will reduce electrical noise and reflections.

Biasing Resistors: One device on the RS-485 line should have biasing resistors. This holds the line in a known state when no devices are transmitting (talking). Typically this would be the Master device.

XR10CX Implementation: Modbus RTU protocol, over RS-485 (EIA-485) physical layer using 2 wires. Each XR10CX should be considers 1 UNIT LOAD for the network loading. The XR10CX devices do not have biasing resistors. XR10CX will operate in slave mode and will only transmit when addressed by the Master device.

Registers: (Holding Registers) Each Register is 1 word defined as 2 bytes or 16 bits. See Table 2 for details of each Register. Holding registers are typically READ/WRITE but can be READ ONLY or WRITE ONLY. The XR10CX uses this type of register. All Registers are in the range of 40001-4999.

Coils: (Output Coils or Boolean Variables) The XR10CX support and uses Output Coils. Each coil is 1 bit and can be read or written to. Despite the name, they can be digital outputs, Boolean variables, or digital inputs.

Register Numbers / Register Addresses: Modbus can be confusing when referencing registers and their addresses. Modbus documentation is not consistent and terms are used interchangeably when they have different meanings. Register Addresses always start at 0 and count up, and Register Numbers are the actual number/name of the Register. Example the first Holding Register is 40001. This would be address 0. For each data type, the address would start at 0 again. To avoid confusion, this manual will reference Register Numbers only. Any reference to a Register less than 40000 simply needs the 40000 added to it. Setpoint ST1 is Register 769 or the actual Register Number 40769. For a list of all available registers, see Table 2.

Input Registers, Discrete Inputs and Coils: MODBUS defines several types of data. The only data used by the XR10CX ® is the Holding Register (40001 – 49999) and Output Coils (00001 – 09999). The XR10CX does not use Input Registers, or Inputs Coils.

CRC ERROR CHECKING: Modbus-RTU includes an error-checking field that is based on a Cyclical Redundancy Checking (CRC) method performed on the message contents. The CRC field checks the contents of the entire message. The CRC field contains a 16-bit value as the last 2 bytes in any message. The low-order byte of the field is appended first, followed by the high-order byte. The CRC value is calculated by the sending device, which appends the CRC to the message. The receiving device recalculated a CRC during receipt of the message, and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, the message will be ignored. Most Energy Management software packages will automatically calculate the CRC values as a normal part of the protocol.

2.2 COMMUNICATION DETAILS:

The MODBUS application data unit is built by the Master (client) that initiates a MODBUS transaction. The function indicates to the slave (server) what kind of action to perform. The function code field of a MODBUS data unit is coded in one byte. When a message is sent from a Master to a Slave device the function code field tells the Slave what kind of action to perform. The message contains information that the Slave uses to take the action defined by the function code. If no error occurs in receiving the message from the Master, the slave will respond to the Master. The response (Slave to Master) message contains the data requested, if no data was requested, it will echo the Master's command. See Figure 1 for a diagram of a successful communication. If the slave is unable to execute the command (example: invalid command, unreachable address), the message contains an error code and an exception code. See Figure 2 for diagram of an exception response. See Table 1 for a list of exception codes.

2.2.1 Normal Communication diagram

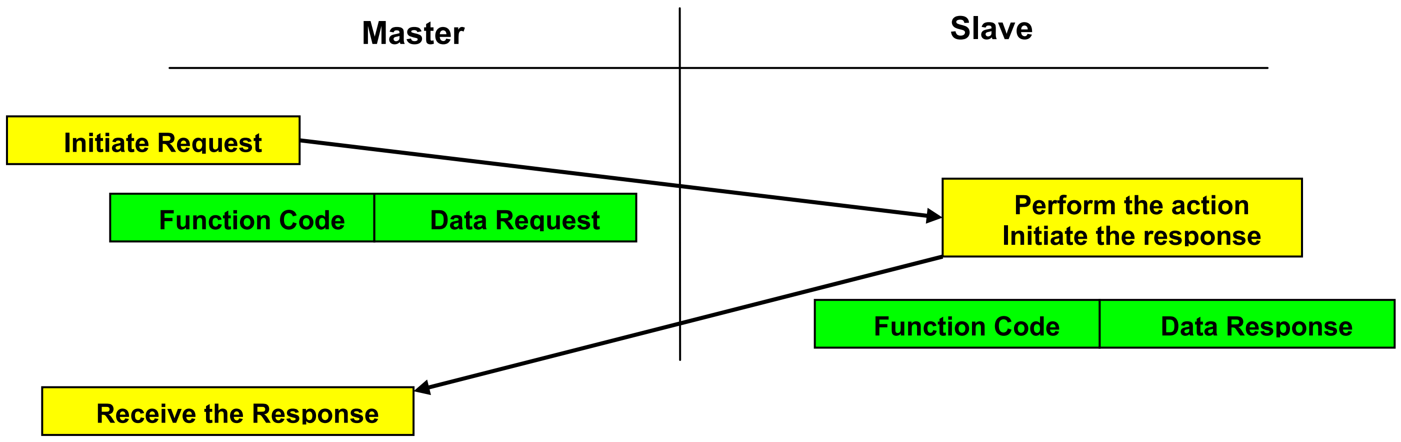

Figure 1
MODBUS Transaction (Error Free)

2.2.2 Communication with exception code

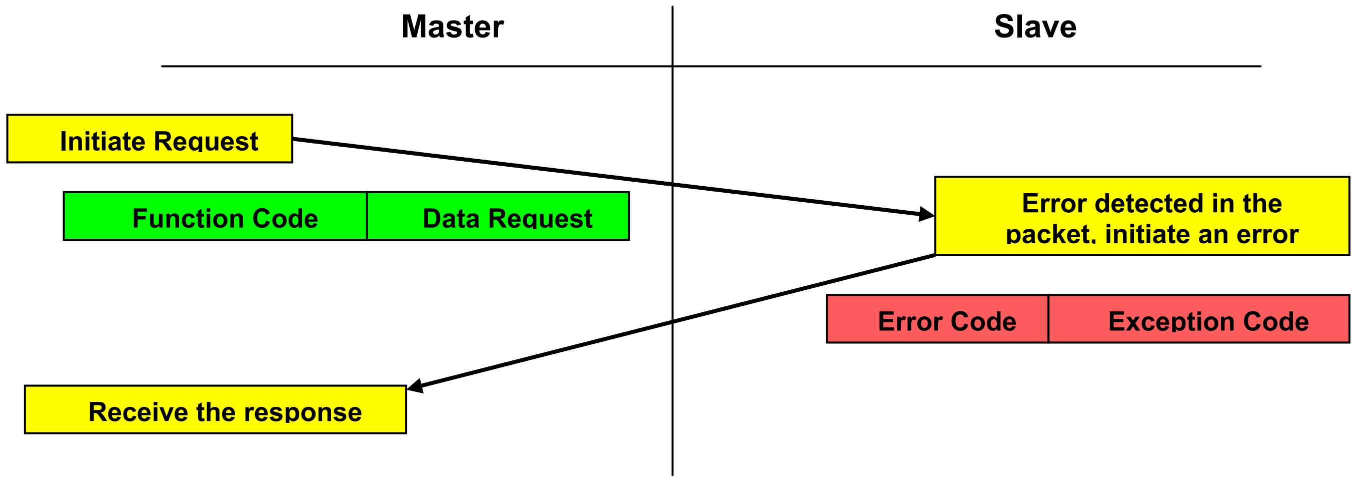

Figure 2 MODBUS Transaction (Error, Exception Code response)

2.3 SEQUENCE OF MODBUS RTU COMMAND AND RESPONSE:

1) Master (Automation System) Sends command to Slave devices (XR10CX)
2) After transmitting, the Master turns off the line driver and listens
3) All Slave Devices receives command. (They are in listen mode)
    a) If address matches Slave address and the CRC is valid, that Slave will process command.
        i) If command was successfully processed
            (1) Slave responds with its own slave address, and echoes the command it received, and includes any data that may have been requested by the Master.
            (2) After sending, the Slave goes back to listen mode.
        ii) If command was not successfully processed
            (1) Slave will respond with and exception response and exception code indicating why it was not processed.
                (a) Possible caused would be invalid address. See Table 1 Exception Codes
            (2) After sending, The Slave goes back to listen mode.
    b) If the command received fails the CRC validation, no response is given.
    c) If the command received does not match the Slave address, no response is given.
    d) If the command is incomplete, no response is given.
    e) If no command is received, no response is given.

3. COMMANDS (MESSAGES)

Modbus standard has several commands used to access different types of data. The XR10CX only uses the data type called Holding Registers (40001 – 49999) and Output Coils (00001 – 09999). There are only 3 commands needed to access the Holding Registers, and 2 commands for the Output Coils. The Modbus specification allow reading and writing to multiple sequential registers, but the maximum number of registers that can be read or written to in one command is 5. The limit for reading Coils is 80 and writing coils is only 1 at a time. This is a limitation of the XR10CX device. In the following commands, the actual address will be used, the command indicates the type of register it is. For example, the first Register 40001 will have address 0 and the first Coil 00001 will also have address 0.

3.1 READ OUTPUT COIL (0x01):

Read Coils from device. Coil status will be packed into the data bytes in the response. Only requested Coil data is valid and the extra bits that may be present for padding the bytes should be ignored.

3.1.1 MASTER COMMAND TO READ COILS (0x01):

Slave Address	Function Code 0x01	Coil Address (MSByte)	Coil Address (LSByte)	Number of Coils (MSByte)	Number of Coils (LSByte)	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): Device address that receives the command. Range: 1-247.
Function Code: (1 byte): Code = 0x01 (Read Output Coil).
Coil Address: (2 bytes): The address of the first coil to be read, reading multiple registers is sequential.
Number of Coils: (2 bytes): Number of Elements (Coils) that the device has to return (3 = 3 Coils or bits). No more than 80 Elements (Coils) allowed. (Each coil is 1 bit).
CRC: (2 bytes): CRC calculated for the frame data received and is used to verify the integrity of data received.

3.1.2 SUCCESSFUL RESPONSE, READ COILS (0x01):

Slave Address	Success echo: Code 0x01	Number of Bytes	Data 1	Data …	Data n	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): The address of the slave responding. Same as the address in the initiating command.
Function Code: (1 byte): Code = 0x01 Echo from the initiating command.
Number of Bytes: (1 byte): Defines the number of bytes followed minus the CRC.
Data: Byte data buffer, length is “Number of Bytes” long. Coil data I contained in this data. The first 8 Coils will be the first Byte.
CRC: (2 bytes):

3.1.3 EXCEPTION RESPONSE, READ HOLDING REGISTER (0x01):

Slave Address	Exception 0x01 + 0x80 Error: 0x81	Exception Code see list	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): The address of the slave responding. Same as the address in the initiating command.
Exception Response: (1 byte): Code = 0x01 Echo from the initiating command plus high bit 0x80 = 0x81.
Exception Code: (1 byte): Defines the number of bytes followed minus the CRC. See Table 1 for explanation of exceptions.
CRC: (2 bytes):

3.2 WRITE SINGLE COIL (0x05):

Write to a single Coil. For multiple coils, use multiple commands. You SET (1) the coil by sending 0xFF00 in the Data field. You RESET (0) the coil by sending 0x0000.

3.2.1 MASTER COMMAND TO WRITE SINGLE COIL (0x05):

Slave Address	Function Code 0x05	Coil Address (MSByte)	Coil Address (LSByte)	Data (MSByte)	Data (LSByte)	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): Device address that receives the command. Range: 1-247.
Function Code: (1 byte): Code = 0x05 (Read Output Coil).
Coil Address: (2 bytes): The address of the first coil to be read, reading multiple registers is sequential.
Data: (2 bytes): 0x000 will RESET (0), and 0xFF00 will SET (1) the coil.
CRC: (2 bytes): CRC calculated for the frame data received and is used to verify the integrity of data received.

3.2.2 SUCCESSFUL RESPONSE, WRITE SINGLE COIL (0x05):

Slave Address	Success Echo Code 0x05	Coil Address (MSByte)	Coil Address (LSByte)	Data (MSByte)	Data (LSByte)	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): The address of the slave responding. Same as the address in the initiating command.
Function Code: (1 byte): Code = 0x05 Echo from the initiating command.
Coil Address: (2 bytes): Echo from initiating command.
Data: (2 bytes): Echo from initiating command.
CRC: (2 bytes):

3.2.3 EXCEPTION RESPONSE, WRITE SINGLE COIL (0x05):

Slave Address	Exception 0x05 + 0x80 Error: 0x85	Exception Code see list	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): The address of the slave responding. Same as the address in the initiating command.
Exception Response: (1 byte): Code = 0x05 Echo from the initiating command plus high bit 0x80 = 0x85.
Exception Code: (1 byte): Defines the number of bytes followed minus the CRC. See Table 1 for explanation of exceptions.
CRC: (2 bytes):

3.3 READ HOLDING REGISTERS (0x03):

Read a single register or multiple registers from the XR10CX. Response will be the values stored in the registers.

3.3.1 MASTER COMMAND TO READ REGISTERS (0x03):

Slave Address	Function Code 0x03	Register Address (MSByte)	Register Address (LSByte)	Number of Registers (MSByte)	Number of Registers (LSByte)	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): Device address that receives the command. Range: 1-247.
Function Code: (1 byte): Code = 0x03 (Read holding register).
Register Address: (2 bytes): The address of the first register to be read, reading multiple registers is sequential.
Number of Registers: (2 bytes): Number of Elements (Registers) that the device has to return (3 = 3 Registers). No more than 5 Elements (registers) allowed. (Each register is 16 bits).
CRC: (2 bytes): CRC calculated for the frame data received and is used to verify the integrity of data received.

3.3.2 SUCCESSFUL RESPONSE, READ HOLDING REGISTERS (0x03):

Slave Address	Success echo: Code 0x03	Number of Bytes	Data 1	Data …	Data n	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): The address of the slave responding. Same as the address in the initiating command.
Function Code: (1 byte): Code = 0x03 Echo from the initiating command.
Number of Bytes: (1 byte): Defines the number of bytes followed minus the CRC.
Data: Byte data buffer, length is “Number of Bytes” long.
CRC: (2 bytes):

3.3.3 EXCEPTION RESPONSE, READ HOLDING REGISTER (0x03):

Slave Address	Exception 0x03 + 0x80 Error: 0x83	Exception Code see list	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): The address of the slave responding. Same as the address in the initiating command.
Exception Response: (1 byte): Code = 0x03 Echo from the initiating command plus high bit 0x80 = 0x83.
Exception Code: (1 byte): Defines the number of bytes followed minus the CRC. See Table 1 for explanation of exceptions.
CRC: (2 bytes):

3.4 WRITE SINGLE REGISTER (0X06):

Command to write a value to a single register. Response will be the data written.

3.4.1 MASTER COMMAND TO WRITE SINGLE REGISTER (0x06):

Slave Address	Function Code 0x06	Register Address (MSByte)	Register Address (LSByte)	Data (MSByte)	Data (LSByte)	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): Device address that receives the command. Range: 1-247.
Function Code: (1 byte): Code = 0x06 (Write single register).
Register Address: (2 bytes): The address of the register to be written to.
Data: (2 bytes): The data to write.
CRC: (2 bytes): CRC calculated for the frame data received and has to be used to verify the integrity of data received.

3.4.2 SUCCESSFUL RESPONSE FROM WRITE SINGLE REGISTER (0x06)

Slave Address	Success: Code 0x06	Register Address (MSByte)	Register Address (LSByte)	Data (MSByte)	Data (LSByte)	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): The address of the slave responding. Same as the address in the initiating command.
Function Code: (1 byte): Code = 0x06 Echo from the initiating command.
Register Address: (2 bytes): The address of the register that was written.
Data: (2 bytes): Byte data buffer, will contain the same data that was sent in initiating command.
CRC: (2 bytes):

3.4.3 EXCEPTION RESPONSE FROM WRITE SINGLE REGISTER (0x06):

Slave Address	Exception 0x06 + 0x80 Error: 0x86	Exception Code see list	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): The address of the slave responding. Same as the address in the initiating command.
Exception Response: (1 byte): Code = 0x06 Echo from the initiating command plus high bit 0x80 = 0x86.
Exception Code: (1 byte): Defines the number of bytes followed minus the CRC. See Table 1 for explanation of exceptions.
CRC: (2 bytes):

3.5 WRITE HOLDING REGISTERS (0x10):

Command to write 1-5 registers. Limit is 5 registers. Response will be the number of registers written.

3.5.1 MASTER COMMAND TO WRITE HOLDING REGISTER (0x10):

Slave Address	Function Code 0x10	Register Address (MSByte)	Register Address (LSByte)	Number Registers (MSByte)	Number Registers (LSByte)	Num Byte	Data	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): Device address that receives the command. Range: 1-247.
Function Code: (1 byte): Code = 0x10 (Write holding registers 1-5).
Register Address: The address of the first register to write to.
Number of Registers: (2 bytes): Defines the number of Elements (Registers) to write to. No more than 5 Elements allowed for the XR10CX.
Num Byte: (1 byte): Defines the number of bytes followed minus the CRC. The number of bytes has to be double the number of addressed Elements (Number of bytes = 2 x number of Registers).
Data (Num Byte or 2 X Number of Registers): Data to be written in MSByte, LSByte order.
CRC: (2 bytes): CRC calculated for the frame data received and has to be used to verify the integrity of data received.

3.5.2 SUCCESSFUL RESPONSE FROM WRITE HOLDING REGISTERS (0x10):

Slave Address	Function Code 0x10	Register Address (MSByte)	Register Address (LSByte)	Number Registers (MSByte)	Number Registers (LSByte)	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): The address of the slave responding. Same as the address in the initiating command.
Function Code: (1 byte): Code = 0x10 Echo from the initiating command.
Register Address: (2 bytes): The address of the register that was written.
Number Registers: (2 bytes): The number of registers written.
CRC: (2 bytes):

3.5.3 EXCEPTION RESPONSE FROM WRITE HOLDING REGISTERS (0x10):

Slave Address	Error, Code 0x10 + 0x80 Error: 0x90	Exception Code, see list	CRC (LSByte)	CRC (MSByte)

Slave Address: (1 byte): The address of the slave responding. Same as the address in the initiating command.
Exception Response: (1 byte): Code = 0x10 Echo from the initiating command plus high bit 0x80 = 0x90.
Exception Code: (1 byte): Defines the number of bytes followed minus the CRC. See Table 1 for explanation of exceptions.
CRC: (2 bytes):

3.6 EXCEPTION RESPONSE:

Exceptions result from a valid packet being received by the Slave device but the slave is unable to complete the command. This can be the result of an invalid address, or a write command to a read only Register. Table 1 provides an explanation of each Exception Code and the possible cause.

Table 1
EXCEPTION CODES:

Exception Code	Name	Meaning
01	Illegal Function	The function code received in the query is not an allowable action for the slave. Only function codes 0x03, 0x06, 0x10 are valid commands.
02	Illegal Data Address	The data address received in the query is not an allowable address for the slave. More specifically, the combination of reference number and transfer length is invalid.
03	Illegal Data Value	Requesting a register that does not exist. More than 5 elements requested. Writing a parameter out of range. Writing to read only register.
04	Slave Device Failure	An unrecoverable error occurred while the slave was attempting to perform the requested action. The device didn't succeed in reading or writing requested operation. Operation (Ram, E2, RTC and etc) is not completing operation correctly.
06	Slave Device Busy	The device can't execute requested operation at this time. Busy in another analogue operation. Master has to repeat the same request at another time.

4. XR10CX REGISTERS & COILS

4.1 LIST OF REGISTERS IN THE XR10CX

Table 2

List of registers for the XR10CX Label is the descriptive text displayed on the XR10CX or on the terminal labels.

Label		Description				Coil	Register
PRB1	Read	Probe 1 Temperature					40257
HY	Read/Write	Differential					40769
ALd	Read/Write	Temperature Alarm Delay					40830
DAo	Read/Write	Temp Alarm Delay at Startup					40831
Did	Read/Write	Dig Input Alarm Delay 0-255min					40847
Adr	Read/Write	Modbus Address 1-247					40852
Set	Read/Write	System Setpoint					40864
						Coil	
	Read/Write	On Off for Control				513	
	Read/Write	Keypad Lock				516	
	Read/Write	Mute Buz and Any-Alarm				517	
	Read	Digital input Pin 9				520	
	Read	Probe 1 Error				521	
	Read	Probe 2 Error				522	
	Read	Probe 3 Error				523	
	Read	Probe 4 Error				524	
	Read	High Temperature Alarm Pb1				525	
	Read	Low Temperature Alarm Pb1				526	
	Read	Alarm from External input				529	
	Read	EEPROM Failure				532	
	Read	ANY-Alarm Can be Muted				538	
	Read	Relay Call For Heat				543	
	Read	Buzzer On with Ext or Hi Alarm				545	

4.2 TYPICAL PARAMETERS FOR ACCESS OVER MODBUS:

Setpoint: Modbus Register 40864
Set Set point,
This is the typical system setpoint. Read/Write.

Modbus network ID or Address. Modbus Register 40852
Adr Serial address 0÷247
This is the Modbus address. You should configure this with the keypad.

Temperature of Probe: Modbus Registers 40257
(TP1) Probe 1 temperature Degrees F (40257)
This is the operating probe that Set references. Read only.

Output Relays (Burner ON): Modbus Coil 543
Statas of Relay, if it is ON the signal for burner is also ON.

Alarms: Modbus Coil 538
Alarms are not typically used on this device.

CONTROL, ON/OFF (Enable/Disable) Modbus Coil 513
ON=257 or 0x0101, OFF=1 or 0x0001
In the OFF state, no heating signal will be present, no alarms, all relay outputs are open, and no temperature are displayed. The display will show the word “OFF”. It is still possible to read the probe temperatures and other parameters over Modbus.
This is a write only register. Reading this register will provide invalid results.

5. SERIAL CONFIGURATION

5.1 PORT SETUP

Baud Rate: 9600bps (Not adjustable)
Data Length: 8 bit (Not adjustable)
Parity: None (Not adjustable)
Stop Bits: 1 (Not adjustable)
Start/Stop: Silent interval of 3 characters minimum
Minimum Time Between Retry: 500 msec

6. WIRING

Modbus RTU uses the same wiring practice and wire as BACNET MSTP.

6.1 TYPICAL WIRING DETAIL

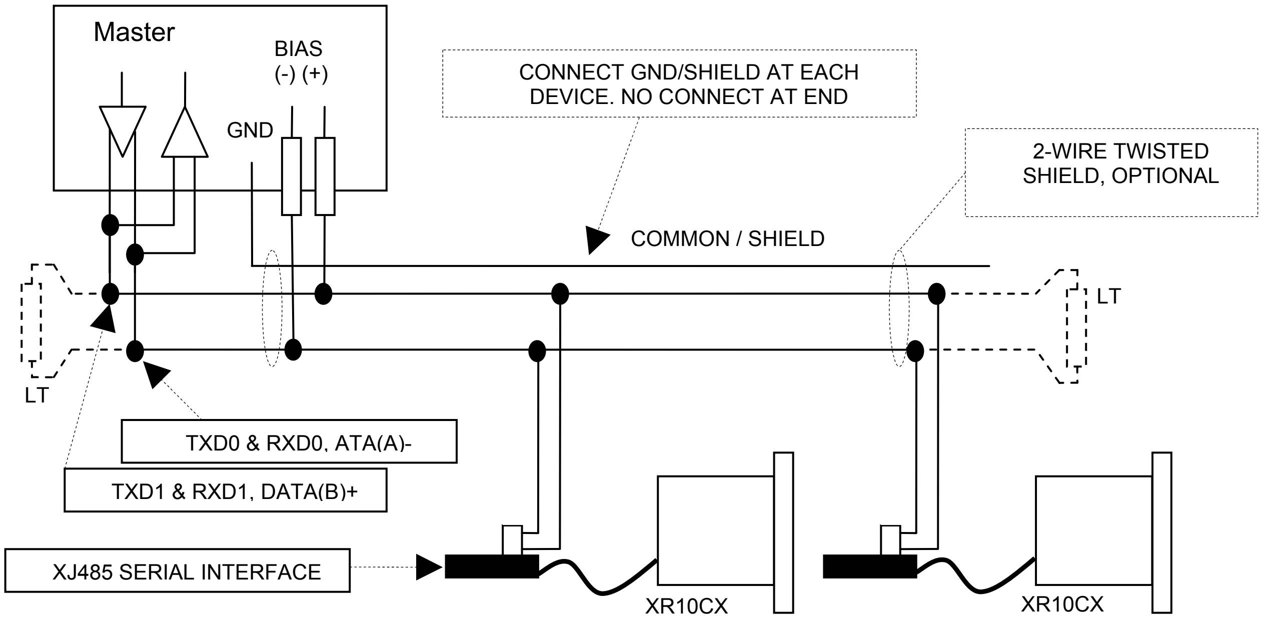

Figure 3 Wiring of a typical Modbus RTU network with XR10CX devices

6.2 PROPER WIRING EXAMPLE

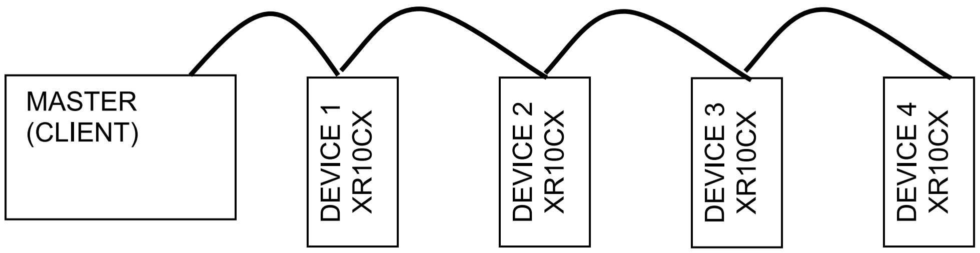

Figure 4 Correctly wired RS-485 Daisy Chain

6.3 IMPROPER WIRING EXAMPLE

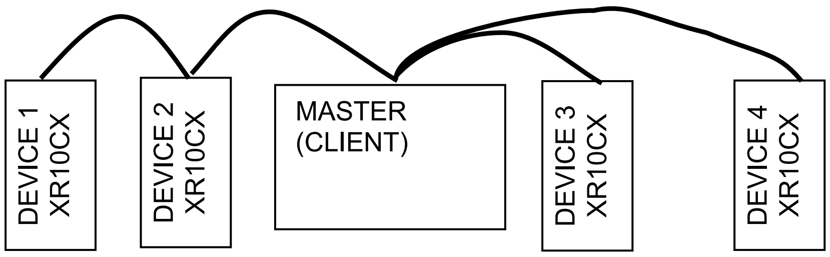

Figure 5 Wiring is not correct. Not a daisy chain and the master is not at an end.

6.4 LINE TERMINATION (LT)

Termination resistors are only required on very long cable runs. A reflection in a transmission line is the result of an impedance discontinuity that a travelling wave sees as it propagates down the line. To minimize the reflections from the end of the RS485-cable, place a Line Termination near each of the 2 ends of the Bus. It is important that the line be terminated at both ends since the propagation is bi-directional, but it is not allowed to place more than 2 LTs on one passive balanced pair.

- Each line termination must be connected between the two conductors of the balanced line: D0 and D1.
- Line termination may be a 150 ohms value (0.5 W) resistor.
- With cable lengths less than 2000', at 9600 baud, reflection is not an issue and does not require LT resistors.

6.5 LINE BIASING

When there is no data activity on an RS-485 balanced pair, the lines are not driven and thus susceptible to external noise or interference. To insure the line is in a known state when the line is not active, one or more devices on the network can provide line biasing by pulling up and down the lines with week resistors. It is common practice for the Master device to provide line biasing. This is generally a jumper setting on the device. The XR10CX does not have line biasing resistors.

- Data(B)+ will be pulled to positive
- Data(A)- will be pulled to negative

6.6 SERIAL INTERFACE XJ485

The XJ485 serial termination is a factory supplied RS485 to TTL connection device. The XR10CX control comes standard with a TTL communication port also used as the HOT KEY programming interface. The termination points are labelled as (+ and –) corresponding to (D1 and D0).

7. REFERENCES

7.1.1 MODBUS INFORMATION:

This document is not intended to be a comprehensive guide for application or installation of a MODBUS solution. The following are some additional resources regarding Modbus:

- ANSI/TIA/EIA-232 Interface between data terminal equipment and data circuit-terminating equipment employing serial binary data interchange.
- ANSI/TIA/EIA-485 Electrical characteristics of generators and receivers for use in balanced digital multipoint systems.
- MODBUS.org MODBUS applications protocol specification.

7.1.2 APPLICATION DETAILS AND SENSOR LOCATIONS:

The XR10CX is used in multiple devices with several configurations. Application and configuration information for the XR10CX as it is applied to your device can be found in the I & O Manual of the device.

7.1.3 XR10CX FEATURES:

General features and programing of the XR10CX can be found in the XR10CX manual.

7.1.4 MORE INFORMATION:

For additional information, contact the PVI Industries Customer Service Dept. at 800-784-8326.