

Modbus Protocol User Guide

Revision F November, 2013

Copyright and Trademark

Copyright © 2004-2013, Grid Connect, Inc. All rights reserved.

No part of this manual may be reproduced or transmitted in any form for any purpose other than the purchaser's personal use, without the express written permission of Grid Connect, Inc. Grid Connect, Inc. has made every effort to provide complete details about the product in this manual, but makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. In no event shall Grid Connect, Inc. be liable for any incidental, special, indirect, or consequential damages whatsoever included but not limited to lost profits arising out of errors or omissions in this manual or the information contained herein.

Grid Connect, Inc. products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of a Grid Connect, Inc. product could create a situation where personal injury, death, or severe property or environmental damage may occur. Grid Connect, Inc. reserves the right to discontinue or make changes to its products at any time without notice.

Grid Connect and the Grid Connect logo, and combinations thereof are registered trademarks of Grid Connect, Inc. All other product names, company names, logos or other designations mentioned herein are trademarks of their respective owners.

Grid Connect

1630 W. Diehl Rd.
Naperville, IL 60563
USA
Phone: 630.245.1445

Technical Support

Phone: 630.245.1445
Fax: 630.245.1717
On-line: gridconnect.com

Disclaimer and Revisions

The information in this guide may change without notice. The manufacturer assumes no responsibility for any errors that may appear in this guide.

Date	Rev.	Author	Comments
11/23/04	A	GR	Preliminary Release
06/02/05	B	GR	Notes for Device Installer and Product Code
01/03/06	C	GR	Notes for XPort-MB and NET232-MB
10/30/09	D	GR	Revisions to Device Installer, Notes for WiPort/WiBox
12/10/12	E	JW	Notes for all hardware added to Introduction
11/06/13	F	EDL	Updates for V3.3.0.0 and web manager

Warranty

Grid Connect warrants the media containing software and technical information to be free from defects and warrants that the software will operate substantially for a period of 60 DAYS after the date of shipment.

In no event will Grid Connect be responsible to the user in contract, in tort (including negligence), strict liability or otherwise for any special, indirect, incidental or consequential damage or loss of equipment, plant or power system, cost of capital, loss of profits or revenues, cost of replacement power, additional expenses in the use of existing software, hardware, equipment or facilities, or claims against the user by its employees or customers resulting from the use of the information, recommendations, descriptions and safety notations supplied by Grid Connect. Grid Connect liability is limited (at its election) to:

- 1) refund of buyer's purchase price for such affected products (without interest)
- 2) repair or replacement of such products, provided that the buyer follows the above procedures.

There are no understandings, agreements, representations or warranties, expressed or implied, including warranties of merchantability or fitness for a particular purpose, other than those specifically set out above or by any existing contract between the parties. The contents of this document shall not become part of or modify any prior or existing agreement, commitment or relationship.

Table of Contents

1. Introduction	1-5
2. Hardware.....	2-5
2.1 NET232	2-6
2.1.1 Serial RS232 Interface.....	2-6
2.1.2 OEM Module.....	2-6
2.2 NET485	2-7
2.2.1 Overview of RS485/RS422	2-7
2.2.2 RS485/RS422 Network Connections	2-11
2.2.3 RS485/RS422 Serial Interface.....	2-12
2.2.4 Full Duplex/Half Duplex Jumper	2-14
2.3 WI232	2-15
2.3.1 Serial RS232 Interface.....	2-15
2.3.2 OEM Module.....	2-15
2.4 Embedded Modules	2-17
2.5 Power Supply.....	2-17
2.6 Ethernet Interface	2-17
3. Modbus	3-19
3.1 Extended Modbus System Example	3-20
3.1.1 Modbus/TCP Master Talking to Modbus/TCP Slave.....	3-20
3.1.2 Modbus/TCP Master Talking to Modbus/RTU Serial Slave.....	3-21
3.1.3 Modbus/RTU Serial Master Talking to Modbus/TCP Slave.....	3-21
3.1.4 Modbus/RTU Serial Master Talking to Modbus/RTU Serial Slave..	3-21
3.2 Network Protocols	3-23
3.3 Packing Algorithm.....	3-23
3.4 IP Address.....	3-23
3.5 Configuration Methods.....	3-23
3.6 Device Server's IP Address	3-24
3.7 Device Installer.....	3-24
3.7.1 RUN Device Installer	3-24
3.7.2 Device Found.....	3-25
4. Web Configuration	4-26
5. Telnet Configuration	5-30
5.1 Basic Commands (D/S/Q)	5-32
5.1.1 Default Settings (D).....	5-32
5.1.2 Save (S)	5-32
5.1.3 Quit Without Saving (Q)	5-32
6. Configuring Modbus	6-32
6.1 Network/IP Settings.....	6-32
6.1.1 IP Address	6-33
6.1.2 Set Gateway IP Address (Y/N).....	6-33

6.1.3 Set Netmask (N for default)	6-33
6.1.4 Telnet/Web Configuration Password	6-33
6.2 Serial and Mode Settings	6-34
6.2.1 Attached Device (1=Slave, 2=Master).....	6-34
6.2.2 Serial Protocol (1=Modbus/RTU, 2=Modbus/ASCII)	6-34
6.2.3 Interface Type (1=RS232 2=RS422/RS485+4-wire 3=RS485+2-wire).....	6-35
6.2.4 Enter Serial Parameters (9600,8,N,1)	6-35
6.3 Modem Control Settings	6-36
6.3.1 RTS/CTS Mode (1=Fixed 2=Variable).....	6-36
6.3.2 Delay after Output of RTS (0-1275 ms, 5ms increments)	6-36
6.3.3 Wait for CTS to Go Active (N/Y).....	6-36
6.3.4 Delay after CTS Going Active (0-1275 ms, 5ms increments)	6-36
6.3.5 Delay Dropping RTS after Transmitting (0-1275 ms, 5 ms increments).....	6-37
6.3.6 DTR Mode (1=Fixed, 2=Active with connection)	6-37
6.4 Modem/Configurable Pin Settings.....	6-37
6.4.1 NET232/NET485/XPort-MB	6-37
6.4.2 XPort-Direct+-MB	6-39
6.4.3 MatchPort/WiPort/Wi232-MB	6-40
6.5 Advanced Modbus Protocol Settings	6-42
6.5.1 Modbus/TCP Port (standard default is 502).....	6-43
6.5.2 Slave Address (0 for auto, or 1..255 fixed otherwise).....	6-43
6.5.3 Allow Modbus Broadcasts (1=Yes 2=No).....	6-43
6.5.4 Use MB/TCP 00BH/00AH Exception Responses (1=No 2=Yes)	6-43
6.5.5 Disable Modbus/TCP pipeline (1=No 2=Yes)	6-44
6.5.6 Character Timeout (0 for auto, or 10-6950 msec) (50)	6-44
6.5.7 Message Timeout (200-65000 msec) (5000)	6-44
6.5.8 Serial TX delay after RX (0-1275 msec) (0).....	6-44
6.5.9 Swap 4x/0x to get 3x/1x (N)	6-44
6.6 Preset Automated Scan Table	6-45
6.6.1 A)dd, D)elete, E)xit Select Function.....	6-46
6.6.2 Modbus Address	6-46
6.6.3 Data Type.....	6-46
6.6.4 Register Offset	6-46
6.6.5 Register Count.....	6-46
6.6.6 Frequency.....	6-46
6.7 Unit ID to IP Address Lookup Table	6-47
6.7.1 Close Idle TCP sockets after (3-60 sec, 0=leave open).....	6-49
6.7.2 Redundant Entry Retries after (15-60 sec. 0=disable feature)	6-49
6.7.3 A)dd, D)elete, E)xit Select Function.....	6-49
6.7.4 Modbus Address From/To	6-49
6.7.5 Slave IP Address	6-49
6.8 Security Settings.....	6-50
6.8.1 Disable SNMP.....	6-50
6.8.2 SNMP Community Name	6-50
6.8.3 Disable Telnet Setup	6-50
6.8.4 Disable Telnet Debug port	6-50
6.8.5 Disable TFTP Firmware Update	6-50
6.8.6 Disable Port 77FEh	6-50
6.8.7 Disable Web Server.....	6-50
6.8.8 Disable Web Setup.....	6-50
6.8.9 Disable ECHO ports.....	6-50

6.8.10 Enable Enhanced Password	6-51
7. Monitor Mode and Firmware Upgrade	7-53
8. Troubleshooting	8-55
8.1 Debug Port 3000	8-55
8.1.1 Modbus Debug Codes	8-55
8.1.2 Modbus Error Codes	8-57
8.2 Troubleshooting Software	8-57
8.3 How fast can I poll?	8-58
8.4 I cannot get a slave response	8-59
8.5 Only Slave ID #1 can be polled	8-59
8.6 Every 2 nd poll seems to fail	8-59
9. Technical Support	9-63

1. Introduction

The Modbus firmware bridges Modbus/TCP on the Ethernet side to Modbus serial on the RS232/RS422/RS485 side. The serial device can be a Modbus slave or master.

When attached to a Modbus serial slave(s) it converts Modbus/TCP requests from up to 10 clients on the network into serial Modbus/RTU or /ASCII requests. Modbus serial responses are converted back to Modbus/TCP and delivered to the original client.

When attached to a Modbus serial master it takes Modbus master serial requests (RTU or ASCII), converts them to Modbus/TCP client requests and routes them to the destination IP address on the network based on a lookup table. The Modbus/TCP response from the server is then converted back to Modbus serial.

This manual provides detailed operation and setup information for products with Modbus firmware. Many device server products are designed with the XPort and other similar device server parts. For example, the NET232 serial to Ethernet converter contains an XPort device server. When the XPort contains Modbus firmware, it is known as an XPort-MB.

The default protocol found in an XPort device server is standard Serial Tunneling protocol, an encapsulation protocol used to transport serial data over an IP network. The XPort device server with Serial Tunneling is used to connect general serial devices to an Ethernet network by packing serial data inside UDP/IP and TCP/IP packets. Changing the standard Serial Tunneling protocol to Modbus will change the setup configuration menus and dialogs. Therefore, this manual provides Modbus protocol specific information for the XPort-MB and other similar device servers.

2. Hardware

The NET232 and NET485 Adapters, designed by Grid Connect, use the XPort Device Server. The NET232 is a complete RS232 Serial to Ethernet interface. The Modbus version of this product is the NET232-MB. The NET485 is an RS422/485 Serial to Ethernet interface. The Modbus version of this product is the NET485-MB.

2.1 NET232

2.1.1 Serial RS232 Interface

The table below lists the RS232 signals for the NET232. The RS232 interface is a 9-pin D-style connector. Male connectors are wired as DTE and female connectors are wired as DCE.

Table 1 - RS232 Signals

NET232 Signal	Direction	DTE DB-9 Male Pin #	DCE DB-9 Female Pin #
Data Out (TXD)	Out	3	2
Data In (RXD)	In	2	3
Ground		5	5
RTS	In	8	7
CTS	Out	7	8
No Connection		1,4,6,9	1,4,6,9

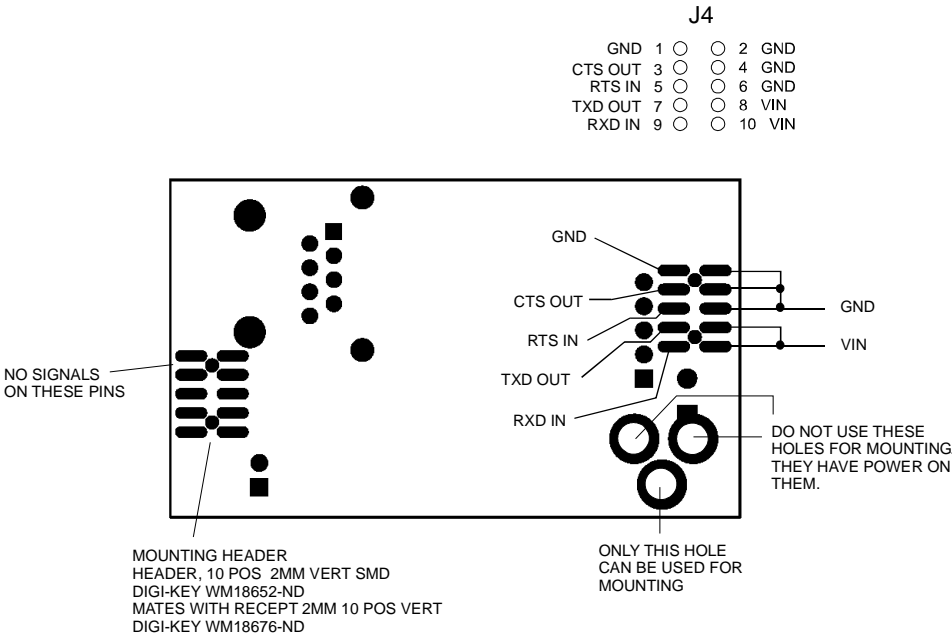
The NET232-DCE kit includes a 9-pin Male/Male Null Modem Adapter if you need a male connector. The NET232-DTE kit includes a 9-pin Female/Female Null Modem Adapter.

The DB9 connector can be used to power the NET232 through one of the unused pins. Please contact the factory for more information.

2.1.2 OEM Module

The NET232 circuit board can be supplied as an OEM module. The module has two headers that are used to secure the module to a motherboard. One hole on the edge of the board can be used for mounting to a standoff.

Note: OEM Modules must be ordered in quantities of 25 or more.



2.2 NET485

2.2.1 Overview of RS485/RS422

RS232 is an EIA standard transmission system and has been around since 1962. RS232 provides single-ended data communications between a transmitter and a receiver. In that era, it allowed for data transmission from one transmitter to one receiver at relatively slow data rates (20k bits / second) and short distances (up to 50 ft. at the maximum data rate).

While RS232 is well-known for connecting PC's to external devices, RS422 and RS485 are not as well known. When communicating at high data rates, or over long distances in real world environments, single-ended methods are often inadequate. RS422 and RS485 were designed to provide data communications over longer distances, higher Baud rates and provide better immunity to external electro-magnetic noise.

RS422 and RS485 use differential data transmission (balanced differential signal). This offers superior performance by canceling the effects of ground shifts and induced noise signals that can appear as common mode voltages on a network. This also allows for data transmission at much higher data rates (up to 460K bits / second) and longer distances (up to 4000 ft.).

What is the difference between RS422 and RS485? Like RS232, RS422 is intended for point-to-point communications. In a typical application, RS422 uses four wires (two separate Twisted Pairs of wires) to transfer data in both directions simultaneously (Full Duplex) or independently (Half Duplex). EIA/TIA-422 specifies the use of one, unidirectional driver (transmitter) with a maximum of 10 receivers. RS422 is often used in noisy industrial environments or to extend a RS232 line.

RS485 is used in applications where multiple devices want to share data communications on a single serial network. RS485 can support up to 32 drivers and 32 receivers on a single two wire (one twisted pair) bus. Most RS485 systems use a Master/Slave architecture, where each slave unit has its unique address and responds only to packets addressed to it. However, peer to peer networks are also possible.

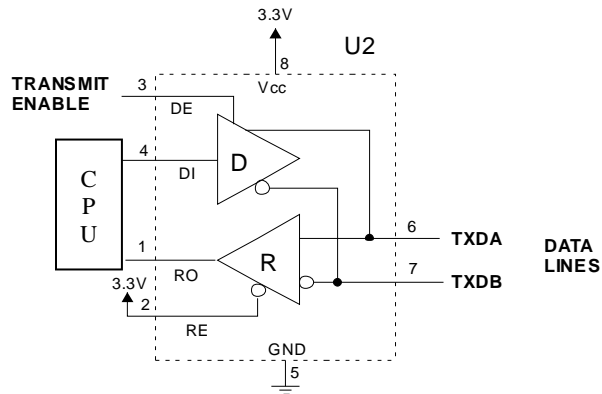
Specification	RS-422	RS-485
Transmission Type	Differential	Differential
Maximum Data Rate	10 MB/s	10 MB/s
Maximum Cable Length	4000 ft.	4000 ft.
Driver Load Impedance	100 Ohm	54 Ohm
Receiver Input Resistance	4 KOhm min	12 KOhm min
Receiver Input Voltage Range	-7V to +7V	-7V to +12V
No of Drivers Per Line	1	32
No of Receivers Per Line	10	32

The RS485 Enable Signal

In a balanced differential system the data signals are produced by a line driver. (See the drawing below) The line driver generates a voltage across a pair of signal wires that transmits the data signals. A balanced line driver can have an optional input signal called an “Enable” signal. The purpose of the enable signal is to connect the driver to its output terminals. If the enable signal is off, the driver is disconnected from the transmission line. When a driver is disconnected from the network it is referred to as being in the “tri-state” condition. Because there are multiple drivers (transmitters) on a RS485 network and only one transmitter can be enabled at a time, the use of this enable control signal is required on all two-wire RS485 networks.

The following drawing shows a typical RS485 driver. Pin 3 is the transmit enable pin. Pin 2, the receive enable pin, is tied to 3.3V which forces the receiver to always be enabled. When Pin 3 is high the transmitter is enabled and data passes out the transmit pins. The design of the circuit allows the receiver section to read

the data that is being transmitted. When Pin 3 is low the transmitter is disabled and the output goes to a “tri-state” condition. In this condition, the receiver section is still listening to the network.



The NET485 provides the Transmit enable signal for RS485 two-wire applications. When configured for RS485 two-wire applications, the NET485 automatically asserts the transmit enable when it is ready to transmit data from its serial port. Once the data has been transmitted, the NET485 automatically de-asserts the enable signal to allow other nodes to transmit their data.

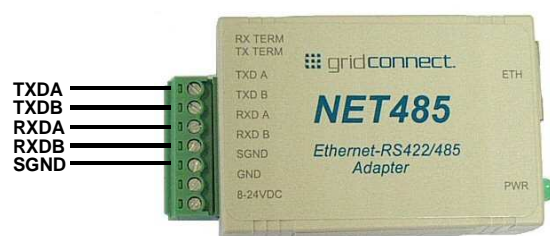
The Transmit enable pin is called a Configurable Pin, meaning it can be selected to do several functions. One of those options is the transmit enable, which is labeled “RS485 TX Enable”. The Quick Start Guide explains how to configure the NET485 for proper RS485 operation.

WARNING: The NET485 comes from the factory already configured for RS485 operation.

The NET485 uses Configurable Pin 1 (**CP1**) for the transmit enable. You must enable **CP1** for **RS485_TXEN** during the configuration process. (See the Quick Start Guide for details)

The NET485 uses **CP1** in the Active High condition. You must set **CP1** to **Active High** during the configuration process. (See the Quick Start Guide for details)

The NET485 can handle both RS485 and RS422 communications. This is done by connecting the processor (CPU) to a pair of RS485 transceivers. The following schematic demonstrates how the circuit is wired for RS422.



The transmit section of the CPU is labeled TX on pin 4. The receive section of the CPU is labeled RX on pin 5. Note that the **CP1** pin on the CPU pin 6, which is configured to control the level of RS485_TXEN, is connected to both transceivers. The transmit section is enabled with a High signal on pin 3, U2 and the receive section is enabled with a Low signal on pin 2, U4.

2-9

RS485 Operation

For RS485 mode, the TXDA (+) signal is connected to the RXDA terminal, and the TXDB (-) signal is connected to the RXDB terminal. The three signals are TXDA, RXDB, and signal ground.

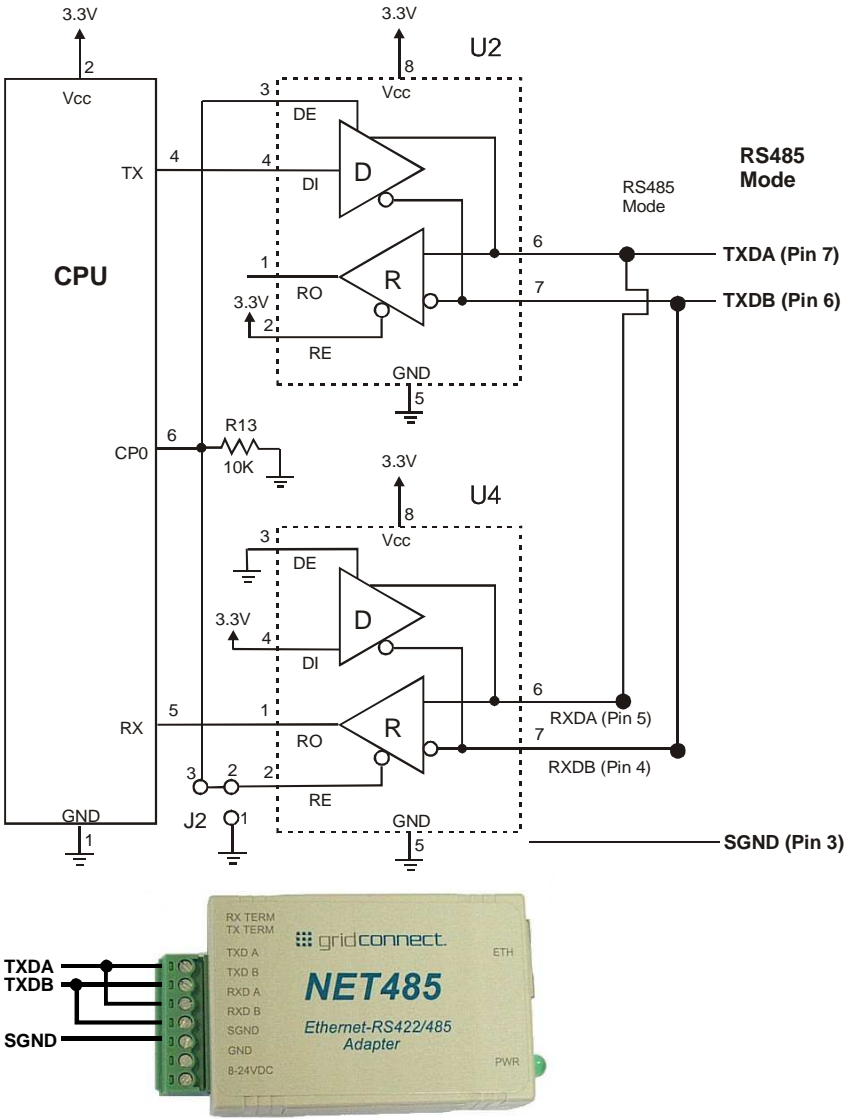


Figure 2 - RS485 Wiring

2.2.2 RS485/RS422 Network Connections

RS422 Networks

A typical RS422 application uses a four-wire interface (two twisted pairs) and a shield. RS422 networks are often used in a half-duplex mode, where a single master in a system sends a command to a slave device and the slave responds with data. Typically one device (node) is addressed by the host computer and a response is received from that device. Systems of this type (4-wire, half-duplex) are often constructed to avoid "data collision" (bus contention) problems on a network. Figure 3 shows a typical RS422 four wire interface.

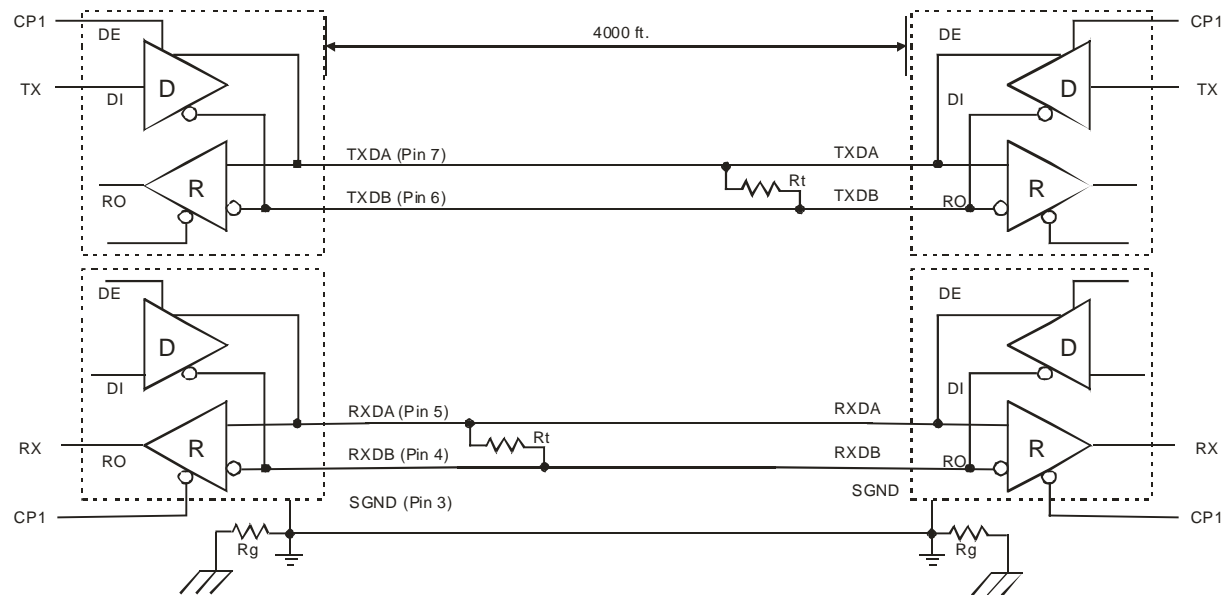


Figure 3 - RS422 Four Wire Interface

Notice that 5 conductors are used (two twisted pairs and a ground wire). Also, when the cable lengths are long and/or the data rates are high, the network must be terminated. To terminate the network, a resistor R_t is added in parallel with the receiver's A and B lines. R_g is an optional resistor between ground and the shield. R_t termination resistors are available as option jumpers on the NET485.

Note: Do NOT install termination resistors on short wire networks. See the Application Notes on the product CD for more information about networks and termination procedures.

RS485 Networks

RS485 permits a balanced transmission line to be shared in a party line or multi-drop configuration. As many as 32 driver / receiver pairs can share a multi-drop network on a single two wire bus. The length of the network is limited to 4,000 ft. between the first node and the last node. RS485 can be used in two-wire or four-wire multi-drop network applications.

Figure 4 shows a typical RS485 two-wire multi-drop network. The tri-state capabilities of 485 allow a single pair of wires to share transmit and receive signals for half-duplex communications. In this configuration, it is important to prevent more than one device from transmitting at the same time. This is controlled by software and the communications protocol. Note that the transmission line is terminated on both ends of the line but not at drop points in the middle of the line. Termination is only required with high data rates and / or long wire runs.

Note: Do NOT install termination resistors on short wire networks. See the Application Notes on the product CD for more information about networks and termination procedures.

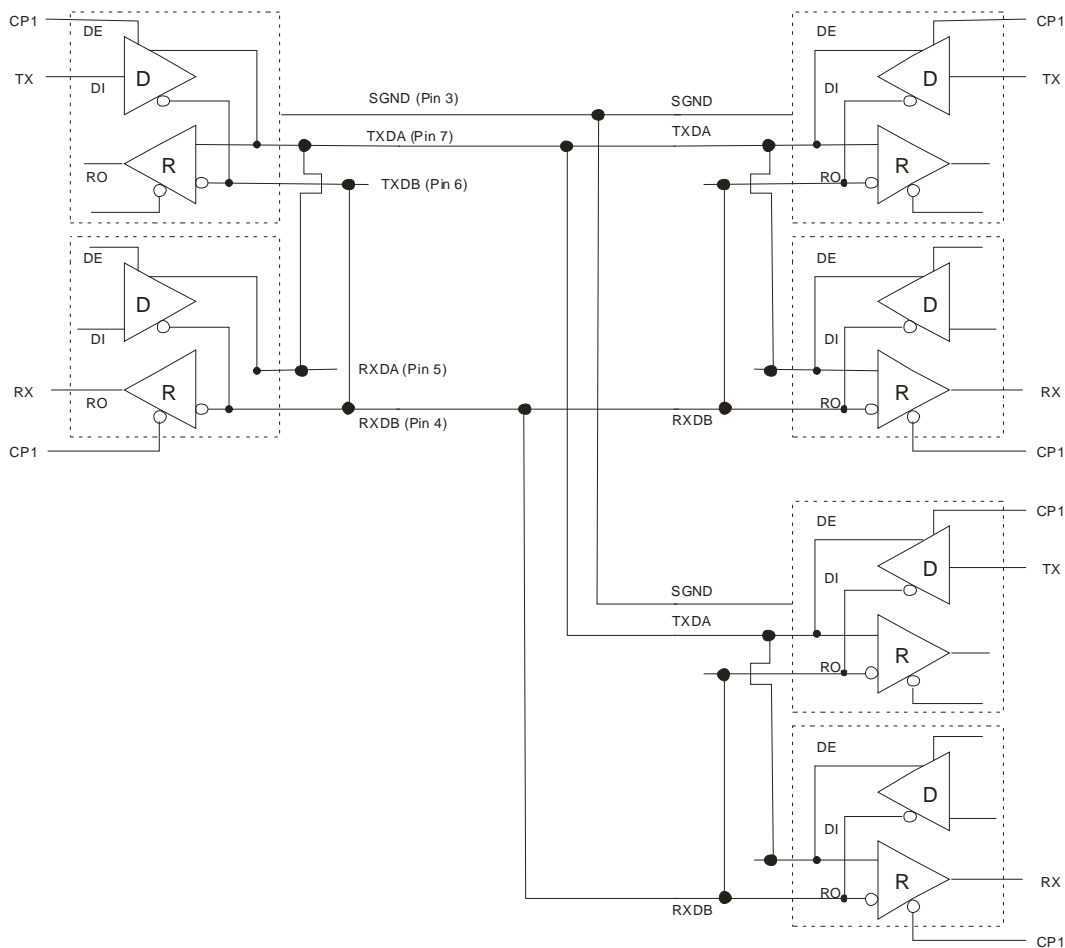


Figure 4 - RS485 Network

2.2.3 RS485/RS422 Serial Interface

The NET485 product allows you to connect an RS422/RS485 device to the Ethernet. Using RS485 two-wire mode, you can connect the NET485's serial interface to multiple devices in a multi-drop network.



Figure 5 - Main Features

The table below lists the RS422/485 signals for the NET485. The RS422/485 and power interface is a 7-pin removable Phoenix connector, with two of the pins used for power.

Table 2 - RS422/485 Signals

NET485 Signal	7-Pin Phoenix
TXDA (+)	7
TXDB (-)	6
RXDA (+)	5
RXDB (-)	4
SGND	3
GND	2
8-24VDC	1



Figure 6 - Phoenix Connector

The NET485 uses protective clamping structures on its inputs and outputs that clamp the voltage to a safe level and dissipate the energy present in ESD (electrostatic) and EFT (electrical fast transients) discharges. This protection structure achieves ESD protection up to 8 kV according to IEC1000-4-2, and EFT protection up to 2 kV on all input/output (I/O) lines.

The NET485 has jumper terminals for adding termination resistors to the RX and TX lines. Add these jumpers **ONLY** if you have long transmission lines and termination resistors are needed.

WARNING: Jumpers must be installed vertically.

Note: Do NOT use RX Term and TX Term jumpers on short transmission lines. Remove these jumpers to remove the 120 Ohm resistors from the transmit and receive lines.

Note: Only use one jumper when wired for RS485 two-wire mode (Figure 2 above).

2.2.4 Full Duplex/Half Duplex Jumper

The NET485 is factory set for RS485 2-wire mode. You can change it to RS422 by changing the protocol for the port setting. See Serial and Mode Settings on page 6-34. You can select Full or Half Duplex by changing the internal jumper J2. The factory default setting is Half Duplex, pins 2 and 3 are connected.

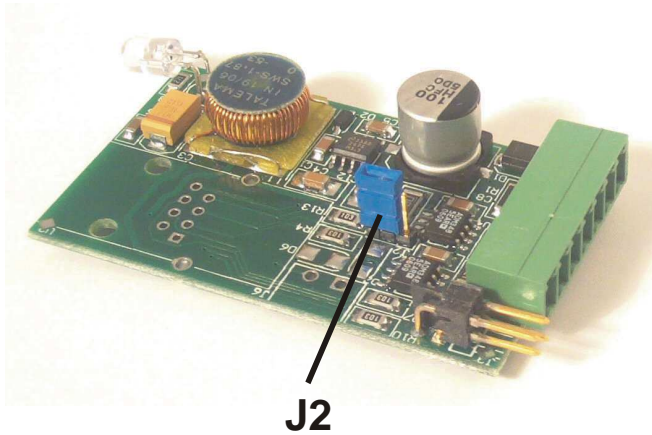


Figure 7 - J2 Setting for Half Duplex

To change the jumper J2 to Full Duplex, open the case and locate the jumper J2. Move the jumper to pins 1 and 2 as shown in the drawing.

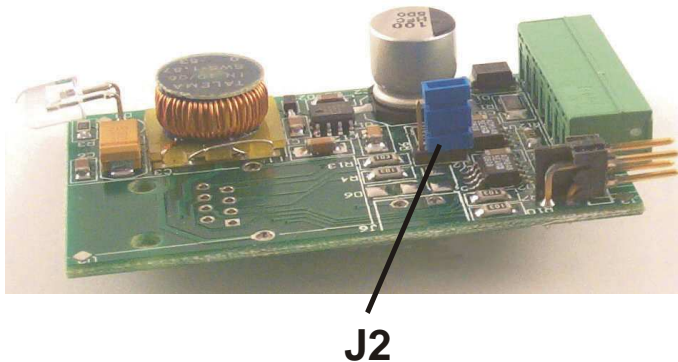


Figure 8 - J2 Setting for Full Duplex

2.3 WI232

2.3.1 Serial RS232 Interface

The table below lists the RS232 signals for the Wi232. The RS232 interface is a 9-pin Male D-style connector (DB9M), configured as a DTE device. DCE configured devices are also available on special order.

Table 3 - RS232 Signals

Wi232 Signal	Direction	DTE DB-9 Male Pin #	DCE DB-9 Female Pin #
Data Out (TXD)	Out	3	2
Data In (RXD)	In	2	3
Ground		5	5
RTS	In	8	7
CTS	Out	7	8
DTR	Use J5 to jumper 4-6	4	6
DSR	Use J5 to jumper 4-6	6	4

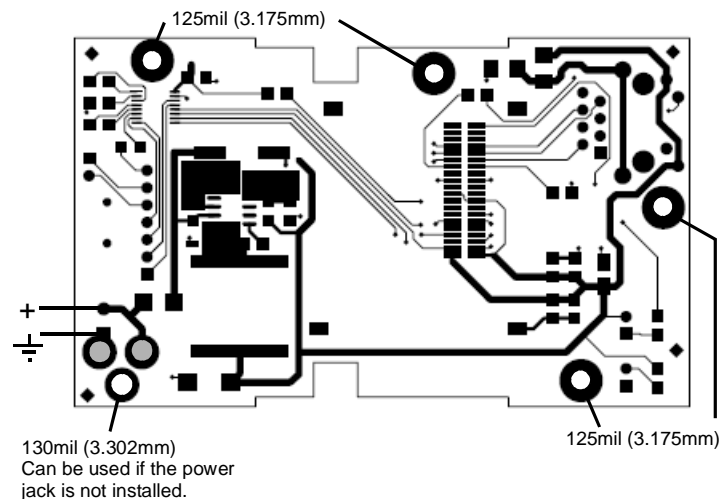
Note: J5 is used to jumper cable pins 4 to 6, which ties DTR to DSR. To locate J5, open the case by removing the two screws. Use the drawing below to locate the jumper.

2.3.2 OEM Module

If you order the Wi232 as an OEM board, you can mount the board to your device and connect the signal wires using the following drawings.

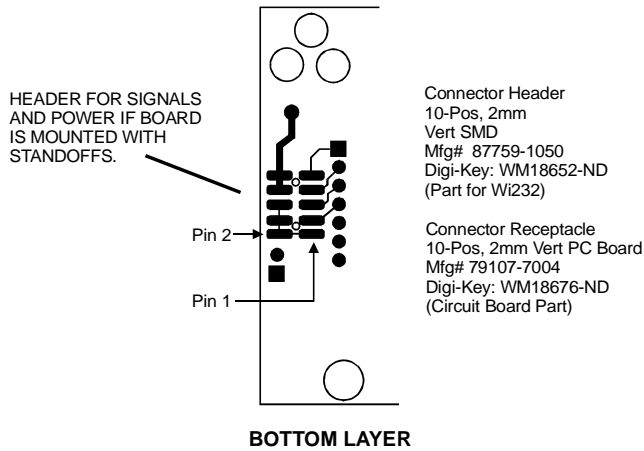
Mounting Holes

The following drawing shows the mounting holes for the Wi232 OEM board. If the power jack is removed, one of the holes can be used for mounting the board.



Header Connector

The header solder pads are located on the bottom layer of the board. The following drawing shows the location of the pads. Note the part number and source for the connector header and receptacle. Use .250" standoffs for proper spacing.



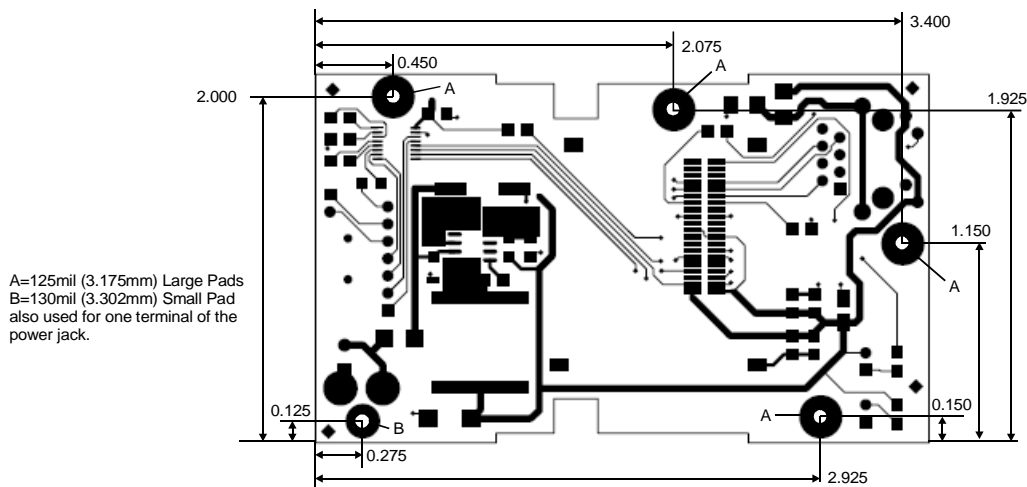
The pads are wired according to the following table.

Table 4 - OEM Header Wiring

Pin #	Description
1	Ground
2	Ground
3	CTS (Out)
4	Ground
5	RTS (In)
6	Ground
7	TXD (Out)
8	V + (In)
9	RXD (In)
10	V+ (In)

Board Dimensions

The dimensions for the mounting holes are shown in the following drawing.



2.4 Embedded Modules

Embedded modules include Xport-MB, Xport-Direct+-MB, and WiPort-MB

For full information on modules regarding hardware please refer to their corresponding integration guides.

2.5 Power Supply

The NET232 can use any DC power source from 5VDC to 24VDC, marked LPS or Class 2. A typical power cube sent with the unit can supply 9VDC at 500 mA. However, there are other units that can be used as long as they are in the range of 5-24VDC and supply the proper wattage. At 9VDC, the NET232 will draw approximately 110mA (.99W) so a 2 Watt power source (9V at 200mA) should be adequate.

NOTE: The NET232 is designed to be used with any properly rated power adapter from 5VDC to 24VDC, 2W maximum, marked Class 2 or LPS.

NOTE: The NET232 power adapter is a 2.1mm positive center power jack. The jack is equivalent to a CUI Inc. PJ-002A power jack.

Grid Connect can supply a special cable adapter to connect the NET232 to a USB jack for +5VDC power.

You can also order the NET232 with a Phoenix right angle power connector. The unit is supplied with a mating Phoenix screw terminal block plug.

The NET485 can also use a DC power source from 5VDC to 24VDC, even though the label shows 8-24VDC. The current draw is determined by network activity and serial port communications. In general, a 2.5W supply will handle the load.

Most modular power supplies use the same method of designating which lead is positive and which one is negative. Generally, the lead with a white stripe, or white markings, is the positive lead. Verify the lead markings with a meter before connecting a power source to the NET485.

Connect the positive lead to the terminal marked 8-24VDC. Connect the negative lead to the terminal marked GND. The power LED will come on when power is supplied.

The unit will go through a self-test and will attempt to connect to a server. The LEDs on the Ethernet connector will indicate the connection status.

2.6 Ethernet Interface

The NET485/NET232 device contains the following LEDs:

- 10BaseT/100BaseTX (Bi-color, Left LED)
- Full/Half Duplex (Bi-color, Right LED)

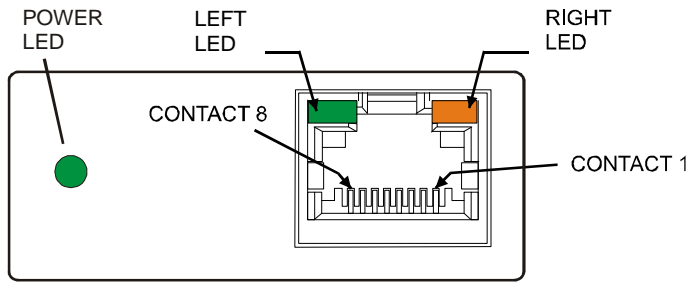
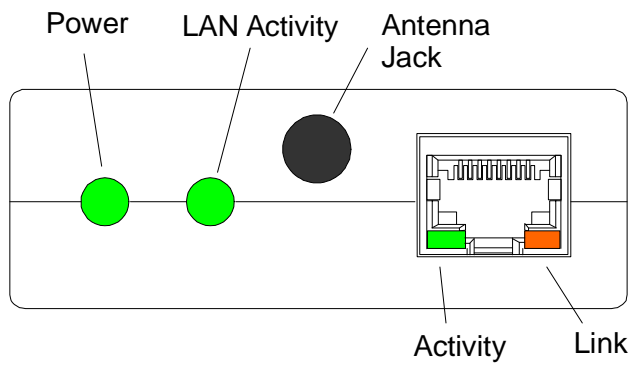


Figure 9 - Ethernet Jack

Table 5 - NET485/NET232 LED Functions

Left LED	Right LED	Meaning
Solid Amber		10BASE-T
Solid Green		100BASE-Tx
	Blinking Amber	Half Duplex Activity
	Blinking Green	Full Duplex Activity

The WI232 device contains the following LEDs



3. Modbus

When it comes to data communications for industrial control systems, Modbus™ is supported by the most manufacturers. The Modbus/RTU protocol defines how a “master” device polls one or more “slave” devices to read and write data in real time over RS-232, RS-422, or RS-485 serial data communication. Although not the most powerful protocol available, its rare simplicity allows not only rapid implementation but also enough flexibility to be applied in a large number of industrial situations. Modbus/TCP, an extension of Modbus/RTU, defines how Modbus/RTU and Modbus/ASCII messages are encoded within and transported over TCP/IP-based networks. Modbus/TCP is just as simple to implement and flexible to apply as the original Modbus/RTU. You can find the specification for both online at www.modicon.com.

The XPort-MB Device Server allows users to integrate new and existing Modbus/RTU and Modbus/ASCII serial devices to newer TCP/IP network-based devices. The next section describes a system that integrates four Modbus/RTU devices with four Modbus/TCP devices.

TM Modbus is a registered trademark of Schneider Automation.

3.1 Extended Modbus System Example

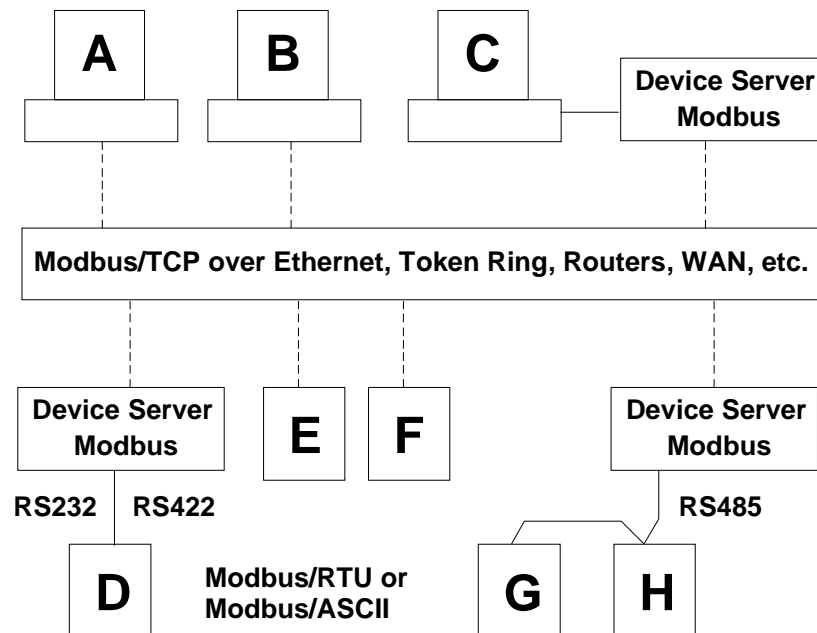


Figure 10 - Extended Modbus System Example

In Figure 0, we can see four specific styles of Modbus operations. Modbus/RTU devices are traditionally split into two groups.

Modbus slave devices generally are the workhorse devices. Often industrially hardened, they tirelessly perform their tasks 24 hours a day, 365 days a year. They perform tasks such as flow metering, temperature control, batch loading, or even running entire automated assembly lines. The slave devices are not called “slaves” because they work all the time; they are called slaves because as far as the data communications is concerned, they function as passive servers. Modbus slave devices passively sit and wait for a remote Modbus master device to ask them to report existing data values (Read) or accept new data values (Write).

Modbus master devices generally are higher-level computers, devices in which data and software are very important. The most common examples of Modbus master devices are the “Human-Machine-Interface” (HMI) computers, which allow human operators to monitor, adjust, and maintain the operations of the field devices. Modbus master devices are clients that actively go out and “Read” from and/or “Write” to remote Modbus slave devices to monitor or adjust slave behavior.

3.1.1 Modbus/TCP Master Talking to Modbus/TCP Slave

Devices A, B, E, and F are all new Modbus/TCP devices, which are improved over Modbus/RTU (see more about Modbus/RTU limitations below). All 4 devices can function concurrently as both Modbus master and Modbus slave. Both computers A and B can treat controller E as a slave, polling data in real-time. Yet controller E can also act as a master and poll data from controller F, which can in turn also act as a master to write alarm data directly up to computers A and B to alert the operators to the alarm condition. Traditional Modbus/RTU requires slave devices even with life threatening alarm conditions to sit patiently and wait for a remote master to poll the specific data that caused the alarm condition.

It is really revolutionary for such a simple and flexible protocol as Modbus to offer such functionality. Therefore, Modbus/TCP offers exciting new design options for industrial users, which the Device Servers extend to traditional Modbus/RTU serial devices.

3.1.2 Modbus/TCP Master Talking to Modbus/RTU Serial Slave

Devices D, G, and H are traditional Modbus/RTU slave devices. Device D uses a point-to-point electrical interface like RS-232. This allows only a single Modbus/RTU master to talk to device D. However, the Device Server makes device D appear on the Modbus/TCP network as a full Modbus/TCP slave device. All Modbus/TCP enabled devices, A, B, E, and F, can actively share access to slave device D. A limitation in traditional Modbus/RTU implementation expects devices to be dedicated as either master or slave devices, so device D can only act as a Modbus slave.

Devices G and H are different from device D. They share a single RS-485 “multi-drop” line that strictly limits them to act as slaves to a single Modbus/RTU master. However, a little of the new Modbus/TCP and Device Server magic still applies—all Modbus/TCP enabled devices A, B, E, and F can actively share access to both slave devices G and H. The Device Server manages and coordinates the shared access. In fact, the Device Server allows up to ten concurrent Modbus masters to share access to the slaves.

3.1.3 Modbus/RTU Serial Master Talking to Modbus/TCP Slave

Device C is a traditional Modbus/RTU master device. Yet the Device Server makes device C appear to the TCP/IP network as a Modbus/TCP master—plus all of the Modbus/TCP slaves on the TCP/IP network (A, B, D, E, F, G, and H) appear as traditional Modbus/RTU slave devices. The only limitation is the traditional Modbus/RTU assumption that device C is dedicated as a master only. Therefore Modbus/TCP master devices A, B, E, and F cannot treat device C as a Modbus/TCP slave.

3.1.4 Modbus/RTU Serial Master Talking to Modbus/RTU Serial Slave

Finally, master device C can poll traditional Modbus/RTU slave devices D, G, and H as if they were directly multi-dropped on an attached RS-485 line. The Device Server transparently bridges traditional Modbus/RTU devices across any TCP/IP network. This means users can start implementing for Modbus/TCP long before all of their required products exist with Modbus/TCP and network interfaces.

3.2 Network Protocols

The XPort-MB Device Server uses TCP/IP protocols for network communication. The supported standards are: ARP, UDP, TCP, ICMP, Telnet, TFTP, DHCP, and SNMP. For transparent connections, TCP/IP (binary stream) or Telnet protocols are used. Firmware upgrades can be made with the TFTP protocol.

The industrial protocol defines addressing, routing and data block handling over the network. The TCP (transmission control protocol) assures that no data is lost or duplicated, and that everything sent into the connection on one side arrives at the target exactly as it was sent.

For typical datagram applications where devices interact with others without maintaining a point-to-point connection, UDP datagram is used.

3.3 Packing Algorithm

Traditional Modbus/RTU requires a “character time-out” to signal the end of a Modbus/RTU packet. This stretches out the overall response cycle. Fortunately, the XPort-MB Device Server uses an intelligent length-predictive algorithm to detect the end of standard Modbus messages. This allows better performance plus the XPort-MB Device Server falls back to using a user definable “character time-out” to manage non-standard or user-defined Modbus functions.

3.4 IP Address

Every device connected to the TCP/IP network including the XPort-MB Device Server must have a unique IP address. When multiple Modbus devices share a single IP, then Modbus/TCP includes an additional address called the Unit ID.

When the XPort-MB Device Server is receiving Modbus/TCP messages from remote masters, the Unit ID is converted to use in the Modbus/RTU message as the slave address.

When the XPort-MB Device Server is receiving Modbus/RTU messages from local serial masters, a user-defined lookup table is used to match the 8-bit Modbus slave address to a remote IP address. The Modbus slave address received is used as the Unit ID.

3.5 Configuration Methods

The XPort-MB Device Server can be configured using remote or local methods. Either use an ASCII terminal or a terminal emulation program to locally access the serial port, or use a Telnet connection to port 9999 to configure the unit over the network using the Setup Menu. Telnet Configuration is also an option inside DeviceInstaller.

You may also use a web browser to access the Device Server’s Web-Manager. You can access Web Configuration using DeviceInstaller or simply type the IP address of the Device Server into your browser’s address bar.

The XPort-MB Device Server configuration is stored in nonvolatile memory and is retained without power. The configuration can be changed any time. The XPort-MB Device Server performs a reset after the configuration has been changed and stored.

3.6 Device Server's IP Address

The XPort-MB Device Server is shipped with a default IP address of 0.0.0.0, which automatically enables DHCP within the XPort-MB Device Server.

With a DHCP-enabled Device Server, if there is a DHCP server to respond to the Device Server's request when it's booting up, the Device Server will then get an IP address, a gateway address, and a subnet mask from the DHCP server. These addresses will not be shown in the Device Server's Setup (configuration) screens (you will still see 0.0.0.0), however if you enter the Monitor Mode and from 0> prompt, type NC (upper case), the IP configuration of the Device Server will display. (See the Monitor Mode and Firmware Upgrade chapter.)

3.7 Device Installer

The XPort-MB is shipped without software or hardware manuals. Device Installer software and documentation can be downloaded from the Lantronix web site. See www.lantronix.com. The XPort-MB user manual can be downloaded from the Grid Connect web page.

If you purchased a NET232-MB, Device Installer software and user manuals are provided on the supplied CD. See the *NET232 User Manual* for Device Installer installation instructions. See the *Device Installer User Manual* for operation details. Since the NET232 MB has different firmware, the operation of Device Installer will be different.

Device Installer uses .NET Framework to adapt XPort embedded servers for Web services. Your system must have .NET Framework installed for Device Installer to work properly. XPort-MB users can download .NET Framework from the Lantronix web page. See www.lantronix.com. NET232-MB users can install the software from the supplied CD.

Device Installer looks for devices on the network and identifies them according to a device code in the firmware. Device Installer looks at the device code from all devices found and compares them to a list of acceptable codes. The list is periodically updated and Device Installer will remind you to check for updates.

3.7.1 RUN Device Installer

Click the Start button on the Task Bar and select **Programs\DeviceInstaller**. From the list of options, select **DeviceInstaller**.

The Device Installer main dialog box appears. The program automatically searches for devices.

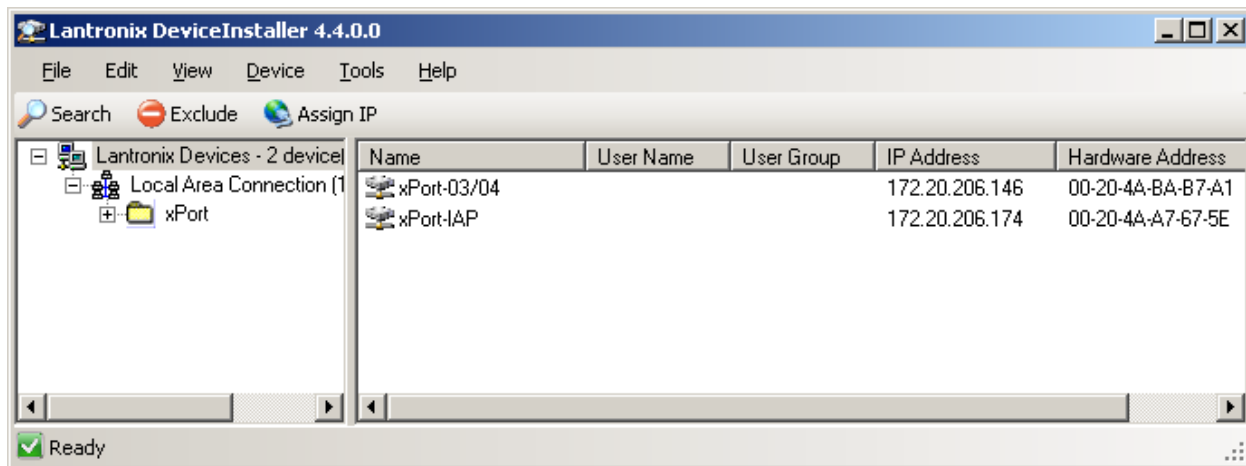
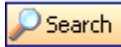


Figure 11 - Device Installer Dialog Box

To search for devices, click the **Search** icon  or select **Search F5** from the Device menu.

3.7.2 Device Found

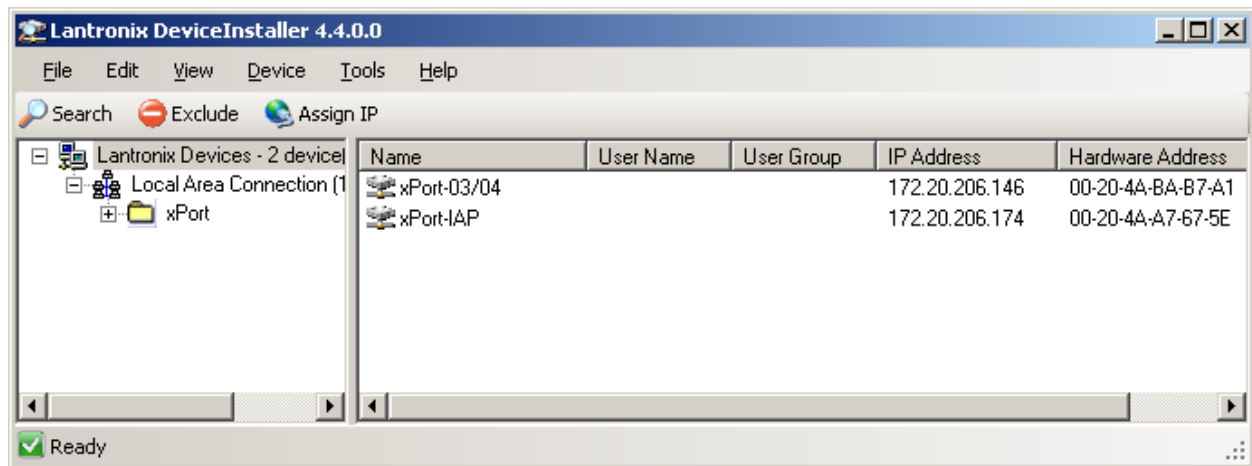


Figure 11 shows two devices found on the network, with the IP addresses assigned by the DHCP server. The device IP Address is normally set to 0.0.0.0 at the factory. The Hardware Address is an individual permanent address assigned to a particular device on the network. The Hardware Address can be found on the product label.

Note: The unit Type shown as XPort-IAP is the Modbus device.

Double-click the XPort-IAP in the Device Installer window to display the expanded window shown in Figure 12. The **Device Details** tab is automatically selected and will display information about the selected device.

Note: The ERROR displayed by DeviceInstaller's Device Details is reported because the Modbus firmware has different setup record data than the standard serial tunneling firmware.

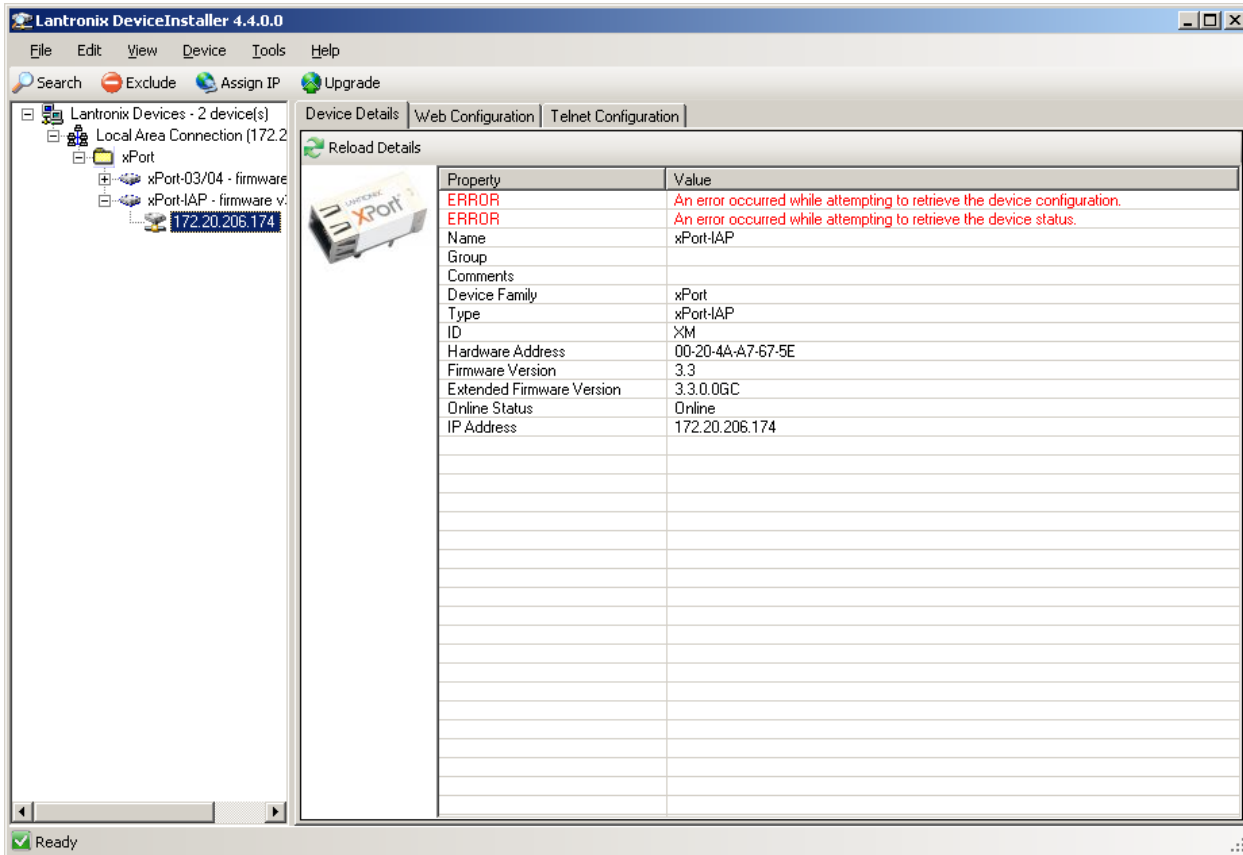


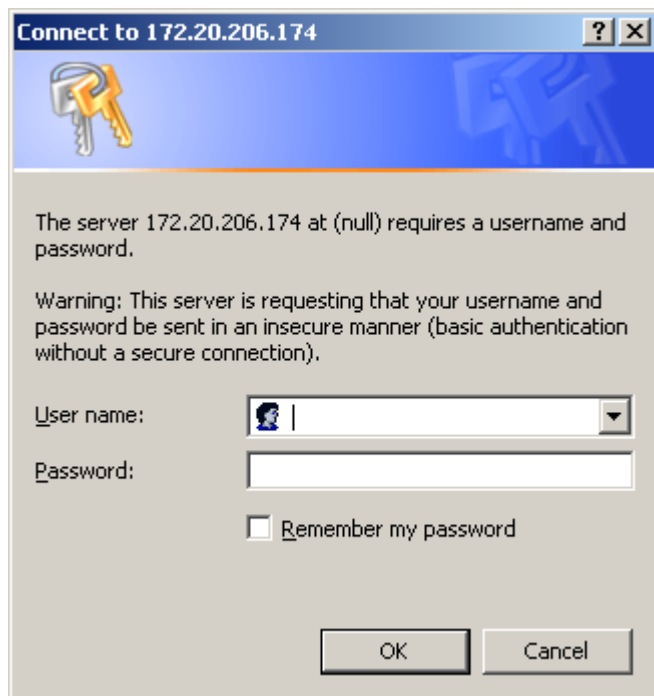


Figure 12 Device Details

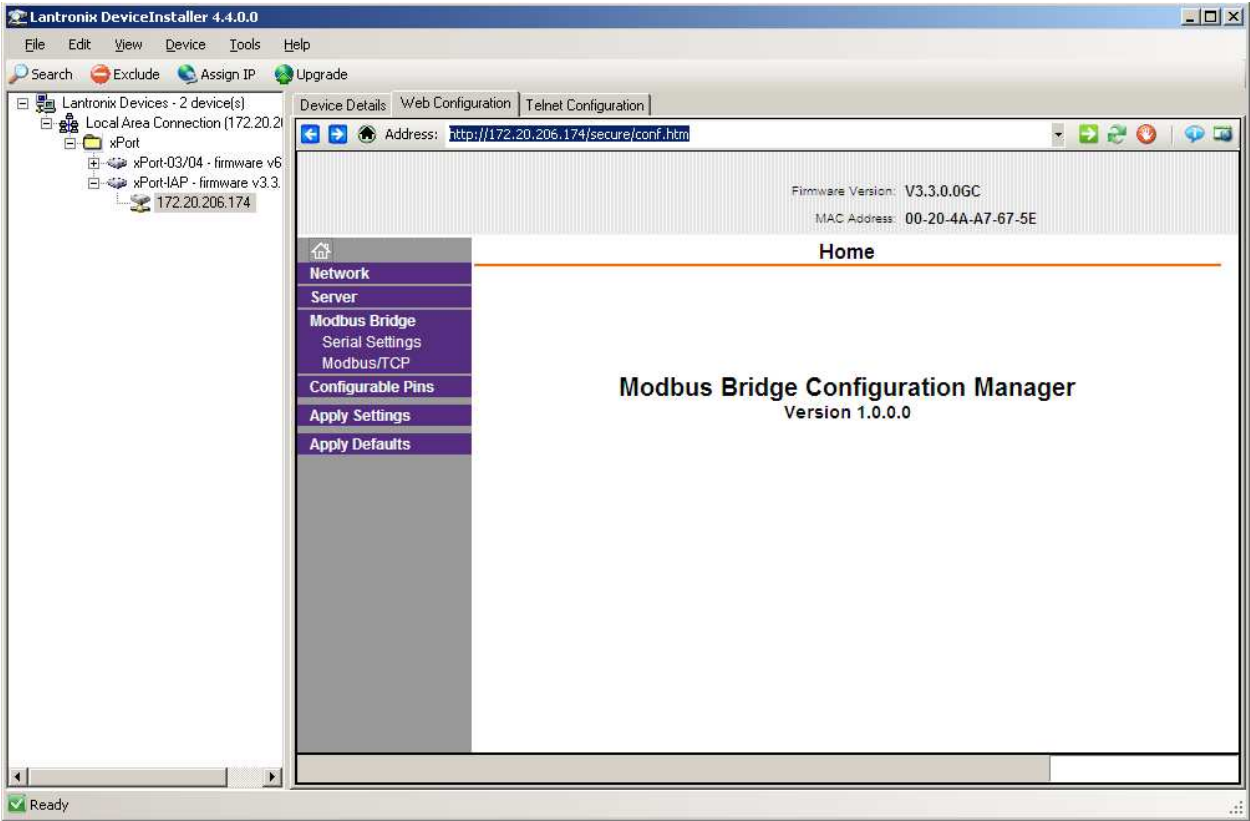
4. Web Configuration

From the DeviceInstaller Utility, click the Web Configuration tab. The IP address and Port number will be displayed. To view the Web-Manager in the current DeviceInstaller window, click the **Navigate** icon . To open the Web-Manager in your default web browser (recommended method), click the **External Browser** icon .

If a password window appears, leave the User name and Password fields empty and click **OK** or press **Enter**. By default the Device Server configuration will not have password protection. If a password has been installed, then you would need to enter it here.



You should then see the Web-Manager Home screen.



The main menu is in the left pane of the Web-Manager window.
Click on one of the configuration page links in the main menu.

Network Settings

Network Mode:

IP Configuration

☒ Obtain IP address automatically

Auto Configuration Methods

BOOTP: ☒ Enable ☐ Disable

DHCP: ☒ Enable ☐ Disable

AutoIP: ☒ Enable ☐ Disable

DHCP Host Name:

☐ Use the following IP configuration:

IP Address:

Subnet Mask:

Default Gateway:

Done!

Click **OK** after completing changes on a configuration page.

Click **Apply Settings** when all page changes are **Done** and the updated configuration settings will be written to the Device Server.

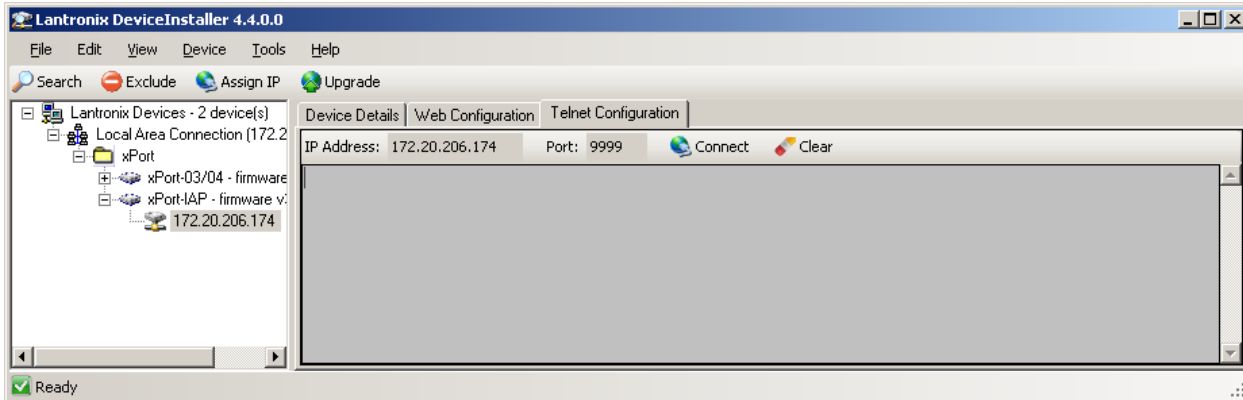
Apply Settings

Please wait while the configuration is saved...

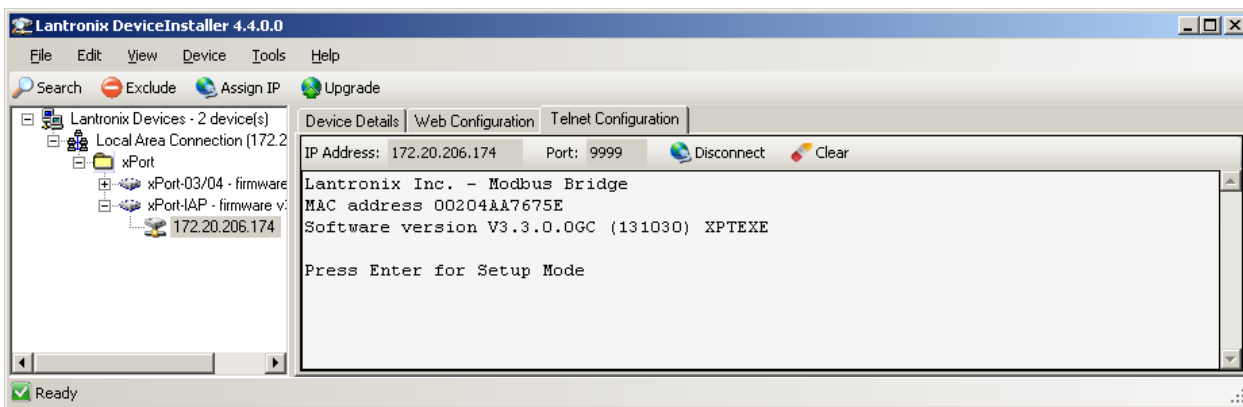
The unit will reboot in order for the settings to be applied.

5. Telnet Configuration

1. From the DeviceInstaller Utility, click the Telnet Configuration button. The IP address and Port number will be displayed. Click the **Connect** button to initiate the connection.



2. You'll see the following lines, which tell you the Device Server's Ethernet MAC address (hardware address).



3. Within 5 seconds, press Enter to display the Setup (configuration) Mode screen. Here you can change the parameters that define how the XPort-MB Device Server does its job.

Note: When you set up a new unit, and especially if you just reflashed the unit with a new firmware type, we recommend that you reset all of the parameters to the factory defaults.

4. To reset the parameters to the factory defaults, type **D** on the command line and press Enter. The default parameters display.
5. Select an option on the menu (1-7) by typing the number of the option.
6. To enter a value for a parameter, type the value and press Enter, or to confirm a default value, press Enter.
7. Review your entries.
8. You have the following options:

To save the configuration and exit, type **S** on the command line and press **Enter**. This saves the parameters to EEPROM.

Caution: *DO NOT POWER CYCLE the unit too fast after doing this. Allow the unit to reboot naturally one time first.*

To quit without saving, type **Q** on the command line and press Enter. The unit reboots.
To restore the default values, type **D** on the command line and press Enter.

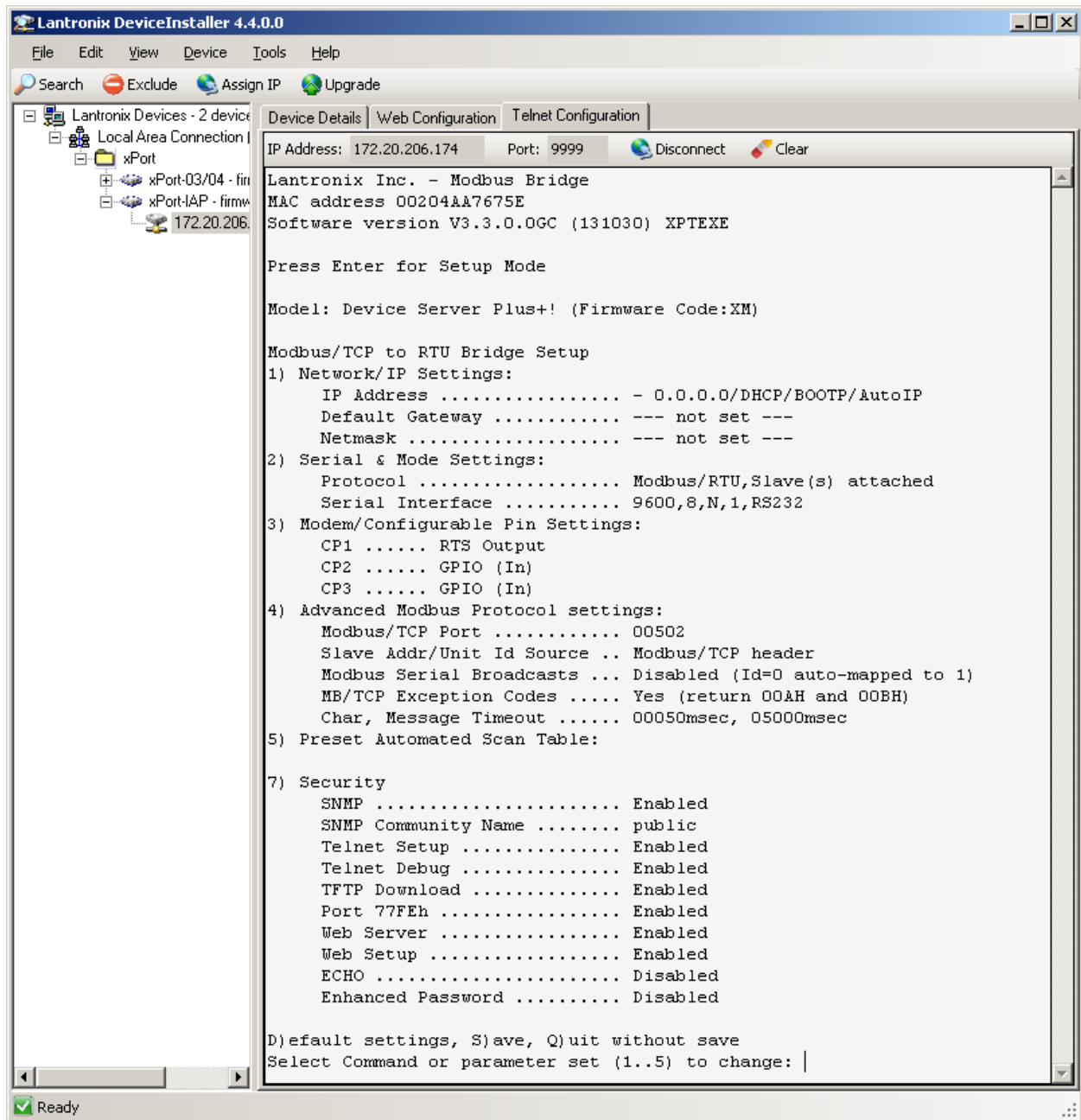


Figure 13 - Setup (Configuration) Mode Screen

5.1 Basic Commands (D/S/Q)

The main Device Server configuration menu is shown above. The Device Server offers three basic options.

5.1.1 Default Settings (D)

Entering **D** resets all parameters to the factory default as shown above. Only the IP address is not changed. Although not required, selecting this option immediately after reloading the firmware and saving it ensures that the unit is reset.

5.1.2 Save (S)

Entering **S** saves the currently displayed parameter settings into non-volatile memory and exits configuration mode. This option will trigger a reset.

5.1.3 Quit Without Saving (Q)

Entering **Q** throws away any parameter changes you have made and exits configuration mode. This option will trigger a reset.

6. Configuring Modbus

6.1 Network/IP Settings

Select **1** in the Telnet Setup menu to configure the Device Server's network parameters. In the Web-Manager click **Network** in the main menu.

The screenshot shows a web browser window with the title "Network Settings". On the left is a vertical navigation menu with the following items: "Network" (highlighted in yellow), "Server", "Modbus Bridge" (with sub-items "Serial Settings" and "Modbus/TCP"), "Configurable Pins", "Apply Settings", and "Apply Defaults". The main content area is titled "Network Settings" and contains the following configuration options:

- Network Mode:** A dropdown menu currently set to "Wired Only".
- IP Configuration:**
 - ☒ Obtain IP address automatically
 - Auto Configuration Methods:**
 - BOOTP:** ☒ Enable ☐ Disable
 - DHCP:** ☒ Enable ☐ Disable
 - AutoIP:** ☒ Enable ☐ Disable
 - DHCP Host Name:** An empty text input field.
 - ☐ Use the following IP configuration:
 - IP Address:** An empty text input field.
 - Subnet Mask:** An empty text input field.
 - Default Gateway:** An empty text input field.

At the bottom right of the configuration area is an "OK" button.

The following values can be set/changed. To understand and select the appropriate values, consult one of the many TCP/IP books available today and your network administrator.

6.1.1 IP Address

The IP address must be set to a unique value on your network. If you are not familiar with IP addressing on your network, please consult your system administrator. Please refer to the User Guide for your Device Server for more details about IP addresses.

If the Device Server is set to an address already in use, it will display an error code with the LEDs and it will not operate properly. If you understand and plan to use DHCP, set the IP to 0.0.0.0 to activate DHCP.

6.1.2 Set Gateway IP Address (Y/N)

Most users could select **N** for this case. You only need to choose **Y** if your Device Server must communicate to remote TCP/IP networks through a router or gateway. If you select **Y**, you must also enter the IP address of the default gateway within your local network.

6.1.3 Set Netmask (N for default)

Most users could select **N**, which causes the Device Server to automatically use the standard netmask appropriate for the IP address you've entered. Users who want a non-standard netmask need to enter the new subnet mask in the traditional form, for example, 255.255.248.000. The selecting of correct IP ranges and subnet masks IS a large enough topic to fill a whole book – we cannot cover it here.

6.1.4 Telnet/Web Configuration Password

A telnet/web configuration password can be set to disable unauthorized access to the setup menu via a Telnet connection to port 9999 or to the configuration web manager. It is not necessary to enter a password to use the setup menu through the serial port.

In the Web-Manager click **Server** in the left-side menu to set the password. The Server page also allows you to configure other advanced network and Device Server options.

Server Settings

Server Configuration

Enhanced Password: ☐ Enable ☒ Disable

Telnet/Web Manager Password:

Retype Password:

Advanced

ARP Cache Timeout (secs):

TCP Keepalive (secs):

Monitor Mode @ Bootup: ☒ Enable ☐ Disable

CPU Performance Mode: ☐ Low ☒ Regular ☐ High

HTTP Server Port:

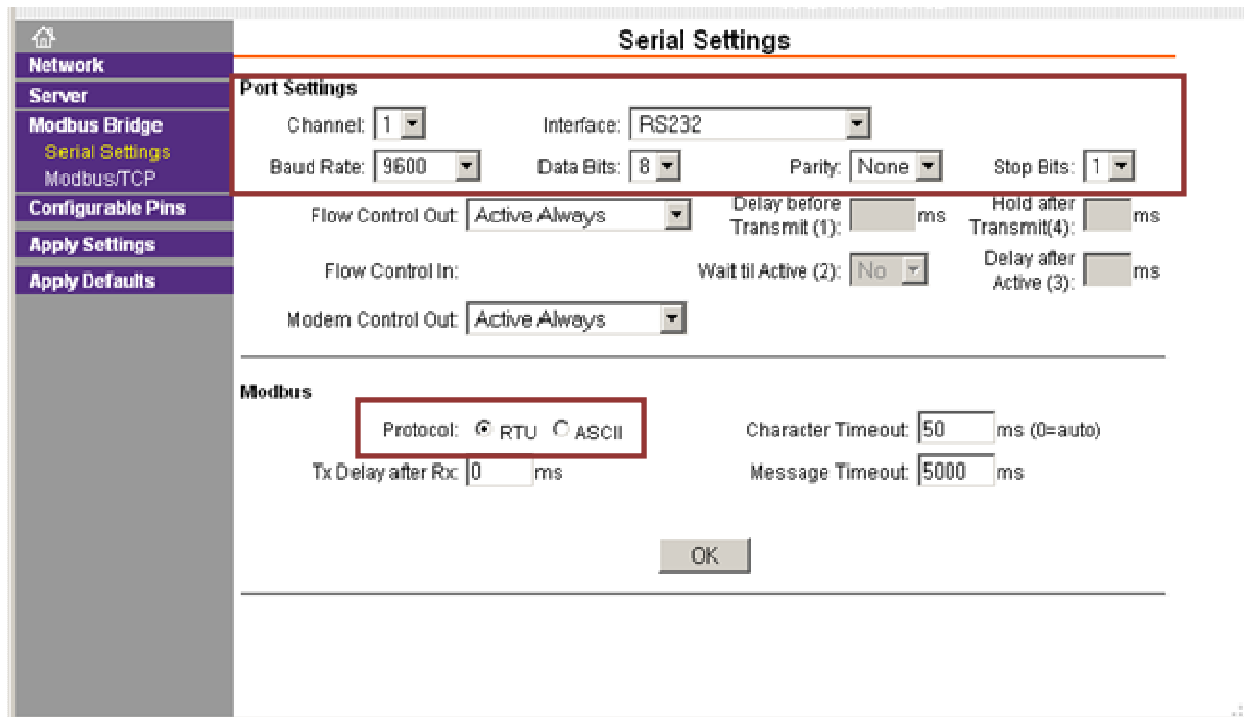
MTU Size:

TCP Re-transmission timeout (ms):

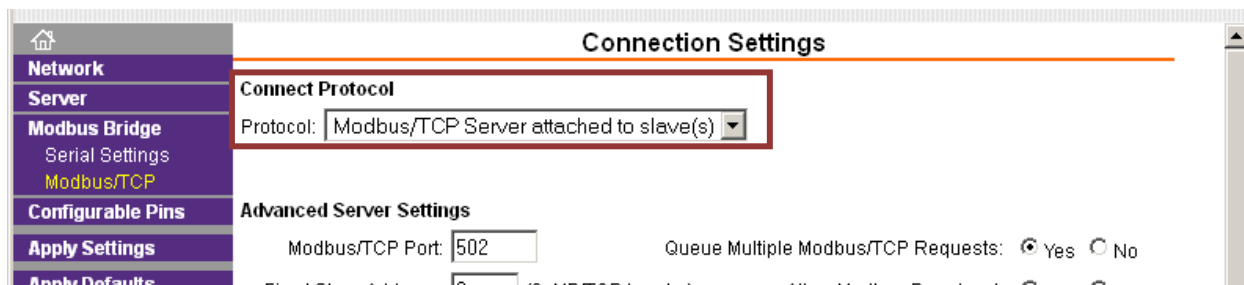
OK

6.2 Serial and Mode Settings

Select **2** in the Telnet Setup menu to change the basic serial parameters. In the Web-Manager click **Serial Settings** in the main menu. The following values can be set/changed.



The screenshot shows the 'Serial Settings' web interface. On the left is a navigation menu with options: Network, Server, Modbus Bridge, Serial Settings (highlighted), Modbus/TCP, Configurable Pins, Apply Settings, and Apply Defaults. The main content area is titled 'Serial Settings' and contains two sections. The 'Port Settings' section is enclosed in a red box and includes fields for Channel (1), Interface (RS232), Baud Rate (9600), Data Bits (8), Parity (None), Stop Bits (1), Flow Control Out (Active Always), Delay before Transmit (1) (0 ms), Hold after Transmit (4) (0 ms), Flow Control In, Wait til Active (2) (No), Delay after Active (3) (0 ms), and Modem Control Out (Active Always). The 'Modbus' section includes a red box around the Protocol field, which has radio buttons for RTU (selected) and ASCII. Other fields in this section are Character Timeout (50 ms), Tx Delay after Rx (0 ms), Message Timeout (5000 ms), and an OK button at the bottom.



The screenshot shows the 'Connection Settings' web interface. The left navigation menu is the same as in the previous screenshot, with 'Modbus/TCP' highlighted. The main content area is titled 'Connection Settings' and contains two sections. The 'Connect Protocol' section is enclosed in a red box and includes a dropdown menu for Protocol, currently set to 'Modbus/TCP Server attached to slave(s)'. The 'Advanced Server Settings' section includes fields for Modbus/TCP Port (502), Queue Multiple Modbus/TCP Requests (radio buttons for Yes and No, with Yes selected), and a partially visible field for First Slave Address.

6.2.1 Attached Device (1=Slave, 2=Master)

As mentioned in the introduction, Modbus/RTU devices are defined as either slave or master devices. Type **1** if the attached device is a slave (such as a sensor, meter or PLC) or **2** if the attached device is a master (such as a PLC or computer running graphical human-machine-interface (HMI) software).

6.2.2 Serial Protocol (1=Modbus/RTU, 2=Modbus/ASCII)

Serial Modbus comes in two forms. Modbus/RTU uses 8-bit data bytes to send binary information. However, some devices cannot handle 8-bit data bytes, so Modbus/ASCII is used. Modbus/ASCII is a slower protocol where each 8-bit data byte is converted to 2 ASCII characters. Since the Device Server converts both to and from Modbus/TCP fully, you can mix any combination of RTU and ASCII devices on a Modbus/TCP network. So a Modbus/RTU Master attached to one Device Server can remotely access a Modbus/ASCII slave attached to another Device Server.

6.2.3 Interface Type (1=RS232 2=RS422/RS485+4-wire 3=RS485+2-wire)

This allows the Device Server to deal with the software-related details of using RS-232, RS-422, and RS-485.

6.2.4 Enter Serial Parameters (9600,8,N,1)

Enter the baud rate, data bits (7/8), parity (N/O/E), and stop bits (1/2) in the classic “DOS Mode Command” style. Examples are: 9600,8,E,1 or 1200,7,O,2. This setting must match the setting on the attached Modbus device.

You will be warned if you try to set an unsupported combination of settings.

Note: After reset, the device server will check the serial port at 9600, 8, N, 1 for 5 seconds.

Note: Do NOT use the Configure button for changing the Baud Rate.

6.3 Modem Control Settings

When using RS232, the Device Server has a number of user-definable “Modem Control” parameters to manage RTS/CTS handshaking for half-duplex radio modems.

Serial Settings

Port Settings

Channel: 1 Interface: RS232

Baud Rate: 9600 Data Bits: 8 Parity: None Stop Bits: 1

Flow Control Out: Active Always Delay before Transmit (1): ms Hold after Transmit (4): ms

Flow Control In: Wait til Active (2): No Delay after Active (3): ms

Modem Control Out: Active Always

Modbus

Protocol: ☒ RTU ☐ ASCII Character Timeout: 50 ms (0=auto)

Tx Delay after Rx: 0 ms Message Timeout: 5000 ms

OK

6.3.1 RTS/CTS Mode (1=Fixed 2=Variable)

Answer **1** and the Device Server output is fixed to high. Answer **2** to enable modem handshaking. The RTS/CTS output is active when the device server is transmitting on the serial port. This setting is very different from the hardware or RTS/CTS flow-control used with printers. This mode cannot work with a direct RS232 cable, as each end only asserts its RTS control signal to power up intermediate transmitters.

6.3.2 Delay after Output of RTS (0-1275 ms, 5ms increments)

Only asked if RTS/CTS mode is variable. After the Device Server asserts the RTS/CTS signal, it delays from 0 to 1275 ms before continuing. Normally this is set to 0. Only set a value here if your device, modem, or interface requires extra initialization time before receiving serial data.

6.3.3 Wait for CTS to Go Active (N/Y)

Only asked if RTS/CTS mode is variable. Answering **N** causes the Device Server to ignore the RTS/CTS response from the modem. Answering **Y** causes the Device Server to wait for the RTS/CTS response from the modem. Do not answer **Y** unless you know that the cable is wired properly to support this signal.

6.3.4 Delay after CTS Going Active (0-1275 ms, 5ms increments)

Only asked if RTS/CTS mode is variable and set to wait for CTS to go active. After the Device Server sees the modem assert an RTS/CTS response input, it delays from 0 to 1275 msec before transmitting. If the Device Server waits without seeing a valid response from the modem, it will return the Modbus exception response 0x0B (hex) to the Modbus/TCP requesting master.

6.3.5 Delay Dropping RTS after Transmitting (0-1275 ms, 5 ms increments)

Only asked if RTS/CTS mode is variable. After the Device Server completes transmission, it delays from 0 to 1275 msec before dropping the RTS/CTS output.

6.3.6 DTR Mode (1=Fixed, 2=Active with connection)

Answer **1** and the Device Server DTR output is fixed to active. Answer **2** and the DTR output is active only when at least one Modbus/TCP client is connected.

6.4 Modem/Configurable Pin Settings

Embedded modules, such as the XPort-MB, have configurable pins that can be set to perform different functions, including modem control functions and general purpose I/O (GPIO). The menu option for Modem Control Settings has been replaced with Modem/Configurable Pin Settings on embedded modules.

Configurable pins assigned as GPIOs can be written and read via Modbus/TCP when in slave attached mode. A Modbus slave address of 255 and starting offset of 0001 are used to direct Read Coil Status, Read Input Status, Force Single Coil and Force Multiple Coils Modbus commands to the embedded module's GPIO. Other commands or unmatched addressing are directed to the serial port.

6.4.1 NET232/NET485/XPort-MB

The Modbus Master/Slave functionality on the XPort-MB is very similar to the Modbus implementation on other device servers. The major difference is that the configurable pins on the XPort-MB (CP1-3) can be configured from the setup menu or web manager.

CP	Function	Direction	Active Level
1	HW Flow Control Out	<input checked="" type="radio"/> Input <input type="radio"/> Output	<input type="radio"/> Low <input checked="" type="radio"/> High
2	General Purpose I/O	<input checked="" type="radio"/> Input <input type="radio"/> Output	<input type="radio"/> Low <input checked="" type="radio"/> High
3	General Purpose I/O	<input checked="" type="radio"/> Input <input type="radio"/> Output	<input type="radio"/> Low <input checked="" type="radio"/> High

OK

6.4.1.1 CP1 Function

This configurable pin can be set to:

CP1 Function (1=GPIO, 2=Status LED Output, 3=RTS Output, 4=RS485 Enable)

Note: Status LED Output is only valid for XPort-MB devices.

Note: CP1 should be set to RS485 Output Enable (active high) for NET485-MB products.

Selecting **GPIO** (General Purpose IO) for CP1 will prompt the user for Input or Output directions.

GPIO (1=Input, 2=Output) (1)

Selecting **GPIO** for any CP option will later prompt you for the active level.

Invert GPIO (active high) (Y)

The **Status LED Output** will provide a status indicator. Normally, this signal is tied to an LED on your device. The LED will blink 4 times, then be off for two seconds to indicate the setup mode is active through the serial port.

Selecting **RTS Output** for CP1 will prompt the user for additional RTS and flow control options.

RTS Mode 1 = Fixed, 2 = Active with transmit.

Selecting **Fixed** will complete the CP1 setup and the menu drops to CP2.

Selecting **Active with Transmit** will display the following option:

Delay after output RTS (0-1275 msec, 5ms resolution) (0)

After the NET232/USB MB asserts the RTS/CTS signal, it delays from 0 to 1275 msec before continuing. Normally this is set to **0**. Only set a value here if your device, modem or cable is non-standard.

Wait for CTS (CP3) to go active (N)

Selecting **Y** to **Wait for CTS** will display the following option:

Delay after CTS going active (0-1275 msec, 5ms resolution) (0)

After the NET232/USB MB sees the modem assert an RTS/CTS response input, it delays from 0 to 1275 msec before transmitting. If the unit waits without seeing a valid response from the modem, it will return the Modbus exception response 0x0B (hex) to the Modbus/TCP requesting master.

Delay dropping RTS after TX (0-1275 msec, 5ms resolution)

After the unit completes transmission, it delays from 0 to 1275 msec before dropping the RTS/CTS output.

Selecting Wait for CTS will cause CP3 to be auto-configured for CTS Input. You will not be able to configure CP3 and the following message will appear:

CP3 Function already configured for CTS Input

Selecting **RS485 Enable** for CP1.

Selecting **RS485 Output Enable** for CP1 will prompt you for additional control options.

Invert RS485 Output Enable(active low) (N)

The RS485 Output Enable function is used for controlling an external RS485 line driver when in RS485 2-wire mode and this output can be configured for active high (default) or active low.

6.4.1.2 CP2 Function

This configurable pin can be set to:

CP2 Function (1=GPIO, 2=DTR Output, 3=RS485 Output Enable).

Note: CP2 should be set to GPIO Input for NET232-MB products.

Selecting **GPIO** (General Purpose IO) for CP2 will prompt the user for Input or Output directions.

GPIO (1=Input, 2=Output) (1)

Selecting **GPIO** for any CP option will later prompt you for the active level.

Invert GPIO (active high) (Y)

Selecting **DTR Output** for CP2 will prompt you for additional options.

DTR Mode (1=Fixed, 2=Active with connection)

Selecting **RS485 Output Enable** for CP2 will prompt you for additional control options.

Invert RS485 Output Enable(active low) (N)

The RS485 Output Enable function is used for controlling an external RS485 line driver when in RS485 2-wire mode and this output can be configured for active high (default) or active low.

6.4.1.3 CP3 Function

This configurable pin can be set to:

CP3 Function (1=Unused, 2=Diagnostic LED Output, 3=RS485 Output Enable)

CP3 can be forced to CTS Input by answering 'Y' to the **Wait for CTS** option under the CP1 Function menu for **RTS Output**.

Note: Selecting Diagnostic LED Output is only a valid option for XPort-MB products.

The Diagnostic LED Output is a status indicator to indicate the unit is in setup mode. When used in conjunction with the Status LED Output, the following conditions can be monitored.

Condition	LED 3	LED 1
No errors	OFF	ON
Network controller error	ON	Blink 3x/4 sec OFF
Duplicate IP address present	ON	Blink 5x/4 sec OFF
No DHCP response	Blink 2x/sec	Blink 5x/4 sec OFF
Setup menu active	Blink 2x/sec	See Note.

Note: During a Telnet connection, CP1 LED (Status LED) is ON. For a serial port connection, CP1 LED (Status LED) blinks for 2 seconds, then OFF for 2 seconds. (It appears as 4 blinks, then OFF for 2 seconds)

6.4.2 XPort-Direct+-MB

The Modbus Master/Slave functionality on the XPort-Direct+ is very similar to the Modbus implementation on other platforms such as the NET232 or NET485. The major difference is that the configurable pins on the XPort-Direct+ (CP1-CP3) can be configured from the setup menu and web manager. The configurable pin options are as follows:

```
CP1 Function (hit space to toggle) GPIO (In)
CP1 Function (hit space to toggle) GPIO (Out)
CP1 Function (hit space to toggle) Diag LED
CP1 Function (hit space to toggle) Status LED-G
CP1 Function (hit space to toggle) Status LED-Y
```

The assignment for each configurable pin is set by cycling through the menu options by entering a space or any key other than <enter>. GPIO assigns the pin as a general purpose input or output. The GPIOs can be written and read via Modbus/TCP when in slave attached mode. Diag LED, Status LED-G and Status LED-Y are the outputs for diagnostic LED (red), green status LED and the yellow status LED. After assigning the applicable function by pressing <enter> you are then asked if the pin is inverted (active low).

```
CP1 Function (hit space to toggle) GPIO (In)      Invert (active low) (Y) ?
```

A function should be assigned to each configurable pin. GPIO (Input) should be the default for all unused or unassigned pins.

```
CP1 Function (hit space to toggle) GPIO (In)      Invert (active low) (N) ?
CP2 Function (hit space to toggle) GPIO (In)      Invert (active low) (N) ?
CP3 Function (hit space to toggle) GPIO (In)      Invert (active low) (N) ?
```

After all the configurable pins have been assigned, the standard modem control settings can be entered if applicable.

```
RTS/CTS Mode (1=Fixed 2=Variable) (1) ?
```

The setting for each configurable pin is displayed in the setup menu.

Modbus/TCP to RTU Bridge Setup

1) Network/IP Settings:

```
IP Address ..... 192.168.0.2
Default Gateway ..... --- not set ---
```

```

Netmask ..... --- not set ---
2) Serial & Mode Settings:
  Protocol ..... Modbus/RTU,Slave(s) attached
  Serial Interface ..... 9600,8,N,1,RS232,CH1
3) Modem/Configurable Pin Settings:
  CP1.. GPIO (In)      CP2.. GPIO (In)      CP3.. GPIO (In)
  RTS Output ..... Fixed High/Active
4) Advanced Modbus Protocol settings:
  Slave Addr/Unit Id Source .. Modbus/TCP header
  Modbus Serial Broadcasts ... Disabled (Id=0 auto-mapped to 1)
  MB/TCP Exception Codes ..... Yes (return 00AH and 00BH)
  Char, Message Timeout ..... 00050msec, 05000msec

```

6.4.3 MatchPort/WiPort/Wi232-MB

The Modbus Master/Slave functionality on the MatchPort, WiPort and WiBox is very similar to the Modbus implementation on other platforms such as the NET232 or NET485. The major difference is that the configurable pins on the MatchPort/WiPort (CP0-CP10) and the WiFi settings on the MatchPort/WiPort/WiBox can be configured from the setup menu and web manager.

The WiPort/WiBox is a 2 serial port device and you can choose which serial port the firmware uses for sending and receiving Modbus/RTU or Modbus/ASCII serial data under the “**Serial & Mode Settings**” menu option. Channel 1 on WiBox only supports RS232 while channel 2 supports RS232 and RS422/RS485 2/4-Wire modes.

```
Use serial connector (1=CH1 2=CH2) (1) ?
```

The configurable pin options are as follows:

```

CP0 Function (hit space to toggle) GPIO (In)
CP0 Function (hit space to toggle) GPIO (Out)
CP0 Function (hit space to toggle) DTR (Out)
CP0 Function (hit space to toggle) Diag LED
CP0 Function (hit space to toggle) Status LED-G
CP0 Function (hit space to toggle) Status LED-Y
CP0 Function (hit space to toggle) RS485 Select
CP0 Function (hit space to toggle) RS485 2-Wire
CP0 Function (hit space to toggle) RS485 4-Wire
CP0 Function (hit space to toggle) Defaults(In)

```

The assignment for each configurable pin is set by cycling through the menu options by entering a space or any key other than <enter>. GPIO assigns the pin as a general purpose input or output. The GPIOs can be written and read via Modbus/TCP when in slave attached mode. DTR is the modem control output (MCO) signal for Data Terminal Ready. Diag LED, Status LED-G and Status LED-Y are the outputs for diagnostic LED (red), green status LED and the yellow status LED. RS485 Select is an output made active when configuring the serial channel for RS422/485 operation. RS485 2-Wire and 4-Wire are outputs made active when configuring RS422/485 2-Wire or 4-Wire operation respectively. Defaults is an input read at startup that tells the firmware to reset configuration to factory defaults. After assigning the applicable function by pressing <enter> you are then asked if the pin is inverted (active low).

```
CP0 Function (hit space to toggle) GPIO (In)      Invert (active low) (Y) ?
```

A function should be assigned to each configurable pin. GPIO (Input) should be the default for all unused or unassigned pins.

```

CP0 Function (hit space to toggle) RS485 Select  Invert (active low) (Y) ?
CP1 Function (hit space to toggle) RS485 2-Wire  Invert (active low) (Y) ?

```

CP2 Function (hit space to toggle)	GPIO (In)	Invert (active low) (N) ?
CP3 Function (hit space to toggle)	GPIO (In)	Invert (active low) (N) ?
CP4 Function (hit space to toggle)	GPIO (In)	Invert (active low) (N) ?
CP5 Function (hit space to toggle)	Diag LED	Invert (active low) (N) ?
CP6 Function (hit space to toggle)	Status LED-G	Invert (active low) (N) ?
CP7 Function (hit space to toggle)	Status LED-Y	Invert (active low) (N) ?
CP8 Function (hit space to toggle)	GPIO (In)	Invert (active low) (N) ?
CP9 Function (hit space to toggle)	GPIO (In)	Invert (active low) (N) ?
CP10 Function (hit space to toggle)	GPIO (Out)	Invert (active low) (N) ?

After all the configurable pins have been assigned, the standard modem control settings can be entered if applicable.

RTS/CTS Mode (1=Fixed 2=Variable) (1) ?

The setting for each configurable pin is displayed in the setup menu.

Modbus/TCP to RTU Bridge Setup

- 1) Network/IP Settings:
 - IP Address 192.168.0.2
 - Default Gateway --- not set ---
 - Netmask --- not set ---
- 2) Serial & Mode Settings:
 - Protocol Modbus/RTU,Slave(s) attached
 - Serial Interface 9600,8,N,1,RS232,CH1
- 3) Modem/Configurable Pin Settings:
 - CP0..!RS485 Select CP1..!RS485 2-Wire CP2.. GPIO (In)
 - CP3.. GPIO (In) CP4.. GPIO (In) CP5.. Diag LED
 - CP6.. Status LED-G CP7.. Status LED-Y CP8.. GPIO (In)
 - CP9.. GPIO (In) CP10.. GPIO (Out)
 - RTS Output Fixed High/Active
- 4) Advanced Modbus Protocol settings:
 - Slave Addr/Unit Id Source .. Modbus/TCP header
 - Modbus Serial Broadcasts ... Disabled (Id=0 auto-mapped to 1)
 - Local Slave Addr for GPIO .. 003 mapped to 0x/1x00100-00110
 - MB/TCP Exception Codes Yes (return 00AH and 00BH)
 - Char, Message Timeout 00050msec, 05000msec
- 6) WLAN Settings:
 - WLAN Enabled, FW Rev 0, network:LTRX_IBSS
 - Ad Hoc network creation Enabled, LTRX_IBSS, Country:US, Channel:11
 - Security None
 - Data rate Up to 11 Mbps
 - Power management Disabled

The menu option for “**WLAN Settings**” has been added to configure the WiFi parameters of the WiPort/WiBox.

6.5 Advanced Modbus Protocol Settings

Changing these parameters takes a bit of thought and planning.

Home

Network

Server

Modbus Bridge

Serial Settings

Modbus/TCP

Configurable Pins

Apply Settings

Apply Defaults

Serial Settings

Port Settings

Channel: 1Interface: RS232

Baud Rate: 9600Data Bits: 8Parity: NoneStop Bits: 1

Flow Control Out: Active AlwaysDelay before Transmit (1): msHold after Transmit(4): ms

Flow Control In:Wait til Active (2): NoDelay after Active (3): ms

Modem Control Out: Active Always

Modbus

Protocol: RTUASCII

Tx Delay after Rx: 0 ms

Character Timeout: 50 ms (0=auto)

Message Timeout: 5000 ms

OK

Home

Network

Server

Modbus Bridge

Serial Settings

Modbus/TCP

Configurable Pins

Apply Settings

Apply Defaults

Connection Settings

Connect Protocol

Protocol: Modbus/TCP Server attached to slave(s)

Advanced Server Settings

Modbus/TCP Port: 502Queue Multiple Modbus/TCP Requests: YesNo

Fixed Slave Address: 0 (0=MB/TCP header)Allow Modbus Broadcast: YesNo

Use Bridge Error Codes (0AH/0BH): YesNo

Swap 4x0x access to get 3x1x: YesNo

Swap Holding Reg (4x) access to Input Reg (3x) after offset: 0 (0 to disable)

Swap Coil Status (0x) access to Input Status (1x) after offset: 0 (0 to disable)

(Example: read of 401 023 maps to 300023 if you enter 1000)

Preset Automated Scan Table (optional)

No.	Unit Id (1-255)	Register Type	Offset	Count (1-124)	Frequency (ms)	
1						Remove

6.5.1 Modbus/TCP Port (standard default is 502)

The standard TCP port for the Modbus/TCP protocol is port 502. The default port number can be modified if needed for local routing or if required by the Modbus/TCP partner.

6.5.2 Slave Address (0 for auto, or 1..255 fixed otherwise)

Modbus/TCP includes a Unit ID field, which is used to address multiple Modbus slaves at a single IP address. Unfortunately, some first generation software drivers assumed a single slave at each IP and always set the Unit ID field to 0. This causes the Device Server problems because it requires the Unit ID for the Modbus/RTU “Slave Address”. To support these older applications, the Device Server allows you to force a fixed address for Modbus/RTU and Modbus/ASCII, but note that this restricts you to a single serial slave device per Device Server.

Setting this value to 0 causes the Device Server to use the Modbus/TCP Unit ID as received. Setting it to any other address causes the Device Server to always use the set value as a fixed address.

6.5.3 Allow Modbus Broadcasts (1=Yes 2=No)

This actually relates to the previous issue. The default is 2/No, in which case Device Server always assumes a Modbus/TCP “Unit ID” of 0 really means Modbus slave address 1. Answering No here is like setting a fixed address of 1 (parameter above), except the fixed address is only used if the Modbus/TCP “Unit ID” is 0.

Note: In the current software version for Device Server, a true Modbus broadcast is only supported when a serial slave device is attached. A Modbus broadcast from a serial master device is discarded regardless of this parameter setting.

6.5.4 Use MB/TCP 00BH/00AH Exception Responses (1=No 2=Yes)

Traditional serial Modbus uses silence to signal some errors. While this works well with direct serial lines, it causes serious problems on a TCP/IP wide-area-network where delays are not so predictable. See the Troubleshooting chapter for a full discussion.

Setting this to **1/No** causes the Modbus bridge to behave like a traditional Modbus serial slave – it answers timeouts, unconfigured slave addresses, and CRC errors with silence.

Setting this to **2/Yes** causes the Modbus bridge to return 1 of 2 new exception codes defined in Modbus/TCP.

Consider exception hex 0A (PATH UNAVAILABLE) a “hard” error where a retry is not likely to succeed. It is returned:

If slave-attached – currently never. However, future firmware may allow the user to define the range of valid slave addresses.

If master-attached – if a Modbus request has a slave address that is not configured in the Unit ID to IP mapping table.

If master-attached – if the TCP socket failed to open. This is really a soft-hard error, as the reason the TCP socket failed to open may be transient or a hard configuration error.

Exception hex 0B (TARGET DEVICE FAILED TO RESPOND) should be considered a “soft” error where a retry may succeed. It is returned:

If slave-attached – if the slave didn't answer or the answer contained a CRC error

If master-attached – if a TCP socket is open, but no response was received in the defined message timeout.

If master-attached – if a TCP socket is open, but the remote Modbus/TCP slave/server returned exception 0x0B.

Note: When using half-duplex RS485 in Master-attached mode with other slaves on the local RS485 line, it is necessary to set MB/TCP 00BH/00AH Exception Responses to 1=No.

6.5.5 Disable Modbus/TCP pipeline (1=No 2=Yes)

While the Modbus/TCP standard specification requires Modbus/TCP masters/clients to only issue 1 poll at a time, the full-duplex flow-controlled nature of TCP/IP allows them to issue more than one at a time, and the TCP socket will happily buffer them. The Modbus Bridge will fetch them one at a time and answer each in turn. See the Troubleshooting chapter for a full discussion of the problem this can cause.

Setting this to **1/No** causes the Modbus Bridge to allow this queuing or pipeline behavior. This is the safest default setting – only change this to **disable** if you are having problems.

Setting this to **2/Yes** causes the Modbus Bridge to always fetch the newest request from the TCP buffer – all older requests are discarded. This allows a Modbus/TCP master/client to issue new requests without risking building up a stale queue of waiting requests.

6.5.6 Character Timeout (0 for auto, or 10-6950 msec) (50)

This sets the timeout between characters received. Official Modbus/RTU defines a 3.5 character time-out, but complex devices have various interrupts that can cause 5 to 10 character “pauses” during transmission. A safe value for general use with Modbus is 50 msec. A setting of 0 will force the Device Server to automatically calculate a minimum timeout based on the baud rate.

Note: Setting this value lower than 50 msec may not improve performance and may even make performance worse. The Device Server uses an intelligent length-predicting algorithm to detect end-of-message in Modbus/RTU. Detecting the end of a Modbus/RTU message with character timeout is only used with user-defined or non-standard Modbus functions.

Note: When using half-duplex RS485 in Master-attached mode with other slaves on the local RS485 line, it is necessary to set Character Timeout to 0 for auto.

6.5.7 Message Timeout (200-65000 msec) (5000)

This sets the timeout for a response from a connected slave both serially and by TCP/IP.

6.5.8 Serial TX delay after RX (0-1275 msec) (0)

This feature inserts a delay between the Modbus/TCP master requests. The first request is sent out of the serial port of the Device Server to the Modbus slave. When the slave's response enters the serial port of the Device Server, it triggers this timer. After the specified delay is reached, the next master request is allowed to pass through the serial port of the Device Server, and the timer is reset. This feature is particularly useful when using RS485 2-wired serial protocol. The delay gives ample time for the RS485 slave devices to turn their transmitters off and their receivers back on. Normally this should be set to 0 – change it only if you are having problems. Raising this value can produce additional message delays.

6.5.9 Swap 4x/0x to get 3x/1x (N)

This setting will convert holding register (4x) data reads to input register (3x) data reads. It also converts coil (0x) reads to contact data (1x) reads. This feature is useful for Modicon I/O scanners that only support reads of holding registers and coils.

6.6 Preset Automated Scan Table

Selecting the Serial Protocol – Attached Device to be Modbus Slave will show Telnet configuration option **5** as Preset Automated Scan Table.

The Preset Automated Scan Table contains optional settings that direct the Device Server to automatically access Modbus registers on the slave(s). This is useful when multiple Modbus/TCP clients are polling the same group of “most interesting” registers. The scan allows the registers to be pre-fetched one time and then delivered without delay to multiple requesting clients. The auto scan operation also allows the TCP connection status and the network settings to be written to the connected slave’s registers.

Firmware Version: **V3.3.0.0GC**
MAC Address: **00-20-4A-A7-67-5E**

🏠

Network

Server

Modbus Bridge

Serial Settings

Modbus/TCP

Configurable Pins

Apply Settings

Apply Defaults

Connection Settings

Connect Protocol

Protocol: Modbus/TCP Server attached to slave(s)

Advanced Server Settings

Modbus/TCP Port: 502 Queue Multiple Modbus/TCP Requests: ☒ Yes ☐ No

Fixed Slave Address: 0 (0=MB/TCP header) Allow Modbus Broadcast: ☐ Yes ☒ No

Use Bridge Error Codes (0AH/0BH): ☒ Yes ☐ No

Swap 4x/0x access to get 3x/1x: ☐ Yes ☒ No

Swap Holding Reg (4x) access to Input Reg (3x) after offset: 0 (0 to disable)

Swap Coil Status (0x) access to Input Status (1x) after offset: 0 (0 to disable)

(Example: read of 401023 maps to 300023 if you enter 1000)

Preset Automated Scan Table (optional)

No.	Unit Id (1-255)	Register Type	Offset	Count (1-124)	Frequency (ms)	
0	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; height: 15px; width: 100%;"></div>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; padding: 2px;">Remove</div>
1	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; height: 15px; width: 100%;"></div>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; padding: 2px;">Remove</div>
2	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; height: 15px; width: 100%;"></div>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; padding: 2px;">Remove</div>
3	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; height: 15px; width: 100%;"></div>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; padding: 2px;">Remove</div>
4	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; height: 15px; width: 100%;"></div>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; padding: 2px;">Remove</div>
5	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; height: 15px; width: 100%;"></div>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; padding: 2px;">Remove</div>
6	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; height: 15px; width: 100%;"></div>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; padding: 2px;">Remove</div>
7	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; height: 15px; width: 100%;"></div>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; padding: 2px;">Remove</div>
8	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; height: 15px; width: 100%;"></div>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; padding: 2px;">Remove</div>
9	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; height: 15px; width: 100%;"></div>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<input style="width: 40px;" type="text"/>	<div style="border: 1px solid gray; padding: 2px;">Remove</div>

OK

Below is an example of adding an entry using Telnet. Select **5** to edit/view settings.

```

0):      001: Holding Reg  (4x)00020-00029/00200msec

A)dd, D)delete, E)xit - select function A
      Modbus addr  (1) ? 1
      Data type (hit space to toggle) Input Reg  (3x)
      Register Offset 55, Count 4, Frequency 250

0):      001: Holding Reg  (4x)00020-00029/00200msec
1):      001: Input Reg    (3x)00055-00058/00250msec

A)dd, D)delete, E)xit - select function

```

Figure 14 – Preset Automated Scan Table Example

6.6.1 A)dd, D)delete, E)xit Select Function

You can either add or delete entries in the scan table. Enter E when you are satisfied with the table to return to the main menu.

6.6.2 Modbus Address

This is the Modbus slave address of the connected slave.

6.6.3 Data Type

Hitting the space bar shows you the available data types for scanning. These include:

- Holding Register (4x) – reads a group of holding registers from the slave
- Input Register (3x) – reads a group of input registers from the slave
- Input Status (1x) – reads a group of input status registers (bits) from the slave
- Coil Status (0x) – reads a group of coil status registers (bits) from the slave
- TCP Status (0x) – writes the TCP client connected status to a coil
(0 = no connections, 1 = client(s) connected)
- IP Config (4x) – writes the network IP settings to holding registers on the first pass and then begins reading back to check for changes. If a change to the IP settings is detected then they are stored and the device server will reset to begin using the new settings. The IP Config options are as follows:
 1. IP address only (4 registers)
 2. IP address + Subnet Mask (8 registers)
 3. IP address + Subnet Mask + Gateway (12 registers)

6.6.4 Register Offset

The starting offset of the Modbus register(s) (Example: Input register offset 10 = address 3x00010).

6.6.5 Register Count

Count is the number of registers to be accessed (124 maximum).

6.6.6 Frequency

Frequency is the period in milliseconds between accesses to the registers (50 – 65000 ms).

6.7 Unit ID to IP Address Lookup Table

Selecting the Attached Device to be a Modbus Master will change Telnet configuration option 5 for updating the Unit ID to IP Address Table.

The screenshot shows a web interface for 'Connection Settings'. At the top, it displays 'Firmware Version: V3.3.0.0GC' and 'MAC Address: 00-20-4A-A7-67-5E'. On the left is a navigation menu with options: Network, Server, Modbus Bridge (with sub-options Serial Settings and Modbus/TCP), Configurable Pins, Apply Settings, and Apply Defaults. The main content area is titled 'Connection Settings' and includes a 'Connect Protocol' dropdown set to 'Modbus/TCP Client attached to master'. Below this is the 'Advanced Client Settings' section, which contains a 'Modbus/TCP Port' input field set to '502', a 'Use Bridge Error Codes (0AH/0BH)' section with 'Yes' selected, and two time settings: 'Close Idle TCP Sockets after: 10 secs (3-60 secs, 0=disable)' and 'Redundant Entry Retries after: 0 secs (15-60 secs, 0=disable)'. A red-bordered box highlights the 'Unit ID to IP Address Mapping' section, which contains a table with 8 rows. Each row has columns for 'No.', 'Start Id', 'End Id', 'Host Address', and a 'Remove' button. The table is currently empty. An 'OK' button is located at the bottom of the settings area.

Firmware Version: V3.3.0.0GC
MAC Address: 00-20-4A-A7-67-5E

Connection Settings

Connect Protocol
Protocol: Modbus/TCP Client attached to master

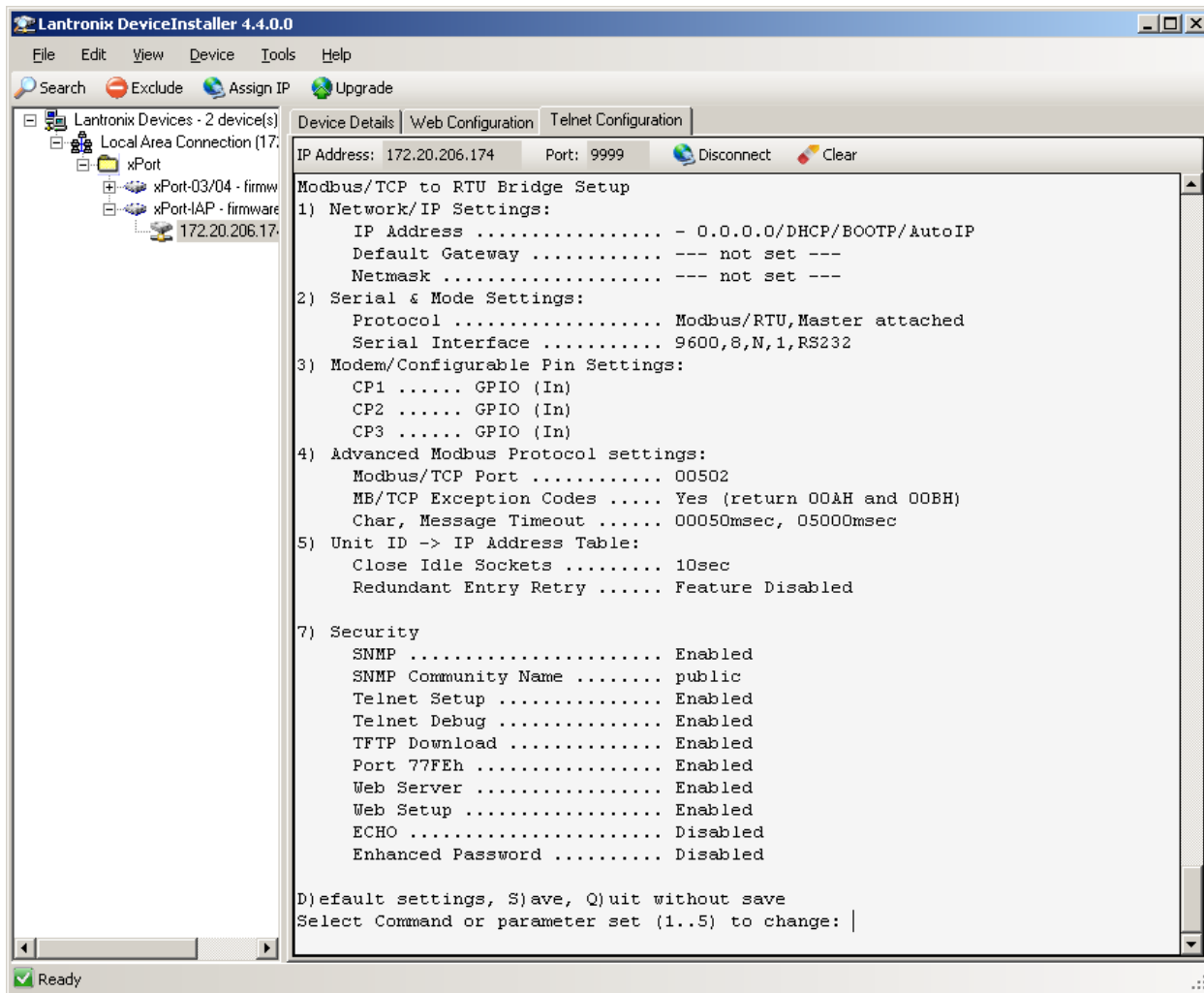
Advanced Client Settings
Modbus/TCP Port: 502
Use Bridge Error Codes (0AH/0BH): ☒ Yes ☐ No
Close Idle TCP Sockets after: 10 secs (3-60 secs, 0=disable)
Redundant Entry Retries after: 0 secs (15-60 secs, 0=disable)

Unit ID to IP Address Mapping

No.	Start Id	End Id	Host Address	
1				Remove
2				Remove
3				Remove
4				Remove
5				Remove
6				Remove
7				Remove
8				Remove

OK

The new Telnet menu appears like this:



Since serial Modbus uses 8-bit slave addresses and a TCP/IP network requires 32-bit IP addresses, the Device Server uses this table to map an 8-bit address into an IP/Unit ID combination. The 8-bit address is used to both select the desired IP and as the Unit ID sent. The table holds 8 entries, and any Modbus slave address not found in the table returns an exception response to the master (if enabled).

Below is an example of adding an entry. Select **5** to edit/view settings.

```
Close Idle TCP sockets after (3-60 sec, 0=leave open) (10)
Redundant entry retries after (15-60 sec. 0=disable feature) (0)
(Set 4th octet to 0 to use Slave Address as part of IP)

1):      001-100: 192.168.000.000+SLV
2):      101-199: 192.168.000.150

A)dd, D)delete, E)xit - select function A
Modbus addr from (102)
Modbus addr to (102) 255
Slave IP address (192) 172.(168) 16.(000) 123.(000)

1):      001-100: 192.168.000.000+SLV
2):      101-199: 192.168.000.050
3):      200-255: 172.016.123.000+SLV

A)dd, D)delete, E)xit - select function
```

Figure 15 - Unit ID to Address Lookup Table Example

6.7.1 Close Idle TCP sockets after (3-60 sec, 0=leave open)

Sockets are opened as required. Entering a 0 holds a single socket open to the last remote Modbus/TCP slave accessed.

Otherwise enter values 3 to 60 to automatically close the last socket after 3 to 60 seconds of idle time.

6.7.2 Redundant Entry Retries after (15-60 sec. 0=disable feature)

Enter the time in seconds for redundant entry retries or set to 0 to disable the feature.

6.7.3 A)dd, D)delete, E)xit Select Function

You can either add or delete entries in the IP address table. They are automatically sorted into increasing order. Enter E when you are satisfied with the table to return to the main menu.

6.7.4 Modbus Address From/To

This is the minimum/maximum Modbus slave address (inclusive) to forward to this IP address.

6.7.5 Slave IP Address

This is the IP address of the remote Modbus/TCP slave. Note the two different ways these IP are interpreted. In the configuration example above you'll see the following results:

- Polls to Slave #12 will go to IP 192.168.0.12 with Unit ID 12.
- Polls to Slave #70 will go to IP 192.168.0.70 with Unit ID 70.
- Polls to Slave #112 will go to IP 192.168.0.50 with Unit ID 112.
- Polls to Slave #155 will go to IP 192.168.0.50 with Unit ID 155.
- Polls to Slave #201 will go to IP 172.16.123.201 with Unit ID 201.
- Polls to Slave #244 will go to IP 172.16.123.244 with Unit ID 244.

Setting the last/4th IP octet to zero is interpreted as a signal to use the Slave ID as part of the IP. This allows a Modbus/RTU master to access up to 255 remote Modbus/TCP slaves. Setting the last/4th octet of the IP to 1-254 causes all slave polls in this group to be sent to the same IP. 255 is not accepted as the last/4th IP octet.

6.8 Security Settings

Select **7** in the Telnet Setup menu to change the security settings. Only the password setting can be adjusted from the Web-Manager. Most of the security options are for disabling ports and features that may be deemed a security risk.

6.8.1 Disable SNMP

This setting allows you to disable the SNMP feature on the Device Server. SNMP is a protocol used by Network Management Systems (NMS).

Press **Enter** to keep the current setting or else type **'y'** for yes and **'n'** for no.

6.8.2 SNMP Community Name

The SNMP community name is like a password that must be matched by a NMS in order to access SNMP MIB data.

6.8.3 Disable Telnet Setup

This setting disables Telnet access to the setup menu (port 9999). Device Server configuration can no longer be accessed via Telnet.

Note: If enabled, this setting can only be turned off using serial port access to the setup menu.

6.8.4 Disable Telnet Debug port

This setting disables Telnet access to debug output (port 3000).

6.8.5 Disable TFTP Firmware Update

This setting disables the ability to update Device Server firmware using TFTP (port 69).

6.8.6 Disable Port 77FEh

Port 77FEh is used for remote configuration of the Device Server. DeviceInstaller also accesses this port when searching for devices on the network.

Note: If enabled, the Device Server can no longer be discovered on the network by DeviceInstaller.

6.8.7 Disable Web Server

This setting disables the web server (port 80).

Note: If enabled, the Web-Manager can no longer be used for configuration.

6.8.8 Disable Web Setup

This setting disables access to the Device Server configuration via the web interface.

Note: If enabled, the Web-Manager can no longer be used for configuration.

6.8.9 Disable ECHO ports

This setting controls whether port 7 echoes characters it receives. This function is normally disabled.

6.8.10 Enable Enhanced Password

The enhanced password expands the default 4 character password to 16 characters. The password protects access to Device Server configuration via Telnet and the web. If the enhanced password is enabled, then you will be prompted if you want to change the password.

7. Monitor Mode and Firmware Upgrade

The easiest way to upgrade your protocol firmware (or “reflash”) is to use the DeviceInstaller Utility.

Note: This procedure should be done with caution.

There are important differences between the industrial protocol firmware (Modbus) files and standard firmware files. Although the NET232 (XPort) and NET232 MB (XPort MB) hardware may be the same, you will not be able to download a standard firmware file to the Modbus version and vice versa. These firmware files are rejected with the error “Sorry, that firmware not supported.” The industrial firmware also has blocked the “SF” command within the Monitor.

8. Troubleshooting

Fortunately, using the Modbus Bridge firmware is normally painless and easy to do.

Unfortunately in some situations it won't be easy or painless at all! In those situations you'll find it difficult to troubleshoot without an in-depth knowledge of Modbus and the system dynamics of polling.

Some general guidelines for trouble-shooting:

- Start polling slowly and work your way up.
- If you're using a custom cable, consider instead first **starting** with a simple, home-made adapter that makes your Device Server's RS-232 port look like a 9-pin DTE port such as on a computer. Then you can use a **known-good** cable to connect your device. Starting with a custom cable is 99 percent guaranteed to be a frustration – first prove everything is set up correctly, then use your custom cable as the final test.
- At a minimum you should have access to something to watch the serial line. Most host applications do a rather POOR job of explaining errors to you. We cannot count how many times we've had customers complain of "No Response," only to find out the device actually **did** respond. It was just the host application declaring "No response" instead of the true "response not understood."

8.1 Debug Port 3000

The Device Server dumps out Modbus traffic data on Telnet port 3000. You will need to use a Telnet tool to open a connection to the Device Server on port 3000.

The first thing to check is that there are "Net_Req[nn]..." debug logs. This indicates the Modbus/TCP master/client is connected and polling the Modbus slave. Next see if there are error conditions reported following the Net_Req such as "No Response!". A Net_Req should be followed by a Net_Rsp. Check the contents of the Net_Req and Net_Rsp data dumps for correct Modbus messaging.

8.1.1 Modbus Debug Codes

" Net_Req[#nn] xx ..." Dump of the received Modbus/TCP request with header (length nn). The Modbus/TCP header is the first 6 bytes followed by the Modbus message. The first byte of the Modbus message is the slave address and the second byte is the function code.

" Net_Rsp[#nn] xx ..." Dump of the Modbus/TCP response with header (length nn). The Modbus/TCP header is the first 6 bytes followed by the Modbus message. The first byte of the Modbus message is the slave address and the second byte is the function code (OR'ed with 80hex if there was an error).

"Mas_Req[#nn] xx ..." A dump of bytes for the master request (Master attached mode).

" no_slave" No matching slave in the Unit ID -> IP Address Table (Master attached mode).

" newsck" Closing TCP socket to reach new IP address (Master attached mode).

" new_opn:n.n.n.n" Open new TCP socket to IP address n.n.n.n (Master attached mode).

" open timeout "	TCP open timeout (Master attached mode).
" discard stale rsp seq_no:xx"	Modbus/TCP response received with the wrong sequence number (xx).
" master impatient"	A new master request available on the serial port before the Modbus/TCP response was received or timeout (Master attached mode).
" Discard 0x0B"	Modbus/TCP response had a gateway error status (0A/OB) and gateway errors were disabled in configuration (Master attached mode).
"Slv_Rsp[#nn] xx ..."	A dump of the Modbus/TCP response (Master attached mode).
" Pipe-N"	The current request is being discarded because more Modbus/TCP requests were queued on the socket and the pipeline option was disabled.
" rsp_delay_msec nnn"	Reports the response delay time in milliseconds (nnn).
" bad slv/cmd "	The slave address or function code of the response did not match the command.
"brcst"	The Modbus request was a broadcast. No response is expected.
" Pause nnn"	The firmware has injected a pause of 'nn' milliseconds. There is a minimum time that must be met of 3.5 character times between commands and responses. This minimum can be increased using the "Serial TX delay after RX" configuration option. Otherwise, it is based on the baud rate.
" CTS_TimeOut!"	A timeout occurred while waiting for CTS to go active (optional modem control).
" sTx[#nn] xx xx ..."	A dump of nn bytes of serial transmit data (debug firmware only).
" Ser_In ccc..."	A dump of received serial characters (Modbus/ASCII mode only).
" Ser_In rxx rxx..."	A dump of received serial bytes (Modbus/RTU, debug firmware only).
" No Response!"	Modbus message timeout with no bytes received on the serial port.
" sRx[#nn] xx xx ..."	A dump of received serial bytes (debug firmware only).
" CharTO"	Character Time Out reached during serial receive. Normally the firmware tries to predict the end of the Modbus message, but if the function is unknown, the CRC is bad or more incoming bytes are available then the character timeout is used.
" Ovr"	Overflow, too many bytes received for Modbus message (>255).
" LftO"	Unexpected byte received, the returned slave address or function code does not match the command (Modbus/RTU only).
"TCP Err err"	Modbus/TCP error reported (see Error Codes).
"CH Err err"	Modbus serial error reported (see Error Codes).

"err? nnn"	Error nnn encountered.
"(opn)"	New TCP socket opened.
"(cls)"	A TCP socket was closed.

8.1.2 Modbus Error Codes

01	Illegal Function
02	Illegal Data Address
03	Illegal Data Value
04	Illegal Response Length
05	ACKnowledge
06	Slave Device Busy
07	Negative ACKnowledge
08	Memory Parity Error (Prog)
10 (0Ah)	Gateway Path Unavailable
11 (0Bh)	Gateway Target Device Failed to Respond
65520	Bad CRC
65521	Timeout
65522	Bad Form – Invalid Modbus message
65524	Overflow – Modbus message too long (>255)
65525	Incomplete – Incomplete Modbus/TCP message received
65526	Out of Sequence – Wrong Modbus address or function received in response

8.2 Troubleshooting Software

Specializing in testing and diagnostic tools for Developers, WinTECH Software offers several products designed for the integration and troubleshooting of communications systems. All applications available from their site are fully functional time-limited demos, and may be freely downloaded and distributed for evaluation purposes. Developed by a Windows Developer for professional use, each application comes complete with an unconditional 30-day money-back guarantee. The web site is www.win-tech.com.

Modscan is a Windows application which operates as a modbus master. It allows you to access and change data points in a connected slave device using either the RTU or ASCII Transmission mode. ModScan is ideally suited for quick and easy compliance testing of the modbus protocol and its built-in display of serial traffic allows effective troubleshooting of field connections.

ModSim32 is a very simple but powerful application for simulating data from modbus slave devices. ModSim is an MDI application which allows you to define multiple blocks of data points which may then be accessed from a connected modbus master. Each document opened within ModSim may be configured to represent data from the same or different slave node thereby providing simulation for an entire group of devices. Useful for compliance testing of new master designs or as a quick simulation of a process.

Note: Grid Connect does not endorse WinTECH software. The information is provided to assist software developers. No support for this software will be provided by Grid Connect.

8.3 How fast can I poll?

First, remember that you still have the serial link in there and therefore cannot expect to poll any faster than you could by a direct serial link. In fact, since you are adding a number of queuing systems between your application and device, you may even lose a bit of performance. For example, some download tests showed remote download by Modbus/TCP bridged to Modbus/RTU ran about 20 percent slower than direct download by Modbus/RTU.

But above all remember that the serial speed (or baud rate) consumes the largest amount of time (see the table below). Suppose you issue a Modbus poll for 125 registers. This requires a 255-byte response, which at 19.2kbps requires over 133 msec just to physically shift across the wire, while at 300 baud it takes nearly 10 seconds!

Table 6 - Baud Rate

Baud Rate	Byte/Sec	Bit Time (msec)	Byte Time (msec)	256 Byte Time (msec)	(in sec)
300	30	3.333333	33.333333	8533.333333	8.53
600	60	1.666667	16.666667	4266.666667	4.27
1200	120	0.833333	8.333333	2133.333333	2.13
2400	240	0.416667	4.166667	1066.666667	1.07
4800	480	0.208333	2.083333	533.333333	0.53
9600	960	0.104167	1.041667	266.666667	0.27
19200	1920	0.052083	0.520833	133.333333	0.13
38400	3840	0.026042	0.260417	66.666667	0.07
57600	5760	0.017361	0.173611	44.444444	0.04
115200	11520	0.008681	0.086806	22.222222	0.02

The overall time it takes to poll is the combined sum of these delays:

1. Delay for Master /Client to recognize need for poll
2. Delay to issue and get the poll onto the Ethernet
3. Delay for the poll to cross Ethernet and arrive error-free at the Modbus Bridge device (may include retries and contention)
4. Delay for Modbus Bridge to process and queue Modbus/RTU poll
5. Delay for the serial link to be free (remember other Masters/Clients may be actively polling)
6. Physical delay to shift poll bit-by-bit across the serial link
7. Delay in the device to recognize, process, and start reply
8. Physical delay to shift response bit-by-bit across the serial link
9. Delay for Modbus Bridge to process and queue Modbus/TCP Response
10. Delay for the response to cross Ethernet and arrive error-free at the Master/Client (may include retries and contention)
11. Delay for Master /Client to recognize need for poll

Delays **a** and **k** are defined by your OPC or DDE driver. For example, a driver that runs only once each 55 msec (using the old DOS timer slice) can have a variable delay here of between 0 to 110 msec.

Delays **c** and **i** are defined by the complexity and load of your TCP/IP network. For example, if you're going thru radio or satellite links, these delays routinely amount to 1000 msec (1 sec) or more per poll and another 1000 msec for a response.

Delays **f** and **h** are defined by the baud rate. Assuming an 8 bytes poll and 255-byte response, at 9600 baud this is at least 275 msec, while at 1200 baud this is at least 2200 msec (2.2 sec).

Delay **g** is defined by the device. Oddly enough, the simpler the device, the faster it tends to reply. Some controllers only allocate fixed time slices to process a response from shared memory – for example once each 100 msec.

Delays **d**, **e**, and **i** are defined by the load on the Modbus Bridge. If other Master/Clients are polling, the queuing delay for **e** can be large (the sum of delays **f**, **g**, and **h**) for each earlier poll waiting.

8.4 I cannot get a slave response

Besides the obvious wrong baud rate, there are many possible causes of this:

- Is your cable set up correctly for RS-232?
- An external Signal Ground connection is often required between devices.
- The Modbus Bridge firmware only expects Modbus/TCP from the network. Some applications just pack Modbus/RTU raw in TCP – this is not supported.

8.5 Only Slave ID #1 can be polled

Your application is setting the Modbus/TCP Unit ID field to 0. This causes the Modbus Bridge firmware to automatically map this to 1.

8.6 Every 2nd poll seems to fail

Your device probably cannot accept a new poll as fast as the Modbus Bridge firmware is sending it. Remember, TCP/IP is a full-duplex channel, plus since you can have up to 8 active sockets it is very easy to have a new request already waiting as your last response is being returned. The only solution to this is to slow down your Modbus/TCP masters so they never poll before the last response has been seen. This manually creates the time delay between polls your device expects.

- My Bridge runs fine - for about 10 minutes and then my applications start reporting slaves going off-line.
- My Bridge runs fine – until a slave goes off-line; then I tend to lose all the slaves or they all poll only intermittently.
- Sometimes my Bridge returns the wrong data from the wrong slave.
- After a while, the Bridge seems to take longer and longer to answer – after a few hours, it takes 10 minutes or more for systems changes to propagate up to the Master/Client.

All these relate to the same issue – a mismatch in queuing behavior and expectation by the Master/Client to the new realities of Ethernet. No, it's not the Modbus Bridge behaving poorly. Yes, resetting the Bridge does "fix" the problem (flushes the bloated TCP queues full of stale requests).

The core problem is that the Master/Client is using the old RS-485 serial assumption that no-answer means poll was lost. However, in the Modbus Bridge case, it could also mean the Bridge has not had time to answer (is being over-worked). Also remember that TCP is reliable – the Bridge receives all polls sent without error. The result is that the Master/Client retries, which like throwing gasoline on a fire, makes it harder for the Bridge to catch up.

Here is the scenario that is hurting you:

1. Master sends out MB/TCP Poll #A with a timeout of 1000 msec.
2. Bridge receives the poll, but the serial link is busy so it waits - possibly another MB/TCP master is being serviced or time-outs waiting on off-line stations are creating a backlog of new requests.
3. After approximately 850 msec, the serial link is now free and the Bridge forwards the MB/RTU request.
4. The Bridge receives the response, and since the timeout on the Bridge and Master are not inherently synchronized, the Bridge sends the MB/TCP response into the TCP socket.
5. In the best of times, it may take 5-10msec for this response to actually go down the Bridge's TCP stack, across the wire, and up the master's TCP stack. If a WAN or satellite is involved, it could take 750 msec or longer.
6. Meanwhile, before the Master receives the Response #A, it gives up and makes the Modbus/RTU assumption that the request must have been lost. The Master sends out a new MB/TCP Poll #B.
7. A few msec later, there is a response that looks like a good Response #B, but really is Response #A. If the Master does not use a sequence number (unfortunately many do not) and has forgotten about pending poll #A, it wrongly assumes this is response #B (possibly with catastrophic results if Poll #B was the same size but different register range). **So here is the source of your “Bridge returns the wrong data for wrong slave” problem.**
8. The Master is idle and has no out-standing polls. Yet the Bridge has received Poll #B by reliable TCP/IP. It sends this out to Modbus/RTU slave and gets an answer. The Bridge is doing its job!
9. The Bridge returns Response #B to the master (if the socket is still open) and there it sits in its TCP/IP buffer. The Master is not expecting more responses, so it neither receives nor purges the "extra" response.
10. Master sends Poll #C and magically finds "a response" waiting as soon as it looks in the receive buffer - yet this is stale Response #B received before poll #C was even issued. If the Master does **not** implement Modbus/TCP sequence numbers, then it accepts the response #B as satisfying poll #C. Imagine if the Master is putting out 300 polls per minute (5 polls per second), but the Bridge can only process on average 290 of those per minute and some carry over. After 10 minutes, you may have up to 100 “stale” responses waiting in your Master’s TCP buffer. This makes it appear as though there is now a 20 second “lag” in data reaching the Master. **So here is the source of your “data taking longer and longer to propagate to Master/Client” problem.**

However, if the Master **does** implement Modbus/TCP sequence numbers, then the stale responses are rejected. If the Master is smart enough to resynchronize itself (response #B doesn't kill poll #C, but Master waits more), then this resynchronization will manifest itself as the slaves **going off-line and back on-line intermittently**. If the Master is not smart enough to resynchronize, once this out-of-sync behavior occurs, your **slaves go permanently off-line**.

As you can see, this Modbus/TCP master is out of sync and the only cure may be to either restart the Master or power cycle our Modbus Bridge. Both actions close the socket and purge the backlogged messages. Most unfortunately, it's the power cycle of our Bridge that is fastest. This always causes the light bulb to go on "Ah, this Bridge device is at fault!"

Our Network-to-Serial product brings out this shortcoming in Master/Client Modbus/TCP designs, but even a pure MB/TCP to MB/TCP network would suffer from this problem if the poll cycle approached the average response time. Any Modbus/TCP network going through WAN will discover this.

Ideally **all** Modbus/TCP Master applications must implement the sequence number and gracefully handle receipt of stale responses with unexpected sequence numbers. Unfortunately, the Modbus/TCP specification says that this sequence number is optional and can be used by a master to match responses to requests; however it can usually be just left as zero. The Modbus/TCP slave just echoes this back in the response. So most Modbus/TCP OPC servers today do not implement the sequence number.

Fortunately, a second generation of Modbus/TCP masters is starting to come that understands the issues of dealing with a Modbus Bridge to serial. So what is your solution if your Modbus/TCP master is first generation?

- Slow down your poll rate. You have to consider the worst-case response time – assume all polls timeout. If you have 5 slaves that normally answer in less than 100 msec each, but you must use a 250 msec message timeout, then polling each of the 5 each 1.25 sec is the only promised safe rate.
- If you are only polling a single slave (or poll one slave at a time), then you can try the “Disable Pipeline” option in the Modbus Bridge firmware. This will either help or make things hopelessly worse. If your OPC server or host application relies on pipelining to send more than one outstanding poll at once, then disabling the pipeline will essentially stop all data communication. (In which case, you can just turn the pipeline back on!)

The ideal solution (the 2nd generation solution) is for your Modbus/TCP Master/Client to not only support the Sequence Number, but also support the receipt of the 00AH and 00BH extended Modbus/TCP exception response. Then the Master/Client never needs to do retries – for each poll, it **will** receive either a value Modbus/TCP response or a Modbus/TCP exception that the slave is unreachable or timed out. This prevents the Master/Client from sending more polls than the Modbus Bridge can process and building the TCP buffer queue up in the first place.

9. Technical Support

If you are experiencing a problem that is not described in this manual, please contact Grid Connect at (630) 245-1445.

Our phone lines are open from 9:00AM - 4:30 PM Central Time Monday through Friday excluding holidays.