LANTRONIX®

Modbus Protocol
User Guide

Copyright and Trademark

© 2013 Lantronix, Inc. All rights reserved. No part of the contents of this book may be transmitted or reproduced in any form or by any means without the written permission of Lantronix.

Lantronix®, WiBox®, and WiPort® are registered trademarks of Lantronix, Inc. xPico™ and xDirect™ are trademarks of Lantronix, Inc.

Windows® and Internet Explorer® are registered trademarks of Microsoft Corporation Mozilla® and Firefox® are registered trademarks of the Mozilla Foundation. Chrome™ is a trademark of Google, Inc. All other trademarks and trade names are the property of their respective holders.

Contacts

Lantronix Corporate Headquarters
167 Technology
Irvine, CA 92618, USA

Toll Free: 800-526-8766
Phone: 949-453-3990
Fax: 949-450-3995

Technical Support
Online: www.lantronix.com/support

Sales Offices
For a current list of our domestic and international sales offices, go to the Lantronix web site at www.lantronix.com/about/contact

Disclaimer

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

Revision History

Date	Rev.	Comments
June 2001	B	Preliminary Release 6/01
September 2002	C	Reformat. Added notes, PN.
August 2004	D	Reformat. Added two advanced settings.
July 2005	E	Added content for WiPort and WiBox.
September 2005	F	Added content for XPort.
April 2012	G	Updated network protocol and platform information.
April 2013	H	Added content for xDirect and xPico. Updated security settings available on IAP.
June 2013	I	Added content for Micro125.

For the latest revision of this product document, please check our online documentation at www.lantronix.com/support/documentation.

If you use a previous version of the firmware, go to the Lantronix FTP site at ftp://ftp.lantronix.com/pub to find the earlier documentation.

Table of Contents

10:   Troubleshooting and Technical Support   29

Figures

Tables

1: Introduction

This protocol manual is for use with Lantronix Industrial Automation Protocol (IAP) Device Servers, such as the XPress DR-IAP, UDS1100-IAP, and xDirect-IAP. In addition to our IAP Device Servers, the Modbus protocol is supported on various embedded products including versions of the XPort, WiPort, xPico, and Micro/Micro125. The default protocol in new IAP Device Servers is the Standard Tunneling protocol, a serial protocol used to connect thousands of intelligent devices to the Ethernet. The user guide for your IAP Device Server provides detailed information for installing and operating the IAP Device Server using Standard Tunnel protocol. Changing that protocol to one of the industrial protocols changes the configuration menus and dialogs.

This User Guide provides Modbus™ protocol-specific information for the embedded and external products listed above.

2: Modbus

When it comes to planning data communication for open, multi-vendor industrial control systems, Modbus is the first choice of end users and integrators alike. The Modbus/RTU protocol defines how a "master" device polls one or more "slave" devices to read and write data in real time by means of RS232, RS422, or RS485 serial data communication. Although not the most powerful protocol available, its rare simplicity allows not only rapid implementation but also enough flexibility to apply in virtually all 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.telemecanique.com.

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

Extended Modbus System Example
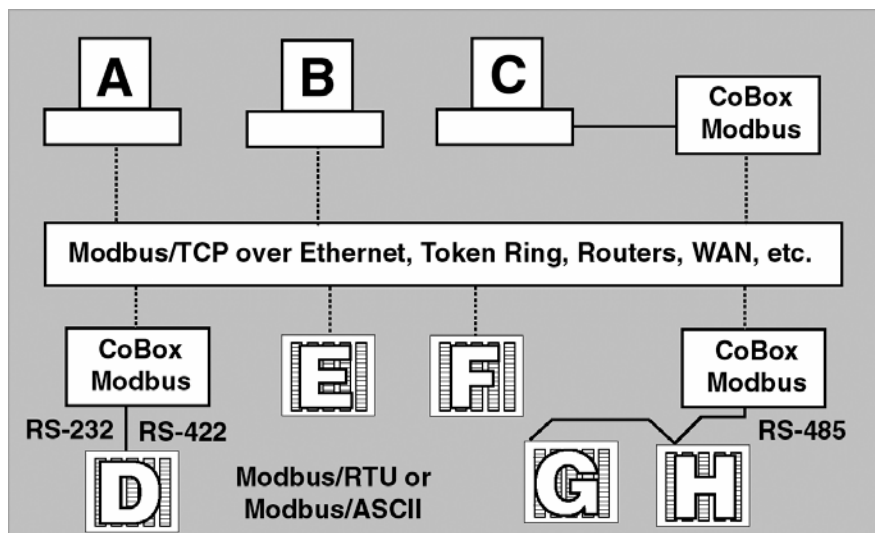
Figure 2-1. Extended Modbus System Example

Figure 2-1. Extended Modbus System Example shows four specific styles of Modbus operations. Modbus/RTU devices are traditionally split into two groups. (CoBox Modbus refers to an IAP Device Server.)

Modbus slave devices generally are the workhorse devices. They perform their tasks 24 hours a day, 365 days a year, for example, 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.

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 four 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 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 Lantronix IAP Device Servers extend to traditional Modbus/RTU serial devices.

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 RS232. This allows only a single Modbus/RTU master to talk to device D. However, the IAP 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 RS485 "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 IAP 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. IAP Device Server manages and coordinates the shared access. In fact, the IAP Device Server allows up to eight concurrent Modbus masters to share access to the slaves.

Modbus/RTU Serial Master Talking to Modbus/TCP Slave

Device C is a traditional Modbus/RTU master device. Yet the IAP 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.

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 RS485 line. The IAP 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: Configuring Modbus

Network Protocols

The IAP 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 IP 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 in which devices interact with others without maintaining a point-to-point connection, UDP datagram is used.

Packing Algorithm

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

IP Address

Every device connected to the TCP/IP network including the IAP 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. See the product user guide for your specific IAP Device Server for a complete description of IP Addressing.

When the IAP 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 IAP 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.

Configuration Methods

The IAP 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. See the product user guide for your IAP Device Server.

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

IAP Device Server's IP Address

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

With a DHCP-enabled IAP Device Server, if there is a DHCP server to respond to IAP Device Server's request when it is booting up, the IAP 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 IAP 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 IAP Device Server will display. . See *4: Monitor Mode and Firmware* for more information.

Using the Setup Mode Screen

1. From the DeviceInstaller configuration utility, click the **Telnet** button to open a Telnet connection to the IAP Device Server. The IAP Device Server's Ethernet hardware address (or HW MAC) displays.

```
Lantronix Inc. – Modbus Bridge
MAC address 00204A8EEA49
Software version V3.3.0.2RC1 (121211)

Press Enter to go into Setup Mode, wait to close
```

2. **Within 5 seconds**, press **Enter** to display the Setup (configuration) Mode screen. Here you can change the parameters that define how the IAP 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.

3. To reset the parameters to the factory defaults, type **D** on the command line and press **Enter**. The default parameters display.

4. Select an option on the menu (1-7) by typing the number of the option.

5. To enter a value for a parameter, type the value and press **Enter**, or to confirm a default value, press **Enter**.

6. Review your entries.

7. 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 3-1. Setup (Configuration) Mode Screen

```
Model: Device Server Plus+! (Firmware Code:AM)

Modbus/TCP to RTU Bridge Setup

1) Network/IP Settings:
    IP Address . . . . . . . . . - 0.0.0.0/DHCP/BOOTP/AutoIP
    Default Gateway  . . . . . . . --- not set ---
    Netmask  . . . . . . . . . . . --- not set ---
2) Serial & Mode Settings:
    Protocol . . . . . . . . . . . Modbus/RTU,Slave(s) attached
    Serial Interface . . . . . . . 9600,8,N,1,RS232
3) Modem Control Settings:
    RTS Output . . . . . . . . . . Fixed High/Active
4) Advanced Modbus Protocol settings:
    Slave Addr/Unit Id Soutce  . . Modbus/TCP header
    Modbus Serial Broadcasts . . . Disabled (Id=0 auto-mapped to 1)
    Modbus/TCP pipeline  . . . . . Enabled (new MB/TCP requests queued in FIFO)
    MB/TCP Exception Codes . . . . Yes (return 0x0A and 0x0B)
    Char, Message Timeout  . . . . 00050msec, 05000msec
7) Security Settings:
    SNMP . . . . . . . . . . . . . Enabled
    SNMP Community Name  . . . . . public
    Telnet Setup . . . . . . . . . Enabled
    TFTP Download . . . . . . . . . Enabled
    Port 77FEh . . . . . . . . . . Enabled
    Web Server . . . . . . . . . . Enabled
    Web Setup . . . . . . . . . . . Enabled
    Enhanced Password. . . . . . . Disabled
    Port 77F0h . . . . . . . . . . Enabled

D)efault settings, S)ave, Q)uit without save
Select Command or parameter set (1. . . 7) to change:
```

Basic Commands (D/S/Q)

Figure 3-1. Setup (Configuration) Mode Screen shows the main IAP Device Server configuration menu. The IAP Device Server offers three basic options.

Default Settings (D)

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

Save (S)

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

Quit Without Saving (Q)

Entering **Q** cancels any parameter changes you have made and exits configuration mode. This option triggers a reset.

Network/IP Settings

Select **1** to configure the Device Server's network parameters.

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

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 network administrator. Please refer to the IAP User Guide for your Device Server for more details about IP addresses.

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

Set Gateway IP Address (Y/N)

Most users could select **N** for this case. You only need to choose **Y** if the IAP 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.

Set Netmask (N for default)

Most users could select **N**, which causes the IAP Device Server to automatically use the standard netmask appropriate for the IP address 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.

Telnet Configuration Password

The Telnet configuration password can be set to disable unauthorized access to the setup menu via a Telnet connection to port 9999. To access the setup menu through the serial port, you do not need to enter the password.

Serial and Mode Settings

Select **2** to change the basic serial parameters. The following values can be set or changed.

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 controller or PLC) or **2** if the attached device is a master (such as a computer running graphical human-machine-interface (HMI) software).

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 IAP 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 IAP Device Server can remotely access a Modbus/ASCII slave attached to another IAP Device Server.

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

This allows the IAP Device Server to deal with the software-related details of using RS232, RS422, and RS485.

With the XPress DR-IAP, you still must set the external red RS232/RS485 switch appropriately. With the UDS-10-IAP or CoBox-FL-IAP, you must still select the correct pins for RS232 or the shared RS422/485 pins.

Enter Serial Parameters (B,D,P,S)

Enter the baud rate (300/ 600/ 1200/ 2400/ 4800/ 9600/ 19,200 or 38,400), 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. These settings must match the settings on the attached Modbus device.

These settings are valid with the XPress DR-IAP only. The UDS-10-IAP and CoBox-FL IAP also support baud rates 57,600 and 115,200. The UDS-10-IAP supports only 1 stop bit. If you try to set an unsupported combination of settings, a warning displays.

Note: Regardless of these settings, IAP Device Server configuration is always done on CH1 with RS232 and a setting of 9600,8,N, 1.

Modem Control Settings

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

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

Answer **1** and the IAP 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.

The XPress DR-IAP has a DTE-style RS232 port, so RTS is an output and CTS is an input. The UDS-10-IAP and CoBox-FL-IAP have a DCE-style RS232 port, so RTS is an input and CTS is an output.

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

Only asked if RTS/CTS mode is variable. After the IAP 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 cable is non-standard.

Wait for CTS to Go Active (N/Y)

Only asked if RTS/CTS mode is variable. Answering **N** causes the IAP Device Server to ignore the RTS/CTS response from the modem. Answering **Y** causes the IAP 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.

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 IAP Device Server sees the modem assert an RTS/CTS response input, it delays from 0 to 1275 msec before transmitting. If the IAP 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.

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

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

Advanced Modbus Protocol Settings

Changing these parameters takes a bit of thought and planning.

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 IAP Device Server problems because it requires the Unit ID for the Modbus/RTU "Slave Address." To support these older applications, the IAP 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 IAP Device Server.

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

Allow Modbus Broadcasts (1=Yes 2=No)

This relates to the previous issue. The default is **2/No**, in which case the IAP 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 IAP 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.

Use MB/TCP 0x0B/0x0A 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 *2: Modbus* for more information.

Setting this parameter to **1/No** causes the IAP Device Server 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 IAP Device Server 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.

Consider exception hex 0B (TARGET DEVICE FAILED TO RESPOND) 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.

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

While the Modbus/TCP standard specification requires Modbus/TCP masters/clients to only issue one 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 buffer them. The IAP Device Server will fetch them one at a time and answer each in turn. See *4: Monitor Mode and Firmware* for more information.

Setting this to **1/No** causes the IAP Device Server 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 IAP Device Server to always fetch the newest request from the TCP buffer – all older requests are discarded. This allows a Modbus/TCP master/client to retry old requests without risking building up a stale queue of waiting requests.

Character Timeout (10-7050 msec)

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.

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

Message Timeout (200-65000 msec)

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

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 IAP Device Server to the Modbus slave. When the slave's response enters the serial port of the IAP 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 IAP 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.

Swap 4x/0x to get 3x//x (1y)

This setting allows the MBF to convert "input" resistor data to "holding" resistor data. It also converts coil and contact data. This feature is useful for Modicon I/O scanners.

Unit ID to IP Address Lookup Table

Figure 3-2. Unit ID to IP Address Lookup Table

```
These parameters only apply when Modbus Master is selected for the Serial
Protocol. The new menu appears like this:
Network/IP Settings:
     IP Address . . . . . . . . . . 192.168.100.77
     Default Gateway  . . . . . . . --- not set ---
     Netmask  . . . . . . . . . . . --- not set ---
2) Serial & Mode Settings:
     Protocol . . . . . . . . . . . Modbus/RTU,Master(s) attached
     Serial Interface . . . . . . . 9600,8,N,1,RS232
3) Modem Control Settings:
     RTS Output . . . . . . . . . . Fixed High/Active
4) Advanced Modbus Protocol settings:
     MB/TCP Exception Codes . . . . Yes (return 0x0A and 0x0B)
     Char, Message Timeout  . . . . 00050msec, 05000msec
5) Unit ID -> IP Address Table
     Close Idle Sockets . . . . . . 10sec
     Redundant Entry Retry  . . . . Feature Disabled
7) Security Settings:
     SNMP . . . . . . . . . . . . . Enabled
     SNMP Community Name . . . . .. public
     Telnet Setup . . . . . . . . . Enabled
     TFTP Download . . . . . . . .. Enabled
     Port 77FEh . . . . . . . . . . Enabled
     Web Server . . . . . . . . . . Enabled
     Web Setup . . . . . . . . . .. Enabled
     Enhanced Password. . . . . . . Disabled
     Port 77F0h . . . . . . . . . . Enabled

D)efault settings, S)ave, Q)uit without save
Select Command or parameter set (1. . . 7) to change:
```

Since serial Modbus uses 8-bit slave addresses and a TCP/IP network requires 32-bit IP addresses, the IAP Device Server uses this table to map an 8-bit address into an IP/Unit ID combination. The 8-bit address is used to 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).

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

Figure 3-3. Unit ID to Address Lookup Table Example

```
Close Idle TCP sockets after (1-60 sec, 0=leave open) (00010)
Redundant entry retries after (15-60 sec. 0=disable feature) (00000)
(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)elete, 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)elete, E)xit - select function
```

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

Unlike earlier versions of the Lantronix Modbus firmware that tried to hold 8 sockets open forever, this version only holds sockets open 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.

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.

A)dd, D)elete, 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.

Modbus Address From/To

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

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 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.

Security Settings

Select **7** to configure the Device Server's security setting parameters.

```
7.Security Settings:
     SNMP ...................... Enabled
```

```
SNMP Community Name ........ public
Telnet Setup .............. Enabled
TFTP Download ............. Enabled
Port 77FEh ................ Enabled
Web Server ................ Enabled
Web Setup ................. Enabled
Enhanced Password ......... Disabled
Port 77F0h ................ Enabled
```

CAUTION: **Disabling both Telnet Setup and Port 77FE will prevent users from accessing the setup menu from the network.**

Below are the security features supported in Modbus IAP.

SNMP

We can enable or disable SNMP. By default it is enabled.

```
Disable SNMP (N) ? __
```

SNMP community

The SNMP Community Name is a required field for NMS to read or write to a device. Enter a string of 1 to 13 characters.

```
SNMP Community Name (public): __
```

The default entry is **public**. The current value is displayed in parentheses.

Telnet Setup

This setting defaults to the **N** (No) option. The **Y** (Yes) option disables access to Setup Mode by Telnet (port 9999). It only allows access locally using the web pages and the serial port of the unit.

```
Disable Telnet Setup (N) ? __
```

Note: If you choose to disable this option, keep in mind that disabling Telnet Setup will prevent users from accessing the setup menu from the network.

TFTP Download

This setting defaults to the **N** (No) option. The **Y** (Yes) option disables the use of TFTP to perform network firmware upgrades. With this option, you can download firmware upgrades over the serial port using DeviceInstaller's Recover Firmware procedure.

```
Disable TFTP Firmware Update (N) ? __
```

Port 77FEh

Port 77FE is a setting that allows DeviceInstaller custom program to locate and configure the unit remotely. You may wish to disable this capability for security purposes.

```
Disable Port 77FEh (N) ? __
```

Note: If you choose to disable this option, keep in mind that disabling both Telnet Setup and Port 77FE will prevent users from accessing the setup menu from the network.

Web Server

This setting defaults to the N (option). The Y (Yes) option disables the web server.

```
Disable Web Server (N) ? __
```

Enable Enhanced Password

This setting defaults to the N (No) option, which allows you to set a 4-character password that protects Setup Mode by means of Telnet and web pages. The Y (Yes) option allows you to set an extended security password of 16-characters for protecting Telnet and Web Page access.

```
Enable Enhanced Password (N) ? Y
```

Port 77F0 (Hex)

Port 77F0 is a setting that allows a custom application to query or set the eight xPico configurable pins when they are functioning as general purpose I/O (GPIO). You may want to disable this capability for security purposes. The default setting is the N (No) option, which enables GPIO control. The Y (Yes) option disables the GPIO control interface.

```
Disable Port 77F0h (N) ? __
```

Note: XPress-DR, xDirect, UDS1100 and Micro125 will not support this security feature.

Default Security Settings

All the security features are enabled by default except the Enhanced Password feature.

Table 3-1. Default Security Settings

Security Feature	Default Setting
Disable SNMP	No
SNMP community name	public
Disable Telnet setup	No
Disable TFTP Firmware Update	No
Disable Port 77FEh	No
Disable Web Server	No
Enable Enhanced Password	No
Disable Port 77F0h	No

4:   Monitor Mode and Firmware

UpgradeThe easiest way to upgrade your protocol firmware (or "reflash") is to use our DeviceInstaller utility. DeviceInstaller is a free utility program provided by Lantronix that discovers, configures, upgrades, and manages Lantronix Device Servers. It can be downloaded from the Lantronix website at www.lantronix.com/support/downloads. For instructions on using DeviceInstaller to configure the IP address and related settings or for more advanced features, see the *DeviceInstaller Online Help*.

The User Guide for the Device Server explains two other ways to reflash. You can do it serially with HyperTerminal or directly by Ethernet with a Trivial-File-Transfer-Protocol (or TFTP) Client. The TFTP method takes just a few seconds – much faster than the 5-plus minutes required serially. With all protocol firmware, use the TFTP destination filename of .ROM (in caps).

There are important differences between the industrial protocol firmware files and the standard Lantronix firmware files. Although the hardware may be the same, you will not be able to download a standard Lantronix firmware (such as the standard "3Q" or the older Modbus versions 1.0, 1.1, or 1.2). These firmware files are rejected with the error "Sorry, that firmware not supported." And although you can download any industrial firmware to non-IAP Lantronix Device Servers, they will not function. When started, an industrial firmware on the wrong hardware will blink the red LED, and you will see the error message "Sorry, this firmware is not supported" from the serial port. The industrial firmware also has blocked the "SF" command within the Monitor.

5:   WiPort and WiBox Implementation

The Modbus master/slave functionality on the WiPort and WiBox is similar to the Modbus implementation on other platforms such as the UDS1100 or XPress-DR. The difference is the configurable pins on the WiPort (CP0-CP10) and the WiFi settings on the WiPort and WiBox are configurable from the Setup Menu.

The WiPort and WiBox have 2 serial ports. Choose the serial port the firmware uses for sending and receiving Modbus/RTU or Modbus/ASCII serial data under the **Serial & Mode Settings** menu option.

Note: Channel 1 on WiBox only supports RS-232 while Channel 2 supports RS-232 and RS-422/RS-485 2/4-wire modes.

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

The menu option for **Modem Control Settings** is replaced with **Modem/Configurable Pin Settings** on the WiPort. The options are:

```
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:
      Network mode ............... Wired Only
      IP Address ................. - 0.0.0.0/DHCP/BOOTP/AutoIP
      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 .. fixed to 136
      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) WLAN Settings:
      WLAN ....................... Disabled, network:MSYS
      Topology ................... Infrastructure, Country: US
      Security ................... Authentication: PSK
                                   Encryption: TKIP
      TX Data rate ............... 54 Mbps auto fallback
      Power management ........... Disabled
      Soft AP Roaming ............ Disabled
      WLAN Max failed packets .... 6
7) Security Settings:
      SNMP ....................... Enabled
      SNMP Community Name ........ public
      Telnet Setup ............... Enabled
      TFTP Download .............. Enabled
      Port 77FEh ................. Enabled
      Web Server ................. Enabled
      Web Setup .................. Enabled
      Enhanced Password .......... Disabled
      Port 77F0h ................. Enabled
```

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

Two new parameters were added under the menu option **Advanced Modbus Protocol Settings** on WiPort. The Modbus slave address and starting offset parameters are used to direct Read Coil Status, Read Input Status, Force Single Coil and Force Multiple Coils Modbus commands to the WiPort's GPIO. Other commands or unmatched addressing are directed to the serial port.

```
Local slave address for GPIO (0 to disable, or 1..255) (0) ? 3
Starting offset (0x/1x0001..9999) (1) 100
```

6: XPort Implementation

The Modbus Master/Slave functionality on the XPort is similar to the Modbus implementation on other platforms (such as the UDS1100 or XPress-DR). A notable difference is the configurable pins on the XPort (CP1-3) are configurable from the setup menu. The menu option for **Modem Control Settings** has been replaced with **Modem/Configurable Pin Settings** on the XPort.

The options are as follows:

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

The **Status LED Output** function for CP1 is an active low output for controlling the device server's Status LED (LED1 in the *XPort Integration Guide*). Selecting **RTS Output** for CP1 prompts for additional options related to controlling a Request to Send (RTS) signal and performing flow control (see *Modem Control Settings on page 13*). Select **Wait for CTS** from these options to auto-configure CP3 for CTS Input. Use the **RS485 Output Enable** function to control an external RS485 line driver when in RS485 2-wire mode. This output is configurable for active high (default) or active low.

```
CP2 Function (1=Unused, 2=DTR Output, 3=RS485 Output Enable)
```

Select **DTR Output** for CP2 prompts for additional options for controlling a Data Terminal Ready (DTR) signal (see *Modem Control Settings on page 13*). **RS485 Output Enable** function controls an external RS485 line driver when in RS485 2-wire mode. This output is configurable for active high (default) or active low.

```
CP3 Function (1=Unused, 2=Diagnostic LED Output, 3=CTS Input)
```

The **Diagnostic LED Output** function for CP3 is an active low output for controlling the device server's Diagnostic LED (LED3 in the *XPort Integration Guide*). Select (**Y**)es on the **Wait for CTS** option under the CP1 function menu for **RTS Output** to automatically select the **CTS Input** function for CP3.

7:   *xPico Implementation*

The Modbus master/slave functionality of xPico is similar to the Modbus implementation on other platforms such as the UDS1100 or XPress-DR. The difference is the configurable pins on the xPico (CP1-CP8) .

The xPico has 2 serial ports. Choose the serial port, hence the firmware uses for sending and receiving Modbus/RTU or Modbus/ASCII serial data under the **Serial & Mode Settings** menu option.

Note: Channel 1 on xPico supports RS-232 and RS-422/RS-485 2/4-wire modes while Channel 2 only supports RS-232.

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

The menu option for **Modem Control Settings** has been replaced with **Modem/Configurable Pin Settings** on the xPico.

```
CP1 Function (hit space to toggle) GPIO (In)

CP2 Function (hit space to toggle) GPIO (Out)

CP3 Function (hit space to toggle) DCD (IN)  Channel-1

CP4 Function (hit space to toggle) DTR (Out) Channel-1

CP5 Function (hit space to toggle) Channel-1 Status LED

CP6 Function (hit space to toggle) Channel-2 Status LED

CP7 Function (hit space to toggle) RS485 Select

CP8 Function (hit space to toggle) RS485 2 Wire Select
```

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.
- ◆   DCD is the Modem Control input (MCI) signal for Data set Ready.
- ◆   Channel 1 and Channel 2 status LED show the activity of Modbus packet transactions happening in the device server.
- ◆   RS485 Select is an output made active when configuring the serial channel for RS422/485 operation.
- ◆   RTS is the out pin for channel 2 for hardware flow control.
- ◆   CTS is the input for channel 2 for hardware flow control.

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)
(Y) ?
CP2 Function (hit space to toggle) GPIO (Out)           Invert (active low)
(N) ?
CP3 Function (hit space to toggle) DCD (IN)  Channel-1   Invert (active low)
(N) ?
CP4 Function (hit space to toggle) DTR (Out) Channel-1   Invert (active low)
(Y) ?
CP5 Function (hit space to toggle) Channel-1 Status LED  Invert (active low)
(Y) ?
CP6 Function (hit space to toggle) Channel-2 Status LED  Invert (active low)
(Y) ?
CP7 Function (hit space to toggle) RS485 Select          Invert (active low)
(Y) ?
CP8 Function (hit space to toggle) RS485 2 Wire Select   Invert (active low)
(Y) ?
```

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

```
DTR Mode (1=Fixed 2=Variable) (1) ?
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 ................. - 0.0.0.0/DHCP/BOOTP/AutoIP
     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 (Out)
     CP3.. DCD (IN)  Channel-1  CP4..!DTR (Out) Channel-1
     CP5..!Channel-1 Status LED  CP6..!Channel-2 Status LED
     CP7..!RS485 Select         CP8..!RS485 2 Wire Select
     DTR Output ................. Fixed High/Active
     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
7) Security Settings:
     SNMP ....................... Enabled
     SNMP Community Name ........ public
     Telnet Setup ............... Enabled
     TFTP Download .............. Enabled
     Port 77FEh ................. Enabled
     Web Server ................. Enabled
     Enhanced Password .......... Disabled
     Port 77F0h ................. Enabled

D)efault settings, S)ave, Q)uit without save
Select Command or parameter set (1..7) to change:
```

8:   *Micro125 Implementation*

The Modbus master/slave functionality on the Micro125 is similar to the Modbus implementation on other platforms such as the xPico-IAP and xDirect-IAP. The Micro125 product does not support configurable pins, so they are not applicable in the Setup Menu.

The Micro125 has 1 serial port. In this serial port, hence the firmware uses for sending and receiving Modbus/RTU or Modbus/ASCII serial data under the **Serial & Mode Settings** menu option.

The setting for each configuration option is displayed in the setup menu.

```
Modbus/TCP to RTU Bridge Setup
1) Network/IP Settings:
     IP Address ................ - 0.0.0.0/DHCP/BOOTP/AutoIP
     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:
     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
7) Security Settings:
     SNMP ...................... Enabled
     SNMP Community Name ........ public
     Telnet Setup .............. Enabled
     TFTP Download ............. Enabled
     Port 77FEh ................ Enabled
     Web Server ................ Enabled
     Enhanced Password ......... Disabled

D)efault settings, S)ave, Q)uit without save
Select Command or parameter set (1..7) to change:
```

9:   Factory Defaults with Reset Pin/Default Pin

xPico Factory Defaults

To restore the factory default configuration of the device server, press the reset pin and then press the default pin for 3 to 5 seconds. Device server will load the factory default configuration and reboot again.

xDirect Factory Defaults

To restore the factory default configuration, press and hold the reset pin for more than 10 seconds. After 10 seconds release the reset pin. The device server will load the factory default configuration.

If the user presses the reset button for 0-2 seconds then the unit simply reboots.

Note: With factory default configuration, IP address, Gateway and Netmask are set to 0.0.0.0.

Micro125 Factory Defaults

To restore the factory default configuration of the device server, press the reset pin for 3 to 5 seconds. Device server will load the factory default configuration and reboot again.

10: Troubleshooting and Technical Support

Using the IAP Device Server firmware is normally easy. However, if a problem occurs, it is difficult to troubleshoot without an in-depth knowledge of Modbus and the system dynamics of polling.

Some general guidelines for troubleshooting:

◆ Start polling slowly and increase speed gradually.

◆ Cabling is the most common problem with device networking. If you have created a custom cable, make sure your pinout is correct.

◆ Ideally, you should have the ability to watch the serial line communications. Most host applications do a poor job of explaining errors. In many situations, the host application declares "No response" when in fact, the device did respond, and the application did not understand the response.

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.

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 9-1. 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:

a. Delay for master/client to recognize need for poll.

b. Delay to issue and get the poll onto the Ethernet.

c. Delay for the poll to cross Ethernet and arrive error-free at the IAP Device Server device (may include retries and contention).

d. Delay for IAP Device Server to process and queue Modbus/RTU poll.

e. Delay for the serial link to be free (remember other master/clients may be actively polling).

f. Physical delay to shift poll bit-by-bit across the serial link.

g. Delay in the device to recognize, process, and start reply.

h. Physical delay to shift response bit-by-bit across the serial link.

i. Delay for IAP Device Server to process and queue Modbus/TCP response.

j. Delay for the response to cross Ethernet and arrive error-free at the master/client (may include retries and contention).

k. 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 are going through 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 IAP Device Server. If other master/client are polling, the queuing delay for **e** can be large (the sum of delays **f**, **g,** and **h)** for each earlier poll waiting.

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 RS232 or RS485? On the XPress DR-IAP, is the external red switch set correctly?

◆ For RS485, you need to short the TX+ to the RX+ and TX- to the RX- externally.

◆ The XPress DR-IAP has a floating ground that is fully isolated from the power supply. An external Signal Ground connection is often required between the IAP and your device.

◆ The IAP Device Server firmware only expects Modbus/TCP from the network. Some applications just pack Modbus/RTU raw in TCP – this is not supported.

◆ Your slave is set for 2 stop bits and your UDS-10-IAP does not support 2 stop bits.

Only Slave ID #1 can be polled

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

Every 2nd poll seems to fail

Most likely you are using RS485, but regardless, your device probably cannot accept a new poll as fast as the IAP Device Server firmware is sending it. TCP/IP is a full-duplex channel, and 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 IAP Device Server runs fine - for about 10 minutes and then my applications start reporting slaves going off-line.

- ◆ My IAP Device Server runs fine – until a slave goes off-line; then I tend to lose all the slaves or they all poll only intermittently.

- ◆ Sometimes my IAP Device Server returns the wrong data from the wrong slave.

- ◆ After a while, the IAP Device Server 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. (It is not the IAP Device Server behaving poorly.) Resetting the IAP Device Server fixes the problem (flushes the bloated TCP queues full of stale requests).

The core problem is that the master/client is using the old RS485 serial assumption that no answer means poll was lost. However, in the IAP Device Server case, it could also mean the IAP Device Server has not had time to answer (is being overworked). Also remember that TCP is reliable – the IAP Device Server receives all polls sent without error. The result is that the master/client retries, which makes it harder for the IAP Device Server to catch up.

Here is the scenario that is causing the problem:

1. Master sends out MB/TCP Poll #A with a timeout of 1000 msec.

2. IAP Device Server receives the poll, but the serial link is busy so it waits - possibly another MB/TCP master is being serviced or timeouts 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 IAP Device Server forwards the MB/RTU request.

4. The IAP Device Server receives the response, and since the timeout on the IAP Device Server and master are not inherently synchronized, the IAP Device Server sends the MB/TCP response into the TCP socket.

5. In the best of times, it may take 5-10 msec for this response to actually go down the IAP Device Server'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 (which 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). Here is the source of the problem **"IAP Device Server returns the wrong data for wrong slave."**

8. The master is idle and has no outstanding polls. Yet the IAP Device Server has received Poll #B by TCP/IP. It sends this out to Modbus/RTU slave and gets an answer. The IAP Device Server is doing its job!

9. The IAP Device Server 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 IAP Device Server 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. **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 does not 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 the IAP Device Server. Both actions close the socket and purge the backlogged messages.

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 an IAP Device Server 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 five slaves that normally answer in less than 100 msec each, but you must use a 250-msec message timeout, then polling each of the five 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 IAP Device Server 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 0x0A and 0x0B 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 IAP Device Server can process and building the TCP buffer queue up in the first place.

Technical Support

If you are experiencing an error that is not described in this chapter, or if you are unable to fix the error, you have the following options:

Technical Support US

Check our online knowledge base or send a question to Technical Support at http://www.lantronix.com/support.

Phone: (800) 422-7044 (US Only)
        (949) 453-7198

Technical Support Europe, Middle East, and Africa

Phone: +33 (0)1 39 30 41 72
        +49 (0) 180 500 13 53 (Germany Only)

Email: eu_techsupp@lantronix.com or eu_support@lantronix.com

Firmware downloads, FAQs, and the most up-to-date documentation are available at www.lantronix.com/support.

When you report a problem, please provide the following information:

◆ Your name, and your company name, address, and phone number

◆ Lantronix model number and serial number

◆ Software version (on the first screen shown when you Telnet to port 9999)

◆ Description of the problem

◆ Debug report (stack dump), if applicable

◆ Status of the unit when the problem occurred (please try to include information on user and network activity at the time of the problem)