

# **Wonderware® Modicon MODBUS I/O Server**

## **User's Guide**

Revision N  
June 2001

**Wonderware Corporation**

All rights reserved. No part of this documentation shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the Wonderware Corporation. No copyright or patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this documentation, the publisher and author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

The information in this documentation is subject to change without notice and does not represent a commitment on the part of Wonderware Corporation. The software described in this documentation is furnished under a license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of these agreements.

Wonderware Modicon MODBUS I/O Server User's Guide ©

**2000 Wonderware Corporation. All Rights Reserved.**

100 Technology Drive  
Irvine, CA 92618  
U.S.A.  
(949) 727-3200  
<http://www.wonderware.com>

**Trademarks**

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Wonderware Corporation cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Wonderware and InTouch are registered trademarks of Wonderware Corporation.

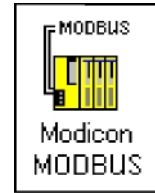
Wonderware FactorySuite, WindowMaker, WindowViewer, SQL Access Manager, Recipe Manager, SPC Pro, DBDump, DBLoad, HDMerge, HistData, Wonderware Logger, InControl, InTrack, InBatch, IndustrialSQL, FactoryOffice, Scout, SuiteLink and NetDDE are trademarks of Wonderware Corporation.

# Contents

Introduction .....	1
Communication Protocols .....	1
Accessing Remote Items via the I/O Server .....	2
Configuring the I/O Server .....	3
Configuring a Communication Port .....	4
Communication Port Settings .....	4
Saving the I/O Server's Configuration File.....	6
Save Configuration .....	6
Saving Multiple Configuration Files.....	6
Configuring a Topic Definition .....	7
Topic Definition .....	7
MODBUS Topic Definition .....	8
Configuring the I/O Server Settings .....	11
Server Settings.....	11
Accessing I/O Server Help .....	13
Contents .....	13
How to Use Help .....	13
About MODBUS .....	13
Item Names .....	14
Special Item/Point Naming Conventions .....	16
Monitoring the Status of Communications with a PLC .....	20
Using the Status Item in Excel .....	20
Monitoring the Status of Communications with InTouch .....	21
Using DDEStatus and IOStatus in Excel .....	21
Reading Values from the I/O Server into Excel .....	22
Writing Values to the I/O Server from Excel .....	23
Troubleshooting I/O Server Communication Problems .....	24
Debugging Communication Between InTouch and an I/O Server .....	24
Debugging Communication Between SuiteLink and an I/O Server .....	26
Debugging Communication Between an I/O Server and a PLC.....	27



# Wonderware Modicon MODBUS I/O Server



## Introduction

The Wonderware Modicon MODBUS I/O Server (referred to as the server through the remainder of this user's guide) is a Microsoft® Windows® application that acts as a communication protocol server. It allows other Windows application programs access to data from Modicon PLCs (also referred to as devices) or other supported Flow Computers (FC). Supported Flow Computers include the Daniel FC, Elliott FC, Omni FC, and Micromotion. (Any reference using the term PLC in this manual is applicable to the aforementioned FCs as well). The Wonderware Modicon MODBUS I/O Server communicates with PLCs via a serial RS-232 or RS-422 connection. Using modems or multi-drop transceivers, the server can support up to 247 PLCs on each serial port.

While the server is primarily intended for use with Wonderware's InTouch (version 3.01 or later), it may be used by any Microsoft Windows program capable of acting as a DDE, FastDDE, or SuiteLink 7 client.

## Communication Protocols

Dynamic Data Exchange (DDE) is a communication protocol developed by Microsoft to allow applications in the Windows environment to send/receive data and instructions to/from each other. It implements a client-server relationship between two concurrently running applications. The server application provides the data and accepts requests from any other application interested in its data. Requesting applications are called clients. Some applications such as InTouch and Microsoft Excel can simultaneously be both a client and a server.

FastDDE provides a means of packing many proprietary Wonderware DDE messages into a single Microsoft DDE message. This packing improves efficiency and performance by reducing the total number of DDE transactions required between a client and a server. Although Wonderware's FastDDE has extended the usefulness of DDE for our industry, this extension is being pushed to its performance constraints in distributed environments.

NetDDE 7 extends the standard Windows DDE functionality to include communication over local area networks and through serial ports. Network extensions are available to allow DDE links between applications running on different computers connected via networks or modems. For example, NetDDE supports DDE between applications running on IBM® compatible computers connected via LAN or modem and DDE-aware applications running on non-PC based platforms under operating environments such as VMS® and UNIX.

SuiteLink uses a TCP/IP based protocol and is designed specifically to meet industrial needs such as data integrity, high-throughput, and easier diagnostics. This

protocol standard is supported on Microsoft Windows NT 4.0 (or higher) and Windows 2000.

SuiteLink is not a replacement for DDE, FastDDE, or NetDDE. The protocol used between a client and a server depends on your network connections and configurations. SuiteLink is designed to be the industrial data network distribution standard and provides the following features:

Value Time Quality (VTQ) places a time stamp and quality indicator on all data values delivered to VTQ-aware clients.

Extensive diagnostics of the data throughput, server loading, computer resource consumption, and network transport are made accessible through the Microsoft Windows NT and Windows 2000 operating system Performance Monitor. This feature is critical for the scheme and maintenance of distributed industrial networks.

Consistent high data volumes can be maintained between applications regardless if the applications are on a single node or distributed over a large node count.

The network transport protocol is TCP/IP using Microsoft's standard WinSock interface.

## Accessing Remote Items via the I/O Server

The communication protocol addresses an element of data in a conversation that uses a three-part naming convention that includes the application name, topic name and item name. The following briefly describes each portion of this naming convention:

*application name* The name of the Windows program (server) that will be accessing the data element. In the case of data coming from or going to Modicon equipment via this server, the application portion of the address is **MODBUS**.

*topic name* Meaningful names are configured in the server to identify specific devices. These names are then used as the topic name in all conversations to that device. For example, **MODSLAVE5**.

---

**Note** You can define multiple topic names for the same device (PLC) to poll different points at different rates.

*item name* A specific data element within the specified topic. For example, when using this server, an item can be a relay, timer, counter, register, etc., in the PLC.

---

**Note** The item/point names are predefined by the server. The term "point" is used interchangeably with the term "item" in this user's guide.

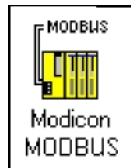
---

! For more information on item/point names, see the "Item Names" section later in this user's guide.

## Configuring the I/O Server

Once the server has been installed, a small amount of configuration is required. Configuring the server automatically creates a configuration file named, **MODBUSDV.CFG**. This file stores the configuration information for the adapter cards or communication ports and all of the topic definitions (described in detail later).

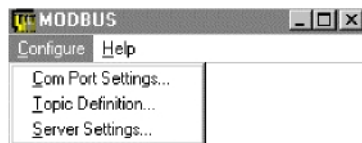
The configuration file is automatically saved to the directory in which the server is installed unless a different directory is specified.



To perform the required configurations, start up the server by double-clicking on its icon. If the server starts up as an icon, double-click on the icon to open the server's window.

To access the options used for the various configurations, open the

**Configure** menu:



---

**Note** If any of the options appear grayed, then these options are not available with this software version.

## Configuring a Communication Port

Use the **Communication Port Settings** option from the Configure Menu to configure the communication ports that will be used to communication with Modicon device. When this option is selected, this dialog box will appear:

## Communication Port Settings

**Note** All communication ports can be configured without leaving this dialog box. Repeat these steps; select a COM Port, set configuration, and click **Save**.

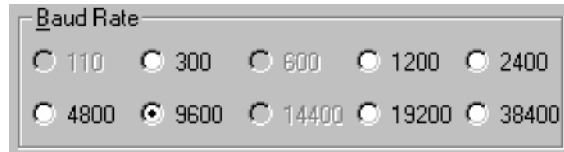
Select a communication port connected to the Modicon device.

Enter the amount of time (in seconds) that all PLCs connected via this communication port will be given to reply to commands from the server.

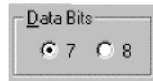
**Note** This timeout is sustained only when the PLC fails to respond. When the PLC is responding normally, there is no penalty. The default value of 3 second should be sufficient for most configurations.

Select the protocol configured for the equipment attached to this communications port. In cases where you have a choice, **RTU** is recommended.





Select the Baud Rate (serial bit rate) setting that matches the configuration of the Modicon device.



Select the number of Data Bits that matches the configuration of the Modicon device.



Select the number of Stop Bits that matches the configuration of the Modicon device. If the Baud Rate is greater than 300, the Stop Bits should be set to **1**.



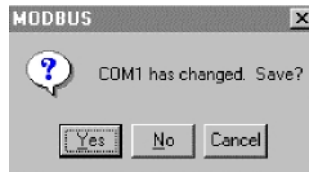
Select the Parity setting that matches the configuration of the Modicon device.

All devices on a single communication port must be configured with the same Baud Rate, Data Bits, Stop Bits, and Parity.

Click **Defaults** to reset the settings to their default values without saving changes.

Click **Save** to save settings for the selected COM Port. The dialog box will remain displayed giving you the option to configure additional COM Ports.

Click **Done** to close the dialog box. If the settings have not been saved, the following dialog box will appear:



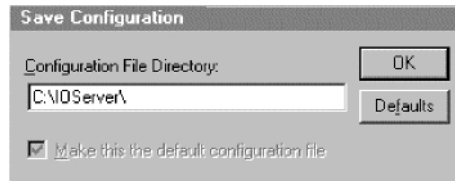
Click **Yes** to save settings for the COM Port.

Click **No** to prevent saving the settings.

Click **Cancel** to return to the **Communication Port Settings** dialog box without saving the settings.

## Saving the I/O Server's Configuration File

If a configuration file does not currently exist in the configuration file directory, the server will automatically display the **Save Configuration** dialog box:



### Save Configuration

This field displays the drive\directory into which the server will save the current configuration file. To save the configuration file to a different directory, enter the path for that directory in this field.



This option is selected and disabled on initial entry to the **Save Configuration** dialog box. This field becomes active if the Configuration File Directory is changed. Once enabled, selecting this option will record a new Configuration File path in the **WIN.INI** file. This option allows the server to find its configuration file automatically each time it is started.

---

**Note** When the server initially starts up, it attempts to locate its default configuration file by first checking the **WIN.INI** file for a previously specified path. If a path is not present in the **WIN.INI** file, the server will assume that the current working directory is to be used.

---

Click **Defaults** to reset the settings to their default values without saving changes.

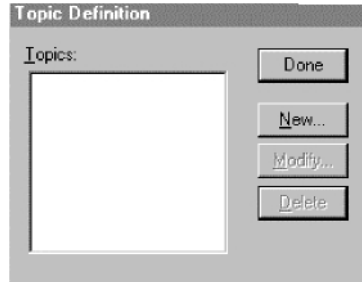
Click **OK** to save the configuration file to the specified directory. **Saving Multiple Configuration Files**

There is no limit to the number of configuration files that you can create as long as they are saved in separate directories. This allows you to have multiple configuration files that can be accessed by using a special switch (/d:). For example, to start the server using a configuration file located in a different directory, click the **Start** button on the taskbar, then choose **Run** and enter the following in the **Open** combo box:

**MODBUS /d:c:\directoryname**

# Configuring a Topic Definition

Use the **Topic Definition** option from the Configure menu to create new, modify, or delete topic definitions. One or more topic definitions must exist for each PLC that the server will communicate with. Each topic definition must contain a unique name for the PLC associated with it. When this option is selected, the **Topic Definition** dialog box will appear:



## Topic Definition

---

**Note** Once topics have been defined, their names will be listed in the **Topics** section of this dialog box.

---



Click this button to close the dialog box and accept any new definitions, modifications or deletions made.



To modify or view an existing topic definition, select its name in the list and click on this button. The **MODBUS Topic Definition** dialog box (described below) will appear displaying the selected topic definition.



To delete an existing topic definition, select its name in the list and click on this button. (A message box will appear prompting you to confirm the deletion.)



To add a new topic definition, click on this button. The **MODBUS Topic Definition** dialog box will appear:

## MODBUS Topic Definition

Topic Name:

Enter a unique name (up to 32-characters long) for the PLC in this field, for example **ModSlave5**.

---

**Note** When communicating with InTouch, this **exact** name is used as the topic name in the Access Name definition.

Com Port:

The Com Port currently associated with this topic will appear in this field. To select a different port, click on the down arrow to open the list of communication ports. Click on the name of the communications port to be associated with this topic.

Slave ID:

Enter the Slave ID of the PLC in this field.

Slave Device Type:

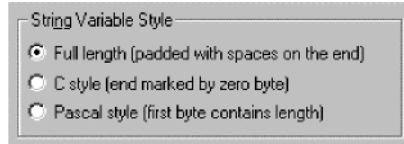
The Slave Device Type currently associated with this topic will appear in this field. To select a different type, click on the down arrow to open the list of Slave Device Types. Click on the name of the type to be associated with this topic.

---

**Note** For more information on the Slave Device Types, see the "Item Names" section later in this user's guide.

Use Concept Data Structures

Select this option to configure the server to be compatible with the Aquatrol Concept software data structures.



String Variable Style

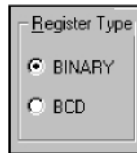
- Full length (padded with spaces on the end)
- C style (end marked by zero byte)
- Pascal style (first byte contains length)

Select the option for the style the PLC uses to store ASCII strings in its registers.

---

**Note** For more information on the String Variable Styles, see the "Item Names" section later in this user's guide.

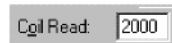
---



Register Type

- BINARY
- BCD

Select the option for the Register Type being used.



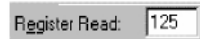
Coil Read:

Enter the maximum number of consecutive coils to be read at one time in this field. In this example, the valid Coil Read values can be between 8 and 2000 and must be an even multiple of 8.



Coil Write:

This field is used to enter the maximum number of consecutive coils that can be written to at one time. In this example, the valid Coil Write values can be between 8 and 800 and must be an even multiple of 8.



Register Read:

Enter the maximum number of consecutive registers to be read at one time in this field. In this example, the valid Register Read values can be between 1 and 125.



Register Write:

Enter the maximum number of consecutive registers that can be written to at one time in this field. In this example, the valid Register Write values can be between 1 and 100.



Update Interval:  msec

Enter the frequency (in milliseconds) that the server will read (poll) the items/points associated with this topic.

---

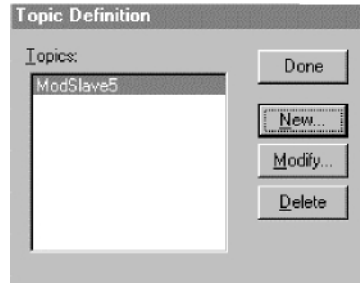
**Note** Different items/points can be polled at different rates by defining multiple topic names for the same PLC and setting different updates rates for each topic.

**Note** If this is the first configuration performed for the server, the **Save Configuration** dialog box will appear prompting you to save the configuration file.

---

Click **Cancel** to close the dialog box without saving changes.

Click **OK** to accept the topic definition and return to the **Topic Definition** dialog box:



Click **Done** to close this dialog.

## Configuring the I/O Server Settings

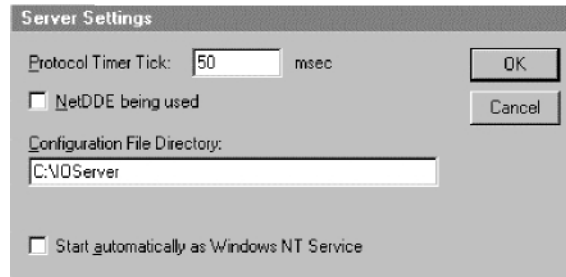
Use the **Server Settings** option from the Configure menu to change the protocol timer, network using Wonderware NetDDE, change the default configuration file path, or to enable the server to start automatically as a Windows NT and Windows 2000 operating system service.

---

**Note** When configuring the server on Windows NT and Windows 2000, the user must be logged on with system administrator privileges. This will ensure that updates to the system registry may be performed.

---

When the **Server Settings** option is selected, the **Server Settings** dialog box will appear:



### Server Settings

Protocol Timer Tick:  msec

Enter the frequency (in milliseconds) that the server is to check for data to process. This should be approximately two to four times faster than the fastest rate desired to update data from the equipment.

---

**Note** The default protocol timer tick value will vary between servers.

---

NetDDE being used

Select this option if you are networking using Wonderware NetDDE.



To create a new default configuration file, enter the complete path for the directory in which the file is to be saved in this field. This new path will automatically be written to the **WIN.INI** file and the server will use this path to load its configuration file the next time it is started.

---

**Note** There is no limit to the number of configuration files created. However, each must be saved in a different directory. When using the server with InTouch, we recommend that you save the configuration file in your application directory. For more information on the Configuration File, see "Saving the I/O Server's Configuration File" in this user's guide.

Start automatically as Windows NT Service

Enabling this option will cause the server to start as a Windows NT and Windows 2000 operating system service.

Windows NT and Windows 2000 offers the capability of running applications even when a user is not logged on to the system. This is valuable when systems must operate in an unattended mode. Enabling this option and rebooting the system will cause the server to run as a Windows NT and Windows 2000 operating system service. However, to view configuration information or to reconfigure the server, the user must log on to the system. Any server related problems that may arise such as missing adapter cards, licensing failures or device drivers not loading will not be visible to the user until a log on is performed. Disabling this option and rebooting the system will cause the server to run as a Windows NT and Windows 2000 application program once again.

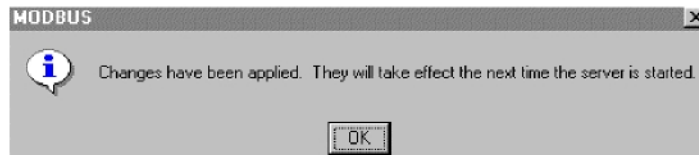
---

**Note** It is highly recommended that the server is configured and communicating successfully prior to running it as a Windows NT and Windows 2000 operating system service.

---

Click **Cancel** to close the dialog box without saving changes.

Click **OK** to accept the server settings. The following message box will appear:



Click **OK** to close the dialog box.

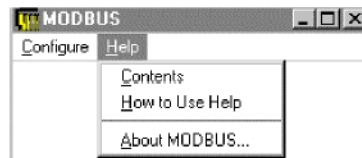
---

**Note** You must restart the server for the changes to take effect.



# Accessing I/O Server Help

The Help menu contains three options that are used to access help for the server.



The following briefly describes the Help menu options.

## Contents

This option is used to display the table of contents for the Help file.

## How to Use Help

This option is used to access a list of basic instructions for using the Help file.

## About MODBUS

This option is used to access miscellaneous information regarding the server, such as the software version, the copyright information, license information, etc.

Your FactorySuite system license information can be viewed through the license viewing utility that is launched from the **About** dialog box.

! For more information on the license viewing utility,  
see your *FactorySuite System Administrator's Guide*.

## Item Names

The Wonderware Modicon MODBUS I/O Server supports item/point names that are consistent with the point naming convention used by Modicon PLCs. The server allows you to select a Slave Type when you configure the topic definition for the PLC. The PLC Address Ranges supported are:

Point Type	484	Mi584/984	6 Digit	Tag Type	Access
Coil	1 - 999	1 - 9999	1 - 65536	Discrete	Read/Write
Contact	1001 - 1999	10001 - 19999	100001 - 165536	Discrete	Read Only
Input Register	3001 - 3999	30001 - 39999	300001 - 365536	Analog	Read Only
Holding Register	4001 - 4999	40001 - 49999	400001 - 465536	Analog	Read/Write
* Ext Memory Register		600001 - 699999	600001 - 69xxxx	Analog	Read/Write

**Note** Analog tagnames can be Integer or Real.

\* Extended Memory Registers are only supported on the 984B and 984-785. Addresses are assigned as follows:

File #	Range
1	600001-609999
2	610001-619999
3	620001-629999
4	630001-639999
5	640001-649999
6	650001-659999
7	660001-669999
8	670001-679999
9	680001-689999
10	690001-699999

The supported Item Names for **Daniel FC**, **Elliott FC**, **Omni FC**, and **Micromotion** slave device types are consistent with the Item Names used by Modicon PLCs. However, the Item Name and Read/Write Access may vary depending upon the application being used. For more information, see your user's guide for the device being used. The Flow Computer Address Ranges supported are:

FC Type	Point Type	Range	Tag Type
Daniel	Coil	1001 - 2999	Discrete
	Holding Register	3001 - 4999	Integer
	Holding Register	5001 - 6999	Integer
	Holding Register	7001 - 65535	Real
Elliot	Coil	1001 - 2999	Discrete
	Holding Register	3001 - 4999	Integer
	Holding Register	5001 - 6999	Integer
	Holding Register	7001 - 65535	Real
Omni	Coil	1001-2999	Discrete
	Holding Register	3001 - 4999	Integer
	Holding Register	4001-4999	String
	Holding Register	5001-6999	Integer
	Holding Register	7001-7999	Real
	Holding Register	13001-13999	Integer
	Holding Register	14001-14999	String
	Holding Register	15001-16999	Integer
	Holding Register	17001-17999	Real
Micromotion	**Coil	00002 - 00020	Discrete
	**Coil	00113 - 00160	Discrete
	**Read Only Discrete	10021 - 10038	Discrete
	Input Register	30001 - 30011	Integer
	Input Register	30120 - 30126	Integer
	Holding Register	40012 - 40051	Integer
	*Strings	40052 - 40119	String
	Holding Register	40124 - 40140	Integer
	***Floating Point Register Pairs	40141 - 40278	Real

\* Registers 50052 - 50119 map to 40052 - 40119. To access the first ASCII string located at 50052 - 50055, use item name 40052 - 40055 M. The other ASCII strings are accessed in the same way.

\*\* Caution! Advise only items within the ranges shown.

\*\*\* To access a float located at register pair 20141 - 20142, use the item name 40141 F. Other floats are accessed in the same way.

Floating Point Registers must be accessed as pairs only.

## Special Item/Point Naming Conventions

Special handling of data from Modicon equipment can be done by using the following conventions:

### Signed Registers

The Wonderware Modicon MODBUS I/O Server normally allows register values in the range of 0 to 65535. Registers may be treated as 16-bit signed quantities having values between -32,768 and 32,767.

To specify that a register is to be treated as a signed quantity, append a space and the letter **S** to the item name. For example, to indicate that the first Holding Register is signed, enter "**40001 S**" for the item name.

### Long Integers

Pairs of registers can be treated as 32-bit signed integers. This is done by appending a space and the letter **L** to the item name. For example: **40001 L**.

### Floating Point

Pairs of registers can be treated as IEEE 32-bit floating point numbers. This is done by appending a space and the letter **F** to the item name. For example: **40001 F**.

### Bits In Registers

Individual bits in registers can be read as discrete tags by using the notation **rrrr:b**.

Where **rrrr** specifies a valid input register or holding register and **b** specifies a bit position between **1** and **16** (**1** specifies the most significant bit of the register).

Examples:

<b>40001:1</b>	Most significant bit of first holding register
<b>30008:16</b>	Least significant bit of an input register
<b>4001:5</b>	5th from the most significant bit of first holding register in a 484 controller.

---

**Note** Bits in registers are read-only. The MODBUS protocol has no commands to write individual register bits.

## Explicit Conversions

Registers are normally treated as integers. This can be altered for specific points by adding a blank space (using the spacebar) and one of the following to the point name:

- L** Long
- F** Float
- I** Integer
- M** Message (ASCII Strings)

## Modulo-10000 Points

Two or three consecutive registers may be interpreted as a single numeric quantity. In doing so, each of the component registers must be in the range of 0-9999. For example, two registers "**40001-40002**" can represent numbers between 0 and 99999999.

Overflow becomes a possibility when grouping three consecutive registers for interpretation as a single numeric quantity. The largest number that may be represented in the PLC with three consecutive Modulo-10000 registers is 999,999,999,999. Unfortunately, the largest number which can be contained in an 'integer' type variable is 2,147,483,647. The latter number is used by the Server to represent an overflow condition. Therefore, the maximum usable value represented in three Modulo-10000 registers is 2,147,483,646 ( or < 21 >< 4748 >< 3646 > ). Any numbers larger than this will be clamped at 2,147,483,647.

## ASCII Strings

Multiple consecutive registers (1-125) can be treated as a string of ASCII characters. The first character of the string is in the high-order 8-bits of the lowest numbered register. Three options are provided for the representation of variable length strings:

- A. The string may be padded with ASCII spaces (hex 20).
- B. The string may be terminated with a zero byte.
- C. The length may be stored in the first byte.

Strings of ASCII characters can be stored in consecutive registers in a PLC. The number of registers that are allocated to the storage of an individual string must be fixed. However, the string contained in those registers can have a variable length up to twice the number of registers allocated. Strings are stored in MODBUS registers from lowest numbered register to the highest and within each register, first the most significant byte, then the least significant byte. The MODBUS I/O Server supports three methods of storing strings of ASCII characters in registers.

## Full Length

If strings are stored in this manner, the string always uses all of the registers allocated. If the string is shorter than the allocation of registers, it is padded with ASCII space characters (hex 20) to the full length. For example, string item **"40001-40010 M"** containing the ASCII string "Wonderware":

	<b>MSB</b>	<b>LSB</b>	
40001	57	6F	"Wo"
40002	6E	64	"nd"
40003	65	72	"er"
40004	77	61	"wa"
40005	72	65	"re"
40006	20	20	" "
40007	20	20	" "
40008	20	20	" " padded to the end
40009	20	20	" " with ASCII spaces
40010	20	20	" "

## C Style

If strings are stored in this manner, the end of the string is marked by a byte of zero immediately following the last character in the string. This option is so named because this is the way strings are stored in the "C" programming language. For example, string item **"40001-40010 M"** containing the ASCII string "Wonderware":

	<b>MSB</b>	<b>LSB</b>	
40001	57	6F	"Wo"
40002	6E	64	"nd"
40003	65	72	"er"
40004	77	61	"wa"
40005	72	65	"re"
40006	0	x	end marked by zero
40007	x	x	the remaining
40008	x	x	bytes
40009	x	x	are
40010	x	x	unused

## Pascal Style

If strings are stored in this manner, the length of the string is denoted by a length byte which occupies the first byte of the string (MSB of the first register). This option is so named because this is the way strings are stored in the "Pascal" programming language (for most compilers). For example: string item "**40001-40010 M**" containing the ASCII string "Wonderware":

	MSB	LSB	
40001	0A	57	length = 10 "W"
40002	6F	6E	"o n"
40003	64	65	"de"
40004	72	77	"r w"
40005	61	72	"ar"
40006	65	x	"e"
40007	x	x	the remaining
40008	x	x	bytes
40009	x	x	are
40010	x	x	unused

## Absolute Notation

The actual PLC register numbers used with absolute notation are offset by a value of +1. For example, Register ;"**400020**" in absolute notation would be **19 HR**.

Absolute item naming conventions are available that are independent of PLC model numbers. The absolute notation allows accessing of the four MODBUS data types, each with an address from 0 to 65535. The data types are indicated by the item name suffix characters as follows:

nnnnn **DO** (Discrete Output) Refers to the same data MODBUS calls "coils" (valid range is 0 **DO** through 65535 **DO**.)

nnnnn **DI** (Discrete Input) Refers to the same data MODBUS calls "contacts" (valid range is 0 **DI** through 65535 **DI**.)

nnnnn **IR** (Input Registers) Refers to the same data MODBUS calls "input registers" (valid range is 0 **IR** through 65535 **IR**.)

nnnnn **HR** (Holding Registers) Refers to the same data MODBUS calls "holding registers" (valid range is 0 **H**) Refers **R** through 65535 **HR**.)

nnnnn **PV** (Process Variable to holding registers but treats them as floating points and assumes 2 registers per floating point number. The valid range is 0 **PV** through 32767 **PV**.)

The **IR** and **HR** absolute notations can also be combined with the following conversions: **L** (long), **F** (floating point) or **S** (signed.) For example:

219 **HRS** 16 bit signed integer 0 **HRL** 32 bit signed integer 100 **HRF** 32 bit floating point

## Monitoring the Status of Communications with a PLC

For each topic name (PLC), there is a built-in discrete item that can be used to monitor the status of communications with the PLC. The discrete item, **Status**, is set to **0** when communication with the PLC fails and is set to **1** when communication is successful.

### Using the Status Item in Excel

The status of the PLC communications can be read into Excel by entering the following DDE reference formula in a cell on a spreadsheet:

```
=MODBUS|ModSlave5!Status
```

where:

<b>MODBUS</b>	Is the name of the server application.
<b>ModSlave5</b>	Is the exact topic name defined in the server for the PLC.
<b>Status</b>	Built-in discrete item used to monitor the status of communications with the PLC.



## Monitoring the Status of Communications with InTouch

InTouch supports built-in topic names called DDEStatus and IOStatus that are used to monitor the status of communications between the server and InTouch. For more information on the built-in topic names DDEStatus and IOStatus, see your online “InTouch User’s Guide.”

### Using DDEStatus and IOStatus in Excel

The status of communication between the server and InTouch can be read into Excel by entering the following DDE reference formula in a cell on a spreadsheet:

`=view|DDEStatus!ModSlave5`

or

`=view|IOStatus!ModSlave5`

where:

<b>view</b>	Is the name of the InTouch application.
<b>[DDE][IO]Status</b>	Built-in topic name used to monitor the status of communications between the server and InTouch.
<b>ModSlave5</b>	The exact topic name defined in the server for the PLC.

## Reading Values from the I/O Server into Excel

Values may be read directly into Excel spreadsheets from the server by entering a DDE formula into a cell using the following format:

**=applicationname|topicname!itemname**

Example formula:

**= MODBUS|ModSlave5!40001**

where:

<b>MODBUS</b>	Is the name of the server application name.
<b>ModSlave5</b>	Is the exact topic name defined in the server for the PLC.
<b>40001</b>	Is the actual location in the PLC that contains the data value. This is the item name.

In this example, each time the value of **40001** changes in the PLC, the server will automatically send the new value to the cell containing the formula in Excel.

---

**Note** Refer to the Microsoft Excel manual for complete details on entering Remote Reference formulas for cells.

## Writing Values to the I/O Server from Excel

Values may be written to the server from Microsoft Excel by creating an Excel macro that uses the **POKE** command. The proper command is entered in Excel as follows:

```
channel=INITIATE("applicationname","topicname")
=POKE(channel,"itemname", Data_Reference)
=TERMINATE (channel)
=RETURN()
```

The following describes each of the above **POKE** macro statements:

```
channel=INITIATE("applicationname","topicname")
```

Opens a channel to a specific topic name (defined in the server) in a particular application name (the executable name less the .EXE) and assigns the number of that opened channel to **channel**.

---

**Note** By using the **channel=INITIATE** statement the word **channel** must be used in the **=POKE** statement instead of the actual cell reference. The "**applicationname**" and "**topicname**" portions of the formula must be enclosed in quotation marks.

---

```
=POKE(channel,"itemname", Data_Reference)
```

**POKEs** the value contained in the **Data\_Reference** to the specified item name (actual location in the PLC) via the **channel** number returned by the previously executed **INITIATE** function. **Data\_Reference** is the row/column ID of the cell containing the data value.

```
=TERMINATE(channel)
```

Closes the channel at the end of the macro. Some applications have a limited number of channels therefore, they should be closed when finished. **Channel** is the channel number returned by the previously executed **INITIATE** function.

```
=RETURN()
```

Marks the end of the macro.

---

**Note** Refer to the **.XLM** sample Excel poke macro provided on the server program disk. Also refer to the Microsoft Excel manual for complete details on entering Remote Reference formulas for cells.

## Troubleshooting I/O Server Communication Problems

This section provides you with some simple steps that can be taken to ascertain and correct communication problems. The problems described here represent the most probable causes of communication failure.

---

**Note** This is a general troubleshooting guide and for the sake of brevity we cannot cover every possible source of communication problems.

---

### Debugging Communication Between InTouch and an I/O Server

This section explains the most common error situations that can occur when attempting to establish communication between InTouch and a server.

Servers are Window applications that communicate with I/O, PLCs, and/or other data sources. If a server supports either the Microsoft Dynamic Data Exchange (DDE) or the Wonderware SuiteLink protocol, it is capable of communicating with the Wonderware InTouch program.

---

**Note** All Wonderware version 7.0 or later servers support both DDE and SuiteLink. However, the SuiteLink protocol is only supported on the Windows NT (version 4.0 or later) and Windows 2000 operating system.

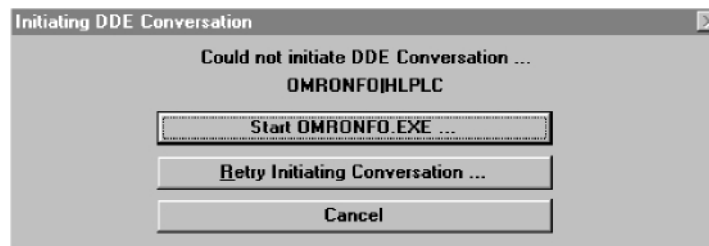
Servers respond to data requests made by other applications. Requesting applications are called clients. When WindowViewer acts as a client and requires the value of an item, it contacts the server and requests the item's value. The server will report the value and update WindowViewer only if a change occurs. All WindowViewer data requests provide information relating an item to a register, coil number, or I/O data point understood by the server. The server uses the information to automatically handle all messages to and from I/O, hardware devices (PLC), and/or other data sources.

---

**Note** We highly recommend starting all the servers required by the InTouch application before starting WindowViewer. InTouch (versions prior to 7.0) will display the **Initiating DDE Conversation** message box for each uninitiated conversation.

For example:

If you start up WindowViewer and cannot successfully establish a conversation with a server, the following Initiating DDE Conversation dialog box will appear:



The information in the second line indicates that you have at least one I/O type tagname defined in your Tagname Dictionary that is associated with an Access Name that defines OMRONFO as the Application Name, and HLPLC as the

Topic Name. Make note of exactly how the application and topic names are spelled.

" This example only applies when using a version of InTouch prior to InTouch 7.0.

To troubleshoot communication problems between WindowViewer and the server, perform the following steps as listed below.

**# Verify the I/O Server is running.**

1. Start the server program.
2. Verify the server is running by checking to see if it is in the Windows Task List.

On Windows NT and Windows 2000 click the right mouse button on the Windows taskbar and select Task Manager from the menu. Click the Applications tab to view all currently running applications. Or press the CTRL+SHIFT+ESC keys.

On Windows 95, press the ALT+TAB keys while holding down the ALT key. On

Windows 3.1 or Windows for Workgroups, press the CTRL+ESC keys.

**# If the I/O Server is running, verify the I/O Server's program name is correct in all WindowMaker Access Name definitions.**

1. Switch to (or start) WindowMaker. Select Access Names from the Special Menu, the Access Name Definitions dialog box appears listing all Access Names defined in the WindowMaker.
2. In the Access Names list, select the Access Name referencing the server and click Modify. The Modify Access Name dialog box will appear.
3. Verify the server's program name in the Application Name box is correct. If it is wrong then correct it and click OK , else click Cancel.

" The server's exact "executable name" must be typed in the Application Name box in all Access Name definitions. The ".exe" extension is not used.

" If you are debugging a remote tagname reference, also verify that the node name for the remote computer in the Node Name box is correct.

4. Repeat steps 2 & 3 and verify the server program name is correct in all Access Names that use it.

**# If you still cannot establish a conversation, verify the exact topic name used in the WindowMaker Access Name definitions are defined in the I/O Server program.**

1. Close WindowViewer if it is running. The server cannot be configured if WindowViewer is running.
2. Start the server program.
3. From the server's Configure menu select Topic Definition. The Topic Definition dialog box appears listing all topic names defined in the server.
4. Verify that the topic name exists and is spelled exactly the same (including spaces) as the topic name referenced in the WindowMaker Access Name definition.

- " Blank spaces cannot follow the topic name in either the server's Topic Definition or the Access Name definition.
- 5. If the topic name is different, either correct it in the server or switch to WindowMaker and correct it in the Access Name definition.
- 6. Once you performed the above procedure, restart WindowViewer and switch to the server program. Data should now appear in the server's program window to indicate that WindowViewer and the server are communicating.
  - " The data in the server's program window indicates the read and write messages the server is sending to and receiving from the PLC. These are not error messages; only status messages are written to the server's program window.
- 7. If no data appears in the server's program window, switch to the Wonderware Logger to check for error messages. For example, a common error message is: "Error for DDE: OMRONFO|HLPLC!<null>("item") Advise failed"

This message appears when the item defined in one or more tagnames is invalid for the server.

" InTouch tagnames use specific naming conventions when accessing data from a server. The valid item names for all Wonderware servers are documented in their respective user's guides. Typically, the item naming conventions used by each server are consistent with the names used by the equipment manufacturer.

! For more information on the Wonderware Logger, see your online *FactorySuite System Administrator's Guide*.

**# If you are still experiencing problems, continue with the following troubleshooting section.**

## Debugging Communication Between SuiteLink and an I/O Server

If you have successfully applied the debug techniques listed in the previous section and are still experiencing communication problems to a server that is attempting to communicate using the SuiteLink protocol, perform the following steps as listed below:

**# Verify the I/O Server supports the Wonderware SuiteLink protocol, that is, the I/O Server is version 7.0 or above.**

**# Try communicating to the I/O Server using the DDE protocol. If this is not possible, then proceed to the next troubleshooting section otherwise continue with the following steps:**

1. Verify Microsoft's TCP/IP stack is installed and configured properly.

" SuiteLink uses the Microsoft TCP/IP stack for its communications even if the client application and the server reside on the same node.

2. If you do not have an Ethernet card to bind to the TCP/IP stack, install the Microsoft Loop Back Adapter.

3. Install the Microsoft TCP/IP stack.

## Debugging Communication Between an I/O Server and a PLC

This section provides you with simple steps to diagnose and correct server to PLC communication problems. The debug techniques listed below address both serial and board servers. Disregard any information that is not applicable to the server type that you are using.

When attempting to establish communication between a server and a PLC, if no data appears in the server's program window and the data items are not updating in WindowViewer, switch to the Wonderware Logger and check for error messages.

~ For more information on the Wonderware Logger,  
see your online *FactorySuite System  
Administrator's Guide*.

For example, some of the most common errors that may appear in the Wonderware Logger for serial servers are:

Response Timeout  
Receive Overrun  
Framing Errors

---

**Note** Unless specified otherwise, most serial communication based servers are full duplex. If you require a server for half duplex (one that monitors the CTS and RTS lines) or if you are not sure whether the PLC's protocol is full or half duplex, call your PLC supplier.

---

### Check your cabling to the PLC.

Is it wired correctly? Check for shorts, loose wires, broken wires, crossed wires, and so on.

:- A continuity tester can be helpful here.

### Verify the I/O Server's serial configuration settings (Parity, Stop Bits, Baud Rate, Handshaking and so on) against the settings in the hardware device.

### Verify the communication port is working properly in Windows.

1. Close the server program.
  - ~ Also, if you are using an AT type computer, two devices cannot share interrupts. Verify that the communication port you are using has a unique interrupt setting.
2. On Windows 95 or Windows NT and Windows 2000 start the HyperTerminal program.
3. Configure the Terminal (or HyperTerminal) program to use the same communication port with the same settings (baud rate, parity, stop bits and so on) as the hardware device.
4. Connect a null modem cable to a second computer's port.

5. On the second computer, start and configure the Terminal (or HyperTerminal) program with the same settings as the first computer.
6. Verify that you can send data between the two computers.
  - " If you do not have two computers and the computer you are using has another port, start two instances of the Terminal (or HyperTerminal) program with each configured to their own port. Then try communicating between them.
  - " If you have an external modem, connect the modem to the communication port that you are testing to see if you can dial out.
7. If the communication port does not appear to be functioning properly, check your environment files (**AUTOEXE.BAT**, **CONFIG.SYS**, **SYSTEM.INI**, and **WIN.INI**). Look for suspicious programs or drivers that might be taking control of the port or its interrupt before the server is loaded. Always keep your environment files as clean as possible. If you are in doubt about an entry, comment it out.
8. If the previous step was unsuccessful, try another communication port or another computer.

---

**Note** A common misconception of connecting to a PLC with a DOS program and the same communication port will work in Windows is not the case! Windows is an entirely different environment than DOS.

---

#### # Does your computer lock up?

Verify the COM port's IRQ's do not conflict with each other or with other communication boards in the computer.

#### # If the PLC or field device has more than one COM port, verify the connection to the correct port.

The COM port on your computer uses the RS-232 hardware communication standard and connects the cable from the COM port to an RS-232 compliant device.

---

**Note** To connect to an RS-422 or RS-485 port on the PLC, you need an RS-232 to RS-422/485 conversion device.

If possible, use an external converter instead of a board-based converter that plugs into a slot in the computer. A board-based converter is difficult to get working for inexperienced users. If a board-based converter is not set up properly, it can conflict with other communication boards in the computer such as, internal modems.

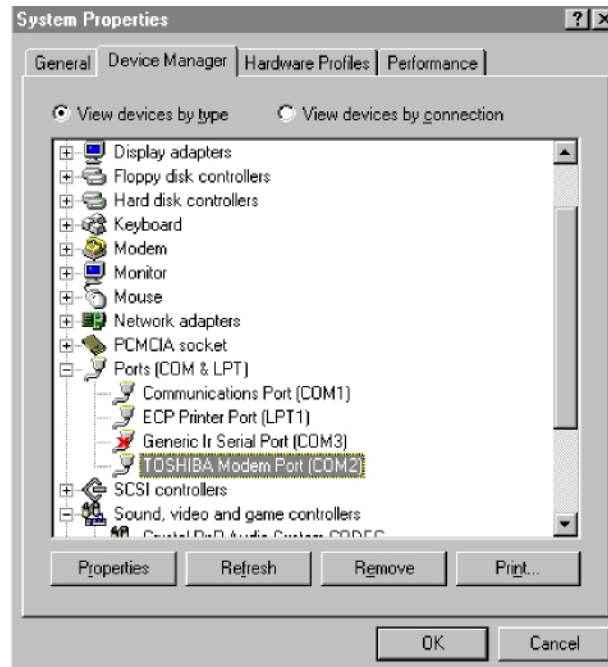
---

#### # If you are using the Windows 95 operating system, verify the following:

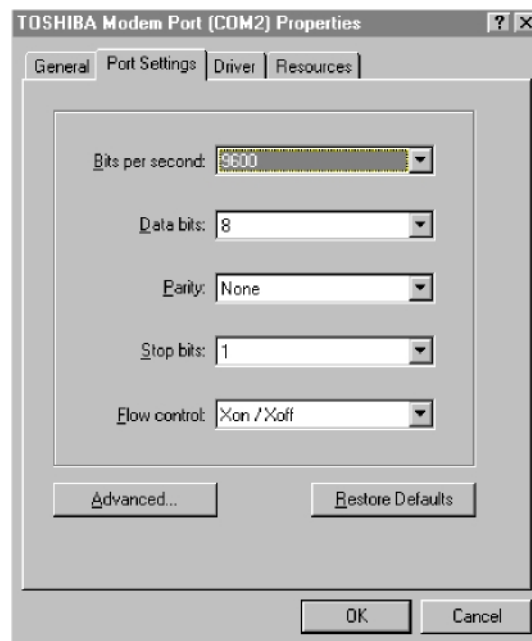
1. Click Start on the Windows taskbar. Point to Settings, then click Control Panel in the menu. The Control Panel dialog box will appear.



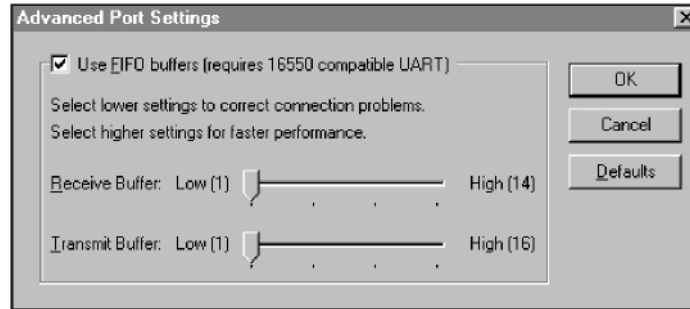
2. Double-click the System icon. The System Properties dialog box will appear. Click the Device Manager tab and select the COM port that you are using for the server. For example:



3. Click Properties. The Properties dialog box will appear. Click the Port Settings tab.



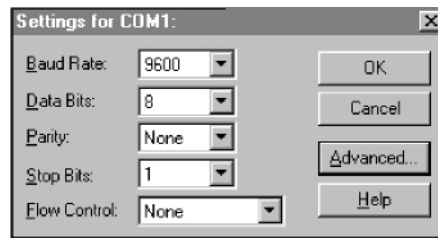
4. Click Advanced. The Advanced Port Settings dialog box appears:



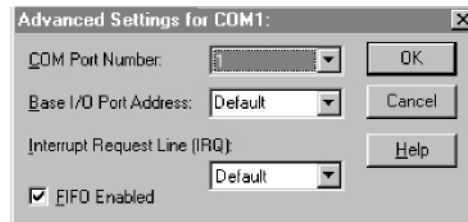
- Lowering the default Receive Buffer and Transmit Buffer settings to their minimum may solve I/O communication problems for portable computers (notebook or laptops) and framing errors for standard computers.

**# If you are using the Windows NT and Windows 2000 operating system, verify the following:**

- Click Start on the Windows taskbar. Point to Settings, then click Control Panel in the menu. The Control Panel dialog box will appear.
- Double-click the Ports icon, the Ports dialog box will appear.
- Select a port and click the Settings button. The Settings for COMx dialog box appears:



- Click Advanced. The Advanced Settings for COMx dialog box appears:



- Lowering the setting for the Interrupt Request Line (IRQ) value to the minimum may solve I/O communication problems for portable computers (notebook or laptops) and framing errors for standard computers.

**# How long is your RS-232 cable?**

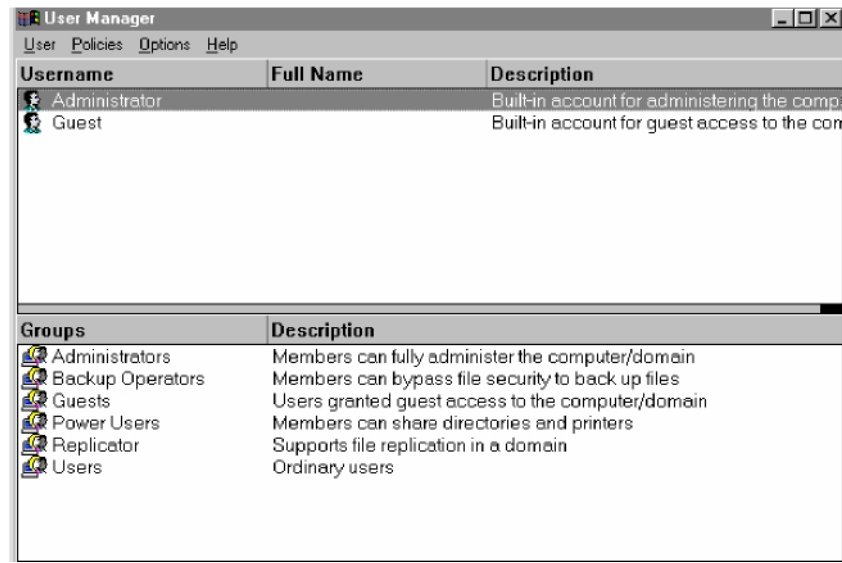
Fifteen meters (fifty feet) is the maximum practical length for the RS-232 standard.

**# Try using a different COM port for the I/O Server.**

**# If you are installing an I/O Server or configuring a board-based I/O Server on a computer running on the Windows NT and Windows 2000 operating system, log on with Administrator privileges.**

" Without Administrator privileges, the server and Server Install program cannot make the necessary edits to the Windows NT and Windows 2000 Registry during installation or board configuration of the server.

1. Click Start on the Windows taskbar. Point to Programs, then to Administrative Tools (Common), and click User Manager in the menu. The User Manager dialog box will appear:



2. Double-click the Username you typed in during log on.
3. If the User Properties dialog box does not appear, you do not have Administrator privileges.
4. If the User Properties dialog box does appear, click on the Groups button and verify "Administrators" is in the "Member of" list.

**# If you experience occasional or random communication errors in the Wonderware Logger, such as "Response Timeouts," check for noise.**

Do the physical cables run past any known noise sources such as photocopiers, fluorescent lamps, fans, pumps, motors or generators? Are the cables properly shielded from its environment? With radio modems and satellite link ups, occasional communications errors in the Wonderware Logger are normal and to be expected as long as they do not adversely impact the flow of data.

**# Increase the Reply Timeout setting in the I/O Server to a value between 5 and 10 seconds.**

Not allowing the PLC or field device enough time to respond to the server's request for data may result in communication errors.

**# Verify the PLC is properly configured and the cable is good by using the programming software for the PLC.**

1. Connect to the PLC with the programming software. The connection must be through the same port and cable. Go into the programming software configuration and write down what the communications parameters are (baud rates, routes, node number, error checking, etc.).
2. Close the programming software. Open the I/O Server and verify the communications settings are the same.
3. Poke data into the PLC with InTouch or WWClient.
4. Shut down the server and use the programming software to verify that the values were correctly poked.

~ Performance of this test depends upon the type of PLC you are using.

**Reinstall the I/O Server and verify that you are using the latest version.**

Wonderware is continually improving our servers and using the latest version will guarantee the best results.

~ New versions of the Wonderware I/O Servers are released regularly on the Knowledge Base CD. These are available to Comprehensive Support customers on the Wonderware WEB site at: <http://www.wonderware.com>

**Move the I/O Server's configuration file to another location on the computer's hard drive. This will clear all configuration for the I/O Server, then reconfigure the I/O Server.**

~ Wonderware server configuration files are typically the exact same name as the server's executable name with the .CFG extension. For example, **OMRONFO.CFG**. Refer to the Configuration File section of the specific server of this user's guide for the exact name of the configuration file.

**If possible, reinstall the Windows operating system.**

Files installed earlier on your computer or the NT registry may have been corrupted or accidentally modified.

**If these troubleshooting suggestions do not solve your problem, there may be a problem with your computer. There are many subtle differences between the various computer hardware brands. Using a computer that is a different brand and meets the following criteria:**

1. Select a different PC manufacturer and if this is not possible, try a different PC model from the same manufacturer.
2. The computer can not use an OEM (Original Equipment Manufacturer) version of Microsoft Windows. We highly recommend using only a Microsoft Windows product. Contact your vendor to determine if installing an off-the-shelf copy of Microsoft Windows will cause any problems.

**If you feel you have tested all possible situations that may be causing your failed I/O communications, contact your local Wonderware distributor for technical support.**

~ For more information on obtaining technical support, see your online *FactorySuite System Administrator's Guide*.

# **LAMPIRAN B**

## Introduction

The Modbus protocol is a master-slave protocol that allows for one, and only one, master to request responses from slaves, or to act based on the request. The master can address individual slaves, or can initiate a broadcast message to all slaves. Slaves return a message (response) to queries that are addressed to them individually. Responses are not returned to broadcast queries from the master.

<b>CAUTION</b>
<hr/> <b>UNEXPECTED EQUIPMENT OPERATION</b> Be sure that there is only one Modbus master controller on the bus and that each Modbus slave has a unique address. Failure to observe this precaution may lead to corrupted data or unexpected and ambiguous results. Be sure that all Modbus slaves have unique addresses. No two slaves should have the same address. Failure to observe this precaution may lead to corrupted data or unexpected and ambiguous results. Failure to follow this precaution can result in injury or equipment damage.

## Hardware Configuration

A Modbus link can be established on either the EIA RS-232 or EIA RS-485 port and can run on as many as two communications ports at a time. Each of these ports can be assigned its own Modbus address, using [system bit %S101 and system words %SW101 and %SW102.](#)

The table below lists the devices that can be used:

<b>Remote</b>	<b>Port</b>	<b>Specifications</b>
TWDLCA10/16/24DRF, TWDLCA40DRF, TWDLMDA20/40DTK, TWDLMDA20DRT	1	Base controller supporting a 3-wire EIA RS-485 port with a miniDIN connector.
TWDNOZ232D	2	Communication module equipped with a 3-wire EIA RS-232 port with a miniDIN connector. Note: This module is only available for the Modular controllers. When the module is attached, the controller cannot have an Operator Display expansion module.
TWDNOZ485D	2	Communication module equipped with a 3-wire EIA RS-485 port with a miniDIN connector. Note: This module is only available for the Modular controllers. When the module is attached, the controller cannot have an Operator Display expansion module.
TWDNOZ485T	2	Communication module equipped with a 3-wire EIA RS-485 port with a terminal.

		Note: This module is only available for the Modular controllers. When the module is attached, the controller cannot have an Operator Display expansion module.
TWDNAC232D	2	Communication adapter equipped with a 3-wire EIA RS-232 port with a miniDIN connector. Note: This adapter is only available for the Compact 16, 24 and 40 I/O controllers and the Operator Display expansion module.
TWDNAC485D	2	Communication adapter equipped with a 3-wire EIA RS-485 port with a miniDIN connector. Note: This adapter is only available for the Compact 16, 24 and 40 I/O controllers and the Operator Display expansion module.
TWDNAC485T	2	Communication adapter equipped with a 3-wire EIA RS-485 port with a terminal connector. Note: This adapter is only available for the Compact 16, 24 and 40 I/O controllers and the Operator Display expansion module.
TWDXCPODM	2	Operator Display expansion module equipped with a 3-wire EIA RS-232 port with a miniDIN connector, a 3-wire EIA RS-485 port with a miniDIN connector and a 3-wire EIA RS-485 port with a terminal. Note: This module is only available for the Modular controllers. When the module is attached, the controller cannot have a Communication expansion module.
Note: The presence and configuration (RS232 or RS485) of Port 2 is checked at power-up or at reset by the firmware executive program.		

### Nominal Cabling

Nominal cable connections are illustrated below for both the EIA RS-232 and the EIA RS-485 types.

Note: If port 1 is used on the Twido controller, the DPT signal on pin 5 must be tied to the circuit common (COM) on pin 7. This signifies to the Twido controller that the communications through port 1 is Modbus and is not the protocol used to communicate with the TwidoSoft software.

The cable connections made to each remote device are shown below.

### EIA RS-485 Line Polarization on TWDLCA•40DRFC Controllers

There is no internal pre-polarization in TWDLCA•40DRF controllers. Therefore,

external line polarization is required when connecting the TWDLCA•40DRF Modbus master controller to the EIA-485 Modbus network.  
 (When there is no data activity on an EIA-485 balanced pair, the lines are not driven and, therefore, susceptible to external noise or interference. To ensure that its receiver stays in a constant state, when no data signal is present, the Modbus master device needs to bias the network via external line polarization.)

Note: EIA RS-485 external line polarization must be implemented on the Modbus Master controller only; you must not implement it on any slave device.

The external line polarization assembly on the TWDLCA•40DRF mini-DIN RS-485 EIA line shall consist in:

- One pull-up resistor to a 5V voltage on D1(A+) circuit,
- One pull-down resistor to the common circuit on D0(B-) circuit.

The following figure illustrates the external line polarization assembly on the TWDLCA•40DRF mini-DIN RS-485 EIA line:

External polarization can be performed in any of two ways:

- Connecting externally the user-provided polarization assembly via mini-DIN fly cable. (Please refer to pin definition for connector.)
- Using a polarization tap (configured for 2-wire polarization) and polarization assembly (available soon from the catalog).

### Software Configuration

To configure the controller to use a serial connection to send and receive characters using the Modbus protocol, you must:

Step	Description
1	Configure the serial port for Modbus using TwidoSoft.
2	Create in your application a transmission/reception table that will be used by the EXCHx instruction.

### Configuring the Port

A Twido controller can use its primary port 1 or an optionally configured port 2 to use the Modbus protocol. To configure a serial port for Modbus:

Step	Action
1	Define any additional communication adapters or modules configured to the base.
2	Right-click on the port and click Edit Controller <b>Comm</b> Setup... and change serial port type to "Modbus".
3	Set the associated communication parameters.



---

## Modbus Master

Modbus master mode allows the controller to send a Modbus query to a slave, and to wait for the response. The Modbus Master mode is only supported via the EXCHx instruction. Both Modbus ASCII and RTU are supported in Modbus Master mode. The maximum size of the transmitted and/or received frames is 250 bytes. Moreover, the word table associated with the EXCHx instruction is composed of the control, transmission and reception tables.

	<b>Most significant byte</b>	<b>Least significant byte</b>
Control table	Command	Length (Transmission/Reception)
	Reception offset	Transmission offset
Transmission table	Transmitted Byte 1	Transmitted Byte 2
	...	...
	...	Transmitted Byte n
	Transmitted Byte n+1	
Reception table	Received Byte 1	Received Byte 2
	...	...
	...	Received Byte p
	Received Byte p+1	

Note: In addition to queries to individual slaves, the Modbus master controller can initiate a broadcast query to all slaves. The command byte in case of a broadcast query must be set to 00, while the slave address must be set to 0.

---

## Control table

The Length byte contains the length of the transmission table (maximum 250 bytes), which is overwritten by the number of characters received at the end of the reception, if reception is requested.

This parameter is the length in bytes of the transmission table. If the Tx Offset parameter is equal to 0, this parameter will be equal to the length of the transmission frame. If the Tx Offset parameter is not equal to 0, one byte of the transmission table (indicated by the offset value) will not be transmitted and this parameter is equal to the frame length itself plus 1.

The Command byte in case of Modbus RTU request (except for broadcast) must always equal to 1 (Tx and Rx).

The Tx Offset byte contains the rank (1 for the first byte, 2 for the second byte, and so on) within the Transmission Table of the byte to ignore when transmitting the bytes. This is used to handle the issues associated with byte/word values within the Modbus protocol. For example, if this byte contains 3, the third byte would be ignored, making the fourth byte in the table the third byte to be transmitted.

The Rx Offset byte contains the rank (1 for the first byte, 2 for the second byte, and so on) within the Reception Table to add when transmitting the packet. This is used to

handle the issues associated with byte/word values within the Modbus protocol. For example, if this byte contains 3, the third byte within the table would be filled with a ZERO, and the third byte was actually received would be entered into the fourth location in the table.

---

#### Transmission/reception tables

When using either mode (Modbus ASCII or Modbus RTU), the Transmission table is filled with the request prior to executing the EXCHx instruction. At execution time, the controller determines what the Data Link Layer is, and performs all conversions necessary to process the transmission and response. Start, end, and check characters are not stored in the Transmission/Reception tables.

Once all bytes are transmitted, the controller switches to reception mode and waits to receive any bytes.

Reception is completed in one of several ways:

- timeout on a character or frame has been detected,
- end of frame character(s) received in ASCII mode,
- the Reception table is full.

Transmitted byte X entries contain Modbus protocol (RTU encoding) data that is to be transmitted. If the communications port is configured for Modbus ASCII, the correct framing characters are appended to the transmission. The first byte contains the device address (specific or broadcast), the second byte contains the function code, and the rest contain the information associated with that function code.

Note: This is a typical application, but does not define all the possibilities. No validation of the data being transmitted will be performed.

Received Bytes X contain Modbus protocol (RTU encoding) data that is to be received. If the communications port is configured for Modbus ASCII, the correct framing characters are removed from the response. The first byte contains the device address, the second byte contains the function code (or response code), and the rest contain the information associated with that function code.

Note: This is a typical application, but does not define all the possibilities. No validation of the data being received will be performed, except for checksum verification.

---

#### Modbus Slave

Modbus slave mode allows the controller to respond to standard Modbus queries from a Modbus master.

When TSXPCX1031 cable is attached to the controller, TwidoSoft communications are started at the port, temporarily disabling the communications mode that was running prior to the cable being connected.

The Modbus protocol supports two Data Link Layer formats: ASCII and RTU. Each is defined by the Physical Layer implementation, with ASCII using 7 data bits, and RTU

using 8 data bits.

When using Modbus ASCII mode, each byte in the message is sent as two ASCII characters. The Modbus ASCII frame begins with a start character (':'), and can end with two end characters (CR and LF). The end of frame character defaults to 0x0A (line feed), and the user can modify the value of this byte during configuration. The check value for the Modbus ASCII frame is a simple two's complement of the frame, excluding the start and end characters.

Modbus RTU mode does not reformat the message prior to transmitting; however, it uses a different checksum calculation mode, specified as a CRC.

The Modbus Data Link Layer has the following limitations:

Address 1-247

Bits: 128 bits on request

Words: 125 words of 16 bits on request

---

## Message Exchange

The language offers two services for communication:

EXCHx instruction: to transmit/receive messages

%MSGx Function Block: to control the message exchanges.

The Twido controller uses the protocol configured for that port when processing an EXCHx instruction.

Note: Each communications port can be configured for different protocols or the same. The EXCHx instruction or %MSGx function block for each communications port is accessed by appending the port number (1 or 2).

---

## EXCHx Instruction

The EXCHx instruction allows the Twido controller to send and/or receive information to/from Modbus devices. The user defines a table of words (%MWi:L) containing control information and the data to be sent and/or received (up to 250 bytes in transmission and/or reception). The format for the word table is described earlier. A message exchange is performed using the EXCHx instruction:

The Twido controller must finish the exchange from the first EXCHx instruction before a second can be launched. The %MSGx function block must be used when sending several messages.

The processing of the EXCHx list instruction occurs immediately, with any transmissions started under interrupt control (reception of data is also under interrupt control), which is considered background processing.

---

## %MSGx Function Block

The use of the %MSGx function block is optional; it can be used to manage data exchanges. The %MSGx function block has three purposes:

**Communications error checking**

The error checking verifies that the parameter L (length of the Word table) programmed with the EXCHx instruction is large enough to contain the length of the message to be sent. This is compared with the length programmed in the least significant byte of the first word of the word table.

**Coordination of multiple messages**

To ensure the coordination when sending multiple messages, the %MSGx function block provides the information required to determine when transmission of a previous message is complete.

**Transmission of priority messages**

The %MSGx function block allows current message transmissions to be stopped in order to allow the immediate sending of an urgent message.

The %MSGx function block has one input and two outputs associated with it:

<b>Input/Output</b>	<b>Definition</b>	<b>Description</b>
R	Reset input	Set to 1: re-initializes communication or resets block (%MSGx.E = 0 and %MSGx.D = 1).
%MSGx.D	Communication complete	0: request in progress. 1: communication done if end of transmission, end character received, error, or reset of block.
%MSGx.E	Error	0: message length OK and link OK. 1: if bad command, table incorrectly configured, incorrect character received (speed, parity, and so on.), or reception table full.

**Limitations**

It is important to note the following limitations:

Port 2 presence and configuration (RS232 or RS485) is checked at power-up or reset

Any message processing on Port 1 is aborted when the TwidoSoft is connected

EXCHx or %MSG can not be processed on a port configured as Remote Link

EXCHx aborts active Modbus Slave processing

Processing of EXCHx instructions is not re-tried in the event of an error

Reset input (R) can be used to abort EXCHx instruction reception processing

EXCHx instructions can be configured with a time out to abort reception

Multiple messages are controlled via %MSGx.D

---

### Error and Operating Mode Conditions

If an error occurs when using the EXCHx instruction, bits %MSGx.D and %MSGx.E are set to 1 and system word %SW63 contains the error code for Port 1, and %SW64 contains the error code for Port 2.

System Words	Use
%SW63	EXCH1 error code: 0 - operation was successful 1 - number of bytes to be transmitted is too great (> 250) 2 - transmission table too small 3 - word table too small 4 - receive table overflowed 5 - time-out elapsed 6 - transmission 7 - bad command within table 8 - selected port not configured/available 9 - reception error 10 - can not use %KW if receiving 11 - transmission offset larger than transmission table 12 - reception offset larger than reception table 13 - controller stopped EXCH processing
%SW64	EXCH2 error code: See %SW63.

---

### Master Controller Restart

If a master/slave controller restarts, one of the following events happens:

A cold start (%S0 = 1) forces a re-initialization of the communications.

A warm start (%S1 = 1) forces a re-initialization of the communications.

In Stop mode, the controller stops all Modbus communications.

---

### Modbus Link Example 1

To configure a Modbus Link, you must:

1. Configure the hardware.
2. Connect the Modbus communications cable.
3. Configure the port.
4. Write an application.
5. Initialize the Animation Table Editor.

The diagrams below illustrate the use of Modbus request code 3 to read a slave's output words. This example uses two Twido Controllers.

### Step 1: Configure the Hardware:

The hardware configuration is two Twido controllers. One will be configured as the Modbus Master and the other as the Modbus Slave.

Note: In this example, each controller is configured to use EIA RS-485 on Port 1 and an optional EIA RS-485 Port 2. On a Modular controller, the optional Port 2 can be either a TWDNOZ485D or a TWDNOZ485T, or if you use TWDXCPODM, it can be either a TWDNAC485D or a TWDNAC485T. On a Compact controller, the optional Port 2 can be either a TWDNAC485D or a TWDNAC485T.

To configure each controller, connect the TSXPCX1031 cable to Port 1 of the controller.

Note: The TSXPCX1031 can only be connected to one controller at a time, on RS-485 EIA port 1 only.

Next, connect the cable to the COM 1 port of the PC. Be sure that the cable is in switch position 2. Download and monitor the application. Repeat procedure for second controller.

### Step 2: Connect the Modbus Communications Cable:

The wiring in this example demonstrates a simple point to point connection. The three signals D1(A+), D0(B-), and COM(0V) are wired according to the diagram.

If using Port 1 of the Twido controller, the DPT signal (pin 5) must be tied to circuit common (pin 7). This conditioning of DPT determines if TwidoSoft is connected. When tied to the ground, the controller will use the port configuration set in the application to determine the type of communication.

### Step 3: Port Configuration:

In both master and slave applications, the optional EIA RS-485 ports are configured. Ensure that the controller's communication parameters are modified in Modbus protocol and at different addresses.

In this example, the master is set to an address of 1 and the slave to 2. The number of bits is set to 8, indicating that we will be using Modbus RTU mode. If this had been set to 7, then we would be using Modbus-ASCII mode. The only other default modified was to increase the response timeout to 1 second.

Note: Since Modbus RTU mode was selected, the "End of Frame" parameter was ignored.

### Step 4: Write the application:

Using TwidoSoft, an application program is written for both the master and the slave. For the slave, we simply write some memory words to a set of known values. In the master, the word table of the EXCHx instruction is initialized to read 4 words from the slave at Modbus address 2 starting at location %MW0.

Note: Notice the use of the RX offset set in %MW1 of the Modbus master. The offset of three will add a byte (value = 0) at the third position in the reception area of the table. This aligns the words in the master so that they fall correctly on word boundaries. Without this offset, each word of data would be split between two words in the exchange block. This offset is used for convenience.

Before executing the EXCH2 instruction, the application checks the communication bit associated with %MSG2. Finally, the error status of the %MSG2 is sensed and stored on the first output bit on the local base controller I/O. Additional error checking using %SW64 could also be added to make this more accurate.

Step 5: Initialize the animation table editor in the master:

After downloading and setting each controller to run, open an animation table on the master. Examine the response section of the table to check that the response code is 3 and that the correct number of bytes was read. Also in this example, note that the words read from the slave (beginning at %MW7) are aligned correctly with the word boundaries in the master.

---

## Modbus Link Example 2

The diagram below illustrates the use of Modbus request 16 to write output words to a slave. This example uses two Twido Controllers.

Step 1: Configure the Hardware:

The hardware configuration is identical to the previous example.

Step 2: Connect the Modbus Communications Cable (RS-485):

The Modbus communications cabling is identical to the previous example.

Step 3: Port Configuration:

The port configurations are identical to those in the previous example.

Step 4: Write the application:

Using TwidoSoft, an application program is created for both the master and the slave. For the slave, write a single memory word %MW18. This will allocate space on the slave for the memory addresses from %MW0 through %MW18. Without allocating the space, the Modbus request would be trying to write to locations that did not exist on the slave.

In the master, the word table of the EXCH2 instruction is initialized to read 4 bytes to the slave at Modbus address 2 at the address %MW16 (10 hexadecimal).

Note: Notice the use of the TX offset set in %MW1 of the Modbus master application. The offset of seven will suppress the high byte in the sixth word (the value 00 hexadecimal in %MW5). This works to align the data values in the transmission table

of the word table so that they fall correctly on word boundaries.

Before executing the EXCH2 instruction, the application checks the communication bit associated with %MSG2. Finally, the error status of the %MSG2 is sensed and stored on the first output bit on the local base controller I/O. Additional error checking using %SW64 could also be added to make this more accurate.

Step 5: Initialize the Animation Table Editor:

Create the following animation table on the master:

Create the following animation table on the slave:

After downloading and setting each controller to run, open an animation table on the slave controller. The two values in %MW16 and %MW17 are written to the slave. In the master, the animation table can be used to examine the reception table portion of the exchange data. This data displays the slave address, the response code, the first word written, and the number of words written

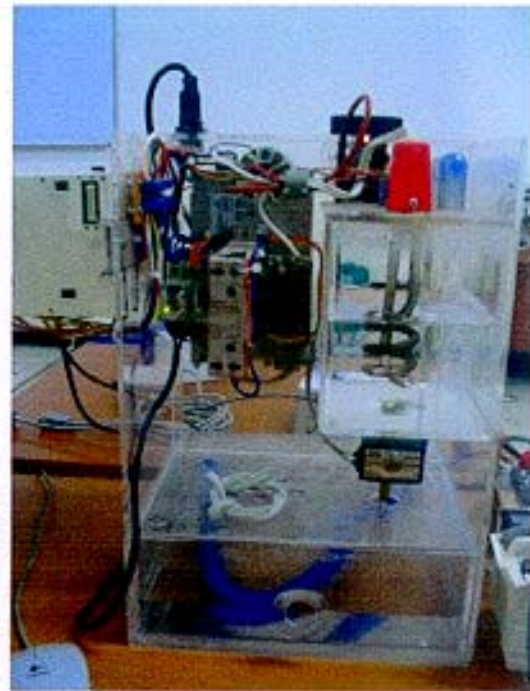


# **LAMPIRAN C**

*Foto Plant Simulator*



*Plant Level Control Simulator*



*Plant Temperature Control Simulator*



*Plant Temperature Control Simulator*