Modbus Unsolicited Serial Driver

Table of Contents

Modbus Unsolicited Serial Driver

Help version 1.045

CONTENTS

Overview

The Modbus Unsolicited Serial Driver provides a reliable way to connect Modbus serial devices to client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It simulates up to 255 Modbus slave devices on a serial communications network. Other devices or PCs can communicate with each simulated Modbus slave device using the Modbus protocol.

Note: For this driver, the terms slave and unsolicited are used interchangeably.

Setup

Supported Devices

Modbus compatible devices

Communication Protocol

Modbus RTU Protocol

Serial Communication / Port Settings
Baud Rate: 1200, 2400, 9600, 19200
Parity: Odd, Even, None
Data Bits: 8
Stop Bits: 1, 2
Flow Control: When using an RS232 / RS485 converter, the type of flow control that is required depends upon the needs of the converter. Some converters do not require any flow control and others require RTS flow. Consult the converter's documentation in order to determine its flow requirements. We recommend using an RS485 converted that provides automatic flow control.

Notes:

1. When using the manufacturer's supplied communications cable, it may be necessary to choose a flow control setting of **RTS** or **RTS Always**.
2. Not all devices support the listed configurations.

Supported Function Codes

- Read Coil Status-code 01H
- Read Input Status-code 02H
- Read Holding Registers-code 03H
- Read Internal Registers-code 04H
- Force Single Coil-code 05H
- Preset Single Register-code 06H
- Diagnostic Loopback-code 08H
- Force Multiple Coils-code 0FH
- Preset Multiple Registers-code 10H

Note: For all other function codes, the driver returns an exception code 01H (function not implemented) to the requesting device.

Broadcast Commands

The Modbus Unsolicited Serial Driver has the ability to receive broadcast write messages. Broadcast messages are defined by using a station ID of 0. When the driver receives a write message (Function 05H, 06H, 0FH, or 10H), with a station ID of 0 the value to be written is placed in all devices defined under the channel on which the command was received. Essentially the broadcast command can be used to send a single piece of data to every device that has been configured in the driver at the same time.

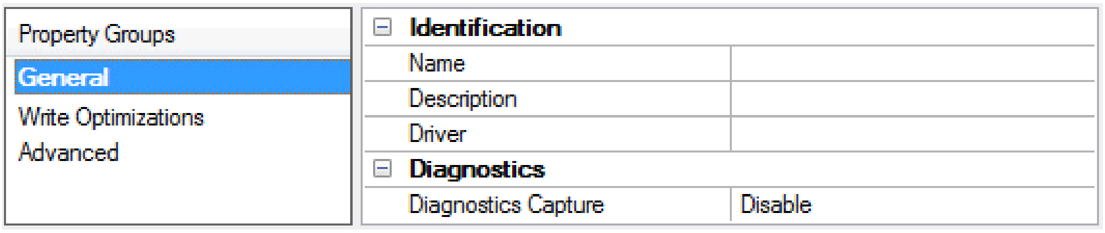
For this driver, the terms slave and unsolicited are used interchangeably.

Channel and Device Limits

The maximum number of channels supported by this driver is 100. The maximum number of devices supported by this driver is 255 per channel. This driver simulates up to 255 Modbus slave devices on a serial communications network.

Channel Properties — General

This server supports the use of simultaneous multiple communications drivers. Each protocol or driver used in a server project is called a channel. A server project may consist of many channels with the same communications driver or with unique communications drivers. A channel acts as the basic building block of an OPC link. This group is used to specify general channel properties, such as the identification attributes and operating mode.



Identification	
Name	
Description	
Driver	
Diagnostics	
Diagnostics Capture	Disable

Identification

Name: User-defined identity of this channel. In each server project, each channel name must be unique. Although names can be up to 256 characters, some client applications have a limited display window when browsing the OPC server's tag space. The channel name is part of the OPC browser information. The property is required for creating a channel.
For information on reserved characters, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in the server help.

Description: User-defined information about this channel.
Many of these properties, including Description, have an associated system tag.

Driver: Selected protocol / driver for this channel. This property specifies the device driver that was selected during channel creation. It is a disabled setting in the channel properties. The property is required for creating a channel.
Note: With the server's online full-time operation, these properties can be changed at any time. This includes changing the channel name to prevent clients from registering data with the server. If a client has already acquired an item from the server before the channel name is changed, the items are unaffected. If, after the channel name has been changed, the client application releases the item and attempts to re-acquire using the old channel name, the item is not accepted. With this in mind, changes to the properties should not be made once a large client application has been developed. Utilize the User Manager to prevent operators from changing properties and restrict access rights to server features.

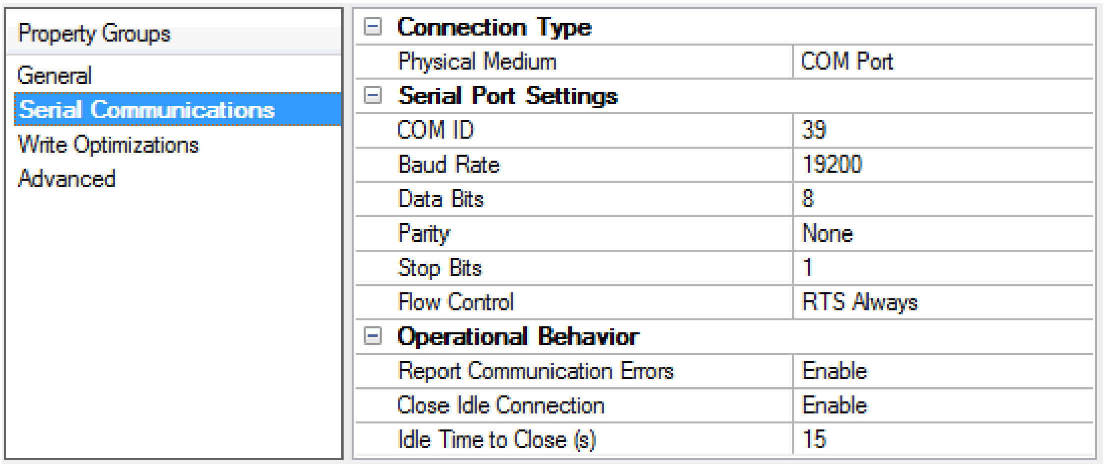
Diagnostics

Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default is disabled.
Note: This property is not available if the driver does not support diagnostics.
For more information, refer to "Communication Diagnostics" and "Statistics Tags" in the server help.

Channel Properties — Serial Communications

Serial communication properties are available to serial drivers and vary depending on the driver, connection type, and options selected. Below is a superset of the possible properties.
Click to jump to one of the sections: **Connection Type**, **Serial Port Settings** or **Ethernet Settings**, and **Operational Behavior**.

Note: With the server's online full-time operation, these properties can be changed at any time. Utilize the User Manager to restrict access rights to server features, as changes made to these properties can temporarily disrupt communications.



Property Groups		
General	⊟ Connection Type	
Serial Communications	Physical Medium	COM Port
Write Optimizations	⊟ Serial Port Settings	
Advanced	COM ID	39
	Baud Rate	19200
	Data Bits	8
	Parity	None
	Stop Bits	1
	Flow Control	RTS Always
	⊟ Operational Behavior	
	Report Communication Errors	Enable
	Close Idle Connection	Enable
	Idle Time to Close (s)	15

Connection Type

Physical Medium: Choose the type of hardware device for data communications. Options include COM Port, None, Modem, and Ethernet Encapsulation. The default is COM Port.

- **None**: Select None to indicate there is no physical connection, which displays the **Operation with no Communications** section.
- **COM Port**: Select Com Port to display and configure the **Serial Port Settings** section.
- **Modem**: Select Modem if phone lines are used for communications, which are configured in the **Modem Settings** section.
- **Ethernet Encap.**: Select if Ethernet Encapsulation is used for communications, which displays the **Ethernet Settings** section.
- **Shared**: Verify the connection is correctly identified as sharing the current configuration with another channel. This is a read-only property.

Serial Port Settings

COM ID: Specify the Communications ID to be used when communicating with devices assigned to the channel. The valid range is 1 to 9991 to 16. The default is 1.

Baud Rate: Specify the baud rate to be used to configure the selected communications port.

Data Bits: Specify the number of data bits per data word. Options include 5, 6, 7, or 8.

Parity: Specify the type of parity for the data. Options include Odd, Even, or None.

Stop Bits: Specify the number of stop bits per data word. Options include 1 or 2.

Flow Control: Select how the RTS and DTR control lines are utilized. Flow control is required to communicate with some serial devices. Options are:

- **None**: This option does not toggle or assert control lines.
- **DTR**: This option asserts the DTR line when the communications port is opened and remains on.
- **RTS**: This option specifies that the RTS line is high if bytes are available for transmission. After all buffered bytes have been sent, the RTS line is low. This is normally used with RS232/RS485 converter hardware.

- **RTS, DTR**: This option is a combination of DTR and RTS.
- **RTS Always**: This option asserts the RTS line when the communication port is opened and remains on.
- **RTS Manual**: This option asserts the RTS line based on the timing properties entered for RTS Line Control. It is only available when the driver supports manual RTS line control (or when the properties are shared and at least one of the channels belongs to a driver that provides this support).
  RTS Manual adds an **RTS Line Control** property with options as follows:
  - **Raise**: This property specifies the amount of time that the RTS line is raised prior to data transmission. The valid range is 0 to 9999 milliseconds. The default is 10 milliseconds.
  - **Drop**: This property specifies the amount of time that the RTS line remains high after data transmission. The valid range is 0 to 9999 milliseconds. The default is 10 milliseconds.
  - **Poll Delay**: This property specifies the amount of time that polling for communications is delayed. The valid range is 0 to 9999. The default is 10 milliseconds.

Tip: When using two-wire RS-485, "echoes" may occur on the communication lines. Since this communication does not support echo suppression, it is recommended that echoes be disabled or a RS-485 converter be used.

Operational Behavior

- **Report Communication Errors**: Enable or disable reporting of low-level communications errors. When enabled, low-level errors are posted to the Event Log as they occur. When disabled, these same errors are not posted even though normal request failures are. The default is Enable.
- **Close Idle Connection**: Choose to close the connection when there are no longer any tags being referenced by a client on the channel. The default is Enable.
- **Idle Time to Close**: Specify the amount of time that the server waits once all tags have been removed before closing the COM port. The default is 15 seconds.

Ethernet Settings

Note: Not all serial drivers support Ethernet Encapsulation. If this group does not appear, the functionality is not supported.

Ethernet Encapsulation provides communication with serial devices connected to terminal servers on the Ethernet network. A terminal server is essentially a virtual serial port that converts TCP/IP messages on the Ethernet network to serial data. Once the message has been converted, users can connect standard devices that support serial communications to the terminal server. The terminal server's serial port must be properly configured to match the requirements of the serial device to which it is attached. *For more information, refer to "Using Ethernet Encapsulation" in the server help.*

- **Network Adapter**: Indicate a network adapter to bind for Ethernet devices in this channel. Choose a network adapter to bind to or allow the OS to select the default.
  Specific drivers may display additional Ethernet Encapsulation properties. For more information, refer to Channel Properties — Ethernet Encapsulation.

Modem Settings

- **Modem**: Specify the installed modem to be used for communications.
- **Connect Timeout**: Specify the amount of time to wait for connections to be established before failing a read or write. The default is 60 seconds.
- **Modem Properties**: Configure the modem hardware. When clicked, it opens vendor-specific modem properties.

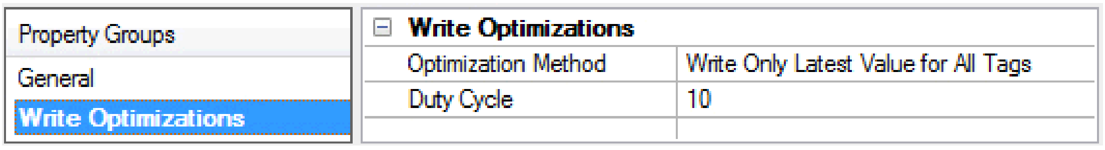
- **Auto-Dial**: Enables the automatic dialing of entries in the Phonebook. The default is Disable. *For more information, refer to "Modem Auto-Dial" in the server help.*
- **Report Communication Errors**: Enable or disable reporting of low-level communications errors. When enabled, low-level errors are posted to the Event Log as they occur. When disabled, these same errors are not posted even though normal request failures are. The default is Enable.
- **Close Idle Connection**: Choose to close the modem connection when there are no longer any tags being referenced by a client on the channel. The default is Enable.
- **Idle Time to Close**: Specify the amount of time that the server waits once all tags have been removed before closing the modem connection. The default is 15 seconds.

Operation with no Communications

- **Read Processing**: Select the action to be taken when an explicit device read is requested. Options include Ignore and Fail. Ignore does nothing; Fail provides the client with an update that indicates failure. The default setting is Ignore.

Channel Properties — Write Optimizations

As with any server, writing data to the device may be the application's most important aspect. The server intends to ensure that the data written from the client application gets to the device on time. Given this goal, the server provides optimization properties that can be used to meet specific needs or improve application responsiveness.


Property Groups
General
Write Optimizations

Write Optimizations	
Optimization Method	Write Only Latest Value for All Tags
Duty Cycle	10

Write Optimizations

Optimization Method: Controls how write data is passed to the underlying communications driver. The options are:

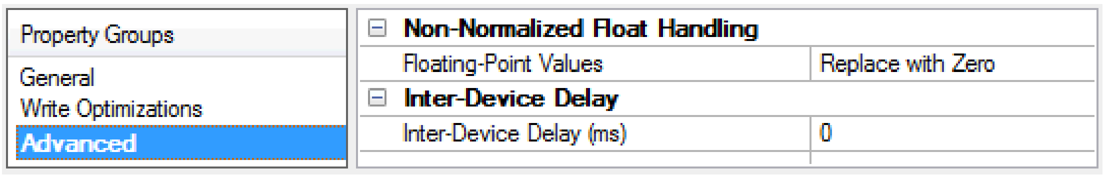
- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing data to the device as quickly as possible. This mode ensures that everything written from the client applications is sent to the target device. This mode should be selected if the write operation order or the write item's content must uniquely be seen at the target device.
- **Write Only Latest Value for Non-Boolean Tags**: Many consecutive writes to the same value can accumulate in the write queue due to the time required to actually send the data to the device. If the server updates a write value that has already been placed in the write queue, far fewer writes are needed to reach the same final output value. In this way, no extra writes accumulate in the server's queue. When the user stops moving the slide switch, the value in the device is at the correct value at virtually the same time. As the mode states, any value that is not a Boolean value is updated in the server's internal write queue and sent to the device at the next possible opportunity. This can greatly improve the application performance.
  Note: This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.

- **Write Only Latest Value for All Tags**: This option takes the theory behind the second optimization mode and applies it to all tags. It is especially useful if the application only needs to send the latest value to the device. This mode optimizes all writes by updating the tags currently in the write queue before they are sent. This is the default mode.

Duty Cycle: is used to control the ratio of write to read operations. The ratio is always based on one read for every one to ten writes. The duty cycle is set to ten by default, meaning that ten writes occur for each read operation. Although the application is performing a large number of continuous writes, it must be ensured that read data is still given time to process. A setting of one results in one read operation for every write operation. If there are no write operations to perform, reads are processed continuously. This allows optimization for applications with continuous writes versus a more balanced back and forth data flow.
Note: It is recommended that the application be characterized for compatibility with the write optimization enhancements before being used in a production environment.

Channel Properties — Advanced

This group is used to specify advanced channel properties. Not all drivers support all properties; so the Advanced group does not appear for those devices.



Non-Normalized Float Handling	
Floating-Point Values	Replace with Zero
Inter-Device Delay	
Inter-Device Delay (ms)	0

Non-Normalized Float Handling: A non-normalized value is defined as Infinity, Not-a-Number (NaN), or as a Denormalized Number. The default is Replace with Zero. Drivers that have native float handling may default to Unmodified. Non-normalized float handling allows users to specify how a driver handles non-normalized IEEE-754 floating point data. Descriptions of the options are as follows:

- **Replace with Zero**: This option allows a driver to replace non-normalized IEEE-754 floating point values with zero before being transferred to clients.
- **Unmodified**: This option allows a driver to transfer IEEE-754 denormalized, normalized, non-number, and infinity values to clients without any conversion or changes.

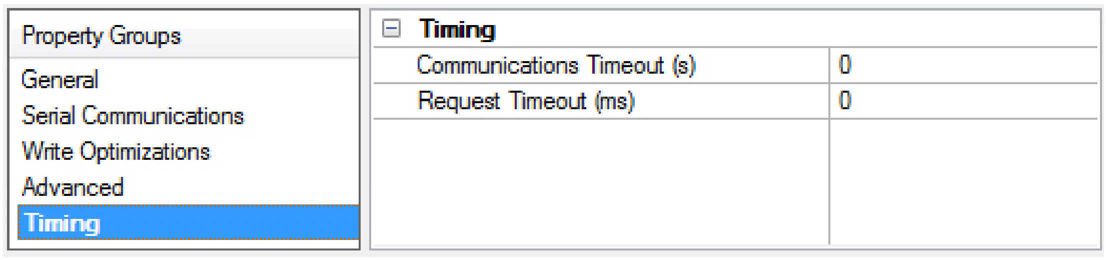
Note: This property is not available if the driver does not support floating point values or if it only supports the option that is displayed. According to the channel's float normalization setting, only real-time driver tags (such as values and arrays) are subject to float normalization. For example, EFM data is not affected by this setting.

For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

Inter-Device Delay: Specify the amount of time the communications channel waits to send new requests to the next device after data is received from the current device on the same channel. Zero (0) disables the delay.

Note: This property is not available for all drivers, models, and dependent settings.

Channel Properties — Timing



Timing	
Communications Timeout (s)	0
Request Timeout (ms)	0

Communications Timeout: Specify the amount of time that the driver waits for an incoming request before setting all unsolicited device tags on the channel to a Bad quality. After the Communications Timeout passes, the only way to reset the timeout and allow all tags be processed normally is to reestablish communications with the device or disable the timeout by setting Communications Timeout to 0 (zero) in the Timing group of channel properties. Disabled: 0; Enabled: 1-->64,800 seconds (18 hours).

Request Timeout: Specify the amount of time that the driver waits for a complete request frame to be received. The elapsed time is calculated starting from the instant the first byte of a new request is received. If a complete request frame is not received during this time, the driver flushes received data buffers and assume the next received byte is the start of a new request.

Tips:

This setting should be chosen carefully. Values for the Request Timeout setting may range from 0 to 30,000 ms, with a default of 0. When 0 is entered, the driver computes a reasonable timeout through the use of the following formula:

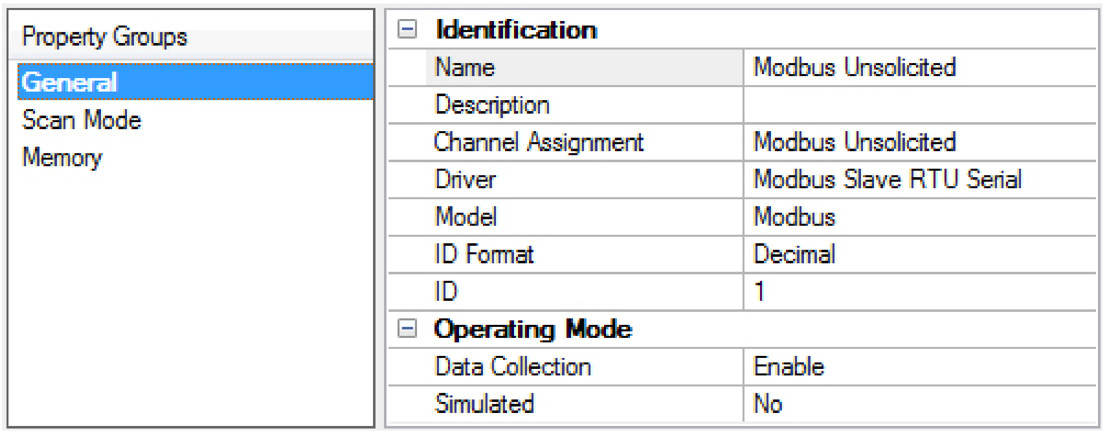
$T_{default}$ = 1000* (Bits per Byte)* 512* 3 / Baud

This is three times the amount of time required to transmit a frame of 512 bytes. The number of bits per byte includes the start bit and the number of data and stop bits specified. For example, a baud rate of 9600 and 8 data bits, and 1 stop bit, results in a default timeout of 1600 ms. If the hardware sends relatively short request frames and would retry a failed request in less than the default calculation (1600 ms in this example), try configuring a shorter Request Timeout.

The Request Timeout should never be shorter than the amount of time it takes to receive the longest request frame sent by any device on the channel. This can be computed using the following formula:

T_{min} = 1000* (Bits per Byte)* (max frame length) / Baud.

Device Properties — General



Identification	
Name	Modbus Unsolicited
Description	
Channel Assignment	Modbus Unsolicited
Driver	Modbus Slave RTU Serial
Model	Modbus
ID Format	Decimal
ID	1
Operating Mode	
Data Collection	Enable
Simulated	No

Identification

Name: User-defined identity of this device.

Description: User-defined information about this device.

Channel Assignment: User-defined name of the channel to which this device currently belongs.

Driver: Selected protocol driver for this device.
For more information on a specific device model, see Supported Devices.

Model: The specific version of the device.

ID Format : Select how the device identity is formatted. Options include Decimal, Octal, and Hex.

ID: Modbus serial devices are assigned device IDs in the range 0 to 255.

Operating Mode

Data Collection: This property controls the device's active state. Although device communications are enabled by default, this property can be used to disable a physical device. Communications are not attempted when a device is disabled. From a client standpoint, the data is marked as invalid and write operations are not accepted. This property can be changed at any time through this property or the device system tags.

Simulated: This option places the device into Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client as valid data. While in Simulation Mode, the server treats all device data as reflective: whatever is written to the simulated device is read back and each OPC item is treated individually. The item's memory map is based on the group Update Rate. The data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

Notes:

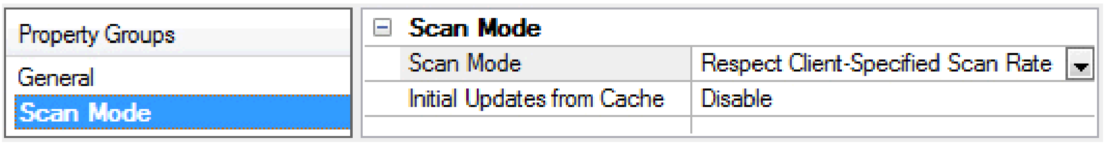
1. This System tag (_Simulated) is read only and cannot be written to for runtime protection. The System tag allows this property to be monitored from the client.

2. In Simulation mode, the item's memory map is based on client update rate(s) (Group Update Rate for OPC clients or Scan Rate for native and DDE interfaces). This means that two clients that reference the same item with different update rates return different data.

Simulation Mode is for test and simulation purposes only. It should never be used in a production environment.

Device Properties — Scan Mode

The Scan Mode specifies the subscribed-client requested scan rate for tags that require device communications. Synchronous and asynchronous device reads and writes are processed as soon as possible; unaffected by the Scan Mode properties.



Property Groups
General
Scan Mode

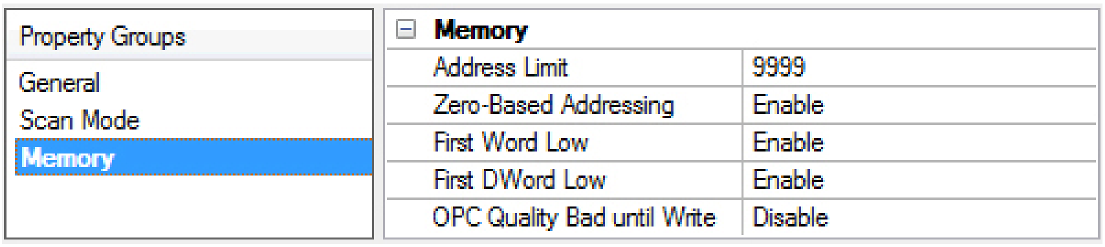
Scan Mode	
Scan Mode	Respect Client-Specified Scan Rate
Initial Updates from Cache	Disable

Scan Mode: Specifies how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  Note: When the server has an active client and items for the device and the scan rate value is increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.
- **Request All Data at Scan Rate**: This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only**: This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the client's responsibility to poll for updates, either by writing to the _DemandPoll tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help*.
- **Respect Tag-Specified Scan Rate**: This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

Initial Updates from Cache: When enabled, this option allows the server to provide the first updates for newly activated tag references from stored (cached) data. Cache updates can only be provided when the new item reference shares the same address, scan rate, data type, client access, and scaling properties. A device read is used for the initial update for the first client reference only. The default is disabled; any time a client activates a tag reference the server attempts to read the initial value from the device.

Device Properties — Memory



Memory	
Address Limit	9999
Zero-Based Addressing	Enable
First Word Low	Enable
First DWord Low	Enable
OPC Quality Bad until Write	Disable

Memory

Address Limit: The address range of coils and registers can be configured with any value between 9999 and 65536. The tags can be addressed up to and including the specified limit.
Notes:

1. The address range cannot be changed when the tags are being processed.
2. When the address range is changed, it is possible that a remote request (from a Modbus master) may get rejected because the requested memory address is outside the new address range.
3. When the address range is changed and the new upper limit is greater than the old one, all old data is preserved and the remaining memory is initialized to '0'. If the new upper limit is smaller then the old one, however, only the data equal to the new memory size is preserved and the remaining data is lost. There may be an exception to this when dealing with Boolean memory type.

*See **Memory Addressing** for details, examples, and diagrams.*

Zero-Based Addressing: By default, addresses have one subtracted when frames are constructed to communicate with a Modbus device. If the device doesn't follow this convention, disable Zero-Based Addressing. The default (enabled) behavior follows the convention of the Modicon PLCs.
Note: Zero-Based Addressing must be disabled when using a Daniels/Enron device.

First Word Low: Two consecutive register addresses in a Modbus device are used for 32-bit data types. Enable if the driver should assume the first word is the low word of the 32-bit value. If First Word Low is enabled (the default), first word low is assumed, which follows the convention of the Modicon Modsoft programming software.
Note: First Word Low must be disabled when using a Daniels/Enron device.

First DWord Low: Four consecutive register addresses (in two groups of two each) are used for 64-bit data types. Users can specify whether the driver should assume the first pair (i.e., first DWord) is the low or the high DWord of the 64-bit value. If First DWord Low is enabled, the first DWord low is assumed; if disabled, the second DWord low is assumed.
Note: First DWord Low must be disabled when using the Daniels/Enron Device.

OPC Quality Bad until Write: This option controls the initial OPC quality of tags attached to this driver. When disabled, all tags have an initial value of 0 and good OPC quality. This is the default condition. When enabled, all tags have an initial value of 0 and Bad OPC quality. The quality of a tag remains Bad until all coils or registers referenced by the tag have been written to by a Modbus master or a client application. For example, a tag with address 400001 and data type DWord references two holding registers: 400001 and 400002. This tag does not show Good quality until both holding registers had been written.

Memory Addressing

Accessible Memory Locations

- Output coils-00001 to 065536
- Input coils-10001 to 165536
- Internal registers-30001 to 365536
- Holding registers-40001 to 465536

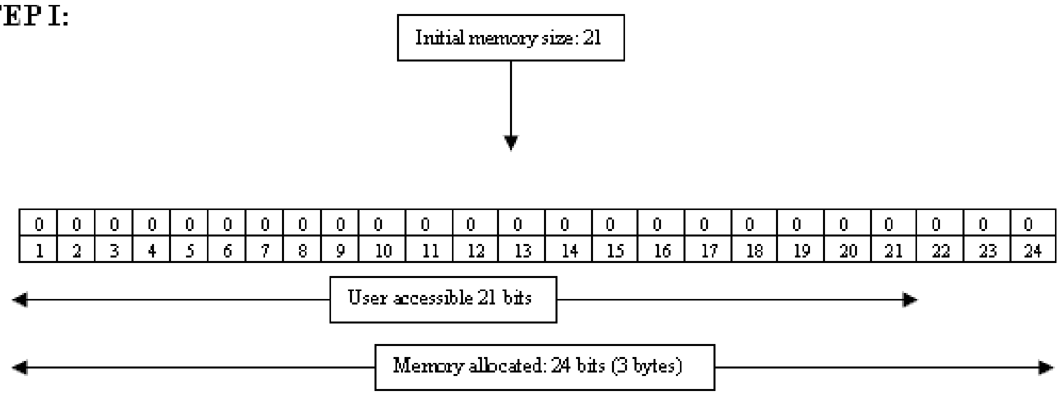
*These settings are configurable. For more information, refer to **Memory**.*

The address range of coils and registers can be configured with any value between 9999 and 65536. The tags can be addressed up to and including the specified limit.

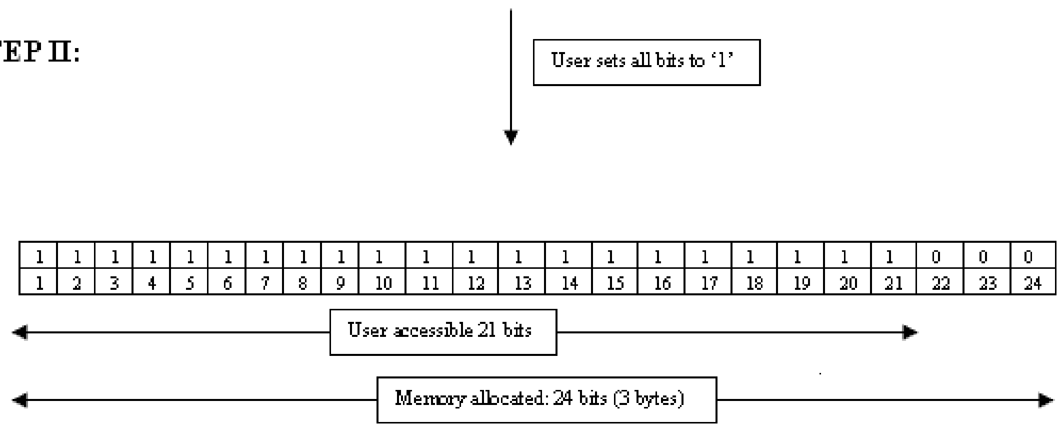
For example, if the initial memory size is 21; this translates into 3 bytes (byte aligned) of memory for Boolean types. If the memory size is changed to 12 (2 bytes), the smaller of the two memory sizes is 2 bytes and that is the amount of data that is preserved. Although users may think only 12 bits have been preserved, 16 bits (2 bytes) have been preserved. Normally this would not be noticed, because with a memory size of 12, memory can be accessed up to index 12 only. If the memory size is increased to 22 (3 bytes) the amount of data preserved from the old memory is 2 bytes (smaller of the two). Even though 12 bits were manipulated earlier (with a memory size of 12), old data for bits 13-16 (which may have been initialized to some values the first time around when the memory size was 21) is preserved.

The images below apply a coil memory type to the example above for a diagrammatic explanation.

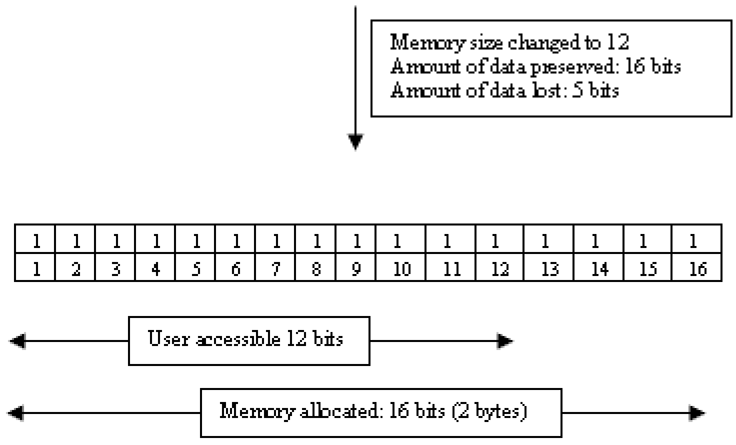
STEP I:



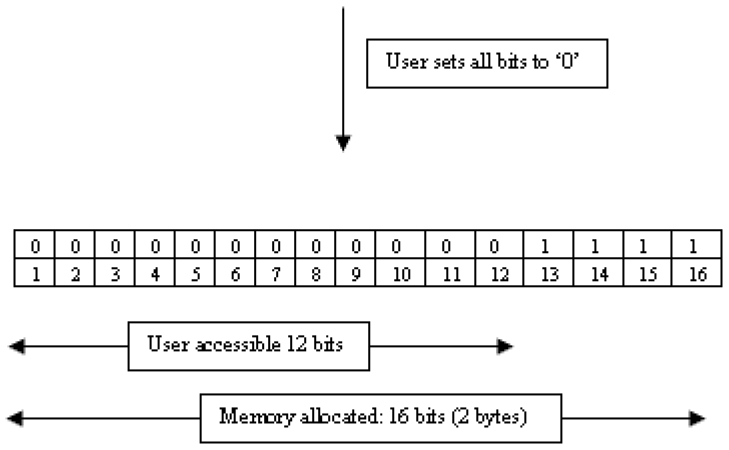
STEP II:



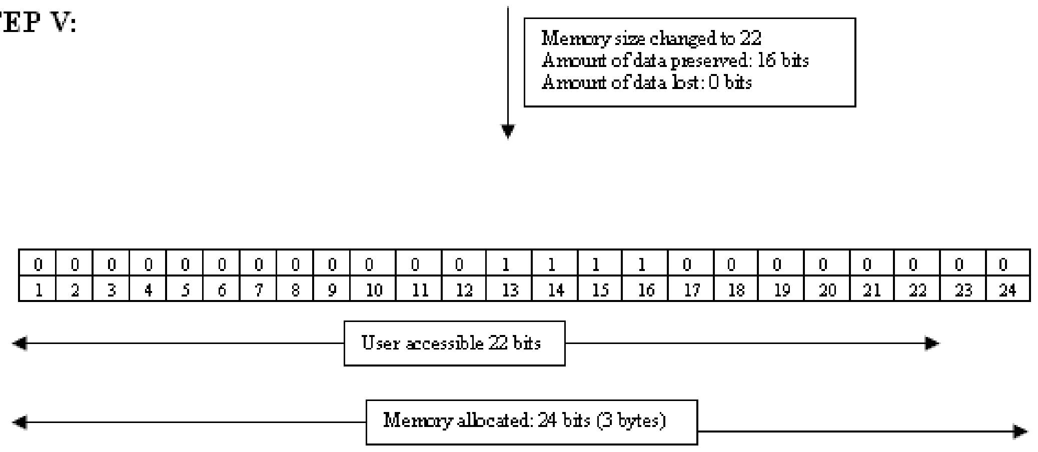
STEP III:


Memory size changed to 12
Amount of data preserved: 16 bits
Amount of data lost: 5 bits
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
User accessible 12 bits
Memory allocated: 16 bits (2 bytes)

STEP IV:



STEP V:



Step V above shows old data for bits 13-16 that were initialized in Step II. Users may expect all the bits in Step V to be '0,' but bits 13-16 have been carried over from Step II and are still set to '1'.

Data Types Description

Data Type	Description
Boolean	Single bit
Word	Unsigned 16-bit value bit 0 is the low bit bit 15 is the high bit
Short	Signed 16-bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
DWord	Unsigned 32-bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
BCD	Two byte packed BCD Value range is 0-9999. Behavior is undefined for values beyond this range.
LBCD	Four byte packed BCD Value range is 0-99999999. Behavior is undefined for values beyond this range.
String	Null terminated ASCII string Supported within the Holding Register Range, includes HiLo LoHi byte order selection.
Double*	64-bit floating point value The driver interprets four consecutive registers as a double precision value by making the last two registers the high DWord and the first two registers the low DWord.
Double Example	If register 40001 is specified as a double, bit 0 of register 40001 would be bit 0 of the 64-bit data type and bit 15 of register 40004 would be bit 63 of the 64-bit data type.
Float*	32-bit floating point value The driver interprets two consecutive registers as a single precision value by making the last register the high word and the first register the low word.
Float Example	If register 40001 is specified as a float, bit 0 of register 40001 would be bit 0 of the 32-bit data type and bit 15 of register 40002 would be bit 31 of the 32-bit data type.

* The descriptions assume the default-first DWord low data handling of 64-bit data types, and first word low data handling of 32-bit data types.

Address Descriptions

Address specifications vary depending on the model in use. Select a link from the following list to obtain specific address information for the model of interest.

Modbus Addressing
Daniels/Enron Addressing

Modbus Addressing

5-Digit Addressing vs. 6-Digit Addressing

In Modbus addressing, the first digit of the address specifies the primary table. The remaining digits represent the device's data item. The maximum value is a two byte unsigned integer (65,535). Six digits are required to represent the entire address table and item. As such, addresses that are specified in the device's manual as 0xxxx, 1xxxx, 3xxxx, or 4xxxx are padded with an extra zero once applied to the Address field of a Modbus tag.

Primary Table	Description
0	Output Coils
1	Input Coils
3	Internal Registers
4	Holding Registers

Modbus Addressing

The following address descriptions apply to the client application's access to each simulated Modbus slave device. The client application controls the memory of the simulated Modbus slave device; therefore, all areas have Read/Write access. The default data types for dynamically defined tags are shown in **bold**.

Address	Range*	Data Type	Access
Output Coils	000001-065536	**Boolean**	Read/Write
Input Coils	100001-165536	**Boolean**	Read/Write
Internal Registers	300001-365536 300001-365535 3xxxxx.0-3xxxxx.15	**Word**, Short, BCD Float, DWord, Long, LBCD Boolean, Double	Read/Write
Internal Registers As String with HiLo Byte Order**	300001.2H-365536.240H .Bit is string length, range 2 to 240 bytes.	**String**	Read Only
Internal Registers As String with LoHi Byte Order **	300001.2L-365536.240L .Bit is string length, range 2 to 240 bytes.	**String**	Read Only
Holding Registers	400001-465536 400001-465535 4xxxxx.0-4xxxxx.15	**Word**, Short, BCD Float, DWord, Long, LBCD Boolean, Double	Read/Write

Address	Range*	Data Type	Access
Holding Registers As String with HiLo Byte Order	400001.2H-465536.240H .Bit is string length, range 2 to 240 bytes.	**String**	Read/Write
Holding Registers As String with LoHi Byte Order	400001.2L-465536.240L .Bit is string length, range 2 to 240 bytes.	**String**	Read/Write

* The maximum range is determined by the value set in the Memory device property. For more information, refer to **Memory**.
** This address supports Function Code 04, and only applies to decimal addressing.

Array Support

Arrays are supported for internal and holding register locations for all data types except for Boolean. Arrays are also supported for input and output coils (Boolean data types). There are two methods of addressing an array. Examples are given using holding register locations.

4xxxx [rows] [cols]

4xxxx [cols] this method assumes rows is equal to one
For Word, Short and BCD arrays, the base address + (rows * cols) cannot exceed 65536.
For Float, DWord, Long and Long BCD arrays, the base address + (rows * cols * 2) cannot exceed 65535.

String Support

The Modbus model supports reading and writing holding register memory as an ASCII string. When using holding registers for string data, each register contains two bytes of ASCII data. The order of the ASCII data within a given register can be selected when the string is defined. The length of the string can be from 2 to 240 bytes and is entered in place of a bit number. The length must be entered as an even number. The byte order is specified by appending either a "H" or "L" to the address.

String Examples

- To address a string starting at 400200 with a length of 100 bytes and HiLo byte order, enter: 400200.100H
- To address a string starting at 400500 with a length of 78 bytes and LoHi byte order, enter: 400500.78L

Note: For this driver, the terms Slave and Unsolicited are used interchangeably.

Daniels/Enron Addressing

The following address descriptions apply to the client application's access to each simulated Daniels/Enron slave device. The client application controls the memory of the simulated slave device; therefore, all areas have Read/Write access.

The default data types for dynamically defined tags are shown in **bold** where appropriate. The following table assumes that the slave device has been configured for the maximum allowed address range of 0 to 65535. For more information, refer to **Memory**.

Address	Range*	Data Type	Access
Output Coils	000000-065535	**Boolean**	Read/Write
Input Coils	100000-165535	**Boolean**	Read/Write
Internal Registers	300000-365535 300000-365534 300000-365532 300000.0-365535.15	**Word**, Short, BCD Float, DWord, Long, LBCD Double Boolean	Read/Write
Holding Registers	400000-405000 406000-407000 408000-465535 400000-404999 406000-406999 408000-465534 400000-404999 405001-405999 406000-406999 408000-465534 400000-404999 406000-406999 407001-407999 408000-465534 400000-404997 406000-406997 408000-465532	**Word**, Short, BCD DWord, LBCD Long Float Double	Read/Write
Holding Registers as Booleans	400000.xx-405000.xx 405001.yy-405999.yy 406000.xx-465535.xx xx is the bit number from 0-15 yy is the bit number from 0-31	**Boolean**	Read/Write
Holding Registers as String with HiLo Byte Order	400000.xxxH-405000.xxxH 406000.xxxH-407000.xxxH 408000.xxxH-465535.xxxH xxx is string length, range 2 to 240 bytes.	**String**	Read/Write
Holding Registers as String with LoHi Byte Order	400000.xxxL-405000.xxxL 406000.xxxL-407000.xxxL	**String**	Read/Write

Address	Range*	Data Type	Access
	408000.xxxL-465535.xxxL xxx is string length, range 2 to 240 bytes.		

* The maximum range is determined by the value in the Memory device property. For more information, refer to **Memory**.

Array Support

Arrays are supported for internal and holding register locations for all data types except for Boolean. Arrays are also supported for input and output coils (Boolean data types). There are two methods of addressing an array. Examples are given using holding register locations.

4xxxx [rows] [cols]

4xxxx [cols] this method assumes rows is equal to one
For Word, Short and BCD arrays, the base address + (rows * cols) cannot exceed 65535.
For Float, DWord, Long and Long BCD arrays, the base address + (rows * cols * 2) cannot exceed 65534.

String Support

The Modbus model supports reading and writing holding register memory as an ASCII string. When using holding registers for string data, each register contains two bytes of ASCII data. The order of the ASCII data within a given register can be selected when the string is defined. The length of the string can be from 2 to 240 bytes and is entered in place of a bit number. The length must be entered as an even number. The byte order is specified by appending either a "H" or "L" to the address.

String Examples

- To address a string starting at 400200 with a length of 100 bytes and HiLo byte order, enter: 400200.100H
- To address a string starting at 400500 with a length of 78 bytes and LoHi byte order, enter: 400500.78L

Note: For this driver, the terms Slave and Unsolicited are used interchangeably.

Event Log Messages

The following information concerns messages posted to the Event Log pane in the main user interface. Consult the server help on filtering and sorting the Event Log detail view. Server help contains many common messages, so should also be searched. Generally, the type of message (informational, warning) and troubleshooting information is provided whenever possible.

Address size changed. | Previous Size = <number>, Current Size = <number>.

Error Type:

Error

Possible Cause:

The address size for the specified device has been changed.

Possible Solution:

Verify the new address size is correct.

Error Mask Definitions

B = Hardware break detected
F = Framing error
E = I/O error
O = Character buffer overrun
R = RX buffer overrun
P = Received byte parity error
T = TX buffer full

Modbus Exception Codes

The following data is from Modbus Application Protocol Specifications documentation.

Code Dec/Hex	Name	Description
01/0x01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the server (or slave). This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type. For example, because it is unconfigured and is being asked to return register values.
02/0x02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 would succeed, a request with offset 96 and length 5 generates exception 02.
03/0x03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does not mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the Modbus protocol is unaware of the significance of any particular value of any particular register.
04/0x04	SLAVE DEVICE FAILURE	An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.
05/0x05	ACKNOWLEDGE	The slave has accepted the request and is processing it, but a long duration of time is required to do so. This response is returned to prevent a timeout error from occurring in the master. The master can next issue a Poll Program Complete message to determine if processing is completed.
06/0x06	SLAVE DEVICE BUSY	The slave is engaged in processing a long-duration program command. The master should retransmit the message later when the slave is free.
07/0x07	NEGATIVE ACKNOWLEDGE	The slave cannot perform the program function received in the query. This code is returned for an unsuccessful programming request using function code 13 or 14 decimal. The master should request diagnostic or error information from the slave.
08/0x08	MEMORY PARITY ERROR	The slave attempted to read extended memory, but detected a parity error in the memory. The master can retry the request, but service may be required on the slave device.
10/0x0A	GATEWAY PATH UNAVAILABLE	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. This usually means that the gateway is misconfigured or overloaded.
11/0x0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network.

Note: For this driver, the terms slave and unsolicited are used interchangeably.

Index