Toyopuc Serial Driver

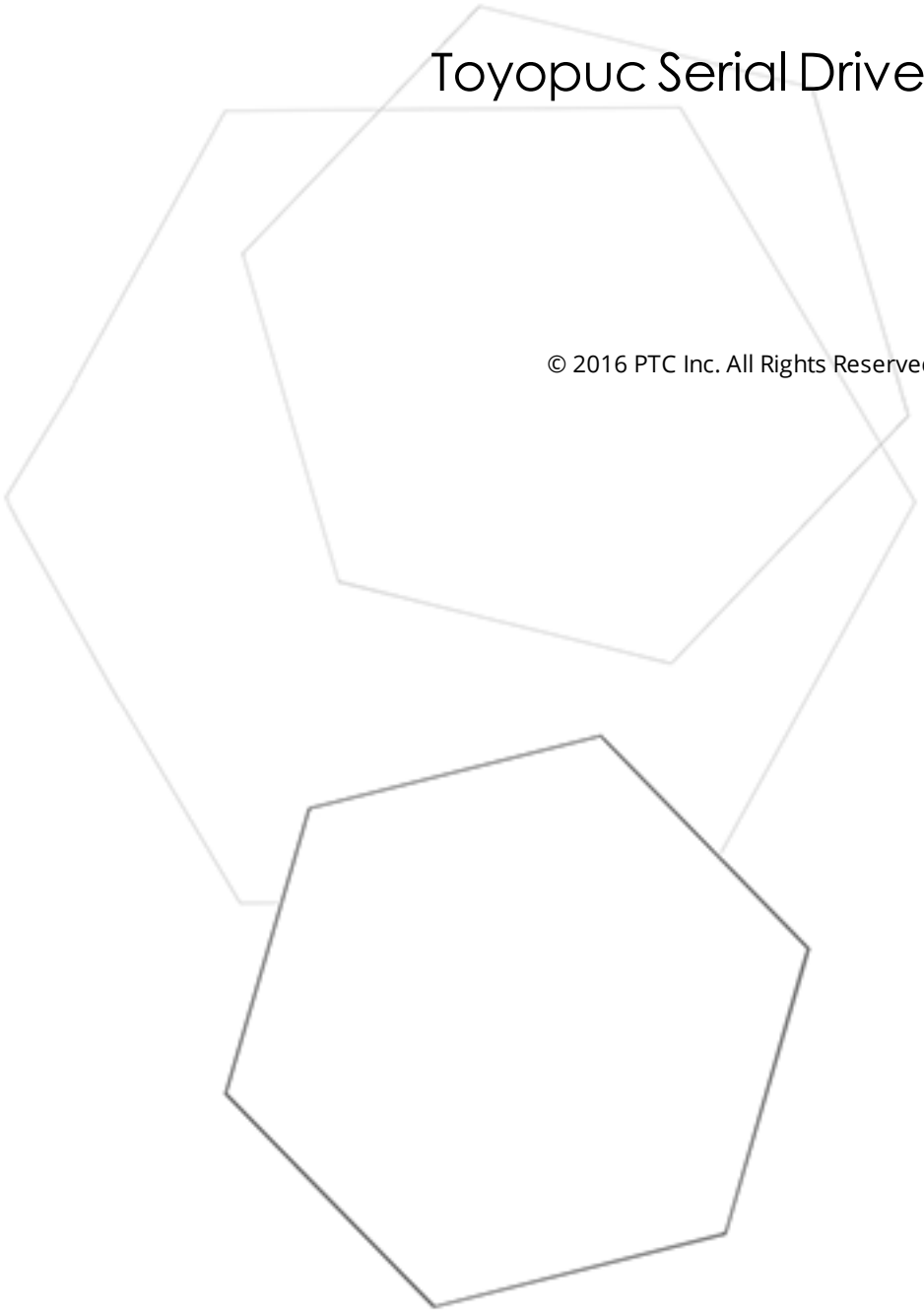


Table of Contents

Toyopuc Serial Driver ... 1
Table of Contents ... 2
Toyopuc Serial Driver ... 3
Overview ... 3
Setup ... 4
Channel Properties - General ... 4
Channel Properties - Serial Communications ... 5
Channel Properties - Write Optimizations ... 8
Channel Properties - Advanced ... 9
Device Properties - General ... 9
Device Properties - Scan Mode ... 11
Device Properties - Timing ... 12
Device Properties - Auto-Demotion ... 13
Device Properties - Redundancy ... 13
Modem Setup ... 13
Diagnostics Tags ... 14
Multi-Point Read Support ... 15
Data Types Description ... 17
Address Descriptions ... 18
Error Descriptions ... 20
Missing address ... 20
Device address '<address>' contains a syntax error ... 20
Address '<address>' is out of range for the specified device or register ... 21
Device address '<address>' is not supported by model '<model name>' ... 21
Data Type '<type>' is not valid for device address '<address>' ... 21
Device address '<address>' is Read Only ... 21
Array size is out of range for address '<address>' ... 21
Array support is not available for the specified address: '<address>' ... 22
COMn does not exist ... 22
Error opening COMn ... 22
COMn is in use by another application ... 22
Unable to set comm properties on COMn ... 23
Communications error on '<channel name>' [<error mask>] ... 23
Device '<device name>' is not responding ... 23
Unable to write to '<address>' on device '<device name>' ... 24
Bad address in block [<start address> to <end address>] on device '<device name>' ... 24
Index ... 25

Toyopuc Serial Driver

Help version 1.015

CONTENTS

Overview
What is the Toyopuc Serial Driver?

Device Setup
How do I configure a device for use with this driver?

Data Types Descriptions
What data types does this driver support?

Address Descriptions
How do I address a data location on a Toyopuc PC2 Serial device?

Error Descriptions
What error messages does the Toyopuc Serial Driver produce?

Overview

The Toyopuc Serial Driver provides a reliable way to connect Toyopuc PC2 Serial devices to OPC client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with any Toyoda Toyopuc PLC that supports Computer Link communications. The Toyopuc Serial Driver supports extensive diagnostics tags and the Toyopuc PC2's Multi Point Read features. For more information, refer to **Diagnostics Tags** and **Multi Point Read Support**.

Setup

Supported Devices

PC2 Series or any Computer Link compatible device

Communication Protocol

Computer Link

Supported Communication Properties

Baud Rate: 300, 600, 1200, 2400, 9600, 19200
Parity: None, Even, or Odd
Data Bits: 5, 6, 7 or 8
Stop Bits: 1 or 2

Note: Not all devices support the listed configurations.

Device IDs (Computer Link Station Number)

The Device ID range is 0 to 37 Octal.

Flow Control

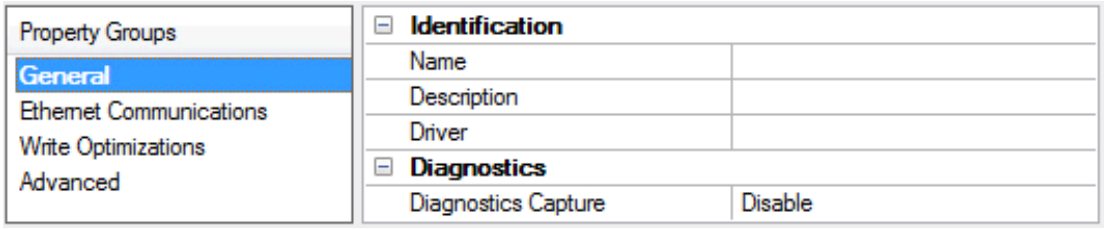
When using an RS232/RS485 converter, the type of flow control that is required depends on the needs of the converter. Some converters do not require any flow control whereas others require RTS flow. To determine the converter's flow requirements, refer to its documentation. An RS485 converter that provides automatic flow control is recommended.

Ethernet Encapsulation

This driver supports Ethernet Encapsulation, which allows the driver to communicate with serial devices attached to an Ethernet network using a terminal server or device server. It may be invoked through the COM ID property group in Channel Properties. For more information, refer to the OPC Server's help documentation.

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: 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.

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.

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.

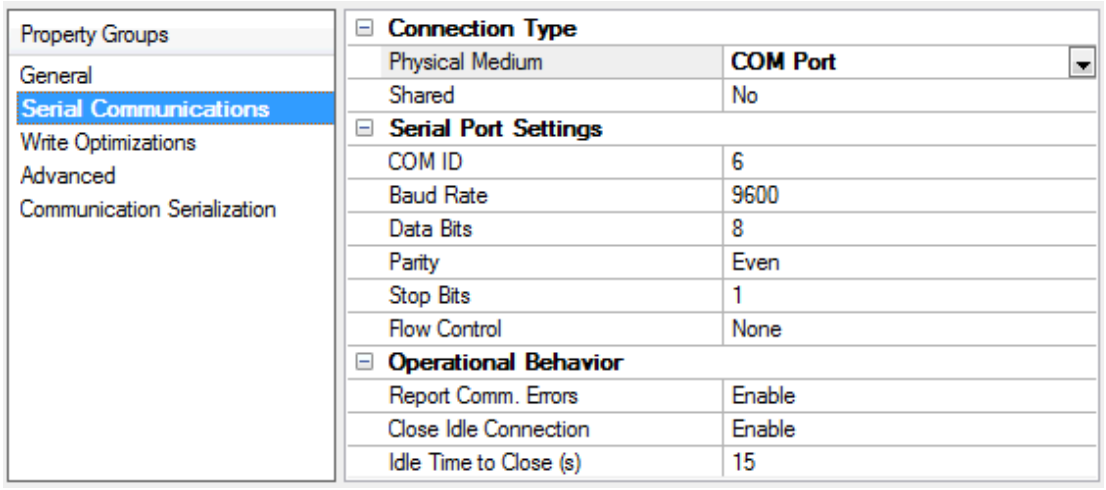
Diagnostics

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.
For more information, refer to "Communication Diagnostics" in the server help.
Not all drivers support diagnostics. To determine whether diagnostics are available for a particular driver, open the driver information and locate the "Supports device level diagnostics" statement.

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.



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 Comm. 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

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 "How To... Use 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

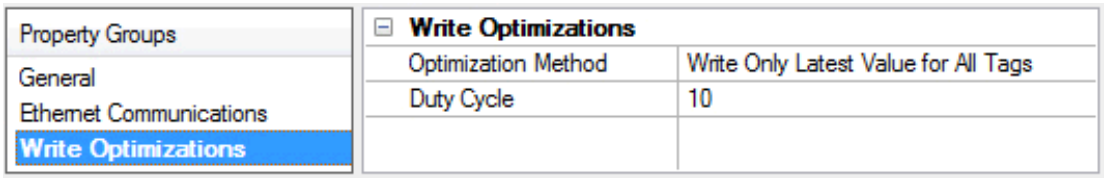
- **Modem**: Specify the installed modem to be used for communications.
- **Connect Timeout**: Specify the amount of time to wait for connections to be established before failing a read or write. The default is 60 seconds.
- **Modem Properties**: Configure the modem hardware. When clicked, it opens vendor-specific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



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.

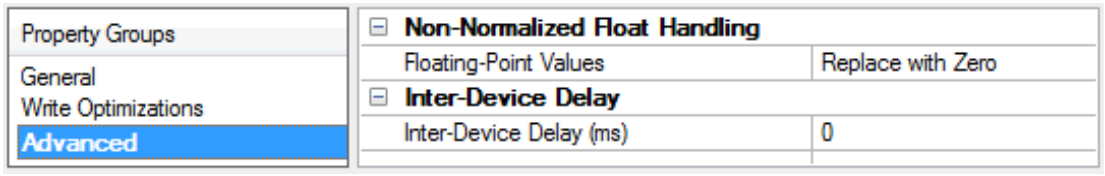
  Note: This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.
- **Write Only Latest Value for All Tags**: This option takes the theory behind the second 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: Non-normalized float handling allows users to specify how a driver handles non-normalized IEEE-754 floating point data. 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. 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 disabled 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.lin

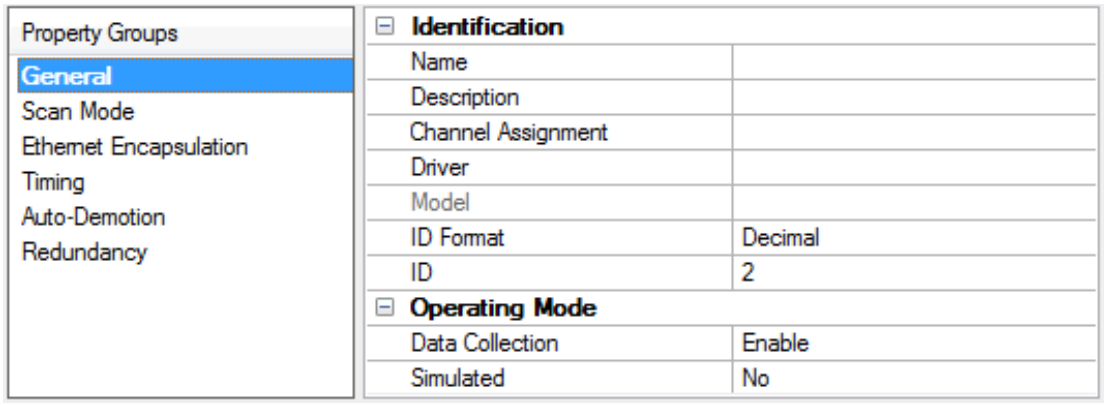
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

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.

Device Properties - General

A device represents a single target on a communications channel. If the driver supports multiple controllers, users must enter a device ID for each controller.



Identification

Name: This property specifies the name of the device. It is a logical user-defined name that can be up to 256 characters long, and may be used on multiple channels.

Note: Although descriptive names are generally a good idea, some OPC client applications may have a limited display window when browsing the OPC server's tag space. The device name and channel name become part of the browse tree information as well. Within an OPC client, the combination of channel name and device name would appear as "ChannelName.DeviceName".
For more information, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in server help.

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

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

Driver: Selected protocol driver for this device.

Model: This property specifies the specific type of device that is associated with this ID. The contents of the drop-down menu depends on the type of communications driver being used. Models that are not supported by a driver are disabled. If the communications driver supports multiple device models, the model selection can only be changed when there are no client applications connected to the device.

Note: If the communication driver supports multiple models, users should try to match the model selection to the physical device. If the device is not represented in the drop-down menu, select a model that conforms closest to the target device. Some drivers support a model selection called "Open," which allows users to communicate without knowing the specific details of the target device. For more information, refer to the driver help documentation.

ID: This property specifies the device's driver-specific station or node. The type of ID entered depends on the communications driver being used. For many communication drivers, the ID is a numeric value. Drivers that support a Numeric ID provide users with the option to enter a numeric value whose format can be changed to suit the needs of the application or the characteristics of the selected communications driver. The ID format can be Decimal, Octal, and Hexadecimal.

Note: If the driver is Ethernet-based or supports an unconventional station or node name, the device's TCP/IP address may be used as the device ID. TCP/IP addresses consist of four values that are separated by periods, with each value in the range of 0 to 255. Some device IDs are string based. There may be additional properties to configure within the ID field, depending on the driver. For more information, refer to the driver's help documentation.

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.

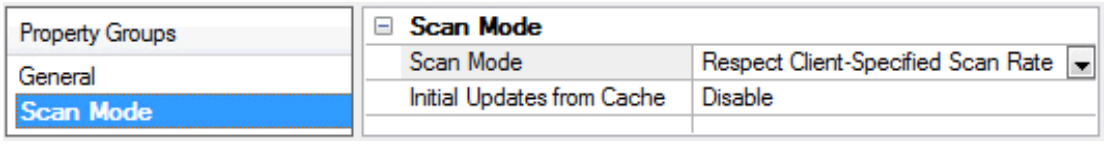
Notes:

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.



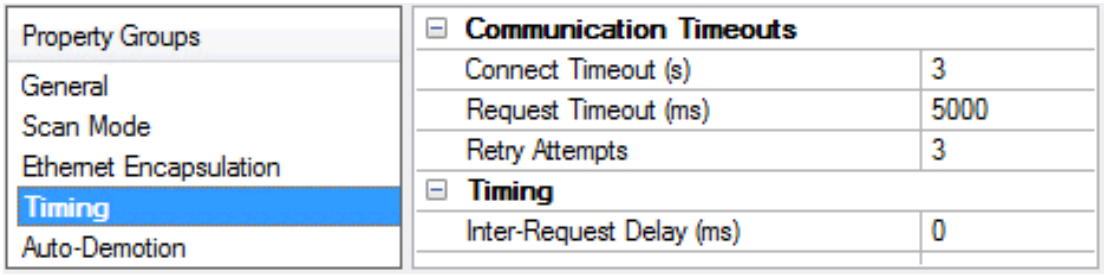
Scan Mode: specifies how tags in the device are scanned for updates sent to subscribed clients. Descriptions of the options are:

- **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 maximum scan rate to be used. 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 - Timing

The device Communications Timeouts properties allow the driver's response to error conditions to be tailored to fit the application's needs. In many cases, the environment requires changes to these properties for optimum performance. Factors such as electrically generated noise, modem delays, and poor physical connections can influence how many errors or timeouts a communications driver encounters. Communications Timeouts properties are specific to each configured device.



Communications Timeouts

Connect Timeout: This property (which is used primarily by Ethernet based drivers) controls the amount of time required to establish a socket connection to a remote device. The device's connection time often takes longer than normal communications requests to that same device. The valid range is 1 to 30 seconds. The default is typically 3 seconds, but can vary depending on the driver's specific nature. If this setting is not supported by the driver, it is disabled.

Note: Due to the nature of UDP connections, the connection timeout setting is not applicable when communicating via UDP.

Request Timeout: This property specifies an interval used by all drivers to determine how long the driver waits for a response from the target device to complete. The valid range is 50 to 9,999,999 milliseconds (167.6667 minutes). The default is usually 1000 milliseconds, but can vary depending on the driver. The default timeout for most serial drivers is based on a baud rate of 9600 baud or better. When using a driver at lower baud rates, increase the timeout to compensate for the increased time required to acquire data.

Retry Attempts: This property specifies how many times the driver retries a communications request before considering the request to have failed and the device to be in error. The valid range is 1 to 10. The default is typically 3, but can vary depending on the driver's specific nature. The number of retries configured for an application depends largely on the communications environment.

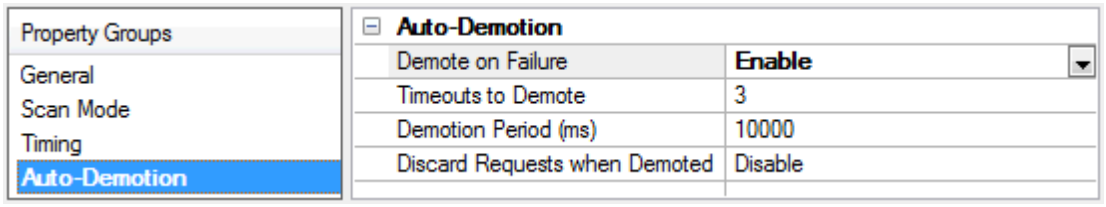
Timing

Inter-Request Delay: This property specifies how long the driver waits before sending the next request to the target device. It overrides the normal polling frequency of tags associated with the device, as well as one-time reads and writes. This delay can be useful when dealing with devices with slow turnaround times and in cases where network load is a concern. Configuring a delay for a device affects communications with all other devices on the channel. It is recommended that users separate any device that requires an inter-request delay to a separate channel if possible. Other communications properties (such as communication serialization) can extend this delay. The valid range is 0 to 300,000 milliseconds; however, some drivers may limit the maximum value due to a function of their particular design. The default is 0, which indicates no delay between requests with the target device.

Note: Not all drivers support Inter-Request Delay. This setting does not appear if it is not supported by the driver.

Device Properties - Auto-Demotion

The Auto-Demotion properties can temporarily place a device off-scan in the event that a device is not responding. By placing a non-responsive device offline for a specific time period, the driver can continue to optimize its communications with other devices on the same channel. After the time period has been reached, the driver re-attempts to communicate with the non-responsive device. If the device is responsive, the device is placed on-scan; otherwise, it restarts its off-scan time period.



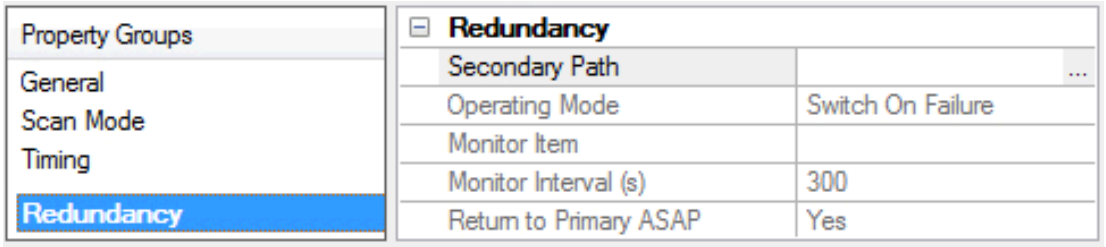
Demote on Failure: When enabled, the device is automatically taken off-scan until it is responding again.
Tip: Determine when a device is off-scan by monitoring its demoted state using the _AutoDemoted system tag.

Timeouts to Demote: Specify how many successive cycles of request timeouts and retries occur before the device is placed off-scan. The valid range is 1 to 30 successive failures. The default is 3.

Demotion Period: Indicate how long the device should be placed off-scan when the timeouts value is reached. During this period, no read requests are sent to the device and all data associated with the read requests are set to bad quality. When this period expires, the driver places the device on-scan and allows for another attempt at communications. The valid range is 100 to 3600000 milliseconds. The default is 10000 milliseconds.

Discard Requests when Demoted: Select whether or not write requests should be attempted during the off-scan period. Disable to always send write requests regardless of the demotion period. Enable to discard writes; the server automatically fails any write request received from a client and does not post a message to the Event Log.

Device Properties - Redundancy



Redundancy is available with the Media-Level Redundancy Plug-in.

Consult the website, a sales representative, or the user manual for more information.

Modem Setup

This driver supports modem functionality. For more information, please refer to the topic "Modem Support" in the OPC Server Help documentation.

Diagnostics Tags

Diagnostics Tags provide information on how the Toyopuc Serial Driver is performing at both the channel and device level. At the channel level, the diagnostics tags provide information that covers all operations performed by the driver when communicating with any PLC on the network. At the device level, the diagnostics tags provide information that pertains only to the device under which the diagnostic tags have been requested.

Channel Level Diagnostics Tags

Tag Name	Functional Description
_ChannelReadTime	Contains the amount of time in milliseconds required to read all currently active data for all devices on this channel. This value is a signed long.
_ChannelHighTime	Contains the amount of time in milliseconds of longest read cycle. This value is a signed long.
_ChannelLowTime	Contains the amount of time in milliseconds of shortest read cycle. This value is a signed long.
_ ChannelReadsPerformed	Contains a count of the reads performed on this channel for all devices. This is a signed long and will roll over.
_ ChannelWritesPerformed	Contains a count of the writes performed on this channel for all devices. This is a signed long and will roll over.
_ChannelTimeouts	Contains a count of the number of timeout/message failures that have occurred for all devices. The _ChannelTimeouts count is for any error that may occur on a message attempt. The value does not necessarily indicate how many messages failed to be sent altogether. It should be used to diagnose possible communication issues with specific devices. This is a signed long and will roll over.

Device Level Diagnostics Tags

Tag Name	Functional Description
_DeviceReadTime	Contains the amount of time in milliseconds required to read a block of data from the specified device. This value is a signed long.
_DeviceHighTime	Contains the longest amount of time in milliseconds required to read a block of data from the specified device. This value is a signed long.
_DeviceLowTime	Contains the shortest amount of time in milliseconds required to read a block of data from the specified device. This value is a signed long.
_ DeviceReadsPerformed	Contains a count of the reads performed on this device. This is a signed long and will roll over.
_ DeviceWritesPerformed	Contains a count of the writes performed on this device. This is a signed long and will roll over.
_DeviceTimeouts	Contains a count of the number of timeout/message failures that have occurred on the specified device. The _DeviceTimeouts count is for any error that may occur on a message attempt. The value does not necessarily indicate how many messages failed to be sent altogether. It should be used to diagnose possible communication issues with this specific device. This is a signed long and will roll over.
_DeviceMultiPointReads	Contains a count of the number of Multi Point read requests that are currently being used to acquire all data that is marked for Multi Point operation. This tag can be used to tune Multi Point read operation. The goal of course being to

Tag Name	Functional Description
	limit the number of Multi Point reads being done to the lowest count possible, preferably 1. This is a signed long and will roll over.

Note: All diagnostics tags are Read/Write. The only value that can be written to the tags is zero (which will clear or reset them).

Multi-Point Read Support

The Toyopuc PLC supports the ability to read data spread randomly throughout the PLC using a single command. This Multi Point command is limited to reading 128 individual 16 bit values in a single request. By using this command, users can read crucial data items quickly and efficiently. The Toyopuc Serial Driver automatically attempts to make the use of the Multi Point command both easy and efficient. Any memory type that can be acquired by the Toyopuc Serial Driver can be part of a Multi Point read command. To mark a particular data item to be part of a Multi Point request simply place the '#' character in front of any current memory type. The table below is shown with the addition of the '#' character to each memory type.

There are some things that should be considered when using the Multi Point read functions. As stated, the Multi Point function can only read 128 16 bit values per request. With this in mind the Multi Point command can increase the speed of data acquisition but if over used the driver will still need to make multiple Multi Point commands to read all the requested data. When this occurs, the overall performance of the driver would again be reduced. The key is to use the Multi Point command wisely.

The driver will automatically group data from memory types like bit memory into 16 bit values. For example if X1, X3, X4, X6, X9, XA, XB are marked as part of a Multi Point read using the '#,' enter an address of #X1, #X3, #X4, #X6, #X9, #XA, #XB, so that these 7 items would be placed into a single 16 bit value. Thus, they would only use one of the 128 16 bit values available in a single Multi Point read command. The 7 items were grouped together because the address of each bit fell within a single 16 bit word value of X memory. If 7 items like #X1,#X20,#X55,#X77,#X99,#XAA,#XBB are entered as part of a Multi Point read, each bit in this case would require an entire 16 bit value in the Multi Point read command to receive the data. Plan the data usage in the controller. If possible make sure that the bits being read are grouped closely. This prudent planning applies primarily to the bit memory types. Register memory requires a single 16 bit value for each register added to the Multi Point read.

Such information will help plan Multi Point reads. The Toyopuc Serial Driver can perform as many Multi Point reads as are needed to acquire all the data that has been have marked for Multi Point operation, but the more reads done, the slower the driver will be. To determine how many Multi Point read request the Toyopuc Serial Driver is using to acquire all currently defined Multi Point data, use the special diagnostic tag "_DeviceMultiPointReads." For more information, refer to **Diagnostics**.

The Multi Point read operation can be combined with the normal data reads of the Toyopuc Serial Driver. If, for example, a block of 50 D registers consecutively ordered is being read, it may be more efficient to read the 50 D registers as part of a normal block read and save the space in the Multi Point read function for data that is spread more randomly throughout the PLC memory. Use the diagnostic tags to help determine the application's best method of acquiring data.

Multi Point Limitations

For tags belonging to the PC2 series device model, the maximum data requested for the data types are as follows:

Boolean: 2048*
Byte: 256

Word: 128
DWord: 64

*If contiguous Booleans are requested, the request will be done in one Multi Point read.

Combination of any of the above data types in a single multi request has to be within the following limit:

(No. of Booleans/16) + (No. of Bytes/2) + No. of Words + (No. of DWords * 2) <= 128

Memory Types Shown with the Multi Point Marker

Memory Type	Syntax	Data Types	Access
Edge Relay (P)	#P0000-#P01FF #P000-#P01F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Keeping Relay (K)	#K0000-#K02FF #K000-#K02F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Specific Relay (V)	#V0000-#V00FF #V000-#V0F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Timer Bits (T)	#T0000-#T01FF #T000- #T01F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Count Bits (C)	#C0000-#C01FF #C000-#C01F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Link Relay (L)	#L0000-#L07FF #L000-#L07F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
I/O Relay (X)	#X0000-#X07FF #X000-#X07F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
I/O Relay (Y)	#Y0000-#Y07FF #Y000-#Y07F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Internal Relay (M)	#M0000-#M07FF #M000-#M07F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Specific Register (S)	#S0000-#S03FF	Byte, **Word**, Short, BCD, DWord, Long, LBCD	Read/Write
Timer/Counter Value (N)	#N0000-#N01FF	Byte, **Word**, Short, BCD, DWord, Long, LBCD	Read/Write
Link Register (R)	#R0000-#R07FF	Byte, **Word**, Short, BCD, DWord, Long, LBCD	Read/Write
Data Register (D)	#D0000-#D2FFF	Byte, **Word**, Short, BCD, DWord, Long, LBCD	Read/Write
File Register (B)	#B0000-#B1FFF	Byte, **Word**, Short, BCD, DWord, Long, LBCD	Read/Write

Data Types Description

Data Type	Description
Boolean	Single bit
Byte	Unsigned 8 bit value bit 0 is the low bit bit 7 is the high 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.

Address Descriptions

The Toyopuc PC2 Series supports the following addresses. The default data types for dynamically defined tags are shown in **bold**.

Memory Type	Syntax	Data Types	Access
Edge Relay (P)	P0000-P01FF P000-P01F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Keeping Relay (K)	K0000-K02FF K000-K02F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Specific Relay (V)	V0000-V00FF V000-V0F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Timer Bits (T)	T0000-T01FF T000- T01F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Count Bits (C)	C0000-C01FF C000-C01F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Link Relay (L)	L0000-L07FF L000-L07F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
I/O Relay (X)	X0000-X07FF X000-X07F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
I/O Relay (Y)	Y0000-Y07FF Y000-Y07F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Internal Relay (M)	M0000-M07FF M000-M07F	**Boolean** Byte, Word, Short, BCD, DWord, Long, LBCD	Read/Write
Specific Register (S)	S0000-S03FF	Byte, **Word**, Short, BCD, DWord, Long, LBCD	Read/Write
Timer/Counter Value (N)	N0000-N01FF	Byte, **Word**, Short, BCD, DWord, Long, LBCD	Read/Write
Link Register (R)	R0000-R07FF	Byte, **Word**, Short, BCD, DWord, Long, LBCD	Read/Write
Data Register (D)	D0000-D2FFF	Byte, **Word**, Short, BCD, DWord, Long, LBCD	Read/Write
File Register (B)	B0000-B1FFF	Byte, **Word**, Short, BCD, DWord, Long, LBCD	Read/Write

Multi Point Read Support

The Multi Point read support allows the Toyopuc Serial Driver to read data from multiple memory types in a single request. For more information, refer to **Multi Point Read Support**.

Diagnostics Tags

The Diagnostics Tags provide information on how the Toyopuc Serial Driver is performing. For more information, refer to **Diagnostics Tags**.

Low/High Byte Modifier

An optional Low (L) or High (H) byte modifier can be appended to any address. This modifier instructs the driver that the low or high byte of the address word is requested.

Note: When adding a static tag with a low/high byte modifier the data type must be set to Byte. For more information on static vs. dynamic tags, refer to the OPC Server's online help.

Examples
1. Request 'Specific Relay 100'--> V100.
2. Request High Byte of 'Data Register 10'--> D10H.
3. Request Long Value (2 consecutive 16 bit registers) starting at 'Link Register 7F'--> L7F@LONG (set datatype to Long for static tags, or append '@LONG' to address for dynamic tags.

Error Descriptions

The following error/warning messages may be generated. Click on the link for a description of the message.

Address Validation

Missing address
Device address '<address>' contains a syntax error
Address '<address>' is out of range for the specified device or register
Device address '<address>' is not supported by model '<model name>'
Data Type '<type>' is not valid for device address '<address>'
Device address '<address>' is Read Only
Array size is out of range for address '<address>'
Array support is not available for the specified address: '<address>'

Serial Communications

COMn does not exist
Error opening COMn
COMn is in use by another application
Unable to set comm properties on COMn
Communications error on '<channel name>' [<error mask>]

Device Status Messages

Device '<device name>' is not responding
Unable to write to '<address>' on device '<device name>'

Device-Specific Messages

Bad address in block [<start address> to <end address>] on device '<device name>'

Missing address

Error Type:

Warning

Possible Cause:

A tag address that has been specified dynamically has no length.

Solution:

Re-enter the address in the client application.

Device address '<address>' contains a syntax error

Error Type:

Warning

Possible Cause:

A tag address that has been specified dynamically contains one or more invalid characters.

Solution:

Re-enter the address in the client application.

Address '<address>' is out of range for the specified device or register

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically references a location that is beyond the range of supported locations for the device.

Solution:
Verify that the address is correct; if it is not, re-enter it in the client application.

Device address '<address>' is not supported by model '<model name>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically references a location that is valid for the communications protocol but not supported by the target device.

Solution:
Verify that the address is correct; if it is not, re-enter it in the client application. Also verify that the selected model name for the device is correct.

Data Type '<type>' is not valid for device address '<address>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically has been assigned an invalid data type.

Solution:
Modify the requested data type in the client application.

Device address '<address>' is Read Only

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically has a requested access mode that is not compatible with what the device supports for that address.

Solution:
Change the access mode in the client application.

Array size is out of range for address '<address>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically is requesting an array size that is too large for the address type or block size of the driver.

Solution:
Re-enter the address in the client application to specify a smaller value for the array or a different starting point.

Array support is not available for the specified address: '<address>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically contains an array reference for an address type that doesn't support arrays.

Solution:
Re-enter the address in the client application to remove the array reference or correct the address type.

COMn does not exist

Error Type:
Fatal

Possible Cause:
The specified COM port is not present on the target computer.

Solution:
Verify that the proper COM port has been selected in the Channel Properties.

Error opening COMn

Error Type:
Fatal

Possible Cause:
The specified COM port could not be opened due to an internal hardware or software problem on the target computer.

Solution:
Verify that the COM port is functional and may be accessed by other Windows applications.

COMn is in use by another application

Error Type:
Fatal

Possible Cause:
The serial port assigned to a device is being used by another application.

Solution:

Verify that the correct port has been assigned to the channel.

Unable to set comm properties on COMn

Error Type:
Fatal

Possible Cause:
The serial properties for the specified COM port are not valid.

Solution:
Verify the serial properties and make any necessary changes.

Communications error on '<channel name>' [<error mask>]

Error Type:
Serious

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.

Possible Cause:

1. The serial connection between the device and the Host PC is bad.
2. The communication properties for the serial connection are incorrect.

Solution:

1. Verify the cabling between the PC and the device.
2. Verify that the specified communication properties match those of the device.

Device '<device name>' is not responding

Error Type:
Serious

Possible Cause:

1. The serial connection between the device and the Host PC is broken.
2. The communication properties for the serial connection are incorrect.
3. The named device may have been assigned an incorrect Network ID.
4. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device property.

Solution:

1. Verify the cabling between the PC and the device.
2. Verify that the specified communication properties match those of the device.
3. Verify that the Network ID given to the named device matches that of the actual device.
4. Increase the Request Timeout property so that the entire response can be handled.

Unable to write to '<address>' on device '<device name>'

Error Type:
Serious

Possible Cause:

1. The serial connection between the device and the Host PC is broken.
2. The communication properties for the serial connection are incorrect.
3. The named device may have been assigned an incorrect Network ID.

Solution:

1. Verify the cabling between the PC and the device.
2. Verify that the specified communication properties match those of the device.
3. Verify that the Network ID given to the named device matches that of the actual device.

Bad address in block [<start address> to <end address>] on device '<device name>'

Error Type:
Serious

Possible Cause:
An attempt has been made to reference a nonexistent location in the specified device.

Solution:
Verify the tags assigned to addresses in the specified range on the device. Eliminate ones that reference invalid locations.

Index

A

Address '<address>' is out of range for the specified device or register 21
Address Descriptions 18
Advanced Channel Properties 9
Array size is out of range for address '<address>' 21
Array support is not available for the specified address: '<address>' 22
Auto Dial 7

B

Bad address in block [<start address> to <end address>] on device '<device name>' 24
Baud Rate 6
BCD 17
Boolean 17

C

Channel Assignment 10
Channel Properties - General 4
Channel Properties - Write Optimizations 8
Close Idle Connection 7
COM ID 6
Communications error on '<channel name>' [<error mask>] 23
Communications Timeouts 12
COMn does not exist 22
COMn is in use by another application 22
Connect Timeout 12
Connection Type 6

D

Data Bits 6
Data Collection 10
Data Type '<type>' is not valid for device address '<address>' 21
Data Types Description 17

Demote on Failure 13
Demotion Period 13
Description 10
Device '<device name>' is not responding 23
Device address '<address>' contains a syntax error 20
Device address '<address>' is not supported by model '<model name> 21
Device address '<address>' is Read Only 21
Device ID 4
Device Properties - Auto-Demotion 13
Device Properties - General 9
Diagnostics 5
Diagnostics Tags 14
Discard Requests when Demoted 13
Do Not Scan, Demand Poll Only 11
Driver 5, 10
Duty Cycle 8
DWord 17

E

Error Descriptions 20
Error opening COMn 22

F

Flow Control 6
Framing 23

I

ID 10
Idle Time to Close 7
IEEE-754 floating point 9
Initial Updates from Cache 11
Inter-Request Delay 12

L

LBCD 17
Long 17

M

Mask 23
Missing address 20
Model 10
Modem 7
Modem Setup 13
Multi Point Read Support 15

N

Name 10
Network 4
Network Adapter 7
Non-Normalized Float Handling 9

O

Operational Behavior 7
Optimization Method 8
Overrun 23
Overview 3

P

Parity 6, 23
Physical Medium 6

R

Read Processing 7
Redundancy 13

Report Comm. Errors 7
Request All Data at Scan Rate 11
Request Data No Faster than Scan Rate 11
Request Timeout 12
Respect Client-Specified Scan Rate 11
Respect Tag-Specified Scan Rate 11
Retry Attempts 12

S

Scan Mode 11
Serial Communications 5
Serial Port Settings 6
Setup 4
Short 17
Simulated 10
Stop Bits 6

T

Timeouts to Demote 13

U

Unable to set comm properties on COMn 23
Unable to write tag '<address>' on device '<device name>' 24

W

Word 17
Write All Values for All Tags 8
Write Only Latest Value for All Tags 8
Write Only Latest Value for Non-Boolean Tags 8
Write Optimizations 8