Cutler-Hammer ELC Serial Driver

Table of Contents

Welcome to the Cutler-Hammer ELC Serial Driver Help Center

This help center is the user documentation for Kepware Cutler-Hammer ELC Serial Driver. This help center is updated regularly to reflect the latest functionality and information.

Version 1.051

Overview

The Cutler-Hammer ELC Serial Driver provides a reliable way to connect Cutler-Hammer Eaton Logic Controllers (ELC) to OPC Client applications, including HMI, SCADA, Historian, MES, ERP and countless custom applications. This driver is intended for use with Cutler-Hammer ELC Serial devices.

Setup

Supported Devices

The following Cutler-Hammer ELC models are supported:

PA10 Series
PB14 Series
PC12 Series
PV28 Series

Supported Communication Parameters

Baud Rate: 9600, 19200, 38400, 57600, 115200
Parity: Odd, Even, None
Data Bits: 7, 8
Stop Bits: 1,2

Note: Some devices may not support the listed configurations.

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 256 per channel. Devices are assigned Device IDs (PLC Network Address) in the range 0 to 255.

Ethernet Encapsulation

This driver supports Ethernet Encapsulation. Ethernet Encapsulation allows the driver to communicate with serial devices attached to an Ethernet network using a terminal server. Ethernet Encapsulation mode is invoked by selecting it from the COM ID dialog in Channel Properties. For more information on Ethernet Encapsulation, refer to the main OPC Server help file.

Flow Control

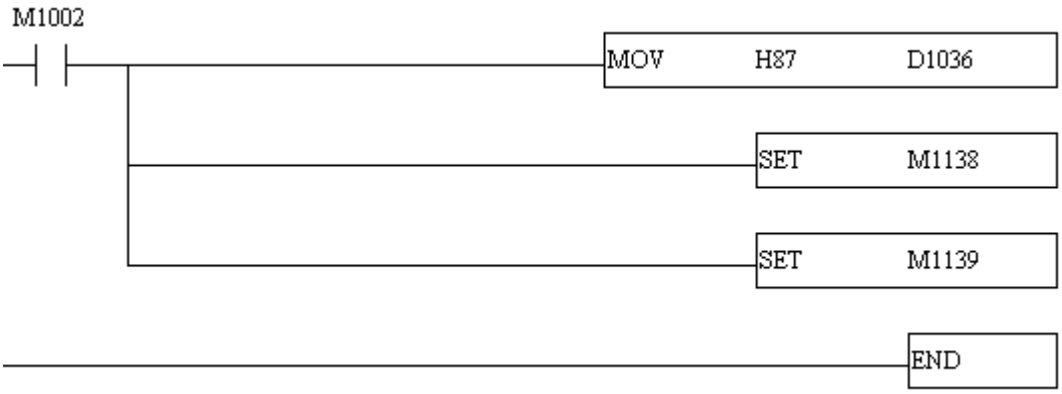
When using an RS232/RS485 converter, the type of flow control that is required depends on the needs of the converter. Some converters do not require any flow control and others will require RTS flow. Consult the converter's documentation to determine its flow requirements. We recommend using an RS485 converter that provides automatic flow control.

Note: When using the manufacturer's supplied communications cable, it is sometimes necessary to choose a flow control setting of **RTS** or **RTS Always** under the Channel Properties.

The Cutler-Hammer ELC Serial Driver supports the RTS Manual flow control option. This selection is used to configure the driver for operation with radio modems that require special RTS timing characteristics. For more information on RTS Manual flow control, refer to the main OPC Server help file topic Channel Wizard.

Enabling Modbus RTU

Whenever the ELC is rebooted, it assumes the communication protocol is Modbus ASCII. To use Modbus RTU, an RTU switching ladder logic needs to be created to toggle from ASCII to RTU.

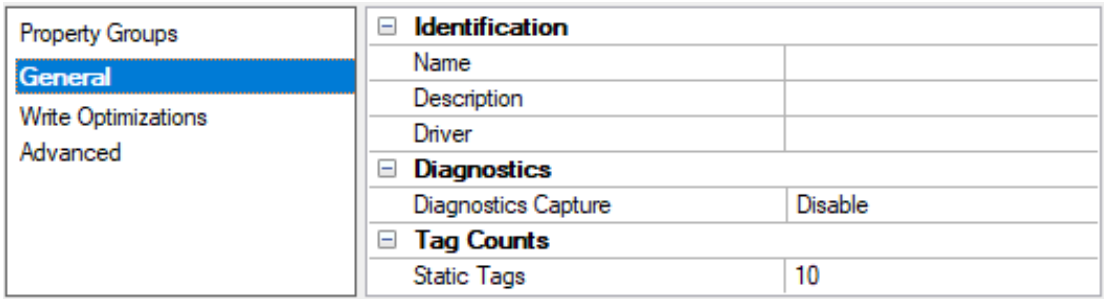


Ladder Logic Notes

- M1002 is normally open and is only pulsed closed once, at the first scan through the logic after the ELC boots up. This drives the rest of the logic.
- MOV H87 D1036 moves the hex value 0x87 into word register D1036, which the PLC examines whenever RTU/ASCII modes are toggled.
- SET M1138 sets the bit memory m1138 ON. M1138 is referred to as "COM1 (RS-232) communication protocol settings" in the ELC help.
- SET 1139 is similar to SET M1138. M1139 is referred to in the ELC memory as "COM1 (RS-232) ASCII/RTU mode when ELC is server. Off = ASCII, On = RT.

Channel Properties – General

This server supports the use of multiple simultaneous 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: Specify the 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: Specify user-defined information about this channel.
Many of these properties, including Description, have an associated system tag.

Driver: Specify the protocol / driver for this channel. Specify 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. Changes to the properties should not be made once a large client application has been developed. Utilize proper user role and privilege management to prevent operators from changing properties or accessing 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 or operating system does not support diagnostics.
For more information, refer to Communication Diagnostics and Statistics Tags in server help.

Tag Counts

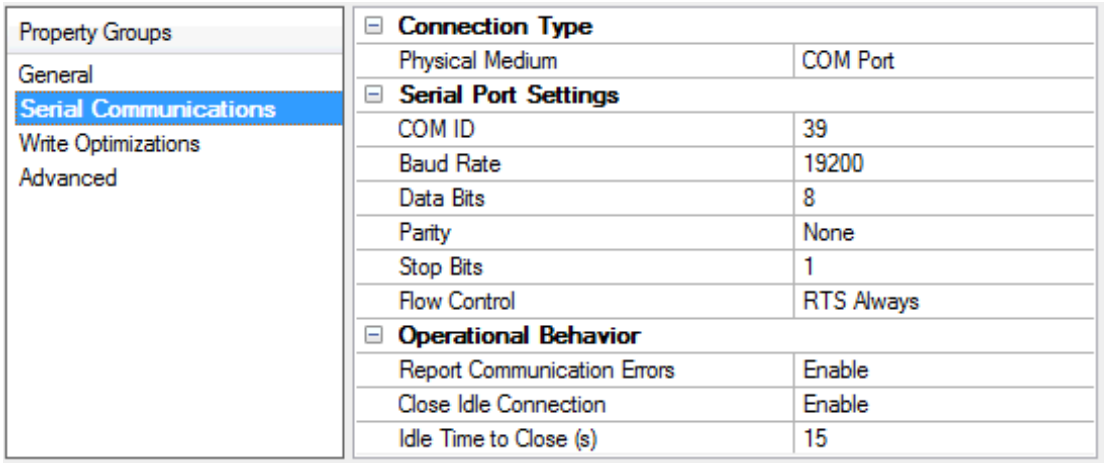
Static Tags: Provides the total number of defined static tags at this level (device or channel). This information can be helpful in troubleshooting and load balancing.

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, and Operational Behavior.

Notes:

- With the server's online full-time operation, these properties can be changed at any time. Utilize proper user role and privilege management to prevent operators from changing properties or accessing server features.
- Users must define the specific communication parameters to be used. Depending on the driver, channels may or may not be able to share identical communication parameters. Only one shared serial connection can be configured for a Virtual Network *(see Channel Properties – Serial Communications)*.



Connection Type

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

1. **None**: Select None to indicate there is no physical connection, which displays the Operation with no Communications section.
2. **COM Port**: Select Com Port to display and configure the Serial Port Settings section.
3. **Modem**: Select Modem if phone lines are used for communications, which are configured in the Modem Settings section.
4. **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**: Specify 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**: Specify 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**: Specify 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.

Modem Settings

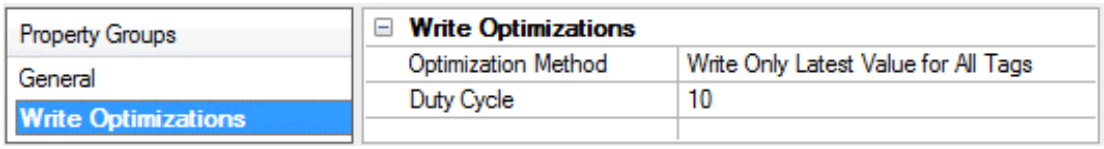
- **Modem**: Specify the installed modem to be used for communications.
- **Connect Timeout**: Specify the amount of time to wait for connections to be established before failing a read or write. The default is 60 seconds.
- **Modem Properties**: Configure the modem hardware. When clicked, it opens vendor-specific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

The server must ensure that the data written from the client application gets to the device on time. Given this goal, the server provides optimization properties to meet specific needs or improve application responsiveness.



Write Optimizations

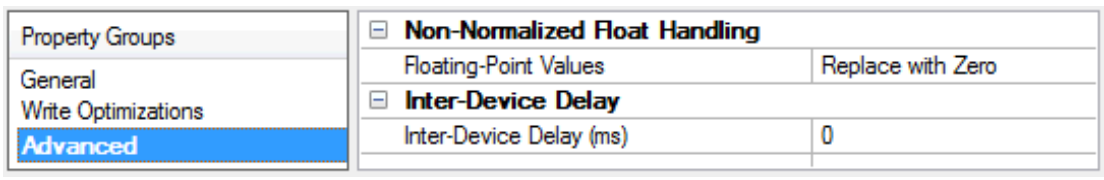
Optimization Method: Controls how write data is passed to the underlying communications driver. The options are:

- **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: 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.

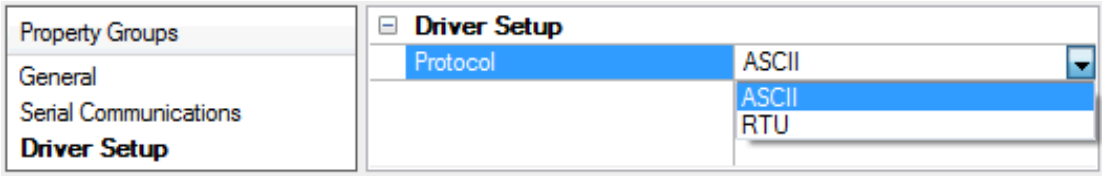
Note: This property is disabled if the driver does not support floating-point values or if it only supports the option that is displayed. According to the channel's float normalization setting, only real-time driver tags (such as values and arrays) are subject to float normalization. For example, EFM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

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.

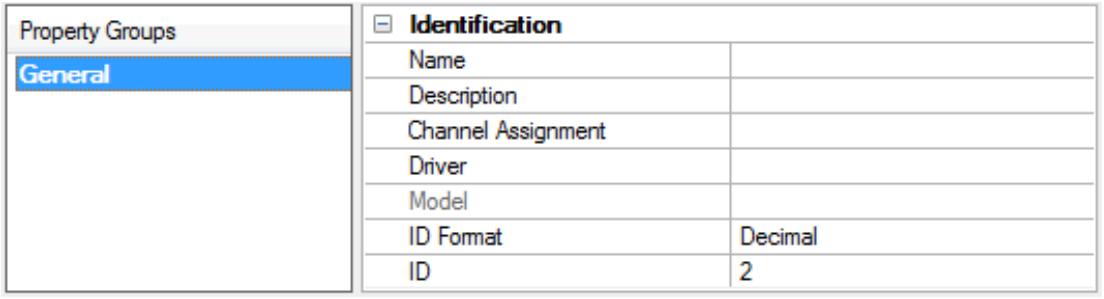
Channel Properties – Driver Setup



Protocol: Options include Modbus ASCII Protocol and Modbus RTU. By default, the ELC assumes communication is in Modbus ASCII. *For information on toggling from ASCII to RTU, refer to Enabling Modbus RTU.*

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

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

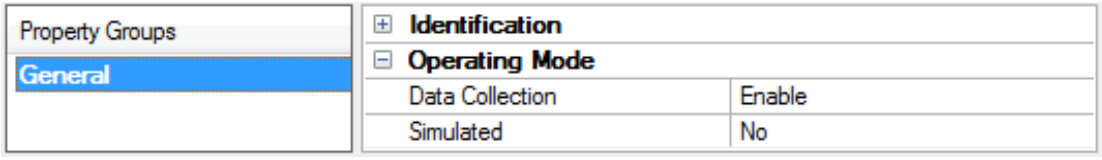
Driver: Selected protocol driver for this device.

Model: Specify the type of device that is associated with this ID. The contents of the drop-down menu depend 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 documentation.*

ID: Specify 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 format is set by the driver by default. Options include Decimal, Octal, and Hexadecimal.

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.

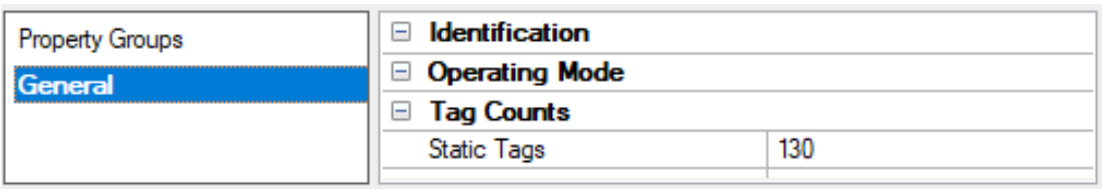
Simulated: Place the device into or out of Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client 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 data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

Notes:

1. Updates are not applied until clients disconnect and reconnect.
2. The 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.
3. 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.
4. When a device is simulated, updates may not appear faster than one (1) second in the client.

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

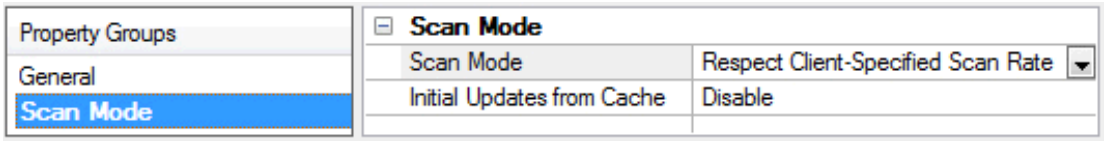
Tag Counts



Static Tags: Provides the total number of defined static tags at this level (device or channel). This information can be helpful in troubleshooting and load balancing.

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.



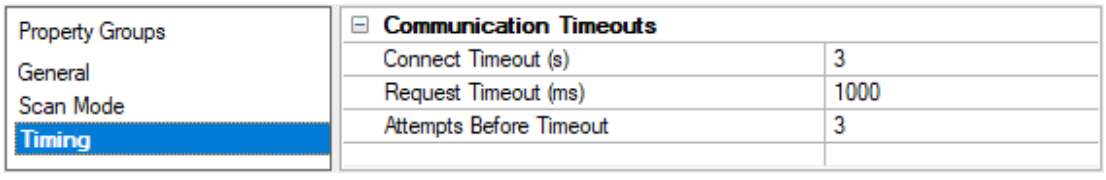
Scan Mode: Specify how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

- **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 OPC 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 Timing 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. Timing 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: Specify 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 9999999 milliseconds (167 minutes). The default is

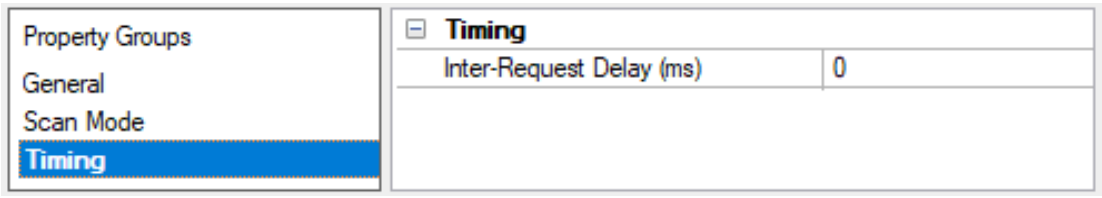
usually 1000 milliseconds, but can vary depending on the driver. The default timeout for most serial drivers is based on a baud rate of 9600 baud or better. When using a driver at lower baud rates, increase the timeout to compensate for the increased time required to acquire data.

Attempts Before Timeout: Specify how many times the driver issues 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 attempts configured for an application depends largely on the communications environment. This property applies to both connection attempts and request attempts.

Timing

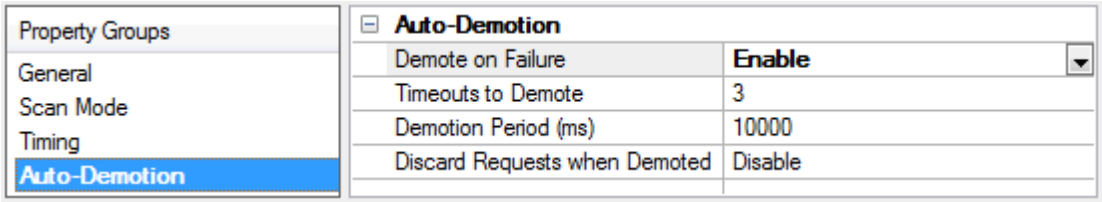
Inter-Request Delay: Specify how long the driver waits before sending the next request to the target device after receiving the response to the previous request. It overrides the normal polling frequency of tags associated with the device, as well as one-time reads and writes. This delay can be useful 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 available.



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 – Block Sizes

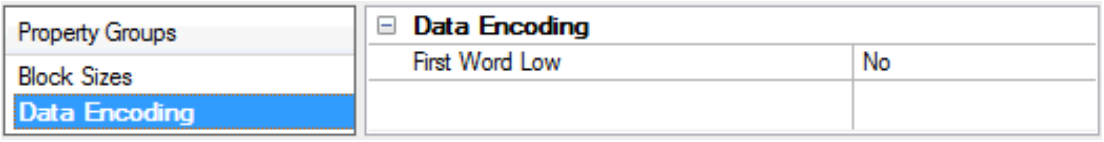
Output: The output block size for ELC memory types S, Y, To, M and Co may be between 8 and 1024 in multiples of 8. The default setting is 256.

Input: The input block size for ELC memory type X may be between 8 and 1024 in multiples of 8. The default setting is 256.

Holding Registers: The holding register size for ELC memory types C, T and D may be between 1 and 120. The default setting is 12. The number of 32-bit counter values that can be blocked together will be one half the number of the holding registers setting, thus yielding an equivalent number of bytes per block.

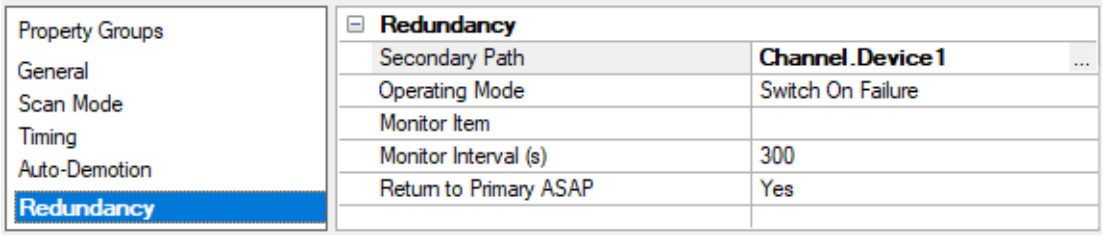
Default values are best for PA10, PB14, and PC12 models. PV28 Series devices are capable of handling larger requests, however, and performance gains may be achieved by increasing the block sizes.

Device Properties – Data Encoding



First Word Low: Two consecutive 16-bit registers' addresses in an ELC device are used for 32-bit data types. Users can specify whether the driver should assume the first word is the low or the high word of the 32-bit value. Check this box to use the same word order as the ELCSoft programming software.

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.

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 on all models and 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

Addressing information is shown in tables below for the PA10/PC12 Series, PB14 Series and PV28 Series. The default data types for dynamically defined tags are shown in **bold** where appropriate.

For notes and restrictions, refer to Write Only Access, String Support and Array Support.

PA10 Series/PC12 Series

Memory Type	Range	Data Type	Access*	Remarks
Discrete Input (external I/O)	X0-X177 (octal)	**Boolean**	Read Only	
Discrete Output (external I/O)	Y0-Y177 (octal)	**Boolean**	Read/Write	
Main Relay	M0-M4095	**Boolean**	Read/Write	
Timer Status	To0-To255	**Boolean**	Read/Write	Contact = ON when timer reaches preset value
Counter Status (16-bit counters)	Co0-Co199	**Boolean**	Read/Write	Contact = ON when counter reaches preset value
Counter Status (32-bit counters)	Co200-Co255	**Boolean**	Read/Write	Contact = ON when counter reaches preset value
Step Point	S0-S1023	**Boolean**	Read/Write	Sequential Function Chart usage
Timer Current Value	T0-T255 Txxx.0-Txxx.15	**Word**, Short, BCD Float, Long, LBCD Double, DWord Boolean	Read/Write	Represents the current value of the 1ms, 10ms, or 100ms timer
Counter Current Value (16-bit counters)	C0-C199 Cxxx.0-Cxxx.15	**Word**, Short, BCD Float, Long, LBCD Double, DWord Boolean	Read/Write	Represents the current value of the counter
Counter Current Value (32-bit counters)	C200-C255 Cxxx.0-Cxxx.31	**DWord**, Float Long, LBCD, Double Boolean	Read/Write Read Only	Represents the current value of the counter
Data Register	D0-D4999 Dxxx.0-Dxxx.15	**Word**, Short, BCD Float, Long, LBCD Double, DWord Boolean	Read/Write	General data storage
Data Register as String with HiLo Byte Order	D0.2H-D4984.32H	**String****	Read/Write	.Bit is string length, range 2 to 32 bytes
Data Register as String with LoHi Byte Order	D0.2L-D4984.32L	**String****	Read/Write	.Bit is string length, range 2 to 32 bytes

PB14 Series

Memory Type	Range	Data Type	Access*	Remarks
Discrete Input (external I/O)	X0-X177 (octal)	**Boolean**	Read Only	
Discrete Output	Y0-Y177 (octal)	**Boolean**	Read/Write	

Memory Type	Range	Data Type	Access*	Remarks
(external I/O)				
Main Relay	M0-M1279	**Boolean**	Read/Write	
Main Relay	M2000-M4095	**Boolean**	Read/Write	
Timer Status	To0-To127	**Boolean**	Read/Write	Contact = ON when timer reaches preset value
Counter Status (16-bit counters)	Co0-Co127	**Boolean**	Read/Write	Contact = ON when counter reaches preset value
Counter Status (32-bit counters)	Co232-Co255	**Boolean**	Read/Write	Contact = ON when counter reaches preset value
Step Point	S0-S127	**Boolean**	Read/Write	Sequential Function Chart usage
Timer Current Value	T0-T127 Txxx.0-Txxx.15	**Word**, Short, BCD Float, Long, LBCD Double, DWord Boolean	Read/Write	Represents the current value of the 1ms, 10ms, or 100ms timer
Counter Current Value (16-bit counters)	C0-C127 Cxxx.0-Cxxx.15	**Word**, Short, BCD Float, Long, LBCD Double, DWord Boolean	Read/Write	Represents the current value of the counter
Counter Current Value (32-bit counters)	C232-C255 Cxxx.0-Cxxx.31	**DWord**, Float Long, LBCD, Double Boolean	Read/Write Read Only	Represents the current value of the counter
Data Register	D0-D599 Dxxx.0-Dxxx.15	**Word**, Short, BCD Float, Long, LBCD Double, DWord Boolean	Read/Write	General data storage
Data Register	D1000-D1311 Dxxx.0-Dxxx.15	**Word**, Short, BCD Float, Long, LBCD Double, DWord Boolean	Read/Write	General data storage
Data Register as String with HiLo Byte Order	D0.2H-D584.32H	**String****	Read/Write	.Bit is string length, range 2 to 32 bytes
Data Register as String with LoHi Byte Order	D0.2L-D584.32L	**String****	Read/Write	.Bit is string length, range 2 to 32 bytes
Data Register as String with HiLo Byte Order	D1000.2H-D1296.32H	**String****	Read/Write	.Bit is string length, range 2 to 32 bytes
Data Register as String with LoHi Byte Order	D1000.2L-D1296.32L	**String****	Read/Write	.Bit is string length, ange 2 to 32 bytes

PV28 Series

Memory Type	Range	Data Type	Access*	Remarks
Discrete Input (external I/O)	X0-X377 (octal)	**Boolean**	Read Only	
Discrete Output (external I/O)	Y0-Y377 (octal)	**Boolean**	Read/Write	
Main Relay	M0-M4095	**Boolean**	Read/Write	
Timer Status	To0-To255	**Boolean**	Read/Write	Contact = ON when timer reaches preset value
Counter Status (16-bit counters)	Co0-Co199	**Boolean**	Read/Write	Contact = ON when counter reaches preset value
Counter Status (32-bit counters)	Co200-Co255	**Boolean**	Read/Write	Contact = ON when counter reaches preset value
Step Point	S0-S1023	**Boolean**	Read/Write	Sequential Function Chart usage
Timer Current Value	T0-T255 Txxx.0-Txxx.15	**Word**, Short, BCD Float, DWord, Long LBCD, Double Boolean	Read/Write	Represents the current value of the 1ms, 10ms, or 100ms timers
Counter Current Value (16-bit counters)	C0-C199 Cxxx.0-Cxxx.15	**Word**, Short, BCD Float, DWord, Long LBCD, Double Boolean	Read/Write	Represents the current value of the counter
Counter Current Value (32-bit counters)	C200-C255 Cxxx.0-Cxxx.31	**DWord**, Float Long, LBCD, Double Boolean	Read/Write Read Only	Represents the current value of the counter
Data Register	D0-D9999 Dxxx.0-Dxxx.15	**Word**, Short, BCD Float, DWord, Long LBCD, Double Boolean	Read/Write	General data storage
Data Register as String with HiLo Byte Order	D0.2H-D9984.32H	**String****	Read/Write	.Bit is string length, range 2 to 32 bytes
Data Register as String with LoHi Byte Order	D0.2L-D9984.32L	**String****	Read/Write	.Bit is string length, range 2 to 32 bytes

*Write Only Access

All Read/Write addresses may be set as Write by prefixing a "W" to the address such as "WD0", which will prevent the driver from reading the register at the specified address. Any attempts by the client to read a write-only tag will result in obtaining the last successful write value to the specified address. If no successful writes have occurred, then the client will receive 0/NULL for numeric/string values for an initial value.

Caution: Setting the "Client access" privileges of write-only tags to Read Only will cause writes to these tags to fail and the client to always receive 0/NULL for numeric/string values.

**String Support

The ELC devices support reading and writing data register memory as an ASCII string. When using data registers for string data, each register will contain 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 32 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 an "H" or "L" to the address.

Examples

1. To address a string starting at D200 with a length of 32 bytes and HiLo byte order, enter: D200.32H
2. To address a string starting at D500 with a length of 32 bytes and LoHi byte order, enter: D500.32L

Note: The string length may be limited by the maximum size of the write request that your device will allow. If while utilizing a string tag you receive an error message in the server event window of "Unable to write to address <address> on device <device>: Device responded with exception code 3." This means the device did not accept the length of your string. If possible, try shortening the string.

Normal Address Example

The 177'th (octal) output coil would be addressed as 'Y177' using octal addressing.

Array Support

Arrays are supported for registers (i.e. T, C, and D addresses) except for bit-within-word and strings. Arrays are also supported for input and output coils (Boolean data types, i.e. M, S, X, Y, To, and Co addresses). There are two methods of addressing an array. Examples are given using register addresses.

Dxxx [rows] [cols]
Dxxx [cols] this method assumes rows is equal to one

For arrays, rows multiplied by cols cannot exceed the block size that has been assigned to the device for the register/coil type. For register arrays of 32-bit data types, rows multiplied by cols multiplied by 2 cannot exceed the block size.

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 parameters 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>'
Unable to write to address '<address>' on device '<device>': Device responded with exception code '<code>'

Device Specific Messages

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

Missing address

Error Type:

Warning

Possible Cause:

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

Solution:
Verify 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 statically 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 statically 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 statically 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 statically 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 statically 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.

Error opening COMn

Error Type:
Fatal

Possible Cause:
The specified COM port could not be opened due 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 and that only one copy of the current project is running.

Unable to set comm parameters on COMn

Error Type:
Fatal

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

Solution:
Verify the serial parameters 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:
Either the serial connection between the device and the Host PC is bad or the communications parameters for the serial connection are incorrect.

Solution:
Verify the cabling between the PC and the PLC device. Also, verify that the specified communications parameters 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 communications parameters 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 setting.

Solution:
1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communications parameters 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 setting 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 communications parameters 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 PLC device.
2. Verify that the specified communications parameters match those of the device.
3. Verify that the Network ID given to the named device matches that of the actual device.

Unable to write to address '<address>' on device '<device>': Device responded with exception code '<code>'

Error Type:
Warning

Possible Cause:
Refer to the Modbus Exception Codes.

Solution:
Solution depends upon the Modbus Exception Codes.

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 and eliminate ones that reference invalid locations.

Bad array spanning ['<address>' to '<address>'] on device '<device name>'

Error Type:
Fatal

Possible Cause:
An array of addresses was defined that spans past the end of the address space.

Solution:
Verify the size of the device's memory space and then redefine the array length accordingly.

Index

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

U

W