AutomationDirect ECOM Driver

Table of Contents

AutomationDirect ECOM Driver

Help version 1.021

CONTENTS

Overview

The AutomationDirect ECOM Driver provides a reliable way to connect AutomationDirect ECOM controllers to OPC client applications, including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with AutomationDirect Programmable Logic Controllers that may be accessed via an ECOM Ethernet module.

Setup

Supported Devices*

DL-05
DL-06
DL-240
DL-250(-1)
DL-260
DL-430
DL-440
DL-450

*All PLCs via an Hx-ECOM module.

Communication Protocol

Ethernet using Winsock V1.1 or higher.

Connection Timeout

This property specifies the time that the driver will wait for a connection to be made with a device. Depending on network load, the connect time may vary with each connection attempt. The default setting is 3 seconds. The valid range is 1 to 60 seconds.

Request Timeout

This property specifies the time the driver will wait on a response from the device before giving up and going on to the next request. Longer timeouts only affect performance if a device is not responding. The default setting is 250 milliseconds. The valid range is 50 to 9999 milliseconds.

Retry Attempts

This property specifies the number of times the driver will retry a message before giving up and going on to the next message. The default setting is 3 retries. The valid range is 1 to 10.

Device IDs

Up to 1024 devices may be defined on a given channel. Each device on the channel must be uniquely identified by its own IP address. In general the Device ID has the following format YYY.YYY.YYY.YYY where YYY designates the device IP address. Each YYY byte should be in the range of 0 to 255.

An ECOM module's IP address can be determined using NetEdit, an AutomationDirect device configuration utility. To launch NetEdit, select the Device ID Wizard button on the General property group in Device Properties.

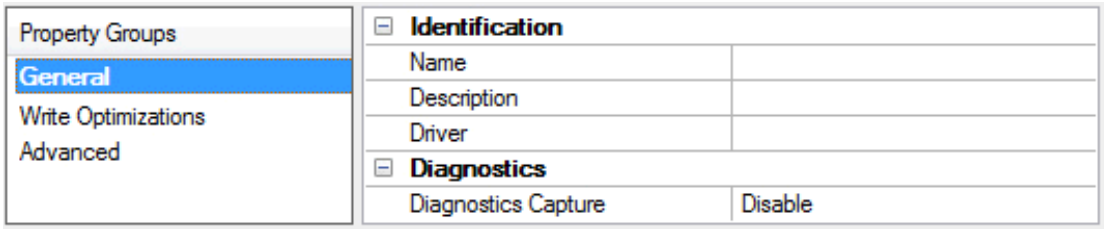
Note: NetEdit has the ability to query the network, configure network parameters and update firmware for ECOM devices.

Automatic Tag Database Generation

Tag Import Settings

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. The property is required for creating a channel.

For information on reserved characters, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in the server help.

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

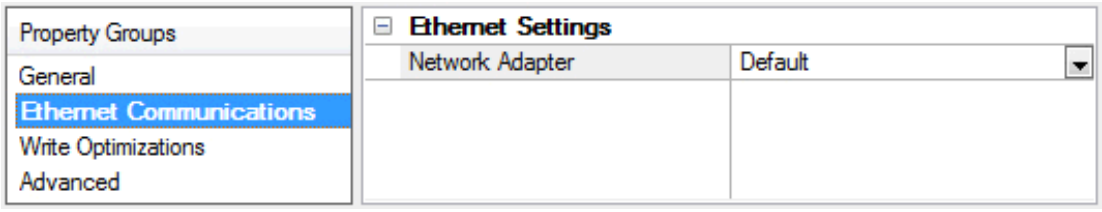
Driver: Selected protocol / driver for this channel. This property specifies the device driver that was selected during channel creation. It is a disabled setting in the channel properties. The property is required for creating a channel.
Note: With the server's online full-time operation, these properties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

Diagnostics

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

Channel Properties — Ethernet Communications

Ethernet Communication can be used to communicate with devices.

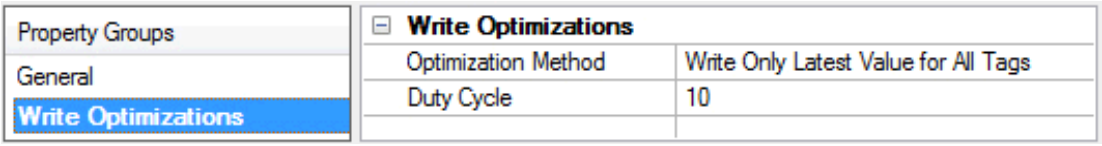


Ethernet Settings

Network Adapter: Specify the network adapter to bind. When left blank or Default is selected, the operating system selects the default adapter.

Channel Properties — Write Optimizations

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



Write Optimizations

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

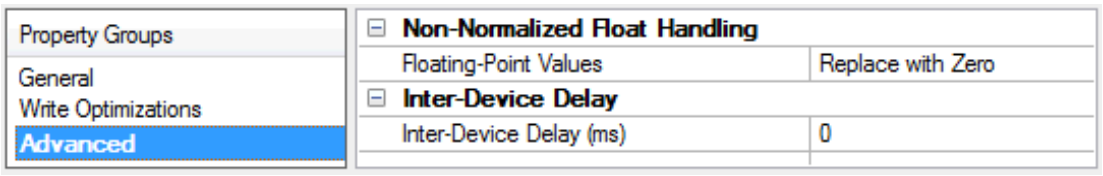
- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing 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 not available if the driver does not support floating point values or if it only supports the option that is displayed. According to the channel's float normalization setting, only real-time driver tags (such as values and arrays) are subject to float normalization. For example, EFM data is not affected by this setting.

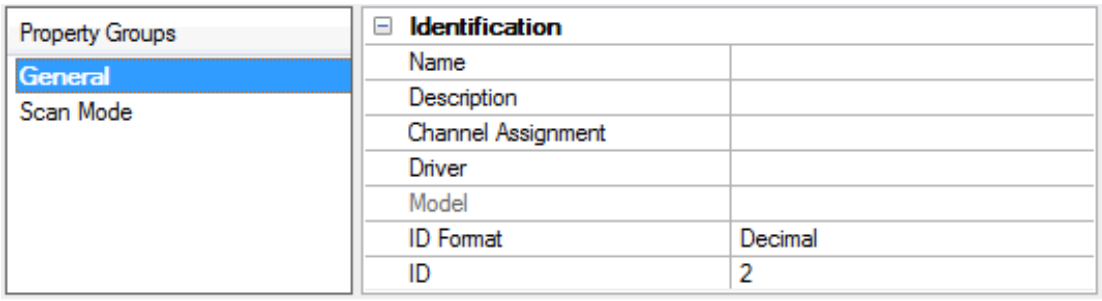
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

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

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

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.

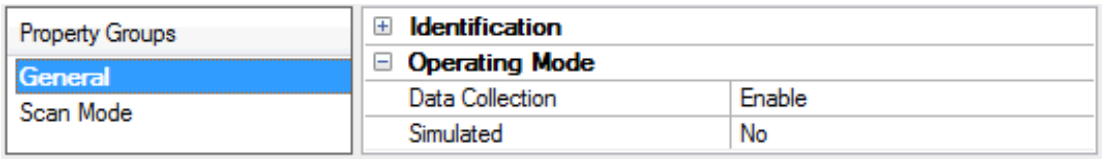
Channel Assignment: User-defined name of the channel to which this device currently belongs.

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 format is set by the driver by default. Options include 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.

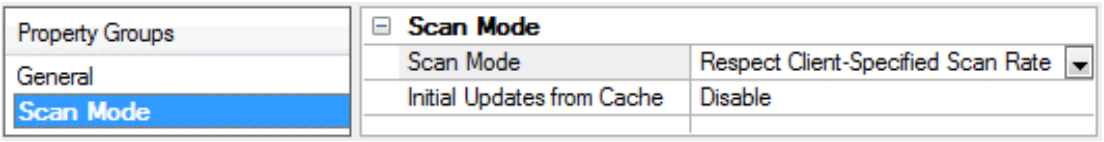
Notes:

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 subscribing clients. Descriptions of the options are:

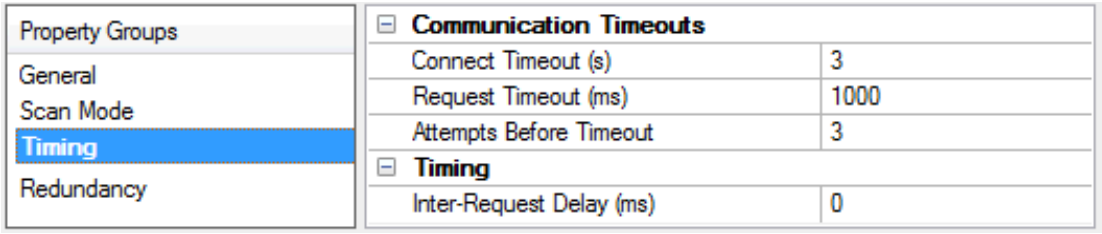
- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  Note: When the server has an active client and items for the device and the scan rate value is increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.

- **Request All Data at Scan Rate**: This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only**: This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the client's responsibility to poll for updates, either by writing to the _DemandPoll tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help*.
- **Respect Tag-Specified Scan Rate**: This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

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

Device Properties — 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: 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.

Attempts Before Timeout: This property specifies 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.

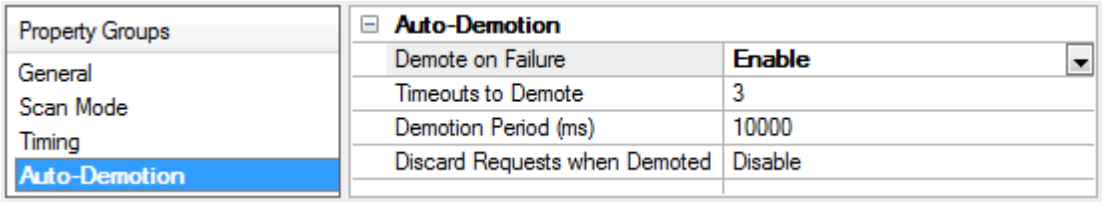
Timing

Inter-Request Delay: 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 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 — Tag Generation

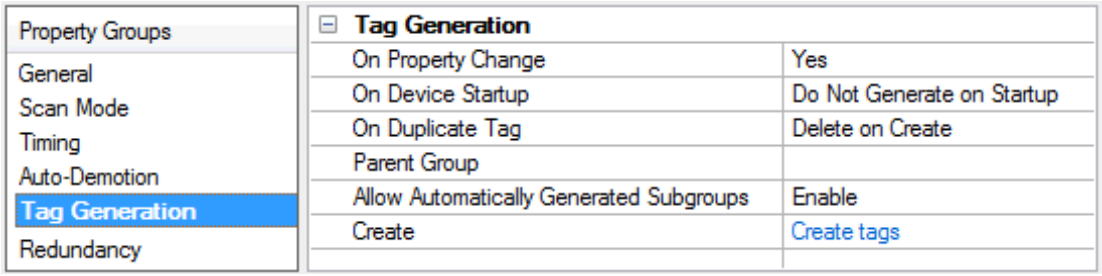
The automatic tag database generation features make setting up an application a plug-and-play operation. Select communications drivers can be configured to automatically build a list of tags that correspond to device-specific data. These automatically generated tags (which depend on the nature of the supporting driver) can be browsed from the clients.

Not all devices and drivers support full automatic tag database generation and not all support the same data types. Consult the data types descriptions or the supported data type lists for each driver for specifics.

If the target device supports its own local tag database, the driver reads the device's tag information and uses the data to generate tags within the server. If the device does not natively support named tags, the driver creates a list of tags based on driver-specific information. An example of these two conditions is as follows:

1. If a data acquisition system supports its own local tag database, the communications driver uses the tag names found in the device to build the server's tags.
2. If an Ethernet I/O system supports detection of its own available I/O module types, the communications driver automatically generates tags in the server that are based on the types of I/O modules plugged into the Ethernet I/O rack.

Note: Automatic tag database generation's mode of operation is completely configurable. *For more information, refer to the property descriptions below.*



On Property Change: If the device supports automatic tag generation when certain properties change, the **On Property Change** option is shown. It is set to **Yes** by default, but it can be set to **No** to control over when tag generation is performed. In this case, the **Create tags** action must be manually invoked to perform tag generation.

On Device Startup: This property specifies when OPC tags are automatically generated. Descriptions of the options are as follows:

- **Do Not Generate on Startup**: This option prevents the driver from adding any OPC tags to the tag space of the server. This is the default setting.
- **Always Generate on Startup**: This option causes the driver to evaluate the device for tag information. It also adds tags to the tag space of the server every time the server is launched.
- **Generate on First Startup**: This option causes the driver to evaluate the target device for tag information the first time the project is run. It also adds any OPC tags to the server tag space as needed.

Note: When the option to automatically generate OPC tags is selected, any tags that are added to the server's tag space must be saved with the project. Users can configure the project to automatically save from the **Tools** | **Options** menu.

On Duplicate Tag: When automatic tag database generation is enabled, the server needs to know what to do with the tags that it may have previously added or with tags that have been added or modified after the communications driver since their original creation. This setting controls how the server handles OPC tags that were automatically generated and currently exist in the project. It also prevents automatically generated tags from accumulating in the server.

For example, if a user changes the I/O modules in the rack with the server configured to **Always Generate on Startup**, new tags would be added to the server every time the communications driver detected a new I/O module. If the old tags were not removed, many unused tags could accumulate in the server's tag space. The options are:

- **Delete on Create**: This option deletes any tags that were previously added to the tag space before any new tags are added. This is the default setting.
- **Overwrite as Necessary**: This option instructs the server to only remove the tags that the communications driver is replacing with new tags. Any tags that are not being overwritten remain in the server's tag space.
- **Do not Overwrite**: This option prevents the server from removing any tags that were previously generated or already existed in the server. The communications driver can only add tags that are completely new.
- **Do not Overwrite, Log Error**: This option has the same effect as the prior option, and also posts an error message to the server's Event Log when a tag overwrite would have occurred.

Note: Removing OPC tags affects tags that have been automatically generated by the communications driver as well as any tags that have been added using names that match generated tags. Users should avoid adding tags to the server using names that may match tags that are automatically generated by the driver.

Parent Group: This property keeps automatically generated tags from mixing with tags that have been entered manually by specifying a group to be used for automatically generated tags. The name of the group can be up to 256 characters. This parent group provides a root branch to which all automatically generated tags are added.

Allow Automatically Generated Subgroups: This property controls whether the server automatically creates subgroups for the automatically generated tags. This is the default setting. If disabled, the server generates the device's tags in a flat list without any grouping. In the server project, the resulting tags are named with the address value. For example, the tag names are not retained during the generation process.
Note: If, as the server is generating tags, a tag is assigned the same name as an existing tag, the system automatically increments to the next highest number so that the tag name is not duplicated. For example, if the generation process creates a tag named "AI22" that already exists, it creates the tag as "AI23" instead.

Create: Initiates the creation of automatically generated OPC tags. If the device's configuration has been modified, **Create tags** forces the driver to reevaluate the device for possible tag changes. Its ability to be accessed from the System tags allows a client application to initiate tag database creation.
Note: **Create tags** is disabled if the Configuration edits a project offline.

Device Properties — Communication Parameters

Port Number

This property specifies the port number that the remote device is configured to use. The default port number is 28784 (0x7070).

Device Properties — Tag Import Settings

Tag Import File

This property is used to specify the exact location of the DirectSoft export file from which tags will be imported. This file will be used when Automatic Tag Database Generation is instructed to create the tag database. The two files types that can be imported are Supported and Not Supported.

Supported Import Files
Program (via Export), .txt extension
Element Documentation (via Export), Standard Format, .csv extension

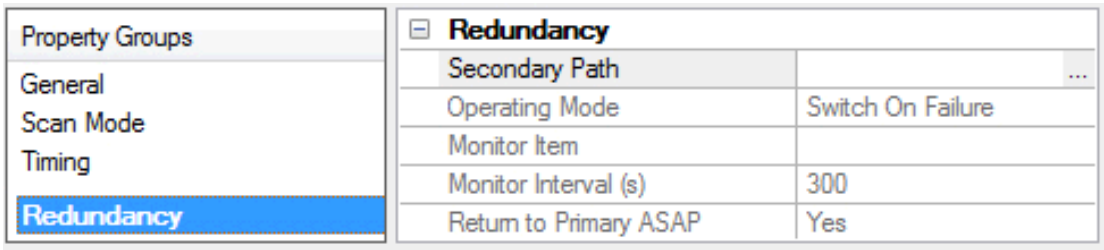
Import Files Not Supported
Element Documentation (via Export), Standard Format, .txt extension
Element Documentation (via Export), EZ-Touch Format, .csv and .txt extension
Element Documentation (auto created), .esd extension
DirectSoft Project, .prj extension

Display Descriptions

When enabled, this option will import tag descriptions. A description will be given to tags with long names that states the original tag name when necessary.

See Also: *Automatic Tag Database Generation*

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.

Automatic Tag Database Generation

The AutomationDirect ECOM Driver generates its tags offline, meaning that a connection to the device is not required to generate tags. Automatic Tag Database Generation is a two-step procedure. First, DirectSoft is used to create a tag export file. Then, the export file is accessed from within the OPC server to generate tags. For more information on creating an export file (*.txt or *.csv) from DirectSoft, refer to **Import Preparation: DirectSoft Steps**. *For more information on configuring the OPC server to use the DirectSoft file for automatic tag database generation, refer to* ***Import Preparation: OPC Server Steps***.

See Also: **Tag Import Settings**

Tag Hierarchy

The tags created by Automatic Tag Database Generation follow a specific hierarchy. The root level groups (or subgroup level of the group specified in "Add generated tags to the following group") are determined by the tag's memory type referenced (such as X, C, V and etc.).

Example
Every variable that is of address type "X" will be placed in a root level group called "X." The only exception applies to counter and timer accumulator addresses CTA and TA, respectively. In these cases, the address is converted to a V-memory reference (TA0 = V0) but the tags generated will be assigned to the root level group CTA or TA, not V. But explicit V-memory references to CTA and TA locations will be assigned to the root level group V as intended.

Import File-to-Server Name Conversions

Leading Underscores

Leading underscores (_) in tag names will be removed. This is required since the server does not accept tag names beginning with an underscore.

Invalid Characters in Name

The only characters allowed in the server tag name are A-Z, a-z, 0-9, and underscore (_). As mentioned above, a tag name cannot begin with an underscore. All other invalid characters encountered will be removed from the tag name.

Importing DirectSoft Elements

This driver uses files generated from DirectSoft via the Program or Element Documentation Export feature to generate the tag database. In both methods, the items of interest are the Elements (such as nickname, address and description) that were created in the DirectSoft Documentation Editor.

How do I create a DirectSoft tag import file (*.txt or *.csv)?

See **Import Preparation: DirectSoft Steps**

How do I configure the OPC Server to use this import file for Automatic Tag Database Generation?

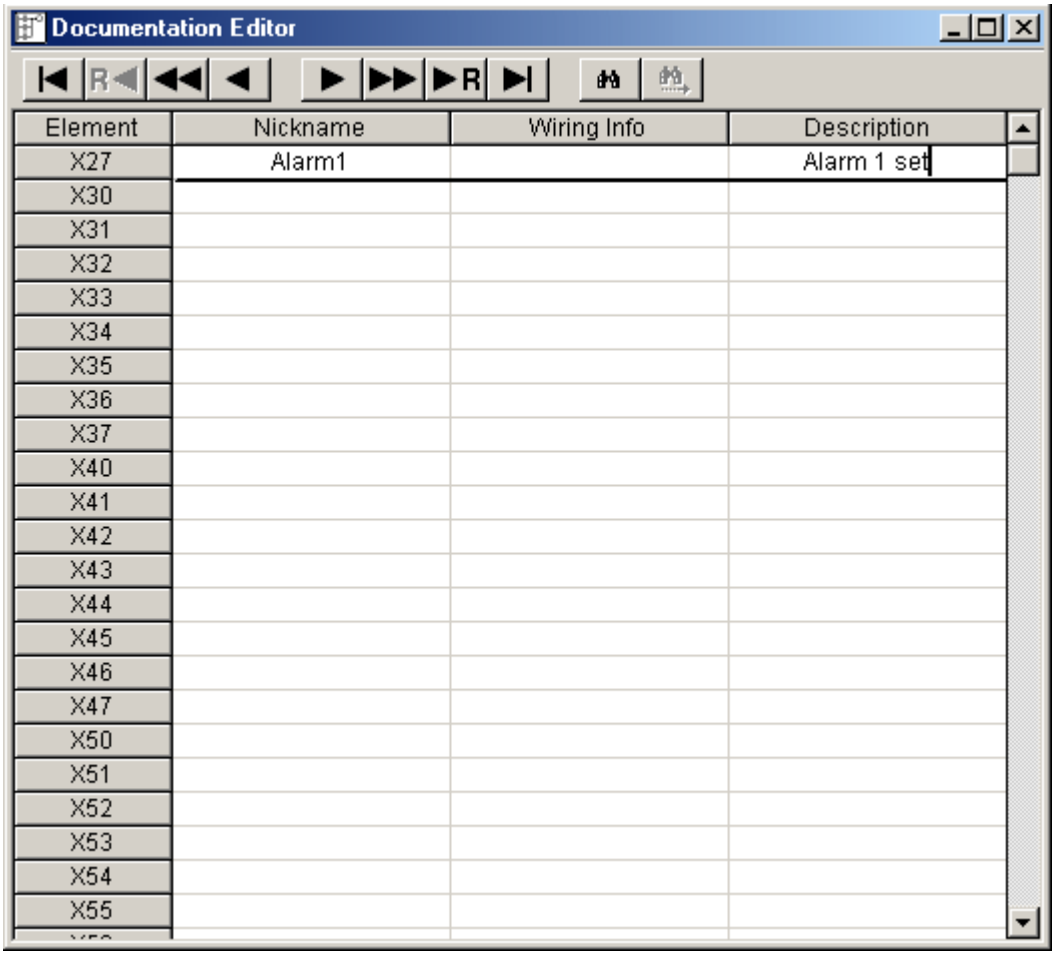
See **Import Preparation: OPC Server Steps**

Import Preparation: DirectSoft Steps

There are two supported methods for generating an export file in DirectSoft for the driver to use as a tag import file: Program Export (*.txt extension) and Element Documentation Export, Standard Format (*.csv extension).

Step 1: Create Nicknames

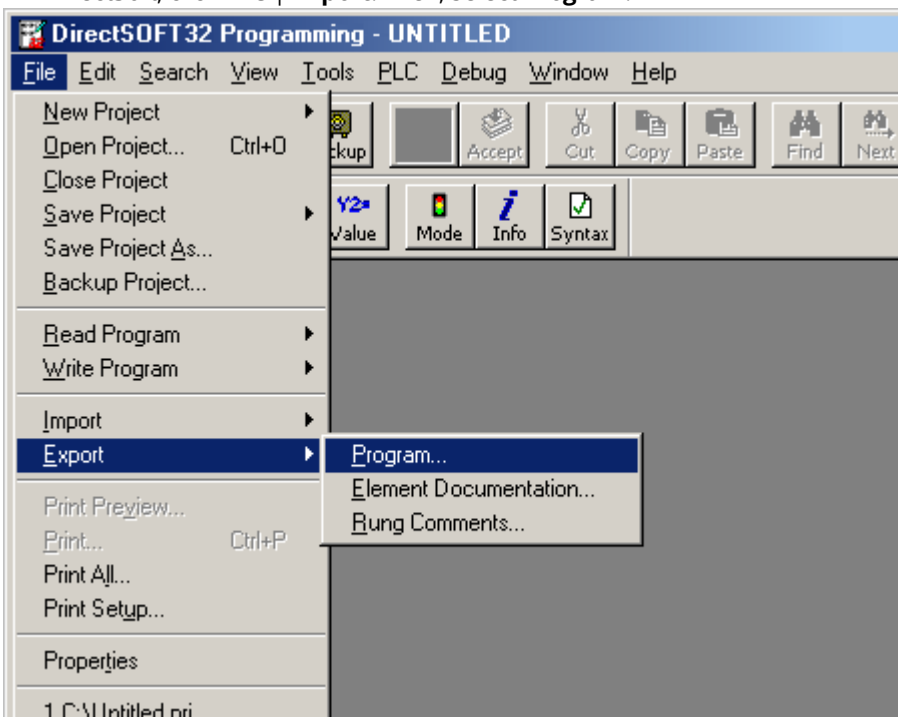
1. Open the DirectSoft project containing the tags (elements) that will be ported to the OPC Server.
2. Launch the **Documentation Editor** by clicking **Tools** | **Documentation Editor**.
3. For each **Memory Reference** of interest, enter both a **Nickname** and a **Description**.



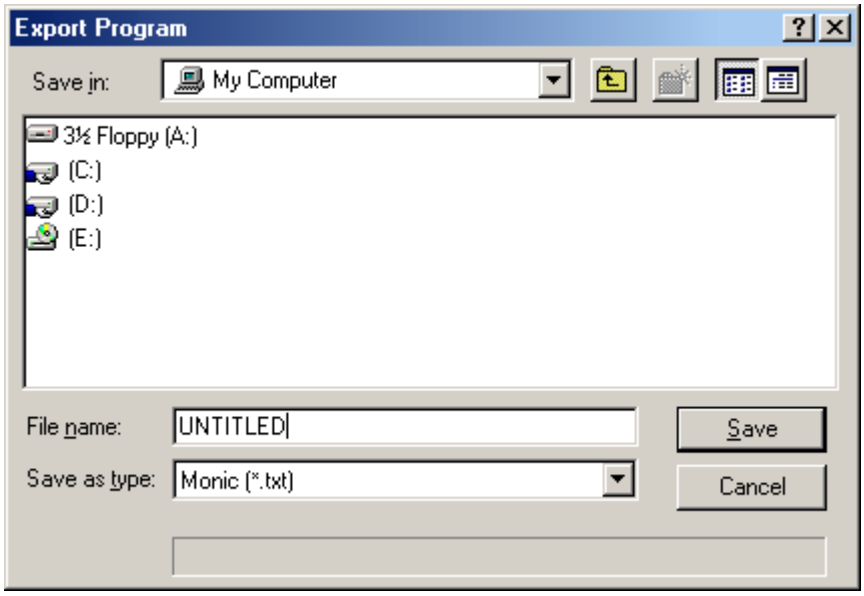
Step 2: Export the Elements

Program Export (.txt)

1. In DirectSoft, click **File** | **Export**. Then, select **Program**.

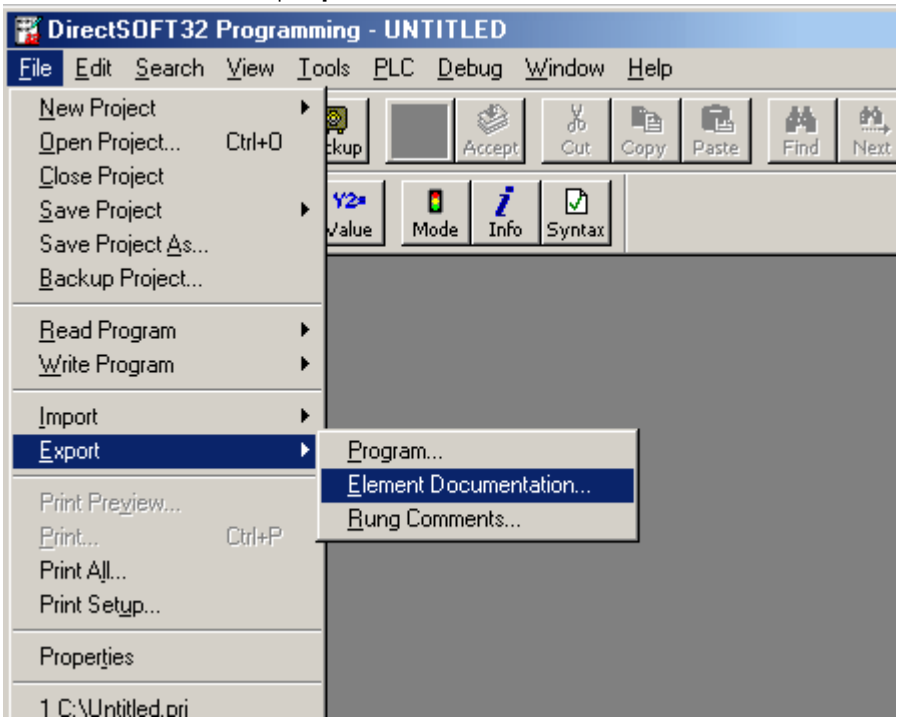


2. The Save dialog will appear, showing the file in text (*.txt) format.

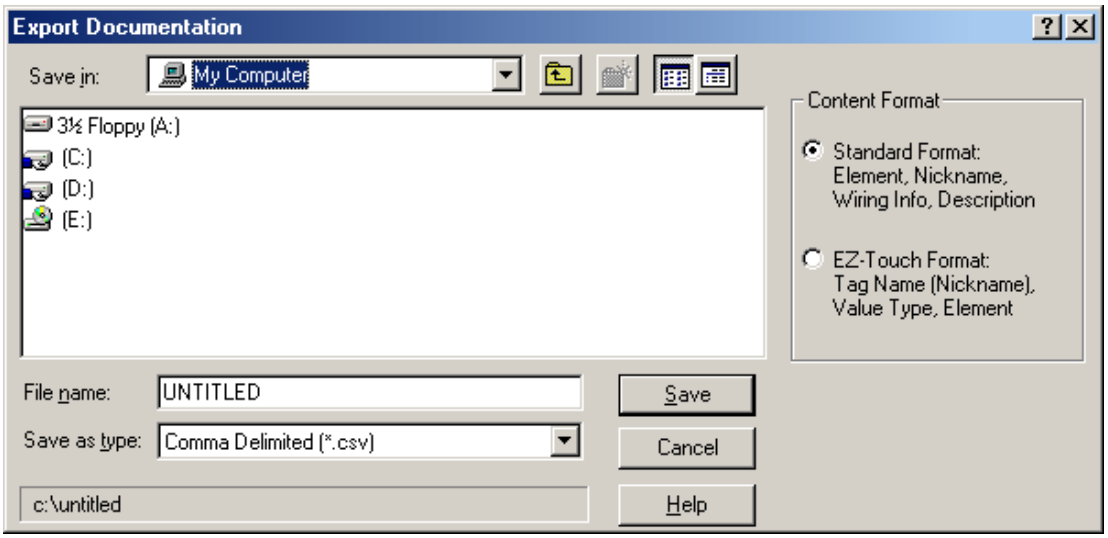


Element Documentation Export (.csv)

1. In DirectSoft, click **File** | **Export**. Then, select **Element Documentation**.



2. The Save dialog will appear. Select **Comma Delimited (*.csv)** and **Standard Format**.



Note: Any other format or file type will not import properly. The file will be in comma separated variable format.

Import Preparation: OPC Server Steps

An export file from DirectSoft must be created before completing the following steps. *For more information, refer to* ***Import Preparation: DirectSoft Steps***.

1. In the driver, click on the device of interest (for which tags will be generated).
2. Next, open its **Device Properties** dialog and select the **Tag Import Settings** property group.
3. Browse and select the location of the DirectSoft export file previously created and then click **Apply**.

4. Next, select the **Tag Generation** property group and configure the Tag Generation properties.
5. Click **Auto Create** to create the tag database.
6. The OPC Server will then attempt to create the tag database while posting messages to the event log on the status of the import. When finished, it will state that the tag import has completed. All elements exported out of DirectSoft will appear in the OPC Server in the layout discussed in **Tag Hierarchy**.

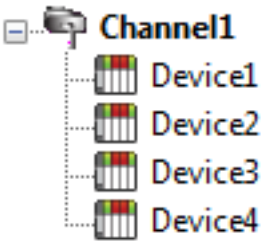
Note: The OPC tags generated are given meaningful names in the OPC server that are based on the variables imported. These tags are also placed in meaningful tag groups to provide a structured and manageable interface. The end result is a well-organized OPC server project that directly reflects the variable import file.

See Also: *Import File-To-Server Name Conversions and Tag Import Settings.*

Optimizing Communications

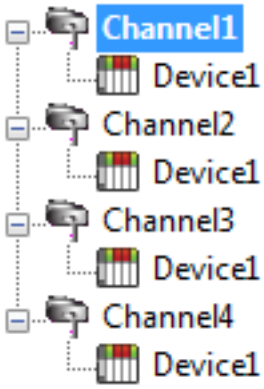
The AutomationDirect ECOM Driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the AutomationDirect ECOM Driver is fast, there are a couple of guidelines that can be used to control and optimize the application and gain maximum performance.

The server refers to communications protocols like AutomationDirect ECOM as a channel. Each channel defined in the application represents a separate path of execution in the server. Once a channel has been defined, a series of devices must then be defined under that channel. Each of these devices represents a single AutomationDirect ECOM from which data will be collected. While this approach to defining the application will provide a high level of performance, it won't take full advantage of the AutomationDirect ECOM Driver or the network. An example of how the application may appear when configured using a single channel is shown below.



Each device appears under a single AutomationDirect ECOM channel. In this configuration, the driver must move from one device to the next as quickly as possible in order to gather information at an effective rate. As more devices are added or more information is requested from a single device, the overall update rate begins to suffer.

If the AutomationDirect ECOM Driver could only define one single channel, then the example shown above would be the only option available; however, the AutomationDirect ECOM Driver can define up to 16 channels. Using multiple channels distributes the data collection workload by simultaneously issuing multiple requests to the network. An example of how the same application may appear when configured using multiple channels to improve performance is shown below.



Each device has now been defined under its own channel. In this new configuration, a single path of execution is dedicated to the task of gathering data from each device. If the application has 16 or fewer devices, it can be optimized exactly how it is shown here.

The performance will improve even if the application has more than 16 devices. While 16 or fewer devices may be ideal, the application will still benefit from additional channels. Although by spreading the device load across all channels will cause the server to move from device to device again, it can now do so with far less devices to process on a single channel.

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
Float	32-bit floating point value. The driver interprets two consecutive registers as a floating-point value by making the second register the high word and the first register the low word.
BCD	Two byte packed BCD Value range is 0-9999. Behavior is undefined for values beyond this range.
String	Null terminated ASCII string. Includes HiLo LoHi byte order selection.

Address Descriptions

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

DL-05 Addressing

The default data types are shown in **bold**.

Memory Type	Reference	Data Types
I/O	X,Y	**Boolean**
Devices	C,SP,T,CT,S	**Boolean**
Data Words	V	Boolean, Word, **Short**, DWord, Long, LBCD, Float, String, **BCD***

*The default is for Timers and Counters only.

Bit Access to V Memory

Bit information can be directly accessed within V memory registers. To access a bit within a V memory register, a bit number can be appended to any V memory address. V memory addressing with bit access would appear as follows: V<xxxxx>.<yy> where xxxxx is the V memory register location and yy is the bit number (0 to 15) within that register. If the V memory location is either a Long or DWord, the bit number yy can be (0 to 31). For more information, refer to the examples below.

Array Support for Data Words

This driver supports array notation for V memory Data Word addresses. To specify an array, append the array size to the address specification as follows: address[array size] or address[rows][cols]. Array size is limited to 63 elements when referenced as a Word, Short, and BCD, and 31 elements when referenced as a DWord, Long, Float, and LBCD.

Examples

1. V1400 [63] @ Word - Array of 63 Words (Maximum allowed) starting at V1400.
2. V1400 [31] @ DWord - Array of 31 DWords (Maximum allowed) starting at V1400.

String Access to Data Words

This driver supports reading and writing V memory Data Words as an ASCII string. When using V memory 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 126 bytes and is entered in place of a bit number. The length must be entered as an even number. The byte order is specified by appending either a "H" or "L" to the address.

Note: The references are in Octal format.

Memory Type	Discrete Memory Reference	Word Memory Reference
Input Points	X0-X377	V40400-V40417
Output Points	Y0-Y377	V40500-V40517
Control Relays	C0-C777	V40600-V40637
Special Relays	SP0-SP777	V41200-V41237
Timer Status Bits	T0-T177	V41100-V41107
Timer Current Values	N/A	V0-V177
Counter Status Bits	CT0-CT177	V41140-V41147
Counter Current Values	N/A	V1000-V1177
Data Words	N/A	V1200-V7377
Data Words String Access HiLo Byte Ordering	N/A	V1200.2H-V7377.126H .Bit is string length, range 2 to 126 bytes.
Data Words String Access LoHi Byte Ordering	N/A	V1200.2L-V7377.126L .Bit is string length, range 2 to 126 bytes.
Data Words (Non-volatile)	N/A	V7400-V7577
Data Words (Non-volatile) String Access HiLo Byte Ordering	N/A	V7400.2H-V7577.126H .Bit is string length, range 2 to 126 bytes.
Data Words (Non-volatile) String Access LoHi Byte Ordering	N/A	V7400.2L-V7577.126L .Bit is string length, range 2 to 126 bytes.
Stages	S0-S377	V41000-V41017
System Parameters	N/A	V7600-V7777

Examples

Example	Description
V40401	Bits 20-27 (octal) of X Input.
V41100	Timer status bits 0-17 (octal).
V1200.1	Bit access to V1200 bit 1.
V1200.100H	String with length 100 and HiLo byte ordering starting at V1200.
V1500.78L	String with length 78 and LoHi byte ordering starting at V1500.

DL-06 Addressing

The default data types are shown in **bold**.

Memory Type	Reference	Data Types
I/O	X,Y,GX,GY	**Boolean**

Memory Type	Reference	Data Types
Devices	C,SP,T,CT,S	**Boolean**
Data Words	V	Boolean, Word, **Short**, DWord, Long, LBCD, Float, String, **BCD***

*The default is for Timers and Counters only.

Bit Access to V Memory

Bit information can be directly accessed within V memory registers. To access a bit within a V memory register, a bit number can be appended to any V memory address. V memory addressing with bit access would appear as follows: V<xxxxx>.<yy> where xxxxx is the V memory register location and yy is the bit number (0 to 15) within that register. If the V memory location is either a Long or DWord, the bit number yy can be (0 to 31). For more information, refer to the examples below.

Array Support for Data Words

This driver supports array notation for V memory Data Word addresses. To specify an array, append the array size to the address specification as follows: address[array size] or address[rows][cols]. Array size is limited to 63 elements when referenced as a Word, Short, and BCD, and 31 elements when referenced as a DWord, Long, Float, and LBCD.

Examples

1. V1200 [63] @ Word - Array of 63 Words (Maximum allowed) starting at V1200.
2. V1200 [31] @ DWord - Array of 31 DWords (Maximum allowed) starting at V1200.

String Access to Data Words

This driver supports reading and writing V memory Data Words as an ASCII string. When using V memory 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 126 bytes and is entered in place of a bit number. The length must be entered as an even number. The byte order is specified by appending either a "H" or "L" to the address.

Note: The references are in Octal format.

Memory Type	Discrete Memory Reference	Word Memory Reference
Input Points	X0-X777	V40400-V40437
Output Points	Y0-Y777	V40500-V40537
Control Relays	C0-C1777	V40600-V40677
Special Relays	SP0-SP777	V41200-V41237
Timer Status Bits	T0-T377	V41100-V41117
Timer Current Values	N/A	V0-V377
Counter Status Bits	CT0-CT177	V41040-V41147
Counter Current Values	N/A	V1000-V1177
Data Words	N/A	V400-V677 V1200-V7377

Memory Type	Discrete Memory Reference	Word Memory Reference
		V10000-V17777
Data Words String Access HiLo Byte Ordering	N/A	V400.2H-V677.126H V1200.2H-V7377.126H V10000.2H-V17777.126H .Bit is string length, range 2 to 126 bytes.
Data Words String Access LoHi Byte Ordering	N/A	V400.2L-V677.126L V1200.2L-V7377.126L V10000.2L-V17777.126L .Bit is string length, range 2 to 126 bytes.
Data Words (Non-volatile)	N/A	V7400-V7577
Data Words (Non-volatile) String Access HiLo Byte Ordering	N/A	V7400.2H-V7577.126H .Bit is string length, range 2 to 126 bytes.
Data Words (Non-volatile) String Access LoHi Byte Ordering	N/A	V7400.2L-V7577.126L .Bit is string length, range 2 to 126 bytes.
Stages	S0-S1777	V41000-V41077
Remote I/O	GX0-GX3777 GY0-GY3777	V40000-V40177 V40200-V40377
System Parameters	N/A	V700-V737 V7600-V7777 V36000-V37777

Examples

Example	Description
V40401	Bits 20-27 (octal) of X Input.
V41100	Timer status bits 0-17 (octal).
V700	System parameter word 700.
V2000.1	Bit access to V2000 bit 1.
V1200.100H	String with length 100 and HiLo byte ordering starting at V1200.
V1500.78L	String with length 78 and LoHi byte ordering starting at V1500.

DL-240 Addressing

The default data types are shown in **bold**.

Memory Type	Reference	Data Types
I/O	X,Y	**Boolean**
Devices	C,SP,T,CT,S	**Boolean**
Data Words	V	Boolean, Word, **Short**, DWord, Long, LBCD, Float, String,

Memory Type	Reference	Data Types
		BCD*

*The default is for Timers and Counters only.

Bit Access to V Memory

Bit information can be directly accessed within V memory registers. To access a bit within a V memory register, a bit number can be appended to any V memory address. V memory addressing with bit access would appear as follows: V<xxxxx>.<yy> where xxxxx is the V memory register location and yy is the bit number (0 to 15) within that register. If the V memory location is either a Long or DWord, the bit number yy can be (0 to 31). For more information, refer to the examples below.

Array Support for Data Words

This driver supports array notation for V memory Data Word addresses. To specify an array, append the array size to the address specification as follows: address[array size] or address[rows][cols]. Array size is limited to 63 elements when referenced as a Word, Short, and BCD, and 31 elements when referenced as a DWord, Long, Float, and LBCD.

Examples

1. V2000 [63] @ Word - Array of 63 Words (Maximum allowed) starting at V2000.
2. V2000 [31] @ DWord - Array of 31 DWords (Maximum allowed) starting at V2000.

String Access to Data Words

This driver supports reading and writing V memory Data Words as an ASCII string. When using V memory 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 126 bytes and is entered in place of a bit number. The length must be entered as an even number. The byte order is specified by appending either a "H" or "L" to the address.

Note: The references are in Octal format.

Memory Type	Discrete Memory Reference	Word Memory Reference
Input Points	X0-X477	V40400-V40423
Output Points	Y0-Y477	V40500-V40523
Control Relays	C0-C377	V40600-V40617
Special Relays	SP0-SP137 SP540-SP617	V41200-V41205 V41226-V41230
Timer Status Bits	T0-T177	V41100-V41107
Timer Current Values	N/A	V0-V177
Counter Status Bits	CT0-CT177	V41140-V41147
Counter Current Values	N/A	V1000-V1177
Data Words	N/A	V2000-V3777
Data Words String Access HiLo Byte Ordering	N/A	V2000.2H-V3777.126H .Bit is string length, range 2 to 126 bytes.
Data Words	N/A	V2000.2L-V3777.126L

Memory Type	Discrete Memory Reference	Word Memory Reference
String Access LoHi Byte Ordering		.Bit is string length, range 2 to 126 bytes.
Data Words (Non-volatile)	N/A	V4000-V4377
Data Words (Non-volatile) String Access HiLo Byte Ordering	N/A	V4000.2H-V4377.126H .Bit is string length, range 2 to 126 bytes.
Data Words (Non-volatile) String Access LoHi Byte Ordering	N/A	V4000.2L-V4377.126L .Bit is string length, range 2 to 126 bytes.
Stages	S0-S777	V41000-V41037
System Parameters	N/A	V7620-V7737 V7746-V7777

Examples

Example	Description
V40500	bits 0-17 (octal) of Y Output.
CT65	Counter contact 65.
S57	Stage control bit 57.
V2000.1	Bit access to V2000 bit 1.
V2000.100H	String with length 100 and HiLo byte ordering starting at V2000.
V2000.78L	String with length 78 and LoHi byte ordering starting at V2000.

DL-250(-1) Addressing

The default data types are shown in **bold**.

Memory Type	Reference	Data Types
I/O	X,Y	**Boolean**
Devices	C,SP,T,CT,S	**Boolean**
Data Words	V	Boolean, Word, **Short**, DWord, Long, LBCD, Float, String, **BCD***

*The default is for Timers and Counters only.

Bit Access to V Memory

Bit information can be directly accessed within V memory registers. To access a bit within a V memory register, a bit number can be appended to any V memory address. V memory addressing with bit access would appear as follows: V<xxxxx>.<yy> where xxxxx is the V memory register location and yy is the bit number (0 to 15) within that register. If the V memory location is either a Long or DWord, the bit number yy can be (0 to 31). For more information, refer to the examples below.

Array Support for Data Words

This driver supports array notation for V memory Data Word addresses. To specify an array, append the array size to the address specification as follows: address[array size] or address[rows][cols]. Array size is limited to 63 elements when referenced as a Word, Short, and BCD, and 31 elements when referenced as a DWord, Long, Float, and LBCD.

Examples

1. V1400 [63] @ Word - Array of 63 Words (Maximum allowed) starting at V1400.
2. V1400 [31] @ DWord - Array of 31 DWords (Maximum allowed) starting at V1400.

String Access to Data Words

This driver supports reading and writing V memory Data Words as an ASCII string. When using V memory 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 126 bytes and is entered in place of a bit number. The length must be entered as an even number. The byte order is specified by appending either a "H" or "L" to the address.

Note: The references are in Octal format.

Memory Type	Discrete Memory Reference	Word Memory Reference
Input Points	X0-X777	V40400-V40437
Output Points	Y0-Y777	V40500-V40537
Control Relays	C0-C1777	V40600-V40677
Special Relays	SP0-SP777	V41200-V41237
Timer Status Bits	T0-T377	V41100-V41117
Timer Current Values	N/A	V0-V377
Counter Status Bits	CT0-CT177	V41140-V41147
Counter Current Values	N/A	V1000-V1177
Data Words	N/A	V1400-V7377 V10000-V17777
Data Words String Access HiLo Byte Ordering	N/A	V1400.2H-V7377.126H V10000.2H-V17777.126H .Bit is string length, range 2 to 126 bytes.
Data Words String Access LoHi Byte Ordering	N/A	V1400.2L-V7377.126L V10000.2L-V17777.126L .Bit is string length, range 2 to 126 bytes.
Stages	S0-S1777	V41000-V41077
System Parameters	N/A	V7400-V7777 V37000-V37777

Examples

Example	Description
V40401	Bits 20-27 (octal) of X Input.
V41100	Timer status bits 0-17 (octal).
V7400	Stage control bit 57.
V1400.1	Bit access to V1400 bit 1.
V1400.100H	String with length 100 and HiLo byte ordering starting at V1400.
V1500.78L	String with length 78 and LoHi byte ordering starting at V1500.

DL-260 Addressing

The default data types are shown in **bold**.

Memory Type	Reference	Data Types
I/O	X,Y,GX,GY	**Boolean**
Devices	C,SP,T,CT,S	**Boolean**
Data Words	V	Boolean, Word, **Short**, DWord, Long, LBCD, Float, String, **BCD***

*The default is for Timers and Counters only.

Bit Access to V Memory

Bit information can be directly accessed within V memory registers. To access a bit within a V memory register, a bit number can be appended to any V memory address. V memory addressing with bit access would appear as follows: V<xxxxx>.<yy> where xxxxx is the V memory register location and yy is the bit number (0 to 15) within that register. If the V memory location is either a Long or DWord, the bit number yy can be (0 to 31). For more information, refer to the examples below.

Array Support for Data Words

This driver supports array notation for V memory Data Word addresses. To specify an array, append the array size to the address specification as follows: address[array size] or address[rows][cols]. Array size is limited to 63 elements when referenced as a Word, Short, and BCD, and 31 elements when referenced as a DWord, Long, Float, and LBCD.

Examples

1. V1400 [63] @ Word - Array of 63 Words (Maximum allowed) starting at V1400.
2. V1400 [31] @ DWord - Array of 31 DWords (Maximum allowed) starting at V1400.

String Access to Data Words

This driver supports reading and writing V memory Data Words as an ASCII string. When using V memory 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 126 bytes and is entered in place of a bit number. The length must be entered as an even number. The byte order is specified by appending either a "H" or "L" to the address.

Note: The references are in Octal format.

Memory Type	Discrete Memory Reference	Word Memory Reference
Input Points	X0-X1777	V40400-V40477
Output Points	Y0-Y1777	V40500-V40577
Control Relays	C0-C3777	V40600-V40777
Special Relays	SP0-SP777	V41200-V41237
Timer Status Bits	T0-T377	V41100-V41117
Timer Current Values	N/A	V0-V377
Counter Status Bits	CT0-CT377	V41140-V41157
Counter Current Values	N/A	V1000-V1377
Data Words	N/A	V400-V777 V1400-V7577 V10000-V35777
Data Words String Access HiLo Byte Ordering	N/A	V400.2H-V777.126H V1400.2H-V7577.126H V10000.2H-V35777.126H .Bit is string length, range 2 to 126 bytes.
Data Words String Access LoHi Byte Ordering	N/A	V400.2L-V777.126L V1400.2L-V7577.126L V10000.2L-V35777.126L .Bit is string length, range 2 to 126 bytes.
Remote I/O	GX0-GX3777 GY0-GY3777	V40000-V40177 V40200-V40377
Stages	S0-S1777	V41000-V41077
System Parameters	N/A	V7600-V7777 V36000-V37777

Examples

Example	Description
V40401	Bits 20-27 (octal) of X Input.
V41100	Timer status bits 0-17 (octal).
V7600	System parameter word 7600.
V2000.1	Bit access to V2000 bit 1.
V400.100H	String with length 100 and HiLo byte ordering starting at V400.
V1400.78L	String with length 78 and LoHi byte ordering starting at V1400.

DL-430 Addressing

The default data types are shown in **bold**.

Memory Type	Reference	Data Types
I/O	X,Y,GX	**Boolean**
Devices	C,SP,T,CT,S	**Boolean**
Data Words	V	Boolean, Word, **Short**, DWord, Long, LBCD, Float, String, **BCD***

*The default is for Timers and Counters only.

Bit Access to V Memory

Bit information can be directly accessed within V memory registers. To access a bit within a V memory register, a bit number can be appended to any V memory address. V memory addressing with bit access would appear as follows: V<xxxxx>.<yy> where xxxxx is the V memory register location and yy is the bit number (0 to 15) within that register. If the V memory location is either a Long or DWord, the bit number yy can be (0 to 31). For more information, refer to the examples below.

Array Support for Data Words

This driver supports array notation for V memory Data Word addresses. To specify an array, append the array size to the address specification as follows: address[array size] or address[rows][cols]. Array size is limited to 63 elements when referenced as a Word, Short, and BCD, and 31 elements when referenced as a DWord, Long, Float, and LBCD.

Examples

1. V1400 [63] @ Word - Array of 63 Words (Maximum allowed) starting at V1400.
2. V1400 [31] @ DWord - Array of 31 DWords (Maximum allowed) starting at V1400.

String Access to Data Words

This driver supports reading and writing V memory Data Words as an ASCII string. When using V memory 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 126 bytes and is entered in place of a bit number. The length must be entered as an even number. The byte order is specified by appending either a "H" or "L" to the address.

Note: The references are in Octal format.

Memory Type	Discrete Memory Reference	Word Memory Reference
Input Points	X0-X477	V40400-V40423
Output Points	Y0-Y477	V40500-V40523
Control Relays	C0-C737	V40600-V40635
Special Relays	SP0-SP137 SP320-SP617	V41200-V41205 V41215-V41230
Timer Status Bits	T0-T177	V41100-V41107
Timer Current Values	N/A	V0-V177
Counter Status Bits	CT0-CT177	V41040-V41147
Counter Current Values	N/A	V1000-V1177

Memory Type	Discrete Memory Reference	Word Memory Reference
Data Words	N/A	V1400-V7377
Data Words String Access HiLo Byte Ordering	N/A	V1400.2H-V7377.126H .Bit is string length, range 2 to 126 bytes.
Data Words String Access LoHi Byte Ordering	N/A	V1400.2L-V7377.126L .Bit is string length, range 2 to 126 bytes.
Remote I/O	GX0-GX737	V40000-V40037
Stages	S0-S577	V41000-V41027
System Parameters	N/A	V7400-V7777

Examples

Example	Description
V40401	Bits 20-27 (octal) of X Input.
T172	Timer contact 172.
GX5	Remote I/O bit 5.
V2000.1	Bit access to V2000 bit 1.
V1400.100H	String with length 100 and HiLo byte ordering starting at V1400.
V1500.78L	String with length 78 and LoHi byte ordering starting at V1500.

DL-440 Addressing

The default data types are shown in **bold**.

Memory Type	Reference	Data Types
I/O	X,Y,GX	**Boolean**
Devices	C,SP,T,CT,S	**Boolean**
Data Words	V	Boolean, Word, **Short**, DWord, Long, LBCD, Float, String, **BCD***

*The default is for Timers and Counters only.

Bit Access to V Memory

Bit information can be directly accessed within V memory registers. To access a bit within a V memory register, a bit number can be appended to any V memory address. V memory addressing with bit access would appear as follows: V<xxxxx>.<yy> where xxxxx is the V memory register location and yy is the bit number (0 to 15) within that register. If the V memory location is either a Long or DWord, the bit number yy can be (0 to 31). For more information, refer to the examples below.

Array Support for Data Words

This driver supports array notation for V memory Data Word addresses. To specify an array, append the array size to the address specification as follows: address[array size] or address[rows][cols]. Array size is

limited to 63 elements when referenced as a Word, Short, and BCD, and 31 elements when referenced as a DWord, Long, Float, and LBCD.

Examples

1. V1400 [63] @ Word - Array of 63 Words (Maximum allowed) starting at V1400.
2. V1400 [31] @ DWord - Array of 31 DWords (Maximum allowed) starting at V1400.

String Access to Data Words

This driver supports reading and writing V memory Data Words as an ASCII string. When using V memory 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 126 bytes and is entered in place of a bit number. The length must be entered as an even number. The byte order is specified by appending either a "H" or "L" to the address.

Note: The references are in Octal format.

Memory Type	Discrete Memory Reference	Word Memory Reference
Input Points	X0-X477	V40400-V40423
Output Points	Y0-Y477	V40500-V40523
Control Relays	C0-C1777	V40600-V40677
Special Relays	SP0-SP137 SP320-SP617 SP620-SP717	V41200-V41205 V41215-V41230 V41231-V41234
Timer Status Bits	T0-T377	V41100-V41117
Timer Current Values	N/A	V0-V377
Counter Status Bits	CT0-CT177	V41040-V41147
Counter Current Values	N/A	V1000-V1177
Data Words	N/A	V1400-V7377 V10000-V17777
Data Words String Access HiLo Byte Ordering	N/A	V1400.2H-V7377.126H V10000.2H-V17777.126H .Bit is string length, range 2 to 126 bytes.
Data Words String Access LoHi Byte Ordering	N/A	V1400.2L-V7377.126L V10000.2L-V17777.126L .Bit is string length, range 2 to 126 bytes.
Remote I/O	GX0-GX1777	V40000-V40077
Stages	S0-S1777	V41000-V41077
System Parameters	N/A	V700-V737 V7400-V7777

Examples

Example	Description
V40401	Bits 20-27 (octal) of X Input.
V41100	Timer status bits 0-17 (octal).
V700	System parameter word 700.
V2000.1	Bit access to V2000 bit 1.
V1400.100H	String with length 100 and HiLo byte ordering starting at V1400.
V1500.78L	String with length 78 and LoHi byte ordering starting at V1500.

DL-450 Addressing

The default data types are shown in **bold**.

Memory Type	Reference	Data Types
I/O	X,Y,GX,GY	**Boolean**
Devices	C,SP,T,CT,S	**Boolean**
Data Words	V	Boolean, Word, **Short**, DWord, Long, LBCD, Float, String, **BCD***

*The default is for Timers and Counters only.

Bit Access to V Memory

Bit information can be directly accessed within V memory registers. To access a bit within a V memory register, a bit number can be appended to any V memory address. V memory addressing with bit access would appear as follows: V<xxxxx>.<yy> where xxxxx is the V memory register location and yy is the bit number (0 to 15) within that register. If the V memory location is either a Long or DWord, the bit number yy can be (0 to 31). For more information, refer to the examples below.

Array Support for Data Words

This driver supports array notation for V memory Data Word addresses. To specify an array, append the array size to the address specification as follows: address[array size] or address[rows][cols]. Array size is limited to 63 elements when referenced as a Word, Short, and BCD, and 31 elements when referenced as a DWord, Long, Float, and LBCD.

Examples

1. V1400 [63] @ Word - Array of 63 Words (Maximum allowed) starting at V1400.
2. V1400 [31] @ DWord - Array of 31 DWords (Maximum allowed) starting at V1400.

String Access to Data Words

This driver supports reading and writing V memory Data Words as an ASCII string. When using V memory 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 126 bytes and is entered in place of a bit number. The length must be entered as an even number. The byte order is specified by appending either a "H" or "L" to the address.

Note: The references are in Octal format.

Memory Type	Discrete Memory Reference	Word Memory Reference
Input Points	X0-X1777	V40400-V40477
Output Points	Y0-Y1777	V40500-V40577
Control Relays	C0-C3777	V40600-V40777
Special Relays	SP0-SP137 SP320-SP717	V41200-V41237
Timer Status Bits	T0-T377	V41100-V41117
Timer Current Values	N/A	V0-V377
Counter Status Bits	CT0-CT377	V41040-V41157
Counter Current Values	N/A	V1000-V1377
Data Words	N/A	V1400-V7377 V10000-V37777
Data Words String Access HiLo Byte Ordering	N/A	V1400.2H-V7377.126H V10000.2H-V37777.126H .Bit is string length, range 2 to 126 bytes.
Data Words String Access LoHi Byte Ordering	N/A	V1400.2L-V7377.126L V10000.2L-V37777.126L .Bit is string length, range 2 to 126 bytes.
Remote I/O	GX0-GX3777 GY0-GY3777	V40000-V40177 V40200-V40377
System Parameters	N/A	V700-V737 V7400-V7777

Examples

Example	Description
V40401	Bits 20-27 (octal) of X Input.
V41100	Timer status bits 0-17 (octal).
V700	System parameter word 700.
V2000.1	Bit access to V2000 bit 1.
V1400.100H	String with length 100 and HiLo byte ordering starting at V1400.
V1500.78L	String with length 78 and LoHi byte ordering starting at V1500.

Error Descriptions

The following 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>'

Device Status Messages

Device Specific Messages

Automatic Tag Database Generation Messages

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:

1. A tag address contains one or more invalid characters.
2. The Bit addressing notation conflicts with the assigned data type.

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.

Device '<device name>' not responding

Error Type:

Serious

Possible Cause:

1. The Ethernet connection between the device and the Host PC is broken.
2. The named device may have been assigned an incorrect IP address.
3. The requested address is not available in the device.
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 ECOM device network.
2. Verify that the IP address given to the named device matches that of the actual device.
3. Verify that the device supports the requested address.
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.

Winsock initialization failed (OS Error = n)

Error Type:

Fatal

OS Error	Possible Solution
10091	Indicates that the underlying network subsystem is not ready for network communication. Wait a few seconds and restart the driver.

OS Error	Possible Solution
10067	Limit on the number of tasks supported by the Windows Sockets implementation has been reached. Close one or more applications that may be using Winsock and restart the driver.

Winsock V1.1 or higher must be installed to use the AutomationDirect ECOM device driver

Error Type:
Fatal

Possible Cause:
The version number of the Winsock DLL found on the system is less than 1.1.

Solution:
Upgrade Winsock to version 1.1 or higher.

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 those that reference invalid locations.

Unable to generate a tag database for device <device name>. Reason: Low memory resources

Error Type:
Warning

Possible Cause:
Memory required for database generation could not be allocated. The process is aborted.

Solution:
Close any unused applications and/or increase the amount of virtual memory and try again.

Unable to generate a tag database for device <device name>. Reason: Import file is invalid or corrupt

Error Type:
Warning

Possible Cause:
The file specified as the Tag Import File in the Database Settings tab in Device Properties is an improperly formatted txt or csv file.

Solution:
If importing Element Documentation, verify that the export file was saved in "Standard Format" with a .csv extension. If problem resumes, try re-exporting the file.

See Also:
Importing DirectSoft Elements

Index

D

E

F

G

I

L

M

N

O

P

R

S

T

U

W