

# AutomationDirect DirectNET Driver

© 2025 PTC Inc. All Rights Reserved.

# Table of Contents

<b>AutomationDirect DirectNET Driver</b>	<b>1</b>
<b>Table of Contents</b>	<b>2</b>
AutomationDirect DirectNET Driver	4
Overview	4
<b>Setup</b>	<b>4</b>
Channel Properties – General	5
Tag Counts	6
Channel Properties – Serial Communications	6
Channel Properties – Write Optimizations	8
Channel Properties – Advanced	9
Device Properties – General	10
Operating Mode	10
Tag Counts	11
Device Properties – Scan Mode	11
Device Properties – Timing	12
Device Properties – Auto-Demotion	13
Device Properties – Tag Generation	13
Device Properties – Tag Import Settings	15
Device Properties – Redundancy	15
<b>Automatic Tag Database Generation</b>	<b>16</b>
Tag Hierarchy	16
Import File-to-Server Name Conversions	16
Importing DirectSoft Elements	16
Import Preparation: DirectSoft Steps	17
Import Preparation: OPC Server Steps	19
<b>Data Types Description</b>	<b>21</b>
<b>Address Descriptions</b>	<b>22</b>
DL-05 Addressing	22
DL-06 Addressing	23
DL-230 Addressing	24
DL-240 Addressing	25
DL-250(-1) Addressing	26
DL-260 Addressing	27
DL-330 Addressing	29
DL-340 Addressing	29
DL-350 Addressing	30
DL-430 Addressing	31
DL-440 Addressing	32
DL-450 Addressing	33
<b>Error Descriptions</b>	<b>35</b>

Missing address .....	35
Device address '<address>' contains a syntax error .....	35
Address '<address>' is out of range for the specified device or register .....	36
Device address '<address>' is not supported by model '<model name>' .....	36
Data Type '<type>' is not valid for device address '<address>' .....	36
Device address '<address>' is Read Only .....	36
COMn does not exist .....	36
Error opening COMn .....	37
COMn is in use by another application .....	37
Unable to set comm properties on COMn .....	37
Communications error on '<channel name>' [<error mask>] .....	37
Device '<device name>' not responding .....	38
Unable to write to '<address>' on device '<device name>' .....	38
Bad address in block [<start address> to <end address>] on device '<device name>' .....	38
Unable to generate a tag database for device <device name> .....	39
Unable to generate a tag database for device <device name> .....	39
<b>Index .....</b>	<b>40</b>

## AutomationDirect DirectNET Driver

Help version 1.041

### CONTENTS

#### [Overview](#)

What is the AutomationDirect DirectNET Driver?

#### [Setup](#)

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

#### [Automatic Tag Database Generation](#)

How can I easily configure tags for the AutomationDirect DirectNET Driver?

#### [Data Types Description](#)

What data types does this driver support?

#### [Address Descriptions](#)

How do I address a data location on an AutomationDirect DirectNet device?

#### [Error Descriptions](#)

What error messages does the AutomationDirect DirectNET Driver produce?

### Overview

---

The AutomationDirect DirectNET Driver provides a reliable way to connect AutomationDirect DirectNet controllers to OPC Client applications; including HMI, SCADA, Historian, MES, ERP and countless custom applications. This driver is intended for use with Automation Direct Logic Programmable Logic Controllers, also known as PLCDirect and Koyo.

### Setup

---

#### Supported Devices

The following PLCs are supported.

DL-05  
DL-06  
DL-230  
DL-240  
DL-250(-1)  
DL-260  
DL-330  
DL-340  
DL-350  
DL-430  
DL-440  
DL-450

#### Communication Protocol

Koyo DirectNet Hex Mode Protocol

#### Supported Communication Parameters\*

Baud Rate - 1200, 2400, 9600, 19200 or 38400  
Parity - None, Even or Odd  
Data Bits - 5, 6, 7 or 8  
Stop Bits - 1 or 2

\*Some of these configurations may not be supported by all devices.

#### Ethernet Encapsulation

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

## 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 90 per channel. Valid Device IDs range from 1 to 90.

## Flow Control

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

**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.

## Automatic Tag Database Generation

### Tag Import Settings

## 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.

Property Groups	<input type="checkbox"/> Identification <input checked="" type="checkbox"/> General <input type="checkbox"/> Write Optimizations <input type="checkbox"/> Advanced	
	Name	
	Description	
	Driver	
	<input type="checkbox"/> Diagnostics <input type="checkbox"/> Tag Counts	
	Diagnostics Capture	Disable
	Static Tags	10

## 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.

**Note:** 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 does not support diagnostics.

● For more information, refer to *Communication Diagnostics in the server help*.

## Diagnostics

**Diagnostics Capture:** When enabled, this option allows the usage of statistics tags that provide feedback to client applications regarding the operation of the channel. 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 *Statistics Tags in the 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](#) or [Ethernet 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](#)).

Property Groups		
General		
<b>Serial Communications</b>		
Write Optimizations		
Advanced		

<input type="checkbox"/> <b>Connection Type</b>		
Physical Medium		COM Port
<input type="checkbox"/> <b>Serial Port Settings</b>		
COM ID		39
Baud Rate		19200
Data Bits		8
Parity		None
Stop Bits		1
Flow Control		RTS Always
<input type="checkbox"/> <b>Operational Behavior</b>		
Report Communication Errors		Enable
Close Idle Connection		Enable
Idle Time to Close (s)		15

## Connection Type

**Physical Medium:** Choose the type of hardware device for data communications. Options include Modem, Ethernet Encapsulation, 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. **Ethernet Encap.:** Select if Ethernet Encapsulation is used for communications, which displays the [Ethernet Settings](#) section.
5. **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.

## Ethernet Settings

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

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

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

## Modem Settings

- **Modem:** Specify the installed modem to be used for communications.
- **Connect Timeout:** Specify the amount of time to wait for connections to be established before failing a read or write. The default is 60 seconds.
- **Modem Properties:** Configure the modem hardware. When clicked, it opens vendor-specific modem properties.
- **Auto-Dial:** Enables the automatic dialing of entries in the Phonebook. The default is Disable. *For more information, refer to "Modem Auto-Dial" in the server help.*
- **Report Communication Errors:** Enable or disable reporting of low-level communications errors. When enabled, low-level errors are posted to the Event Log as they occur. When disabled, these same errors are not posted even though normal request failures are. The default is Enable.
- **Close Idle Connection:** Choose to close the modem connection when there are no longer any tags being referenced by a client on the channel. The default is Enable.
- **Idle Time to Close:** Specify the amount of time that the server waits once all tags have been removed before closing the modem connection. The default is 15 seconds.

## Operation with no Communications

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

## Channel Properties – Write Optimizations

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.

Property Groups	<input checked="" type="checkbox"/> <b>Write Optimizations</b>	
General	Optimization Method	Write Only Latest Value for All Tags
<b>Write Optimizations</b>	Duty Cycle	10

## Write Optimizations

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

- **Write All Values for All Tags:** This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing data to the device as quickly as possible. This mode ensures that everything written from the client applications is sent to the target device. This mode should be selected if the write operation order or the write item's content must uniquely be seen at the target device.
- **Write Only Latest Value for Non-Boolean Tags:** Many consecutive writes to the same value can accumulate in the write queue due to the time required to actually send the data to the device. If the server updates a write value that has already been placed in the write queue, far fewer writes are needed to reach the same final output value. In this way, no extra writes accumulate in the server's queue. When the user



stops moving the slide switch, the value in the device is at the correct value at virtually the same time. As the mode states, any value that is not a Boolean value is updated in the server's internal write queue and sent to the device at the next possible opportunity. This can greatly improve the application performance.

● **Note:** This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.

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

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

● **Note:** It is recommended that the application be characterized for compatibility with the write optimization enhancements before being used in a production environment.

## Channel Properties – Advanced

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

Property Groups	
General	
Write Optimizations	
<b>Advanced</b>	

<input checked="" type="checkbox"/> <b>Non-Normalized Float Handling</b>	
Floating-Point Values	Replace with Zero
<input checked="" type="checkbox"/> <b>Inter-Device Delay</b>	
Inter-Device Delay (ms)	0

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

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

● **Note:** This property is 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.

## 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.

Property Groups <b>General</b>	[-] <b>Identification</b>	
	Name	
	Description	
	Channel Assignment	
	Driver	
	Model	
	ID Format	Decimal
	ID	2

### 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.

**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.

### Operating Mode

Property Groups <b>General</b>	[+] <b>Identification</b>	
	[-] <b>Operating Mode</b>	
	Data Collection	Enable
	Simulated	No

**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

Property Groups	[-] Identification	
General	[-] Operating Mode	
	[-] Tag Counts	
	Static Tags	130

**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.

Property Groups	[-] Scan Mode	
General	Scan Mode	Respect Client-Specified Scan Rate ▼
Scan Mode	Initial Updates from Cache	Disable

**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.

Property Groups	☐ <b>Communication Timeouts</b>	
General	Connect Timeout (s)	3
Scan Mode	Request Timeout (ms)	1000
<b>Timing</b>	Attempts Before Timeout	3

### 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. 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.


Property Groups General Scan Mode <b>Timing</b>	<input checked="" type="checkbox"/> <b>Timing</b>	
	Inter-Request Delay (ms)	0

## 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.

Property Groups General Scan Mode Timing <b>Auto-Demotion</b>	<input checked="" type="checkbox"/> <b>Auto-Demotion</b>	
	Demote on Failure	Enable <input type="button" value="v"/>
	Timeouts to Demote	3
	Demotion Period (ms)	10000
	Discard Requests when Demoted	Disable

**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.*

Property Groups	<b>Tag Generation</b>	
General	On Device Startup	Do Not Generate on Startup
Timing	On Duplicate Tag	Delete on Create
Auto-Demotion	Parent Group	
<b>Tag Generation</b>	Allow Automatically Generated Subgroups	Enable
Communications	Create	Create tags
Redundancy		

**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. To invoke via the Configuration API service, access `/config/v1/project/channels/{name}/devices/{name}/services/TagGeneration`.

**On Device Startup:** Specify 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 – Tag Import Settings

Property Groups	<input checked="" type="checkbox"/> <b>Tag Import Settings</b>	
General	Tag Import File	*.csv
Scan Mode	Display Descriptions	Enable
Timing		
Auto-Demotion		
Tag Generation		
<b>Tag Import Settings</b>		
Redundancy		

### Tag Import File

This property is used to enter the exact location of the DirectSoft export file from which tags will be imported. It is this file that will be used when Automatic Tag Database Generation is instructed to create the tag database. There are two types of import files: 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 property imports tag descriptions. If necessary, a description is given to tags with long names that state their original tag name.

● **See Also:** [Automatic Tag Database Generation](#)

## Device Properties – Redundancy

Property Groups	<input checked="" type="checkbox"/> <b>Redundancy</b>	
General	Secondary Path	Channel.Device1 ...
Scan Mode	Operating Mode	Switch On Failure
Timing	Monitor Item	
Auto-Demotion	Monitor Interval (s)	300
<b>Redundancy</b>	Return to Primary ASAP	Yes

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

---

### Generating a New Tag Database

The AutomationDirect DirectNET Driver uses files generated from DirectSoft via the **Program** or **Element Documentation Export** feature to generate the tag database. Before attempting to automatically create the tag database, a file must be created, selected and exported from DirectSoft. Tags are generated offline, meaning that a connection to the device is not required. Instead, the device driver imports a tag file generated from the DirectSoft export file to create the tag database. For more information, refer to [Importing DirectSoft Elements](#) and [Tag Import Settings](#).

Thus, there are two steps necessary for automatic tag database generation. First, an export file (\*.txt or \*.csv) must be created from DirectSoft. Second, tags must be generated based on that DirectSoft export file within the OPC server. For more information, refer to [Import Preparation: DirectSoft Steps](#) and [Import Preparation: OPC Server Steps](#).

### Generating Tag Database While Preserving Previously Generated Tag Databases

Under certain circumstances, multiple imports into the server are required to import all tags of interest. This is the case with importing VersaPro System variables and non-System variables into the same OPC server project. In the Database Creation property group, under Device Properties, click on the selection **Perform the following action**. The options available are "Delete on create," "Overwrite as necessary," "Do not overwrite" and "Do not overwrite, log error." After the first OPC server import/database creation is done, check that the action is set to "Do not overwrite" or "Do not overwrite, log error" for future imports. This will allow tags to be imported without deleting or overwriting ones that have been imported previously.

## Tag Hierarchy

---

The tags created via 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.). For 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 (nickname, address and description) created in the DirectSoft Documentation Editor.

### See Also:

- For information on how to create a DirectSoft tag import file (\*.txt or \*.csv), refer to [Import Preparation: DirectSoft Steps](#).
- For information on how to configure the OPC server to use this import file for Automatic Tag Database Generation, refer to [Import Preparation: OPC Server Steps](#).

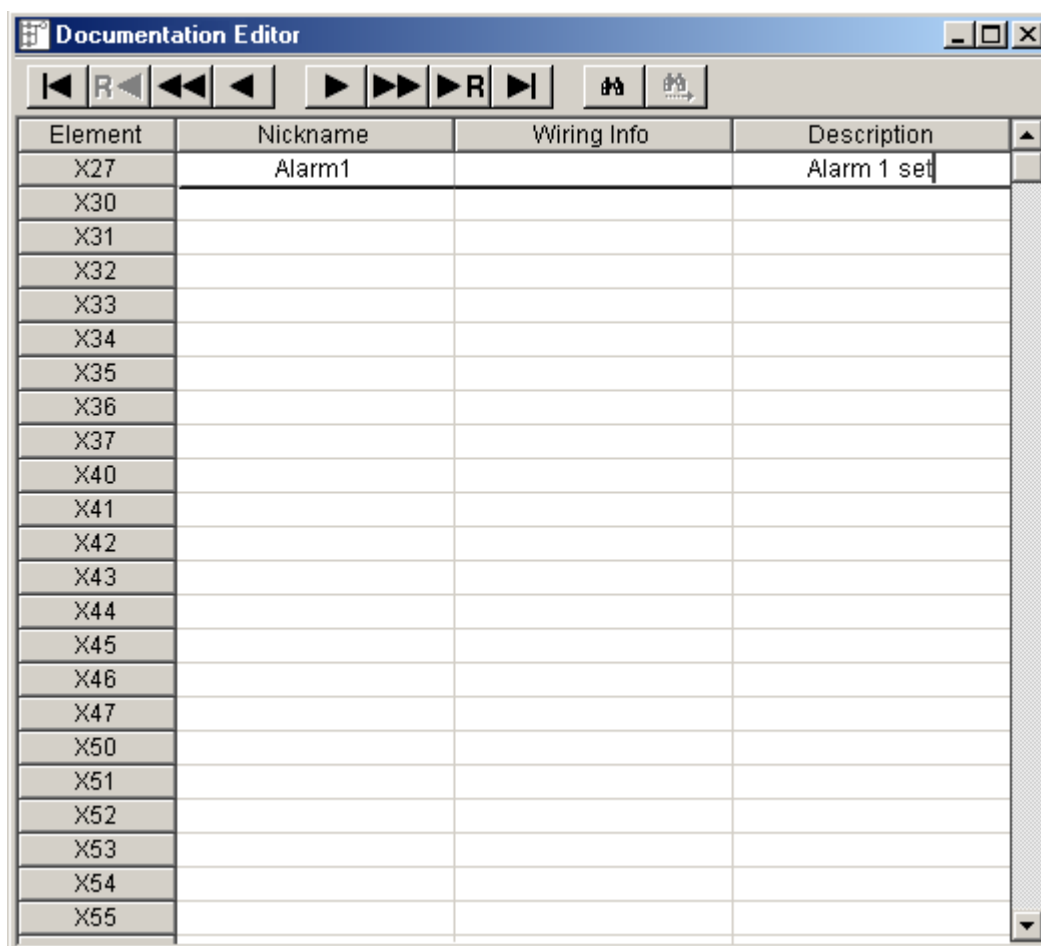


## Import Preparation: DirectSoft Steps

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).

### Creating Nicknames

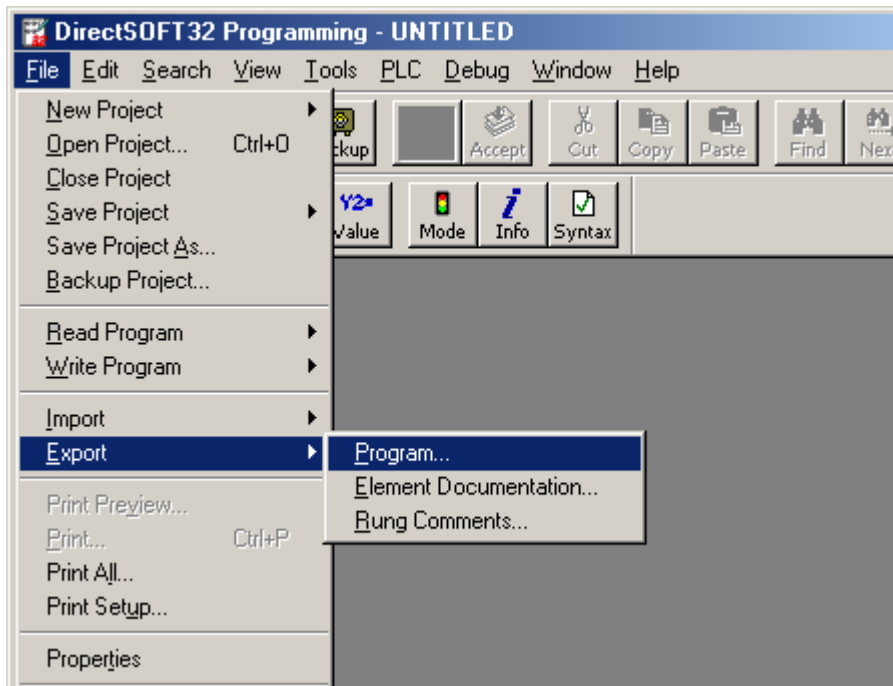
1. Open the **DirectSoft** project containing the tags (elements) that will be ported to the OPC server.
2. Open the **Documentation Editor** by clicking **Menu | Tools | Documentation Editor**.
3. Enter a nickname and description for each memory reference of interest.



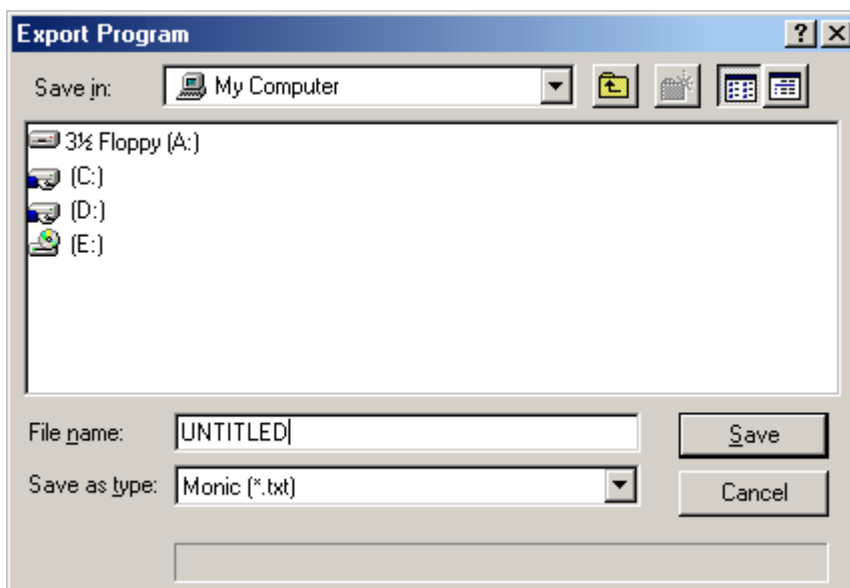
### Exporting the Elements Through Program Export (.txt)

1. Open **DirectSoft**.
2. On **Menu**, click **File | Export**.

3. Select **Program**.



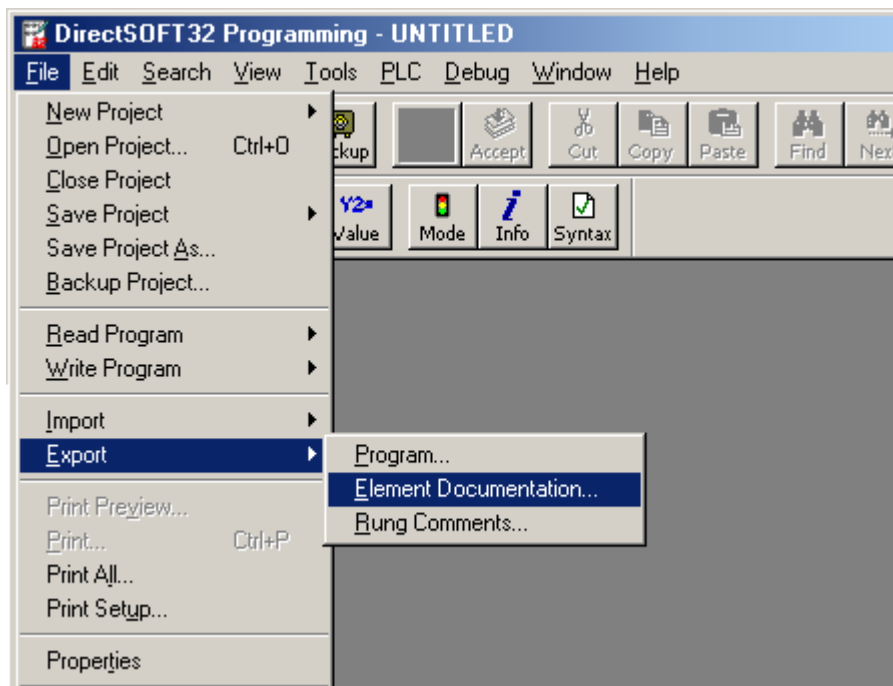
4. The **Save** dialog will show the file in text (\*.txt) format.



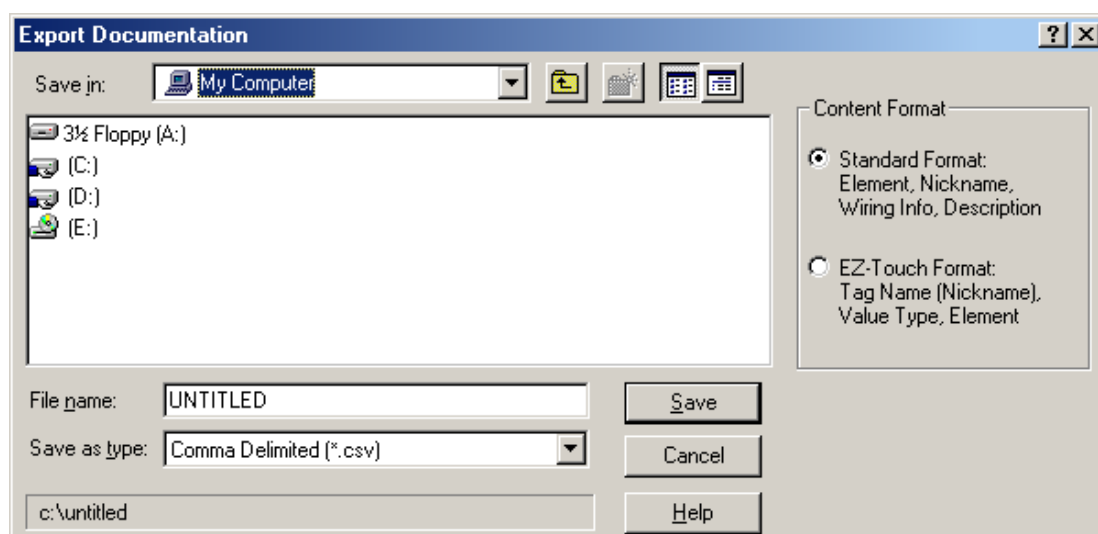
### Exporting the Elements Through Element Documentation Export (.csv)

1. Open **DirectSoft**.
2. On **Menu**, click **File | Export**.

3. Select **Element Documentation**.



4. In the **Save** dialog, select **Comma Delimited (\*.csv)** and **Standard Format**. Any other format or file type will not import properly. The file will be in **comma-separated variable format**.




• See Also: [Import Preparation: OPC Server Steps](#)

## Import Preparation: OPC Server Steps

• **Important:** An export file from DirectSoft must be created before the following OPC server steps can be completed. For more information, refer to [Import Preparation: DirectSoft Steps](#).

1. In the driver, open up the **Device Properties** for the device of interest.
2. Select the **Tag Import Settings** property group.
3. Browse and select the location of the newly created DirectSoft export file. Click **Apply**.

4. Select the **Database Creation** property group.
5. Configure the Database Creation settings.
6. Click **Auto Create** to create the tag database.

 **Note:** 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 been completed. All elements exported out of DirectSoft will appear in the OPC server in the layout discussed in [Tag Hierarchy](#).

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

 **See Also:** [Tag Import Settings](#), [Import File-to-Server Name Conversions](#)

## Data Types Description

Data Type	Description
Boolean	Single bit
Byte	Unsigned 8-bit value bit 0 is the low bit bit 7 is the high bit
Char	Signed 8-bit value bit 0 is the low bit bit 6 is the high bit bit 7 is the sign 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.
LBCD	Four-byte packed BCD Value range is 0-99999999. Behavior is undefined for values beyond this range.

## 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](#)  
[DL-06](#)  
[DL-230](#)  
[DL-240](#)  
[DL-250\(-1\)](#)  
[DL-260](#)  
[DL-330](#)  
[DL-340](#)  
[DL-350](#)  
[DL-430](#)  
[DL-440](#)  
[DL-450](#)

## DL-05 Addressing

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

I/O	X, Y	<b>Boolean</b>
Devices	C, SP, T, CT, S	<b>Boolean</b>
Data Words	V	Word, <b>Short</b> , DWord, Long, Float, LBCD <b>BCD</b> (default 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<xxxx>.<yy> where xxxx 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).

## Array Support for V Memory

This driver supports array notation for V memory 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 128 elements when referenced as a Word, Short, and BCD, and 64 elements when referenced as a DWord, Long, Float, and LBCD.

## Examples

1. V0 [3][4] @ BCD - Array of 12 Timer Values.
2. V1200 [128] @ Word - Array of 128 Words (Maximum allowed) starting at V1200.
3. V1200 [64] @ DWord - Array of 64 DWords (Maximum allowed) starting at V1200.

## Address Specifications

All address ranges are specified in Octal.

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Input Points*	X<xxx>	X0-X377	V<xxxxx>	V40400-V40417
Output Points	Y<xxx>	Y0-Y377	V<xxxxx>	V40500-V40517
Control Relays	C<xxx>	C0-C777	V<xxxxx>	V40600-V40637
Special Relays	SP<xxx>	SP0-SP777	V<xxxxx>	V41200-V41237
Timer Status Bits	T<xxx>	T0-T177	V<xxxxx>	V41100-V41107

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Timer Current Values	None	None	V<xxxxx>	V0-V177
Counter Status Bits	CT<xxx>	CT0-CT177	V<xxxxx>	V41140-V41147
Counter Current Values	None	None	V<xxxxx>	V1000-V1177
Data Words	None	None	V<xxxxx>	V1200-V7377
Data Words Non-Volatile	None	None	V<xxxxx>	V7400-V7577
Stages	S<xxx>	S0-S377	V<xxxxx>	V41000-V41017
System Parameters	None	None	V<xxxxx>	V7600-V7777

\*Read Only.

### Examples

1. V40401 - bits 20 - 27 (octal) of X Input.
2. V41100 - Timer status bits 0 - 17 (octal).
3. V7600 - System parameter word 7600.
4. V2000.1 - Bit access to V2000 bit 1.
5. V2000.30 @ Long - Bit access to V2000 as a Long bit 30.

## DL-06 Addressing

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

I/O	X, Y, GX, GY	<b>Boolean</b>
Devices	C, SP, T, CT, S	<b>Boolean</b>
Data Words	V	Word, <b>Short</b> , DWord, Long, Float, LBCD <b>BCD</b> (default 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).

### Array Support for V Memory

This driver supports array notation for V memory 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 128 elements when referenced as a Word, Short, and BCD, and 64 elements when referenced as a DWord, Long, Float, and LBCD.

### Examples

1. V0 [3][4] @ BCD - Array of 12 Timer Values.
2. V1200 [128] @ Word - Array of 128 Words (Maximum allowed) starting at V1200.
3. V1200 [64] @ DWord - Array of 64 DWords (Maximum allowed) starting at V1200.

### Address Specifications

All address ranges are specified in Octal.

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Input Points*	X<xxx>	X0-X777	V<xxxxx>	V40400-V40437
Output Points	Y<xxx>	Y0-Y777	V<xxxxx>	V40500-V40537
Control Relays	C<xxx>	C0-C1777	V<xxxxx>	V40600-V40677
Special Relays	SP<xxx>	SP0-SP777	V<xxxxx>	V41200-V41237
Timer Status Bits	T<xxx>	T0-T377	V<xxxxx>	V41100-V41117
Timer Current Values	None	None	V<xxxxx>	V0-V377
Counter Status Bits	CT<xxx>	CT0-CT177	V<xxxxx>	V41040-V41147
Counter Current Values	None	None	V<xxxxx>	V1000-V1177
Data Words	None	None	V<xxxxx>	V400-V677 V1200-V7377 V10000-V17777
Data Words Non-Volatile	None	None	V<xxxxx>	V7400-V7577
Stages	S<xxx>	S0-S1777	V<xxxxx>	V41000-V41077
Remote I/O	GX<xxx> GY<xxx>	GX0-GX3777 GY0-GY3777	V<xxxxx>	V40000-V40177 V40200-V40377
System Parameters	None	None	V<xxxxx>	V700-V777 V7600-V7777 V36000-V37777

\*Read Only.

### Examples

1. V40401 - bits 20 - 27 (octal) of X Input.
2. V41100 - Timer status bits 0 - 17 (octal).
3. V700 - System parameter word 700.
4. V2000.1 - Bit access to V2000 bit 1.
5. V2000.30@Long - Bit access to V2000 as a Long bit 30.

## DL-230 Addressing

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

I/O	X, Y	<b>Boolean</b>
Devices	C, SP, T, CT, S	<b>Boolean</b>
Data Words	V	Word, <b>Short</b> , DWord, Long, Float, LBCD <b>BCD</b> (default 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).

### Array Support for V Memory

This driver supports array notation for V memory 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 128 elements when referenced as a Word, Short, and BCD, and 64 elements when referenced as a DWord, Long, Float, and LBCD.



## Examples

1. V0 [3][4] @ BCD - Array of 12 Timer Values.
2. V2000 [128] @ Word - Array of 128 Words (Maximum allowed) starting at V2000.
3. V2000 [64] @ DWord - Array of 64 DWords (Maximum allowed) starting at V2000.

## Address Specifications

All address ranges are specified in Octal.

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Input Points*	X<xxx>	X0-X177	V<xxxxx>	V40400-V40407
Output Points	Y<xxx>	Y0-Y177	V<xxxxx>	V40500-V40507
Control Relays	C<xxx>	C0-C377	V<xxxxx>	V40600-V40617
Special Relays	SP<xxx>	SP0-SP117 SP540-SP577	V<xxxxx>	V41200-V41204 V41226-V41227
Timer Status Bits	T<xxx>	T0-T77	V<xxxxx>	V41100-V41103
Timer Current Values	None	None	V<xxxxx>	V0-V77
Counter Status Bits	CT<xxx>	CT0-CT77	V<xxxxx>	V41140-V41143
Counter Current Values	None	None	V<xxxxx>	V1000-V1077
Data Words	None	None	V<xxxxx>	V2000-V2377
Data Words Non-Volatile	None	None	V<xxxxx>	V4000-V4177
Stages	S<xxx>	S0-S377	V<xxxxx>	V41000-V41017
System Parameters	None	None	V<xxxxx>	V7620-V7647 V7750-V7777

\*Read Only.

## Examples

1. V40500 bits 0 - 17 (octal) of Y Output.
2. CT65 Counter contact 65.
3. S57 Stage control bit 57.
4. V2000.1 Bit access to V2000 bit 1.
5. V2000.30@Long Bit access to V2000 as a Long bit 30.

## DL-240 Addressing

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

I/O	X, Y	<b>Boolean</b>
Devices	C, SP, T, CT, S	<b>Boolean</b>
Data Words	V	Word, <b>Short</b> , DWord, Long, Float, LBCD <b>BCD</b> (default 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).

## Array Support for V Memory

This driver supports array notation for V memory 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 128 elements when referenced as a Word, Short, and BCD, and 64 elements when referenced as a DWord, Long, Float, and LBCD.

### Examples

1. V0 [3][4] @ BCD- Array of 12 Timer Values.
2. V2000 [128] @ Word - Array of 128 Words (Maximum allowed) starting at V2000.
3. V2000 [64] @ DWord - Array of 64 DWords (Maximum allowed) starting at V2000.

### Address Specifications

All address ranges are specified in Octal.

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Input Points	X<xxx>	X0-X177	V<xxxxx>	V40400-V40407
Output Points	Y<xxx>	Y0-Y177	V<xxxxx>	V40500-V40507
Control Relays	C<xxx>	C0-C377	V<xxxxx>	V40600-V40617
Special Relays	SP<xxx>	SP0-SP137	V<xxxxx>	V41200-V41205
		SP540-SP617		V41226-V41230
Timer Status Bits	T<xxx>	T0-T177	V<xxxxx>	V41100-V41107
Timer Current Values	None	None	V<xxxxx>	V0-V177
Counter Status Bits	CT<xxx>	CT0-CT177	V<xxxxx>	V41140-V41147
Counter Current Values	None	None	V<xxxxx>	V1000-V1177
Data Words	None	None	V<xxxxx>	V2000-V3777
Data Words Non-Volatile	None	None	V<xxxxx>	V4000-V4377
Stages	S<xxx>	S0-S777	V<xxxxx>	V41000-V41037
System Parameters	None	None	V<xxxxx>	V7620-V7737
				V7746-V7777

\*Read Only.

### Examples

1. V40500 bits 0 - 17 (octal) of Y Output.
2. CT165 Counter contact 165.
3. S57 Stage control bit 57.
4. V2000.1 Bit access to V2000 bit 1.
5. V2000.30@Long Bit access to V2000 as a Long bit 30.

### DL-250(-1) Addressing

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

I/O	X, Y	<b>Boolean</b>
Devices	C, SP, T, CT, S	<b>Boolean</b>
Data Words	V	Word, <b>Short</b> , DWord, Long, Float, LBCD <b>BCD</b> (default 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<xxxx>.<yy> where xxxx 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).

### Array Support for V Memory

This driver supports array notation for V memory 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 128 elements when referenced as a Word, Short, and BCD, and 64 elements when referenced as a DWord, Long, Float, and LBCD.

### Examples

1. V0 [3][4] @ BCD- Array of 12 Timer Values.
2. V1400 [128] @ Word - Array of 128 Words (Maximum allowed) starting at V1400.
3. V1400 [64] @ DWord - Array of 64 DWords (Maximum allowed) starting at V1400.

### Address Specifications

All address ranges are specified in Octal.

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Input Points*	X<xxx>	X0-X777	V<xxxx>	V40400-V40437
Output Points	Y<xxx>	Y0-Y777	V<xxxx>	V40500-V40537
Control Relays	C<xxx>	C0-C1777	V<xxxx>	V40600-V40677
Special Relays	SP<xxx>	SP0-SP777	V<xxxx>	V41200-V41237
Timer Status Bits	T<xxx>	T0-T377	V<xxxx>	V41100-V41117
Timer Current Values	None	None	V<xxxx>	V0-V377
Counter Status Bits	CT<xxx>	CT0-CT177	V<xxxx>	V41140-V41147
Counter Current Values	None	None	V<xxxx>	V1000-V1177
Data Words	None	None	V<xxxx>	V1400-V7377 V10000-V17777
Stages	S<xxx>	S0-S1777	V<xxxx>	V41000-V41077
System Parameters	None	None	V<xxxx>	V7400-V7777 V37000-V37777

\*Read Only.

### Examples

1. V40401 bits 20 - 27 (octal) of X Input.
2. V41100 Timer status bits 0 - 17 (octal).
3. V7400 System parameter word 7400.
4. V2000.1 Bit access to V2000 bit 1.
5. 2000.30@Long Bit access to V2000 as a Long bit 30.

### DL-260 Addressing

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

I/O	X, Y	<b>Boolean</b>
Devices	C, SP, T, CT, S	<b>Boolean</b>
Data Words	V	Word, <b>Short</b> , DWord, Long, Float, LBCD

		BCD (default 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<xxxx>.<yy> where xxxx 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).

### Array Support for V Memory

This driver supports array notation for V memory 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 128 elements when referenced as a Word, Short, and BCD, and 64 elements when referenced as a DWord, Long, Float, and LBCD.

### Examples

1. V0 [3][4] @ BCD- Array of 12 Timer Values.
2. V1400 [128] @ Word - Array of 128 Words (Maximum allowed) starting at V1400.
3. V1400 [64] @ DWord - Array of 64 DWords (Maximum allowed) starting at V1400.

### Address Specifications

All address ranges are specified in Octal.

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Input Points*	X<xxx>	X0-X1777	V<xxxxx>	V40400-V40477
Output Points	Y<xxx>	Y0-Y1777	V<xxxxx>	V40500-V40577
Control Relays	C<xxx>	C0-C3777	V<xxxxx>	V40600-V40777
Special Relays	SP<xxx>	SP0-SP777	V<xxxxx>	V41200-V41237
Timer Status Bits	T<xxx>	T0-T377	V<xxxxx>	V41100-V41117
Timer Current Values	None	None	V<xxxxx>	V0-V377
Counter Status Bits	CT<xxx>	CT0-CT377	V<xxxxx>	V41140-V41157
Counter Current Values	None	None	V<xxxxx>	V1000-V1377
Data Words	None	None	V<xxxxx>	V400-V777 V1400-V7377 V10000-V35777
Stages	S<xxx>	S0-S1777	V<xxxxx>	V41000-V41077
Remote I/O	GX<xxx>	GX0-GX3777	V<xxxxx>	V40000-V40177
	GY<xxx>	GY0-GY3777	V<xxxxx>	V40200-V40377
System Parameters	None	None	V<xxxxx>	V7600-V7777 V36000-V37777

\*Read Only.

### Examples

1. V40401 bits 20 - 27 (octal) of X Input.
2. V41100 Timer status bits 0 - 17 (octal).
3. V7400 System parameter word 7400.
4. V2000.1 Bit access to V2000 bit 1.
5. V2000.30@Long Bit access to V2000 as a Long bit 30.

## DL-330 Addressing

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

<b>I/O</b>	<b>IO</b>	<b>Boolean</b>
<b>Devices</b>	C, SP, CT, SR	<b>Boolean</b>
<b>8-bit Registers</b>	R	Byte, Char, Word, <b>Short</b> , DWord, Long, Float, BCD, LBCD <b>BCD</b> (default for Timers and Counters only)

## Address Specifications

All address ranges are specified in Octal.

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Input / Output Points	IO<xxx>	IO0-IO157 IO700-IO767	R<xxx>	R0-R15 R70-R76
Control Relays	C<xxx>	C160-C373	R<xxx>	R16-R37
Special Relays	SP<xxx>	SP374-SP377 SP770-SP777	R<xxx>	R37 R77
Timer / Counter Status Bits	CT<xxx>	CT600-CT677	None	None
Timer / Counter Pre-set Values	None	None	R<xxx>	R564-R573
Timer / Counter Current Values	None	None	R<xxx>	R600-R677
Data Words*	None	None	R<xxx>	R400-R563
Shift Registers	SR<xxx>	SR400 - SR577	None	None
Special Registers	None	None	R<xxx>	R574-V577

\*Only even registers are writable.

## Examples

1. R37 bits 374 - 377 (octal) of the special relays.
2. IO157 bit 157 of the I/O points.
3. R16 bits 160 - 167 of the control relays.

## DL-340 Addressing

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

<b>I/O</b>	<b>IO</b>	<b>Boolean</b>
<b>Devices</b>	C, SP, CT, SR	<b>Boolean</b>
<b>Data Words</b>	R	Byte, Char, Word, <b>Short</b> , DWord, Long, Float, BCD, LBCD <b>BCD</b> (default for Timers and Counters only)

## Address Specifications

All address ranges are specified in Octal.

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Input / Output Points	IO<xxx>	IO0-IO157	R<xxx>	R0-R15

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
		IO700-IO767		R70-R76
Control Relays	C<xxx>	C160-C373 C1000-C1067	R<xxx>	R16-R37 R100-R106
Special Relays	SP<xxx>	SP374-SP377 SP770-SP777 SP1070-SP1077	R<xxx>	R37 R77 R107
Timer / Counter Status Bits	CT<xxx>	CT600-CT677	None	None
Timer / Counter Pre-set Values	None	None	R<xxx>	R564-R573
Timer / Counter Current Values	None	None	R<xxx>	R600-R677
Data Words*	None	None	R<xxx>	R400-R563 R700-R767
Shift Registers	SR<xxx>	SR400-SR577	None	None
Special Registers	None	None	R<xxx>	R574-V577 R770-R777

\*Only even registers are writable.

## Examples

1. R37 bits 374 - 377 (octal) of the special relays.
2. IO157 bit 157 of the I/O points.
3. R100 bits 1000 - 1007 of the control relays.

## DL-350 Addressing

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

I/O	X, Y	<b>Boolean</b>
Devices	C, SP, T, CT, S	<b>Boolean</b>
Data Words	V	Word, <b>Short</b> , DWord, Long, Float, LBCD <b>BCD</b> (default 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<xxxx>.<yy> where xxxx 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).

## Array Support for V Memory

This driver supports array notation for V memory 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 128 elements when referenced as a Word, Short, and BCD, and 64 elements when referenced as a DWord, Long, Float, and LBCD.

## Examples

1. V0 [3][4] @ BCD- Array of 12 Timer Values
2. V1400 [128] @ Word - Array of 128 Words (Maximum allowed) starting at V1400
3. V1400 [64] @ DWord - Array of 64 DWords (Maximum allowed) starting at V1400

## Address Specifications

All address ranges are specified in Octal.

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Input Points*	X<xxx>	X0-X777	V<xxxxx>	V40400-V40437
Output Points	Y<xxx>	Y0-Y777	V<xxxxx>	V40500-V40537
Control Relays	C<xxx>	C0-C1777	V<xxxxx>	V40600-V40677
Special Relays	SP<xxx>	SP0-SP777	V<xxxxx>	V41200-V41237
Timer Status Bits	T<xxx>	T0-T377	V<xxxxx>	V41100-V41117
Timer Current Values	None	None	V<xxxxx>	V0-V377
Counter Status Bits	CT<xxx>	CT0-CT177	V<xxxxx>	V41140-V41147
Counter Current Values	None	None	V<xxxxx>	V1000-V1177
Data Words	None	None	V<xxxxx>	V1400-V7377 10000-V17777
Stages	S<xxx>	S0-S1777	V<xxxxx>	V41000-V41077
System Parameters	None	None	V<xxxxx>	V7400-V7777 V36000-V37777

\*Read Only.

## Examples

1. V40401 bits 20 - 27 (octal) of X Input.
2. V41100 Timer status bits 0 - 17 (octal).
3. V7400 System parameter word 7400.
4. V2000.1 Bit access to V2000 bit 1.
5. V2000.30@Long Bit access to V2000 as a Long bit 30.

## DL-430 Addressing

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

I/O	X, Y, GX	<b>Boolean</b>
Devices	C, SP, T, CT, S	<b>Boolean</b>
Data Words	V	Word, <b>Short</b> , DWord, Long, Float, LBCD <b>BCD</b> (default 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).

## Array Support for V Memory

This driver supports array notation for V memory 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 128 elements when referenced as a Word, Short, and BCD, and 64 elements when referenced as a DWord, Long, Float, and LBCD.

## Examples

1. V0 [3][4] @ BCD- Array of 12 Timer Values.
2. V1400 [128] @ Word - Array of 128 Words (Maximum allowed) starting at V1400.
3. V1400 [64] @ DWord - Array of 64 DWords (Maximum allowed) starting at V1400.

## Address Specifications

All address ranges are specified in Octal.

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Input Points	X<xxx>	X0-X477	V<xxxxx>	V40400-V40423
Output Points	Y<xxx>	Y0-Y477	V<xxxxx>	V40500-V40523
Control Relays	C<xxx>	C0-C737	V<xxxxx>	V40600-V40635
Special Relays	SP<xxx>	SP0-SP137 SP320-SP617	V<xxxxx>	V41200-V41205 V41215-V41230
Timer Status Bits	T<xxx>	T0-T177	V<xxxxx>	V41100-V41107
Timer Current Values	None	None	V<xxxxx>	V0-V177
Counter Status Bits	CT<xxx>	CT0-CT177	V<xxxxx>	V41040-V41147
Counter Current Values	None	None	V<xxxxx>	V1000-V1177
Data Words	None	None	V<xxxxx>	V1400-V7377
Stages	S<xxx>	S0-S577	V<xxxxx>	V41000-V41027
Remote I/O	GX<xxx>	GX0-GX737	V<xxxxx>	V40000-V40037
System Parameters	None	None	V<xxxxx>	V7400-V7777

\*Read Only.

## Examples

1. V40401 bits 20 - 27 (octal) of X Input.
2. T172 Timer contact 172.
3. GX5 Remote I/O bit 5.
4. V2000.1 Bit access to V2000 bit 1.
5. V2000.30@Long Bit access to V2000 as a Long bit 30.

## DL-440 Addressing

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

I/O	X, Y GX	<b>Boolean</b>
Devices	C, SP, T, CT, S	<b>Boolean</b>
Data Words	V	Word, <b>Short</b> , DWord, Long, Float, LBCD <b>BCD</b> (default 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).

## Array Support for V Memory



This driver supports array notation for V memory 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 128 elements when referenced as a Word, Short, and BCD, and 64 elements when referenced as a DWord, Long, Float, and LBCD.

### Examples

1. V0 [3][4] @ BCD- Array of 12 Timer Values.
2. V1400 [128] @ Word - Array of 128 Words (Maximum allowed) starting at V1400.
3. V1400 [64] @ DWord - Array of 64 DWords (Maximum allowed) starting at V1400.

### Address Specifications

All address ranges are specified in Octal.

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Input Points*	X<xxx>	X0-X477	V<xxxxx>	V40400-V40423
Output Points	Y<xxx>	Y0-Y477	V<xxxxx>	V40500-V40523
Control Relays	C<xxx>	C0-C1777	V<xxxxx>	V40600-V40677
Special Relays	SP<xxx>	SP0-SP137	V<xxxxx>	V41200-V41205
		SP320-SP617		V41215-V41230
		SP620-SP717		V41231-V41234
Timer Status Bits	T<xxx>	T0-T377	V<xxxxx>	V41100-V41117
Timer Current Values	None	None	V<xxxxx>	V0-V377
Counter Status Bits	CT<xxx>	CT0-CT177	V<xxxxx>	V41040-V41147
Counter Current Values	None	None	V<xxxxx>	V1000-V1177
Data Words	None	None	V<xxxxx>	V1400-V7377
				V10000-V17777
Stages	S<xxx>	S0-S1777	V<xxxxx>	V41000-V41077
Remote I/O	GX<xxx>	GX0-GX1777	V<xxxxx>	V40000-V40077
System Parameters	None	None	V<xxxxx>	V700-V737
				V7400-V7777

\*Read Only.

### Examples

1. V40401 bits 20 - 27 (octal) of X Input.
2. V41100 Timer status bits 0 - 17 (octal).
3. V700 System parameter word 700.
4. V2000.1 Bit access to V2000 bit 1.
5. V2000.30@Long Bit access to V2000 as a Long bit 30.

### DL-450 Addressing

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

I/O	X, Y, GX, GY	<b>Boolean</b>
Devices	C, SP, T, CT, S	<b>Boolean</b>
Data Words	V	Word, <b>Short</b> , DWord, Long, Float, LBCD

		BCD (default 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<xxxx>.<yy> where xxxx 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).

### Array Support for V Memory

This driver supports array notation for V memory 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 128 elements when referenced as a Word, Short, and BCD, and 64 elements when referenced as a DWord, Long, Float, and LBCD.

### Examples

1. V0 [3][4] @ BCD- Array of 12 Timer Values.
2. V1400 [128] @ Word - Array of 128 Words (Maximum allowed) starting at V1400.
3. V1400 [64] @ DWord - Array of 64 DWords (Maximum allowed) starting at V1400.

### Address Specifications

All address ranges are specified in Octal.

Memory Type	Discrete Memory Reference	Discrete Memory Range	Word Memory Reference	Word Memory Range
Input Points*	X<xxx>	X0-X1777	V<xxxxx>	V40400-V40477
Output Points	Y<xxx>	Y0-Y1777	V<xxxxx>	V40500-V40577
Control Relays	C<xxx>	C0-C3777	V<xxxxx>	V40600-V40777
Special Relays	SP<xxx>	SP0-SP137 SP320-SP717	V<xxxxx>	V41200-V41237
Timer Status Bits	T<xxx>	T0-T377	V<xxxxx>	V41100-V41117
Timer Current Values	None	None	V<xxxxx>	V0-V377
Counter Status Bits	CT<xxx>	CT0-CT377	V<xxxxx>	V41040-V41157
Counter Current Values	None	None	V<xxxxx>	V1000-V1377
Data Words	None	None	V<xxxxx>	V1400-V7377 V10000-V37777
Stages	S<xxx>	S0-S1777	V<xxxxx>	V41000-V41077
Remote I/O	GX<xxx>	GX0-GX3777	V<xxxxx>	V40000-V40177
	GY<xxx>	GY0-GY3777	V<xxxxx>	V40200-V40377
System Parameters	None	None	V<xxxxx>	V700-V737 V7400-V7777

\*Read Only.

### Examples

1. V40401 bits 20 - 27 (octal) of X Input.
2. V41100 Timer status bits 0 - 17 (octal).
3. V700 System parameter word 700.
4. V2000.1 Bit access to V2000 bit 1.
5. V2000.30@Long Bit access to V2000 as a Long bit 30.

# 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>'](#)

[Data Type '<type>' is not valid for device address '<address>'](#)

[Device address '<address>' is Read Only](#)

## Serial Communications

[COMn does not exist](#)

[Error opening COMn](#)

[COMn is in use by another application](#)

[Unable to set comm properties on COMn](#)

[Communications error on '<channel name>' \[<error mask>\]](#)

## Device Status Messages

[Device '<device name>' is not responding](#)

[Unable to write to '<address>' on device '<device name>'](#)

## Device Specific Messages

[Bad address in block \[<start address> to <end address>\] on device '<device name>'](#)

## Automatic Tag Database Generation Messages

[Unable to generate a tag database for device <device name>](#)

[Unable to generate a tag database for 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 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.

---

**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.

---

**Unable to set comm properties on COMn**

---

**Error Type:**

Fatal

**Possible Cause:**

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

**Solution:**

Verify the serial properties and make any necessary changes.

---

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

---

**Error Type:**

Serious

**Error Mask Definitions:**

**B** = Hardware break detected.

**F** = Framing error.

**E** = I/O error.

**O** = Character buffer overrun.

**R** = RX buffer overrun.

**P** = Received byte parity error.

**T** = TX buffer full.

**Possible Cause:**

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

**Solution:**

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

---

**Device '<device name>' not responding**

---

**Error Type:**

Serious

**Possible Cause:**

1. The serial connection between the device and the Host PC is broken.
2. The communications properties for the serial connection are incorrect.
3. The named device may have been assigned an incorrect Network ID.
4. The requested address is not available in the device.
5. 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 the specified communications properties match those of the device.
3. Verify the Network ID given to the named device matches that of the actual device.
4. Verify that the device supports the requested address.
5. 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 properties for the serial connection are incorrect.
3. The named device may have been assigned an incorrect Network ID.

**Solution:**

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

---

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

---

**Error Type:**

Serious

**Possible Cause:**

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

**Solution:**

Verify the tags assigned to addresses in the specified range on the device and eliminate ones that reference invalid locations.

---

**Unable to generate a tag database for device <device name>****Error Type:**

Warning

**Possible Cause:**

Memory required for database generation could not be allocated. The process is cancelled.

**Solution:**

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

---

**Unable to generate a tag database for device <device name>****Error Type:**

Warning

**Possible Cause:**

The file specified as the Tag Import File in the Database Settings property group 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 the problem resumes, try re-exporting the file.

**See Also:**

[Importing DirectSoft Elements](#)

# Index

## A

Address '<address>' is out of range for the specified device or register 36  
Address Descriptions 22  
Allow Sub Groups 14  
Attempts Before Timeout 12  
Auto-Demotion 13  
Auto-Dial 8  
Automatic Tag Database Generation 16

## B

Bad address in block [<start address> to <end address>] on device '<device name>' 38  
Baud Rate 7  
BCD 21  
Boolean 21  
Byte 21

## C

Channel Assignment 10  
Channel Properties – Advanced 9  
Channel Properties – General 5  
Channel Properties – Serial Communications 6  
Channel Properties – Write Optimizations 8  
Char 21  
Close Idle Connection 7-8  
COM ID 7  
COM Port 6  
Communications error on '<channel name>' [<error mask>] 37  
Communications Timeouts 12  
COMn does not exist 36  
COMn is in use by another application 37  
Connect Timeout 8, 12  
Connection Type 6  
Create 15

## D

Data Bits 7



Data Collection 11  
Data Type '<type>' is not valid for device address '<address>' 36  
Data Types Description 21  
Delete 14  
Demote on Failure 13  
Demotion Period 13  
Device '<device name>' not responding 38  
Device address '<address>' contains a syntax error 35  
Device address '<address>' is not supported by model '<model name>' 36  
Device address '<address>' is Read Only 36  
Device ID 5  
Device Properties – Auto-Demotion 13  
Device Properties – General 10  
Device Properties – Redundancy 15  
Device Properties – Tag Generation 13  
Device Properties – Timing 12  
Diagnostics 5  
DirectSoft Steps 17  
Discard Requests when Demoted 13  
DL-05 Addressing 22  
DL-06 Addressing 23  
DL-230 Addressing 24  
DL-240 Addressing 25  
DL-250 Addressing 26  
DL-260 Addressing 27  
DL-330 Addressing 29  
DL-340 Addressing 29  
DL-350 Addressing 30  
DL-430 Addressing 31  
DL-440 Addressing 32  
DL-450 Addressing 33  
Do Not Scan, Demand Poll Only 12  
Driver 10  
Drop 7  
DTR 7  
Duty Cycle 9  
DWord 21

## E

Element Documentation 15  
Error Descriptions 35  
Error opening COMn 37  
Ethernet Encap. 7

Ethernet Settings 7

## F

Float 21

Flow Control 7

Framing 37

## G

General 10

Generate 14

## I

ID 10

Identification 5, 10

Idle Time to Close 7-8

Import File-To-Server Name Conversions 16

Import Preparation 17, 19

Importing DirectSoft Elements 16

Initial Updates from Cache 12

Inter-Device Delay 9

## L

Long 21

## M

Mask 37

Missing address 35

Model 10

Modem 7-8

Modem Settings 8

## N

Name 10

Network 5

Network Adapter 8

Non-Normalized Float Handling 9

None 6

## O

On Device Startup 14

On Duplicate Tag 14

On Property Change 14

OPC Server Steps 19

Operating Mode 10

Operation with no Communications 8

Operational Behavior 7

Optimization Method 8

Overrun 37

Overview 4

Overwrite 14

## P

Parent Group 14

Parity 7, 37

Physical Medium 6

Poll Delay 7

## R

Raise 7

Read Processing 8

Redundancy 15

Replace with Zero 9

Report Communication Errors 7-8

Request Timeout 12

Respect Tag-Specified Scan Rate 12

RS-485 7

RTS 7

## S

Scan Mode 11

Serial Communications 6

Serial Port Settings 7

Setup 4

Shared 7

Short 21  
Simulated 11  
Stop Bits 7

## T

Tag Counts 6, 11  
Tag Generation 13  
Tag Hierarchy 16  
Tag Import Settings 15  
Timeouts to Demote 13  
Timing 12

## U

Unable to generate a tag database for device <device name> 39  
Unable to set comm properties on COMn 37  
Unable to write tag '<address>' on device '<device name>' 38  
Unmodified 9

## W

Word 21  
Write All Values for All Tags 8  
Write Only Latest Value for All Tags 9  
Write Only Latest Value for Non-Boolean Tags 9