IDEC Serial Driver

Table of Contents

IDEC Serial Driver

Help version 1.031

CONTENTS

Overview

The IDEC Serial Driver provides a reliable way to connect IDEC Serial controllers to OPC client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with IDEC Programmable Logic Controllers.

Setup

Supported Devices

Micro1
Micro3
MicroSmart
OpenNet
FA2J
FA2
FA3S-CP11
FA3S-CP12

Communication Protocol

IDEC ASCII Protocol

Supported Communication Parameters

Baud Rate: 300, 600, 1200, 2400, 9600, 19200
Parity: Even, None
Data Bits: 7 (Micro3, OpenNet); 8 (All other models)
Stop Bits: 1

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 (such as the Lantronix DR1). It may be invoked through the COM ID property group. in Channel Properties. For more information, refer to the OPC server's help documentation.

Channel and Device Limits

The maximum number of channels supported by this driver is 100. The maximum number of devices supported by this driver is 32 per channel.

Device IDs

Micro3 networks support up to 32 devices. Each device can be assigned a Device ID in the range of 0 to 255. All other models support up to 32 devices and should be assigned a Device ID of 0 for 1:1 communications (or a Device ID of 1-255 for 1:N communications).

Note: When multidropping, do not assign a Device ID of 0.

Flow Control

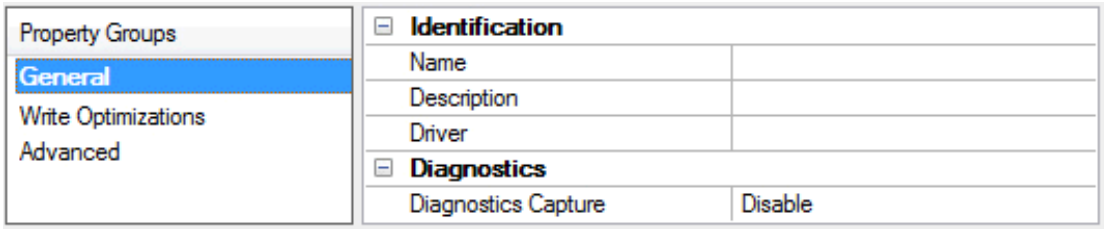
When using an RS232 / RS485 converter, the type of flow control that is required depends on the needs of the converter. Some converters do not require any flow control whereas others require RTS flow. To determine the converter's flow requirements, refer to its documentation. An RS485 converter that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

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.

See Also: 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.



Identification

Name: Specify the user-defined identity of this channel. In each server project, each channel name must be unique. Although names can be up to 256 characters, some client applications have a limited display window when browsing the OPC server's tag space. The channel name is part of the OPC browser information. The property is required for creating a channel.

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

Description: Specify user-defined information about this channel.

Many of these properties, including Description, have an associated system tag.

Driver: Specify the protocol / driver for this channel. 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 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 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.

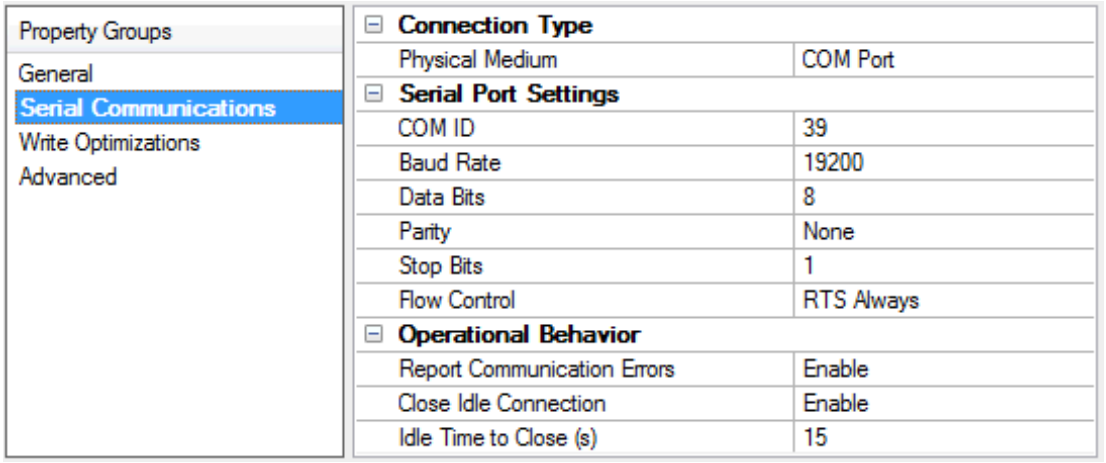
For more information, refer to "Communication Diagnostics" and "Statistics Tags" in the server help.

Channel Properties — Serial Communications

Serial communication properties are available to serial drivers and vary depending on the driver, connection type, and options selected. Below is a superset of the possible properties.

Click to jump to one of the sections: Connection Type, Serial Port Settings or Ethernet Settings, and Operational Behavior.

Note: With the server's online full-time operation, these properties can be changed at any time. Utilize proper user role and privilege management to prevent operators from changing properties or accessing server features.



Connection Type

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

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

Serial Port Settings

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

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

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

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

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

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

- **None**: This option does not toggle or assert control lines.
- **DTR**: This option asserts the DTR line when the communications port is opened and remains on.
- **RTS**: This option specifies that the RTS line is high if bytes are available for transmission. After all buffered bytes have been sent, the RTS line is low. This is normally used with RS232/RS485 converter hardware.
- **RTS, DTR**: This option is a combination of DTR and RTS.
- **RTS Always**: This option asserts the RTS line when the communication port is opened and remains on.
- **RTS Manual**: This option asserts the RTS line based on the timing properties entered for RTS Line Control. It is only available when the driver supports manual RTS line control (or when the properties are shared and at least one of the channels belongs to a driver that provides this support).
  RTS Manual adds an **RTS Line Control** property with options as follows:
  - **Raise**: This property specifies the amount of time that the RTS line is raised prior to data transmission. The valid range is 0 to 9999 milliseconds. The default is 10 milliseconds.
  - **Drop**: This property specifies the amount of time that the RTS line remains high after data transmission. The valid range is 0 to 9999 milliseconds. The default is 10 milliseconds.
  - **Poll Delay**: This property specifies the amount of time that polling for communications is delayed. The valid range is 0 to 9999. The default is 10 milliseconds.

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

Operational Behavior

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

Ethernet Settings

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

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

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

Modem Settings

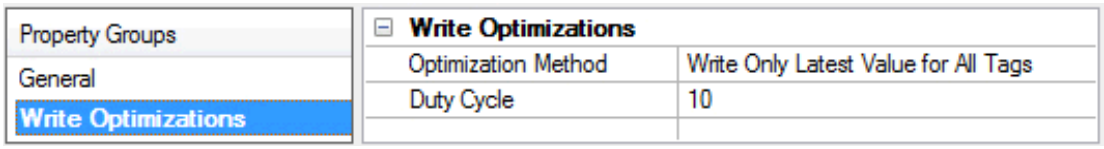
- **Modem**: Specify the installed modem to be used for communications.
- **Connect Timeout**: Specify the amount of time to wait for connections to be established before failing a read or write. The default is 60 seconds.
- **Modem Properties**: Configure the modem hardware. When clicked, it opens vendor-specific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

Operation with no Communications

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

Channel Properties — Write Optimizations

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



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.

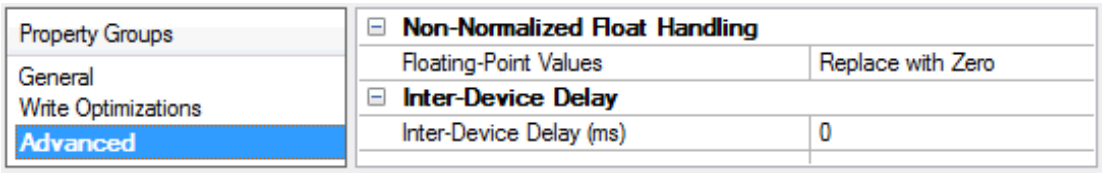
- **Write Only Latest Value for All Tags**: This option takes the theory behind the second optimization mode and applies it to all tags. It is especially useful if the application only needs to send the latest value to the device. This mode optimizes all writes by updating the tags currently in the write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

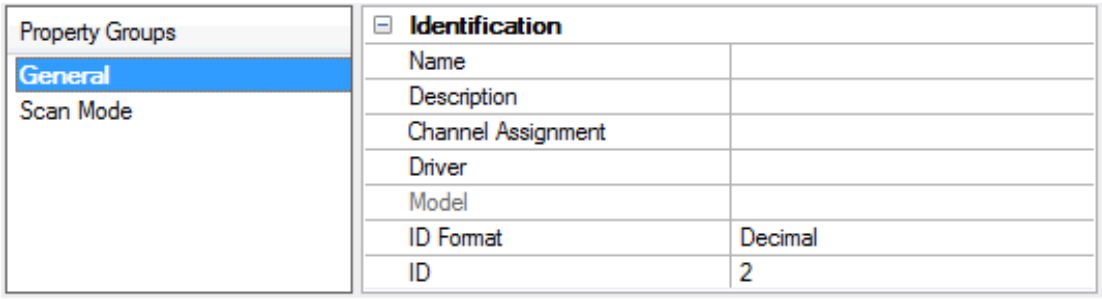
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

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: 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.

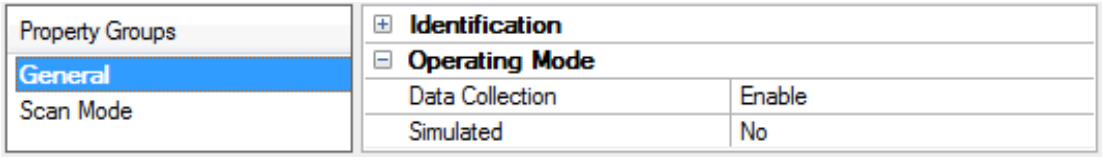
Driver: Selected protocol driver for this device.

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

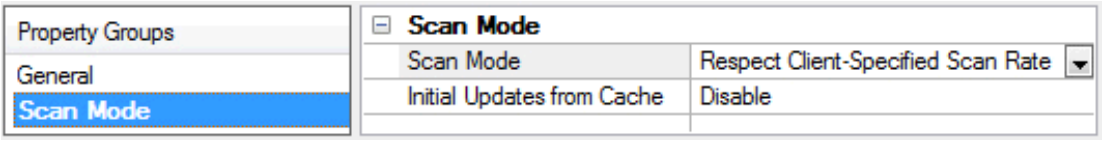
Notes:

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.



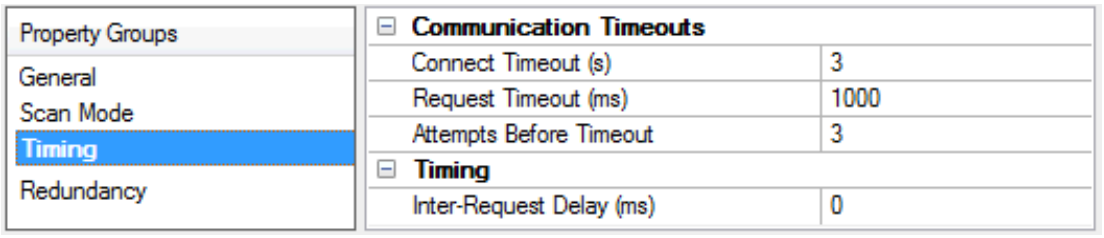
Scan Mode: Specify how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

- **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 client's responsibility to poll for updates, either by writing to the _DemandPoll tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help*.
- **Respect Tag-Specified Scan Rate**: This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

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

Device Properties — Timing

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



Communications Timeouts

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

Request Timeout: Specify an interval used by all drivers to determine how long the driver waits for a response from the target device to complete. The valid range is 50 to 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: 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.

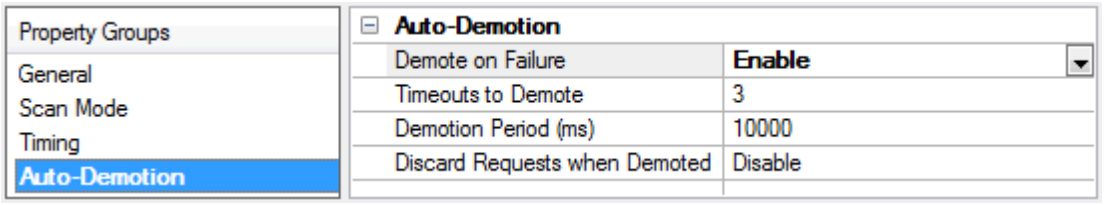
Timing

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.

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

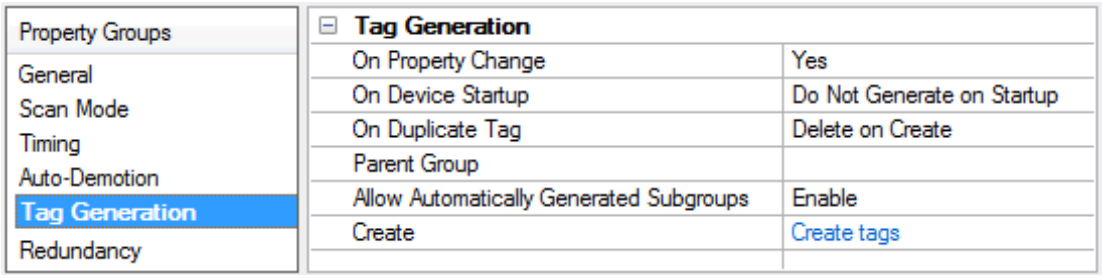
The automatic tag database generation features make setting up an application a plug-and-play operation. Select communications drivers can be configured to automatically build a list of tags that correspond to device-specific data. These automatically generated tags (which depend on the nature of the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

Automatic Tag Database Generation

The IDEC Serial Driver utilizes the OPC server's Automatic Tag Database Generation feature, which automatically creates tags that correspond to data points used by the device's ladder program. A tag information

file is required to generate tags, and is created with the IDEC WindLDR programming application (version 4.14 or higher).

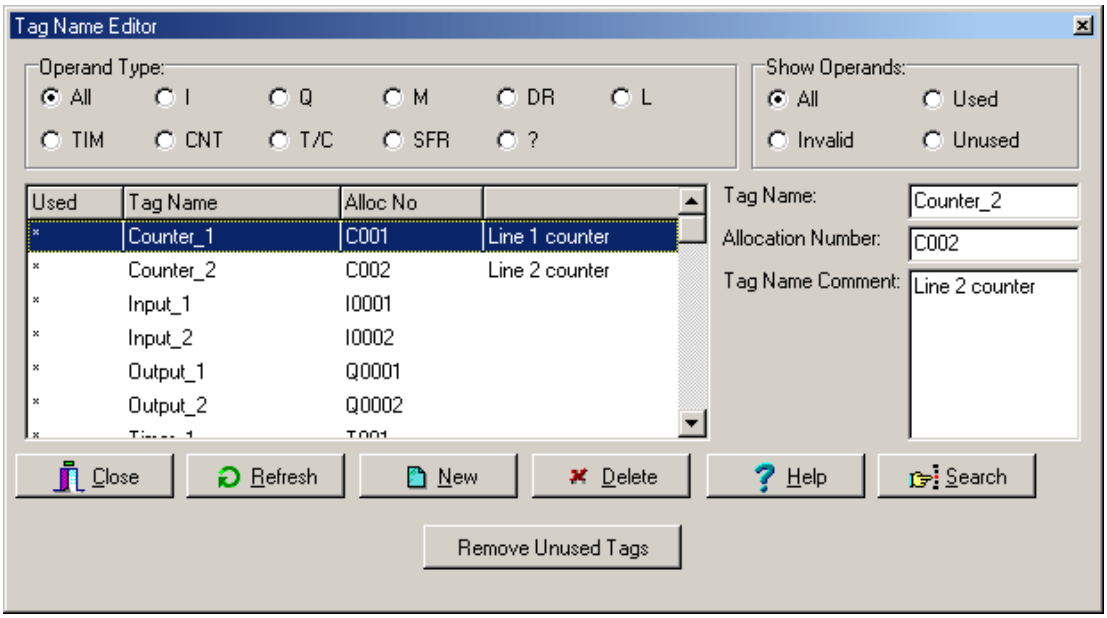
The automatic tag database generation feature is customizable. The tag generation settings can be accessed by clicking **Device Properties** | **Database Creation**. For more information, refer to the OPC server's help documentation.

Creating the Tag Information File

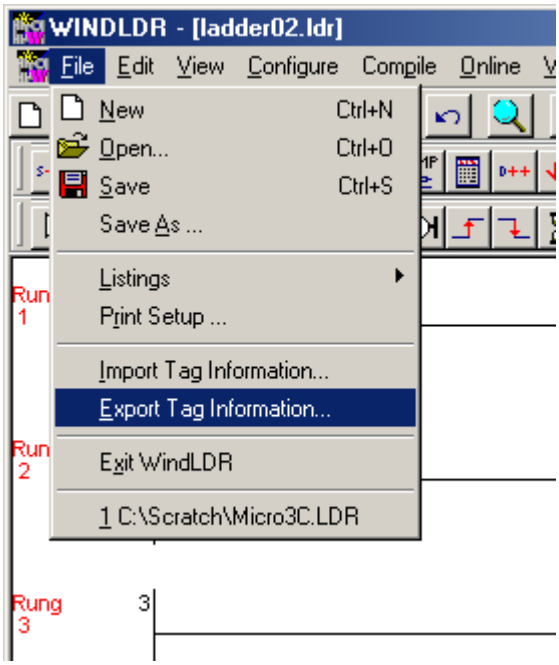
A tag information file must be prepared for the target device before tags can be automatically generated. For more information, refer to the instructions below.

1. Start the WndLDR device programming application. Then, load the ladder project that will be run by the target device.
2. In the main menu, click **Edit** | **Tag Name Editor** to view the current list of tags.

   Note: This list is automatically updated as the ladder program is created. Tags not used by the current ladder may be added or removed, and existing tags may be edited. The driver will create tags with names as they appear in the editor. Allocation numbers will be converted to the appropriate address strings. Tag name comments will appear as tag descriptions in the server. Users should review all operands because they will be used by the automatic tag generator.



3. When satisfied with the tag name list, click **Close**.
4. To save the tag name data to a tag information file, select **File** | **Export Tag Information**.



Configuring the Driver

Once the tag information file has been created for a particular device, the driver must be configured to use it. For more information, refer to the instructions below.

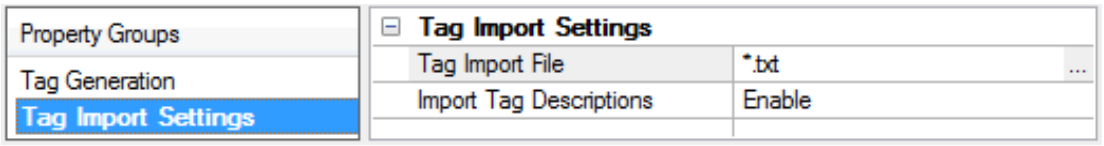
1. To start, open the **Device Properties** for the device of interest.
2. Next, select the **Tag Import Settings** property group. For more information, refer to Tag Import Settings.
3. Manually enter the location of the device's tag information file. Alternatively, click the **Browse** button and then locate the tag information file.
4. Select the **Tag Generation** property group and configure as desired.

The OPC server's Event Log will show when the automatic tag generation feature is enabled, any errors that occurred during processing, and when the generation process completed. For more information, refer to the OPC server's help documentation.

Notes:

- At this time, WndLDR does not explicitly export data type information. As such, the driver must assign its default data type to all generated tags. Address mnemonics and data types of generated tags may need to be adjusted to match the ladder program.
- Records for numerical timer and counter presets will be ignored.
- Three tags will be created for each reference to a timer and counter: current value, output bit, and preset.

Device Properties — Tag Import Settings

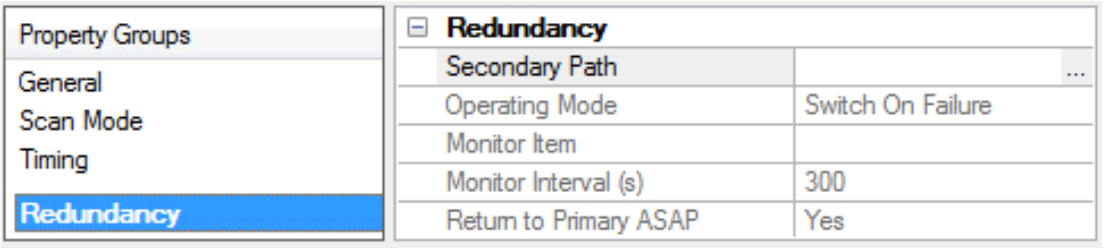


Tag Import File: Specify the exact location of the WindLDR tag information file that the driver should use when automatically generating tags for the device.

Import Tag Descriptions: Enable tag descriptions for import. The default setting is enabled.

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.

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
Float	32-bit floating point 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

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.

Micro1
Micro3
MicroSmart
OpenNet Controller
FA2
FA2J
FA3S-CP11
FA3S-CP12

Micro1 Addressing

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

Data Types

Device Type	Reference	Data Type
I/O	I, Q, IF	**Boolean**
Internal Relays	M	**Boolean**
Shift Registers	R	**Boolean**
Timers/Counters	T, C TW, TP, CW, CP	**Boolean** **Word**, Short

Address Specifications

Address	Reference	Range	Access
Input Bits	I<xx> xx-Bit Number*	I0-I27	Read Only
Force Input Bits	IF<xx> xx-Bit Number*	IF0-IF27	Read/Write
Output Bits	Q<xx> xx-Bit Number*	Q0-Q27	Read/Write
Internal Relay Bits	M<xxx> xxx-Bit Number*	M0-M197, M300-317	Read/Write
Shift Register Bits	R<xxx> xxx-Bit Number	R0-R127	Read/Write
Counter Bits	C<xx> xx-Bit Number	C0-C46	Read Only
Counter Words	CW<xx> xx-Word Number**	CW0-CW46	Read Only
Counter Presets	CP<xx> xx-Word Number**	CP0-CP46	Read/Write
Timer Bits	T<xx>	T0-T79	Read Only

Address	Reference	Range	Access
	xx-Bit Number		
Timer Words	TW<xx> xx-Word Number**	TW0-TW79	Read Only
Timer Presets	TP<xx> xx-Word Number**	TP0-TP79	Read/Write

* Low digit octal coded.
** 14 bit number.

Examples

M180	Internal relay bits 180.*
TP65	Preset value for timer 65.**
TW65	Accumulator value for timer 65.**

* Lowest digit octal coded. M108 would not be valid.
** 14 bit number.

Micro3 Addressing

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

Data Types

Device Type	Reference	Data Type
I/O	I, Q, IF IW, QW	**Boolean** **Word**, Short
Internal Relays	M MW	**Boolean** **Word**, Short
Shift Registers	R RW	**Boolean** **Word**, Short
Timers/Counters	T, C TW, TP, CW, CP HW, HP	**Boolean** **Word**, Short **DWord**, Long
Data Registers	DW DS	**Word**, Short, DWord, Long, Float **String**
Calendar/Clock	WW	**Word**, Short

Address Specifications

The following memory map corresponds to the Micro3 device. The driver expands this memory range from 0 to 9999 for all memory types to support newer devices such as the Micro3C and any other future Micro3 compatible configurations.

Address	Reference	Range	Access
Input Bits/Words	I<xxx>, IW<xxx> xxx-Bit Number*	I0-I37, IW0-IW20	Read Only
Force Input Bits	IF<xxx>	IF0-IF37	Read/Write

Address	Reference	Range	Access
	xxx-Bit Number*		
Output Bits/Words	Q<xxx>, QW<xxx> xx-Bit Number*	Q0-Q37, QW0-QW20	Read/Write
Internal Relay Bits/Words	M<xxx>, MW<xxx> xxx-Bit Number*	M0-M317, MW0-MW300	Read/Write
Shift Register Bits/Words	R<xxx>, RW<xxx> xxx-Bit Number	R0-R63, RW0-RW48	Read/Write
Counter Bits	C<xx> xx-Bit Number	C0-C31	Read Only
Counter Words	CW<xx> xx-Word Number**	CW0-CW31	Read Only
Counter Presets	CP<xx> xx-Word Number**	CP0-CP31	Read/Write
Timer Bits	T<xx> xx-Bit Number	T0-T31	Read Only
Timer Words	TW<xx> xx-Word Number**	TW0-TW31	Read Only
Timer Presets	TP<xx> xx-Word Number**	TP0-TP31	Read/Write
High Speed Counter Accumulators	HW<xx> xx-HSC Accumulator Number***	HW0-HW9999	Read Only
High-Speed Counter Presets	HP<xx> xx-HSC Preset Number***	HP0-HP9999	Read Only
Data Registers	DW<xxx> xxx-Word Number DS<xxx>.<yyy> xxx-Word Number yyy-String Length (characters)	DW0-DW9999 DS0.0 …DS0.128 DS99.0 …DS99.128	Read/Write Read/Write
Calendar/Clock	WW<xxx> xxx-Word Number	WW0-WW6	Read/Write

* Low digit octal coded.
** 14 bit number.
*** 32 bit number.

Calendar/Clock Word Numbers

0	Year (0=2000, 1=2001, etc)
1	Month
2	Day
3	Day of week (0=Sunday, 1=Monday, etc)
4	Hour (24 hour format)
5	Minute

0	Year (0=2000, 1=2001, etc)
6	Second

Examples

M180	Internal relay bits 180.*
TP65	Preset value for timer 65.**
TW65	Accumulator value for timer 65.**
DS10.24	String data stored in DW10-DW21.
WW2	Day of the month.

* Lowest digit octal coded. M108 would not be valid.
** 14 bit number.

MicroSmart Addressing

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

Data Types

Device Type	Reference	Data Type
I/O	I, Q, IF IW, QW	**Boolean** **Word**, Short
Internal Relays	M MW	**Boolean** **Word**, Short
Shift Registers	R RW	**Boolean** **Word**, Short
Timers/Counters	T, C TW, TP, CW, CP HW, HP	**Boolean** **Word**, Short **DWord**, Long
Data Registers	DW DS	**Word**, Short, DWord, Long, Float **String**
Calendar/Clock	WW	**Word**, Short

Address Specifications

The following memory map corresponds to the MicroSmart device. The driver expands this memory range from 0 to 9999 for all memory types to support newer devices such as the Micro3C and any other future Micro3 compatible configurations.

Address	Reference	Range	Access
Input Bits/Words	I<xxx>, IW<xxx> xxx-Bit Number*	I0-I627, IW0-IW610	Read Only
Force Input Bits	IF<xxx> xxx-Bit Number*	IF0-IF627	Read/Write
Output Bits/Words	Q<xxx>, QW<xxx> xx-Bit Number*	Q0-Q627, QW0-QW610	Read/Write
Internal Relay Bits/Words	M<xxxx>, MW<xxxx> xxxx-Bit Number*	M0-M8317, MW0-MW8300	Read/Write

Shift Register Bits/Words	R<xxx>, RW<xxx> xxx-Bit Number	R0-R255, RW0-RW240	Read/Write
Counter Bits	C<xx> xx-Bit Number	C0-C255	Read Only
Counter Words	CW<xx> xx-Word Number**	CW0-CW255	Read Only
Counter Presets	CP<xx> xx-Word Number**	CP0-CP255	Read/Write
Timer Bits	T<xx> xx-Bit Number	T0-T255	Read Only
Timer Words	TW<xx> xx-Word Number**	TW0-TW255	Read Only
Timer Presets	TP<xx> xx-Word Number**	TP0-TP255	Read/Write
High-Speed Counter Accumulators	HW<xx> xx-HSC Accumulator Number***	HW0-HW9998	Read Only
High-Speed Counter Presets	HP<xx> xx-HSC Preset Number***	HP0-HP9998	Read Only
Data Registers	DW<xxxx> xxxx-Word Number DS<xxxx>.<yyy> xxxx-Word Number yyy-String Length (characters)	DW0-DW49999 DS0.0...DS0.128- DS49999.0...DS49999.128	Read/Write Read/Write
Calendar/Clock	WW<x> x-Word Number	WW0-WW6	Read/Write

* Low digit octal coded.
** 16 bit number.
*** 32 bit number.

Calendar/Clock Word Numbers

0	Year (0 = 2000, 1 = 2001, etc)
1	Month
2	Day
3	Day of week (0 = Sunday, 1 = Monday, etc)
4	Hour (24 hour format)
5	Minute
6	Second

Examples

M180	Internal relay bits 180.*
TP65	Preset value for timer 65.**
TW65	Accumulator value for timer 65.**

M180	Internal relay bits 180.*
DS10.24	String data stored in DW10-DW21.
WW2	Day of the month.

* Lowest digit octal coded. M108 would not be valid.
* * 16 bit number.

OpenNet Controller Addressing

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

Data Types

Device Type	Reference	Data Type
I/O	I, Q, IF IW, QW	**Boolean** **Word**, Short
Internal Relays	M MW	**Boolean** **Word**, Short
Shift Registers	R RW	**Boolean** **Word**, Short
Link Registers	Lxxxx Lxxxx.yy	**Word**, Short **Boolean**
Timers/Counters	T, C TW, TP, CW, CP HW, HP	**Boolean** **Word**, Short **DWord**, Long
Data Registers	DW	**Word**, Short, DWord, Long, Float

Address Specifications

The following memory map is open from 0 to 9999 for all memory types to support newer devices. Consult the manufacturer's documentation for device specific address ranges.

Address	Reference	Range	Access
Input Bits/Words	I<xxxx>, IW<xxxx> xxxx-Bit Number*	I0-I9997, IW0-IW9997	Read Only
Force Input Bits	IF<xxxx> xxxx-Bit Number*	IF0-IF9997	Read/Write
Output Bits/Words	Q<xxxx>, QW<xxxx> xxxx-Bit Number*	Q0-Q9997, QW0-QW9997	Read/Write
Internal Relay Bits/Words	M<xxxx>, MW<xxxx> xxxx-Bit Number*	M0-M9997, MW0-MW9997	Read/Write
Shift Register Bits/Words	R<xxxx>, RW<xxxx> xxxx-Bit Number	R0-R9999, RW0-RW9999	Read/Write
Link Register	L<xxxx> xxxx-Bit Number*	L0-L9997	Read/Write
Link Register Bit Access	L<xxxx>.<yy> <xxxx>-register number* <yy>-bit number	<xxxx>: 100-127, 200-227, 300-327, 400-427, 500-527, 600-627, 700-727, 1000-	Read/Write

Address	Reference	Range	Access
		1317 <yy>: 0-15	
Counter Bits	C<xxxx> xxxx-Bit Number	C0-C9999	Read Only
Counter Words	CW<xxxx> xxxx-Word Number**	CW0-CW9999	Read Only
Counter Presets	CP<xxxx> xxxx-Word Number**	CP0-CP9999	Read/Write
Timer Bits	T<xx> xxxx-Bit Number	T0-T9999	Read Only
Timer Words	TW<xxxx> xxxx-Word Number**	TW0-TW9999	Read Only
Timer Presets	TP<xxxx> xxxx-Word Number**	TP0-TP9999	Read/Write
High-Speed Counter Accumulators	HW<xxxx> xxxx-HSC Accumulator Number***	HW0-HW9999	Read Only
High-Speed Counter Presets	HP<xxxx> xxxx-HSC Preset Number***	HP0-HP9999	Read Only
Data Registers	DW<xxxx> xxxx-Word Number DS<xxxx>.<yyy> xxxx-Word Number yyy-String Length (characters)	DW0-DW9999 DS0.0 ...DS0.128 DS9999.0 ...DS9999.128	Read/Write Read/Write
Calendar/Clock	WW<x> x-Word Number	WW0-WW6	Read/Write

* Low digit octal coded.
** 16 bit number.
*** 32 bit number.

Calendar/Clock Word Numbers

0	Year (0=2000, 1=2001, etc)
1	Month
2	Day
3	Day of week (0=Sunday, 1=Monday, etc)
4	Hour (24 hour format)
5	Minute
6	Second

Examples

M180	Internal relay bits 180.*
TP65	Preset value for timer 65.**
TW65	Accumulator value for timer 65.**
DS10.24	String data stored in DW10-DW21.
WW2	Day of the month.

* Lowest digit octal coded. M108 would not be valid.
** 16 bit number.

FA2 Addressing

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

Data Types

Device Type	Reference	Data Type
I/O	I, Q, IF IB, QB	**Boolean** **Byte**, Char
Internal Relays	M MB	**Boolean** **Byte**, Char
Shift Registers	R RB	**Boolean** **Byte**, Char
Timers/Counters	T, C TW, TP, CW, CP	**Boolean** **Word**, Short
Data Registers	DW	**Word**, Short

Address Specifications

Address	Reference	Range	Access
Input Bits/Bytes	I<xxx>, IB<xxx> xxx-Bit Number*	I0-I317, IB0-IB310	Read Only
Force Input Bits	IF<xxx> xxx-Bit Number*	IF0-IF317	Read/Write
Output Bits/Bytes	Q<xxx>, QB<xxx> xx-Bit Number*	Q0-Q317, QB0-QB310	Read/Write (bits) Read Only (bytes)
Internal Relay Bits/Bytes	M<xxx>, MB<xxx> xxx-Bit Number*	M0-M637, MB0-MB630	Read/Write (bits) Read Only (bytes)
Shift Register Bits/Bytes	R<xxx>, RB<xxx> xxx-Bit Number	R0-R127, RB0-RB120	Read/Write (bits) Read Only (bytes)
Counter Bits	C<xx> xx-Bit Number	C0-C46	Read Only
Counter Words	CW<xx> xx-Word Number**	CW0-CW46	Read Only
Counter Presets	CP<xx> xx-Word Number**	CP0-CP46	Read/Write

Address	Reference	Range	Access
Timer Bits	T<xx> xx-Bit Number	T0-T79	Read Only
Timer Words	TW<xx> xx-Word Number**	TW0-TW79	Read Only
Timer Presets	TP<xx> xx-Word Number**	TP0-TP79	Read/Write
Data Registers	DW<xxx> xxx-Word Number	DW0-DW399	Read/Write

* Low digit octal coded.
** 14-bit number.

Examples

M180	Internal relay bits 180.*
TP65	Preset value for timer 65**
TW65	Accumulator value for timer 65**

* Lowest digit octal coded. M108 would not be valid.
** 14-bit number.

FA2J Addressing

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

Data Types

Device Type	Reference	Data Type
I/O	I, Q, IF IB, QB	**Boolean** **Byte**, Char
Internal Relays	M MB	**Boolean** **Byte**, Char
Shift Registers	R RB	**Boolean** **Byte**, Char
Timers/Counters	T, C TW, TP, CW, CP	**Boolean** **Word**, Short
Data Registers	DW	**Word**, Short

Address Specifications

Address	Reference	Range	Access
Input Bits/Bytes	I<xxx>, IB<xxx> xxx-Bit Number*	I0-I157, IB0-IB150	Read Only
Force Input Bits	IF<xxx> xxx-Bit Number*	IF0-IF157	Read/Write
Output Bits/Bytes	Q<xxx>, QB<xxx> xx-Bit Number*	Q0-Q157, QB0-QB150	Read/Write (bits) Read Only (bytes)

Address	Reference	Range	Access
Internal Relay Bits/Bytes	M<xxx>, MB<xxx> xxx-Bit Number*	M0-M637, MB0-MB630	Read/Write (bits) Read Only (bytes)
Shift Register Bits/Bytes	R<xxx>, RB<xxx> xxx-Bit Number	R0-R127, RB0-RB120	Read/Write (bits) Read Only (bytes)
Counter Bits	C<xx> xx-Bit Number	C0-C46	Read Only
Counter Words	CW<xx> xx-Word Number**	CW0-CW46	Read Only
Counter Presets	CP<xx> xx-Word Number**	CP0-CP46	Read/Write
Timer Bits	T<xx> xx-Bit Number	T0-T79	Read Only
Timer Words	TW<xx> xx-Word Number**	TW0-TW79	Read Only
Timer Presets	TP<xx> xx-Word Number**	TP0-TP79	Read/Write
Data Registers	DW<xxx> xxx-Word Number	DW0-DW399	Read/Write

* Low digit octal coded.
** 14 bit number.

Examples

M180	Internal relay bits 180.*
TP65	Preset value for timer 65.**
TW65	Accumulator value for timer 65.**

* Lowest digit octal coded. M108 would not be valid.
** 14 bit number.

FA3S-CP11 Addressing

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

Data Types

Device Type	Reference	Data Type
I/O	I, Q, IF IB, QB	**Boolean** **Byte**, Char
Internal Relays	M MB	**Boolean** **Byte**, Char
Shift Registers	R RB	**Boolean** **Byte**, Char
Timers/Counters	T, C	**Boolean**

Device Type	Reference	Data Type
	TW, TP, CW, CP	**Word**, Short
Data Registers	DW	**Word**, Short

Address Specifications

Address	Reference	Range	Access
Input Bits/Bytes	I<xxx>, IB<xxx> xxx-Bit Number*	I0-I157, IB0-IB150	Read Only
Force Input Bits	IF<xxx> xxx-Bit Number*	IF0-IF157	Read/Write
Output Bits/Bytes	Q<xxx>, QB<xxx> xx-Bit Number*	Q0-Q157, QB0-QB150	Read/Write (bits) Read Only (bytes)
Internal Relay Bits/Bytes	M<xxx>, MB<xxx> xxx-Bit Number*	M0-M637, MB0-MB630	Read/Write (bits) Read Only (bytes)
Shift Register Bits/Bytes	R<xxx>, RB<xxx> xxx-Bit Number	R0-R127, RB0-RB120	Read/Write (bits) Read Only (bytes)
Counter Bits	C<xx> xx-Bit Number	C0-C46	Read Only
Counter Words	CW<xx> xx-Word Number**	CW0-CW46	Read Only
Counter Presets	CP<xx> xx-Word Number**	CP0-CP46	Read/Write
Timer Bits	T<xx> xx-Bit Number	T0-T79	Read Only
Timer Words	TW<xx> xx-Word Number**	TW0-TW79	Read Only
Timer Presets	TP<xx> xx-Word Number**	TP0-TP79	Read/Write
Data Registers	DW<xxx> xxx-Word Number	DW0-DW399	Read/Write

* Low digit octal coded.
** 14 bit number.

Examples

M180	Internal relay bits 180.*
TP65	Preset value for timer 65.**
TW65	Accumulator value for timer 65.**

* Lowest digit octal coded. M108 would not be valid.
** 14 bit number.

FA3S-CP12 Addressing

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

Data Types

Device Type	Reference	Data Type
I/O	I, Q, IF IB, QB	**Boolean** **Byte**, Char
Internal Relays	M MB	**Boolean** **Byte**, Char
Shift Registers	R RB	**Boolean** **Byte**, Char
Timers/Counters	T, C TW, TP, CW, CP	**Boolean** **Word**, Short
Data Registers	DW	**Word**, Short

Address Specifications

Address	Reference	Range	Access
Input Bits/Bytes	I<xxx>, IB<xxx> xxx-Bit Number*	I0-I317, IB0-IB310	Read Only
Force Input Bits	IF<xxx> xxx-Bit Number*	IF0-IF317	Read/Write
Output Bits/Bytes	Q<xxx>, QB<xxx> xx-Bit Number*	Q0-Q317, QB0-QB310	Read/Write (bits) Read Only (bytes)
Internal Relay Bits/Bytes	M<xxx>, MB<xxx> xxx-Bit Number*	M0-M1317, MB0-MB1310	Read/Write (bits) Read Only (bytes)
Shift Register Bits/Bytes	R<xxx>, RB<xxx> xxx-Bit Number	R0-R223, RB0-RB216	Read/Write (bits) Read Only (bytes)
Counter Bits	C<xx> xx-Bit Number	C0-C46 C48-C102	Read Only
Counter Words	CW<xx> xx-Word Number**	CW0-CW46 CW48-CW102	Read Only
Counter Presets	CP<xx> xx-Word Number**	CP0-CP46 CP48-CP102	Read/Write
Timer Bits	T<xx> xx-Bit Number	T0-T255	Read Only
Timer Words	TW<xx> xx-Word Number**	TW0-TW255	Read Only
Timer Presets	TP<xx> xx-Word Number**	TP0-TP255	Read/Write
Data Registers	DW<xxx> xxx-Word Number	DW0-DW999 DW3000-DW3071	Read/Write

* Low digit octal coded.
** 14 bit number.

Examples

M180	Internal relay bits 180.*
TP65	Preset value for timer 65.**
TW65	Accumulator value for timer 65.**

* Lowest digit octal coded. M108 would not be valid.
** 14 bit number.

Error Descriptions

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

Address Validation

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

Serial Communications

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

Device Status Messages

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

Device Specific Messages

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

Automatic Tag Database Generation Messages

Tag '<tag name>' not imported because model does not support type
Error parsing import file record number <record number>
Import file record <record number> could not be processed due to buffer overflow
Exception encountered during tag import. Bad tag information file
Tag import failed due to low memory resources
Description truncated for import file record number <record number>
Imported tag name '<old name>' is invalid. Name changed to '<new 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 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 statically references a location that is valid for the communications protocol but not supported by the target device.

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

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

Error Type:
Warning

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

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

Device address '<address>' is Read Only

Error Type:
Warning

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

Solution:
Change the access mode in the client application.

COMn does not exist

Error Type:
Fatal

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

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

Error opening COMn

Error Type:
Fatal

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

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

COMn is in use by another application

Error Type:
Fatal

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

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

Unable to set comm parameters on COMn

Error Type:
Fatal

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

Solution:

Verify the serial parameters and make any necessary changes.

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

Error Type:

Serious

Error Mask Definitions:

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

Possible Cause:

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

Solution:

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

Solution:

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

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

Error Type:
Serious

Possible Cause:

1. The serial connection between the device and the Host PC is broken.
2. The communication 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 communication parameters match those of the device.
3. Verify that the Network ID given to the named device matches that of the actual device.

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

Error Type:
Serious

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

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

Tag '<tag name>' not imported because model does not support type

Error Type:
Warning

Possible Cause:
The tag information file contains references to operand/memory types not supported by the currently selected model.

Solution:
Verify that the model selected for the device in the driver and the WindLDR project from which the tag information file was created agree. Make appropriate model changes.

See Also:
Address Descriptions

Error parsing import file record number <record number>

Error Type:
Warning

Possible Cause:
A record in the tag information file has an incorrect format or a field was longer than expected.

Solution:
Verify the format of the tag information file using a text editor. Fields should be delineated by tabs and end with a new line character. Tag name and allocation numbers are assumed to be 31 characters or less and tag descriptions are assumed to be 64 characters or less. Make necessary changes to file.

Import file record <record number> could not be processed due to buffer overflow

Error Type:
Warning

Possible Cause:
A tag information file record is too long to fit into the memory buffer used for record processing.

Solution:
Records are assumed to be 128 characters long or less. Make necessary changes to the file.

Exception encountered during tag import. Bad tag information file

Error Type:
Warning

Possible Cause:
The tag information file is unreadable.

Solution:
Recreate the tag information file.

Tag import failed due to low memory resources

Error Type:
Warning

Possible Cause:
Memory needed to process the tag information file could not be allocated.

Solution:
Free up system resources and try again.

Description truncated for import file record number <record number>

Error Type:
Warning

Possible Cause:
The specified tag description is longer than the 64 character limit imposed by the OPC server.

Solution:
The driver will automatically truncate the description. Shortening tag descriptions by using WindLDR's tag name editor will ensure that all important information is maintained.

Imported tag name '<old name>' is invalid. Name changed to '<new name>'

Error Type:
Warning

Possible Cause:
The tag name given in the tag information file is not a valid OPC server name. Valid names must be composed entirely of letters, number, and underscores, and not start with an underscore.

Solution:
The driver will automatically rename the tag by replacing invalid characters. If the first character is invalid, it will be substituted with a zero. All other invalid characters will be substituted with underscores. If desired, change the tag names in the ladder project using WindLDR's tag name editor for consistency.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

W