Fuji Flex Driver

Table of Contents

Fuji Flex Driver

Help version 1.019

CONTENTS

Overview

The Fuji Flex Driver provides a reliable way to connect Fuji Flex devices to OPC client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with Fuji Flex N series devices.

Setup

Supported Devices

Fuji Flex N Series PLCs-NB0, NB1, NB2, NB3, NJ, and NS

Communication Protocol

Fuji Computer Link

Supported Communication Parameters

Baud Rate-300, 600, 1200, 2400, 4800, 9600, and **19200**
Parity - None, Even, and **Odd**
Data Bits-7 and **8**
Stop Bits-**1** and 2

Note: The default values are shown in **bold**.

Ethernet Encapsulation

This driver supports Ethernet Encapsulation. Ethernet Encapsulation allows the driver to communicate with serial devices attached to an Ethernet network using a terminal server. Ethernet Encapsulation mode is invoked by selecting it from the COM ID dialog on the channel properties page. More help on Ethernet Encapsulation can be found in the main OPC Server help file.

Channel and Device Limits

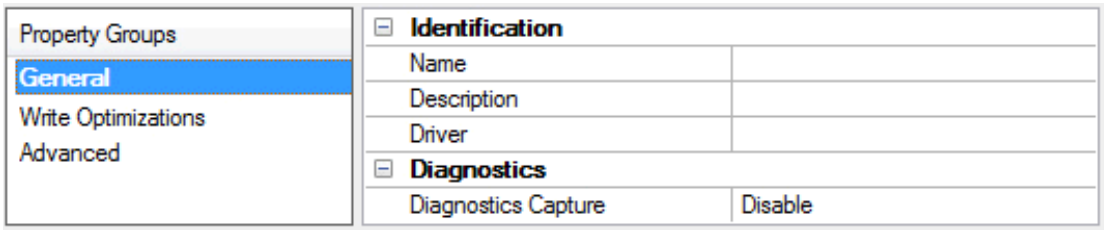
The maximum number of channels supported by this driver is 100. The maximum number of devices supported by this driver is 32 per channel.
Valid Device IDs range from 0 to 31.

Flow Control

When using an RS232/RS422 converter, the type of flow control that is required will depend upon the needs of the converter. Some converters do not require any flow control and others will require RTS flow. Consult the documentation of the converter to determine what its flow requirements are.

Channel Properties — General

This server supports the use of simultaneous multiple communications drivers. Each protocol or driver used in a server project is called a channel. A server project may consist of many channels with the same communications driver or with unique communications drivers. A channel acts as the basic building block of an OPC link. This group is used to specify general channel properties, such as the identification attributes and operating mode.



Identification

Name: User-defined identity of this channel. In each server project, each channel name must be unique. Although names can be up to 256 characters, some client applications have a limited display window when browsing the OPC server's tag space. The channel name is part of the OPC browser information. The property is required for creating a channel.
For information on reserved characters, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in the server help.

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

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

Diagnostics

Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications 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.

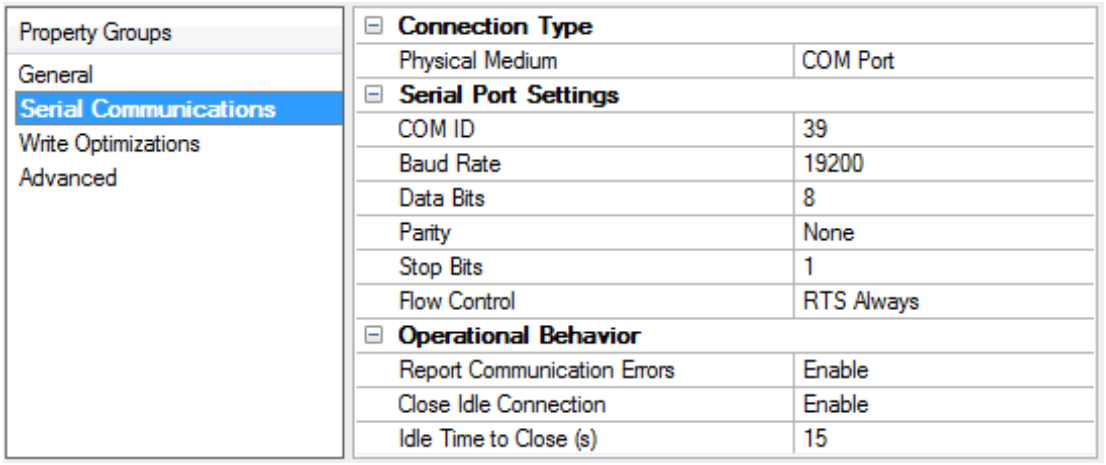
For more information, refer to "Communication Diagnostics" and "Statistics Tags" in the server help.

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 the User Manager to restrict access rights to server features, as changes made to these properties can temporarily disrupt communications.



Connection Type

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

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

Serial Port Settings

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

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

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

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

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

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

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

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

Operational Behavior

- **Report 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

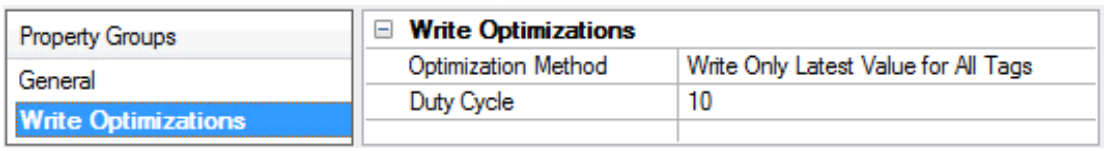
- **Modem**: Specify the installed modem to be used for communications.
- **Connect Timeout**: Specify the amount of time to wait for connections to be established before failing a read or write. The default is 60 seconds.
- **Modem Properties**: Configure the modem hardware. When clicked, it opens vendor-specific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

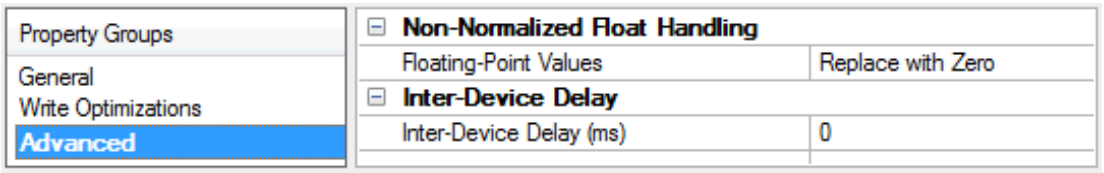
internal write queue. The server processes the write queue and attempts to empty it by writing data to the device as quickly as possible. This mode ensures that everything written from the client applications is sent to the target device. This mode should be selected if the write operation order or the 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.

Channel Properties — Interface Options

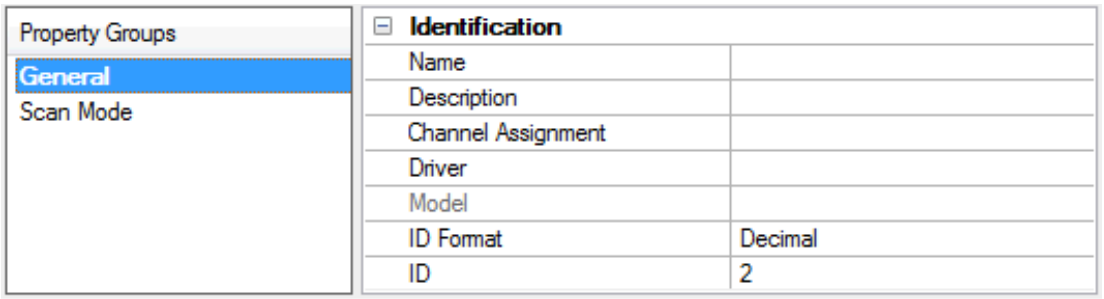
Communication with an N series PLC may be directly through the loader port or through an optional communications unit such as the NB-RS1. If devices on a given channel are equipped with communications units, enable **Using Communication Unit** in channel properties.

Networking multiple N series PLCs requires the use of communication units and an RS-485 line. There should be only one device on a channel if communication is through the loader port or through a communications unit with an RS-232C line.

If communication units are used, they must be configured to use ":" (0x3A) for the start code, a carriage return (0x0D) for the end code, and no BCC.

Device Properties — General

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



Identification

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

Note: Although descriptive names are generally a good idea, some OPC client applications may have a limited display window when browsing the OPC server's tag space. The device name and channel name

become part of the browse tree information as well. Within an OPC client, the combination of channel name and device name would appear as "ChannelName.DeviceName".

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

Description: User-defined information about this device.

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

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

Driver: Selected protocol driver for this device.

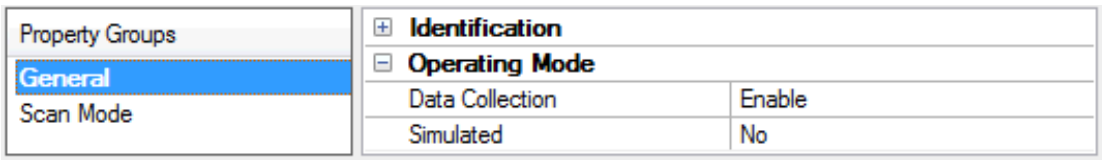
Model: This property specifies the specific type of device that is associated with this ID. The contents of the drop-down menu depends on the type of communications driver being used. Models that are not supported by a driver are disabled. If the communications driver supports multiple device models,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

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

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

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

Operating Mode



Data Collection: This property controls the device's active state. Although device communications are enabled by default, this property can be used to disable a physical device. Communications are not attempted when a device is disabled. From a client standpoint, the data is marked as invalid and write operations are not accepted. This property can be changed at any time through this property or the device system tags.

Simulated: This option places the device into Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client as valid data. While in Simulation Mode, the server treats all device data as reflective: whatever is written to the simulated device is read back and each OPC item is treated individually. The item's memory map is based on the group

Update Rate. The data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

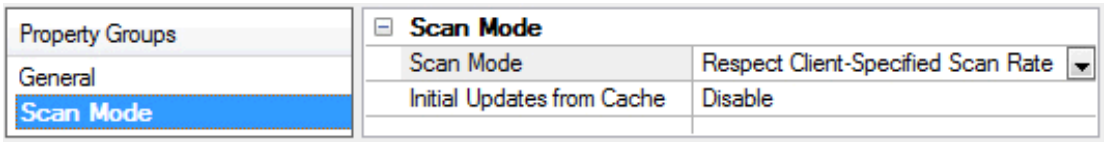
Notes:

1. This System tag (_Simulated) is read only and cannot be written to for runtime protection. The System tag allows this property to be monitored from the client.
2. In Simulation mode, the item's memory map is based on client update rate(s) (Group Update Rate for OPC clients or Scan Rate for native and DDE interfaces). This means that two clients that reference the same item with different update rates return different data.

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

Device Properties — Scan Mode

The Scan Mode specifies the subscribed-client requested scan rate for tags that require device communications. Synchronous and asynchronous device reads and writes are processed as soon as possible; unaffected by the Scan Mode properties.



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

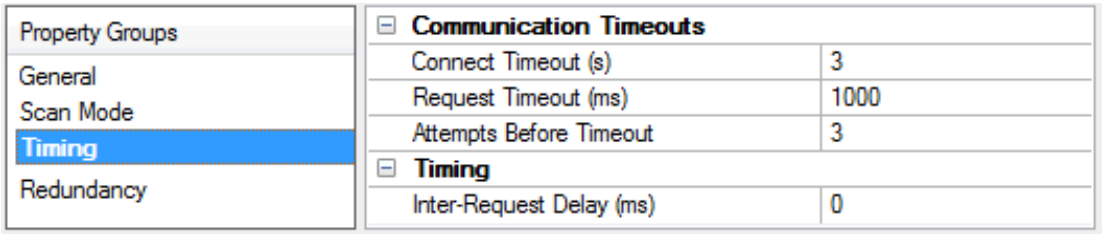
- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  Note: When the server has an active client and items for the device and the scan rate value is increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.
- **Request All Data at Scan Rate**: This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only**: This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the client's responsibility to poll for updates, either by writing to the _DemandPoll tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help*.
- **Respect Tag-Specified Scan Rate**: This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

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

Device Properties — Timing

The device Timing properties allow the driver's response to error conditions to be tailored to fit the application's needs. In many cases, the environment requires changes to these properties for optimum

performance. Factors such as electrically generated noise, modem delays, and poor physical connections can influence how many errors or timeouts a communications driver encounters. Timing properties are specific to each configured device.



Communications Timeouts

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

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

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

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

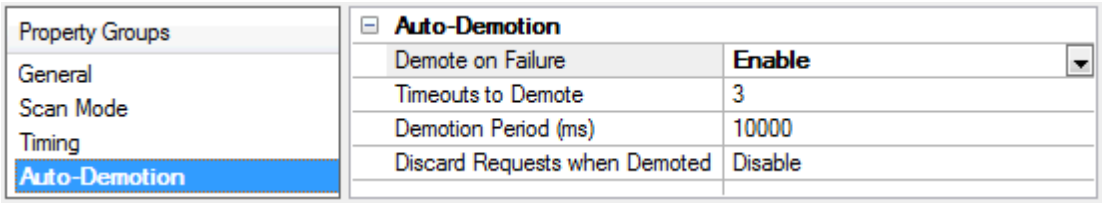
Timing

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

Note: Not all drivers support Inter-Request Delay. This setting does not appear if it is not available.

Device Properties — Auto-Demotion

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



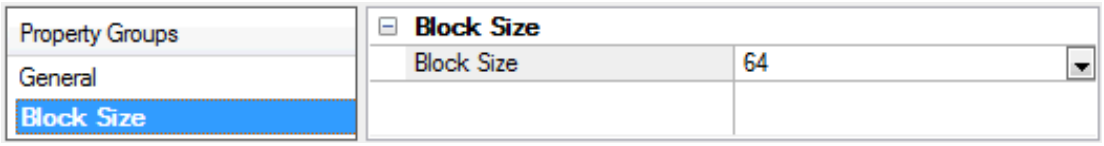
Demote on Failure: When enabled, the device is automatically taken off-scan until it is responding again.
Tip: Determine when a device is off-scan by monitoring its demoted state using the _AutoDemoted system tag.

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

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

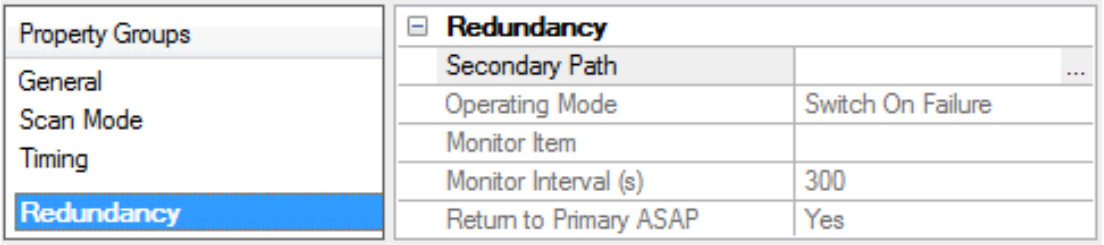
Discard Requests when Demoted: Select whether or not write requests should be attempted during the off-scan period. Disable to always send write requests regardless of the demotion period. Enable to discard writes; the server automatically fails any write request received from a client and does not post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

Device Properties — Block Size



The block size is the number of bytes that may be requested from a device at one time. This setting refines the performance of the driver. If a large number of consecutive data points are being read, a large block size may improve performance, whereas if a few scattered data points are being read, a smaller block size may improve performance. Block sizes range: 16, 32, 64, or 128. 64 is the default.

Device Properties — Redundancy



Redundancy is available with the Media-Level Redundancy Plug-In.
Consult the website, a sales representative, or the user manual for more information.

Data Types Description

Data Type	Description
Boolean	Single bit
Word	Unsigned 16-bit value bit 0 is the low bit bit 15 is the high bit
Short	Signed 16-bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
DWord	Unsigned 32-bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
BCD	Two byte packed BCD Value range is 0-9999. Behavior is undefined for values beyond this range.
LBCD	Four byte packed BCD Value range is 0-99999999. Behavior is undefined for values beyond this range.
String	Null terminated ASCII string.

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.

Flex-PC NB0
Flex-PC N Series Open

NB0 Model Address Descriptions

The **Flex-PC NB0** model option must be selected if you are communicating with an NB0 device. Use the **Flex-PC N Series Open** model for all other devices. All addresses are in hexadecimal. Bit numbers, string lengths, and array dimensions are in decimal. Default data types are shown in **bold**.

Address Type	Range	Data Type	Access
Input relay (as bits)	X0–X1F	**Boolean**	Read Only
Input relay (as words)*	WX0–WX1 WX0.b–WX1.b (b is bit number 0–15)	**Word**, Short* 3 **Boolean**	Read Only
Output relay (as bits)	Y0–Y1F	**Boolean**	Read/Write
Output relay (as words)*	WY0–WY1 WY0.b–WY1.b (b is bit number 0–15)	**Word**, Short* 3 **Boolean**	Read/Write
Internal relay (as bits)	M0–M81FF	**Boolean**	Read/Write
Internal relay (as words)*	WM0–WM81F WM0.b–WM81F.b (b is bit number 0–15)	**Word**, Short* 3 **Boolean**	Read/Write
Latch relay (as bits)	L0–LFF	**Boolean**	Read/Write
Latch relay (as words)*	WL0–WLF WL0.b–WLF.b (b is bit number 0–15)	**Word**, Short* 3 **Boolean**	Read/Write
Timer contact	T0–T1F	**Boolean**	Read/Write
Timer current value	WT0–WT1F	**Word**, Short, BCD* 3	Read/Write
Counter contact	C0–C1F	**Boolean**	Read/Write
Counter current value	WC0–WC1F	**Word**, Short, BCD* 3	Read/Write
Data register	D0–D803F D0–D803E D0.b–D803F.b (b is bit number 0–15) D0.l–D7FFF.l (l is string length 0–64)	**Word**, Short, BCD* 3 DWord, Long, LBCD* 2,* 3 **Boolean** String* 4	Read/Write

* When addressing discrete data as words, addresses correspond to word offsets. For example, WX0 references X0–XF, and WX1 references X10–X1F, etc.

* 2 When 32-bit data types are specified, two consecutive 16-bit registers will be used. For example, if D0 is declared as type DWord, registers D0 and D1 will both be used.

* 3 These addresses may be also be referenced as arrays. The syntax for declaring an array, using data registers as an example, is: Dxxxx[rows][cols] or Dxxxx[cols] with an assumed row count of 1. For Word, Short and BCD arrays, the base address + (rows* cols) cannot exceed 0x803F. For DWord, Long and LBCD, the base address + (rows* cols* 2) cannot exceed 0x803F. In all cases, the total number of bytes being requested cannot exceed the block size. **See Also:** Block Size.

* 4 ASCII strings can be stored in data registers. When using data registers for string data, each register will contain two bytes of ASCII data. Characters are stored in "low byte to high byte" order. For example, writing the string "ABCD" to D0.4 would result in D0=0x4142 and D1=0x4344.

Open Model Address Descriptions

The **Flex-PC N Series Open** model option allows this driver to handle a wide range of Fuji N series PLCs without specific control over the range of addresses available from the device. If you are using an NB0 device, you must select the Flex-PC NB0 model. The address ranges shown below may exceed the range available for your particular device. If an address is requested that is not supported by your device, the Fuji Flex Driver will mark the requested data item in error. All addresses are in hexadecimal. Bit numbers, string lengths, and array dimensions are in decimal. Default data types are shown in **bold**.

Address Type	Range	Data Type	Access
Input relay (as bits)	X0–XFFFF	**Boolean**	Read Only
Input relay (as words)*	WX0–WXFFFF WX0.b–WXFFFF.b (b is bit number 0–15)	**Word**, Short* 3 **Boolean**	Read Only
Output relay (as bits)	Y0–YFFFF	**Boolean**	Read/Write
Output relay (as words)*	WY0–WYFFFF WY0.b–WYFFFF.b (b is bit number 0–15)	**Word**, Short* 3 **Boolean**	Read/Write
Internal relay (as bits)	M0–MFFFF	**Boolean**	Read/Write
Internal relay (as words)*	WM0–WMFFFF WM0.b–WMFFFF.b (b is bit number 0–15)	**Word**, Short* 3 **Boolean**	Read/Write
Latch relay (as bits)	L0–LFFFF	**Boolean**	Read/Write
Latch relay (as words)*	WL0–WLFFFF WL0.b–WLFFFF.b (b is bit number 0–15)	**Word**, Short* 3 **Boolean**	Read/Write
Timer contact	T0–TFFFF	**Boolean**	Read/Write
Timer current value	WT0–WTFFFF	**Word**, Short, BCD* 3	Read/Write
Counter contact	C0–CFFFF	**Boolean**	Read/Write
Counter current value	WC0–WCFFFF	**Word**, Short, BCD* 3	Read/Write
Step relay	S0–SFFFF	**Boolean**	Read/Write
Data register	D0–DFFFF D0–DFFFE D0.b–DFFFF.b (b is bit number 0–15) D0.l–D7FFF.l (l is string length 0–64)	**Word**, Short, BCD* 3 DWord, Long, LBCD* 2,* 3 **Boolean** String* 4	Read/Write

* When addressing discrete data as words, addresses correspond to word offsets. For example, WX0 references X0–XF, and WX1 references X10–X1F, etc.

* 2 When 32-bit data types are specified, two consecutive 16-bit registers will be used. For example, if D0 is declared as type DWord, registers D0 and D1 will both be used.

* 3 These addresses may be also be referenced as arrays. The syntax for declaring an array, using data registers as an example, is: Dxxxx[rows][cols] or Dxxxx[cols] with an assumed row count of 1. For Word, Short and BCD arrays, the base address + (rows* cols) cannot exceed 0x803F. For DWord, Long and LBCD, the base address + (rows* cols* 2) cannot exceed 0x803F. In all cases, the total number of bytes being requested cannot exceed the block size. ***See Also:*** ***Block Size***.

* 4 ASCII strings can be stored in data registers. When using data registers for string data, each register will contain two bytes of ASCII data. Characters are stored in "low byte to high byte" order. For example, writing the string "ABCD" to D0.4 would result in D0=0x4142 and D1=0x4344.

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 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>'

Driver Specific Messages

Device '<device name>' returned 'Parameter Error' probably caused by a bad address in block (Tag: '<address>', Size: <size> bytes). Block deactivated
Device '<device name>' returned 'Processing is impossible due to transmission interlock by another device or loader' (Tag: '<address>', Size: <size> bytes)
Device '<device name>' returned 'Incorrect module number' probably due to unsupported memory type (Tag: '<address>', Size: <size> bytes). Block deactivated
Device '<device name>' returned 'Address exceeding the module's range was specified during write' (Tag: '<address>', Size: <size> bytes)
Device '<device name>' returned 'Connection to network is impossible' (Tag: '<address>', Size: <size> bytes)
Device '<device name>' returned 'Another loader is communicating over the network' (Tag: '<address>', Size: <size> bytes)
Device '<device name>' returned 'Transmission error' (Tag: '<address>', Size: <size> bytes)
Device '<device name>' returned error code: '<code>' (Tag: '<address>', Size: <size> bytes)
Received a bad check sum (Device: '<device name>', Tag: '<address>', Size: <size> bytes)
Response had unexpected format (Device: '<device name>', Tag: '<address>', Size: <size> bytes)
Response had incorrect data size (Device: `<device name>', Tag: '<tag name>', Size: <size> bytes)

Missing address

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically has no length.

Solution:
Re-enter the address in the client application.

Device address '<address>' contains a syntax error

Error Type:
Warning

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

Solution:
Re-enter the address in the client application.

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

Error Type:
Warning

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

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

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

Error Type:
Warning

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

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

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

Error Type:
Warning

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

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

Device address '<address>' is Read Only

Error Type:
Warning

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

Solution:
Change the access mode in the client application.

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 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 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. 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.

Device '<device name>' returned 'Parameter Error' probably caused by a bad address in block (Tag: '<address>', Size: <size> bytes). Block deactivated

Error Type:
Warning

Possible Cause:
It's likely that an attempt was 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.

Device '<device name>' returned 'Processing is impossible due to transmission interlock by another device or loader' (Tag: '<address>', Size <size> bytes)

Error Type:
Warning

Possible Cause:
An attempt was made to communicate with a device that has already established a 1-to-1 link with another device.

Solution:
Disconnect the offending device from the target device.

Device '<device name>' returned 'Incorrect module number' probably due to unsupported memory type (Tag: '<address>', Size: <size> bytes). Block deactivated

Error Type:
Warning

Possible Cause:
It's likely that an attempt was 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.

Device '<device name>' returned 'Connection to network is impossible' (Tag: '<address>', Size: <size> bytes)

Error Type:
Warning

Possible Cause:
The device could not connect to the network.

Solution:
Verify communications unit settings.

Device '<device name>' returned 'Address exceeding the module's range was specified during write' (Tag: '<address>', Size: <size> bytes)

Error Type:
Warning

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

Solution:

Unless the offending tag is not active in the client, the driver will detect this problem during a read and deactivate the tag's block. Verify the tags assigned to addresses in the specified range on the device and eliminate ones that reference invalid locations.

Device '<device name>' returned 'Another loader is communicating over the network' (Tag: '<address>', Size: <size> bytes)

Error Type:

Warning

Possible Cause:

An attempt was made to communicate with a device that has already established a 1-to-1 link with a loader.

Solution:

Disconnect the loader.

Device '<device name>' returned 'Transmission error' (Tag: '<address>', Size: <size> bytes)

Error Type:

Warning

Possible Cause:

A communications error was detected by the device.

Solution:

The driver will automatically retry the request. If error message is frequent, take measures to reduce noise.

See Also:

Device Setup

Device '<device name>' returned error code: '<code>' (Tag: '<address>', Size: <size> bytes)

Error Type:

Warning

Possible Cause:

The device encountered an error while processing a request.

Solution:

Refer to device documentation for meaning of error code and take appropriate actions.

Received a bad check sum (Device: '<device name>', Tag: '<address>', Size: <size> bytes)

Error Type:

Warning

Possible Cause:
The driver detected a communications error.

Solution:
The driver will automatically retry the request. If error message is frequent, take measures to reduce noise.

See Also:
Setup

Response had unexpected format (Device: '<device name>', Tag: '<address>', Size: <size> bytes)

Error Type:
Warning

Possible Cause:
The device issued a response that is not in the format expected by the driver.

Solution:
Contact Technical Support.

Response had incorrect data size (Device: '<device name>', Tag: '<tag name>', Size: <size> bytes)

Error Type:
Warning

Possible Cause:
The device issued a response that did not contain the expected number of data points.

Solution:
It's likely that an attempt was made to reference a nonexistent location in the specified device. Verify the tags assigned to addresses in the specified range on the device and eliminate ones that reference invalid locations.

Index

A

B

C

D

E

G

I

L

M

N

O

P

R

S

T

U

W