Yokogawa MX Driver

Table of Contents

Yokogawa MX Driver

Help version 1.030

CONTENTS

Overview

The Yokogawa MX Driver provides a reliable way to connect Yokogawa MX Ethernet devices to OPC client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with Yokogawa MX Data Acquisition devices that support Ethernet TCP communications.

Setup

Supported Devices
MX100

Timing
Device timing properties are common to the server and described in the server help under Device Properties – Timing. Below are ranges and defaults that the driver has customized:
Request Timeout: Valid range is 100 to 30000 milliseconds. The default setting is 5000 milliseconds.

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 1024 per channel.

Device ID
Yokogawa MX Series devices are networked using standard IP addressing. In general, the Device ID has the format YYY.YYY.YYY.YYY, where YYY designates the device IP address. Each YYY byte should be in the range of 0 to 255.

Channel Properties — General

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

Property Groups	☐ **Identification**	
General	Name	
Write Optimizations	Description	
Advanced	Driver	
	☐ **Diagnostics**	
	Diagnostics Capture	Disable
	☐ **Tag Counts**	
	Static Tags	10

Identification

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

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

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

Diagnostics

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

Diagnostics

Diagnostics Capture: When enabled, this option allows the usage of statistics tags that provide feedback to client applications regarding the operation of the channel. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default is disabled.

 Note: This property is not available if the driver does not support diagnostics.

 For more information, refer to Statistics Tags in the server help.

Tag Counts

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

Channel Properties — Ethernet Communications

Ethernet Communication can be used to communicate with devices.

Ethernet Settings

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

Channel Properties — Write Optimizations

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

Write Optimizations

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

- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing data to the device as quickly as possible. This mode ensures that everything written from the client applications is sent to the target device. This mode should be selected if the write operation order or the write item's content must uniquely be seen at the target device.
- **Write Only Latest Value for Non-Boolean Tags**: Many consecutive writes to the same value can accumulate in the write queue due to the time required to actually send the data to the device. If the server updates a write value that has already been placed in the write queue, far fewer writes are needed to reach the same final output value. In this way, no extra writes accumulate in the server's queue. When the user stops moving the slide switch, the value in the device is at the correct value at virtually the same time. As the mode states, any value that is not a Boolean value is updated in the server's internal write queue and sent to the device at the next possible opportunity. This can greatly improve the application performance.
  Note: This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.
- **Write Only Latest Value for All Tags**: This option takes the theory behind the second optimization mode and applies it to all tags. It is especially useful if the application only needs to send the latest

value to the device. This mode optimizes all writes by updating the tags currently in the write queue before they are sent. This is the default mode.

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

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

Channel Properties — Advanced

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

Property Groups	Non-Normalized Float Handling	
General	Floating-Point Values	Replace with Zero
Write Optimizations	Inter-Device Delay	
Advanced	Inter-Device Delay (ms)	0

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

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

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

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

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

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

Device Properties — General

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

Property Groups	⊟ **Identification**	
General	Name	
Scan Mode	Description	
	Channel Assignment	
	Driver	
	Model	
	ID Format	Decimal
	ID	2

Identification

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

 Note: Although descriptive names are generally a good idea, some OPC client applications may have a limited display window when browsing the OPC server's tag space. The device name and channel name become part of the browse tree information as well. Within an OPC client, the combination of channel name and device name would appear as "ChannelName.DeviceName".
 For more information, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in server help.

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

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

Driver: Selected protocol driver for this device.

Model: Specify the type of device that is associated with this ID. The contents of the drop-down menu depend on the type of communications driver being used. Models that are not supported by a driver are disabled. If the communications driver supports multiple device models, the model selection can only be changed when there are no client applications connected to the device.
 Note: If the communication driver supports multiple models, users should try to match the model selection to the physical device. If the device is not represented in the drop-down menu, select a model that conforms closest to the target device. Some drivers support a model selection called "Open," which allows users to communicate without knowing the specific details of the target device. *For more information, refer to the driver documentation.*

ID: Specify the device's driver-specific station or node. The type of ID entered depends on the communications driver being used. For many communication drivers, the ID is a numeric value. Drivers that support a Numeric ID provide users with the option to enter a numeric value whose format can be changed to suit the needs of the application or the characteristics of the selected communications driver. The format is set by the driver by default. Options include Decimal, Octal, and Hexadecimal.
 Note: If the driver is Ethernet-based or supports an unconventional station or node name, the device's TCP/IP address may be used as the device ID. TCP/IP addresses consist of four values that are separated by periods, with each value in the range of 0 to 255. Some device IDs are string based. There may be additional properties to configure within the ID field, depending on the driver.

Operating Mode

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

Simulated: Place the device into or out of Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client as valid data. While in Simulation Mode, the server treats all device data as reflective: whatever is written to the simulated device is read back and each OPC item is treated individually. The data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

⚙ **Notes:**

1. Updates are not applied until clients disconnect and reconnect.

2. The System tag (_Simulated) is read only and cannot be written to for runtime protection. The System tag allows this property to be monitored from the client.

3. In Simulation mode, the item's memory map is based on client update rate(s) (Group Update Rate for OPC clients or Scan Rate for native and DDE interfaces). This means that two clients that reference the same item with different update rates return different data.

4. When a device is simulated, updates may not appear faster than one (1) second in the client.

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

Tag Counts

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

Device Properties — Scan Mode

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

Property Groups	⊟ Scan Mode	
General	Scan Mode	Respect Client-Specified Scan Rate ▾
Scan Mode	Initial Updates from Cache	Disable

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

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

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

Device Properties — Timing

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

Property Groups	⊟ Communication Timeouts	
General	Connect Timeout (s)	3
Scan Mode	Request Timeout (ms)	1000
Timing	Attempts Before Timeout	3

Communications Timeouts

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

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

Request Timeout: Specify an interval used by all drivers to determine how long the driver waits for a response from the target device to complete. The valid range is 50 to 9999999 milliseconds (167 minutes). The default is usually 1000 milliseconds, but can vary depending on the driver. The default timeout for most serial drivers is based on a baud rate of 9600 baud or better. When using a driver at lower baud rates, increase the timeout to compensate for the increased time required to acquire data.

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

Timing

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

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

Property Groups	⊟ Timing	
General	Inter-Request Delay (ms)	0
Scan Mode		
Timing		

Device Properties — Auto-Demotion

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

Property Groups	⊟ Auto-Demotion	
General	Demote on Failure	**Enable** ▼
Scan Mode	Timeouts to Demote	3
Timing	Demotion Period (ms)	10000
Auto-Demotion	Discard Requests when Demoted	Disable

Demote on Failure: When enabled, the device is automatically taken off-scan until it is responding again.

⬤ **Tip**: Determine when a device is off-scan by monitoring its demoted state using the _AutoDemoted system tag.

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

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

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

Device Properties — Tag Generation

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

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

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

1. If a data acquisition system supports its own local tag database, the communications driver uses the tag names found in the device to build the server's tags.

2. If an Ethernet I/O system supports detection of its own available I/O module types, the communications driver automatically generates tags in the server that are based on the types of I/O modules plugged into the Ethernet I/O rack.

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

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

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

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

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

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

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

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

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

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

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

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

Allow Automatically Generated Subgroups: This property controls whether the server automatically creates subgroups for the automatically generated tags. This is the default setting. If disabled, the server gen-

erates the device's tags in a flat list without any grouping. In the server project, the resulting tags are named with the address value. For example, the tag names are not retained during the generation process.

⬤ **Note**: If, as the server is generating tags, a tag is assigned the same name as an existing tag, the system automatically increments to the next highest number so that the tag name is not duplicated. For example, if the generation process creates a tag named "AI22" that already exists, it creates the tag as "AI23" instead.

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

⬤ **Note**: **Create tags** is disabled if the Configuration edits a project offline.

Device Properties — Device Configuration

⊟ **General**		
Stop MX on Shutdown	Disable	
Special Data Handling	-INF/+INF	
⊟ **Time Settings**		
Date and Time	Device Time	
Date Format	MM/DD/YY	

Stop MX on Shutdown: When enabled, this option will inform the driver to send a command to the device to stop measuring when the server is exited. The default setting is disabled.

Special Data Handling: This property specifies how to handle special data values for channels that return a special condition status (such as "Data out of range"). Options include -INF/+INF or -99999/+99999. A negative sign indicates an under range condition, whereas a positive sign is used for all other special conditions (including over range conditions). The default setting is -INF/+INF.

🔧 **Note**: The client software being used may not have the ability to interpret -INF or +INF values. In this case, users will need to select -99999/+99999.

Date and Time: This property specifies the origin of the data value of the Date and Time data types (which represent the date and time of the latest data). Options include Device Time and System Time. The default setting is Device Time. Descriptions of the options are as follows:

- **Device Time**: When selected, the Date and Time tags will return the date and time read from the device. This date and time represents the date and time that the latest data was measured or computed based on the device.
- **System Time**: When selected, the Date and Time tags will return the date and time that the requested data was returned from the device based on the internal system clock.

Date Format: This property specifies the format of the return string for the Date data type. Options include MM/DD/YY (month/day/year), YY/MM/DD (year/month/day), or DD/MM/YY (day/month/year). The default setting is MM/DD/YY.

Device Properties — Redundancy

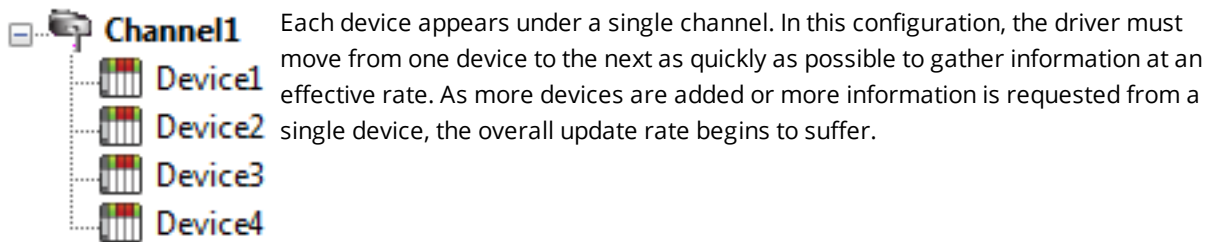|		
⊟ **Redundancy**		
Secondary Path	**Channel.Device1**	...
Operating Mode	Switch On Failure	
Monitor Item		
Monitor Interval (s)	300	
Return to Primary ASAP	Yes	

Redundancy is available with the Media-Level Redundancy Plug-In.

🔧 *Consult the website, a sales representative, or the **user manual** for more information.*
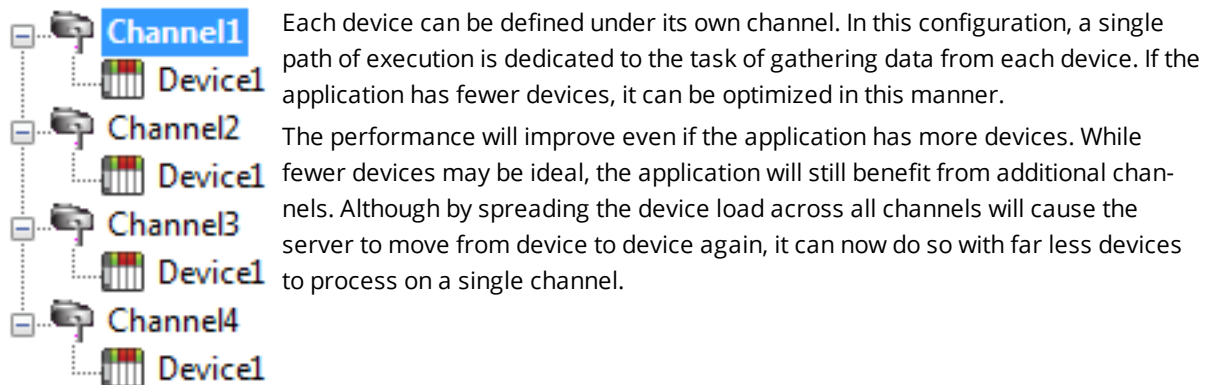
Optimizing Communications

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

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

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

If the Yokogawa MX Driver could only define one single channel, then the example shown above would be the only option available; however, the Yokogawa MX Driver can define up to 100 channels. Using multiple channels distributes the data collection workload by simultaneously issuing multiple requests to the network. An example of how the same application may appear when configured using multiple channels to improve performance is shown below.

 Each device can be defined under its own channel. In this configuration, a single path of execution is dedicated to the task of gathering data from each device. If the application has fewer devices, it can be optimized in this manner.

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

Data Types Description

Data Type	Description
Boolean	Single bit
Byte	Unsigned 8-bit value bit 0 is the low bit bit 7 is the high bit

Data Type	Description
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
Long	Signed 32-bit value
Float	32-bit floating point value
Double	64-bit floating point value
String	Null-terminated ASCII string

MX100 Addressing

The driver supports the following addresses for this device. The default data type for each address is shown in **bold**.

Measured Channels

Process Value of Channel	CHxxxxx or CHxxxxx.PV	00001-00060	**Double**, Float	Read Only
Alarm Summary of Channel	CHxxxxx.Alarm	00001-00060	**Short**, Word, Byte	Read Only
Alarm Level1 Status of Channel	CHxxxxx.Alarm1	00001-00060	**Short**, Word, Byte	Read Only
Alarm Level2 Status of Channel	CHxxxxx.Alarm2	00001-00060	**Short**, Word, Byte	Read Only
Alarm Level1 Setpoint*	CHxxxxx.ASP1	00001-00060	**Double**, Float	Read Only
Alarm Level2 Setpoint*	CHxxxxx.ASP2	00001-00060	**Double**, Float	Read Only
Scan interval of Channel*	CHxxxxx.interval	00001-00060	**Short**, Word, Byte	Read Only
Upper Scale Value of Channel*	CHxxxxx.scale_Hi	00001-00060	**Double**, Float	Read Only
Lower Scale Value of Channel*	CHxxxxx.scale_Lo	00001-00060	**Double**, Float	Read Only
Status of Channel	CHxxxxx.status	00001-00060	**Short**, Word, Byte	Read Only
Tag name of Channel*	CHxxxxx.tag	00001-00060	**String**	Read Only
Tag comment of Channel*	CHxxxxx.comment	00001-00060	**String**	Read Only
Unit String of Channel*	CHxxxxx.unit	00001-00060	**String**	Read Only
Precision of Channel*	CHxxxxx.Precision	00001-00060	**Short**, Word, Byte	Read Only
Digital Output on Channel	CHxxxxx.DO	00001-00060	**Boolean**, Short, Word, Byte	Read/Write
Analog Output (Volt) on Channel	CHxxxxx.AOVolt	00001-00060	**Double**, Float	Read/Write
Analog Output (mA) on Channel	CHxxxxx.AOmA	00001-00060	**Double**, Float	Read/Write
Pulse Width Modulation on Channel	CHxxxxx.PWM	00001-00060	**Double**, Float	Read/Write
Set Retransmission Start/Stop for Channel	CHxxxxx.Retransmission	00001-0060	**Boolean**, Short, Word, Byte	Read/Write
Lowest Measuring Channel*	CH.Low		**Short**, Word, Byte	Read Only

Highest Measuring Channel*	CH.High		**Short**, Word, Byte	Read Only

*The data that is associated with these addresses will only be read at the start of a communications session. Once read, the values will not be refreshed until the server has been restarted or the "Reset" tag has been invoked. To invoke a reset, a non zero value must be written to the Reset tag. Once the Reset tag has been invoked, the driver will reinitialize all startup data from the device.

◗**Note:** This will result in a pause of the update of process values and alarms until all initialized data has been read from the device.

Channel Addressing

The Yokogawa MX Ethernet Device is arranged as a 6 slot (numbers 0 to 5) unit. Each slot may hold a module consisting of up to 10 channels (1 to 10). The first slot has possible channel numbers of CH00001-00010, the second slot CH000011-00020, the third slot CH00021-00030, the fourth slot CH00031-00040, the fifth slot CH00041-00050, and the last slot has possible channel numbers of 00051-00060. There is potential for gaps in channel numbering because not all slots must contain a module, and not all modules are equipped with 10 channels (as in the case of a 4-channel module).

Tag Names

For channels that have unspecified tag names, the driver will construct an internal tag name based on the channel number. For example, the tag name of address "CH00001" will be returned as "CH00001".

Digital Output

"CHxxxxx.DO" is only valid for channel numbers in slots containing a Digital Output Module.

General Device Data

Address Type	Format	Range	Data Types	Access
Date of Last Data	Date		**String**	Read Only
Time of Last Data	Time		**String**	Read Only
7 segment display pattern	Dispx	1-2	**Long**, Short, Word	Read/Write
Set display blink	Disp.Blink		**Boolean**, Short, Word, Byte	Write Only
Set interval for display blink (mSec)	Disp.Interval		**Integer**	Write Only
Serial Number of Unit	Unit.Serial		**String**	Read Only
Serial Number of Module	Modulex.Serial	0-5	**String**	Read Only
IP Address of Device	IP		**String**	Read Only
Set backup	Backup		**Boolean**, Short, Word, Byte	Read/Write
Reset Alarms	AlarmReset		**Boolean**, Short, Word, Byte	Write Only
Set Retransmission Start/Stop for all channels	RetransAll		**Boolean**, Short, Word, Byte	Write Only
Direct Reloading of Configuration	Reset		**Boolean**, Short, Word, Byte	Write Only
Unit Style	Unit.Style		**Long**	Read Only

Address Type	Format	Range	Data Types	Access
Unit firmware version in *xx.xx.xx.xx* format	Unit.Firmware		**String**	Read Only

7 Segment Display Pattern

"'Disp1" refers to the left indicator of the display pattern on the front of the unit. "Disp2" refers to the right indicator.

Set Display Blink and Set Interval for Display Blink

Setting display blink on (non-zero value) will cause the display on the front of the unit to blink for the set interval.

Write Only Items

Addresses that have Write Only access will be assigned a default access of Read/Write; however, data values will be unreadable for these addresses (and the associated tags will not be included in the scan list). The current data value for these tags will always be 0 for numeric data types with the exception of "Disp.Interval," which has a default value of 1000.

 Note: The actual number of addresses available for of each type depends on the configuration of the Yokogawa device. If the driver finds that an address is not present in the device at Runtime, it will post an error message and remove the tag from its scan list.

Error Descriptions

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

Address Validation

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

Device Status Messages

Detected Yokogawa MX 100 Unit Style '<version>' with '<number of modules>' modules and a total of '<number of channels>' channels on device '<device name>'
Device '<device>' response contains framing errors (Tag '<tag address>')
Device '<device name>' has no FIFO data for channel (Tag '<address>')
Device '<device name>' is not responding
Device '<device name>' responded with error '<error code>' (Tag '<address>', Size '<size>'
Unable to write to '<address>' on device '<device name>

Driver Error Messages

Winsock initialization failed (OS Error = n)
Winsock V1.1 or higher must be installed to use the Yokogawa MX Ethernet device driver

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.

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>' 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.

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.

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.

Detected Yokogawa MX 100 Unit Style '<version>' with '<number of modules>' modules and a total of '<number of channels>' channels on device '<device name>'

Error Type:
Informational

Possible Cause:
Communications have occurred with the device, and the unit and channel information requests have completed successfully.

Solution:
N/A.

⬤ Note:
This message provides a summary of the device configuration.

Device '<device>' response contains framing errors (Tag '<tag address>')

Error Type:

Warning

Possible Cause:

The Yokogawa MX Ethernet device response contained unexpected data. This could be caused by the following:

1. The packets are misaligned due to the connection between the PC and device.

2. Bad cabling between the PC and controller is causing noise.

Solution:

The driver will recover from the error without user intervention. If this error occurs frequently, there may be an issue with the cabling or the device itself.

Device '<device name>' has no FIFO data for channel (Tag '<address>')

Error Type:

Serious

Possible Cause:

Requested FIFO data is not available from the device.

Solution:

N/A

Device '<device name>' is not responding

Error Type:

Serious

Possible Cause:

1. The connection between the device and the Host PC is broken.

2. The IP address assigned to the device is incorrect.

3. The device Ethernet port is already in use.

4. The connection cannot be established in the specified timeout period.

5. 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 IP address given to the named device matches that configured in the actual device. If applicable, verify the subnet mask and default gateway properties configured in the actual device.

3. Check for another connected application (such as MX Standard software) and disconnect. The MX device's TCP/IP port only supports a single connection.

4. Increase the Connect Timeout value in the Timeout property group of Device Properties.

5. Increase the Request Timeout property value so that the entire response can be handled.

Device '<device name>' responded with error '<error code>' (Tag '<address>', Size '<size>')

Error Type:
Serious

Possible Cause:
The Yokogawa MX Ethernet device responded with the indicated error code.

Solution:
The solution will depend upon the device error code.

See Also:
The Yokogawa MX Ethernet documentation of the specific error code.

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

Error Type:
Serious

Possible Cause:

1. The connection between the device and the Host PC is broken.

2. The named device may have been assigned an incorrect IP address.

3. The address specified may be Read Only or may not exist in the current device.

Solution:

1. Verify the cabling between the PC and the PLC device.

2. Verify the IP address given to the named device matches that of the actual device.

3. Check address availability for the device.

Winsock initialization failed (OS Error = n)

Error Type:
Fatal

OS Error	Indication	Possible Solution
10091	Indicates that the underlying network sub-system is not ready for network communication.	Wait a few seconds and restart the driver.
10067	Limit on the number of tasks supported by the Windows Sockets implementation has been reached.	Close one or more applications that may be using Winsock and restart the driver.

Winsock V1.1 or higher must be installed to use the Yokogawa MX Ethernet device driver

Error Type:

Fatal

Possible Cause:

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

Solution:

Upgrade Winsock to version 1.1 or higher.

Index

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

Short 18

Simulated 10

T

Tag Counts 6, 10

Tag Generation 13

Timeouts to Demote 13

Timing 11

U

Unable to write tag '<address>' on device '<device name>' 25

Unmodified 8

W

Winsock initialization failed (OS Error = n) 25

Winsock V1.1 or higher must be installed to use the Yokogawa MX Ethernet device driver 26

Word 18

Write All Values for All Tags 7

Write Only Latest Value for All Tags 8

Write Only Latest Value for Non-Boolean Tags 7