Mitsubishi FX Driver

Table of Contents

Mitsubishi FX Driver

Help version 1.041

CONTENTS

Overview

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

Setup

Supported Device Models
FX
FX0
FX0N
FX2N
FX3U

⬤ **Note:** FX3U is not supported in Windows CE.

Communication Protocol
Direct Serial

Supported Communication Parameters
Baud Rate: 9600
Parity: Even
Data Bits: 7
Stop Bits: 1

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

When used directly with a serial port, this driver supports only a single connection to a single controller per serial port. When operating in the Ethernet Encapsulation mode, the driver supports up to 100 controllers per channel. In this mode, a single controller can be paired with a terminal server (device server) to form a single node. For more information, refer to **Appendix: Ethernet Encapsulation Examples**.

Note: Ethernet Encapsulation is not supported by the FX3U model.

Channel and Device Limits

The maximum number of channels supported by this driver is 256. The maximum number of devices supported by this driver is 100 per channel.

Device IDs

This protocol does not support simultaneous communication with multiple devices.

Flow Control

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

Note: When using the manufacturer's supplied communications cable, it is sometimes necessary to choose a flow control setting of **RTS** or **RTS Always** in the channel properties.

Channel Properties

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.*

Tag Counts

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

Channel Properties — Serial Communications

Serial communication properties are available to serial drivers and vary depending on the driver, connection type, and options selected. Below is a superset of the possible properties.
Click to jump to one of the sections: **Connection Type**, **Serial Port Settings** or **Ethernet Settings**, and **Operational Behavior**.

⚪ **Notes**:

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

- Users must define the specific communication parameters to be used. Depending on the driver, channels may or may not be able to share identical communication parameters. Only one shared serial connection can be configured for a Virtual Network *(see **Channel Properties — Serial Communications**)*.

Property Groups			
General	⊟	**Connection Type**	
Serial Communications		Physical Medium	COM Port
Write Optimizations	⊟	**Serial Port Settings**	
Advanced		COM ID	39
		Baud Rate	19200
		Data Bits	8
		Parity	None
		Stop Bits	1
		Flow Control	RTS Always
	⊟	**Operational Behavior**	
		Report Communication Errors	Enable
		Close Idle Connection	Enable
		Idle Time to Close (s)	15

Connection Type

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

1. **None**: Select None to indicate there is no physical connection, which displays the **Operation with no Communications** section.

2. **COM Port**: Select Com Port to display and configure the **Serial Port Settings** section.

3. **Modem**: Select Modem if phone lines are used for communications, which are configured in the **Modem Settings** section.

4. **Ethernet Encap.**: Select if Ethernet Encapsulation is used for communications, which displays the **Ethernet Settings** section.

5. **Shared**: Verify the connection is correctly identified as sharing the current configuration with another channel. This is a read-only property.

Serial Port Settings

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

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

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

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

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

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

- **None**: This option does not toggle or assert control lines.
- **DTR**: This option asserts the DTR line when the communications port is opened and remains on.

- **RTS**: This option specifies that the RTS line is high if bytes are available for transmission. After all buffered bytes have been sent, the RTS line is low. This is normally used with RS232/RS485 converter hardware.
- **RTS, DTR**: This option is a combination of DTR and RTS.
- **RTS Always**: This option asserts the RTS line when the communication port is opened and remains on.
- **RTS Manual**: This option asserts the RTS line based on the timing properties entered for RTS Line Control. It is only available when the driver supports manual RTS line control (or when the properties are shared and at least one of the channels belongs to a driver that provides this support).
  RTS Manual adds an **RTS Line Control** property with options as follows:
  - **Raise**: Specify the amount of time that the RTS line is raised prior to data transmission. The valid range is 0 to 9999 milliseconds. The default is 10 milliseconds.
  - **Drop**: Specify the amount of time that the RTS line remains high after data transmission. The valid range is 0 to 9999 milliseconds. The default is 10 milliseconds.
  - **Poll Delay**: Specify the amount of time that polling for communications is delayed. The valid range is 0 to 9999. The default is 10 milliseconds.

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

Operational Behavior

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

Ethernet Settings

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

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

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

Modem Settings

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

Operation with no Communications

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

Channel Properties — Write Optimizations

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

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.

Identification	
Name	
Description	
Channel Assignment	
Driver	
Model	
ID Format	Decimal
ID	2

Property Groups: General, Scan Mode

Identification

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

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

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

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

Driver: Selected protocol driver for this device.

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

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

periods, with each value in the range of 0 to 255. Some device IDs are string based. There may be additional properties to configure within the ID field, depending on the driver.

Operating Mode

Property Groups	Identification	
General	**Operating Mode**	
Scan Mode	Data Collection	Enable
	Simulated	No

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

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

 Notes:

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

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

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

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

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

Tag Counts

Property Groups	Identification	
General	Operating Mode	
	Tag Counts	
	Static Tags	130

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

Device Properties — Scan Mode

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

unaffected by the Scan Mode properties.

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 sub-scribed 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.

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 sup-
ported by the driver, it is disabled.
● **Note**: Due to the nature of UDP connections, the connection timeout setting is not applicable when com-
municating 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 con-
sidering the request to have failed and the device to be in error. The valid range is 1 to 10. The default is typ-
ically 3, but can vary depending on the driver's specific nature. The number of attempts configured for an
application depends largely on the communications environment. This property applies to both connection
attempts and request attempts.

Timing

Inter-Request Delay: Specify how long the driver waits before sending the next request to the target
device. It overrides the normal polling frequency of tags associated with the device, as well as one-time
reads and writes. This delay can be useful when dealing with devices with slow turnaround times and in
cases where network load is a concern. Configuring a delay for a device affects communications with all
other devices on the channel. It is recommended that users separate any device that requires an inter-
request delay to a separate channel if possible. Other communications properties (such as communication
serialization) can extend this delay. The valid range is 0 to 300,000 milliseconds; however, some drivers may
limit the maximum value due to a function of their particular design. The default is 0, which indicates no
delay between requests with the target device.
● **Note**: Not all drivers support Inter-Request Delay. This setting does not appear if it is not available.

Device Properties — Auto-Demotion

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

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

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

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

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

Device Properties — Redundancy

Property Groups	⊟ **Redundancy**	
General	Secondary Path	**Channel.Device 1** ...
Scan Mode	Operating Mode	Switch On Failure
Timing	Monitor Item	
Auto-Demotion	Monitor Interval (s)	300
Tag Generation	Return to Primary ASAP	Yes
Tag Import Settings		
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

The Mitsubishi FX Driver supports the following data types.

Data Type	Description
Boolean	Single bit
Word	Unsigned 16-bit value bit 0 is the low bit bit 15 is the high bit
Short	Signed 16-bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
DWord	Unsigned 32-bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
Float	32-bit Floating point value The driver interprets two consecutive registers as a single precision value by making the last register the high word and the first register the low word.
String	Null terminated ASCII string Support includes HiLo LoHi byte order selection and string lengths up to 64 bytes.

Address Descriptions

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

FX Addressing
FX0 Addressing
FX0N Addressing
FX2N Addressing
FX3U Addressing

FX Addressing

The default data types for dynamically defined tags are shown in **bold**.

Device Type	Range	Data Type	Access
Inputs	X000-X377*	**Boolean**	Read Only
Outputs	Y000-Y377*	**Boolean**	Read/Write
Auxiliary Relays	M0000-M1535	**Boolean**	Read/Write
Special Aux. Relays	M8000-M8255	**Boolean**	Read/Write
States	S000-S999	**Boolean**	Read/Write
Timer Contacts	TS000-TS255	**Boolean**	Read Only
Counter Contacts	CS000-CS255	**Boolean**	Read Only
Timer Reset	TR000-TR255	**Boolean**	Read/Write
Counter Reset	CR000-CR255	**Boolean**	Read/Write
Timer Value	T000-T255	Short, **Word**	Read/Write
Counter Value	C000-C199	Short, **Word**	Read/Write
32-bit Counter Value**	C200-C255	Long, **DWord**	Read/Write
Data Registers**	D000-D999 D000-D998	**Short**, Word, Long, DWord, Float	Read/Write
Data Registers String Access HiLo Byte Ordering	DSH0000.02-DSH0967.64 The string length may also be specified using a colon. The string length can range from 2 to 64 bytes, and must be an even number.***	**String**	Read/Write
Data Registers String Access LoHi Byte Ordering	DSL0000.02-DSL0967.64 The string length may also be specified using a colon. The string length can range from 2 to 64 bytes, and must be an even number.***	**String**	Read/Write

Device Type	Range	Data Type	Access
Special Data Registers**	D8000-D8255 D8000-D8254	**Short**, Word, Long, DWord, Float	Read/Write

*Octal

**Users can specify a Long data type by appending a space and an "L" to the address. For example, "D000" would be entered as "D000 L". This does not apply to arrays or bit accessed registers.

***For example, a string with HiLo byte ordering at D10 for a length of 12 bytes would be DSH10.12.

FX0 Addressing

The default data types for dynamically defined tags are shown in **bold**.

Device Type	Range	Data Type	Access
Inputs	X000-X017*	**Boolean**	Read Only
Outputs	Y000-Y15*	**Boolean**	Read/Write
Auxiliary Relays	M0000-M0511	**Boolean**	Read/Write
Special Aux. Relays	M8000-M8255	**Boolean**	Read/Write
States	S00-S63	**Boolean**	Read/Write
Timer Contacts	TS00-TS55	**Boolean**	Read Only
Counter Contacts	CS00-CS15 CS235-CS254	**Boolean**	Read Only
Timer Reset	TR00-TR55	**Boolean**	Read/Write
Counter Reset	CR00-CR15 CR235-CR254	**Boolean**	Read/Write
Timer Value	T00-T55	Short, **Word**	Read/Write
Counter Value	C00-C15	Short, **Word**	Read/Write
32-bit Counter Value**	C235-C254	Long, **DWord**	Read/Write
Data Registers**	D00-D31 D00-D30	**Short**, Word, Long, DWord, Float	Read/Write
Data Registers String Access HiLo Byte Ordering	DSH0000.02–DSH30.62 The string length may also be specified using a colon. The string length can range from 2 to 62 bytes, and must be an even number.***	**String**	Read/Write

Device Type	Range	Data Type	Access
Data Registers String Access LoHi Byte Ordering	DSL0000.02–DSL30.62 The string length may also be specified using a colon. The string length can range from 2 to 62 bytes, and must be an even number.***	**String**	Read/Write
Special Data Registers**	D8000-D8069 D8000-D8068	**Short**, Word Long, DWord, Float	Read/Write

*Octal.

**Users can specify a Long data type by appending a space and an "L" to the address. For example, "D00" would be entered as "D00 L". This does not apply to arrays or bit accessed registers.

***For example, a string with HiLo byte ordering at D10 for a length of 12 bytes would be DSH10.12.

FX0N Addressing

The default data types for dynamically defined tags are shown in **bold**.

Device Type	Range	Data Type	Access
Inputs	X000-X177*	**Boolean**	Read Only
Outputs	Y000-Y177*	**Boolean**	Read/Write
Auxiliary Relays	M0000-M0511	**Boolean**	Read/Write
Special Aux. Relays	M8000-M8255	**Boolean**	Read/Write
States	S000-S127	**Boolean**	Read/Write
Timer Contacts	TS00-TS63	**Boolean**	Read Only
Counter Contacts	CS00-CS31 CS235-CS254	**Boolean**	Read Only
Timer Reset	TR00-TR63	**Boolean**	Read/Write
Counter Reset	CR00-CR31 CR235-CR254	**Boolean**	Read/Write
Timer Value	T00-T63	Short, **Word**	Read/Write
Counter Value	C00-C31	Short, **Word**	Read/Write
32-bit Counter Value**	C235-C254	Long, **DWord**	Read/Write
Data Registers**	D000-D255 D000-D254	**Short**, Word Long, DWord, Float	Read/Write

Device Type	Range	Data Type	Access
Data Registers String Access HiLo Byte Ordering	DSH0000.02-DSH223.64 The string length may also be specified using a colon. The string length can range from 2 to 64 bytes, and must be an even number.***	**String**	Read/Write
Data Registers String Access LoHi Byte Ordering	DSL0000.02-DSL223.64 The string length may also be specified using a colon. The string length can range from 2 to 64 bytes, and must be an even number.***	**String**	Read/Write
Special Data Registers**	D8000-D8255 D8000-D8254	**Short**, Word Long, DWord, Float	Read/Write

*Octal.
**Users can specify a Long data type by appending a space and an "L" to the address. For example, "D000" would be entered as "D000 L". This does not apply to arrays or bit accessed registers.
***For example, a string with HiLo byte ordering at D10 for a length of 12 bytes would be DSH10.12.

FX2N Addressing

The default data types for dynamically defined tags are shown in **bold**.

Device Type	Range	Data Type	Access
Inputs	X000-X377*	**Boolean**	Read Only
Outputs	Y000-Y377*	**Boolean**	Read/Write
Auxiliary Relays	M0000-M3071	**Boolean**	Read/Write
Special Aux. Relays	M8000-M8255	**Boolean**	Read/Write
States	S000-S999	**Boolean**	Read/Write
Timer Contacts	TS000-TS255	**Boolean**	Read Only
Counter Contacts	CS000-CS255	**Boolean**	Read Only
Timer Reset	TR000-TR255	**Boolean**	Read/Write
Counter Reset	CR000-CR255	**Boolean**	Read/Write
Timer Value	T000-T255	Short, **Word**	Read/Write
Counter Value	C000-C199	Short, **Word**	Read/Write
32-bit Counter Value**	C200-C255	Long, **DWord**	Read/Write
Data Registers**	D000-D7999 D000-D7998	**Short**, Word,	Read/Write

Device Type	Range	Data Type	Access
		Long, DWord, Float	
Data Registers String Access HiLo Byte Ordering	DSH0000.02-DSH7967.64 The string length may also be specified using a colon. The string length can range from 2 to 64 bytes, and must be an even number.***	**String**	Read/Write
Data Registers String Access LoHi Byte Ordering	DSL0000.02-DSL7967.64 The string length may also be specified using a colon. The string length can range from 2 to 64 bytes, and must be an even number.***	**String**	Read/Write
Special Data Registers**	D8000-D8255 D8000-D8254	**Short**, Word, Long, DWord, Float	Read/Write

*Octal.
**Users can specify a Long data type by appending a space and an "L" to the address. For example, "D000" would be entered as "D000 L". This does not apply to arrays or bit accessed registers.
***For example, a string with HiLo byte ordering at D10 for a length of 12 bytes would be DSH10.12.

FX3U Addressing

The default data types for dynamically defined tags are shown in **bold**.

🔹 **Note:** The FX3U model is not supported in Windows CE. It also does not support Ethernet Encapsulation.

Device Type	Range	Data Type	Access
Inputs	X000-X377*	**Boolean**	Read Only
Outputs	Y000-Y377*	**Boolean**	Read/Write
Auxiliary Relays	M0000-M7679	**Boolean**	Read/Write
Special Aux. Relays	M8000-M8511	**Boolean**	Read/Write
States	S000-S4095	**Boolean**	Read/Write
Timer Contacts	TS000-TS511	**Boolean**	Read Only
Counter Contacts	CS000-CS255	**Boolean**	Read Only
Timer Reset	TR000-TR511	**Boolean**	Read/Write
Counter Reset	CR000-CR255	**Boolean**	Read/Write
Timer Value	T000-T511	Short, **Word**	Read/Write
Counter Value	C000-C199	Short, **Word**	Read/Write

Device Type	Range	Data Type	Access
32-bit Counter Value**	C200-C255	Long, **DWord**	Read/Write
Data Registers**	D000-D7999 D000-D7998	**Short**, Word, Long, DWord, Float	Read/Write
Data Registers String Access HiLo Byte Ordering	DSH0000.02-DSH7967.64 The string length may also be specified using a colon. The string length can range from 2 to 64 bytes, and must be an even number.***	**String**	Read/Write
Data Registers String Access LoHi Byte Ordering	DSL0000.02-DSL7967.64 The string length may also be specified using a colon. The string length can range from 2 to 64 bytes, and must be an even number.***	**String**	Read/Write
File Registers**	R000-R32767 R000-R32766	**Short**, Word, Long, DWord, Float	Read/Write
File Registers String Access HiLo Byte Ordering	RSH0000.02-RSH32735.64 The string length may also be specified using a colon. The string length can range from 2 to 64 bytes, and must be an even number.***	**String**	Read/Write
File Registers String Access LoHi Byte Ordering	RSL0000.02-RSL32735.64 The string length may also be specified using a colon. The string length can range from 2 to 64 bytes, and must be an even number.***	**String**	Read/Write
Special Data Registers**	D8000-D8511 D8000-D8510	**Short**, Word, Long, DWord, Float	Read/Write

*Octal.

**Users can specify a Long data type by appending a space and an "L" to the address. For example, "D000" would be entered as "D000 L". This does not apply to arrays or bit accessed registers.

***For example, a string with HiLo byte ordering at D10 for a length of 12 bytes would be DSH10.12

Event Log Messages

The following information concerns messages posted to the Event Log pane in the main user interface. Consult the OPC server help on filtering and sorting the Event Log detail view. Server help contains many common messages, so should also be searched. Generally, the type of message (informational, warning) and troubleshooting information is provided whenever possible.

Received block length does not match expected length. | Received block length = <number>, Expected block length = <number>.

Error Type:
Warning

Possible Cause:
The data type maximum length or the length dictated in the address definition set a range that the results do not fit.

Possible Solution:
Verify the data type is correct and check the address definition for a length definition and correct or update.

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

Appendix: Ethernet Encapsulation Examples

🔘 **Note**: The following is provided for convenience only. Refer to the manufacturer's documentation for current and official instructions.

Click on a link below for a specific Ethernet encapsulation example.

Using COM-ET10-T Ethernet Module
Using the FX2NC-ENET-ADP Ethernet Module
Using the FX3U-ENET Ethernet Module
Writing Network Parameters to the PLC

🔘 **Note:** Ethernet Encapsulation is not supported by the FX3U device model.

Using COM-ET10-T Ethernet Module

Mitsubishi Electric has an Ethernet module that allows peer-to-peer Ethernet connection to FX1S, FX1N, FX2NC or FX2NC model controllers. The COM-ET10-T module connects to the controller via the FX1N-CNV-BD or the FX2N-CNV-BD communications adapter. For equipment information, refer to the FX hardware manual.

Example Parameters

```
The Module ID is Fixed at ET10
 E = 45
 T = 54
 1 = 31
 0 = 30


IP Address: 192.168.108.10
 192 = C0
 168 = A8
 108 = 6C
 10 = 0A
Subnet Mask: 255.255.255.0
 255 = FF
 255 = FF
 255 = FF
 0 = 00
Gateway IP Address: 192.168.108.01
 192 = C0
 168 = A8
 108 = 6C
 1 = 01
UPD Port: 1024
TCP Port1: 1025
TCP Port2: 1026
The module currently supports only connecting to TCP Port 1025.
```

Mitsubishi FX 2N() Series Ladder

Ladder Logic for initializing the module. The configuration parameters are stored in data registers. Any thousand range with the D register area beginning with D0, D1000, D2000 and etc can be used.

```
|---| M8002 |---------------[ MOV  H81   D8120 ]  'Set communication register
|   |--------[ DMOV H45543130  D1000 ]   'Module ID ET10 (Fixed)
| |-------[ DMOV H0C0A86C0A  D1002 ]  'Module IP Address
|   |-------[ DMOV H0FFFFFF00  D1004 ]  'Subnet Mask
```

```
|   |-------[ DMOV H0C0A86C01  D1006 ]  'Gateway IP Address
|   |-------------[ MOV K1024  D1008 ]  'UDP Port Number
|   |-------------[ MOV K1025  D1009 ]  'TCP Port Number 1
|   |-------------[ MOV K1026  D1010 ]  'TCP Port Number 2
|   |----------------[ MOV K0  D1011 ]  'System Area Default Value 0
|   |----------------[ MOV K0  D1012 ]  'System Area Default Value 0
|  |----------------[ MOV K0   D1013 ]   'System Area Default Value 0
|   |----------------[ MOV K0  D1014 ]  'System Area Default Value 0
|   |----------------[ MOV K0  D1015 ]  'System Area Default Value 0
```

⬤ **Note:** This information has been taken from the *Mitsubishi Electric Ethernet Communications Module COM ET10-T* manual.

Using the FX2NC-ENET-ADP Ethernet Module

Mitsubishi has an Ethernet module that allows peer to peer Ethernet connection to FX1S, FX1N, FX2NC or FX2NC model controllers. The FX2NC-ENET_ADP module connects to the controller via the FX1N-CNV-BD or the FX2N-CNV-BD communications adapter. For equipment information, refer to the FX hardware manual.

Example Parameters
```
The Module ID is Fixed at ENET
 E = 45
 N = 4E
 E = 45
 T = 54
IP Address: 192.168.108.10
 192 = C0
 168 = A8
 108 = 6C
 10 = 0A
Subnet Mask: 255.255.255.0
 255 = FF
 255 = FF
 255 = FF
 0 = 00
Gateway IP Address: 192.168.108.01
 192 = C0
 168 = A8
 108 = 6C
 1 = 01
TCP Port: 1024
```

Mitsubishi FX1s Series Ladder

Ladder Logic for initializing the module on a FX1s. Set the Ethernet parameters in 9 data registers starting from D128 to D136.

```
|---| M8002 |-----------[ DMOV H454E4554  D128 ]  'Module ID ENET (Fixed)
|   |-------[ DMOV H0C0A86C0A  D130 ]   'Module IP Address
|   |-------[ DMOV H0FFFFFF00  D132 ]   'Subnet Mask
|   |-------[ DMOV H0C0A86C01  D134 ]   'Gateway IP Address
|   |-------------[ MOV K1024  D136 ]   'TCP Port Number
```

Mitsubishi FX1N/2N/2NC Series Ladder

Ladder Logic for initializing the module on a FX1s. Set the Ethernet parameters in 9 data registers starting from D1000 to D1008. If these registers are used for other purposes, set to 9 data registers starting at D2000, D3000, D4000, D5000, D6000 or D7000.

```
|---| M8002 |-----------[ DMOV H454E4554  D1000 ]  'Module ID ENET (Fixed)
|   |-------[ DMOV H0C0A86C0A  D1002 ]  'Module IP Address
|   |-------[ DMOV H0FFFFFF00  D1004 ]  'Subnet Mask
|   |-------[ DMOV H0C0A86C01  D1006 ]  'Gateway IP Address
```

```
|   |-------------[ MOV K1024   D1008 ]   'TCP Port Number
```

🔹 **Note:** This information has been taken from the *Mitsubishi FX2NC-ENET-ADP Ethernet Adapter Users* manual.

Using the FX3U-ENET Ethernet Module

To connect to the FX3U PLC via the FX3U-ENET Ethernet module, use Ethernet Encapsulation. To do so, the FX3U-ENET module must be configured using the Mitsubishi GX Developer-FX software. For more information, follow the instructions below.

1. To start, create a new GX Developer project for a FX3U model.

2. Then, select **Tools** | **FX special functions utility** | **FX Configurator-EN**.
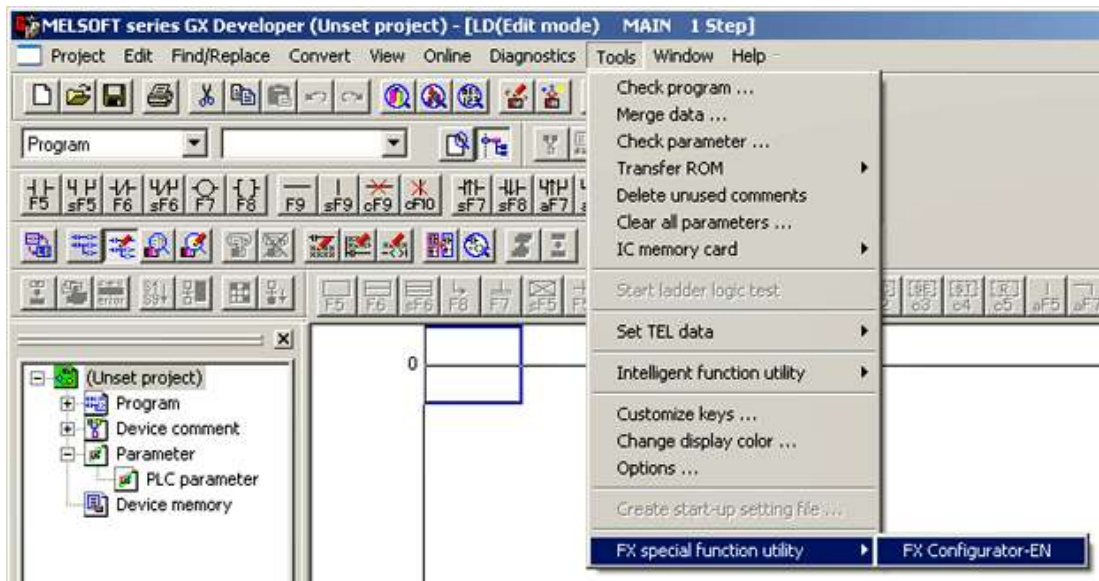
3. The FX Configurator-EN dialog should appear as shown below.
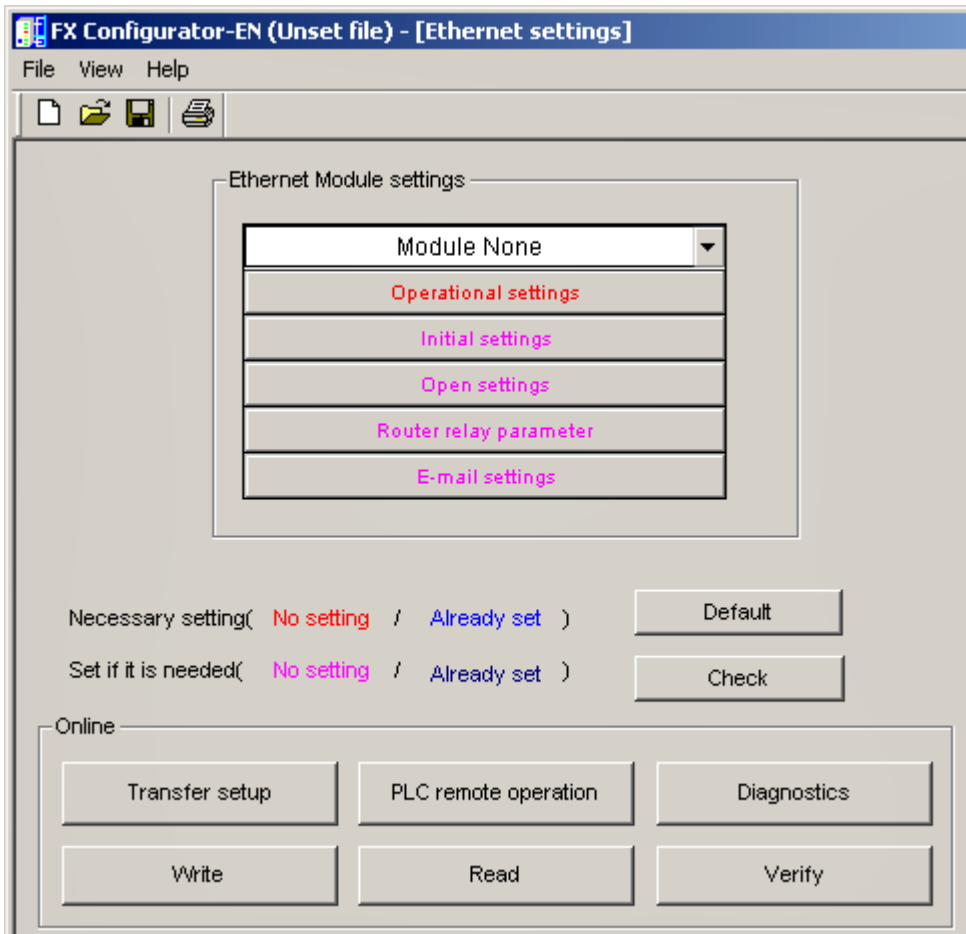
4. Next, fill in the minimum required configuration information for the FX3U-ENET block. Select a module from the first drop-down list, and then click **Operational Settings**. The settings should be similar to those shown in the image below.
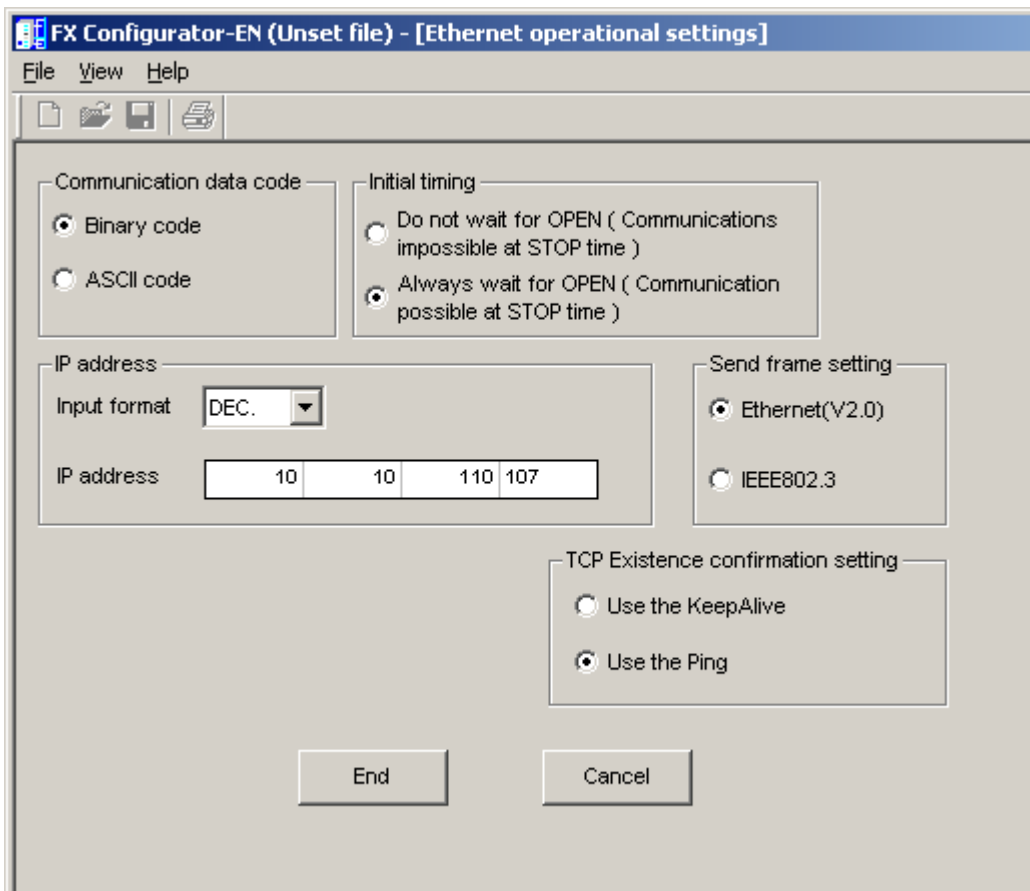
5. Click **End** to continue.

6. In the **FX Configurator-EN** dialog window, click **Open Settings**.
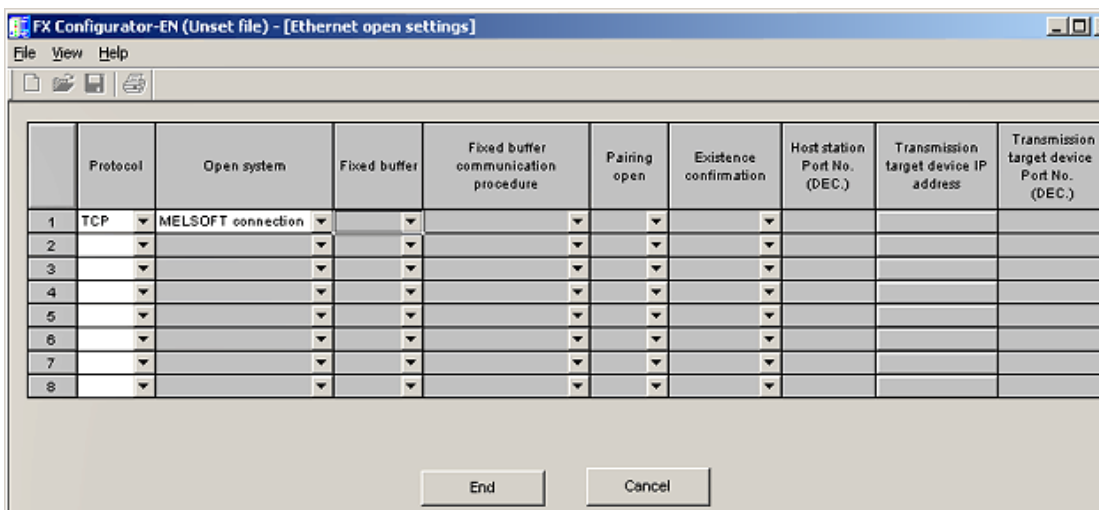
Writing Network Parameters to the PLC

After the network parameters have been specified, they must be written to the PLC. To do so, click **Write** from the **FX-Configurator-EN** window.
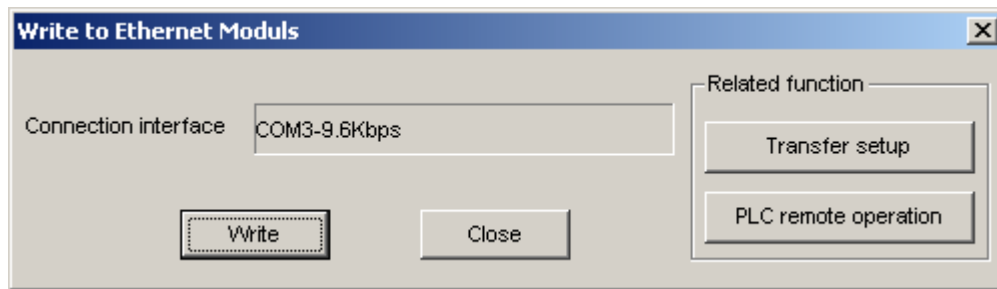
There must be a serial connection to the FX3U PLC because the configuration settings are written to the PLC via this serial link. Make sure that the communication parameters are correct by using the **Transfer setup** button above or by clicking **Online** | **Transfer setup** from the main menu.

After the parameters have been written, cycle the power on the PLC for the network parameter changes to take effect. Now, the FX3U PLC can be communicated with using Ethernet encapsulation via the FX3U-ENET Ethernet module. Use the following parameters:

- **IP Address:** 10.10.110.107 (or the IP address used above)
- **Port Number:** 5551
- **Protocol:** TCP/IP

Index

N

Name 10

Network 3

Network Adapter 7

Non-Normalized Float Handling 9

None 6

O

Operating Mode 11

Operation with no Communications 8

Operational Behavior 7

Optimization Method 8

Outputs 17-21

Overrun 23

Overview 3

P

Parity 6, 23

Physical Medium 6

Poll Delay 7

R

Raise 7

Read Processing 8

Received block length does not match expected length. | Received block length = <number>, Expected block length = <number>. 23

Redundancy 15

Registers 17

Replace with Zero 9

Report Communication Errors 7-8

Request Timeout 13

Reset 17-21

Respect Tag-Specified Scan Rate 12

RS-485 7