Yaskawa MP Series Ethernet Driver

© 2021 PTC Inc. All Rights Reserved.

Table of Contents

Yaskawa MP Series Ethernet Driver ... 1
Table of Contents ... 2
Yaskawa MP Series Ethernet Driver ... 4
Overview ... 4
Setup ... 5
Channel Properties — General ... 5
Channel Properties — Ethernet Communications ... 6
Channel Properties — Write Optimizations ... 6
Channel Properties — Advanced ... 7
Device Properties — General ... 8
Operating Mode ... 9
Device Properties — Scan Mode ... 9
Device Properties — Timing ... 10
Device Properties — Auto-Demotion ... 11
Device Properties — Communications Parameters ... 12
Device Properties — Block Sizes ... 12
Device Properties — Redundancy ... 13
Memory Mapping for MPxxxxiec Devices ... 14
Optimizing Communications ... 15
Data Types Description ... 16
Address Descriptions for MPxxxx (218IF Module) Devices ... 17
Address Descriptions for MPxxxxiec Devices ... 19
Error Descriptions ... 21
Address '<address>' is out of range for the specified device or register ... 22
Array size is out of range for address '<address>' ... 22
Array support is not available for the specified address: '<address>' ... 22
Data Type '<type>' is not valid for device address '<address>' ... 22
Device address '<address>' contains a syntax error ... 23
Device address '<address>' is not supported by model '<model name>' ... 23
Device address '<address>' is Read Only ... 23
Missing address ... 23
Device '<device name>' is not responding ... 23
Unable to write to '<address>' on device '<device name>' ... 24
Device '<device name>' block request [<start address> to <end address>] responded with exception '<exception response>' ... 24
Failure to start Winsock communications ... 25

Illegal data address for tag '<tag address>' on device '<device name>' 25
Illegal data address in block [<start address> to <end address>] on device '<device name>' 25
Illegal data value for tag '<tag address>' on device '<device name>' 25
Illegal data value in block [<start address> to <end address>] on device '<device name>' 26
Illegal function code '<function code (hex)>' in block [<start address> to <end address>] on device '<device name>' 26
Illegal function code '<hex function code>' for tag '<tag address>' on device '<device name>' 26
Modbus server device '<device name>' detected a memory parity error 27
Modbus server device '<device name>' has failed 27
Modbus server device '<device name>' is busy 27
Tag '<tag address>' on device '<device name>' responded with exception '<exception code>' 27
Unable to bind to adapter: '<adapter>'. Connect failed. 28
Unable to create a socket connection for Device '<device>' 28
Unexpected response frame received for block [<start address> to <end address>] on device '<device name>' 28
Unexpected response frame received for tag '<tag address>' on device '<device name>' 28
Winsock initialization failed (OS Error = <error code>) 29
Winsock shut down failed (OS Error = <error code>) 29
Winsock V1.1 or higher must be installed to use the Yaskawa MP Series Ethernet device driver 29
Appendix: Hardware Configuration for MPxxxx (218IF Module) 30
Hardware Configuration for MPxxxxiec 34
Hardware Configuration for MPxxxx (218IF Module) - Ladders 40
Index 46

Yaskawa MP Series Ethernet Driver

Help version 1.041

CONTENTS

Overview
What is the Yaskawa MP Series Ethernet Driver?

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

Optimizing Communications
How do I get the best performance from the Yaskawa MP Series Ethernet?

Data Types Description
What data types does the Yaskawa MP Series Ethernet Driver support?

Address Descriptions
How do I reference a data location in a Yaskawa MP Series Ethernet device?

Error Descriptions
What error messages does the Yaskawa MP Series Ethernet Driver produce?

Overview

The Yaskawa MP Series Ethernet Driver provides a reliable way to connect Yaskawa MP Series Ethernet devices to OPC client applications; including HMI, SCADA, Historian, MES, ERP and countless custom applications. It is intended for any MPxxxx Series controller that uses the 218IF module or any MPxxxxiec Series controller that communicates via Modbus TCP.

Setup

Supported Devices

MPxxxx (218IF Module). This includes any controller that uses the 218IF module
MPxxxxiec. This includes all devices from the MP2000iec Series
CPxxxx series (using the MPxxxx protocol)

Communication Protocol

MPxxxx (218IF Module) Protocol: Memobus over TCP
MPxxxxiec Protocol: Modbus TCP

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 2048 per channel.

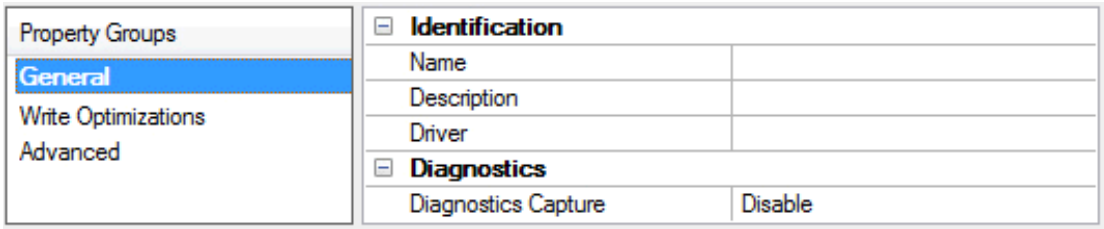
Device ID

Yaskawa MP Series Ethernet devices are networked using standard IP addressing. The Device ID has the following format: YYY.YYY.YYY.YYY, where YYY designates the device IP address. Each YYY byte must be in the range of 0 to 255.

See Also: *Hardware Configuration*

Channel Properties — General

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



Identification

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

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

Driver: Specify the protocol / driver for this channel. This property specifies the device driver that was selected during channel creation. It is a disabled setting in the channel properties. The property is required for creating a channel.

Note: With the server's online full-time operation, these properties can be changed at any time. This includes changing the channel name to prevent clients from registering data with the server. If a client has already acquired an item from the server before the channel name is changed, the items are unaffected. If, after the channel name has been changed, the client application releases the item and attempts to re-acquire using the old channel name, the item is not accepted. 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

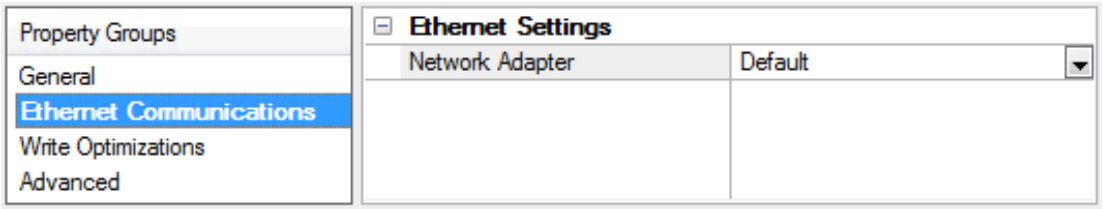
Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default is disabled.

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

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

Channel Properties — Ethernet Communications

Ethernet Communication can be used to communicate with devices.

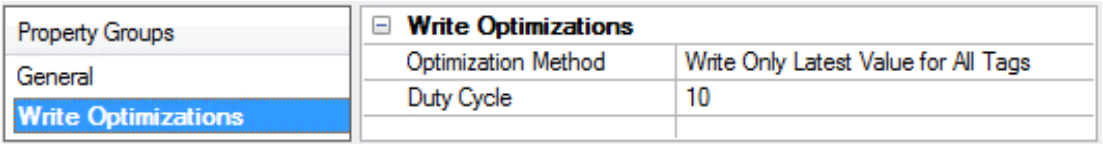


Ethernet Settings

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

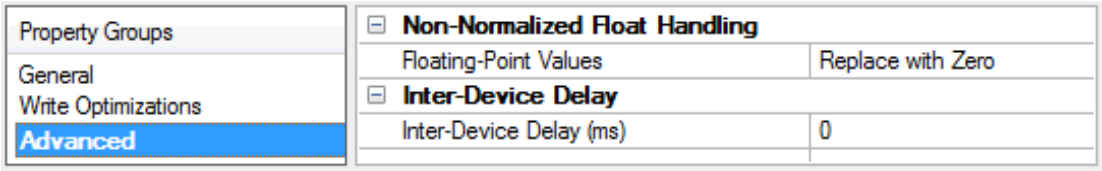
Optimization Method: Controls how write data is passed to the underlying communications driver. The options are:

- **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.



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

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

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

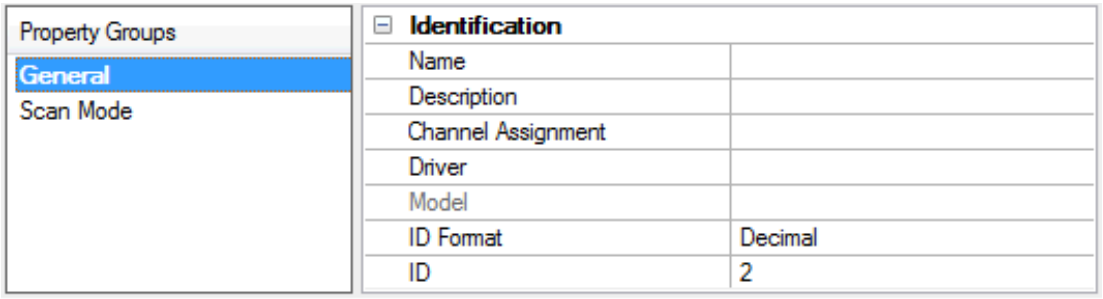
For more information on the floating-point values, refer to "How To ... Work with Non-Normalized Floating-Point Values" in the server help.

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

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

Device Properties — General

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



Identification

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

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

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

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

Driver: Selected protocol driver for this device.

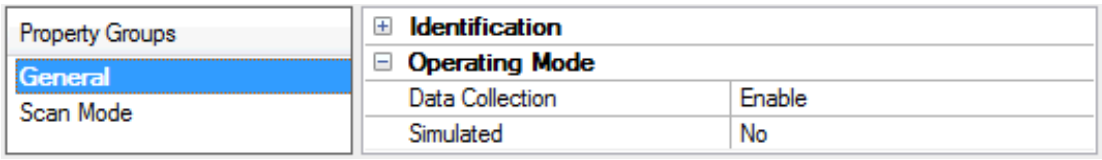
Model: Specify the type of device that is associated with this ID. The contents of the drop-down menu depend on the type of communications driver being used. Models that are not supported by a driver are disabled. If the communications driver supports multiple device models, the model selection can only 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 help documentation.

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

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

Operating Mode



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

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

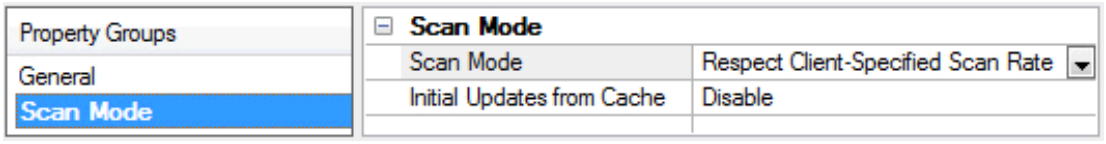
Notes:

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

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

Device Properties — Scan Mode

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



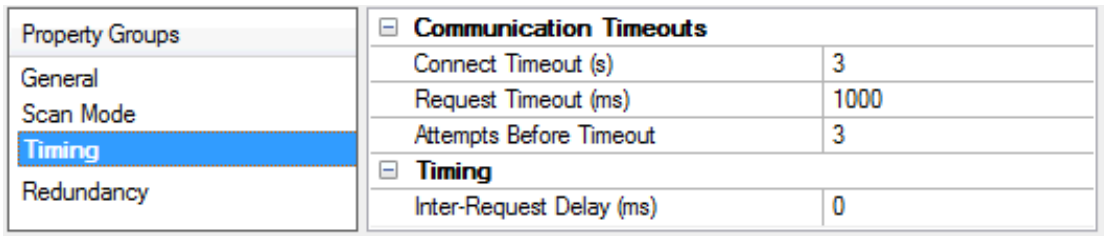
Scan Mode: Specify how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  Note: When the server has an active client and items for the device and the scan rate value is increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.
- **Request All Data at Scan Rate**: This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only**: This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the 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 supported by the driver, it is disabled.

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

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

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

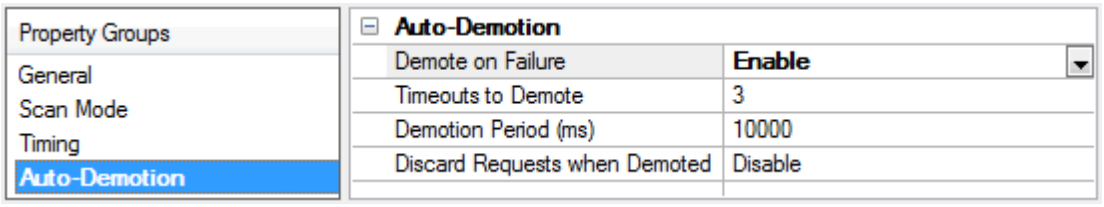
Timing

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

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

Device Properties — Auto-Demotion

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



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

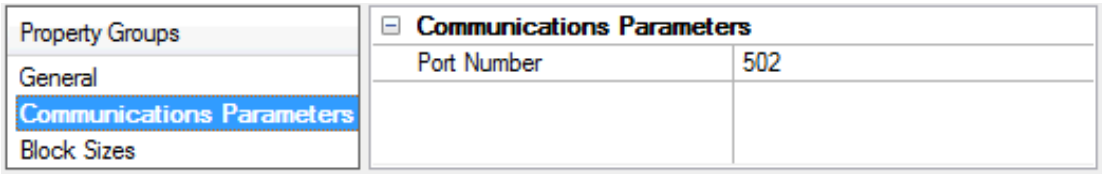
Tip: Determine when a device is off-scan by monitoring its demoted state using the _AutoDemoted system tag.

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

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

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

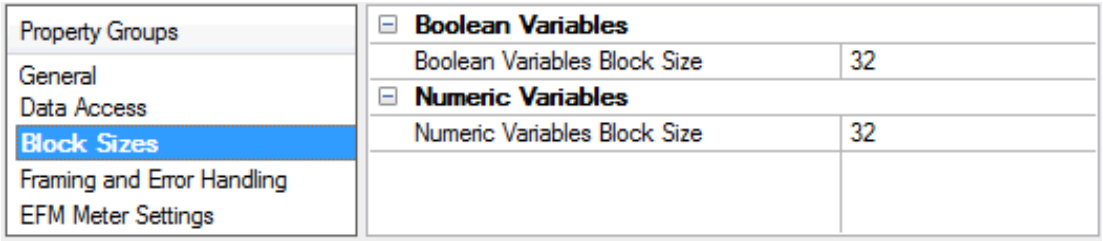
Device Properties — Communications Parameters



Port Number: This property specifies the TCP/IP port number that the remote device is configured to use. The default port number is 502.

Note: There should only be one device defined in the PLC per connection.

Device Properties — Block Sizes



Block Sizes for MPxxxx (218IF Module) Controllers

Bits

Input bits (IB) and output bits (MB) can be read from 8 to 800 points (bits) at a time. The default value is 32.

Registers

Input registers (IW, IL, IF) and output registers (MW, ML, MF) can be read from 1 to 120 locations (words) at a time. The default value is 32.

Block Sizes for MPxxxxiec Controllers

Bit Addresses

Input bits (IX) and Output Bits (QX) can be read 8 to 128 points (bits) at a time. The default value is 32.

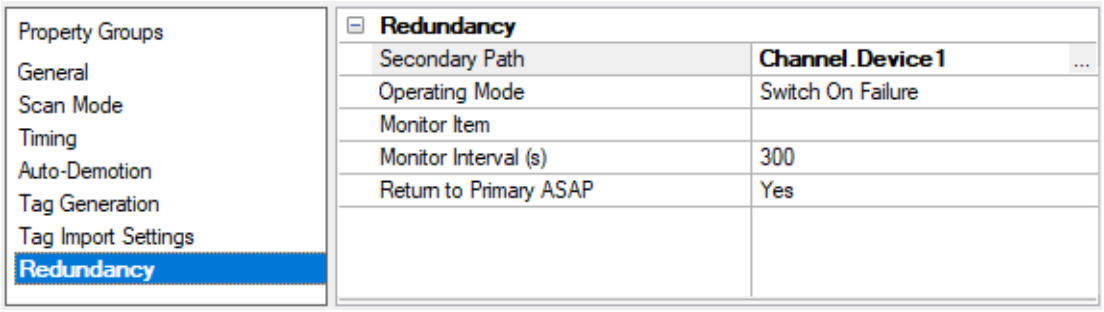
Register Addresses

Input registers (IW, ID, IL) and output registers (QW, QD, QL) can be read from 1 to 120 locations (words) at a time. The default value is 32.

Reasons for Changing the Default Block Sizes

1. Future versions of the device may not support block Read/Write operations of the default size.
2. The device may contain non-contiguous addresses (such as when using binary space module). If this is the case and the driver attempts to read a block of data that encompasses undefined memory, the device will most likely reject the request.

Device Properties — Redundancy



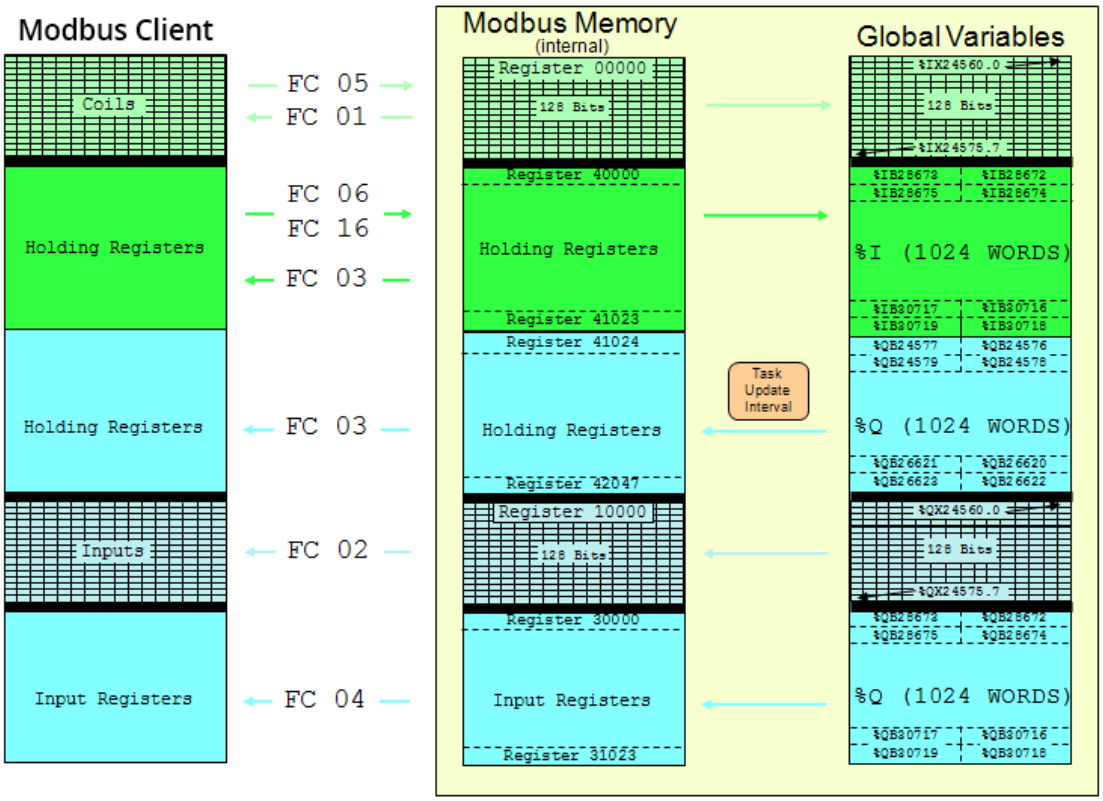
Redundancy is available with the Media-Level Redundancy Plug-In.

Consult the website, a sales representative, or the user manual for more information.

Memory Mapping for MPxxxxiec Devices

This driver (Modbus client) communicates via Modbus TCP to MPxxxxiec Series controllers that are configured as a Modbus server. The image below displays the Modbus memory map: it not only shows how it relates to the Global Variables (iec memory) in the controller, but also shows the Modbus function codes (FC) that are used to communicate between devices' application memory.

Note: The Yaskawa MP Series Ethernet Driver supports MPxxxxiec series devices, which includes the Yaskawa MP2000iec Series. For more information, refer to the MP2000iec Series at the manufacturer's website.

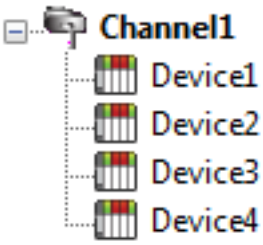


See Also: Address Descriptions for MPxxxxiec Devices

Optimizing Communications

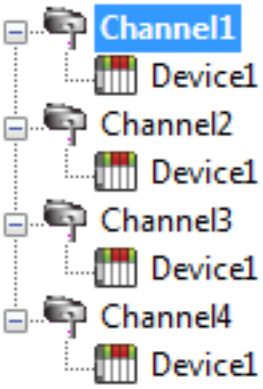
The Yaskawa MP Series Ethernet Driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the driver is fast, there are a couple of guidelines that can be used to control and optimize the application and gain maximum performance.

The server refers to communications protocols like Yaskawa MP Series Ethernet 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 Yaskawa Memobus Plus controller 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 Yaskawa MP Series Ethernet 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 Yaskawa MP Series Ethernet Driver could only define one single channel, then the example shown above would be the only option available; however, the 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 exactly how it is shown here.

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 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.

Block Size, which is available on each defined device, can also affect the Aromat Matsushita / NAIS Ethernet driver's performance. Block Size refers to the number of bytes that may be requested from a device at one time. To refine the performance of this driver, configure **Block Size** to 1 to 120 registers and 8 to 800 bits.

Data Types Description

Data Type	Description
Boolean	Single bit
Word	Unsigned 16-bit value bit 0 is the low bit bit 15 is the high bit
Short	Signed 16-bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
DWord	Unsigned 32-bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
BCD	Two byte packed BCD Value range is 0-9999. Behavior is undefined for values beyond this range.
LBCD	Four byte packed BCD Value range is 0-99999999. Behavior is undefined for values beyond this range.
Float	32-bit floating point value The driver interprets two consecutive registers as a floating point value by making the second register the high word and the first register the low word. When accessing data from a thermocouple or resistance sensor input module, the driver will use a single 16-bit register as a floating point value.
Float Example	If register 40001 is specified as a float, bit 0 of register 40001 would be bit 0 of the 32-bit word, and bit 15 of register 40002 would be bit 31 of the 32-bit word.
Double	64-bit floating point value
String	Null terminated ASCII string Support, includes HiLo and LoHi byte order selection and even string lengths from 2 to 240 bytes.

Address Descriptions for MPxxxx (218IF Module) Devices

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

Memory Type	Range	Data Type	Access
Input Bits	IB0000.b-IBFFFE.b (b is bit number 0x0-0xF)	**Boolean**	Read Only
Output Bits	MB00000.b-MB65534.b (b is bit number 0x0-0xF)	**Boolean**	Read/Write
Input Registers	IW0000-IWFFFE IW0000-IWFFFD IW0000.b-IWFFFE.b (b is bit number 0x0-0xF) IL0000-ILFFFD IF0000-IFFFFD	**Short**, Word, BCD **Long**, DWord, LBCD, Float **Boolean** **Long**, DWord, LBCD **Float**	Read Only
Output Registers	MW00000-MW65534 MW00000-MW65533 MW00000.b-MW65534.b (b is bit number 0x0-0xF) ML00000-ML65533 MF00000-MF65533	**Short**, Word, BCD **Long**, DWord, LBCD, Float **Boolean** **Long**, DWord, LBCD **Float**	Read/Write
Output Registers as strings with HiLo byte order	MSH00000.bbb-MSH65534.ddd (ddd is string length 2-240, even decimal)	String	Read/Write
Output Registers as strings with LoHi byte order	MSL00000.bbb-MSL65534.bbb (ddd is string length 2-240, even decimal)	String	Read/Write

Arrays

Arrays are supported for register addresses (IW, IL, IF, MW, ML, and MF). The syntax for declaring an array is *MMxxxxx[cols]* and *MMxxxxx[rows][cols]*, where "*MM*" is the memory type mnemonic and "*xxxxx*" is the base address of the array data.

The last register of the array cannot exceed the end of the address range. The formula for the final address in an array is *base address + (rows * columns * x) - x*.

Notes:

1. "*x*" is the total number of bytes in the data type. Word equals 2, DWord equals 4, and Double equals 8. Users should note the following:

- The last register of a Word, Short, and BCD array cannot exceed 65534.
- Float, DWord, Long, and Long BCD arrays cannot exceed 65533.
- Arrays do not allow the total number of registers being requested to exceed the register block size that was specified for the device.

2. Arrays are not supported for Boolean types (Ibxxxx.b, MBxxxxx.b, IWxxxx.b, and MWxxxxx.b).

String Support

This driver supports reading and writing output register memory as an ASCII string. When using output registers for string data, each register will contain two bytes of ASCII data. The order of the ASCII data within a given register can be selected when the string is defined. The length of the string can be from 2 to 240 bytes and is entered in place of a bit number. The length must be entered as an even decimal number.

Examples

1. To address a string starting at MW40200 with a length of 100 bytes and Hi-Lo byte order, enter: MSH40200.100
2. To address a string starting at MW40500 with a length of 78 bytes and Lo-Hi byte order, enter: MSL40500.78

Notes:

1. Input addresses (IB, IW, IL, IF) are in hex, while output addresses (MB, MW, ML, MF) are in decimal. Bit numbers, "b" are always in hex. Array "rows" and "cols" are always in decimal.
2. All output addresses map to the same memory area. For example, MB00001.F is the same as MW00001.F. ML00001 and MF00001 both map to the same memory as MW00001 and MW00002. The same is true for input addresses.
3. Writes to MB00000-MB04095 are faster than writes to MB04096-MB65534 because they can take advantage of direct bit access Memobus commands. Writes to the higher bits requires the driver to perform a read modify write operation, taking approximately twice as long.

Important: The actual range of valid addresses is hardware specific and may be smaller than the range allowed by this driver.

Address Descriptions for MPxxxxiec Devices

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

Memory Type	Range	Data Type	Access
Input Bits	%IX24560.b-%IX24575.b (b is bit number 0-7)	**Boolean**	Read/Write
Output Bits	%QX24560.b-%QX24575.b (b is bit number 0-7)	**Boolean**	Read Only
Input Registers	%IW28672-%IW30718 %IW28672.b-%IW30718.b (b is bit number 0-15) %ID28672-%ID30716 %IL28672-%IL30712	Short, **Word**, BCD **Boolean** Long, **DWord**, Float, LBCD **Double**	Read/Write
Output Registers	%QW24576-%QW26622 %QW28672-%QW30718 %QW24576.b-%QW26622.b %QW28672.b-%QW30718.b (b is bit number 0-15) %QD24576-%QD26620 %QD28672-%QD30716 %QL24576-%QL26616 %QL28672-%QL30712	Short, **Word**, BCD **Boolean** Long, **DWord**, Float, LBCD **Double**	Read Only

Note: Strings are not supported.

See Also: Memory Mapping for MPxxxxiec Devices

Arrays

Arrays are supported for register addresses IW, IL, ID, QW, QL, QD. The syntax for declaring an array is *%MMxxxxx[cols]* and *%MMxxxxx[rows][cols]*, where "*MM*" is the memory type mnemonic and "*xxxxx*" is the base address of the array data. When creating an array, the total number of registers requested by an array cannot exceed the register block size that was specified for this device.

The last register of the array cannot exceed the end of the address range. The final address in an array may be calculated through the following: *base address + (rows * columns * x) - x.*

Note: "*x*" is the total number of bytes in the data type. Word equals 2, DWord equals 4, and Double equals 8.

Input Register Arrays IW, IL, and ID Examples

1. Word Address %IW30712 [2][2] would be %IW30712 + ([2]*[2] * 2) - 2 = %IW30718.
2. DWord Address %ID30704 [2][2] would be %ID30704 + ([2]*[2]* 4) – 4 = %ID30716.

3. Double Address %IL30688 [2][2] would be %IL30688 + ([2]* [2] * 8) – 8 = %IL30712.

Output Register Arrays QW, QL, and QD Examples
1. Word Address %QW26616 [2][2] would be %QW26616 + ([2]* [2] * 2) - 2 = %QW26622.
2. DWord Address %QD26608 [2][2] would be %QD26608 + ([2]* [2]* 4) – 4 = %QD26620.
3. Double Address %QL26592 [2][2] would be %QL26592 + ([2]* [2] * 8) – 8 = %QL26616.

Note: Arrays are not supported for Boolean types (%IXxxxx.b, %QXxxxxx.b, %IWxxxx.b, %QWxxxxx.b).

Location and Size Prefixes

The following iec memory types' ranges utilize both location and size prefixes. Description of the location prefixes are as follows:

- **I:** The physical input.
- **Q:** The physical output.

Description of the size prefixes are as follows:

- **X:** Single bit.
- **W:** Word (16 bits).
- **D:** Double Word (32 bits).
- **L:** Long Word (64 bits).

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
Array size is out of range for address '<address>'
Array support is not available for the specified address: '<address>'
Data Type '<type>' is not valid for device address '<address>'
Device address '<address>' contains a syntax error
Device address '<address>' is not supported by model '<model name>'
Device address '<address>' is Read Only
Missing address

Device Status Messages

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

Device Specific Messages

Device '<device name>' block request [<start address> to <end address>] responded with exception '<exception response>'
Failure to start Winsock communications
Illegal data address for tag '<tag address>' on device '<device name>'
Illegal data address in block [<start address> to <end address>] on device '<device name>'
Illegal data value for tag '<tag address>' on device '<device name>'
Illegal data value in block [<start address> to <end address>] on device '<device name>'
Illegal function code '<function code (hex)>' in block [<start address> to <end address>] on device '<device name>'
Illegal function code '<hex function code>' for tag '<tag address>' on device '<device name>'
Modbus server device '<device name>' detected a memory parity error
Modbus server device '<device name>' has failed
Modbus server device '<device name>' is busy
Tag '<tag address>' on device '<device name>' responded with exception '<exception code>'
Unable to bind to adapter: '<adapter>'. Connect failed
Unable to create a socket connection for Device '<device>'
Unexpected response frame received for block [<start address> to <end address>] on device '<device name>'
Unexpected response frame received for tag '<tag address>' on device '<device name>'
Winsock initialization failed (OS Error = <error code>)
Winsock shut down failed (OS Error = <error code>)
Winsock V1.1 or higher must be installed to use the Yaskawa MP Series 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.

Array size is out of range for address '<address>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically is requesting an array size that is too large for the address type or block size of the driver.

Solution:
Re-enter the address in the client application to specify a smaller value for the array or a different starting point.

Array support is not available for the specified address: '<address>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically contains an array reference for an address type that doesn't support arrays.

Solution:
Re-enter the address in the client application to remove the array reference or correct the address type.

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:
An invalid tag address has been specified in a dynamic request.

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

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

Error Type:
Warning

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

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

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.

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 communication parameters for the connection are incorrect.
3. The named device may have been assigned an incorrect Network ID.
4. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device setting.

Solution:

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

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

Error Type:
Serious

Possible Cause:

1. The named device may not be connected to the network.
2. The named device may have been assigned an incorrect Network ID.
3. The named device is not responding to write requests.
4. The address does not exist in the PLC.

Solution:

1. Check the PLC network connections.
2. Verify that the Network ID given to the named device matches that of the actual device.

Device '<device name>' block request [<start address> to <end address>] responded with exception '<exception response>'

Error Type:
Warning

Possible Cause:
The device returned a Modbus exception code.

Solution:
Determine the meaning of the exception code, and then fix accordingly.

Failure to start Winsock communications

Error Type:
Fatal

Possible Cause:
Could not negotiate with the operating systems Winsock 1.1 functionality.

Solution:
Verify that the winsock.dll is properly installed on the system.

Illegal data address for tag '<tag address>' on device '<device name>'

Error Type:
Warning

Possible Cause:
The data address received in this query is not allowed for the Modbus server device because the reference number and transfer length combination is invalid.

Solution:
Ensure that the range of memory exists in the PLC.

Note:
For a controller with 100 registers, a request with offset 96 and length 4 would succeed. A request with an offset 96 and length 5, however, will generate exception 02.

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

Error Type:
Warning

Possible Cause:
The data address received in this query is not allowed for the Modbus server device because the reference number and transfer length combination is invalid.

Solution:
Ensure that the range of memory exists in the PLC.

Note:
For a controller with 100 registers, a request with offset 96 and length 4 would succeed. A request with an offset 96 and length 5, however, will generate exception 02.

Illegal data value for tag '<tag address>' on device '<device name>'

Error Type:
Warning

Possible Cause:

A value contained in the query data field is not an allowed value for Modbus server device. This indicates an error in the structure of the remainder of a complex request, such as that the implied length is incorrect.

Solution:
Correct the error in the structure , and then retry the complex request.

Note:
This error message does not mean that a data item submitted for storage in a register has a value outside the expectation of the application program. The Modbus TCP Protocol is not aware of the significance of any particular value of any particular register.

Illegal data value in block [<start address> to <end address>] on device '<device name>'

Error Type:
Warning

Possible Cause:
A value contained in the query data field is not an allowed value for Modbus server device. This indicates an error in the structure of the remainder of a complex request, such as that the implied length is incorrect.

Solution:
Correct the error in the structure , and then retry the complex request.

Note:
This error message does not mean that a data item submitted for storage in a register has a value outside the expectation of the application program. The Modbus TCP Protocol is not aware of the significance of any particular value of any particular register.

Illegal function code '<function code (hex)>' in block [<start address> to <end address>] on device '<device name>'

Error Type:
Fatal

Possible Cause:
The function code received in the query is not allowed for the Modbus server device. This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It may also indicate that the device is in the wrong state to process this type of request (such as if it is not configured but is being asked to return register values).

Solution:
Correct the function code, and then retry the query.

Illegal function code '<hex function code>' for tag '<tag address>' on device '<device name>'

Error Type:
Fatal

Possible Cause:
The function code received in the query is not allowed for the Modbus server device. This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It may also indicate that the device is in the wrong state to process this type of request (such as if it is not configured but is being asked to return register values).

Solution:
Correct the function code, and then retry the query.

Modbus server device '<device name>' detected a memory parity error

Error Type:
Warning

Possible Cause:
The Modbus server device attempted to read extended memory and detected a parity error.

Solution:
Retry the request, but service may be required on the Modbus server device if the error continues to occur.

Modbus server device '<device name>' has failed

Error Type:
Fatal

Possible Cause:
An unrecoverable error occurred while the Modbus server device was attempting to perform the requested action.

Solution:
Locate the cause of the error, and then re-attempt the action.

Modbus server device '<device name>' is busy

Error Type:
Warning

Possible Cause:
The Modbus server device is processing a long duration program command.

Solution:
Retry the request when the Modbus server device is free.

Tag '<tag address>' on device '<device name>' responded with exception '<exception code>'

Error Type:
Warning

Possible Cause:

The device returned a Modbus exception code.

Solution:
Determine the meaning of the exception code, and then fix accordingly.

Unable to bind to adapter: '<adapter>'. Connect failed.

Error Type:
Fatal

Possible Cause:
The device could not bind local IP address of specified adapter.

Solution:

1. Verify that the winsock.dll is properly installed on the system.
2. Verify that TCP/IP and Ethernet adapter is properly configured on the system.

Unable to create a socket connection for Device '<device>'

Error Type:
Fatal

Possible Cause:
The device could not create a socket for TCP/IP Ethernet communication.

Solution:
Verify that the winsock.dll is properly installed on the system.

Unexpected response frame received for block [<start address> to <end address>] on device '<device name>'

Error Type:
Serious

Possible Cause:

1. The data is corrupted.
2. An unexpected frame was received.

Solution:
Re-attempt the query.

Unexpected response frame received for tag '<tag address>' on device '<device name>'

Error Type:
Serious

Possible Cause:

1. The data is corrupted.
2. An unexpected frame was received.

Solution:

Re-attempt the query.

Winsock initialization failed (OS Error = <error code>)

Error Type:

Fatal

Possible Cause:

Could not negotiate with the operating systems winsock 1.1 functionality.

Solution:

Verify that the winsock.dll is properly installed on the system.

Winsock shut down failed (OS Error = <error code>)

Error Type:

Serious

Possible Cause:

The device could not negotiate with the operating systems winsock 1.1 functionality.

Solution:

Verify that the winsock.dll is properly installed on the system.

Winsock V1.1 or higher must be installed to use the Yaskawa MP Series Ethernet device driver

Error Type:

Fatal

Possible Cause:

The device could not negotiate with the operating systems winsock 1.1 functionality.

Solution:

Verify that the winsock.dll is properly installed on the system.

Appendix: Hardware Configuration for MPxxxx (218IF Module)

The 218IF module must be configured before Ethernet connections to it may be established. Note the following:

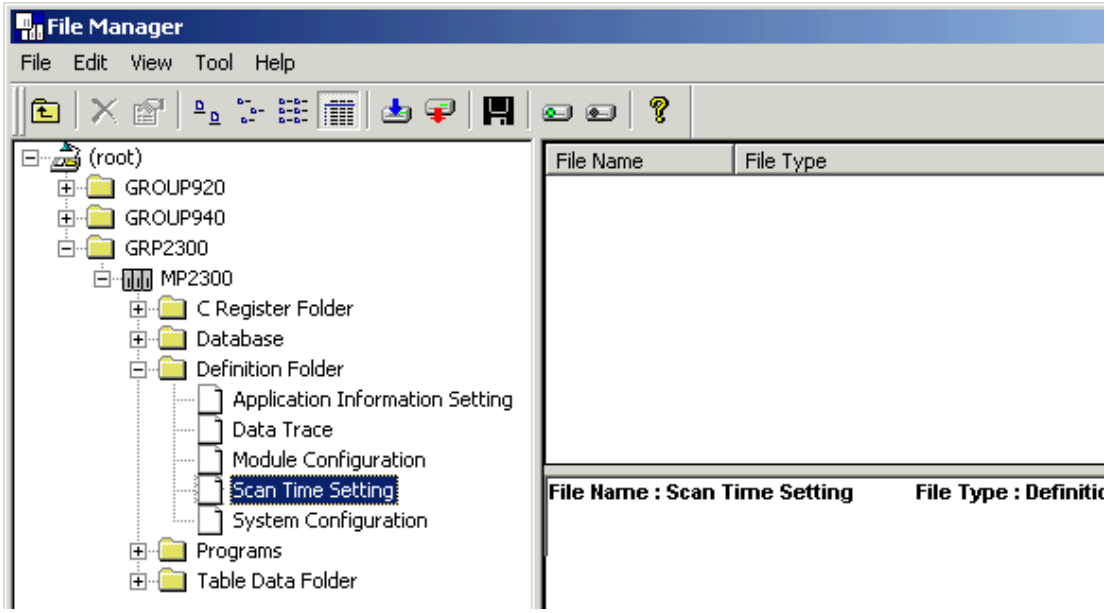
- The Yaskawa MP Series Ethernet Driver will support any controller that uses the 218IF module.
- Each connection point must be configured individually.
- Up to 20 connections may be configured, though only 10 may be used at a time.
- In the OPC server project, each device requires a corresponding connection configuration in a 218IF module. Connections for remote stations and other software applications must be configured in addition to those required by the OPC server.

Configuring a 218IF Module

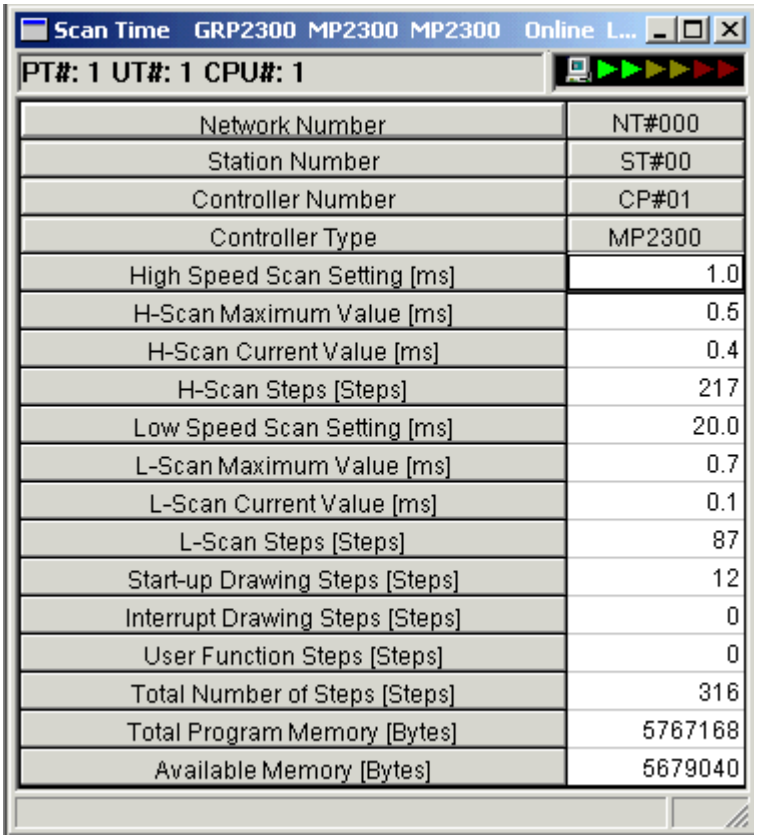
Follow the instructions below for information on configuring a 218IF Module.

Note: The examples shown are from MotionWorks (MPE720) v6.02.

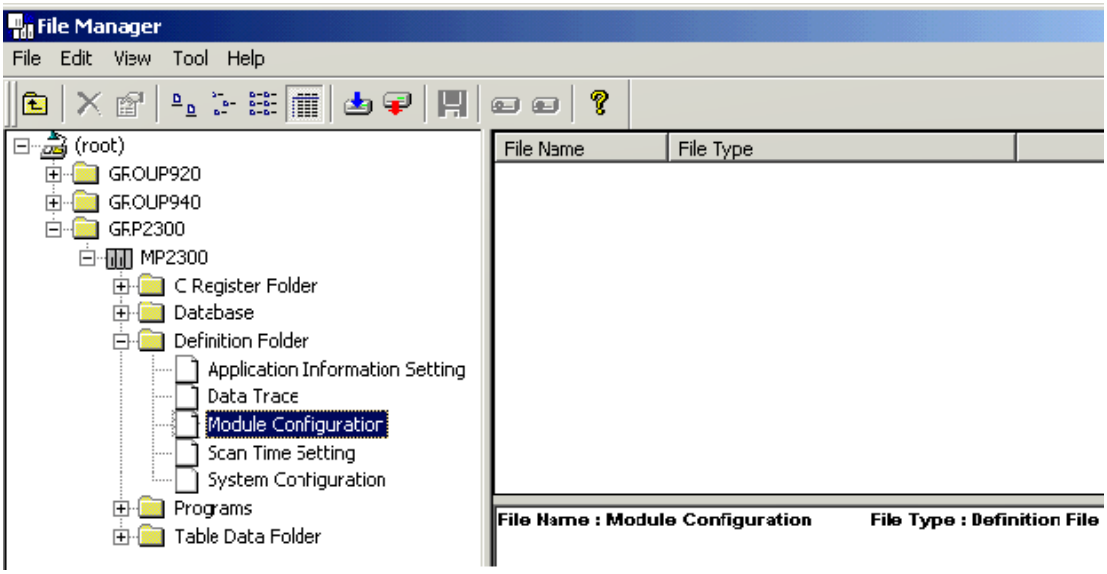
1. Open the **File Manager**.
2. Create a project for the controller.
3. Double-click on **Scan Time Setting**.



4. In **Scan Time**, note the **High Speed Scan Setting** and **Low Speed Scan Setting** fields. Yaskawa recommends 1ms for the High Scan setting and 20ms for the Low Scan setting.

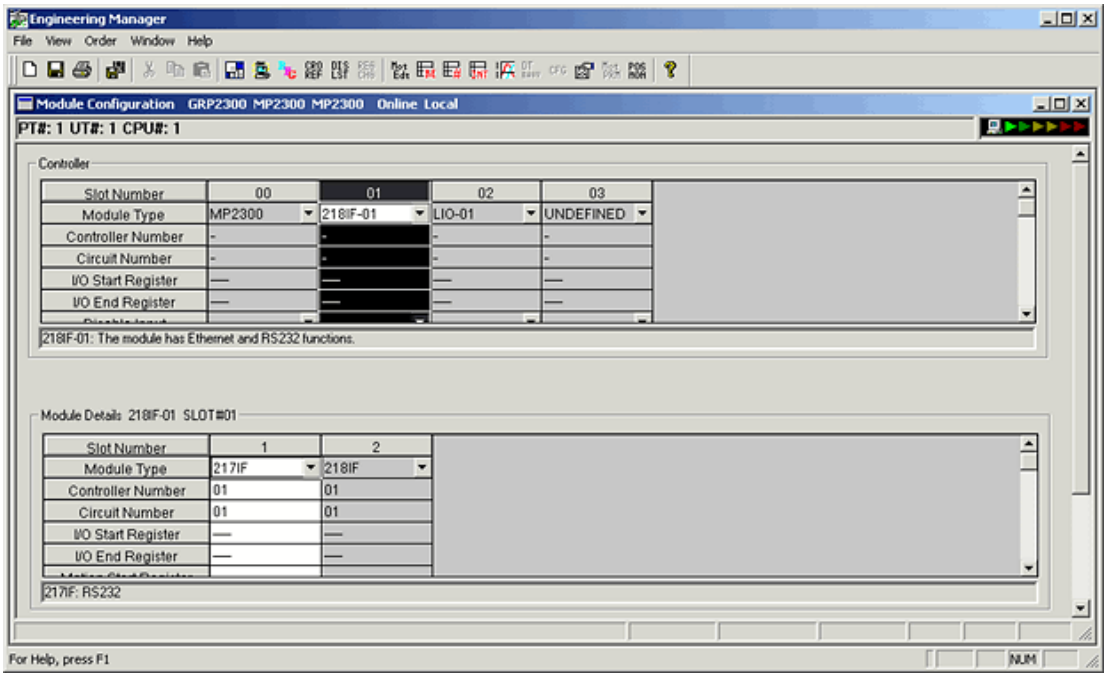


5. When the scan time settings are set, close the Scan Time dialog.
6. Double-click on **Module Configuration** to open the **Engineering Manager**.



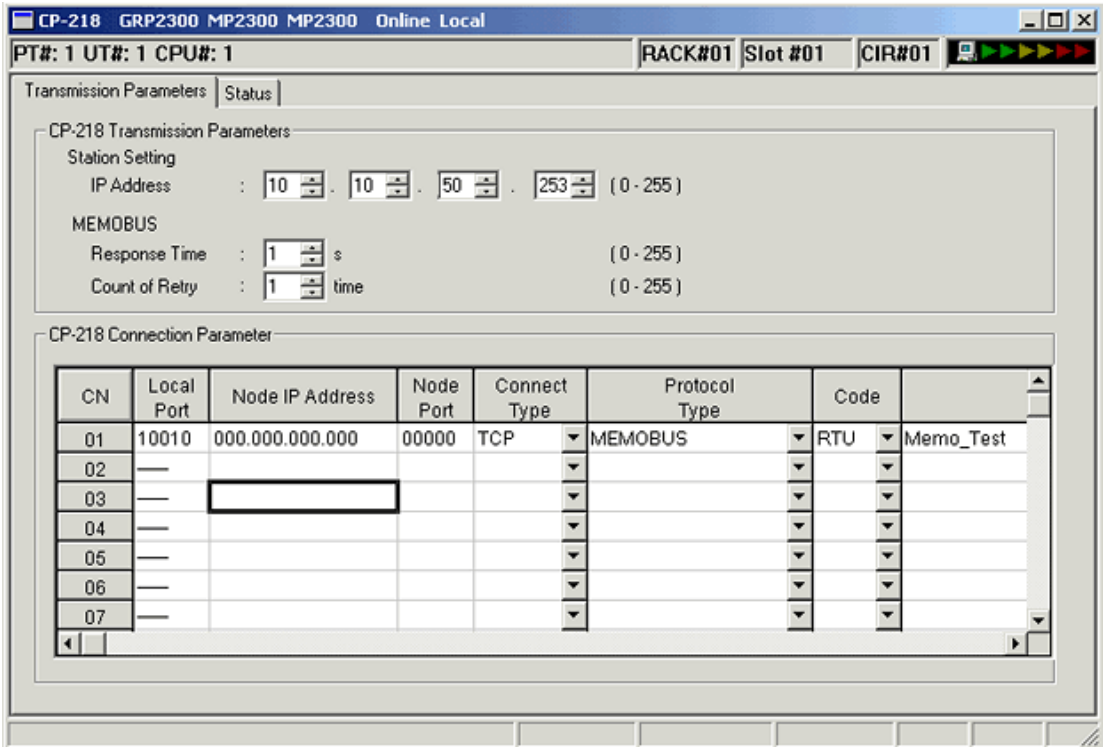
7. In the Engineering Manager, select the **218IF module**.

8. In **Slot** details, double-click 218IF to open the **Ethernet Interface** dialog.

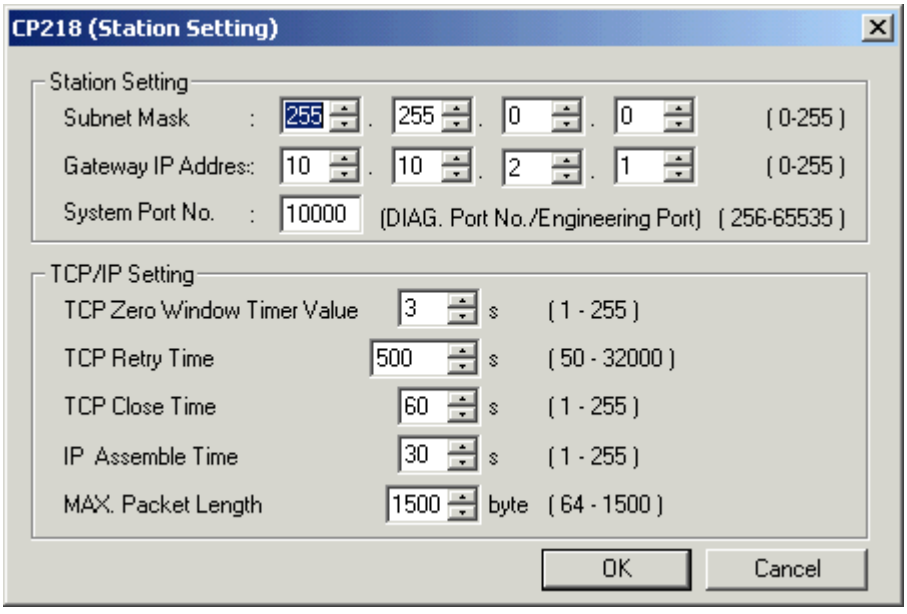


9. Click on the **Transmission Parameters** property group.

   - By default, the IP address will be set to 192.168.1.200; Response Time and Count of Retry will be 0. Set the IP address to one that is supported by the network.
   - Set Response Time to 1, and Count of Retry to 1.
   - No connections will have been created in a new project. Select Connection 1 and enter a port number in the Local Port field. The OPC server defaults to Port 502. If a different port number has been set, make sure the device in the OPC server project also has the same port number.
   - Since this is a Modbus server connection port, the Node IP Address and Node Port should both be set to 0.
   - Connect Type: TCP; Protocol Type: MEMOBUS; and Code: RTU.



10. If there is the possibility of a device connection from another LAN, set the **Network Subnet Mask** and **Gateway IOP Address**. To do so, select **Edit | Local Port: TCP/IP Setting** from the **File Manager** main menu. In the **Station Setting** dialog, set the Subnet Mask and Gateway IP values. Then, click **OK** to set and close.



11. Load the new configuration into the controller's **Flash Memory**.

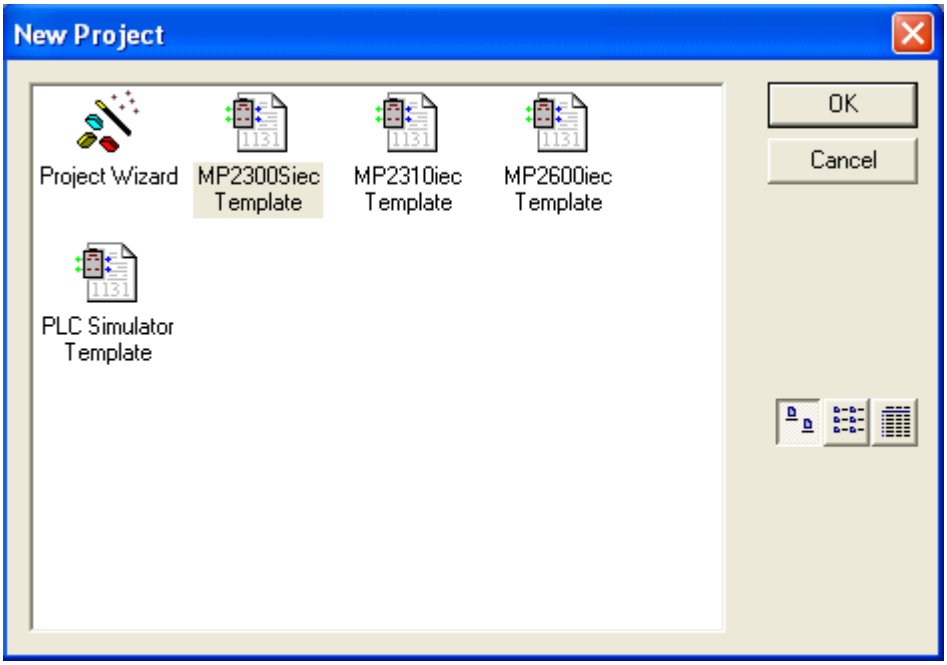
12. Next, create the ladder programs as described in Hardware Configuration - Ladders.

Important: The ladders are absolutely necessary; otherwise, the driver will not be able to connect to the 218IF module or exchange data with it.

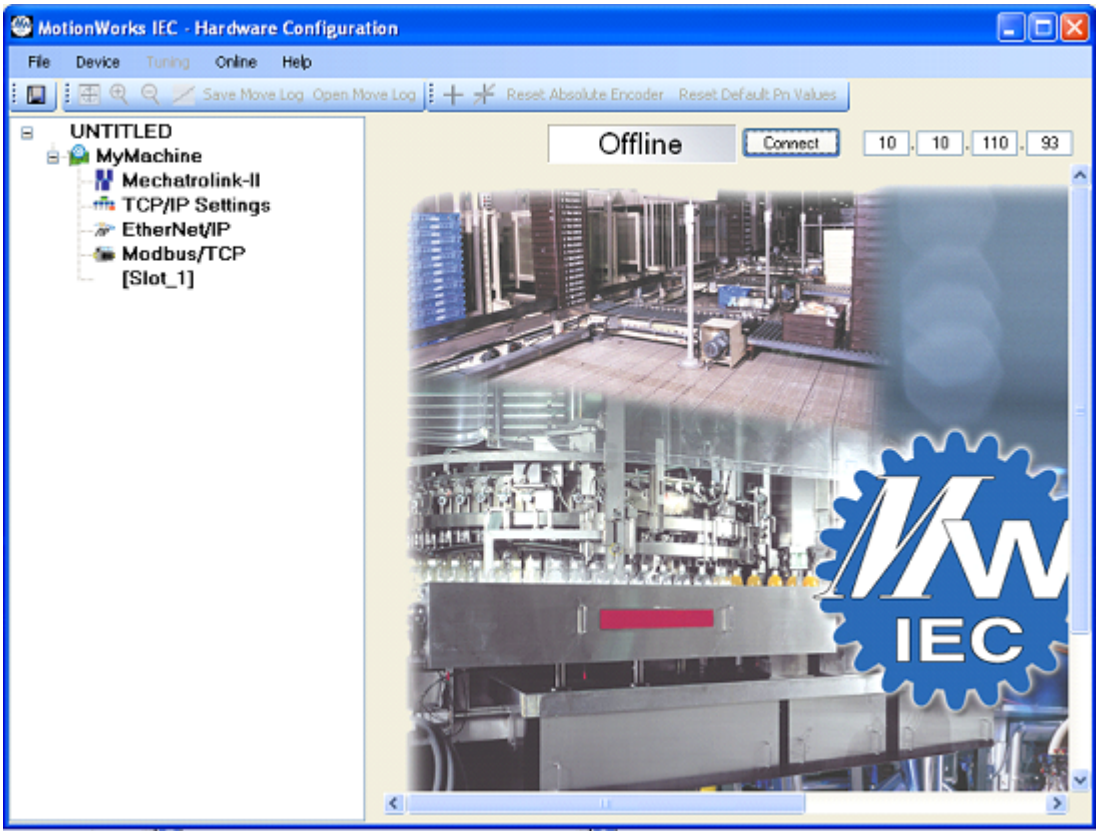
Hardware Configuration for MPxxxxiec

For information on configuring an MPxxxiec device, refer to the instructions below.

1. To start, install the programming software. In the following examples, MotionWorks IEC Pro 1.2.3.12 is used.
2. Next, create a project. In the following examples, an MP2300Siec device is used.

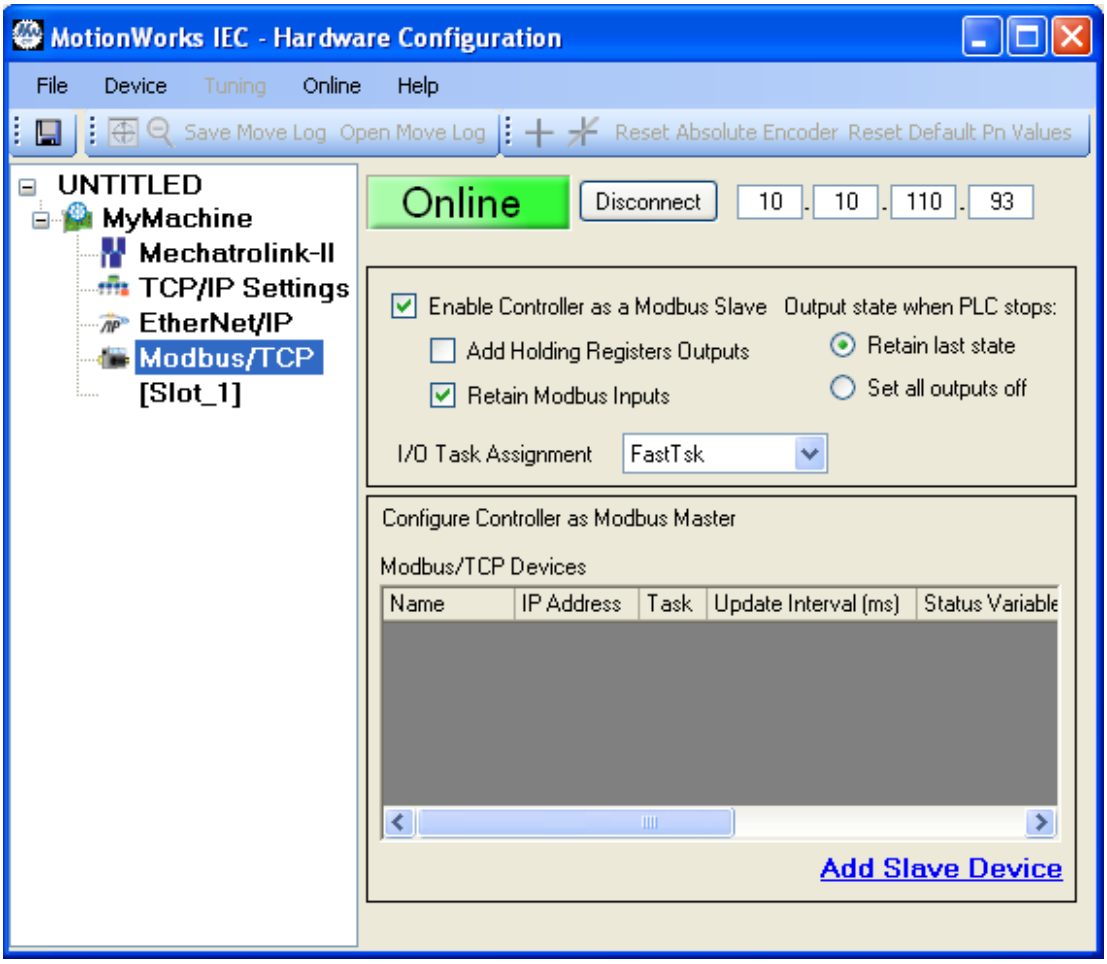


3. Launch the Hardware Configuration by clicking the Hardware Configuration icon . The Hardware Configuration window should appear as shown below.



4. In the upper-right corner, enter the device's IP Address. Then, click **Connect**.

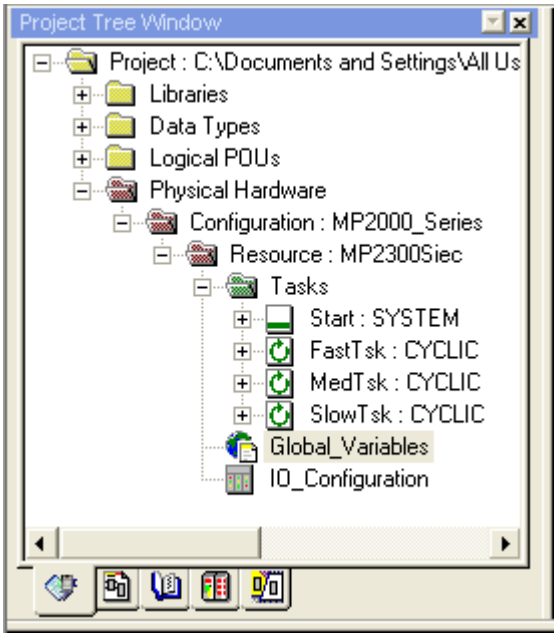
5. In the **Project Window Tree**, beneath the name of the device, double-click on **Modbus/TCP**. In this example, the device name is "MyMachine".



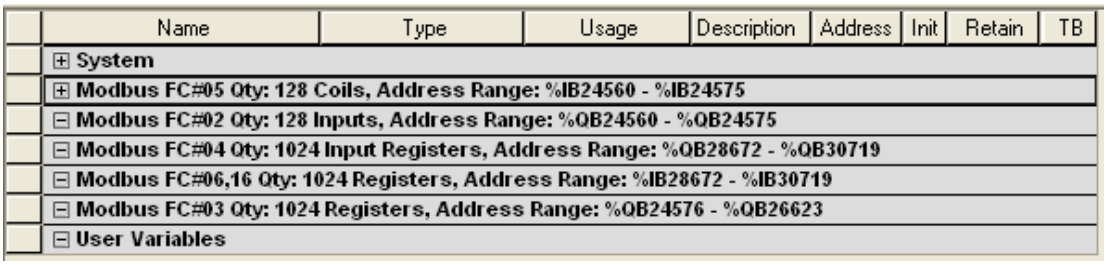
6. Then, do the following:

    - Check **Enable Controller as a Modbus Slave**.
    - In **I/O Task Assignment**, select **FastTsk**.
    - Leave the remaining parameters at their default setting.

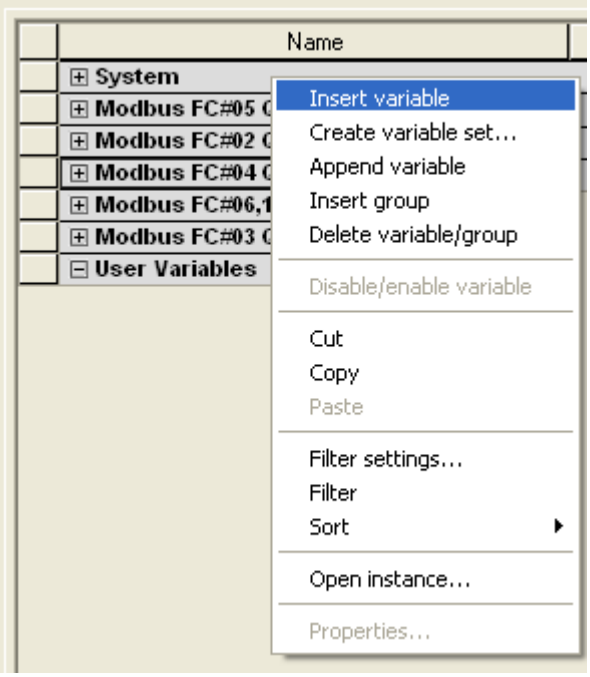
7. Once finished, save the changes by clicking **File** | **Save**.
8. Next, return to the MotionWorks IEC Pro 1.2.3.12 main project window. Locate the **Project Tree Window** and then browse to **Physical Hardware**.
9. Expand the project tree by clicking **Configuration: MP2000_Series** | **Resource:MP2300Siec**. Then, select **Global_Variables**.



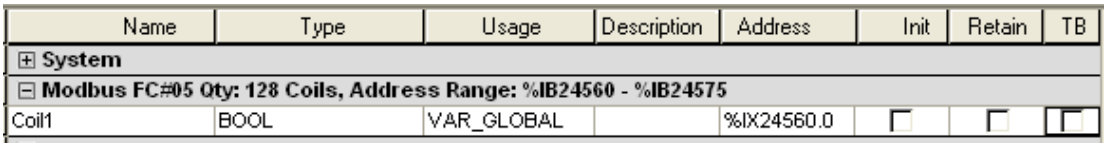
10. The following categories should be displayed:



11. To create a new variable, right-click on the desired group and then select **Insert Variable**.



12. In the **Address** column, specify the MotionWorks IEC Pro 1.2.3.12 address.



Name	Type	Usage	Description	Address	Init	Retain	TB
System							
Modbus FC#05 Qty: 128 Coils, Address Range: %IB24560 - %IB24575							
Coil1	BOOL	VAR_GLOBAL		%IX24560.0	☐	☐	☐

13. Once all changes are complete, build the project by clicking **Build | Make**.

14. Next, launch the **Project Control** dialog by clicking the **Project Control** icon . To download the project, click **Download| Download**. The following dialog will be invoked:

Note: The **Error** button will be enabled if the project contains any errors. When clicked, it will display a description of the error.

15. Next, locate **State** and verify that it is set to **Run**. Then, select **Cold**, **Warm**, or **Hot**. Descriptions of the starts are as follows:

    - **Cold:** In this start, all data will be initialized.
    - **Warm:** In this start, only non-retentive data will be initialized.
    - **Hot:** In this start, no data will be initialized.

16. When finished, click **File** | **Save**.

Viewing Data

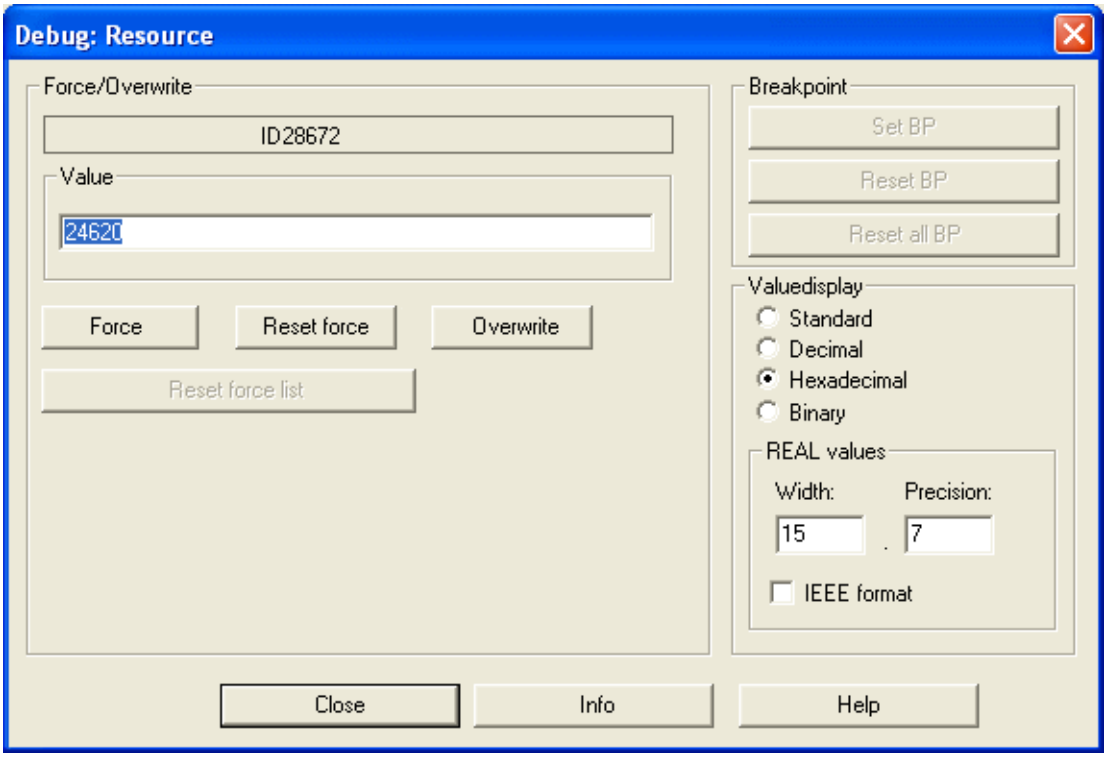
Users must be in debug mode to view data. For more information, refer to the instructions below.

1. First, click the **Debug On/Off** icon .
2. Then, return to the **Global_Variables** window.

Note: Data should now be visible.

Editing Read/Write Data

1. First, right-click on the variable that will be edited. Then, select **Debug Dialog**.
2. In **Value**, select or type the new value. Then, click **Overwrite**.



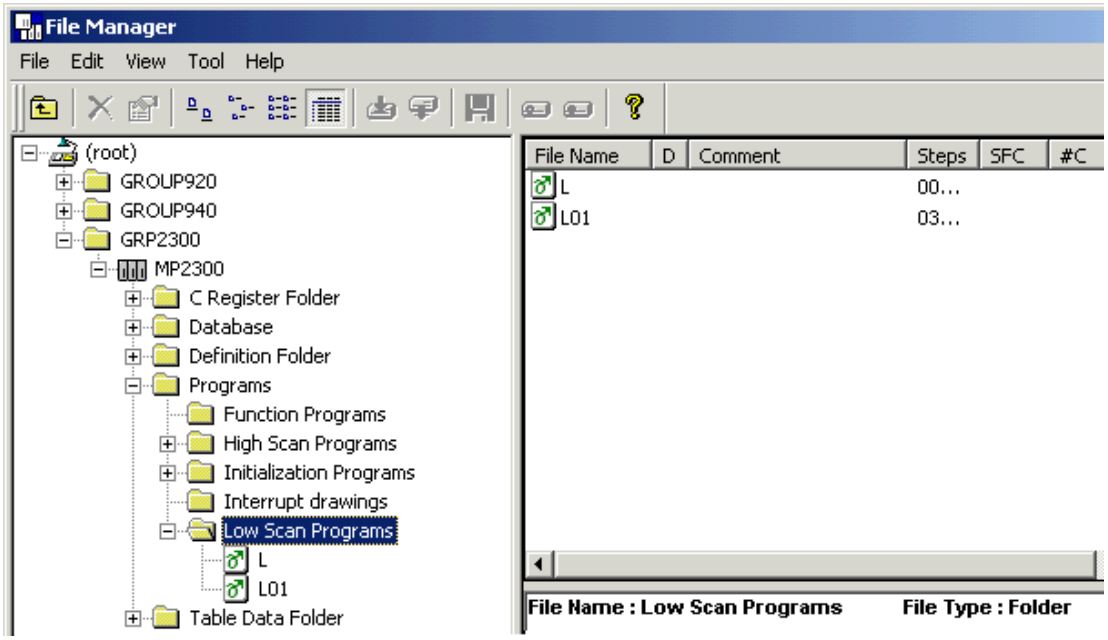
Hardware Configuration for MPxxxx (218IF Module) - Ladders

After the 218IF module has been configured, two ladders need to be created to handle communications. **See Also:** Hardware Configuration.

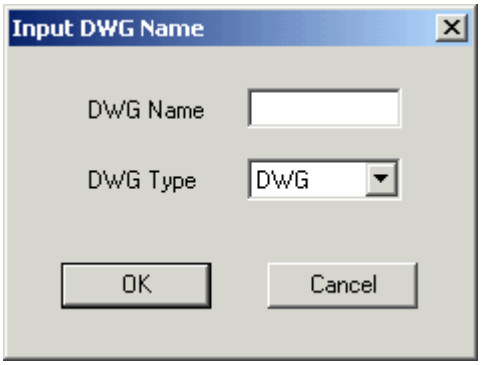
Important: The ladders are absolutely necessary: otherwise, the driver will not be able to connect to the 218IF module or exchange data.

Adding Ladders

To add a new ladder drawing, select **Low Scan Programs** in the **Controller project.** Then, select **File| New Drawing** from the main menu.

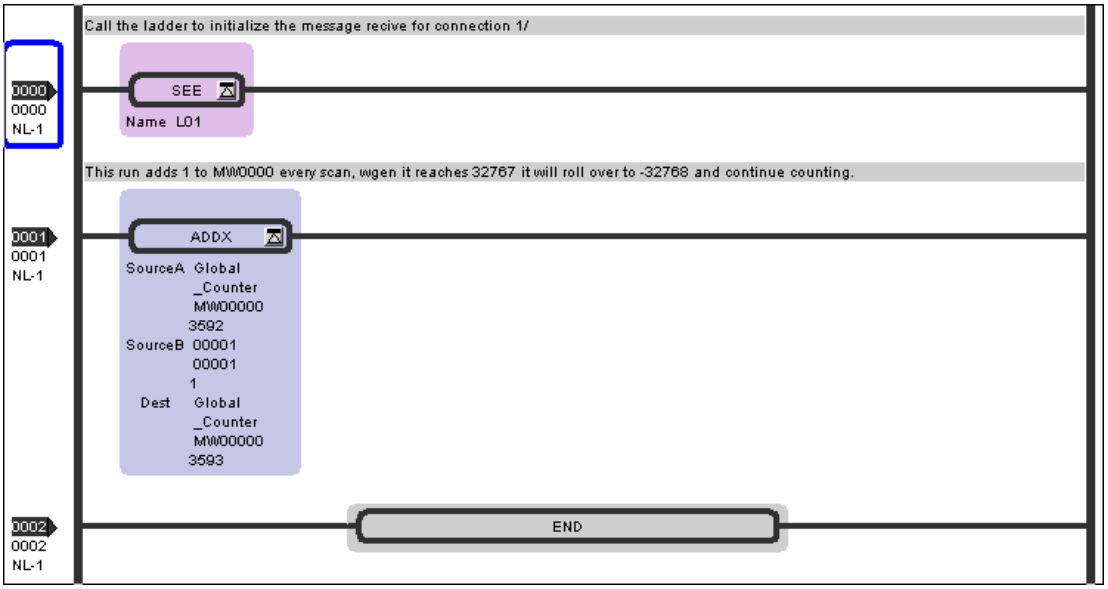


Note: The image below shows how the Input DWG Name dialog is displayed.



Drawing L in Low Scan Programs

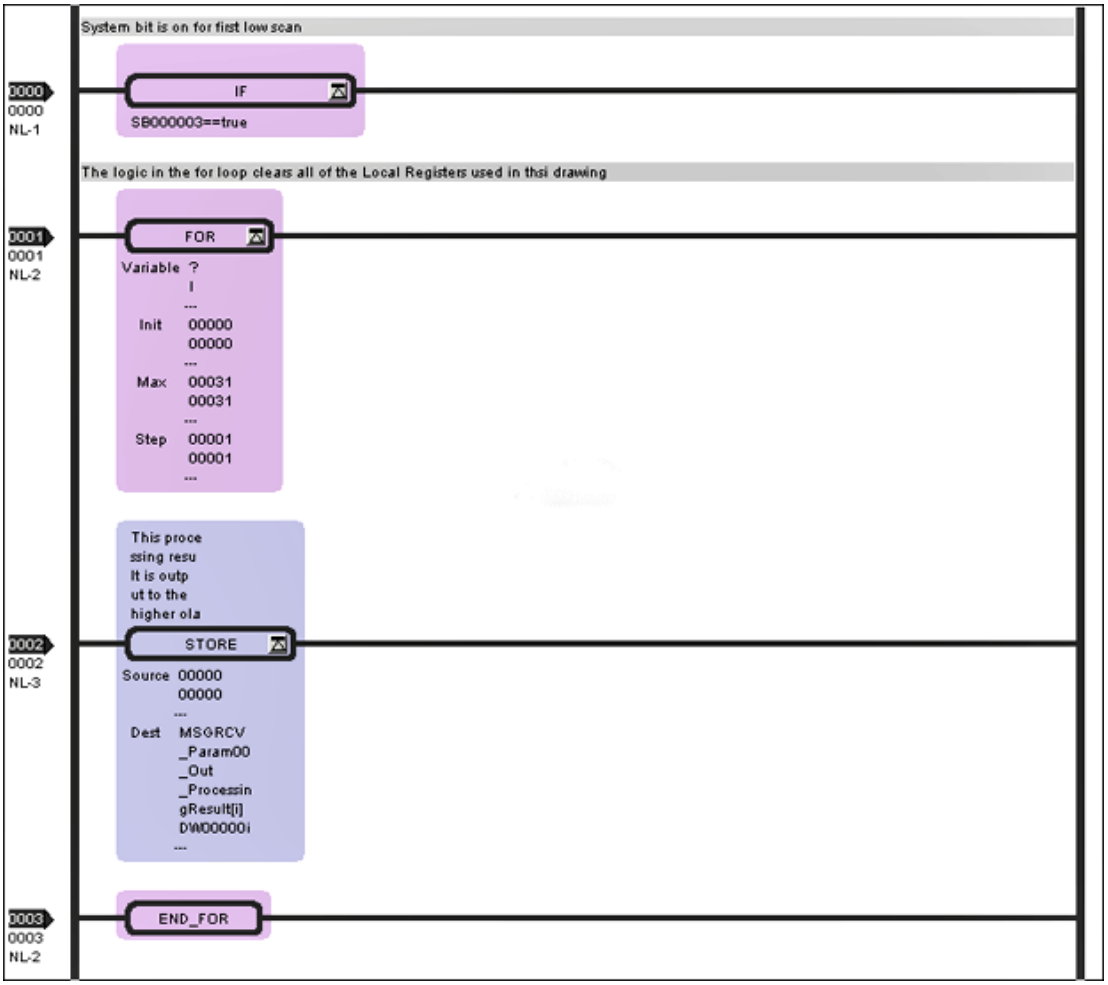
The first ladder is Drawing L in the Low Scan Programs. It calls the ladder that initializes the Message Receive function for the connection and also initializes and increments a scan counter.

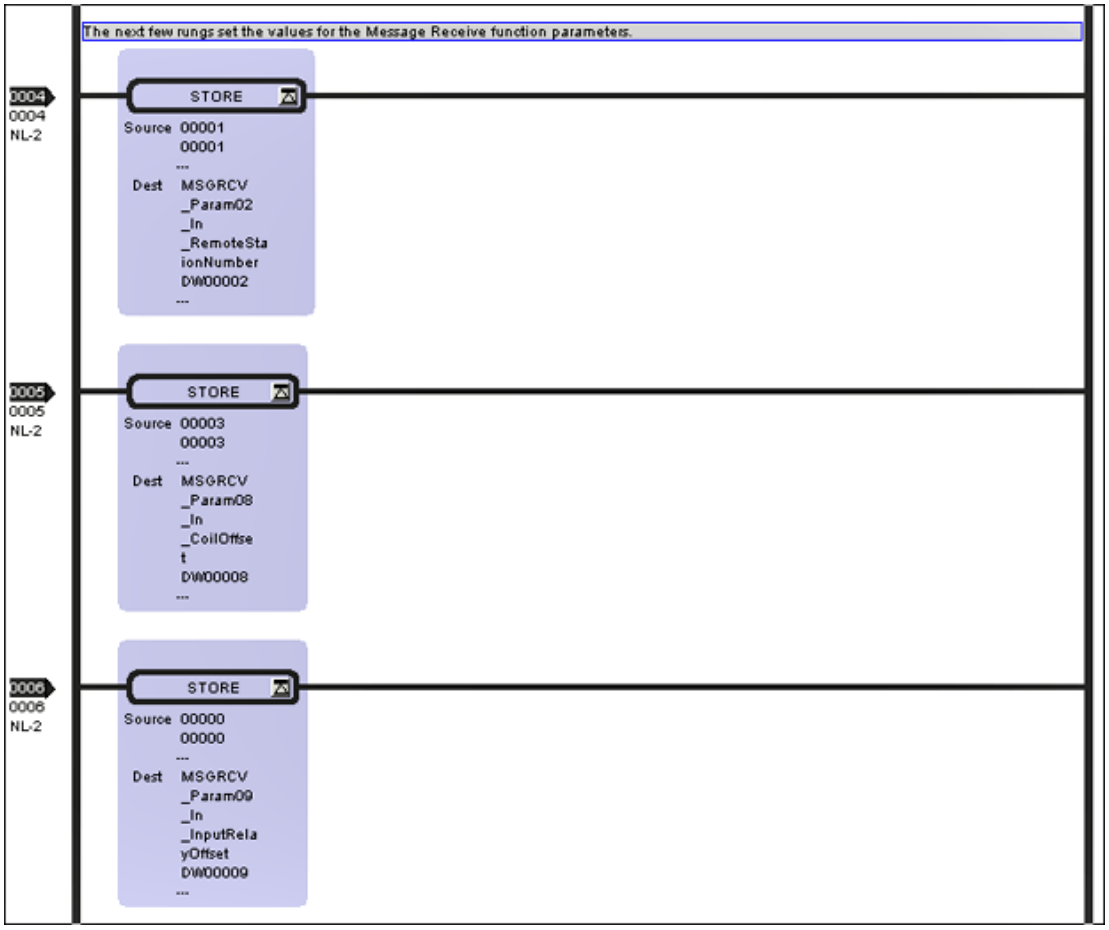


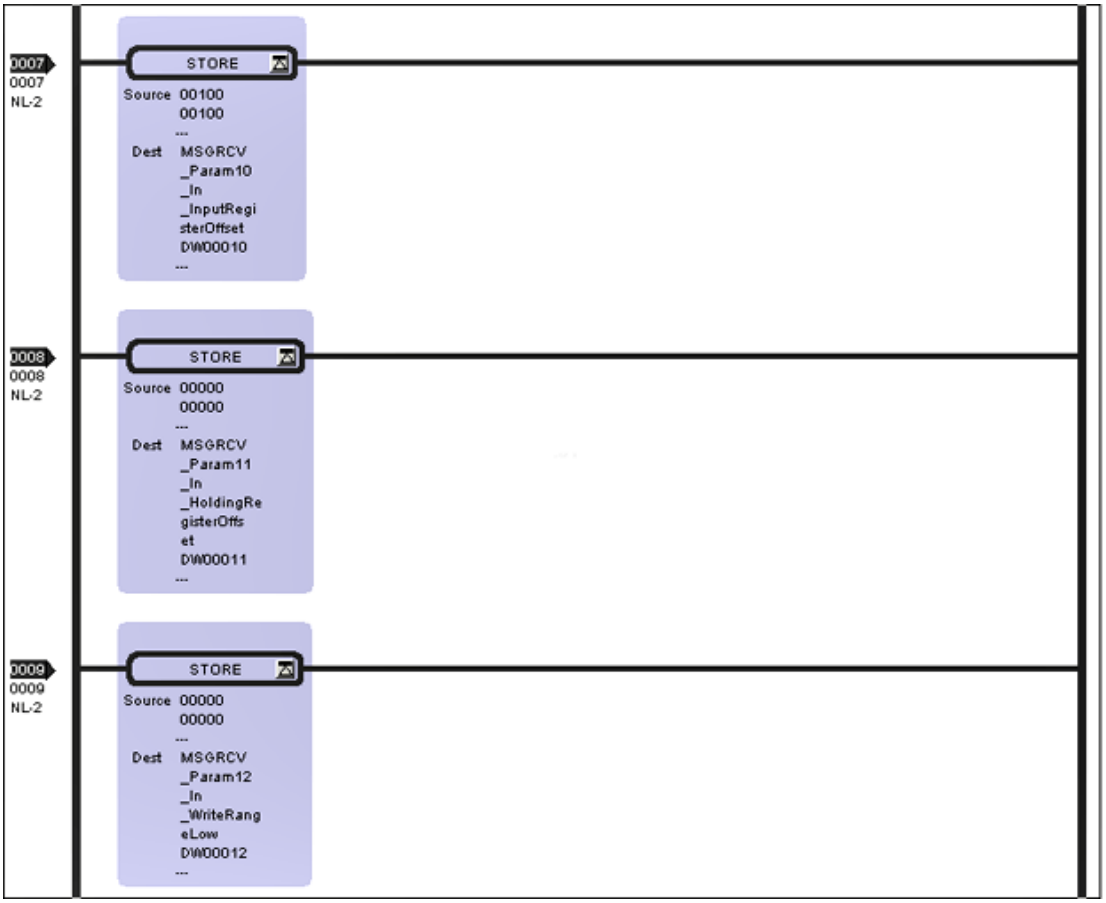
Ladder L01

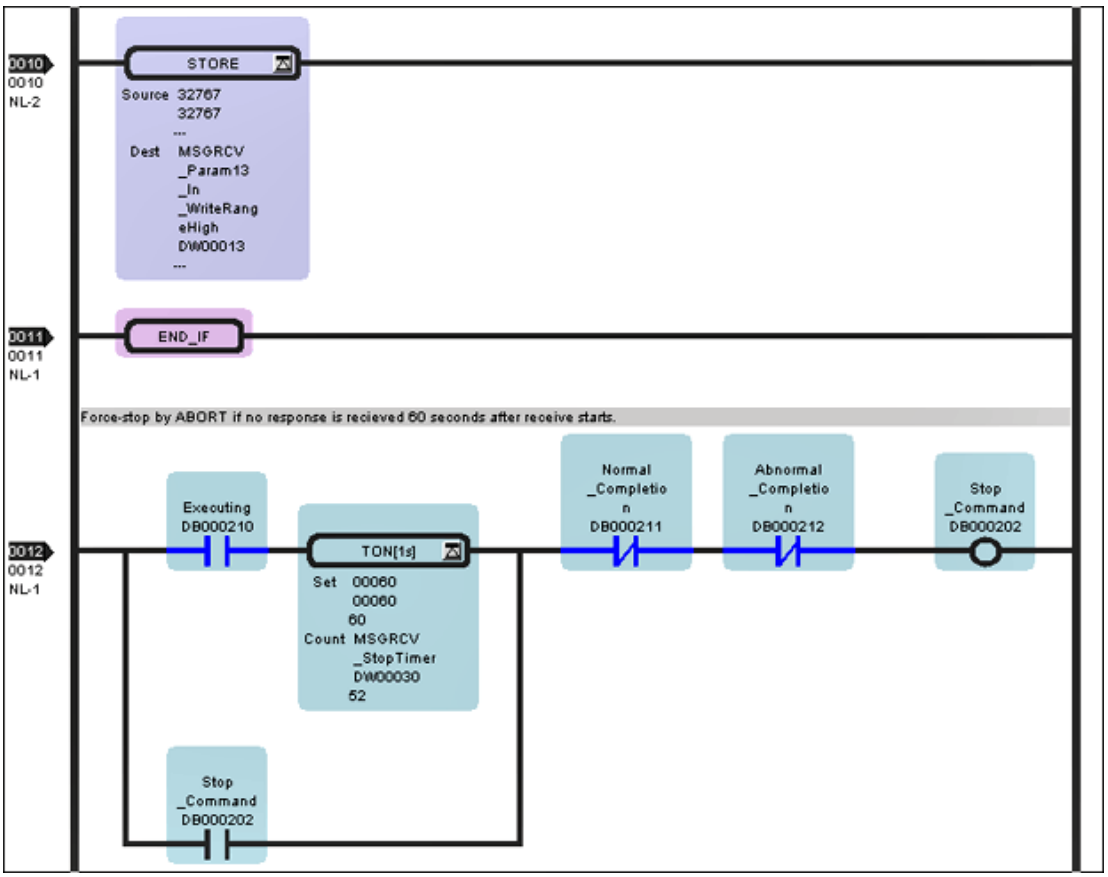
The next ladder program is Ladder L01, which initializes the Message Receive parameters and manages the process.

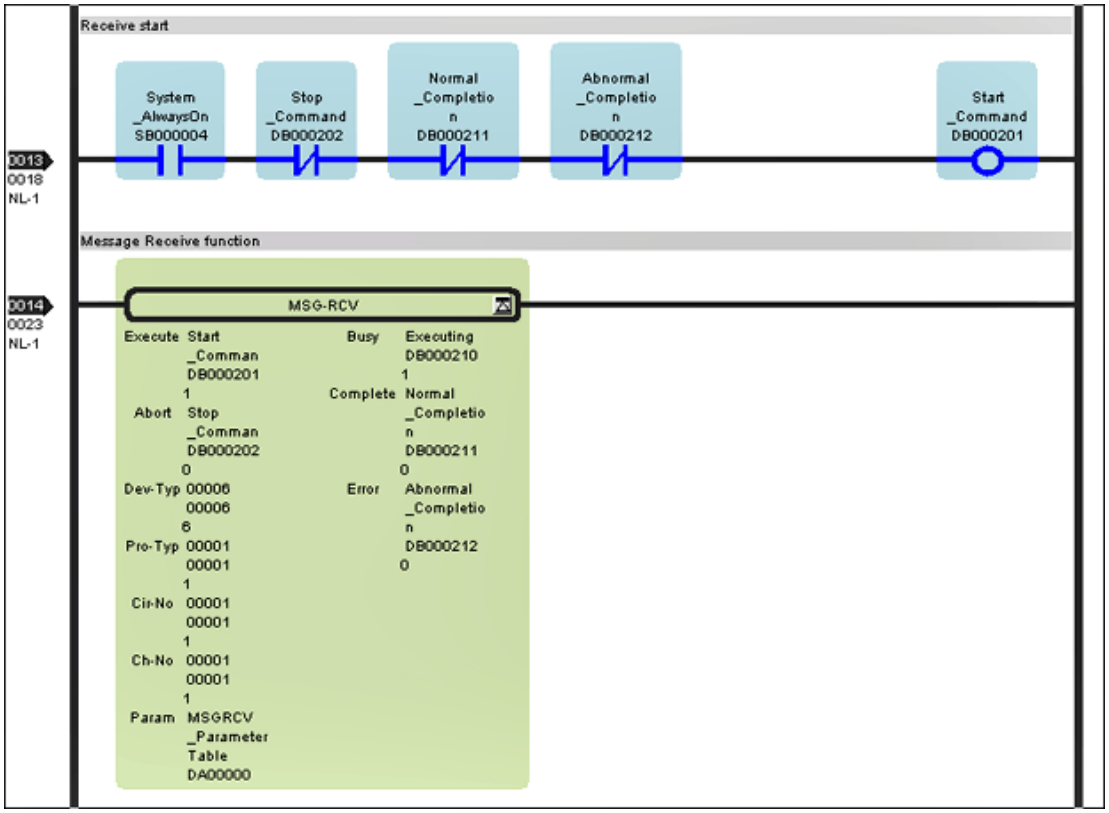
Note: The ladder program image has been separated to simplify viewing and printing.

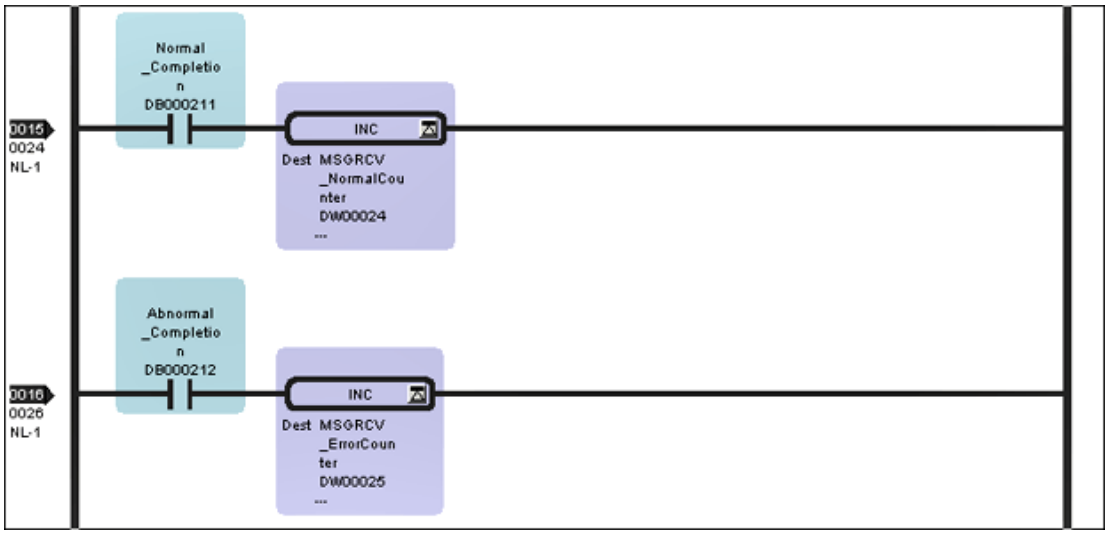


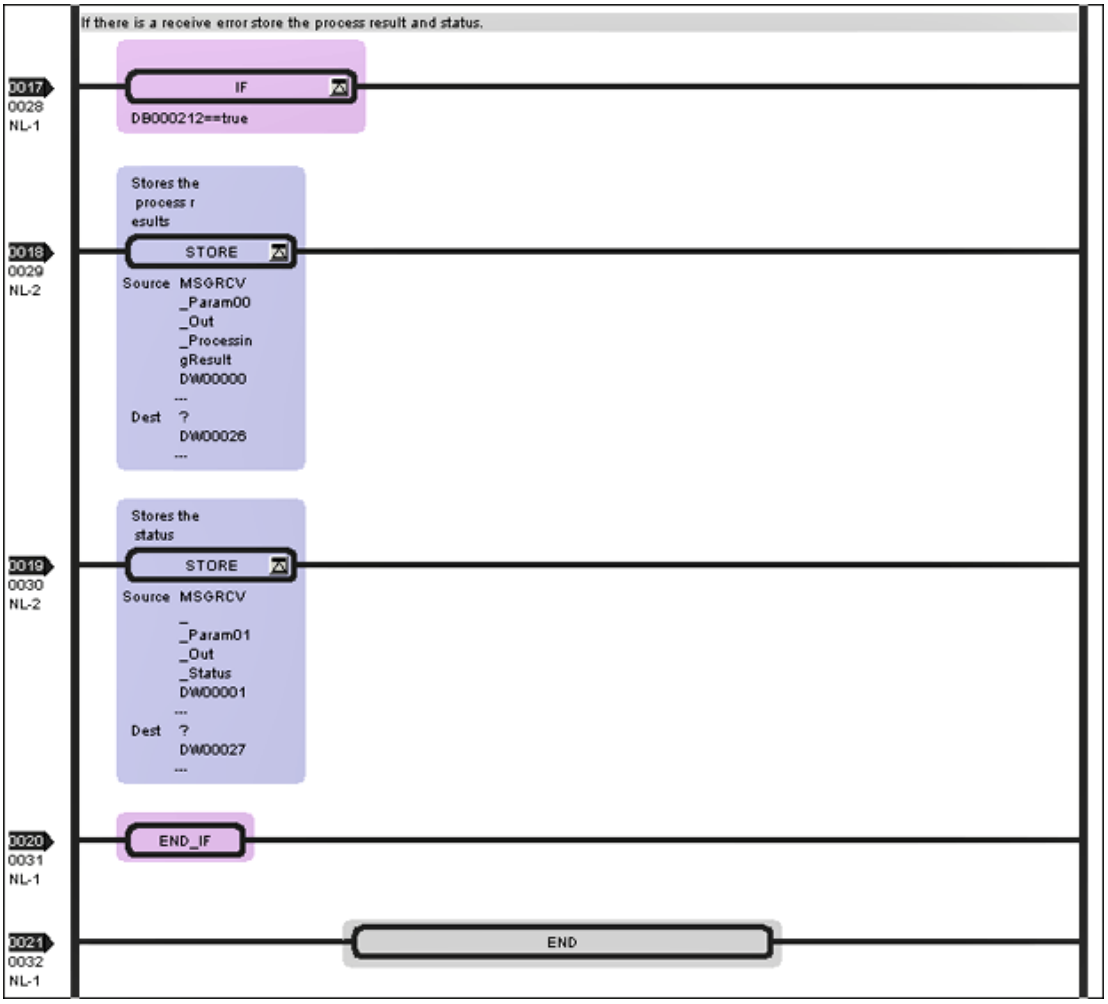

The next few rungs set the values for the Message Receive function parameters.
0004
0004
NL-2
STORE
Source 00001
00001
...
Dest MSGRCV
_Param02
_In
_RemoteSta
ionNumber
DW00002
...
0005
0005
NL-2
STORE
Source 00003
00003
...
Dest MSGRCV
_Param08
_In
_CoilOffse
t
DW00008
...
0006
0006
NL-2
STORE
Source 00000
00000
...
Dest MSGRCV
_Param09
_In
_InputRela
yOffset
DW00009
...


0007
0007
NL-2
STORE
Source 00100
00100
...
Dest MSGRCV
_Param10
_In
_InputRegi
sterOffset
DW00010
...
0008
0008
NL-2
STORE
Source 00000
00000
...
Dest MSGRCV
_Param11
_In
_HoldingRe
gisterOffs
et
DW00011
...
0009
0009
NL-2
STORE
Source 00000
00000
...
Dest MSGRCV
_Param12
_In
_WriteRang
eLow
DW00012
...


0010
0010
NL-2
STORE
Source 32767
32767
...
Dest MSGRCV
_Param13
_In
_WriteRang
eHigh
DW00013
...
0011
0011
NL-1
END_IF
Force-stop by ABORT if no response is recieved 60 seconds after receive starts.
0012
0012
NL-1
Executing
DB000210
TON[1s]
Set 00060
00060
60
Count MSGRCV
_StopTimer
DW00030
52
Normal
_Completio
n
DB000211
Abnormal
_Completio
n
DB000212
Stop
_Command
DB000202
Stop
_Command
DB000202


Receive start
System _AlwaysOn SB000004
Stop _Command DB000202
Normal _Completion DB000211
Abnormal _Completion DB000212
Start _Command DB000201
0013
0018
NL-1
Message Receive function
0014
0023
NL-1
MSG-RCV
Execute Start _Comman DB000201 1
Abort Stop _Comman DB000202 0
Dev-Typ 00006 00006 6
Pro-Typ 00001 00001 1
Cir-No 00001 00001 1
Ch-No 00001 00001 1
Param MSGRCV _Parameter Table DA00000
Busy Executing DB000210 1
Complete Normal _Completion DB000211 0
Error Abnormal _Completion DB000212 0


Normal _Completion DB000211
0015
0024
NL-1
INC
Dest MSGRCV _NormalCounter DW00024
Abnormal _Completion DB000212
0016
0026
NL-1
INC
Dest MSGRCV _ErrorCounter DW00025



Load and Save the Configuration

After the ladders have been edited, verify and load them to the Controller. Then, cycle the power on the controller to initialize the Ethernet module.

Note: The Yaskawa MP Series Ethernet device may release the communications socket connection by default after a two-minute interval from when the last communications occurred between the device and the server.

To maintain a responsive connection to the Yaskawa MP Series Ethernet module, it is recommended that the **Set** point value of the **On Delay Timer** at rung 12 of ladder L01 should be set greater than the slowest group update rate occurring in the client application by ten or more seconds.

For example, if the client has a group update rate of 120 seconds (120000ms), the On Delay Timer set point should be changed to at least 130 seconds (130000ms).

Index

A

Address '<address>' is out of range for the specified device or register 22
Address Descriptions for MPxxxx (218IF Module) 19
Address Descriptions for MPxxxx (218IF Module) Devices 17
Array size is out of range for address '<address>' 22
Array support is not available for the specified address 22
Attempts Before Timeout 11
Auto-Demotion 11

B

BCD 16
Block Sizes 12
Boolean 16

C

Channel Assignment 8
Communications Parameters 12
Communications Timeouts 10-11
Connect Timeout 11

D

Data Collection 9
Data Type '<type>' is not valid for device address '<address>' 22
Data Types Description 16
Demote on Failure 11
Demotion Period 12
Device '<device name>' block request [<start address> to <end address>] responded with exception '<exception response>' 24
Device '<device name>' is not responding 23
Device address '<address>' contains a syntax error 23
Device address '<address>' is not supported by model '<model name>' 23
Device address '<address>' is Read Only 23

Discard Requests when Demoted 12
Do Not Scan, Demand Poll Only 10
Double 16
Driver 8
DWord 16

E

Error Descriptions 21

F

Failure to start Winsock communications 25
Float 16

G

General 8

H

Hardware Configuration for MPxxxx (218IF Module) 30
Hardware Configuration for MPxxxx (218IF Module) - Ladders 40
Hardware Configuration for MPxxxxiec 34

I

ID 9
Identification 8
Illegal data address for tag '<tag address>' on device '<device name>' 25
Illegal data address in block [<start address> to <end address>] on device '<device name>' 25
Illegal data value for tag '<tag address>' on device '<device name>' 25
Illegal data value in block [<start address> to <end address>] on device '<device name>' 26
Illegal function code '<function code (hex)>' in block [<start address> to <end address>] on device '<device name>' 26
Illegal function code '<hex function code>' for tag '<tag address>' on device '<device name>' 26
Initial Updates from Cache 10
Inter-Request Delay 11

L

LBCD 16
Long 16

M

Memory Mapping for MPxxxxiec Devices 14
Missing address 23
Modbus server device '<device name>' detected a memory parity error 27
Modbus server device '<device name>' has failed 27
Modbus server device '<device name>' is busy 27
Model 8

N

Name 8

O

Operating Mode 9
Optimizing Yaskawa MP Series Ethernet Communications 15
Overview 4

R

Redundancy 13
Request Timeout 11
Respect Tag-Specified Scan Rate 10

S

Scan Mode 10
Setup 5
Short 16
Simulated 9
String 16

T

Tag '<tag address>' on device '<device name>' responded with exception '<exception code>' 27
Timeouts to Demote 11

U

Unable to bind to adapter <adapter>. Connect failed. 28
Unable to create a socket connection for Device <device> 28
Unable to write to '<address>' on device '<device name>' 24
Unexpected response frame received for block [<start address> to <end address>] on device '<device name>' 28
Unexpected response frame received for tag '<tag address>' on device '<device name>' 28

W

Winsock initialization failed OS Error = <error code> 29
Winsock shut down failed OS Error = <error code> 29
Winsock V1.1 or higher must be installed to use the Yaskawa MP Series Ethernet device driver 29
Word 16