Yaskawa MP Series Ethernet Driver Help

Table of Contents

Yaskawa MP Series Ethernet Driver Help

Help version 1.032

CONTENTS

Overview

The Yaskawa MP Series Ethernet Driver provides an easy and 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.

Device Setup

Supported Devices

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

Communication Protocol

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

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 should be in the range of 0 to 255.

Connection Timeout

This parameter specifies the time that the driver will wait for a connection to be made with a device. Depending on network load, the connect time may vary with each connection attempt. The default setting is 3 seconds. The valid range is 1 to 60 seconds.

Request Timeout

This parameter specifies the time that the driver will wait on a response from the device before giving up and going on to the next request. Longer timeouts only affect performance if a device is not responding. The default setting is 1000 milliseconds. The valid range is 100 to 9999 milliseconds.

Retry Attempts

This parameter specifies the number of times that the driver will retry a message before giving up and going on to the next message. The default setting is 3 retries. The valid range is 1 to 10.

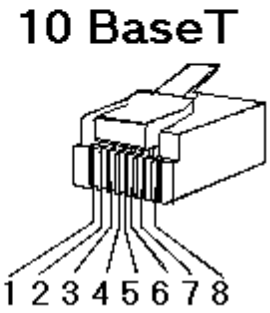
See Also: Hardware Configuration

Cable Diagrams

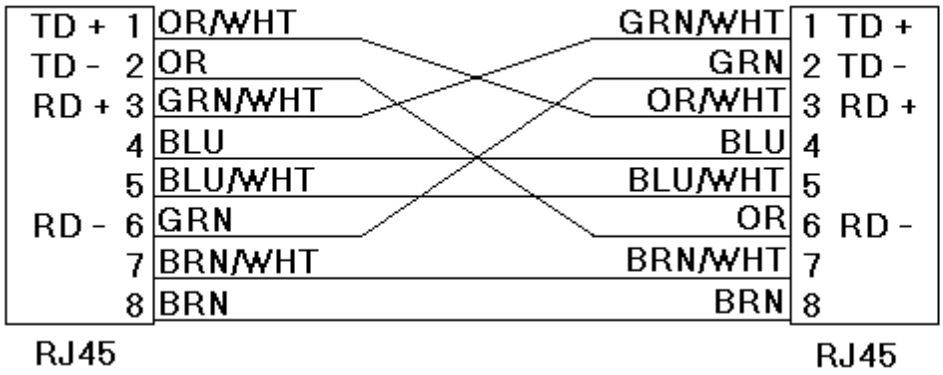
Both the MPxxxx 218IF Ethernet communications module and the MPxxxxiec are connected to the network via an Ethernet twisted pair transceiver. The pinout of the 218IF Ethernet connector is displayed in the table below.

Pin Number	Signal Name	Remarks
1	GND	Shield ground
2	CI+	Collision detection (+)
3	DO+	Send data (+)
4	GND	Shield ground
5	DI+	Receive data (+)
6	PWRGND	12 V power supply
7	N.C.	Not Connected
8	GND	Shield ground
9	CI-	Collision detection (-)
10	DO-	Send data (-)
11	GND	Shield ground
12	DI-	Receive data (-)
13	+12 V	12 V power supply
14	GND	Shield ground
15	N.C.	Not Connected

Patch Cable (Straight Through)



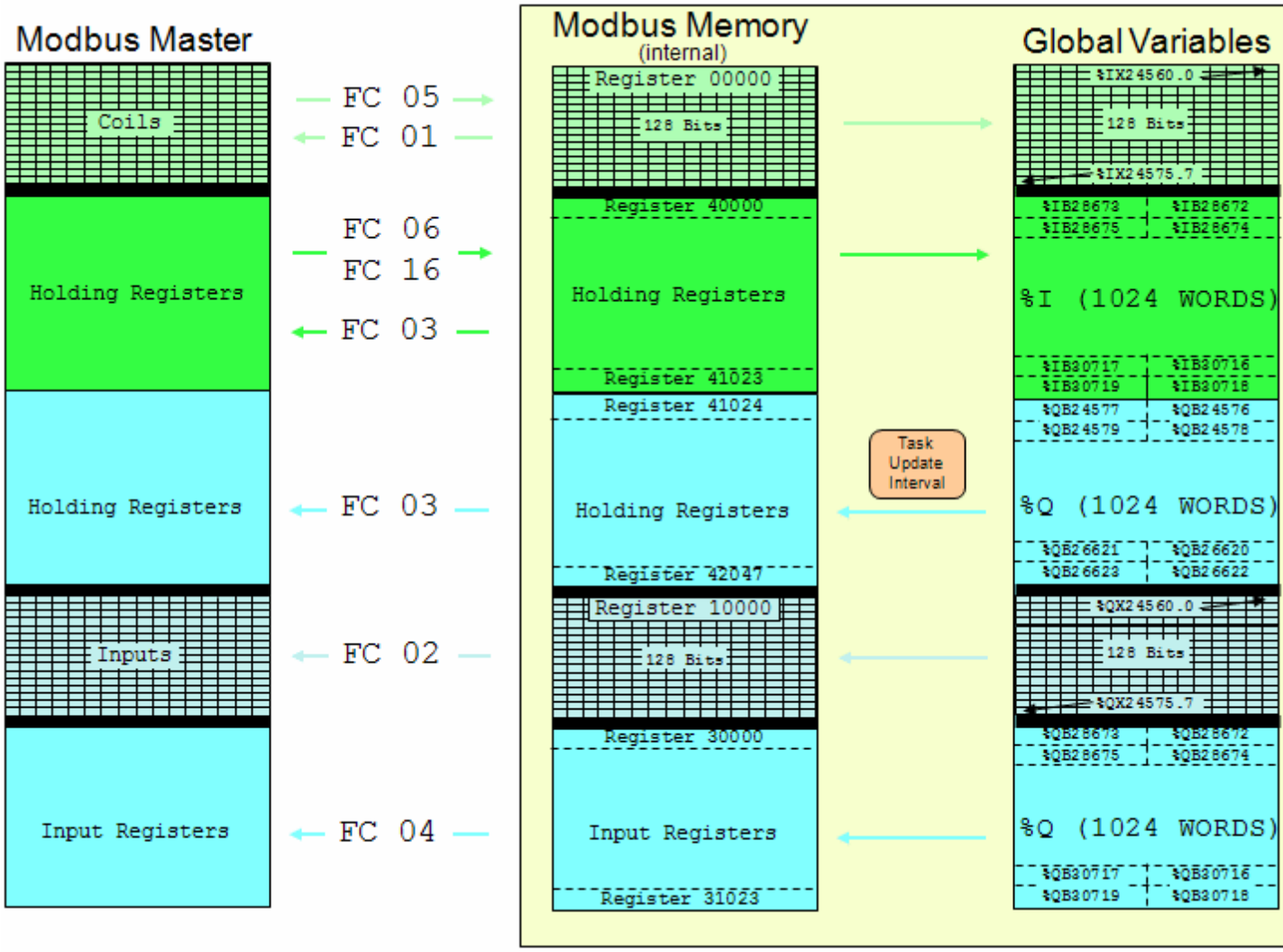
Crossover Cable



Memory Mapping for MPxxxxiec Devices

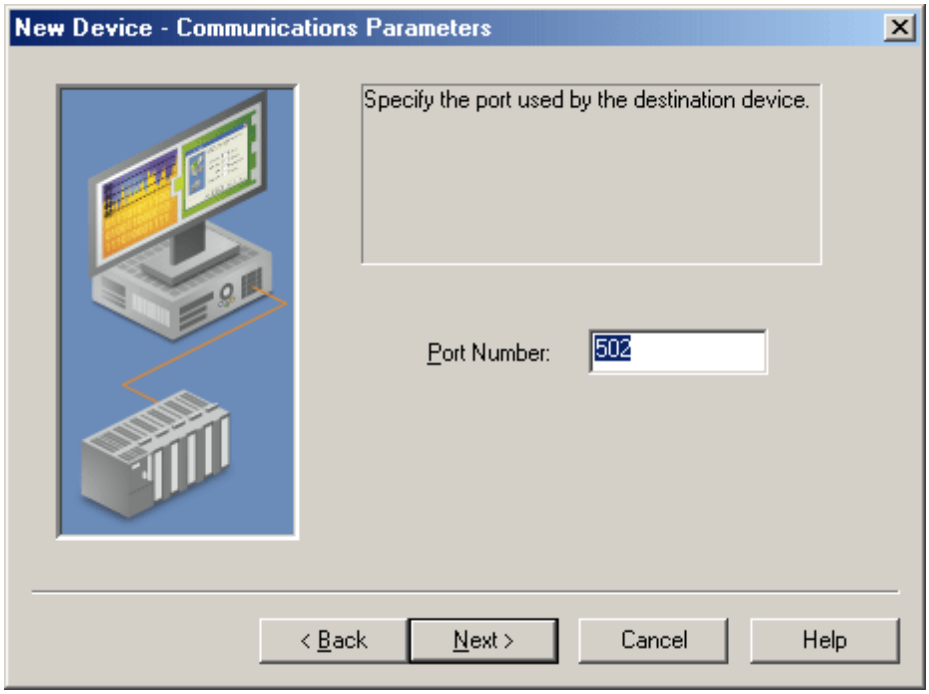
This driver (Master) communicates via Modbus TCP to MPxxxxiec Series controllers that are configured as a Modbus Server/Slave. The image below displays the Modbus memory map: it not only shows how it relates to the Global Variables (iec memory) in the controller, but also shows the Modbus function codes (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]

Communications Parameters

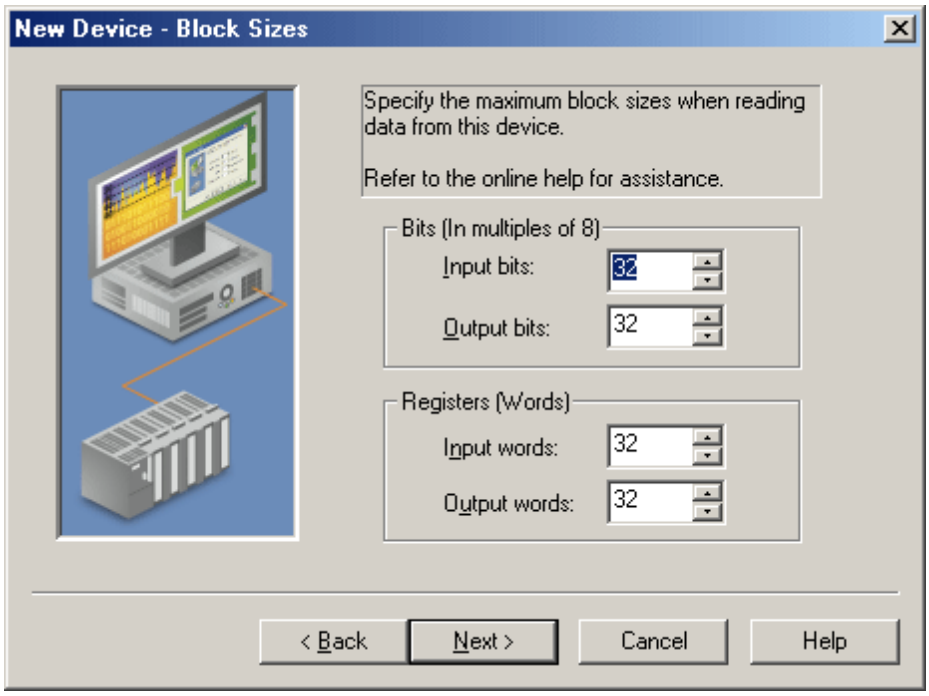


Description of the parameter is as follows:

- **Port Number:** This parameter 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.

Block Sizes



Block Sizes for MPxxxx (218IF Module) Controllers

Bit Addresses

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

Register Addresses
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.

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.

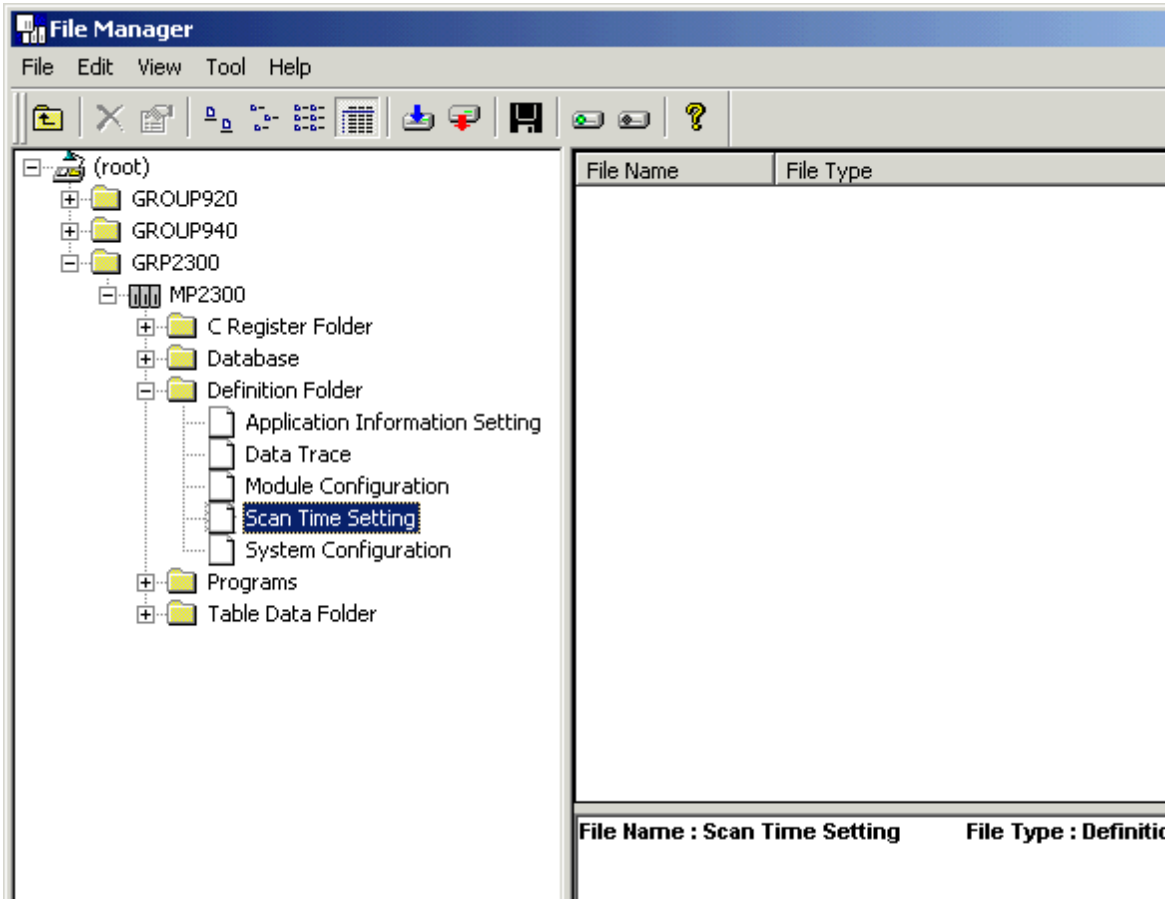
Configuring a 218IF Module

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

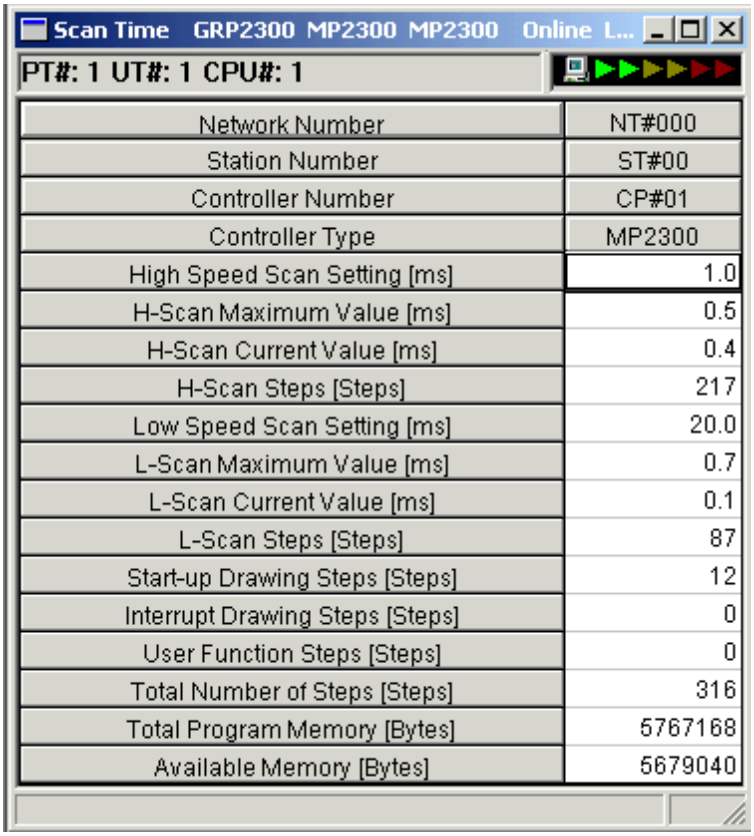
Note: The examples shown are from MotionWorks (MPE720) v6.02.

1. Open the **File Manager**.
2. Create a project for the controller.

3. Double-click on **Scan Time Setting**.

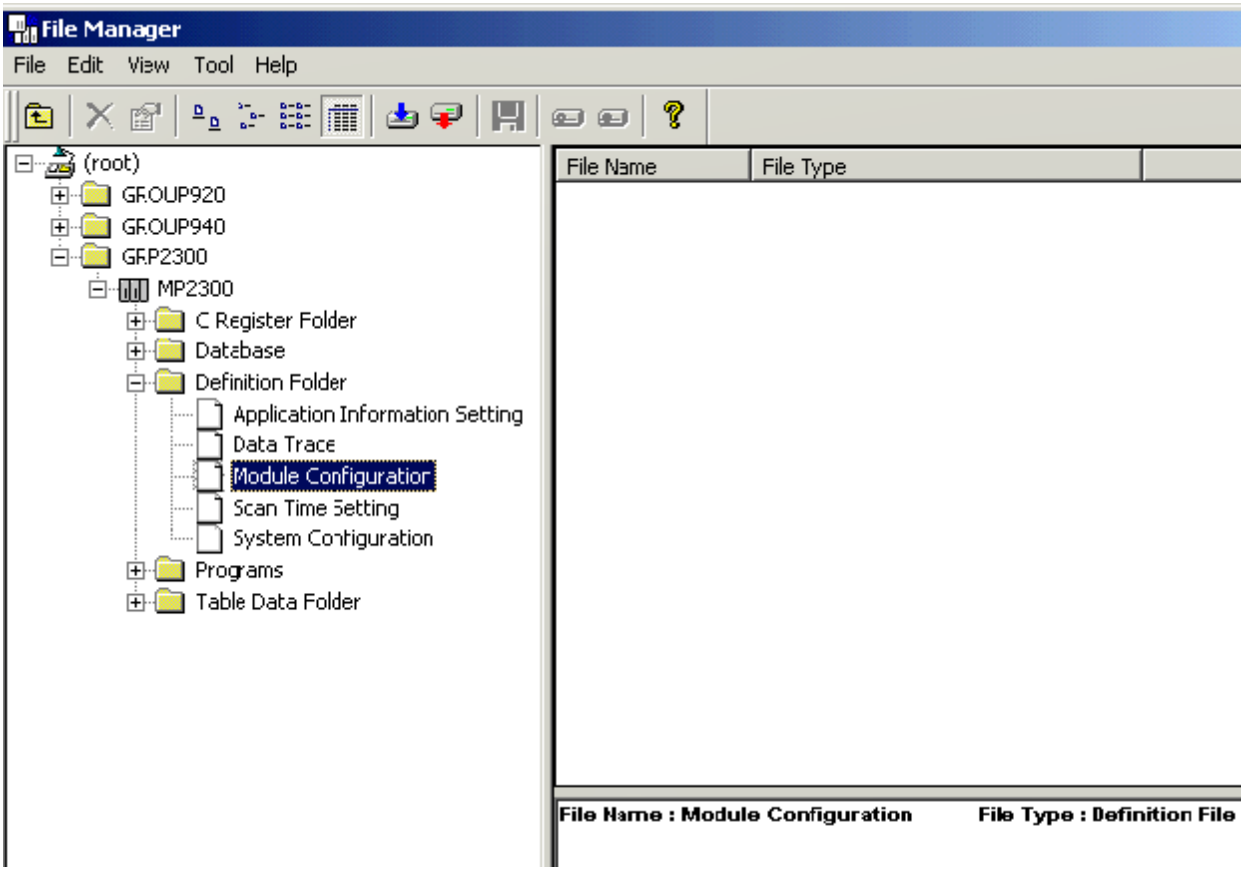


4. In **Scan Time**, note the **High Speed Scan Setting** and **Low Speed Scan Setting** fields. Yaskawa recommends 1ms for the High Scan setting and 20ms for the Low Scan setting.



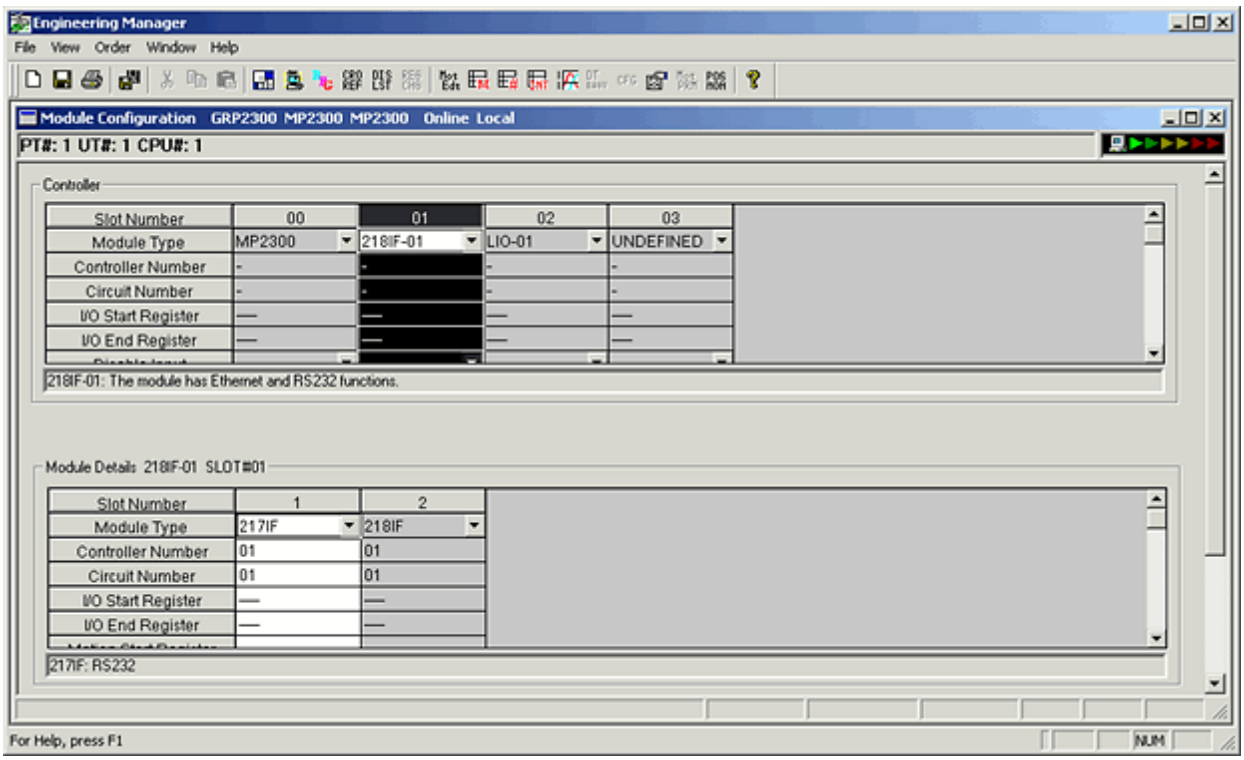
5. When the scan time settings are set, close the Scan Time dialog.

6. Double-click on **Module Configuration** to open the **Engineering Manager**.



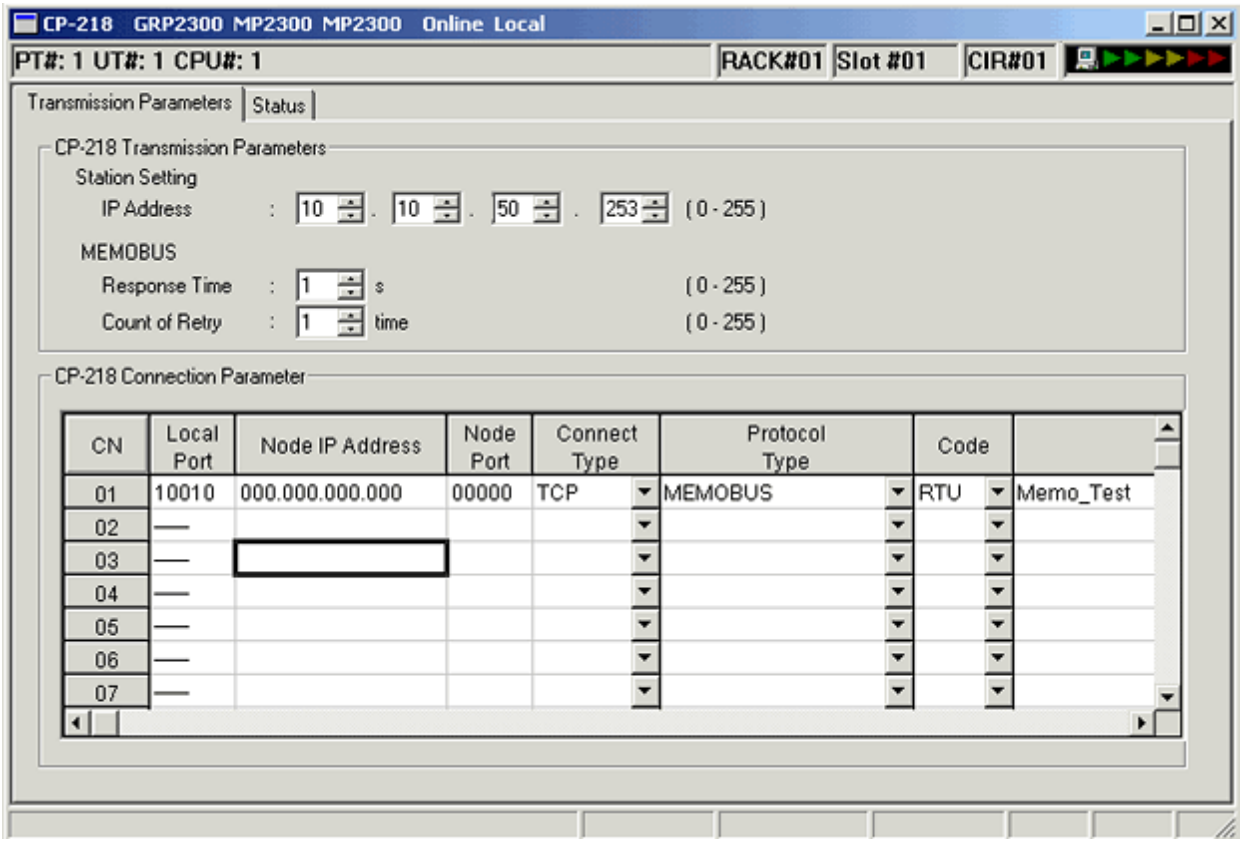
7. In the Engineering Manager, select the **218IF module**.

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

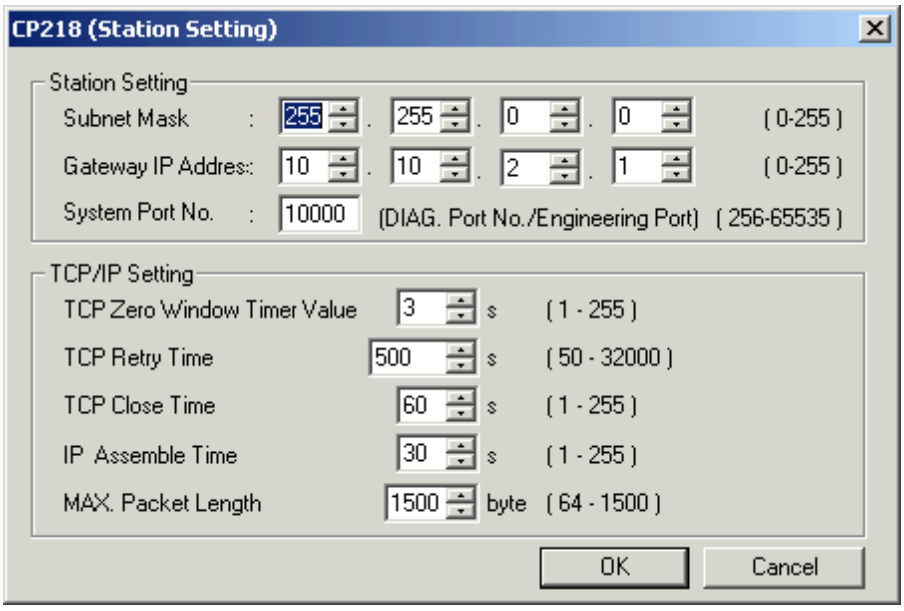


9. Click on the **Transmission Parameters** tab.

   - 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 slave 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**.

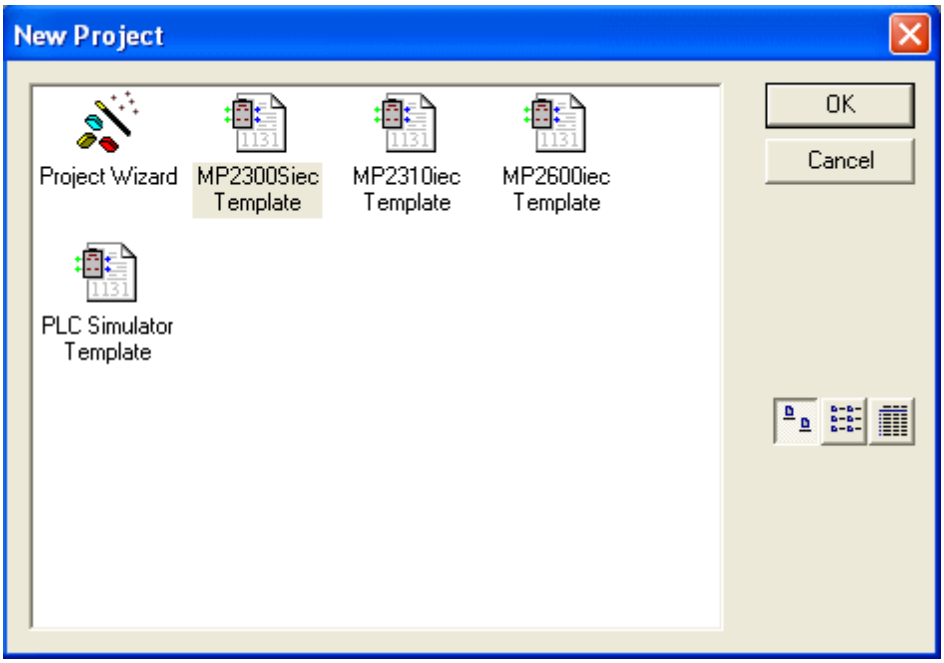
12. Next, create the ladder programs as described in **Hardware Configuration - Ladders**.

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

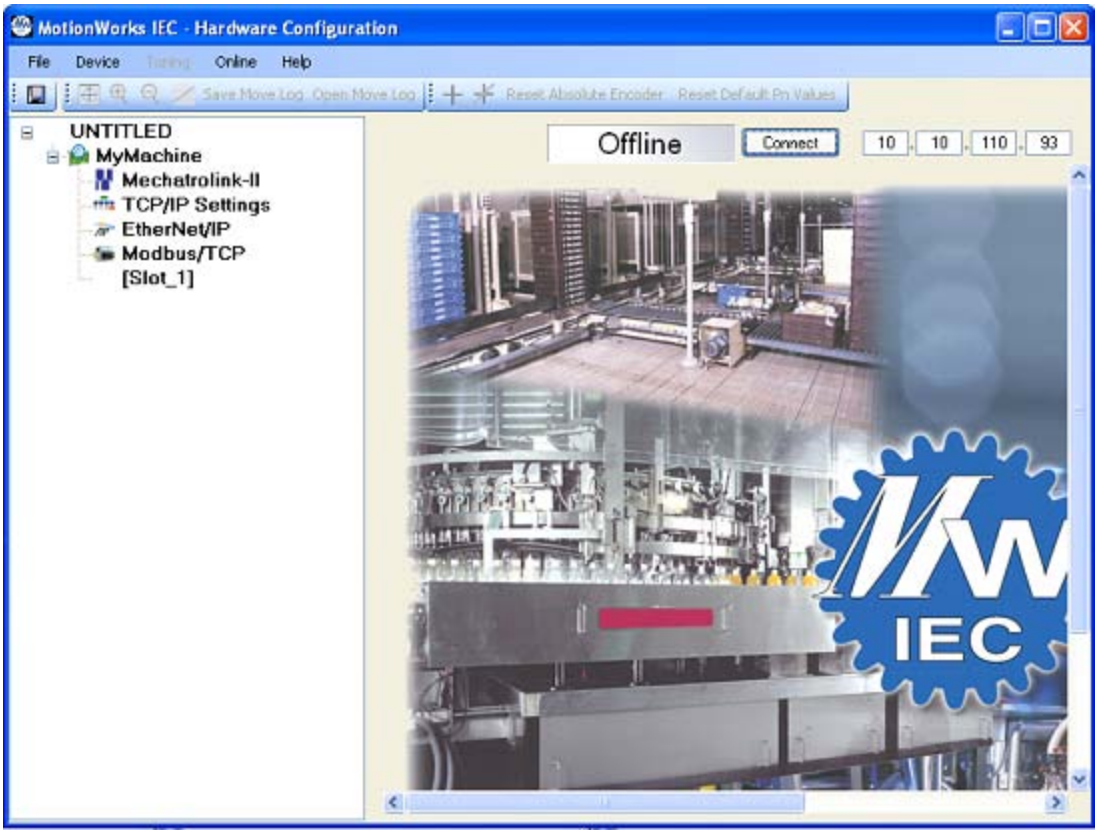
Hardware Configuration for MPxxxxiec

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.

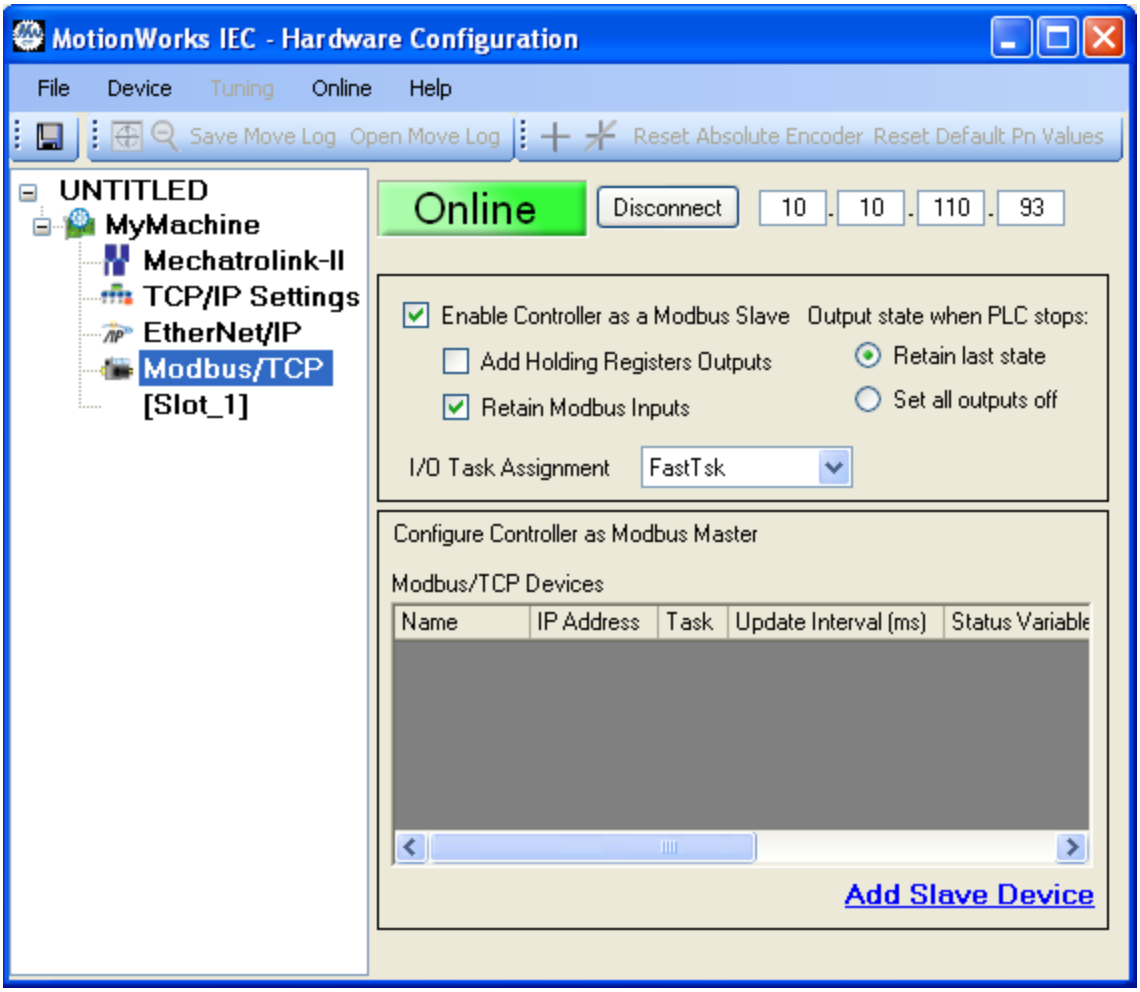


3. Launch the Hardware Configuration by clicking the Hardware Configuration icon . The Hardware Configuration window should appear as shown below.



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".



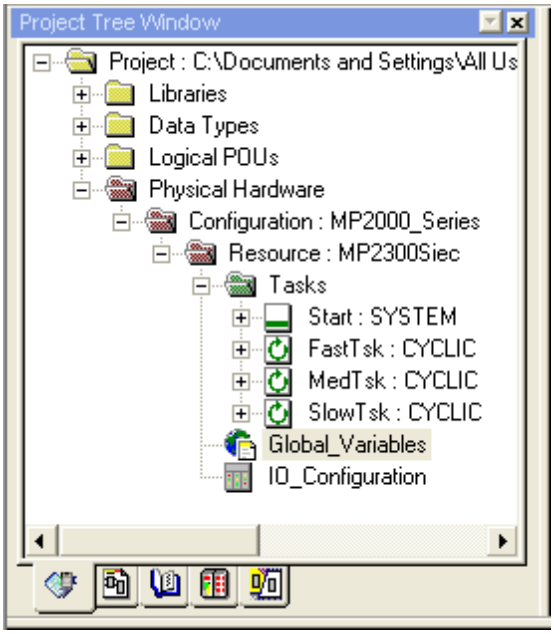
6. Then, do the following:

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

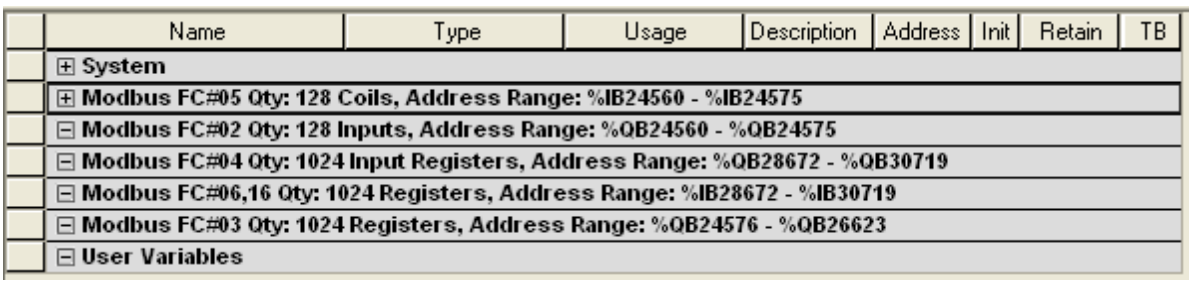
7. Once finished, save the changes by clicking **File** | **Save**.

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**.

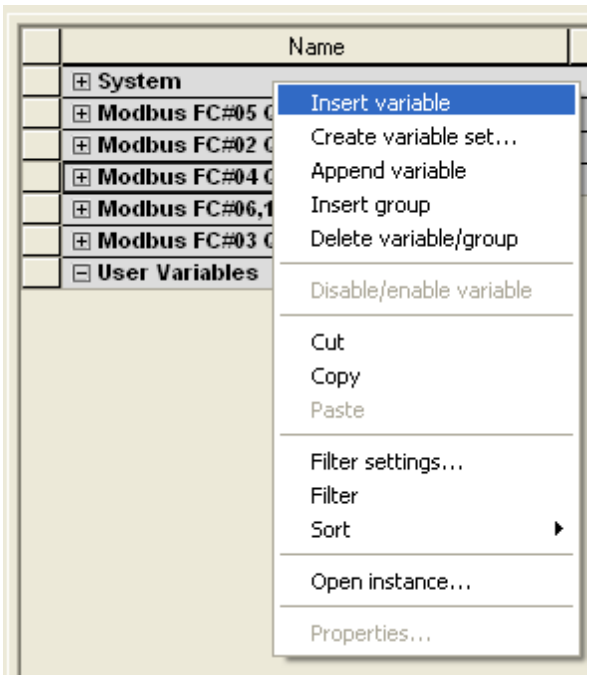
9. Expand the project tree by clicking **Configuration: MP2000_Series** | **Resource:MP2300Siec**. Then, select **Global_Variables**.



10. The following categories should be displayed:



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

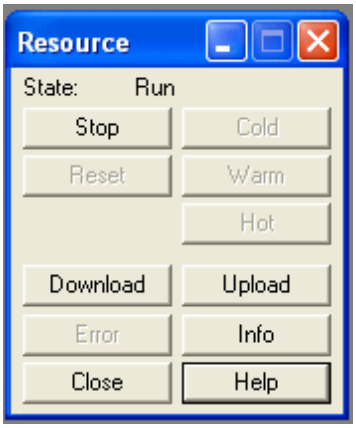


12. In the **Address** column, specify the MotionWorks IEC Pro 1.2.3.12 address.

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 in order to view data. For more information, refer to the instructions below.

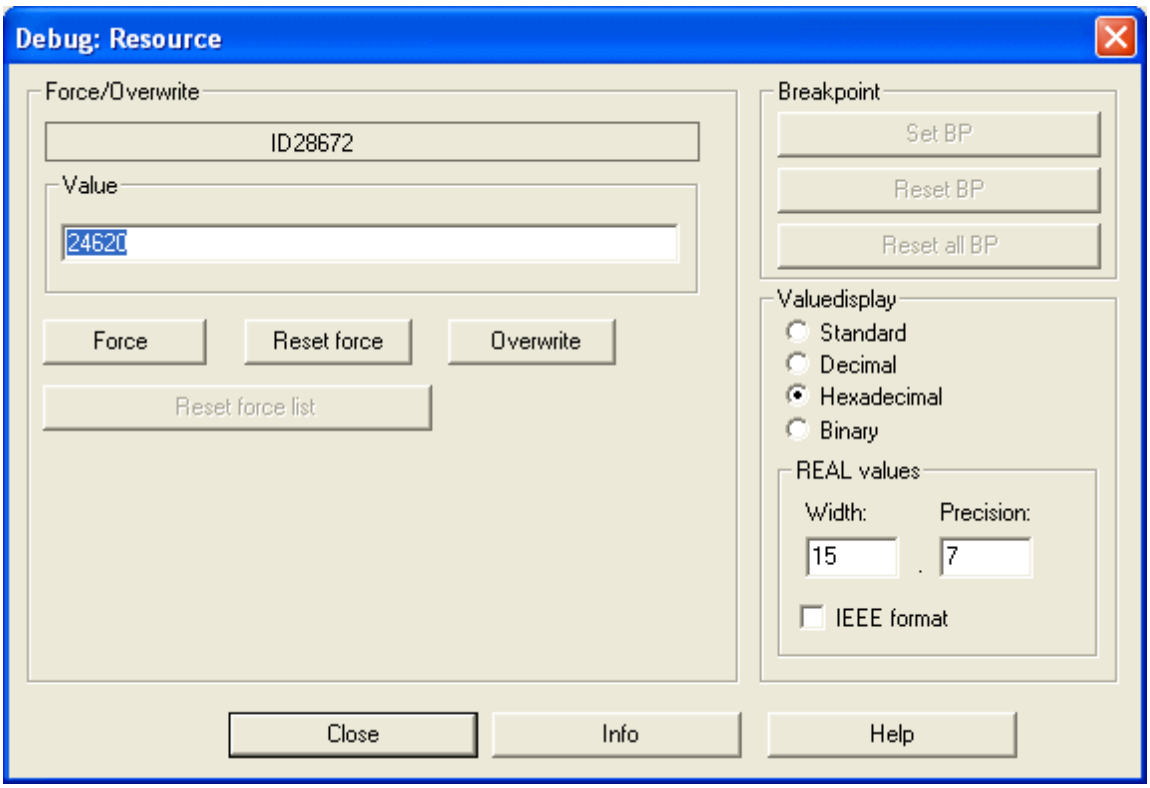
1. First, click the **Debug On/Off** icon  .
2. Then, return to the **Global_Variables** window.

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**.



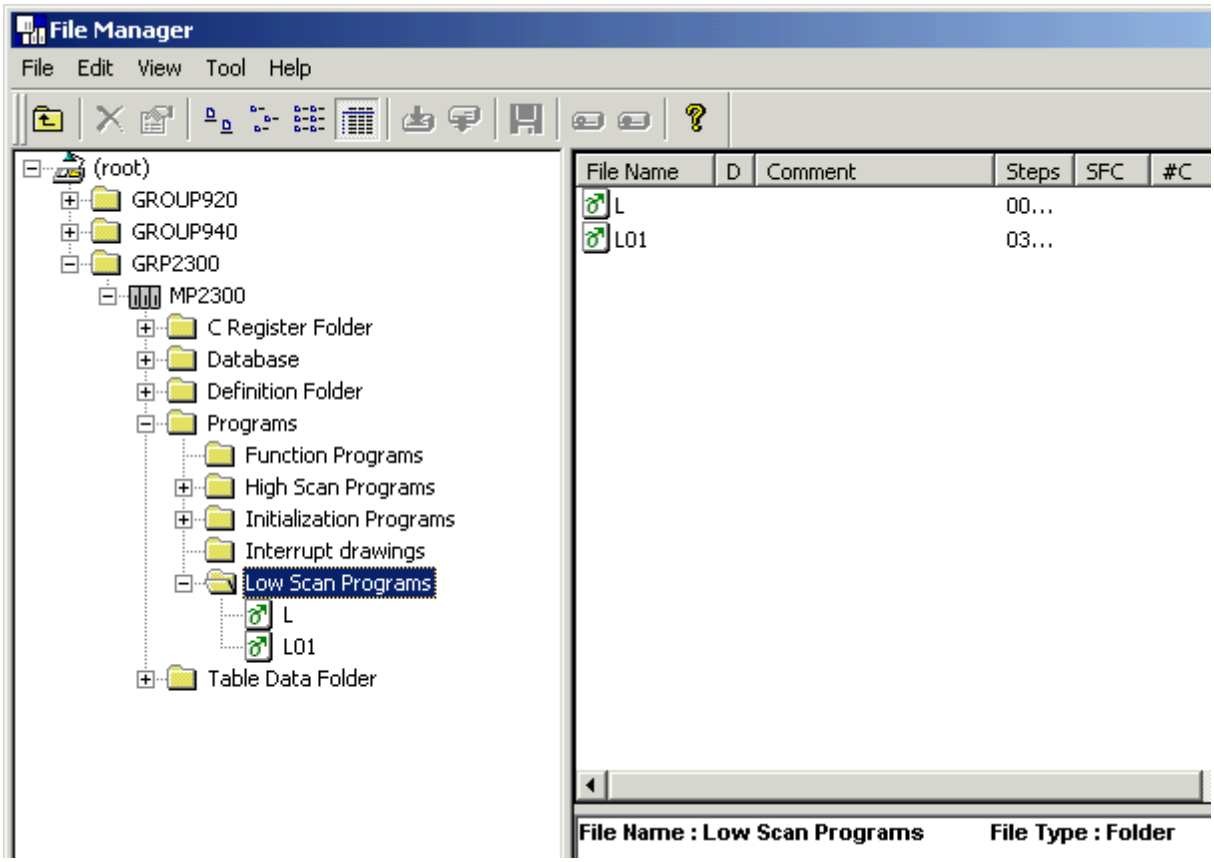
Hardware Configuration for MPxxxx (218IF Module) - Ladders

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

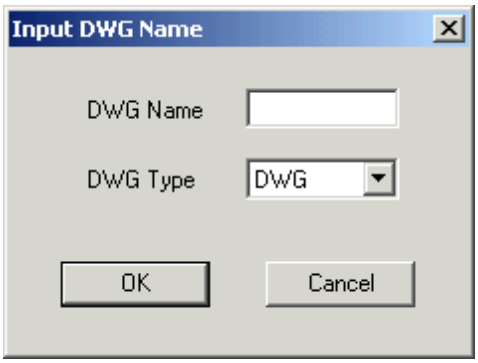
Important: The ladders are absolutely necessary: otherwise, the driver will not be able to connect to the 218IF module or exchange data.

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.

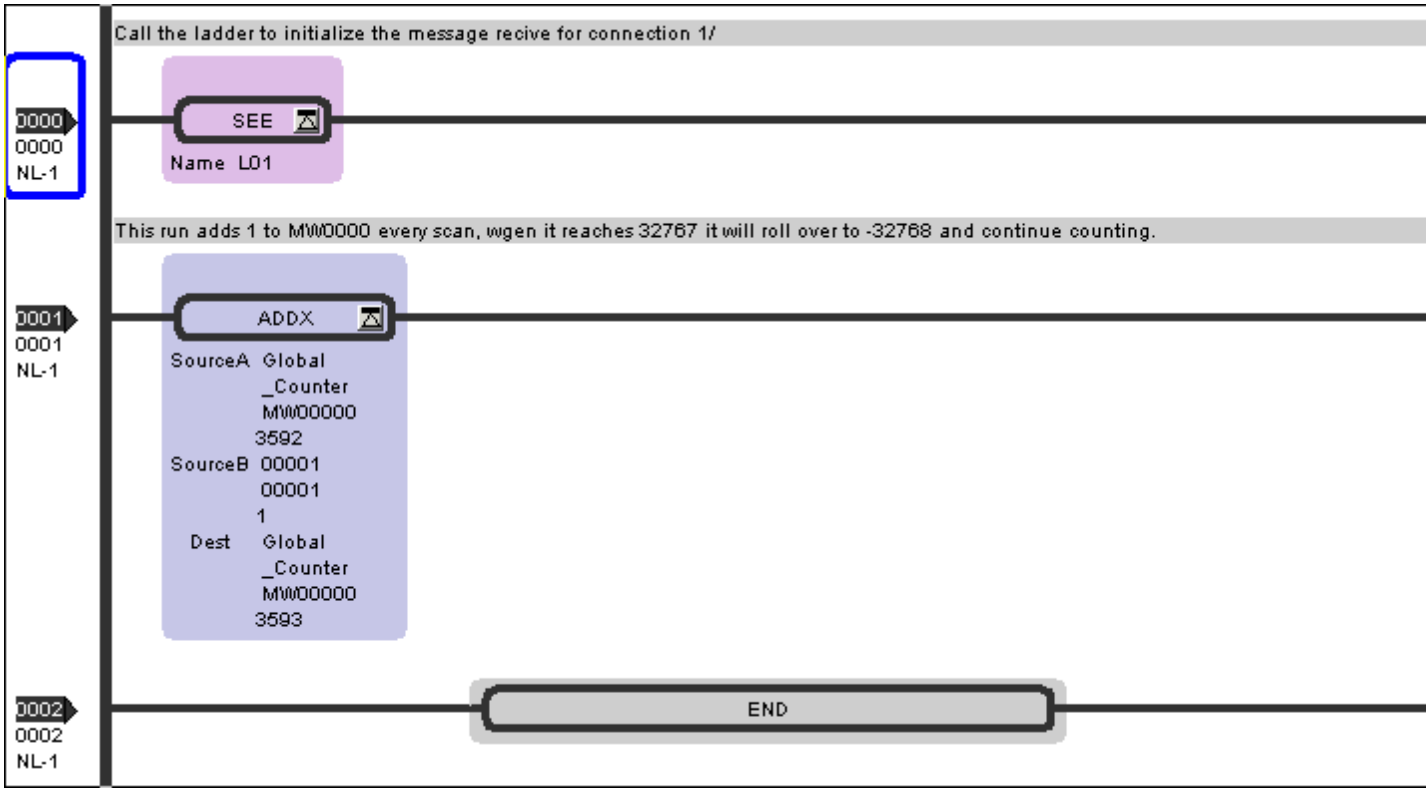


Note: The image below shows how the Input DWG Name dialog is displayed.



Drawing L in Low Scan Programs

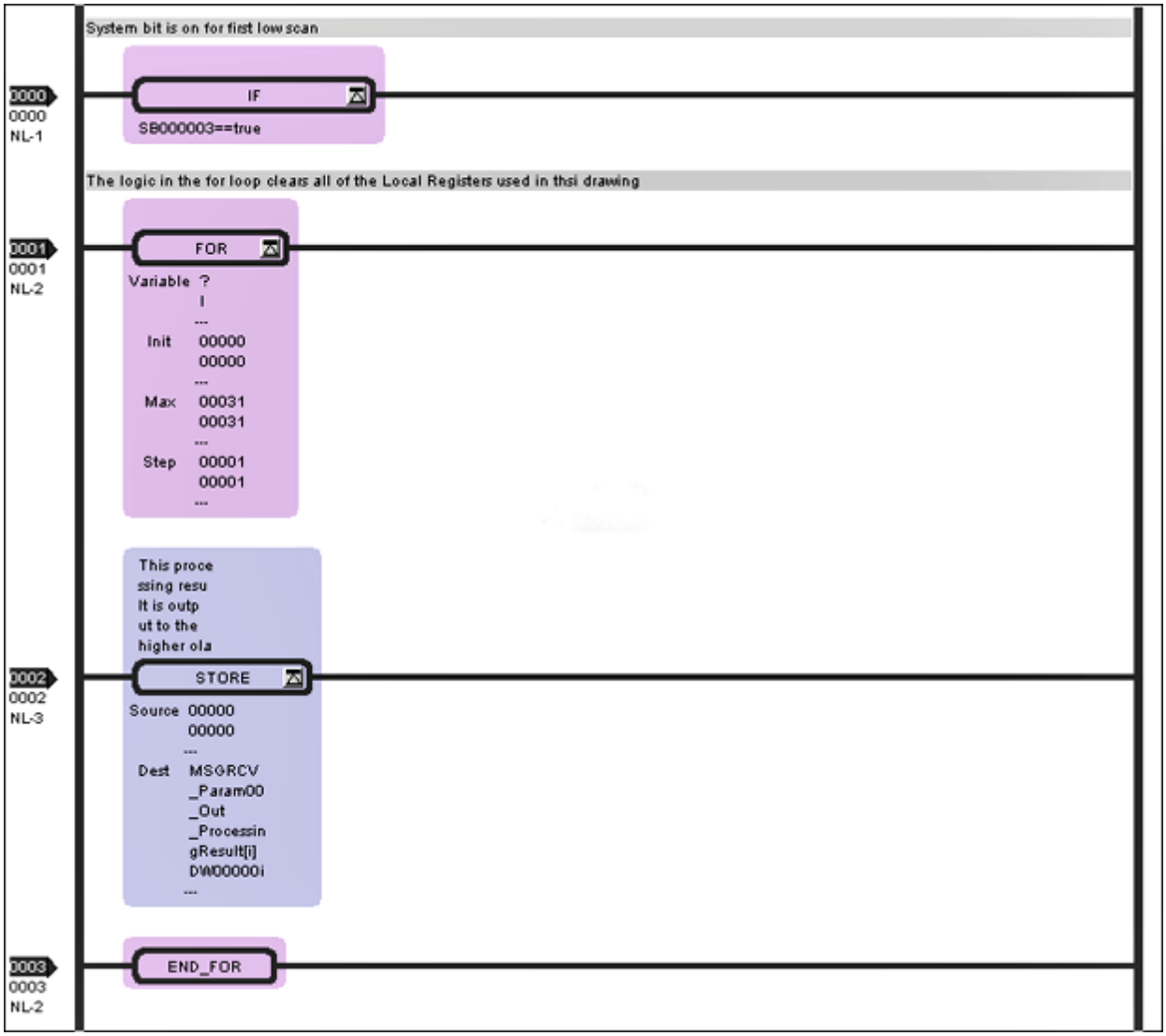
The first ladder is Drawing L in the Low Scan Programs. It calls the ladder that initializes the Message Receive function for the connection and also initializes and increments a scan counter.



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 in order to simplify viewing and printing.


System bit is on for fiist low scan
0000
0000
NL-1
IF
SB000003==true
The logic in the for loop clears all of the Local Registers used in thsi drawing
0001
0001
NL-2
FOR
Variable ?
I

Init 00000
00000

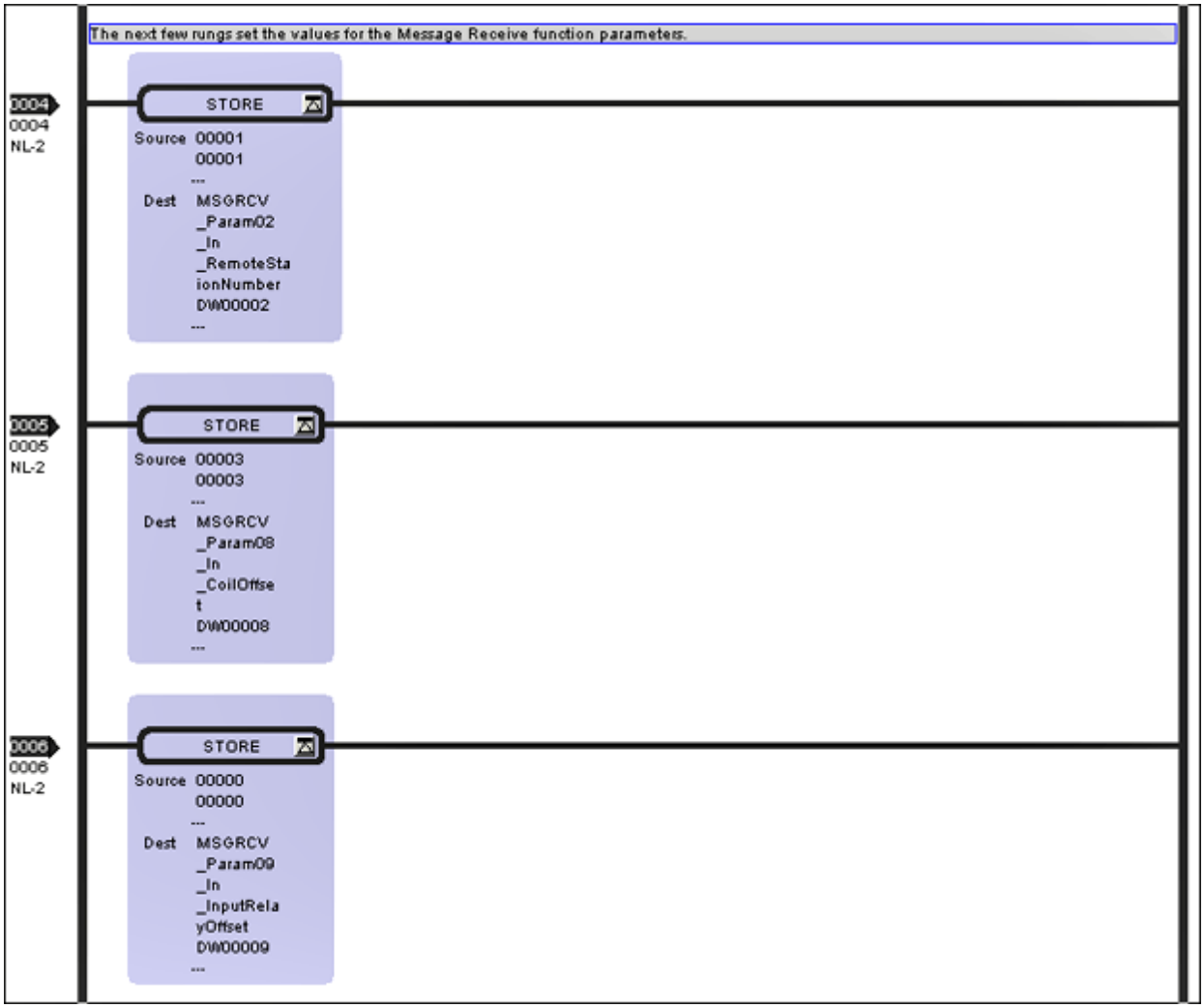
Max 00031
00031

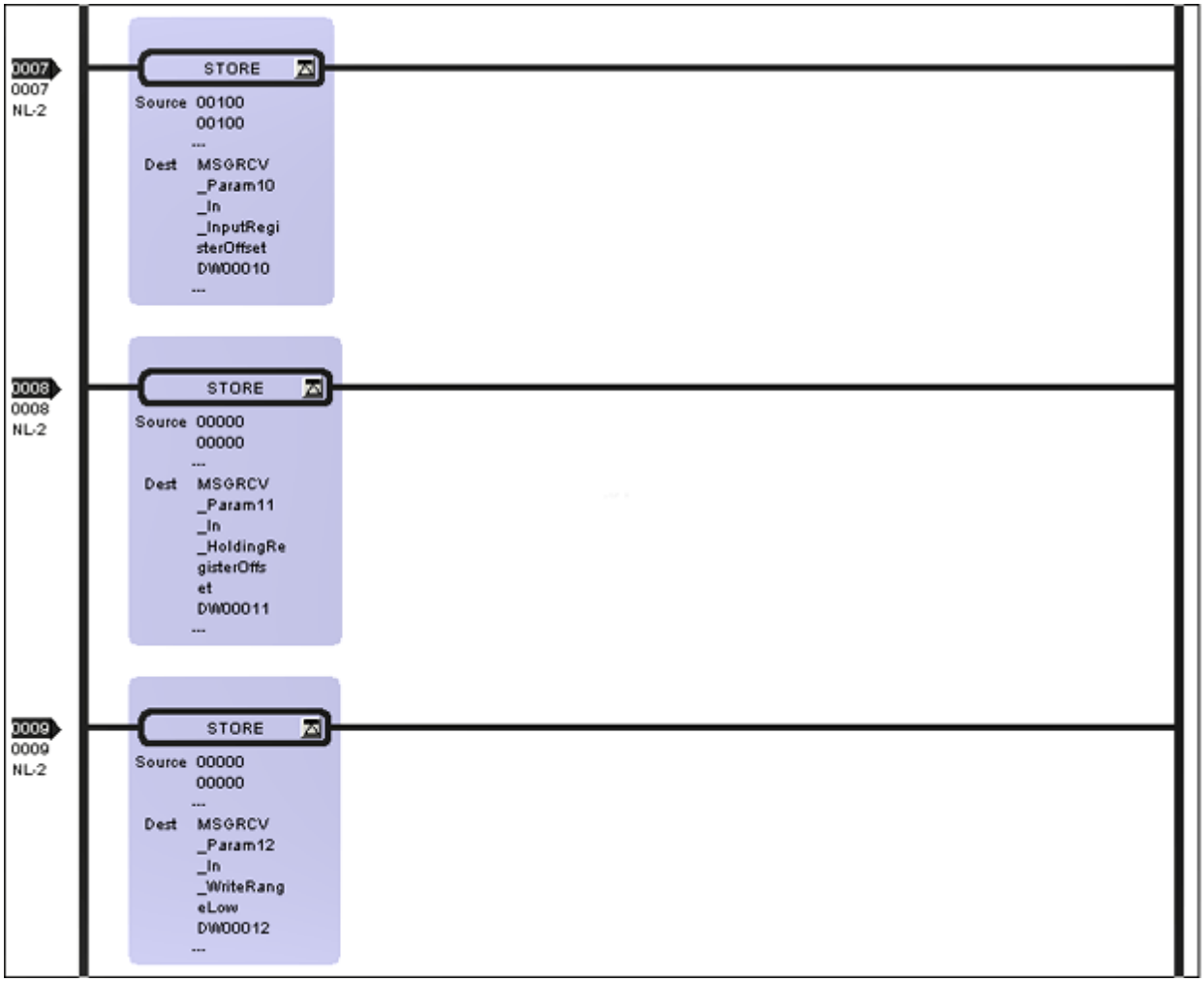
Step 00001
00001

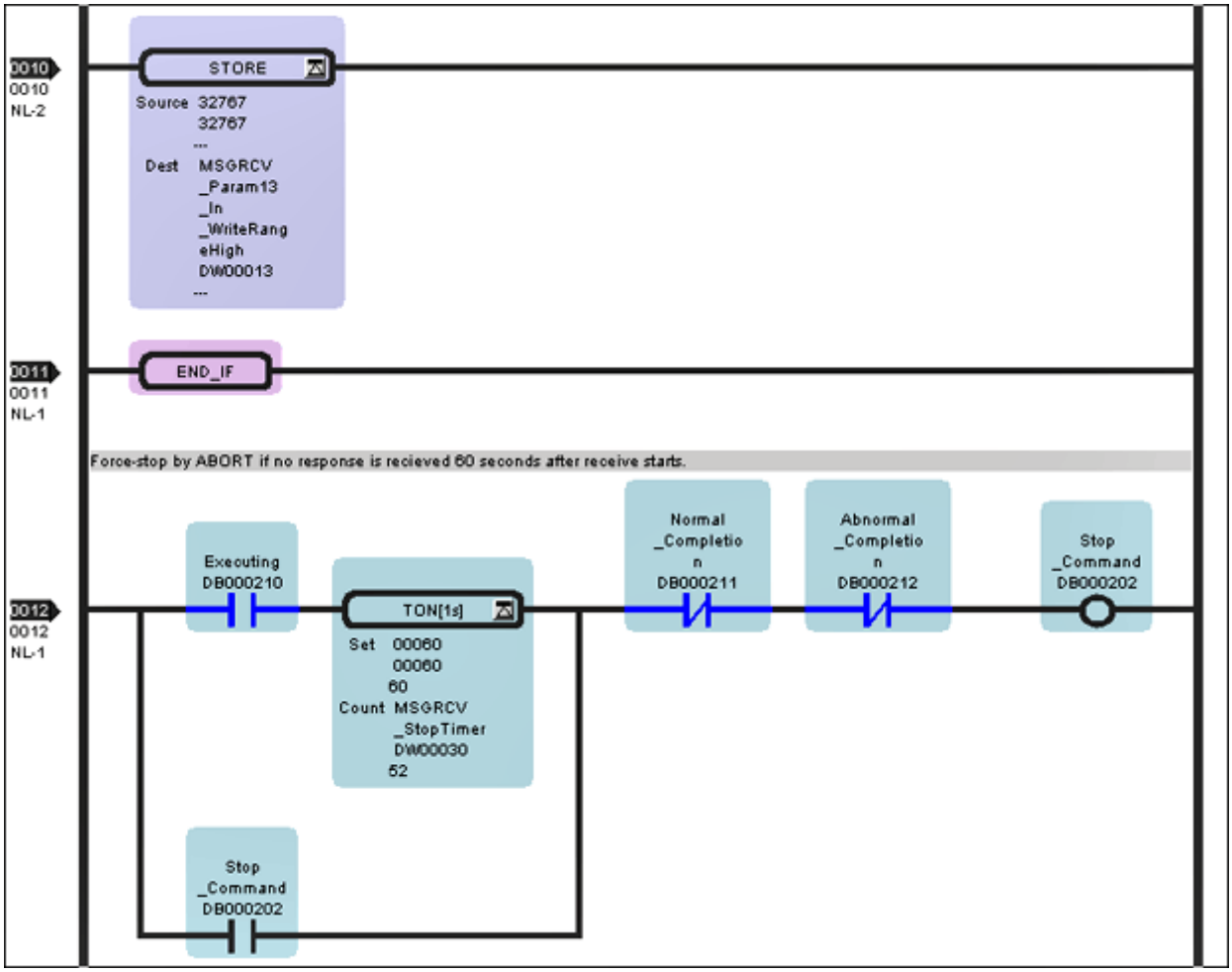
This proce
ssing resu
lt is outp
ut to the
higher ola
0002
0002
NL-3
STORE
Source 00000
00000

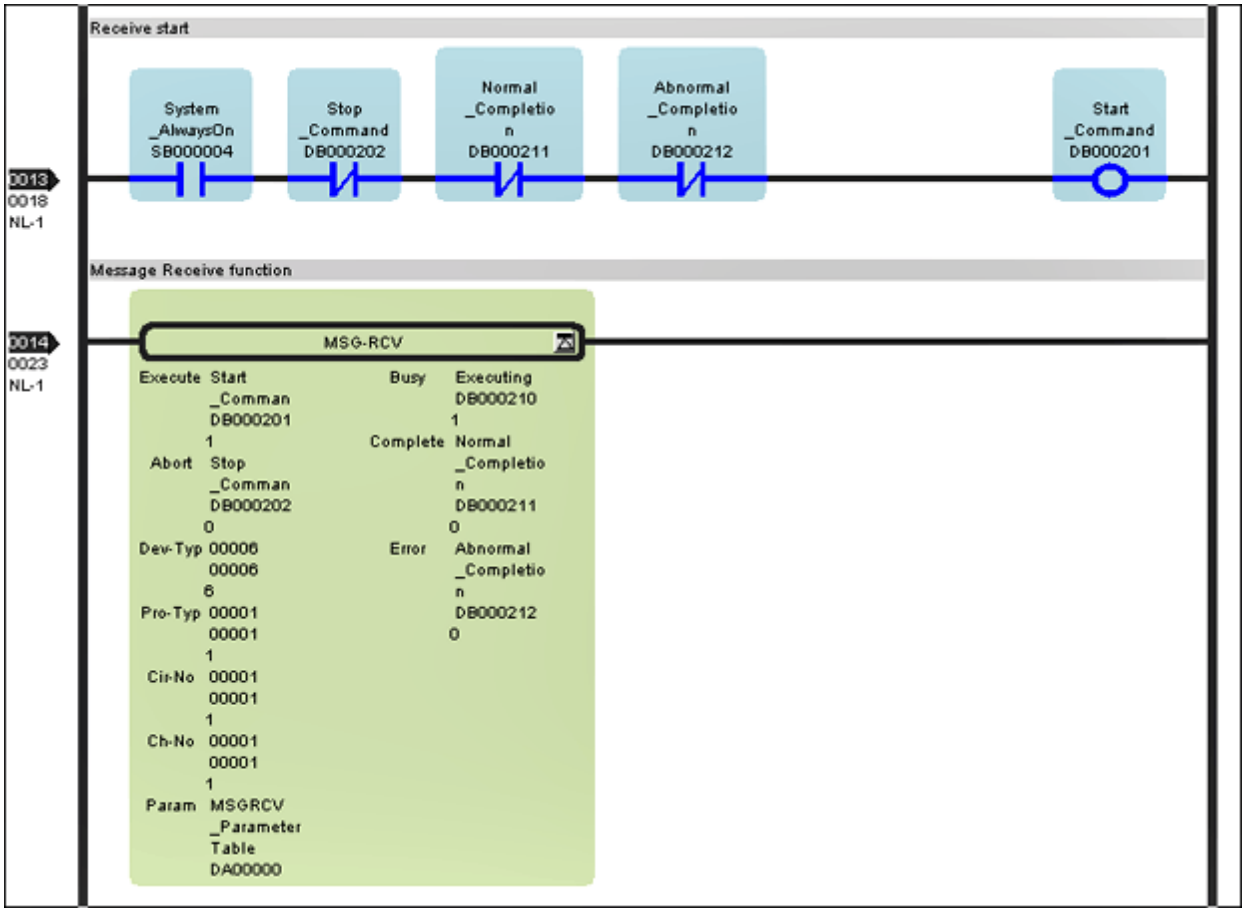
Dest MSGRCV
_Param00
_Out
_Processin
gResult[i]
DW00000i

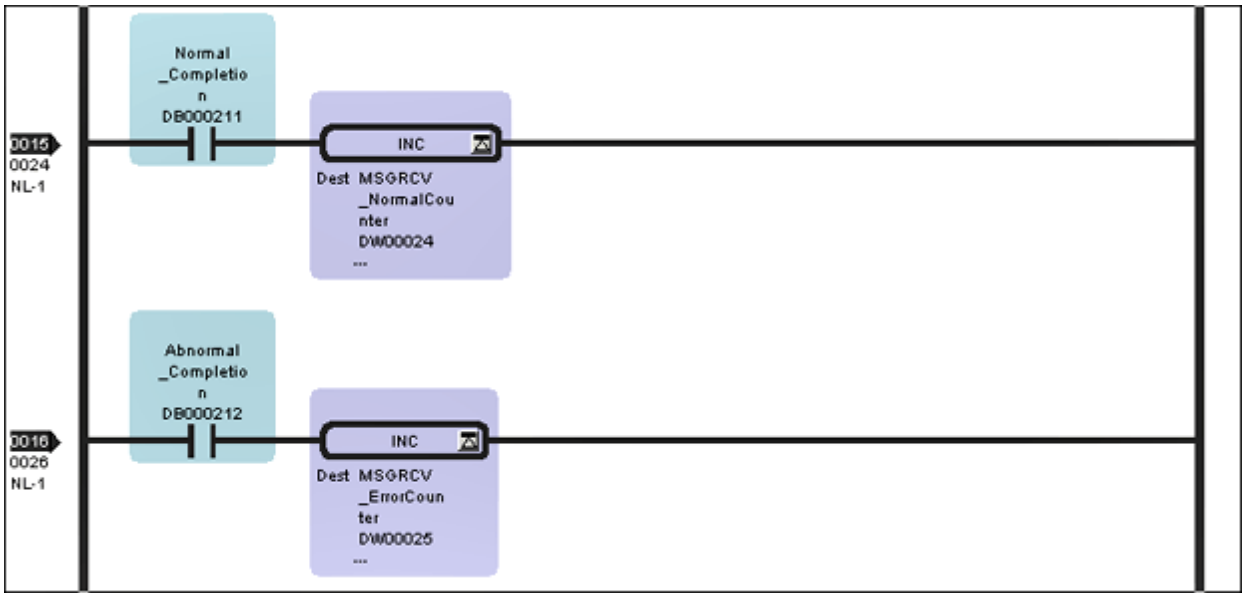
0003
0003
NL-2
END_FOR

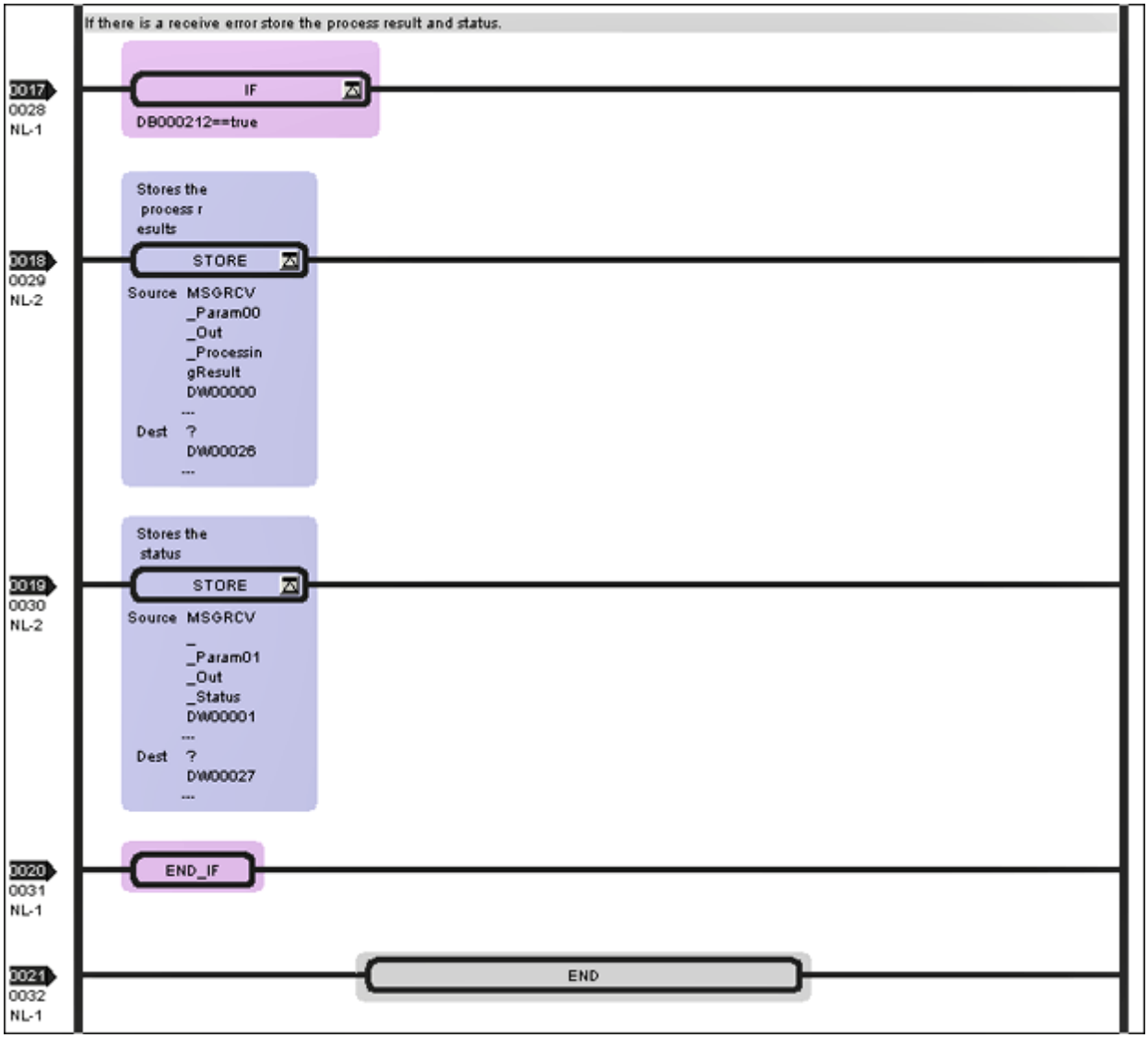

The next few rungs set the values for the Message Receive function parameters.
0004
0004
NL-2
STORE
Source 00001
00001
...
Dest MSGRCV
_Param02
_In
_RemoteStation
Number
DW00002
...
0005
0005
NL-2
STORE
Source 00003
00003
...
Dest MSGRCV
_Param08
_In
_CoilOffset
DW00008
...
0006
0006
NL-2
STORE
Source 00000
00000
...
Dest MSGRCV
_Param09
_In
_InputRelayOffset
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 2 minute interval from when the last communications occurred between the device and the server.

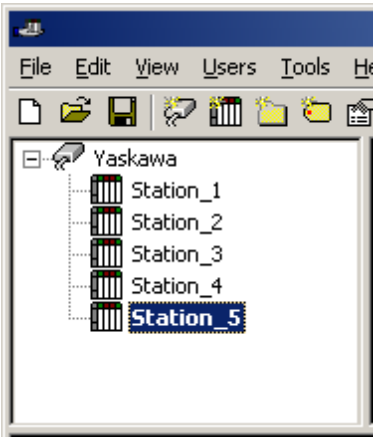
In order to maintain a responsive connection to the Yaskawa MP Series Ethernet module, it is recommended that the **Set** point value of the **On Delay Timer** at rung 12 of ladder L01 should be set greater than the slowest group update rate occurring in the client application by ten or more seconds.

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).

Optimizing Yaskawa MP Series Ethernet Communications

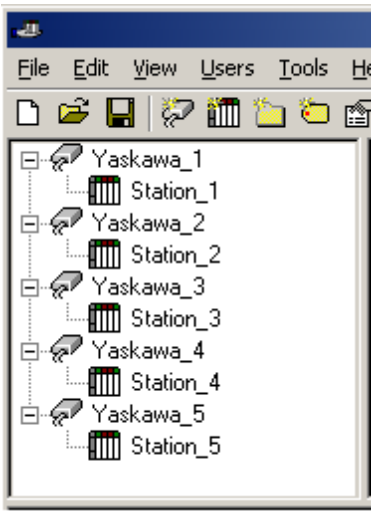
The Yaskawa MP Series Ethernet Driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the Yaskawa MP Series Ethernet Driver is fast, there are a couple of guidelines that can be used in order to control and optimize the application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

Our 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 Yaskawa MP Series Ethernet channel. In this configuration, the driver must move from one device to the next as quickly as possible in order 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 Yaskawa MP Series Ethernet Driver can define up to 16 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 has now been defined under its own channel. In this new configuration, a single path of execution is dedicated to the task of gathering data from each device. If the application has 16 or fewer devices, it can be optimized exactly how it is shown here.

The performance will improve even if the application has more than 16 devices. While 16 or 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*.

Note 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.

Note 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

Note 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.

Note 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.

Note 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>'
Slave device '<device name>' detected a memory parity error
Slave device '<device name>' has failed
Slave 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 Validation

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

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 Status Messages

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

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

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 Specific Messages

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

Device Specific Messages

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 server or slave 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 server or slave 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 server or slave. 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 server or slave. 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 server or slave. 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 server or slave 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 server or slave. 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 server or slave 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.

Slave device '<device name>' detected a memory parity error

Error Type:
Warning

Possible Cause:
The slave attempted to read extended memory, and detected a parity error.

Solution:
The master can retry the request, but service may be required on the slave device.

Slave device '<device name>' has failed

Error Type:
Fatal

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

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

Slave device '<device name>' is busy

Error Type:
Warning

Possible Cause:
The slave is processing a long duration program command.

Solution:
The master should retransmit the message later when the slave 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.

Index

A

B

C

D

E

F

H

I

L

M

O

S

T

U

W