

# Mitsubishi CNC Ethernet Driver

© 2017 PTC Inc. All Rights Reserved.

# Table of Contents

<b>Mitsubishi CNC Ethernet Driver</b> .....	<b>1</b>
<b>Table of Contents</b> .....	<b>2</b>
Mitsubishi CNC Ethernet Driver .....	4
Overview .....	4
<b>Setup</b> .....	<b>5</b>
Channel Properties .....	5
Channel Properties - General .....	5
Channel Properties - Ethernet Communications .....	6
Channel Properties - Write Optimizations .....	6
Channel Properties - Advanced .....	7
Driver Device Properties .....	8
Device Properties - General .....	8
Device Properties - Scan Mode .....	10
Device Properties - Timing .....	11
Device Properties - Auto-Demotion .....	12
Device Properties - Network Parameters .....	13
Device Properties - Redundancy .....	13
Multi-Level Networks .....	15
<b>Optimizing Communications</b> .....	<b>16</b>
<b>Data Types Description</b> .....	<b>17</b>
<b>Address Descriptions</b> .....	<b>18</b>
<b>Event Log Messages</b> .....	<b>21</b>
Failed to read tag on device.   Tag address = '<address>'. .....	21
Write failed for tag on device. Device responded with error code.   Tag address = '<address>', Error code = <code>. .....	21
Block read failed on device. Device responded with error code.   Block size = <number> (points), Block start address = '<address>', Error code = <code>. .....	21
Write failed for tag on device. Framing error.   Tag address = '<address>'. .....	21
Write failed for tag on device. Device responded with an incorrect transaction ID.   Tag address = '<address>'. .....	22
Block read failed on device. Framing error.   Block size = <number> (points), Block start address = '<address>'. .....	22
Block read failed on device. Device responded with an incorrect transaction ID.   Block size = <number> (points), Block start address = '<address>'. .....	22
Write failed for tag on device. Connection error.   Tag address = '<address>'. .....	22
Block read failed on device. Connection error.   Block size = <number> (points), Block start address = '<address>'. .....	22

**Index** ..... **23**

---

## Mitsubishi CNC Ethernet Driver

---

Help version 1.037

### CONTENTS

#### Overview

What is the Mitsubishi CNC Ethernet Driver?

#### Device Setup

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

#### Optimizing Your Ethernet Communications

How do I get the best performance from the Mitsubishi CNC Ethernet Driver?

#### Data Types Description

What data types does this driver support?

#### Address Descriptions

How do I address a data location on a Mitsubishi CNC Ethernet device?

#### Error Descriptions

What error messages does the driver produce?

### Overview

---

The Mitsubishi CNC Ethernet Driver provides a reliable way to connect Mitsubishi CNC Ethernet controllers to OPC Client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications.

## Setup

---

### Supported Devices

C64 CNC Controller

### Communication Protocol

Ethernet with Winsock V1.1 or higher  
TCP/IP

### Supported Communication Parameters

Binary Format only

### Model

Mitsubishi C64 with an AJ71QE71 compatible Ethernet module

### [Channel Properties](#)

### [Device Properties](#)

## Channel Properties

---

This server supports the use of simultaneous multiple communications drivers. Each protocol or driver used in a server project is called a channel. A server project may consist of many channels with the same communications driver or with unique communications drivers. A channel acts as the basic building block of an OPC link.

The properties associated with a channel are broken in to logical groupings. While some groups are specific to a given driver or protocol, the following are the common groups:

### [General](#)

### [Ethernet or Serial Communications](#)

### [Write Optimization](#)

### [Advanced](#)

## Channel Properties - General

---

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

Property Groups	<input type="checkbox"/> <b>Identification</b>	
<b>General</b>	Name	
Write Optimizations	Description	
Advanced	Driver	
	<input type="checkbox"/> <b>Diagnostics</b>	
	Diagnostics Capture	Disable

## Identification

---

**Name:** User-defined identity of this channel. In each server project, each channel name must be unique. Although names can be up to 256 characters, some client applications have a limited display window when browsing the OPC server's tag space. The channel name is part of the OPC browser information.

• For information on reserved characters, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in the server help.

**Description:** User-defined information about this channel.

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

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

• **Note:** With the server's online full-time operation, these properties can be changed at any time. This includes changing the channel name to prevent clients from registering data with the server. If a client has already acquired an item from the server before the channel name is changed, the items are unaffected. If, after the channel name has been changed, the client application releases the item and attempts to re-acquire using the old channel name, the item is not accepted. With this in mind, changes to the properties should not be made once a large client application has been developed. Utilize the User Manager to prevent operators from changing properties and restrict access rights to server features.

## Diagnostics

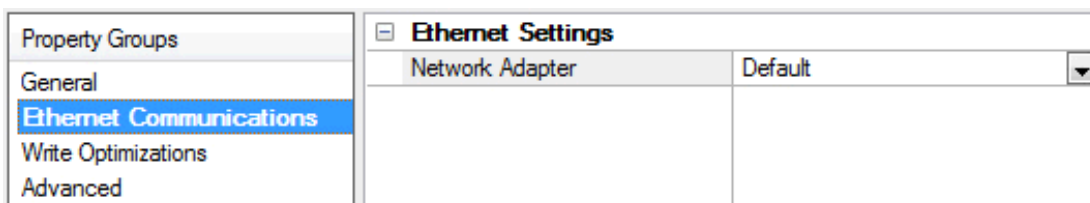
**Diagnostics Capture:** When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default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

• **Note:** This property is disabled if the driver does not support diagnostics.

• For more information, refer to "Communication Diagnostics" 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 Default is selected, the operating system selects the default adapter.

## Channel Properties - Write Optimizations

As with any OPC server, writing data to the device may be the application's most important aspect. The server intends to ensure that the data written from the client application gets to the device on time. Given this goal, the server provides optimization properties that can be used to meet specific needs or improve application responsiveness.

Property Groups	[-] <b>Write Optimizations</b>	
General	Optimization Method	Write Only Latest Value for All Tags
<b>Write Optimizations</b>	Duty Cycle	10

## Write Optimizations

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

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

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

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

## Channel Properties - Advanced

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

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

**Non-Normalized Float Handling:** Non-normalized float handling allows users to specify how a driver handles non-normalized IEEE-754 floating point data. 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. Descriptions of the options are as follows:

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

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

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

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

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

## Driver Device Properties

Device properties are organized into the following groups. Click on a link below for details about the settings in that group.

### Identification

[Operating Mode](#)

[Scan Mode](#)

[Timing](#)

[Auto-Demotion](#)

[32-Bit Data](#)

[Network Parameters](#)

[Redundancy](#)

## Device Properties - General

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



Property Groups	Identification	
General	Name	
Scan Mode	Description	
Auto-Demotion	Channel Assignment	
Redundancy	Driver	
	Model	
	ID Format	Decimal
	ID	2
	Operating Mode	
	Data Collection	Enable
	Simulated	No

## Identification

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

● **Note:** Although descriptive names are generally a good idea, some OPC client applications may have a limited display window when browsing the OPC server's tag space. The device name and channel name become part of the browse tree information as well. Within an OPC client, the combination of channel name and device name would appear as "ChannelName.DeviceName".

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

**Description:** User-defined information about this device.

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

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

**Driver:** Selected protocol driver for this device. This property specifies the driver selected during channel creation. It is disabled in the channel properties.

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

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

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

● **Note:** If the driver is Ethernet-based or supports an unconventional station or node name, the device's TCP/IP address may be used as the device ID. TCP/IP addresses consist of four values that are separated by

periods, with each value in the range of 0 to 255. Some device IDs are string based. There may be additional properties to configure within the ID field, depending on the driver. For more information, refer to the driver's help documentation.

## Operating Mode

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

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

### Notes:

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

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

## Device Properties - Scan Mode

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

Property Groups	<b>Scan Mode</b>	
General	Scan Mode	Respect Client-Specified Scan Rate ▼
<b>Scan Mode</b>	Initial Updates from Cache	Disable

**Scan Mode:** specifies how tags in the device are scanned for updates sent to subscribed 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 maximum scan rate to be used. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Note:** When the server has an active client and items for the device and the scan rate value is increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.

- **Request All Data at Scan Rate:** This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only:** This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the client's responsibility to poll for updates, either by writing to the `_DemandPoll` tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help.*
- **Respect Tag-Specified Scan Rate:** This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

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

## Device Properties - Timing

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

Property Groups	<input type="checkbox"/> <b>Communication Timeouts</b>	
General	Connect Timeout (s)	3
Scan Mode	Request Timeout (ms)	5000
<b>Timing</b>	Retry Attempts	3
Auto-Demotion	<input type="checkbox"/> <b>Timing</b>	
	Inter-Request Delay (ms)	0

### Communications Timeouts

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

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

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

**Retry Attempts:** This property specifies how many times the driver retries 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 retries configured for an application depends largely on the communications environment.

## Timing

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

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

## Device Properties - Auto-Demotion

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

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

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

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

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

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

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

## Device Properties - 32-Bit Data

Property Groups	[-] <b>32-Bit Data</b>	
<b>32-Bit Data</b>	First Word Low	Enable
Network Parameters		

**First Word Low:** Indicate if the first data word is low in the environment (as opposed to high). Two consecutive registers addresses in a Mitsubishi device are used for 32-bit data types. It can be specified whether the driver should assume the first word is the low or the high word of the 32-bit value. The default is Enable First Word Low.

● **Note:** This property can't be changed while there are active references on the device.

## Device Properties - Network Parameters

Property Groups	[-] <b>Network Parameters</b>	
General	Port Number	5001
Scan Mode	Source Network	1
Timing	Source Station	1
Auto-Demotion	Destination Network	1
<b>Network Parameters</b>	Destination Station	2

**Port Number:** This property specifies the UDP port for the destination CNC (or the gateway device if a multi-layered network is configured to receive requests). The default setting is 5001.

● **Note:** It is recommended that the default setting be used because it is always available for programming and monitoring tools.

**Source Network:** This parameter specifies the source network number on which the PC resides. The valid range is 1 to 239. The default setting is 1. This setting is irrelevant if the driver communicates directly to CNC (no gateway device).

**Source Station:** This parameter specifies the station number assigned to the PC. The valid range is 1 to 239. The default setting is 1. All devices on the source network should have unique station numbers. This setting is irrelevant if the driver communicates directly to CNC (no gateway device).

**Destination Network:** This parameter specifies the network number on which the CNC resides. The valid range is 0 to 239. The default setting is 1. This setting is irrelevant if the driver communicates directly to CNC (no gateway device).

**Destination Station:** This parameter specifies the station number assigned to CNC. The valid range is 0 to 239. The default setting is 1. All devices on the destination network should have unique station numbers. This setting is irrelevant if the driver communicates directly to CNC (no gateway device).

## Device Properties - Redundancy

Property Groups	[-] <b>Redundancy</b>	
General	Secondary Path	...
Scan Mode	Operating Mode	Switch On Failure
Timing	Monitor Item	
<b>Redundancy</b>	Monitor Interval (s)	300
	Return to Primary ASAP	Yes

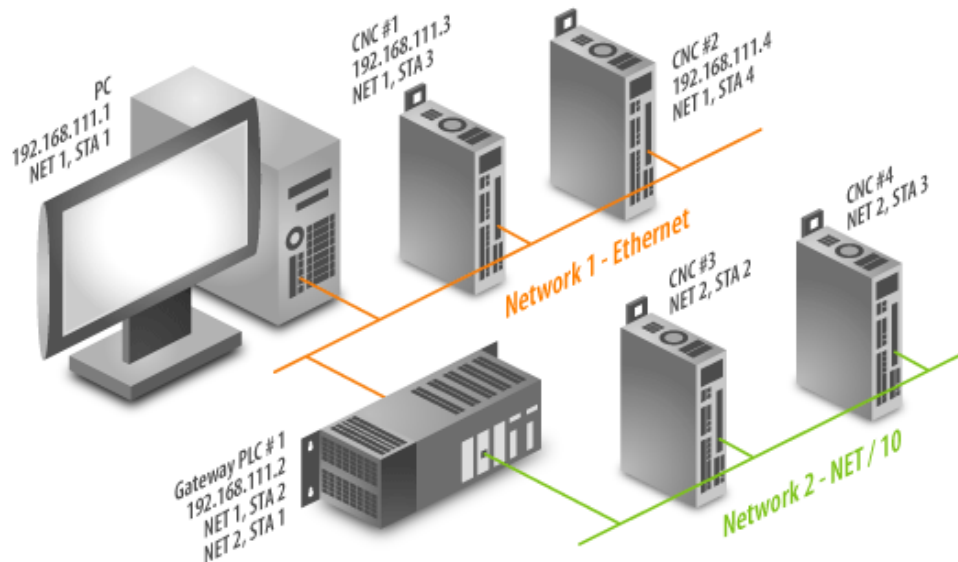
Redundancy is available with the Media-Level Redundancy Plug-In.

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

## Multi-Level Networks

This driver can be used to communicate with devices on remote networks. In the example shown below, CNC 1 and CNC 2 are on the local Ethernet network. CNC 3 and CNC 4 are on Network 2 - NET/10. PLC 1 serves as a relay device connecting the two networks.

● **See Also:** [Device Setup](#)



The gateway has an AJ71QE71 Ethernet module and NET/10 module. CNC 1 and CNC 2 have an AJ71QE71 Ethernet module and NET/10 module. CNC 3 and CNC 2 have a NET/10 module. In this example, four devices are created in the server project using the Device IDs listed in the table below.

CNC	Device ID	SRC NET	SRC STA	DST NET	DST STA	Comment
1	192.168.111.2	1	1	1	3	Direct
2	192.168.111.2	1	1	1	4	Direct
3	192.168.111.2	1	1	2	2	Via PLC 1
4	192.168.111.2	1	1	2	3	Via PLC 1

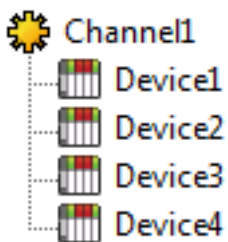
Users can configure the Ethernet card in gateway (Open Method UDP and IP 192.168.111.2). The destination IP (255.255.255.255) and destination port (0xFFFF) can be used to accommodate any IP and Port that may be utilized by the PC.

● **Note:** A relay device may take 5 or more seconds to report a failed read and write to a remote device. It is recommended that the request timeout be set for remote devices accordingly.

## Optimizing Communications

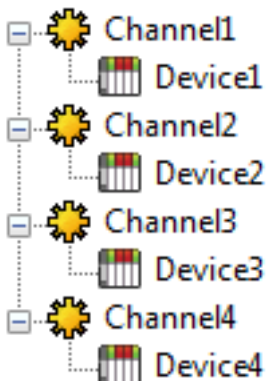
The Mitsubishi CNC Ethernet Driver is designed to provide the best performance with the least amount of impact on the system's overall performance. While the driver is fast, there are a couple of guidelines to control and optimize the application and gain maximum performance.

This server refers to communications protocols like Mitsubishi CNC Ethernet Driver 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 can then be defined under that channel. Each of these devices represents a single Ethernet device from which data is to be collected. While this approach to defining the application provides a high level of performance, it won't take full advantage of the 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 device 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 Mitsubishi CNC 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 256 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 256 or fewer devices, it can be optimized exactly how it is shown here.

The performance improves even if the application has more than 256 devices. While 256 or fewer devices may be ideal, the application still benefits from additional channels. Although by spreading the device load across all 256 channels causes the server to move from device to device again, it can do so with far less devices to process on a single channel.



## Data Types Description

The Mitsubishi CNC Ethernet Driver supports the following data types.

Data Type	Description
Word	Unsigned 16-bit value bit 0 is the low bit bit 15 is the high bit
Short	Signed 16-bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
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
Long Example	If register 40001 is specified as a long, bit 0 of register 40001 would be bit 0 of the 32-bit data type and bit 15 of register 40002 would be bit 31 of the 32-bit data type. The reverse is true when this is not selected.
Float*	32-bit 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 data type and bit 15 of register 40002 would be bit 31 of the 32-bit data type. The reverse is true when this is not selected.

\*The driver interprets two consecutive registers as a single precision value by making the first register the low word and the second register the high word. The reverse is true when this is not selected.

## Address Descriptions

Address specifications vary depending on the model in use. The default data types for dynamically defined tags are shown in **bold**.

Device Type	Range*	Data Type	Access
Inputs	X0000-X0AFF (Hex)	<b>Boolean</b> , Short, Word, Long, DWord	Read/Write
Outputs	Y0000-Y0E7F (Hex)	<b>Boolean</b> , Short, Word, Long, DWord	Read/Write
Link Relays	B0000-B1FFF (Hex)	<b>Boolean</b> , Short, Word, Long, DWord	Read/Write
Special Link Relays	SB0000-SB01FF (Hex)	<b>Boolean</b> , Short, Word, Long, DWord	Read/Write
Internal Relays	M0000-M8191	<b>Boolean</b> , Short, Word, Long, DWord	Read/Write
Special Internal Relays	SM0000-SM0127	<b>Boolean</b> , Short, Word, Long, DWord	Read/Write
Latch Relays	L0000-L0255	<b>Boolean</b> , Short, Word, Long, DWord	Read/Write
Annunciator Relays	F0000-F0127	<b>Boolean</b> , Short, Word, Long, DWord	Read/Write
Timer Contacts	TS0000-TS0255	<b>Boolean</b> , Short, Word, Long, DWord	Read/Write
Timer Coils	TC0000-TC0255	<b>Boolean</b> , Short, Word, Long, DWord	Read/Write
Counter Contacts	CS0000-CS0127	<b>Boolean</b> , Short, Word, Long, DWord	Read/Write
Counter Coils	CC0000-CC0127	<b>Boolean</b> , Short, Word, Long, DWord	Read/Write

\*This device responds to block reads that extend past the memory range if the starting address is within the valid memory range. When this happens, the device returns zeros for all values outside this memory range.

● **Note:** All Boolean device types can be accessed as Short, Word, Long, and DWord. However, the device must be addressed on a 16-bit boundary.

Device Type	Range	Data Type	Access
Timer Value	TN0000-TN0255	<b>Short</b> , Word	Read/Write
Counter Value	CN0000-CN0127	Short, <b>Word</b>	Read/Write
Data Register	D00000-D08191 D00000-D08190	<b>Short</b> , Word Long, DWord, Float	Read/Write
Data Register Bit Access	D00000.00-D08191.15* D00000.00-D08190.31*	<b>Short</b> , Word, Boolean** Long, DWord	Read/Write
Data Register String Access HiLo Byte Ordering	DSH00000.002-DSH08190.002 DSH00000.128-DSH08127.128  The .bit number is used to specify the string length 2-128 bytes. Length must be even.	<b>String</b>	Read/Write
Data Register String Access LoHi Byte Ordering	DSL00000.002-DSL08190.002 DSL00000.128-DSL08127.128  The .bit number is used to specify the string length 2-128 bytes. Length must be even.	<b>String</b>	Read/Write
Special Data Register	SD0000-SD0127 SD0000-SD0126	<b>Short</b> , Word Long,	Read/Write

Device Type	Range	Data Type	Access
		DWord, Float	
Special Data Register Bit Access	SD0000.00-SD0127.15* SD0000.00-SD0126.31*	<b>Short</b> , Word, Boolean** Long, DWord	Read/Write
Link Register	W0000-W1FFF (Hex) W0000-W1FFE (Hex)	<b>Short</b> , Word Long, DWord, Float	Read/Write
Link Register Bit Access	W0000.00-W1FFF.15* W0000.00-W1FFE.31*	<b>Short</b> , Word, Boolean** Long, DWord	Read/Write
Link Register String Access HiLo Byte Ordering	WSH0000.002-WSH1FFE.002 WSH0000.128-WSH1FBF.128  The .bit number is used to specify the string length 2-128 bytes. Length must be even.	<b>String</b>	Read/Write
Link Register String Access LoHi Byte Ordering	WSL0000.002-WSL1FFE.002 WSL0000.128-WSL1FBF.128  The .bit number is used to specify the string length 2-128 bytes. Length must be even.	<b>String</b>	Read/Write
Special Link Register	SW0000-SW01FF (Hex) SW0000-SW01FE (Hex)	<b>Short</b> , Word Long, DWord, Float	Read/Write
Special Link Register Bit Access	SW0000.00-SW01FF.15* SW0000.00-SW01FE.31*	<b>Short</b> , Word, Boolean** Long, DWord	Read/Write
File Register	R00000-R08191 R00000-R08190	<b>Short</b> , Word Long, DWord, Float	Read/Write
File Register Bit Access	R00000.00-R08191.15* R00000.00-R08190.31*	<b>Short</b> , Word, Boolean** Long, DWord	Read/Write
File Register String Access HiLo Byte Ordering	RSH0000.002-RSH08190.002 RSH0000.128-RSH08127.128  The .bit number is used to specify the string length 2-128 bytes. Length must be even.	<b>String</b>	Read/Write
File Register String Access LoHi Byte Ordering	RSL0000.002-RSL08190.002 RSL0000.128-RSL08127.128  The .bit number is used to specify the string length 2-128 bytes. Length must be even.	<b>String</b>	Read/Write
Index Register	Z00-Z13 Z00-Z12	<b>Short</b> , Word Long, DWord, Float	Read/Write
Index Register Bit Access	Z00.00-Z13.15*	<b>Short</b> , Word,	Read/Write

Device Type	Range	Data Type	Access
	Z00.00-Z12.31*	Boolean** Long, DWord	

\*For register memory, the data types Short, Word, DWord, Long, and Boolean may use an optional .bb (dot bit) that can be appended to the address to reference a bit in a particular value. The valid ranges for the optional bit are 0 to 15 for Short, Word, Boolean, and 0 to 31 for Long and DWord. Strings use the bit number to specify length. The valid length of a string in D memory is 2 to 128 bytes. The string length must also be an even number. Float types do not support bit operations. The bit number is always in decimal notation.

\*\*When accessing register memory as Boolean, a bit number is required.

### Array Access

All device types can be accessed in arrays of Short, Word, Long, DWord, or Float format. The size of the array depends on the data type and device type. All Register device types can access up to 254 elements for Short and Word and 127 elements for Long, DWord, and Floats. All Bit memory types can be accessed with up to 125 elements for Short and Word and 62 elements for Long, DWord, and Float. Arrays can either 1 dimension or 2. Regardless of the dimensions, the array size must not exceed the limits already stated. Appending array notation onto a normal device reference enters arrays.

● **Note:** The default for array tags (all device types) is Word.

### Examples

1. D100[4] Single dimension includes the following register addresses: D100, D101, D102, and D103.
2. M016[3][4] Two Dimensions includes the following device addresses as words: M016, M032, M048, M064, M080, M096, M112, M128, M144, M160, M176, M192 3 rows x 4 columns = 12 words 12 x 16 (word) = 192 total bits.

### Additional Device Examples

1. Access X device memory as Word : X??? where the ??? is a hex number on 16-bit boundaries such as 010, 020, 030, and so forth.
2. Access M device memory as Long : M???? where the ??? is a decimal number on 16-bit boundaries such as 0, 16, 32, 48, and so forth.

## Event Log Messages

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

---

### **Failed to read tag on device. | Tag address = '<address>'.**

**Error Type:**

Warning

**Possible Cause:**

The specified address is out of range of device.

**Possible Solution:**

Verify the address range supported by device and modify tag configuration accordingly.

---

### **Write failed for tag on device. Device responded with error code. | Tag address = '<address>', Error code = <code>.**

**Error Type:**

Warning

**Possible Cause:**

The error code should indicate the reason for the error message.

**Possible Solution:**

Consult the documentation on error codes.

---

### **Block read failed on device. Device responded with error code. | Block size = <number> (points), Block start address = '<address>', Error code = <code>.**

**Error Type:**

Warning

**Possible Cause:**

The error code should indicate the reason for the error message.

**Possible Solution:**

Consult the documentation on error codes.

---

### **Write failed for tag on device. Framing error. | Tag address = '<address>'.**

**Error Type:**

Warning

**Possible Cause:**

A packet has been received with incorrect values for fields such as source station number and so forth.

**Write failed for tag on device. Device responded with an incorrect transaction ID. | Tag address = '<address>'.**

---

**Error Type:**

Warning

**Block read failed on device. Framing error. | Block size = <number> (points), Block start address = '<address>'.**

---

**Error Type:**

Warning

**Possible Cause:**

A packet has been received with incorrect values for fields such as source station number and so forth.

**Block read failed on device. Device responded with an incorrect transaction ID. | Block size = <number> (points), Block start address = '<address>'.**

---

**Error Type:**

Warning

**Write failed for tag on device. Connection error. | Tag address = '<address>'.**

---

**Error Type:**

Warning

**Possible Cause:**

This error is due to a Winsock error (such as socket creation failure and so forth).

**Block read failed on device. Connection error. | Block size = <number> (points), Block start address = '<address>'.**

---

**Error Type:**

Warning

**Possible Cause:**

This error is due to a Winsock error (such as socket creation failure and so forth).

# Index

## A

Address Descriptions 18  
Advanced Channel Properties 7

## B

Binary 5  
Block read failed on device. Connection error. | Block size = <number> (points), Block start address = '<address>'. 22  
Block read failed on device. Device responded with an incorrect transaction ID. | Block size = <number> (points), Block start address = '<address>'. 22  
Block read failed on device. Device responded with error code. | Block size = <number> (points), Block start address = '<address>', Error code = <code>. 21  
Block read failed on device. Framing error. | Block size = <number> (points), Block start address = '<address>'. 22

## C

Channel Assignment 9  
Channel Properties 5  
Channel Properties - Ethernet Communications 6  
Channel Properties - General 5  
Channel Properties - Write Optimizations 6  
Communication Protocol 5  
Communications Timeouts 11-12  
Connect Timeout 11

## D

Data Collection 10  
Data Types Description 17  
Demote on Failure 12  
Demotion Period 12  
Description 9  
Destination Network 13  
Destination Station 13

Device Properties 8  
Device Properties - Auto-Demotion 12  
Device Properties - General 8  
Diagnostics 6  
Discard Requests when Demoted 12  
Do Not Scan, Demand Poll Only 11  
Driver 6, 9  
Duty Cycle 7  
DWord 17

## **E**

Event Log Messages 21

## **F**

Failed to read tag on device. | Tag address = '<address>'. 21  
First Word Low 13

## **I**

ID 9  
IEEE-754 floating point 8  
Initial Updates from Cache 11  
Inter-Request Delay 12

## **L**

Long 17

## **M**

Model 5, 9  
Multi-Level Networks 15

## **N**

Name 9



Network Adapter 6  
Network Parameters 13  
Non-Normalized Float Handling 8

## **O**

Optimization Method 7  
Optimizing Communications 16  
Overview 4

## **P**

Port 13

## **R**

Redundancy 13  
Request All Data at Scan Rate 11  
Request Data No Faster than Scan Rate 10  
Request Timeout 11  
Respect Client-Specified Scan Rate 10  
Respect Tag-Specified Scan Rate 11  
Retry Attempts 11

## **S**

Scan Mode 10  
Setup 5  
Short 17  
Simulated 10  
Source Network 13  
Source Station 13  
Supported Devices 5

## **T**

Timeouts to Demote 12

**W**

Word 17

Write All Values for All Tags 7

Write failed for tag on device. Connection error. | Tag address = '<address>'. 22

Write failed for tag on device. Device responded with an incorrect transaction ID. | Tag address = '<address>'. 22

Write failed for tag on device. Device responded with error code. | Tag address = '<address>', Error code = <code>. 21

Write failed for tag on device. Framing error. | Tag address = '<address>'. 21

Write Only Latest Value for All Tags 7

Write Only Latest Value for Non-Boolean Tags 7

Write Optimizations 7