Fanuc Focas Ethernet Driver

Table of Contents

Fanuc Focas Ethernet Driver

Help version 1.062

CONTENTS

Overview

The Fanuc Focas Ethernet Driver provides a reliable way to connect Fanuc Focas Ethernet controllers to OPC Client applications, including HMI, SCADA, Historian, MES, ERP, and countless custom applications. This driver is intended for use with Fanuc Focas Computer Numerical Control (CNC) system.

*For more information on the additional software that is required for use with this driver, refer to **Additional Software Requirements**.*

External Dependencies

This driver has external dependencies. For this driver to communicate with the hardware, the Fanuc CNC Focas 1 / Ethernet Library (part number A02B-0207-K732) or Fanuc Focas 2 Library (part number A02B-0207-K737) must be **installed** on the system. For more information, refer to **Additional Software Requirements**.

Note: The Focas 2 Library combines both Ethernet and HSSB capabilities and can be obtained from the FANUC distributor or by calling 1-888-326-8287. Choose CNC, PARTS, place the order, and request the part number.

Install a Focas Library

This driver requires a Focas library to communicate with the hardware, either FANUC CNC Focas 1 / Ethernet Library (part number A02B-0207-K732) or FANUC Focas 2 Library (part number A02B-0207-K737). Only 32-bit DLL files are compatible with Fanuc Focas Ethernet Driver. Follow these steps to install a library:

1. Obtain a library from the distributor (typically Fwlib* .zip).
2. Move or paste the Fwlib* .zip file to the appropriate Windows system folder:

   On a 64-bit Windows OS, the destination folder is C:\Windows\SysWOW64

   On a 32-bit Windows OS, the destination folder is C:\Windows\System32
3. Once the zip file is in the appropriate destination folder, unzip / extract the contents of the Fwlib* .zip.
4. Verify the 32-bit DLL files are located in the folder; in particular FWLIB32.DLL and FWLIBE1.DLL, which are used by the Fanuc Focas Ethernet Driver.
5. Restart the computer.

See Also: External Dependencies

Additional Software Requirements

Winsock

The host computer must have Winsock version 1.1 or later installed. This is normally done by default when Windows is installed.

Focas 1 / HSSB or Focas 2 Library

This driver requires that either the FANUC CNC Focas 1 / Ethernet Library (part number A02B-0207-K732) or FANUC Focas 2 Library (part number A02B-0207-K737) is installed on the system. Although the library does not need to be **installed** to create a server project, the project will not run without it. This software may be obtained from the FANUC distributor or by calling 1-888-326-8287.

Note: The Focas 2 Library combines both Ethernet and HSSB capabilities.

Unsolicited Message Server

This driver requires that the unsolicited message server application "UMsgServ.exe" (version 1.0.0.1 or later) is installed to use unsolicited messaging. This application is available from the distributor. For more information, refer to the instructions below.

1. To start the installation, copy the executable file to the System folder.
2. Access the command prompt to type "UMsgServ.exe –Install." This causes the message server to automatically launch every time the computer is started.
3. Once the message server is running, its icon is visible in the System Tray.
4. Next, configure the TCP port number on which the message server listens (in addition to the message timeout and the maximum number of CNCs). To do so, right-click the icon and then select **Setting**. The default settings are as follows:

- **TCP Port Number:** 8196
- **Message Timeout:** 30 seconds
- **Maximum Number of CNCs:** 32

5. To uninstall the message server, use the command prompt to type "UMsgServ.exe –Remove" and then delete the executable file.

CNC Control Software

The table below displays the software that must be installed on the controller to use its unsolicited messaging capabilities.

Refer to Fanuc CNC documentation for other model compatibility.

Model	Software
16i	B0F4/K1(M) or later B0H1/P5(M) or later
18i	DDF4/K1(M) or later BDH1/P5(M) or later
21i	BDF4/K1(M) or later DDH1/P5(M) or later

Note: Set the CNC parameter 904, bit 4 to 1 for unsolicited messaging by using the controller's programming software.

Fast Ethernet Firmware

Firmware 6567/E2 or later (registered onto F-ROM of CNC) must be installed on the controller to use its unsolicited messaging capabilities.

Ladder

A ladder program must be created that constructs and controls the transmission of unsolicited messages to use the controllers unsolicited messaging capabilities. For more information, refer to **Unsolicited Messaging**.

Setup

Supported Devices

The Fanuc Focas Ethernet Driver can communicate to controllers that are compatible with the Focas 1 or Focas 2 CNC / PMC data window control libraries. This includes, but is not limited to, the following:

Series 0i
Series 15
Series 15i
Series 16
Series 16i
Series 18
Series 18i
Series 21
Series 21i
Series 30i
Series 31i
Series 32i
Power Mate i
Open Addressing

Channel and Device Limits

The maximum number of channels supported by this driver is 256. The maximum number of devices supported by this driver is 20 per channel. Each device on the channel must be uniquely identified by its own IP address.

Connect Timeout

Specify the amount of time that the driver waits for a connection to be made with a device. The connection time depends on the network load and may vary with each connection attempt. The valid range is 1 to 60 seconds. The default setting is 3 seconds.

Request Timeout

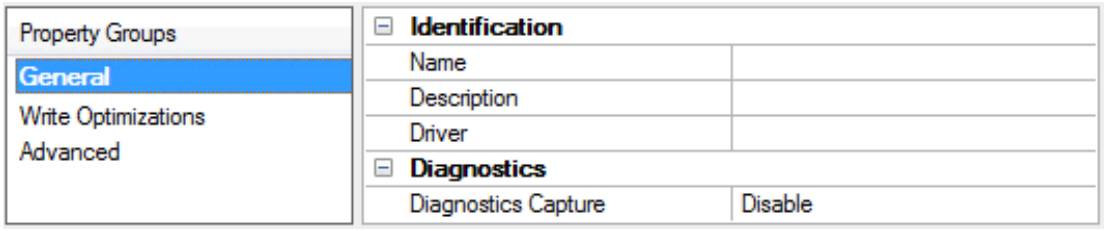
Specify the amount of time that the driver waits on a response from the device before giving up and going on to the next request. Longer timeouts only affect performance when a device is not responding. The valid range is 100 to 9999 milliseconds. The default setting is 1000 milliseconds.

Retry Attempts

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

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.



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. 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: 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. 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. 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

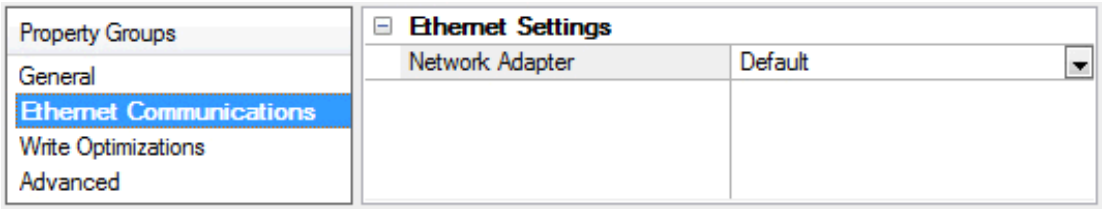
Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default is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

Note: This property is 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.

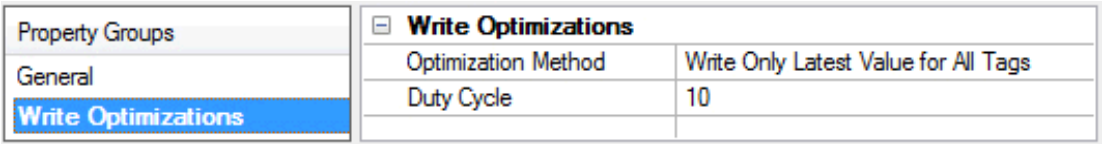


Ethernet Settings

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

As with any 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.



Write Optimizations

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

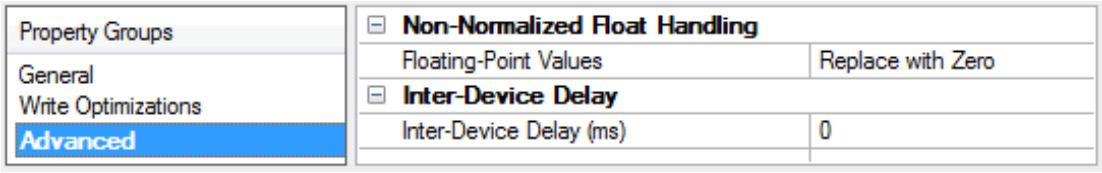
- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing 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.

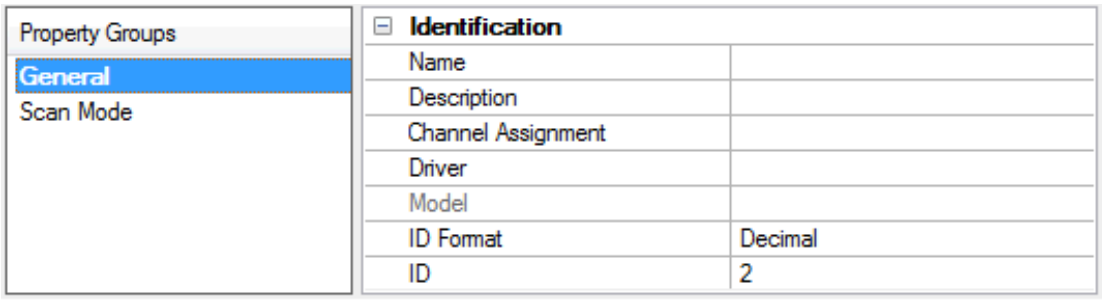
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

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: 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.

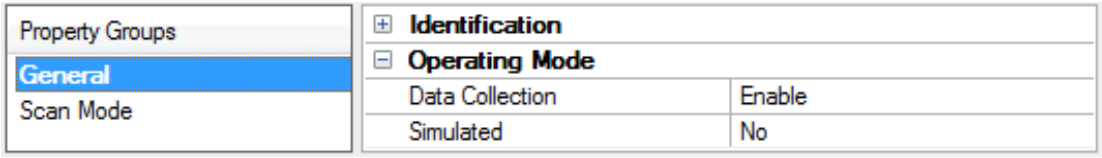
Driver: Selected protocol driver for this device.

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 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: 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.

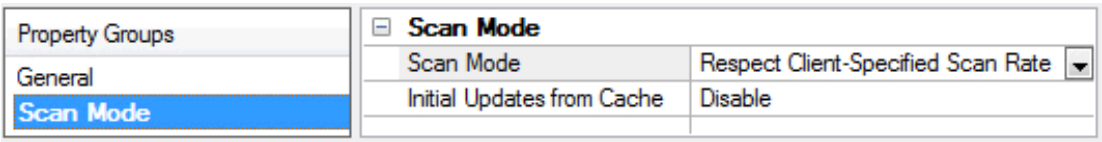
Notes:

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.



Scan Mode: Specifies 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

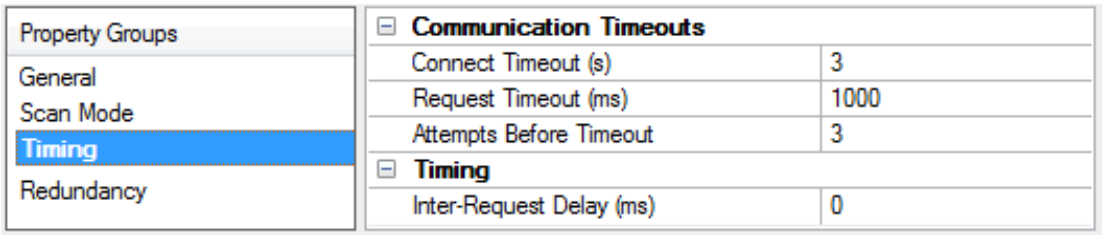
increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.

- **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 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: 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.

Attempts Before Timeout: This property specifies 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.

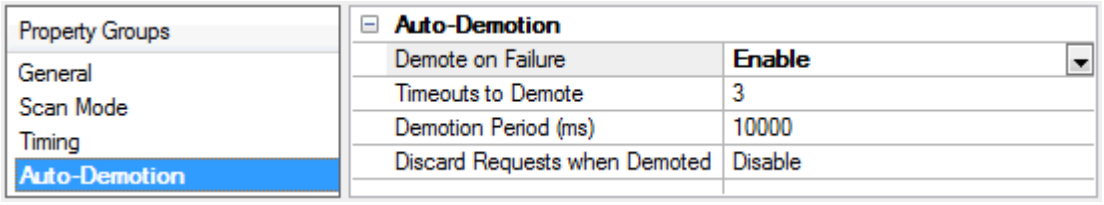
Timing

Inter-Request Delay: 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.



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.

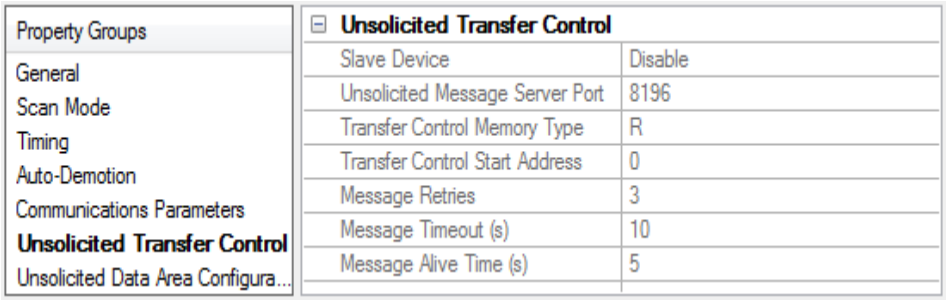
Discard Requests when Demoted: Select whether or not write requests should be attempted during the off-scan period. Disable to always send write requests regardless of the demotion period. Enable to discard writes; the server automatically fails any write request received from a client and does not post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

Device Properties — Communications Parameters

TCP/IP Port: Specify the TCP/IP port number that the remote device is configured to use. The default setting is 8193.

Maximum Request Size: Specify the number of bytes that may be requested from a device at one time. To refine the driver's performance, configure the request size to one of the following settings: 8, 16, 32, 64, 128, 256, or 512 bytes. The default setting is 256 bytes.

Device Properties — Unsolicited Transfer Control



Slave Device: This option should be enabled if the device receives unsolicited data from the CNC. All tags belonging to a slave device read data cached in the driver, not directly from the CNC. The slave device's tags displays a value of zero until it receives its first unsolicited data update. When disabled, all tags belonging to the device read and write directly to the CNC. The default setting is disabled.

Unsolicited Message Server Port: Specify the port that the unsolicited message server application has been configured to use. The default setting is 8196.

Transfer Control Memory Type: Specify the registers' PMC memory type for unsolicited message transfer control. Options include R (internal relay) and E (extended relay). The default setting is R.

Transfer Control Start Address: Specify the start address of the registers used for unsolicited message transfer control. The valid range is 0 to 7999, although the actual range of valid addresses depends on the hardware. The default setting is 0.

Message Retries: Specify the number of times that the CNC should retry sending unsolicited messages. The valid range is 0 to 10. The default setting is 3.

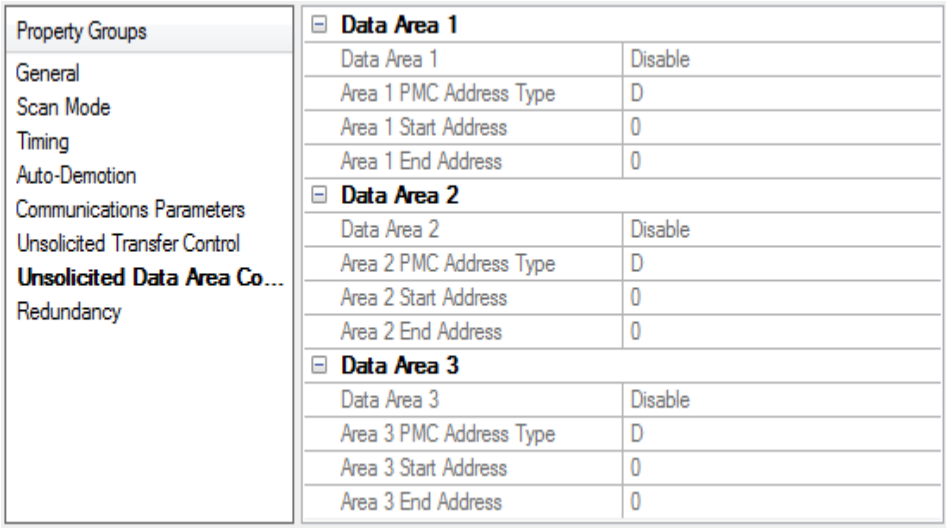
Message Timeout: Specify the unsolicited message timeout, which is the amount of time that the CNC waits for the driver to respond to an unsolicited message. The valid range is 0 to 30. The default setting is 10 seconds.

Message Alive Time: Specify the unsolicited message alive time, which is the amount of time that the CNC retains an unsolicited message for the driver to read. This setting must be less than the message timeout. The valid range is 0 to 30. The default setting is 5 seconds.

See Also: Unsolicited Messaging

Device Properties — Unsolicited Data Areas

Users can configure up to three areas in PMC memory for unsolicited messaging. These areas' data content is sent to the driver in each unsolicited message. As such, these areas should be made as small as possible.



Data Area *n*

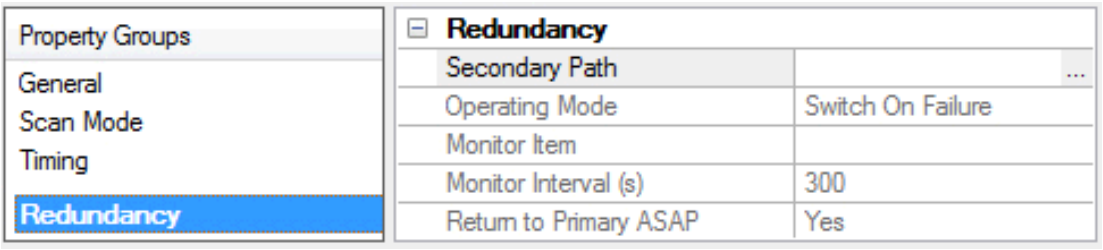
Note: All tags that are created for a slave device are validated according to the configured data areas. For example, if a slave device is configured with a single area with D1000 to D1100, then a tag with address D1000 would be valid, but tags with addresses D1101 or C0001 would be invalid. All tags belonging to a slave device are Read Only.

- **Data Area *n***: When enabled, this data area is in use.
- **PMC Address Type:** Specify the area's PMC address type. The default setting is D. Supported types include the following:
    - **G:** Signal to PMC->CNC
    - **F:** Signal to CNC->PMC
    - **Y:** Signal to PMC->machine
    - **X:** Signal to machine->PMC
    - **A:** Message demand
    - **R:** Internal relay
    - **T:** Changeable timer
    - **K:** Keep relay
    - **C:** Counter
    - **D:** Data table

- **Start Address:** Specify an area's start address. The valid range is 0 and 7999, although the actual range of valid addresses depends on the hardware. The default setting is 0.
- **End Address:** Specify an area's end address. The valid range is 0 and 7999, although the actual range of valid addresses depends on the hardware. The total number of bytes must not exceed 1430, 1414, or 1398 for areas 1, 2, or 3 respectively. The default value is 0.

See Also: Unsolicited Messaging

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.

Unsolicited Messaging

Before configuring a system for unsolicited messaging, it is important that users understand how the various hardware and software components work together to transfer data. These components include one or more CNC controllers equipped with Fast Ethernet communications boards, firmware that supports unsolicited messaging, and a ladder program. To receive unsolicited data, a host computer must be equipped with the OPC server, its Fanuc Focas Ethernet Driver , the Focas 1 Data Window Library software, and the Unsolicited Message Server. Data can be read from the OPC server with OPC or DDE client applications running on the host or remote computer. For more information, refer to Additional Software Requirements.

During an unsolicited messaging session, the controller is only in direct communication with the Unsolicited Message Server. The message server notifies the driver when the controller makes a request to send unsolicited data. The Focas 1 Data Window Library allows the driver to receive the unsolicited data via the message server, and also enables direct communications with the controllers for starting and ending unsolicited messaging sessions.

Note: If unsolicited messaging is not used, the driver uses the library to issue read and write requests directly to the controller. It is possible to simultaneously use both types of communication with a controller by creating a slave and a non-slave device in the OPC server project.

Unsolicited Data Transmission

Ladder programs coordinate the transfer of unsolicited data from their respective controllers, and must be tailored to each application. For more information on unsolicited data transmission, refer to the instructions below.

1. To start, the ladder program places the message contents in a designated area of the PMC memory. Once the message is prepared, the ladder controls the data transmission by setting and monitoring bits in the Unsolicited Transfer Control area of PMC memory.

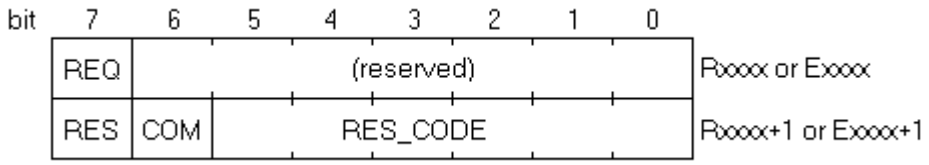
2. To trigger the unsolicited message transmission, the ladder sets the "REQ" (request to send) transfer control bit. The controller sends an "Unsolicited data ready" notification to the message server immediately after.
3. The message server relays the notification to the driver, which then responds by issuing a "Read unsolicited message" command to the message server.
4. The driver receives the unsolicited message data in response to this command, and then replies to the data ready notification with a response code indicating success or failure. The message server then passes this response code to the controller.
5. When the controller receives the response code, it copies it to the "RES_CODE" memory area. The controller sets the "RES" (response ready) bit to indicate that the transaction completed immediately after.

   Note: The ladder program must be designed to detect when the RES bit has been set. Once it detects that the RES bit has been set, it can read the response code and react as needed. If the data fails to reach the driver, the controller will place its own response code that describes the problem in RES_CODE and set the RES bit.
6. Once the ladder has read the response code, it must clear the REQ bit to ready the system for the next message.

PMC Memory

The PMC memory area included in unsolicited messages is defined with the Fanuc Focas Ethernet Driver . The driver transfers these area properties to the controller at the start of an unsolicited messaging session. Data from the range of all areas is included each time an unsolicited message is sent. As such, these areas should be made as small as possible. The ladder program must be written to use the exact address ranges specified in the driver configuration.

Unsolicited data transmission is coordinated between the ladder program and communications board via 2 bytes of PMC R or E memory. This transfer control area is defined with the Fanuc Focas Ethernet Driver . The transfer control area properties are then sent to the controller along with the data area properties when an unsolicited messaging session starts. The ladder program must be written to use the specific addresses specified in the driver configuration. The various bits in the transfer control area, with starting address xxxx, have the following locations and meanings:



REQ

Rxxxx.7 (or *Exxxx.7*)

After the ladder program constructs a message, it must set this bit to 1. This signals the communications board to issue a notification that a new message is ready to be read.

COM

Rxxxx+1.6 (or *Exxxx+1.6*)

The communications board sets this bit to 1 when message transmission begins. The communications board sets this bit back to 0 immediately before it sets RES to 1, and places the response code in RES_CODE.

RES

Rxxxx+1.7 (or *Exxxx+1.7*)

The communications board sets this bit to 1 immediately after message transmission completes. When the ladder program detects that this bit is set to 1, it can read the response code from RES_CODE. The ladder then acts depending on the value of RES_CODE. Once this is done, the ladder must set REQ back to 0. This causes the communications board to clear RES_CODE and set the bit back to 0. Then the communications board is ready to perform the next unsolicited transaction.

RES_CODE

Rxxxx+1.0 to Rxxxx+1.5 (or *Exxxx+1.0 to Exxxx+1.5*)

The result of the unsolicited transaction is placed here. It can be a code passed down from the driver, or a code set by the communications board if there was a communications failure. The possible values are displayed in the table below.

RES_CODE	Meaning
0x00	Success. The communications board did not detect any failures in communication, and the driver reported that it processed the received data successfully.
0x01	The transmission control properties are invalid or the unsolicited messaging session was not started.
0x02	The unsolicited message server is not running.
0x03	The CNC failed to transmit message.
0x04	The CNC failed to receive response.
0x05	The transmission retry count was exceeded.
0x06	The CNC failed to construct the message data.
0x07	The CNC received an invalid packet.
0x08	The CNC accepted termination of unsolicited messaging session.
0x10	The driver experienced a Focas 1 Library error while reading message.
0x11	The driver found the message data to be invalid or experienced other problems while processing message data.
0x21	The PC application to receive the message does not exist, though the message was received by the PC. Either the OPC server or this driver is not running.
0x22	The PC application to receive the message was not recognized by the unsolicited message server, though the message was received by the PC. The unsolicited message server may need be restarted, or there may have been a problem when the driver was starting the unsolicited messaging session.
0x23	The CNC failed in writing the received message to the PC.
0x24	The timeout period and retry count have expired.
0x25	Illegal data was included in the received message.

The Fanuc Focas Ethernet Driver stores unsolicited data in a memory cache. The OPC server makes this data visible to client applications via tags, whose addresses must be the same as the controller's data source. For example, if an unsolicited data area is configured to include the byte at D1000, then a tag with an address of D1000 must be used to view the data. These tags show the last values sent to the driver, which may not necessarily be the current values in the controller. To poll current values directly from the con-

troller, users would need additional tags belonging to a non-slave device. Slave device tags display a value of zero until the driver receives its first unsolicited data update from the controller.

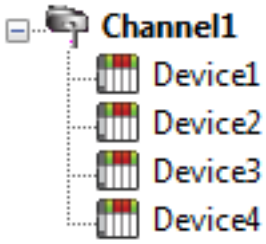
Notes:

1. In addition to data ready notifications, the Unsolicited Message Server also apprises the driver of other important events. These include CNC power up, CNC power down, Unsolicited Message Server shutdown, and communications error notifications. The driver responds to each of these events in such a way that communication with the hardware is maintained, if possible.
2. The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restarts the session after detecting a communications problem. Tags that belong to a slave device in an error state will continue to display the last value received from the device or the initial value of zero.

Optimizing Communications

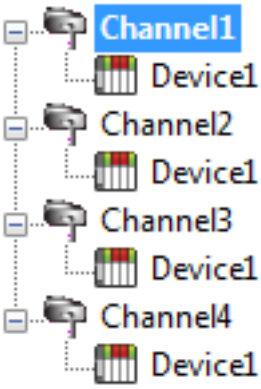
The Fanuc Focas Ethernet Driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the driver is fast, there are a couple of guidelines that can be used to control and optimize the application and gain maximum performance.

This server refers to communications protocols like Fanuc Focas 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 Fanuc Focas controller from which data iscollected. While this approach to defining the application provides a high level of performance, it won't take full advantage of the Fanuc Focas 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 Fanuc Focas 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 fewer devices, it can be optimized exactly how it is shown here.

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

Request Size can also affect the Fanuc Focas Ethernet Driver performance. The request size refers to the number of bytes that may be requested from a device at one time, and is available on every defined device. To refine the driver's performance, configure the request size to one of the following settings: 8, 16, 32, 64, 128, 256, or 512 bytes. Depending on the model of the device being used, the setting chosen for request size can dramatically affect the application. The default value of 256 bytes is recommended. If the application consists of large requests for consecutively ordered data, try increasing the request size setting for the device. For more information, refer to **Setup**.

Data Types Description

Data Type	Description
Boolean	Single bit
Byte	Unsigned 8-bit value bit 0 is the low bit bit 7 is the high bit
Word	Unsigned 16-bit value bit 0 is the low bit bit 15 is the high bit
Short	Signed 16-bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
DWord	Unsigned 32-bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
Float	32-bit floating point value

Data Type	Description
String	Null terminated ASCII string

Address Descriptions

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

Note: If the model of interest is listed as supported but is not selectable, use the Open model.

Series 15i
Series 16i
Series 18i
Series 21i
Power Mate i
Open

Series 15i

The following addresses are supported for this model. Not all address ranges may be valid for the particular device being used. For more information, refer to the specific device's documentation. To jump to a specific section, select a link from the list below.

CNC Data
Arrays
Strings

PMC Data

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

Address Type	Range	Data Type	Access
A (Message demand)	A00000-A00124 A00000-A00123 A00000-A00121 Axxxxx.0-Axxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
C (Counter)	C00000-C00199 C00000-C00198 C00000-C00196 Cxxxxx.0-Cxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
D (Data table)	D00000-D09999 D00000-D09998 D00000-D09996 Dxxxxx.0-Dxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
F (Signal to CNC->PMC)	F00000-F00511 F00000-F00510 F00000-F00508 Fxxxxx.0-Fxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
G (Signal to PMC->CNC)	G00000-G00511	**Byte**, Char	Read/Write

Address Type	Range	Data Type	Access
	G00000-G00510 G00000-G00508 Gxxxxx.0-Gxxxxx.7	Word, Short DWord, Long, Float Boolean	
K (Keep relay)	K00000-K00909 K00000-K00908 K00000-K00906 Kxxxxx.0-Kxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
R (Internal relay)	R00000-R09199 R00000-R09198 R00000-R09196 Rxxxxx.0-Rxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
T (Changeable timer)	T00000-T00299 T00000-T00298 T00000-T00296 Txxxxx.0-Txxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
X (Signal to machine->PMC)	X00000-X00127 X00000-X00126 X00000-X00124 Xxxxxx.0-Xxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
Y (Signal to PMC->machine)	Y00000-Y00127 Y00000-Y00126 Y00000-Y00124 Yxxxxx.0-Yxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
Custom Macro Value (common range)	#0100-#0999	**Float**, Double	Read/Write
Custom Macro Value (local range)	#0001-#0033	**Float**, Double	Read Only
Custom Macro Value (system range)	#1000-#9999	**Float**, Double	Read/Write

CNC Data

Status Info Tags
Tool Offset
Workpiece Zero Offset
Parameter Read Values
Alarms Values
Diagnostic Values
Path Value
Read Axis Data Values
Program Value
Read Dynamic2 Data Values

Arrays

Arrays are supported for all PMC addresses, except for Custom Macros in the system range and where Boolean or string data types are used. Tool Offset data cannot be addressed as an array. The syntax for declaring an array is as follows:

Mxxxx[cols] with assumed row count of 1.

Mxxxxx[rows][cols] where M is the address type and *xxxxx* is the byte offset of the first element in the array.

Note: For all arrays, the total number of bytes being requested cannot exceed the specified request size.

Strings

All address types can be read and written to as ASCII strings. Each byte of memory contains one ASCII character. The length of strings can range from 1 to 120 and is entered in place of the bit number. An additional character "M" is appended to the address to distinguish string addresses from bit addresses.

Example
To address a string of length 100 characters, starting at D00200, enter: D00200.100 M.

Note: Use caution when modifying Word, Short, DWord, Long, and Float types. Since all addresses start at a byte offset within the device, it is possible for the memory associated with tags to overlap. For example, word tags D00000 and D00001 overlap at byte 1. Writing to D00000 also modifies the value held in D00001. It is recommended that these memory types be used such that each value to be read and written to by the driver occupy a unique range of memory in the device. For example, map 3 Word values to bytes D00000-D00001, D00002-D00003, and D00004-D00005. Tags to access these values would then have addresses D00000, D00002, and D00004 respectively, and a data type of Word.

Series 16i

The following addresses are supported for this model. Not all address ranges may be valid for the particular device being used. For more information, refer to the specific device's documentation. To jump to a specific section, select a link from the list below.

PMC Data

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

Address Type	Range	Data Type	Access
A (Message demand)	A00000-A00124 A00000-A00123 A00000-A00121 Axxxxx.0-Axxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
C (Counter)	C00000-C00199 C00000-C00198 C00000-C00196 Cxxxxx.0-Cxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
D (Data table)	D00000-D09999 D00000-D09998 D00000-D09996 Dxxxxx.0-Dxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
E (Extended relay)	E00000-E07999 E00000-E07998	**Byte**, Char Word, Short	Read/Write

Address Type	Range	Data Type	Access
	E00000-E07996 Exxxxx.0-Exxxxx.7	DWord, Long, Float Boolean	
F (Signal to CNC->PMC)	F00000-F02511 F00000-F02510 F00000-F02508 Fxxxxx.0-Fxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
G (Signal to PMC->CNC)	G00000-G02511 G00000-G02510 G00000-G02508 Gxxxxx.0-Gxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
K (Keep relay)	K00000-K00909 K00000-K00908 K00000-K00906 Kxxxxx.0-Kxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
M (Input signal from other devices)	M00000-M00511 M00000-M00510 M00000-M00508 Mxxxxx.0-Mxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
N (Output signal from other devices)	N00000-N00511 N00000-N00510 N00000-N00508 Nxxxxx.0-Nxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
R (Internal relay)	R00000-R09119 R00000-R09118 R00000-R09116 Rxxxxx.0-Rxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
T (Changeable timer)	T00000-T00299 T00000-T00298 T00000-T00296 Txxxxx.0-Txxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
X (Signal to machine->PMC)	X00000-X00127 X00000-X00126 X00000-X00124 Xxxxxx.0-Xxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
Y (Signal to PMC->machine)	Y00000-Y00127 Y00000-Y00126 Y00000-Y00124 Yxxxxx.0-Yxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
Custom Macro Value (common range)	#0100-#0999	**Float**, Double	Read/Write
Custom Macro Value (local range)	#0001-#0033	**Float**, Double	Read Only
Custom Macro Value (system range)	#1000-#9999	**Float**, Double	Read/Write

CNC Data

Status Info Tags

Tool Offset

Workpiece Zero Offset

Parameter Read Values
Alarms Values
Diagnostic Values
Path Value
Read Axis Data Values
Program Value
Read Dynamic2 Data Values

Arrays

Arrays are supported for all PMC addresses, except for Custom Macros in the system range and where Boolean or string data types are used. Tool Offset data cannot be addressed as an array. The syntax for declaring an array is as follows:

Mxxxxx[cols] with assumed row count of 1.
Mxxxxx[rows][cols] where M is the address type and *xxxxx* is the byte offset of the first element in the array.

Note: For all arrays, the total number of bytes being requested cannot exceed the specified request size.

Strings

All address types can be read and written to as ASCII strings. Each byte of memory contains one ASCII character. The length of strings can range from 1 to 120 and is entered in place of the bit number. An additional character "M" is appended to the address to distinguish string addresses from bit addresses.

Example
To address a string of length 100 characters, starting at D00200, enter D00200.100 M.

Unsolicited Data

If tags belong to a slave device, then their address types and ranges are validated according to the data areas configured for that device) For example, if the slave device is configured with a single area with D01000 to D01100, then a tag with address D01000 would be valid, but tags with addresses D01101 or C00001 would be invalid. All tags belonging to a slave device are Read Only.

See Also: Unsolicited Data Areas

Note: Use caution when modifying Word, Short, DWord, Long, and Float types. Since all addresses start at a byte offset within the device, it is possible for the memory associated with tags to overlap. For example, word tags D00000 and D00001 overlap at byte 1. Writing to D00000 also modifies the value held in D00001. It is recommended that these memory types be used such that each value to be read and written to by the driver occupy a unique range of memory in the device. For example, map 3 Word values to bytes D00000-D00001, D00002-D00003, and D00004-D00005. Tags to access these values would then have addresses D00000, D00002, and D00004 respectively, and a data type of Word.

Series 18i

The following addresses are supported for this model. Not all address ranges may be valid for the particular device being used. For more information, refer to the specific device's documentation. To jump to a specific section, select a link from the list below.

CNC Data
Arrays

Strings
Unsolicited Data

PMC Data

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

Address Type	Range	Data Type	Access
A (Message demand)	A00000-A00124 A00000-A00123 A00000-A00121 Axxxxx.0-Axxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
C (Counter)	C00000-C00199 C00000-C00198 C00000-C00196 Cxxxxx.0-Cxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
D (Data table)	D00000-D09999 D00000-D09998 D00000-D09996 Dxxxxx.0-Dxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
E (Extended relay)	E00000-E07999 E00000-E07998 E00000-E07996 Exxxxx.0-Exxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
F (Signal to CNC->PMC)	F00000-F02511 F00000-F02510 F00000-F02508 Fxxxxx.0-Fxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
G (Signal to PMC->CNC)	G00000-G02511 G00000-G02510 G00000-G02508 Gxxxxx.0-Gxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
K (Keep relay)	K00000-K00909 K00000-K00908 K00000-K00906 Kxxxxx.0-Kxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
M (Input signal from other devices)	M00000-M00511 M00000-M00510 M00000-M00508 Mxxxxx.0-Mxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
N (Output signal from other devices)	N00000-N00511 N00000-N00510 N00000-N00508 Nxxxxx.0-Nxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
R (Internal relay)	R00000-R09119 R00000-R09118 R00000-R09116 Rxxxxx.0-Rxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write

Address Type	Range	Data Type	Access
T (Changeable timer)	T00000-T00299 T00000-T00298 T00000-T00296 Txxxxx.0-Txxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
X (Signal to machine->PMC)	X00000-X00127 X00000-X00126 X00000-X00124 Xxxxxx.0-Xxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
Y (Signal to PMC->machine)	Y00000-Y00127 Y00000-Y00126 Y00000-Y00124 Yxxxxx.0-Yxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
Custom Macro Value (common range)	#0100-#0999	**Float**, Double	Read/Write
Custom Macro Value (local range)	#0001-#0033	**Float**, Double	Read Only
Custom Macro Value (system range)	#1000-#9999	**Float**, Double	Read/Write

CNC Data

Status Info Tags
Tool Offset
Workpiece Zero Offset
Parameter Read Values
Alarms Values
Diagnostic Values
Path Value
Read Axis Data Values
Program Value
Read Dynamic2 Data Values

Arrays

Arrays are supported for all PMC addresses, except for Custom Macros in the system range and where Boolean or string data types are used. Tool Offset data cannot be addressed as an array. The syntax for declaring an array is as follows:

Mxxxxx[cols] with assumed row count of 1.
Mxxxxx[rows][cols] where M is the address type and *xxxxx* is the byte offset of the first element in the array.

Note: For all arrays, the total number of bytes being requested cannot exceed the specified request size.

Strings

All address types can be read and written to as ASCII strings. Each byte of memory contains one ASCII character. The length of strings can range from 1 to 120 and is entered in place of the bit number. An additional character "M" is appended to the address to distinguish string addresses from bit addresses.

Example
To address a string of length 100 characters, starting at D00200, enter D00200.100 M.

Unsolicited Data

If tags belong to a slave device, then their address types and ranges are validated according to the data areas configured for that device. For example, if the slave device is configured with a single area with D01000 to D01100, then a tag with address D01000 would be valid, but tags with addresses D01101 or C00001 would be invalid. All tags belonging to a slave device are Read Only.

See Also: Unsolicited Data Areas

Note: Use caution when modifying Word, Short, DWord, Long, and Float types. Since all addresses start at a byte offset within the device, it is possible for the memory associated with tags to overlap. For example, word tags D00000 and D00001 overlap at byte 1. Writing to D00000 also modifies the value held in D00001. It is recommended that these memory types be used such that each value to be read and written to by the driver occupy a unique range of memory in the device. For example, map 3 Word values to bytes D00000-D00001, D00002-D00003, and D00004-D00005. Tags to access these values would then have addresses D00000, D00002, and D00004 respectively, and a data type of Word.

Series 21i

The following addresses are supported for this model. Not all address ranges may be valid for the particular device being used. For more information, refer to the specific device's documentation. To jump to a specific section, select a link from the list below.

PMC Data

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

Address Type	Range	Data Type	Access
A (Message demand)	A00000-A00124 A00000-A00123 A00000-A00121 Axxxxx.0-Axxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
C (Counter)	C00000-C00199 C00000-C00198 C00000-C00196 Cxxxxx.0-Cxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
D (Data table)	D00000-D09999 D00000-D09998 D00000-D09996 Dxxxxx.0-Dxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
E (Extended relay)	E00000-E07999 E00000-E07998 E00000-E07996 Exxxxx.0-Exxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
F (Signal to CNC->PMC)	F00000-F02511	**Byte**, Char	Read Only

Address Type	Range	Data Type	Access
	F00000-F02510 F00000-F02508 Fxxxxx.0-Fxxxxx.7	Word, Short DWord, Long, Float Boolean	
G (Signal to PMC->CNC)	G00000-G02511 G00000-G02510 G00000-G02508 Gxxxxx.0-Gxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
K (Keep relay)	K00000-K00909 K00000-K00908 K00000-K00906 Kxxxxx.0-Kxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
M (Input signal from other devices)	M00000-M00511 M00000-M00510 M00000-M00508 Mxxxxx.0-Mxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
N (Output signal from other devices)	N00000-N00511 N00000-N00510 N00000-N00508 Nxxxxx.0-Nxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
R (Internal relay)	R00000-R09119 R00000-R09118 R00000-R09116 Rxxxxx.0-Rxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
T (Changeable timer)	T00000-T00299 T00000-T00298 T00000-T00296 Txxxxx.0-Txxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
X (Signal to machine->PMC)	X00000-X00127 X00000-X00126 X00000-X00124 Xxxxxx.0-Xxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
Y (Signal to PMC->machine)	Y00000-Y00127 Y00000-Y00126 Y00000-Y00124 Yxxxxx.0-Yxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
Custom Macro Value (common range)	#0100-#0999	**Float**, Double	Read/Write
Custom Macro Value (local range)	#0001-#0033	**Float**, Double	Read Only
Custom Macro Value (system range)	#1000-#9999	**Float**, Double	Read/Write

CNC Data

Status Info Tags

Tool Offset

Workpiece Zero Offset

Parameter Read Values

Alarms Values

Arrays

Arrays are supported for all PMC addresses, except for Custom Macros in the system range and where Boolean or string data types are used. Tool Offset data cannot be addressed as an array. The syntax for declaring an array is as follows:

Mxxxxx[cols] with assumed row count of 1.
Mxxxxx[rows][cols] where M is the address type and *xxxxx* is the byte offset of the first element in the array.

Note: For all arrays, the total number of bytes being requested cannot exceed the specified request size.

Strings

All address types can be read and written to as ASCII strings. Each byte of memory contains one ASCII character. The length of strings can range from 1 to 120 and is entered in place of the bit number. An additional character "M" is appended to the address to distinguish string addresses from bit addresses.

Example
To address a string of length 100 characters, starting at D00200, enter D00200.100 M.

Unsolicited Data

If tags belong to a slave device, then their address types and ranges are validated according to the data areas configured for that device. For example, if the slave device is configured with a single area with D01000 to D01100, then a tag with address D01000 would be valid, but tags with addresses D01101 or C00001 would be invalid. All tags belonging to a slave device are Read Only.

See Also: Unsolicited Data Areas

Note: Use caution when modifying Word, Short, DWord, Long, and Float types. Since all addresses start at a byte offset within the device, it is possible for the memory associated with tags to overlap. For example, word tags D00000 and D00001 overlap at byte 1. Writing to D00000 also modifies the value held in D00001. It is recommended that these memory types be used such that each value to be read and written to by the driver occupy a unique range of memory in the device. For example, map 3 Word values to bytes D00000-D00001, D00002-D00003, and D00004-D00005. Tags to access these values would then have addresses D00000, D00002, and D00004 respectively, and a data type of Word.

Power Mate i

The following addresses are supported for this model. Not all address ranges may be valid for the particular device being used. *For more information, refer to the specific device's documentation.* To jump to a specific section, select a link from the list below.

PMC Data

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

Address Type	Range	Data Type	Access
A (Message demand)	A00000-A00124 A00000-A00123 A00000-A00121 Axxxxx.0-Axxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
C (Counter)	C00000-C00199 C00000-C00198 C00000-C00196 Cxxxxx.0-Cxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
D (Data table)	D00000-D09999 D00000-D09998 D00000-D09996 Dxxxxx.0-Dxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
E (Extended relay)	E00000-E07999 E00000-E07998 E00000-E07996 Exxxxx.0-Exxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
F (Signal to CNC->PMC)	F00000-F02511 F00000-F02510 F00000-F02508 Fxxxxx.0-Fxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
G (Signal to PMC->CNC)	G00000-G02511 G00000-G02510 G00000-G02508 Gxxxxx.0-Gxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
K (Keep relay)	K00000-K00909 K00000-K00908 K00000-K00906 Kxxxxx.0-Kxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
M (Input signal from other devices)	M00000-M00511 M00000-M00510 M00000-M00508 Mxxxxx.0-Mxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
N (Output signal from other devices)	N00000-N00511 N00000-N00510 N00000-N00508 Nxxxxx.0-Nxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
R (Internal relay)	R00000-R09119 R00000-R09118 R00000-R09116 Rxxxxx.0-Rxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
T (Changeable timer)	T00000-T00299 T00000-T00298 T00000-T00296 Txxxxx.0-Txxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write

Address Type	Range	Data Type	Access
X (Signal to machine->PMC)	X00000-X00127 X00000-X00126 X00000-X00124 Xxxxxx.0-Xxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
Y (Signal to PMC->machine)	Y00000-Y00127 Y00000-Y00126 Y00000-Y00124 Yxxxxx.0-Yxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
Custom Macro Value (common range)	#0100-#0999	**Float**, Double	Read/Write
Custom Macro Value (local range)	#0001-#0033	**Float**, Double	Read Only
Custom Macro Value (system range)	#1000-#9999	**Float**, Double	Read/Write

CNC Data

Status Info Tags
Tool Offset
Workpiece Zero Offset
Parameter Read Values
Alarms Values
Diagnostic Values
Path Value
Read Axis Data Values
Program Value
Read Dynamic2 Data Values

Arrays

Arrays are supported for all PMC addresses, except for Custom Macros in the system range and where Boolean or string data types are used. Tool Offset data cannot be addressed as an array. The syntax for declaring an array is as follows:

Mxxxx[cols] with assumed row count of 1.
Mxxxxx[rows][cols] where M is the address type and *xxxxx* is the byte offset of the first element in the array.

Note: For all arrays, the total number of bytes being requested cannot exceed the specified request size.

Strings

All address types can be read and written to as ASCII strings. Each byte of memory contains one ASCII character. The length of strings can range from 1 to 120 and is entered in place of the bit number. An additional character "M" is appended to the address to distinguish string addresses from bit addresses.

Example
To address a string of length 100 characters, starting at D00200, enter D00200.100 M.

Note: Use caution when modifying Word, Short, DWord, Long, and Float types. Since all addresses start at a byte offset within the device, it is possible for the memory associated with tags to overlap. For example, word tags D00000 and D00001 overlap at byte 1. Writing to D00000 also modifies the value held in D00001. It is recommended that these memory types be used such that each value to be read and written to by the driver occupy a unique range of memory in the device. For example, map 3 Word values to bytes D00000-

D00001, D00002-D00003, and D00004-D00005. Tags to access these values would then have addresses D00000, D00002, and D00004 respectively, and a data type of Word.

Open

The following addresses are supported for this model. Not all address ranges may be valid for the particular device being used. For more information, refer to the specific device's documentation. To jump to a specific section, select a link from the list below.

CNC Data
Arrays
Strings
Unsolicited Data

PMC Data

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

Address Type	Range	Data Type	Access
A (Message demand)	A00000-A32767 A00000-A32766 00000-A32764 Axxxxx.0-Axxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
C (Counter)	C00000-C32767 C00000-C32766 C00000-C32764 Cxxxxx.0-Cxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
D (Data table)	D00000-D32767 D00000-D32766 D00000-D32764 Dxxxxx.0-Dxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
E (Extended relay)	E00000-E32767 E00000-E32766 E00000-E32764 Exxxxx.0-Exxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
F (Signal to CNC->PMC)	F00000-F32767 F00000-F32766 F00000-F32764 Fxxxxx.0-Fxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
G (Signal to PMC->CNC)	G00000-G32767 G00000-G32766 G00000-G32764 Gxxxxx.0-Gxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
K (Keep relay)	K00000-K32767 K00000-K32766 K00000-K32764 Kxxxxx.0-Kxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
M (Input signal from other devices)	M00000-M32767 M00000-M32766	**Byte**, Char Word, Short	Read Only

Address Type	Range	Data Type	Access
	M00000-M32764 Mxxxxx.0-Mxxxxx.7	DWord, Long, Float Boolean	
N (Output signal from other devices)	N00000-N32767 N00000-N32766 N00000-N32764 Nxxxxx.0-Nxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
R (Internal relay)	R00000-R32767 R00000-R32766 R00000-R32764 Rxxxxx.0-Rxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
T (Changeable timer)	T00000-T32767 T00000-T32766 T00000-T32764 Txxxxx.0-Txxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
X (Signal to machine->PMC)	X00000-X32767 X00000-X32766 X00000-X32764 Xxxxxx.0-Xxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read Only
Y (Signal to PMC->machine)	Y00000-Y32767 Y00000-Y32766 Y00000-Y32764 Yxxxxx.0-Yxxxxx.7	**Byte**, Char Word, Short DWord, Long, Float Boolean	Read/Write
Custom Macro Value (common range)	#0100-#0999	**Float**, Double	Read/Write
Custom Macro Value (local range)	#0001-#0033	**Float**, Double	Read Only
Custom Macro Value (system range)	#1000-#9999	**Float**, Double	Read/Write

CNC Data

Status Info Tags
Tool Offset
Workpiece Zero Offset
Parameter Read Values
Alarms Values
Diagnostic Values
Path Value
Read Axis Data Values
Program Value
Program Name
Read Dynamic2 Data Values

Arrays

Arrays are supported for all PMC addresses, except for Custom Macros in the system range and where Boolean or string data types are used. Tool Offset data cannot be addressed as an array. The syntax for declaring an array is as follows:

Mxxxxx[cols] with assumed row count of 1.
Mxxxxx[rows][cols] where M is the address type and *xxxxx* is the byte offset of the first element in the array.

Note: For all arrays, the total number of bytes being requested cannot exceed the specified request size.

Strings

All address types can be read and written to as ASCII strings. Each byte of memory contains one ASCII character. The length of strings can range from 1 to 120 and is entered in place of the bit number. An additional character "M" is appended to the address to distinguish string addresses from bit addresses.

Example
To address a string of length 100 characters starting at D00200, enter D00200.100 M.

Unsolicited Data

If tags belong to a slave device, then their address types and ranges are validated according to the data areas configured for that device. For example, if the slave device is configured with a single area with D01000 to D01100, then a tag with address D01000 would be valid, but tags with addresses D01101 or C00001 would be invalid. All tags belonging to a slave device are Read Only.

See Also: Unsolicited Data Areas

Note: Use caution when modifying Word, Short, DWord, Long, and Float types. Since all addresses start at a byte offset within the device, it is possible for the memory associated with tags to overlap. For example, word tags D00000 and D00001 overlap at byte 1. Writing to D00000 also modifies the value held in D00001. It is recommended that these memory types be used such that each value to be read and written to by the driver occupy a unique range of memory in the device. For example, map 3 Word values to bytes D00000-D00001, D00002-D00003, and D00004-D00005. Tags to access these values would then have addresses D00000, D00002, and D00004 respectively, and a data type of Word.

Status Info Tags

The values in a CNC machine's status information are read using the *cnc_statinfo* Focas API call.

Series 15i

Tag Name	Data Type	Access	Description
statinfo_alarm	Short	Read Only	Status of alarm
statinfo_aut	Short	Read Only	Automatic mode selection
statinfo_battery	Short	Read Only	Status of battery
statinfo_dummy1	Short	Read Only	Reserved for future use
statinfo_dummy2	Short	Read Only	Reserved for future use
statinfo_edit	Short	Read Only	Status of program editing
statinfo_emergency	Short	Read Only	Status of emergency
statinfo_labelskip	Short	Read Only	Status of label skip
statinfo_manual	Short	Read Only	Manual mode selection
statinfo_motion	Short	Read Only	Status of axis movement, dwell
statinfo_mstb	Short	Read Only	Status of M,S,T,B function
statinfo_run	Short	Read Only	Status of automatic operation

Tag Name	Data Type	Access	Description
statinfo_warning	Short	Read Only	Status of warning
statinfo_write	Short	Read Only	Status of writing backed up memory

Returned Status Codes

Tag Name	Status Code
statinfo_alarm	0 : No alarm 1 : Alarm
statinfo_aut	0 : No selection 1 : MDI 2 : DNC 3 : Memory 4 : Edit 5 : Teach In
statinfo_battery	0 : Normal 1 : Battery low (backed up memory) 2 : Battery low (absolute position detector)
statinfo_dummy1	Not used
statinfo_dummy2	Not used
statinfo_edit	0 : Not editing 1 : Edit 2 : Search 3 : Verify 4 : Condense 5 : Read 6 : Punch
statinfo_emergency	0 : Not emergency 1 : Emergency
statinfo_labelskip	0 : Label skip 1 : Not label skip
statinfo_manual	0: No selection 1 : Reference 2 : INC feed 3 : Handle 4 : Jog 5 : Angular Jog 6 : Inc+Handl 7 : Jog+Handl
statinfo_motion	1 : Motion 2 : Dwell 3 : Wait (Waiting: only TT)
statinfo_mstb	1 : FIN
statinfo_run	0 : Stop 1 : Hold 2 : Start 3 : MSTR (Jog MDI)

Tag Name	Status Code
	4 : Restart (Not blinking)* 5 : PRSR (Program restart) 6 : NSRC (Sequence number search) 7 : Restart (Blinking)** 8 : Reset 13 : HPCC (During RISC operation)
statinfo_warning	0 : No warning 1 : Warning
statinfo_write	0 : Not writing 1 : @Writing

* Except under manual mode and under cutter radius compensation outside corner.
** Under manual mode or under cutter radius compensation outside corner.

Series 16i/18i/21i/Power Mate i/Open

Tag Name	Data Type	Access	Description
statinfo_alarm	Short	Read Only	Status of alarm
statinfo_aut	Short	Read Only	Automatic / Manual mode selection
statinfo_edit	Short	Read Only	Status of program editing
statinfo_emergency	Short	Read Only	Status of emergency
statinfo_hdck	Short	Read Only	Status of manual handle re-trace
statinfo_motion	Short	Read Only	Status of axis movement, dwell
statinfo_mstb	Short	Read Only	Status of M,S,T,B function
statinfo_run	Short	Read Only	Status of automatic operation
statinfo_tmmode	Short	Read Only	T/M mode selection

Returned Status Codes

Tag Name	Status Code
statinfo_alarm	0 : Others 1 : Alarm 2 : Battery low 3 : Fan alarm
statinfo_aut	0 : MDI 1 : Memory 3 : Edit 4 : Handle 5 : Jog 6 : Teach in Jog 7 : Teach in Handle 8 : INC feed 9 : Reference 10 : Remote 11 : TEST(test operation mode)

Tag Name	Status Code
statinfo_edit	M Series 0 : Not editing 1 : Edit 2 : Search 3 : Output 4 : Input 5 : Compare 6 : Label Skip (Label skip status) 7 : Restart (During program restart) 8 : HPCC (During RISC operation) 9 : PTRR (During tool retraction and recovery mode) 10 : RVRS (During retracing) 11 : RTRY (During reprogressing) 12 : RVED (End of retracing) 13 : Handle (During handle overlapping) 14 : Offset (During tool length measurement mode) 15 : Work Offset (During work zero point measurement mode) 16 : AICC (During AI contour control) 17 : Memory-Check (Checking tape memory) 18 : Customer's Board (During customer's board control) 19 : Save (Saving fine torque sensing data) 20 : AI NANO (During AI nano contour control) 21 : AI APC (During AI advanced preview control) 22 : MBL APC (During multi blocks advanced preview control) 23 : NANO HP (Running of AI High-precision Contour Control. 24 : AI HPCC (Running of AI Nano High-precision Contour Control) 25 : 5-AXIS (Running of 5-axes machining) 26 : LEN (Change the manual active offset value: length offset change mode) 27 : RAD (Change the manual active offset value: radius offset change mode) 28 : WZR (Change the manual active offset value: workpiece origin offset change mode) 39 : TCP (During tool center point control of 5-axes machining) 40 : TWP (During tilted working plane command) 41 : TCP+TWP (During tool center point control of 5-axes machining and tilted working plane command) 42 : APC (Advanced Preview Control) T Series 0 : Not editing 1 : Edit 2 : Search 3 : Output 4 : Input 5 : Compare 6 : Label Skip (Label skip status) 7 : Offset (During writing mode of tool length compensation amount) 8 : Work Shift (During writing mode of work shift amount) 9 : Restart (During program restart) 14 : PTRR (During tool retraction and recovery mode)

Tag Name	Status Code
	17 : Memory-Check (Checking tape memory) 19 : Save (Saving fine torque sensing data) 23 : NANO HP (Running of AI High-precision Contour Control) 24 : AI HPCC (Running of AI Nano High-precision Contour Control) 26 : OFSX (Change the manual active offset value: X-axis offset change mode) 27 : OFSZ (Change the manual active offset value: Z-axis offset change mode) 28 : WZR (change the manual active offset value: workpiece origin offset change mode) 29 : OFSY (Change the manual active offset value: Y-axis offset change mode) 31 : TOFS (Change the manual active offset value: Tool offset change mode) 39 : TCP (During tool center point control of 5-axes machining) 40 : TWP (During tilted working plane command) 41 : TCP+TWP (During tool center point control of 5-axes machining and tilted working plane command) 42 : APC (Advanced Preview Control)
statinfo_emergency	0 : Not emergency 1 : Emergency 2 : Reset
statinfo_hdck	N/A
statinfo_motion	1 : Motion 2 : Dwell
statinfo_mstb	0 : Others 1 : FIN
statinfo_run	0 : Reset 1 : Stop 2 : Hold 3 : Start 4 : MSTR*
statinfo_tmmode	0 : T Mode 1 : M Mode

* During operation of Jog MDI, and retraction and re-positioning of tool retraction and recovery.

Tool Offset

The values in a CNC machine's tool offset information are read using the *cnc_rdtofs* and written using the *cnc_wrtofs* Focas API calls.

CNC Data

Address Type	Range	Data Type	Access
Tool Offset	TOFS:nn:o nn = Tool Number (01-64) o = Offset Type (0-9)	**Long**, DWord	Read/Write

Tool Offset Types

The meaning of the tool offset type depends upon the hardware. The following tables summarize the various offset types.

	Cutter Radius	Tool Length
Wear	0	2
Geometry	1	3

Lathe Series (T series)

	X-Axis	Z-Axis	Nose R	Imaginary Tool Nose	Y-Axis
Wear	0	2	4	6	8
Geometry	1	3	5	7	9

Tool Offset Values

Series 15, 150i

6007#0 (OFE)	6004#0 (OFD)	6002#1 (OFC)	6002#0 (OFA)	Linear axis mm input [mm]	Linear axis inch input [inch]	Rotation axis [deg]
0	0	0	1	0.01	0.001	0.01
0	0	0	0	0.001	0.0001	0.001
0	0	1	0	0.0001	0.00001	0.0001
0	1	0	0	0.00001	0.000001	0.00001
1	0	0	0	0.000001	0.0000001	0.000001

Series 16/18/21, 160/180/210, 160i/180i/210i, 0i, Power Mate, Open

	1004#1 (ISC)	1004#0 (ISA)	Linear axis mm input [mm]	Linear axis inch input [inch]	Rotation axis [deg]
IS-A*	0	1	0.01	0.001	0.01
IS-B	0	0	0.001	0.0001	0.001
IS-C**	1	0	0.0001	0.00001	0.0001

* IS-A is effective for Power Mate i-H.
** IS-C is effective for Power Mate i-D.

Workpiece Zero Offset

The values in a CNC machine's workpiece zero offset information are read using the *cnc_rdzofs* and written using the *cnc_wrzofs* Focas API calls.

Note: Not all addresses are valid for all device models.

CNC Data

Address Type	Range	Data Type	Access
Workpiece Zero Offset	ZOFS:aa:ooo aa = axis (01-32) ooo = offset (000-306)	**Long**, DWord	Read/Write

Workpiece Zero Offset Values

Series 150

	1009#1 (ISE)	1004#5 (ISD)	1004#1 (ISF)	1004#0 (ISR)	Linear axis mm input [mm]	Linear axis inch input [inch]	Rotation axis [deg]
IS-A	0	0	0	1	0.01	0.001	0.01
IS-B	0	0	0	0	0.001	0.0001	0.001
IS-C	0	0	1	0	0.0001	0.00001	0.0001
IS-D	0	1	0	0	0.00001	0.000001	0.00001
IS-E	1	0	0	0	0.000001	0.0000001	0.000001

Series 15, 150i

	1012#3 (ISE)	1012#2 (ISD)	1012#1 (ISC)	1012#0 (ISA)	Linear axis mm input [mm]	Linear axis inch input [inch]	Rotation axis [deg]
IS-A	0	0	0	1	0.01	0.001	0.01
IS-B	0	0	0	0	0.001	0.0001	0.001
IS-C	0	0	1	0	0.0001	0.00001	0.0001
IS-D	0	1	0	0	0.00001	0.000001	0.00001
IS-E	1	0	0	0	0.000001	0.0000001	0.000001

Series 16/18/21, 160/180/210, 160i/180i/210i, 0i, Power Mate, Open

	1004#1 (ISC)	1004#0 (ISA)	Linear axis mm input [mm]	Linear axis inch input [inch]	Rotation axis [deg]
IS-A	0	1	0.01	0.001	0.01
IS-B	0	0	0.001	0.0001	0.001
IS-C	1	0	0.0001	0.00001	0.0001

Series 300i

	1013#3 (ISE)	1013#2 (ISD)	1013#1 (ISC)	1013#0 ISA)	Linear axis mm input [mm]	Linear axis inch input [inch]	Rotation axis [deg]
IS-	0	0	0	1	0.01	0.001	0.01

	1013#3 (ISE)	1013#2 (ISD)	1013#1 (ISC)	1013#0 ISA)	Linear axis mm input [mm]	Linear axis inch input [inch]	Rotation axis [deg]
A							
IS-B	0	0	0	0	0.001	0.0001	0.001
IS-C	0	0	1	0	0.0001	0.00001	0.0001
IS-D	0	1	0	0	0.00001	0.000001	0.00001
IS-E	1	0	0	0	0.000001	0.0000001	0.000001

Parameter Read Values

The values in a CNC machine's parameters can be read using the *cnc_rdparam* tags and the *cnc_rdparam* Focas API call.

Refer to the machine manual for specific parameters and which data type to assign to each.

For axis access, append .A to the parameter number, followed by the axis number. The first axis in the parameter is axis 1, the second axis 2, and so on. The highest axis to access is 8. If no axis is added to the address, the first axis is read.

For bit in parameter access, append a dot (.) to the parameter number, followed by the bit number. The first bit is bit 0. The maximum accessible bit is the 32nd bit. Bit access is allowed on axis access. Bit access is not allowed on parameters with default data types of float or double.

Address Type	Range	Data Type	Access
CNC Parameter	cnc_rdparam.0 - cnc_rdparam.37267 cnc_rdparam.0.A1 – cnc_rdparam.32767.A8 cnc_rdparam.0.0** – cnc_rdparam.32767.31** cnc_rdparam.0.A1.0**-cnc_rdparam.32767.A8.31**	**Default*** Byte, Char, Word, Short, DWord, Long, Float, Double	Read Only

* The correct data type for each parameter is dependent on which machine is being accessed. When creating a CNC Parameter tag, the data type must be manually assigned. Otherwise, the Default data type is assigned. If Default is the assigned data type, then the data type of the tag is determined and set at runtime.

** Bit in parameter tags can only be assigned the Boolean data type.

Alarm Values

Specific alarm information can be read using the *cnc_alarm2* tag and using the *cnc_alarm2* Focas API call.

Refer to the machine manual for which bits correspond with which alarms. When a bit is set to true, there is one or more alarms of that type present in the machine.

For bit access, append a dot (.) to the cnc_alarm2 tag. The range is 32-bits starting at bit 0 (0-31). All bit tags can only be assigned the Boolean data type.

Address Type	Range	Data Type	Access
Alarm Reporting	cnc_alarm2 cnc_alarm2.0	DWord, **Long**, Boolean	Read Only

Address Type	Range	Data Type	Access
	through cnc_alarm2.31		

Diagnostic Values

The diagnostic values of the machine can be read with *cnc_diagnoss* tags and uses the *cnc_diagnoss* Focas API call. The user must provide a diagnosis number and the desired axis. For values that do not have an axis, 0 must be used. An axis value of -1 (ALL_AXES) is not supported.

The format for these tag addresses are in the form cnc_diagnoss: A:B.C where A is the diagnosis number, B is the axis, and C is optional bit access. Tag address examples include:

- cnc_diagnoss:301:1
- cnc_diagnoss:301:2
- cnc_diagnoss:301:3
- cnc_diagnoss:1140:0 (as a Byte)
- cnc_diagnoss:1140:0.0 (as a Boolean)

For bit access, append a dot (.) to the cnc_diagnoss tag. The range is 8-bits starting at bit 0 (0-7). All bit tags can only be assigned the Boolean data type.

Address Type	Range	Data Type	Access
Diagnostic Reporting	cnc_diagnoss:1:0 through cnc_diagnoss:A:B where A is the maximum diagnosis number and B is the maximum axis defined. These maximums are machine dependent. For bit access to values: cnc_diagnoss:1:0.0 through cnc_diagnoss:A:B.7	Boolean, Byte, Char, Word, Short, DWord, **Long**, Float, Double Boolean	Read Only

Path Values

The *target_path* tag can be used to get and set the current path for the controller. Writing a value to the target_path tag produces a *cnc_setpath* Focas API call to the machine. A read produces a *cnc_getpath* Focas API call.

Tag Name	Data Type	Access	Description
target_path	Byte, Char, Word, **Short**, DWord, Long	Read / Write	Used to get or set the current path for the controller.

Read Axis Data Values

The axis data values of the machine can be read with *cnc_rdaxisdata* tags and uses the *cnc_rdaxisdata* Focas API call. The user must provide a class, type, axis, and whether they want the data or flag value.
Refer to the documentation of cnc_rdaxisdata for the specific class and type combinations that retrieve the target information.

Note: The *cnc_rdaxisdata* call is only supported for the 0i and the 30i and higher Fanuc Focas controller models.
Refer to the Fanuc Focas API documentation for sub-model support information for this call.

The format for these tag addresses are in the form *cnc_rdaxisdata: A:B:C:data* or *cnc_rdaxisdata: A:B:C:flags.D* where A is the class, B is the type, C is the axis, and D is optional bit access on the flags value.

Data is organized by classes with the type of data dependent on the class selected.
Below are the definitions of each class, the types available, and the flags available (as documented by the Fanuc Focas API):

Class	Types Available	Flags Available
1: Position Value	0: Absolute position 1: Machine position 2: Relative position 3: Distance to go 4: Handle interruption (input unit) 5: Handle interruption (output unit) 6: Start point of program restart 7: Distance to go of program restart 8: Start point of block restart 9: Distance to go of block restart	Bit 0: Display state 1 = Displayed on CNC screen : 0 = Not displayed on CNC screen Bit 1: Axis detaching state 1 = enabled : 0 = disabled Bit 2: Interlock state 1 = enabled : 0 = disabled Bit 3: Machine lock state 1 = enabled : 0 = disabled Bit 4 : Servo off state 1 = enabled : 0 = disabled Bit 5: In-position check 1 = Not in-position : 0 = In-position Bit 6: Mirror image state 1 = enabled : 0 = disabled Bit 7: Diameter and radius setting switching function 1 = switching : 0 = not switching Bit 8: High speed program check mode(only machine coordinate) 1 = enabled : 0 = disabled Bit 9: Optional one axis approach by program restart 1 = executing : 0 = not executing Bit 10: Restart coordinates display on the program restart screen(except machine coordinate) 1 = possible : 0 = impossible (display "**********") Bit 11: Release state of axis (Flexible path axis assignment function) 1 = enabled : 0 = disabled Bit 12-15 : not used
2: Servo	0: Servo load meter 1: Load current (% unit) 2: Load current (Ampere unit)	None

Class	Types Available	Flags Available
3: Spindle	0: Spindle load meter 1: Spindle motor speed 2: Spindle speed (according to parameter 3799#2) 3: Spindle speed (got from Spindle motor speed) 4: Spindle load meter (average of each 250ms) 5: Spindle load meter (maximum value) 6: Spindle load meter (maximum value) (average of each 250ms) 7: Time that spindle can continue processing	For Types 5-7 Only: Bit 0: Data judgment result 1 = It has judged : 0 = Unjudgment Bit 1: Presence of data 1 = Exist : 0 = None Bit 2-15 : not used
4: Selected Spindle	0: Spindle load meter 1: Spindle motor speed 2: Spindle speed (according to parameter 3799#2) 3: Spindle speed (got from Spindle motor speed) 4: Spindle load meter (average of each 250ms) 5: Spindle load meter (maximum value) 6: Spindle load meter (maximum value) (average of each 250ms) 7: Time that spindle can continue processing	For Types 5-7 Only: Bit 0: Data judgment result 1 = It has judged : 0 = Unjudgment Bit 1: Presence of data 1 = Exist : 0 = None Bit 2-15 : not used
5: Speed	0: Feed rate (F) (Feed per minute) 1: Spindle speed (S) 2: Jog speed / Dry run speed 3: Tool tip speed 4: Rotation speed of servo motor 5: Feed rate (F/S)	Bit 0: Spindle speed 1 = spindle exists : 0 = spindle not exist Bit 1: Jog speed / Dry run speed 1 = Dry run speed : 0 = Jog speed Bit 2-15 : not used

Address examples include:

- cnc_rdaxisdata:1:1:1:data (Machine position of the x axis)
- cnc_rdaxisdata:1:1:2:data (Machine position of the y axis)
- cnc_rdaxisdata:1:1:3:data (Machine position of the z axis)
- cnc_rdaxisdata:1:1:1:flags.0 (Whether the machine position of the x axis is displayed on the screen or not)
- cnc_rdaxisdata:2:1:2:data (Load current as a percentage of the y axis)

For data that is not specific to an axis use an axis value of 1.

Address Type	Range	Data Type	Access
Axis Data Reporting	cnc_rdaxisdata:1:0:1:data through cnc_rdaxisdata:5:9:32:data cnc_rdaxisdata:1:0:1:flags through cnc_rdaxisdata:5:9:32:flags These maximums are device dependent. For ease of use, a maximum of 32 axes can be defined; note that the device may not allow that many. For bit access to flags value: cnc_rdaxisdata:1:0:1:flags.0 through cnc_rdaxisdata:5:9:32:flags.15	Boolean, Byte, Char, Word, Short, DWord, **Long**, Float, Double **Boolean**	Read Only

Program Values

The currently running program number along with the main program number can be read. These values can be the same or reflect different programs or sub-programs running. These values are read through the *cnc_rdprgnum* Focas API call.

Tag Name	Data Type	Access	Description
cnc_rdprgnum_Main	Word, Short, DWord, **Long**	Read Only	The main program set on the machine.
cnc_rdprgnum_Running	Word, Short, DWord, **Long**	Read Only	The currently running program on the machine. This can be a sub-program.

Program Name

The name of the program being executed can be read. These values are read through the *cnc_exeprgname* Focas API call.

Note: The *cnc_exeprgname* call is only supported for the Fanuc Focas controller models 0i and 30i and above. *Refer to the Fanuc Focas API documentation for sub-model support information for this call.*

Tag Name	Data Type	Access	Description
cnc_exeprg-name_Name	**String**	Read Only	The name of the program being executed.
cnc_exeprg-name_Number	Word, Short, DWord, **Long**	Read Only	The O number of the program being executed. It will be 0 if there is not an O number.

Read Dynamic2 Data Values

The read dynamic2 data represents many different values in a single call. There is positional data for each axis and data for the device itself. Positional data units can depend on the device. These values are read

through the *cnc_rddynamic2* Focas API call.

Tag Name	Data Type	Access	Description
cnc_rddynamic2_ActF	DWord, **Long**	Read Only	The actual feed rate.
cnc_rddynamic2_ActS	DWord, **Long**	Read Only	The actual spindle speed.
cnc_rddynamic2_Alarm cnc_rddynamic2_Alarm.0 through cnc_rddynamic2_Alarm.31	DWord, **Long Boolean**	Read Only	The alarm status. The bits represent different alarms based on the model. Refer to the device documentation for the specific meaning of each bit. Used for bit access to the different alarms.
cnc_rddynamic2_PrgMNum	DWord, **Long**	Read Only	The main program number.
cnc_rddynamic2_PrgNum	DWord, **Long**	Read Only	The current program number.
cnc_rddynamic2_SeqNum	DWord, **Long**	Read Only	The current sequence number.
cnc_rddynamic2:1:Abs through cnc_rddynamic2:32:Abs	DWord, **Long**	Read Only	The absolute position for the axis specified. The axis number can go up to 32 axes. The number of axes that have data depends on the number of axes defined on the controller.
cnc_rddynamic2:1:Dis through cnc_rddynamic2:32:Dis	DWord, **Long**	Read Only	The distance to go for the axis specified. The axis number can go up to 32 axes. The number of axes that have data depends on the number of axes defined on the controller.
cnc_rddynamic2:1:Mac through cnc_rddynamic2:32:Mac	DWord, **Long**	Read Only	The machine position for the axis specified. The axis number can go up to 32 axes. The number of axes that have data depends on the number of axes defined on the controller.
cnc_rddynamic2:1:Rel through cnc_rddynamic2:32:Rel	DWord, **Long**	Read Only	The relative position for the axis specified. The axis number can go up to 32 axes. The number of axes that have data depends on the number of axes defined on the controller.

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.

Address Validation

The following messages may be generated. Click the link for a description of the message.

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:
A tag address that has been specified statically contains one or more invalid characters.

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

Device address <address> is read only.

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically has a requested access mode that is not compatible with what the device supports for that address.

Solution:
Change the access mode in the client application.

Missing address.

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically has no length.

Solution:

Re-enter the address in the client application.

Device Status Messages

The following messages may be generated. Click the link for a description of the message.

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 IP address assigned to the device is incorrect.
3. 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 PLC device.
2. Verify that the IP address given to the named device matches that of the actual device.
3. 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 connection between the device and the host PC is broken.
2. The named device may have been assigned an incorrect IP address.

Solution:

1. Verify the cabling between the PC and the PLC device.
2. Verify that the IP address given to the named device matches that of the actual device.

General Driver Messages

The following messages may be generated. Click the link for a description of the message.

Could not acquire library handle for device <channel.device>. FWLIB error: <code>.

Could not acquire library handle for device <channel.device>. FWLIB error: <code>.

Error Type:
Warning

Possible Cause:

1. Call to Focas 1 Data Window Library to connect to device failed.
2. Invalid device IP or port number.
3. The device may not be running.
4. The device may be busy processing other requests.
5. There may be a cabling problem.

Solution:
The error code provided by the library should help diagnose the problem. If the problem is transient, the driver should be able to connect on a subsequent retry.

See Also:
Focas 1 Data Window Library Error Codes

Could not read one or more vacant macros in range starting at <address> on device <device>.

Error Type:
Warning

Possible Cause:
The macro number is not configured in the device.

Solution:
Check the tag address and device configuration.

Could not set request timeout for device <channel.device>. FWLIB error: <code>.

Error Type:
Warning

Possible Cause:

1. Call to Focas 1 Data Window Library to set request timeout failed.
2. Invalid timeout.
3. The device may be busy processing other requests.
4. There may be a cabling problem.

Solution:
The error code provided by the library should help diagnose the problem. If the problem is transient, the driver should be able to set the timeout on a subsequent retry.

See Also:
Focas 1 Data Window Library Error Codes

Invalid XML document. Reason: Error loading Unsolicited Data Areas for device <device-name>. End address can not be less than start address for area <area-number>.

Error Type:
Fatal

Possible Cause:
The end address for the given unsolicited data area is greater than the start address.

Solution:
Make the end address greater than or equal to the start address.

Invalid XML document. Reason: Error loading Unsolicited Data Areas for device <device-name>. Invalid area order or duplicate area number.

Error Type:
Fatal

Possible Cause:

1. The unsolicited data areas are not listed in the correct order.
2. There are duplicate area numbers.

Solution:

1. Correct the ordering of the unsolicited data areas to be in the increasing order starting at 1.
2. Renumber the duplicate area number.

See Also:
Unsolicited Data Areas

Invalid XML document. Reason: Error loading Unsolicited Data Areas for device <device-name>. Maximum size of area <area-number> is <size> bytes.

Error Type:
Fatal

Possible Cause:
The maximum number of bytes for areas 1, 2, or 3 are 1430, 1414, or 1398 respectively. The range that has been defined for these areas is outside this limit.

Solution:
Correct the unsolicited data area size based on the limits above.

See Also:
Unsolicited Data Areas

Read error occurred for address starting at <address> on device <channel.device>. FWLIB error: <code>.

Error Type:
Warning

Possible Cause:

1. Call to Focas 1 Data Window Library to read data failed.
2. Invalid PMC type.
3. Invalid address.
4. Invalid request size.
5. The device may be busy processing other requests.
6. There may be a cabling problem.

Solution:
The error code provided by the library should help diagnose the problem. If the problem is transient, the driver should be able to read the data on a subsequent retry.

See Also:
Focas 1 Data Window Library Error Codes

Unable to start the Fanuc Focas Data Window Library services.

Error Type:
Fatal

Possible Cause:
The driver was unable to load the Fanuc Focas Data Window Library.

Solution:
Make sure that the library is installed on the computer. Contact this software's distributor.

Write error occurred for address <address> on device <channel.device>. FWLIB error: <code>.

Error Type:
Warning

Possible Cause:
1. Call to Focas 1 Data Window Library to write data failed.
2. Invalid PMC type.
3. Invalid address.
4. Invalid request size.
5. The device may be busy processing other requests.
6. There may be a cabling problem.

Solution:
The error code provided by the library should help diagnose the problem. If the problem is transient, the driver should be able to write the data on a subsequent retry.

See Also:
Focas 1 Data Window Library Error Codes

Slave Device Driver Messages

The following messages may be generated. Click the link for a description of the message.

Attempt to launch unsolicited message server failed.
Could not access necessary system resources for slave device: <channel.device>.
Failed to connect slave device <channel.device>. Could not acquire library handle. FWLIB error: <code>.
Failed to connect slave device <channel.device>. Could not determine host IP address.
Failed to connect slave device <channel.device>. Could not set data area size.
Failed to connect slave device <channel.device>. Could not set data area start address.
Failed to connect slave device <channel.device>. Could not set data area type.
Failed to connect slave device <channel.device>. Could not set host IP.
Failed to connect slave device <channel.device>. Could not set host port.

Attempt to launch unsolicited message server failed.

Error Type:

Warning

Possible Cause:

The driver was not able to start the Unsolicited Message Server.

Solution:

For the restart to succeed, the message server executable file "UMsgServ.ext" must be located in the host computer's system folder.

Note:

The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:

Unsolicited Messaging

Could not access necessary system resources for slave device: <channel.device>.

Error Type:
Serious

Possible Cause:
The driver could not create data objects needed for unsolicited communications.

Solution:
Close down all unnecessary applications running on the host computer.

Failed to connect slave device <channel.device>. Could not acquire library handle. FWLIB error: <code>.

Error Type:
Warning

Possible Cause:
1. Call to Focas 1 Data Window Library to connect to device failed.
2. Invalid device IP or port number.
3. The device may not be running.
4. The device may be busy processing other requests.
5. There may be a cabling problem.

Solution:
The error code provided by the library should help diagnose the problem. If the problem is transient, the driver should be able to connect on a subsequent retry.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Focas 1 Data Window Library Error Codes

Failed to connect slave device <channel.device>. Could not determine host IP address.

Error Type:
Warning

Possible Cause:
Part of starting an unsolicited messaging session with a device includes informing the device of the IP of the host computer. This message will be posted if the driver fails to determine the default IP of the host computer.

Solution:
Make sure that an IP address is configured for the computer.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

Failed to connect slave device <channel.device>. Could not set data area size.

Error Type:
Warning

Possible Cause:
Call to Focas 1 Data Window Library to set unsolicited message data area size failed.

Solution:
Ensure that the specified range of addresses is valid.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Unsolicited Data Areas

Failed to connect slave device <channel.device>. Could not set data area start address.

Error Type:
Warning

Possible Cause:
Call to Focas 1 Data Window Library to set unsolicited message data area start address failed.

Solution:
Ensure that the specified start address is valid.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Unsolicited Data Areas

Failed to connect slave device <channel.device>. Could not set data area type.

Error Type:
Warning

Possible Cause:
Call to Focas 1 Data Window Library to set unsolicited message data area type failed.

Solution:
Ensure that the specified type is valid.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Unsolicited Data Areas

Failed to connect slave device <channel.device>. Could not set host IP.

Error Type:
Warning

Possible Cause:
Call to Focas 1 Data Window Library to set host IP of Unsolicited Message Server failed.

Solution:
Check the IP of the host computer.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

Failed to connect slave device <channel.device>. Could not set host port.

Error Type:
Warning

Possible Cause:
Call to Focas 1 Data Window Library to set host port of Unsolicited Message Server failed.

Solution:
Ensure that the specified port number is valid.

Note:

The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Unsolicited Transfer Control

Failed to connect slave device <channel.device>. Could not set message alive time.

Error Type:
Warning

Possible Cause:
Call to Focas 1 Data Window Library to set unsolicited message alive time failed.

Solution:
Ensure that the specified alive time value is valid.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Unsolicited Transfer Control

Failed to connect slave device <channel.device>. Could not set message retries.

Error Type:
Warning

Possible Cause:
Call to Focas 1 Data Window Library to set transmission control message retries failed.

Solution:
Ensure that the specified retries value is valid.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Unsolicited Transfer Control

Failed to connect slave device <channel.device>. Could not set message timeout.

Error Type:
Warning

Possible Cause:
Call to Focas 1 Data Window Library to set unsolicited message timeout failed.

Solution:
Ensure that the specified timeout value is valid.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Unsolicited Transfer Control

Failed to connect slave device <channel.device>. Could not set messaging properties. FWLIB data error <code>.

Error Type:
Warning

Possible Cause:
1. Call to Focas 1 Data Window Library to set unsolicited messaging properties failed due to a data related error.
2. Invalid transfer control parameters.
3. The device may not be running.
4. The device may be busy processing other requests.
5. There may be a cabling problem.

Solution:
The error code provided by the library should help diagnose the problem. If the problem is transient, the driver should be able to connect on a subsequent retry.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Focas 1 Data Window Library Error Codes
Unsolicited Transfer Control

Failed to connect slave device <channel.device>. Could not set messaging properties. FWLIB error: <code>.

Error Type:
Warning

Possible Cause:
1. Call to Focas 1 Data Window Library to set unsolicited messaging properties failed due to a non-data related error.
2. Invalid transfer control properties.
3. The device may not be running.
4. The device may be busy processing other requests.
5. There may be a cabling problem.

Solution:
The error code provided by the library should help diagnose the problem. If the problem is transient, the driver should be able to connect on a subsequent retry.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Focas 1 Data Window Library Error Codes
Unsolicited Transfer Control

Failed to connect slave device <channel.device>. Could not set number of data areas.

Error Type:
Warning

Possible Cause:
Call to Focas 1 Data Window Library to set number of unsolicited message data areas failed.

Solution:
Ensure that the device supports the number of data areas configured.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Unsolicited Data Areas

Failed to connect slave device <channel.device>. Could not set request timeout. FWLIB error: <code>.

Error Type:
Warning

Possible Cause:
1. Call to Focas 1 Data Window Library to set request timeout failed.
2. Invalid timeout.
3. The device may be busy processing other requests.
4. There may be a cabling problem.

Solution:
The error code provided by the library should help diagnose the problem. If the problem is transient, the driver should be able to set the timeout on a subsequent retry.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Focas 1 Data Window Library Error Codes

Failed to connect slave device <channel.device>. Could not set transmission control PMC type.

Error Type:
Warning

Possible Cause:
Call to Focas 1 Data Window Library to set transmission control PMC type failed.

Solution:
Ensure that the specified PMC type is valid.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Unsolicited Transfer Control

Failed to connect slave device <channel.device>. Could not set transmission control start address.

Error Type:

Warning

Possible Cause:
Call to Focas 1 Data Window Library to set transmission control start address failed.

Solution:
Ensure that the specified start address is valid.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Unsolicited Transfer Control

Failed to connect slave device <channel.device>. Could not start messaging session. FWLIB error: <code>.

Error Type:
Warning

Possible Cause:
1. Call to Focas 1 Data Window Library to start unsolicited messaging session failed.
2. Invalid transfer control parameters.
3. The device may not be running.
4. The device may be busy processing other requests.
5. There may be a cabling problem.

Solution:
The error code provided by the library should help diagnose the problem. If the problem is transient, the driver should be able to connect on a subsequent retry.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Focas 1 Data Window Library Error Codes
Unsolicited Transfer Control

Installed version of Focas Data Window Library does not support unsolicited communication. Device <device> deactivated.

Error Type:

Serious

Possible Cause:
The server project includes slave devices, but the version of the Focas 1 Data Window Library software installed on the system does not support unsolicited communication.

Solution:
Contact the distributor for library update. The device's firmware may also need to be upgraded.

See Also:
Additional Software Requirements

Received CNC power down notification from unsolicited message server. Reconnecting slave devices.

Error Type:
Warning

Possible Cause:
The Unsolicited Message Server has notified the driver that with one of the devices it has started an unsolicited messaging session with no longer appears to be running. The likely reason is that the CNC was powered down. Network problems could also be responsible.

Solution:
Restart the CNC and check for networking problems such as cable breaks. The driver should automatically resume communication with the device.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

Received CNC power up notification from unsolicited message server.

Error Type:
Warning

Possible Cause:
The Unsolicited Message Server has notified the driver that one of the devices it has started an unsolicited messaging session with has been powered up. This message likely follows a "Received CNC power down notification" message.

Solution:
This is for information only. The driver should automatically resume communication with the device.

Received socket error notification from unsolicited message server.

Error Type:
Warning

Possible Cause:
The Unsolicited Message Server experienced a socket error for one or more of the connections to a device on the network.

Solution:
If the problem is transient, the driver should recover from this error by restarting all unsolicited messaging sessions. If not, investigate cabling, CNC power supply, and I/F board.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

Received unsolicited message server shutdown notification.

Error Type:
Warning

Possible Cause:
The Unsolicited Message Server was shutdown while the driver was using it.

Solution:
This is for information only. The driver automatically restarts the message server. For the restart to succeed, the message server executable file "UMsgServ.exe" must be located in the host computer's system directory.

Note:
The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Unsolicited Messaging

Unsolicited message server does not seem to be running. Attempting to launch.

Error Type:
Warning

Possible Cause:
The Unsolicited Message Server has stopped running, and was not shutdown normally.

Solution:
This is for information only. The driver automatically restarts the message server. For the restart to succeed, the message server executable file "UMsgServ.exe" must be located in the host computer's system folder.

Note:

The device's _System._Error Tag is set if the driver fails to start an unsolicited messaging session, or restart the session after detection of a communications problem. Tags belonging to a slave device in an error state continue to show the last value received from the device or their initial value of zero.

See Also:
Unsolicited Messaging

Focas 1 Data Window Library Error Codes

The Fanuc Focas Ethernet Driver uses the Fanuc Focas 1 Data Window Library software to communicate with devices on the network. When the library cannot complete a request made by this driver, it returns an error code describing the reason. These error codes are included in the relevant driver error messages. This table is provided to aid in diagnosing the hardware or software configuration problem causing these errors. For more information, refer to **Additional Software Requirements**.

Error Code	Error Type	Meaning
-17	Protocol	Data from Ethernet board is incorrect.
-16	Socket	Investigate CNC power supply, Ethernet cable, and I/F board.
-15	DLL	There is no DLL file for CNC series.
-8	Handle	Invalid connection handle.
-7	Version	The CNC / PMC version does not match that of the library. Replace the library or the CNC / PMC control software.
-6	Unexpected	An unanticipated error occurred.
-2	Reset	The RESET or STOP button was pressed.
-1	Busy	The CNC was busy processing another request. This commonly occurs during slave device connect attempts. The driver retries until a connection is made.
0	Normal	Function was completed without error.
1 (CNC)	Function	Function was not executed or is not available. This can occur if the Unsolicited Message Server goes down while the driver is using it. The driver then attempts to restart the message server.
1 (PMC)	No PMC	The PMC does not exist.
2	Length	Invalid data block length.
3 (CNC)	Number	Invalid data number.
3 (PMC)	Range	Invalid address range.
4 (CNC)	Attribute	Invalid data attribute. This could result from a bad address type or range for data Read/Write.
4 (PMC)	Type	Invalid address type.
5	Data	Invalid data.
6	No Option	Invalid CNC option.
7	Protection	Write operation is prohibited.

Error Code	Error Type	Meaning
8	Overflow	CNC tape memory is overflowed.
9	Parameter	CNC parameter is set incorrectly.
10	Buffer	The buffer is empty or full. This can occur if there are more slave devices than the Unsolicited Message Server is configured to handle.
11	Path	Invalid path number.
12	Mode	Invalid CNC mode.
13	Reject	CNC rejected request. This can occur if an attempt is made to start multiple unsolicited messaging sessions with the same device.
14	Data Server	Data server error occurred.
15	Alarm	Function cannot be executed due to an alarm in CNC.
16	Stop	CNC status is stop or emergency.
17	Password	Data is protected by the CNC data protection function.

Index

E

F

G

I

L

M

N

O

P

R

S

T

U

W