

GE Fanuc Ethernet Driver Help

© 2009 Kepware Technologies

Table of Contents

1	Getting Started.....	3
	Help Contents.....	3
	Overview	3
2	Device Setup.....	3
	Device Setup	3
	Variable Import Settings.....	4
	PLC Settings	5
3	Automatic Tag Database Generation.....	5
	Automatic Tag Database Generation.....	5
	Tag Hierarchy	6
	Import File-to-Server Name Conversions.....	6
	Importing VersaPro Tags.....	7
	Importing VersaPro Tags.....	7
	VersaPro Import Preparation: VersaPro Steps.....	8
	VersaPro Import Preparation: OPC Server Steps.....	9
	Highlighting VersaPro Variables.....	9
	VersaPro Array Tag Import.....	10
	Importing Cimplicity Logic Developer Tags.....	10
	Importing Cimplicity Logic Developer Tags.....	10
	Cimplicity Logic Developer Import Preparation: Logic Developer Steps.....	11
	Cimplicity Logic Developer Import Preparation: OPC Server Steps.....	12
	Highlighting Cimplicity Logic Developer Variables.....	13
	Cimplicity Logic Developer Array Tag Import.....	13
	Importing Proficy Logic Developer Tags.....	13
	Importing Proficy Logic Developer Tags.....	13
	Proficy Logic Developer Import Preparation: Logic Developer Steps.....	13
	Proficy Logic Developer Import Preparation: OPC Server Steps.....	16
	Highlighting Proficy Logic Developer Variables.....	16
	Proficy Logic Developer Array Tag Import.....	16
4	Optimizing Your GE Fanuc Ethernet Communications.....	17
	Optimizing GE Fanuc Ethernet Communications.....	17
5	Data Types Description.....	18
	Data Types Description.....	18
6	Address Descriptions.....	19
	Address Descriptions.....	19
	PACSystems Addressing.....	19
	Symbolic Variables.....	20
	311 Addressing.....	23
	313 Addressing.....	23
	331 Addressing.....	24
	341 Addressing.....	25
	350 Addressing.....	26
	360 Addressing.....	26
	731 Addressing.....	27
	732 Addressing.....	28
	771 Addressing.....	29
	772 Addressing.....	30

781 Addressing.....	31
782 Addressing.....	32
GE OPEN Addressing.....	33
Horner OCS Addressing.....	34
VersaMax Addressing.....	35
Advanced Addressing.....	36
Special Items.....	37
7 Error Descriptions.....	39
Error Descriptions.....	39
Address Validation.....	40
Address Validation Error Messages.....	40
Missing address.....	40
Device address '<address>' contains a syntax error.....	40
Address '<address>' is out of range for the specified device or register.....	40
Device address '<address>' is not supported by model '<model name>'.....	41
Data Type '<type>' is not valid for device address '<address>'.....	41
Device address '<address>' is Read Only.....	41
Array size is out of range for address '<address>'.....	41
Array support is not available for the specified address: '<address>'.....	41
Device Status Messages.....	42
Device Status Messages.....	42
Device '<device name>' not responding.....	42
Unable to write to '<address>' on device '<device name>'.....	42
Attempting to reacquire symbolic variable mapping information for device '<device name>'. Cannot read or write symbolic variables until complete.....	42
Driver Error Messages.....	43
Driver Error Messages.....	43
Winsock V1.1 or higher must be installed to use the GE Fanuc Ethernet device driver.....	43
Device '<device name>' returned error code <error num> reading in byte(s) starting at <add.....	43
The symbolic variable name in tag address '<address>' does not exist in current configuration of device '<device name>'.....	43
The address length of the symbolic variable addressed by '<tag name>' on device '<device>' is bigger than the configured block size for this device. Deactivating tag.....	44
The array size of address '<address>' on device '<device name>' must be the same as symbolic variable's array size of [rows][cols].....	44
The data type of the symbolic variable addressed by '<address>' on device '<device name>' is not compatible with tag.....	44
Automatic Tag Database Generation Messages.....	44
Automatic Tag Database Generation Messages.....	44
Unable to generate a tag database for device <device name>. Reason: Low memory resources.....	45
Unable to generate a tag database for device <device name>. Reason: Import file is invalid.....	45
Database Error: Array tags '<orig. tag name><dimensions>' exceed 31 characters. Tags renamed.....	45
Database Error: Tag '<orig. tag name>' exceeds 31 characters. Tag renamed to '<new tag name>'.....	45
Database Error: Datatype '<type>' for tag '<tag name>' not found in import file.....	46
Database Error: Logic Developer Variable Arrays are currently not supported. Array Tag(s) '.....	46
Database Error: No Reference Address found for tag '<tag name>' in import file. Tag not created.....	46
Database Error: Only variables with Data Source '<data source name>' are imported.....	46
Database Error: Data type '<type>' arrays are currently not supported. Tag '<array tag name>' not created... ..	47

GE Fanuc Ethernet Driver Help

Help version 1.037

CONTENTS

[Overview](#)

What is the GE Fanuc Ethernet Driver?

[Device Setup](#)

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

[Automatic Tag Database Generation](#)

How can I easily configure tags for the GE Fanuc Ethernet driver?

[Optimizing Your GE Fanuc Ethernet Communications](#)

How do I get the best performance from the GE Fanuc Ethernet driver?

[Data Types Description](#)

What data types does this driver support?

[Address Descriptions](#)

How do I address a data location on a GE device?

[Error Descriptions](#)

What error messages does the GE Fanuc Ethernet driver produce?

Overview

The GE Fanuc Ethernet Driver was designed specifically for use with 32 bit OPC server products running on Intel microprocessor based computers. For operating system (OS) requirements, please refer to the OPC server help documentation.

This driver is intended for use with GE Programmable Logic Controllers that may be accessed via an Ethernet module.

Device Setup

Supported Devices

Series 90-30 311/313, 331/341, 350,360
Series 90-70 731/732, 771/772, 781/782
GE OPEN (Wide range model support)
Horner OCS (Horner's Operator Control Stations)
PACSystems RX3i, RX7i
VersaMax family

Communication Protocol

Ethernet; using Winsock V1.1 or higher.
Supported Communication Parameters

Connection Timeout

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

Request Timeout

This parameter specifies the time that the driver will wait on a response from the device before giving up and going on to the next request. Longer timeouts will only affect performance if a device is not responding. The default setting is 1000 milliseconds. The valid range is 50 to 30000 milliseconds.

Retry Attempts

This parameter specifies the number of times that the driver will retry a message before giving up and going on to the

next message. The default setting is 3 retries. The valid range is 1 to 10.

TCP/IP Port Number

This parameter specifies the TCP/IP port number that the remote device is configured to use. The default port number is 18245.

Maximum Bytes per Request

This parameter specifies the number of bytes that may be requested from a device at one time. To refine this driver's performance, configure the request size to one of the following settings: 32, 64, 128, 256, 512, 1024, or 2048 bytes. The default value is 256 bytes.

Device IDs

Up to 1024 devices may be defined on a given channel. Each device on the channel must be uniquely identified by its own IP address. The LogicMaster 90 TCP software supplied by GE can be used to configure the IP address of an Ethernet module. In general the Device ID has the following format YYY.YYY.YYY.YYY, where YYY designates the device IP address (each YYY byte should be in the range of 0 to 255).

Automatic Tag Database Generation

[GE Fanuc Ethernet Variable Import Settings](#)

Cable Diagrams

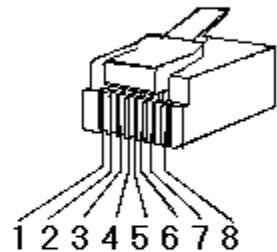
Patch Cable (Straight Through)

TD + 1	OR/WHT	OR/WHT	1 TD +
TD - 2	OR	OR	2 TD -
RD + 3	GRN/WHT	GRN/WHT	3 RD +
4	BLU	BLU	4
5	BLU/WHT	BLU/WHT	5
RD - 6	GRN	GRN	6 RD -
7	BRN/WHT	BRN/WHT	7
8	BRN	BRN	8

RJ45

RJ45

10 BaseT



Crossover Cable

TD + 1	OR/WHT	GRN/WHT	1 TD +
TD - 2	OR	GRN	2 TD -
RD + 3	GRN/WHT	OR/WHT	3 RD +
4	BLU	BLU	4
5	BLU/WHT	BLU/WHT	5
RD - 6	GRN	OR	6 RD -
7	BRN/WHT	BRN/WHT	7
8	BRN	BRN	8

RJ45

RJ45

8-pin RJ45

Variable Import Settings

Tag Import File

Enter the exact location of the variable import file (.snf or .csv file extension) or Logic Developer variable import file (.txt or other file extension) from which variables will be imported. It is this file that will be used when Automatic Tag

Database Generation is instructed to create the tag database. All tags will be imported and expanded according to their respective data types.

Display Descriptions

Check this option in order to import tag descriptions. If necessary, a description will be given to tags with long names stating the original tag name.

See Also: [Automatic Tag Database Generation](#).

PLC Settings

Program Name

The Program Name, which is also referred to as the Control Program Task Name, is required in order to access program block registers (P) or subprogram block registers (L). It may be obtained by accessing _PROGNAME or from the following programming packages.

VersaPro

The program name is referred to as the Folder Nickname. It is located under **PLC | Status Info**.

Proficy Machine Edition - Logic Developer Cimplicity Machine Edition - Logic Developer

For 90-70 PLCs:

The program name is referred to as the PLC Target Name. It is located under Target Properties (Inspector).

For PACSystems:

The program name is located under Properties for Program Block _MAIN.

Note: The program name must be 7 characters or less in length. The name will default to uppercase.

See Also: [Special Items](#)

CPU Slot Location

The CPU Slot Location is the physical location of the device on the rack. The default setting is slot 1. The valid range is 0 to 15.

Note: For CPUs that take up two slots, the leftmost slot covered is the CPU slot.

Automatic Tag Database Generation

The GE Fanuc Ethernet Device Driver generates its tags offline and is based on variables imported from a text file. It is offline in the sense that no connection to the device is required to generate tags. The text file (variables to import) can originate from one of the following applications:

1. Proficy Machine Edition - Logic Developer
2. Cimplicity Machine Edition - Logic Developer
3. VersaPro

There are two parts to Automatic Tag Database Generation: creating a variable import file from the application in use and generating tags based on the variable import file from the OPC Server. For information on creating variable import files, refer to [Importing VersaPro Tags](#), [Importing Proficy Logic Developer Tags](#) or [Importing Cimplicity Logic Developer Tags](#). It is recommended that users become familiar with the second part before dealing with the first part.

Overview

If the target device supports its own local tag database the driver will read the device's tag information and use this data to generate OPC tags within the OPC Server. If the device does not natively support its own named tags, the driver will create a list of tags based on information that is specific to the driver. An example of these two conditions is as follows.

1. If a **data acquisition system** supports its own local tag database, the driver will use the tag names found in the device to build the server's **OPC tags**.

2. If an **Ethernet I/O system** supports detection of its own available **I/O module types**, the driver will automatically generate OPC tags in the server. These tags will be based on the types of I/O modules plugged into the Ethernet I/O rack.

The OPC tags generated are given meaningful names in the OPC Server and are based on the variables imported. These tags are also placed in meaningful tag groups to provide a structured and manageable interface to the tags. The end result is a well-organized OPC Server project that directly reflects the variable import file.

See Also: [Tag Hierarchy](#) and [Import File-To-Server Name Conversions](#).

Generating Tag Database While Preserving Previously Generated Tag Databases

Under certain circumstances, multiple imports into the server are required to import all tags of interest. This is the case with importing VersaPro System variables and non-System variables into the same OPC Server project. In the Database Creation dialog under Device Properties, click on the selection **Perform the following action**. The options available are "Delete on create," "Overwrite as necessary," "Do not overwrite," and "Do not overwrite, log error". After the first OPC Server import/database creation is done, check that the action is set to "Do not overwrite" or "Do not overwrite, log error" for future imports. This will allow tags to be imported without deleting or overwriting tags that were previously imported.

Tag Hierarchy

The tags created via Automatic Tag Generation follow a specific hierarchy.

The root level groups (or subgroup levels of the group specified in **Add generated tags to the following group**) are determined by the variable addresses (i.e. R, G, M, etc.) referenced. For example, every variable that is of address type "R", will be placed in a root level group called "R". Each array tag is provided in its own subgroup of the parent group. The name of the array subgroup provides a description of the array. For instance an array R10[6] defined in the import file would have a subgroup name "R10_x"; x signifies dimension 1 exists.

Tags in "R10_x" Group

Tag Name	Tag Address	Comment
R10_x	R10[6]	Full array
R10_10	R10	Array element 1
R10_11	R11	Array element 2
R10_12	R12	Array element 3
R10_13	R13	Array element 4
R10_14	R14	Array element 5
R10_15	R15	Array element 6

Symbolic variable tags (PACSystems only) are placed in a group called **Symbolic**. Symbolic variable arrays are not broken out into individual element tags and will not be placed in a separate group. A single array tag will be generated for each symbolic variable array in the Symbolic group. For example, if a 2x3 array of symbolic variables named "MySymbolicArray" is defined in the import file, then a single tag with name "MySymbolicArray" and address "!MySymbolicArray"[2][3] would be generated in the Symbolic group.

See Also: [Symbolic Variables](#)

Import File-to-Server Name Conversions

Leading Underscores, Percents, Pound, and Dollar Signs

Leading underscores (_) in tag names will be replaced with **U_**. This is required since the server does not accept tag/group names beginning with an underscore.

Leading percents (%) in tag names will be replaced with **P_**. This is required since the server does not accept tag/group names beginning with a percent sign.

Leading pound signs (#) in tag names will be replaced with **PD_**. This is required since the server does not accept tag/group names beginning with a pound sign.

Leading dollar signs (\$) in tag names will be replaced with **D_**. This is required since the server does not accept tag/group names beginning with a dollar sign.

Invalid Characters in Name

The only characters allowed in the server tag name are A-Z, a-z, 0-9, and underscore (_). As mentioned above, a tag name cannot begin with an underscore. All other invalid characters encountered will be converted to a sequence of characters that are valid. Below is a table showing the invalid character and the sequence of characters that it is replaced with when encountered in the import file variable name.

Invalid Character	Replace With
\$	D_
%	P_
+	PL_
-	M_
#	PD_
@	A_
<	L_
>	G_
=	E_

Long Names

The GE Fanuc Ethernet driver is limited to 31 character group and tag names. Therefore, if a tag name exceeds 31 characters, it must be clipped. Names are clipped as follows.

Non-Array

1. Determine a 5-digit Unique ID for this tag.
2. Given a tag name: ThisIsALongTagNameAndProbablyExceeds31
3. Clip tag at 31: ThisIsALongTagNameAndProbablyEx
4. Room is made for the Unique ID: ThisIsALongTagNameAndProba#####
5. Insert this id: ThisIsALongTagNameAndProba00000

Array

1. Determine a 5-digit Unique ID for this array.
2. Given an array tag name: ThisIsALongTagNameAndProbablyExceeds31_23
3. Clip tag at 31 while holding on to the element values: ThisIsALongTagNameAndPr_23
4. Room is made for the Unique ID: ThisIsALongTagName#####_23
5. Insert this id: ThisIsALongTagName00001_23

Importing VersaPro Tags

The device driver uses files generated from VersaPro called Shared Name Files (SNF) to generate the tag database. Certain aspects of the Automatic Tag Database Generation process are specific to the application from which variables are imported. The following topics are VersaPro specific. In order to import tags from an application other than VersaPro, refer to [Automatic Tag Database Generation](#) to see if the application is supported.

How do I create a VersaPro variable import file (*.SNF)?

See [VersaPro Import Preparation: VersaPro Steps](#)

How do I configure the OPC Server to use this import file for Automatic Tag Database

Generation?

See [VersaPro Import Preparation: OPC Server Steps](#)

How do I import System Variables since they are not included with All Variables?

See [Generating Tag Database While Preserving Previously Generated Tag Databases](#)

How do I highlight variables in VersaPro?

See [Highlighting VersaPro Variables](#)

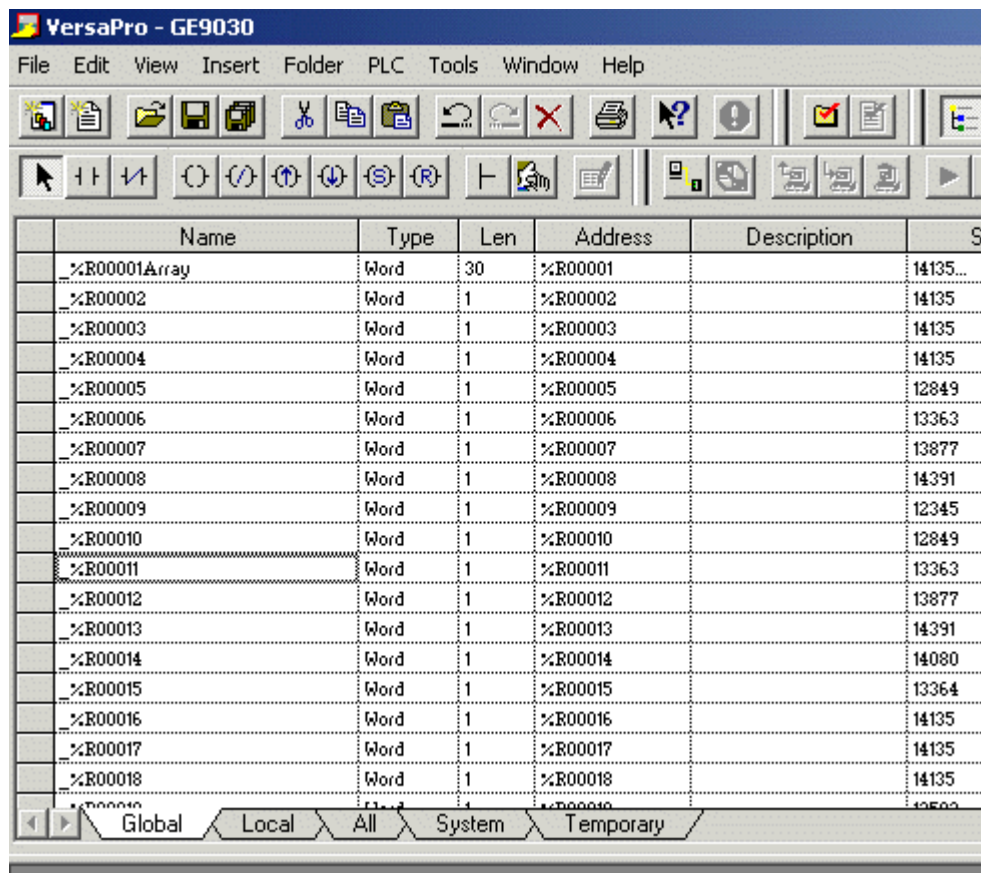
How are VersaPro array variables imported?

See [VersaPro Array Tag Import](#)

VersaPro Import Preparation: VersaPro Steps

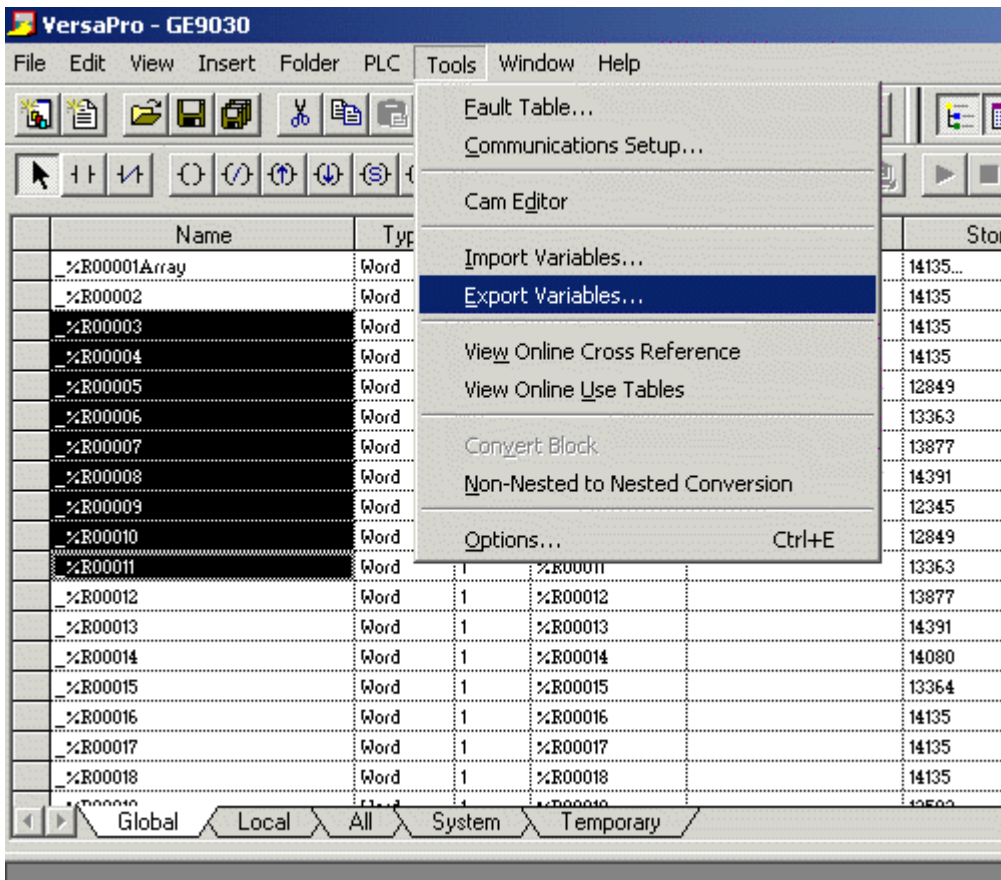
1. Open the VersaPro project containing the tags (variables) that will be ported to the OPC Server.
2. If the **Variable Declaration Table** is not already open, click **View | Variable Declaration Table**.
3. Next, specify which group the tags of interest belong to. The default groups available are **Global**, **Local**, **All**, **System** and **Temporary**.

Note: The group **All** does not include the variables from the System group. Multiple Imports, or multiple SNF files, are required to import System variables and Global/Local/All variables.



4. Click on the group's tab to bring its variables to the front. Then, highlight the tags of interest using either the mouse or menu.

5. Next, click **Tools | Export Variables**.



6. When prompted, select **Shared Name File (*.snf)** and specify a name. VersaPro will export the project's contents into this SNF file.

See Also: [Highlighting VersaPro Variables](#)

VersaPro Import Preparation: OPC Server Steps

1. Open Device Properties in the device of interest for which tags will be generated.
2. Select the **Database Settings** tab.
3. Enter or browse for the location of the **VersaPro SNF file** newly created.
4. Select the **Database Creation** tab and utilize as instructed above.
5. The OPC Server will state in the event log that it is attempting to perform a tag import, when finished, it will state that the tag import has completed.

Note: All variable exported out of VersaPro will appear in the OPC Server in the layout discussed in [Tag Hierarchy](#).

See Also: [Variable Import Settings](#)

Highlighting VersaPro Variables

Highlighting variables in VersaPro can be performed in following ways.

Single Variable Selecting

Left-click on a variable of interest while pressing CTRL.

Pick-n-Choose

N/A.

Selecting a Range of Variables

Left-click on the first variable in the range of interest. Then, press SHIFT while left-clicking on the last variable in the range. All variables in the range will be highlighted.

Selecting All Variables

Left-click on a variable within the group of interest in the Variable Declaration Table. The variable chosen is irrelevant. Next, click **Edit | Select All**. All variables will be highlighted with that group.

VersaPro Array Tag Import

Variables in VersaPro have a **Length** specification. Length is the number of elements for the given array variable. In the device driver, this element count can be used to create tags in two ways. The first is to create an array tag with data in a row x column format. The second is an expanded group of tags, Length in number. The following applies for variables with a Length > 1.

Array Tag

Since VersaPro arrays are 1-dimensional, the number of columns is always 1. Thus, an array tag would have the following syntax:

```
<array variable>[#rows = Length].
```

This single array tag would retrieve Length elements starting at the base address defined in <array variable>. The data will come back formatted in array form in order to be used in HMIs that support arrays.

Individual Elements

Element tags are simply the base address + element number. This has the following form, where $n = \text{Length} - 1$.

```
<array variable><base address + 0>  
<array variable><base address + 1>  
<array variable><base address + 2>  
...  
<array variable><base address + n>
```

These tags are not array tags; they are just the reference tags for the <array variable>. Think of it as a listing of all the addresses being referenced in the <array variable>.

Example

Variable Imported:
MyArrayTag, Length = 10, Address = R1

Result as Array Tag:
MyArrayTag [10]

Result as Individual Elements:
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10

Note: Variables of type BIT array cannot be accessed as an array tag, only as an expanded group of tags.

Importing Cimplicity Logic Developer Tags

The device driver uses user-generated ASCII text files from Logic Developer to generate the tag database. Certain

aspects of the Automatic Tag Database Generation process are specific to the application from which variables are imported. The following topics are Logic Developer specific. In order to import tags from an application other than Logic Developer, refer to [Automatic Tag Database Generation](#) to see if the application is supported.

How do I create a Logic Developer variable import file (*.TXT)?

See [Cimplicity Logic Developer Import Preparation: Logic Developer Steps](#)

How do I configure the OPC Server to use this import file for Automatic Tag Database Generation?

See [Cimplicity Logic Developer Import Preparation: OPC Server Steps](#)

How do I highlight variables in Logic Developer?

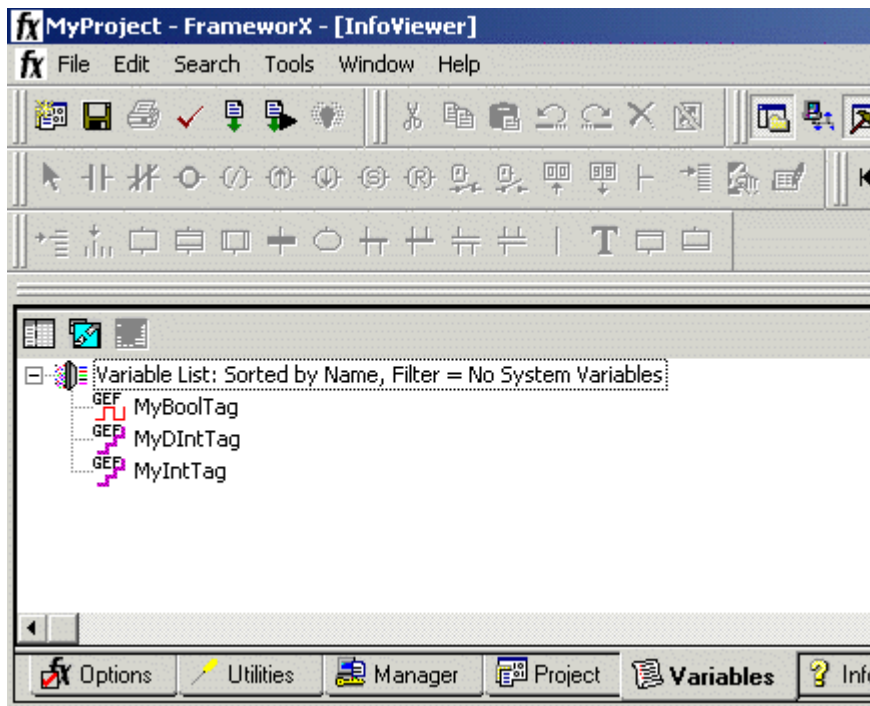
See [Highlighting Cimplicity Logic Developer Variables](#)

How are Logic Developer array variables imported?

See [Cimplicity Logic Developer Array Tag Import](#)

Cimplicity Logic Developer Import Preparation: Logic Developer Steps

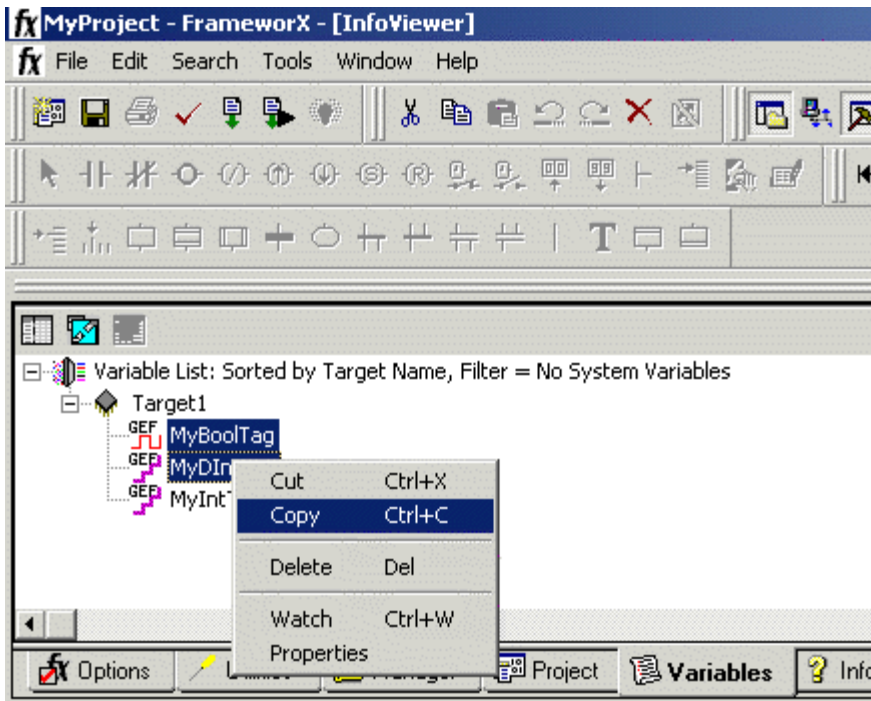
1. Open the **FrameworkX** project containing the tags (variables) that will be ported to the OPC Server.
2. Open the **Navigator** window by pressing **Shift+F4**.
3. Click on the **Variables** tab and select **Variable List View**.



Note: Each FrameworkX project contains one or more targets, which essentially is the device on which the application will run. Variables are created on the target-level, so a target of interest must be chosen in order to specify the variables that will be exported. In order for the target variables to be imported, the variables must have **GE FANUC PLC** as the **Data Source**. Verify this by left-clicking on the variable and then looking at its Data Source property in the **Inspector** window. Internal variables will not be imported.

5. Sort the variables by target. To do so, right-click on the **Variable List** and click **Sort | Target**.
6. Highlight the tags of interest in the Target of interest.

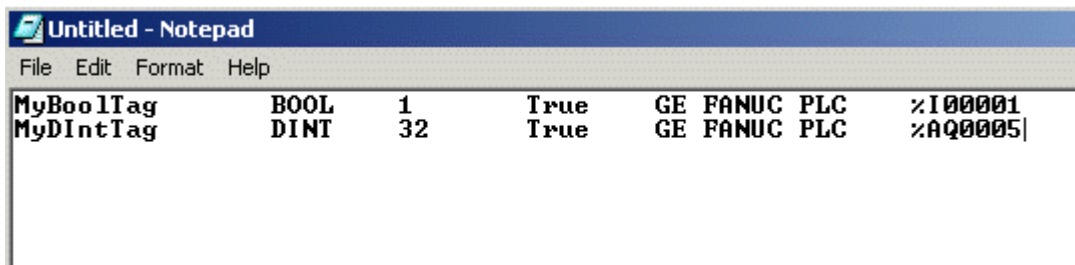
7. Next, click **Edit | Copy**.



8. With the highlighted variables now copied to the Clipboard, open a word processing program like Notepad or Wordpad.

9. In Notepad/Wordpad, click **Edit | Paste**.

10. The variables on the Clipboard will now be pasted to the document, TAB delimited. Do not modify the contents. Modifications may cause the import to fail.



11. Save the text document with the **TXT extension (.txt) in ANSI form**.

12. The variables are now contained within the text document and can be imported into the OPC Server.

Cimplicity Logic Developer Import Preparation: OPC Server Steps

1. Open Device Properties in the device of interest for which tags will be generated.
2. Select the **Database Settings** tab.
3. Enter or browse for the location of the **Logic Developer TXT file** newly created.
4. Select the **Database Creation** tab and utilize as instructed above.
5. The OPC Server will state in the event log that it is attempting to perform a tag import. When finished, it will state that the tag import has completed.

Note: All variables exported out of Logic Developer will appear in the OPC Server in the layout discussed in [Tag Hierarchy](#).

See Also: [Variable Import Settings](#)

Highlighting Cimplicity Logic Developer Variables

Variables in Logic Developer can be highlighted in the following ways.

Single Variable Selecting

Left-click on a variable of interest.

Pick-n-Choose

Left-click on the first variable of interest. Then, press CTRL while left-clicking on each successive variable of interest. Do this until all variables of interest are highlighted.

Selecting a Range of Variables

Left-click on the first variable in the range of interest. Then, press SHIFT while left-clicking on the last variable in the range. All variables in the range will be highlighted.

Selecting All Variables

Left-click on a variable within the target of interest in the Variable List View. The variable chosen is irrelevant. Next, click **Edit** | **Select All**. All variables will be highlighted within that target.

Cimplicity Logic Developer Array Tag Import

Array tags, or individual element breakdowns of array variables, are not supported when importing from Cimplicity Logic Developer.

Importing Proficy Logic Developer Tags

The device driver uses user-generated ASCII text files from Proficy Logic Developer to generate the tag database. Certain aspects of the Automatic Tag Database Generation process are specific to the application from which variables are imported. The following topics are specific to Proficy Logic Developer. In order to import tags from an application other than Proficy Logic Developer, refer to [Automatic Tag Database Generation](#) to see if the application is supported.

How do I create a Proficy Logic Developer variable import file (*.snf or *.csv)?

See [Proficy Logic Developer Import Preparation: Logic Developer Steps](#)

How do I configure the OPC Server to use this import file for Automatic Tag Database Generation?

See [Proficy Logic Developer Import Preparation: OPC Server Steps](#)

How do I highlight variables in Proficy Logic Developer?

See [Highlighting Proficy Logic Developer Variables](#)

How are Proficy Logic Developer array variables imported?

See [Proficy Logic Developer Array Tag Import](#)

Proficy Logic Developer Import Preparation: Logic Developer Steps

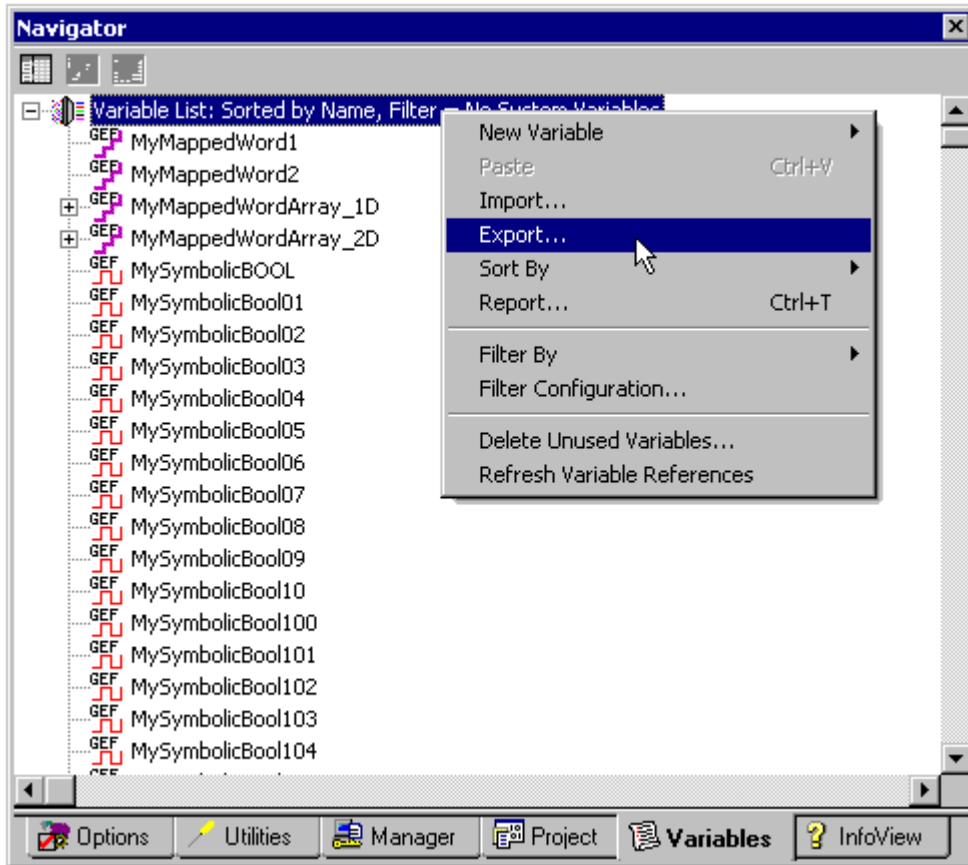
1. Open the Proficy Logic Developer project containing the tags (variables) that will be ported to the OPC Server.
2. Open the **Navigator** window by pressing Shift-F4 (unless it is already open).
3. Click on the **Variables** tab in order to bring the project's variables to the front. Then, click **Variable List View**.

Note: Each Logic Developer project contains one or more targets. A target is essentially the device on which the

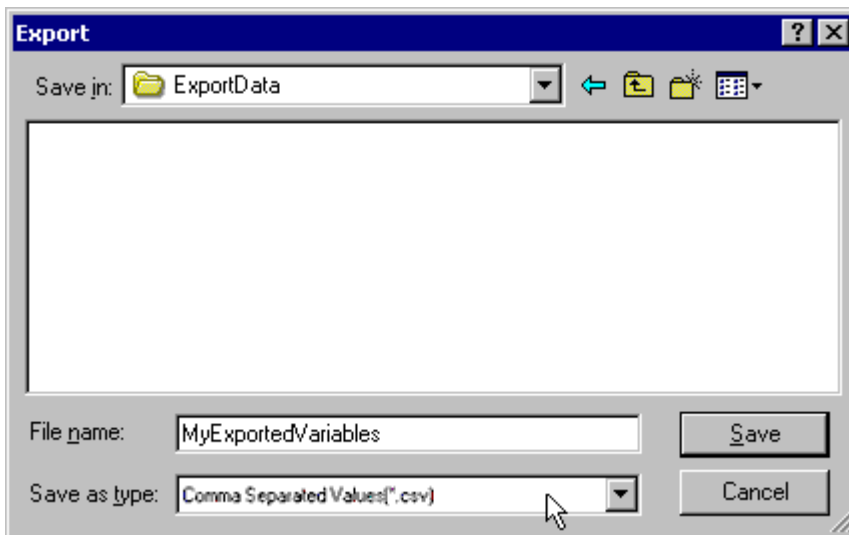
application will run. Because variables are created on the target-level, the target of interest must be specified beforehand in order to specify the variables to export. Furthermore, the target variables must have **GE FANUC PLC** chosen as the **Data Source** in order to be imported. Verify these settings by left-clicking on the variable and then looking at its Data Source properties in the **Inspector** window. Internal variables will not be imported.

4. Next, sort the variables by target by right-clicking on the **Variable List** header and selecting **Sort | Target**.

5. To export all of the variables, right-click on the **Variable List** and click **Export....**



6. To export only selected variables, **Highlight** the tags of interest in the target of interest. Then, right-click on one of the selected variables and click **Export....**



Proficy Logic Developer Import Preparation: OPC Server Steps

1. Open up the Device Properties in the device of interest for which tags will be generated.
2. Select the [Variable Import Settings](#) tab.
3. Enter or browse for the location of the export file newly created (*.snf or *.csv).
4. Select the **Database Creation** tab and then click **Auto Create** to import variables. Alternatively, other settings in that dialog can be selected to automatically create the database later.
5. The OPC Server will state in the event log that it is attempting to perform a tag import. When finished, it will state that the tag import has completed. All variables exported out of Logic Developer will appear in the OPC Server in the layout discussed in [Tag Hierarchy](#).

Highlighting Proficy Logic Developer Variables

Variables in Logic Developer can be highlighted in the following ways.

Single Variable Selecting

Left-click on a variable of interest.

Pick-n-Choose

Left-click on a variable of interest. Then, press CTRL while left-clicking on each successive variable of interest. Do this until all variables of interest are highlighted.

Selecting a Range of Variables

Left-click on the first variable in the range of interest. Then, press SHIFT while left-clicking on the last variable in the range. All variables in the range will be highlighted.

Selecting All Variables

Left-click on a variable within the target of interest in the Variable List View. The variable chosen is irrelevant. Next, click **Edit | Select All**. All variables will be highlighted with that target.

Proficy Logic Developer Array Tag Import

Arrays of referenced variables and arrays of symbolic variables will be imported differently.

Referenced Variable Arrays

Arrays of referenced variables will be imported as described in [VersaPro Array Tag Import](#). A group will be created for each array. Each group will contain a single array tag, plus a number of tags addressing the individual array

elements.

Symbolic Variable Arrays

A single array tag will be generated for each symbolic variable array in the import file. All symbolic variable array tags will be placed in the Symbolic group along with all other symbolic variable tags. The driver will not generate tags for BOOL and STRING symbolic variable arrays.

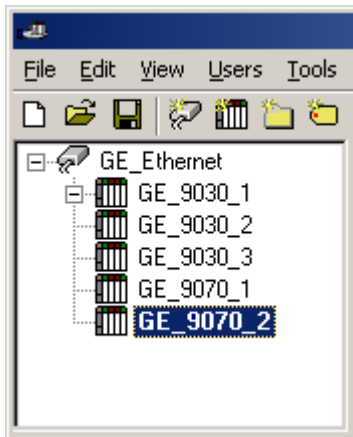
Note on importing array elements (index): When importing array elements (e.g., MyArrayTag[1,2]) into the driver, the characters '[' and ']' are replaced with '{' and '}' respectively for internal reasons ('[' and ']' are reserved by this driver for array notation). At runtime, however, while communicating with the PLC, the replacement will be reversed by the driver to comply with the standard GE syntax.

See Also: [Note on addressing array elements \(index\)](#) and [Symbolic Variables](#).

Optimizing Your GE Fanuc Ethernet Communications

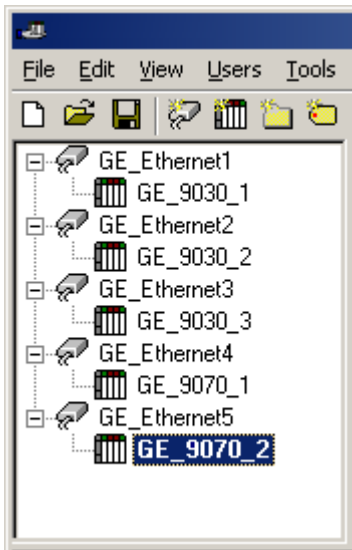
The GE Fanuc Ethernet driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the GE Fanuc Ethernet driver is fast, there are a couple of guidelines that can be used in order to control and optimize the application and gain maximum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

Our server refers to communications protocols like GE Fanuc 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 GE controller from which data will be collected. While this approach to defining the application will provide a high level of performance, it won't take full advantage of the GE Fanuc 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 GE Fanuc Ethernet channel. In this configuration, the driver must move from one device to the next as quickly as possible in order to gather information at an effective rate. As more devices are added or more information is requested from a single device, the overall update rate begins to suffer.

If the GE Fanuc Ethernet driver could only define one single channel, then the example shown above would be the only option available; however, the GE Fanuc Ethernet driver can define up to 256 channels. Using multiple channels distributes the data collection workload by simultaneously issuing multiple requests to the network. An example of how the same application may appear when configured using multiple channels to improve performance is shown below.



Each device has now been defined under its own channel. In this new configuration, a single path of execution is dedicated to the task of gathering data from each device. If the application has 256 or fewer devices, it can be optimized exactly how it is shown here.

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

Note for PACSystem Models: It is recommended that referenced (mapped) variables and symbolic variables be placed on separate devices. For more information, refer to [Symbolic Variables](#).

Block Size, available on each device, can also affect the performance of the GE Fanuc Ethernet driver. Block Size refers to the number of bytes that may be requested from a device at one time. To refine the performance of this driver, configure Block Size to one of the following settings: 32, 64, 128, 256, 512, 1024, or 2048 bytes. Depending on the specific GE device, the block size setting can have a dramatic effect on the application. We recommend a default value of 256 bytes. If the application consists of large requests for consecutively ordered data, try increasing the block size setting.

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
BCD	Two byte packed BCD Value range is 0-9999. Behavior is undefined for values

	beyond this range.
LBCD	Four byte packed BCD Value range is 0-99999999. Behavior is undefined for values beyond this range.
Float	32 bit floating point value. The driver interprets two consecutive registers as a floating point value by making the second register the high word and the first register the low word.
String	Null terminated ASCII string Support includes HiLo LoHi byte order selection.

Address Descriptions

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

[PACSystems Addressing](#)

[Symbolic Variables](#)

[311](#)

[313](#)

[331](#)

[341](#)

[350](#)

[360](#)

[731](#)

[732](#)

[771](#)

[772](#)

[781](#)

[782](#)

[GE OPEN](#)

[Horner OCS](#)

[VersaMax Addressing](#)

[Advanced Addressing](#)

[Special Items](#)

PACSystems Addressing

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

Device Address	Range	Data Type	Access
Discrete Inputs	I00001 to I32768 I00001 to I32761 (every 8th bit) I00001 to I32753 (every 8th bit)	Boolean Word , Short, BCD Byte	Read/Write
Discrete Outputs	Q00001 to Q32768 Q00001 to Q32761 (every 8th bit) Q00001 to Q32753 (every 8th bit)	Boolean Word , Short, BCD Byte	Read/Write
Discrete Globals	G00001 to G7680 G00001 to G7673 (every 8th bit) G00001 to G7665 (every 8th bit)	Boolean Word , Short, BCD Byte	Read/Write
Internal Coils	M00001 to M32768 M00001 to M32761 (every 8th bit) M00001 to M32753 (every 8th bit)	Boolean Word , Short, BCD Byte	Read/Write
Temporary Coils	T0001 to T1024 T0001 to T1017 (every 8th bit)	Boolean Word , Short, BCD	Read/Write

	T0001 to T1009 (every 8th bit)	Byte	
Status References (Same for SA, SB, SC)	S001 to S128 S001 to S121 (every 8th bit) S001 to S113 (every 8th bit)	Boolean Word , Short, BCD Byte	Read Only
Register References	R1.0 to R32640.15 R00001 to R32640 R00001 to R32639	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Inputs	AI1.0 to AI32640.15 AI00001 to AI32640 AI00001 to AI32639	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ32640.15 AQ00001 to AQ32640 AQ00001 to AQ32639	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Program Block Registers*	P1.0 to P8192.15 P00001 to P8192 P00001 to P8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
SubProgram Block Registers**	L00001<SUBPRGM> to L8192<SUBPRGM> L00001<SUBPRGM> to L8191<SUBPRGM> L00001<SUBPRGM>.0 to L8192<SUBPRGM>.15	Word , Short, BCD DWord, Long, LBCD, Float Boolean	Read/Write
Bulk Memory	W1.0 to W65534.15 W00001 to W65534 W00001 to W65533	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write

*Program specified in [PLC Settings](#).

**The SubProgram name/Block name <SUBPRGM> must match the name assigned in the PLC. This name can be found in VersaPro under Block Properties.

See Also: [Symbolic Variables](#)

Symbolic Variables

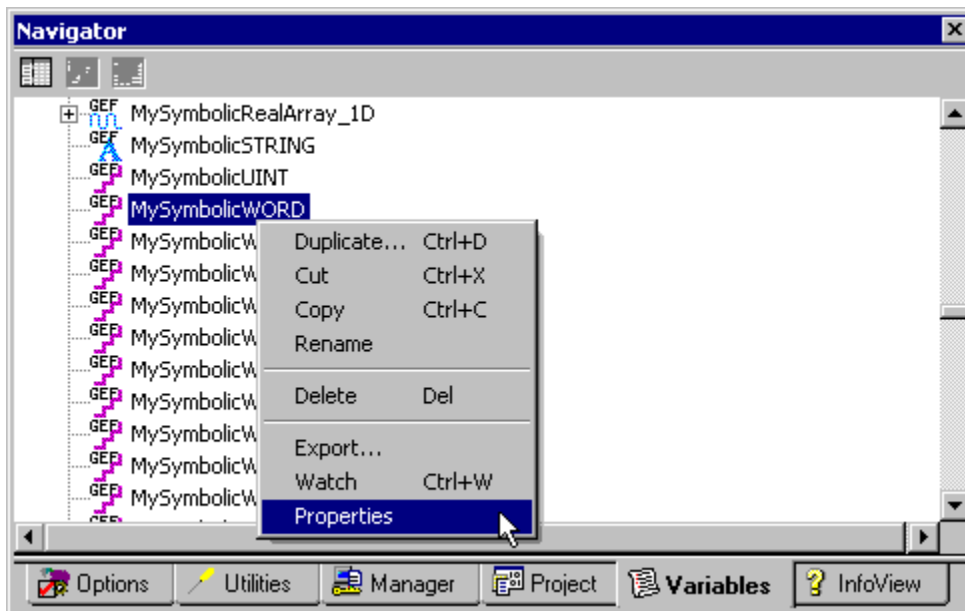
A variable is a named storage space for data in the PLC. A **Reference Variable** is a variable that is mapped to a specific I/O or internal register location in the PLC (I00001, R00100, etc). A **Symbolic Variable** is a variable that is not mapped to a specific register. The PLC will allocate memory for symbolic variables, as needed, from its managed memory area. This driver has the ability to read and write to symbolic variables defined in the PLC.

Note: Symbolic variables are available in PAC Systems PLCs only.

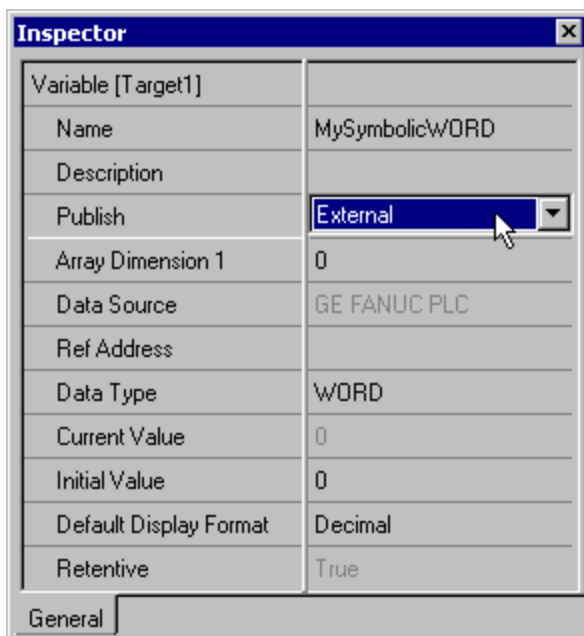
Create a Symbolic Variable

1. Load the PLC's current configuration project into Proficy Logic Developer.
2. Open the **Navigator View** by pressing Shift-F4 (if it isn't already open).
3. Click on the **Variables** tab. To create a new variable, right-click on the **Variables View** and select **New Variable**.

Note: To edit an existing variable, right-click on it and then select **Properties**.



4. In either case, the variable's **Properties Inspector** dialog will appear as shown below.



Note: The variable shown above is symbolic because the **Ref Address** property is left blank. The variable is named "MySymbolicWORD" and has a data type of Word. An array of variables can be created by setting the **Array Dimension 1** property to a value greater than zero. The **Array Dimension 2** property will appear in the variable inspector if Array Dimension 1 is set. A single dimension array is defined when Array Dimension 2 is left at zero. The total number of array elements is limited by the **Maximum Bytes Per Request** setting in Device Properties. For more information, refer to [Device Setup](#).

Important: In order for a symbolic variable to be visible to this driver, **Publish** must be set to **External**.

5. Download the modified project to the PLC.

6. In order to access a symbolic variable with the OPC Server, a tag will need to be created that has an address that references the variable by name, preceded by the "!" character.

Examples

1. !MySymbolicArray1D [4] – addresses the symbolic variable named "MySymbolicArray1D", which is assumed to have its Array Dimension 1 property set to 4.
2. !MySymbolicArray1D {3} – addresses the index (3) of the above 1D symbolic array variable "MySymbolicArray1D"
3. !MySymbolicArray2D [3][4] – addresses the symbolic variable named "MySymbolicArray2D", which is assumed to have its Array Dimension 1 property set to 3, and its "Array Dimension 2" property set to 4.
4. !MySymbolicArray2D {2,3} – addresses the index (2,3) of the above 2D symbolic array variable "MySymbolicArray2D."

Note on addressing array elements (index): As shown above, the syntax for addressing array elements uses '{' and '}'. These characters are specific to KEPServerEX. The correct GE syntax to access the array elements instead uses '[' and ']'. The appropriate conversion will be done internally by the driver at runtime while communicating with the PLC. However, for addressing purposes due to internal reasons ('[' and ']' are reserved by this driver for array notation), users will need to use the characters '{' and '}'.

Important: The tag must be assigned a data type that is compatible with the Data Type property of the symbolic variable, and dimensions of array tags must be the same as the dimensions of the variable. These tag properties can only be validated during run time.

Compatible Data Types*

Native Type (device)	Compatible Types (server)
Bool	Bool (no arrays)
Byte	Byte, Char
Int	Word, Short
UInt	Word, Short
Word	Word, Short
Dint	DWord, Long
DWord	DWord, Long
Real	Float
String	String (no arrays)

*The driver will assign a default type of Word.

Be sure to specify an appropriate data type override for each dynamic tag. (Dynamic tags are defined in the client application only, and are created in the server on demand by the client. See the OPC Server help for details.) The data type of a dynamic tag can be specified by appending one of the following strings to the item name: "@Boolean", "@Byte", "@Char", "@Short", "@Word", "@Long", "@DWord", "@Float", or "@String".

Example

ItemName = "Channel1.Device1.!MySymbolicWordVariable @DWord"

Performance Considerations

The driver will attempt to optimize performance by reading blocks of symbolic variable memory. (It is generally faster to read a single large block of data, where only the first and last few bytes are needed to update two tags for example, then it is to perform two separate reads for few bytes needed for each tag.) The amount of symbolic variable memory read per request will be limited by the **Maximum Bytes Per Request** device property setting. It is generally advantageous to maximize this setting. However, in some cases, if the variables most frequently read are widely scattered in the controller's managed memory area, performance may be increased by reducing this setting. Unfortunately, the mapping of symbolic variable data in the controller's memory is not under the user's control. Mapping depends largely on when the variables are added to the configuration project. If performance is a primary concern, some experimentation with this setting is recommended.

This driver will also attempt to optimize performance by performing multi-item writes. The number of tags included in a request is limited by the Maximum Bytes Per Request device property setting. However, unlike the block reads described above, the location of the variables in memory is of little consequence. Writes can further be optimized with the **Write Optimizations** setting in Channel Properties.

If the driver finds that the variable named in a tag's address does not exist in the controller, or if its data type or array dimension does not match the tag's, its OPC quality will be set to BAD. The driver will continue to check the controller for the named variable should a new configuration be downloaded at some point. Do not use such tags in the client applications as they will remain invalid.

311 Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I001 to I512 I001 to I505 (every 8th bit) I001 to I497 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q001 to Q512 Q001 to Q505 (every 8th bit) Q001 to Q497 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Globals	G0001 to G1280 G0001 to G1273 (every 8th bit) G0001 to G1265 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M0001 to M1024 M0001 to M1017 (every 8th bit) M0001 to M1009 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S01 to S32 S01 to S25 (every 8th bit) S01 to S17 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R512.15 R001 to R512 R001 to R511	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Inputs	AI1.0 to AI64.15 AI01 to AI64 AI01 to AI63	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ032.15 AQ001 to AQ032 AQ001 to AQ031	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		

*Default data type of Boolean becomes Byte when an array specification is given.

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)

313 Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I001 to I512 I001 to I505 (every 8th bit) I001 to I497 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q001 to Q512 Q001 to Q505 (every 8th bit)	Boolean Byte	Read/Write

	Q001 to Q497 (every 8th bit)	Word, Short, BCD	
Discrete Globals	G0001 to G1280 G0001 to G1273 (every 8th bit) G0001 to G1265 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M0001 to M1024 M0001 to M1017 (every 8th bit) M0001 to M1009 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S01 to S32 S01 to S25 (every 8th bit) S01 to S17 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R1024.15 R001 to R1024 R001 to R1023	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Inputs	AI1.0 to AI64.15 AI01 to AI64 AI01 to AI63	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ032.15 AQ001 to AQ032 AQ001 to AQ031	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		

*Default data type of Boolean becomes Byte when an array specification is given.

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)

331 Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I001 to I512 I001 to I505 (every 8th bit) I001 to I497 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q001 to Q512 Q001 to Q505 (every 8th bit) Q001 to Q497 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Globals	G0001 to G1280 G0001 to G1273 (every 8th bit) G0001 to G1265 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M0001 to M1024 M0001 to M1017 (every 8th bit) M0001 to M1009 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S01 to S32 S01 to S25 (every 8th bit) S01 to S17 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R2048.15 R0001 to R2048	Boolean Word , Short, BCD	Read/Write

	R0001 to R2047	DWord, Long, LBCD, Float	
Analog Inputs	AI1.0 to AI128.15 AI001 to AI128 AI001 to AI127	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ64.15 AQ01 to AQ64 AQ01 to AQ63	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		

*Default data type of Boolean becomes Byte when an array specification is given.

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)

341 Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I001 to I512 I001 to I505 (every 8th bit) I001 to I497 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q001 to Q512 Q001 to Q505 (every 8th bit) Q001 to Q497 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Globals	G0001 to G1280 G0001 to G1273 (every 8th bit) G0001 to G1265 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M0001 to M1024 M0001 to M1017 (every 8th bit) M0001 to M1009 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S01 to S32 S01 to S25 (every 8th bit) S01 to S17 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R9999.15 R0001 to R9999 R0001 to R9998	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Inputs	AI1.0 to AI1024.15 AI0001 to AI1024 AI0001 to AI1023	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ256.15 AQ0001 to AQ256 AQ0001 to AQ255	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		

*Default data type of Boolean becomes Byte when an array specification is given.

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)
[Array Support](#)

350 Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I0001 to I2048 I0001 to I2041 (every 8th bit) I0001 to I2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q0001 to Q2048 Q0001 to Q2041 (every 8th bit) Q0001 to Q2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Globals	G0001 to G1280 G0001 to G1273 (every 8th bit) G0001 to G1265 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M0001 to M4096 M0001 to M4089 (every 8th bit) M0001 to M4081 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S01 to S32 S01 to S25 (every 8th bit) S01 to S17 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R9999.15 R0001 to R9999 R0001 to R9998	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Inputs	AI1.0 to AI2048.15 AI0001 to AI2048 AI0001 to AI2047	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ512.15 AQ001 to AQ512 AQ001 to AQ511	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		

*Default data type of Boolean becomes Byte when an array specification is given.

Advanced Addressing

[Bit Access to Registers](#)
[Default Data Type Override](#)
[String Access to Registers](#)
[Array Support](#)

360 Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I0001 to I2048 I0001 to I2041 (every 8th bit) I0001 to I2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q0001 to Q2048 Q0001 to Q2041 (every 8th bit) Q0001 to Q2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write

Discrete Globals	G0001 to G1280 G0001 to G1273 (every 8th bit) G0001 to G1265 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M0001 to M4096 M0001 to M4089 (every 8th bit) M0001 to M4081 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S01 to S32 S01 to S25 (every 8th bit) S01 to S17 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R32768.15 R00001 to R32768 R00001 to R32767	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Inputs	AI1.0 to AI2048.15 AI0001 to AI2048 AI0001 to AI2047	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ512.15 AQ001 to AQ512 AQ001 to AQ511	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		

*Default data type of Boolean becomes Byte when an array specification is given.

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)

731 Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I001 to I512 I001 to I505 (every 8th bit) I001 to I497 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q001 to Q512 Q001 to Q505 (every 8th bit) Q001 to Q497 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Globals	G0001 to G1280 G0001 to G1273 (every 8th bit) G0001 to G1265 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M0001 to M2048 M0001 to M2041 (every 8th bit) M0001 to M2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S001 to S128 S001 to S121 (every 8th bit) S001 to S113 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R16384.15 R00001 to R16384 R00001 to R16383	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write

Program Block Registers***	P00001 to P16384 P00001 to P16383 P1.0 to P16384.15	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
SubProgram Block Registers**	L00001<SUBPRGM> to L16384<SUBPRGM> L00001<SUBPRGM> to L16383<SUBPRGM> L00001<SUBPRGM>.0 to L16384<SUBPRGM>.15	Word , Short, BCD DWord, Long, LBCD, Float Boolean	Read/Write
Analog Inputs	AI1.0 to AI8192.15 AI0001 to AI8192 AI0001 to AI8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ8192.15 AQ0001 to AQ8192 AQ0001 to AQ8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items		

*Default data type of Boolean becomes Byte when an array specification is given.

**The SubProgram name/Block name <SUBPRGM> must match the name assigned in the PLC. This name can be found in VersaPro under Block Properties.

***Program specified in [PLC Settings](#).

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)

732 Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I001 to I512 I001 to I505 (every 8th bit) I001 to I497 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q001 to Q512 Q001 to Q505 (every 8th bit) Q001 to Q497 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Globals	G0001 to G1280 G0001 to G1273 (every 8th bit) G0001 to G1265 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M0001 to M2048 M0001 to M2041 (every 8th bit) M0001 to M2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S001 to S128 S001 to S121 (every 8th bit) S001 to S113 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R16384.15 R00001 to R16384 R00001 to R16383	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write

Program Block Registers***	P00001 to P16384 P00001 to P16383 P1.0 to P16384.15	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
SubProgram Block Registers**	L00001<SUBPRGM> to L16384<SUBPRGM> L00001<SUBPRGM> to L16383<SUBPRGM> L00001<SUBPRGM>.0 to L16384<SUBPRGM>.15	Word , Short, BCD DWord, Long, LBCD, Float Boolean	Read/Write
Analog Inputs	AI1.0 to AI8192.15 AI0001 to AI8192 AI0001 to AI8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ8192.15 AQ0001 to AQ8192 AQ0001 to AQ8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		

*Default data type of Boolean becomes Byte when an array specification is given.

**The SubProgram name/Block name <SUBPRGM> must match the name assigned in the PLC. This name can be found in VersaPro under Block Properties.

***Program specified in [PLC Settings](#).

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)

771 Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I0001 to I2048 I0001 to I2041 (every 8th bit) I0001 to I2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q0001 to Q2048 Q0001 to Q2041 (every 8th bit) Q0001 to Q2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Globals	G0001 to G7680 G0001 to G7673 (every 8th bit) G0001 to G7665 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M0001 to M4096 M0001 to M4089 (every 8th bit) M0001 to M4081 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S001 to S128 S001 to S121 (every 8th bit) S001 to S113 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R16384.15 R00001 to R16384 R00001 to R16383	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write

Program Block Registers***	P00001 to P16384 P00001 to P16383 P1.0 to P16384.15	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
SubProgram Block Registers**	L00001<SUBPRGM> to L16384<SUBPRGM> L00001<SUBPRGM> to L16383<SUBPRGM> L00001<SUBPRGM>.0 to L16384<SUBPRGM>.15	Word , Short, BCD DWord, Long, LBCD, Float Boolean	Read/Write
Analog Inputs	AI1.0 to AI8192.15 AI0001 to AI8192 AI0001 to AI8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ8192.15 AQ0001 to AQ8192 AQ0001 to AQ8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		

*Default data type of Boolean becomes Byte when an array specification is given.

**The SubProgram name/Block name <SUBPRGM> must match the name assigned in the PLC. This name can be found in VersaPro under Block Properties.

***Program specified in [PLC Settings](#).

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)

772 Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I0001 to I2048 I0001 to I2041 (every 8th bit) I0001 to I2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q0001 to Q2048 Q0001 to Q2041 (every 8th bit) Q0001 to Q2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Globals	G0001 to G7680 G0001 to G7673 (every 8th bit) G0001 to G7665 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M0001 to M4096 M0001 to M4089 (every 8th bit) M0001 to M4081 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S001 to S128 S001 to S121 (every 8th bit) S001 to S113 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R16384.15 R00001 to R16384 R00001 to R16383	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write

Program Block Registers***	P00001 to P16384 P00001 to P16383 P1.0 to P16384.15	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
SubProgram Block Registers**	L00001<SUBPRGM> to L16384<SUBPRGM> L00001<SUBPRGM> to L16383<SUBPRGM> L00001<SUBPRGM>.0 to L16384<SUBPRGM>.15	Word , Short, BCD Boolean	Read/Write
Analog Inputs	AI1.0 to AI8192.15 AI0001 to AI8192 AI0001 to AI8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ8192.15 AQ0001 to AQ8192 AQ0001 to AQ8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		

*Default data type of Boolean becomes Byte when an array specification is given.

**The SubProgram name/Block name <SUBPRGM> must match the name assigned in the PLC. This name can be found in VersaPro under Block Properties.

***Program specified in [PLC Settings](#).

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)

781 Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I00001 to I12288 I00001 to I12281 (every 8th bit) I00001 to I12273 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q00001 to Q12288 Q00001 to Q12281 (every 8th bit) Q00001 to Q12273 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Globals	G0001 to G7680 G0001 to G7673 (every 8th bit) G0001 to G7665 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M00001 to M12288 M00001 to M12281 (every 8th bit) M00001 to M12273 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S001 to S128 S001 to S121 (every 8th bit) S001 to S113 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R16384.15 R00001 to R16384 R00001 to R16383	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write

Program Block Registers***	P00001 to P16384 P00001 to P16383 P1.0 to P16384.15	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
SubProgram Block Registers**	L00001<SUBPRGM> to L16384<SUBPRGM> L00001<SUBPRGM> to L16383<SUBPRGM> L00001<SUBPRGM>.0 to L16384<SUBPRGM>.15	Word , Short, BCD DWord, Long, LBCD, Float Boolean	Read/Write
Analog Inputs	AI1.0 to AI8192.15 AI0001 to AI8192 AI0001 to AI8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ8192.15 AQ0001 to AQ8192 AQ0001 to AQ8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		

*Default data type of Boolean becomes Byte when an array specification is given.

**The SubProgram name/Block name <SUBPRGM> must match the name assigned in the PLC. This name can be found in VersaPro under Block Properties.

***Program specified in [PLC Settings](#).

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)

782 Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I00001 to I12288 I00001 to I12281 (every 8th bit) I00001 to I12273 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q00001 to Q12288 Q00001 to Q12281 (every 8th bit) Q00001 to Q12273 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Globals	G0001 to G7680 G0001 to G7673 (every 8th bit) G0001 to G7665 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M00001 to M12288 M00001 to M12281 (every 8th bit) M00001 to M12273 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S001 to S128 S001 to S121 (every 8th bit) S001 to S113 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R16384.15 R00001 to R16384 R00001 to R16383	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write

Program Block Registers***	P00001 to P16384 P00001 to P16383 P1.0 to P16384.15	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
SubProgram Block Registers**	L00001<SUBPRGM> to L16384<SUBPRGM> L00001<SUBPRGM> to L16383<SUBPRGM> L00001<SUBPRGM>.0 to L16384<SUBPRGM>.15	Word , Short, BCD DWord, Long, LBCD, Float Boolean	Read/Write
Analog Inputs	AI1.0 to AI8192.15 AI0001 to AI8192 AI0001 to AI8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ8192.15 AQ0001 to AQ8192 AQ0001 to AQ8191	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		

*Default data type of Boolean becomes Byte when an array specification is given.

**The SubProgram name/Block name <SUBPRGM> must match the name assigned in the PLC. This name can be found in VersaPro under Block Properties.

***Program specified in [PLC Settings](#).

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)

GE OPEN Addressing

The GE OPEN model selection has been provided to supply support for any GE SNP compatible device that is not currently listed in the standard model selection menu. The ranges of data for each data type have been expanded to allow a wide range of GE PLCs to be addressed. Although the address ranges shown here may exceed the specific PLC's capability, the driver will respect all messages from the PLC regarding memory range limits.

Note: The default data types for dynamic tags are shown in **bold**.

Device Address	Range	Data Type*	Access
Discrete Inputs	I0001 to I32768 I0001 to I32761 (every 8th bit) I0001 to I32753 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q00001 to Q32768 Q00001 to Q32761 (every 8th bit) Q00001 to Q32753 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Globals	G00001 to G32768 G00001 to G32761 (every 8th bit) G00001 to G32753 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M00001 to M32768 M00001 to M32761 (every 8th bit) M00001 to M32753 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T00001 to T32768 T00001 to T32761 (every 8th bit) T00001 to T32753 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References	S00001 to S32768 S00001 to S32761 (every 8th bit)	Boolean Byte	Read Only

(Same for SA, SB, SC)	S00001 to S32753 (every 8th bit)	Word, Short, BCD	
Register References	R1.0 to R32768.15 R00001 to R32768 R00001 to R32767	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Program Block Registers***	P00001 to P32768 P00001 to P32767	Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
SubProgram Block Registers**	L00001<SUBPRGM> to L32768<SUBPRGM> L00001<SUBPRGM> to L32767<SUBPRGM>	Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Inputs	AI1.0 to AI32768.15 AI00001 to AI32768 AI00001 to AI32767	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ32768.15 AQ00001 to AQ32768 AQ00001 to AQ32767	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		
Bulk Memory	W1.0 to W65534.15 W00001 to W65534 W00001 to W65533	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write

*Default data type of Boolean becomes Byte when an array specification is given.

**The SubProgram name/Block name <SUBPRGM> must match the name assigned in the PLC. This name can be found in VersaPro under Block Properties.

***Program specified in [PLC Settings](#).

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)

Horner OCS Addressing

The Horner OCS model selection has been provided to supply support for Horner Operator Control Stations. The ranges of data for each data type have been expanded to allow a wide range of Horner OCS devices to be addressed. Although the address ranges shown here may exceed the specific device's capability, the driver will respect all messages from the OCS regarding memory range limits.

Note: The default data types for dynamic tags are **bold**.

Device Address	Range	Data Type*	Access
Discrete Inputs	I0001 to I2048 I0001 to I2041 (every 8th bit) I0001 to I2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q00001 to Q20488 Q00001 to Q2041 (every 8th bit) Q00001 to Q2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M00001 to M2048 M00001 to M2041 (every 8th bit) M00001 to M2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T00001 to T2048 T00001 to T2041 (every 8th bit) T00001 to T2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Register References	R1.0 to R32768.15	Boolean	Read/Write

	R00001 to R32768 R00001 to R32767	Word , Short, BCD DWord, Long, LBCD, Float	
Analog Inputs	AI1.0 to AI512.15 AI00001 to AI512 AI00001 to AI511	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ512.15 AQ00001 to AQ512 AQ00001 to AQ511	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write

*Default data type of Boolean becomes Byte when an array specification is given.

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)

VersaMax Addressing

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

Device Address	Range	Data Type*	Access
Discrete Inputs	I001 to I2048 I001 to I2041 (every 8th bit) I001 to I2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Outputs	Q001 to Q2048 Q001 to Q2041 (every 8th bit) Q001 to Q2033 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Discrete Globals	G0001 to G1280 G0001 to G1273 (every 8th bit) G0001 to G1265 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Internal Coils	M0001 to M1024 M0001 to M1017 (every 8th bit) M0001 to M1009 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Temporary Coils	T001 to T256 T001 to T249 (every 8th bit) T001 to T241 (every 8th bit)	Boolean Byte Word, Short, BCD	Read/Write
Status References (Same for SA, SB, SC)	S01 to S32 S01 to S25 (every 8th bit) S01 to S17 (every 8th bit)	Boolean Byte Word, Short, BCD	Read Only
Register References	R1.0 to R2048.15 R001 to R2048 R001 to R2047	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Inputs	AI1.0 to AI128.15 AI01 to AI128 AI01 to AI127	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Analog Outputs	AQ1.0 to AQ128.15 AQ001 to AQ128 AQ001 to AQ127	Boolean Word , Short, BCD DWord, Long, LBCD, Float	Read/Write
Special Items (PLC Status, Time, etc)	See Also: Special Items .		

*The default data type, Boolean, changes to Byte when an array specification is given.

Advanced Addressing

[Bit Access to Registers](#)

[Default Data Type Override](#)

[String Access to Registers](#)

[Array Support](#)**Advanced Addressing**

The following advanced addressing topics are discussed below:

[Bit Access to Registers](#)[Default Data Type Override](#)[String Access to Registers](#)[Array Support](#)**Bit Access to Registers**

Register based device types such as R, P, L, AI, and AQ can be accessed at the bit level by appending a bit number to the register address. The valid bit number range is 0 to 15. The bit number must be preceded by either a "." or "a : " .

Examples

1. R100.12 allows Read/Write access to bit 12 of Register 100.
2. L50SUB1.3 allows Read/Write access to bit 3 of local register L50 in subprogram SUB1.

Note: All device addresses can be prefixed with a % sign if needed, such as %R100. This can aid in converting from other OPC servers or communications drivers.

Default Data Type Override

The default data types for each device type are shown in [PACSystems Addressing](#). The defaults can be overridden by appending data type indicators to the device address. The possible data type indicators are as follows.

Indicators	Data type
F	Float
S	Short
L	Long
M	String
(BCD)	BCD

Examples

Address	Description
R100 F	Access R100 as a floating point value
R300 L	Access R300 as a long
R400-R410 M	Access R400-R410 as a string with a length of 22 bytes. (LoHi byte order is assumed.)
P100 S	Access P100 as a short
L200SUB1-L210SUB1 M	Access L200-L210 in subprogram SUB1 as a string with a length of 22 bytes. (LoHi byte order is assumed.)

Note: There must be a space between the register number and the data type indicator.

String Access to Registers

Register space can be accessed as string data by appending the "M" data indicator. The length of the string is based on how the device address reference is entered. Each register addressed can contain 2 characters. The byte order of characters in registers can be specified by appending an optional "H" for HiLo or "L" for LoHi after the "M" data indicator. If no byte order is specified, LoHi order is assumed. Here are some examples:

Examples

Address	Description
R100-R200 M	Access Register R100 as string with a length of 202 bytes. (LoHi byte order is assumed.)
R400 M	Access Register R400 as a string with a length of 4 bytes. (LoHi byte order is assumed.)

R405-R405 M	Access Register R405 as a string with a length of 2 bytes. (LoHi byte order is assumed.)
L100SUB1-L100SUB1 M	Access local register L100 in subprogram SUB1 as a string with a length of 2 bytes. (LoHi byte order is assumed.)
R100-R200 M H	Access Register R100 as string with a length of 202 bytes. HiLo byte order is explicitly specified.
R100-R200 M L	Access Register R100 as string with a length of 202 bytes. LoHi byte order is explicitly specified.

Notes: The maximum string length is 255 bytes. For HiLo byte ordering, the string "AB" would be stored in a register as 0x4142. For LoHi byte ordering, the string "AB" would be stored in a register as 0x4241. There must be a space between the "M" data type indicator and the byte order indicator.

Array Support

An array is a collection of contiguous elements of a given data type. Arrays are supported by the following data types: byte, word, short, DWord, long, float, and char.

Maximum Array Size for Referenced or Mapped Variables	512 DWords (longs, floats), 1024 words (shorts) or 2048 bytes and chars for a total of 16384 bits
Maximum Array Size for Symbolic Variables	512 DWords (longs, floats), 1024 words (shorts) or 1024 bytes and chars for a total of 16384 bits

Note: The number of usable bytes is directly reflective of the block size.

Examples

Address	Address Breakdown
G1 [4] includes the following byte addresses Note: G25 indicates the fourth byte beginning at bit 25.>	G1, G9, G17, G25 1 row implied = 4 bytes 4 x 8 (byte) = 32 total bits
R16 [3][4] includes the following Word addresses:	R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27 3 rows x 4 columns = 12 words 12 x 16 (word) = 192 total bits
P10 [5] includes the following Word addresses:	P10, P11, P12, P13, P14 1 rows x 5 columns = 5 words 5 x 16 (word) = 80 total bits

Special Items

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

Note: Though not visible in the server configuration, tags with these addresses are automatically created and can be browsed by the OPC client. They will be found under the <Channel Name>.<Device Name>._InternalTags group. If the OPC client does not support browsing or a non-OPC client is being used, users can create their own static and dynamic tags using these addresses.

Device Address	Range	Data Type	Access
_OVERSWEEP	Oversweep flag; meaningful only when constant sweep mode is active. 1 = Constant Sweep value exceeded 0 = No oversweep condition exists	Boolean	Read Only
_CONSWEEP	Constant sweep mode. 1 = Constant Sweep Mode active 0 = Constant Sweep Mode not active	Boolean	Read Only
_NEWFT	PLC Fault entry since last read. 1 = PLC Fault Table has changed since last read by the server	Boolean	Read Only

	0 = PLC Fault Table is unchanged since last read by the server		
_NEWIOFT	IO Fault entry since last read. 1 = IO Fault Table has changed since last read by the server 0 = IO Fault Table is unchanged since last read by the server	Boolean	Read Only
_FTSTATUS	PLC Fault entry present 1 = One or more fault entries in PLC Fault Table 0 = PLC Fault table is empty	Boolean	Read Only
_IOSTATUS	IO Fault entry present 1 = One or more fault entries in IO Fault Table 0 = IO Fault Table is empty	Boolean	Read Only
_PROGATTACH	Programmer attachment in system flag. 1 = Programmer attachment found in system 0 = No programmer attachment found in system	Boolean	Read Only
_ENSWITCH	Front panel ENABLED/DISABLED switch setting. 1 = Outputs disabled 0 = Outputs enabled	Boolean	Read Only
_RUN	Front panel RUN/STOP switch setting. 1 = Run 0 = Stop		
_OEMPROT	OEM protected bit. 1 = OEM protection in effect 0 = No OEM protection	Boolean	Read Only
_SPRGCHG	Program changed flag. 1 = Program changed 0 = No program change (90-70 release 2.x and later)	Boolean	Read Only
_STATE	Current PLC State 0 = Run I/O enabled 1 = Run I/O disabled 2 = Stop I/O disabled 3 = CPU stop faulted 4 = CPU halted 5 = CPU suspended 6 = Stop I/O enabled	Byte, Char	Read Only
_PROGNUM	Control program number SNP Master is currently logged into. -1 = SNP Master is not logged into a task 0 = SNP Master is logged into task 0	Byte, Char	Read Only
_PRIVLEVEL	Current privilege level of server for accessing memory in PLC CPU (0-4)	Byte, Char	Read Only
_SWEEPTIME	Time taken by the last complete sweep for the main control program task. The value returned is a whole number representing the number of tenths of a millisecond. The Sweep Time in milliseconds is 1/10 of this value. For example, if _SWEEPTIME is 123, the actual Sweep Time is 12.3 milliseconds.	Word, Short	Read Only
_SNPID	CPU Controller ID.	String	Read Only
_PROGNAME	Name of control program task in the PLC CPU.	String	Read Only
_SNUM_PROGS	Number of control program tasks defined for the PLC CPU.	Byte, Char	Read Only
_SPROG_FLAGS	Bit flags indicating which control programs tasks have programmers	Byte, Char	Read Only

	attached. Bit 0 = control program 1.		
_TIME	Internal time and date of the PLC CPU. Format: SSMMHHDDMoYYDw where SS = Seconds (0-59) MM == Minutes (0-59) HH = Hours (0-23) DD = Day (1- 31) Mo = Month (1-12) YY = Year (0-99) Wd = Day of week (1-Sunday, 2-Monday, etc)	String	Read Only
_SECOND	PLC Time: Seconds (0-59)	Byte, Char	Read Only
_MINUTE	PLC Time: Minutes (0-59)	Byte, Char	Read Only
_HOUR	PLC Time: Hours (0-23)	Byte, Char	Read Only
_DAY	PLC Date: Day (1- 31)	Byte, Char	Read Only
_MONTH	PLC Date: Month (1-12)	Byte, Char	Read Only
_YEAR	PLC Date: Year (0-99)	Byte, Char	Read Only
_DOW	PLC Date: Day of week (1-Sunday, 2-Monday, etc)	Byte, Char	Read Only

Error Descriptions

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

Address Validation

[Missing address](#)

[Device address '<address>' contains a syntax error](#)

[Address '<address>' is out of range for the specified device or register](#)

[Device address '<address>' is not supported by model '<model name>'](#)

[Data Type '<type>' is not valid for device address '<address>'](#)

[Device address '<address>' is Read Only](#)

[Array size is out of range for address '<address>'](#)

[Array support is not available for the specified address: '<address>'](#)

Device Status Messages

[Device '<device name>' is not responding](#)

[Unable to write to '<address>' on device '<device name>'](#)

[Attempting to reacquire symbolic variable mapping information for device '<device name>'. Cannot read or write symbolic variables until complete](#)

Driver Error Messages

Winsock initialization failed (OS Error = n)

[Winsock V1.1 or higher must be installed to use the GE Fanuc Ethernet device driver](#)

[Device '<device name>' returned error code <error num> reading n byte\(s\) starting at <address>](#)

[The symbolic variable name in tag address '<address>' does not exist in current configuration of device '<device name>'](#)

[The address length of the symbolic variable addressed by '<tag name>' on device '<device>' is bigger than the configured block size for this device, Deactivating tag](#)

[The array size of address '<address>' on device '<device name>' must be the same as symbolic variable's array size of \[rows\]\[cols\]](#)

[The data type of the symbolic variable addressed by '<address>' on device '<device name>' is not compatible with tag](#)

Automatic Tag Database Generation Messages

[Unable to generate a tag database for device <device name>. Reason: Low memory resources](#)

[Unable to generate a tag database for device <device name>. Reason: Import file is invalid or corrupt](#)

[Database Error: Tag '<orig. tag name>' exceeds 31 characters. Tag renamed to '<new tag name>'](#)

[Database Error: Array tags '<orig. tag name><dimensions>' exceed 31 characters. Tags renamed to '<new tag name><dimensions>'](#)

[Database Error: Data type '<type>' for tag '<tag name>' not found in import file. Setting to default](#)

[Database Error: Data type '<type>' for tag '<tag name>' is currently not supported. Tag not created](#)

[Database Error: Logic Developer Variable Arrays are currently not supported. Array Tag\(s\) '<array tag name>' not created](#)

[Database Error: No Reference Address found for tag '<tag name>' in import file. Tag not created](#)

[Database Error: Only variables with Data Source '<data source name>' are imported. Data Source '<data source name>' is not supported. Tag '<tag name>' not created](#)

[Database Error: Data type '<type>' arrays are currently not supported. Tag '<array tag name>' not created](#)

Address Validation Error Messages

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

Address Validation

[Missing address](#)

[Device address '<address>' contains a syntax error](#)

[Address '<address>' is out of range for the specified device or register](#)

[Device address '<address>' is not supported by model '<model name>'](#)

[Data Type '<type>' is not valid for device address '<address>'](#)

[Device address '<address>' is Read Only](#)

[Array size is out of range for address '<address>'](#)

[Array support is not available for the specified address: '<address>'](#)

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 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.

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 via DDE references a location that is beyond the range of supported locations for the device.

Solution:

Verify the address is correct; if it is not, re-enter it in the client application.

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

Error Type:

Warning

Possible Cause:

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

Solution:

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

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>' 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.

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

Error Type:

Warning

Possible Cause:

1. The number of elements in an array definition exceeds the range of elements supported by the address type.
2. The number of elements in an array definition exceeds the maximum number of data values that may be obtained from the device in a single request.

Solution:

Reduce the number of referenced elements in the array definition.

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.

Device Status Messages

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

Device Status Messages

[Device '<device name>' is not responding](#)

[Unable to write to '<address>' on device '<device name>'](#)

[Attempting to reacquire symbolic variable mapping information for device '<device name>'. Cannot read or write symbolic variables until complete](#)

Device '<device name>' 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.

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.

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.

Attempting to reacquire symbolic variable mapping information for device '<device name>'. Cannot read or write symbolic variables until complete

Error Type:

Warning

Possible Cause:

A new configuration is being downloaded to the device or the configuration has been lost (possibly due to power failure).

Solution:

The driver should automatically recover once a new device configuration has been restored.

Driver Error Messages

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

Driver Error Messages

Winsock initialization failed (OS Error = n)

[Winsock V1.1 or higher must be installed to use the GE Fanuc Ethernet device driver](#)

[Device '<device name>' returned error code <error num> reading n byte\(s\) starting at <address>](#)

[The symbolic variable name in tag address '<address>' does not exist in current configuration of device '<device name>'](#)

[The address length of the symbolic variable addressed by '<tag name>' on device '<device>' is bigger than the configured block size for this device. Deactivating tag](#)

[The array size of address '<address>' on device '<device name>' must be the same as symbolic variable's array size of \[rows\]\[cols\]](#)

[The data type of the symbolic variable addressed by '<address>' on device '<device name>' is not compatible with tag](#)

Winsock V1.1 or higher must be installed to use the GE Fanuc Ethernet device driver

Error Type:

Fatal

Possible Cause:

The version number of the Winsock DLL found on the system is less than 1.1.

Solution:

Upgrade Winsock to version 1.1 or higher.

Device '<device name>' returned error code <error num> reading n byte(s) starting at <address>

Error Type:

Error

Possible Cause:

An attempt has been made to read a location that does not exist.

Solution:

Review the address map for the device in question and make necessary adjustments in the client application.

The symbolic variable name in tag address '<address>' does not exist in current configuration of device '<device name>'

Error Type:

Warning

Possible Cause:

The named variable does not exist in the current device configuration.

Solution:

1. Verify that variable is defined in current device configuration.
2. Check the spelling of the variable name.

Note:

To optimize driver performance, it is recommended that all tags with invalid symbolic variable addresses be removed from the server configuration or not be used by a client application.

The address length of the symbolic variable addressed by '<tag name>' on device '<device>' is bigger than the configured block size for this device. Deactivating tag

Error Type:

Error

Possible Cause:

The requested tag is larger than the configured block size for this device. This can be caused by the name being too big or requested packet being too big.

Solution:

Increase the block size or change name size.

The array size of address '<address>' on device '<device name>' must be the same as symbolic variable's array size of [rows][cols]

Error Type:

Warning

Possible Cause:

The array dimensions given in the tag address, if any, are not the same as the array size of the referenced symbolic variable.

Solution:

Determine the array dimensions of the variable as currently defined in the device. Adjust the tag address to match.

The data type of the symbolic variable addressed by '<address>' on device '<device name>' is not compatible with tag

Error Type:

Warning

Possible Cause:

The native data type of the symbolic variable, as currently defined in the device, is not compatible with the data type of the tag.

Solution:

Adjust the data type of the tag to match the variable's native data type.

See Also:

[Symbolic Variables](#)

Automatic Tag Database Generation Messages

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

Automatic Tag Database Generation Messages

[Unable to generate a tag database for device <device name>. Reason: Low memory resources](#)

[Unable to generate a tag database for device <device name>. Reason: Import file is invalid or corrupt](#)

[Database Error: Tag '<orig. tag name>' exceeds 31 characters. Tag renamed to '<new tag name>'](#)

[Database Error: Array tags '<orig. tag name><dimensions>' exceed 31 characters. Tags renamed to '<new tag name><dimensions>'](#)

[Database Error: Data type '<type>' for tag '<tag name>' not found in import file. Setting to default](#)

[Database Error: Data type '<type>' for tag '<tag name>' is currently not supported. Tag not created](#)

[Database Error: Logic Developer Variable Arrays are currently not supported. Array Tag\(s\) '<array tag name>' not created](#)

[Database Error: No Reference Address found for tag '<tag name>' in import file. Tag not created](#)

Database Error: Only variables with Data Source '<data source name>' are imported. Data Source '<data source name>' is not supported. Tag '<tag name>' not created
Database Error: Data type '<type>' arrays are currently not supported. Tag '<array tag name>' not created

Unable to generate a tag database for device <device name>. Reason: Low memory resources

Error Type:

Warning

Possible Cause:

Memory required for database generation could not be allocated. The process is aborted.

Solution:

Close any unused applications and/or increase the amount of virtual memory. Then, try again.

Unable to generate a tag database for device <device name>. Reason: Import file is invalid or corrupt

Error Type:

Warning

Possible Cause:

The file specified as the Tag Import File (in the Database Settings device properties page) is a corrupt import file (*.snf or *.csv) or improperly formatted Logic Developer text file.

Solution:

Select a valid, properly formatted VersaPro/Logic Developer variable import file or retry the tag export process in the respective application to produce a new import file.

See Also:

[Automatic Tag Database Generation Preparation](#)

Database Error: Array tags '<orig. tag name><dimensions>' exceed 31 characters. Tags renamed to '<new tag name><dimensions>'

Error Type:

Warning

Possible Cause:

The name assigned to an array tag originates from the variable name in the import file. This name exceeds the 31-character limitation and will be renamed to one that is valid. <Dimensions> define the number of dimensions for the given array tag. XXX for 1 Dimension, XXX_YYY for 2 Dimensions. The number of X's and Y's approximates the number of elements for the respective dimensions. Since such an error will occur for each element, generalizing with XXX and YYY implies all array elements will be affected.

Solution:

None.

See Also:

[Import File-to-Server Name Conversions](#)

Database Error: Tag '<orig. tag name>' exceeds 31 characters. Tag renamed to '<new tag name>'

Error Type:

Warning

Possible Cause:

The name assigned to a tag originates from the variable name in the import file. This name exceeds the 31 character limitation and will be renamed to one that is valid.

Solution:

None.

See Also:

[Import File-to-Server Name Conversions](#)

Database Error: Data type '<type>' for tag '<tag name>' not found in import file. Setting to default

Error Type:

Warning

Possible Cause:

The definition of data type '<type>', for tag <tag name>, could not be found in the import file.

Solution:

This tag will take on the Default type for the given address type as assigned by the GE Fanuc Ethernet Driver.

Database Error: Logic Developer Variable Arrays are currently not supported. Array Tag (s) '<array tag name>' not created

Error Type:

Warning

Possible Cause:

Array tags of 1 or 2 dimensions originating from a Logic Developer import file, are not supported at this time. The array tag(s) were not automatically generated.

Solution:

For applicable tags, avoid using arrays in the Logic Developer projects.

Database Error: No Reference Address found for tag '<tag name>' in import file. Tag not created

Error Type:

Warning

Possible Cause:

Variables without a reference address cannot have a tag created since the reference address determines the tag's address. The tag was not automatically generated.

Solution:

Verify the <tag name> has a PLC as a data source and that reference address (PLC memory location) has been assigned to it.

Database Error: Only variables with Data Source '<data source name>' are imported. Data Source '<data source name>' is not supported. Tag '<tag name>' not created

Error Type:

Warning

Possible Cause:

In Logic Developer, variables can take on a data value from a number of sources. For use in the OPC Server, the source must be a GE Fanuc Ethernet PLC. The tag was not automatically generated.

Solution:

Verify the <tag name> has a PLC as a data source.

Database Error: Data type '<type>' arrays are currently not supported. Tag '<array tag name>' not created

Error Type:

Warning

Possible Cause:

Boolean or String array tags of 1 or 2 dimensions are not supported at this time.

Solution:

For Boolean array tags, individual array elements of the tag if specified in the import file will be generated. Furthermore, the driver will also automatically create individual elements for the array tag (except for bit within word type Boolean array tags).

Note:

For String array tags, neither the array tag or the individual elements will be generated. String data type is currently not supported by the driver. Thus, avoid using String data type if possible.

Index

- 3 -

311 Addressing	23
313 Addressing	23
331 Addressing	24
341 Addressing	25
350 Addressing	26
360 Addressing	26

- 7 -

731 Addressing	27
732 Addressing	28
771 Addressing	29
772 Addressing	30
781 Addressing	31
782 Addressing	32

- A -

Address	33, 39
Address '<address>' is out of range for the specified device or register	40
Address Descriptions	19
Address Validation Error Messages	40
Address Validation Errors	40
Addressing Notes	36
Advanced Addressing	36
Array size is out of range for address '<address>'	41
Automatic Tag Database Generation	5
Automatic Tag Database Generation Messages	44

- B -

BCD	18
block	3
block size	3
Boolean	18
Byte	18

- C -

communications	3
----------------	---

- D -

Data Type '<type>' is not valid for device address '<address>'	41
Data Types Description	18
Database Error	
Array tags '<orig. tag name><dimensions>' exceed 31 characters. Tags renamed to '<new tag name><dimensions>'	45
Data type '<type>' arrays are currently not supported. Tag '<array tag name>' not created	47
Datatype '<type>' for tag '<tag name>' not found in import file. Setting to Default Type '<type>'	46
Logic Developer Variable Arrays are currently not supported. Array Tag(s) '<array tag name>' not created	46
No Reference Address found for tag '<tag name>' in import file. Tag not created	46
Only variables with Data Source '<data source name>' are imported. Data Source '<data source name>' is not..	46
Tag '<orig. tag name>' exceeds 31 characters. Tag renamed to '<new tag name>'	45
device	39
Device '<device name>' not responding	42
Device '<device name>' returned error code <error num> reading in byte(s) starting at <address>	43
Device address '<address>' contains a syntax error	40
Device address '<address>' is not supported by model '<model name>'	41
Device address '<address>' is Read Only	41
Device ID	3
Device Status Messages	42
Driver Error Messages	43
DWord	18
- E -	
error	39, 42, 43
Error Descriptions	39

- F -

Float 18

- G -

GE OPEN Addressing 33

- H -

Highlighting LogicDeveloper Variables 13
 Highlighting Proficy Logic Developer Variables 16
 Highlighting VersaPro Variables 9
 Horner OCS Addressing 34

- I -

Import File-to-Server Name Conversions 6
 Importing LogicDeveloper Tags 10
 Importing Proficy Logic Developer Tags 13
 Importing VersaPro Tags 7

- L -

LBCD 18
 LogicDeveloper Array Tag Import 13
 LogicDeveloper Import Preparation
 LogicDeveloper Steps 11
 LogicDeveloper Import Preparation OPC Server
 Steps 12
 Long 18

- M -

Missing address 40

- N -

Network 3
 Nickname 5

- O -

Optimizing GE Fanuc Ethernet Communications 17
 Overview 3

- P -

PACSystems Addressing 19
 PLC Settings 5
 Proficy Logic Developer Array Tag Import 16
 Proficy Logic Developer Import Preparation: Logic
 Developer Steps 13
 Proficy Logic Developer Import Preparation: OPC
 Server Steps 16
 protocol 3

- S -

Short 18
 Special Items 37
 Symbolic Variables 20

- T -

Tag Hierarchy 6
 Target Name 5
 The address length of the symbolic variable
 addressed by '<tag name>' on device '<device>' is
 bigger 44
 The array size of address '<address>' on device
 '<device name>' must be the same as 44
 The data type of the symbolic variable addressed
 by '<address>' on device '<device name>' is not
 compatible with tag 44
 The symbolic variable name in tag address
 '<address>' does not exist in current configuration
 of device '<device name>' 43

- U -

Unable to generate a tag database for device
 <device name>. Reason
 Import file is invalid or corrupt 45
 Low memory resources 45
 Unable to write tag '<address>' on device '<device
 name>' 42

- V -

Variable Import Settings 4
VersaMax Addressing 35
VersaPro Array Tag Import 10
VersaPro Import Preparation OPC Server Steps
9
VersaPro Import Preparation VersaPro Steps 8

- W -

Winsock 39
Winsock V1.1 or higher must be installed to use
the GE Fanuc Ethernet device driver 43
Word 18