GE SNP Driver Help

Table of Contents

GE SNP Driver Help

Help version 1.026

CONTENTS

Overview

The GE SNP Driver provides an easy and reliable way to connect GE SNP controllers to OPC Client applications, including HMI, SCADA, Historian, MES, ERP and countless custom applications. This driver is intended for use with GE Programmable Logic Controllers.

Device Setup

Supported Devices

Series GE Micro
Series 90-30 311/313, 331/341, 350, 360
Series 90-70 731/732, 771/772, 781/782
GE OPEN Wide range model support

Communication Protocol

GE SNP

Supported Communication Parameters

Baud Rate: 300, 600, 1200, 2400, 9600 and 19200
Parity: Odd, None
Data Bits: 8
Stop Bits: 1

Device IDs

Series 90-30 PLCs support up to 6-character strings. For example, 1 and Ge3.
Series 90-70 PLCs support up to 7-character strings. For example, 1, Ge7 and Ge.

Note: For peer-to-peer communications, an empty string is a valid Device ID.

Flow Control

When using an RS232/RS485 converter, the type of flow control that is required will depend upon the needs of the converter. Some converters do not require any flow control whereas others require RTS flow. Consult the converter's documentation to determine its flow requirements. An RS485 converter that provides automatic flow control is recommended.

Note: When using the manufacturer's supplied communications cable, it is sometimes necessary to choose a flow control setting of **RTS** or **RTS Always** under the Channel Properties.

Automatic Tag Database Generation

For more information, refer to **GE SNP Variable Import Settings**.

Cabling

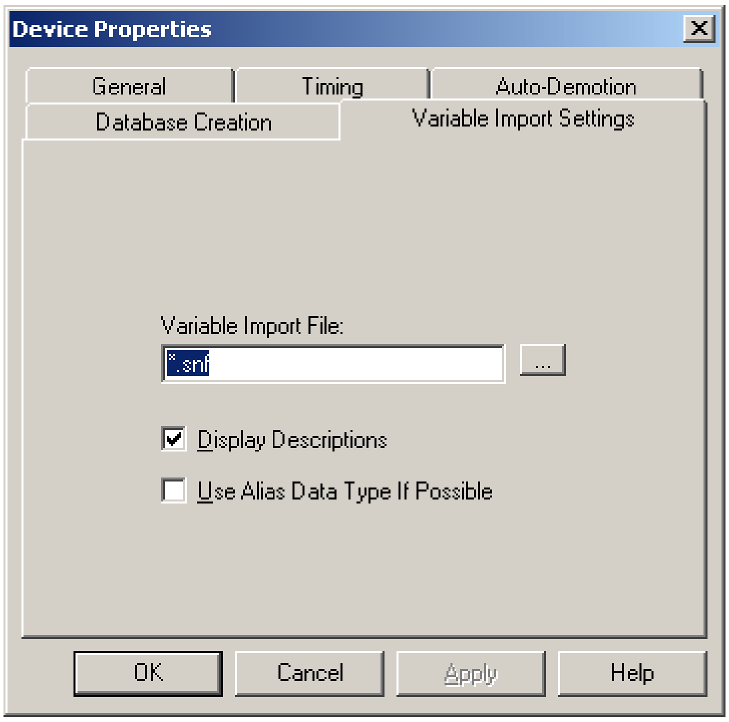
Follow the manufacturer's suggested cabling for the communications port and communications module.

Modem Setup

This driver supports modem functionality. For more information, please refer to the topic "Modem Support" in the OPC server Help documentation.

Variable Import Settings

The Variable Import Settings page is used to define the Variable Import File, as well as to specify tag description import and the use of an alias data type.



Variable Import File: *.snf

[✓] Display Descriptions

[ ] Use Alias Data Type If Possible

Descriptions of the parameters are as follows.

- **Variable Import File:** This parameter is used to 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 to have tag descriptions imported. If necessary, a description will be given to tags with long names stating the original tag name.
- **Use Alias Data Type If Possible:** Check this option to use the data type assigned to an alias tag in the import file. If the alias data type is incompatible with the source tag data type, the source tag data type will be used instead.

See Also: Automatic Tag Database Generation

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.
Double	64 bit floating point value

Automatic Tag Database Generation

The GE SNP Device Driver generates its tags offline based on variables imported from a text file. It is offline in the sense that a connection to the device is not required to generate tags. The text file (variables to import) can originate from either VersaPro or Cimplicity ME Logic Developer. For information on creating variable import files, refer to **Importing VersaPro Tags** and **Importing LogicDeveloper Tags**.

Overview

If the target device supports its own local tag database, the driver will read the device's tag information and then use that 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 specific to the driver. An example of these two conditions is as follows:

1. A data acquisition system that supports its own local tag database. The driver will use the tags names found in the device to build the OPC server's OPC tags.
2. An Ethernet I/O system that supports detection of I/O module type. The driver in this case will automatically generate OPC tags in the OPC server that are 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. 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 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 import tags without deleting or overwriting tags 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 referenced (such as R, G, M and so forth). 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 example, an array R10[6] defined in the import file would have a subgroup name "R10_x". X signifies that 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

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. The table below lists the invalid character and the sequence of characters that it will be replaced with when encountered in the import file variable name.

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

Long Names

The GE SNP 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_45_8
3. Clip tag at 31 while holding on to the element values: ThisIsALongTagNameAndPr_23_45_8
4. Room is made for the Unique ID: ThisIsALongTagName#####_23_45_8
5. Insert this ID: ThisIsALongTagName00001_23_45_8

Importing VersaPro Tags

The 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. 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**

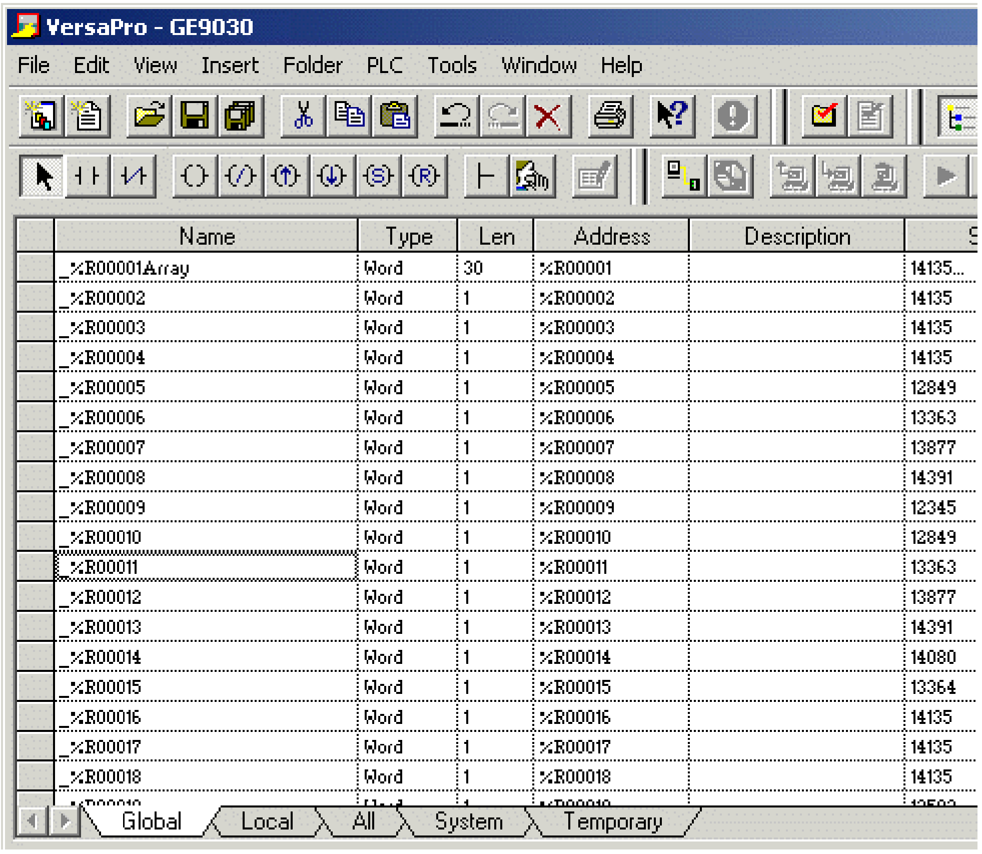
VersaPro Import Preparation: VersaPro Steps

1. Open the **VersaPro** project containing the tags (variables) that will be ported to OPC server.

2. Open the **Variable Declaration Table** by clicking **View** | **Variable Declaration Table**.

3. Next, decide which group the tags of interest belong to. The default groups available are **Global**, **Local**, **All**, **System** and **Temporary**.

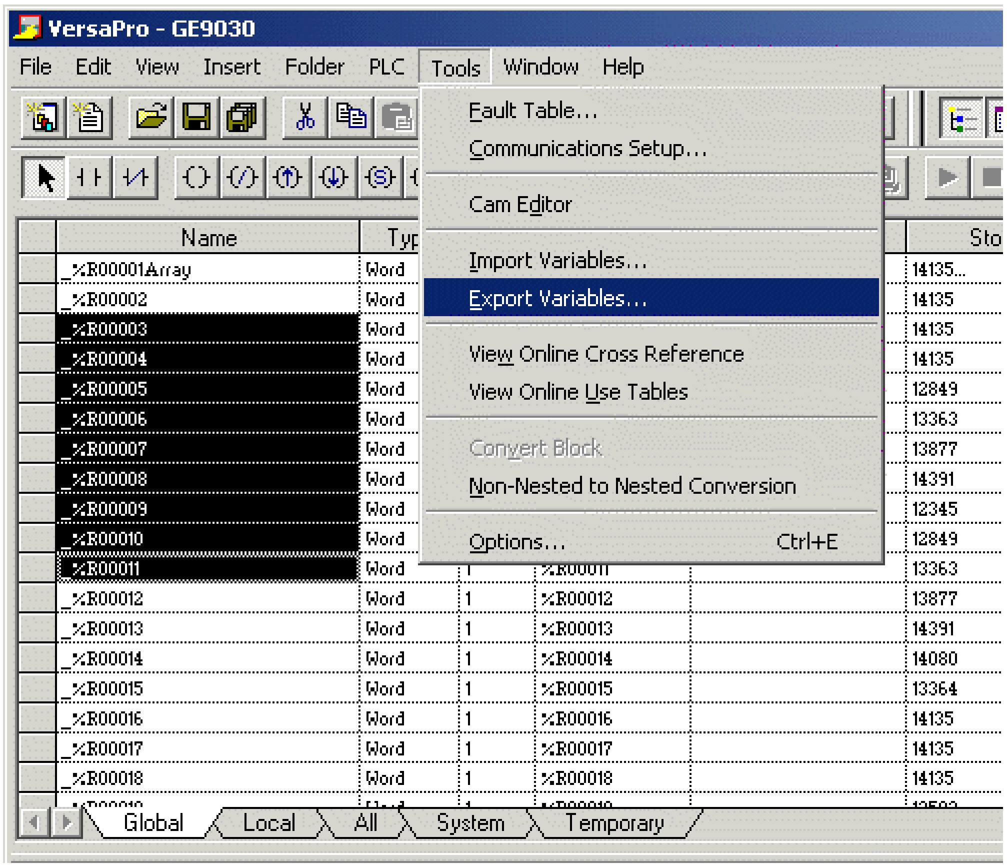
Note: All does not include the variables from the System group. To import System variables and Global/Local/All variables, multiple imports (that is, multiple SNF files) are required.



Name	Type	Len	Address	Description	S
_%R00001Array	Word	30	%R00001		14135...
_%R00002	Word	1	%R00002		14135
_%R00003	Word	1	%R00003		14135
_%R00004	Word	1	%R00004		14135
_%R00005	Word	1	%R00005		12849
_%R00006	Word	1	%R00006		13363
_%R00007	Word	1	%R00007		13877
_%R00008	Word	1	%R00008		14391
_%R00009	Word	1	%R00009		12345
_%R00010	Word	1	%R00010		12849
_%R00011	Word	1	%R00011		13363
_%R00012	Word	1	%R00012		13877
_%R00013	Word	1	%R00013		14391
_%R00014	Word	1	%R00014		14080
_%R00015	Word	1	%R00015		13364
_%R00016	Word	1	%R00016		14135
_%R00017	Word	1	%R00017		14135
_%R00018	Word	1	%R00018		14135

4. Click on the group's tab to bring its variables to the front. Next, highlight the tags of interest via the mouse or menu. For more information, refer to **Highlighting VersaPro Variables**.

5. Select **Tools** | **Export Variables**.



Name	Typ				Sto
_%R00001Array	Word				14135...
_%R00002	Word				14135
_%R00003	Word				14135
_%R00004	Word				14135
_%R00005	Word				12849
_%R00006	Word				13363
_%R00007	Word				13877
_%R00008	Word				14391
_%R00009	Word				12345
_%R00010	Word				12849
_%R00011	Word	1	%R00011		13363
_%R00012	Word	1	%R00012		13877
_%R00013	Word	1	%R00013		14391
_%R00014	Word	1	%R00014		14080
_%R00015	Word	1	%R00015		13364
_%R00016	Word	1	%R00016		14135
_%R00017	Word	1	%R00017		14135
_%R00018	Word	1	%R00018		14135

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

VersaPro Import Preparation: OPC Server Steps

1. Open up the **Device Properties** for 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 that was just 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. All variables 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 the following ways.

- **Single Variable Selecting:** Press the CTRL key and then left-click on the variable of interest.
- **Pick-n-Choose:** N/A.
- **Selecting a Range of Variables:** Start by left-clicking on the first variable in the range of interest. Then, press the SHIFT key while left-clicking on the last variable in the range. All variables in the range will be highlighted.
- **Selecting All Variables:** Start by left-clicking 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 within 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 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 information 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 return formatted in array form for use in HMIs that support arrays.

Individual Elements

Element tags are simply the base address + element number. This has the following form, where n = 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>. It may be thought of 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. They may only be accessed as an expanded group of tags.

Importing LogicDeveloper Tags

The 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 LogicDeveloper specific. For information on how to import tags from an application other than LogicDeveloper, refer to **Automatic Tag Database Generation** to see if the application is supported.

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

See **LogicDeveloper Import Preparation: LogicDeveloper Steps**

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

See **LogicDeveloper Import Preparation: OPC Server Steps**

How do I highlight variables in LogicDeveloper?

See **Highlighting LogicDeveloper Variables**

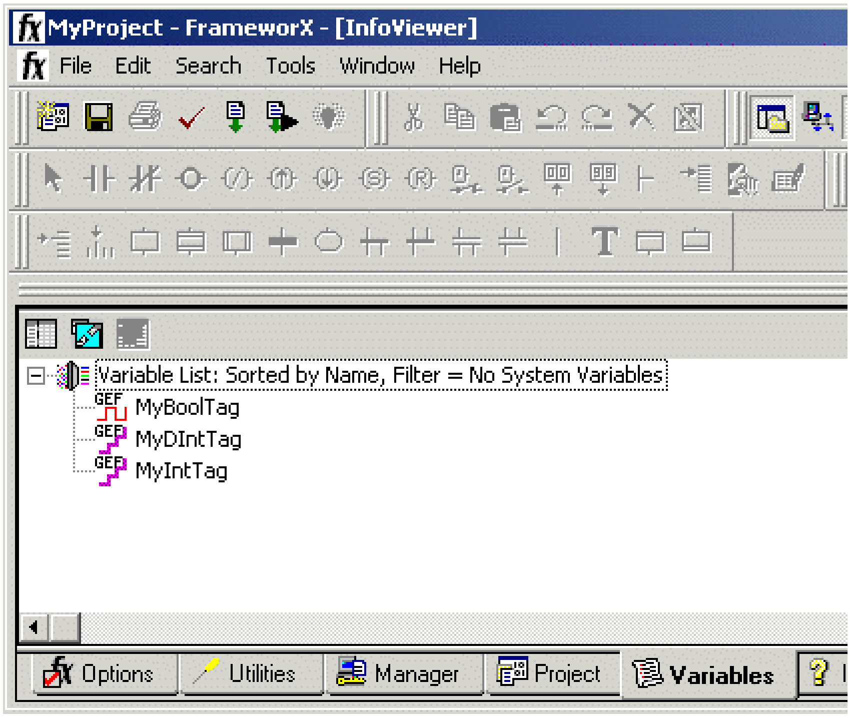
How are LogicDeveloper array variables imported?

See **LogicDeveloper Array Tag Import**

LogicDeveloper Import Preparation: LogicDeveloper Steps

1. Open the FrameworX project containing the tags (variables) that will be ported over to OPC server.

2. Open the **Navigator** window if it's not already open by pressing Shift-F4. Click **Variables** | **Variable List View**.



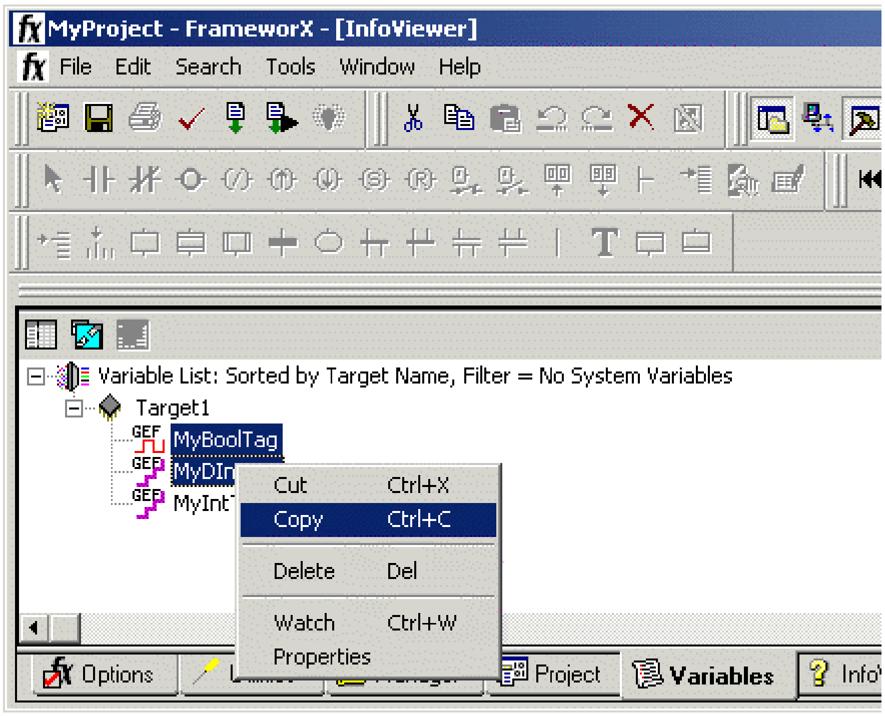
Variable List: Sorted by Name, Filter = No System Variables
MyBoolTag
MyDIntTag
MyIntTag

Important: Each FrameworX project contains one or more targets. A target is essentially the device on which the application will run. Variables are created on the target-level, so in order to specify the variables to export the target of interest must be specified. In order for the target variables to be imported, the variables must have **GE PLC** as the **Data Source**. This can be verified by left-clicking on the variable and looking at its Data Source property in the **Inspector** window. Internal variables will not be imported.

3. Next, sort the variables by target. Right-click on the **Variable List** header and then click **Sort** | **Target**.

4. Highlight the tags of interest in the target of interest. For more information, refer to **Highlighting LogicDeveloper Variables**.

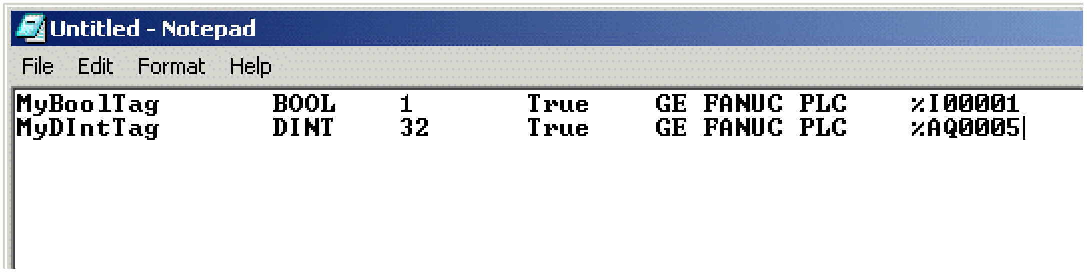
5. Select **Edit** | **Copy** to copy the variable to the Clipboard.


MyProject - FrameworX - [InfoViewer]
File Edit Search Tools Window Help
Variable List: Sorted by Target Name, Filter = No System Variables
Target1
MyBoolTag
MyDIn
MyInt
Cut Ctrl+X
Copy Ctrl+C
Delete Del
Watch Ctrl+W
Properties
Options
Project
Variables
Info

6. Next, open a word-processing program like Notepad or Wordpad. This will be the place where the variables will ultimately be saved for importing.

7. Click **Edit** | **Paste**.

8. 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.



```
MyBoolTag          BOOL      1        True      GE FANUC PLC      %I00001
MyDIntTag          DINT      32       True      GE FANUC PLC      %AQ0005
```

9. Save this text document with the text extension (*.txt) in ANSI form.

10. The variables highlighted and copied in Logic Developer are now contained within this text document to be imported into the OPC server.

LogicDeveloper Import Preparation: OPC Server Steps

1. Open up the **Device Properties** for 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 that was just 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. 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 LogicDeveloper Variables

Highlighting variables in Logic Developer can be performed in the following ways.

- **Single Variable Selecting:** Left-click on a variable of interest.
- **Pick-n-Choose:** Start by left-clicking on the first variable of interest. Then, press the CTRL key while left-clicking on each successive variable of interest. Do this until all variables of interest are highlighted.
- **Selecting a Range of Variables:** Start by left-clicking on the first variable in the range of interest. Then, press the SHIFT key while left-clicking on the last variable in the range. All variables in the range will be highlighted.
- **Selecting All Variables:** Start by left-clicking 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.

LogicDeveloper Array Tag Import

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

Importing Proficy Logic Developer Tags

This driver uses import 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. To import tags from an application other than Proficy Logic Developer, refer to **Automatic Tag Database Generation** to see if the application is supported.

Note: This driver supports the importing of structured data types.

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 if it's not already open by pressing Shift-F4.

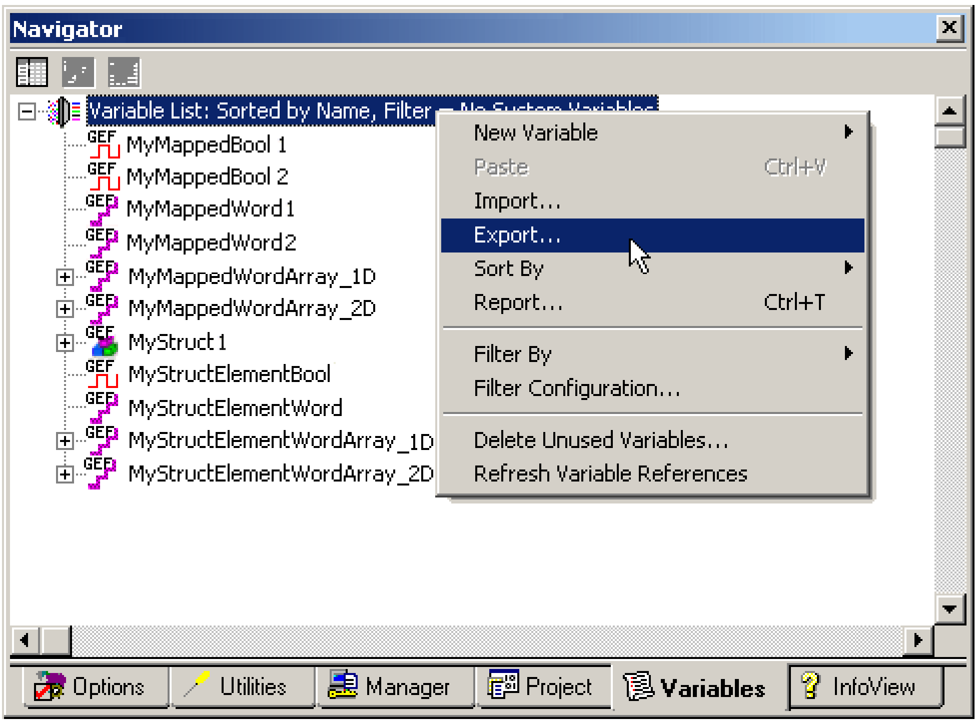
3. Click on the **Variables** tab.

4. Select **Variable List View** on the left-hand side of the window.

Important: Each Logic Developer project contains one or more targets. A target is essentially the device on which the application will run. Variables are created on the target-level, so in order to specify the variables to export, users must first decide the target of interest. In order for the target variables to be imported, the variables must have **GE PLC** as the **Data Source**. This can be verified by left clicking on the variable and looking at its Data Source properties in the **Inspector** window. Internal variables will not be imported.

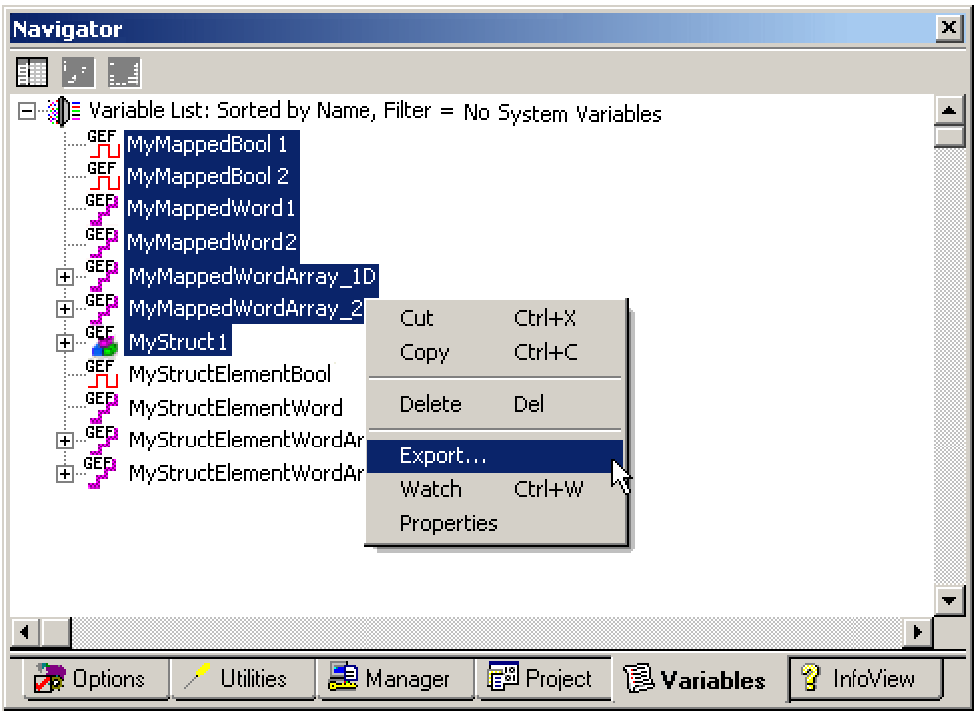
5. Next, sort the variables by target. To do so, right-click on the **Variable List** header and then select **Sort** | **Target**.

6. To export all of the variables, right-click on the **Variable List** tree root and then click **Export...**.

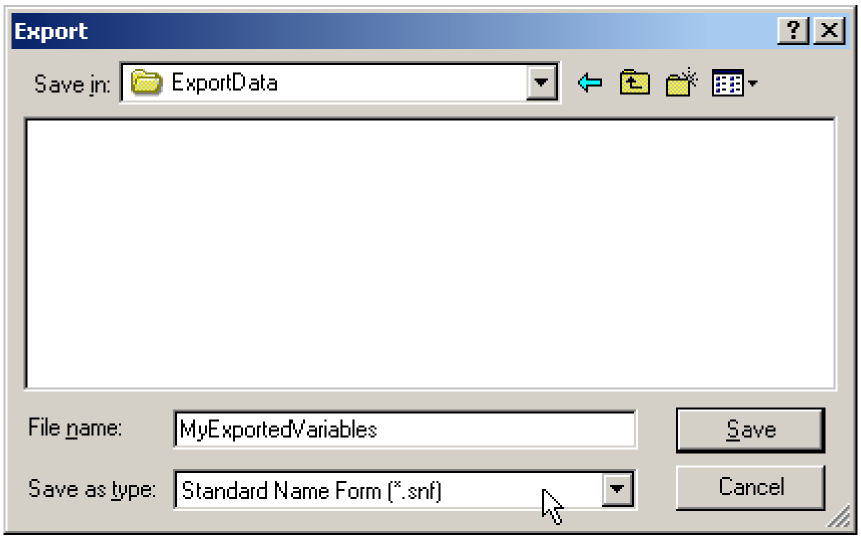


7. To export only selected variables, highlight the tags of interest in the target of interest. For more information, refer to **Highlighting Proficy Logic Developer Variables**.

8. Then, right-click on one of the selected variables and then click **Export...**.



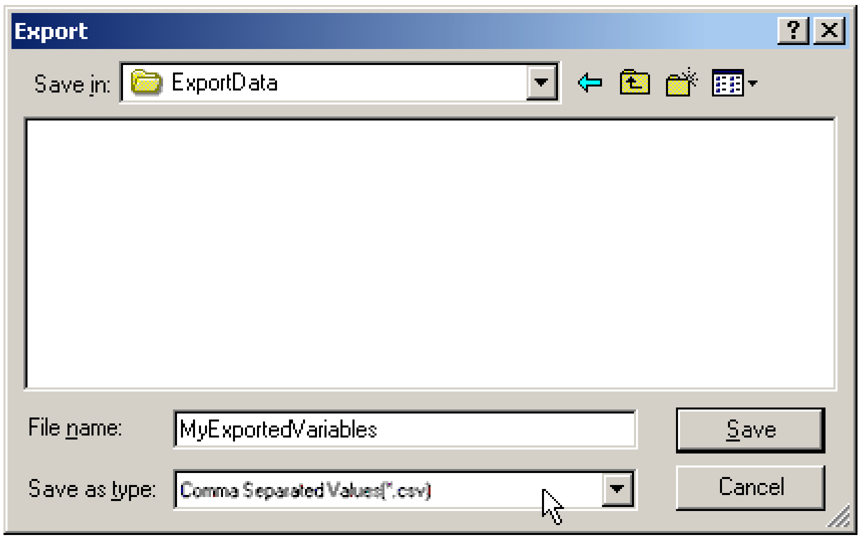
9. In the **Save as Type** drop-down list, select the export file type: either **Standard Name Form (*.snf)** or **Comma Separated Variable (*.csv)**. Both dialogs are shown below.



File name: MyExportedVariables

Save as type: Standard Name Form (*.snf)

Save in: ExportData



File name: MyExportedVariables

Save as type: Comma Separated Values(*.csv)

Save in: ExportData

Proficy Logic Developer Import Preparation: OPC Server Steps

1. Open up the **Device Properties** for 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 (*.snf or*.csv) that was just created.

4. Select the **Database Creation** tab and then click **Auto Create** to import the variables. Alternatively, select other settings that will 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**.

See Also: Variable Import Settings

Highlighting Proficy Logic Developer Variables

Highlighting variables in Logic Developer can be performed in the following ways.

- **Single Variable Selecting:** Left-click on a variable of interest.
- **Pick-n-Choose:** Start by left-clicking on the first variable of interest. Then, press the CTRL key while left-clicking on each successive variable of interest. Do this until all variables of interest are highlighted.

- **Selecting a Range of Variables:** Start by left-clicking on the first variable in the range of interest. Then, press the SHIFT key while left-clicking on the last variable in the range. All variables in the range will be highlighted.
- **Selecting All Variables:** Start by left-clicking 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.

Proficy Logic Developer Array Tag Import

Arrays of referenced variables and arrays of symbolic variables will be imported differently. Descriptions are as follows:

- **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 places in the **Symbolic** group along with all other symbolic variable tags. The driver will not generate tags for **Bool** and **String** symbolic variable arrays.

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.

GE Micro
311
313
331
341
350
360
731
732
771
772
781
782
GE OPEN
Advanced Addressing

Note: Each topic contains tables that display the ranges that are supported by the driver for that model; however, the actual range may vary depending on the configuration of the device.

GE Micro 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	R0001 to R9999 R0001 to R9998 R0001 to R9996	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI0001 to AI1024 AI0001 to AI1023 AI0001 to AI1021	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ0001 to AQ256 AQ0001 to AQ255 AQ0001 to AQ253	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing

Default Data Type Override
String Access to Registers
Array Support

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	R001 to R512 R001 to R511 R001 to R509	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI01 to AI64 AI01 to AI63 AI01 to AI61	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ001 to AQ032 AQ001 to AQ031 AQ001 to AQ029	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing
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) 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	S01 to S32 S01 to S25 (every 8th bit)	**Boolean** Byte	Read Only

(Same for SA, SB, SC)	S01 to S17 (every 8th bit)	Word, Short, BCD	
Register References	R001 to R1024 R001 to R1023 R001 to R1021	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI01 to AI64 AI01 to AI63 AI01 to AI61	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ001 to AQ032 AQ001 to AQ031 AQ001 to AQ029	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing
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	R0001 to R2048 R0001 to R2047 R0001 to R2045	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI001 to AI128 AI001 to AI127 AI001 to AI125	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ01 to AQ64 AQ01 to AQ63 AQ01 to AQ61	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing
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	R0001 to R9999 R0001 to R9998 R0001 to R9996	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI0001 to AI1024 AI0001 to AI1023 AI0001 to AI1021	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ0001 to AQ256 AQ0001 to AQ255 AQ0001 to AQ253	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing
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	R0001 to R9999 R0001 to R9998 R0001 to R9996	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI0001 to AI2048 AI0001 to AI2047 AI0001 to AI2045	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ001 to AQ512 AQ001 to AQ511 AQ001 to AQ509	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing
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	R00001 to R32768 R00001 to R32767 R00001 to R32765	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI0001 to AI2048 AI0001 to AI2047 AI0001 to AI2045	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ001 to AQ512 AQ001 to AQ511 AQ001 to AQ509	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing
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	R00001 to R16384 R00001 to R16383 R00001 to R16381	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI0001 to AI8192 AI0001 to AI8191 AI0001 to AI8189	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ0001 to AQ8192 AQ0001 to AQ8191 AQ0001 to AQ8189	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing
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	R00001 to R16384 R00001 to R16383 R00001 to R16381	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI0001 to AI8192	**Word**, Short, BCD	Read/Write

	AI0001 to AI8191 AI0001 to AI8189	DWord, Long, LBCD, Float Double	
Analog Outputs	AQ0001 to AQ8192 AQ0001 to AQ8191 AQ0001 to AQ8189	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing
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	R00001 to R16384 R00001 to R16383 R00001 to R16381	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI0001 to AI8192 AI0001 to AI8191 AI0001 to AI8189	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ0001 to AQ8192 AQ0001 to AQ8191 AQ0001 to AQ8189	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing
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	**Boolean**	Read/Write

	Q0001 to Q2041 (every 8th bit) Q0001 to Q2033 (every 8th bit)	Byte Word, Short, BCD	
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	R00001 to R16384 R00001 to R16383 R00001 to R16381	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI0001 to AI8192 AI0001 to AI8191 AI0001 to AI8189	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ0001 to AQ8192 AQ0001 to AQ8191 AQ0001 to AQ8189	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing
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	R00001 to R16384 R00001 to R16383 R00001 to R16381	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI0001 to AI8192 AI0001 to AI8191 AI0001 to AI8189	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ0001 to AQ8192 AQ0001 to AQ8191	**Word**, Short, BCD DWord, Long, LBCD, Float	Read/Write

	AQ0001 to AQ8189	Double	

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

Advanced Addressing
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	R00001 to R16384 R00001 to R16383 R00001 to R16381	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI0001 to AI8192 AI0001 to AI8191 AI0001 to AI8189	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ0001 to AQ8192 AQ0001 to AQ8191 AQ0001 to AQ8189	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing
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.

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	**Boolean**	Read/Write

	Q00001 to Q32761 (every 8th bit) Q00001 to Q32753 (every 8th bit)	Byte Word, Short, BCD	
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 (Same for SA, SB, SC)	S00001 to S32768 S00001 to S32761 (every 8th bit) S00001 to S32753 (every 8th bit)	**Boolean** Byte Word, Short, BCD	Read Only
Register References	R00001 to R32768 R00001 to R32767 R00001 to R32765	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Inputs	AI00001 to AI32768 AI00001 to AI32767 AI00001 to AI32765	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write
Analog Outputs	AQ00001 to AQ32768 AQ00001 to AQ32767 AQ00001 to AQ32765	**Word**, Short, BCD DWord, Long, LBCD, Float Double	Read/Write

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

Advanced Addressing
Default Data Type Override
String Access to Registers
Array Support

Advanced Addressing

Default Data Type Override

The default data types for each device type are shown in the table below. These 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.)

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 type indicator. The length of the string is based on how the device address reference is entered. Each register addressed can contain two 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.

Examples

Address	Description
R100-R150 M	Access Register R100 as string with a length of 102 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.)
R100-R150 M H	Access Register R100 as string with a length of 102 bytes. HiLo byte order is explicitly specified.
R100-R150 M L	Access Register R100 as string with a length of 102 bytes. LoHi byte order is explicitly specified.

Note: The maximum string length is 128 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. The maximum array size is 16 Doubles, 32 DWords (Longs, Floats), 64 Words (Shorts), or 128 Bytes for a total of 1024 bits. The following data types support arrays: Byte, Word, Short, DWord, Long, Float, and Double.

Examples

Address	Address Breakdown
G1 [4] includes the following byte addresses*	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

*G25 indicates the fourth byte beginning at bit 25.

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>'

Serial Communications

COMn does not exist
Error opening COMn
COMn is in use by another application
Unable to set comm parameters on COMn
Communications error on '<channel name>' [<error mask>]

Device Status Messages

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

Device Specific Messages

Invalid tag in block starting at <address> on device <device name>. Block deactivated

Write Error Messages

Unable to write to tag '<tag address>' on device '<device name>'. The service requested is either not defined or not supported
Unable to write to tag '<tag address>' on device '<device name>'. The user does not have sufficient privileges to process the request. Minor status error code = '<hexadecimal error code>'
Unable to write to tag '<tag address>' on device '<device name>'. The CPU has received a message that is out of order
Unable to write to tag '<tag address>' on device '<device name>'. Service request error. Minor status error code = '<hexadecimal error code>'
Unable to write to tag '<tag address>' on device '<device name>'. Service request mailbox type is either undefined or unexpected
Unable to write to tag '<tag address>' on device '<device name>'. The PLC CPU's service request queue is full: please wait a minimum of 10 ms before sending another service request
Unable to write to tag '<tag address>' on device '<device name>'. A framing error has occurred
Unable to write to tag '<tag address>' on device '<device name>'. Device returned major error code '<hexadecimal error code>' and minor error code '<hexadecimal error code>'

Blocked Read Error Messages

Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. The service requested is either not defined or not supported
Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. The user does not have sufficient privileges to process the request. Minor status error code = '<hexadecimal error code>'
Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. The CPU has received a message that is out of order
Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. Service request error. Minor status error code = '<hexadecimal error code>'
Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. Service request mailbox type is either undefined or unexpected
Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. The PLC CPU's service request queue is full: please wait a minimum of 10 ms before sending another service request
Unable to read '<byte count>' bytes starting at address '<start tag>' on device '<device name>'. A framing error has occurred

Address Validation

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

Address Validation

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 via DDE 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:

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

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

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

Error Type:
Warning

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

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

Serial Communications

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

Serial Communications
COMn does not exist
Error opening COMn
COMn is in use by another application
Unable to set comm parameters on COMn
Communications error on '<channel name>' [<error mask>]

COMn does not exist

Error Type:
Fatal

Possible Cause:
The specified COM port is not present on the target computer.

Solution:
Verify that the proper COM port has been selected.

Error opening COMn

Error Type:
Fatal

Possible Cause:
The specified COM port could not be opened due an internal hardware or software problem on the target computer.

Solution:
Verify that the COM port is functional and may be accessed by other Windows applications.

COMn is in use by another application

Error Type:
Fatal

Possible Cause:
The serial port assigned to a device is being used by another application.

Solution:
1. Verify that the correct port has been assigned to the channel.
2. Verify that only one copy of the current project is running.

Unable to set comm parameters on COMn

Error Type:
Fatal

Possible Cause:
The serial parameters for the specified COM port are not valid.

Solution:
Verify the serial parameters and make any necessary changes.

Communications error on '<channel name>' [<error mask>]

Error Type:
Serious

Error Mask Definitions:
B = Hardware break detected.
F = Framing error.
E = I/O error.
O = Character buffer overrun.
R = RX buffer overrun.
P = Received byte parity error.
T = TX buffer full.

Possible Cause:
1. The serial connection between the device and the Host PC is bad.
2. The communications parameters for the serial connection are incorrect.

Solution:
1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communications parameters match those of the device.

Device Status Messages

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

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

Device '<device name>' not responding

Error Type:
Serious

Possible Cause:
1. The serial connection between the device and the Host PC is broken.
2. The communications parameters for the serial connection or incorrect.
3. The named device may have been assigned an incorrect Network ID.
4. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device setting.

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

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

Error Type:
Serious

Possible Cause:
1. The serial connection between the device and the Host PC is broken.

2. The communications parameters for the serial connection or incorrect.
3. The named device may have been assigned an incorrect Network ID.

Solution:

1. Verify the cabling between the PC and the PLC device.
2. Verify the specified communications parameters match those of the device.
3. Verify that the Network ID given to the named device matches that of the actual device.

Device Specific Messages

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

Device Specific Messages

Invalid tag in block starting at <address> on device <device name>. Block deactivated

Write Error Messages

Unable to write to tag '<tag address>' on device '<device name>'. The service requested is either not defined or not supported
Unable to write to tag '<tag address>' on device '<device name>'. The user does not have sufficient privileges to process the request. Minor status error code = '<hexadecimal error code>'
Unable to write to tag '<tag address>' on device '<device name>'. The CPU has received a message that is out of order
Unable to write to tag '<tag address>' on device '<device name>'. Service request error. Minor status error code = '<hexadecimal error code>'
Unable to write to tag '<tag address>' on device '<device name>'. Service request mailbox type is either undefined or unexpected
Unable to write to tag '<tag address>' on device '<device name>'. The PLC CPU's service request queue is full: please wait a minimum of 10 ms before sending another service request
Unable to write to tag '<tag address>' on device '<device name>'. A framing error has occurred
Unable to write to tag '<tag address>' on device '<device name>'. Device returned major error code '<hexadecimal error code>' and minor error code '<hexadecimal error code>'

Blocked Read Error Messages

Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. The service requested is either not defined or not supported
Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. The user does not have sufficient privileges to process the request. Minor status error code = '<hexadecimal error code>'
Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. The CPU has received a message that is out of order
Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. Service request error. Minor status error code = '<hexadecimal error code>'
Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. Service request mailbox type is either undefined or unexpected
Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. The PLC CPU's service request queue is full: please wait a minimum of 10 ms before sending another service request
Unable to read '<byte count>' bytes starting at address '<start tag>' on device '<device name>'. A framing error has occurred
Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. Device returned major error code '<hexadecimal error code>' and minor error code '<hexadecimal error code>'

Non-Blocked Error Messages

Unable to read tag '<tag address>' on device '<device name>'. The service requested is either not defined or not supported
Unable to read tag '<tag address>' on device '<device name>'. The user does not have sufficient privileges to process the request. Minor status error code = '<hexadecimal error code>'
Unable to read tag '<tag address>' on device '<device name>'. The CPU has received a message that is out of order
Unable to read tag '<tag address>' on device '<device name>'. Service request error. Minor status error code = '<hexadecimal error code>'
Unable to read tag '<tag address>' on device '<device name>'. Service request mailbox type is either undefined or unexpected

Unable to read tag '<tag address>' on device '<device name>'. The PLC CPU's service request queue is full: please wait a minimum of 10 ms before sending another service request
Unable to read tag '<tag address>' on device '<device name>'. A framing error has occurred
Unable to read tag '<tag address>' on device '<device name>'. Device returned major error code '<hexadecimal error code>' and minor error code '<hexadecimal error code>'

Invalid tag in block starting at <address> on device <device name>. Block deactivated

Error Type:
Serious

Possible Cause:
An attempt has been made to reference a nonexistent location in the specified device.

Solution:
Verify the tags assigned to addresses in the specified range on the device and eliminate ones that reference invalid locations.

Unable to write to tag '<tag address>' on device '<device name>'. The service requested is either not defined or not supported

Error Type:
Warning

Possible Cause:
The requested service is either not defined or not supported.

Solution:
1. Determine whether writes are supported for the tag address.
2. Verify that the device has the latest Firmware revision.

Unable to write to tag '<tag address>' on device '<device name>'. The user does not have sufficient privileges to process the request. Minor status error code = '<hexadecimal error code>'

Error Type:
Warning

Possible Cause:
The user does not have sufficient privileges to complete the service request.

Solution:
For the privilege level required to complete the service request, refer to the Minor Status field.

Unable to write to tag '<tag address>' on device '<device name>'. The CPU has received a message that is out of order

Error Type:
Warning

Possible Cause:
The CPU received a malformed message.

Solution:
Resend the write request. This will automatically resend the polled tags, as well.

Unable to write to tag '<tag address>' on device '<device name>'. Service request error. Minor status error code = '<hexadecimal error code>'

Error Type:
Warning

Possible Cause:
The CPU cannot process the service request correctly.

Solution:
For the specific error code, refer to the Minor Status field.

Unable to write to tag '<tag address>' on device '<device name>'. Service request mailbox type is either undefined or unexpected

Error Type:
Warning

Possible Cause:
The service request mailbox is either undefined or unexpected.

Solution:
Make sure that a valid mailbox type has been specified. Valid values used by master SNP implementations are 0xC0 and 0x80. Valid values used by slave SNP implementations are 0xD4, 0x94, and 0xD1.

Unable to write to tag '<tag address>' on device '<device name>'. The PLC CPU's service request queue is full: please wait a minimum of 10 ms before sending another service request

Error Type:
Warning

Possible Cause:
The PLC CPU's service request queue is full.

Solution:
Resend the write request at a later time. This will automatically resend the polled tags, as well.

Unable to write to tag '<tag address>' on device '<device name>'. A framing error has occurred

Error Type:
Warning

Possible Cause:
1. The packets are misaligned due to the connection between the PC and the device.
2. Bad cabling connects the device and is causing noise.

Solution:
1. Place the device on a less noisy network.
2. Increase the Request Timeout and/or Retry Attempts.

Unable to write to tag '<tag address>' on device '<device name>'. Device returned major error code '<hexadecimal error code>' and minor error code '<hexadecimal error code>'

Error Type:
Warning

Possible Cause:
An unknown error has occurred.

Solution:
For the meaning of the error code, refer to the manufacturer's documentation.

Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. The service requested is either not defined or not supported

Error Type:
Warning

Possible Cause:
The requested service is either not defined or not supported.

Solution:
1. Determine whether reads are supported for the tag address.
2. Verify that the device has the latest Firmware revision.

Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. The user does not have sufficient privileges to process the request. Minor status error code = '<hexadecimal error code>'

Error Type:
Warning

Possible Cause:
The user does not have sufficient privileges to complete the service request.

Solution:
For the privilege level required to complete the service request, refer to the Minor Status field.

Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. The CPU has received a message that is out of order

Error Type:
Warning

Possible Cause:
The CPU received a malformed message.

Solution:
Resend the read request. This will automatically resend the polled tags, as well.

Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. Service request error. Minor status error code = '<hexadecimal error code>'

Error Type:
Warning

Possible Cause:
The CPU cannot process the service request correctly.

Solution:
For the specific error code, refer to the Minor Status field.

Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. Service request mailbox type is either undefined or unexpected

Error Type:
Warning

Possible Cause:
The service request mailbox is either undefined or unexpected.

Solution:
Make sure that a valid mailbox type has been specified. Valid values used by master SNP implementations are 0xC0 and 0x80. Valid values used by slave SNP implementations are 0xD4, 0x94, and 0xD1.

Unable to read ' <number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. The PLC CPU's service request queue is full: please wait a minimum of 10 ms before sending another service request

Error Type:
Warning

Possible Cause:
The PLC CPU's service request queue is full.

Solution:
Resend the read request at a later time. This will automatically resend the polled tags, as well.

Unable to read '<byte count>' bytes starting at address '<start tag>' on device '<device name>'. A framing error has occurred

Error Type:
Warning

Possible Cause:
1. The packets are misaligned due to the connection between the PC and the device.
2. Bad cabling connects the device and is causing noise.

Solution:
1. Place the device on a less noisy network.
2. Increase the Request Timeout and/or Retry Attempts.

Unable to read '<number of bytes>' bytes starting at address '<tag address>' on device '<device name>'. Device returned major error code '<hexadecimal error code>' and minor error code '<hexadecimal error code>'

Error Type:
Warning

Possible Cause:
An unknown error has occurred.

Solution:
For the meaning of the error code, refer to the manufacturer's documentation.

Unable to read tag '<tag address>' on device '<device name>'. The service requested is either not defined or not supported

Error Type:
Warning

Possible Cause:
The requested service is either not defined or not supported.

Solution:
1. Determine whether reads are supported for the tag address.
2. Verify that the device has the latest Firmware revision.

Unable to read tag '<tag address>' on device '<device name>'. The user does not have sufficient privileges to process the request. Minor status error code = '<hexadecimal error code>'

Error Type:

Warning

Possible Cause:
The user does not have sufficient privileges to complete the service request.

Solution:
For the privilege level required to complete the service request, refer to the Minor Status field.

Unable to read tag '<tag address>' on device '<device name>'. The CPU has received a message that is out of order

Error Type:
Warning

Possible Cause:
The CPU received a malformed message.

Solution:
Resend the read request. This will automatically resend the polled tags, as well.

Unable to read tag '<tag address>' on device '<device name>'. Service request error. Minor status error code = '<hexadecimal error code>'

Error Type:
Warning

Possible Cause:
The CPU cannot process the service request correctly.

Solution:
For the specific error code, refer to the Minor Status field.

Unable to read tag '<tag address>' on device '<device name>'. Service request mailbox type is either undefined or unexpected

Error Type:
Warning

Possible Cause:
The service request mailbox is either undefined or unexpected.

Solution:
Make sure that a valid mailbox type has been specified. Valid values used by master SNP implementations are 0xC0 and 0x80. Valid values used by slave SNP implementations are 0xD4, 0x94, and 0xD1.

Unable to read tag '<tag address>' on device '<device name>'. The PLC CPU's service request queue is full: please wait a minimum of 10 ms before sending another service request

Error Type:
Warning

Possible Cause:
The PLC CPU's service request queue is full.

Solution:
Resend the read request at a later time. This will automatically resend the polled tags, as well.

Unable to read tag '<tag address>' on device '<device name>'. A framing error has occurred

Error Type:
Warning

Possible Cause:
1. The packets are misaligned due to the connection between the PC and the device.
2. Bad cabling connects the device and is causing noise.

Solution:
1. Place the device on a less noisy network.
2. Increase the Request Timeout and/or Retry Attempts.

Unable to read tag '<tag address>' on device '<device name>'. Device returned major error code '<hexadecimal error code>' and minor error code '<hexadecimal error code>'

Error Type:
Warning

Possible Cause:
An unknown error has occurred.

Solution:
For the meaning of the error code, refer to the manufacturer's documentation.

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: Import file is invalid or corrupt

Error Type:
Warning

Possible Cause:
1. The file specified as the Tag Import File in the Database Settings tab of Device Properties is a corrupt import file (*.snf or*.csv).
2. The file specified as the Tag Import File is an improperly formatted Logic Developer text file.

Solution:
1. Select a valid, properly formatted VersaPro/Logic Developer variable import file.
2. Retry the tag export process in the respective application to produce a new import file.

See Also:
Automatic Tag Database Generation Preparation

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 unused applications and/or increase the amount of virtual memory and try again.

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: 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 and, XXX_YYY for 2. 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: Datatype '<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 SNP Driver.

Database Error: Datatype '<type>' for tag '<tag name>' is currently not supported. Tag not created

Error Type:
Warning

Possible Cause:
The data type <type> as specified in the import file cannot be resolved or isn't natively supported by the GE SNP Driver. The tag was not automatically generated.

Solution:
For applicable tags, avoid using data type <type> in the VersaPro/Logic Developer projects.

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

Note:
The String data type is not currently supported by this driver. Therefore, neither the array tag or the individual elements will be generated for String array tags. Avoid using String data type when possible.

Index