

Honeywell HC900 Ethernet Device Driver Help

© 2009 Kepware Technologies

Table of Contents

1	Getting Started.....	3
	Help Contents.....	3
	Overview.....	3
2	Device Setup.....	3
	Device Setup.....	3
	TCP/IP.....	4
	Settings.....	4
	Blocks.....	5
3	Data Types Description.....	6
	Data Types Description.....	6
4	Automatic Tag Database Generation.....	6
	Automatic Tag Database Generation.....	6
	Tag Generation.....	7
	Tag Generation SP Programmer Details.....	8
	Tag Import.....	9
	Add Import File.....	11
	Creating Tag Import Files.....	11
5	Address Descriptions.....	15
	Address Descriptions.....	15
	Output Coils.....	15
	Input Coils.....	15
	Internal Registers.....	15
	Holding Registers.....	16
6	Optimizing Your Honeywell HC900 Ethernet Communications.....	18
	Optimizing Your Honeywell HC900 Ethernet Communications.....	18
7	Error Descriptions.....	18
	Error Descriptions.....	18
	Address Validation.....	19
	Address Validation.....	19
	Missing address.....	19
	Device address '<address>' contains a syntax error.....	20
	Address '<address>' is out of range for the specified device or register.....	20
	Device address '<address>' is not supported by model '<model name>'.....	20
	Data Type '<type>' is not valid for device address '<address>'.....	20
	Device address '<address>' is Read Only.....	20
	Array size is out of range for address '<address>'.....	21
	Array support is not available for the specified address: '<address>'.....	21
	Device Status Messages.....	21
	Device Status Messages.....	21
	Device '<device name>' is not responding.....	21
	Unable to write to '<address>' on device '<device name>'.....	21
	Device Specific Messages.....	22
	Device Specific Messages.....	22
	Failure to initiate 'winsock.dll'.....	22
	Bad address in block [x to y] on device '<device name>'.....	22
	Bad received length [x to y] on device '<device name>'.....	22

Tag Import Specific Messages.....	23
Tag Import Specific Messages.....	23
Could not read record <record> - Buffer length exceeded.....	23
No tags imported - Unsupported file format.....	23
Could not parse expected data (field <field>, record <record>).....	23
Invalid decimal address (field <field>, record <record>).....	24
Invalid tag name '<name>' (field <field>, record <record>) could not be coerced into.....	24
Invalid tag name '<old name>' (field <field>, record <record>) changed to '<new name>'.....	24
Invalid datatype (field <field>, record <record>).....	24
Invalid access (field <field>, record <record>).....	25
Invalid type (field <field>, record <record>).....	25
Invalid tag type (field <field>, record <record>).....	25
Could not import tag in record <record> - unknown block name.....	26
Invalid block name '<name>' (field <field>, record <record>) could not be coerced.....	26
Invalid block name '<old name>' (field <field>, record <record>) changed to '<new>'.....	26

Index

Honeywell HC900 Ethernet Device Driver Help

Help version 1.009

CONTENTS

Overview

What is the Honeywell HC900 Ethernet Device Driver?

Device Setup

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

Data Types Description

What data types does the Honeywell HC900 Ethernet driver support?

Automatic Tag Database Generation

How can I easily configure tags for the Honeywell HC900 Ethernet driver?

Address Descriptions

How do I reference a data location in a Honeywell HC900 Ethernet device?

Optimizing Your Honeywell HC900 Ethernet Communications

How do I get the best performance from the Honeywell HC900 Ethernet driver?

Error Descriptions

What error messages does the Honeywell HC900 Ethernet driver produce?

Overview

The Honeywell HC900 Ethernet Device 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. The Honeywell HC900 Ethernet Device Driver is intended for use with Honeywell HC900 Hybrid Controllers and similar devices.

Note 1: The driver will post messages when a failure occurs during operation.

Note 2: The driver is limited to 8192 devices.

Note 3: TCP/IP must be properly installed in order to use this driver. For information on setting up TCP/IP, refer to the Windows documentation.

Device Setup

Supported Devices

HC900 Hybrid Controller

Communication Protocol

Modbus Ethernet using Winsock V1.1 or higher

Connection Timeout

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

Request Timeout

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

Retry Attempts

This parameter specifies the number of times 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.

Device ID (PLC Network Address)

The Device ID is used to specify the device IP in standard YYY.YYY.YYY.YYY format.

See Also:

- [TCP/IP](#)
- [Block Sizes](#)
- [Settings](#)
- [Tag Generation](#)
- [Tag Import](#)

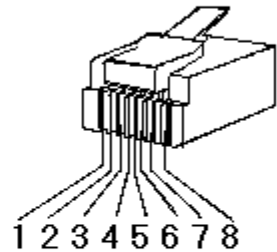
Cable Diagrams

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

TCP/IP

Port

This parameter specifies the TCP/IP port number that the remote device is configured to use. The default port number is 502.

Settings

First Word Low in 32 Bit Data Types (Float)

Two consecutive register addresses are used for 32-bit data types such as floats. Users can specify whether the driver should treat the contents of the first register as the low or high word in 32-bit values. The HC900 can be configured to

use a number of Double Register Formats. The formats are as follows.

Format	Description	Byte order	Notes
FP B	Floating Point Big Endian	4, 3, 2, 1	Honeywell default
FP BB	Floating Point Big Endian with byte-swap	3, 4, 1, 2	
FP L	Floating Point Little Endian	1, 2, 3, 4	
FP LB	Floating Point Little Endian with byte-swap	2, 1, 4, 3	Modbus standard

Examples of Data in FP B Format

Value (decimal)	Value (hex)	Register N		Register N+1	
		High	Low	High	Low
100.0	0x42C80000	0x42	0xC8	0x00	0x00
55.32	0x425D47AE	0x42	0x5D	0x47	0xAE
2.0	0x40000000	0x40	0x00	0x00	0x00
1.0	0x3F800000	0x3F	0x80	0x00	0x00
-1.0	0xBF800000	0xBF	0x80	0x00	0x00

Note: The driver will use the Honeywell default "FP B" if this device property is left unchecked. If checked, the "FP LB" format will be used. The driver does not currently support the Honeywell "FP BB" and "FP L" double register formats.

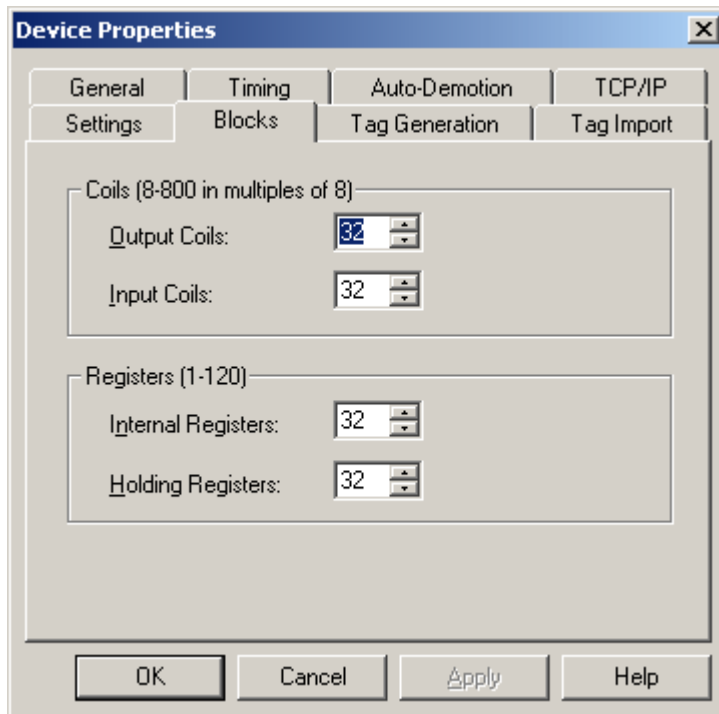
Blocks

Coil Block Sizes

Coils can be read from 8 to 800 points (bits) at a time.

Register Block Sizes

Registers can be read from 1 to 120 locations (words) at a time.



Given the overhead involved in sending data via TCP/IP, it is generally advantageous to keep the block size large.

However, if data will be read from non-contiguous locations within the device, reducing the block size may improve performance.

Data Types Description

Data Type	Description
Boolean	Single bit
Word	Unsigned 16 bit value bit 0 is the low bit bit 15 is the high bit
Short	Signed 16 bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
DWord	Unsigned 32 bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32 bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
BCD	Two byte packed BCD Value range is 0-9999. Behavior is undefined for values beyond this range.
LBCD	Four byte packed BCD Value range is 0-99999999. Behavior is undefined for values beyond this range.
Float	32 bit floating point value. The driver interprets two consecutive registers as a floating-point value by making the second register the high word and the first register the low word.
Float	If register 40001 is specified as a float, bit 0 of register 40001 would be bit 0 of the 32 bit word, and bit 15 of register 40002 would be bit 31 of the 32 bit word.

Automatic Tag Database Generation

The Honeywell HC900 Ethernet driver can automatically create most, if not all, of the tags needed for the application. Tags can be generated either individually or in combination, although the preferred method is **Tag Import**. In this method, the driver reads controller configuration data from one or more CSV files exported by the Honeywell Hybrid Control Designer application (version 2.1 or later). Users receive the exact set of tags that are being used in used by the controller. Alternatively, users can still take advantage of the driver's Automatic Tag Database Generation feature by using the **Tag Generation** method. In this method, the driver is given a general description of the controller's configuration (such as the number of loops used, number of set point programmers used and etc.). The driver then generates a set of tags based on this description. Users may then add or subtract tags manually after the tag generation operation is completed.

Users may configure the driver to use Automatic Tag Database Generation either with the New Device Wizard or with an existing device object's Device Properties dialog. Tag import settings can be found under Tag Import; tag generation settings can be found under Tag Generation in both the New Device Wizard and Device Properties dialogs. The driver will create the tags upon completion of the Device Wizard or the pressing of "OK" or "Apply" in Device Properties. The driver will then place messages in the server's event log as the tag generation process proceeds.

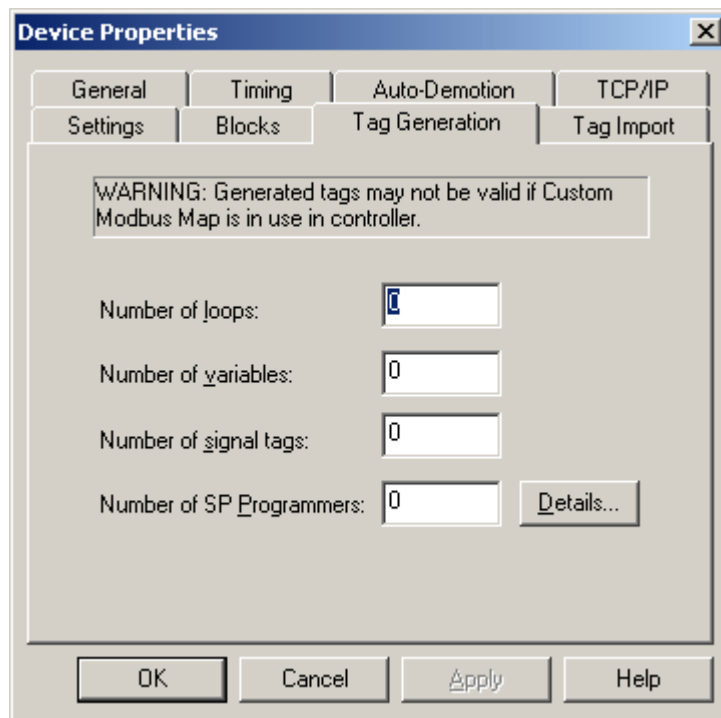
See Also: [Tag Import](#) and [Tag Generation](#).

Tag Generation

The Honeywell HC900 Ethernet driver has the ability to automatically create tags in the OPC server by using either **Tag Generation** or **Tag Import**. Tag Generation creates a set of tags that is based on a general description of the controller's configuration. Tag Import involves importing tags that have been defined in a Hybrid Control Designer application (version 2.1 or later). For more information, refer to [Tag Import](#).

Note: The driver will generate tags assuming the Universal Modbus Map is in use in the controller. Newer versions of the HC900 may use the Custom Modbus Map where various control parameters can be mapped to user-defined Modbus Partitions. The driver assumes the Universal Modbus Map is in use by default; thus, if the controller uses the Custom Modbus Map, it is recommended that Tag Import be utilized instead.

The Tag Generation Device Properties dialog is used to specify how many loops, variables, signal tags and Set Point Programmers are used in the project. The driver creates a set of tags based on these settings. After all parameters have been specified, click **Apply** or **OK** to begin the tag generation process.



Number of loops

Enter the number of control loops used in the controller (1-32). Tags for the most important loop parameters are created for each loop. The default parameter names are similar to those listed in the HC900 Communications Manual. Loops are listed according to their loop number, e.g., Loop_01, in the order entered in the controller configuration. The notes below describe how to associate the control loop number with the tag name assigned in the controller configuration.

Number of variables

Enter the number of variables (read/write) used in your controller configuration. Variables are listed by number, in the same sequence as in the controller configuration. Each variable will have an analog (float) and a digital (Boolean) parameter listing. The notes below describe how to select the appropriate type plus how to associate the variable number with that assigned in the controller configuration.

Number of signal tags

Enter the number of signal tags (read only) used in your controller configuration. Signal tags are listed by number, in the same sequence as in the controller configuration. Each signal tag will have an analog (float) and a digital (Boolean) parameter listing. The notes below describe how to select the appropriate type plus how to associate the signal tag

number with that assigned in the controller configuration.

Number of SP Programmers

Enter the number of SP programmers used in the controller (1– 8). The notes below describe how to associate the set point programmer number with the tag name assigned in the controller configuration.

Details...

Click this button to specify additional tag generation parameters specific to each set point programmer. For more information, refer to [Tag Generation SP Programmer Details](#).

Tags for each set point programmer are categorized into 3 groups. The Parameters group contains the current programmer status parameters including those specific to the current segment and programmer start, hold, advance and reset. The Additional Parameters group contains other setup parameters plus the current program number, the program save request that assigns a set of parameters previously written to the programmer block to a stored profile number, and the auxiliary output status (if used). The other groups are specific to each segment for profile setup. Consult the HC900 Communications User Manual for further description on use of these parameters for program monitoring and profile setup using the parameter groups.

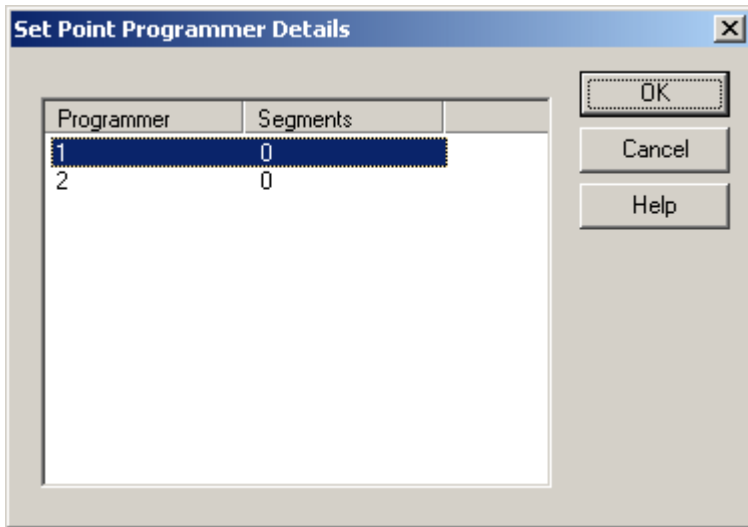
Notes:

1. Variables and signal tags can be used for either analog or digital data, depending on the device's configuration. In either case, the actual data stored in the device will be in IEEE floating point format. For digital data, TRUE/ON will be stored as 1.0, and FALSE/OFF will be stored as 0.0. Thus, users may read and write to any of these locations using tags with a data type of float. However, many client applications will not be able to handle "digital" values represented as floats, especially since TRUE is typically coerced into a float value of –1.0. The driver will automatically do the required conversion to or from a true Boolean if the tag is created with the Boolean data type.
2. To make the configuration of the automatic tag generation feature simple, each variable or signal tag is created with two listings having different tag names ending in B (for Boolean) or F (for Float), representing the digital and analog data types respectively. Users do not need to specify ahead of time what type of data each tag will be accessing. Simply configure the client application to use the appropriate version of the tag. Users may also delete any unused tags from the server: they are of no consequence as long as the client does not use them.
3. All tags generated by this driver are given generic names. To help associate the tags generated by this driver with the tags configured in the controller, we recommend that two reports be printed from the Honeywell Hybrid Control Designer application relative to the controller configuration. To do so, click **File | Print Report Preview** and then **FBD's**. To get information about variables and signal tags, select **Tag Information**. The Tag Generation Wizard will list the variables and signal tags in the same numerical order as documented. To get information about loop order and SP programmer order, select the **Block Modbus Addresses** report. The Tag Generation Wizard will list the loops and SP programmers in the same numerical order as documented. Note the associated tag names for these blocks.
4. If Hybrid Control Designer version 2.1 or later is being used, the Tag Import method may be preferable because it will import all of the tags defined in the project and no others. Additionally, the imported tags will have the same, or very similar, names as the tags in the controller.

See Also: [Tag Import](#)

Tag Generation SP Programmer Details

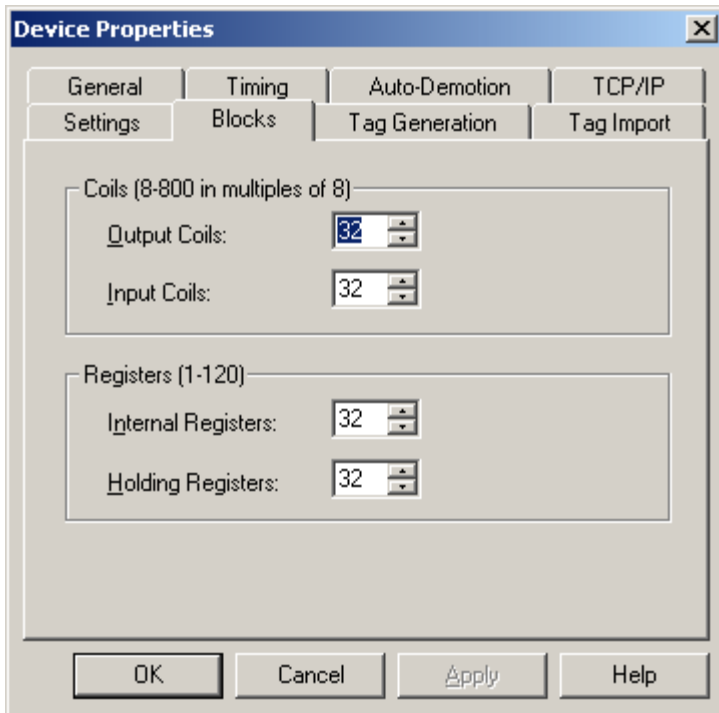
At least two groups of tags, **Parameters** and **Additional Parameters**, will be generated for each set point programmer specified on the Tag Generation Device Properties dialog. In addition, the driver may be configured to create groups of tags for each set point programmer segment used in the controller configuration. To do so, click on the **Details...** button on the Tag Generation dialog in order to show the number segments configured for each set point programmer. Next, double-click on a programmer row to edit. 0 to 50 segments may be specified for each set point programmer.



Tag Import

The Honeywell HC900 Ethernet driver has the ability to automatically create tags in the OPC server by using either **Tag Generation** or **Tag Import**. Tag Generation creates a set of tags based on a general description of the controller's configuration. For more information, refer to [Tag Generation](#). Tag Import involves importing tags that have been defined in a Hybrid Control Designer application (version 2.1 or later). Tag Import is the preferred method because it imports only the tags defined in the controller and no others.

The **Tag Import** Device Properties dialog is first used to specify a list of files from which tags will be imported. After the tag import file list has been specified, click **Apply** or **OK** to begin the tag generation process.



Add...

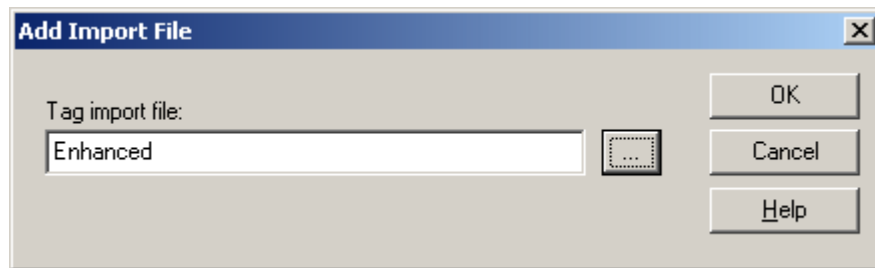
This button is used to add a tag import file to the list. For more information, refer to [Add Import File](#).

Remove

This button is used to remove the selected tag import file from the list.

Options...

Click the **Options** button under **Add...** and **Remove** for further options displayed in the Tag Import Options dialog.



Display Descriptions

Check this box to include tag descriptions defined in the Hybrid Control Designer as tag descriptions in the OPC Server. Descriptions for OPC Server tags are limited to 64 characters and will be truncated if necessary.

Regenerate Imported Tags

The driver will automatically regenerate tags when changes are made to the import file list, to the display descriptions options or to any of the settings in the Tag Generation dialog. Checking this box will also force the driver to regenerate the tags if none of the settings have changed but the contents of one or more of the tag import files have. All automatically generated tags will be recreated when **OK** or **Apply** is selected.

Tag Naming

Use the **Tag Naming** drop-down list to select a tag naming option; either **Enhanced** or **Legacy**. The setting chosen will become the default setting for all new devices added thereafter. When this driver is first installed, the default for any existing projects will be **Legacy**.

- **Enhanced:** This option has fewer naming constraints and is consistent with the naming requirements of the current OPC server. Tag names cannot have a period, double quotes, or start with an underscore.
- **Legacy:** This option enforces the stricter naming requirements of previous versions of this driver. Tag names must start with a letter, and the name must consist of letters and digits only.

Tag Import Notes:

The Hybrid Control Designer can generate many different types of export files, which are only used by this driver and are given as Hybrid Control Designer menu options. For more information, refer to See [Creating Tag Import Files](#).

1. FBD | Modbus Register Map | Detailed Function Block Report
2. FBD | Modbus Register Map | User-Defined Signals and Variables
3. FBD | Tag Information | Signal Tags
4. FBD | Tag Information | Variables
5. FBD | Tag Information | Signal Tags and Variables
6. FBD | Modbus Partitions | (Selected Partition Name)(Used for Custom Map)
7. FBD | All Modbus Registers
8. FBD | All Modbus Partitions (Used for Custom Map)

Additional Notes:

1. Using options 3 + 5, or 4 + 5, can create files with duplicate information and if used in the driver's import tag file list, will result in the creation of duplicate tags.
2. Option 6 refers to the selection of an individual named partition (from a list of all named partitions), while option 8 refers to the selection of all named partitions.
3. Using options 7 + any other option will result in duplicate tags.
4. The preferred method for importing all tags is a single file. Use option 6 for applications utilizing the Fixed Modbus Map (default for HC900). If the Custom Modbus Map (for Modbus Partitions) is utilized in HC Designer,

use option 7.

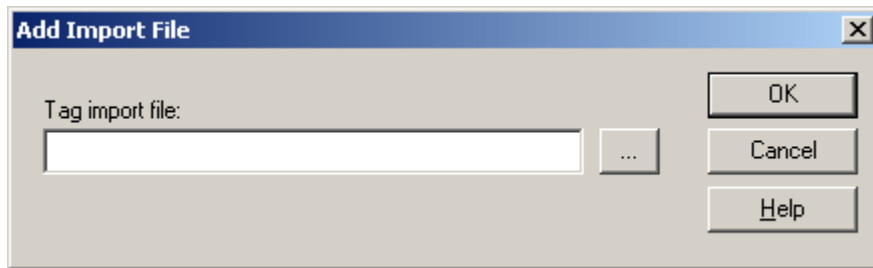
5. Tags generated from an import operation will be placed in groups. For file type 2, the driver will create groups called UserDefinedVariables and UserDefinedSignalTags. For file types 3, 4, and 5, the driver will create groups called TagInfoVariables and TagInfoSignalTags. For file type 1, the driver will create groups based on the block names given in the file. For Custom Map partition import, the Partition name will be used as a primary group name.
6. A leading underscore in group names will be replaced with A.
7. Tag names may be modified slightly to create valid OPC Server tag names. Digits may be appended to the end of a tag name to make the name unique. Any name changes will be reported in the server's event log. A leading underscore in tag names will be replaced with 0 (zero).

OPC Server Requirements for a Valid Tag Name:

- The tag name cannot start with an underscore_.
- Periods, double quotes, and back slashes are not allowed.
- The tag name must be no longer than 256 characters.
- Tags may have the same name, as long as they are in different groups.

Add Import File

To add a file to the Tag Import configuration, type the full path and filename in the box provided. Alternatively, click "... " button to browse to the desired file.

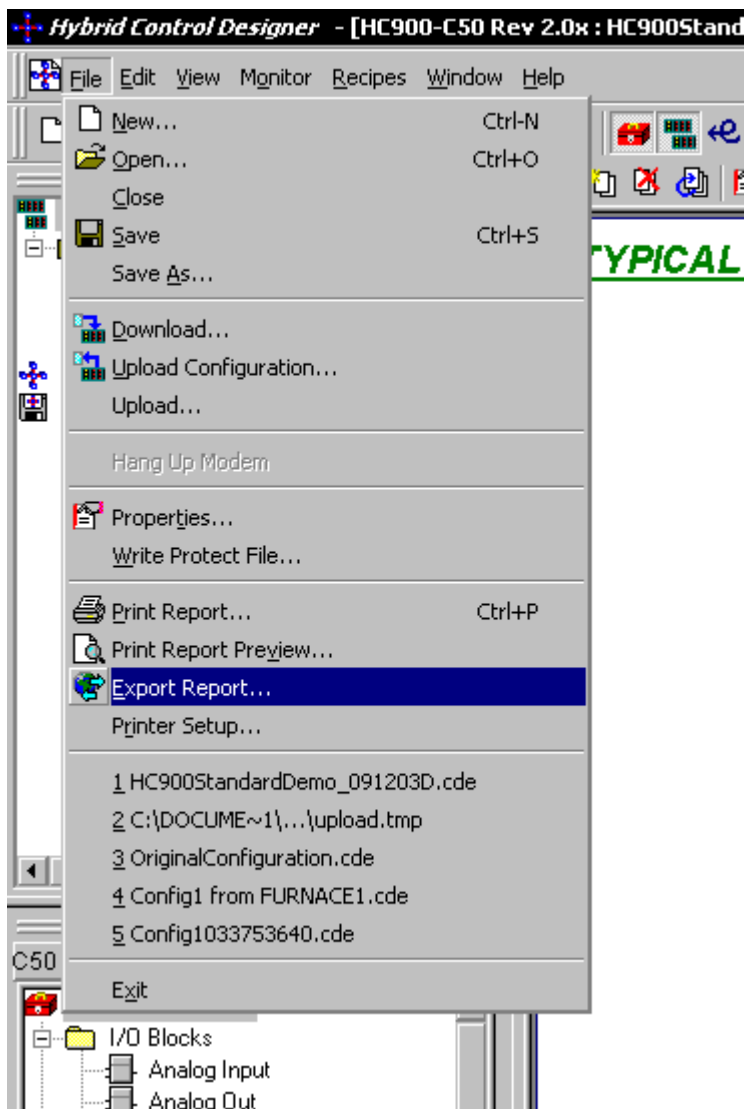


Note: For more information on the driver's tag import capability, refer to [Tag Import](#).

Creating Tag Import Files

The Honeywell Hybrid Control Designer version 2.1 and later is used to export controller configuration data to CSV (comma separated variable format) files. The files that contain tag information can in turn be imported by this driver. For more information on driver configuration, refer to [Tag Import](#). Instructions on how to create a tag import file are given below.

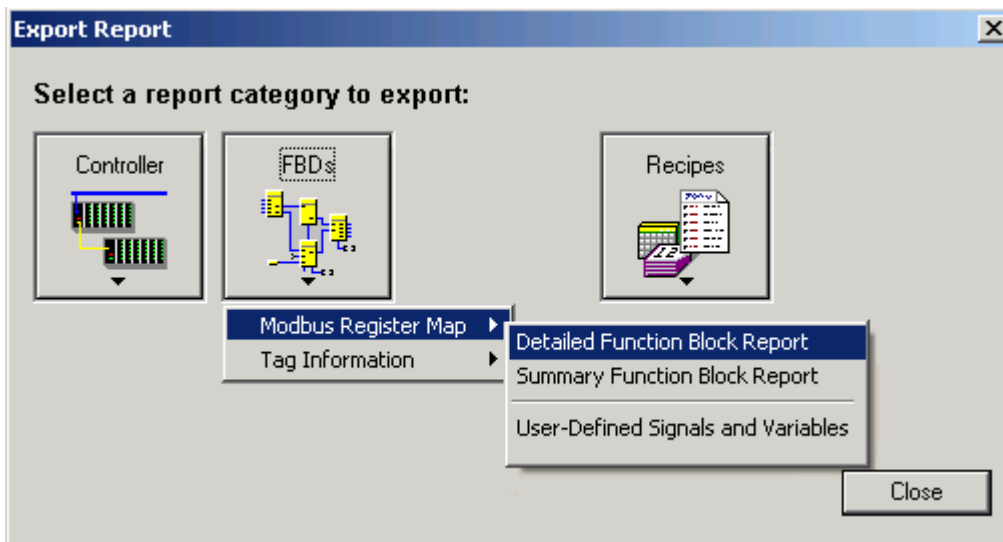
1. To create an export file, start the Hybrid Control Designer and then open a **configuration file**. Alternatively, upload a **controller configuration**.
2. Next, click **File | Export Report....**



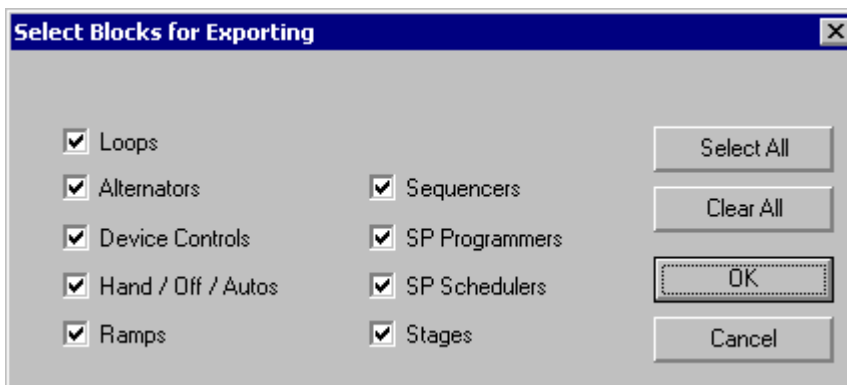
Note: Users will then be given the option to export Controller, FBDs, or Recipes data. Controller and Recipes data will not be of use to the driver. Of the FBDs options, only the **Summary Function Block Report** will not be of use to the driver. One or more of the files generated by any of the other FBDs options may be imported.

Example One

1. To create a file describing the tags used by the various function blocks defined in the controller, click **Modbus Register Map | Detailed Function Block Report**.



2. Next, specify the types of function blocks that will be included in the export file.

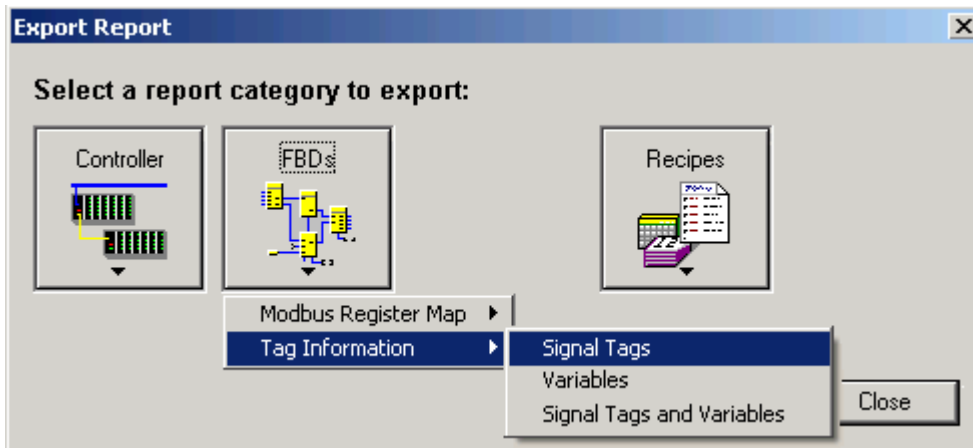


3. Click **OK** and then specify a file name.

Note: The file created can now be used directly by the driver's tag import feature.

Example Two

1. To create a file describing all of the Signal Tags in use by the controller, click **Tag Information | Signal Tags**.

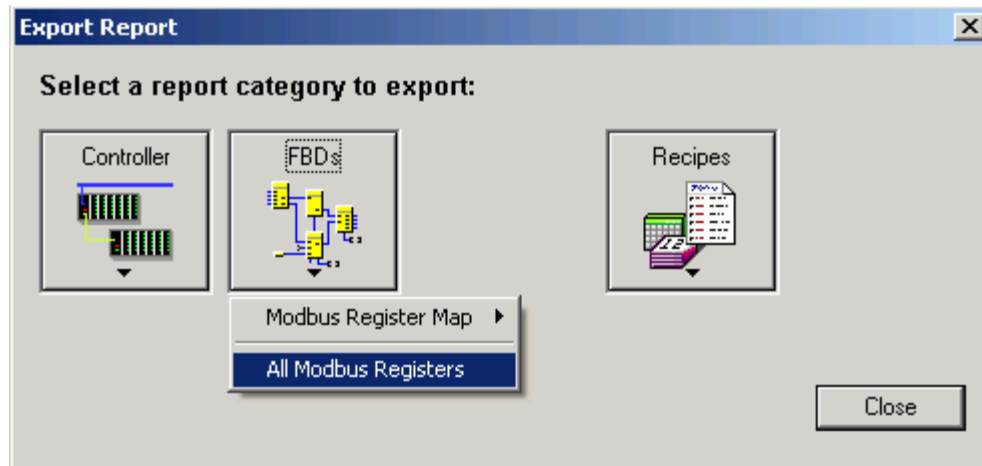


2. After all selections have been made, users will be prompted to specify a file name.

Note: The file created can now be used directly by the driver's tag import feature.

Example Three

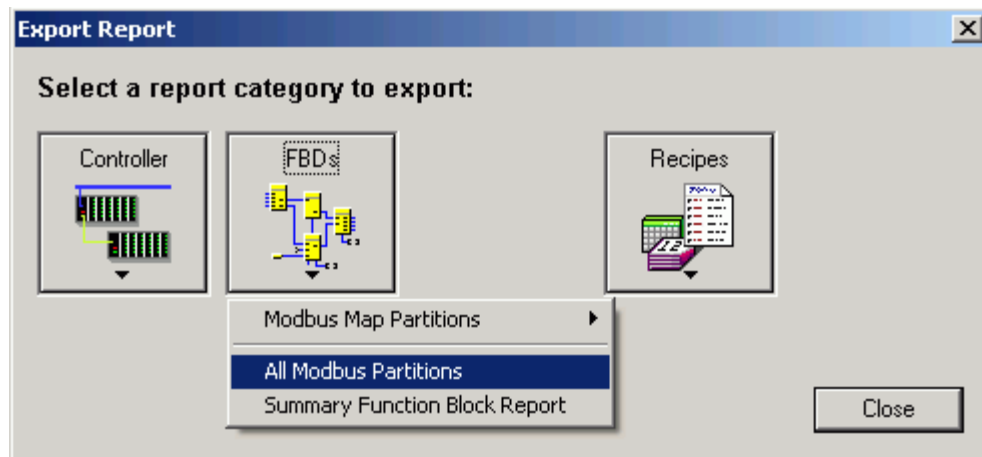
1. To create a file describing all of the tags for an HC900 configuration using the Fixed Modbus Map (the default), select **All Modbus Registers**.



2. Next, specify the file name for import.

Example Four

1. To create a file for all named Modbus partitions using a Custom Modbus Map (HC900 ver. 4.0 firmware and later), select **All Modbus Partitions**.



2. Next, define the file name for import. Users may choose to delete partitions that do not contain pertinent tags or that are empty. They may also select individual partitions for import.

Note 1: Data for Signal Tags and Variables may be interpreted as 32-bit floating point values, or Boolean values as floats (with float value 0.0 for FALSE/OFF or 1.0 for TRUE/ON). The driver will automatically assign the appropriate data type (float or Boolean) using the data in the Tag Information export file(s) and perform any necessary type conversions during runtime. Boolean-to-float conversions are not performed for User-Defined Signal Tags and Variables. **See Also:** [Holding Registers](#).

Note 2: The driver uses the header records in each import file to determine its content. Do not modify these headers.

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.

[Output Coils](#)

[Input Coils](#)

[Internal Registers](#)

[Holding Registers](#)

Output Coils

The default data type is shown in **bold**.

	Decimal Addressing	Hexadecimal Addressing
Type	Boolean	
Format	0xxxxx	H0yyyyy
Format	Read/Write	
Range	1-65536	1-10000

Note: Honeywell documentation often gives the location of parameters in device memory as a 1-based coil or register number in decimal and a 0-based modbus address in hex. The coil or register number will be the modbus address plus 1. For example, they may speak of "Variable #1", which is at "holding register 6337 with modbus address 0x18C0". This can be a point of confusion when creating tags with this driver. Make sure you **specify the coil or register number, and not the given modbus address** when creating tags with this driver. This number may be expressed in decimal, as in the Honeywell documentation, or in hex if you prefer. However, the coil or register number in hex will not be the same value as the modbus address given in the Honeywell documentation – it will be 1 greater.

Example:

The 255'th output coil would be addressed as '0255' using decimal addressing or 'H0FF' using hexadecimal addressing.

Input Coils

	Decimal Addressing	Hexadecimal Addressing
Type	Boolean	
Format	1xxxxx	H1yyyyy
Security	Read Only	
Range	1-65536	1-10000

Note: Honeywell documentation often gives the location of parameters in device memory as a 1-based coil or register number in decimal and a 0-based modbus address in hex. The coil or register number will be the modbus address plus 1. For example, they may speak of "Variable #1", which is at "holding register 6337 with modbus address 0x18C0". This can be a point of confusion when creating tags with this driver. Make sure you **specify the coil or register number, and not the given modbus address** when creating tags with this driver. This number may be expressed in decimal, as in the Honeywell documentation, or in hex if you prefer. However, the coil or register number in hex will not be the same value as the modbus address given in the Honeywell documentation – it will be 1 greater.

Example:

The 127'th input coil would be addressed as '1127' using decimal addressing or 'H17F' using hexadecimal addressing.

Internal Registers

The default data types are shown in **bold**.

	Decimal Addressing	Hexadecimal Addressing
Type	Word , Short, BCD	
Format	3xxxxx	H3yyyyy

Format	Read Only	
Range	1-65536	1-10000

Type	Boolean	
Format	3xxxx.bb	H3yyyyy.c
Format	Read Only	
Range	xxxxx.0-xxxxx.15	yyyyy.0-yyyyy.F

Type	Float, DWord, Long, LBCD	
Format	3xxxxx	H3yyyyy
Format	Read Only	
Range	1-65535	1-FFFF

Note: Honeywell documentation often gives the location of parameters in device memory as a 1-based coil or register number in decimal and a 0-based modbus address in hex. The coil or register number will be the modbus address plus 1. For example, they may speak of "Variable #1", which is at "holding register 6337 with modbus address 0x18C0". This can be a point of confusion when creating tags with this driver. Make sure you **specify the coil or register number, and not the given modbus address** when creating tags with this driver. This number may be expressed in decimal, as in the Honeywell documentation, or in hex if you prefer. However, the coil or register number in hex will not be the same value as the modbus address given in the Honeywell documentation – it will be 1 greater.

Arrays

Arrays are also supported for the holding register addresses. The syntax for declaring an array (using decimal addressing) is as follows:

3xxxx[cols]

with assumed row count of 1 and 3xxxx[rows][cols].

For Word, Short and BCD arrays, the base address + (rows * cols)-1 cannot exceed 65536.

For Float, DWord, Long and Long BCD arrays, the base address + (rows * cols * 2)-1 cannot exceed 65535.

For all arrays, the total number of registers being requested cannot exceed the internal register block size that was specified for this device.

Holding Registers

The default data types are shown in **bold**.

	Decimal Addressing	Hexadecimal Addressing
Type	Word , Short, BCD	
Format	4xxxxx	H4yyyyy
Security	Read/Write	
Range	1-65536	1-10000

Type	Boolean	
Format	4xxxx.bb	H4yyyyy.c
Security	Read/Write	
Range	xxxxx.0-xxxxx.15	yyyyy.0-yyyyy.F

Type	Boolean (Variables)	
Format	4xxxxx	H4yyyyy
Security	Read/Write	
Range	6337 – 7535	18C1 – 1D6F

Type	Boolean (Signal Tags-HC900 Range)	
Format	4xxxxx	H4yyyyy
Security	Read Only	
Range	15201-23200	3B61 – 5AA0

Type	Boolean (Signal Tags-Legacy Range)	
Format	4xxxxx	H4yyyyy
Security	Read Only	
Range	8193-10192	2001-27D0

Type	Float, DWord, Long, LBCD	
Format	4xxxxx	H4yyyyy
Security	Read/Write	
Range	1-65535	1-FFFF

Note: Honeywell documentation often gives the location of parameters in device memory as a 1-based coil or register number in decimal, and a 0-based modbus address in hex. The coil or register number will be the modbus address plus 1. For example, they may speak of "Variable #1", which is at "holding register 6337 with modbus address 0x18C0". This can be a point of confusion when creating tags with this driver. Make sure you specify the coil or register number, and not the given modbus address when creating tags with this driver. This number may be expressed in decimal, as in the Honeywell documentation, or in hex if you prefer. However, the coil or register number in hex will not be the same value as the modbus address given in the Honeywell documentation – it will be 1 greater.

This driver is derived from our Modbus Ethernet driver. Because of this, it allows tremendous flexibility in addressing and data type assignment. No effort is made to restrict address and data type options to reflect the complexities of the data mapping in Honeywell HC900 devices. Thus, it is possible to configure tags that will not access data stored in the device correctly. For example, you could create a tag addressing register 6337 with a data type of word. Such a tag would not be useful here since this is the starting address of "Variable #1", and will always contain floating point data. Likewise, a tag addressing register 6338 as a float would not be useful either, since variables are mapped to registers 6337, 6339, 6341, etc. (These are starting addresses. Each value uses two registers.) Refer to your device documentation for the correct starting address and data type for each parameter you will be creating a tag for.

Variables and Signal Tags

Variables (registers 6337 – 7535) and signal tags (registers 15201 – 23200 and 8193-10192) can be used for either analog or digital data, depending on your device configuration. In either case, the actual data stored in the device will be in IEEE floating point format. For digital data, TRUE/ON will be stored as 1.0, and FALSE/OFF will be stored as 0.0. Thus, you may read and write to any of these locations using tags with a data type of float. However, many client applications will not be able to handle "digital" values represented as floats, especially since TRUE is typically coerced into a float value of -1.0. For these cases, create tags that address digital variables and signals with a data type of Boolean. The driver will automatically convert these values to and from true Booleans.

Arrays

Arrays are also supported for the holding register addresses. The syntax for declaring an array (using decimal addressing) is as follows:

```
4xxxx[cols]
with assumed row count of 1 and 4xxxx[rows][cols].
```

For Word, Short and BCD arrays, the base address + (rows * cols)-1 cannot exceed 65536.

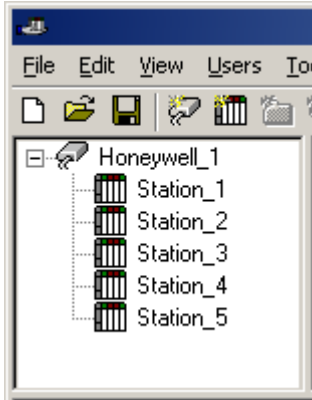
For Float, DWord, Long and Long BCD arrays, the base address + (rows * cols * 2)-1 cannot exceed 65535.

For all arrays, the total number of registers being requested cannot exceed the holding register block size that was specified for this device.

Optimizing Your Honeywell HC900 Ethernet Communications

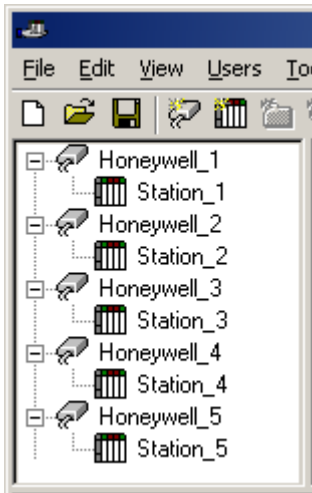
The Honeywell HC900 Ethernet driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the Honeywell HC900 Ethernet driver is fast, there are a couple of guidelines that can be used in order to control and optimize the application and gain maximum performance.

Our server refers to communications protocols like Honeywell HC900 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 Honeywell HC900 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 Honeywell HC900 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 Honeywell HC900 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 Honeywell HC900 Ethernet driver could only define one single channel, then the example shown above would be the only option available; however, the Honeywell HC900 Ethernet driver can define up to 16 channels. Using multiple channels distributes the data collection workload by simultaneously issuing multiple requests to the network. An example of how the same application may appear when configured using multiple channels to improve performance is shown below.



Each device has now been defined under its own channel. In this new configuration, a single path of execution is dedicated to the task of gathering data from each device. If the application has 16 or fewer devices, it can be optimized exactly how it is shown here.

The performance will improve even if the application has more than 16 devices. While 16 or fewer devices may be ideal, the application will still benefit from additional channels. Although 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.

Block Size, which is available on each defined device, can also affect the Honeywell HC900 Ethernet driver's performance. Block Size refers to the number of bytes that may be requested from a device at one time. To refine the performance of this driver, may be configured to 1 to 120 registers and 8 to 800 bits.

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>'](#)

Device Specific Messages

[Failure to initiate 'winsock.dll'](#)
[Bad address in block \[x to y\] on device '<device name>'](#)
[Bad received length \[x to y\] on device '<device name>'](#)

Tag Import Specific Messages

[Could not read record <record>-Buffer length exceeded](#)
[No tags imported-Unsupported file format](#)
[Could not parse expected data \(field <field>, record <record>\)](#)
[Invalid decimal address \(field <field>, record <record>\)](#)
[Invalid tag name '<name>' \(field <field>, record <record>\) could not be coerced into valid name](#)
[Invalid tag name '<old name>' \(field <field>, record <record>\) changed to '<new name>'](#)
[Invalid datatype \(field <field>, record <record>\)](#)
[Invalid access \(field <field>, record <record>\)](#)
[Invalid type \(field <field>, record <record>\)](#)
[Invalid tag type \(field <field>, record <record>\)](#)
[Could not import tag in record <record>-unknown block name](#)
[Invalid block name '<name>' \(field <field>, record <record>\) could not be coerced into valid group name](#)
[Invalid block name '<old name>' \(field <field>, record <record>\) changed to '<new name>'](#)

Address Validation

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:

An invalid tag address has been specified in a dynamic request.

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

Solution:

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

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.

Device Status Messages

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

Device Status Messages

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

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

Device '<device name>' is not responding

Error Type:

Serious

Possible Cause:

1. The connection between the device and the host PC is broken.
2. The communication parameters for the connection are incorrect.
3. The named device may have been assigned an incorrect Network ID.

Solution:

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

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

Error Type:

Serious

Possible Cause:

1. The named device may not be connected to the network.
2. The named device may have been assigned an incorrect Network ID.
3. The named device is not responding to write requests.
4. The address does not exist in the PLC.

Solution:

1. Check the PLC network connections.
2. Verify that the Network ID given to the named device matches that of the actual device.

Device Specific Messages

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

Device Specific Messages

[Failure to initiate 'winsock.dll'](#)

[Bad address in block \[x to y\] on device '<device name>'](#)

[Bad received length \[x to y\] on device '<device name>'](#)

Failure to initiate 'winsock.dll'

Error Type:

Fatal

Possible Cause:

Could not negotiate with the operating systems winsock 1.1 functionality.

Solution:

Verify that the winsock.dll is properly installed on the system.

Bad address in block [x to y] on device '<device name>'

Error Type:

Fatal addresses falling in this block.

Cause:

This error is reported when the driver attempts to read a location in a PLC that does not exist. For example, in a PLC that only has holding registers 40001 to 41400, requesting address 41405 would generate this error. Once this error is generated, the driver will not request the specified block of data from the PLC again. Any other addresses being requested that are in this same block will also go invalid.

Solution:

The client application should be modified to ask for addresses within the range of the device.

Bad received length [x to y] on device '<device name>'

Error Type:

Fatal addresses falling in this block.

Cause:

The driver attempted to read a block of memory in the PLC. The PLC responded with no error, but did not provide the driver with the requested block size of data.

Solution:

Ensure that the range of memory exists for the PLC.

Tag Import Specific Messages

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

Tag Import Specific Messages

[Could not read record <record>-Buffer length exceeded](#)

[No tags imported-Unsupported file format](#)

[Could not parse expected data \(field <field>, record <record>\)](#)

[Invalid decimal address \(field <field>, record <record>\)](#)

[Invalid tag name '<name>' \(field <field>, record <record>\) could not be coerced into valid name](#)

[Invalid tag name '<old name>' \(field <field>, record <record>\) changed to '<new name>'](#)

[Invalid datatype \(field <field>, record <record>\)](#)

[Invalid access \(field <field>, record <record>\)](#)

[Invalid type \(field <field>, record <record>\)](#)

[Invalid tag type \(field <field>, record <record>\)](#)

[Could not import tag in record <record>-unknown block name](#)

[Invalid block name '<name>' \(field <field>, record <record>\) could not be coerced into valid group name](#)

[Invalid block name '<old name>' \(field <field>, record <record>\) changed to '<new name>'](#)

Could not read record <record>-Buffer length exceeded

Error Type:

Warning

Possible Cause:

1. A record exceeded the maximum allowed length of 1024 characters.
2. File may be corrupted.

Solution:

1. Edit tag name and description data to reduce length.
2. Regenerate file.

No tags imported-Unsupported file format

Error Type:

Warning

Possible Cause:

The import file is not one of the types the driver is able to read.

Solution:

Change the import file type to one that is supported.

See Also:

[Creating Tag Import Files](#)

Could not parse expected data (field <field>, record <record>)

Error Type:

Warning

Possible Cause:

1. The field exceed the maximum allowed length of 256 characters.
2. The field delimiter (comma) is missing, possibly due to a file editing error.
3. The file is corrupted.

Solution:

1. Edit the specified field to reduce length if possible.
2. Edit the file and replace the delimiter.
3. Regenerate the file.

Invalid decimal address (field <field>, record <record>)

Error Type:

Warning

Possible Cause:

1. The data in the specified field is not a decimal value, possibly due to a file editing error.
2. The file is not in one of the supported formats.
3. The file is corrupted.

Solution:

1. Edit the file and correct the specified field.
2. Regenerate the file.

See Also:

[Creating Tag Import Files](#)

Invalid tag name '<name>' (field <field>, record <record>) could not be coerced into valid name

Error Type:

Warning

Possible Cause:

The tag name specified in the file is not a valid OPC Server Tag name, and the driver's tag name modification mechanism was unable to create a unique and valid name based on the field.

Solution:

Manually create the tag.

Invalid tag name '<old name>' (field <field>, record <record>) changed to '<new name>'

Error Type:

Information

Possible Cause:

This is not truly an error. Tag names that are valid in the Hybrid Control Designer are not necessary valid in the OPC Server. The driver created a valid tag name based on the name read from the file.

Solution:

N/A

Invalid datatype (field <field>, record <record>)

Error Type:

Warning

Possible Cause:

1. The datatype in the specified field is not one of the types supported by the driver, possibly due to a file editing error.
2. The file is not in one of the supported formats.
3. The file is corrupted.

Solution:

1. Edit the file and change the specified field to a supported datatype. Supported types are: "unsigned 16", "signed 16", "unsigned 32", "signed 32", and "float 32".
2. Regenerate the file.

See Also:[Creating Tag Import Files](#)

Invalid access (field <field>, record <record>)

Error Type:

Warning

Possible Cause:

1. The access type in the specified field is not one of the types supported by the driver, possible due to a file editing error.
2. The file is not in one of the supported formats.
3. The file is corrupted.

Solution:

1. Edit the file and change the specified field to a supported access types. Supported types are: "R", "W", and "R/W".
2. Regenerate the file.

See Also:[Creating Tag Import Files](#)

Invalid type (field <field>, record <record>)

Error Type:

Warning

Possible Cause:

1. The type in the specified field is not one of the types expected by the driver, possible due to a file editing error.
2. The file is not in one of the supported formats.
3. The file is corrupted.

Solution:

1. Edit the file and change the specified field to a supported type. Expected types are "Variable" and "Signal Tag". There are many other types used for function block data, but the driver should not be trying to process these when importing a function block file.
2. Regenerate the file.

See Also:[Creating Tag Import Files](#)

Invalid tag type (field <field>, record <record>)

Error Type:

Warning

Possible Cause:

1. The tag type in the specified field is not one of the types expected by the driver, possible due to a file editing error.
2. The file is not in one of the expected formats.
3. The file is corrupted.

Solution:

1. Edit the file and change the specified field to a supported tag type. Expected tag types are "Digital" and "Analog".
2. Regenerate the file.

See Also:

[Creating Tag Import Files](#)

Could not import tag in record <record>-unknown block name

Error Type:

Warning

Possible Cause:

The driver read a header that indicated that the file was a Detailed Function Block Report. The driver did not find an expected block sub-header record.

Solution:

Verify that the file is a Detailed Function Block Report, and that the block sub-headers are present. Regenerate the file if needed.

Invalid block name '<name>' (field <field>, record <record>) could not be coerced

Error Type:

Warning

Possible Cause:

The tag name specified in the file is not a valid OPC Server Group name, and the driver's tag name modification mechanism was unable to create a unique and valid name based on field.

Solution:

Edit the file and modify the tag name in the block subheader.

Invalid block name '<old name>' (field <field>, record <record>) changed to '<new>'

Error Type:

Information

Possible Cause:

This is not truly an error. Tag names that are valid in the Hybrid Control Designer are not necessary valid in the OPC Server. The driver created a valid group name based on the name read from the file.

Solution:

N/A

Index

- A -

Add Import File 11
 Address '<address>' is out of range for the specified device or register 20
 Address Descriptions 15
 Array size is out of range for address '<address>' 21
 Array support is not available for the specified address: '<address>' 21
 Automatic Tag Database Generation 6

- B -

Bad address in block [x to y] on device '<device name>' 22
 Bad received length [x to y] on device '<device name>' 22
 BCD 6
 Block Sizes 5
 Boolean 6

- C -

Could not import tag in record <record> - unknown block name 26
 Could not parse expected data (field <field>_ record <record>) 23
 Could not read record <record> - Buffer length exceeded 23
 Creating Tag Import Files 11

- D -

Data Type '<type>' is not valid for device address '<address>' 20
 Data Types Description 6
 Device '<device name>' is not responding 21
 Device address '<address>' contains a syntax error 20
 Device address '<address>' is not supported by model '<model name>' 20
 Device address '<address>' is Read Only 20
 Device Setup 3

DWord 6

- E -

Error Descriptions 18

- F -

Failure to initiate 'winsock.dll' 22
 Float 6

- H -

Holding Register 15
 Holding Registers 16

- I -

Input Coils 15
 Internal Registers 15
 Invalid access (field <field>_ record <record>) 25
 Invalid block name '<name>' (field <field>_ record <record>) could not be coerced 26
 Invalid block name '<old name>' (field <field>_ record <record>) changed to '<new>' 26
 Invalid datatype (field <field>_ record <record>) 24
 Invalid decimal address (field <field>_ record <record>) 24
 Invalid tag name '<name>' (field <field>_ record <record>) could not be coerced into valid name 24
 Invalid tag name '<old name>' (field <field>_ record <record>) changed to '<new name>' 24
 Invalid tag type (field <field>_ record <record>) 25
 Invalid type (field <field>_ record <record>) 25

- L -

LBCD 6
 Long 6

- M -

Missing address 19

Modbus Partitions - Custom Modbus Map 7, 9

- N -

No tags imported - Unsupported file format 23

- O -

Optimizing Your Honeywell HC Ethernet
Communications 18

Output Coils 15

Overview 3

- S -

Settings 4

Short 6

- T -

Tag Generation 7

Tag Generation SP Programmer Details 8

Tag Import 9

TCP/IP 4

- U -

Unable to write to '<address>' on device '<device
name>.' 21

- W -

Word 6