# Hilscher Universal Driver Help

# Table of Contents

# Hilscher Universal Driver Help

Help version 1.020

**CONTENTS**



## Overview

The Hilscher Universal Driver provides an easy and reliable way to connect Hilscher Universal devices to OPC client applications, including HMI, SCADA, Historian, MES, ERP and countless custom applications. It is intended for use with Hilscher Communications Interface (CIF) cards. I/O and diagnostic information is available through the OPC server. The CIF cards currently supported are DeviceNet Master/Slave and Profibus DP Master/Slave.

## External Dependencies

This driver has external dependencies. It requires that SyCon (Hilscher's System Configuration Software) and the correct PCI interface card (CIF 50 models) be installed on the same machine as the OPC server.

## Channel Setup

A channel represents the SyCon Configuration Database, which includes the board assignment and device/module definitions. The device/module definitions are imported from the configuration file. For more information, refer to **SyCon Database Import**.

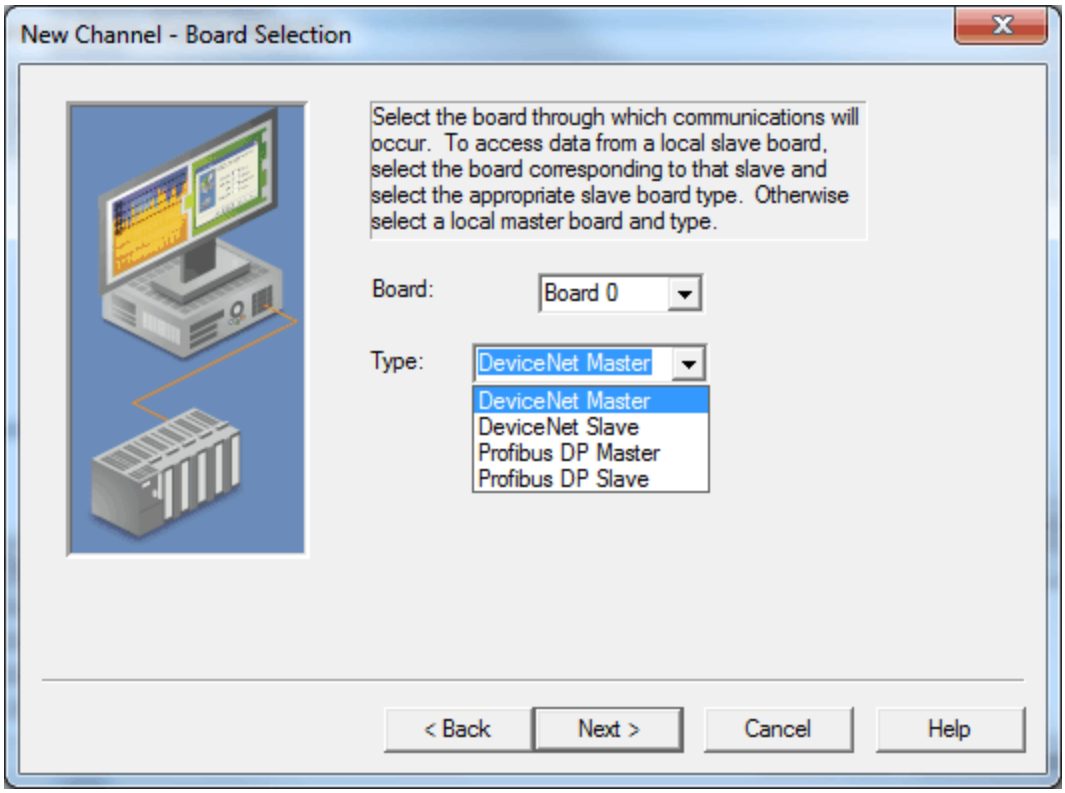
Select a link from the list below for information on a specific aspect of Channel Setup.

**Board Selection**
How do I select the board number and bus type over which communications will occur?

**Slave Board Configuration**
How do I configure a local Hilscher Slave board? How do I configure the server to communicate directly with the local Slave board?

**SyCon Database Import**
How do I specify the location of the SyCon Configuration Database?

## Board Selection




Descriptions of the parameters are as follows:

- **Board:** This parameter specifies the board on which communications will occur for the given channel. In the drop-down menu, Board x correlates to Board x in SyCon.
- **Type:** This parameter specifies the selected board's bus type and master/slave type (if applicable).

### Supported Board Types

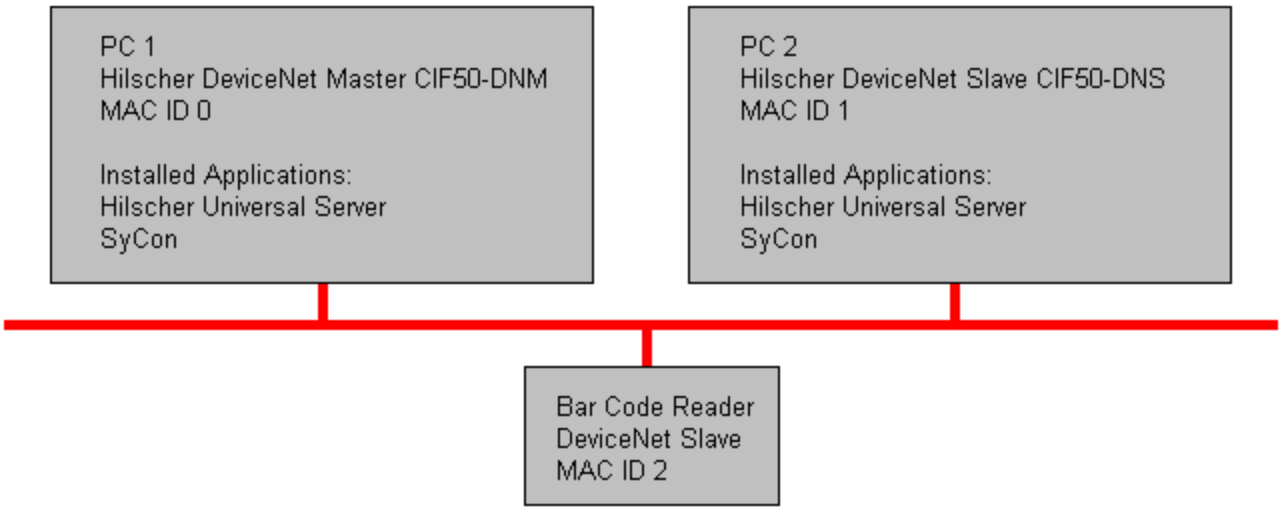
| | | |
|---|---|---|
| DeviceNet Master | Local DeviceNet Master board (such as CIF50-DNM). | Read/Write to DeviceNet slave I/O. |
| DeviceNet Slave | Local DeviceNet Slave board (CIF50-DNS). | Read/Write to local board I/O only. |
| Profibus DP Master | Local Profibus Master board (such as CIF50-PB). | Read/Write to Profibus slave I/O. |
| Profibus DP Slave | Local Profibus Slave board (CIF50-DPS). | Read/Write to local board I/O only. |

## Slave Board Configuration

Once configured, a local Hilscher Slave board can communicate both locally and with the Master. For information on configuration (and local/Master communication) refer to the instructions below.

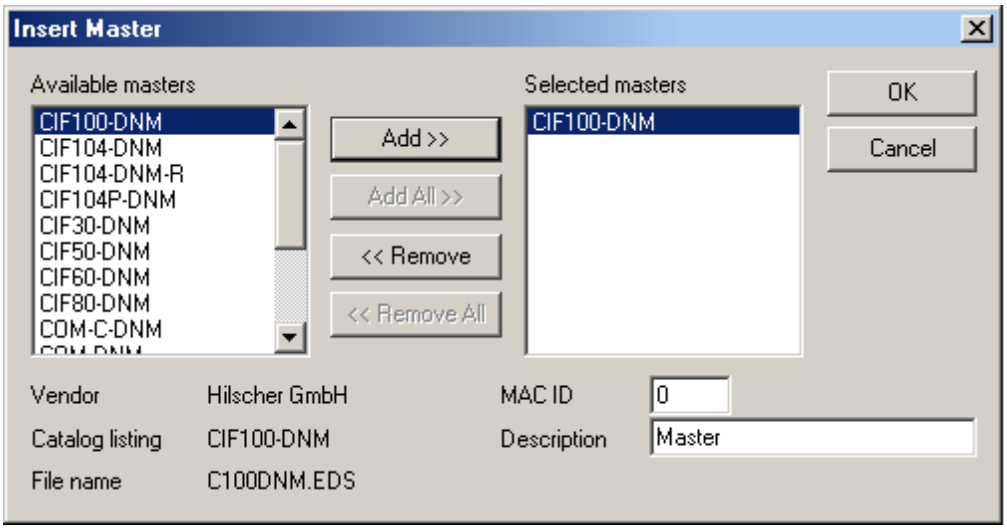
**Note:** This tutorial uses DeviceNet as the example network, but the same steps may be applied to Profibus.

**Important:** Two SyCon configurations are required in order to communicate to the Slave board locally and from the Master. One must be from the Master's perspective and one must be from the Slave's perspective. In the example below, the Master board is in PC 1 and the Slave board is in PC 2. The Slave board is configured before the Master.



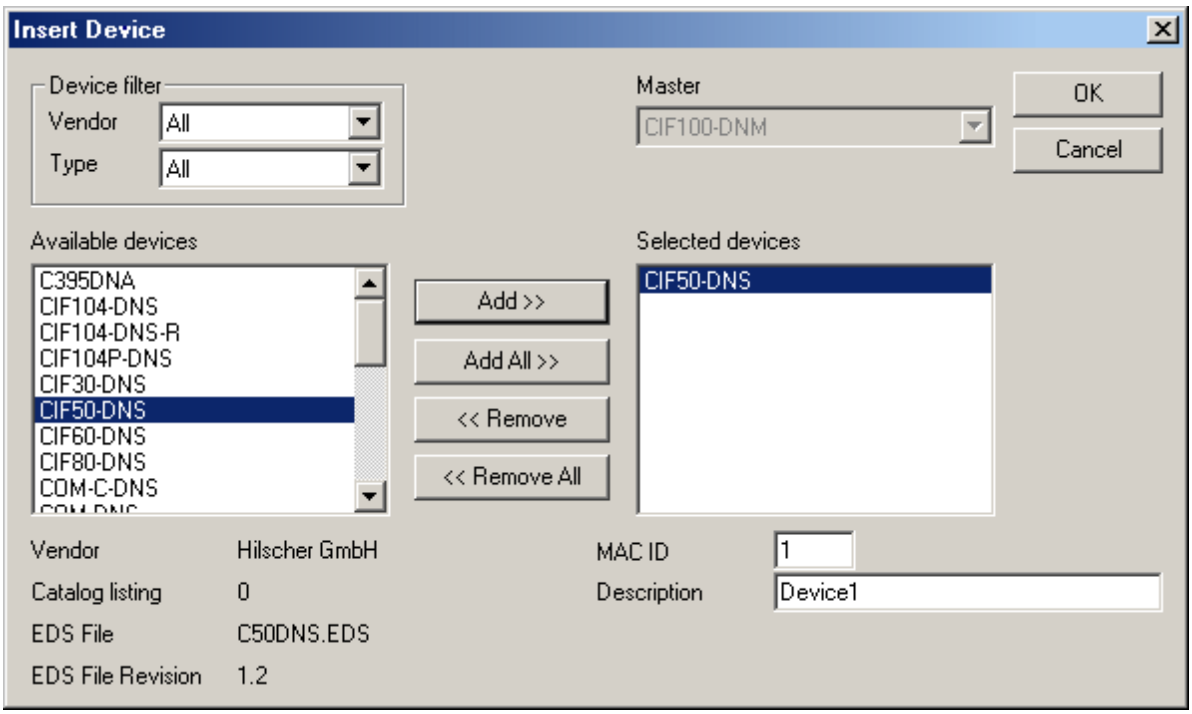

### PC 2 SyCon Configuration (Slave)

1. To start, open an empty SyCon project and insert a "dummy" Master. The master chosen is irrelevant since the configuration will be downloaded to the Slave.

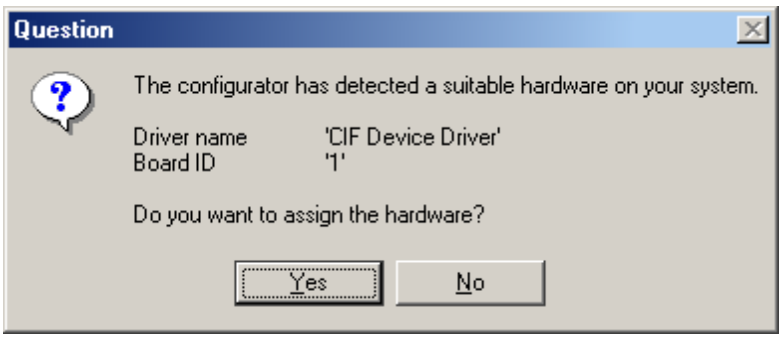



**Note:** When prompted, do not assign the hardware if the Master board is in the same machine as the Slave board.

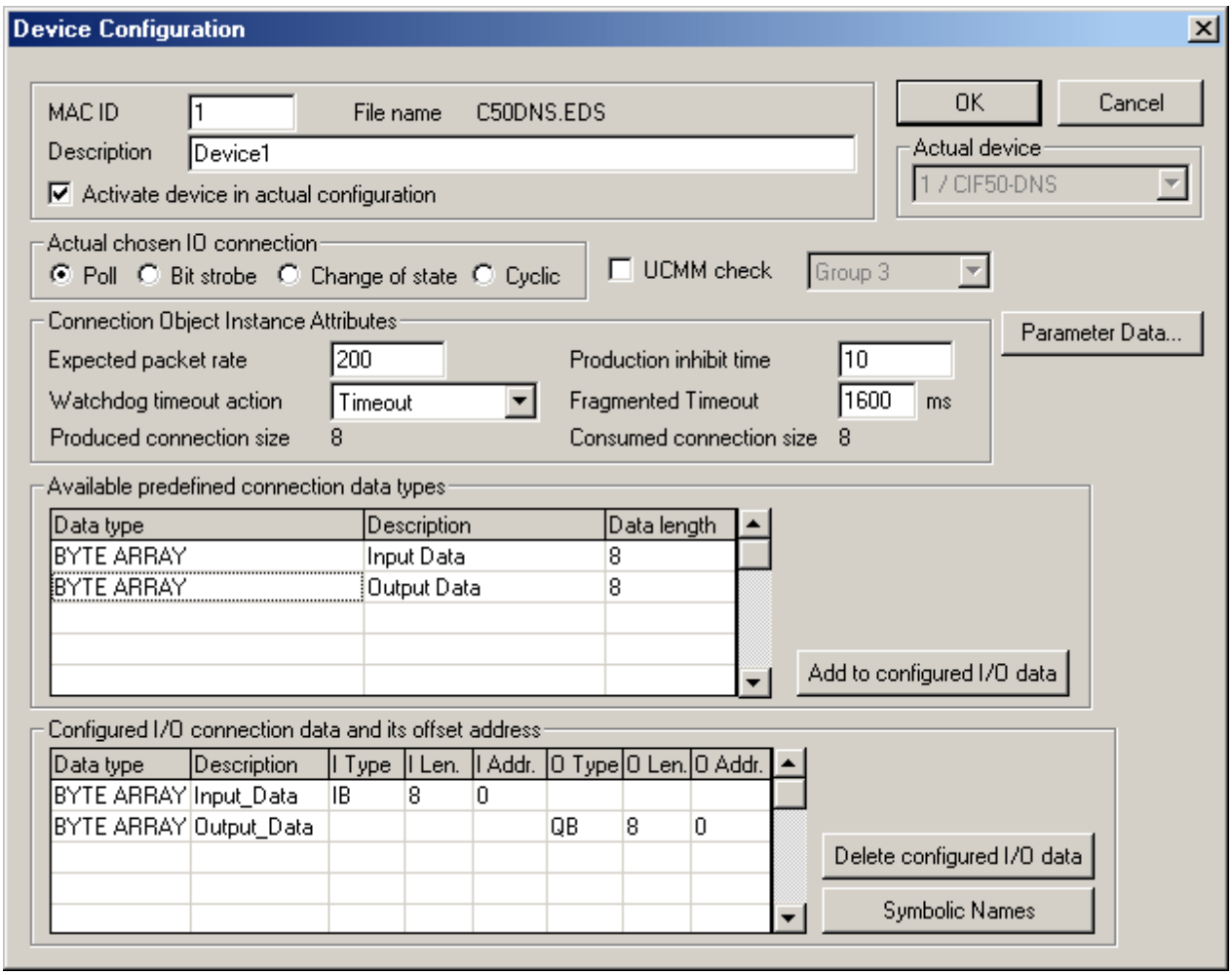
2. Next, insert the Slave and select a unique MAC ID.



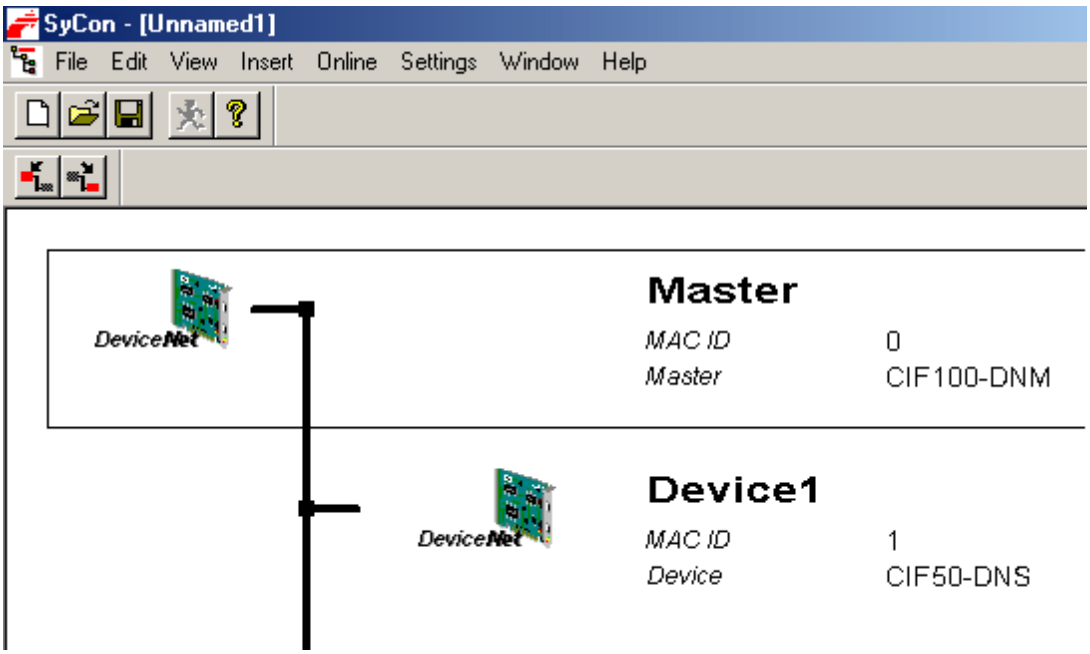

3. Assign the MAC ID to the local Slave board.



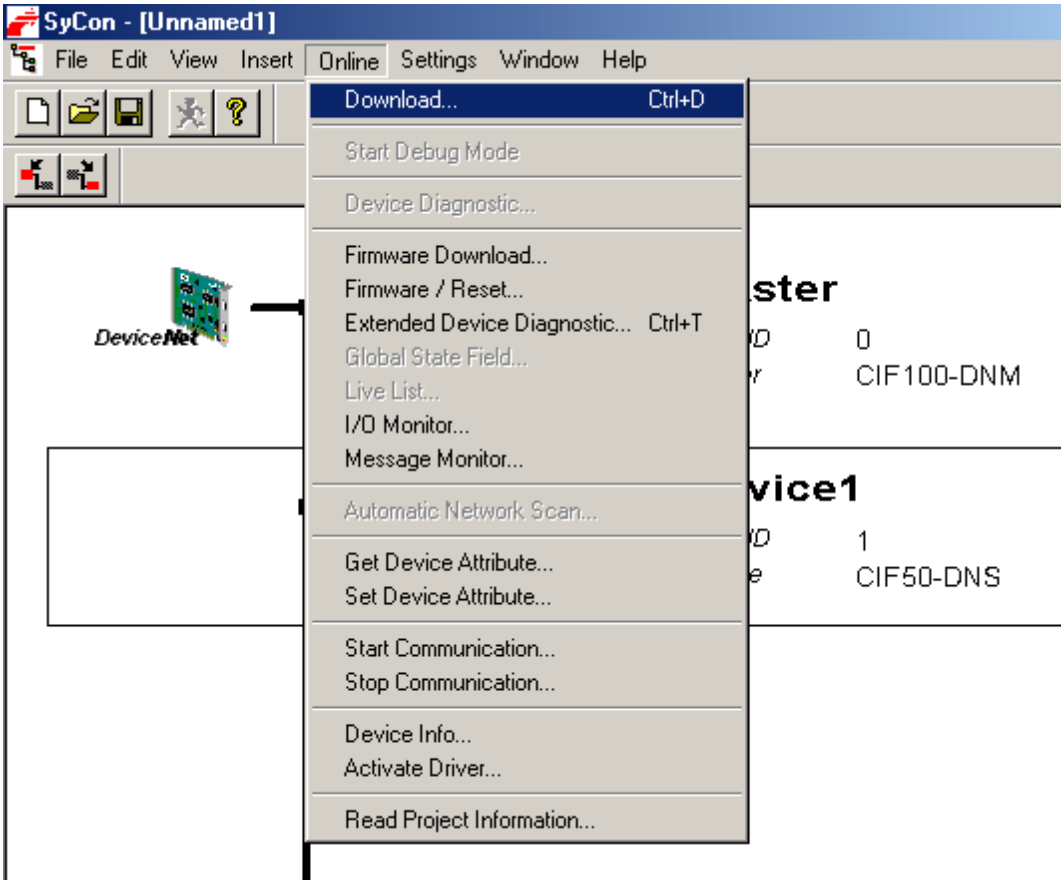

4. Then, configure the I/O.




**Note:** The network should appear as shown below.



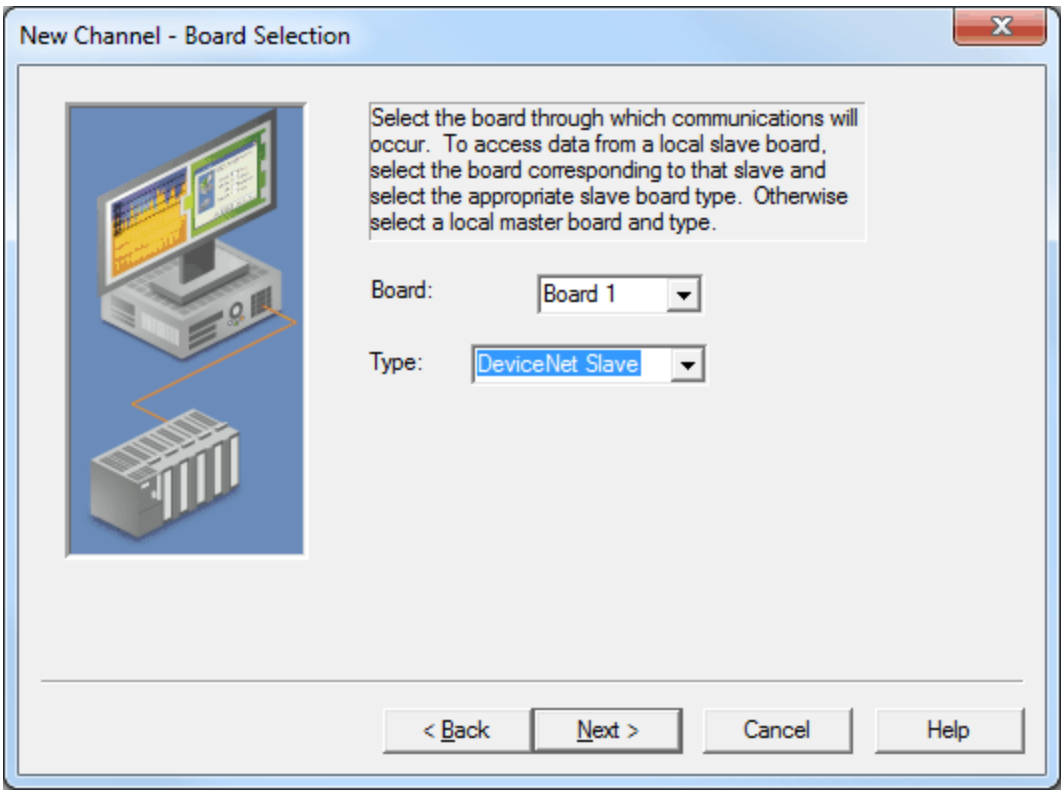
5. Save and then download the configuration to the Slave board by clicking **Online** | **Download**. This is the configuration that will be imported into the server later.



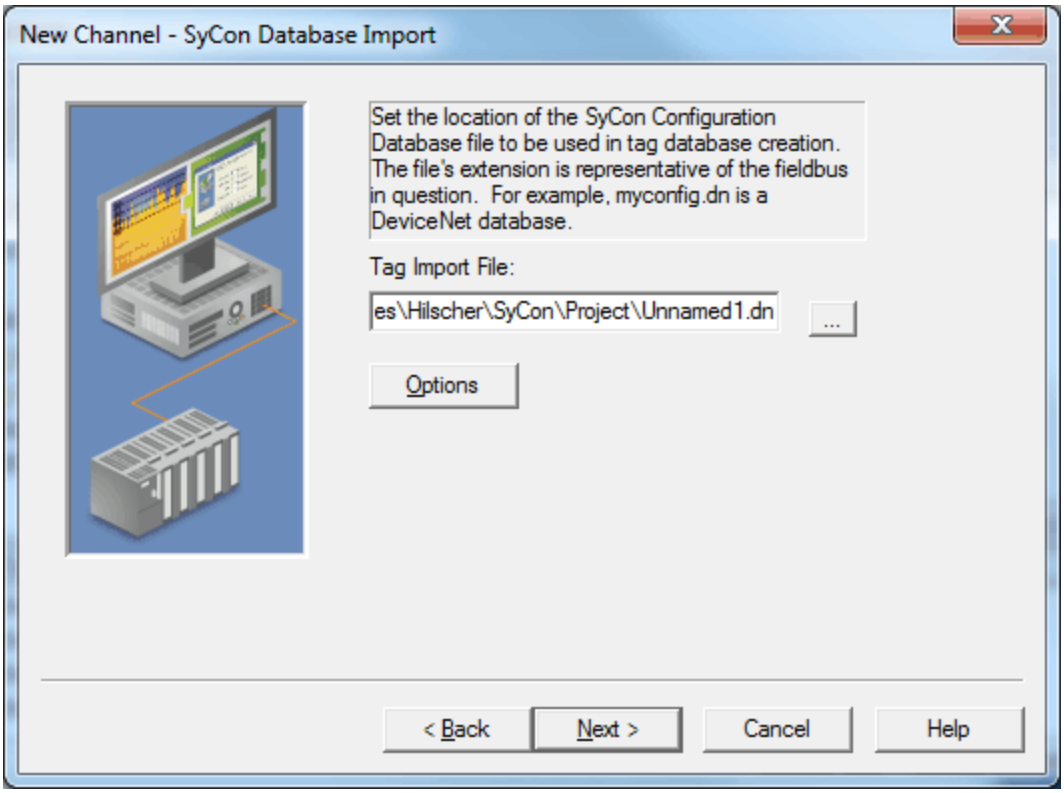

### PC 2 Server Configuration (Slave)

Once the Slave board is configured, it can be accessed locally using the Hilscher Universal Driver.

1. In the server, create a new channel. In the **Board Selection** dialog, choose the location of the board in the PC and then choose **DeviceNet Slave** as a board type.



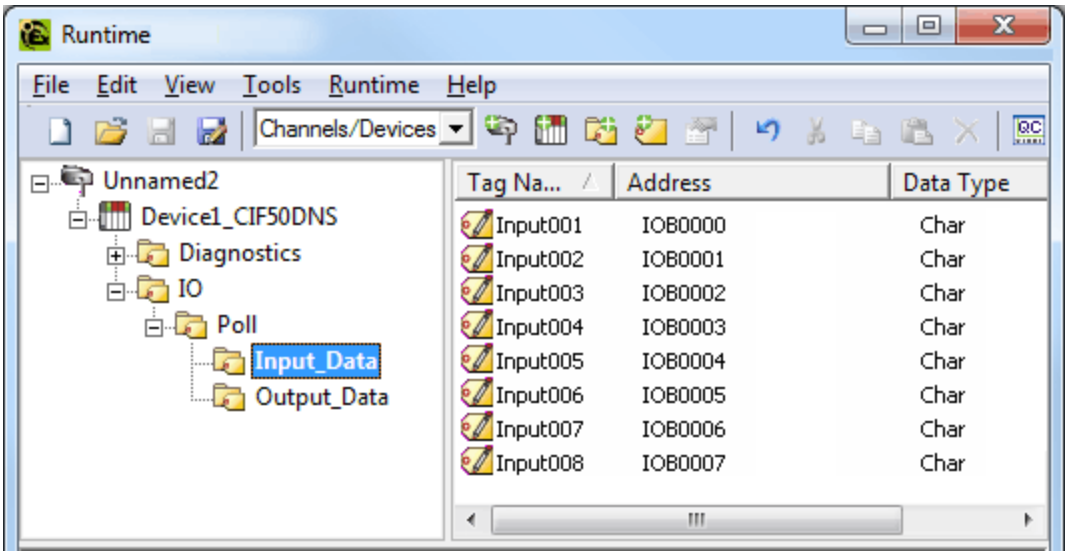

2. In the **SyCon Database Import** dialog, choose the SyCon configuration that was previously created.




3. Complete the New Channel Wizard.
4. Next, create a new device and set the Device ID to the MAC ID of the Slave board. In this example, it is 1.

5. Automatically generate the tags for the Slave. To do so, click **Channel Properties** | **SyCon Database**.

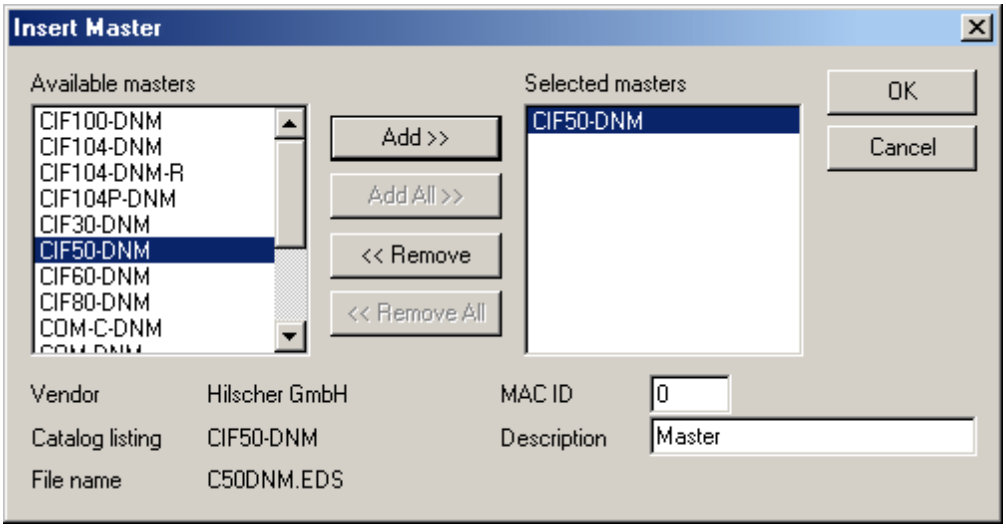
   **Note:** The image below displays the tags created for the configured Slave board.



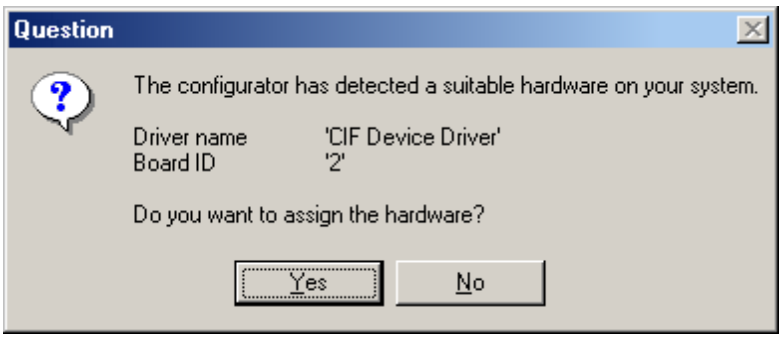

**PC 1 SyCon Configuration (Master)**

Assume that the Slave board in PC 2 is configured for communications and access it from the DeviceNet Master in PC 1. A Master board configuration is required.

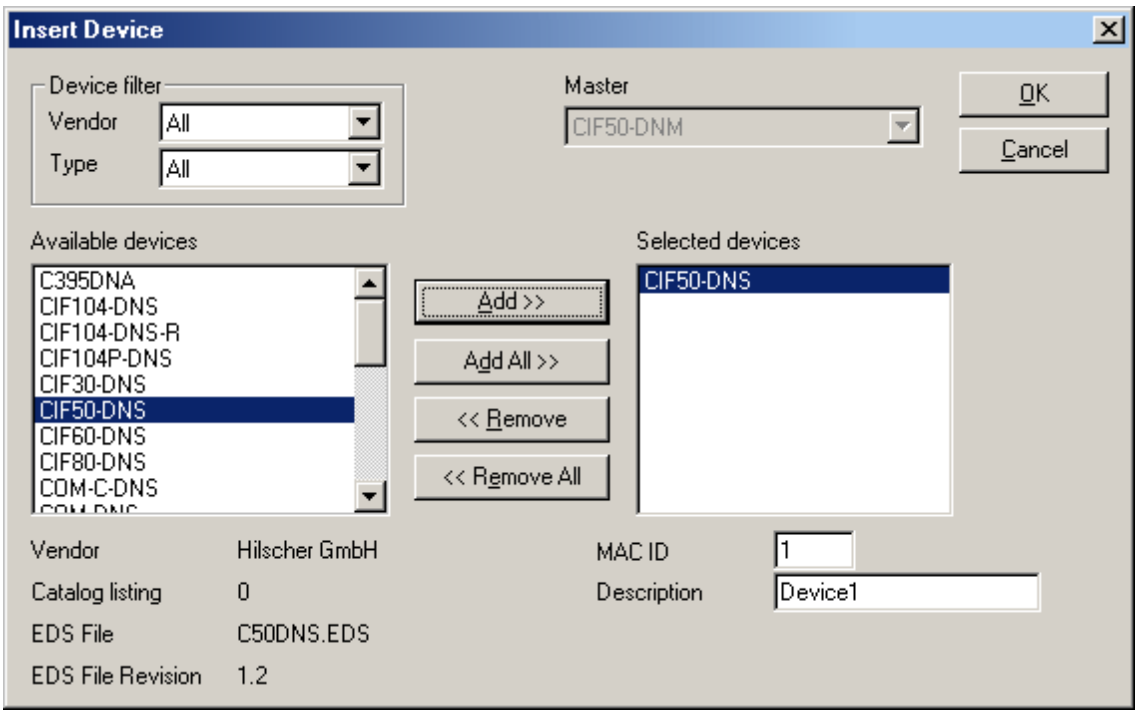
1. First, open an empty SyCon project and insert the Master.



2. Assign it to the local Master board.

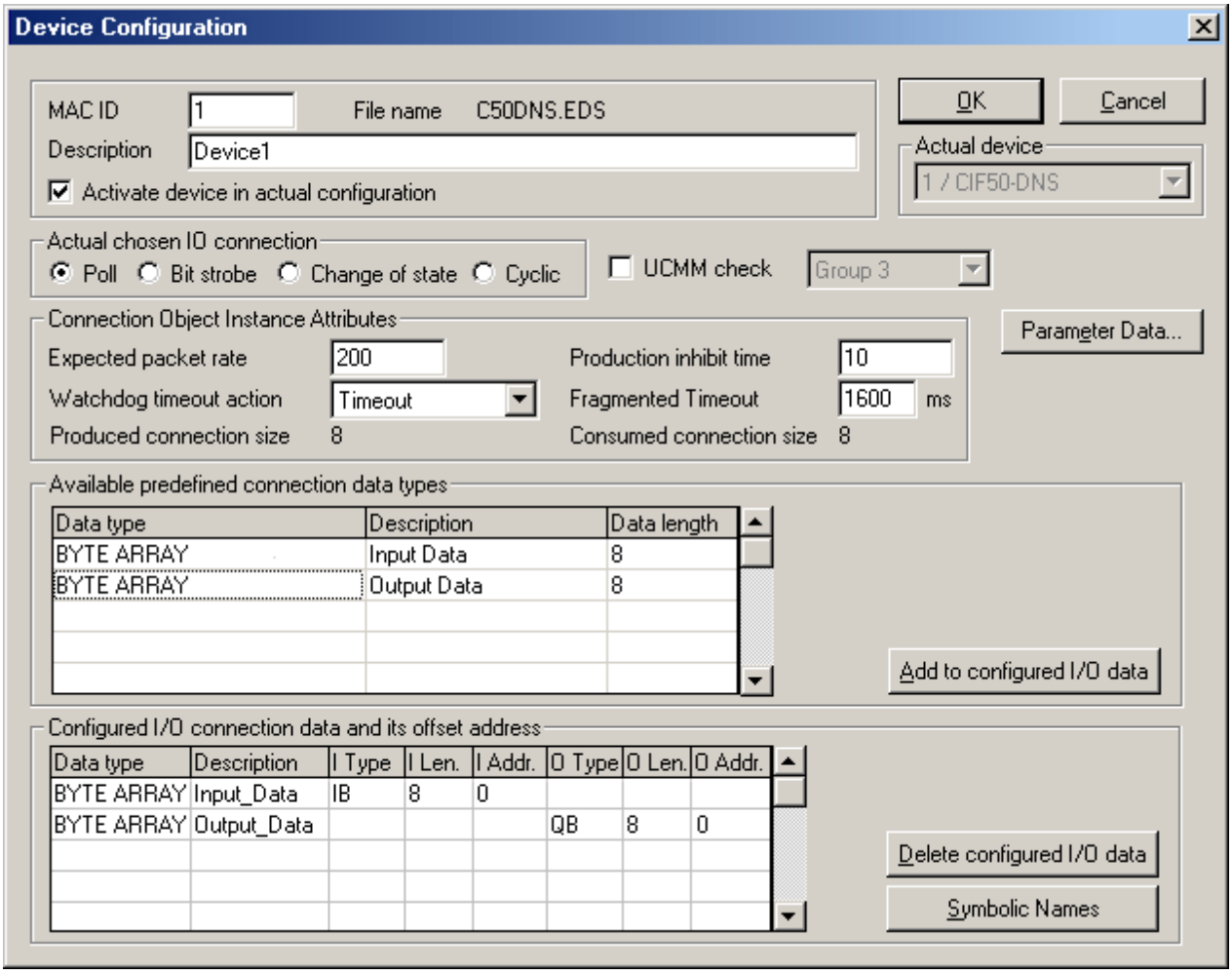



3. Next, insert the Slave. Alternatively, perform **Automatic Network Configuration**.
4. Assign the MAC ID chosen in Step 2 of PC 2 SyCon Configuration (Slave).

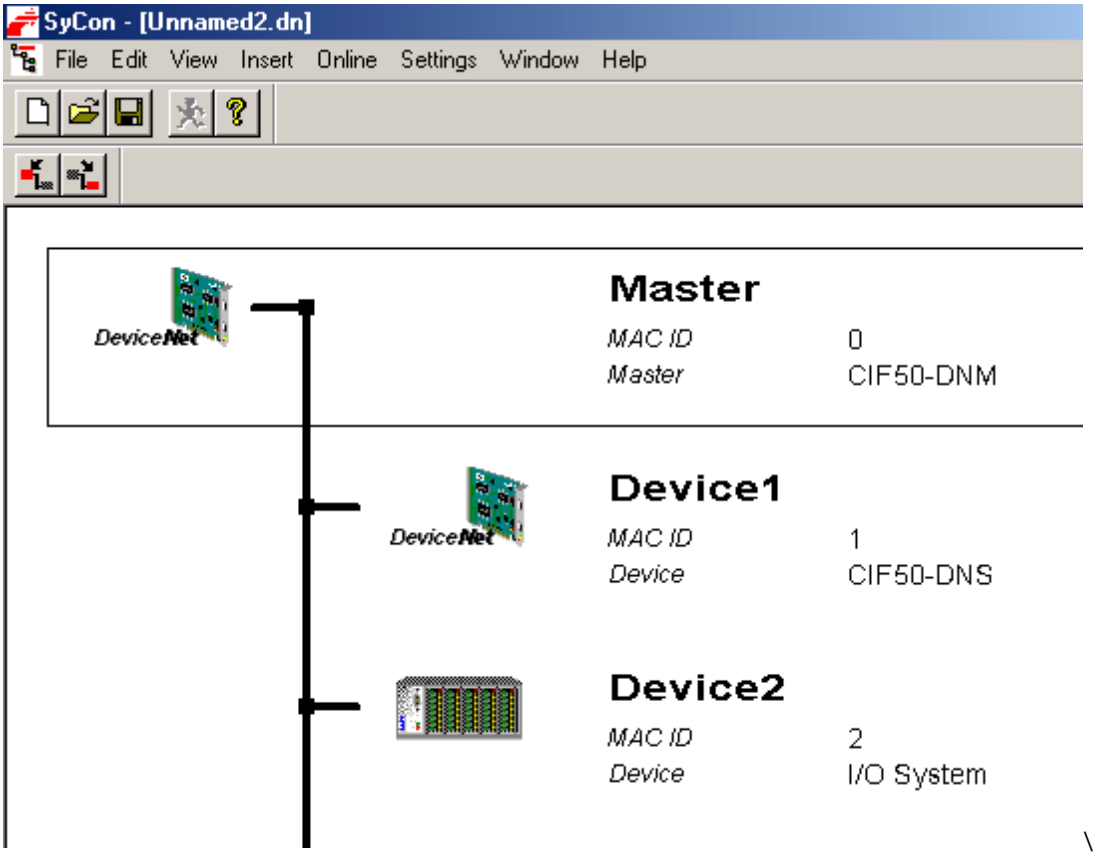



**Note:** Do not assign the hardware when prompted if the Master board is in the same machine as the Slave board.

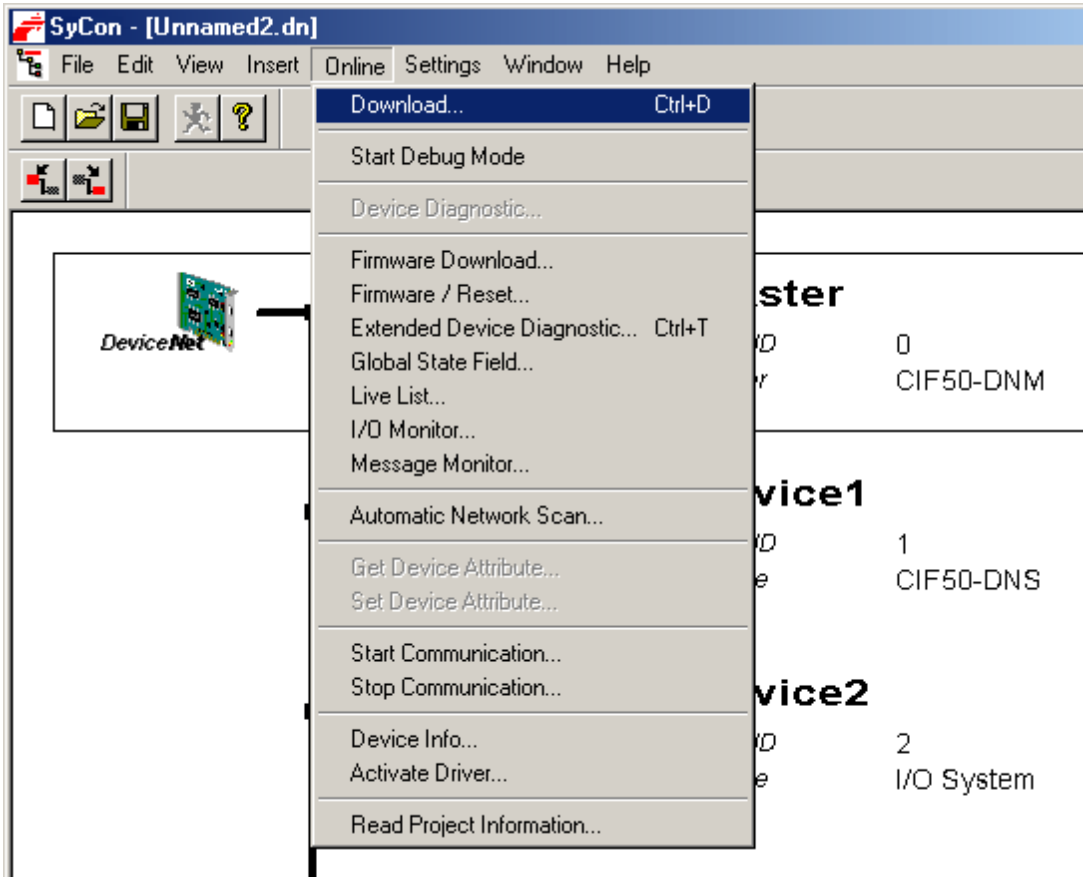
5. Configure the I/O. Its values must match the configured I/O in the Slave SyCon configuration created earlier.




6. If desired, insert the **Bar Code Reader Slave**. This is not required. The resulting network should appear as displayed below.


SyCon - [Unnamed2.dn]
File Edit View Insert Online Settings Window Help
DeviceNet
Master
MAC ID 0
Master CIF50-DNM
DeviceNet
Device1
MAC ID 1
Device CIF50-DNS
Device2
MAC ID 2
Device I/O System

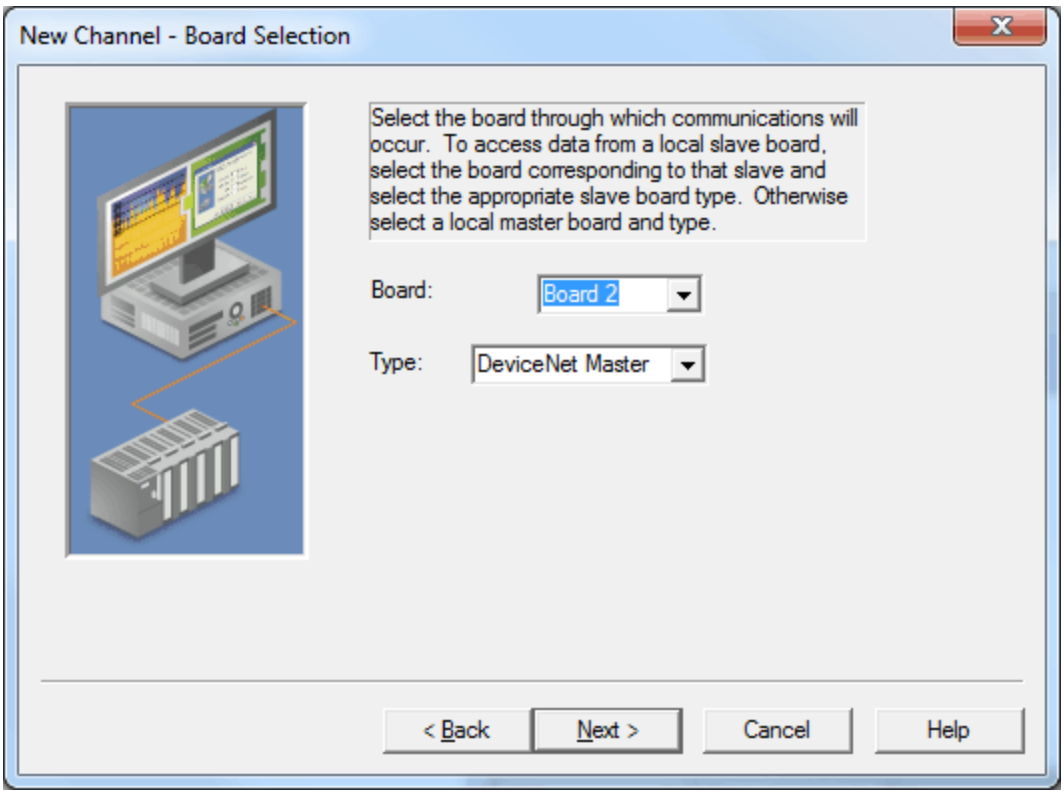
7. Save and then download the configuration to the Master board by clicking **Online** | **Download**.



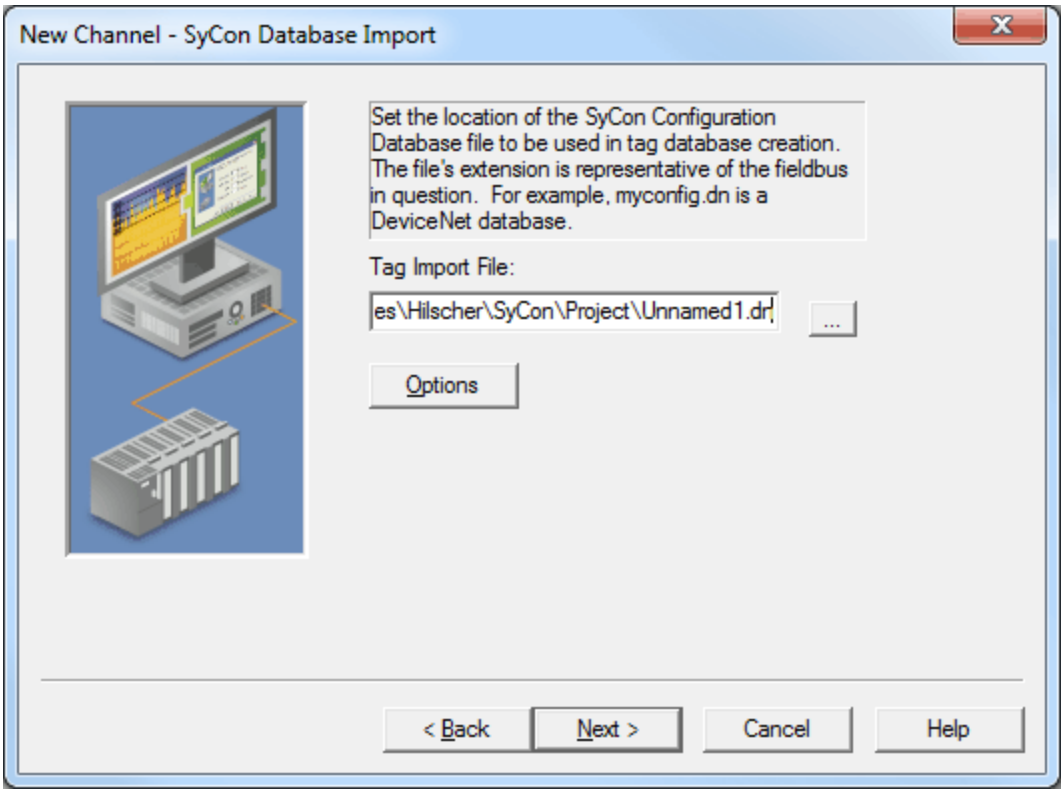

**PC 1 Server Configuration (Master)**

Now that the Master board is configured, the Slave board can be accessed remotely using the Hilscher Universal Driver.

1. In the server, create a new channel. In the **Board Selection** dialog, choose the location of the board in the PC. For board type, select **DeviceNet Master**.

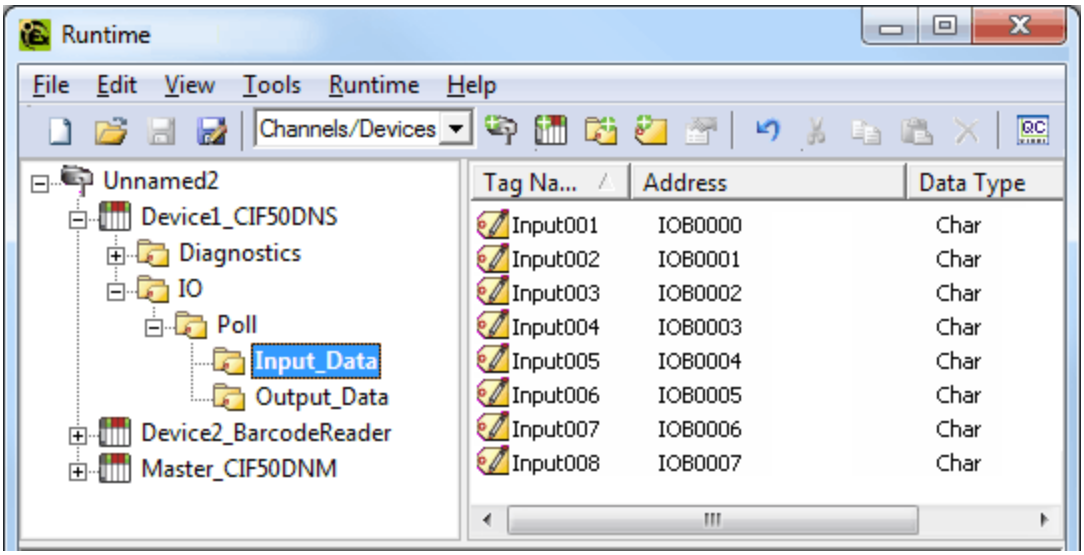



2. In the **SyCon Database Import** dialog, choose the SyCon configuration that was previously created.



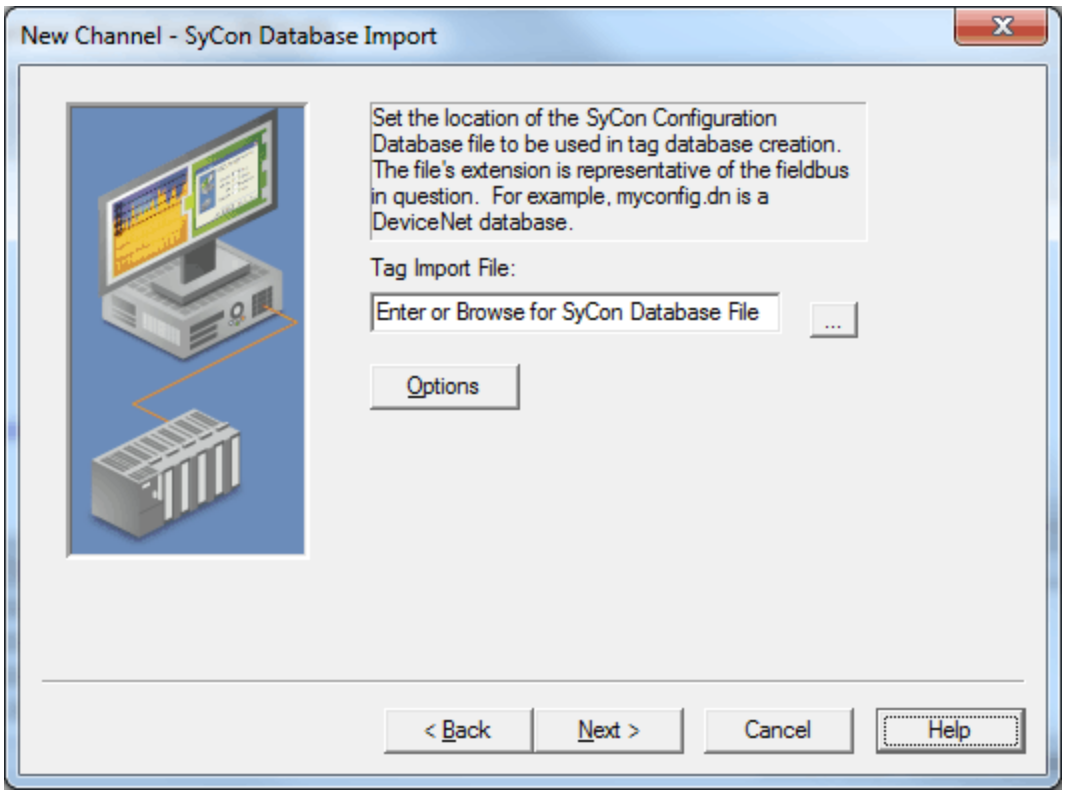

3. Complete the New Channel Wizard.

4. Next, create a new device and set the Device ID to the MAC ID of the Slave board in PC 2. In this example, it is 1. Then, create a new device and set the Device ID to the MAC ID of the Bar Code Reader. In this example, it is 2.
5. Automatically generate the tags for the Slaves. To do so, click **Channel Properties** | **SyCon Database**.
6. The image below displays the tags created for the Slave board in PC 2.




**Important:** If the Slave board is in the same machine as the Master board, the Slave configuration may need to be redownloaded after downloading the Master configuration.

## SyCon Database Import




Descriptions of the parameters are as follows:

- **Tag Import File:** This parameter specifies the exact location of the SyCon configuration database from which the tags will be imported. This file will be used when Automatic Tag Database Generation is instructed to create the tag database.

  **Note:** In order to configure I/O for a Hilscher Slave board (CIF50-DNS), a separate SyCon configuration database must be created for each Slave board. This is not to be confused with the SyCon configuration database set up for the Master. When a channel's board type is specified as a Slave-type, the SyCon Database specified must correspond specifically to this Slave. For more information, refer to **Slave Board Configuration**.
- **Options:** This button invokes the SyCon Database Import OPtions. For more information, refer to **Database Options**.

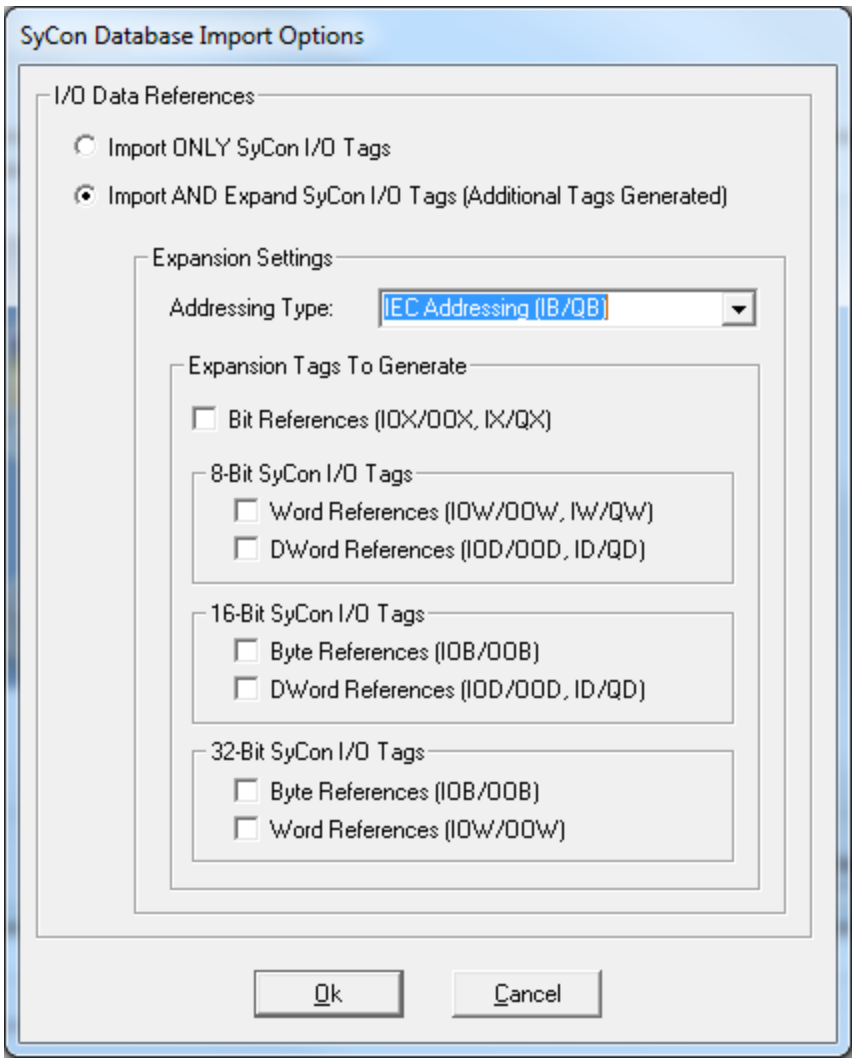
### Supported Databases

Profibus-DP Database Extension: .pb
DeviceNet Database Extension: .dn

**See Also:** **Automatic Tag Database Generation**

## Database Options

For more information on a specific parameter or topic, click on the region in the image below.




## I/O Data References

### Import ONLY SyCon I/O Tags

When selected, this option only imports the I/O symbolic names that are configured in SyCon. Tags will be generated under the following folder:

*IO\\<bus dependant module path>*

### Import AND Expand SyCon I/O Tags

When selected, this option imports the I/O Symbolic Names configured in SyCon and also generates alternative references for each symbolic name. These alternative references (or Expansion Tags) provide different ways of looking at the same piece of data. Tags will be generated under the following folders:

*IO\\<bus dependant module path>\\PI*
*IO\\<bus dependant module path>\\IEC*

**Note:** Depending on the Expansion Tag addressing type, tags may be generated to both of these folders. For more information, refer to **Addressing Type**.

### Examples

**SyCon**

Byte Addressing should be assumed. For more information on Byte and Word addressing modes, refer to **Address Descriptions**. Details are as follows:

- **Configured I/O Module:** Word Array, QW, Length 4, Offset 0.
- **Symbolic Names:** Default is Output001, Offset 0, Word.

**OPC Server**

For Expansion Tags, 16 bit Module Data / Byte References should be assumed. The following tags will be generated in the OPC server under the specific options:

- **Import only SyCon I/O Tags:** Output001, Offset 0, Word.
- **Import and Expand SyCon I/O Tags:** Output001, Offset 0, Word; Output001_B0_QB, Offset 0, Byte; Output001_B1_QB, Offset 1, Byte.

### Symbolic Names and Expansion Tags Differences

SyCon creates default symbolic names based on the Configured I/O. Users can create additional symbolic names (such as Bit, Word, DWord and String tags) within SyCon. Each SyCon symbolic name is imported as an individual tag in the OPC server. Expansion tags are an expansion of the symbolic name tags imported, regardless of whether or not the symbolic name tag is an expansion of the Configured I/O.

### Examples

**SyCon**

Byte addressing should be assumed.

- **Configured I/O Module:** Byte Array, IB, Length 10, Offset 0.
- **Symbolic Names:** Default: Input001, Offset 0, Byte.

  **Note:** These were created for the module MyWordTag, Offset 0, Word.

**OPC Server**

For Expansion Tags, Generate Bit References should be assumed. The following tags will be generated in the OPC server under the specific options:

- **Import only SyCon I/O Tags:** Input001, Offset 0, Byte; MyWordTag, Offset 0, Word.
- **Import and Expand SyCon I/O Tags:**
  Input001, Offset 0, Byte
  Input001_IX_00, Offset 0, Bit 0
  Input001_IX_01, Offset 0, Bit 1
  ...
  Input001_IX_07, Offset 0, Bit 7
  MyWordTag, Offset 0, Word
  MyWordTag_B0_IX_00, Offset 0, Bit 0
  ...
  MyWordTag_B0_IX_07, Offset 0, Bit 7
  MyWordTag_B1_IX_00, Offset 1, Bit 0
  ...
  MyWordTag_B1_IX_07, Offset 0, Bit 7

**Note:** The example above shows how all symbolic name tags are expanded based on the chosen settings. Only bit references were generated additionally.

**Important:** Using Expansion Tags should eliminate the need to create additional symbolic names in SyCon. The exception to this is Strings, since these are not included in the Expansion Settings.

## Expansion Tag Addressing Type

There are two types of I/O addresses supported in the Hilscher Universal Driver: Process Image Offset Addressing and IEC Addressing. Descriptions are as follows:

- **Process Image Offset Addressing:** Address mnemonic and offsets are based on the physical offset into the Master's Process Image Memory Map (I/O Data). Addresses are always byte-based, regardless of the addressing mode selected in SyCon's Master Settings.

  *IO\\<bus dependant module path>\\PI*

  **Note:** Expansion Tags will be generated in a tag group labeled "PI". All tag addresses in this group will be based on Process Image Addressing.
- **IEC Addressing:** Address mnemonic and offsets are based on standard Siemens addressing (IB, IW, ID). Addresses are byte-based or word-based, depending on the addressing mode selected in SyCon's Master Settings.

  *IO\\<bus dependant module path>\\IEC*

  **Note:** Expansion Tags will be generated in a tag group labeled "IEC". All tag addresses in this group will be based on IEC Addressing.

**Important:** Expansion tags will be generated for both addressing types. Two tag groups, "PI" and "IEC," will exist along with their respective Expansion Tags.

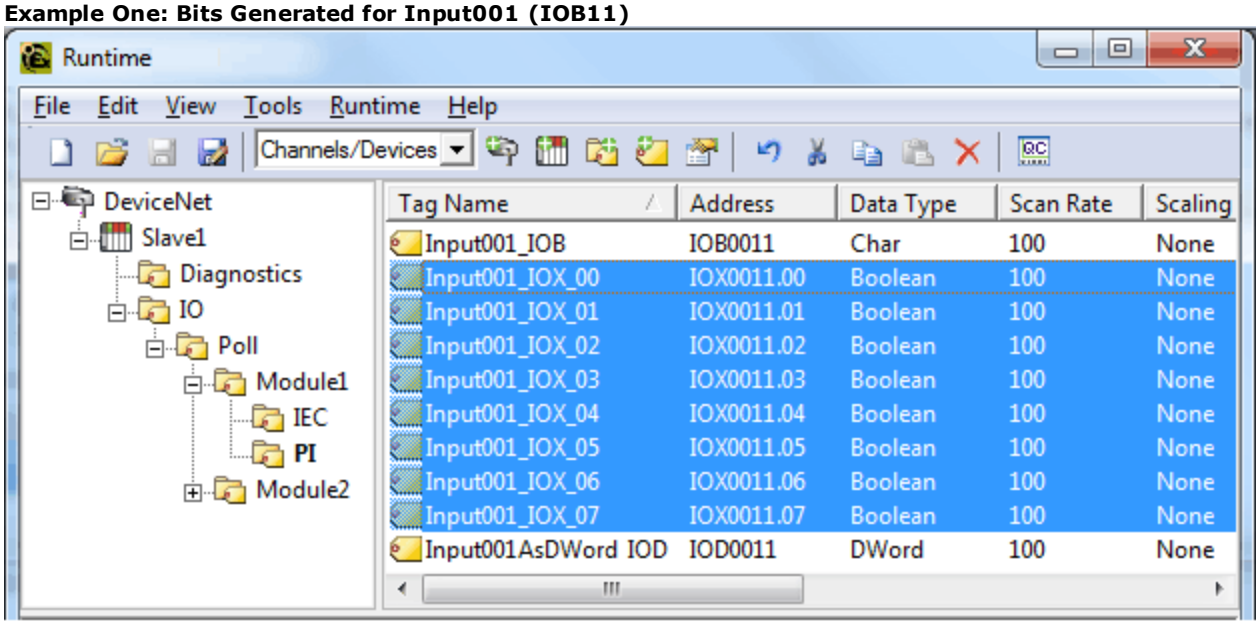
**Note:** For more information, refer to **Address Descriptions**.

## Generate Bit References

A module's Byte, Word and DWord I/O data can have individual bits referenced. This option will automatically generate bit references at the same offsets as the Byte, Word and DWord reference. These tags will have the mnemonic IOX/OOX for Process Image Offset and IX/QX for IEC Offsets.

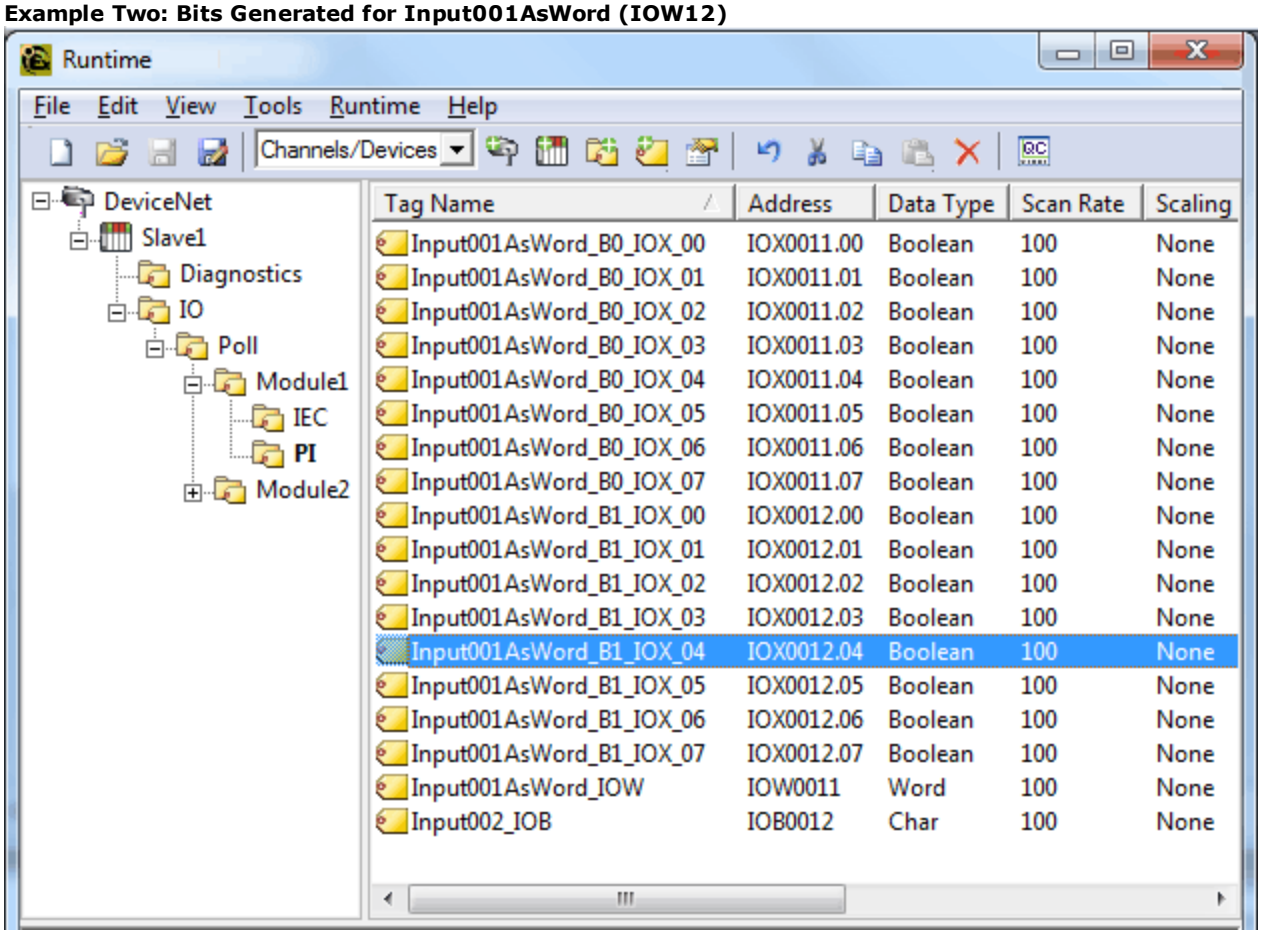
The examples below display how Bit tags are generated for Byte, Word and DWord I/O data. Input001, Input001AsWord and Input001AsDWord are symbolic names defined in the SyCon database. Module1 is a Byte Module.

**Example One: Bits Generated for Input001 (IOB11)**



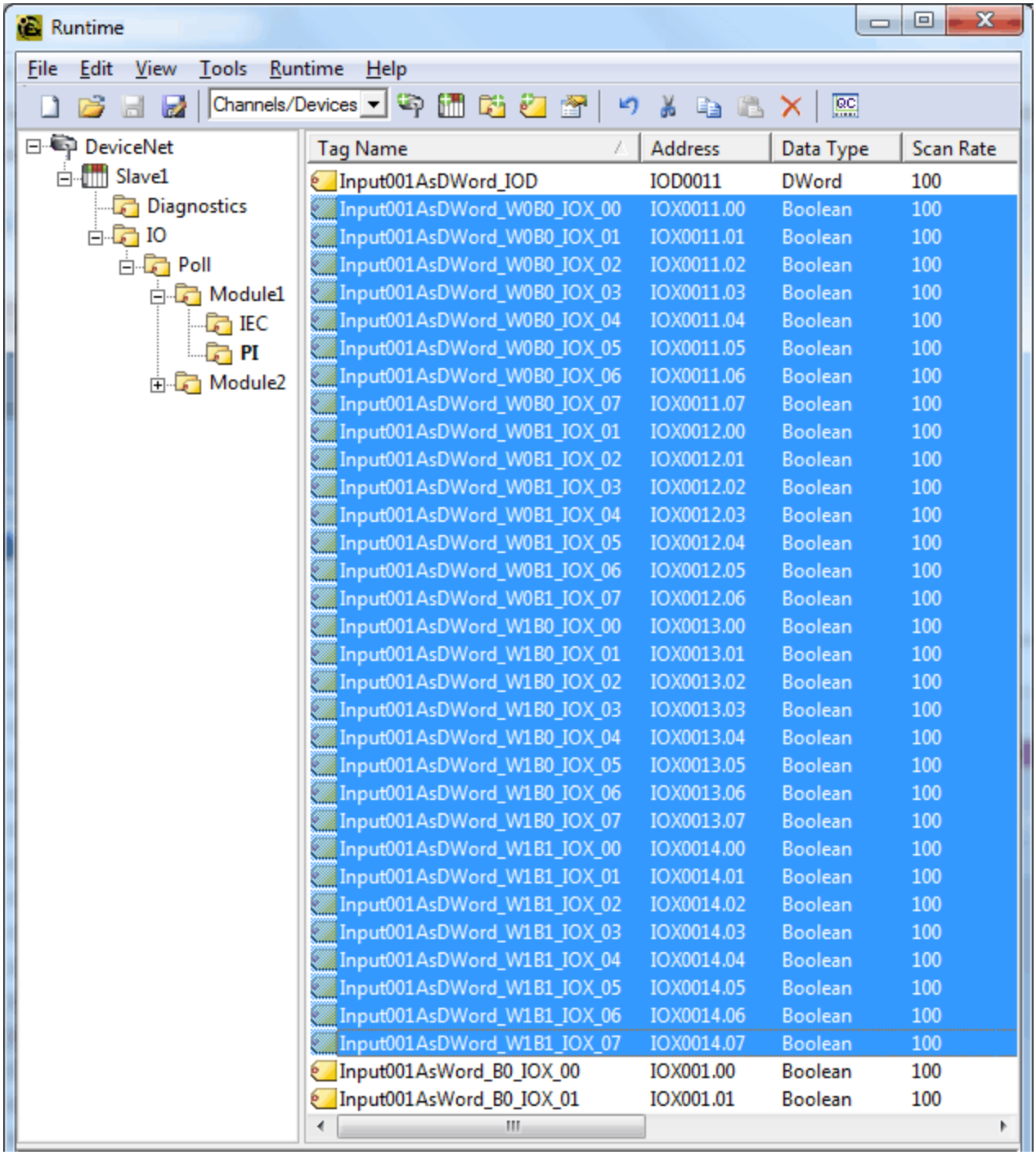

**Note:** Bits 0-7 of Byte 11 are referenced individually in IOX tags.

**Example Two: Bits Generated for Input001AsWord (IOW12)**




**Note:** Bits 0-15 of Word 11 are referenced individually in IOX tags. Since 'Module 1' is a Byte Module, Word 12 must be broken up into its individual Bytes (which are further referenced as Bits 0-7). If 'Module 1' were a Word Module, Bits 0-15 could be referenced.

**Example Three: Bits Generated for Input001AsDWord (IOD11)**




**Note:** Bits 0-31 of DWord 11 are referenced individually in IOX tags. Because 'Module 1' is a Byte Module, DWord 12 must be broken up into its individual Bytes which are further referenced as Bits 0-7. If 'Module 1' were a DWord Module, Bits 0-31 could be referenced.

## 8 Bit Data Expansion

Byte references are automatically generated for 8 Bit I/O Data when the "Import AND Expand SyCon I/O Tags" option is selected under Database Options. Additionally, 8 Bit I/O Data can be referenced as 16 bit and 32 bit entities. These options will automatically generate Word and/or DWord references at the same offsets as the Byte reference.

### Word Reference

These tags will have the mnemonic IOW/OOW for Process Image Offset and IW/QW for IEC Offsets. In order to access the Byte I/O data as a Word, these tags must be generated.

### DWord Reference

These tags will have the mnemonic IOD/OOD for Process Image Offset and ID/QD for IEC Offsets. In order to access the Byte I/O data as a DWord, these tags must be generated.

Word and DWord references are only generated if the size of module in question (including the offset into the module) is big enough to reference as a Word or DWord. The module must be at least 2 bytes in size to reference the module's base offset as a Word and 4 bytes in size to reference the module's base offset as a DWord. If a module is 2 bytes in size, no DWord references will be generated. Likewise, if the module is 1 byte in size, no Word references will be generated.

### Examples

**SyCon**

Byte Addressing is assumed. For more information on Byte and Word Addressing Modes, refer to **Address Descriptions**.

**Configured I/O Module:** Byte Array, IB, Length 6, Offset 0.

**OPC Server**

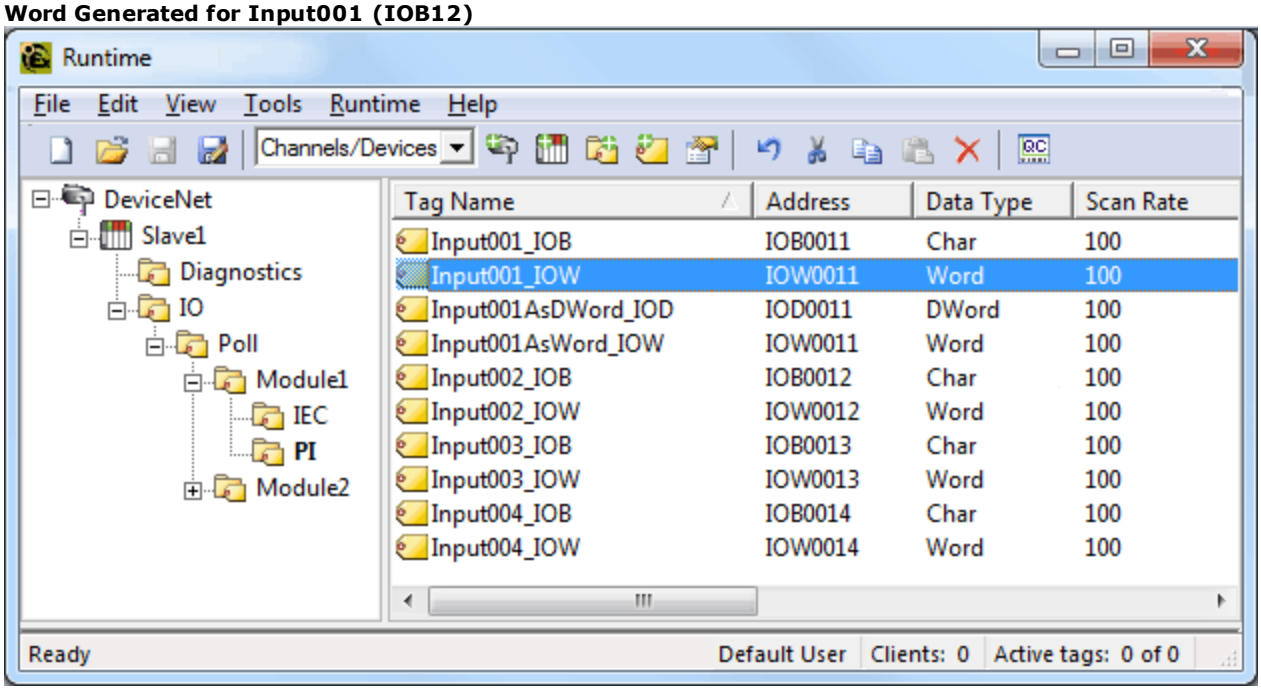
The following tags will be generated in the OPC server.

| Tag | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 |
|---|---|---|---|---|---|---|
| Byte Tags | IOB0 | IOB1 | IOB2 | IOB3 | IOB4 | IOB5 |
| Word Tags | IOW0 | IOW1 | IOW2 | IOW3 | IOW4 | * |
| DWord Tags | IOD0 | IOD1 | IOD2 | * | * | * |

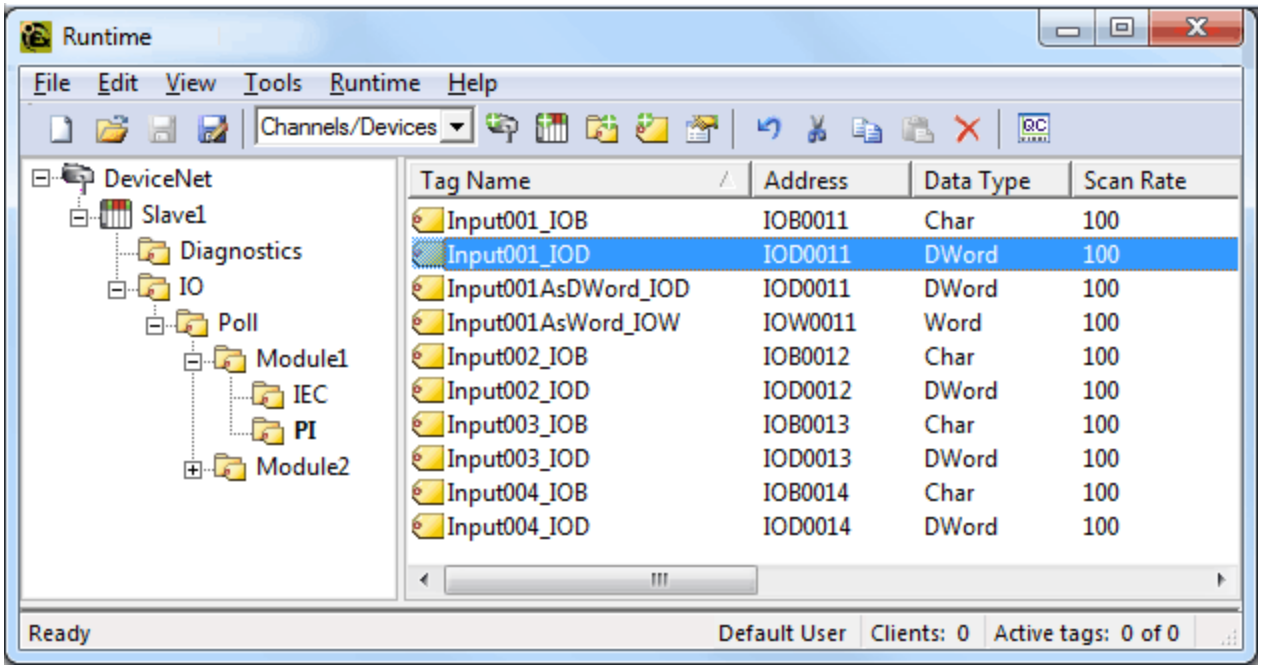
*No tag is generated because it would exceed the size of the module.

The images below illustrate how Word and DWord tags are generated for Byte data. Input001 is a symbolic name defined in the SyCon database. When expanded, a Byte reference (Input001_IOB) is automatically generated for Input001.

**Word Generated for Input001 (IOB12)**




**DWord Generated for Input001 (IOB12)**




## 16 bit SyCon Tag Data Expansion

Word references are automatically generated for 16 bit I/O Data if the "Import AND Expand SyCon I/O Tags" option is selected under Database Options. Additionally, 16 bit I/O Data can be referenced as 8 Bit and 32 bit entities. These options will automatically generate Byte and/or DWord references at the same offsets as the Word reference.

### Byte Reference

These tags will have the mnemonic IOB/OOB for Process Image Offset and IB/QB for IEC Offsets. In order to access the Bytes of Word I/O data, these tags must be generated.

### DWord Reference

These tags will have the mnemonic IOD/OOD for Process Image Offset and ID/QD for IEC Offsets. In order to access the Word I/O data as a DWord, these tags must be generated.

Byte and DWord references are only generated if the size of module in question (including the offset into the module) is big enough to reference as a DWord. The module must be at least 2 words in size to reference the module's base offset as a DWord. If a module is 1 word in size, no DWord references will be generated.

### Examples

**SyCon**

Byte addressing is assumed.

**Configured I/O Module:** Word Array, IW, Length 6, Offset 0.

**OPC Server**

The following tags will be generated in the OPC server.

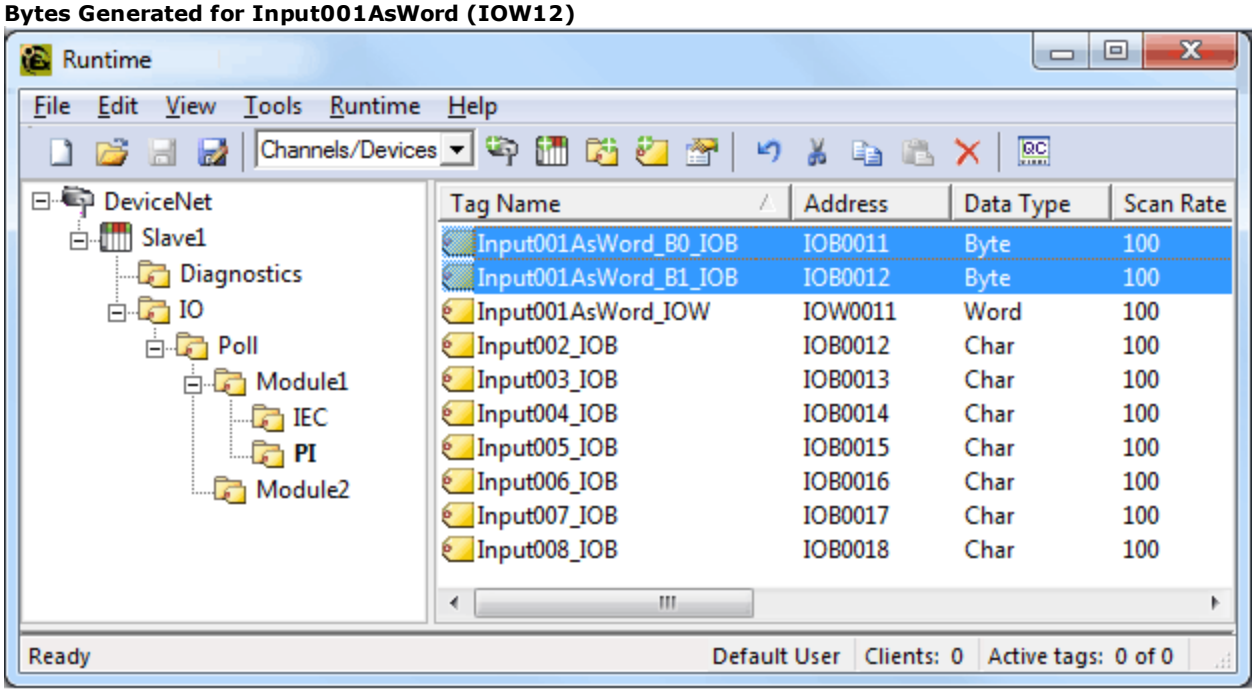
| Tags | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 |
|---|---|---|---|---|---|---|
| Byte Tags | IOB0 | IOB1 | IOB2 | IOB3 | IOB4 | IOB5 |
| Word Tags | IOW0 | N/A* | IOW2 | N/A* | IOW4 | N/A* |
| DWord Tags | IOD0 | N/A* | IOD2 | N/A* | ** | N/A* |

*No tag will be generated regardless of the size of the module. This follows the definition of a Word Module. For more information on Byte/Word/DWord Module Addressing, refer to **Address Descriptions**.

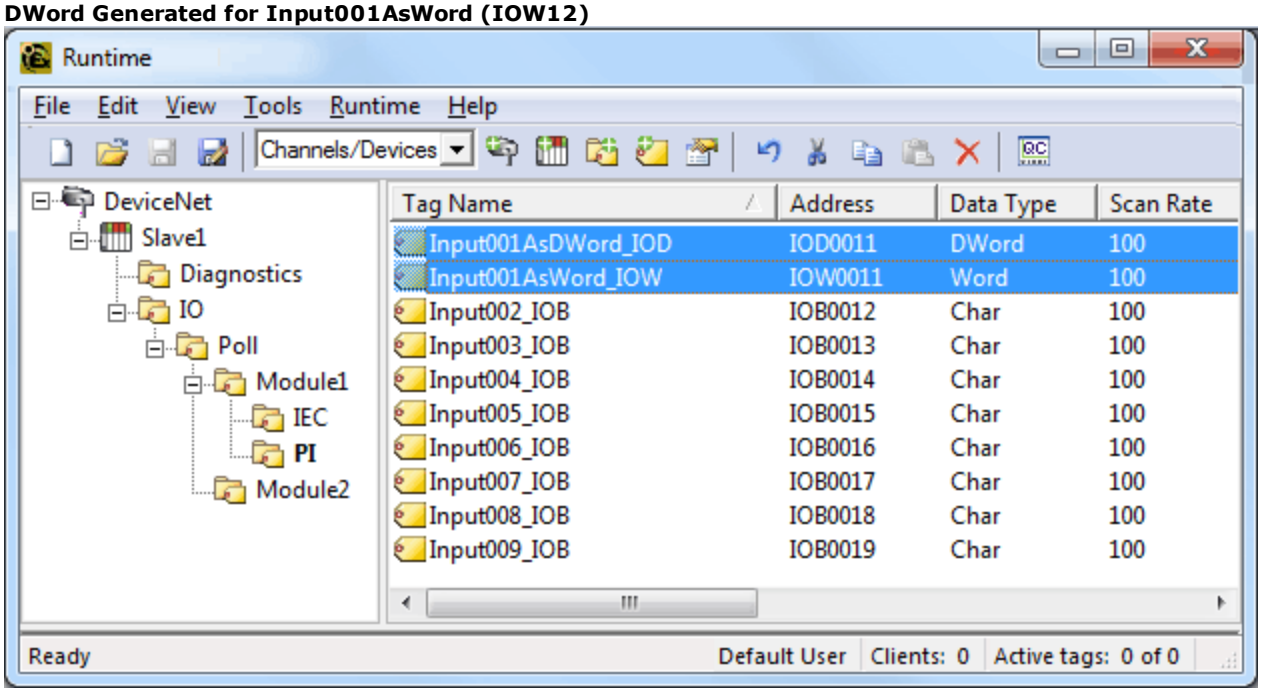
**No tag is generated because it would exceed the size of the module.

The images below illustrate how Byte and DWord tags are generated for Word data. Input001AsWord is a symbolic name defined in the SyCon database. When expanded, a Word reference (Input001AsWord_IOW) is automatically generated for Input001AsWord.

**Bytes Generated for Input001AsWord (IOW12)**




**DWord Generated for Input001AsWord (IOW12)**




## 32 bit SyCon Tag Data Expansion

DWord references are automatically generated for 32 bit I/O Data if the "Import AND Expand SyCon I/O Tags" option is selected under Database Options. Additionally, 32 bit I/O Data can be referenced as 8-Bit and 16 bit entities. These options will automatically generate Byte and/or Word references at the same offsets as the DWord reference.

### Byte Reference

These tags will have the mnemonic IOB/OOB for Process Image Offset and IB/QB for IEC Offsets. In order to access the Bytes of DWord Module data, these tags must be generated.

### Word Reference

These tags will have the mnemonic IOW/OOW for Process Image Offset and IW/QW for IEC Offsets. In order to access the Words of DWord Module data, these tags must be generated.

## Examples

### SyCon

Byte Addressing is assumed.

**Configured I/O Module:** DWORD Array, ID, Length 4, Offset 0

### OPC Server

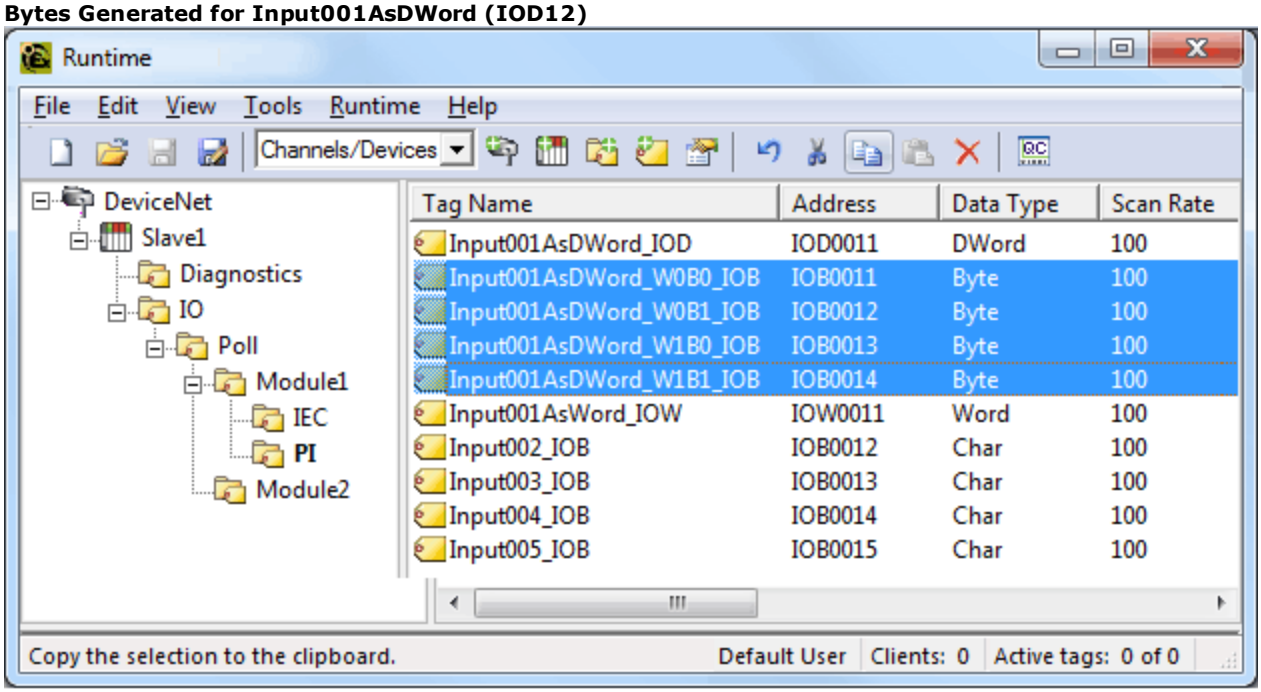
The following tags will be generated in the OPC server.

| Tags | Byte 0 | Byte 1 | Byte 2 | Byte 3 |
|---|---|---|---|---|
| Byte Tags | IOB0 | IOB1 | IOB2 | IOB3 |
| Word Tags | IOW0 | N/A* | IOW2 | N/A* |
| DWord Tags | IOD0 | N/A* | N/A* | N/A* |

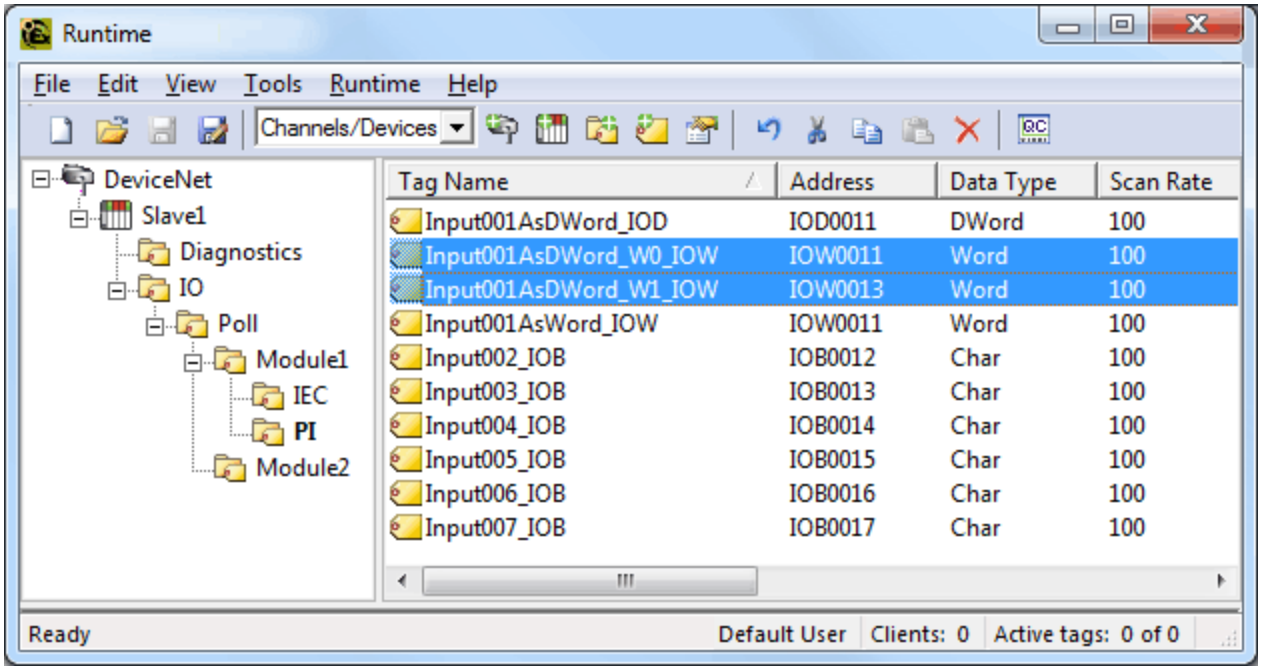
*No tag will be generated regardless of the size of the module. This follows the definition of a Word Module. For more information on Byte/Word/DWord Module Addressing, refer to **Address Descriptions**.

The examples below illustrate how Byte and Word tags are generated for DWord data. Input001AsDWord is a symbolic name defined in the SyCon database. When expanded, a Byte reference (Input001AsDWord_IOD) is automatically generated for Input001AsDWord.

**Bytes Generated for Input001AsDWord (IOD12)**




**Words Generated for Input001AsDWord (IOD12)**


Runtime
File Edit View Tools Runtime Help
Channels/Devices
DeviceNet
Slave1
Diagnostics
IO
Poll
Module1
IEC
PI
Module2
Tag Name
Address
Data Type
Scan Rate
Input001AsDWord_IOD IOD0011 DWord 100
Input001AsDWord_W0_IOW IOW0011 Word 100
Input001AsDWord_W1_IOW IOW0013 Word 100
Input001AsWord_IOW IOW0011 Word 100
Input002_IOB IOB0012 Char 100
Input003_IOB IOB0013 Char 100
Input004_IOB IOB0014 Char 100
Input005_IOB IOB0015 Char 100
Input006_IOB IOB0016 Char 100
Input007_IOB IOB0017 Char 100
Ready
Default User
Clients: 0
Active tags: 0 of 0

## Device Setup

The device represents a single device in the SyCon Configuration Database. It can be a Master or a Slave.

### 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 30 seconds.

### Request Timeout

This parameter specifies the time that the driver will wait for 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 100 to 30000 milliseconds.

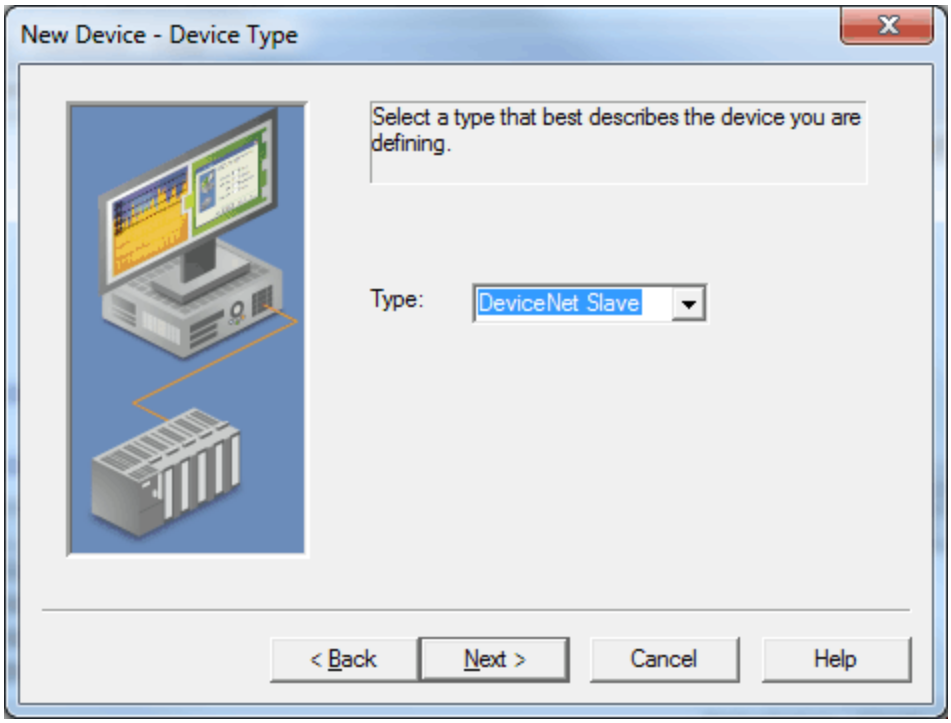
### Retry Attempts

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

### Device IDs

The Device ID represents the MAC ID in SyCon. Its range varies from bus to bus. The Device ID allows Automatic Tag Database Generation to import the proper tags for a given device.

## Device Type




Description of the parameter is as follows:

- **Type:** When applicable, this parameter defines the device's bus type and master/slave type. Supported device types include DeviceNet Master, DeviceNet Slave, Profibus-DP Master, and Profibus-DP Slave.

**Note:** Diagnostics may be accessed from the device specified as the Master.

## Data Types Description

| Data Type | Description |
|---|---|
| Boolean | Single bit |
| Byte | Unsigned 8 bit value |
| Char | Signed 8 bit value |
| Word | Unsigned 16 bit value<br><br>bit 0 is the low bit<br>bit 15 is the high bit |
| Short | Signed 16 bit value<br><br>bit 0 is the low bit<br>bit 14 is the high bit<br>bit 15 is the sign bit |
| BCD | Two byte packed BCD<br><br>Value range is 0-9999. Behavior is undefined for values beyond this range. |
| DWord | Unsigned 32 bit value<br><br>bit 0 is the low bit<br>bit 31 is the high bit |
| Long | Signed 32 bit value<br><br>bit 0 is the low bit<br>bit 30 is the high bit<br>bit 31 is the sign bit |
| LBCD | Four byte packed BCD<br><br>Value range is 0-99999999. Behavior is undefined for values beyond this range. |
| Float | 32 bit floating point value.<br><br>The driver interprets two consecutive 16 bit 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 |

## Address Descriptions

Select an addressing type from the list below for specific address descriptions.

**Process Image Address Descriptions**
**IEC Address Descriptions**

**Note:** Process Image refers to addresses with syntax IOx and OOx. IEC refers to addresses with syntax Ix and Qx.

## Process Image Address Descriptions

Address mnemonic and offsets are based on the physical offset into the Master's Process Image Memory Map (I/O Data). Addresses are always byte-based, regardless of the addressing mode selected in SyCon's Master Settings. The default data types for dynamically defined tags are shown in **bold**.

**Note:** The address ranges listed below are based on an 8K Dual-Port Memory.

| Device Type | Range Data | Type | Access |
|---|---|---|---|
| Process Image Inputs | IOX0.b-IOX3583.b*<br>.b is Bit 0-7 Byte Module<br>.b is Bit 0-15 Word Module<br>.b is Bit 0-31 DWord Module | **Boolean** | Read Only |
| | IOB0-IOB3583 | **Byte**, Char, String** | Read Only |
| | IOW0-IOW3582 | **Word**, Short, BCD | Read Only |
| | IOD0-IOD3580 | **DWord**, Long, LBCD, Float | Read Only |
| Process Image Outputs | OOX0.b-OOX3583.b*<br>.b is Bit 0-7 Byte Module<br>.b is Bit 0-15 Word Module<br>.b is Bit 0-31 DWord Module | **Boolean** | Write Only |
| | OOB0-OOB3583 | **Byte**, Char, String** | Write Only |
| | OOW0-OOW3582 | **Word**, Short, BCD | Write Only |
| | OOD0-OOD3580 | **DWord**, Long, LBCD, Float | Write Only |

*These memory types/subtypes do not support arrays.

**Byte memory types (IOB) support Strings. The syntax for strings is *<address>.<length>* where 0 < length <= 246.

**Note 1:** All offsets for memory types IO and OO represent a byte starting location within the specified memory type.

**Note 2:** Use caution when modifying Word, Short, DWord and Long types. For I and Q memory types, addresses may overlap depending on the module format and addressing type. It is recommended that these memory types be used so that overlapping does not occur.

**Note 3:** For information on the proper referencing of Process Image data, refer to **Module Format vs. Byte/Word Addressing**.

### Arrays

All memory types support arrays, excepting those marked with an asterisk (*). The syntax below is valid for declaring an array. If no rows are specified, row count of 1 is assumed.

*<address>[rows][cols]*

For Word, Short and BCD arrays, the base address + (rows *cols *2) cannot exceed 247. The array's elements are words and are located on a word boundary. For example, IOW0[4] will return IOW0, IOW2, IOW4, and IOW6 for both Byte and Word Addressing.

For Float, DWord, Long and Long BCD arrays, the base address + (rows *cols *4) cannot exceed 247. The array's elements are DWords and are located on a DWord boundary. For example, IOD0[4] will return IOD0, IOD4, IOD8, IOD12 for both Byte and Word Addressing.

For all arrays, the total number of bytes being requested cannot exceed the internal block size of 247 bytes.

### Byte Swapping

Bytes can be swapped for 16 bit (Word, Short and BCD) and 32 bit (DWord, Long, Float and LBCD) data by appending an 'S' to the end of an address reference:

*<address>S*

For arrays (each element will be Byte swapped):

*<address>S [rows][cols]*

Below are examples to illustrate how Bytes are swapped for 16 and 32 bit data for both Little and Big Endian. Byte-ordering is unaffected by the Addressing Mode.

**16 Bit Data**

In the example below, an analog sensor maps to Offset 0. Sensor value=0x1234 (hex) == 4660 (dec).

**Little Endian (LSB-MSB) - No Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x34 | IOB0=0x34 | IOW0=0x1234 | IOD0=0x00001234 |
| 1 | 0x12 | IOB1=0x12 | | |
| 2 | 0x00 | IOB2=0x00 | | |
| 3 | 0x00 | IOB3=0x00 | | |

**Little Endian (LSB-MSB) - Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x34 | IOB0=0x34 | IOW0=0x3412 | IOD0=0x34120000 |
| 1 | 0x12 | IOB1=0x12 | | |
| 2 | 0x00 | IOB2=0x00 | | |
| 3 | 0x00 | IOB3=0x00 | | |

**Big Endian (MSB-LSB) - No Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x12 | IOB0=0x12 | IOW0=0x3412 | IOD0=0x00003412 |
| 1 | 0x34 | IOB1=0x34 | | |
| 2 | 0x00 | IOB2=0x00 | | |
| 3 | 0x00 | IOB3=0x00 | | |

**Big Endian (MSB-LSB) - Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x12 | IOB0=0x12 | IOW0=0x1234 | IOD0=0x12340000 |
| 1 | 0x34 | IOB1=0x34 | | |
| 2 | 0x00 | IOB2=0x00 | | |
| 3 | 0x00 | IOB3=0x00 | | |

**32 Bit Data**

In the example below, an analog sensor maps to Offset 0. Sensor value=0x12345678 (hex) == 305,419,896 (dec).

**Little Endian (LSB-MSB) - No Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x78 | IOB0=0x78 | IOW0=0x5678 | IOD0=0x12345678 |

| | | | | |
|---|---|---|---|---|
| 1 | 0x56 | IOB1=0x56 | | |
| 2 | 0x34 | IOB2=0x34 | IOW2=0x1234 | |
| 3 | 0x12 | IOB3=0x12 | | |

**Little Endian (LSB-MSB) - Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x78 | IOB0=0x78 | IOW0=0x7856 | IOD0=0x78563412 |
| 1 | 0x56 | IOB1=0x56 | | |
| 2 | 0x34 | IOB2=0x34 | IOW2=0x3412 | |
| 3 | 0x12 | IOB3=0x12 | | |

**Big Endian (MSB-LSB) - No Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x12 | IOB0=0x12 | IOW0=0x3412 | IOD0=0x78563412 |
| 1 | 0x34 | IOB1=0x34 | | |
| 2 | 0x56 | IOB2=0x56 | IOW2=0x7856 | |
| 3 | 0x78 | IOB3=0x78 | | |

**Big Endian (MSB-LSB) - Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x12 | IOB0=0x12 | IOW0=0x1234 | IOD0=0x12345678 |
| 1 | 0x34 | IOB1=0x34 | | |
| 2 | 0x56 | IOB2=0x56 | IOW2=0x5678 | |
| 3 | 0x78 | IOB3=0x78 | | |

**Note 1:** Caution must be exercised when referencing overlapped memory. For example, IW1S will corrupt IW0S (and vice-versa). Overlapping references is not recommended.

**Note 2:** SyCon allows for swapping via Symbolic Names under Long and Word Details

### Module Format vs. Byte/Word Addressing

Module Format refers to the width of the Module configured in SyCon. Supported formats include Byte, Word and DWord. Addressing Modes below refer to the Addressing Mode under Master Settings in SyCon. Byte 0 -Byte n refer to the Byte Offsets in the Process Image. Regions highlighted in Yellow exemplify the Bytes involved in a reference at Offset 0 for the given memory type (i.e., IOD0 will contain Byte 0-Byte 3). They also illustrates how Words and DWords can overlap. Exercise caution when referencing overlapped Words and DWords.

### Byte Module (8-Bit Module Data)

For Byte Modules, Byte memory is treated as a Byte. There are no N/A references.

### Byte Addressing

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|---|
| IOX0.0-7 | IOX1.0-7 | IOX2.0-7 | IOX3.0-7 | IOX4.0-7 |
| IOB0 | IOB1 | IOB2 | IOB3 | IOB4 |
| IOW0 | IOW1 | IOW2 | IOW3 | IOW4 |
| IOD0 | IOD1 | IOD2 | IOD3 | IOD4 |

**Table x: Byte Module Byte PI Addressing**

### Word Addressing

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|---|
| IOX0.0-7 | IOX1.0-7 | IOX2.0-7 | IOX3.0-7 | IOX4.0-7 |
| IOB0 | IOB1 | IOB2 | IOB3 | IOB4 |
| IOW0 | IOW1 | IOW2 | IOW3 | IOW4 |
| IOD0 | IOD1 | IOD2 | IOD3 | IOD4 |

**Table x: Byte Module Word PI Addressing**

### Word Module (16 bit Module Data)

For Word Modules, Byte memory is treated as a Word with the exception of IOB. N/A references are due to this Word alignment for the module.

### Byte Addressing

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|---|
| IOX0.0-15 | N/A | IOX2.0-15 | N/A | IOX4.0-15 |
| IOB0 | IOB1 | IOB2 | IOB3 | IOB4 |
| IOW0 | N/A | IOW2 | N/A | IOW4 |
| IOD0 | N/A | IOD2 | N/A | IOD4 |

**Table x: Word Module Byte PI Addressing**

### Word Addressing

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|---|
| IOX0.0-15 | N/A | IOX2.0-15 | N/A | IOX4.0-15 |
| IOB0 | IOB1 | IOB2 | IOB3 | IOB4 |
| IOW0 | N/A | IOW2 | N/A | IOW4 |
| IOD0 | N/A | IOD2 | N/A | IOD4 |

**Table x: Word Module Word PI Addressing**

### DWord Module (32 bit Module Data)

For DWord Modules, Byte memory is treated as a DWord with the exception of IOB and IOW. N/A references are due to this DWord alignment for the module.

### Byte Addressing

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|---|
| IOX0.0-31 | N/A | N/A | N/A | IOX4.0-31 |
| IOB0 | IOB1 | IOB2 | IOB3 | IOB4 |
| IOW0 | N/A | IOW2 | N/A | IOW4 |
| IOD0 | N/A | N/A | N/A | IOD4 |

**Table x: DWord Module Byte PI Addressing**

### Word Addressing

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|---|
| IOX0.0-31 | N/A | N/A | N/A | IOX4.0-31 |
| IOB0 | IOB1 | IOB2 | IOB3 | IOB4 |
| IOW0 | N/A | IOW2 | N/A | IOW4 |
| IOD0 | N/A | N/A | N/A | IOD4 |

**Table x: DWord Module Word PI Addressing**

## IEC Address Descriptions

Address mnemonic and offsets are based on standard Siemens addressing (IB, IW, ID). Addresses are byte-based or word-based, depending on the addressing mode selected in SyCon's Master Settings. The default data types for dynamically defined tags are shown in **bold**.

**Note:** The address ranges listed below are based on an 8K Dual-Port Memory.

| Device Type | Range Data | Type | Access |
|---|---|---|---|
| Process Image Inputs | IX0.b-IX3583.b*<br>.b is Bit 0-7 Byte Module<br>.b is Bit 0-15 Word Module<br>.b is Bit 0-31 DWord Module | **Boolean** | Read Only |
| | IB0-IB3583 | **Byte**, Char, String** | Read Only |
| | IW0-IW3582 | **Word**, Short, BCD | Read Only |
| | ID0-ID3580 | **DWord**, Long, LBCD, Float | Read Only |
| Process Image Outputs | QX0.b-QX3583.b*<br>.b is Bit 0-7 Byte Module | **Boolean** | Write Only |

| | | | |
|---|---|---|---|
| | .b is Bit 0-15 Word Module<br>.b is Bit 0-31 DWord Module | | |
| | QB0-QB3583 | **Byte**, Char, String** | Write Only |
| | QW0-QW3582 | **Word**, Short, BCD | Write Only |
| | QD0-QD3580 | **DWord**, Long, LBCD, Float | Write Only |

*These memory types/subtypes do not support arrays.

**Byte memory types (IB) support Strings. The syntax for strings is *<address>.<length>* where 0 < length <= 246.

**Note 1:** All offsets for memory types I and Q represent a byte starting location within the specified memory type.

**Note 2:** Use caution when modifying Word, Short, DWord and Long types. For I and Q memory types, addresses may overlap depending on the module format and addressing type. It is recommended that these memory types be used so that overlapping does not occur.

**Note 3:** For information on the proper referencing of Process Image data, refer to **Module Format vs. Byte/Word Addressing**.

### Arrays

All memory types support arrays, excepting those marked with an asterisk (*). The syntax below is valid for declaring an array. If no rows are specified, row count of 1 is assumed.

*<address>[rows][cols]*

For Word, Short and BCD arrays, the base address + (rows *cols *2) cannot exceed 247. The array's elements are words and are located on word boundaries. For example, IW0[4] would return IW0, IW2, IW4, and IW6 assuming Byte Addressing, IW0, IW1, IW2, IW3 (assuming Word Addressing).

For Float, DWord, Long and Long BCD arrays, the base address + (rows *cols *4) cannot exceed 247. The array's elements are DWords and are located on DWord boundaries. For example, ID0[4] will return ID0, ID4, ID8, ID12 (assuming Byte Addressing) and ID0, ID2, ID4, ID6 (assuming Word Addressing).

For all arrays, the total number of bytes being requested cannot exceed the internal block size of 247 bytes.

### Byte Swapping

Bytes can be swapped for 16 bit (Word, Short and BCD) and 32 bit (DWord, Long, Float and LBCD) data by appending an 'S' to the end of an address reference:

*<address>S*

For arrays (each element will be Byte swapped):

*<address>S [rows][cols]*

The examples below illustrate how Bytes are swapped for 16 and 32 bit data for both Little and Big Endian. Byte-ordering is unaffected by the addressing ,ode.

#### 16 bit Data

In the example below, an analog sensor maps to Offset 0. Sensor value=0x1234 (hex) == 4660 (dec).

**Little Endian (LSB-MSB) - No Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x34 | IB0=0x34 | IW0=0x1234 | ID0=0x00001234 |
| 1 | 0x12 | IB1=0x12 | | |
| 2 | 0x00 | IB2=0x00 | | |
| 3 | 0x00 | IB3=0x00 | | |

**Little Endian (LSB-MSB) - Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x34 | IB0=0x34 | IW0=0x3412 | ID0=0x34120000 |
| 1 | 0x12 | IB1=0x12 | | |
| 2 | 0x00 | IB2=0x00 | | |
| 3 | 0x00 | IB3=0x00 | | |

**Big Endian (MSB-LSB) - No Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x12 | IB0=0x12 | IW0=0x3412 | ID0=0x00003412 |
| 1 | 0x34 | IB1=0x34 | | |
| 2 | 0x00 | IB2=0x00 | | |
| 3 | 0x00 | IB3=0x00 | | |

**Big Endian (MSB-LSB) - Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x12 | IB0=0x12 | IW0=0x1234 | ID0=0x12340000 |
| 1 | 0x34 | IB1=0x34 | | |
| 2 | 0x00 | IB2=0x00 | | |
| 3 | 0x00 | IB3=0x00 | | |

**32 Bit Data**
In the example below, an analog sensor maps to Offset 0. Sensor value=0x12345678 (hex) == 305,419,896 (dec).

**Little Endian (LSB-MSB) - No Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x78 | IB0=0x78 | IW0=0x5678 | ID0=0x12345678 |
| 1 | 0x56 | IB1=0x56 | | |
| 2 | 0x34 | IB2=0x34 | IW1=0x1234 | |
| 3 | 0x12 | IB3=0x12 | | |

**Little Endian (LSB-MSB) - Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x78 | IB0=0x78 | IW0=0x7856 | ID0=0x78563412 |
| 1 | 0x56 | IB1=0x56 | | |
| 2 | 0x34 | IB2=0x34 | IW1=0x3412 | |
| 3 | 0x12 | IB3=0x12 | | |

**Big Endian (MSB-LSB) - No Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x12 | IB0=0x12 | IW0=0x3412 | ID0=0x78563412 |
| 1 | 0x34 | IB1=0x34 | | |
| 2 | 0x56 | IB2=0x56 | IW1=0x7856 | |
| 3 | 0x78 | IB3=0x78 | | |

**Big Endian (MSB-LSB) - Swap**

| DPM Byte Offset | Data | Byte Reference | Word Reference | DWord Reference |
|---|---|---|---|---|
| 0 | 0x12 | IB0=0x12 | IW0=0x1234 | ID0=0x12345678 |
| 1 | 0x34 | IB1=0x34 | | |
| 2 | 0x56 | IB2=0x56 | IW1=0x5678 | |
| 3 | 0x78 | IB3=0x78 | | |

**Note 1:** Caution must be exercised when referencing overlapped memory. For example, IW1S will corrupt IW0S (and vice-versa). Overlapping references is not recommended.

**Note 2:** SyCon allows for swapping via symbolic names under Long and Word Details

### Module Format vs. Byte/Word Addressing

Module Format refers to the width of the Module configured in SyCon. Supported formats include Byte, Word and DWord. The addressing modes below refer to SyCon's Addressing Mode settings. Byte 0-Byte n refer to the Byte Offsets in the Process Image. Regions highlighted in Yellow exemplify the Bytes involved in a reference at Offset 0 for the given memory type (ie ID0 will contain Byte 0-Byte 3). They also illustrates how Words and DWords can overlap. Exercise caution when referencing overlapped Words and DWords.

### Byte Module (8-Bit Module Data)

For Byte Modules, Byte memory is treated as a Byte. There are no N/A references.

#### Byte Addressing

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|---|
| IX0.0-7 | IX1.0-7 | IX2.0-7 | IX3.0-7 | IX4.0-7 |
| IB0 | IB1 | IB2 | IB3 | IB4 |
| IW0 | IW1 | IW2 | IW3 | IW4 |
| ID0 | ID1 | ID2 | ID3 | ID4 |

**Table x: Byte Module Byte IEC Addressing**

#### Word Addressing

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|---|
| IX0.0-7 | N/A | IX1.0-7 | N/A | IX2.0-7 |
| IB0 | N/A | IB1 | N/A | IB2 |
| IW0 | N/A | IW1 | N/A | IW2 |
| ID0 | N/A | ID1 | N/A | ID2 |

**Table x: Byte Module Word IEC Addressing**

### Word Module (16 bit Module Data)

For Word Modules, Byte memory is treated as a Word (with the exception of IOB). N/A references are due to the Word alignment for the module.

#### Byte Addressing

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|---|
| IX0.0-15 | N/A | IX2.0-15 | N/A | IX4.0-15 |
| IB0 | IB1 | IB2 | IB3 | IB4 |
| IW0 | N/A | IW2 | N/A | IW4 |
| ID0 | N/A | ID2 | N/A | ID4 |

**Table x: Word Module Byte IEC Addressing**

#### Word Addressing

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|---|
| IX0.0-15 | N/A | IX1.0-15 | N/A | IX2.0-15 |
| IB0 | N/A | IB1 | N/A | IB2 |
| IW0 | N/A | IW1 | N/A | IW2 |
| ID0 | N/A | ID1 | N/A | ID2 |

**Table x: Word Module Word IEC Addressing**

### DWord Module (32 bit Module Data)

For DWord Modules, Byte memory is treated as a DWord (with the exception of IOB and IOW). N/A references are due to the DWord alignment for the module.

#### Byte Addressing

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|---|
| IX0.0-31 | N/A | N/A | N/A | IX4.0-31 |
| IB0 | IB1 | IB2 | IB3 | IB4 |

| | | | | |
|---|---|---|---|---|
| IW0 | N/A | IW2 | N/A | IW4 |
| ID0 | N/A | N/A | N/A | ID4 |

**Table x: DWord Module Byte IEC Addressing**

### Word Addressing

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|---|
| IX0.0-31 | N/A | N/A | N/A | IX2.0-31 |
| IB0 | N/A | IB1 | N/A | IB2 |
| IW0 | N/A | IW1 | N/A | IW2 |
| ID0 | N/A | N/A | N/A | ID2 |

**Table x: DWord Module Word IEC Addressing**

## Automatic Tag Database Generation

Automatic Tag Database Generation is a feature of the Hilscher Universal Driver that provides in order to import the SyCon Configuration Database into the OPC server. The specification of the SyCon Configuration file is made at the channel level; meaning, all devices under a channel will be based on the same SyCon Configuration file.

Module definitions, module settings, configured I/O and SyCon symbolic names are imported in tag database generation. In addition, diagnostics for both Master and Slaves are generated. This import capability leaves all the configuration to be done in SyCon with little configuration necessary in the OPC server.

**See Also: Database Options**

### How to Perform Automatic Tag Generation

To begin, ensure that devices have been defined under a channel. Remember that the Device ID is the MAC ID in SyCon. To perform automatic tag generation, navigate to **Channel Properties** | **SyCon Database**. Tags (I/O and Diagnostics) will be generated for each device under the given channel. Wait until the process is complete before editing Device Properties, Channel Properties and the SyCon Configuration Database.

### Information Imported From Database

**Master**
Addressing Mode (Byte vs Word-Based Addressing)

**Slave**
Module Format (Byte, Word, or DWord Module)
Byte Swapping (Symbolic Name Swap Option)

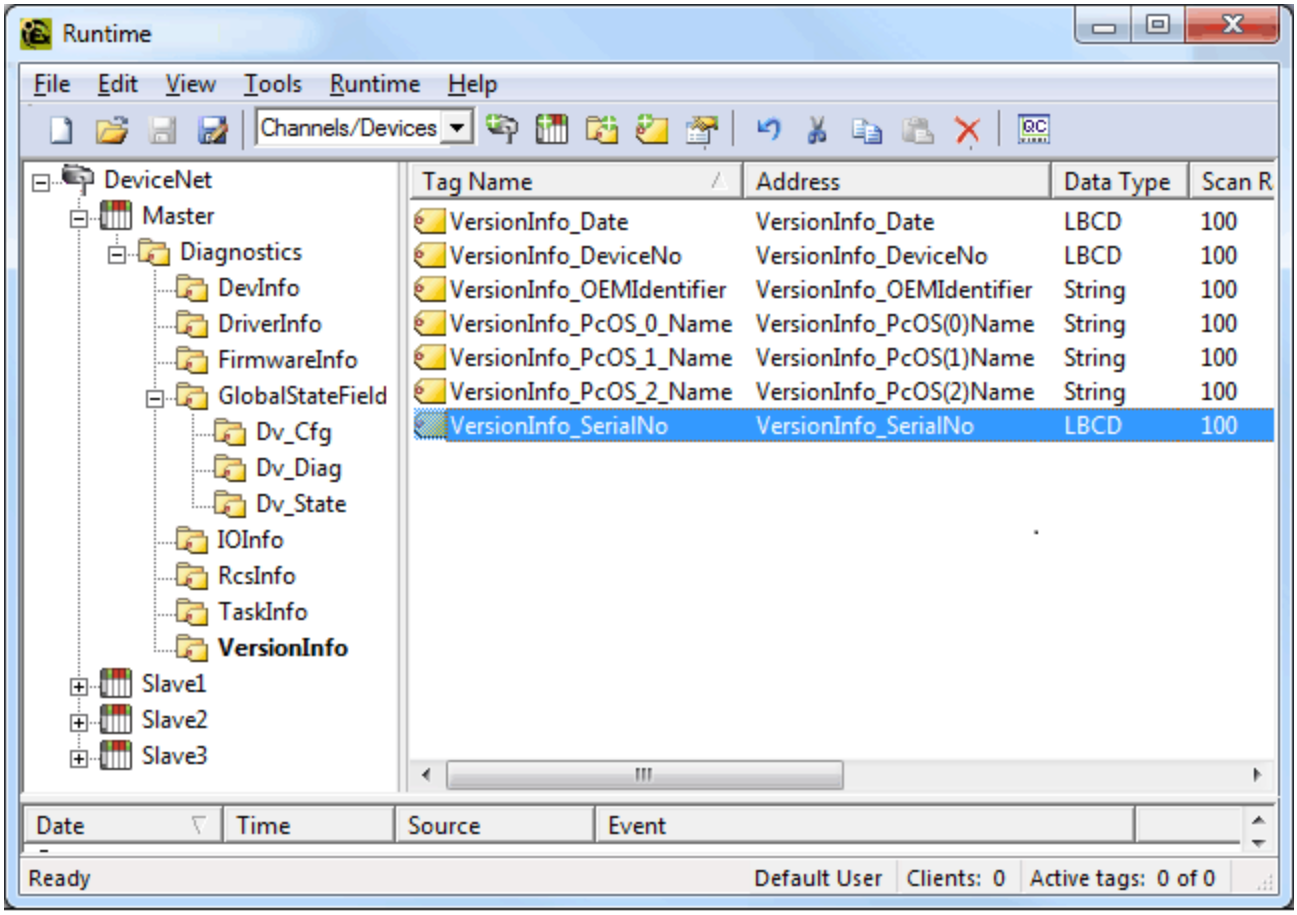
### Information Not Imported From Database

Device Names
Message Definitions (Explicit Messages, DPV1, etc)
String-type Symbolic Names

### Tags Generated In Server

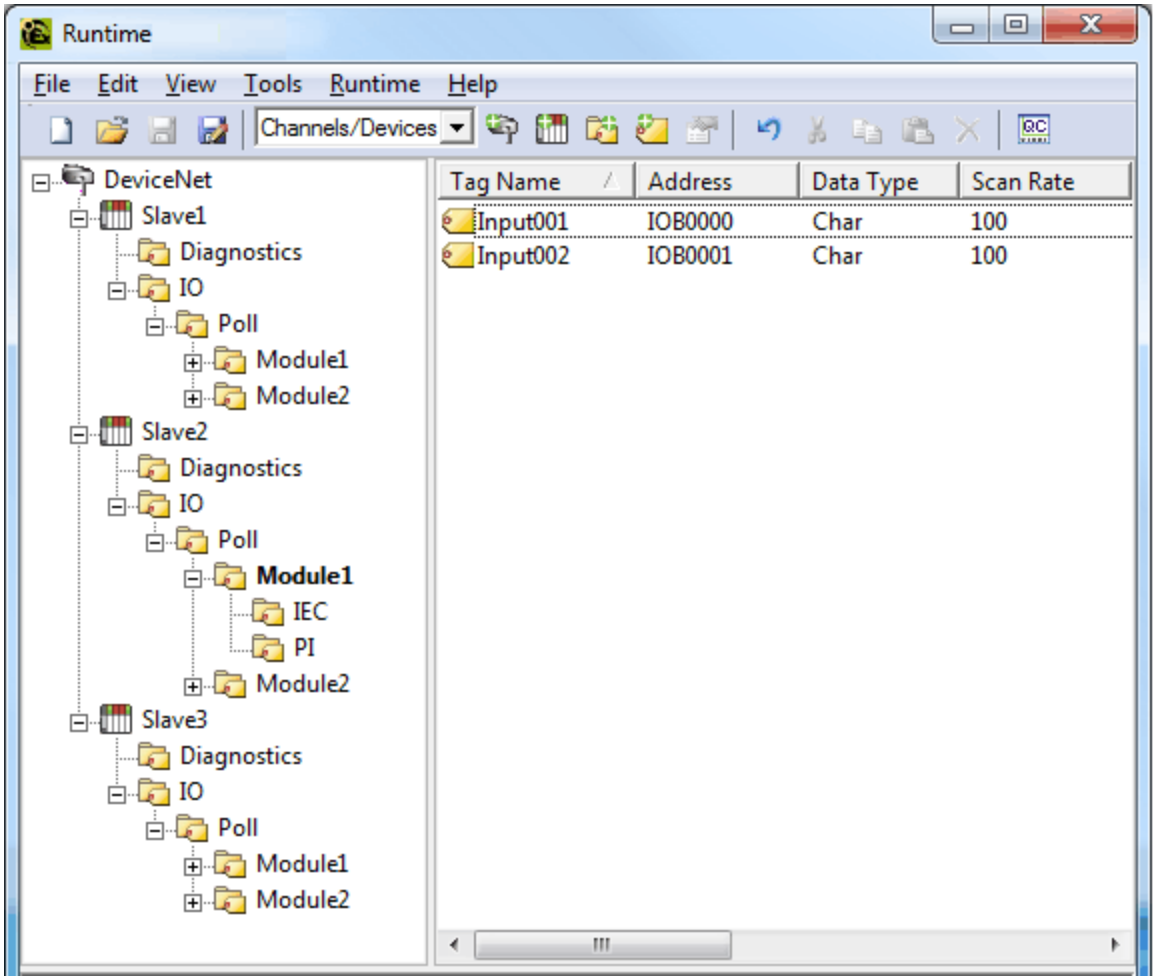
**Master**
Diagnostic Tags

The image below is an example of the diagnostic tags available for a DeviceNet Master. The exact diagnostic tags generated depends on the bus as specified under **Channel Properties** | **Board Type**.




**Slave**
I/O Data Tags (SyCon Symbolic Names)
Diagnostic Tags (if applicable)

SyCon I/O tags defined in the SyCon Configuration Database will be imported and a server tag generated for each. Also, if **I/O Tag Expansion** is selected, additional tags will be generated based on both the SyCon I/O tags and the expansion settings selected. The image below displays the tags and tag groups generated for three DeviceNet devices with I/O Tag Expansion set ON.




**Note:** For more information on SyCon I/O tag expansion, refer to **I/O Data References** and **Expansion Tag Addressing Type**.

## 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
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>'

### Driver Error Messages

Unable to load '<dll>'
Unable to import from '<dll>'
DevOpenDriver () failed with error code '<code>'
Memory allocation error

### Device Status Messages

Device '<device name>' is not responding
Unable to read device info data in area '<area>'. Board '<board>' returned Error Code '<code>'
Unable to read '<block size>' device info bytes in area '<area>'. Board '<board>' returned Error Code '<code>'
Unable to read task state data in task '<task num>'. Board '<board>' returned Error Code '<code>'
Unable to read '<block size>' task state bytes in task '<task num>'. Board '<board>' returned Error Code '<code>'
Unable to read tag '<address>' from device '<device>'. Board '<board>' returned Error Code '<code>'
Unable to read '<block size>' bytes starting at '<address>' from device '<device>'. Board '<board>' returned Error Code '<code>'
Unable to write to tag '<address>' from device '<device>'. Board '<board>' returned Error Code '<code>'
Unable to read tag '<name>': msg.b <command>, msg.device_adr <Device ID>
Unable to read '<block size>' message bytes: msg.b <command>, msg.device_adr <Device ID>...
Unable to write to tag '<address>': msg.b <command>, msg.device_adr <Device ID>...
Unable to read tag '<address>' from device '<device>'. Board '<board>' returned DPM Diagnostics [Global Bits='<Global Bits>', Node='<Remote Address>', Code='<Error Event>']
Unable to read '<block size>' bytes starting at '<address>' from device '<device>'. Board '<board>' returned DPM Diagnostics [Global Bits='<Global Bits>', Node='<Remote Address>', Code='<Error Event>']
Unable to write to tag '<address>' from device '<device>'. Board '<board>' returned DPM Diagnostics [Global Bits='<Global Bits>', Node='<Remote Address>', Code='<Error Event>']
Unable to read tag '<address>' from device '<device>'. Board '<board>' returned DNM Diagnostics [Global Bits='<Global Bits>', Node='<Device Address>', Code='<Error Event>']
Unable to read '<block size>' bytes starting at '<address>' from device '<device>'. Board '<board>' returned DNM Diagnostics [Global Bits='<Global Bits>', Node='<Device Address>', Code='<Error Event>']
Unable to write to tag '<address>' from device '<device>'. Board '<board>' returned DNM Diagnostics [Global Bits='<Global Bits>', Node='<Device Address>', Code='<Error Event>']

### Automatic Tag Database Generation Messages

The file is not a valid Sycon database or may be corrupt
Auto tag database generation cannot be performed while the driver is processing tags
Board Type for Board '<board number>' does not match the actual board installed. Verify Board Type and/or Board Selection
Board Type for Board '<board number> does not match the Slave Type for one or more Slaves configured. Delete or edit Slaves accordingly
'dbm32.dll' is not loaded and is required for auto tag generation. Verify SyCon is installed

## Error Codes

### CIF Device Driver Errors

| Error Code | Source | Description |
|---|---|---|
| -1 | CIF Driver | The communication board is not initialized by the driver.<br><br>-Check the driver configuration.<br>-Driver function used without calling DevOpenDriver () first. |
| -2 | CIF Driver | Error in internal 'Init State'. |
| -3 | CIF Driver | Error in internal 'Read State'. |
| -4 | CIF Driver | Command on this channel is active. |
| -5 | CIF Driver | Unknown parameter in function occurred. |
| -6 | CIF Driver | Version is incompatible. The device driver version does not correspond to the driver DLL version. From version V1.200 the internal command structure between DLL and driver has changed. Make sure to use the same version of the device driver and the driver DLL. |
| -10 | Device | Dual port memory RAM is not accessible/no hardware found. This error occurs when the driver is not able to read or write to the Dual port memory.<br><br>-Check the BIOS setting of the PC.<br><br>Memory address conflict with other PC components, try another memory address.<br><br>-Check the driver configuration for this board<br>-Check the jumper settings of the board. |
| -11 | Device | Not ready (RDY flag=Ready flag failed). Board is not ready. This could be a hardware malfunction or another program writes inadmissible to the dual port memory. |
| -12 | Device | Not running (RUN flag=Running flag failed). The board is ready but not all tasks are running because of an initialization error.<br><br>-No database is loaded into the device or an invalid parameter has been set so that a task cannot initialize. |
| -13 | Device | Watchdog test failed. |
| -14 | Device | Signals wrong Operating System version. No license code found on the communication board.<br><br>-Device has no license for the used operating system or customer software.<br>-No firmware or no database on the device is loaded. |
| -15 | Device | Error in dual port memory flags. |
| -16 | Device | Send mailbox is full. |
| -17 | Device | Function PutMessage timeout.<br><br>-If using an interrupt, check the interrupt on the device and in driver setup. These settings have to be the same. Is an interrupt on the board set? Is the right interrupt set? The interrupt could already be used by another PC component.<br>-If using polling mode, make sure that no interrupt is set on the board and that polling is set in the driver setup. The settings have to be the same.<br>-Device internal segment buffer full.<br>DevSetHostState not called. |
| -18 | Device | Function GetMessage timeout.<br><br>-If using an interrupt, check the interrupt on the device and in driver setup. These settings have to be the same. Is an interrupt on the board set? Is the right interrupt set? The interrupt could already be used by another PC component.<br>-If using polling mode, make sure that no interrupt is set on the board and that polling is set in the driver setup. The settings have to be the same. |
| -19 | Device | No message available. |
| -20 | Device | Reset command timeout.<br><br>-The board is ready but not all tasks are running because of an initialization error.<br>-No database is loaded into the device.<br>-If using an interrupt, check the interrupt on the device and in driver setup. These settings have to be the same. Is an interrupt on the board set? Is the right |

| | | |
|---|---|---|
| | | interrupt set? The interrupt could already be used by another PC component.<br>-If using polling mode, make sure that no interrupt is set on the board and that polling is set in the driver setup. The settings have to be the same. |
| -21 | Device | COM flag not set. The device cannot reach communication state.<br><br>-Device not connected to the fieldbus.<br>-No station found on the fieldbus.<br>-Wrong configuration on the device. |
| -22 | Device | I/O data exchange failed. |
| -23 | Device | I/O data exchange timeout.<br><br>-If using an interrupt, check the interrupt on the device and in driver setup. These settings have to be the same. Is an interrupt on the board set? Is the right interrupt set? The interrupt could already be used by another PC component.<br>-If using polling mode, then make sure that no interrupt is set on the board and that polling is set in the driver setup. The settings have to be the same. |
| -24 | Device | I/O data mode unknown. |
| -25 | Device | Function call failed. |
| -26 | Device | Dual port memory size differs from configuration. |
| -27 | Device | State mode unknown. |
| -30 | User | Driver not opened (device driver not loaded).<br><br>-Device driver not installed.<br>-Wrong parameters in the driver configuration. |
| -31 | User | Can't connect with device board. |
| -32 | User | Board not initialized.<br><br>-DevInitBoard () not called. |
| -33 | User | IOCTRL function failed.<br><br>-Make sure to use a device driver and DLL with the same version. |
| -34 | User | Parameter DeviceNumber invalid. |
| -35 | User | Parameter InfoArea unknown. |
| -36 | User | Parameter Number invalid. |
| -37 | User | Parameter Mode invalid. |
| -38 | User | NULL pointer assignment. |
| -39 | User | Message buffer too short. |
| -40 | User | Size parameter invalid. |
| -42 | User | Size parameter with zero length. |
| -43 | User | Size parameter too long. |
| -44 | User | Device address null pointer. |
| -45 | User | Pointer to buffer is a null pointer. |
| -46 | User | SendSize parameter too long. |
| -47 | User | ReceiveSize parameter too long. |
| -48 | User | Pointer to send buffer is a null pointer. |
| -49 | User | Pointer to receive buffer is a null pointer. |

## 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**
**Data Type '<type>' is not valid for device address '<address>'**
**Device address '<address>' is Read Only**
**Array size is out of range for address '<address>'**
**Array support is not available for the specified address: '<address>'**

### Missing address

**Error Type:**
Warning

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

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

### Device address '<address>' contains a syntax error

**Error Type:**
Warning

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

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

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

**Error Type:**
Warning

**Possible Cause:**
A tag address that has been specified statically 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.

### 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.

## Driver Error Messages

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

**Driver Error Messages**
**Unable to load '<dll>'**
**Unable to import from '<dll>'**
**DevOpenDriver () failed with error code '<code>'**
**Memory allocation error**

## Unable to load '<dll>'

**Error Type:**
Serious

**Possible Cause:**
A software component necessary to communicate with the Hilscher card or SyCon configuration database cannot be loaded.

**Solution:**
Verify that the latest version of SyCon is installed on the same machine as the OPC server and then try again.

## Unable to import from '<dll>'

**Error Type:**
Serious

**Possible Cause:**
The interface necessary to communicate with the Hilscher card or SyCon configuration database, cannot be loaded from <dll>.

**Solution:**
Verify that the latest version of SyCon is installed on the same machine as the OPC server and try again.

## DevOpenDriver () failed with error code '<code>'

**Error Type:**
Serious

**Possible Cause:**
Unable to load the device drivers necessary for card communications.

**Solution:**
Refer to the Error Codes table for specific information.

**See Also:**
**Error Codes**

## Memory allocation error

**Error Type:**
Serious

**Possible Cause:**
Memory required to driver operations could not be allocated.

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

## 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

**Error Type:**
Warning

**Result:**
**If the tag was being read:**

- If tag is a block tag, the entire block will be invalidated. All tags within that block will be invalidated.
- If tag is an array tag or string tag, just this tag is invalidated.

**If the tag was being written:**

- Write operation for the given tag will not take place.

**Possible Cause:**
1. The connection between the device and the Host PC is broken.
2. Device CPU work load is too high.

3. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device setting.

**Solution:**
1. Verify the cabling between the PC and the PLC device.
2. If this error occurs frequently, decrease the tag group scan rate to reduce the work load on the PLC CPU.
3. Increase the Request Timeout setting so that the entire response can be handled.

## Unable to read device info data in area '<area>'. Board '<board>' returned Error Code '<code>'

**Error Type:**
Warning

**Possible Cause:**
Device info data contains diagnostics information for most Master cards. Some device info data is read individually while other device info data are structured and read as a block. This error pertains to the former. Read access to this data failed.

**Solution:**
Refer to the Error Codes table for specific information.

**See Also:**
Error Codes

## Unable to read '<block size>' device info bytes in area '<area>'. Board '<board>' returned Error Code '<code>'

**Error Type:**
Warning

**Possible Cause:**
Device info data contains diagnostics information for most Master cards. Some device info are read individually while other device info data are structured and read as a block. This error pertains to the latter. Read access to this data failed.

**Solution:**
Refer to the Error Codes table for specific information.

**See Also:**
Error Codes

## Unable to read task state data in task '<task num>'. Board '<board>' returned Error Code '<code>'

**Error Type:**
Warning

**Possible Cause:**
Task state data contains diagnostics information for some Slave cards. Some task data is read individually while other device info data are structured and read as a block. This error pertains to the former. Read access to this data failed.

**Solution:**
Refer to the Error Codes table for specific information.

**See Also:**
Error Codes

## Unable to read '<block size>' task state bytes in task '<task num>'. Board '<board>' returned Error Code '<code>'

**Error Type:**
Warning

**Possible Cause:**
Task state data contains diagnostics information for some Slave cards. Some task data is read individually while other device info data are structured and read as a block. This error pertains to the latter. Read access to this data failed.

**Solution:**
Refer to the Error Codes table for specific information.

**See Also:**
Error Codes

## Unable to read tag '<address>' from device '<device>'. Board '<board>' returned Error Code '<code>'

**Error Type:**
Warning

**Possible Cause:**
The driver was unable to read the data at offset '<address>' in '<board>'s Input image. This data location corresponds to Input data mapped for device '<device>'. This error pertains to all string and array I/O tags.

**Solution:**
Refer to the Error Codes table for specific information.

**See Also:**
Error Codes

## Unable to read '<block size>' bytes starting at '<address>' from device '<device>'. Board '<board>' returned Error Code '<code>'

**Error Type:**
Warning

**Possible Cause:**
The driver was unable to read '<block size>' bytes of data, starting at offset '<address>', in '<board>'s Input image. These data locations correspond to Input data mapped for device '<device>'. This error pertains to all non-string and non-array I/O tags.

**Solution:**
Refer to the Error Codes table for specific information.

**See Also:**
Error Codes

## Unable to write to tag '<address>' from device '<device>'. Board '<board>' returned Error Code '<code>'

**Error Type:**
Warning

**Possible Cause:**
A write failed to offset '<address>' in '<board>'s Output image, corresponding to Output data mapped to device '<device>'.

**Solution:**
Refer to the Error Codes table for specific information.

**See Also:**
Error Codes

## Unable to read tag '<name>': msg.b=<command>, msg.device_adr=<Device ID>...

**Error Type:**
Warning

**Possible Cause:**
The driver was unable to read tag '<name>' because the message associated with '<name>' failed or the response from the message was invalid. The message command and destination device are listed as <command> and <Device ID> respectively. Applicable for diagnostics only.

- Refer to the Error Codes table if the following message is received:
  "...Get message failed on Board '<board>' with Error Code '<code>"
- Either the destination for the message was invalid or unexpected data was received the following message is received:
  "...Response from Board '<board>' contains a framing error"

**Solution:**
Re-download the configuration database in SyCon.

**See Also:**
Error Codes

## Unable to read '<block size>' message bytes: msg.b=<command>, msg.device_adr=<Device ID>...

**Error Type:**
Warning

**Possible Cause:**
The driver was unable to read <block size> bytes of data requested in message <command>, from device <Device ID> because the message failed or the response from the message was invalid. The message command and destination device are listed as <command> and <Device ID> respectively. Applicable for diagnostics only.

- Refer to the Error Codes table if the following message is received:
  "...Get message failed on Board '<board>' with Error Code '<code>"
- Either the destination for the message was invalid or unexpected data was received if the following message is received:
  "...Response from Board '<board>' contains a framing error"

**Solution:**
Re-download the configuration database in SyCon.

**See Also:**
Error Codes

## Unable to write to tag '<address>': msg.b=<command>, msg.device_adr=<Device ID>...

**Error Type:**
Warning

**Possible Cause:**
The driver was unable to write to tag '<name>' because the message associated with '<name>' failed or the response from the message was invalid. The message command and destination device are listed as <command> and <Device ID> respectively. Applicable for diagnostics only.

- Refer to the Error Codes table if the following message is received:
  "...Put message failed on Board '<board>' with Error Code '<code>"
- Either the destination for the message was invalid or unexpected data was received if the following message is received:
  "...Response from Board '<board>' contains a framing error"

**Solution:**

Re-download the configuration database in SyCon.

**See Also:**
Error Codes

### Unable to read tag '<address>' from device '<device>'. Board '<board>' returned DPM Diagnostics [Global Bits='<Global Bits>', Node='<Remote Address>', Code='<Error Event>']

**Error Type:**
Warning

**Possible Cause:**
The driver was unable to read the data at offset '<address>' in '<board>'s Input image. This data location corresponds to Input data mapped for device '<device>'. This error pertains to all non-string and non-array I/O tags.

**Solution:**
Please contact Technical Support for information specific to the returned DPM Diagnostics information.

### Unable to read '<block size>' bytes starting at '<address>' from device '<device>'. Board '<board>' returned DPM Diagnostics [Global Bits='<Global Bits>', Node='<Remote Address>', Code='<Error Event>']

**Error Type:**
Warning

**Possible Cause:**
The driver was unable to read '<block size>' bytes of data, starting at offset '<address>', in '<board>'s Input image. These data locations correspond to Input data mapped for device '<device>'. This error pertains to all non-string and non-array I/O tags.

**Solution:**
Please contact Technical Support for information specific to the returned DPM Diagnostics information.

### Unable to write to tag '<address>' from device '<device>'. Board '<board>' returned DPM Diagnostics [Global Bits='<Global Bits>', Node='<Remote Address>', Code='<Error Event>']

**Error Type:**
Warning

**Possible Cause:**
A write failed to offset '<address>' in '<board>'s Output image, corresponding to Output data mapped to device '<device>'.

**Solution:**
Please contact Technical Support for information specific to the returned DPM Diagnostics information.

### Unable to read tag '<address>' from device '<device>'. Board '<board>' returned DNM Diagnostics [Global Bits='<Global Bits>', Node='<Device Address>', Code='<Error Event>']

**Error Type:**
Warning

**Possible Cause:**
The driver was unable to read the data at offset '<address>' in '<board>'s Input image. This data location corresponds to Input data mapped for device '<device>'. This error pertains to all non-string and non-array I/O tags.

**Solution:**
Please contact Technical Support for information specific to the returned DPM Diagnostics information.

## Unable to read '<block size>' bytes starting at '<address>' from device '<device>'. Board '<board>' returned DNM Diagnostics [Global Bits='<Global Bits>', Node='<Device Address>', Code='<Error Event>']

**Error Type:**
Warning

**Possible Cause:**
The driver was unable to read '<block size>' bytes of data, starting at offset '<address>', in '<board>'s Input image. These data locations correspond to Input data mapped for device '<device>'. This error pertains to all non-string and non-array I/O tags.

**Solution:**
Please contact Technical Support for information specific to the returned DPM Diagnostics information.

## Unable to write to tag '<address>' from device '<device>'. Board '<board>' returned DNM Diagnostics [Global Bits='<Global Bits>', Node='<Device Address>', Code='<Error Event>']

**Error Type:**
Warning

**Possible Cause:**
A write failed to offset '<address>' in '<board>'s Output image, corresponding to Output data mapped to device '<device>'.

**Solution:**
Please contact Technical Support for information specific to the returned DPM Diagnostics information.

## 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**
**The file is not a valid Sycon database or may be corrupt**
**Auto tag database generation cannot be performed while the driver is processing tags**
**Board Type for Board '<board number>' does not match the actual board installed. Verify Board Type and/or Board Selection**
**Board Type for Board '<board number> does not match the Slave Type for one or more Slaves configured. Delete or edit Slaves accordingly**
**'dbm32.dll' is not loaded and is required for auto tag generation. Verify SyCon is installed**

## The file is not a valid Sycon database or may be corrupt

**Error Type:**
Warning

**Possible Cause:**
1. The file is not a valid Sycon database.
2. The file is corrupt.

**Solution:**
1. Ensure that the file is a valid Sycon database.
2. Ensure that the file is not corrupt.
3. Attempt using a new and valid file.

## Auto tag database generation cannot be performed while the driver is processing tags

**Error Type:**
Warning

**Possible Cause:**

Automatic tag database generation was attempted while the driver was processing tags.

**Solution:**
Ensure that the driver is not processing tags and then reattempt automatic tag database generation.

## Board Type for Board '<board number>' does not match the actual board installed. Verify Board Type and/or Board Selection

**Error Type:**
Warning

**Possible Cause:**
The Board Type that is being used does not match the board that is being installed.

**Solution:**
Verify the Board Type and/or the Board Selection.

## Board Type for Board '<board number> does not match the Slave Type for one or more Slaves configured. Delete or edit Slaves accordingly

**Error Type:**
Warning

**Possible Cause:**
The Board Type does not match the Slave Type for one or more of the slaves being configured.

**Solution:**
Edit or delete the slaves as necessary in order to ensure that the Board Type matches the Slave Type.

## 'dbm32.dll' is not loaded and is required for auto tag generation. Verify SyCon is installed

**Error Type:**
Warning

**Possible Cause:**
'dbm32.dll' is not loaded.

**Solution:**
1. Load 'dbm32.dll' and then reattempt automatic tag database generation.
2. Verify that Sycon is installed.

# Index

## C



## D

## E



## F



## G



## I

## L



## M



## O



## P



## R



## S

## T



## U

## W