

SattBus Ethernet Driver

© 2025 PTC Inc. All Rights Reserved.

Table of Contents

| | |
|---|-----------|
| SattBus Ethernet Driver | 1 |
| Table of Contents | 2 |
| Welcome to the SattBus Ethernet Driver Help Center | 3 |
| Overview | 4 |
| Setup | 4 |
| Channel Properties – General | 5 |
| Tag Counts | 5 |
| Channel Properties – Ethernet Communications | 5 |
| Channel Properties – Write Optimizations | 6 |
| Channel Properties – Advanced | 6 |
| Device Properties – General | 7 |
| Operating Mode | 8 |
| Tag Counts | 8 |
| Device Properties – Scan Mode | 9 |
| Device Properties – Timing | 9 |
| Device Properties – Communication Parameters | 11 |
| Changing the Local Port Number and Network Adapter Properties | 11 |
| Device Properties – Settings | 12 |
| Bit Ordering for Memory Cells | 12 |
| Device Properties – Redundancy | 15 |
| Data Types Description | 16 |
| Address Descriptions | 17 |
| Overlapped vs. None-Overlapped Addressing | 18 |
| Error Descriptions | 19 |
| Address '<address>' is out of range for the specified device or register | 19 |
| Array size is out of range for address '<address>' | 19 |
| Array support is not available for the specified address: '<address>' | 19 |
| Data Type '<type>' is not valid for device address '<address>' | 19 |
| 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 |
| Missing address | 20 |
| Device '<device name>' is not responding | 20 |
| Unable to bind to adapter: '<adapter name>' for device '<device name>'. Connection failed | 21 |
| Unable to write to '<address>' on device '<device name>' | 21 |
| Index | 23 |

Welcome to the SattBus Ethernet Driver Help Center

This help center is the user documentation for Kepware SattBus Ethernet Driver. This help center is updated regularly to reflect the latest functionality and information.

[Overview](#)

What is the SattBus Ethernet Driver?

[Setup](#)

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

[Data Types Description](#)

What data types are supported by this driver?

[Address Descriptions](#)

How do I address a data location on a device?

[Error Descriptions](#)

What messages are produced by the SattBus Ethernet Driver?

Version 1.030

© 2025 PTC Inc. All Rights Reserved.

Overview

The SattBus Ethernet Driver provides a reliable way to connect SattBus Ethernet devices to OPC client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with Sattcon devices communicating via the SattBus Ethernet interface.

● **Note:** A standard Ethernet interface is used as the hardware for connecting PCs to a SattBus system with Ethernet capability.

Setup

Supported Devices

Sattcon 200

● **Note:** Any device that supports the SattBus protocol (and has the SattBus Ethernet interface/module present) will be supported.

Communication Protocol

SattBus

Channel and Device Limits

The maximum number of channels supported by this driver is 16. The maximum number of devices supported by this driver is 2048 per channel.

Device IDs

The Device ID is the device IP address.

Connection

Standard Ethernet connection

Overlapping

Overlapping affects the procedure of reading and writing certain addresses. By default, this option is set to "Yes" for compatibility with the SattBus Serial port driver. *For more information, refer to [Overlapped Addressing vs. None-Overlapped Addressing](#).*

Channel Properties – General

This server supports the use of multiple simultaneous communications drivers. Each protocol or driver used in a server project is called a channel. A server project may consist of many channels with the same communications driver or with unique communications drivers. A channel acts as the basic building block of an OPC link. This group is used to specify general channel properties, such as the identification attributes and operating mode.

| | | |
|---------------------|--|---------|
| Property Groups | <input type="checkbox"/> Identification | |
| General | Name | |
| Write Optimizations | Description | |
| Advanced | Driver | |
| | <input type="checkbox"/> Diagnostics | |
| | Diagnostics Capture | Disable |
| | <input type="checkbox"/> Tag Counts | |
| | Static Tags | 10 |

Identification

Name: Specify the user-defined identity of this channel. In each server project, each channel name must be unique. Although names can be up to 256 characters, some client applications have a limited display window when browsing the OPC server's tag space. The channel name is part of the OPC browser information. The property is required for creating a channel.

• For information on reserved characters, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in the server help.

Description: Specify user-defined information about this channel.

• Many of these properties, including Description, have an associated system tag.

Driver: Specify the protocol / driver for this channel. Specify the device driver that was selected during channel creation. It is a disabled setting in the channel properties. The property is required for creating a channel.

• **Note:** With the server's online full-time operation, these properties can be changed at any time. This includes changing the channel name to prevent clients from registering data with the server. If a client has already acquired an item from the server before the channel name is changed, the items are unaffected. If, after the channel name has been changed, the client application releases the item and attempts to re-acquire using the old channel name, the item is not accepted. Changes to the properties should not be made once a large client application has been developed. Utilize proper user role and privilege management to prevent operators from changing properties or accessing server features.

Diagnostics

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

• **Note:** This property is not available if the driver or operating system does not support diagnostics.

• For more information, refer to *Communication Diagnostics and Statistics Tags* in server help.

Tag Counts

Static Tags: Provides the total number of defined static tags at this level (device or channel). This information can be helpful in troubleshooting and load balancing.

Channel Properties – Ethernet Communications

Ethernet Communication can be used to communicate with devices.

| | | |
|--------------------------------|-------------------|---------|
| Property Groups | Ethernet Settings | |
| General | Network Adapter | Default |
| Ethernet Communications | | |
| Write Optimizations | | |
| Advanced | | |

Ethernet Settings

Network Adapter: Specify the network adapter to bind. When left blank or Default is selected, the operating system selects the default adapter.

Channel Properties – Write Optimizations

The server must ensure that the data written from the client application gets to the device on time. Given this goal, the server provides optimization properties to meet specific needs or improve application responsiveness.

| | | |
|----------------------------|---------------------|--------------------------------------|
| Property Groups | Write Optimizations | |
| General | Optimization Method | Write Only Latest Value for All Tags |
| Write Optimizations | Duty Cycle | 10 |
| | | |

Write Optimizations

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

- **Write All Values for All Tags:** This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing data to the device as quickly as possible. This mode ensures that everything written from the client applications is sent to the target device. This mode should be selected if the write operation order or the write item's content must uniquely be seen at the target device.
- **Write Only Latest Value for Non-Boolean Tags:** Many consecutive writes to the same value can accumulate in the write queue due to the time required to actually send the data to the device. If the server updates a write value that has already been placed in the write queue, far fewer writes are needed to reach the same final output value. In this way, no extra writes accumulate in the server's queue. When the user stops moving the slide switch, the value in the device is at the correct value at virtually the same time. As the mode states, any value that is not a Boolean value is updated in the server's internal write queue and sent to the device at the next possible opportunity. This can greatly improve the application performance.
 - **Note:** This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.
- **Write Only Latest Value for All Tags:** This option takes the theory behind the second optimization mode and applies it to all tags. It is especially useful if the application only needs to send the latest value to the device. This mode optimizes all writes by updating the tags currently in the write queue before they are sent. This is the default mode.

Duty Cycle: is used to control the ratio of write to read operations. The ratio is always based on one read for every one to ten writes. The duty cycle is set to ten by default, meaning that ten writes occur for each read operation. Although the application is performing a large number of continuous writes, it must be ensured that read data is still given time to process. A setting of one results in one read operation for every write operation. If there are no write operations to perform, reads are processed continuously. This allows optimization for applications with continuous writes versus a more balanced back and forth data flow.

● **Note:** It is recommended that the application be characterized for compatibility with the write optimization enhancements before being used in a production environment.

Channel Properties – Advanced

This group is used to specify advanced channel properties. Not all drivers support all properties; so the Advanced group does not appear for those devices.

| | | |
|---------------------|---|-------------------|
| Property Groups | <input type="checkbox"/> Non-Normalized Float Handling | |
| General | Floating-Point Values | Replace with Zero |
| Write Optimizations | <input type="checkbox"/> Inter-Device Delay | |
| Advanced | Inter-Device Delay (ms) | 0 |

Non-Normalized Float Handling: A non-normalized value is defined as Infinity, Not-a-Number (NaN), or as a Denormalized Number. The default is Replace with Zero. Drivers that have native float handling may default to Unmodified. Non-normalized float handling allows users to specify how a driver handles non-normalized IEEE-754 floating point data. Descriptions of the options are as follows:

- **Replace with Zero:** This option allows a driver to replace non-normalized IEEE-754 floating point values with zero before being transferred to clients.
- **Unmodified:** This option allows a driver to transfer IEEE-754 denormalized, normalized, non-number, and infinity values to clients without any conversion or changes.

● **Note:** This property is disabled if the driver does not support floating-point values or if it only supports the option that is displayed. According to the channel's float normalization setting, only real-time driver tags (such as values and arrays) are subject to float normalization. For example, EFM data is not affected by this setting.

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

Inter-Device Delay: Specify the amount of time the communications channel waits to send new requests to the next device after data is received from the current device on the same channel. Zero (0) disables the delay.

● **Note:** This property is not available for all drivers, models, and dependent settings.

Device Properties – General

A device represents a single target on a communications channel. If the driver supports multiple controllers, users must enter a device ID for each controller.

| | | |
|-----------------|--|---------|
| Property Groups | <input type="checkbox"/> Identification | |
| General | Name | |
| | Description | |
| | Channel Assignment | |
| | Driver | |
| | Model | |
| | ID Format | Decimal |
| | ID | 2 |

Identification

Name: Specify the name of the device. It is a logical user-defined name that can be up to 256 characters long and may be used on multiple channels.

● **Note:** Although descriptive names are generally a good idea, some OPC client applications may have a limited display window when browsing the OPC server's tag space. The device name and channel name become part of the browse tree information as well. Within an OPC client, the combination of channel name and device name would appear as "ChannelName.DeviceName".

● *For more information, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in server help.*

Description: Specify the user-defined information about this device.

● Many of these properties, including Description, have an associated system tag.

Channel Assignment: Specify the user-defined name of the channel to which this device currently belongs.

Driver: Selected protocol driver for this device.

Model: Specify the type of device that is associated with this ID. The contents of the drop-down menu depend on the type of communications driver being used. Models that are not supported by a driver are disabled. If the communications driver supports multiple device models, the model selection can only be changed when there are no client applications connected to the device.

● **Note:** If the communication driver supports multiple models, users should try to match the model selection to the physical device. If the device is not represented in the drop-down menu, select a model that conforms closest to the target device. Some drivers support a model selection called "Open," which allows users to communicate without knowing the specific details of the target device. *For more information, refer to the driver documentation.*

ID: Specify the device's driver-specific station or node. The type of ID entered depends on the communications driver being used. For many communication drivers, the ID is a numeric value. Drivers that support a Numeric ID provide users with the option to enter a numeric value whose format can be changed to suit the needs of the application or the characteristics of the selected communications driver. The format is set by the driver by default.

Options include Decimal, Octal, and Hexadecimal.

● **Note:** If the driver is Ethernet-based or supports an unconventional station or node name, the device's TCP/IP address may be used as the device ID. TCP/IP addresses consist of four values that are separated by periods, with each value in the range of 0 to 255. Some device IDs are string based. There may be additional properties to configure within the ID field, depending on the driver.

Operating Mode

| | | |
|-----------------|---|--------|
| Property Groups | <div> <div>+</div> Identification </div> <div> <div>-</div> Operating Mode </div> | |
| General | Data Collection | Enable |
| | Simulated | No |

Data Collection: This property controls the device's active state. Although device communications are enabled by default, this property can be used to disable a physical device. Communications are not attempted when a device is disabled. From a client standpoint, the data is marked as invalid and write operations are not accepted. This property can be changed at any time through this property or the device system tags.

Simulated: Place the device into or out of Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client as valid data. While in Simulation Mode, the server treats all device data as reflective: whatever is written to the simulated device is read back and each OPC item is treated individually. The data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

● Notes:

1. Updates are not applied until clients disconnect and reconnect.
2. The System tag (_Simulated) is read only and cannot be written to for runtime protection. The System tag allows this property to be monitored from the client.
3. In Simulation mode, the item's memory map is based on client update rate(s) (Group Update Rate for OPC clients or Scan Rate for native and DDE interfaces). This means that two clients that reference the same item with different update rates return different data.
4. When a device is simulated, updates may not appear faster than one (1) second in the client.

● Simulation Mode is for test and simulation purposes only. It should never be used in a production environment.

Tag Counts

| | | |
|-----------------|---|-----|
| Property Groups | <input type="checkbox"/> Identification <input type="checkbox"/> Operating Mode <input type="checkbox"/> Tag Counts | |
| General | | |
| | Static Tags | 130 |

Static Tags: Provides the total number of defined static tags at this level (device or channel). This information can be helpful in troubleshooting and load balancing.

Device Properties – Scan Mode

The Scan Mode specifies the subscribed-client requested scan rate for tags that require device communications. Synchronous and asynchronous device reads and writes are processed as soon as possible; unaffected by the Scan Mode properties.

| | | |
|-----------------|------------------------------------|--------------------------------------|
| Property Groups | <input type="checkbox"/> Scan Mode | |
| General | | |
| Scan Mode | Scan Mode | Respect Client-Specified Scan Rate ▼ |
| | Initial Updates from Cache | Disable |

Scan Mode: Specify how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

- **Respect Client-Specified Scan Rate:** This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate:** This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
 - **Note:** When the server has an active client and items for the device and the scan rate value is increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.
- **Request All Data at Scan Rate:** This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only:** This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the OPC client's responsibility to poll for updates, either by writing to the _DemandPoll tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help.*
- **Respect Tag-Specified Scan Rate:** This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

Initial Updates from Cache: When enabled, this option allows the server to provide the first updates for newly activated tag references from stored (cached) data. Cache updates can only be provided when the new item reference shares the same address, scan rate, data type, client access, and scaling properties. A device read is used for the initial update for the first client reference only. The default is disabled; any time a client activates a tag reference the server attempts to read the initial value from the device.

Device Properties – Timing

The device Timing properties allow the driver's response to error conditions to be tailored to fit the application's needs. In many cases, the environment requires changes to these properties for optimum performance. Factors such as electrically generated noise, modem delays, and poor physical connections can influence how many errors or timeouts a communications driver encounters. Timing properties are specific to each configured device.

| | | |
|-----------------|---|------|
| Property Groups | <input type="checkbox"/> Communication Timeouts | |
| General | | |
| Scan Mode | | |
| Timing | Connect Timeout (s) | 3 |
| | Request Timeout (ms) | 1000 |
| | Attempts Before Timeout | 3 |

Communications Timeouts

Connect Timeout: This property (which is used primarily by Ethernet based drivers) controls the amount of time required to establish a socket connection to a remote device. The device's connection time often takes longer than normal communications requests to that same device. The valid range is 1 to 30 seconds. The default is typically 3 seconds, but can vary depending on the driver's specific nature. If this setting is not supported by the driver, it is disabled.

● **Note:** Due to the nature of UDP connections, the connection timeout setting is not applicable when communicating via UDP.

Request Timeout: Specify an interval used by all drivers to determine how long the driver waits for a response from the target device to complete. The valid range is 50 to 9999999 milliseconds (167 minutes). The default is usually 1000 milliseconds, but can vary depending on the driver. The default timeout for most serial drivers is based on a baud rate of 9600 baud or better. When using a driver at lower baud rates, increase the timeout to compensate for the increased time required to acquire data.

Attempts Before Timeout: Specify how many times the driver issues a communications request before considering the request to have failed and the device to be in error. The valid range is 1 to 10. The default is typically 3, but can vary depending on the driver's specific nature. The number of attempts configured for an application depends largely on the communications environment. This property applies to both connection attempts and request attempts.

Timing

Inter-Request Delay: Specify how long the driver waits before sending the next request to the target device after receiving the response to the previous request. It overrides the normal polling frequency of tags associated with the device, as well as one-time reads and writes. This delay can be useful when dealing with devices with slow turn-around times and in cases where network load is a concern. Configuring a delay for a device affects communications with all other devices on the channel. It is recommended that users separate any device that requires an inter-request delay to a separate channel if possible. Other communications properties (such as communication serialization) can extend this delay. The valid range is 0 to 300,000 milliseconds; however, some drivers may limit the maximum value due to a function of their particular design. The default is 0, which indicates no delay between requests with the target device.

● **Note:** Not all drivers support Inter-Request Delay. This setting does not appear if it is not available.

| | | |
|--|--------------------------|---|
| Property Groups General Scan Mode Timing | Timing | |
| | Inter-Request Delay (ms) | 0 |

Device Properties – Communication Parameters

These properties specify the device's protocol and port numbers.

| | | |
|----------------------------------|------------------------------------|------|
| Property Groups | ☐ Communications Parameters | |
| General | Port Number | 2999 |
| Scan Mode | IP Protocol | UDP |
| Timing | Overlapped addressing | Yes |
| Communications Parameters | Write to | 2999 |

Port Number: Specify the local port number to which the device will send messages. The default setting is 2999.

IP Protocol: Specify the protocol type. The only protocol used for SattBus over Ethernet is UDP.

Overlapped Addressing: Specify the addressing of register double-words. The default setting is Yes. *For more information, refer to [Overlapped vs. None-Overlapped Addressing](#).*

Write To: Specify the port number on which the device will receive messages. The default setting is 2999.

● **Note:** Although the driver supports the configuration of communication port numbers, the actual port numbers that the Sattbus device expects for successful communication are fixed. To communicate with the Sattbus device, both the Local and Write To port numbers must be 2999.

Changing the Local Port Number and Network Adapter Properties

For information on changing the Local Port Number and the Network Adapter properties, refer to the instructions below.

Changing the Local Port Number Property

1. To start, right-click on the device and then select **Properties**.
2. Next, click on the **Communication Parameters** property group and refer to the **Port Number** field.
3. Make changes to the property as desired by typing the number or using the arrows to the right.

| | | |
|----------------------------------|------------------------------------|------|
| Property Groups | ☐ Communications Parameters | |
| General | Port Number | 2999 |
| Scan Mode | IP Protocol | UDP |
| Timing | Overlapped addressing | Yes |
| Communications Parameters | Write to | 2999 |

● **Caution:** Some SattBus Ethernet modules are configured to send responses to a fixed, predefined port number (such as 2999) on the PC. As such, when the local port number is changed, the bind may succeed although the actual communications fail.

Changing the Network Adapter Property

1. To start, right-click on the channel and then select **Properties**.
2. Next, click on the **Ethernet Settings** property group and refer to the **Network Adapter** field.

| | |
|----------------------------|---------|
| ☐ Ethernet Settings | |
| Network Adapter | Default |

● **Note:** If only a single Network Interface Card (NIC) is in the machine, users will have to multi-home the PC. Once completed, the "Default" adapter should not be selected for any of the other channels.

Device Properties – Settings

| | |
|---------------------------|--|
| Property Groups | <input type="checkbox"/> Settings |
| General | Bit ordering MSBit 15 14 13 12 11 10 9 8 7 ... |
| Scan Mode | <input type="checkbox"/> Block size |
| Communications Parameters | Register Block Size 20 |
| Settings | I/O RAM/Memory Cell Block Size 20 |

Bit Ordering: These properties configure the bit ordering that will be used when reading or writing Word memory cell tags. The default setting is MSBit 7| 6| 5| 4| 3| 2| 1| 0| 15| 14| 13| 12| 11| 10| 9| 8| LSBit.


Register Block Size: Specify the number of bytes of Register data (such as R tags) that may be requested from a device at one time. To refine the driver's performance, configure the block size to one of the following settings: 20, 32, 64, 128, 256, or 510. The default setting is 20 bytes.

I/O RAM/Memory Cell Block Size: Specify the number of bytes of I/O RAM/Memory Cell data (such as X, XB, XW, M, MW, I, O or IO tags) that may be requested from a device at one time. To refine the driver's performance, configure the block size to one of the following settings: 20, 32, 64, 128, or 255. The default setting is 20 bytes.

Bit Ordering for Memory Cells

For more information on a selection, select a link from the list below.

[MSBit 0| 1| 2| 3| 4| 5| 6| 7| 8| 9| 10| 11| 12| 13| 14| 15 LSBit](#)
[MSBit 15| 14| 13| 12| 11| 10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0 LSBit](#)
[MSBit 7| 6| 5| 4| 3| 2| 1| 0| 15| 14| 13| 12| 11| 10| 9| 8 LSBit](#)

 **Note:** Addresses are in octal.

MSBit 0| 1| 2| 3| 4| 5| 6| 7| 8| 9| 10| 11| 12| 13| 14| 15 LSBit

This is how the DOX10 programming software represents the memory cells. In this condition, the most significant bit (MSBit) of the tag being read or written is equal to the address of the tag.

1. For the Word tag 'MW00' (address '0'), the most significant bit (MSBit) is the memory cell address '0'. The least significant bit (LSBit) is memory cell address '17' as shown below:

| <----- MW00 -----> | | | | | | | | | | | | | | | | | |
|--------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|-------|
| MSBit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | LSBit |

Read

If memory addresses '2' and '13' are ON, and rest all are OFF as shown below:

| <----- MW00 -----> | | | | | | | | | | | | | | | | | |
|--------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|---------|
| State | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | =>16416 |
| MSBit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | LSBit |

then the value that the tag 'MW00' will read is: 16416.

Write

Writing a value '1' to the tag 'MW00', will set the bit memory addresses as shown below:

| <----- MW00 -----> | | | | | | | | | | | | | | | | | |
|--------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|-------|
| State | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | =>1 |
| MSBit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | LSBit |

2. For the Word tag 'MW11'*(address '11'), the most significant bit (MSBit) is the memory bit '11'. The least significant bit (LSBit) is memory bit '31' as shown below:

● **Note:** Word access at non-byte boundaries is not allowed in the DOX10 programming software.

| <----- MW11 -----> | | | | | | | | | | | | | | | | | |
|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| MSBit | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 20 | 21 | 22 | 23 | 25 | 26 | 27 | 30 | 31 | LSBit |

Read

If memory addresses '14' and '27' are ON, and rest all are OFF as shown below:

| <----- MW11 -----> | | | | | | | | | | | | | | | | | |
|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------|
| State | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | =>4100 |
| MSBit | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 20 | 21 | 22 | 23 | 25 | 26 | 27 | 30 | 31 | LSBit |

then the value that the tag 'MW11' will read is: 4100.

Write

Writing a value '258' to the tag 'MW11', will set the bit memory addresses as shown below:

| <----- MW11 -----> | | | | | | | | | | | | | | | | | |
|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| State | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | =>258 |
| MSBit | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 20 | 21 | 22 | 23 | 25 | 26 | 27 | 30 | 31 | LSBit |

● **Note:** Due to the nature of this bit ordering option, care is needed when addressing 8-bit memory that is close to 16-bit addresses.

MSBit 15| 14| 13| 12| 11| 10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0 LSBit

In this condition, the least significant bit (LSBit) of the tag being read or written is equal to the address of the tag.

1. For the Word tag 'MW00' (address '0'), the most significant bit (MSBit) is the memory cell address '17'. The least significant bit (LSBit) is memory cell address '0' as shown below:

| <----- MW00 -----> | | | | | | | | | | | | | | | | | |
|--------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|-------|
| LSBit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | MSBit |

Read

If memory addresses '2' and '13' are ON, and rest all are OFF as shown below:

| <----- MW00 -----> | | | | | | | | | | | | | | | | | |
|--------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|--------|
| State | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | =>2052 |
| LSBit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | MSBit |

then the value that the tag 'MW00' will read is: 2052.

Write

Writing a value '1' to the tag 'MW00', will set the bit memory addresses as shown below:

| <----- MW00 -----> | | | | | | | | | | | | | | | | | |
|--------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|-------|
| State | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | =>1 |
| LSBit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | MSBit |

2. For the Word tag 'MW11'(address '11'), the most significant bit (MSBit) is the memory bit '31'. The least significant bit (LSBit) is memory bit '11' as shown below:

Note: Word access at non-byte boundaries is not allowed in the DOX10 programming software.

| <----- MW11 -----> | | | | | | | | | | | | | | | | | |
|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| LSBit | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 20 | 21 | 22 | 23 | 25 | 26 | 27 | 30 | 31 | MSBit |

Read

If memory addresses '14' and '27' are ON, and rest all are OFF as shown below:

| <----- MW11 -----> | | | | | | | | | | | | | | | | | |
|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------|
| State | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | =>8200 |
| LSBit | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 20 | 21 | 22 | 23 | 25 | 26 | 27 | 30 | 31 | MSBit |

then the value that the tag 'MW11' will read is: 8200.

Write

Writing a value '258' to the tag 'MW11', will set the bit memory addresses as shown below:

| <----- MW11 -----> | | | | | | | | | | | | | | | | | |
|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| State | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | =>258 |
| LSBit | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 20 | 21 | 22 | 23 | 25 | 26 | 27 | 30 | 31 | MSBit |

MSBit 7| 6| 5| 4| 3| 2| 1| 0| 15| 14| 13| 12| 11| 10| 9| 8 LSBit

This condition, which is the default, is similar to the second selection described above, but with the bytes swapped.

1. For the Word tag 'MW00' (address '0'), the most significant bit (MSBit) is the memory cell address '7'. The least significant bit (LSBit) is memory cell address '10' as shown below:

| <----- MW00 -----> | | | | | | | | | | | | | | | | | |
|--------------------|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-------|
| LSBit | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | MSBit |

Read

If memory addresses '2' and '13' are ON, and rest all are OFF as shown below:

| <----- MW00 -----> | | | | | | | | | | | | | | | | | |
|--------------------|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|--------|
| State | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | =>1032 |
| LSBit | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | MSBit |

then the value that the tag 'MW00' will read is: 1032.

Write

Writing a value '1' to the tag 'MW00', will set the bit memory addresses as shown below:

| <----- MW00 -----> | | | | | | | | | | | | | | | | | |
|--------------------|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-------|
| State | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | =>1 |
| LSBit | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | MSBit |

2. For the Word tag 'MW11'*(address '11'), the most significant bit (MSBit) is the memory bit '20'. The least significant bit (LSBit) is memory bit '21' as shown below:

● **Note:** Word access at non-byte boundaries is not allowed in the DOX10 programming software.

| <----- MW11 -----> | | | | | | | | | | | | | | | | | |
|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| LSBit | 21 | 22 | 23 | 25 | 26 | 27 | 30 | 31 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 20 | MSBit |

Read

If memory addresses '14' and '27' are ON, and rest all are OFF as shown below:

| <----- MW11 -----> | | | | | | | | | | | | | | | | | |
|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------|-------|
| State | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | =>2080 | |
| LSBit | 21 | 22 | 23 | 25 | 26 | 27 | 30 | 31 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 20 | MSBit |

then the value that the tag 'MW11' will read is: 2080.

Write

Writing a value '258' to the tag 'MW11', will set the bit memory addresses as shown below:

| <----- MW11 -----> | | | | | | | | | | | | | | | | | |
|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| State | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | =>258 |
| LSBit | 21 | 22 | 23 | 25 | 26 | 27 | 30 | 31 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 20 | MSBit |

● **Note:** Due to the nature of this bit ordering option, care is needed when addressing 8-bit memory that is close to 16-bit addresses.

Device Properties – Redundancy

| | | |
|-------------------|------------------------|----------------------|
| Property Groups | Redundancy | |
| General | Secondary Path | Channel.Device 1 ... |
| Scan Mode | Operating Mode | Switch On Failure |
| Timing | Monitor Item | |
| Auto-Demotion | Monitor Interval (s) | 300 |
| Redundancy | Return to Primary ASAP | Yes |

Redundancy is available with the Media-Level Redundancy Plug-In.

● Consult the website, a sales representative, or the [user manual](#) for more information.

Data Types Description

| Data Type | Description |
|-----------|---|
| Boolean | Single bit |
| Byte | Unsigned 8-bit value bit 0 is the low bit bit 7 is the high bit |
| Char | Signed 8-bit value bit 0 is the low bit bit 6 is the high bit bit 7 is the sign bit |
| Word | Unsigned 16-bit value byte 0 is the high byte byte 1 is the low byte |
| Short | Signed 16-bit value byte 0 is the high byte byte 1 is the low byte bit 0 is the sign bit |
| DWord | Unsigned 32-bit value byte 0 is the high byte byte 7 is the low byte |
| Long | Signed 32-bit value byte 0 is the high byte byte 7 is the low byte bit 0 is the sign bit |

Address Descriptions

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

| Address Type | Range | Data Type | Access |
|---|---|---|--|
| Register Addresses are in decimal. | R0-R32767 R0.00-R32767.00 ...R0.15-R32767.15 RW0-RW32767 RW0.00-RW32767.00 ...RW0.15-RW32767.15 RD0-RD32766** | Word , Short Boolean Word , Short Boolean DWord , Long | Read/Write Read Only Read/Write Read Only Read/Write |
| I/O RAM bits Addresses are in octal. | 0-77777 IO0-IO77777 IO-IO77777 OO-O77777 X0-X77777 | Boolean Boolean Boolean Boolean Boolean | Read/Write |
| I/O RAM Addresses are in octal; bit numbers are in decimal.* | XB0-XB77770 XW0.00-XW77760.00 ...XW0.15-XW77760.15 XW0-XW77760 | Byte , Char Boolean Word , Short | Read/Write |
| Memory Cell in I/O RAM Bit numbers are in decimal. | M0-M77760 M0.00-M77760.00 ...M0.15-M77760.15 MW0-MW77760 MW0.00-MW77760.00 ...MW0.15-MW77760.15 | Word , Short Boolean Word , Short Boolean | Read/Write |

*Addresses can be on non-byte boundary (non-multiples of 10 octal).

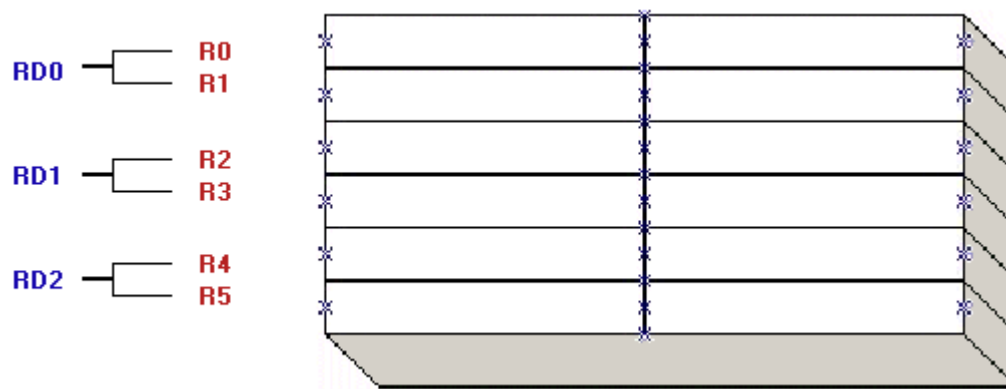
**For more information, refer to [Overlapped vs. None-Overlapped Addressing](#).

Overlapped vs. None-Overlapped Addressing

This device property influences the addressing of register double-words. By default, the driver is set to Overlapped Mode for compatibility with the SattBus Serial driver.

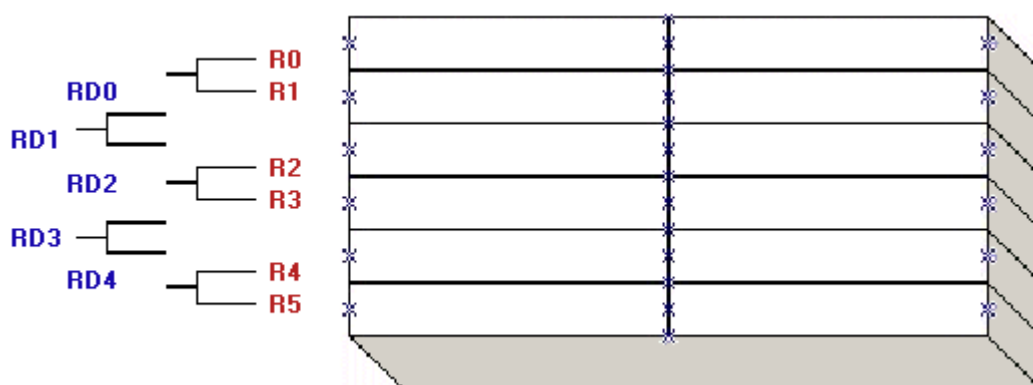
None-Overlapped Mode

In None-Overlapped Mode, RD1 starts with the byte following the second byte of RD0.



Overlapped Mode

In Overlapped Mode, the byte address matches the first byte of the double word. For example, RD3 is the first byte of double word RD3.



Error Descriptions

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

Address Validation

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

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

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

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

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

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

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

[Missing address](#)

Device Status Messages

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

[Unable to bind to adapter: '<adapter name>' for device '<device name>'. Connection failed](#)

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

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.

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

Error Type:

Warning

Possible Cause:

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

Solution:

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

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

Error Type:

Warning

Possible Cause:

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

Solution:

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

Data Type '<type>' is not valid for device address '<address>'

Error Type:

Warning

Possible Cause:

A tag address that has been specified statically has been assigned an invalid data type.

Solution:

Modify the requested data type in the client application.

Device address '<address>' contains a syntax error

Error Type:

Warning

Possible Cause:

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

Solution:

Re-enter the address in the client application.

Device address '<address>' is 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.

Device address '<address>' is Read Only

Error Type:

Warning

Possible Cause:

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

Solution:

Change the access mode in the client application.

Missing address

Error Type:

Warning

Possible Cause:

A tag address that has been specified statically has no length.

Solution:

Re-enter the address in the client application.

Device '<device name>' is not responding

Error Type:

Serious

Possible Cause:

1. The Ethernet connection between the device and the Host PC is broken.
2. The named device may have been assigned an incorrect Network ID.
3. The communication properties for the Ethernet connection are incorrect.
4. The Ethernet configuration for the LAN adapter is incorrect.
5. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device property.

Solution:

1. Verify the cabling between the PC and the device.
2. Verify that the Network ID given to the named device matches that of the actual device.
3. Verify that the specified communication properties match those of the device.
4. Verify that the LAN adapter is correctly configured: that the device driver functions properly, the TCP/IP protocol is installed and that the PLC has a valid IP address.
5. Increase the Request Timeout property so that the entire response can be handled.

Unable to bind to adapter: '<adapter name>' for device '<device name>'. Connection failed

Error Type:

Fatal

Possible Cause:

Some other device in the driver has already been bound to the adapter/port combination.

Solution:

Change the adapter/port combination so that it is unique.

Note:

In order for the device to successfully bind to the local adapter, the combination of adapter and local port number must be unique.

See Also:

[Changing the Local Port Number and Network Adapter Properties](#)

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

Error Type:

Serious

Possible Cause:

1. The Ethernet connection between the device and the Host PC is broken.
2. The named device may have been assigned an incorrect Network ID.
3. The communication properties for the Ethernet connection are incorrect.
4. The Ethernet configuration for the LAN adapter incorrect.

Solution:

1. Verify the cabling between the PC and the device.
2. Verify that the Network ID given to the named device matches that of the actual device.
3. Verify that the specified communication properties match those of the device.
4. Verify that the LAN adapter is correctly configured: that the device driver functions properly, the TCP/IP protocol is installed and that the PLC has a valid IP address.

Index

A

Address '<address>' is out of range for the specified device or register 19

Address Descriptions 17

Array size is out of range for address '<address>' 19

Array support is not available for the specified address: '<address>' 19

Attempts Before Timeout 10

B

Bit Ordering for Memory Cells 12

Boolean 16

C

Changing the Local Port Number and Network Adapter Properties 11

Channel Assignment 7

Channel Properties – Advanced 6

Channel Properties – Ethernet Communications 5

Channel Properties – General 5

Channel Properties – Write Optimizations 6

Communications Timeouts 9

Connect Timeout 10

D

Data Collection 8

Data Type '<type>' is not valid for device address '<address>' 19

Data Types Description 16

Device '<device name>' is not responding 20

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 ID 4

Device Properties – General 7

Device Properties – Redundancy 15

Device Properties – Timing 9

Diagnostics 5

Do Not Scan, Demand Poll Only 9

Driver 7

Duty Cycle 6

DWord 16

E

Error Descriptions 19

Ethernet Settings 6

G

General 7

I

ID 8

Identification 5, 7

Initial Updates from Cache 9

Inter-Device Delay 7

L

Long 16

M

Missing address 20

Model 8

N

Name 7

Network Adapter 6

Non-Normalized Float Handling 7

O

Operating Mode 8

Optimization Method 6

Overlapped vs. None-overlapped addressing 18

Overview 4

P

Protocol 4

R

Redundancy 15

Replace with Zero 7

Request Timeout 10

Respect Tag-Specified Scan Rate 9

S

Scan Mode 9

Settings 12

Setup 4

Short 16

Simulated 8

T

Tag Counts 5, 8

Timing 9

U

Unable to bind to adapter: '<adapter name>' for device '<device name>'. Connection failed 21

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

Unmodified 7

W

Word 16

Write All Values for All Tags 6

Write Only Latest Value for All Tags 6

Write Only Latest Value for Non-Boolean Tags 6