SattBus Serial Driver

Table of Contents

SattBus Serial Driver 1

Table of Contents 2

SattBus Serial Driver 4

Overview 4

External Dependencies 4

Setup 4

Channel Properties — General 5

Channel Properties — Write Optimizations 6

Channel Properties — Advanced 7

Channel Properties — Interface Board 8

Device Properties — General 8

Operating Mode 9

Device Properties — Scan Mode 10

Device Properties — Timing 10

Device Properties — Auto-Demotion 11

Device Properties — Settings 12

Data Types Description 17

Address Descriptions 18

Error Descriptions 19

Error Codes 20

Missing address 20

Device address '<address>' contains a syntax error 20

Address '<address>' is out of range for the specified device or register 20

Device address '<address>' is not supported by model '<model name>' 21

Data Type '<type>' is not valid for device address '<address>' 21

Device address '<address>' is Read Only 21

Device '<device name>' is not responding 21

Unable to write to '<address>' on device '<device name>' 22

Unable to read '<number>' bytes starting at address '<address>' on device '<device name>'. <reason> 22

Unable to write to tag '<tag name>' on device '<device name>' <reason> 23

Failed to start Board '<channel name>'. Possible resource conflicts 23

Failed to connect to Board '<channel name>' 24

Failed to map memory for Board '<channel name>' 24

Failed to allocate Board '<channel name>' 24

Failed to allocate memory for Board '<channel name>' 24

Failed to stop Board '<channel name>' 25
Failed to initialize Board '<channel name>' 25
Failed to initialize Communications for Board '<channel name>' 25
Failed to get Version for Board '<channel name>' 25
Board '<channel name>' not present at location specified 25
Failed to set Communication Properties for Board '<channel name>' 26
Invalid XML document. Reason: Error loading 'Interface Board Configuration' for channel '<channel name>'. Memory segment value has to be in increments of 1000 (hex values) 26
Invalid XML document. Reason: Error loading 'InterfaceBoard Configuration' for channel '<channel name>'. Memory segment value has to be between '<start address>' and '<end address>' 26
Invalid XML document. Reason: Error loading 'Interface Board Configuration' for channel '<channel name>'. The Sattbus Node ID cannot be 1. Valid Node ID values are 0 and 2-127 26
Index **28**

SattBus Serial Driver

Help version 1.026

CONTENTS

Overview
What is the SattBus Serial Driver?

Interface Board Properties
How do I configure the interface board?

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

Data Types Description
What data types does this driver support?

Address Descriptions
How do I address a data location on a SattBus Serial Driver?

Error Descriptions
What error messages does the SattBus Serial Driver produce?

Overview

The SattBus Serial Driver provides a reliable way to connect SattBus Serial 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 interface.

The SattBus PC-board is used as the hardware interface for connecting PCs to a SattBus system. The SattBus PC-board is implemented as a standard I/O channel board. It is equipped with a dual port RAM chip (which is mapped into the host processor memory area).

External Dependencies

This driver has external dependencies. It requires the SattBus PC-board as the hardware interface for connecting PCs to a SattBus system.

For information on configuring the interface in the OPC server, refer to ***Interface Board Properties****.*

Setup

System Restrictions

Note: This driver does not work on 64-bit operating systems.

Supported Devices

SattCon 05
SattCon 15
SattCon 31
SattCon 35
SattCon 200

Satt Control OP45

Communication Protocol

SattBus

Channel and Device Limits

The maximum number of channels supported by this driver is 1. The maximum number of devices supported by this driver is 126 per channel.

Device IDs

This property specifies the unique ID that will be used to communicate with other devices. The valid range is from 2 to 127. The default setting is 2.

Request Timeout

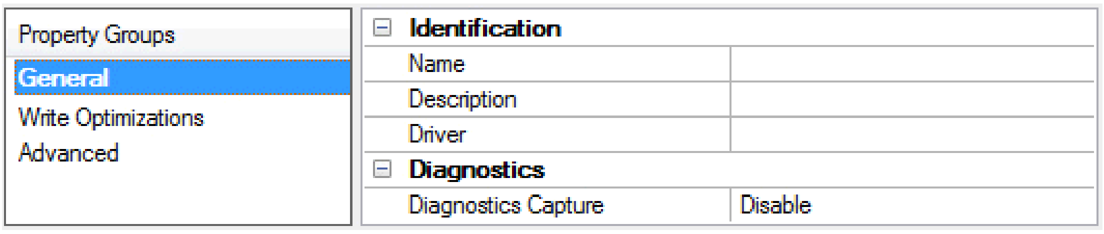
Specify the amount of time that the driver will wait on a response from the device before retrying or giving up and moving on to the next request. The valid range is from 2000 to 30000 ms.

Select Bit Ordering for Memory Cell (M, MW, XW)

This property is used to configure the bit ordering that will be used when reading or writing Word memory cell tags (M, MW and XW). The default setting is MSBit 7| 6| 5| 4| 3| 2| 1| 0| 15| 14| 13| 12| 11| 10| 9| 8 LSBit. For more information, refer to **Settings - Bit Ordering for Memory Cells**.

Channel Properties — General

This server supports the use of simultaneous multiple 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.



Identification	
Name	
Description	
Driver	
Diagnostics	
Diagnostics Capture	Disable

Identification

Name: 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: User-defined information about this channel.
● Many of these properties, including Description, have an associated system tag.

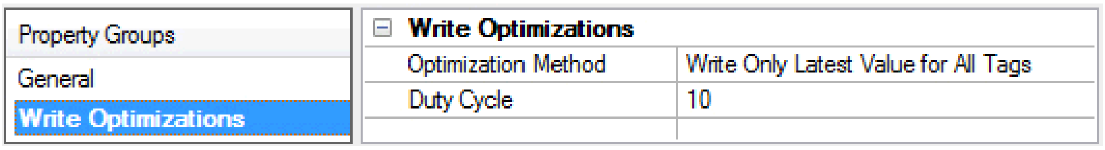
Driver: Selected protocol / driver for this channel. This property specifies the device driver that was selected during channel creation. It is a disabled setting in the channel properties. The property is required for creating a channel.
Note: With the server's online full-time operation, these properties 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. With this in mind, changes to the properties should not be made once a large client application has been developed. Utilize the User Manager to prevent operators from changing properties and restrict access rights to server features.

Diagnostics

Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications allows the usage of statistics tags that provide feedback to client applications regarding the operation of the channel. 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 does not support diagnostics.
For more information, refer to "Communication Diagnostics" and "Statistics Tags" in the server help.

Channel Properties — Write Optimizations

As with any server, writing data to the device may be the application's most important aspect. The server intends to ensure that the data written from the client application gets to the device on time. Given this goal, the server provides optimization properties that can be used 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

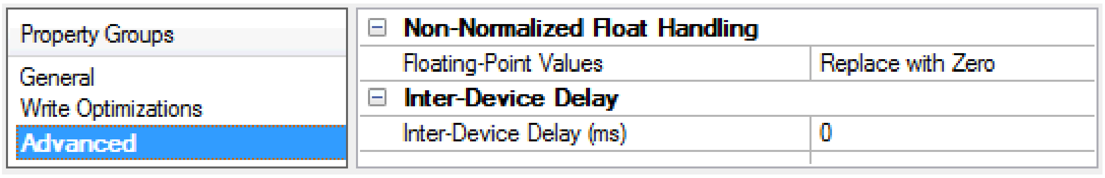
improve the application performance.
Note: This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.

- **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.



Floating-Point Values	Replace with Zero
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 not available 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.

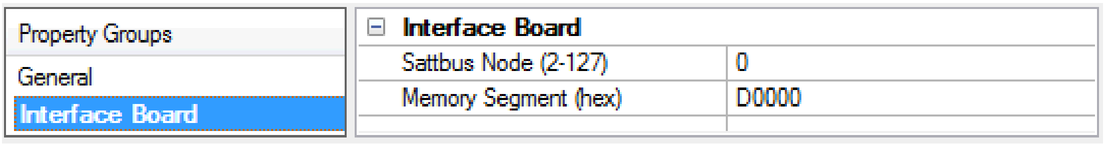
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

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.

Channel Properties — Interface Board

These channel properties must be set to match the SattBus Interface Board's configuration.



Property Groups
- General
- Interface Board

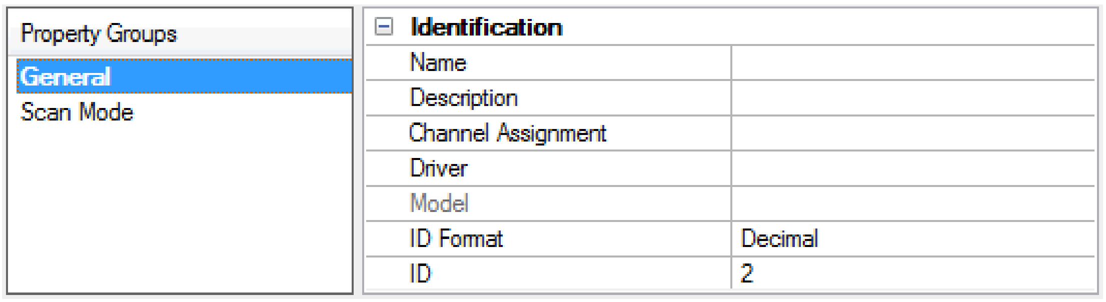
Interface Board	
Sattbus Node (2-127)	0
Memory Segment (hex)	D0000

SattBus Node: Specify the network address that will be assigned to the server channel. The valid range is 2 to 127. The default setting is 0.
Note: If 0 is specified, the network address configured with the interface board DIP switches will be used.

Memory Segment (hex): The dual port RAM of the interface board is mapped onto the host processor RAM area as a 2 KB memory segment. The start address of the dual port memory segment can be selected from a range between 0x00000 and 0xFF000 in increments of 0x1000. The valid range is 0xC0000 to 0xEF000. The default setting is D0000.
For information on system restrictions, refer to Setup.

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
- General
- Scan Mode

Identification	
Name	
Description	
Channel Assignment	
Driver	
Model	
ID Format	Decimal
ID	2

Identification

Name: This property specifies 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: User-defined information about this device.
Many of these properties, including Description, have an associated system tag.

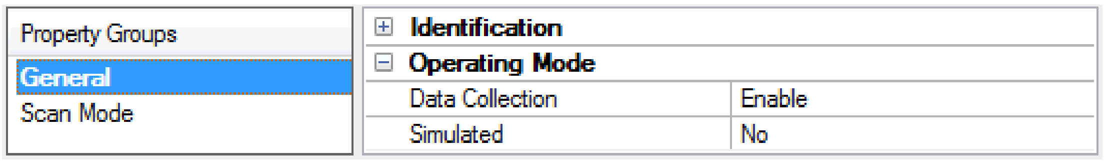
Channel Assignment: User-defined name of the channel to which this device currently belongs.

Driver: Selected protocol driver for this device.

Model: This property specifies the specific type of device that is associated with this ID. The contents of the drop-down menu depends 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 help documentation.

ID: This property specifies 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. *For more information, refer to the driver's help documentation.*

Operating Mode



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: This option places the device into 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 item's memory map is based on the group Update Rate. The data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.
Notes:

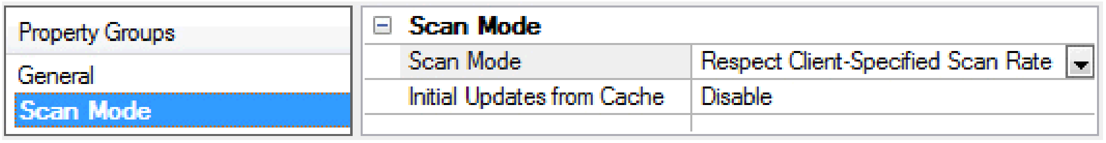
1. This System tag (_Simulated) is read only and cannot be written to for runtime protection. The System tag allows this property to be monitored from the client.

2. 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.

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

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.



Scan Mode	
Scan Mode	Respect Client-Specified Scan Rate
Initial Updates from Cache	Disable

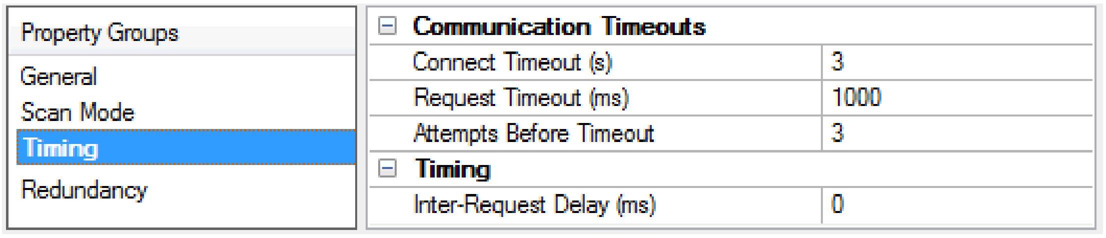
Scan Mode: Specifies how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

- **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 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.



Communication Timeouts	
Connect Timeout (s)	3
Request Timeout (ms)	1000
Attempts Before Timeout	3
Timing	
Inter-Request Delay (ms)	0

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: This property specifies 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 9,999,999 milliseconds (167.6667 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.

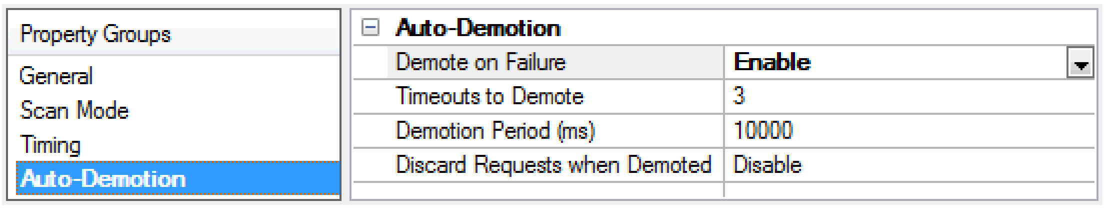
Attempts Before Timeout: This property specifies how many times the driver issues a communications request before considering the request to have failed and the device to be in error. The valid range is 1 to 10. The default is typically 3, but can vary depending on the driver's specific nature. The number 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: This property specifies how long the driver waits before sending the next request to the target device. 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 turnaround 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.

Device Properties — Auto-Demotion

The Auto-Demotion properties can temporarily place a device off-scan in the event that a device is not responding. By placing a non-responsive device offline for a specific time period, the driver can continue to optimize its communications with other devices on the same channel. After the time period has been reached, the driver re-attempts to communicate with the non-responsive device. If the device is responsive, the device is placed on-scan; otherwise, it restarts its off-scan time period.



Auto-Demotion	
Demote on Failure	Enable
Timeouts to Demote	3
Demotion Period (ms)	10000
Discard Requests when Demoted	Disable

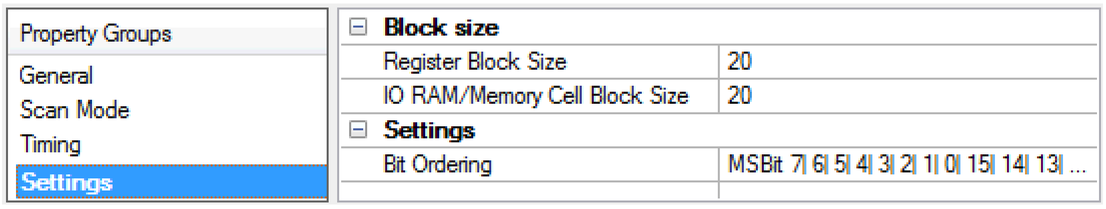
Demote on Failure: When enabled, the device is automatically taken off-scan until it is responding again.
Tip: Determine when a device is off-scan by monitoring its demoted state using the _AutoDemoted system tag.

Timeouts to Demote: Specify how many successive cycles of request timeouts and retries occur before the device is placed off-scan. The valid range is 1 to 30 successive failures. The default is 3.

Demotion Period: Indicate how long the device should be placed off-scan when the timeouts value is reached. During this period, no read requests are sent to the device and all data associated with the read requests are set to bad quality. When this period expires, the driver places the device on-scan and allows for another attempt at communications. The valid range is 100 to 3600000 milliseconds. The default is 10000 milliseconds.

Discard Requests when Demoted: Select whether or not write requests should be attempted during the off-scan period. Disable to always send write requests regardless of the demotion period. Enable to discard writes; the server automatically fails any write request received from a client and does not post a message to the Event Log.

Device Properties — Settings



Block size	
Register Block Size	20
IO RAM/Memory Cell Block Size	20
Settings	
Bit Ordering	MSBit 7| 6| 5| 4| 3| 2| 1| 0| 15| 14| 13| ...

Block Size

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

Settings

Bit Ordering: Select the bit ordering for 16-bit memory cell tags to be used when reading or writing Word memory cell tags (M, MW and XW). The three options are described below. The default setting is MSBit 7| 6| 5| 4| 3| 2| 1| 0| 15| 14| 13| 12| 11| 10| 9| 8 LSBit.

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: The 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. The following examples will illustrate this condition for reads and writes:

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'.

<------------------------- 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, the rest are OFF.

<------------------------- 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**

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'.

Note: Word access at non-byte boundaries is not allowed in the DOX10 programing 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, the rest are OFF.

<------------------------- 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**

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. The following examples will illustrate this condition for reads and writes.

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'.

<------------------------- **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, the rest are OFF.

<------------------------- **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**

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'.

   Note: Word access at non-byte boundaries is not allowed in the DOX10 programing 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, the rest are OFF.

<------------------------- **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**

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 is similar to the second selection described above, but with the bytes swapped. It is the default condition. The following examples will illustrate this condition for reads and writes.

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'.

<------------------------- 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, the rest are OFF.

<------------------------- 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**

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'.

Note: Word access at non-byte boundaries is not allowed in the DOX10 programing 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, the rest are OFF.

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

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.

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 bit 0 is the low bit bit 15 is the high bit
Short	Signed 16-bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
DWord	Unsigned 32-bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit

Address Descriptions

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

Address Type	Range	Data Type	Access
Register*	R0-R32767	**Word**, Short	Read/Write
	R0.00-R32767.00 ... R0.15-R32767.15	**Boolean**	Read Only
	RW0-RW32767	**Word**, Short	Read/Write
	RW0.00-RW32767.00 ... RW0.15-RW32767.15	**Boolean**	Read Only
	RD0-RD32766	**DWord**, Long	Read/Write
I/O RAM bits**	0-77777	**Boolean**	Read/Write
	IO0-IO77777	**Boolean**	
	I0-I77777	**Boolean**	
	O0-O77777	**Boolean**	
	X0-X77777	**Boolean**	
I/O RAM***	XB0-XB77770	**Byte**, Char	Read/Write
	XW0.00-XW77760.00 ... XW0.15-XW77760.15	**Boolean**	
	XW0-XW77760	**Word**, Short	
Memory Cell in I/O RAM****	M0-M77760	**Word**, Short	Read/Write
	M0.00-M77760.00 ... M0.15-M77760.15	**Boolean**	
	MW0-MW77760	**Word**, Short	
	MW0.00-MW77760.00 ... MW0.15-MW77760.15	**Boolean**	

* Addresses are in decimal.
** Addresses are in octal.
*** Addresses are in octal, bit numbers are in decimal. Addresses can be on a non-byte boundary (non-multiples of 10 octal).
**** Bit numbers are in decimal.

Error Descriptions

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

Address Validation

Missing address
Device address '<address>' contains a syntax error
Address '<address>' is out of range for the specified device or register
Device address '<address>' is not supported by model '<model name>'
Data Type '<type>' is not valid for device address '<address>'
Device address '<address>' is Read Only

Device Status Messages

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

Device Specific Messages

Unable to read '<number>' bytes starting at address '<address>' on device '<device name>'. <reason>
Unable to write to tag '<tag name>' on device '<device name>' <reason>
Failed to start Board '<channel name>'. Possible resource conflicts
Failed to connect to Board '<channel name>'
Failed to map memory for Board '<channel name>'
Failed to allocate Board '<channel name>'
Failed to allocate memory for Board '<channel name>'
Failed to stop Board '<channel name>'
Failed to initialize Board '<channel name>'
Failed to initialize Communications for Board '<channel name>'
Failed to get Version for Board '<channel name>'
Board '<channel name>' not present at location specified
Failed to set Communication Properties for Board '<channel name>'
Invalid XML document. Reason: Error loading 'Interface Board Configuration' for channel '<channel name>'. Memory segment value has to be between '<start address>' and '<end address>'
Invalid XML document. Reason: Error loading 'Interface Board Configuration' for channel '<channel name>'. Memory segment value has to be in increments of 1000 (hex values)
Invalid XML document. Reason: Error loading 'Interface Board Configuration' for channel '<channel name>'. The Sattbus Node ID cannot be 1. Valid Node ID values are 0 and 2-127

See Also:
Error Codes

Error Codes

Error Code	Description
0	No error
1	General error
2	Checksum error
3	Limit error
4	Message length error
5	Illegal function
6	Illegal address
7	Illegal type
8	Illegal operation
10	Queue full
11	Queue ready
17	Function occupied.
18	No such variable.
19	Buffer error.
20	Module restart.
21	Device not responding.
22	Temporarily inoperable.

Missing address

Error Type:

Warning

Possible Cause:

A tag address that has been specified dynamically 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 dynamically 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 dynamically references a location that is beyond the range of supported locations for the device.

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

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

Error Type:
Warning

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

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

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

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically 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 dynamically 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.

Device '<device name>' is not responding

Error Type:
Serious

Possible Cause:

1. The SattBus Serial connection between the device and the Host PC is broken.
2. The SattBus Interface board properties are incorrect.
3. The SattBus board may not be configured correctly.
4. 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 PLC device.
2. Verify that the specified properties match those of the SattBus card and device.
3. Verify the configuration of the SattBus card.
4. Increase the Request Timeout property so that the entire response can be handled.

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

Error Type:

Serious

Possible Cause:

1. The SattBus Serial connection between the device and the Host PC is broken.
2. The SattBus Interface board properties are incorrect.
3. The SattBus board may not be configured correctly.

Solution:

1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified properties match those of the SattBus card and device.
3. Verify the configuration of the SattBus card.

Unable to read '<number>' bytes starting at address '<address>' on device '<device name>'. <reason>

Error Type:

Serious

Possible Cause:

The driver was unable to complete a read operation because of the reason specified.

Solution:

Refer to the reasons below.

Reasons:

1. The frame returned contains errors.
2. The board response timed-out.
3. The board returned incomplete data.
4. An unsupported message was used.
5. The board returned error code '<code>'.
6. An unexpected error occurred.
7. The device is busy.
8. No buffers were available to transmit request.

See Also:
Error Codes

Unable to write to tag '<tag name>' on device '<device name>' <reason>

Error Type:
Serious

Possible Cause:
The driver was unable to complete a write operation because of the reason specified.

Solution:
Refer to the reasons below.

Reasons:

1. The frame returned contains errors.
2. The board response timed-out.
3. The board returned incomplete data.
4. An unsupported message was used.
5. The board returned error code '<code>'.
6. An unexpected error occurred.
7. The device is busy.
8. No buffers were available to transmit request.

See Also:
Error Codes

Failed to start Board '<channel name>'. Possible resource conflicts

Error Type:
Serious

Possible Cause:
The driver was unable to start the SattBus interface board.

Solution:
Check for conflicting devices installed on the computer.

Failed to connect to Board '<channel name>'

Error Type:
Serious

Possible Cause:
The driver was unable to connect to the SattBus interface board.

Solution:
Verify that the interface board is configured and installed correctly.

Failed to map memory for Board '<channel name>'

Error Type:
Warning

Possible Cause:
The SattBus kernel mode interface failed to provide a card memory mapping to the driver.

Solution:
Verify that the interface board is configured and installed correctly.

Failed to allocate Board '<channel name>'

Error Type:
Warning

Possible Cause:
Resources could not be allocated for kernel mode interface to the SattBus board.

Solution:
Shut down any unused applications and restart the server.

Failed to allocate memory for Board '<channel name>'

Error Type:
Warning

Possible Cause:
Resources could not be allocated for required memory map of SattBus card.

Solution:
Shut down any unused applications and restart the server.

Failed to stop Board '<channel name>'

Error Type:
Serious

Possible Cause:
The driver was unable to stop the SattBus interface board.

Solution:
Verify that the interface board is configured and installed correctly.

Failed to initialize Board '<channel name>'

Error Type:
Serious

Possible Cause:
The driver was unable to initialize the SattBus interface board.

Solution:
Verify that the interface board is configured and installed correctly.

Failed to initialize Communications for Board '<channel name>'

Error Type:
Serious

Possible Cause:
The driver was unable to initialize communications for the SattBus interface board.

Solution:
Verify that the interface board is configured and installed correctly.

Failed to get Version for Board '<channel name>'

Error Type:
Serious

Possible Cause:
The driver was unable get version information from the SattBus interface board.

Solution:
Verify that the interface board is configured and installed correctly.

Board '<channel name>' not present at location specified

Error Type:
Serious

Possible Cause:
The SattBus interface card may not be configured or installed correctly.

Solution:
Verify that the interface board is configured and installed correctly.

Failed to set Communication Properties for Board '<channel name>'

Error Type:
Serious

Possible Cause:
The driver was unable to set communications properties for the SattBus interface board.

Solution:
Verify that the interface board is configured and installed correctly.

Invalid XML document. Reason: Error loading 'Interface Board Configuration' for channel '<channel name>'. Memory segment value has to be in increments of 1000 (hex values)

Error Type:
Serious

Possible Cause:
The memory segment address is not a value that is divisible by 0x1000.

Solution:
Make sure the memory segment address is divisible by 0x1000 (i.e last three digits are zeros).

Invalid XML document. Reason: Error loading 'InterfaceBoard Configuration' for channel '<channel name>'. Memory segment value has to be between '<start address>' and '<end address>'

Error Type:
Serious

Possible Cause:
The memory segment address is outside the driver supported range of 0xC000-0xEF000.

Solution:
Make sure the memory segment address is in the range 0xC000-0xEF000.

See Also:
Interface Board Properties

Invalid XML document. Reason: Error loading 'Interface Board Configuration' for channel '<channel name>'. The Sattbus Node ID cannot be 1. Valid Node ID values are 0 and 2-127

Error Type:

Serious

Possible Cause:

The Sattbus Node ID is set to 1.

Solution:

Make sure the Sattbus Node ID is either 0 or 2-127.

See Also:

Interface Board Properties

Index

6

64-bit operating systems 4

A

Address '<address>' is out of range for the specified device or register 20

Address Descriptions 18

Attempts Before Timeout 11

Auto-Demotion 11

B

Block Size 12

Board '<channel name>' not present at location specified 25

Boolean 17

C

Channel Assignment 8

Communications Timeouts 10-11

Connect Timeout 11

D

Data Collection 9

Data Type '<type>' is not valid for device address '<address>' 21

Data Types Description 17

Demote on Failure 11

Demotion Period 12

Device '<device name> is not responding 21

Device address '<address>' contains a syntax error 20

Device address '<address>' is not supported by model '<model name>' 21

Device address '<address>' is Read Only 21

Device ID 5

Device Properties — Settings 12

Discard Requests when Demoted 12

Do Not Scan, Demand Poll Only 10

Driver 8

DWord 17

E

Error Codes 20

Error Descriptions 19

External Dependencies 4

F

Failed to allocate Board '<channel name>' 24

Failed to allocate memory for Board '<channel name>' 24

Failed to connect to Board '<channel name>' 24

Failed to get Version for Board '<channel name>' 25

Failed to initialize Board '<channel name>' 25

Failed to initialize Communications for Board '<channel name>' 25

Failed to map memory for Board '<channel name>' 24

Failed to set Communication Properties for Board '<channel name>' 26

Failed to start Board '<channel name>'. Possible resource conflicts 23

Failed to stop Board '<channel name>' 25

G

General 8

I

ID 9

Identification 8

Initial Updates from Cache 10

Inter-Request Delay 11

Interface Board 8

Invalid XML document. Reason: Error loading for channel. Memory segment value has to be between '<start>' and '<end> 26

Invalid XML document. Reason: Error loading for channel. Memory segment value has to be in increments of 1000 (hex values) 26

Invalid XML document. Reason: Error loading for channel. The Sattbus node ID cannot be 1 26

L

Long 17

M

Memory Cell 5

Missing address 20

Model 9

N

Name 8

O

Operating Mode 9

Overview 4

P

Protocol 5

R

Request Timeout 11

Respect Tag-Specified Scan Rate 10

S

Scan Mode 10

Setup 4

Short 17

Simulated 9

T

Timeout 5

Timeouts to Demote 12

U

Unable to read '<number>' bytes starting at address '<address>' on device '<device name>'. <reason> 22

Unable to write tag '<address>' on device '<device name>' 22

Unable to write to tag '<tag name>' on device '<device name>' <reason> 23

W

Word 17