Siemens S7-200 Driver

Table of Contents

Siemens S7-200 Driver

Help version 1.042

CONTENTS

Overview

What is the Siemens S7-200 Driver?

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 Siemens S7-200 device?

Event Log Messages

What messages does the Siemens S7-200 Driver produce?

Overview

The Siemens S7-200 Driver provides a reliable way to connect Siemens S7-200 devices to OPC Client applications, including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with Siemens S7-200 devices, and supports a 10 or 11-bit setting for the PPI programming cable. When using the 10-bit mode (specifically, the EM 241 Modem Module), the S7-200 PPM mode should be selected. When using the 11-bit mode, the S7-200 model should be selected.

Setup

Channel and Device Limits

The maximum number of channels supported by this driver is 256. The maximum number of devices supported by this driver is 127 per channel.

Supported Communication Parameters
Baud Rate: 9600 or 19200
Parity: Even (11-bit mode) and None (10-bit PPM mode)
Data Bits: 8
Stop Bits: 1
● **Note:** Not all devices support the listed configurations.

Ethernet Encapsulation

This driver supports Ethernet Encapsulation, which allows the driver to communicate with serial devices attached to an Ethernet network using a terminal or device server. It may be invoked through the COM ID in Channel Properties. *For more information, refer to the main server's help documentation.*

Communication Protocols

Point-to-Point (PPI) S7-200 Communications Protocol (11-bit mode).
Point-to-Point Modem (PPM) S7-200 Communications Protocol (10-bit mode).

The Siemens S7-200 Driver normally operates using the standard 11-bit PPI protocol. If the EM 241 modem module is required, the S7-200 PPM model must be selected. This model allows the driver to operate in a 10-bit mode that is compatible with many off-the-shelf modems. The 10-bit PPM mode can also be used directly on the PLC's programming port. To enable 10-bit PPM mode, set the S7-200 programming cable to 10-bit mode.

● *See Also:Device Properties*

Flow Control

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

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

● *See Also:Channel Communication Properties*

Supported Devices

Siemens S7-200 devices

Supported Cables

A special cable is required to communicate with the S7-200 PLC. The cable recommended by the manufacturer should be used.

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	⊟ **Identification**	
General	Name	
Write Optimizations	Description	
Advanced	Driver	
	⊟ **Diagnostics**	
	Diagnostics Capture	Disable
	⊟ **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 does not support diagnostics.
 ◆ *For more information, refer to "Communication Diagnostics" and "Statistics Tags" in the 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 — Serial Communications

Serial communication properties are available to serial drivers and vary depending on the driver, connection type, and options selected. Below is a superset of the possible properties.
Click to jump to one of the sections: **Connection Type**, **Serial Port Settings** or **Ethernet Settings**, and **Operational Behavior**.

Notes:

- With the server's online full-time operation, these properties can be changed at any time. Utilize proper user role and privilege management to prevent operators from changing properties or accessing server features.

- Users must define the specific communication parameters to be used. Depending on the driver, channels may or may not be able to share identical communication parameters. Only one shared serial connection can be configured for a Virtual Network (see **Channel Properties — Serial Communications**).

Property Groups	□ **Connection Type**	
General	Physical Medium	COM Port
Serial Communications	□ **Serial Port Settings**	
Write Optimizations	COM ID	39
Advanced	Baud Rate	19200
	Data Bits	8
	Parity	None
	Stop Bits	1
	Flow Control	RTS Always
	□ **Operational Behavior**	
	Report Communication Errors	Enable
	Close Idle Connection	Enable
	Idle Time to Close (s)	15

Connection Type

Physical Medium: Choose the type of hardware device for data communications. Options include Modem, Ethernet Encapsulation, COM Port, and None. The default is COM Port.

1. **None**: Select None to indicate there is no physical connection, which displays the **Operation with no Communications** section.

2. **COM Port**: Select Com Port to display and configure the **Serial Port Settings** section.

3. **Modem**: Select Modem if phone lines are used for communications, which are configured in the **Modem Settings** section.

4. **Ethernet Encap.**: Select if Ethernet Encapsulation is used for communications, which displays the **Ethernet Settings** section.

5. **Shared**: Verify the connection is correctly identified as sharing the current configuration with another channel. This is a read-only property.

Serial Port Settings

COM ID: Specify the Communications ID to be used when communicating with devices assigned to the channel. The valid range is 1 to 9991 to 16. The default is 1.

Baud Rate: Specify the baud rate to be used to configure the selected communications port.

Data Bits: Specify the number of data bits per data word. Options include 5, 6, 7, or 8.

Parity: Specify the type of parity for the data. Options include Odd, Even, or None.

Stop Bits: Specify the number of stop bits per data word. Options include 1 or 2.

Flow Control: Select how the RTS and DTR control lines are utilized. Flow control is required to communicate with some serial devices. Options are:

- **None**: This option does not toggle or assert control lines.
- **DTR**: This option asserts the DTR line when the communications port is opened and remains on.
- **RTS**: This option specifies that the RTS line is high if bytes are available for transmission. After all buffered bytes have been sent, the RTS line is low. This is normally used with RS232/RS485 converter hardware.
- **RTS, DTR**: This option is a combination of DTR and RTS.
- **RTS Always**: This option asserts the RTS line when the communication port is opened and remains on.
- **RTS Manual**: This option asserts the RTS line based on the timing properties entered for RTS Line Control. It is only available when the driver supports manual RTS line control (or when the properties are shared and at least one of the channels belongs to a driver that provides this support).
  RTS Manual adds an **RTS Line Control** property with options as follows:
    - **Raise**: Specify the amount of time that the RTS line is raised prior to data transmission. The valid range is 0 to 9999 milliseconds. The default is 10 milliseconds.
    - **Drop**: Specify the amount of time that the RTS line remains high after data transmission. The valid range is 0 to 9999 milliseconds. The default is 10 milliseconds.
    - **Poll Delay**: Specify the amount of time that polling for communications is delayed. The valid range is 0 to 9999. The default is 10 milliseconds.

Tip: When using two-wire RS-485, "echoes" may occur on the communication lines. Since this communication does not support echo suppression, it is recommended that echoes be disabled or a RS-485 converter be used.

Operational Behavior

- **Report Communication Errors**: Enable or disable reporting of low-level communications errors. When enabled, low-level errors are posted to the Event Log as they occur. When disabled, these same errors are not posted even though normal request failures are. The default is Enable.
- **Close Idle Connection**: Choose to close the connection when there are no longer any tags being referenced by a client on the channel. The default is Enable.
- **Idle Time to Close**: Specify the amount of time that the server waits once all tags have been removed before closing the COM port. The default is 15 seconds.

Ethernet Settings

Note: Not all serial drivers support Ethernet Encapsulation. If this group does not appear, the functionality is not supported.

Ethernet Encapsulation provides communication with serial devices connected to terminal servers on the Ethernet network. A terminal server is essentially a virtual serial port that converts TCP/IP messages on the Ethernet network to serial data. Once the message has been converted, users can connect standard devices that support serial communications to the terminal server. The terminal server's serial port must be properly configured to match the requirements of the serial device to which it is attached. *For more information, refer to "Using Ethernet Encapsulation" in the server help.*

- **Network Adapter**: Indicate a network adapter to bind for Ethernet devices in this channel. Choose a network adapter to bind to or allow the OS to select the default.
  Specific drivers may display additional Ethernet Encapsulation properties. For more information, refer to **Channel Properties — Ethernet Encapsulation***.*

Modem Settings

- **Modem**: Specify the installed modem to be used for communications.
- **Connect Timeout**: Specify the amount of time to wait for connections to be established before failing a read or write. The default is 60 seconds.
- **Modem Properties**: Configure the modem hardware. When clicked, it opens vendor-specific modem properties.
- **Auto-Dial**: Enables the automatic dialing of entries in the Phonebook. The default is Disable. *For more information, refer to "Modem Auto-Dial" in the server help.*
- **Report Communication Errors**: Enable or disable reporting of low-level communications errors. When enabled, low-level errors are posted to the Event Log as they occur. When disabled, these same errors are not posted even though normal request failures are. The default is Enable.
- **Close Idle Connection**: Choose to close the modem connection when there are no longer any tags being referenced by a client on the channel. The default is Enable.
- **Idle Time to Close**: Specify the amount of time that the server waits once all tags have been removed before closing the modem connection. The default is 15 seconds.

Operation with no Communications

- **Read Processing**: Select the action to be taken when an explicit device read is requested. Options include Ignore and Fail. Ignore does nothing; Fail provides the client with an update that indicates failure. The default setting is Ignore.

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.

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.

Channel Properties — Communication Serialization

The server's multi-threading architecture allows channels to communicate with devices in parallel. Although this is efficient, communication can be serialized in cases with physical network restrictions (such as Ethernet radios). Communication serialization limits communication to one channel at a time within a virtual network.

The term "virtual network" describes a collection of channels and associated devices that use the same pipeline for communications. For example, the pipeline of an Ethernet radio is the client radio. All channels using the same client radio associate with the same virtual network. Channels are allowed to communicate each in turn, in a "round-robin" manner. By default, a channel can process one transaction before handing communications off to another channel. A transaction can include one or more tags. If the controlling channel contains a device that is not responding to a request, the channel cannot release control until the transaction times out. This results in data update delays for the other channels in the virtual network.

Property Groups	Channel-Level Settings	
General	Virtual Network	None
Serial Communications	Transactions per Cycle	1
Communication Serialization	Global Settings	
	Network Mode	Load Balanced

Channel-Level Settings

Virtual Network: Specify the channel's mode of communication serialization. Options include None and Network 1 - Network 500. The default is None. Descriptions of the options are as follows:

- **None**: This option disables communication serialization for the channel.
- **Network 1 - Network 500**: This option specifies the virtual network to which the channel is assigned.

Transactions per Cycle: Specify the number of single blocked/non-blocked read/write transactions that can occur on the channel. When a channel is given the opportunity to communicate, this is the number of transactions attempted. The valid range is 1 to 99. The default is 1.

Global Settings

Network Mode: This property is used to control how channel communication is delegated. In **Load Balanced** mode, each channel is given the opportunity to communicate in turn, one at a time. In **Priority** mode,

channels are given the opportunity to communicate according to the following rules (highest to lowest priority):

1. Channels with pending writes have the highest priority.

2. Channels with pending explicit reads (through internal plug-ins or external client interfaces) are prioritized based on the read's priority.

3. Scanned reads and other periodic events (driver specific).

The default is Load Balanced and affects *all* virtual networks and channels.
🔶 Devices that rely on unsolicited responses should not be placed in a virtual network. In situations where communications must be serialized, it is recommended that Auto-Demotion be enabled.

Due to differences in the way that drivers read and write data (such as in single, blocked, or non-blocked transactions); the application's Transactions per cycle property may need to be adjusted. When doing so, consider the following factors:

- How many tags must be read from each channel?
- How often is data written to each channel?
- Is the channel using a serial or Ethernet driver?
- Does the driver read tags in separate requests, or are multiple tags read in a block?
- Have the device's Timing properties (such as Request timeout and Fail after *x* successive timeouts) been optimized for the virtual network's communication medium?

Channel Properties — Siemens Client ID

Siemens Client ID: Specify the node number used by the Siemens S7-200 Driver on the network. Each channel must have a unique Siemens client ID. The valid range is 0 to 126.

Device Properties — General

Property Groups	☐ **Identification**	
General	Name	Siemens S7-200
Scan Mode	Description	
Timing	Channel Assignment	Siemens S7-200
Auto-Demotion	Driver	Siemens S7-200
Redundancy	Model	S7-200
	ID Format	Decimal
	ID	1
	☐ **Operating Mode**	
	Data Collection	Enable
	Simulated	No

Identification

Name: User-defined identity of this device.

Description: User-defined information about this device.

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

Driver: Selected protocol driver for this device.

Model: Select the specific version of the device.

ID Format: Select how the device identity is formatted. Options include Decimal, Octal, and Hex.

ID: the unique identity of the device for communication with the driver. The valid range is 0 to 126. Any devices defined under this channel should not use an ID that conflicts with the Siemens client ID.

Operating Mode

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.

Property Groups	Scan Mode	
General	Scan Mode	Respect Client-Specified Scan Rate ▼
Scan Mode	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	Communication Timeouts	
General	Connect Timeout (s)	3
Scan Mode	Request Timeout (ms)	1000
Timing	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 9999 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. 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.
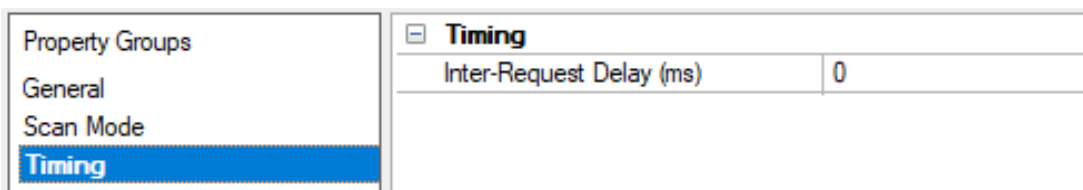
Device Properties — Auto-Demotion

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.

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 — Redundancy

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 of a 16-bit value*
Byte	Unsigned 8-bit value bit 0 is the low bit bit 7 is the high bit
Word	Unsigned 16-bit value bit 0 is the low bit bit 15 is the high bit
Short	Signed 16-bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
DWord	Unsigned 32-bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
Float	32-bit floating point value The driver interprets two consecutive registers as a floating-point value by making the second register the high word and the first register the low word.
String	Null-terminated ASCII string

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

Address Descriptions

Address specifications vary depending on the model in use. Select a link from the list below to obtain specific address information for the model of interest.

S7-200 Addressing

S7-200 PPM Addressing

S7-200 Addressing

The S7-200 addressing format is the same as the S7-200 PPM addressing format. The model selection in this case determines whether the driver is using PPI protocol (normal S7-200 Mode) or PPM (S7-200 in Point to Point Modem) mode. In both cases, the addressing is the same.

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

Address Type	Range	Type	Access
Discrete Inputs	I00000-I65535 I00000-I65534 I00000-I65532 I00000.bb-I65535.bb I00000.bb-I65534.bb I00000.bb-I65532.bb	Byte **Word**, Short DWord, Long, Float **Byte** Boolean, **Word**, Short **DWord**, Long	Read/Write
Discrete Outputs	Q00000-Q65535 Q00000-Q65534 Q00000-Q65532 Q00000.bb-Q65535.bb Q00000.bb-Q65534.bb Q00000.bb-Q65532.bb	Byte **Word**, Short DWord, Long, Float **Byte** Boolean, **Word**, Short **DWord**, Long	Read/Write
Internal Memory	M00000-M65535 M00000-M65534 M00000-M65532 M00000.bb-M65535.bb M00000.bb-M65534.bb M00000.bb-M65532.bb	Byte, **Word**, Short DWord, Long, Float **Byte** Boolean, **Word**, Short **DWord**, Long	Read/Write
Special Memory	SM00000-SM65535 SM00000-SM65534 SM00000-SM65532 SM00000.bb- SM65535.bb SM00000.bb- SM65534.bb SM00000.bb- SM65532.bb	Byte **Word**, Short DWord, Long, Float **Byte** Boolean, **Word**, Short **DWord**, Long	Read/Write SM0-SM29 are Read Only
Variable Memory	V00000-V65535 V00000-V65534	Byte, **Word**, Short	Read/Write

Address Type	Range	Type	Access
	V00000-V65532 V00000.bb-V65535.bb V00000.bb-V65534.bb V00000.bb-V65532.bb	DWord, Long, Float, String **Byte** Boolean, **Word**, Short **DWord**, Long, String	
Timer Current Values	T00000-T65535	**DWord**, Long	Read/Write
Timer Status Bits	T00000-T65535	Boolean*	Read Only
Counter Current Values	C00000-C65535	**Word**, Short	Read/Write
Counter Status Bits	C00000-C65535	Boolean*	Read Only
High Speed Counters	HC00000-HC65535	**DWord**, Long	Read Only
Analog Inputs	AI00000-AI65534**	**Word**, Short	Read Only
Analog Outputs	AQ00000-AQ65534**	**Word**, Short	Write Only

* For Timer and Counter status bits, dot bit notation is not used. The status bit for timer 7 would be T7 declared as Boolean.

** For Analog Inputs and Outputs, the address must be even (AI0, AI2, AI4...). Analog Outputs (AQ) are Write Only: there is no method for reading the value of Analog Outputs from the device. Write-only data types of this driver return the last value written when read if an initial write to device has completed. If an initial write has not completed, the driver returns a value of 0 when read. This only applies while a client is connected to the server.

The actual number of addresses of each type depends on the Siemens S7-200 device in use. Each type does not necessarily support an address of 0 to 65535. For address ranges, refer to the device's documentation.

Optional Dot Bits

For Byte, Word, Short, DWord or Long data types, an optional .bb (dot bit) can be appended to the address to reference a bit in a particular value. The valid ranges for the optional bit is 0-7 for Byte types; 0-15 for Word, Short, and Boolean types; 0-31 for DWord and Long types; and 1-211 for String types. Float types do not support bit operations. Boolean and String types require a bit number. The bit number for String types specifies the number of characters in the string.

Dynamic addresses with bit numbers in the range of 0-7 default to Byte; 8-15 default to Word; 16-31 defaults to DWord. V Memory addresses with a bit number larger than 31 defaults to String. The following diagram illustrates how the driver maps bits within the controller.
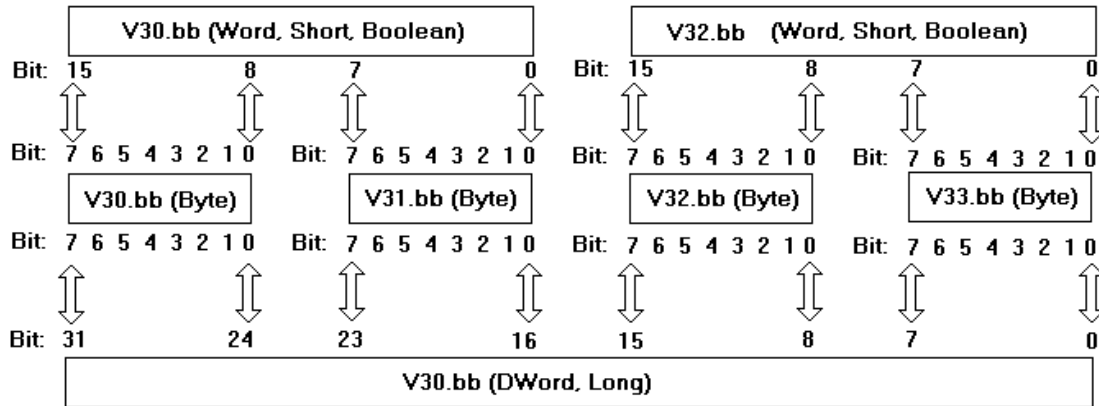
● **Note**: V30.10@bool, V30.2@byte, and V30.26@DWord all reference the same bit in the controller.

Arrays

Certain memory types (I, Q, M, SM, V, AI, and AQ) support an array operation. Boolean arrays are not allowed at this time. To specify an array address, append *[rows][cols]* to the end of an address. If only [cols] is specified, [rows] defaults to 1. With the array type, it is possible to read and write a block of 200 bytes at one time.

The maximum array size for Word and Short types is 100, and for DWord, Long and Float types is 50. The array size is determined by the multiplication of rows and cols.

● **Note**: The maximum array size also depends on the maximum block size of the device being used.

Examples

1. To read and write an array of 10 Variable Memory Float values starting with V10, declare an address as follows: V10 [1][10]. Choose Float for the data type.

● **Note**: This array reads and writes values to registers V10, V14, V18, V22 ... V46.

2. To read and write to bit 23 of Internal Memory Long M20, declare an address as follows: M20.23. Choose Long for the data type.

Strings

The driver allows for variable length strings to be stored in Variable Memory locations. The bit number specifies the string length (1-211) in characters. String data that is sent to the device, but is smaller in length than the string character count (bit number), is null terminated. String data that meets or exceeds the character length is truncated to the character count and sent to the device without a null terminator.

To read and write a string starting at V5 for a length of 10 characters, declare an address as follows: V5.10. Choose string for the data type.

● **Notes**:

1. V Memory locations V5-V14 would be used to store this 10 character string.

2. Not all devices support up to 211 character requests in a single transaction. To determine the maximum number of characters that can be requested in a transaction, consult the device's documentation. This value is the largest string the driver can Read/Write to and from the device.

🔶 **Caution**: When modifying Word, Short, DWord, Long and Float types remember that each address starts at a byte offset within the device. Therefore, Words V0 and V1 overlap at byte 1. Writing to V0 modifies the value held in V1. Similarly, DWord, Long, and Float types can also overlap. It is recommended that these memory types be used so that overlapping does not occur. As an example, when using DWords, use V0, V4, V8, and so on to prevent overlapping bytes.

S7-200 PPM Addressing

The S7-200 PPM addressing format is the same as the S7-200 addressing format. The model selection in this case determines whether the driver is using PPI protocol (normal S7-200 Mode) or PPM (S7-200 in Point to Point Modem) mode. In both cases, the addressing is the same. PPM mode is used when the target PLC is connected via the EM241 Modem module or via the programming port running in 10-bit mode.

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

Address Type	Range	Type	Access
Discrete Inputs	I00000-I65535 I00000-I65534 I00000-I65532 I00000.bb-I65535.bb I00000.bb-I65534.bb I00000.bb-I65532.bb	Byte **Word**, Short DWord, Long, Float **Byte** Boolean, **Word**, Short **DWord**, Long	Read/Write
Discrete Outputs	Q00000-Q65535 Q00000-Q65534 Q00000-Q65532 Q00000.bb-Q65535.bb Q00000.bb-Q65534.bb Q00000.bb-Q65532.bb	Byte **Word**, Short DWord, Long, Float **Byte** Boolean, **Word**, Short **DWord**, Long	Read/Write
Internal Memory	M00000-M65535 M00000-M65534 M00000-M65532 M00000.bb-M65535.bb M00000.bb-M65534.bb M00000.bb-M65532.bb	Byte **Word**, Short DWord, Long, Float **Byte** Boolean, **Word**, Short **DWord**, Long	Read/Write
Special Memory	SM00000-SM65535 SM00000-SM65534 SM00000-SM65532 SM00000.bb- SM65535.bb SM00000.bb- SM65534.bb SM00000.bb- SM65532.bb	Byte **Word**, Short DWord, Long, Float **Byte** Boolean, **Word**, Short **DWord**, Long	Read/Write SM0-SM29 are Read Only
Variable Memory	V00000-V65535 V00000-V65534 V00000-V65532	Byte **Word**, Short DWord, Long, Float,	Read/Write

Address Type	Range	Type	Access
	V00000.bb-V65535.bb V00000.bb-V65534.bb V00000.bb-V65532.bb	String **Byte** Boolean, **Word**, Short **DWord**, Long, String	
Timer Current Values	T00000-T65535	**DWord**, Long	Read/Write
Timer Status Bits	T00000-T65535	Boolean*	Read Only
Counter Current Values	C00000-C65535	**Word**, Short	Read/Write
Counter Status Bits	C00000-C65535	Boolean*	Read Only
High Speed Counters	HC00000-HC65535	**DWord**, Long	Read Only
Analog Inputs	AI00000-AI65534***	**Word**, Short	Read Only
Analog Outputs	AQ00000-AQ65534***	**Word**, Short	Write Only

* For Timer and Counter status bits, dot bit notation is not used. The status bit for timer 7 would be "T7" declared as Boolean.

** For Analog Inputs and Outputs the address must be even (AI0, AI2, AI4...). Analog Outputs (AQ) are Write Only: there is no method for reading the value of Analog Outputs from the device. Write Only types in this driver returns the last value written when read if an initial write to device has completed. If an initial write has not completed, the driver returns a value of 0 when read. This only applies while a client is connected to the server.

The actual number of addresses of each type depends on the Siemens S7-200 device in use. Each type does not necessarily support an address of 0 to 65535. For address ranges, refer to the device documentation.

Optional Dot Bits

For Byte, Word, Short, DWord, or Long data types, an optional .bb (dot bit) can be appended to the address to reference a bit in a particular value. The valid ranges for the optional bit is 0-7 for Byte types; 0-15 for Word, Short, and Boolean types; 0-31 for DWord and Long types; 1-211 for String types. Float types do not support bit operations. Boolean and String types require a bit number. The bit number for String types specifies the number of characters in the string.

Dynamic addresses with bit numbers in the range of 0-7 defaults to Byte; 8-15 defaults to Word; 16-31 defaults to DWord. V Memory addresses with a bit number larger than 31 defaults to String. The following diagram illustrates how the driver maps bits within the controller.
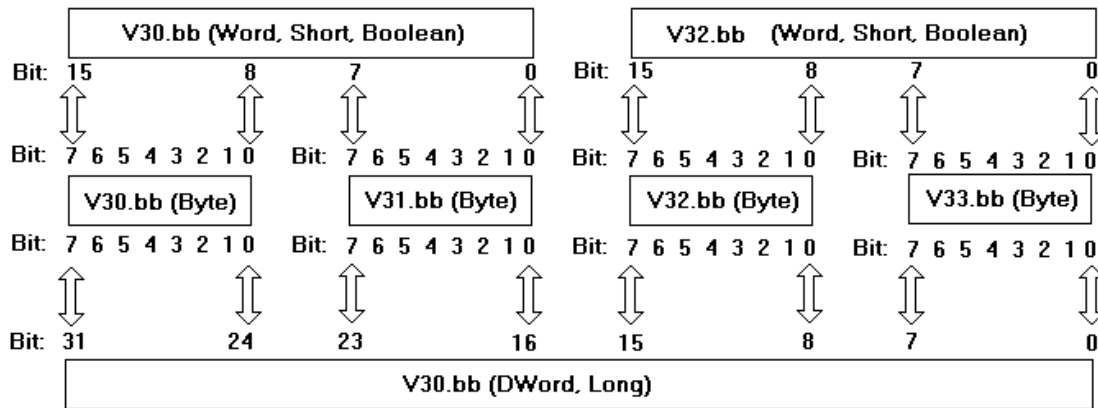
> **Note**: V30.10@bool, V30.2@byte, and V30.26@DWord all reference the same bit in the controller.

Arrays

In addition to the address formats listed above, certain memory types (I, Q, M, SM, V, AI, AQ) support an array operation. Boolean arrays are not allowed at this time. To specify an array address, append *[rows][cols]* to the end of an address. If only [cols] is specified, [rows] defaults to 1. With the array type, it is possible to read and write a block of 200 bytes at one time.

The maximum array size for Word and Short types is 100, and for DWord, Long and Float types is 50. The array size is determined by the multiplication of rows and cols.

> **Note**: The maximum array size also depends on the maximum block size of the device being used.

Examples

1. To read and write an array of 10 Variable Memory Float values starting with V10, declare an address as follows:
V10 [1][10]. Choose Float for the data type. This array reads and writes values to registers V10, V14, V18, V22 ... V46.

2. To read and write to bit 23 of Internal Memory Long M20, declare an address as follows: M20.23. Choose Long for the data type.

Strings

The driver allows for variable length strings to be stored in Variable Memory locations. The bit number specifies the string length (1-211) in characters. String data that is sent to the device, but is smaller in length than the string character count (bit number) is null terminated. String data that meets or exceeds the character length is truncated to the character count and sent to the device without a null terminator.

To read and write a string starting at V5 for a length of 10 characters, declare an address as follows: V5.10. Choose string for the data type.

> **Notes**:
>
> 1. V Memory locations V5-V14 would be used to store this 10 -character string.
>
> 2. Not all devices support up to 211-character requests in a single transaction. To determine the maximum number of characters that can be requested in a transaction, refer to the device documentation. This value is the largest string the driver can Read / Write to and from the device.

🔶 When modifying Word, Short, DWord, Long and Float types remember that each address starts at a byte offset within the device. Therefore, Words V0 and V1 overlap at byte 1. Writing to V0 modifies the value held in V1. Similarly, DWord, Long, and Float types can also overlap. It is recommended that these memory types be used so that overlapping does not occur. For example, when using DWords, use V0, V4, V8 ... and so on, to prevent overlapping bytes.

Event Log Messages

The following information concerns messages posted to the Event Log pane in the main user interface. Consult the OPC server help on filtering and sorting the Event Log detail view. Server help contains many common messages, so should also be searched. Generally, the type of message (informational, warning) and troubleshooting information is provided whenever possible.

Block may have addresses out of range. | Block start address = '<address>', Block size = <count> (bytes).

Error Type:
Warning

Possible Cause:
An attempt has been made to reference a block of memory that contains at least one non-existent location in the specified device.

Possible Solution:
Verify that the tags assigned to addresses are within the specified range on the device and eliminate any that reference invalid locations.

Error Mask Definitions

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

Index

A

B

C

T

U

V

W