Table of Contents

Siemens TCP/IP Ethernet Driver ................................................................. 1

Table of Contents ......................................................................................... 2

Siemens TCP/IP Ethernet Driver ................................................................. 5
Overview ........................................................................................................ 5

Setup ............................................................................................................... 5

Channel Properties — General ................................................................. 6
Channel Properties — Ethernet Communications ...................................... 7
Channel Properties — Write Optimizations .................................................. 7
Channel Properties — Advanced ................................................................. 8
Device Properties — Identification ............................................................... 9
Operating Mode ............................................................................................. 9

Device Properties — Scan Mode ................................................................. 10
Device Properties — Timing ........................................................................ 11
Device Properties — Auto-Demotion ............................................................. 12
Device Properties — Tag Generation ............................................................ 12
Device Properties — Communications Parameters ...................................... 14
Device Properties — S7 Communication Parameters .................................... 15
S7-300/400/1200/1500 ................................................................................ 16
Device Properties — Addressing Options .................................................... 16
Device Properties — Tag Import .................................................................. 18
Device Properties — Redundancy ................................................................. 19

Channel Configuration API Commands ..................................................... 19

Siemens TCP/IP Ethernet Channel Setup with API Commands .................... 20

Device Configuration API Commands ......................................................... 20

Siemens TCP/IP Ethernet Device Setup with API Commands ...................... 21

Siemens TCP/IP Ethernet API Example ....................................................... 23

Optimizing Communications ...................................................................... 24

Data Types Description .............................................................................. 25

Address Descriptions .................................................................................. 25

S7-200 Address Descriptions .................................................................... 26
S7-300 Address Descriptions .................................................................... 28
S7-400 Address Descriptions .................................................................... 29
S7-1200 Address Descriptions ................................................................ 29
S7-1500 Address Descriptions ................................................................ 29
NetLink: S7-300 Address Descriptions ..................................................... 30
NetLink: S7-400 Address Descriptions ..................................................... 30
Event Log Messages ........................................................................................................45

Reason = 'Frame contains errors'. ..................................................................................45
Reason = 'Device returned transport error'. Error code = <error>. .................................45
Reason = 'Device returned protocol error'. Error class = <class>, Error code = <error>. ......................................................................................................................................46
Reason = 'Device returned data access error'. Error code = <error>. .................................46
Reason = 'Device is not responding'. ................................................................................46
Reason = 'Unknown error occurred'. ................................................................................47
Reason = NetLink returned error. Error code = <error>. ..................................................47
Failed to resolve host. | Host = '<host name>'. .................................................................48
Auto-generated tag names and descriptions may not appear as expected due to string conversion error. ..............................................................................................................48
A required code page is unavailable on this machine. Tag generation may fail or tag names and descriptions may not appear as expected. | Required code page = <page>. .............................................................................................................48
Unable to load the Step 7 language file. ..........................................................................49
Memory exception reading the Step 7 language file. .........................................................49
Step 7 language file failed to open. | OS error = '<error>'. ................................................49
Tag generation failure. | Data block name = '<block name>', data block number = <block number>. .................................................................................................................................50
Created tag in group due to internal block size. | Tag address = '<address>', tag name = 'name>', group name = '<name>'. .................................................................................................................................50
Tag not created because arrays are not supported with specified data type. | Tag name = 'name>', group name = 'name>', data type = '<type>'. .................................................................................................................................50
Unable to connect to device. | ............................................................................................51
Unable to establish association with device. | .................................................................51
Unable to read from address on device. | Address = '<address>', ............................................52
Unable to read from address on device. Tag deactivated. | Address = '<address>', ..........53
Unable to read data from device. | Data block = '<block>', block start = <address>, block size = <size>, .................................................................................................................................53
Unable to read data from device. Block deactivated. | Data block = '<block>', block start = <address>, block size = <size>, .................................................................................................................................54
Unable to read data from device. | Memory type = '<type>', block start = <address>, block size = <size> (bytes), .................................................................................................................................55
Unable to read data from device. Block deactivated. | Memory type = '<type>', block start = <address>, block size = <size> (bytes), .................................................................................................................................56
Unable to write to address on device. | Address = '<address>', ................................................57
Unable to write to address on device. HEXSTRING length is different from tag length. | Address = '<address>', HEXSTRING length = <length> (bytes), tag length = <length> (bytes). .................................................................................................................................58
Unable to write to address on device. HEXSTRING contains a non-hexadecimal character. |
Address = '<address>'. ................................................................. 58

Unable to write to address on device. HEXSTRING length must be an even number of characters. | Address = '<address>', ................................................................. 58

Unable to write to address on device. Time of Day string contains a syntax error. Expected 'hh:mm:ss.sss' format. | Address = '<address>', Time of Day string = '<string>'. .............................. 59

Error Codes .................................................................................. 59

Appendix: Configuring Siemens Connections ........................................... 61

How To Configure S7-200 Connections in Micro/WIN .................................. 61
How To Configure S7-300/400 Connections in STEP 7 .................................... 76
How To Configure S7-1200 Connections with the Totally Integrated Automation (TIA) Portal ...... 82
How To Configure S7-1500 Connections .................................................... 85

Appendix: Alternate Syntax Support ......................................................... 87

Applicom Direct-Link Item Syntax ...................................................... 87
INAT OPC Server TCPIPH1 Item Syntax ................................................ 89
Siemens Simatic Net Item Syntax ........................................................... 93
Siemens STEP 7 Item Syntax ............................................................... 95
Softing S7/S5 OPC Server Item Syntax .................................................. 98

Index ............................................................................................. 101
Siemens TCP/IP Ethernet Driver

Overview
What is the Siemens TCP/IP Ethernet Driver?

Getting Started
How do I configure a channel and device?

Using the API
How do I configure a channel and device via the API?

Optimizing Communications
How do I get the best performance from the driver?

Data Types Description
What data types does this driver support?

Address Descriptions
How do I address a data location on a Siemens TCP/IP device?

Event Log Messages
What messages does the Siemens TCP/IP Ethernet Driver produce?

Overview
The Siemens TCP/IP Ethernet Driver provides a reliable way to connect Siemens TCP/IP Ethernet devices to OPC client applications, including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with Siemens S7-200, 300, 400, 1200, and 1500 PLCs. There are two options for communications:

- Industrial Ethernet TCP/IP interface communication processor (CP). The protocol used is S7 Messaging on Industrial Ethernet (ISO 8073 Class 0) over TCP/IP as defined in RFC1006.
- Hilscher’s NetLink adapter. Only an MPI port is required. The NetLink adapter does not support the S7-200 model.

The driver requires no special libraries or hardware. A standard Ethernet card is all that is needed.

Setup
A maximum of 1024 devices may be defined on a channel.

Supported Devices
S7-200 via CP243
S7-300 via CP343
S7-400 via CP443
S7-1200*
S7-1500*
S7-300 via NetLink
S7-400 via NetLink
*This device has a built-in Ethernet module.

**Supported NetLink Cables and Gateways**

NT 50-MPI  
NL 50-MPI  
NL-MPI

*Note:* For NetLink users, NetLink communication parameters (such as IP Address, Subnet Mask, and Baud Rate) can be configured using the NetLink Configuration utility. This application is located in the server’s Utilities sub-directory and can be launched using the Start menu shortcut.

**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:** 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. 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" in the server help.

---

**Channel Properties — Ethernet Communications**

Ethernet Communication can be used to communicate with devices.

---

**Ethernet Settings**

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

---

**Channel Properties — Write Optimizations**

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.

---

**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

---

www.ptc.com
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.

<table>
<thead>
<tr>
<th>Property Groups</th>
<th>Non-Normalized Float Handling</th>
<th>Inter-Device Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Floating-Point Values</td>
<td>Replace with Zero</td>
</tr>
<tr>
<td>Write Optimizations</td>
<td>Inter-Device Delay (ms)</td>
<td>0</td>
</tr>
</tbody>
</table>

**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.

**Device Properties — 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:** the unique identity of the device for communication with the driver. The device ID is formatted as YYY.YYY.YYY.YYY, where YYY designates the device's IP address. Each YYY byte should be in the range of 0 to 255. If the device supports host name resolution, the device ID may also be specified as a standard UNC/DNS name.

*See Also:* Operating Mode

**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.

**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.

![Property Groups and Communication Timeouts](image)

**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.

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 — Tag Generation

The automatic tag database generation features make setting up an application a plug-and-play operation. Select communications drivers can be configured to automatically build a list of tags that correspond to device-specific data. These automatically generated tags (which depend on the nature of the supporting driver) can be browsed from the clients.

Not all devices and drivers support full automatic tag database generation and not all support the same data types. Consult the data types descriptions or the supported data type lists for each driver for specifics.

If the target device supports its own local tag database, the driver reads the device’s tag information and uses the data to generate tags within the server. If the device does not natively support named tags, the driver creates a list of tags based on driver-specific information. An example of these two conditions is as follows:

1. If a data acquisition system supports its own local tag database, the communications driver uses the tag names found in the device to build the server’s tags.

2. If an Ethernet I/O system supports detection of its own available I/O module types, the com-
communications driver automatically generates tags in the server that are based on the types of I/O modules plugged into the Ethernet I/O rack.

Note: Automatic tag database generation's mode of operation is completely configurable. For more information, refer to the property descriptions below.

<table>
<thead>
<tr>
<th>Property Groups</th>
<th>Tag Generation</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>On Property Change</td>
<td>Yes</td>
</tr>
<tr>
<td>Scan Mode</td>
<td>On Device Startup</td>
<td>Do Not Generate on Startup</td>
</tr>
<tr>
<td>Timing</td>
<td>On Duplicate Tag</td>
<td>Delete on Create</td>
</tr>
<tr>
<td>Auto-Demotion</td>
<td>Parent Group</td>
<td></td>
</tr>
<tr>
<td>Tag Generation</td>
<td>Allow Automatically Generated Subgroups</td>
<td>Enable</td>
</tr>
<tr>
<td></td>
<td>Create</td>
<td>Create tags</td>
</tr>
</tbody>
</table>

On Property Change: If the device supports automatic tag generation when certain properties change, the On Property Change option is shown. It is set to Yes by default, but it can be set to No to control over when tag generation is performed. In this case, the Create tags action must be manually invoked to perform tag generation.

On Device Startup: This property specifies when OPC tags are automatically generated. Descriptions of the options are as follows:

- **Do Not Generate on Startup**: This option prevents the driver from adding any OPC tags to the tag space of the server. This is the default setting.
- **Always Generate on Startup**: This option causes the driver to evaluate the device for tag information. It also adds tags to the tag space of the server every time the server is launched.
- **Generate on First Startup**: This option causes the driver to evaluate the target device for tag information the first time the project is run. It also adds any OPC tags to the server tag space as needed.

Note: When the option to automatically generate OPC tags is selected, any tags that are added to the server’s tag space must be saved with the project. Users can configure the project to automatically save from the Tools | Options menu.

On Duplicate Tag: When automatic tag database generation is enabled, the server needs to know what to do with the tags that it may have previously added or with tags that have been added or modified after the communications driver since their original creation. This setting controls how the server handles OPC tags that were automatically generated and currently exist in the project. It also prevents automatically generated tags from accumulating in the server.

For example, if a user changes the I/O modules in the rack with the server configured to Always Generate on Startup, new tags would be added to the server every time the communications driver detected a new I/O module. If the old tags were not removed, many unused tags could accumulate in the server’s tag space. The options are:

- **Delete on Create**: This option deletes any tags that were previously added to the tag space before any new tags are added. This is the default setting.
- **Overwrite as Necessary**: This option instructs the server to only remove the tags that the communications driver is replacing with new tags. Any tags that are not being overwritten remain in the server’s tag space.
- **Do not Overwrite**: This option prevents the server from removing any tags that were previously generated or already existed in the server. The communications driver can only add tags that are completely new.

- **Do not Overwrite, Log Error**: This option has the same effect as the prior option, and also posts an error message to the server's Event Log when a tag overwrite would have occurred.

  **Note**: Removing OPC tags affects tags that have been automatically generated by the communications driver as well as any tags that have been added using names that match generated tags. Users should avoid adding tags to the server using names that may match tags that are automatically generated by the driver.

**Parent Group**: This property keeps automatically generated tags from mixing with tags that have been entered manually by specifying a group to be used for automatically generated tags. The name of the group can be up to 256 characters. This parent group provides a root branch to which all automatically generated tags are added.

**Allow Automatically Generated Subgroups**: This property controls whether the server automatically creates subgroups for the automatically generated tags. This is the default setting. If disabled, the server generates the device's tags in a flat list without any grouping. In the server project, the resulting tags are named with the address value. For example, the tag names are not retained during the generation process.

  **Note**: If, as the server is generating tags, a tag is assigned the same name as an existing tag, the system automatically increments to the next highest number so that the tag name is not duplicated. For example, if the generation process creates a tag named "AI22" that already exists, it creates the tag as "AI23" instead.

**Create**: Initiates the creation of automatically generated OPC tags. If the device's configuration has been modified, **Create tags** forces the driver to reevaluate the device for possible tag changes. Its ability to be accessed from the System tags allows a client application to initiate tag database creation.

  **Note**: **Create tags** is disabled if the Configuration edits a project offline.

### Device Properties — Communications Parameters

**Port Number**: This parameter specifies the port number that the remote CP is configured to use. The default setting for IE TCP/IP is 102 (TSAP). The default setting for NetLink is 1099.

  **Note**: It is recommended that the default port be used for most applications, where the server and the PLC exist on the same network. For an application using the Internet through firewalls and advanced routers, the port number can be changed to allow these operations to occur. In most cases, however, the PLC only accepts a connection on port 102/1099 and may require router forwarding.

**MPI ID**: This parameter is for NetLink only, and is configured for the port in which the NetLink adapter is connected. It does not apply to models utilizing the IE TCP/IP CPs (such as S7-300 and S7-400). A maximum of two connections or devices via TCP are possible when using the NetLink adapter.
Device Properties — S7 Communication Parameters

The S7 family includes specific parameters, which are broken out into the following groups: **S7 Communication Parameters**, **S7-200**, and **S7-300/400/1200/1500**.

### S7 Communication Parameters

**Maximum PDU Size**: This parameter establishes the maximum Protocol Data Unit (PDU) size that will be requested from the device. The actual PDU used for communication depends on what the device supports. Typically, the driver and device negotiate the highest supported PDU size. However, this parameter can force a lower PDU size than would normally be negotiated.

*Note*: To observe the PDU value negotiated with the device, use the _CurrentPDUSize internal tag (See Internal Tags).

### S7-200

S7-200 enables communication with S7-200 devices on an Ethernet network. There are two options:

- PG connection (such as, a connection utilized by Micro/WIN). One connection is available.
- Configured connection (such as, a connection configured in Micro/WIN via the Ethernet wizard). Eight connections are available.

*Note*: Configured connections are recommended because they free the PG port for Micro/WIN and also provide flexibility to make multiple concurrent connections.

#### Local TSAP

<table>
<thead>
<tr>
<th>Link Type</th>
<th>TSAP Value (hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG</td>
<td>4B57 (‘KW’)</td>
</tr>
<tr>
<td>Configured</td>
<td>A remote (client) TSAP configured in Micro/WIN’s Ethernet wizard. If Micro/WIN remote TSAP=xx.yy*, set local TSAP to xxyy.</td>
</tr>
</tbody>
</table>

#### Remote TSAP

<table>
<thead>
<tr>
<th>Link Type</th>
<th>TSAP Value (hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG</td>
<td>4B57 (‘KW’)</td>
</tr>
<tr>
<td>Configured</td>
<td>A local (server) TSAP configured in Micro/WIN’s Ethernet wizard. If Micro/WIN remote TSAP=xx.yy*, set local TSAP to xxyy.</td>
</tr>
</tbody>
</table>
*TSAP as displayed in Micro/WIN’s Ethernet wizard. When accessed from V memory, the value may be in decimal form. For example, if TSAP is 10.00, the V memory value is 1000 hex or 4096 decimal. The values entered for Local TSAP must be in hexadecimal notation; in this example, the value 1000 would be entered.

**Tip:** Local TSAP==Micro/WIN remote TSAP, Remote TSAP==Micro/WIN local TSAP.

For information on using the CP243-1 module, refer to *How to Configure S7-200 Connections in Micro/WIN.*

### S7-300/400/1200/1500

**Link Type:** Defines the communication connection between the driver and the CP. The type of link chosen determines the number of simultaneous requests allowed. The greater the number of simultaneous requests, the greater the data throughput. Each device connection is allowed one outstanding request. To achieve multiple simultaneous requests, multiple connections must be configured. This is achieved by defining the device multiple times in the server (identical device properties). The devices can be defined within the same channel or under separate channels.

For more information, refer to *Optimizing Communication.*

Channel.Device=1 CP connection

There are three types of links: PC (applications), OP (operator panel), and PG (programming device). OP and PG are usually reserved, but can be used if all PC connections are taken.

<table>
<thead>
<tr>
<th>Type</th>
<th>S7-300 CPU 314, 315</th>
<th>S7-400 CPU 412, 413</th>
<th>S7-400 CPU 414</th>
<th>S7-400 CPU 416</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>2</td>
<td>14</td>
<td>30</td>
<td>62</td>
</tr>
<tr>
<td>OP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PG</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Example**

Given an S7-400 CPU 412 device, 14 simultaneous requests can be achieved by defining 14 identical devices in the server with all configured for Link Type PC. In addition to the PC connections, two more devices can be configured for Link Type OP and PG. This provides 16 connections overall.

Connection resources are shared amongst applications communicating with the CP. If another application such as STEP 7 is configured to use Industrial Ethernet over TCP/IP, at least one PG/PC connection must be left open for that application.

For information on increasing the number of PG, OP, and PC type connections, refer to *How to Configure S7-300/400 Connections in STEP 7.*

**CPU Rack:** The number of the rack in which the CPU of interest resides.

**CPU Slot:** The number of the slot in the rack in which the CPU of interest resides.

For information on how to read or write the rack number or slot number using an internal tag, refer to *Internal Tags.*

### Device Properties — Addressing Options
**Byte Order**: establishes the order for 16-bit and 32-bit values. Options include Big Endian (S7 Default) or Little Endian, explained below.

### Big Endian

<table>
<thead>
<tr>
<th>DWord 1</th>
<th>Word 1</th>
<th>Word 3</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Byte 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td>1- 1- 1- 1- 1- 1- 1- 0</td>
<td>2- 2- 2- 2- 1- 1- 1- 0</td>
<td>7 6 5 4 3 2 1 0</td>
<td>7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Bits
- The bit range for DWord 1 is 31-0.
- The bit range for Word 1 and Word 3 is 15-0.
- The bit range for Byte 1, Byte 2, Byte 3, and Byte 4 is 7-0.

*Note*: Big Endian uses bytes ordered from highest to lowest. The bit order is never changed.

### Little Endian

<table>
<thead>
<tr>
<th>DWord 1</th>
<th>Word 3</th>
<th>Word 1</th>
<th>Byte 4</th>
<th>Byte 3</th>
<th>Byte 2</th>
<th>Byte 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3- 3- 2- 2- 1- 0</td>
<td>1- 1- 1- 1- 1- 0</td>
<td>1- 1- 1- 1- 1- 0</td>
<td>7 6 5 4 3 2 1 0</td>
<td>7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Bits
- The bit range for DWord 1 is 31-0.
- The bit range for Word 3 and Word 1 is 15-0.
- The bit range for Byte 4, Byte 3, Byte 2, and Byte 1 is 7-0.

*Note*: Little Endian uses bytes ordered from lowest to highest. The bit order is never changed.
**Device Properties — Tag Import**

The Tag Import parameters allow automatic tag database generation from projects defined in Siemens STEP 7 or in Siemens TIA Portal that were exported via the TIA Portal Exporter.

**Supported Models via Siemens STEP 7**

S7-300  
S7-400

**Supported Models via Siemens TIA Portal**

S7-300  
S7-400  
S7-1200  
S7-1500

**STEP 7 Tag Import**

<table>
<thead>
<tr>
<th>Property Groups</th>
<th>Tag Import Type</th>
<th>Step 7 Project File</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scan Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tag Import</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tag Import Type:** Select "Step 7 Project File" from the drop-down menu.

**STEP 7 Project (.S7P):** Locate and select the desired STEP 7 project file (.S7P) from which to import tags.  

*Note:* The desired STEP 7 project file (.S7P) must be located in the Step 7 project directory to import tags.

**Program Path:** Specify the PLC program within the project for which tags should be generated. The program path format is "Station Name\CPU Name\Program Name" and the program path selection drop-down list can be populated with stations that have single or multiple CPUs. When a station with multiple CPUs is found, each CPU associated with the station has a CPU ID designator formatted as " @ ID x" appended to the CPU name portion of program path.  

For example: A SIMATIC 400 controller with two 400 CPUs would have two entries:

- "=K51 S7-400\CPU @ ID 3K34"
- "=K51 S7-400\CPU 416-3 DP @ ID 2K51"

*Note:* Tag import for the Siemens S7-300 and S7-400 devices have been qualified for use with projects created from Siemens Simatic STEP 7 versions 5.3, 5.4, and 5.5.

*Important:* Tag import for the Siemens TCP/IP Ethernet Driver supports tag names and comments in the native character set as specified by the Windows code page in the Siemens STEP 7 project’s language file. A missing, altered, corrupt, or incorrect Siemens STEP 7 language file may cause tag names and comments to import incorrectly. Utilizing the STEP 7 language-neutral option (which allows text to be entered in a different character set than what is used in the STEP 7 language file) may also cause tag names and comments to import incorrectly. The STEP 7 language file can be located in the Global sub-directory of the STEP 7 project root. Automatic tag generation may result in the display of incorrect characters if the necessary language packs are not installed on the system.

*See Also:* Appendix: Configuring Siemens Connections
**TIA Portal Tag Import**

The TIA Portal Exporter can export tags from a Siemens TIA Portal project into the server. This utility opens a project, and allows the selection of program blocks, tag tables, or individual tags for export. Tags are exported into a format which can be consumed by the Siemens TCP/IP Ethernet Driver Automatic Tag Generation process.

When Siemens TCP/IP Ethernet Driver is installed, the application installer for TIA Portal Exporter is saved in the Server's "Utilities" folder. Copy the installer to a computer that has Siemens TIA Portal and the Openness API installed. Run the installer and refer to the instructions in the Help documentation to create a *.TPE export file.

**Tag Import Type:** Select “TIA Portal Exporter file” from the drop-down menu.

**TIA Portal Exporter file (*.TPE):** Locate and select the desired TIA Portal Exporter file (*.TPE) from which to import tags.

**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.*

**Channel Configuration API Commands**

The following commands define a channel using the Configuration API service.

**General Properties**

common.ALLTYPES_NAME * Required parameter.

**Note:** Changing this property causes the API endpoint URL to change.

common.ALLTYPES_DESCRIPTION

servermain.MULTIPLE_TYPES DEVICE DRIVER * Required parameter

servermain.CHANNEL_DIAGNOSTICS_CAPTURE

**Ethernet Communication Properties**
Siemens TCP/IP Ethernet Driver

servermain.CHANNEL ETHERNET COMMUNICATIONS NETWORK_ADAPTER_STRING

Advanced Properties

servermain.CHANNEL_NON_NORMALIZED_FLOATING_POINT_HANDLING * Required parameter

Write Optimizations

servermain.CHANNEL_WRITE_OPTIMIZATIONS_METHOD
servermain.CHANNEL_WRITE_OPTIMIZATIONS_DUTY_CYCLE

See Also: The server help system Configuration API Service section.

Siemens TCP/IP Ethernet Channel Setup with API Commands

The minimum required properties to create a Siemens TCP/IP Ethernet device under a channel are common.ALLTYPES_NAME, and servermain.MULTIPLE_TYPES_DEVICE_DRIVER. All other properties will be set to their default values.

JSON Example

Below is an example JSON blob that can be used in a POST to the channel endpoint to make a channel named Channel1.

```
{
    "common.ALLTYPES_NAME": "Channel1",
    "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Siemens TCP/IP Ethernet"
}
```

If there is a channel named Channel1 and a device name Device1 with the server running on the local host, the GET request query would be sent to:

```
/config/v1/project/channels/Channel1/devices/Device1?content=property_definitions
```

/config/v1/doc/drivers/Siemens%20TCP~IP%20Ethernet/Devices has the enumerated details for devices.

Device Configuration API Commands

The following commands define a channel using the Configuration API service.

General Properties

common.ALLTYPES_NAME
common.ALLTYPES_DESCRIPTION
servermain.DEVICE_CHANNEL_ASSIGNMENT
servermain.MULTIPLE_TYPES_DEVICE_DRIVER
servermain.DEVICE_MODEL
servermain.DEVICE_ID_STRING
servermain.DEVICE_DATA_COLLECTION
servermain.DEVICE_SIMULATED

**Scan Mode**

servermain.DEVICE_SCAN_MODE * Required parameter
servermain.DEVICE_SCAN_MODE_RATE_MS

**Auto Demotion**

servermain.DEVICE_AUTO_DEMOTION_ENABLE_ON_COMMUNICATIONS_FAILURES
servermain.DEVICE_AUTO_DEMOTION_DEMOTE_AFTER_SUCCESSIVE_TIMEOUTS
servermain.DEVICE_AUTO_DEMOTION_PERIOD_MS
servermain.DEVICE_AUTO_DEMOTION_discardWrites

**Tag Generation**

servermain.DEVICE_TAG_GENERATION_ON_STARTUP * Required parameter
servermain.DEVICE_TAG_GENERATION_DUPLICATE_HANDLING * Required parameter
servermain.DEVICE_TAG_GENERATION_GROUP
servermain.DEVICE_TAG_GENERATION_ALLOW_SUB_GROUPS

**Tip:** To invoke automatic Tag Generation, send a PUT with an empty body to the TagGeneration service endpoint on the device.

**See Also:** For more information see Services help.

**Timing**

servermain.DEVICE_CONNECTION_TIMEOUT_SECONDS
servermain.DEVICE_REQUEST_TIMEOUT_MILLISECONDS
servermain.DEVICE_RETRY_ATTEMPTS
servermain.DEVICE_INTER_REQUEST_DELAY_MILLISECONDS

**See Also:** The server help system Configuration API Service section.

**Siemens TCP/IP Ethernet Device Setup with API Commands**
The API commands to set unique properties to create a Siemens TCP/IP Ethernet device under a channel are:

**General**
common.ALLTYPES_NAME
common.ALLTYPES_DESCRIPTION
servermain.DEVICE_CHANNEL_ASSIGNMENT
servermain.MULTIPLE_TYPES DEVICE_DRIVER
servermain.DEVICE_MODEL
servermain.DEVICE_ID_STRING

Communication Parameters

siemens_tcpip_ethernet.DEVICE_COMMUNICATIONS_PORT_NUMBER
siemens_tcpip_ethernet.DEVICE_COMMUNICATIONS_MPI_ID
siemens_tcpip_ethernet.DEVICE_S7_COMMUNICATIONS_MAX_PDU
siemens_tcpip_ethernet.DEVICE_S7_COMMUNICATIONS_200_LOCAL_TSAP
siemens_tcpip_ethernet.DEVICE_S7_COMMUNICATIONS_200_REMOTE_TSAP
siemens_tcpip_ethernet.DEVICE_S7_COMMUNICATIONS_300_400_1200_1500_LINK_TYPE
siemens_tcpip_ethernet.DEVICE_S7_COMMUNICATIONS_CPU_RACK
siemens_tcpip_ethernet.DEVICE_S7_COMMUNICATIONS_CPU_SLOT

Addressing Options

siemens_tcpip_ethernet.DEVICE_ADDRESSING_BYTE_ORDER

Tag Import

siemens_tcpip_ethernet.DEVICE_TAG_IMPORT_STEP_7_PROJECT_FILE
siemens_tcpip_ethernet.DEVICE_TAG_IMPORT_PROGRAM_PATH
siemens_tcpip_ethernet.DEVICE_TAG_IMPORT_TYPE: This is always set to “TIA Portal Exporter file”.

siemens_tcpip_ethernet.DEVICE_TAG_IMPORT_TIA_EXPORT_FILE: Set the desired TIA Portal Exporter file (*.TPE) from which to import tags.

Notes:

Items marked with an * have values that are enumerated. The valid list of enumerations and values can be found by querying the device endpoint with content=property_definitions. This query returns a list of all the properties and their valid enumerations as well as a description for each.

For example, if there is a channel named Channel1 and a device name Device1 with the server running on the local host, the GET request query would be sent to:

/config/v1/project/channels/Channel1/devices/Device1?content=property_definitions
Siemens TCP/IP Ethernet Driver

/config/v1/doc/drivers/Siemens%20TCP~IP%20Ethernet/Devices has the enumerated details for devices.

### Siemens TCP/IP Ethernet API Example

For a list of channel and device definitions and enumerations, access the following endpoints with the REST client:

#### Channel Definitions

/config/v1/doc/drivers/Siemens%20TCP%2FIP%20Ethernet/channels

#### Device Definitions

/config/v1/doc/drivers/Siemens%20TCP%2FIP%20Ethernet/devices

#### Create Siemens Channel

Endpoint (POST):

/config/v1/project/channels

Body:

```json
{
  "common.ALLTYPES_NAME": "MyChannel",
  "servermain.MULTIPLE_TYPES_DEVICE.Driver": "Siemens TCP/IP Ethernet"
}
```

#### Create Siemens Device

Endpoint (POST):

/config/v1/project/channels/MyChannel/devices

Body:

```json
{
  "common.ALLTYPES_NAME": "MyDevice",
  "servermain.DEVICE_ID_STRING": "<192.160.0.1>.0",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Siemens TCP/IP Ethernet",
  "servermain.DEVICE_MODEL": <model enumeration>
}
```

#### Create Siemens Tags Example

Endpoint (POST):

/config/v1/project/channels/MyChannel/devices/MyDevice/tags

Body:
See server help for more information on configuring projects over the Configuration API.

Optimizing Communications

The Siemens TCP/IP Ethernet Driver was designed to provide the best performance with the least amount of impact on the system’s overall performance. While the Siemens TCP/IP Ethernet Driver is fast, there are a couple of guidelines that can be used to optimize the application and gain maximum performance.

This server refers to communications protocols like Siemens TCP/IP Ethernet as a channel. Each channel defined in the application represents a separate path of execution in the server. Once a channel has been defined, a series of devices can then be defined under that channel. Each of these devices represents a single Siemens TCP/IP Ethernet controller from which data will be collected. Although this approach to defining the application provides a high level of performance, it does not take full advantage of the Siemens TCP/IP Ethernet Driver or the network. An example of how the application may appear when configured using a single channel is shown below.

Each device appears under a single Siemens TCP/IP Ethernet channel. In this configuration, the driver must move from one device to the next as quickly as possible to gather information at an effective rate. As more devices are added or more information is requested from a single device, the overall update rate begins to suffer.

If the Siemens TCP/IP Ethernet Driver could only define one channel, then the example above would be the only option available; however, the driver can define up to 256 channels. Using multiple channels distributes the data collection workload by simultaneously issuing multiple requests to the network. An example of how the same application may appear when configured using multiple channels is shown below.

Each device has now been defined under its own channel. In this new configuration, a single path of execution is dedicated to the task of gathering data from each device. The performance will improve even if the application has more than 256 devices. While 256 or fewer devices may be ideal, the application will still benefit from additional channels. Although spreading the device load across all channels will cause the server to move from device to device again, it can now do so with far less devices to process on a single channel.

Although the server limits the number of channels to 256, the device ultimately determines the number of allowed connections. This constraint comes from the fact that some devices cannot support 256 connections. For these devices, the maximum number of channels defined should equal the maximum number of connections allowed. For devices that support more than 256 connections, the maximum 256 channels should be defined, with devices spread evenly over these 256 channels.
For more information on device connections, refer to Device Properties.

Data Types Description

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Single bit</td>
</tr>
<tr>
<td>Byte</td>
<td>Unsigned 8-bit value</td>
</tr>
<tr>
<td>Char</td>
<td>Signed 8-bit value</td>
</tr>
<tr>
<td>Word</td>
<td>Unsigned 16-bit value</td>
</tr>
<tr>
<td></td>
<td>bit 0 is the low bit</td>
</tr>
<tr>
<td></td>
<td>bit 15 is the high bit</td>
</tr>
<tr>
<td>Short</td>
<td>Signed 16-bit value</td>
</tr>
<tr>
<td></td>
<td>bit 0 is the low bit</td>
</tr>
<tr>
<td></td>
<td>bit 14 is the high bit</td>
</tr>
<tr>
<td></td>
<td>bit 15 is the sign bit</td>
</tr>
<tr>
<td>BCD</td>
<td>Two byte packed BCD</td>
</tr>
<tr>
<td></td>
<td>Value range is 0-9999. Behavior is undefined for values beyond this range</td>
</tr>
<tr>
<td>DWord</td>
<td>Unsigned 32-bit value</td>
</tr>
<tr>
<td></td>
<td>bit 0 is the low bit</td>
</tr>
<tr>
<td></td>
<td>bit 31 is the high bit</td>
</tr>
<tr>
<td>Long</td>
<td>Signed 32-bit value</td>
</tr>
<tr>
<td></td>
<td>bit 0 is the low bit</td>
</tr>
<tr>
<td></td>
<td>bit 30 is the high bit</td>
</tr>
<tr>
<td></td>
<td>bit 31 is the sign bit</td>
</tr>
<tr>
<td>LBCD</td>
<td>Four byte packed BCD</td>
</tr>
<tr>
<td></td>
<td>Value range is 0-99999999. Behavior is undefined for values beyond this range</td>
</tr>
<tr>
<td>Float</td>
<td>32-bit floating point value</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Date</td>
<td>64-bit floating-point value</td>
</tr>
<tr>
<td>String</td>
<td>Null-terminated ASCII string*</td>
</tr>
</tbody>
</table>

*The Data Block subtype, String, is a NULL padded ASCII string.

Address Descriptions

Address specifications vary depending on the model in use. Select a link from the following list to obtain information for the model of interest.
### S7-200 Address Descriptions

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

<table>
<thead>
<tr>
<th>Address Type</th>
<th>Range</th>
<th>Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Inputs (IEC)</td>
<td>I0.b-I65535.b</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>.b is Bit Number 0-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IB0-I0W65535</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IW0-IW65534</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ID0-ID65532</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete Inputs (SIMATIC)</td>
<td>E0.b-E65535.b</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>.b is Bit Number 0-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EB0-EB65535**</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EW0-EW65534</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ED0-ED65532</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete Outputs (IEC)</td>
<td>Q0.b-Q65535.b</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>.b is Bit Number 0-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>QB0-QB65535</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>QW0-QW65534</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>QD0-QD65532</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete Outputs (SIMATIC)</td>
<td>A0.b-A65535.b</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>.b is Bit Number 0-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AB0-AB65535</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AW0-AW65534</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AD0-AD65532</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>**</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> I and E access the same memory area.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog Inputs (IEC)</td>
<td>AI0-A165534***</td>
<td>Word, Short</td>
<td>Read Only</td>
</tr>
<tr>
<td></td>
<td>AIW0-AIW65534</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog Inputs (SIMATIC)</td>
<td>AE0-AE65534***</td>
<td>Word, Short</td>
<td>Read Only</td>
</tr>
<tr>
<td></td>
<td>AEW0-AEW65534</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Q and A access the same memory area.
<table>
<thead>
<tr>
<th>Address Type</th>
<th>Range</th>
<th>Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note</strong>: AI and AE access the same memory area.</td>
<td>AQ0-AQ65534*** AQW0-AQW65534</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Analog Outputs (IEC)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog Outputs (SIMATIC)</td>
<td>AA0-AA65534*** AAW0-AAW65534</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td><strong>Note</strong>: AQ and AA access the same memory area.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal Memory</td>
<td>M0.b-M65535.b .b is Bit Number 0-7 MB0-MB65535 MW0-MW65534 MD0-MD65532</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Special Memory (Bytes 0-29 are Read Only)</td>
<td>SM0.b-SM65535.b .b is Bit Number 0-7 SMB0-SMB65535 SMW0-SMW65534 SMD0-SMD65532</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Sequence Control Relay (SCR)</td>
<td>S0.b-S65535.b .b is Bit Number 0-7 SB0-SB65535 SW0-SW65534 SD0-SD65532</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Variable Memory</td>
<td>V0.b-V65535.b .b is Bit Number 0-7 VB0-VB65535 VW0-VW65535 VD0-VD65535</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Timer Current Values</td>
<td>T0-T65535*</td>
<td>DWord, Long</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Timer Status Bit</td>
<td>T0-T65535*</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>Counter Current Values (IEC)</td>
<td>C0-C65535*</td>
<td>Word</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Counter Status Bit (IEC)</td>
<td>C0-C65535*</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td>Counter Current Values (SIMATIC)</td>
<td>Z0-Z65535*</td>
<td>Word</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Counter Status Bit (SIMATIC)</td>
<td>Z0-Z65535*</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td><strong>Note</strong>: C and Z access the same memory area.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Address Type

<table>
<thead>
<tr>
<th>Address Type</th>
<th>Range</th>
<th>Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-Speed Counter</td>
<td>HC0-HC65535*</td>
<td>DWord, Long</td>
<td>Read Only</td>
</tr>
</tbody>
</table>

*These memory types/subtypes do not support arrays.

**Byte memory types (MB) support strings. The syntax for strings is *<address>.<length> where 0 < length <= 932* (see notes below).

***For Analog Inputs and Outputs, the address must be even (AI0, AI2, AI4, and so forth).

#### Notes:

1. All offsets for memory types I, Q, M, S, and SM represent a byte starting location within the specified memory type.

2. Use caution when modifying Word, Short, DWord, and Long types. For I, Q, and F, each address starts at a byte offset within the device. Therefore, Words MW0 and MW1 overlap at byte 1. Writing to MW0 will also modify the value held in MW1. Similarly, DWord, and Long types can also overlap. It is recommended that these memory types be used so that overlapping does not occur. For example, DWord MD0, MD4, MD8, and so on can be used to prevent overlapping bytes.

3. The total number of bytes being requested cannot exceed the data portion of the negotiated PDU size. For example, for a 960-byte PDU size, the largest single array that may be read or written is 932 bytes. If arrays exceed the negotiated PDU size, they may fail to be read or written.

### Arrays

All memory types/subtypes with the exception of those marked with an asterisk support arrays. The valid syntax for declaring an array is as follows:

*<address>[rows][cols]*

*<address>.rows cols*

*<address>,rows.cols*

*<address>_rows_cols*

#### Notes:

1. If no rows are specified, a row count of 1 is assumed.

2. For Word, Short, and BCD arrays, the base address + *(rows * cols * 2)* cannot exceed 65536. Keep in mind that the elements of the array are words, located on a word boundary. For example, IW0[4] would return IW0, IW2, IW4, and IW6.

3. For Float, DWord, Long, and Long BCD arrays, the base address + *(rows * cols * 4)* cannot exceed 65536. Keep in mind that the elements of the array are DWord, located on a DWord boundary. For example, ID0[4] will return ID0, ID4, ID8, and ID12.

4. For all arrays, the total number of bytes requested cannot exceed the data portion of the negotiated PDU size. For example, for a 960-byte PDU size, the largest single array that may be read or written is 932 bytes. If arrays exceed the negotiated PDU size, they may fail to be read or written.

#### Note: The offset for an atomic type tag in a data block is denoted by the column "Address" in Step 7, as shown above. This offset is denoted by the column "Offset" in the Siemens TIA Portal programming environment.

### S7-300 Address Descriptions

#### Standard Support

www.ptc.com
Siemens TCP/IP Ethernet Driver

**S7-300/400/1200/1500 Item Syntax**

**Internal Tags**

**Third-Party Support**
For users familiar with third-party applications, limited addressing support is available.

**Legacy Support**
**Legacy S7-300/400 Item Syntax**

All brand and product names are trademarks, registered trademarks, or service marks of their respective holders.

**S7-400 Address Descriptions**

**Standard Support**
**S7-300/400/1200/1500 Item Syntax**

**Internal Tags**

**Third-Party Support**
For users familiar with third-party applications, limited addressing support is available.

**Legacy Support**
**Legacy S7-300/400 Item Syntax**

All brand and product names are trademarks, registered trademarks, or service marks of their respective holders.

**S7-1200 Address Descriptions**

**Standard Support**
**S7-300/400/1200/1500 Item Syntax**

**Internal Tags**

**Third-Party Support**
For users familiar with third-party applications, limited addressing support is available.

**Legacy Support**
**Legacy S7-300/400 Item Syntax**

All brand and product names are trademarks, registered trademarks, or service marks of their respective holders.

**S7-1500 Address Descriptions**

**Standard Support**
**S7-300/400/1200/1500 Item Syntax**

**Internal Tags**

**Third-Party Support**
For users familiar with third-party applications, limited addressing support is available.

**Legacy Support**
Legacy S7-300/400 Item Syntax

All brand and product names are trademarks, registered trademarks, or service marks of their respective holders.

NetLink: S7-300 Address Descriptions

Standard Support
S7-300/400/1200/1500 Item Syntax

Third-Party Support
For users familiar with third-party applications, limited addressing support is available.

Legacy Support
Legacy S7-300/400 Item Syntax

All brand and product names are trademarks, registered trademarks, or service marks of their respective holders

NetLink: S7-400 Address Descriptions

Standard Support
S7-300/400/1200/1500 Item Syntax

Third-Party Support
For users familiar with third-party applications, limited addressing support is available.

Legacy Support
Legacy S7-300/400 Item Syntax

All brand and product names are trademarks, registered trademarks, or service marks of their respective holders

Internal Tags

Although the following internal tags are not visible in the server configuration, they can be browsed by the OPC client. They can be found under the `<Channel Name>`.<Device Name>._InternalTags group. If the OPC client does not support browsing, or if a non-OPC client is being used, the tags can be created dynamically and statically by using the addresses given below.

**Note:** The tags listed in the following table are valid for the S7-300, S7-400, S7-1200, and S7-1500 device models. The default data types are shown in **bold**.

<table>
<thead>
<tr>
<th>Device Address</th>
<th>Description</th>
<th>Range</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>_RACK</td>
<td>Number of the rack in which the CPU of interest resides. On changing this device property, the connection with the CPU is re-established.</td>
<td>0-7</td>
<td>Byte, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td>_SLOT</td>
<td>Number of the slot in which the CPU of interest resides. On changing this device property, the connection with the CPU is re-established.</td>
<td>2-31</td>
<td>Byte, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td>_CurrentPDUSize</td>
<td>Subsequent to connection, this tag shows the size of the Protocol Data Unit which has been negotiated with the</td>
<td>240, 480,</td>
<td>Word</td>
<td>Read</td>
</tr>
</tbody>
</table>

www.ptc.com
See Also: Examples, String Support
**S7 Data Types**

The S7 data type is used to coerce the data type for a tag. It does not apply to Timers and Counters. The default data types are shown in **bold**.

<table>
<thead>
<tr>
<th>S7 Data Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Unsigned Byte</td>
<td>B0-B65535, BYTE0-BYTE65535</td>
<td><strong>Byte, Char</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>B0.b-B65535.b, BYTE0.b-</td>
<td>Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BYTE65535.b.b, .b is Bit Number 0-7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>B0.n-B65535.n, BYTE0.n-</td>
<td>String*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BYTE65535.n.n, .n is string length, 0 &lt; n &lt;= 932</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Signed Byte</td>
<td>C0-C65535, CHAR0-CHAR65535</td>
<td><strong>Byte, Char</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>C0.b-C65535.b, CHAR0.b-</td>
<td>Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR65535.b.b, .b is Bit Number 0-7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>C0.n-C65535.n, CHAR0.n-</td>
<td>String*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR65535.n.n, .n is string length, 0 &lt; n &lt;= 932</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Unsigned Double Word</td>
<td>D0-D65532, DWORD0-DWORD65532</td>
<td><strong>DWORD, Long, LBCD, Float</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D0.b-D65532.b, DWORD0.b-</td>
<td>Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DWORD65532.b.b, .b is Bit Number 0-31</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>S7 Date</td>
<td>DATE0-DATE65534</td>
<td><strong>String</strong></td>
</tr>
<tr>
<td></td>
<td>Stored as WORD in steps of 1 day since January 1, 1990.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S7 Data Type</td>
<td>Description</td>
<td>Address Range</td>
<td>Data Type</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>---------------</td>
<td>-----------</td>
</tr>
<tr>
<td>DI DINT</td>
<td>Signed Double Word</td>
<td>DIO-DI65532 DINT0-DINT65532</td>
<td>DWord, Long, LBCD, Float</td>
</tr>
<tr>
<td>DT</td>
<td>S7 Date_And_Time</td>
<td>0 year, 1 month, 2 days, 3 hours, 4 minutes, 5 seconds, 6 two most significant digits of MSEC, 7 (4MSB) two least significant digits of MSEC, 7 (4LSB) day of week (1=Sunday). Displayed as string format &quot;m/d/y h:mm:ss &lt;AM/PM&gt;&quot; with range &quot;1/1/1990 0:00:00 AM&quot; to &quot;12/31/2089 23:59:59 PM&quot;. Displayed as date format &quot;yyyy-mm-ddThh:mm:ss.hhh&quot; with range &quot;1990-01-01T00:00:00.000&quot; to &quot;2089-12-31T23:59:59.998&quot;.</td>
<td>DT0-DT65528</td>
</tr>
<tr>
<td>I INT</td>
<td>Signed Word</td>
<td>I0-I65534 INT0-INT65534</td>
<td>Word, Short, BCD</td>
</tr>
<tr>
<td>REAL</td>
<td>IEEE Float</td>
<td>REAL0-REAL65532</td>
<td>Float</td>
</tr>
<tr>
<td>String</td>
<td>S7 String</td>
<td>STRING0.n-STRING65532.n .n is string length 0&lt;n&lt; 254 (PDU size of 480 and above) 0&lt;n&lt; 210 (PDU size of 480 and above)</td>
<td>String</td>
</tr>
<tr>
<td>S7 Data Type</td>
<td>Description</td>
<td>Address Range</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>T TIME</td>
<td>S7 TIME. Stored as DWORD in steps of milliseconds. Displayed as string format &quot;+/-ddD_hhH_mmM_ssS_hhhMS&quot; with range &quot;+24D_20H_31M_23S_648MS&quot; to &quot;24D_20H_31M_23S_647MS&quot;.</td>
<td>T0-T65532 TIME0-TIME65532 String</td>
<td></td>
</tr>
<tr>
<td>TOD</td>
<td>S7 Time_Of_Day. Stored as DWORD, representing milliseconds since midnight. Displayed as string format &quot;h:m:s.mmm&quot; with range &quot;0:0:0.0&quot; to &quot;23:59:59.999&quot;.</td>
<td>TOD0-TOD65532 String</td>
<td></td>
</tr>
<tr>
<td>W Word</td>
<td>Unsigned Word</td>
<td>W0-W65534 WORD0-WORD65534 Word, Short, BCD</td>
<td></td>
</tr>
<tr>
<td>X Bit</td>
<td></td>
<td>X0.b-X65534.b .b is Bit Number 0-15 Boolean</td>
<td></td>
</tr>
</tbody>
</table>

*These are raw strings that differ in structure and usage from the STEP 7 string data type.

Use caution when modifying Word, Short, DWord, and Long type as each address starts at a byte offset within the device. Therefore, Words MW0 and MW1 overlap at byte 1. Writing to MW0 will also modify the value held in MW1. Similarly, DWord, and Long types can also overlap. It is recommended that these memory types be used so that overlapping does not occur. For example, DWord MD0, MD4, MD8, and so on can be used to prevent overlapping bytes.

See Also: Examples

String Support
Raw Strings
For an address DBx,By.n @ string, string values read and written are stored at byte offset y.
Raw strings are null-terminated. If the maximum string length is 10 and 3 characters are written, the fourth character is set to NULL, while characters 5-10 are left untouched.

**Note:** For raw strings, the total number of bytes requested cannot exceed the data portion of the negotiated PDU size. If raw strings exceed the negotiated PDU size, they may fail to be read or written.

### String Support

The string subtype follows the STEP 7 string data type definition. The syntax for the string S7 data type is \( STRING_{y,n} \) where \( y \) is the Byte offset, and \( n \) is the maximum string length. If \( n \) is not specified, the maximum string length will be 254 characters when PDU size is \( >= 480 \), otherwise it will be 210. String values read and written are stored at byte offset \( y+2 \) in data block \( x \). The actual string length gets updated with every write based on the string length of the string being written.

<table>
<thead>
<tr>
<th>( y )</th>
<th>( y+1 )</th>
<th>( y+2 )</th>
<th>...</th>
<th>( y+\text{n}-1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>maximum string length (n)</th>
<th>actual string length</th>
<th>( .. )</th>
<th>( .. )</th>
<th>( .. )</th>
<th>...</th>
<th>( .. )</th>
</tr>
</thead>
</table>

**Notes:**

1. String strings are NULL padded. If the maximum string length is 10 and 3 characters are written, characters 4-10 are set to NULL.
2. If a PDU of 240 is negotiated, STEP 7 strings with a length greater than 222 may fail to be read and strings with a length greater than 212 may fail to be written.

### Hex Strings

The HEXSTRING subtype is specific to the Siemens TCP/IP Ethernet Driver. The syntax for the HEXSTRING subtype is \( HEXSTRING_{y,n} \), where \( y \) is the byte offset and \( n \) is the length. The \( n \) value must be specified in the range of 1 through 932. String is the only valid data type for a HEXSTRING tag.

The value assigned to a HEXSTRING must be an even number of characters. There is no padding, so the entire string must be specified. For example, tag HexStr defined as DB1,STRING0.10 uses 10 bytes of storage and has a display length of 20. To assign a value, the string must be 20 characters long and contain only valid hexadecimal characters. An example valid hex string for this tag is “56657273696f6E353137”.

### Array Support

The \([\text{rows}]\,[\text{cols}]\) notation is appended to an address to specify an array (such as MW0[2][5]). If no rows are specified, row count of 1 is assumed. Boolean arrays and string arrays are not supported.

For Word, Short, and BCD arrays, the base address + (rows * cols * 2) cannot exceed 65536. Keep in mind that the elements of the array are words, located on a word boundary. For example, IW0[4] would return IW0, IW2, IW4, and IW6.

For Float, DWord, Long, and Long BCD arrays, the base address + (rows * cols * 4) cannot exceed 65536. Keep in mind that the elements of the array are DWord, located on a DWord boundary. For example, ID0[4] will return ID0, ID4, ID8, ID12.

For all arrays, the total number of bytes requested cannot exceed the data portion of the negotiated PDU size. For example, for a 960-byte PDU size, the largest single array that may be read or written is 932 bytes. If arrays exceed the negotiated PDU size, they may fail to be read or written.
Timers
The Siemens TCP/IP Ethernet Driver automatically scales T values based on the Siemens S5 time format. Timer data is stored as a Word in the PLC but scaled to a DWord in the driver. The value returned will already be scaled using the appropriate Siemens time base. As a result, the values are always returned as a count of milliseconds. When writing to T memory, the Siemens time base will also be applied. To assign a value to a timer in the controller, write the desired value as a count of milliseconds to the appropriate timer.

Counters
The value returned for C memory will automatically be converted to a BCD value.

Examples

<table>
<thead>
<tr>
<th>S7 Data Type</th>
<th>Data Type</th>
<th>Input</th>
<th>Flags</th>
<th>Data Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Byte</td>
<td>IB0</td>
<td>MB0</td>
<td>DB1,B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBYTE0</td>
<td>MBYTE0</td>
<td>DB1,BYTE0</td>
</tr>
<tr>
<td></td>
<td>Boolean</td>
<td>IB0.7</td>
<td>MB0.7</td>
<td>DB1,B0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBYTE0.7</td>
<td>MBYTE0.7</td>
<td>DB1,BYTE0.7</td>
</tr>
<tr>
<td></td>
<td>String</td>
<td>IB0.64</td>
<td>MB0.64</td>
<td>DB1,B0.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBYTE0.64</td>
<td>MBYTE0.64</td>
<td>DB1,BYTE0.64</td>
</tr>
<tr>
<td></td>
<td>Array</td>
<td>IB0[2][5]</td>
<td>MB0[2][5]</td>
<td>DB1,B0[2][5]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBYTE0[2][5]</td>
<td>MBYTE0[2][5]</td>
<td>DB1,BYTE0[2][5]</td>
</tr>
<tr>
<td>C</td>
<td>Char</td>
<td>IC0</td>
<td>MC0</td>
<td>DB1,C0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ICHAR0</td>
<td>MCHAR0</td>
<td>DB1,CHAR0</td>
</tr>
<tr>
<td></td>
<td>Boolean</td>
<td>IC0.7</td>
<td>MC0.7</td>
<td>DB1,C0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ICHAR0.7</td>
<td>MCHAR0.7</td>
<td>DB1,CHAR0.7</td>
</tr>
<tr>
<td></td>
<td>String</td>
<td>IC0.64</td>
<td>MC0.64</td>
<td>DB1,C0.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ICHAR0.64</td>
<td>MCHAR0.64</td>
<td>DB1,CHAR0.64</td>
</tr>
<tr>
<td></td>
<td>Array</td>
<td>IC0[10]</td>
<td>MC0[10]</td>
<td>DB1,C0[10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ICHAR0[10]</td>
<td>MCHAR0[10]</td>
<td>DB1,CHAR0[10]</td>
</tr>
<tr>
<td></td>
<td>DWord</td>
<td>ID0</td>
<td>MD0</td>
<td>DB1,D0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDWORD0</td>
<td>MDWORD0</td>
<td>DB1,DWORD0</td>
</tr>
<tr>
<td></td>
<td>Boolean</td>
<td>ID0.31</td>
<td>MD0.31</td>
<td>DB1,D0.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDWORD0.31</td>
<td>MDWORD0.31</td>
<td>DB1,DWORD0.31</td>
</tr>
<tr>
<td></td>
<td>Array</td>
<td>ID0[10]</td>
<td>MD0[10]</td>
<td>DB1,D0[10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDWORD0[10]</td>
<td>MDWORD0[10]</td>
<td>DB1,DWORD0[10]</td>
</tr>
<tr>
<td>DATE</td>
<td>String</td>
<td>IDATE0</td>
<td>MDATE0</td>
<td>DB1,DATE0</td>
</tr>
<tr>
<td>DI</td>
<td>Long</td>
<td>IDI0</td>
<td>MDI0</td>
<td>DB1,DI0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDINT0</td>
<td>MDINT0</td>
<td>DB1,DINT0</td>
</tr>
<tr>
<td></td>
<td>Boolean</td>
<td>IDI0.31</td>
<td>MDI0.31</td>
<td>DB1,DI0.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDINT0.31</td>
<td>MDINT0.31</td>
<td>DB1,DINT0.31</td>
</tr>
<tr>
<td>S7 Data Type</td>
<td>Data Type</td>
<td>Input</td>
<td>Flags</td>
<td>Data Blocks</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>-------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>DT</td>
<td>IDT0</td>
<td>MDT0</td>
<td>DBT0</td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td>I[0]</td>
<td>M[15]</td>
<td>DB[0]</td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>IREAL0</td>
<td>MREAL0</td>
<td>DBREAL0</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>ISTRING0.10</td>
<td>MSTRING0.10</td>
<td>DBSTRING0.10</td>
<td></td>
</tr>
<tr>
<td>TOD</td>
<td>ITOD0</td>
<td>MTOD0</td>
<td>DBTOD0</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>ITIME4</td>
<td>MTIME4</td>
<td>DBTIME4</td>
<td></td>
</tr>
<tr>
<td>Word</td>
<td>IW[0]</td>
<td>MW[0]</td>
<td>DB[0]</td>
<td></td>
</tr>
<tr>
<td>Boolean</td>
<td>IW[0.15]</td>
<td>MW[0.15]</td>
<td>DB[0.15]</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>IX[0.7]</td>
<td>MX[0.7]</td>
<td>DB[0.7]</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The offset for an atomic type tag in a data block is denoted by the column "Address" in Step 7, as shown above. This offset is denoted by the column "Offset" in the Siemens TIA Portal programming environment.

### Legacy S7-300/400 Item Syntax

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

*For preferred item syntax, refer to Standard S7-300/400/1200/1500 Item Syntax.*

<table>
<thead>
<tr>
<th>Address Type</th>
<th>Range</th>
<th>Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Inputs</td>
<td>I0.b:65535.b</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>.b is Bit Number 0-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IB0:IB65535</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IW0-IW65534</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IW:KT0-IW:KT65534</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IW:KC0-IW:KC65534</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Byte, Char, String**</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Word, Short, BCD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DWord, Long Word</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Short, Long</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Read/Write</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Read/Write</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Read/Write</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Read/Write</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

www.ptc.com
<table>
<thead>
<tr>
<th>Address Type</th>
<th>Range</th>
<th>Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Inputs</td>
<td>ID0-ID65532</td>
<td>DWord, Long, LBCD, Float</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>E0.b-E65535.b .b is Bit Number 0-7</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>EB0-EB65535** EW0-EW65534 EW:kT0-EW:kT65534 EW:KC0-EW:KC65534 ED0-ED65532</td>
<td>Byte, Char, String** Word, Short, BCD DWord, Long Word, Short DWord, Long, LBCD, Float</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>Note: I and E access the same memory area.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete Outputs</td>
<td>Q0.b-Q65535.b .b is Bit Number 0-7</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>QB0-QB65535(QW0-QW65534 QW:kT0-QW:kT65534 QW:KC0-QW:KC65534 QD0-QD65532)</td>
<td>Byte, Char, String** Word, Short, BCD DWord, Long Word, Short DWord, Long, LBCD, Float</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>Note: Q and A access the same memory area.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete Outputs</td>
<td>A0.b-A65535.b .b is Bit Number 0-7</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Peripheral Inputs</td>
<td>PI0.b-PI65535.b .b is Bit Number 0-7</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td></td>
<td>PIB0-PIB65535 PIW0-PIW65534 PIW:kT0-PIW:kT65534 PIW:KC0-PIW:KC65534 PID0-PID65532</td>
<td>Byte, Char, String** Word, Short, BCD DWord, Long Word, Short DWord, Long, LBCD, Float</td>
<td>Read Only</td>
</tr>
<tr>
<td>Peripheral Inputs</td>
<td>PE0.b-PE65535.b .b is Bit Number 0-7</td>
<td>Boolean</td>
<td>Read Only</td>
</tr>
<tr>
<td></td>
<td>PEB0-PEB65535** PEW0-PEW65534 PEW:kT0-PEW:kT65534 PEW:KC0-PEW:KC65534</td>
<td>Byte, Char, String** Word, Short, BCD DWord, Long Word, Short</td>
<td>Read Only</td>
</tr>
</tbody>
</table>

www.ptc.com
<table>
<thead>
<tr>
<th>Address Type</th>
<th>Range</th>
<th>Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PED0-PED65532</td>
<td>DWord, Long, LBCD, Float</td>
<td>Read Only</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: PI and PE access the same memory area.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peripheral Outputs</td>
<td>PQ0.b-PQ65535.b .b is Bit Number 0-7</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>PQB0-PQ65535</td>
<td>Byte, Char, String**</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>PQW:KT0-PQW:KT65534</td>
<td>Word, Short, BCD</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>PQW:KC0-PQW:KC65534</td>
<td>DWord, Long</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>PQD0-PQD65532</td>
<td>Word</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>PA0.b-PA65535.b .b is Bit Number 0-7</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>PAB0-PAB65535</td>
<td>Byte, Char, String**</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>PAW:KT0-PAW:KT65534</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>PAW:KC0-PAW:KC65534</td>
<td>DWord, Long</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>PAD0-PAD65532</td>
<td>Word</td>
<td>Read/Write</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: PQ and PA access the same memory area.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal Memory</td>
<td>F0.b-F65535.b .b is Bit Number 0-7</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>FB0-FB65535</td>
<td>Byte, Char, String**</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>FW0-FW65534</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>FW:KT0-FW:KT65534</td>
<td>DWord, Long</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>FW:KC0-FW:KC65534</td>
<td>Word</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>FD0-FD65532</td>
<td>DWord, Long, LBCD, Float</td>
<td>Read/Write</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: F and M access the same memory area.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal Memory</td>
<td>M0.b-M65535.b .b is Bit Number 0-7</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>MB0-MB65535</td>
<td>Byte, Char, String**</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>MW0-MW65534</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>MW:KT0-MW:KT65534</td>
<td>DWord, Long</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>MW:KC0-MW:KC65534</td>
<td>Word</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>MD0-MD65532</td>
<td>DWord, Long, LBCD, Float</td>
<td>Read/Write</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: F and M access the same memory area.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Block Boolean</td>
<td>DB1:N:KM0.b-KM65534.b 1-N is Block Number .b is Bit Number 0-15</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

www.ptc.com
<table>
<thead>
<tr>
<th>Address Type</th>
<th>Range</th>
<th>Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Block Left Byte</td>
<td>DB1DBX0.b-DBNDBX65534.b 1-N is Block Number</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>.b is Bit Number 0-15</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB1D0.b-DBND65534.b 1-N is Block Number</td>
<td>Boolean</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>.b is Bit Number 0-15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Block Right Byte</td>
<td>DB1-N:KL0-KL65535 1-N is Block Number</td>
<td>Byte, Char, String**</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>Alternates</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB1DBB0-DBNDBB65535 1-N is Block Number</td>
<td>Byte, Char, String**</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>DB1DL0-DBNDL65535 1-N is Block Number</td>
<td>Byte, Char, String**</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Data Block Unsigned Word</td>
<td>DB1-N:KH0-KH65534 1-N is Block Number</td>
<td>Word, Short, BCD</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>Alternates</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB1DR0-DBNDR65534 1-N is Block Number</td>
<td>Byte, Char, String**</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Data Block Signed Word</td>
<td>DB1-N:KF0-KF65534 1-N is Block Number</td>
<td>Word, Short, BCD</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>Alternates</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB1DBW0-DBNDBW65534 1-N is Block Number</td>
<td>Word, Short, BCD</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>DB1DW0-DBNDW65534 1-N is Block Number</td>
<td>Word, Short, BCD</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Data Block Signed Long</td>
<td>DB1-N:KD0-KD65532 1-N is Block Number</td>
<td>DWord, Long, BCD, Float</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>Alternates</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB1DBD0-DB1DBD65532 1-N is Block Number</td>
<td>DWord, Long, BCD, Float</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>DB1DD0-DB1DD65532 1-N is Block Number</td>
<td>DWord, Long, BCD, Float</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Address Type</td>
<td>Range</td>
<td>Type</td>
<td>Access</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------</td>
<td>---------------</td>
<td>------------</td>
</tr>
<tr>
<td>Data Block Float</td>
<td>DB1-N:KG0-KG65532 1-N is Block Number</td>
<td>Float</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Data Block BCD</td>
<td>DB1-N:BCD0-BCD65534 1-N is Block Number</td>
<td>Word, Short, BCD</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Data Block S5 Timer as DB</td>
<td>DB1-N:KT0-KT65534 1-N is Block Number</td>
<td>DWord, Long</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Data Block S5 Counter as DB</td>
<td>DB1-N:KC0-KC65534 1-N is Block Number</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Data Block String***</td>
<td>DB1S0.n-DB1S65535.n* .n is string length 0&lt;n&lt;= 932</td>
<td>String</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Data Block String***</td>
<td>DB1STRING0.n-DB1STRING65535.n* .n is string length 0&lt;n&lt;= 254 (PDU size of 480 and above) 0&lt;n&lt;= 210 (PDU size below 480) 0&lt;n&lt;= 254 (Netlink S7300 and Netlink S7400 Models)</td>
<td>String</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Timer Current Values***</td>
<td>T0-T65535*</td>
<td>DWord, Long</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Counter Current Values*****</td>
<td>C0-C65535*</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Counter Current Values*****</td>
<td>Z0-Z65535*</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
</tbody>
</table>

*These memory types/subtypes do not support arrays.
**Byte memory types (like MB) support Strings. The syntax for strings is <address>.<length> where 0 < length <=932.
***For more information, refer to Data Block Strings.
****For more information, refer to Timers.
*****For more information, refer to Counters.

**Notes:**

1. All offsets for memory types I, Q, and F represent a byte starting location within the specified memory type.

2. Use caution when modifying Word, Short, DWord, and Long types. For I, Q, and F each address starts at a byte offset within the device. Therefore, Words FW0 and FW1 overlap at byte 1. Writing to FW0 will also modify the value held in FW1. Similarly, DWord, and Long types can also overlap. It is recommended that these memory types be used so that overlapping does not occur. For example, DWord, FD0, FD4, FD8 and so on can be used to prevent overlapping bytes.

**Data Block Strings**

Data block Strings can be referenced by using S subtypes or String subtypes.

**S Subtype**

The syntax for the S subtype is DBxSy.n where x is the data block, y is the byte offset, and n is the maximum String length. String values read and written are stored at byte offset y in data block x.
S Strings are null terminated. If the maximum string length is 10 and 3 characters are written, the fourth character is set to NULL, while characters 5-10 are left untouched.

★ **Note:** For raw strings the total number of bytes requested cannot exceed the data portion of the negotiated PDU size. If raw strings exceed the negotiated PDU size, they may fail to be read or written.

### String Subtype

The String subtype follows the STEP 7 String data type definition. The syntax for the String subtype is DBx.STRINGy.n, where x is the data block, y is the Byte offset, and n is the maximum String length. If n is not specified, the maximum String length will be 254 characters when PDU size is >= 480, otherwise it will be 210. String values read and written are stored at Byte offset y+2 in data block x. The first two bytes contain the maximum string length (n) and the actual string length. The actual string length gets updated with every write based on the string length of the string being written.

<table>
<thead>
<tr>
<th>y</th>
<th>y+1</th>
<th>y+2</th>
<th>...</th>
<th>y+n-1</th>
</tr>
</thead>
</table>

**Notes:**

1. String Strings are NULL padded. If the maximum string length is 10 and 3 characters are written, characters 4-10 are set to NULL.
2. If a PDU of 240 is negotiated, STEP 7 strings with a length greater than 222 may fail to be read and strings with a length greater than 212 may fail to be written.

### Hex Strings

The HEXSTRING subtype is specific to the Siemens TCP/IP Ethernet Driver. The syntax for the HEXSTRING subtype is HEXSTRINGy.n, where y is the byte offset and n is the length. The n value must be specified in the range of 1 through 932. String is the only valid data type for a HEXSTRING tag.

The value assigned to a HEXSTRING must be an even number of characters. There is no padding, so the entire string must be specified. For example, tag HexStr defined as DB1,STRINGO.10 uses 10 bytes of storage and has a display length of 20. To assign a value, the string must be 20 characters long and contain only valid hexadecimal characters. An example valid hex string for this tag is “56657273696f6e353137”.

★ **Note:** For HEXSTRINGs, the total number of bytes requested cannot exceed the data portion of the negotiated PDU size. If raw strings exceed the negotiated PDU size, they may fail to be read or written.

### Arrays

All memory types/subtypes with the exception of those marked with an asterisk support arrays. The syntax below are valid for declaring an array. If no rows are specified, a row count of 1 is assumed.

```
<address>[rows][cols]
<address>.rows.cols
<address>,rows.cols
<address>_rows cols
```

For Word, Short, BCD and "KT" arrays, the base address + (rows * cols * 2) cannot exceed 65536. Keep in mind that the elements of the array are words, located on a word boundary. For example, IW0[4] would
return IW0, IW2, IW4, and IW6. "KT" subtypes fall into the 16-bit category because the data stored in the PLC is contained within a Word. For more information, refer to Timers.

For Float, DWord, Long, and Long BCD arrays (excluding "KT" subtypes), the base address + (rows * cols * 4) cannot exceed 65536. Keep in mind that the elements of the array are DWord, located on a DWord boundary. For example, ID0[4] will return ID0, ID4, ID8, ID12.

For all arrays, the total number of bytes being requested cannot exceed the data portion of the negotiated PDU size. For example, for a 960-byte PDU size, the largest single array that may be read or written is 932 bytes. If arrays exceed the negotiated PDU size, they may fail to be read or written.

KL vs. KR vs. DBB
KL and KR determine whether the left byte or right byte of the data block word is returned.

<table>
<thead>
<tr>
<th>Value</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

The following examples are from the table above.

**Example 1**

DB1:KH0=0x89  
DB1:KL0=0x8  
DB1:KR0=0x9  
DB1:DBB0=0x8

**Example 2**

DB1:KH1=0x9A  
DB1:KL1=0x9  
DB1:KR1=0xA  
DB1:DBB1=0x9

**Timers**

The Siemens TCP/IP Ethernet Driver automatically scales T and KT values based on the Siemens S5 time format. Timer data is stored as a Word in the PLC but scaled to a DWord in the driver. The value returned for either a T or KT memory type will already be scaled using the appropriate Siemens time base. As a result, the values are always returned as a count of milliseconds. When writing to T or KT memory types, the Siemens time base will also be applied. To assign a value to a timer in the controller, write the desired value as a count of milliseconds to the appropriate timer.

**Counters**

The value returned for either C or KC memory type will automatically be converted to a BCD value. DB1:KH0 @ BCD=DB1:KC0 @ Word.

**Examples**

- To access bit 3 of internal memory F20, declare an address as follows: F20.3  
- To access data block 5 as word memory at byte 30, declare an address as follows: DB5:KH30  
- To access data block 2 byte 20 and bit 7, declare an address as follows: DB2:KM20.7  
- To access data block 1 as left byte memory at byte 10, declare an address as follows: DB1:KL10
To access internal memory F20 as a DWORD, declare an address as follows: FD20
To access Input memory I10 as a Word, declare an address as follows: IW10

**Note:** The offset for an atomic type tag in a data block is denoted by the column "Address" in Step 7, as shown above. This offset is denoted by the column "Offset" in the Siemens TIA Portal programming environment.
Event Log Messages

The following information concerns messages posted to the Event Log pane in the main user interface. Consult the 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.

**Reason = 'Frame contains errors'.**

**Error Type:**
Warning

**Possible Cause:**
1. An unexpected frame was received. The response code may be incorrect.
2. The frame sequence is out of order.

**Possible Solution:**
Cable noise may cause distortion in the frame, resulting in erroneous data or dropped frames. Verify the cabling between the PC and the PLC device.

● See Also:
1. Error Matrix
2. Error Codes

**Reason = 'Device returned transport error'. Error code = <error>.**

**Error Type:**
Warning

**Possible Cause:**
An RFC1006 (ISO over TCP/IP) error occurred. This is the portion of the packet that encapsulates the S7 Messaging packet.

**Possible Solution:**
Follow guidance from the rest of the error message or contact Technical Support.

● Note:
No protocol or data access errors can occur for this operation.

● See Also:
1. Error Matrix
2. Error Codes
Reason = 'Device returned protocol error'. Error class = <class>, Error code = <error>.

Error Type:
Warning

Possible Cause:
1. An S7 Messaging error occurred. This can occur if a portion is malformed or contains incorrect packet lengths.
2. The tag is too large to be read or written with the currently negotiated PDU.

Possible Solution:
1. Follow guidance from the rest of the error message or contact Technical Support.
2. Adjust the size of the tag, or check the Maximum PDU Size device property.

See Also:
1. Error Matrix
2. Error Codes
3. Device Properties – S7 Comm Parameters

Reason = 'Device returned data access error'. Error code = <error>.

Error Type:
Warning

Possible Cause:
A requested address may be out of range or referenced incorrectly.

Possible Solution:
1. Verify the range is correct and correctly referenced.
2. Follow guidance from the rest of the error message or contact Technical Support.

See Also:
1. Error Matrix
2. Error Codes

Reason = 'Device is not responding'.

Error Type:
Warning

Possible Cause:
1. The connection between the device and the host PC is invalid.
2. The named device may have an incorrect IP address assigned.
3. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device setting.
4. The device CPU load is too high.

**Possible Solution:**

1. Verify the cabling between the PC and the PLC device.
2. Verify the IP address for the named device matches the actual device.
3. Decrease the tag group scan rate to reduce the load on the PLC CPU.
4. Increase the values for properties: Request Timeout, Scan Cycle Load from Communication, and/or Scan Cycle Monitoring Time.

See Also:
Error Matrix

**Reason = 'Unknown error occurred'.**

**Error Type:**
Warning

**Possible Cause:**
Process could not complete.

**Possible Solution:**
Follow guidance from the rest of the error message or retry the process.

See Also:
Error Matrix

**Reason = NetLink returned error. Error code = <error>**.

**Error Type:**
Warning

**Possible Cause:**
An error was returned from the PLC or NetLink adapter.

**Possible Solution:**

1. If error code is 0x11, an incorrect MPI ID may be set. Determine the MPI ID through which communications are occurring and enter it in the MPI ID device property field.
2. If error code is 0x87, the requested data may be out of range for the device. Verify the device address limits and correct the tag references.
See Also:
1. Error Matrix
2. Error Codes

Failed to resolve host. | Host = '<host name>'.

Error Type:
Warning

Possible Cause:
1. The named device may have been assigned an incorrect IP address.
2. Communication with the host failed. Connection may have been lost, a port conflict occurred, or some communication parameter is not valid.

Possible Solution:
1. Verify the IP address given to the named device matches that of the actual device.
2. Verify or correct connections, port number, MPI ID, and other communication parameters.

Auto-generated tag names and descriptions may not appear as expected due to string conversion error.

Error Type:
Warning

Possible Cause:
The Unicode character conversion failed.

Possible Solution:
Verify that the Step 7 language file is present and reflects a character set that can display the Step 7 tag and comment strings.

A required code page is unavailable on this machine. Tag generation may fail or tag names and descriptions may not appear as expected. | Required code page = <page>.

Error Type:
Warning

Possible Cause:
1. The computer is not configured with support for the specified Windows code page specified.
2. The language file was not found.

Possible Solution:
1. Install any language packs necessary for displaying the Windows code page character set.

2. Try again on a computer that supports the Windows code page specified.

3. Modify the Step 7 project to generate a Language file.

**Note:**
When Language file was not found the value of required code page will be: 0.

**Unable to load the Step 7 language file.**

**Error Type:**
Warning

**Possible Cause:**
The Step 7 language file is altered or corrupt.

**Possible Solution:**
Verify that the Step 7 project is not corrupt and can be opened in Simatic Step 7.

**Memory exception reading the Step 7 language file.**

**Error Type:**
Warning

**Possible Cause:**
The operating system has insufficient memory to read the Step 7 language file.

**Possible Solution:**
Ensure that the system resources are adequate for all applications running on the computer.

**Step 7 language file failed to open. | OS error = '<error>'.**

**Error Type:**
Warning

**Possible Cause:**

1. The Step 7 language file is altered or corrupt.

2. The language file is missing.

**Possible Solution:**

1. Verify that the Step 7 project is not corrupt and can be opened in Simatic Step 7.

2. Modify the Step 7 project to generate a language file.
Tag generation failure. | Data block name = '<block name>', data block number = <block number>.

Error Type:
Warning

Possible Cause:
An unexpected data type or other issue occurred during the parsing of the Step 7 project for the specified data block.

Possible Solution:
Compare the tags that were automatically generated with those in the project for the specified data block to determine which tag caused the incomplete generation. Correct issues with the block and retry.

See Also:
Error Codes

Created tag in group due to internal block size. | Tag address = '<address>', tag name = '<name>', group name = '<name>'.

Error Type:
Warning

Possible Cause:
While parsing the data blocks of the Step 7 project for automatic tag generation, an array variable was encountered that exceeds the internal block size. Although all individual array element tags generate as expected, the array tag itself is generated with a dimension that allows it to fit within the block size.

Possible Solution:
To use array tags and not the individual array element tags, determine the address where the array tag ends, then manually generate another tag to address the remainder of the array. For example, if data block 1 begins with an array of 250 REAL, there would be 250 array element tags with addresses DB1,REAL0; DB1,REAL4;... DB1,REAL992; DB1,REAL996. Because the size of the array exceeds the maximum data payload of 932 bytes, the array tag would only be created with 233 dimensions (DB1,REAL0[233]). The array tag does not provide the client with the data for the last 17 elements. If the client wants to use array tags and not the individual array element tags, another tag with the address "DB1,REAL932[17]" must be created. This warning message only occurs for tags of the first element of the complex type array during automatic tag generation for arrays of complex types (such as structures, user-defined types, function blocks, or system function blocks).

Tag not created because arrays are not supported with specified data type. | Tag name = '<name>', group name = '<name>', data type = '<<type>>'.

Error Type:
Warning

Possible Cause:
1. A tag address that has been specified dynamically has been assigned an invalid data type.

2. While parsing the data blocks of the Step 7 project for automatic tag generation, an array variable was encountered with a data type for which the driver does not support arrays.

Possible Solution:

1. Modify the requested data type in the client application.

2. The client must access the data using the array element tags that were generated. Variables with the Step 7 data types of DATE, DATE_AND_TIME, STRING, TiME, and TIME_OF_DAY generate tags with the string data type (for which arrays are not supported). During automatic tag generation for arrays of complex types (such as structures, user-defined types, function blocks, or system function blocks), this warning message only occurs for tags of the first element of the complex type array.

Unable to connect to device. |

Error Type:
Warning

Possible Cause:

1. An RFC1006 error (ISO over TCP/IP) occurred. This is the portion of the packet that encapsulates the S7 Messaging packet.

2. The device's CPU work load is too high.

3. This portion is malformed or contains incorrect packet lengths.

Possible Solution:

1. Cable noise may cause distortion in the frame, resulting in erroneous data or dropped frames. Verify the cabling between the PC and the PLC device.

2. Reduce network traffic or increase the Request Timeout and/or Fail After Attempt count.

3. Decrease the tag group Scan Rate to reduce the load on the PLC CPU.

4. Increase the values for properties: Scan Cycle Load from Communication and Scan Cycle Monitoring Time.

See Also:
Error Matrix

Unable to establish association with device. |

Error Type:
Warning

Possible Cause: 
1. An S7 Messaging error occurred. This will occur if this portion is malformed or contains incorrect packet lengths.
2. An RFC1006 error (ISO over TCP/IP) occurred. This is the portion of the packet that encapsulates the S7 Messaging packet.
3. The TPDU response size is incorrect.
4. An unexpected frame was received. The response code may be incorrect.
5. The frame sequence is out of order.
6. The device CPU workload is too high.

Possible Solution:

1. Cable noise may cause distortion in the frame, resulting in erroneous data. It may also cause dropped frames. Verify the cabling between the PC and the PLC device.
2. Reduce network traffic. If this error occurs frequently, increase the Request Timeout and/or Fail After attempt count.
3. If this error occurs frequently, decrease the tag group scan rate to reduce the work load on the PLC’s CPU.
4. Increase the Scan Cycle Load from Communication and Scan Cycle Monitoring Time.

See Also:

Error Matrix

Unable to read from address on device. | Address = '<address>',

Error Type:

Warning

Possible Cause:

1. A data access error occurred. The requested address may be out of range or referenced incorrectly.
2. An S7 messaging error occurred. A portion is malformed or contains incorrect packet lengths.
3. A TCP/IP error occurred. A portion is malformed or contains incorrect packet lengths.
4. An attempt was made to read an array larger than the PDU size negotiated with the device.

Possible Solution:

1. Verify and correct the address range.
2. Verify and correct the packet format and length.
3. Verify and correct the communications configuration and connections.
4. Verify and correct the data type, values, and ranges.
5. Verify the device’s address limits and correct the tag references causing the error.
Unable to read from address on device. Tag deactivated. | Address = '<address>',

Error Type:
Warning

Possible Cause:
1. A data access error occurred. The requested address may be out of range or referenced incorrectly.
2. An S7 Messaging error occurred. A portion is malformed or contains incorrect packet lengths.
3. A TCP/IP error occurred. A portion is malformed or contains incorrect packet lengths.
4. The device CPU work load is too high.
5. If the tag address references a TOD data type, the DWORD value may be larger than the number of milliseconds in a day. For example, 86400000.
6. If the error code=0x11, an incorrect MPI ID may be set.
7. If the error code=0x87, users may be accessing data out of range in the device.

Possible Solution:
1. Verify and correct the address range.
2. Verify and correct the packet format and length.
3. Verify and correct the communications configuration and connections.
4. Verify and correct the data type, values, and ranges.
5. Reduce network traffic or increase the Request Timeout and/or Fail After attempt count.
6. Decrease the tag group scan rate to reduce the workload on the PLC CPU.
7. Increase the Scan Cycle Load from Communication and Scan Cycle Monitoring Time.
8. Change the value in the device to a valid DWORD that can be converted to a time that is less than or equal to 23:59:59.999.
9. Determine the MPI ID in use for communications and re-enter it in the MPI ID Device Property field.
10. Verify the device's address limits and correct the tag references causing the error.

Unable to read data from device. | Data block = '<block>', block start = <address>, block size = <size>,

Error Type:
Warning

Possible Cause:

1. A TCP/IP error occurred. A portion is malformed or contains incorrect packet lengths.
2. The device CPU work load is too high.
3. If the tag address references a TOD data type, the DWORD value may be larger than the number of milliseconds in a day. For example, 86400000.
4. An error was returned from the PLC or NetLink adapter.
5. Cable noise may cause distortion in the frame, resulting in erroneous data or dropped frames. Verify the cabling between the PC and the PLC device.
6. If the error code=0x11, an incorrect MPI ID may be set.
7. If the error code=0x87, users may be accessing data out of range in the device.

Possible Solution:

1. Verify and correct the address range.
2. Verify and correct the packet format and length.
3. Verify and correct the communications configuration and connections.
4. Verify and correct the data type, values, and ranges.
5. Reduce network traffic or increase the Request Timeout and/or Fail After attempt count.
6. Decrease the tag group scan rate to reduce the workload on the PLC CPU.
7. Increase the Scan Cycle Load from Communication and Scan Cycle Monitoring Time.
8. Change the value in the device to a valid DWORD that can be converted to a time that is less than or equal to 23:59:59.999.
9. Determine the MPI ID in use for communications and re-enter it in the MPI ID Device Property field.
10. Verify the device’s address limits and correct the tag references causing the error.

See Also:
Error Matrix

Unable to read data from device. Block deactivated. | Data block = 'block>', block start = <address>, block size = <size>,

Error Type:
Warning

Possible Cause:

1. A TCP/IP error occurred. A portion is malformed or contains incorrect packet lengths.
2. The device CPU work load is too high.
3. If the tag address references a TOD data type, the DWORD value may be larger than the number of milliseconds in a day. For example, 86400000.

4. An error was returned from the PLC or NetLink adapter.

5. Cable noise may cause distortion in the frame, resulting in erroneous data or dropped frames. Verify the cabling between the PC and the PLC device.

6. If the error code=0x11, an incorrect MPI ID may be set.

7. If the error code=0x87, users may be accessing data out of range in the device.

**Possible Solution:**

1. Verify and correct the address range.

2. Verify and correct the packet format and length.

3. Verify and correct the communications configuration and connections.

4. Verify and correct the data type, values, and ranges.

5. Reduce network traffic or increase the Request Timeout and/or Fail After attempt count.

6. Decrease the tag group scan rate to reduce the workload on the PLC CPU.

7. Increase the Scan Cycle Load from Communication and Scan Cycle Monitoring Time.

8. Change the value in the device to a valid DWORD that can be converted to a time that is less than or equal to 23:59:59.999.

9. Determine the MPI ID in use for communications and re-enter it in the MPI ID Device Property field.

10. Verify the device's address limits and correct the tag references causing the error.

**See Also:**

Error Matrix

**Unable to read data from device. | Memory type = '<type>', block start = <address>, block size = <size> (bytes),**

**Error Type:**

Warning

**Possible Cause:**

1. An S7 Messaging error occurred. A portion is malformed or contains incorrect packet lengths.

2. The device CPU work load is too high.

3. If the tag address references a TOD data type, the DWORD value may be larger than the number of milliseconds in a day. For example, 86400000.

4. An error was returned from the PLC or NetLink adapter.
5. Cable noise may cause distortion in the frame, resulting in erroneous data or dropped frames. Verify the cabling between the PC and the PLC device.

6. If the error code=0x11, an incorrect MPI ID may be set.

7. If the error code=0x87, users may be accessing data out of range in the device.

**Possible Solution:**

1. Verify and correct the address range.
2. Verify and correct the packet format and length.
3. Verify and correct the communications configuration and connections.
4. Verify and correct the data type, values, and ranges.
5. Reduce network traffic or increase the Request Timeout and/or Fail After attempt count.
6. Decrease the tag group scan rate to reduce the workload on the PLC CPU.
7. Increase the Scan Cycle Load from Communication and Scan Cycle Monitoring Time.
8. Change the value in the device to a valid DWORD that can be converted to a time that is less than or equal to 23:59:59.999.
9. Determine the MPI ID in use for communications and re-enter it in the MPI ID Device Property field.
10. Verify the device's address limits and correct the tag references causing the error.

**See Also:**

Error Matrix

**Unable to read data from device. Block deactivated. | Memory type = '<type>', block start = <address>, block size = <size> (bytes),**

**Error Type:**

Warning

**Possible Cause:**

1. An S7 Messaging error occurred. A portion is malformed or contains incorrect packet lengths.
2. The device CPU work load is too high.
3. If the tag address references a TOD data type, the DWORD value may be larger than the number of milliseconds in a day. For example, 86400000.
4. An error was returned from the PLC or NetLink adapter.
5. Cable noise may cause distortion in the frame, resulting in erroneous data or dropped frames. Verify the cabling between the PC and the PLC device.
6. If the error code=0x11, an incorrect MPI ID may be set.
7. If the error code=0x87, users may be accessing data out of range in the device.
Possible Solution:

1. Verify and correct the address range.
2. Verify and correct the packet format and length.
3. Verify and correct the communications configuration and connections.
4. Verify and correct the data type, values, and ranges.
5. Reduce network traffic or increase the Request Timeout and/or Fail After attempt count.
6. Decrease the tag group scan rate to reduce the workload on the PLC CPU.
7. Increase the Scan Cycle Load from Communication and Scan Cycle Monitoring Time.
8. Change the value in the device to a valid DWORD that can be converted to a time that is less than or equal to 23:59:59.999.
9. Determine the MPI ID in use for communications and re-enter it in the MPI ID Device Property field.
10. Verify the device's address limits and correct the tag references causing the error.

See Also:
Error Matrix

Unable to write to address on device. | Address = '<address>',

Error Type:
Warning

Possible Cause:

1. The connection between the device and the host PC is broken.
2. The named device may have been assigned an incorrect IP address.
3. The device CPU workload is too high.
4. An attempt was made to write to an array larger than the PDU size negotiated with the device.

Possible Solution:

1. Verify the cabling between the PC and the PLC device.
2. Verify the IP address given to the named device matches that of the actual device.
3. Decrease the tag group scan rate to reduce the work load on the PLC CPU.
4. Increase the Scan Cycle Load from Communication and Scan Cycle Monitoring Time.

See Also:
Error Matrix
Unable to write to address on device. HEXSTRING length is different from tag length. | Address = '<address>', HEXSTRING length = <length> (bytes), tag length = <length> (bytes).

Error Type:
Warning

Possible Cause:
The tag and hexstring length do not match. The syntax for the HEXSTRING subtype is HEXSTRINGy.n, where y is the byte offset and n is the length. The n value must be specified in the range of 1 through 932. String is the only valid data type for a HEXSTRING tag. The value assigned to a HEXSTRING must be an even number of characters. There is no padding, so the entire string must be specified. For example, tag HexStr defined as DB1,STRING0.10 uses 10 bytes of storage and has a display length of 20. To assign a value, the string must be 20 characters long and contain only valid hexadecimal characters.

Possible Solution:
Correct the mismatch between the tag and hexstring length.

Unable to write to address on device. HEXSTRING contains a non-hexadecimal character. | Address = '<address>'.

Error Type:
Warning

Possible Cause:
The hexstring format is invalid. The syntax for the HEXSTRING subtype is HEXSTRINGy.n, where y is the byte offset and n is the length. The n value must be specified in the range of 1 through 932. String is the only valid data type for a HEXSTRING tag. To assign a value, the string must be 20 characters long and contain only valid hexadecimal characters.

Possible Solution:
Correct the format and syntax of the hexstring.

Unable to write to address on device. HEXSTRING length must be an even number of characters. | Address = '<address>'.

Error Type:
Warning

Possible Cause:
The hexstring length contains on odd number of characters, which is not valid.

Possible Solution:
Correct the hexstring to contain an even number of hexadecimal characters.
Unable to write to address on device. Time of Day string contains a syntax error. Expected 'hh:mm:ss.hhh' format. | Address = '<address>', Time of Day string = '<string>'.

Error Type:
Warning

Possible Cause:
The string written is not in the correct hh:mm:ss.hhh format.

Possible Solution:
Format the string as hh:mm:ss.hhh and retry.

Error Codes

NetLink Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td></td>
<td>Service could be executed without an error</td>
</tr>
<tr>
<td>0x01</td>
<td>Remote Station</td>
<td>Timeout from remote station</td>
</tr>
<tr>
<td>0x02</td>
<td>Remote Station</td>
<td>Resource unavailable</td>
</tr>
<tr>
<td>0x03</td>
<td>Remote Station</td>
<td>Requested function of master is not activated within the remote station</td>
</tr>
<tr>
<td>0x11</td>
<td>Remote Station</td>
<td>No response of the remote station</td>
</tr>
<tr>
<td>0x12</td>
<td>Network</td>
<td>Master not into the logical token ring</td>
</tr>
<tr>
<td>0x14</td>
<td>Host</td>
<td>Resource of the local FDL controller not available or not sufficient</td>
</tr>
<tr>
<td>0x15</td>
<td>Host</td>
<td>The specified msg.data_cnt parameter is invalid</td>
</tr>
<tr>
<td>0x30</td>
<td>Remote Station</td>
<td>Timeout. The requested message was accepted but no indication was sent back by the remote station</td>
</tr>
<tr>
<td>0x39</td>
<td>Remote Station</td>
<td>Sequence fault, internal state machine error</td>
</tr>
<tr>
<td>0x85</td>
<td>Host</td>
<td>Specified offset address out of limits or unknown in the remote station</td>
</tr>
<tr>
<td>0x86</td>
<td>Device</td>
<td>Wrong PDU coding in the MPI response of the remote station</td>
</tr>
<tr>
<td>0x87</td>
<td>Host</td>
<td>Specified length to write or to read results in an access outside of limits</td>
</tr>
</tbody>
</table>

Transport Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Error reason not specified</td>
</tr>
<tr>
<td>0x01</td>
<td>Invalid parameter code</td>
</tr>
<tr>
<td>0x02</td>
<td>Invalid TPDU type</td>
</tr>
<tr>
<td>0x03</td>
<td>Invalid parameter value</td>
</tr>
</tbody>
</table>
### Protocol Errors

<table>
<thead>
<tr>
<th>Error Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>No error</td>
</tr>
<tr>
<td>0x81</td>
<td>Error in the application ID of the request</td>
</tr>
<tr>
<td>0x82</td>
<td>Error in the object definition (e.g. bad data type)</td>
</tr>
<tr>
<td>0x83</td>
<td>No resources available</td>
</tr>
<tr>
<td>0x84</td>
<td>Error in the structure of the service request</td>
</tr>
<tr>
<td>0x85</td>
<td>Error in the communication equipment</td>
</tr>
<tr>
<td>0x87</td>
<td>Access error</td>
</tr>
<tr>
<td>0xD2</td>
<td>OVS error</td>
</tr>
<tr>
<td>0xD4</td>
<td>Diagnostic error</td>
</tr>
<tr>
<td>0xD6</td>
<td>Protection system error</td>
</tr>
<tr>
<td>0xD8</td>
<td>BuB error</td>
</tr>
<tr>
<td>0xEF</td>
<td>Layer 2 specific error</td>
</tr>
</tbody>
</table>

### Data Access Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFF</td>
<td>No error</td>
</tr>
<tr>
<td>0x01</td>
<td>Hardware fault</td>
</tr>
<tr>
<td>0x03</td>
<td>Illegal object access</td>
</tr>
<tr>
<td>0x05</td>
<td>Invalid address (incorrect variable address)</td>
</tr>
<tr>
<td>0x06</td>
<td>Data type is not supported</td>
</tr>
<tr>
<td>0x07</td>
<td>Invalid data size / too much data</td>
</tr>
<tr>
<td>0x0A</td>
<td>Object does not exist or length error</td>
</tr>
</tbody>
</table>
Appendix: Configuring Siemens Connections

How To Configure S7-200 Connections in Micro/WIN

How To Configure S7-300/400 Connections in STEP 7

How To Configure S7-1200 Connections with the Totally Integrated Automation (TIA) Portal

How To Configure S7-1500 Connections

How To Configure S7-200 Connections in Micro/WIN

Configured connections are accomplished through the Ethernet wizard in Micro/WIN. The following instructions illustrate each step in the Ethernet wizard and also describe any precautions that should be taken. Follow these instructions closely to use configured connections with the Siemens TCP/IP Ethernet Driver correctly.

Note: The Micro/WIN software may require an upgrade before the Ethernet wizard is made available.

Step 1: Launching the Ethernet Wizard
1. In the Micro/WIN main menu, click Tools | Ethernet Wizard.
2. Then, click Next.
3. Click **Yes** to proceed.

**Note:** The program must be compiled before the Ethernet wizard can execute. Correct any errors in the program before continuing.

**Step 2: Setting CP243-1 Module Position**

1. Click **Read Modules**.
Note: While it is recommended that the Read Modules function be used, this does require that the PLC be connected to the PC either serially or by Ethernet. In either case, the communications parameters for Micro/WIN must be properly set for the Read Modules operation to occur.

2. To view the results of Read Modules, select the Ethernet module. Click Next.

Step 3: Assigning Module Address
1. Enter the IP Address, Subnet Mask and Gateway Address if applicable. Alternatively, enable BOOTP if applicable.
2. Select **Auto Detect Communications** to allow the module to automatically select either **10BaseT** or **100BaseT**. In rare cases where there may be a cable issue that does not allow the module to operate properly at 100BaseT, force the module to use 10BaseT. This will increase the modules' tolerance to a bad Ethernet line.

3. The image shown below contains demonstration values.

Step 4: Number of Configured Connections

1. Enter the number of desired available connections for this device. If 0 is entered, the only connection available will be the PG connection used by Micro/WIN.
Note: The number of connections selected determines how many simultaneous connections the PLC can support. When intending to have only one OPC server talking to the PLC, set up only one connection. This will ensure the best performance for the OPC server. When intending to have more than one active connection to the PLC, use multiple connections. Keep in mind, however, that the performance of the module will be impacted as each connection is used.

2. The image shown below contains 4 connections.

Step 5: Configuring Connections
Each connection is configured individually. For this example, 4 connections have been selected.
**Step 5a: Connection 0**

There are two types of connections, client and server. In a client connection, the device is a client and makes request with servers (other devices). In a server connection, the device is a server and handles requests from clients (such as the OPC server and other devices). The latter is required for communications with the Siemens TCP/IP Ethernet Driver.

1. Select **This is a Server Connection**.

![Configure Connections](image)

- Connection 0 (1 connections requested):
  - This is a Client Connection: Client connections request data transfers between the local PLC and a remote server.
  - This is a Server Connection: Servers respond to connection requests from remote clients.

   **Local Properties (Client)**
   - TSAP: 10.00
   - You may define up to 32 data transfers between this connection and the remote server.

   **Remote Properties (Server)**
   - TSAP: 10.00
   - Specify the IP address of the server for this connection.
   
   **Enable the Keep Alive function for this connection.**

   Please specify a symbolic name for this client connection. Your program can reference this connection symbolically when initiating data transfers with the remote server.

   **Connection0_0**

2. From this standpoint, the CP243-1 is considered the **Server** (local) and the **OPC Server Channel**. The device is considered the **Client (Remote)**.

3. Enter a **Remote TSAP** or accept the default. This will be the **Local TSAP** in the **OPC Server**.

4. **Optional**: Accept all connection requests or limit to a particular remote machine. It is recommended that **Accept all connection requests** be selected. If concerned about device security (or if intending to access this device over the Internet) select a specific IP address. Users must ensure that the OPC server is running on a PC that has a known and fixed IP address.
5. Select **Enable the Keep Alive**.

6. Click **Next Connection**.

---

**Configure Connections**

You have requested 4 connection(s). For each connection, specify whether the connection should act as a client or server, and configure its associated properties.

- **Connection 0 (4 connections requested)**
  - **This is a Client Connection**: Client connections request data transfers between the local PLC and a remote server.
  - **This is a Server Connection**: Servers respond to connection requests from remote clients.

**Local Properties (Server)**

- **TSAP**: 10.00
- **This server will connection with an Operator Panel (OP)**.
- **Accept all connection requests**.

**Remote Properties (Client)**

- **TSAP**: 10.00

**Step 5b: Connection 1**

1. Select **This is a Server Connection**.
2. Notice the Local TSAP automatically incremented to 11.00.

3. Enter a Remote TSAP or accept the default. This will be the Local TSAP in the OPC Server.

4. Optional: Accept all connection requests.

5. Select Enable the Keep Alive.

6. Click Next Connection.
Step 5c: Connection 2
1. Select **This is a Server Connection.**
2. Notice the **Local TSAP** automatically incremented to 12.00.

3. Enter a **Remote TSAP** or accept the default. This will be the **Local TSAP** in the **OPC Server**.

4. **Optional**: Accept all connection requests.

5. Select **Enable the Keep Alive**.

6. Click **Next Connection**.
Step 5d: Connection 3

1. Select This is a Server Connection....
2. **Notice** the **Local TSAP** automatically incremented to 13.00.

3. Enter a **Remote TSAP** or accept the default. This will be the Local TSAP in the OPC server.

4. **Optional**: Accept all connection requests.

5. Select **Enable the Keep Alive**.

6. Click **Next Connection**.
That completes the configuration of the four connections that were selected.

**Note:** Notice that the Local TSAP in the Connection dialog was automatically advanced for each connection. This TSAP number will need to be used in the OPC server setup when defining a device as the remote TSAP number.

**Step 6: CRC and Keep Alive Interval**

**Optional:** Enable CRC protection to monitor for accidental configuration corruption.

1. Set the Keep Alive Interval. The longer the interval, the longer the connection between the device and the OPC server will exist during idle time. A long Keep Alive Interval may not be desirable if connections are being shared (nonconcurrent). Each remote client will need to wait this amount of time before it will be able to connect with the device once the last connected remote client is finished communications. The 30 second default is suggested.
Step 7: Memory for Configuration

1. Click **Suggest Address** to let the wizard find the best available location to store the **Ethernet configuration**.

2. The image shown below displays the results.
Note: It is recommended that the Micro/WIN software pick this location for the application. If a CRC was not generated for the configuration data, please take steps to ensure that no other aspect of the PLC program will overwrite this area of memory.

Step 8: Ethernet Wizard Summary
1. Click Finish or Prev to modify the Configured Connections.

2. To review what the Ethernet wizard produced, double-click ETH0_CTRL under the Program Block. All TSAPs configured are listed for future reference. Remember, the Local TSAP below is the Remote TSAP in the OPC server and the remote TSAP below is the Local TSAP in the OPC server.
3. The image below shows ETH0_CTRL.

4. The image below shows ETH0_CFG.

5. Now that the results of the Ethernet wizard have been confirmed, a connection can be made using the OPC server.

How To Configure S7-300/400 Connections in STEP 7

To configure the S7-300/400 for communications with the Siemens TCP/IP Ethernet Driver, both the CPU and the Ethernet module will need to be configured as well. To do so, follow the directions below.

1. From the Simatic Manager, launch HW Config by double-clicking Hardware under the SIMATIC Station.
2. If this is a new Simatic project, add the necessary modules to the **Rack** in HW Config. For the Siemens TCP/IP Ethernet Driver to communicate with the CPU, there will need to be at least one Ethernet module capable of **S7 Communications**. This may be built into the CPU.

### Configuring an Ethernet Module

1. Right-click on the particular module in the rack and then select **Object Properties**.

2. The dialog should appear as shown below.
3. From the **General** tab, click the **Interface** | **Properties** button.
4. Specify the **IP** and **Subnet Mask** for this module.

5. To network this module, click **New** under **Subnet**. Next, select the network created and the click **OK**.

6. Return to the HW Config main window.

**Configuring Connections**

1. Right-click on the CPU module in the rack and select **Object Properties**.
2. The dialog should appear as shown below.
3. Configure the desired number of **PG/OP** and **PC (S7 Communication)** connections.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PG Communication</td>
<td>Used for program loading, diagnostics</td>
</tr>
<tr>
<td>OP Communication</td>
<td>Used for operator control and monitoring</td>
</tr>
<tr>
<td>S7 Standard Communication</td>
<td>Communication connections not configured, MPI communications with PUT/GET function blocks</td>
</tr>
<tr>
<td>S7 Communication (PC)</td>
<td>Configured connections, data communications</td>
</tr>
</tbody>
</table>

*Note:* The maximum number of PC connections for the CPU equals the Maximum Number of Connection Resources **minus** the S7 Standard Communication resources **minus** the OP Communication resources **minus** the PG Communication resources. Note that the Maximum Number of Connection Resources is based on the CPU/version/firmware.

In the example shown above, there are six S7 communication (PC) connections available (12-4-1-1=6). Likewise, the number of PG and OP connections can be increased using the same concept.

If the **Device returned protocol [Class=0x83, Code=0x04]** error is encountered, increase the number of S7 Standard Communication connections, thereby decreasing the number of S7 Communication connections.

4. After the connections have been configured, click **OK**. Next, in the main HW Config window click **Station | Save and Compile**.
5. Click **PLC | Download** to commit to the changes.

**How To Configure S7-1200 Connections with the Totally Integrated Automation (TIA) Portal**

To configure the S7-1200 for communications with the Siemens TCP/IP Ethernet Driver, an online connection is required between the programming device and the target system. Users may have to configure the programming device to talk to the target system. For more information, follow the instructions below.

**Note:** For new Simatic projects, refer to the PLC’s documentation for information on the default IP address settings.

1. Start the TIA Portal. In the Portal View, click **Create new project**.

2. Next, select the **Online & Diagnostics** tab and then click **Accessible Devices**.

3. Select the appropriate PG/PC interface. This will prompt the TIA to scan the network for the device.
4. Once the scan is complete, select the device and then click **Show**. This will invoke the Project View.

5. In the project tree, locate the IP address and then open **Online & Diagnostics**.

6. Next, double-click **Online & Diagnostics** to invoke **Online Access**.

7. Select **Functions** and then click **Assign IP Address**.

8. Enter the communication settings and click **Assign IP Address**.

   *Note:* The device is now ready to be configured.
9. Return to the Portal View and then select the **Device & Networks** tab. Then, click **Add new device**.

![Device & Networks Tab](image1)

10. Next, select the device's configuration and then click **Add device**. This will invoke the Project View, where the device's hardware can be further configured.

![Project View](image2)

11. Once finished, view the project tree. Locate **Program Blocks** and **PLC Tags** and then configure the addresses that will be used in the PLC project.

![Program Blocks and PLC Tags](image3)
Note: The device is now configured and can be placed in Run Mode for communications.

How To Configure S7-1500 Connections

The S7-1500 controller has an embedded Ethernet port that supports S7 communications over the Ethernet protocol. It must be configured to allow access from the server, however. For more information, refer to the instructions below.

1. To start, open the PLC project in the S7 TIA Portal software.
2. Next, open PLC Properties.
3. In the General tab, select Protection. Then, ensure that Full access (no protection) is enabled.
   
   Note: At this time, the Siemens TCP/IP Ethernet Driver does not support the use of a password.
4. Next, check Permit access with PUT/GET communication from remote partner. Then, save the settings.
5. In the server, create a new channel. In Device Driver, select Siemens TCP/IP Ethernet Driver from the drop-down list. Continue through the channel wizard, specifying channel properties as needed. Then, click Finish.

6. Next, create a new device. In Model, select S7-1500 from the Device Model drop-down list. Continue through the device wizard, specifying device properties as needed. Then, click Finish.

Note: The controller Protection settings are directly related to the device returning protocol error [Class-s=0x81, Code=0x04]. If this error occurs, the controller’s memory (such as the Simatic memory card) may need to be reset. Once the controller’s memory is reset, downloading the S7 project to the controller should not prompt for a password.
Appendix: Alternate Syntax Support

Limited addressing support is available for the following applications.

**Applicom Direct-Link Item Syntax**
**INAT OPC-Server TCPIPH1 Item Syntax**
**Siemens Simatic Net Item Syntax**
**Softing S7/S5 OPC Server Item Syntax**

**Note:** All brand and product names are trademarks, registered trademarks, or service marks of their respective holders.

**Note:** The offset for an atomic type tag in a data block is denoted by the column "Address" in Step 7, as shown above. This offset is denoted by the column "Offset" in the Siemens TIA Portal programming environment.

**Applicom Direct-Link Item Syntax**

The following support for the Applicom Direct-Link OPC server is considered to be limited. Care must be taken because the data type for a given S7 data type/suffix may differ from the data type for the same S7 data type/suffix in the specified product. Applicom ASCII strings are not supported by this driver. The following information is intended to be a guideline for users that are already familiar with and/or prefer the syntax of the specified product. For preferred item syntax, refer to Standard S7-300/400/1200/1500 Item Syntax.

**Address Syntax**

**Input, Output, Peripheral, Flag Memory Types**

<memory type><S7 data type><address>[<Data Type suffix>][<Byte Switching suffix>]
<memory type><S7 data type><address>[<Data Type suffix>]<_row_ col>

**Timer and Counter Memory Types**

<memory type><address>

**DB Memory Type**

DB<num>.<S7 data type><address>[<Data Type suffix>][< Byte Switching suffix>]
DB<num>.<S7 data type><address>[<Data Type suffix>]<_row_ col>

where <num> ranges from 1 to 65535.

**See Also:** Examples

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Inputs</td>
<td></td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>Outputs</td>
<td></td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>Peripheral Inputs</td>
<td>Dependent on S7 Data Type</td>
<td>Read Only</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>Peripheral Outputs</td>
<td></td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

www.ptc.com
<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Flag Memory</td>
<td></td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>DB</td>
<td>Data Blocks</td>
<td></td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>Timers</td>
<td>T0-T65535</td>
<td>DWord, Long</td>
<td>Read/Write</td>
</tr>
<tr>
<td>C</td>
<td>Counters</td>
<td>C0-C65535</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Z</td>
<td></td>
<td>Z0-Z65535</td>
<td>Read/Write</td>
<td></td>
</tr>
</tbody>
</table>

**See Also:** [Examples]

**S7 Data Types**

The S7 data type is used to coerce the data type for a tag. It does not apply to Timers and Counters. The default data types are shown in **bold**. Suffixes are not required.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
<th>Suffix</th>
<th>Data Type w/ Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>None*</td>
<td>Bit</td>
<td>0.b-65534.b</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBX0.b-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBX65534.b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>.b is Bit Number 0-15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Unsigned Byte</td>
<td>B0-B65535</td>
<td>Byte</td>
<td>Char</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBB0-DBB65535</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>Unsigned Word</td>
<td>W0-W65534</td>
<td>Word</td>
<td>Short, BCD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBW0-DBW65534</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Unsigned Double Word</td>
<td>D0-D65532</td>
<td>DWord, Long, LBCD, Float</td>
<td>F</td>
<td>Float</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBD0-DBD65532</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*No S7 data type specified. Applies to non-DB memory types only.
**Applies to DB memory types only.

**See Also:** [Examples]

**Data Type Suffixes**

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>32-bit IEEE floating point value</td>
<td>Float</td>
</tr>
</tbody>
</table>

**Byte Switching Suffixes**

These suffixes are used to switch the bytes that compose data of type 16-bit Word, 32-bit DWord, or 32-bit Float. The switching is applied after the device-level addressing option for byte Order is applied. For more information, refer to [Addressing Options].

Byte Switching Suffixes can be used with all memory types except Timers and Counters (T, C, and Z) and Peripheral Inputs and Outputs (PI, PE, PQ, and PA). For information on the various types of switching that depend on the suffix and data type of the item, refer to the table below.
Siemens TCP/IP Ethernet Driver

<table>
<thead>
<tr>
<th>Suffix</th>
<th>16-Bit Data Types (Word, Short, BCD)</th>
<th>32-Bit Data Types (DWord, Long, LBCD, Float)</th>
</tr>
</thead>
<tbody>
<tr>
<td>_X1</td>
<td>O1 O2 -&gt; O2 O1 (byte switching)</td>
<td>O1 O2 O3 O4 -&gt; O4 O3 O2 O1 (byte switching)</td>
</tr>
<tr>
<td>_X2</td>
<td>O1 O2 -&gt; O2 O1 (byte switching)</td>
<td>O1 O2 O3 O4 -&gt; O3 O4 O1 O2 (Word switching)</td>
</tr>
<tr>
<td>_X3</td>
<td>O1 O2 -&gt; O2 O1 (byte switching)</td>
<td>O1 O2 O3 O4 -&gt; O2 O1 O4 O3 (Switching bytes in the words)</td>
</tr>
</tbody>
</table>

Array Support

The `<array size>` notation is appended to an address to specify an array (such as "MW0.10"). Boolean arrays and string arrays are not supported.

Examples

<table>
<thead>
<tr>
<th>S7 Data Type</th>
<th>Data Type</th>
<th>Input</th>
<th>Flags</th>
<th>Data Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>None DBX</td>
<td>Boolean</td>
<td>I0.7</td>
<td>M0.7</td>
<td>DB1.DBX0.7</td>
</tr>
<tr>
<td>B DBB</td>
<td>Byte</td>
<td>IBO</td>
<td>MB0</td>
<td>DB1.DBB0</td>
</tr>
<tr>
<td></td>
<td>Array</td>
<td>IBO_2_5</td>
<td>MB0_2_5</td>
<td>DB1.DBB0_2_5</td>
</tr>
<tr>
<td>W DBW</td>
<td>Word</td>
<td>IW0</td>
<td>MW0</td>
<td>DB1.DBW0</td>
</tr>
<tr>
<td></td>
<td>Array</td>
<td>IW0_10</td>
<td>MW0_10</td>
<td>DB1.DBW0_10</td>
</tr>
<tr>
<td>D DBD</td>
<td>DWord</td>
<td>ID0</td>
<td>MD0</td>
<td>DB1.DBD0</td>
</tr>
<tr>
<td></td>
<td>Float (F)</td>
<td>ID0F</td>
<td>MD0F</td>
<td>DB1.DBD0F</td>
</tr>
<tr>
<td></td>
<td>Array</td>
<td>ID0_4_3</td>
<td>MD0_4_3</td>
<td>DB1.DBD0F_4_3</td>
</tr>
</tbody>
</table>

All brand and product names are trademarks, registered trademarks, or service marks of their respective holders.

INAT OPC Server TCPIPH1 Item Syntax

The following support for the INAT OPC Server TCPIPH1 (V1.22 and up) is considered to be limited. Care must be taken as the data type for a given S7 data type/suffix may differ from the data type for the same S7 data type/suffix in the specified product. S7 data type and suffixes not included below are not supported by this driver. The following information is intended to be a guideline for users that are already familiar with and/or prefer the syntax of the specified product. For preferred item syntax, refer to Standard S7-300/400/1200/1500 Item Syntax.
Address Syntax
Input, Output, Peripheral, Flag Memory Types

<memory type><S7 data type><address>[<suffix>]
<memory type><S7 data type><address><string length>*
<memory type><S7 data type><address><array size>[<suffix>]

Timer and Counter Memory Types
<memory type><address>

DB Memory Type
DB<num>.<S7 data type><address>[<suffix>]
DB<num>.<S7 data type><address><string length>*
DB<num>.<S7 data type><address><array size>[<suffix>]

where <num> ranges from 1 to 65535.

*Applies to S7 data types that support string.

See Also: Examples

Memory Types

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Inputs</td>
<td>Dependent on S7 Data Type</td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>Outputs</td>
<td></td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PI</td>
<td>Peripheral Inputs</td>
<td></td>
<td>Read Only</td>
<td></td>
</tr>
<tr>
<td>PE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ</td>
<td>Peripheral Outputs</td>
<td></td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>PA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Flag Memory</td>
<td></td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB</td>
<td>Data Blocks</td>
<td></td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>Timers</td>
<td>T0-T65535</td>
<td>DWord, Long</td>
<td>Read/Write</td>
</tr>
<tr>
<td>C</td>
<td>Counters</td>
<td>C0-C65535</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Z</td>
<td></td>
<td>Z0-Z65535</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See Also: Examples

S7 Data Types
The S7 data type is used to coerce the data type for a tag. It does not apply to Timers and Counters. Default data types are shown in **bold**. A suffix is not required.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
<th>Suffix</th>
<th>Data Type w/ Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Bit</td>
<td>X0.b-X65534.b</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>.b is Bit Number 0-15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Unsigned</td>
<td>B0-B65535</td>
<td>Byte, Char</td>
<td>KF</td>
<td>Char</td>
</tr>
</tbody>
</table>

www.ptc.com
### Data Type

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
<th>Suffix</th>
<th>Data Type w/ Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>Byte</td>
<td>BYTE0-BYTE65535</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>Unsigned Word</td>
<td>W0-W65534</td>
<td>Word, Short, BCD</td>
<td>BCD</td>
<td>BCD Short</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WORD0-WORD65534</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Signed Word</td>
<td>I0-I65534</td>
<td>Word, Short, BCD</td>
<td>BCD</td>
<td>BCD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INT0-INT65534</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Unsigned Double Word</td>
<td>D0-D65532</td>
<td>DWord, Long, LBCD, Float</td>
<td>BCD IEEE</td>
<td>LBCD Float Long</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DWORD0-DWORD65532</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DI</td>
<td>Signed Double Word</td>
<td>DI0-DI65532</td>
<td>DWord, Long, LBCD, Float</td>
<td>BCD IEEE</td>
<td>LBCD Float Long</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DINT0-DINT65532</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>IEEE Float</td>
<td>R0-R65532</td>
<td>Float</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>REAL0-REAL65532</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>S7 String</td>
<td>G0.n-G65532.n</td>
<td>String</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>STRING0.n-STRING65532.n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>.n is string length</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0&lt;n&lt;= 254 (PDU size of 480 and above)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0&lt;n&lt;= 210 (PDU size below 480)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0&lt;n&lt;= 254 (Netlink S7300 and Netlink S7400 Models)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### See Also: Examples

### Suffixes

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCD</td>
<td>Two byte packed BCD for Word references Value range is 0-9999</td>
<td>BCD</td>
</tr>
<tr>
<td></td>
<td>Four byte packed BCD for DWord references Value range is 0-99999999</td>
<td>LBCD</td>
</tr>
<tr>
<td>IEEE</td>
<td>32-bit IEEE floating point value</td>
<td>Float</td>
</tr>
<tr>
<td>KF</td>
<td>Signed</td>
<td>Char Short Long</td>
</tr>
</tbody>
</table>

### Array Support

The `<array size>` notation is appended to an address to specify an array (such as MW0.10). Boolean arrays and string arrays are not supported.

### Examples

<table>
<thead>
<tr>
<th>S7 Data Type</th>
<th>Data Type</th>
<th>Input</th>
<th>Flags</th>
<th>Data Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Boolean</td>
<td>IX0.7</td>
<td>MX0.7</td>
<td>DB1.X0.7</td>
</tr>
<tr>
<td>B</td>
<td>Byte</td>
<td>IB0</td>
<td>MB0</td>
<td>DB1.B0</td>
</tr>
<tr>
<td>S7 Data Type</td>
<td>Data Type</td>
<td>Input</td>
<td>Flags</td>
<td>Data Blocks</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>Byte</td>
<td>Char (KF)</td>
<td>IB0KF</td>
<td>MB0KF</td>
<td>DB1.B0KF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBYTE0KF</td>
<td>MBYTE0KF</td>
<td>DB1.BYTE0KF</td>
</tr>
<tr>
<td></td>
<td>Array</td>
<td>IB0KF.10</td>
<td>MB0KF.10</td>
<td>DB1.B0KF.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBYTE0KF.10</td>
<td>MBYTE0KF.10</td>
<td>DB1.BYTE0KF.10</td>
</tr>
<tr>
<td></td>
<td>Word</td>
<td>IW0</td>
<td>MW0</td>
<td>DB1.W0</td>
</tr>
<tr>
<td></td>
<td>BCD (BCD)</td>
<td>IW0BCD</td>
<td>MW0BCD</td>
<td>DB1.W0BCD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IWORD0BCD</td>
<td>MWORD0BCD</td>
<td>DB1.WORD0BCD</td>
</tr>
<tr>
<td></td>
<td>Short (KF)</td>
<td>IW0KF</td>
<td>MW0KF</td>
<td>DB1.W0KF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IWORD0KF</td>
<td>MWORD0KF</td>
<td>DB1.WORD0KF</td>
</tr>
<tr>
<td></td>
<td>Array</td>
<td>IW0BCD.10</td>
<td>MW0BCD.10</td>
<td>DB1.W0BCD.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IWORD0BCD.10</td>
<td>MWORD0BCD.10</td>
<td>DB1.WORD0BCD.10</td>
</tr>
<tr>
<td></td>
<td>Short</td>
<td>II0</td>
<td>MI0</td>
<td>DB1.I0</td>
</tr>
<tr>
<td></td>
<td>BCD (BCD)</td>
<td>II0BCD</td>
<td>MI0BCD</td>
<td>DB1.I0BCD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IINT0BCD</td>
<td>MIINT0BCD</td>
<td>DB1.INT0BCD</td>
</tr>
<tr>
<td></td>
<td>Array</td>
<td>II0.10</td>
<td>MI0.10</td>
<td>DB1.I0.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IINT0.10</td>
<td>MIINT0.10</td>
<td>DB1.INT0.10</td>
</tr>
<tr>
<td></td>
<td>DWord</td>
<td>ID0</td>
<td>MD0</td>
<td>DB1.D0</td>
</tr>
<tr>
<td></td>
<td>LBCD (BCD)</td>
<td>ID0BCD</td>
<td>MD0BCD</td>
<td>DB1.D0BCD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDWORD0BCD</td>
<td>MDWORD0BCD</td>
<td>DB1.DWORD0BCD</td>
</tr>
<tr>
<td></td>
<td>Float (IEEE)</td>
<td>ID0EEE</td>
<td>MD0EEE</td>
<td>DB1.D0EEE</td>
</tr>
<tr>
<td></td>
<td>Long (KF)</td>
<td>ID0KF</td>
<td>MD0KF</td>
<td>DB1.D0KF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDWORD0KF</td>
<td>MDWORD0KF</td>
<td>DB1.DWORD0KF</td>
</tr>
<tr>
<td></td>
<td>Array</td>
<td>ID0EEE.10</td>
<td>MD0EEE.10</td>
<td>DB1.D0EEE.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDWORD0EEE.10</td>
<td>MDWORD0EEE.10</td>
<td>DB1.DWORD0EEE.10</td>
</tr>
<tr>
<td></td>
<td>Long</td>
<td>IDI0</td>
<td>MDI0</td>
<td>DB1.DI0</td>
</tr>
<tr>
<td></td>
<td>LBCD (BCD)</td>
<td>IDI0BCD</td>
<td>MDI0BCD</td>
<td>DB1.DI0BCD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDINT0BCD</td>
<td>MDINT0BCD</td>
<td>DB1.DINT0BCD</td>
</tr>
<tr>
<td></td>
<td>Float (IEEE)</td>
<td>IDI0EEE</td>
<td>MDI0EEE</td>
<td>DB1.DI0EEE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDINT0EEE</td>
<td>MDINT0EEE</td>
<td>DB1.DINT0EEE</td>
</tr>
</tbody>
</table>
All brand and product names are trademarks, registered trademarks, or service marks of their respective holders.

Siemens Simatic Net Item Syntax

The following support for the Siemens Simatic Net OPC server is considered to be limited. Care must be taken as the data type for a given S7 data type may differ from the data type for the same S7 data type in the specified product. S7 data types not included below are not supported by this driver. The following information is intended to be a guideline for users that are already familiar with and/or prefer the syntax of the specified product. For preferred item syntax, refer to Standard S7-300/400/1200/1500 Item Syntax.

Address Syntax

Input, Output, Peripheral, Flag Memory Types

<memory type><S7 data type><address>
<memory type><S7 data type><address>.string length>
<memory type><S7 data type><address>,array size>

Timer and Counter Memory Types

<memory type><address>

DB Memory Type

DB<num>,<S7 data type><address>
DB<num>,<S7 data type><address>.string length>*
DB<num>,<S7 data type><address>,array size>

where <num> ranges from 1 to 65535.

*Applies to S7 data types that support String.

See Also: Examples

Memory Types

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Inputs</td>
<td>Dependent on S7 Data Type</td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>Q</td>
<td>Outputs</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory Type</td>
<td>Description</td>
<td>Address Range</td>
<td>Data Type</td>
<td>Access</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------</td>
<td>---------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>PI</td>
<td>Peripheral Inputs</td>
<td></td>
<td></td>
<td>Read Only</td>
</tr>
<tr>
<td>PE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ</td>
<td>Peripheral Outputs</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>PA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Flag Memory</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB</td>
<td>Data Blocks</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>T</td>
<td>Timers</td>
<td>T0-T65535</td>
<td>DWord, Long</td>
<td>Read/Write</td>
</tr>
<tr>
<td>C</td>
<td>Counters</td>
<td>C0-C65535</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td>Z</td>
<td></td>
<td>Z0-Z65535</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**See Also: Examples**

**S7 Data Types**
The S7 data type is used to coerce the data type for a tag. It does not apply to Timers and Counters. The default data types are shown in **bold**.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Bit</td>
<td>X0.b-X65534.b</td>
<td>Boolean</td>
</tr>
<tr>
<td></td>
<td>.b is Bit Number 0-15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Unsigned Byte</td>
<td>B0-B65535</td>
<td>Byte, Char</td>
</tr>
<tr>
<td></td>
<td>BYTE0-BYTE65535</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Char</td>
<td>Signed Byte</td>
<td>CHAR0-CHAR65535</td>
<td>Byte, Char</td>
</tr>
<tr>
<td>W</td>
<td>Unsigned Word</td>
<td>W0-W65534</td>
<td>Word, Short, BCD</td>
</tr>
<tr>
<td></td>
<td>WORD0-WORD65534</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td>Signed Word</td>
<td>INTO-INT65534</td>
<td>Word, Short, BCD</td>
</tr>
<tr>
<td>D</td>
<td>Unsigned Double Word</td>
<td>D0-D65532</td>
<td>DWord, Long, LBCD, Float</td>
</tr>
<tr>
<td></td>
<td>DWORD0-DWORD65532</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DINT</td>
<td>Signed Double Word</td>
<td>DINT0-DINT65532</td>
<td>DWord, Long, LBCD, Float</td>
</tr>
<tr>
<td>REAL</td>
<td>IEEE Float</td>
<td>REAL0-REAL65532</td>
<td>Float</td>
</tr>
<tr>
<td>String</td>
<td>S7 String</td>
<td>STRING0.n-STRING65532</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>.n is string length.</td>
<td>0&lt;n&lt;= 254.</td>
<td></td>
</tr>
</tbody>
</table>

**See Also: Examples**

**Array Support**
The `<.array size>` notation is appended to an address to specify an array (such as MW0.10). Boolean arrays and string arrays are not supported.

**Examples**
### Siemens STEP 7 Item Syntax

The following support for Siemens STEP 7 Variable Table (VAT) syntax is considered to be limited. Care must be taken as the data type for a given S7 data type/suffix may differ from the data type for the same S7 data type/suffix in the specified product. S7 data types not included below are not supported by this driver. The following information is intended to be a guideline for users that are already familiar with and/or prefer the syntax of the specified product. For preferred item syntax, refer to [Standard S7-300/400/1200/1500 Item Syntax](#).

### Address Syntax

**Input, Output, Peripheral, Flag Memory Types**

```
<memory type><S7 data type><address>
```

**Timer and Counter Memory Types**

```
<memory type><address>
```
DB Memory Type
DB<num>.<S7 data type><address>

where <num> ranges from 1 to 65535.

See Also: Examples

Memory Types

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Inputs</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>Q</td>
<td>Outputs</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>M</td>
<td>Flag Memory</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>DB</td>
<td>Data Blocks</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>T</td>
<td>Timers</td>
<td>T0-T65535</td>
<td>DWord, Long</td>
<td>Read/Write</td>
</tr>
<tr>
<td>C</td>
<td>Counters</td>
<td>C0-C65535, Z0-Z65535</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
<tr>
<td>P</td>
<td>Peripheral Inputs</td>
<td></td>
<td></td>
<td>Read Only</td>
</tr>
<tr>
<td>P</td>
<td>Peripheral Outputs</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
</tbody>
</table>

See Also: Examples

Accessing Structured Elements in STEP 7
For the Siemens S7-1200 model, STEP 7 accesses the Counter and Timer structured elements as complete whole structures instead of individual tags. For more information on the element offset and its server address equivalent, refer to the tables below.

Counters

<table>
<thead>
<tr>
<th>Element</th>
<th>Data Type</th>
<th>Offset</th>
<th>Server Address Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count_UP</td>
<td>Boolean</td>
<td>0.0</td>
<td>DB1,C0.0</td>
</tr>
<tr>
<td>Count_Down</td>
<td>Boolean</td>
<td>0.1</td>
<td>DB1,C0.1</td>
</tr>
<tr>
<td>Reset</td>
<td>Boolean</td>
<td>0.2</td>
<td>DB1,C0.2</td>
</tr>
<tr>
<td>Load</td>
<td>Boolean</td>
<td>0.3</td>
<td>DB1,C0.3</td>
</tr>
<tr>
<td>Q_UP</td>
<td>Boolean</td>
<td>0.4</td>
<td>DB1,C0.4</td>
</tr>
<tr>
<td>Q_Down</td>
<td>Boolean</td>
<td>0.5</td>
<td>DB1,C0.5</td>
</tr>
<tr>
<td>PAD</td>
<td>Byte</td>
<td>1.0</td>
<td>DB1,B1</td>
</tr>
<tr>
<td>Preset_Value</td>
<td>Short</td>
<td>2.0</td>
<td>DB1,I2</td>
</tr>
<tr>
<td>Count_Value</td>
<td>Short</td>
<td>4.0</td>
<td>DB1,I4</td>
</tr>
</tbody>
</table>

Timers
<table>
<thead>
<tr>
<th>Element</th>
<th>Data Type</th>
<th>Offset</th>
<th>Server Address Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>DWord</td>
<td>0.0</td>
<td>DB1,D0</td>
</tr>
<tr>
<td>Preset</td>
<td>DWord</td>
<td>4.0</td>
<td>DB1,D4</td>
</tr>
<tr>
<td>Elapsed</td>
<td>DWord</td>
<td>8.0</td>
<td>DB1,D8</td>
</tr>
<tr>
<td>Running</td>
<td>Bool</td>
<td>12.0</td>
<td>DB1,DBX12.0</td>
</tr>
<tr>
<td>IN</td>
<td>Bool</td>
<td>12.1</td>
<td>DB1,DBX12.1</td>
</tr>
<tr>
<td>Q</td>
<td>Bool</td>
<td>12.2</td>
<td>DB1,DBX12.2</td>
</tr>
<tr>
<td>PAD</td>
<td>Byte</td>
<td>13.0</td>
<td>DB1,DBB13</td>
</tr>
<tr>
<td>PAD_2</td>
<td>Byte</td>
<td>14.0</td>
<td>DB1,DBB14</td>
</tr>
<tr>
<td>PAD_3</td>
<td>Byte</td>
<td>15.0</td>
<td>DB1,DBB15</td>
</tr>
</tbody>
</table>

For more information, refer to [Standard S7-300/400/1200/1500 Item Syntax](#).

**S7 Data Types**
The S7 data type is used to coerce the data type for a tag. It does not apply to Counters and Timers. The default data types are shown in **bold**.

*No S7 data type specified. Applies to non-DB memory types only.

**Applies to DB memory types only.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
<th>Suffix</th>
<th>Data Type w/ Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>None* DBX**</td>
<td>Bit</td>
<td>0.b-65534.b DBX0.b-DBX65534.b .b is Bit Number 0-15</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B DBB**</td>
<td>Unsigned Byte</td>
<td>B0-B65535 DBB0-DBB65535</td>
<td>Byte, Char</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W DBW**</td>
<td>Unsigned Word</td>
<td>W0-W65534 DBW0-DBW65534</td>
<td>Word, Short, BCD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D DBD**</td>
<td>Unsigned Double Word</td>
<td>D0-D65532 DBD0-DBD65532</td>
<td>DWord, Long, BCD, Float, F Float</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**See Also:** Examples

### Examples

<table>
<thead>
<tr>
<th>S7 Data Type</th>
<th>Data Type</th>
<th>Input</th>
<th>Flags</th>
<th>Data Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>None DBX</td>
<td>Boolean</td>
<td>I0.7</td>
<td>M0.7</td>
<td>DB1,DBX0.7</td>
</tr>
<tr>
<td>B DBB</td>
<td>Byte</td>
<td>IB0</td>
<td>MB0</td>
<td>DB1,DBB0</td>
</tr>
<tr>
<td>W DBW</td>
<td>Word</td>
<td>IW0</td>
<td>MW0</td>
<td>DB1,DBW0</td>
</tr>
<tr>
<td>D</td>
<td>DWord</td>
<td>ID0</td>
<td>MD0</td>
<td></td>
</tr>
</tbody>
</table>
All brand and product names are trademarks, registered trademarks, or service marks of their respective holders.

Note: The offset for an atomic type tag in a data block is denoted by the column "Address" in Step 7, as shown above. This offset is denoted by the column "Offset" in the Siemens TIA Portal programming environment.

### Softing S7/S5 OPC Server Item Syntax

The following support for the Softing S7/S5 OPC server is considered to be limited. Care must be taken as the data type for a given S7 data type/suffix may differ from the data type for the same S7 data type/suffix in the specified product. The following information is intended to be a guideline for users that are already familiar with and/or prefer the syntax of the specified product. For preferred item syntax, refer to [Standard S7-300/400/1200/1500 Item Syntax](#).

#### Address Syntax

**Input, Output, Peripheral, Flag Memory Types**

<memory type><S7 data type><address>[:<suffix>]

**Timer and Counter Memory Types**

<memory type><address>

**DB Memory Type**

DB<num>.<S7 data type><address>[:<suffix>]

Where <num> ranges from 1 to 65535.

#### See Also: Examples

### Memory Types

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Inputs</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>Q</td>
<td>Outputs</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>P</td>
<td>Peripheral Inputs</td>
<td>Dependent on S7 Data Type</td>
<td></td>
<td>Read Only</td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>P</td>
<td>Peripheral Outputs</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>Q</td>
<td></td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>M</td>
<td>Flag Memory</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>DB</td>
<td>Data Blocks</td>
<td></td>
<td></td>
<td>Read/Write</td>
</tr>
<tr>
<td>T</td>
<td>Timers</td>
<td>T0-T65535</td>
<td>DWord, Long</td>
<td>Read/Write</td>
</tr>
<tr>
<td>C</td>
<td>Counters</td>
<td>C0-C65535 Z0-Z65535</td>
<td>Word, Short</td>
<td>Read/Write</td>
</tr>
</tbody>
</table>
**See Also:** Examples

**S7 Data Types**
The S7 data type is used to coerce the data type for a tag. It does not apply to Timers and Counters. The default data types are shown in **bold**. Suffix is not required.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Address Range</th>
<th>Data Type</th>
<th>Suffix</th>
<th>Data Type w/ Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>None* DBX**</td>
<td>Bit</td>
<td>0.b-65534.b DBX0.b-DBX65534.b .b is Bit Number 0-15</td>
<td>Boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B DBB**</td>
<td>Unsigned Byte</td>
<td>B0-B65535 DBB0-DBB65535</td>
<td>Byte, Char</td>
<td>Byte Char String</td>
<td>Byte Char String</td>
</tr>
<tr>
<td>W DBW**</td>
<td>Unsigned Word</td>
<td>W0-W65534 DBW0-DBW65534</td>
<td>Word, Short, BCD</td>
<td>Word INT BCD</td>
<td>Word INT BCD</td>
</tr>
<tr>
<td>D DBD**</td>
<td>Unsigned Double Word</td>
<td>D0-D65532 DBD0-DBD65532</td>
<td>DWord, Long, LBCD, Float</td>
<td>DWord DINT BCD REAL</td>
<td>DWord DINT BCD REAL</td>
</tr>
</tbody>
</table>

*No S7 data type specified. Applies to non-DB memory types only.
**Applies to DB memory types only.

**See Also:** Examples

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>Unsigned Byte</td>
<td>Byte</td>
</tr>
<tr>
<td>Char</td>
<td>Signed Byte</td>
<td>Char</td>
</tr>
<tr>
<td>Word</td>
<td>Unsigned Word</td>
<td>Word</td>
</tr>
<tr>
<td>INT</td>
<td>Signed Word</td>
<td>Short</td>
</tr>
<tr>
<td>DWORD</td>
<td>Unsigned DWord</td>
<td>DWord</td>
</tr>
<tr>
<td>DINT</td>
<td>Signed DWord</td>
<td>Long</td>
</tr>
<tr>
<td>BCD</td>
<td>Two byte packed BCD for Word references Value range is 0-9999</td>
<td>BCD</td>
</tr>
<tr>
<td></td>
<td>Four byte packed BCD for DWord references Value range is 0-99999999</td>
<td>LBCD</td>
</tr>
<tr>
<td>REAL</td>
<td>32-bit IEEE floating point value</td>
<td>Float</td>
</tr>
<tr>
<td>String</td>
<td>S7 String</td>
<td>String</td>
</tr>
</tbody>
</table>

**Examples**
<table>
<thead>
<tr>
<th>S7 Data Type</th>
<th>Data Type</th>
<th>Input</th>
<th>Flags</th>
<th>Data Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Boolean</td>
<td>I0.7</td>
<td>M0.7</td>
<td>DB1,DBX0.7</td>
</tr>
<tr>
<td>B</td>
<td>Byte</td>
<td>IB0</td>
<td>MB0</td>
<td>DB1,DBB0</td>
</tr>
<tr>
<td></td>
<td>String (String)</td>
<td>IB0:String</td>
<td>MB0:String</td>
<td>DB1,DBB0:String</td>
</tr>
<tr>
<td>W</td>
<td>Word</td>
<td>IW0</td>
<td>MW0</td>
<td>DB1,DBW0</td>
</tr>
<tr>
<td></td>
<td>BCD (BCD)</td>
<td>IW0:BCD</td>
<td>MW0:BCD</td>
<td>DB1,DBW0:BCD</td>
</tr>
<tr>
<td>D</td>
<td>DWord</td>
<td>ID0</td>
<td>MD0</td>
<td>DB1,DBD0</td>
</tr>
<tr>
<td></td>
<td>LBCD (BCD)</td>
<td>ID0:BCD</td>
<td>MD0:BCD</td>
<td>DB1,DBD0:BCD</td>
</tr>
<tr>
<td></td>
<td>Float (REAL)</td>
<td>ID0:REAL</td>
<td>MD0:REAL</td>
<td>DB1,DBD0:REAL</td>
</tr>
</tbody>
</table>

All brand and product names are trademarks, registered trademarks, or service marks of their respective holders.
Index

A

A required code page is unavailable on this machine. Tag generation may fail or tag names and descriptions may not appear as expected. | Required code page = <page>. 48

Address Descriptions 25
Address Syntax 31, 87, 90
Addressing Options 16
Allow Sub Groups 14
Applicom Direct-Link Item Syntax 87
Array Support 35, 89, 91
Arrays 42
Attempts Before Timeout 11
Auto-Demotion 12
Auto-generated tag names and descriptions may not appear as expected due to string conversion error. 48

B

BCD 25
BigEndian 17
Bits 17
Boolean 25
Byte 25
Byte Switching Suffix 88

C

Channel Assignment 9
Char 25
Communications Parameters 14
Communications Timeouts 11
Configured connection 15
Configuring Connections 65
Configuring Siemens Connections 61
Connect Timeout 11
Counters 36, 43
CPU Rack 16
CPU Slot 16
Create 14
Created tag in group due to internal block size. | Tag address = '<address>', tag name = '<name>', group name = '<name>'. 50

D

Data Access Errors 60
Data Block 25
Data Collection 9
Data Types Description 25
Date 25
DB Memory Type 31
Delete 13
Demote on Failure 12
Demotion Period 12
Device Properties — Tag Generation 12
Discard Requests when Demoted 12
Do Not Scan, Demand Poll Only 10
Driver 9
DWord 25

E

Error Codes 59
Ethernet Wizard 61
Event Log Messages 45

F

Failed to resolve host. | Host = '<host name>'. 48
Float 25

G

Generate 13
H

Help Contents 5
Hex Strings 35, 42
How To Configure S7-1200 Connections with the Totally Integrated Automation (TIA) Portal 82
How To Configure S7-1500 Connections 85
How To Configure S7-200 Connections in Micro/WIN 61
How To Configure S7-300/400 Connections in STEP 7 76

I

ID 9
Identification 9
INAT OPC Server TCPIP H1 Item Syntax 89
Initial Updates from Cache 10
Inter-Request Delay 11
Internal Tags 30

L

LBCD 25
Legacy S7-300/400 Item Syntax 37
Link Type 16
Little Endian 17
Local TSAP 15
Long 25

M

Memory exception reading the Step 7 language file. 49
Memory Types 31, 87, 90
Micro/WIN 16
MicroWin 61
Model 9
MPI ID 14
NetLink 14
NetLink adapter 14
NetLink Errors 59
NetLink: S7-300 Address Descriptions 30
NetLink: S7-400 Address Descriptions 30

On Device Startup 13
On Duplicate Tag 13
On Property Change 13
Operating Mode 9
Optimizing Communications 24
Overview 5
Overwrite 13

Parent Group 14
PG connection 15
Port Number 14
Program Path 18
Protocol Errors 60

Reason = 'Device is not responding'. 46
Reason = 'Device returned data access error'. Error code = <error>. 46
Reason = 'Device returned protocol error'. Error class = <class>, Error code = <error>. 46
Reason = 'Device returned transport error'. Error code = <error>. 45
Reason = 'Frame contains errors'. 45
Reason = 'Unknown error occurred'. 47
Reason = NetLink returned error. Error code = <error>. 47
Redundancy 19
Remote TSAP 15
Request Timeout 11
Respect Tag-Specified Scan Rate 10

S

S7-1200 Address Descriptions 29
S7-1500 Address Descriptions 29
S7-200 Address Descriptions 26
S7-300 Address Descriptions 28
S7-300/400/1200/1500 16
S7-400 Address Descriptions 29
S7 Communications Parameters 15
Scan Mode 10
Setup 5
Short 25
Siemens Simatic Net Item Syntax 93
Siemens STEP 7 Item Syntax 95
Siemens TCP/IP Ethernet Channel Setup with API Commands 20
Siemens TCP/IP Ethernet Device Setup with API Commands 21, 23
Simulated 10
Softing S7/S5 OPC Server Item Syntax 98
Standard S7-300/400/1200/1500 Item Syntax 31
STEP 7 76
Step 7 language file failed to open. | OS error = '<error>'. 49
STEP 7 Project 18-19
STEP 7 Tag Import 18
String 25
String Subtype 42
String Support 34
Suffixes 91
Supported Devices 5
Supported Models via Siemens STEP 7 18
Supported Models via Siemens TIA Portal 18
Supported NetLink Cables and Gateways 6
Syntax 87

T

Tag Generation 12
Tag generation failure. | Data block name = '<block name>', data block number = <block number>. 50
Tag Import 18
Tag Import Type 18-19
Tag not created because arrays are not supported with specified data type. | Tag name = '<name>', group name = '<name>', data type = '<type>'. 50

Third-Party 87
TIA Portal 82
TIA Portal Tag Import 19
Timeouts to Demote 12
Timers 36, 43
Transport Errors 59
TSAP 16

Unable to connect to device. | 51
Unable to establish association with device. | 51
Unable to load the Step 7 language file. 49
Unable to read data from device. | Data block = '<block>', block start = <address>, block size = <size>, 53
Unable to read data from device. | Memory type = '<type>', block start = <address>, block size = <size> (bytes), 55
Unable to read data from device. Block deactivated. | Data block = '<block>', block start = <address>, block size = <size>, 54
Unable to read data from device. Block deactivated. | Memory type = '<type>', block start = <address>, block size = <size> (bytes), 56
Unable to read from address on device. | Address = '<address>', 52
Unable to read from address on device. Tag deactivated. | Address = '<address>', 53
Unable to write to address on device. | Address = '<address>', 57
Unable to write to address on device. HEXSTRING contains a non-hexadecimal character. | Address = '<address>'. 58
Unable to write to address on device. HEXSTRING length is different from tag length. | Address = '<address>', HEXSTRING length = <length> (bytes), tag length = <length> (bytes). 58
Unable to write to address on device. HEXSTRING length must be an even number of characters. | Address = '<address>'. 58
Unable to write to address on device. Time of Day string contains a syntax error. Expected 'hh mm ss.hhh' format. | Address = '<address>', Time of Day string = '<string>'. 59
W

Word 25