# Allen-Bradley Data Highway Plus Driver

# Table of Contents

# Allen-Bradley Data Highway Plus Driver

Help version 1.060

## CONTENTS

## Overview

The Allen-Bradley Data Highway Plus Driver provides a reliable way to connect Allen-Bradley DH+ devices to client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications.

This driver supports the Allen Bradley SLC family and PLC5-series PLCs, excluding the PLC5/250 series. Address ranges are open to support future models of these series of PLCs.

## Setup



**See Also**: Technical and knowledge base articles available online discuss appropriate settings for a variety of hardware configurations.

## Requirements and Dependencies

### System Restrictions

The Allen-Bradley Data Highway Plus Driver does not work on 64-bit operating systems.

### Supported Devices

SLC Family processors
PLC5 series excluding the PLC5/250 series

**Tip**: Address ranges have been opened up in the driver to allow for new devices. The driver may support a device even if it is not listed above.

### Communication Protocol

Allen-Bradley DH+/DH-485

### Supported Network Cards

This driver requires an interface card by Allen-Bradley or S-S Technologies (SST).
AB 1784-KT
AB 1784-KTX(D)
AB 1784-PKTX
AB-1784-PKTX(D)
AB 1784-PCMK/B
AB 1784-U2DHP USB
SST 5136-SD-ISA
SST 5136-SD-PCI and SST 5136-DHP-PCI

### Supported Networks

Data Highway Plus (DH+)
Data Highway-485 (DH-485)

**Note:** Data Highway-485 (DH-485) is only applicable to Allen-Bradley cards.

## Channel Setup

The maximum number of channels supported is 100.

### Supported Communication Protocol

Allen-Bradley DH+/DH-485

Channel setup includes configuration of the following property groups:

**General**

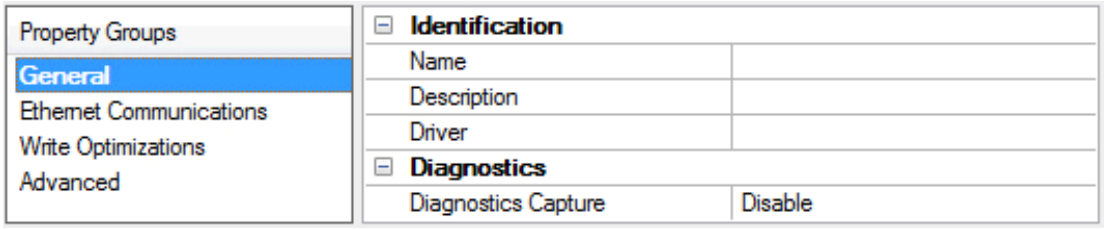
**Write Optimizations**

**Advanced**

**Interface Card**

## 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.
*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.

**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

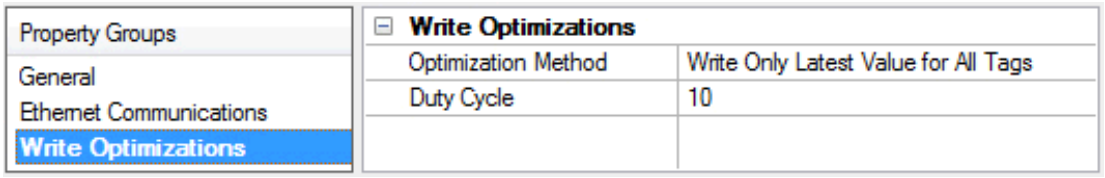
**Diagnostics Capture**: When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default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

*For more information, refer to "Communication Diagnostics" in the server help.*

*Not all drivers support diagnostics. To determine whether diagnostics are available for a particular driver, open the driver information and locate the "Supports device level diagnostics" statement.*

## Channel Properties - Write Optimizations

As with any OPC 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 needed to reach the same final output value. In this way, no extra writes accumulate in the server's queue. When the user stops moving the slide switch, the value in the device is at the correct value at virtually the same time. As the mode states, any value that is not a Boolean value is updated in the server's internal write queue and sent to the device at the next possible opportunity. This can greatly improve the application performance.

  **Note**: This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.
- **Write Only Latest Value for All Tags**: This option takes the theory behind the second optimization mode and applies it to all tags. It is especially useful if the application only needs to send the latest value to the device. This mode optimizes all writes by updating the tags currently in the write queue before they are sent. This is the default mode.

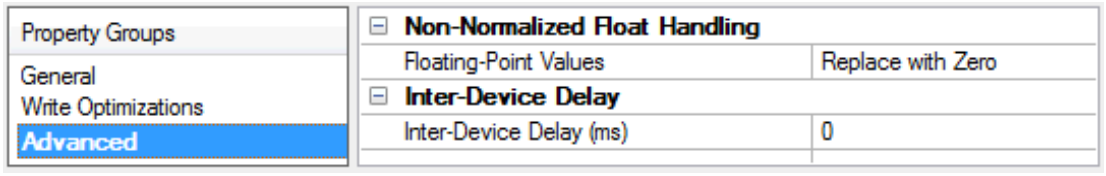
**Duty Cycle**: is used to control the ratio of write to read operations. The ratio is always based on one read for every one to ten writes. The duty cycle is set to ten by default, meaning that ten writes occur for each read operation. Although the application is performing a large number of continuous writes, it must be ensured

that read data is still given time to process. A setting of one results in one read operation for every write operation. If there are no write operations to perform, reads are processed continuously. This allows optimization for applications with continuous writes versus a more balanced back and forth data flow.

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

## Channel Properties - Advanced

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




**Non-Normalized Float Handling**: Non-normalized float handling allows users to specify how a driver handles non-normalized IEEE-754 floating point data. 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. Descriptions of the options are as follows:

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

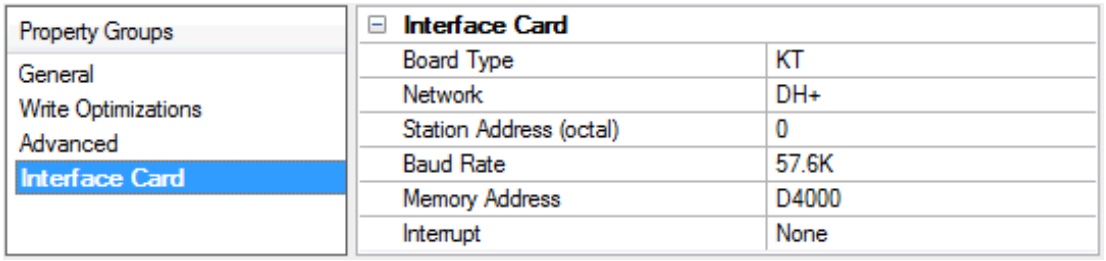
**Note:** This property is disabled if the driver does not support floating point values or if it only supports the option that is displayed. According to the channel's float normalization setting, only real-time driver tags (such as values and arrays) are subject to float normalization. For example,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

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

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

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

## Channel Properties - Interface Card



**Board Type**: Specify the board type of the eight supported by Allen-Bradley and SST. Allen-Bradley types include KT, KTX, KTX-D, PKTX, PKTX-D and PCMK/B. SST board types include 5136-SD-ISA and 5136-SD-PCI.

**Network**: Specify the supported network type: DH+ or DH-485.
**Note**: Allen-Bradley cards support the DH+ and DH-485 networks: SST cards only support the DH+ network.

**Station Address**: Specifies the device's unique node ID: 0-77 octal for DH+ and 0-31 decimal for DH-485.
**Caution**: Ensure that this ID does not conflict with any other Node ID on the network.

**Baud Rate**: Select the baud rate for the device. Speeds available are based on the network type and card specified.

**Memory Address**: Select the location of the card's memory address. The ISA cards (KT, KTX, KTX-D and 5136-SD-ISA) require that the user manually sets the memory address on the card.
*For more information, refer to the appropriate card details: Allen Bradley- 1784 KTX/KTX-D Cards or 5136-SD-ISA Cards.*

**I/O Port Address**: Select the port address the configured for this card.

**Interrupt**: Set the interrupt level based on card type and desired behavior. The PCI card types automatically set up the interrupts. The ISA cards (KT, KTX, KTX-D and 5136-SD-ISA) require that a unique interrupt level be selected. This setting must match the hardware configuration. To disable interrupts, select None.

**PCI Card Instance**: Select the installed card instance number (based on similar PCI card instances on the computer). If the card is the first PCI installed, the PCI Card Instance should be 0. If the next PCI card installed is from the same vendor, the instance for that card should be 1 (and so forth). If the next PCI card is from a different vendor and is the first card of that type installed, the instance should be 0. Up to four concurrent card instances are allowed (0, 1, 2, 3)

**PKTX Channel**: Indicate the PKTX channel that the server should use for communications. PKTX cards come in single channel (PKTX) or dual channel card (PKTX-D). When a PKTX/PKTX-D card / dual-channel card is used, the driver must know which channel to use. When using the PKTX-D card, select the channel for the device: 1 for PKTX Channel 1A and 2 for PKTX Channel 2.

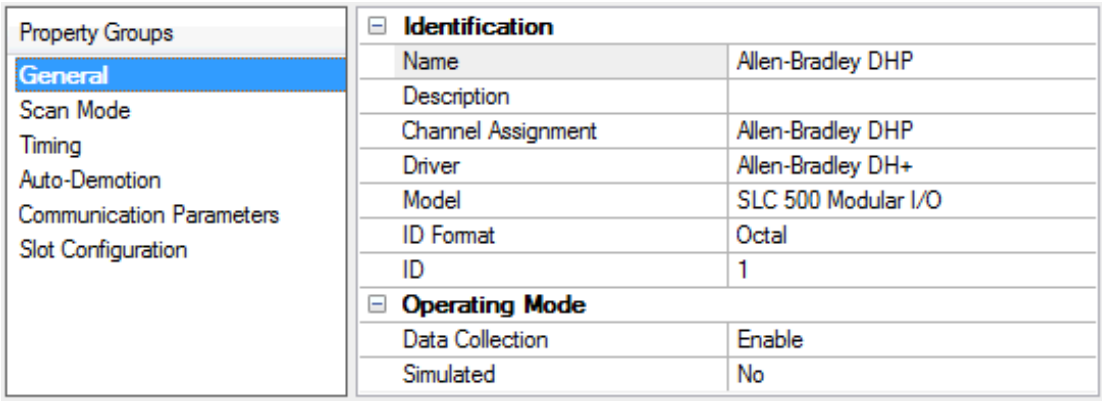
## Device Setup

The maximum number of devices supported per channel is 1024.

Device setup includes configuration of the following property groups:



## Device Properties - General




### 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**: The specific version of the device.

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

**ID**: Enter the unique network address of the device.

### 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.

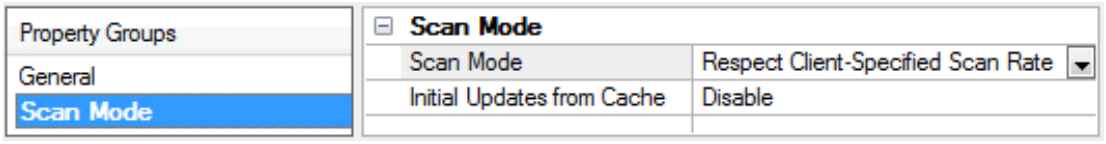
**Notes:**

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 subscribed clients. Descriptions of the options are:

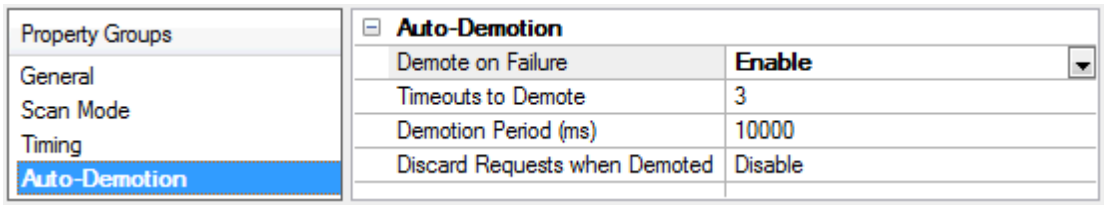
- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the maximum scan rate to be used. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  **Note**: When the server 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 - 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.

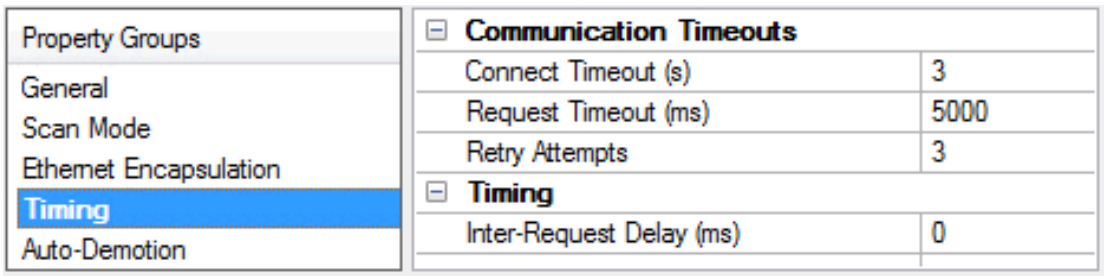
**Timeouts to Demote**: Specify how many successive cycles of request timeouts and retries occur before the device is placed off-scan. The valid range is 1 to 30 successive failures. The default is 3.

**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 - Timing

The device Communications Timeouts 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. Communications Timeouts properties are specific to each configured device.



### 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.

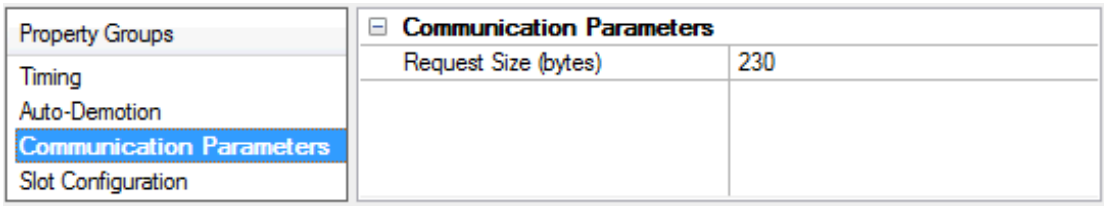
**Retry Attempts**: This property specifies how many times the driver retries a communications request before considering the request to have failed and the device to be in error. The valid range is 1 to 10. The default is typically 3, but can vary depending on the driver's specific nature. The number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

### 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 supported by the driver.

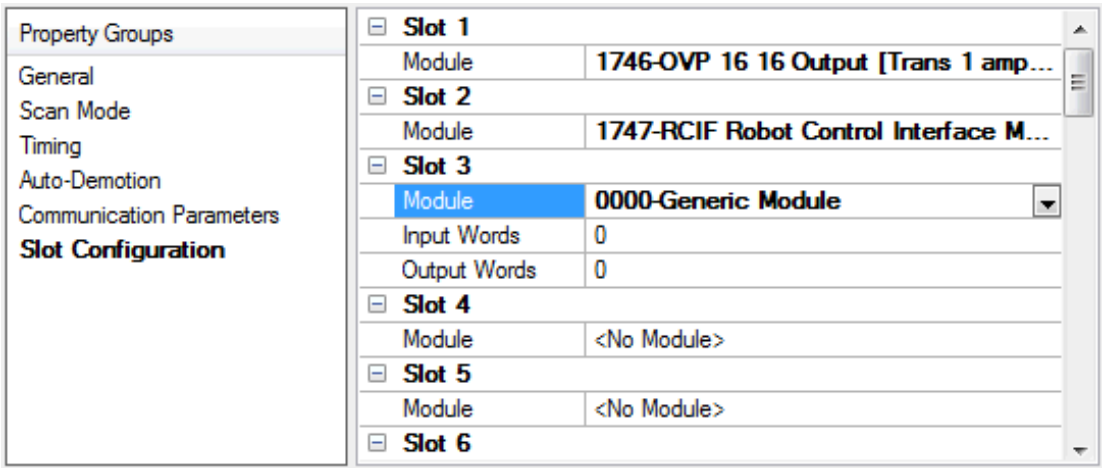
## Device Properties - Communications Parameters




**Request Size**: Indicate the number of bytes that may be requested from a device at one time. Driver performance can be refined by configuring the request size to one of the following settings: 32, 64, 128, or 230 bytes. The default is 230 bytes.

## Slot Configuration

SLC500 models (modular I/O racks) must be configured if the I/O is to be accessed by the driver. Up to 30 slots can be configured per device.




**Slot *n***: the physical slot being configured. Use the plus icon to expand the properties.

**Module**: Select the type of module in the slot from the options available in the drop-down list.

**Input Words**: If required by the module selected, enter the maximum number of Input Words for this module.

**Output Words**: If required by the module selected, enter the maximum number of Output Words for this module.

To use slot configuration, follow the instructions below:

1. Select the slot to be configured by clicking on the row in the module list box.
2. To select a module, click on it from the available modules drop-down list.
3. Configure the Input Words and Output Words if necessary.
4. To remove a slot/module, select **No Module** from the available modules drop-down list.
5. When complete, click **OK**.

**Tips:**

- Use the 0000-Generic Module to configure I/O that is not contained in the list of Available Modules.
- The module selections available are the same as those in the Allen Bradley APS software.

**Note:** It is common to have open slots in the rack that do not contain a physical module. To correctly access data for the various slots that do contain a module, the preceding module(s) must have the correct number of words mapped. For example, if only interested in the I/O in slot 3, but slots 1 and 2 contain I/O modules, the correct modules must be selected for slots 1, 2, and 3 from this slot configuration group.

**0000-Generic Module**
Use the Generic Module to map Input and Output words for modules that are not represented in the list of available modules. To correctly use the Generic Module, users must know the number of Input and Output words required for each module.

- *Consult Allen-Bradley I/O user manual documentation to confirm Input and Output requirements and be aware that requirements may be different based on Class 1 or Class 3 operation.*
- *For information on the number of input and output words available for each I/O module, refer to* ***Modular I/O Selection Guide****.*

## Modular I/O Selection Guide

The following table lists the number of input and output words available for each I/O module available for slot configuration.

| Module Type | Input Words | Output Words |
|---|---|---|
| 1746-I*8 Any 8 pt Discrete Input Module | 1 | 0 |
| 1746-I*16 Any 16 pt Discrete Input Module | 1 | 0 |
| 1746-I*32 Any 32 pt Discrete Input Module | 2 | 0 |
| 1746-O*8 Any 8 pt Discrete Output Module | 0 | 1 |
| 1746-O*16 Any 16 pt Discrete Output Module | 0 | 1 |
| 1746-O*32 Any 32 pt Discrete Output Module | 0 | 2 |
| 1746-IA4 4 Input 100/120 VAC | 1 | 0 |
| 1746-IA8 8 Input 100/120 VAC | 1 | 0 |
| 1746-IA16 16 Input 100/120 VAC | 1 | 0 |
| 1746-IB8 8 Input (Sink) 24 VDC | 1 | 0 |
| 1746-IB16 16 Input (Sink) 24 VDC | 1 | 0 |
| 1746-IB32 32 Input (Sink) 24 VDC | 2 | 0 |
| 1746-IG16 16 Input [TTL] (Source) 5VDC | 1 | 0 |
| 1746-IM4 4 Input 200/240 VAC | 1 | 0 |
| 1746-IM8 8 Input 200/240 VAC | 1 | 0 |
| 1746-IM16 16 Input 200/240 VAC | 1 | 0 |
| 1746-IN16 16 Input 24 VAC/VDC | 1 | 0 |
| 1746-ITB16 16 Input [Fast] (Sink) 24 VDC | 1 | 0 |
| 1746-ITV16 16 Input [Fast] (Source) 24 VDC | 1 | 0 |
| 1746-IV8 8 Input (Source) 24 VDC | 1 | 0 |
| 1746-IV16 16 Input (Source) 24 VDC | 1 | 0 |
| 1746-IV32 32 Input (Source) 24 VDC | 2 | 0 |
| 1746-OA8 8 Output (Triac) 100/240 VAC | 0 | 1 |
| 1746-OA16 16 Output (Triac) 100/240 VAC | 0 | 1 |
| 1746-OB8 8 Output [Trans] (Source) 10/50 VDC | 0 | 1 |
| 1746-OB16 16 Output [Trans] (Source) 10/50 VDC | 0 | 1 |
| 1746-OB32 32 Output [Trans] (Source) 10/50 VDC | 0 | 2 |
| 1746-OBP16 16 Output [Trans 1 amp] (SRC) 24 VDC | 0 | 1 |
| 1746-OV8 8 Output [Trans] (Sink) 10/50 VDC | 0 | 1 |
| 1746-OV16 16 Output [Trans] (Sink) 10/50 VDC | 0 | 1 |
| 1746-OV32 32 Output [Trans] (Sink) 10/50 VDC | 0 | 2 |
| 1746-OW4 4 Output [Relay] VAC/VDC | 0 | 1 |
| 1746-OW8 8 Output [Relay] VAC/VDC | 0 | 1 |
| 1746-OW16 16 Output [Relay] VAC/VDC | 0 | 1 |
| 1746-OX8 8 Output [Isolated Relay] VAC/VDC | 0 | 1 |

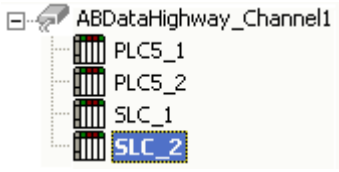
| Module Type | Input Words | Output Words |
|---|---|---|
| 1746-OVP16 16 Output [Trans 1 amp] (Sink) 24VDC3 | 0 | 1 |
| 1746-IO4 2 In 100/120 VAC 2 Out [Rly] VAC/VDC3 | 1 | 1 |
| 1746-IO8 4 In 100/120 VAC 4 Out [Rly] VAC/VDC4 | 1 | 1 |
| 1746-IO12 6 In 100/120 VAC 6 Out [Rly] VAC/VDC | 1 | 1 |
| 1746-NI4 4 Ch Analog Input | 4 | 0 |
| 1746-NIO4I Analog Comb 2 in & 2 Current Out | 2 | 2 |
| 1746-NIO4V Analog Comb 2 in & 2 Voltage Out | 2 | 2 |
| 1746-NO4I 4 Ch Analog Current Output | 0 | 4 |
| 1746-NO4V 4 Ch Analog Voltage Output | 0 | 4 |
| 1746-NT4 4 Ch Thermocouple Input Module | 8 | 8 |
| 1746-NR4 4 Ch Rtd/Resistence Input Module | 8 | 8 |
| 1746-HSCE High Speed Counter/Encoder | 8 | 1 |
| 1746-HS Single Axis Motion Controller | 4 | 4 |
| 1746-OG16 16 Output [TLL] (SINK) 5 VDC | 0 | 1 |
| 1746-BAS Basic Module 500 5/01 Configuration | 8 | 8 |
| 1746-BAS Basic Module 5/02 Configuration | 8 | 8 |
| 1747-DCM Direct Communication Module (1/4 Rack) | 2 | 2 |
| 1747-DCM Direct Communication Module (1/2 Rack) | 4 | 4 |
| 1747-DCM Direct Communication Module (3/4Rack) | 6 | 6 |
| 1747-DCM Direct Communication Module (Full Rack) | 8 | 8 |
| 1747-SN Remote I/O Scanner | 32 | 32 |
| 1747-DSN Distributed I/O Scanner 7 Blocks | 8 | 8 |
| 1747-DSN Distributed I/O Scanner 30 Blocks | 32 | 32 |
| 1747-KE Interface Module, Series A | 1 | 0 |
| 1747-KE Interface Module, Series B | 8 | 8 |
| 1746-NI8 8 Ch Analog Input, Class 1 | 8 | 8 |
| 1746-NI8 8 Ch Analog Input, Class 3 | 16 | 12 |
| 1746-IC16 16 Input (Sink) 48 VDC | 1 | 0 |
| 1746-IH16 16 Input [Trans] (Sink) 125 VDC | 1 | 0 |
| 1746-OAP12 12 Output [Triac] 120/240 VDC | 0 | 1 |
| 1746-OB6EI 6 Output [Trans] (Source) 24 VDC | 0 | 1 |
| 1746-OB16E 16 Output [Trans] (Source) Protected | 0 | 1 |
| 1746-OB32E 32 Output [Trans] (Source) 10/50 VDC | 0 | 2 |
| 1746-OBP8 8 Output [Trans 2 amp] (Source) 24 VDC | 0 | 1 |
| 1746-IO12DC 6 Input 12 VDC, 6 Output [Rly | 1 | 1 |
| 1746-INI4I Analog 4 Ch. Isol. Current Input | 8 | 8 |
| 1746-INI4VI Analog 4 Ch. Isol. Volt./Current Input | 8 | 8 |
| 1746-INT4 4 Ch. Isolated Thermocouple Input | 8 | 8 |
| 1746-NT8 Analog 8 Ch Thermocouple Input | 8 | 8 |
| 1746-HSRV Motion Control Module | 12 | 8 |
| 1746-HSTP1 Stepper Controller Module | 8 | 8 |
| 1747-MNET MNET Network Comm Module | 0 | 0 |
| 1746-QS Synchronized Axes Control Module | 32 | 32 |

| Module Type | Input Words | Output Words |
|---|---|---|
| 1747-QV Open Loop Velocity Control | 8 | 8 |
| 1747-RCIF Robot Control Interface Module | 32 | 32 |
| 1747-SCNR ControlNet SLC Scanner | 32 | 32 |
| 1747-SDN DeviceNet Scanner Module | 32 | 32 |
| 1394-SJT GMC Turbo System | 32 | 32 |
| 1203-SM1 SCANport Comm Module - Basic | 8 | 8 |
| 1203-SM1 SCANport Comm Module - Enhanced | 32 | 32 |
| AMCI-1561 AMCI Series 1561 Resolver Module | 8 | 8 |

## Optimizing Communications

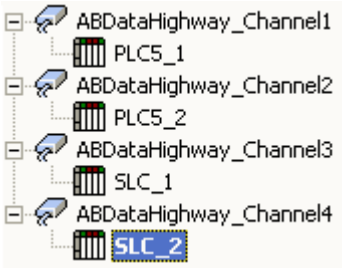
The Allen-Bradley Data Highway Plus Driver is designed to provide the best performance with the least amount of impact on the system's overall performance. While the driver is fast, there are a couple of guidelines that can be used to control and optimize the application and gain maximum performance.

The server refers to communications protocols like Allen-Bradley 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 must be defined under that channel. Each of these devices represents a single Allen-Bradley PLC from which data is collected. While this approach to defining the application provides a high level of performance, it doesn't take full advantage of the 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 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 Allen-Bradley Data Highway Plus Driver could only define one single channel, the example shown above would be the only option available; however, the driver can define up to 100 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 to improve performance is shown below.




Each device has now been defined under its own channel. In this configuration, a single path of execution is dedicated to the task of gathering data from each device. If the application has 100 or fewer channels, it can be optimized as shown here.

The performance improves even if the application has more than 100 channels. While 100 or fewer channels may be ideal, the application still benefits from additional channels. Although spreading the device load across all channels causes the server to move from device to device again, it can now do so with far fewer devices to process on a single channel.

## Address Descriptions

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

General Addressing
SLC Family Open Addressing
PLC-5 Family Addressing

## General Addressing

These general addresses pertain to both the SLC50/5 and PLC-5.

Output Files
Input Files
Status Files
Binary Files
Timer Files
Counter Files
Control Files
Integer Files
Float Files
ASCII Files
String Files

*See Also: SLC Family Open Addressing, PLC-5 Family Addressing*

## Output Files

The syntax for accessing data in the output file depends on the PLC model. Data locations are Read/Write for PLC-5 Models and Read Only for all other models. Arrays are not supported. The default data type for all syntax is shown in **bold**.

**Note:** PLC-5 Model Word and bit address information is in octal. This follows the convention of the programming software.

### PLC-5 Model Syntax

| Syntax | Data Type |
|---|---|
| O:<word> | Short, **Word**, BCD |
| O:<word>/<bit> | **Boolean** |
| O/bit | **Boolean** |

### SLC Family Open Models (Modular I/O) Syntax

| Syntax | Data Type |
|---|---|
| O:<slot> | Short, **Word**, BCD |
| O:<slot>.<word> | Short, **Word**, BCD |
| O:<slot>/<bit> | **Boolean** |
| O:<slot>.<word>/<bit> | **Boolean** |

The following slot and word locations are allowed for each model.

| PLC Model | Min Slot | Max Slot | Max Word |
|---|---|---|---|
| SLC Family | 1 | 30 | * |
| PLC-5 Family | NA | NA | 277 |

**For information on the number of input or output words available for each I/O module, refer to **Modular I/O Selection Guide**.*

*For more information on slot configuration, refer to **Device Setup**.*

### Examples

| PLC-5 | Addresses |
|---|---|
| O:0 | word 0 |
| O:37 | word 31 (37 octal=31 decimal) |
| O/42 | bit 34 (42 octal=34 decimal) |
| O:2/2 | bit 2 word 2 (same as O/42) |

| SLC Family | Addresses |
|---|---|
| O:1 | word 0 slot 1 |
| O:1.0 | word 0 slot 1 (same as O:1) |
| O:12 | word 0 slot 12 |
| O:12.2 | word 2 slot 12 |
| O:4.0/0 | bit 0 word 0 slot 4 |
| O:4/0 | bit 0 slot 4 (same as O:4.0/0) |
| O:4.2/0 | bit 0 word 2 slot 4 |
| O:4/32 | bit 32 slot 4 (same as O:4.2/0) |

## Input Files

The syntax for accessing data in the input file depends on the PLC model. Data locations are Read/Write for PLC-5 Models and Read Only for all other models. Arrays are not supported. The default data type for all syntax is shown in **bold**.

**Note:** PLC-5 Model Word and bit address information is in octal. This follows the convention of the programming software.

### PLC-5 Model Syntax

| Syntax | Data Type |
|---|---|
| I:<word> | Short, **Word**, BCD |
| I:<word>/<bit> | **Boolean** |
| I/bit | **Boolean** |

### SLC Family Open Models (Modular I/O) Syntax

| Syntax | Data Type |
|---|---|
| I:<slot> | Short, **Word**, BCD |
| I:<slot>.<word> | Short, **Word**, BCD |

| Syntax | Data Type |
|---|---|
| I:<slot>/<bit> | **Boolean** |
| I:<slot>.<word>/<bit> | **Boolean** |

The following slot and word locations are allowed for each model.

| PLC Model | Min Slot | Max Slot | Max Word |
|---|---|---|---|
| SLC 5/05 | 1 | 30 | * |
| PLC-5 Family | NA | NA | 277 |

*For information on the number of input or output words available for each I/O module, refer to **Modular I/O Selection Guide**.

*For more information on slot configuration, refer to **Device Setup**.*

### Examples

| PLC-5 | Addresses |
|---|---|
| I:0 | word 0 |
| I:10 | word 8 (10 octal=8 decimal) |
| I/20 | bit 16 (20 octal=16 decimal) |
| I:1/0 | bit 0 word 1 (same as I/20) |

| SLC Family | Addresses |
|---|---|
| I:1 | word 0 slot 1 |
| I:1.0 | word 0 slot 1 (same as I:1) |
| I:12 | word 0 slot 12 |
| I:12.2 | word 2 slot 12 |
| I:4.0/0 | bit 0 word 0 slot 4 |
| I:4/0 | bit 0 slot 4 (same as I:4.0/0) |
| I:4.2/0 | bit 0 word 2 slot 4 |
| I:4/32 | bit 32 slot 4 (same as I:4.2/0) |

## Binary Files

To access a binary file, specify a file number, a word and (optionally) a bit within the word. The default data type for all syntax is shown in **bold**.

| Syntax | Data Type |
|---|---|
| B<file>:<word> | Short, **Word**, BCD, DWord, Long, LBCD |
| B<file>:<word> [rows][cols] | Short, **Word**, BCD, DWord, Long, LBCD* |
| B<file>:<word> [cols] | Short, **Word**, BCD, DWord, Long, LBCD* |
| B<file>:<word>/<bit> | **Boolean** |
| B<file>/<bit> | **Boolean** |

*Array types.

The number of array elements (in bytes) cannot exceed the block request size specified. This means that array size cannot exceed 16 words given a block request size of 32 bytes. For more information, refer to **Communications Parameters**.

The following file numbers and maximum word locations are allowed for each model. The maximum word location is one less when accessing as a 32-bit data type (Long, DWord or Long BCD).

| PLC Model | File Number | Max Word |
|---|---|---|
| SLC 5/05 Open | 3, 9-999 | 999 |
| PLC-5 Family | 3-999 | 1999 |

| Example | Description |
|---|---|
| B3:0 | word 0 |
| B3/26 | bit 26 |
| B12:4/15 | bit 15 word 4 |
| B3:10 [20] | 20 element array starting at word 10 |
| B15:0 [6] [6] | 6 by 6 element array starting at word 0 |

## Status Files

To access a status file, specify a word and (optionally) a bit within the word. The default data type for all syntax is shown in **bold**.

| Syntax | Data Type |
|---|---|
| S:<word> | Short, **Word**, BCD, DWord, Long, LBCD |
| S:<word> [rows][cols] | Short, **Word**, BCD, DWord, Long, LBCD* |
| S:<word> [cols] | Short, **Word**, BCD, DWord, Long, LBCD* |
| S:<word>/<bit> | **Boolean** |
| S/bit | **Boolean** |

*Array types.

The number of array elements (in bytes) cannot exceed the block request size specified. This means that array size cannot exceed 16 words given a block request size of 32 bytes. For more information, refer to **Communications Parameters**.

The following word locations are allowed for each model. The maximum word location is one less when accessing as a 32-bit data type (Long, DWord or Long BCD).

| PLC Model | Max Word |
|---|---|
| SLC Family | 999 |
| PLC-5 Family | 999 |

| Example | Description |
|---|---|
| S:0 | word 0 |
| S/26 | bit 26 |
| S:4/15 | bit 15 word 4 |

| Example | Description |
|---|---|
| S:10 [16] | 16 element array starting at word 10 |
| S:0 [4] [8] | 4 by 8 element array starting at word 0 |

## Timer Files

Timer files are a structured type whose data is accessed by specifying a file number, an element and a field. The default data type depends on the field being accessed. Integer fields receive a default data type of Word.

| Syntax | Data Type |
|---|---|
| T<file>:<element>.<field> | Depends on field |

The following file numbers and maximum element are allowed for each model.

| PLC Model | File Number | Max Element |
|---|---|---|
| SLC Family | 4, 9-999 | 999 |
| PLC-5 Family | 3-999 | 1999 |

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation. The default data type is shown in **bold**.

| Element Field | Data Type | Access |
|---|---|---|
| ACC | Short, **Word**, Boolean | Read/Write |
| PRE | Short, **Word**, Boolean | Read/Write |
| DN | **Boolean** | Read Only |
| TT | **Boolean** | Read Only |
| EN | **Boolean** | Read Only |
| S | **Word**, Short, Boolean | Read/Write |

| Example | Description |
|---|---|
| T4:0.ACC | Accumulator of timer 0 file 4. |
| T4:3.ACC/0 | Accumulator of timer 3 file 4 (bit 0). |
| T4:10.DN | Done bit of timer 10 file 4. |
| T15:0.PRE | Preset of timer 0 file 15. |
| T4:3.PRE/1 | Preset of timer 3 file 4 (bit 1). |
| T4:0.S | Status word of timer 0 file 4. |
| T4:0.S/12 | Status word of timer 0 file 4 (bit 12). |

## Counter Files

Counter files are a structured type whose data is accessed by specifying a file number, an element and a field. The default data type depends on the field being accessed. Integer fields receive a default data type of Word.

| Syntax | Data Type |
|---|---|
| C<file>:<element>.<field> | Depends on field |

The following file numbers and maximum element are allowed for each model.

| PLC Model | File Number | Max Element |
|---|---|---|
| SLC Family | 5, 9-999 | 999 |
| PLC-5 Family | 3-999 | 1999 |

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation. The default data type is shown in **bold**.

| Element Field | Data Type | Access |
|---|---|---|
| ACC | **Word**, Short, Boolean | Read/Write |
| PRE | **Word**, Short, Boolean | Read/Write |
| UA | **Boolean** | Read Only |
| UN | **Boolean** | Read Only |
| OV | **Boolean** | Read Only |
| DN | **Boolean** | Read Only |
| CD | **Boolean** | Read Only |
| CU | **Boolean** | Read Only |
| S | **Word**, Short, Boolean | Read/Write |

| Example | Description |
|---|---|
| C5:0.ACC | Accumulator of counter 0 file 5 |
| C5:2.ACC/2 | Accumulator of counter 2 file 5 (bit 2) |
| C5:10.DN | Done bit of counter 10 file 5 |
| C15:0.PRE | Preset of counter 0 file 15 |
| C5:2.PRE/3 | Preset of counter 2 file 5 (bit 3) |
| C5:0.S | Status word of counter 0 file 5 |
| C5:0.S/9 | Status word of counter 0 file 5 (bit 9) |

## Control Files

Control files are a structured type whose data is accessed by specifying a file number, an element and a field. The default data type depends on the field being accessed. Integer fields receive a default data type of Word.

| Syntax | Data Type |
|---|---|
| R<file>:<element>.<field> | Depends on field |

The following file numbers and maximum element are allowed for each model.

| PLC Model | File Number | Max Element |
|---|---|---|
| SLC Family | 6, 9-999 | 999 |
| PLC-5 Family | 3-999 | 1999 |

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation. The default data type is shown in **bold**.

| Element Field | Data Type | Access |
|---|---|---|
| LEN | **Word**, Short, Boolean | Read/Write |
| POS | **Word**, Short, Boolean | Read/Write |
| FD | **Boolean** | Read Only |
| IN | **Boolean** | Read Only |
| UL | **Boolean** | Read Only |
| ER | **Boolean** | Read Only |
| EM | **Boolean** | Read Only |
| DN | **Boolean** | Read Only |
| EU | **Boolean** | Read Only |
| EN | **Boolean** | Read Only |
| S | **Word**, Short, Boolean | Read/Write |

| Example | Description |
|---|---|
| R6:0.LEN | Length field of control 0 file 6 |
| R6:1.LEN/5 | Length field of control 1 file 6 (bit 5) |
| R6:10.DN | Done bit of control 10 file 6 |
| R15:18.POS | Position field of control 18 file 15 |
| R6:1.POS/4 | Position field of control 1 file 6 (bit 4) |
| R6:0.S | Status word of control 0 file 6 |
| R6:0.S/6 | Status word of control 0 file 6 (bit 6) |

## Integer Files

To access an integer file, specify a file number, a word and an optional bit in the word. The default data type for all syntax is shown in **bold**.

| Syntax | Data Type |
|---|---|
| N<file>:<word> | Short, **Word**, BCD, DWord, Long, LBCD |
| N<file>:<word> [rows][cols] | Short, **Word**, BCD, DWord, Long, LBCD* |
| N<file>:<word> [cols] | Short, **Word**, BCD, DWord, Long, LBCD* |
| N<file>:<word>/<bit> | **Boolean** |
| N<file>/bit | **Boolean** |

*Array types.

The number of array elements (in bytes) cannot exceed the block request size specified. This means that array size cannot exceed 16 words given a block request size of 32 bytes. For more information, refer to **Communications Parameters**.

The following file numbers and maximum word locations are allowed for each model. The maximum word location is one less when accessing a 32-bit data type (Long, DWord or Long BCD).

| PLC Model | File Number | Max Word |
|---|---|---|
| SLC Family | 7, 9-999 | 999 |
| PLC-5 Family | 3-999 | 1999 |

| Example | Description |
|---|---|
| N7:0 | word 0 |
| N7/26 | bit 26 |
| N12:4/15 | bit 15 word 4 |
| N7:10 [8] | 8 element array starting at word 10 |
| N15:0 [4] [5] | 4 by 5 element array starting at word 0 |

## Float Files

To access data in a Float file, specify a file number and an element. The only data type allowed is Float.

| Syntax | Data Type |
|---|---|
| F<file>:<element> | Float |
| F<file>:<element> [rows][cols] | Float array |
| F<file>:<element> [cols] | Float array |

The number of array elements (in bytes) cannot exceed the block request size specified. This means that array size cannot exceed 8 Floats given a block request size of 32 bytes). For more information, refer to **Communications Parameters**.

The following file numbers and maximum word locations are allowed for each model.

| PLC Model | File Number | Max Word |
|---|---|---|
| SLC Family | 8-999 | 999 |
| PLC-5 Family | 3-999 | 1999 |

| Example | Description |
|---|---|
| F8:0 | Float 0 |
| F8:10 [16] | 16 element array starting at word 10 |
| F15:0 [4] [4] | 16 element array starting at word 0 |

## ASCII Files

To access data in an ASCII file, specify a file number and character location. The default data type for all syntax is shown in **bold**.

| Syntax | Data Type |
|---|---|
| A<file>:<char> | **Char**, Byte |
| A<file>:<char> [rows][cols] | **Char**, Byte |
| A<file>:<char> [cols] | **Char**, Byte |
| A<file>:<word> offset>/length | **String** |

**Note:** The number of array elements cannot exceed the block request size specified. For more information, refer to **Communications Parameters**.

Internally, the PLC packs two characters per word in the file, with the high byte containing the first character and the low byte containing the second character. The PLC programming software allows access at the word level or two-character level. The Allen-Bradley Data Highway Plus Driver allows accessing to the character level. This means that if the programming software A10:0=AB is used,'A' would be stored in the high byte of

A10:0 and'B' would be stored in the low byte. If the driver is used, two assignments would be made: A10:0=A and A10:1=B. This would result in the same data being stored in the PLC memory.

Referencing this file as string data allows access to data at word boundaries like the programming software. The length can be up to 236 characters. If a string that is sent to the device is smaller in length than the length specified by the address, the driver null terminates the string before sending it down to the controller.

The following file numbers and maximum character locations are allowed for each model.

| PLC Model | File Number | Max Character |
|---|---|---|
| SLC Family | 9-999 | 1999 |
| PLC-5 Family | 3-999 | 1999 |

**Note:** All SLC Family PLCs do not support ASCII file types. For more information, refer to PLC documentation.

| Example | Description |
|---|---|
| A9:0 | Character 0 (high byte of word 0) |
| A27:10 [80] | 80 character array starting at character 10 |
| A15:0 [4] [16] | 4 by 16 character array starting at character 0 |
| A62:0/32 | 32 character string starting at word offset 0 |

## String Files

To access data in a string file, specify a file number and an element. The only data type allowed is string, which are 82 character null terminated arrays. The driver places the null terminator based on the string length returned by the PLC.

**Note:** Arrays of strings are not supported.

| Syntax | Data Type |
|---|---|
| ST<file>:<element> | String |

The following file numbers and maximum word locations are allowed for each model.

| PLC Model | File Number | Max Word | Element Field |
|---|---|---|---|
| SLC Family | 9-999 | 999 | NA |
| PLC-5 Family | 3-999 | 999 | .LEN |

| Example | Description |
|---|---|
| ST9:0 | string 0 |
| ST18:10 | string 10 |

## SLC Family Open Addressing

### Open Addressing

The actual number of addresses available depends on the model of the PLC being used. The ranges have been opened up to allow for maximum flexibility with future models. If the driver finds at Runtime that an address is not present in the device, it posts an error message and remove the tag from its scan list.

**Note:** This model has no model specific addressing.

***See Also:*** *General Addressing*

## PLC-5 Family Addressing

### General Addressing

General Addressing

### Model Specific Addressing

BCD Files
PID Files
Message Files
Block Transfer Files
SC Files

## BCD Files

To access data in a BCD file, specify a file number and a word. The only data types allowed are BCD and Long BCD. The default data type is always BCD.

| Syntax | Data Type |
|---|---|
| D<file>:<word> | **BCD**, LBCD, Boolean |
| D<file>:<word> [rows][cols] | **BCD**, LBCD* |
| D<file>:<word> [cols] | **BCD**, LBCD* |

*Array types.

The number of array elements (in bytes) cannot exceed the block request size specified. This means that array size cannot exceed 16 BCDs given a block request size of 32 bytes. For more information, refer to Communications Parameters.

The following file numbers and maximum word locations are allowed for each model.

| PLC Model | File Number | Max Word |
|---|---|---|
| SLC 5/05 Open | NA | NA |
| PLC-5 Family | 3-999 | 999 |

| Example | Description |
|---|---|
| D9:0 | Word 0 |
| D8:9/10 | File 8 BCD element 9 (bit 10) |
| D27:10 [16] | 16 element array starting at word 10 |
| D15:0 [4] [8] | 32 element array starting at word 0 |

## PID Files

PID files are a structured type whose data is accessed by specifying a file number, an element and a field. The default data type depends on the field being accessed. Integer fields receive a default data type of Word.

| Syntax | Data Type |
|---|---|
| PD<file>:<element>.<field> | Depends on field |

The following file numbers and maximum element are allowed for each model.

| PLC Model | File Number | Max Element |
|---|---|---|
| SLC 5/05 Open | NA | NA |
| PLC-5 Family | 3-999 | 999 |

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation.

| Element Field | Data Type | Access |
|---|---|---|
| SP | Real | Read/Write |
| KP | Real | Read/Write |
| KI | Real | Read/Write |
| KD | Real | Read/Write |
| BIAS | Real | Read/Write |
| MAXS | Real | Read/Write |
| MINS | Real | Read/Write |
| DB | Real | Read/Write |
| SO | Real | Read/Write |
| MAXO | Real | Read/Write |
| MINO | Real | Read/Write |
| UPD | Real | Read/Write |
| PV | Real | Read/Write |
| ERR | Real | Read/Write |
| OUT | Real | Read/Write |
| PVH | Real | Read/Write |
| PVL | Real | Read/Write |
| DVP | Real | Read/Write |
| DVN | Real | Read/Write |
| PVDB | Real | Read/Write |
| DVDB | Real | Read/Write |
| MAXI | Real | Read/Write |
| MINI | Real | Read/Write |
| TIE | Real | Read/Write |
| FILE | Word, Short, Boolean | Read/Write |
| ELEM | Word, Short, Boolean | Read/Write |
| EN | Boolean | Read/Write |
| CT | Boolean | Read/Write |
| CL | Boolean | Read/Write |
| PVT | Boolean | Read/Write |
| DO | Boolean | Read/Write |

| Element Field | Data Type | Access |
|---|---|---|
| SWM | Boolean | Read/Write |
| CA | Boolean | Read/Write |
| MO | Boolean | Read/Write |
| PE | Boolean | Read/Write |
| INI | Boolean | Read/Write |
| SPOR | Boolean | Read/Write |
| OLL | Boolean | Read/Write |
| OLH | Boolean | Read/Write |
| EWD | Boolean | Read/Write |
| DVNA | Boolean | Read/Write |
| DVHA | Boolean | Read/Write |
| PVLA | Boolean | Read/Write |
| PVHA | Boolean | Read/Write |

| Example | Description |
|---|---|
| PD14:0.SP | Set point field of PD 0 file 14 |
| PD18:6.EN | Status enable bit of PD 6 file 18 |
| PD21:5.FILE/8 | File number for PD 5 file 21 (bit 8) |
| PD21:5.ELEM/9 | Element number for PD 5 file 21 (bit 9) |

## Message Files

Message files are a structured type whose data is accessed by specifying a file number, an element and a field. The default data type depends on the field being accessed. Integer fields receive a default data type of Word.

| Syntax | Data Type |
|---|---|
| MG<file>:<element>.<field> | Depends on field |

The following file numbers and maximum element are allowed for each model.

| PLC Model | File Number | Max Element |
|---|---|---|
| SLC 5/05 Open | NA | NA |
| PLC-5 Family | 3-999 | 999 |

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation. The default data types are shown in **bold**.

| Element Field | Data Type | Access |
|---|---|---|
| ERR | Short, **Word** | Read/Write |
| RLEN | Short, **Word** | Read/Write |
| DLEN | Short, **Word** | Read/Write |
| EN | Boolean | Read/Write |
| ST | Boolean | Read/Write |
| DN | Boolean | Read/Write |

| Element Field | Data Type | Access |
|---|---|---|
| ER | Boolean | Read/Write |
| CO | Boolean | Read/Write |
| EW | Boolean | Read/Write |
| NR | Boolean | Read/Write |
| TO | Boolean | Read/Write |

| Example | Description |
|---|---|
| MG14:0.RLEN | Requested length field of MG 0 file 14 |
| MG18:6.CO | Continue bit of MG 6 file 18 |

## Block Transfer Files

Block transfer files are a structured type whose data is accessed by specifying a file number, an element and a field. The default data type depends on the field being accessed. Integer fields receive a default data type of Word.

| Syntax | Data Type |
|---|---|
| BT<file>:<element>.<field> | Depends on field |

The following file numbers and maximum element are allowed for each model.

| PLC Model | File Number | Max Word |
|---|---|---|
| SLC 5/05 Open | NA | NA |
| PLC-5 Family | 3-999 | 1999 |

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation.

| Element Field | Data Type | Access |
|---|---|---|
| RLEN | **Word**, Short | Read/Write |
| DLEN | **Word**, Short | Read/Write |
| FILE | **Word**, Short, Boolean | Read/Write |
| ELEM | **Word**, Short, Boolean | Read/Write |
| RW | **Boolean** | Read/Write |
| ST | **Boolean** | Read/Write |
| DN | **Boolean** | Read/Write |
| ER | **Boolean** | Read/Write |
| CO | **Boolean** | Read/Write |
| EW | **Boolean** | Read/Write |
| NR | **Boolean** | Read/Write |
| TO | **Boolean** | Read/Write |
| EN | **Boolean** | Read/Write |
| S | **Word**, Short, Boolean | Read/Write |

| Example | Description |
|---|---|
| BT14:0.RLEN | Requested length field of BT 0 file 14 |
| BT18:6.CO | Continue bit of BT 6 file 18 |
| BT12:4.FILE/6 | File number for BT 4 file 12 (bit 6) |
| BT12:4.ELEM/7 | Element number for BT 4 file 12 (bit 7) |
| BT12:0.S | Status word of BT 0 file 12 |
| BT12:0.S/7 | Status word of BT 0 file 12 (bit 7) |

## SC Files

SC files are a structured type whose data is accessed by specifying a file number, an element and a field. The default data type depends on the field being accessed.

| Syntax | Data Type |
|---|---|
| SC<file>:<element>.<field>/<bit> | Depends on field |

The following file numbers and maximum element are allowed for each model.

| PLC Model | File Number | Max Element |
|---|---|---|
| SLC-500 | NA | NA |
| PLC-5 | 3-999 | 1999 |

The following fields are allowed for each element. For the meaning of each field, refer to the PLC's documentation. The default data type is shown in **bold**.

| Element Field | Data Type | Access |
|---|---|---|
| PRE | **Word**, Short, Boolean | Read/Write |
| TIM | **Word**, Short, Boolean | Read/Write |
| S | **Word**, Short, Boolean | Read/Write |
| DN | **Boolean** | Read/Write |
| ER | **Boolean** | Read/Write |
| OV | **Boolean** | Read/Write |
| LS | **Boolean** | Read/Write |
| SA | **Boolean** | Read/Write |

| Example | Description |
|---|---|
| SC9:0.PRE | Preset word |
| SC9:0.TIM | Active-time bit |
| SC9:0.SA | Scan-active bit |
| SC9:0.FS | First-scan bit |
| SC9:0.LS | Last-scan bit |
| SC9:0.OV | Timer-overflow bit |
| SC9:0.ER | Step-errored bit |
| SC9:0.DN | Done bit |
| SC21:0.S | Status word of SC 0 file 21 |
| SC21:0.S/1 | Status word of SC 0 file 21 (bit 1) |

## Data Types Description

| Data Type | Description |
|---|---|
| Boolean | Single bit |
| Byte | Unsigned 8-bit value |
| Char | Signed 8-bit value |
| Word | Unsigned 16-bit value |
| Short | Signed 16-bit value |
| DWord | Unsigned 32-bit value |
| Long | Signed 32-bit value |
| BCD | Two byte packed BCD, four decimal digits |
| LBCD | Four byte packed BCD, eight decimal digits |
| Float | 32-bit IEEE Floating point |
| String | Null terminated character array |

**Note:** The DWord, Long, and LBCD data types are not native to any of the PLC models.

When referencing a 16-bit location as a 32-bit value, the location referenced is the low word and the next successive location is the high word. For example, if N7:10 is selected as a DWord data type, N7:10 would be the low word and N7:11 would be the high word.

# 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.

## Failed to allocate board.

**Error Type:**
Error

## Failed to allocate memory for board.

**Error Type:**
Error

## Failed to stop board.

**Error Type:**
Error

## Failed to start card. Possible resource conflicts. | Card = '<card>'.

**Error Type:**
Error

**Possible Cause:**

1. Memory conflict.
2. Temporary network problem.
3. Bad memory address selected.
4. The properties selected for channel setup may be incorrect.
5. The card is corrupt.

**Possible Solution:**

1. Some other device may be using the same memory address. Configure the device with a different memory address.
2. Restart the driver.
3. The memory address selected is not within the range. Try a different memory address.
4. Modify the channel properties with appropriate values.
5. Replace the card.

## Failed to perform M16 diagnostic. The card is in 8-bit mode. Change jumper configuration to 16-bit mode. | Card = '<card>'.

### Error Type:
Error

### Possible Cause:
This occurs when the jumper settings on the KTXD card are in 8-bit mode.

### Possible Solution:
Change the jumper settings to 16-bit mode. Refer to KTXD user guide for more information.

## Failed to write to dual port memory on device. | Card = '<card>'.

### Error Type:
Error

### Possible Cause:

1. A network failure occurred.
2. Memory conflict.
3. The properties selected for the channel setup may be incorrect.
4. The card is corrupt.

### Possible Solution:

1. Check for broken links between the card and the host. Make sure that the card is inserted properly in the appropriate slot and restart the driver.
2. Assign a different memory base address.
3. Correct the channel setup properties to appropriate values.
4. Replace the card.

## Failed RAM test on device. | Card = '<card>'.

### Error Type:
Error

### Possible Cause:

1. Memory conflict.
2. Temporary network problem.
3. Bad memory address selected.
4. The properties selected for channel setup may be incorrect.
5. Card is corrupt.

### Possible Solution:

1. Some other device may be using the same memory address. Configure the device with a different memory address.
2. Restart the driver.
3. The memory address selected is not within the range: try a different memory address.
4. Correct the channel setup properties to appropriate values.
5. Replace the card.

## Failed CTC test on device. | Card = '<card>'.

### Error Type:
Error

### Possible Cause:

1. Memory conflict.
2. Temporary network problem.
3. Bad memory address selected.
4. The properties selected for channel setup may be incorrect.
5. Card is corrupt.

### Possible Solution:

1. Some other device may be using the same memory address. Configure the device with a different memory address.
2. Restart the driver.
3. The memory address selected is not within the range: try a different memory address.
4. Correct the channel setup properties to appropriate values.
5. Replace the card.

## Failed SIO test on device. | Card = '<card>'.

### Error Type:
Error

### Possible Cause:

1. Memory conflict.
2. Temporary network problem.
3. Bad memory address selected.
4. The properties selected for channel setup may be incorrect.
5. Card is corrupt.

### Possible Solution:

1. Some other device may be using the same memory address. Configure the device with a different memory address.
2. Restart the driver.
3. The memory address selected is not within the range: try a different memory address.
4. Correct the channel setup properties to appropriate values.
5. Replace the card.

## Failed to enable card on device. | Card = '<card>'.

### Error Type:
Error

### Possible Cause:

1. Channel properties, such as baud rate, may have been set inaccurately.
2. A network failure occurred.
3. A memory conflict occurred.
4. The card is corrupted.

### Possible Solution:

1. Modify the channel properties with appropriate values.
2. Check for any broken links between the card and the host. Make sure that the card is inserted properly on the appropriate slot and then restart the driver.
3. Assign a different memory base address.
4. Replace the card.
5. Check channel baud rate settings and Device Manager.

## Interrupt is not configured for device. | Card = '<card>'.

### Error Type:
Error

### Possible Cause:

1. The interrupt may not be configured on the device.
2. The interrupt configured on the device may not match the one selected in Channel Setup.

### Possible Solution:

1. Enable the interrupt by setting the correct jumper/DIP switch combination.
2. Set the interrupt property on the channel properties to match the card set up.

## Failed to load KTXPCL.BIN on device. | Card = '<card>'.

### Error Type:
Error

### Possible Cause:
KTXPCL.BIN is unavailable or corrupt.

### Possible Solution:
Locate or reinstall KTXPCL.BIN, then try again.

## Failed to allocate device for board.

### Error Type:
Error

### Possible Cause:
1. The interface card is being used by another application.
2. The interface card is not functional.
3. If it is an ISA interface card, the memory address selected may be inappropriate.

### Possible Solution:
1. Uninstall any other application (such as RSLinx) that might be using the same interface card.
2. Verify that the card is functional and inserted properly in the appropriate slot.
3. Assign the correct memory address.

## Failed to complete diagnosis while performing M16 tests. | Card = '<card>'.

### Error Type:
Error

### Possible Cause:
1. The jumper settings on the KTXD card may be in 8-bit mode.
2. The ISA bus slot may be corrupted.
3. The card memory may be corrupted.
4. One or more binary files related to M16 diagnosis has been corrupted.

### Possible Solution:
1. Change the KTXD card to 16-bit mode. Refer to KTXD user guide for more information.
2. Try a different ISA slot.
3. Replace the card.
4. Reinstall the binary files for M16 diagnosis.

## Failed to complete loading protocol while performing M16 tests. | Card = '<card>'.

**Error Type:**
Error

**Possible Cause:**

1. The jumper settings on the KTXD card may be in 8-bit mode.
2. The ISA bus slot may be corrupted.
3. The card memory may be corrupted.
4. One or more binary files related to M16 diagnosis has been corrupted.

**Possible Solution:**

1. Change the KTXD card to 16-bit mode. Refer to KTXD user guide for more information.
2. Try a different ISA slot.
3. Replace the card.
4. Reinstall the binary files for M16 diagnosis.

## Failed to connect to device. | Card = '<card>'.

**Error Type:**
Error

**Possible Cause:**

1. The card may not be installed correctly.
2. An inappropriate *.inf file may be loaded for this device.

**Possible Solution:**

1. Verify that all the card pins are connected to the slot.
2. Use Device Manager to verify that the right driver is installed for this device. The driver supplied by the OPC server for the card being used must be installed. For more information, refer to Channel Setup.

## Failed to load resource for PCI setup under device. | Card = '<card>'.

**Error Type:**
Error

## Duplicate station ID detected by device. | Card = '<card>'.

**Error Type:**
Error

### Possible Cause:
There is more than one device with the same device ID.

### Possible Solution:
Assign a different station ID (device ID) to one of the duplicate stations.

## Invalid station ID detected by device. | Card = '<card>'.

### Error Type:
Error

### Possible Cause:
The device with the specified ID could not be located in the network.

### Possible Solution:
1. Verify the specified device ID is correct or modify it to the correct ID.
2. Check for broken connections between the card and the device.
3. Verify the device is on the appropriate network.
4. Verify that the device is powered on.

## Error writing to SST ports for device. | Card = '<card>'.

### Error Type:
Error

### Possible Cause:
The ports selected are not working properly, which usually results from conflicts with other devices.

### Possible Solution:
Assign a different port range and restart the driver.

## Failed to take the SST card offline for device. | Card = '<card>'.

### Error Type:
Error

### Possible Cause:
The driver cannot take the card offline due to an invalid or non-existent device.

### Possible Solution:
1. Check for memory conflicts. If there is a conflict, try a different memory address.
2. Check for broken connections between the card and the host.
3. Verify the card is installed correctly.
4. Restart the driver.
5. Restart the computer.

## Bad port/memory detected on SST card while loading module on device. | Card = '<card>'.

**Error Type:**
Error

**Possible Cause:**
A conflict occurred on memory/ports between devices.

**Possible Solution:**

1. Check for memory conflicts. If there is a conflict, try a different memory address.
2. Check for I/O ports conflicts. If there is a conflict, try different port address.
3. The card or section of a card may be corrupted. If so, try different memory/port address.

## Protocol is not reflected onto memory after being loaded on device. | Card = '<card>'.

**Error Type:**
Error

**Possible Cause:**

1. The device is not configured properly.
2. A memory conflict occurred.
3. A problem occurred with the network.
4. The card is corrupt.

**Possible Solution:**

1. Reconfigure the device with appropriate properties.
2. Assign a different unique memory address.
3. Restart the driver.
4. Replace the card.

## SST module could not be loaded because of a card error on device. | Card = '<card>'.

**Error Type:**
Error

## SST card execution error occured on device. | Card = '<card>'.

**Error Type:**
Error

**Possible Cause:**

1. There may be a memory conflict.
2. There may be a temporary problem with the network.
3. Some binary file for the driver may be corrupted.
4. The card may be corrupted.

**Possible Solution:**

1. Reconfigure the device with a different memory address.
2. Restart the driver.
3. Replace the card.

## Failed to load binary resource for device. | Resource = <resource>, Card = '<card>'.

**Error Type:**
Error

**Possible Cause:**
One of the binary files required for loading the protocol may be corrupted.

## Failed to clear dual port memory while performing functionality test for device. | Card = '<card>'.

**Error Type:**
Error

**Possible Cause:**
There may be a conflict between this server and another OPC server supporting the drivers.

**Possible Solution:**
Verify all other OPC servers (such as RSLinx) are completely uninstalled, then restart the PC.

## Failed to remove device from channel map. | Channel ID = <channel>, Card = '<card>'.

**Error Type:**
Error

## Failed to add device to channel map. | Channel ID = <channel>, Card = '<card>'.

**Error Type:**
Error

## Failed to locate PKTXChannel for device. | PKTXChannel = <channel>, Card = '<card>'.

**Error Type:**

Error

### Possible Cause:

1. The card is not installed properly.
2. Card instance ID may be used by other card.
3. The properties selected for channel setup may be incorrect.
4. The card is corrupt.

### Possible Solution:

1. Verify all the pins are connected into the slot.
2. Verify that the instance ID is not used by any other PKTX card.
3. Modify the channel properties with appropriate values.
4. Replace the card.

## Error performing diagnostic test.

### Error Type:

Error

### Possible Cause:

1. The card is not installed properly.
2. The card is corrupt.

### Possible Solution:

1. Verify that all the pins are connected into the slot.
2. Replace the card.

## Unable to read block from device. Frame received contains errors. | Block start address = '<address>'.

### Error Type:

Warning

### Possible Cause:

1. An incorrect frame size was received.
2. A TNS mismatch occurred.
3. Invalid response command was returned from device.
4. There was a misalignment of packets due to connection/disconnection between PC and device.
5. There is a problem with the cabling connecting the devices, causing noise.

### Possible Solution:
While the driver attempts to recover from this error without intervention, there may be an issue with the cabling or the device itself.

## Unable to read block from device. Block deactivated. | Block start address = '<address>', Status code = <code>, Extended status code = <code>.

### Error Type:
Warning

### Possible Cause:
The address requested in the block does not exist in the PLC.

### Possible Solution:
Check the status and extended status codes returned by the PLC. An extended status code may not always be returned and the error information is contained within the status code. The codes are displayed in hexadecimal. Status code errors in the low nibble of the status code indicate errors found by the local node. The driver continues to retry reading these blocks of data periodically. Errors found by the local node occur when the KF module cannot see the destination PLC on the network. Status code errors in the high nibble of the status code indicate errors found by the PLC. These errors are generated when the block of data the driver is referncing is not available in the PLC. The driver does not ask for these blocks again after receiving this kind of error. This kind of error can be generated if the address does not exist in the PLC.

## Unable to write to address on device. Frame received contains errors. | Tag address = '<address>'.

### Error Type:
Warning

### Possible Cause:
1. An incorrect frame size was received.
2. A TNS mismatch occurred.
3. Invalid response command was returned from device.
4. There was a misalignment of packets due to connection/disconnection between PC and device.
5. There is a problem with the cabling connecting the devices, causing noise.

### Possible Solution:
While the driver attempts to recover from this error without intervention, there may be an issue with the cabling or the device itself.

## Unable to read block from device. | Block start address = '<address>', Status code = <code>, Extended status code = <code>.

### Error Type:
Warning

### Possible Cause:
The address requested in the block does not exist in the PLC.

### Possible Solution:
Check the status and extended status codes returned by the PLC. An extended status code may not always be returned and the error information is contained within the status code. The codes are displayed in hexadecimal. Status code errors in the low nibble of the status code indicate errors found by the local node. The driver continues to retry reading these blocks of data periodically. Errors found by the local node occur when the KF module cannot see the destination PLC on the network. Status code errors in the high nibble of the status code indicate errors found by the PLC. These errors are generated when the block of data the driver is referncing is not available in the PLC. The driver does not ask for these blocks again after receiving this kind of error. This kind of error can be generated if the address does not exist in the PLC.

## Unable to write to address on device. | Tag address = '<address>', Status code = <code>, Extended status code = <code>.

### Error Type:
Warning

### Possible Cause:
The address requested in the block does not exist in the PLC.

### Possible Solution:
Check the status and extended status codes returned by the PLC. An extended status code may not always be returned and the error information is contained within the status code. The codes are displayed in hexadecimal. Status code errors in the low nibble of the status code indicate errors found by the local node. The driver continues to retry writing these blocks periodically. Errors found by the local node occur when the KF module cannot see the destination PLC on the network. Status code errors in the high nibble of the status code indicate errors found by the PLC. These errors are generated when the block of data the driver is referncing is not available in the PLC. The driver does not ask for these blocks again after receiving this kind of error. This kind of error can be generated if the address does not exist in the PLC.

## Unable to read block from device. | Block start address = '<address>', Status code = <code>.

### Error Type:
Warning

### Possible Cause:
The address requested in the block does not exist in the PLC.

### Possible Solution:

1. Check the status codes returned by the PLC. The codes are displayed in hexadecimal.
2. Status code errors in the low nibble of the status code indicate errors found by the local node. The driver continues to retry reading these blocks of data periodically. Errors found by the local node occur when the KF module cannot see the destination PLC on the network.
3. Status code errors in the high nibble of the status code indicate errors found by the PLC. These errors are generated when the block of data the driver is referncing is not available in the PLC. The driver does not ask for these blocks again after receiving this kind of error. This kind of error can be generated if the address does not exist in the PLC.

## Unable to write to address on device. | Tag address = '<address>', Status code = <code>.

**Error Type:**
Warning

**Possible Cause:**
The address requested in the block does not exist in the PLC.

**Possible Solution:**

1. Check the status codes returned by the PLC. The codes are displayed in hexadecimal.
2. Status code errors in the low nibble of the status code indicate errors found by the local node. The driver continues to retry writing these blocks periodically. Errors found by the local node occur when the KF module cannot see the destination PLC on the network.
3. Status code errors in the high nibble of the status code indicate errors found by the PLC. These errors are generated when the block of data the driver is referncing is not available in the PLC. The driver does not ask for these blocks again after receiving this kind of error. This kind of error can be generated if the address does not exist in the PLC.

## The XML file contains an invalid network value for card type. Setting default network.

**Error Type:**
Warning

## The XML file contains invalid baud rate for card type.

**Error Type:**
Warning

## Channel is in use, unable to perform synchronization.

**Error Type:**
Warning

## Loading SST protocol.

**Error Type:**
Informational

## SST load completed.

**Error Type:**
Informational

## Loading AB DH+ protocol on device. | Card = '<card>'.

**Error Type:**
Informational

### AB DH+ protocol load on device completed. | Card = '<card>'.

**Error Type:**
Informational

### Loading AB DH+ protocol on PKTXChannel of device. | PKTXChannel = <channel>, Card = '<card>'.

**Error Type:**
Informational

### AB DH+ protocol load on PKTXChannel of device completed. | PKTXChannel = <channel>, Card = '<card>'.

**Error Type:**
Informational

### Windows NT platform detected.

**Error Type:**
Informational

### All matching channels have been updated with the new configured properties.

**Error Type:**
Informational

# Index

## 0



## A



## B



## C

## D



## E



## F

## G



## H



## I

## L



## M



## N



## O



## P

## R



## S



## T

## U



## W