

# Omron Host Link Driver

© 2023 PTC Inc. All Rights Reserved.

# Table of Contents

<b>Omron Host Link Driver</b> .....	<b>1</b>
<b>Table of Contents</b> .....	<b>2</b>
Omron Host Link Driver .....	3
Overview .....	3
<b>Setup</b> .....	<b>4</b>
Channel Properties — General .....	5
Tag Counts .....	6
Channel Properties — Serial Communications .....	7
Channel Properties — Write Optimizations .....	9
Channel Properties — Advanced .....	10
Device Properties — General .....	11
Device Properties — Scan Mode .....	12
Device Properties — Timing .....	14
Device Properties — Auto-Demotion .....	16
Device Properties — Inter-Character Delay .....	16
Device Properties — Redundancy .....	17
<b>Data Types Description</b> .....	<b>18</b>
<b>Address Descriptions</b> .....	<b>19</b>
C20H Addressing .....	19
C200H Addressing .....	23
CQM1 Addressing .....	27
Open Addressing .....	30
<b>Event Log Messages</b> .....	<b>35</b>
Mismatch between type requested and type received.   Tag address = '<address>', Type requested = <type>, Type received = <type>. .....	35
Bad address in block.   Tag address = '<address>', Block = <start> to <end>. .....	35
Error Mask Definitions .....	35
<b>Index</b> .....	<b>36</b>

## Omron Host Link Driver

---

Help version 1.039

### CONTENTS

#### Overview

What is the Omron Host Link Driver?

#### Setup

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

#### Data Types Description

What data types does this driver support?

#### Address Descriptions

How do I address a data location on an Omron Host Link device?

#### Event Log Messages

What messages does the Omron Host Link Driver produce?

### Overview

---

The Omron Host Link Driver provides a reliable way to connect Omron Host Link devices to client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with SYSMAC C-Series devices.

---

## Setup

---

### Supported Devices

C20H  
C200H  
CQM1  
Open

### Communication Protocol

Omron Host Link

### Supported Communication Parameters

Baud Rate: 9600  
Parity: Even or Odd  
Data Bits: 7 or 8  
Stop Bits: 1 or 2

### Channel and Device Limits

The maximum number of channels supported by this driver is 100. The maximum number of devices supported by this driver is 32 per channel.

### Ethernet Encapsulation

This driver supports Ethernet Encapsulation, which allows the driver to communicate with serial devices attached to an Ethernet network using a terminal server. It may be set through the channel properties. For more information, refer to the server help documentation.

### Flow Control

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

#### ● Notes:

1. When using the manufacturer's supplied communications cable, it is sometimes necessary to choose a flow control setting of **RTS** or **RTS Always** in the channel properties.
2. When running on platforms that do not enforce proper flow control, it may be necessary to set the flow control in the server's communications settings.

## Channel Properties — General

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

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

### Identification

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

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

**Description:** Specify user-defined information about this channel.

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

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

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

### Diagnostics

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

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

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

## Tag Counts

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

## Channel Properties — Serial Communications

Serial communication properties are available to serial drivers and vary depending on the driver, connection type, and options selected. Below is a superset of the possible properties.

Click to jump to one of the sections: [Connection Type](#), [Serial Port Settings](#) or [Ethernet Settings](#), and [Operational Behavior](#).

### Notes:

- With the server's online full-time operation, these properties can be changed at any time. Utilize proper user role and privilege management to prevent operators from changing properties or accessing server features.
- Users must define the specific communication parameters to be used. Depending on the driver, channels may or may not be able to share identical communication parameters. Only one shared serial connection can be configured for a Virtual Network (see [Channel Properties — Serial Communications](#)).

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

### Connection Type

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

1. **None:** Select None to indicate there is no physical connection, which displays the [Operation with no Communications](#) section.
2. **COM Port:** Select Com Port to display and configure the [Serial Port Settings](#) section.
3. **Modem:** Select Modem if phone lines are used for communications, which are configured in the [Modem Settings](#) section.
4. **Ethernet Encap.:** Select if Ethernet Encapsulation is used for communications, which displays the [Ethernet Settings](#) section.
5. **Shared:** Verify the connection is correctly identified as sharing the current configuration with another channel. This is a read-only property.

### Serial Port Settings

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

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

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

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


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

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

- **None:** This option does not toggle or assert control lines.
- **DTR:** This option asserts the DTR line when the communications port is opened and remains on.
- **RTS:** This option specifies that the RTS line is high if bytes are available for transmission. After all buffered bytes have been sent, the RTS line is low. This is normally used with RS232/RS485 converter hardware.
- **RTS, DTR:** This option is a combination of DTR and RTS.
- **RTS Always:** This option asserts the RTS line when the communication port is opened and remains on.
- **RTS Manual:** This option asserts the RTS line based on the timing properties entered for RTS Line Control. It is only available when the driver supports manual RTS line control (or when the properties are shared and at least one of the channels belongs to a driver that provides this support).

RTS Manual adds an **RTS Line Control** property with options as follows:


- **Raise:** Specify the amount of time that the RTS line is raised prior to data transmission. The valid range is 0 to 9999 milliseconds. The default is 10 milliseconds.
- **Drop:** Specify the amount of time that the RTS line remains high after data transmission. The valid range is 0 to 9999 milliseconds. The default is 10 milliseconds.
- **Poll Delay:** Specify the amount of time that polling for communications is delayed. The valid range is 0 to 9999. The default is 10 milliseconds.

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

## Operational Behavior

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

## Ethernet Settings

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



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

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

## Modem Settings

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

## Operation with no Communications

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

## Channel Properties — Write Optimizations

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

Property Groups	[-] <b>Write Optimizations</b>	
General	Optimization Method	Write Only Latest Value for All Tags
<b>Write Optimizations</b>	Duty Cycle	10

## Write Optimizations

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

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

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

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

### Channel Properties — Advanced

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

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

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

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

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

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

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

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

## Device Properties — General

Property Groups	<input type="checkbox"/> <b>Identification</b>	
<b>General</b>	Name	Omron Host Link
Scan Mode	Description	
Timing	Channel Assignment	Omron Host Link
Auto-Demotion	Driver	Omron Host Link
Inter-Character Delay	Model	C20H
Redundancy	ID Format	Decimal
	ID	0
	<input type="checkbox"/> <b>Operating Mode</b>	
	Data Collection	Enable
	Simulated	No

### Identification

**Name:** User-defined identity of this device.

**Description:** User-defined information about this device.

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

**Driver:** Selected protocol driver for this device.

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

● For a list of models that support the FINS Communications Service, refer to the manufacturer's website.

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

**ID:** The ID specifies the unique unit number of the device. The valid range is 0 to 31.

### 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.

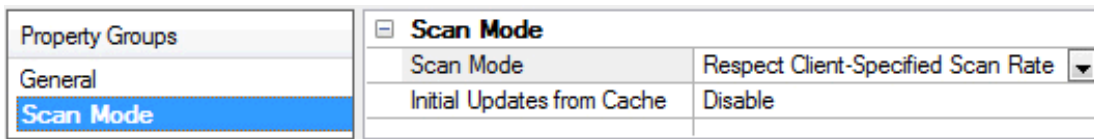
**Notes:**

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

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

**Initial Updates from Cache:** When enabled, this option allows the server to provide the first updates for newly activated tag references from stored (cached) data. Cache updates can only be provided when the new item reference shares the same address, scan rate, data type, client access, and scaling properties. A

device read is used for the initial update for the first client reference only. The default is disabled; any time a client activates a tag reference the server attempts to read the initial value from the device.

## Device Properties — Timing

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

Property Groups	☐ <b>Communication Timeouts</b>	
General	Connect Timeout (s)	3
Scan Mode	Request Timeout (ms)	1000
<b>Timing</b>	Attempts Before Timeout	3

### Communications Timeouts

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

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

**Request Timeout:** Specify an interval used by all drivers to determine how long the driver waits for a response from the target device to complete. The valid range is 50 to 9999 milliseconds (167 minutes). The default is usually 1000 milliseconds, but can vary depending on the driver. The default timeout for most serial drivers is based on a baud rate of 9600 baud or better. When using a driver at lower baud rates, increase the timeout to compensate for the increased time required to acquire data.

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

### Timing

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

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

Property Groups	[-] <b>Timing</b>	
General	Inter-Request Delay (ms)	0
Scan Mode		
<b>Timing</b>		

## 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.

Property Groups	[-] <b>Auto-Demotion</b>	
General	Demote on Failure	Enable
Scan Mode	Timeouts to Demote	3
Timing	Demotion Period (ms)	10000
<b>Auto-Demotion</b>	Discard Requests when Demoted	Disable

**Demote on Failure:** When enabled, the device is automatically taken off-scan until it is responding again.

**Tip:** Determine when a device is off-scan by monitoring its demoted state using the `_AutoDemoted` system tag.

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

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

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

## Device Properties — Inter-Character Delay

The inter-character delay is intended for modem communications, but some specific devices may not need a delay when communicating via modem. The delay introduced by this value reduces communication speed.

Property Groups	[-] <b>Inter-Character Delay</b>	
Scan Mode	Delay (ms)	0
Timing		
Auto-Demotion		
<b>Inter-Character Delay</b>		

**Delay:** Specify the controlled delay between each character the driver sends to an Omron device. The valid range is 0 to 200 milliseconds. The default setting is 0 milliseconds.

**Tip:** It is generally recommended that this value be left at the default setting; however, it can be adjusted for successful communications when using the driver over a modem. For a modem connection of 1200 baud, a typical setting would be 30 milliseconds.



## Device Properties — Redundancy

Property Groups	☐ <b>Redundancy</b>	
General	Secondary Path	<b>Channel.Device 1</b> ...
Scan Mode	Operating Mode	Switch On Failure
Timing	Monitor Item	
Auto-Demotion	Monitor Interval (s)	300
Tag Generation	Return to Primary ASAP	Yes
Tag Import Settings		
<b>Redundancy</b>		

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

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

## Data Types Description

Data Type	Description
Boolean	Single bit
Word	Unsigned 16-bit value bit 0 is the low bit bit 15 is the high bit
Short	Signed 16-bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
BCD	Two byte packed BCD Value range is 0-9999. Behavior is undefined for values beyond this range.
LBCD	Four byte packed BCD Value range is 0-99999999. Behavior is undefined for values beyond this range.
Long	Signed 32-bit value
DWord	Unsigned 32-bit value
Float	32-bit floating point value The driver interprets two consecutive registers as a floating point value by making the second register the high word and the first register the low word.
Float Example	If register DM100 is specified as a float, bit 0 of register DM100 would be bit 0 of the 32-bit float, and bit 15 of register DM101 would be bit 31 of the 32-bit float.
String	Null terminated ASCII string. Support includes string lengths up to 58 characters, and selection of HiLo byte order, LoHi byte order, Only High byte, and Only Low byte.

## Address Descriptions

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

[C20H Addressing](#)

[C200H Addressing](#)

[CQM1 Addressing](#)

[Open Addressing](#)

## C20H Addressing

The default data types are shown in **bold**.

• For more information, refer to [String Support](#) and [Array Support](#).

Device Type	Range	Data Type	Access
Auxiliary Relay	AR00-AR27	<b>Word</b> , Short, BCD, Long,	Read/Write
	AR00-AR26	DWord, Float, LBCD	
	ARxx.00-ARxx.15	Boolean	
Auxiliary Relay as String with HiLo Byte Order	AR00.56H-AR27.02H .I is string length, range 2 to 56 chars	<b>String</b>	Read/Write
Auxiliary Relay as String with LoHi Byte Order	AR00.56L-AR27.02L .I is string length, range 2 to 56 chars	<b>String</b>	Read/Write
Auxiliary Relay as String using Only the High Order byte of each word	AR00.28D-AR27.01D .I is string length, range 1 to 28 chars	<b>String</b>	Read/Write
Auxiliary Relay as String using Only the Low Order byte of each word	AR00.28E-AR27.01E .I is string length, range 1 to 28 chars	<b>String</b>	Read/Write
Data Memory	DM0000-DM0999	<b>Word</b> , Short, BCD, Long,	Read/Write
	DM0000-DM0998	DWord, Float, LBCD	
	DMxxxx.00-DMxxxx.15	Boolean	
	DM1000-DM1999	<b>Word</b> , Short, BCD, Long,	Read Only
	DM1000-DM1998	DWord, Float, LBCD	
DMxxxx.00-DMxxxx.15			

Device Type	Range	Data Type	Access
		Boolean	
Data Memory as String with HiLo Byte Order	DM0000.58H- DM0999.02H  .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
	DM1000.58H- DM1999.02H  .I is string length, range 2 to 58 chars	<b>String</b>	Read Only
Data Memory as String with LoHi Byte Order	DM0000.58L- DM0999.02L  .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
	DM1000.58L- DM1999.02L  .I is string length, range 2 to 58 chars	<b>String</b>	Read Only
Data Memory as String using Only the High Order byte of each word	DM0000.29D- DM0999.01D  .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
	DM1000.29D- DM1999.01D  .I is string length, range 1 to 29 chars	<b>String</b>	Read Only
Data Memory as String using Only the Low Order byte of each word	DM0000.29E- DM0999.01E  .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
	DM1000.29E- DM1999.01E  .I is string length, range 1 to 29 chars	<b>String</b>	Read Only
Holding Relay	HR00-HR99	<b>Word</b> , Short, BCD, Long, DWord, Float,	Read/Write
	HR00-HR98		

Device Type	Range	Data Type	Access
	HRxx.00-HRxx.15	LBCD Boolean	
Holding Relay as String with HiLo Byte Order	HR00.58H-HR99.02H .l is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Holding Relay as String with LoHi Byte Order	HR00.58L-HR99.02L .l is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Holding Relay as String using Only the High Order byte of each word	HR00.29D-HR99.01D .l is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Holding Relay as String using Only the Low Order byte of each word	HR00.29E-HR99.01E .l is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Internal Relay	IR000-IR255 IR000-IR254 IRxxx.00-IRxxx.15	<b>Word</b> , Short, BCD, Long, DWord, Float, LBCD Boolean	Read/Write
Internal Relay as String with HiLo Byte Order	IR000.58H-IR255.02H .l is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Internal Relay as String with LoHi Byte Order	IR000.58L-IR255.02L .l is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Internal Relay as String using Only the High Order byte of each word	IR000.29D-IR255.01D .l is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Internal Relay as String using Only the Low Order byte of each word	IR000.29E-IR255.01E .l is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Link Relays	LR00-LR63 LR00-LR62 LRxx.00-LRxx.15	<b>Word</b> , Short, BCD, Long, DWord, Float, LBCD	Read/Write

Device Type	Range	Data Type	Access
		Boolean	
Link Relays as String with HiLo Byte Order	LR00.58H-LR63.02H .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Link Relays as String with LoHi Byte Order	LR00.58L-LR63.02L .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Link Relays as String using Only the High Order byte of each word	LR00.29D-LR63.01D .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Link Relays as String using Only the Low Order byte of each word	LR00.29E-LR63.01E .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Timer/Counter	RC000-RC511 RCxxx.00-RCxxx.15	Word, Short, <b>BCD</b> Boolean	Read/Write
Timer/Counter status	TC000-TC511	<b>Boolean</b>	Read/Write

### String Support

The C20H model supports reading and writing auxiliary relay (AR), data memory (DM), holding relay (HR), internal relay (IR) and link relays (LR) as an ASCII string. When using any of these device types for string data, each register can contain either two bytes (two characters) of ASCII data or one. The order of the ASCII data within a given register, or the byte to use within a given register can be selected when the string is defined.

When using two bytes of ASCII data per register the length of the string can be from 2 to 58 characters (or 2 to 56 for AR) and is entered in place of a bit number. The length must be entered as an even number. The range of registers spanned by the string cannot exceed the range of the device type. The byte order is specified by appending either a "H" or "L" to the address.

When using one byte of ASCII data per register the length of the string can be from 1 to 29 characters (or 1 to 28 for AR) and is entered in place of a bit number. The range of registers spanned by the string cannot exceed the range of the device type. The byte to use within a register is specified by appending either a "D" or "E" to the address.

### Examples

- To address a string starting at DM100 with a length of 50 bytes and HiLo byte order, enter:  
DM100.50H
- To address a string starting at DM110 with a length of 8 bytes and LoHi byte order, enter: DM110.08L
- To address a string starting at DM200 with a length of 15 bytes and Only the High Order byte, enter:  
DM200.15D

- To address a string starting at DM220 with a length of 7 bytes and Only the Low Order byte, enter:  
DM220.07E

## Array Support

Arrays are supported for all data types except Boolean and String. There are two methods of addressing an array. Examples are given using data memory locations.

*DMxxxx [rows] [cols]*

*DMxxxx [cols]\**

\* This method assumes "rows" is equal to one.

Rows multiplied by cols multiplied by data size in bytes cannot exceed 116 bytes. This limit is imposed by the protocol. Since this driver uses an ASCII protocol, there are 4 bytes for each word, short and BCD, and 8 bytes for each DWord, long, LBCD and float. For example, a 4 X 7 array of words results in an array size of 28 words times 4 bytes for each word = 112 bytes, which would fit within the maximum request size of 116 bytes.

Use caution when modifying 32-bit values (DWord, Long, LBCD, and Float). Each address that allows these data types starts at a word offset within the device. Therefore, DWords DM0 and DM1 overlap at word DM1. Writing to DM0 also modifies the value held in DM1. It is recommended that users utilize these data types so that overlapping does not occur. As an example, when using DWords, users may want to use DM0, DM2, DM4 and so on to prevent overlapping Words.

## C200H Addressing

The default data types are shown in **bold**.

For more information, refer to [String Support](#) and [Array Support](#).

Device Type	Range	Data Type	Access
Auxiliary Relay	AR000-AR999 AR000-AR998 ARxxx.00-ARxxx.15	<b>Word</b> , Short, BCD, Long, DWord, Float, LBCD  Boolean	Read/Write
Auxiliary Relay as String with HiLo Byte Order	AR000.58H-AR999.02H .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Auxiliary Relay as String with LoHi Byte Order	AR000.58L-AR999.02L .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Auxiliary Relay as String using Only the High Order byte of each word	AR000.29D-AR999.01D .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Auxiliary Relay as String using Only the Low	AR000.29E-AR999.01E	<b>String</b>	Read/Write

Device Type	Range	Data Type	Access
Order byte of each word	.I is string length, range 1 to 29 chars		
Data Memory	DM0000-DM9999 DM0000-DM9998 DMxxxx.00-DMxxxx.15	<b>Word</b> , Short, BCD, Long, DWord, Float, LBCD  Boolean	Read/Write
Data Memory as String with HiLo byte order	DM0000.58H- DM0999.02H  .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Data Memory as String with LoHi Byte Order	DM0000.58L- DM0999.02L  .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Data Memory as String using Only the High Order byte of each word	DM0000.29D- DM0999.01D  .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Data Memory as String using Only the Low Order byte of each word	DM0000.29E- DM0999.01E  .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Holding Relay	HR000-HR999 HR000-HR998 HRxxx.00-HRxxx.15	<b>Word</b> , Short, BCD, Long, DWord, Float, LBCD  Boolean	Read/Write
Holding Relay as String with HiLo byte order	HR000.58H- HR999.02H  .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Holding Relay as String with LoHi Byte Order	HR000.58L-HR999.02L  .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Holding Relay as String using Only the High Order byte of each word	HR000.29D-HR999.01D  .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write



Device Type	Range	Data Type	Access
Holding Relay as String using Only the Low Order byte of each word	HR000.29E-HR999.01E .l is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Internal Relay	IR000-IR999 IR000-IR998 IRxxx.00-IRxxx.15	<b>Word</b> , Short, BCD, Long, DWord, Float, LBCD  Boolean	Read/Write
Internal Relay as String with HiLo byte order	IR000.58H-IR999.02H .l is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Internal Relay as String with LoHi Byte Order	IR000.58L-IR999.02L .l is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Internal Relay as String using Only the High Order byte of each word	IR000.29D-IR999.01D .l is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Internal Relay as String using Only the Low Order byte of each word	IR000.29E-IR999.01E .l is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Link Relays	LR000-LR999 LR000-LR998 LRxxx.00-LRxxx.15	<b>Word</b> , Short, BCD, Long, DWord, Float, LBCD  Boolean	Read/Write
Link Relays as String with HiLo Byte Order	LR000.58H-LR999.02H .l is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Link Relays as String with LoHi byte order	LR000.58L-LR999.02L .l is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Link Relays as String using Only the High Order byte of each word	LR000.29D-LR999.01D .l is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Link Relays as String using Only the Low Order byte of each word	LR000.29E-LR999.01E	<b>String</b>	Read/Write

Device Type	Range	Data Type	Access
	.l is string length, range 1 to 29 chars		
Timer/Counter	RC000-RC999 RCxxx.00-RCxxx.15	Word, Short, <b>BCD</b>  Boolean	Read/Write
Timer/Counter status	TC000-TC999	<b>Boolean</b>	Read/Write

### String Support

The C200H model supports reading and writing auxiliary relay (AR), data memory (DM), holding relay (HR), internal relay (IR) and link relays (LR) as an ASCII string. When using any of these device types for string data, each register can contain either two bytes (two characters) of ASCII data or one. The order of the ASCII data within a given register, or the byte to use within a given register can be selected when the string is defined.

When using two bytes of ASCII data per register the length of the string can be from 2 to 58 characters and is entered in place of a bit number. The length must be entered as an even number. The range of registers spanned by the string cannot exceed the range of the device type. The byte order is specified by appending either a "H" or "L" to the address.

When using one byte of ASCII data per register the length of the string can be from 1 to 29 characters and is entered in place of a bit number. The range of registers spanned by the string cannot exceed the range of the device type. The byte to use within a register is specified by appending either a "D" or "E" to the address.

### Examples

- To address a string starting at DM100 with a length of 50 bytes and HiLo byte order, enter:  
DM100.50H
- To address a string starting at DM110 with a length of 8 bytes and LoHi byte order, enter: DM110.08L
- To address a string starting at DM200 with a length of 15 bytes and Only the High Order byte, enter:  
DM200.15D
- To address a string starting at DM220 with a length of 7 bytes and Only the Low Order byte, enter:  
DM220.07E

### Array Support

Arrays are supported for all data types except Boolean and String. There are two methods of addressing an array. Examples are given using data memory locations.

*DMxxxx [rows] [cols]*  
*DMxxxx [cols]\**

\* This method assumes "rows" is equal to one.

Rows multiplied by cols multiplied by data size in bytes cannot exceed 116 bytes. This limit is imposed by the protocol. Since this driver uses an ASCII protocol, there are 4 bytes for each word, short and BCD, and 8 bytes for each DWord, long, LBCD and float. For example, a 4 X 7 array of words results in an array size of 28 words times 4 bytes for each word = 112 bytes, which would fit within the maximum request size of 116 bytes.

Use caution when modifying 32-bit values (DWord, Long, LBCD, and Float). Each address that allows these data types starts at a word offset within the device. Therefore, DWords DM0 and DM1 overlap at word DM1. Writing to DM0 also modifies the value held in DM1. It is recommended that users utilize these data types so that overlapping does not occur. When using DWords, for example, users may want to use DM0, DM2, DM4 and so on to prevent overlapping Words.

## CQM1 Addressing

The default data types are shown in **bold**.

For more information, refer to [String Support](#) and [Array Support](#).

Device Type	Range	Data Type	Access
Auxiliary Relay	AR00-AR27	<b>Word</b> , Short, BCD, Long	Read/Write
	AR00-AR26	DWord, Float, LBCD	
	ARxx.00-ARxx.15	Boolean	
Auxiliary Relay as String with HiLo Byte Order	AR00.56H-AR27.02H .l is string length, range 2 to 56 chars	<b>String</b>	Read/Write
Auxiliary Relay as String with LoHi Byte Order	AR00.56L-AR27.02L .l is string length, range 2 to 56 chars	<b>String</b>	Read/Write
Auxiliary Relay as String using Only the High Order byte of each word	AR00.28D-AR27.01D .l is string length, range 1 to 28 chars	<b>String</b>	Read/Write
Auxiliary Relay as String using Only the Low Order byte of each word	AR00.28E-AR27.01E .l is string length, range 1 to 28 chars	<b>String</b>	Read/Write
Data Memory	DM0000-DM6655	<b>Word</b> , Short, BCD, Long	Read/Write
	DM0000-DM6654	DWord, Float, LBCD	
	DMxxxx.00-DMxxxx.15	Boolean	
Data Memory as String with HiLo Byte Order	DM0000.58H-DM6655.02H .l is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Data Memory as String with LoHi Byte Order	DM0000.58L-DM6655.02L	<b>String</b>	Read/Write

Device Type	Range	Data Type	Access
	.J is string length, range 2 to 58 chars		
Data Memory as String using Only the High Order byte of each word	DM0000.29D-DM6655.01D .J is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Data Memory as String using Only the Low Order byte of each word	DM0000.29E-DM6655.01E .J is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Holding Relay	HR00-HR99 HR00-HR98 HRxx.00-HRxx.15	<b>Word</b> , Short, BCD, Long  DWord, Float, LBCD  Boolean	Read/Write
Holding Relay as String with HiLo Byte Order	HR00.58H-HR99.02H .J is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Holding Relay as String with LoHi Byte Order	HR00.58L-HR99.02L .J is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Holding Relay as String using Only the High Order byte of each word	HR00.29D-HR99.01D .J is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Holding Relay as String using Only the Low Order byte of each word	HR00.29E-HR99.01E .J is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Internal Relay	IR000-IR255 IR000-IR254 IRxxx.00-IRxxx.15	<b>Word</b> , Short, BCD, Long  DWord, Float, LBCD  Boolean	Read/Write
Internal Relay as String with HiLo Byte Order	IR000.58H-IR255.02H .J is string length, range 2 to 58 chars	<b>String</b>	Read/Write

Device Type	Range	Data Type	Access
Internal Relay as String with LoHi Byte Order	IR000.58L-IR255.02L .l is string length, range 2 to 58 chars	String	Read/Write
Internal Relay as String using Only the High Order byte of each word	IR000.29D-IR255.01D .l is string length, range 1 to 29 chars	String	Read/Write
Internal Relay as String using Only the Low Order byte of each word	IR000.29E-IR255.01E .l is string length, range 1 to 29 chars	String	Read/Write
Link Relays	LR00-LR63 LR00-LR62 LRxx.00-LRxx.15	Word, Short, BCD, Long DWord, Float, LBCD Boolean	Read/Write
Link Relays as String with HiLo Byte Order	LR00.58H-LR63.02H .l is string length, range 2 to 58 chars	String	Read/Write
Link Relays as String with LoHi Byte Order	LR00.58L-LR63.02L .l is string length, range 2 to 58 chars	String	Read/Write
Link Relays as String using Only the High Order byte of each word	LR00.29D-LR63.01D .l is string length, range 1 to 29 chars	String	Read/Write
Link Relays as String using Only the Low Order byte of each word	LR00.29E-LR63.01E .l is string length, range 1 to 29 chars	String	Read/Write
Timer/Counter	RC000-RC511 RCxxx.00-RCxxx.15	Word, Short, BCD Boolean	Read/Write
Timer/Counter status	TC000-TC511	Boolean	Read/Write

### String Support

The CQM1 model supports reading and writing auxiliary relay (AR), data memory (DM), holding relay (HR), internal relay (IR) and link relays (LR) as an ASCII string. When using any of these device types for string data, each register can contain either two bytes (two characters) of ASCII data or one. The order of the ASCII data within a given register, or the byte to use within a given register can be selected when the string is defined.

When using two bytes of ASCII data per register the length of the string can be from 2 to 58 characters (or 2 to 56 for AR) and is entered in place of a bit number. The length must be entered as an even number. The range of registers spanned by the string cannot exceed the range of the device type. The byte order is specified by appending either a "H" or "L" to the address.

When using one byte of ASCII data per register the length of the string can be from 1 to 29 characters (or 1 to 28 for AR) and is entered in place of a bit number. The range of registers spanned by the string cannot exceed the range of the device type. The byte to use within a register is specified by appending either a "D" or "E" to the address.

### Examples

- To address a string starting at DM100 with a length of 50 bytes and HiLo byte order, enter:  
DM100.50H
- To address a string starting at DM110 with a length of 8 bytes and LoHi byte order, enter: DM110.08L
- To address a string starting at DM200 with a length of 15 bytes and Only the High Order byte, enter:  
DM200.15D
- To address a string starting at DM220 with a length of 7 bytes and Only the Low Order byte, enter:  
DM220.07E

### Array Support

Arrays are supported for all data types except Boolean and String. There are two methods of addressing an array. Examples are given using data memory locations.

*DMxxxx [rows] [cols]*

*DMxxxx [cols]\**

\* This method assumes "rows" is equal to one.

Rows multiplied by cols multiplied by data size in bytes cannot exceed 116 bytes. This limit is imposed by the protocol. Since this driver uses an ASCII protocol, there are 4 bytes for each word, short and BCD, and 8 bytes for each DWord, long, LBCD and float. For example, a 4 X 7 array of words results in an array size of 28 words times 4 bytes for each word = 112 bytes, which would fit within the maximum request size of 116 bytes.

⚠ Use caution when modifying 32-bit values (DWord, Long, LBCD, and Float). Each address that allows these data types starts at a word offset within the device. Therefore, DWords DM0 and DM1 overlap at word DM1. Writing to DM0 also modifies the value held in DM1. It is recommended that users utilize these data types so that overlapping does not occur. When using DWords, for example, users may want to use DM0, DM2, DM4 and so on to prevent overlapping Words.

### Open Addressing

The following memory map is open for all memory types to support newer devices. Consult the manufacturer's documentation for device specific address ranges. The default data types are shown in **bold**.

• For more information, refer to [String Support](#) and [Array Support](#).

Device Type	Range	Data Type	Access
Auxiliary Relay	AR0000-AR9999 AR0000-AR9998	<b>Word</b> , Short, BCD, Long, DWord, Float, LBCD	Read/Write

Device Type	Range	Data Type	Access
	ARxxxx.00-ARxxxx.15	Boolean	
Auxiliary Relay as String with HiLo Byte Order	AR0000.58H-AR9999.02H .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Auxiliary Relay as String with LoHi Byte Order	AR0000.58L-AR9999.02L .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Auxiliary Relay as String using Only the High Order byte of each word	AR0000.29D-AR9999.01D .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Auxiliary Relay as String using Only the Low Order byte of each word	AR0000.29E-AR9999.01E .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Data Memory	DM0000-DM9999 DM0000-DM9998 DMxxxx.00-DMxxxx.15	<b>Word</b> , Short, BCD, Long, DWord, Float, LBCD  Boolean	Read/Write
Data Memory as String with HiLo Byte Order	DM0000.58H-DM9999.02H .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Data Memory as String with LoHi Byte Order	DM0000.58L-DM9999.02L .I is string length, range 2 to 58 chars	<b>String</b>	Read/Write
Data Memory as String using Only the High Order byte of each word	DM0000.29D-DM9999.01D .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Data Memory as String using Only the Low Order byte of each word	DM0000.29E-DM9999.01E .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Holding Relay	HR0000-HR9999 HR0000-HR9998 HRxxxx.00-HRxxxx.15	<b>Word</b> , Short, BCD, Long, DWord, Float, LBCD  Boolean	Read/Write
Holding Relay as String with HiLo Byte Order	HR0000.58H-HR9999.02H	<b>String</b>	Read/Write

Device Type	Range	Data Type	Access
	.l is string length, range 2 to 58 chars		
Holding Relay as String with LoHi Byte Order	HR0000.58L-HR9999.02L .l is string length, range 2 to 58 chars	String	Read/Write
Holding Relay as String using Only the High Order byte of each word	HR0000.29D-HR9999.01D .l is string length, range 1 to 29 chars	String	Read/Write
Holding Relay as String using Only the Low Order byte of each word	HR0000.29E-HR9999.01E .l is string length, range 1 to 29 chars	String	Read/Write
Internal Relay	IR0000-IR9999 IR0000-IR9998 IRxxxx.00-IRxxxx.15	Word, Short, BCD, Long, DWord, Float, LBCD  Boolean	Read/Write
Internal Relay as String with HiLo Byte Order	IR0000.58H-IR9999.02H .l is string length, range 2 to 58 chars	String	Read/Write
Internal Relay as String with LoHi Byte Order	IR0000.58L-IR9999.02L .l is string length, range 2 to 58 chars	String	Read/Write
Internal Relay as String using Only the High Order byte of each word	IR0000.29D-IR9999.01D .l is string length, range 1 to 29 chars	String	Read/Write
Internal Relay as String using Only the Low Order byte of each word	IR0000.29E-IR9999.01E .l is string length, range 1 to 29 chars	String	Read/Write
Link Relays	LR0000-LR9999 LR0000-LR9998 LRxxxx.00-LRxxxx.15	Word, Short, BCD, Long, DWord, Float, LBCD  Boolean	Read/Write
Link Relays as String with HiLo Byte Order	LR0000.58H-LR9999.02H .l is string length, range 2 to 58 chars	String	Read/Write
Link Relays as String with LoHi Byte Order	LR0000.58L-LR9999.02L	String	Read/Write



Device Type	Range	Data Type	Access
	.I is string length, range 2 to 58 chars		
Link Relays as String using Only the High Order byte of each word	LR0000.29D-LR9999.01D .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Link Relays as String using Only the Low Order byte of each word	LR0000.29E-LR9999.01E .I is string length, range 1 to 29 chars	<b>String</b>	Read/Write
Timer/Counter	RC0000-RC9999 RCxxxx.00-RCxxxx.15	Word, Short, <b>BCD</b> Boolean	Read/Write
Timer/Counter Status*	TC0000-TC9999	<b>Boolean</b>	Read/Write

\* **Note:** Some models do not support writes to Timer/Counter status (TCxxxx).

### String Support

The Open model supports reading and writing auxiliary relay (AR), data memory (DM), holding relay (HR), internal relay (IR) and link relays (LR) as an ASCII string. When using any of these device types for string data, each register can contain either two bytes (two characters) of ASCII data or one. The order of the ASCII data within a given register, or the byte to use within a given register can be selected when the string is defined.

When using two bytes of ASCII data per register the length of the string can be from 2 to 58 characters and is entered in place of a bit number. The length must be entered as an even number. The range of registers spanned by the string cannot exceed the range of the device type. The byte order is specified by appending either a "H" or "L" to the address.

When using one byte of ASCII data per register the length of the string can be from 1 to 29 characters and is entered in place of a bit number. The range of registers spanned by the string cannot exceed the range of the device type. The byte to use within a register is specified by appending either a "D" or "E" to the address.

### Examples

- To address a string starting at DM100 with a length of 50 bytes and HiLo byte order, enter:  
DM100.50H
- To address a string starting at DM110 with a length of 8 bytes and LoHi byte order, enter: DM110.08L
- To address a string starting at DM200 with a length of 15 bytes and Only the High Order bytes enter:  
DM200.15D
- To address a string starting at DM220 with a length of 7 bytes and Only the Low Order byte, enter:  
DM220.07E

### Array Support

Arrays are supported for all data types except Boolean and String. There are two methods of addressing an array. Examples are given using data memory locations.

*DMxxxx [rows] [cols]*  
*DMxxxx [cols]\**

\* This method assumes "rows" is equal to one.

Rows multiplied by cols multiplied by data size in bytes cannot exceed 116 bytes. This limit is imposed by the protocol. Since this driver uses an ASCII protocol, there are 4 bytes for each word, short and BCD, and 8 bytes for each DWord, long, LBCD and float. For example, a 4 X 7 array of words results in an array size of 28 words times 4 bytes for each word = 112 bytes, which would fit within the maximum request size of 116 bytes.

⚠ Use caution when modifying 32-bit values (DWord, Long, LBCD, and Float). Each address that allows these data types starts at a word offset within the device. Therefore, DWords DM0 and DM1 overlap at word DM1. Writing to DM0 also modifies the value held in DM1. It is recommended that users utilize these data types so that overlapping does not occur. When using DWords, for example, users may want to use DM0, DM2, DM4 and so on to prevent overlapping Words.

## Event Log Messages

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

**Mismatch between type requested and type received. | Tag address = '<address>', Type requested = <type>, Type received = <type>.**

---

**Error Type:**

Error

**Bad address in block. | Tag address = '<address>', Block = <start> to <end>.**

---

**Error Type:**

Warning

**Possible Cause:**

An attempt was made to reference a non-existent location in the specified device.

**Possible Solution:**

Verify that the addresses assigned to the tags are in the specified range on the device. Eliminate or correct tags that reference invalid locations.

### Error Mask Definitions

---

**B** = Hardware break detected

**F** = Framing error

**E** = I/O error

**O** = Character buffer overrun

**R** = RX buffer overrun

**P** = Received byte parity error

**T** = TX buffer full

# Index

## A

Address Descriptions 19  
Attempts Before Timeout 14  
Auto-Demotion 16  
Auto-Dial 9

## B

Bad address in block. | Tag address = '<address>', Block = <start> to <end>. 35  
Baud Rate 4, 8  
BCD 18  
Boolean 18

## C

C200H Addressing 23  
C20H 19  
Channel Assignment 11  
Channel Properties — Advanced 10  
Channel Properties — General 5  
Channel Properties — Serial Communications 7  
Channel Properties — Write Optimizations 9  
Close Idle Connection 8-9  
COM ID 8  
COM Port 7  
Communication Protocol 4  
Communications Timeouts 14  
Connect Timeout 9, 14  
Connection Type 7  
CQM1 Addressing 27

## D

Data Bits 4, 8  
Data Collection 11

Data Types Description 18  
Delay 16  
Demote on Failure 16  
Demotion Period 16  
Device Properties — Auto-Demotion 16  
Device Properties — Redundancy 17  
Device Properties — Timing 14  
Diagnostics 5  
Discard Requests when Demoted 16  
Do Not Scan, Demand Poll Only 12  
Driver 11  
Drop 8  
DTR 8  
Duty Cycle 10  
DWord 18

## **E**

ERP 3  
Error Mask Definitions 35  
Ethernet Encap. 7  
Ethernet Encapsulation 4  
Ethernet Settings 8  
Event Log Messages 35

## **F**

Float 18  
Flow Control 4, 8  
Framing 35

## **H**

Hardware break 35  
Historian 3  
HMI 3

**I**

I/O error 35  
ID 11  
Identification 5, 11  
Idle Time to Close 8-9  
Initial Updates from Cache 13  
Inter-Character Delay 16  
Inter-Device Delay 11

**L**

LBCD 18  
Long 18

**M**

Mask 35  
MES 3  
Mismatch between type requested and type received. | Tag address = '<address>', Type requested = <type>, Type received = <type>. 35  
Model 11  
Modem 7, 9  
Modem Settings 9

**N**

Network 4  
Network Adapter 9  
Non-Normalized Float Handling 10  
None 7

**O**

Open Addressing 30  
Operation with no Communications 9  
Operational Behavior 8  
Optimization Method 9

Overrun 35

Overview 3

## **P**

Parity 4, 8, 35

Physical Medium 7

Poll Delay 8

## **R**

Raise 8

Read Processing 9

Redundancy 17

Replace with Zero 10

Report Communication Errors 8-9

Request Timeout 14

Respect Tag-Specified Scan Rate 12

RS-485 8

RTS 8

RXbuffer overrun 35

## **S**

SCADA 3

Scan Mode 12

Serial Communications 7

Serial Port Settings 7

Setup 4

Shared 7

Short 18

Simulated 12

Stop Bits 4, 8

String 18

SYSMAC C-Series 3

## **T**

Tag Counts 6

Timeouts to Demote 16

Timing 14

TXbuffer full 35

## **U**

Unmodified 10

## **W**

Word 18

Write All Values for All Tags 10

Write Only Latest Value for All Tags 10

Write Only Latest Value for Non-Boolean Tags 10