Honeywell UDC Serial Driver

Table of Contents

Honeywell UDC Serial Driver

Help version 1.022

CONTENTS

Overview

The Honeywell UDC Serial Driver provides a reliable way to connect Honeywell UDC Serial devices to OPC client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with Honeywell UDC Serial devices that support the Modbus RTU protocol. The driver will support the UDC 3000 and UDC 3300. To support the Honeywell UDC Serial 3300, select Modbus 3K communications when configuring the UDC 3300.

This driver can also control the operation of the RTS line for use with radio modems that require specific RTS timing.

Setup

Supported Devices

Honeywell UDC Serial 3000, Honeywell UDC Serial 3300 (MODB3K mode).

Communication Protocol

Modbus RTU Protocol with Honeywell UDC Serial extensions.

Supported Communication Parameters*

The default settings are shown in **bold** where appropriate.

Baud Rate: 1200, 2400, 9600, 19200
Parity: Odd, Even, **None**
Data Bits: **8**
Stop Bits: **1**,2

*Not all devices support the listed configurations.

Ethernet Encapsulation

This driver supports Ethernet Encapsulation, which allows the driver to communicate with serial devices attached to an Ethernet network using a terminal server or device server. Ethernet Encapsulation mode may be invoked through the COM ID dialog in Channel Properties. For more information, refer to the main OPC Server help file.

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 247 per channel.

Device ID (PLC Network Address)

Honeywell UDC Serial devices are assigned Device IDs in the range 1 to 99.

Flow Control

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

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

The Honeywell UDC Serial Driver supports the **RTS Manual** flow control option. This selection is used to configure the driver for operation with radio modems that require special RTS timing characteristics. *For more information on RTS Manual flow control, refer to **Channel Properties — Serial Communication**.*

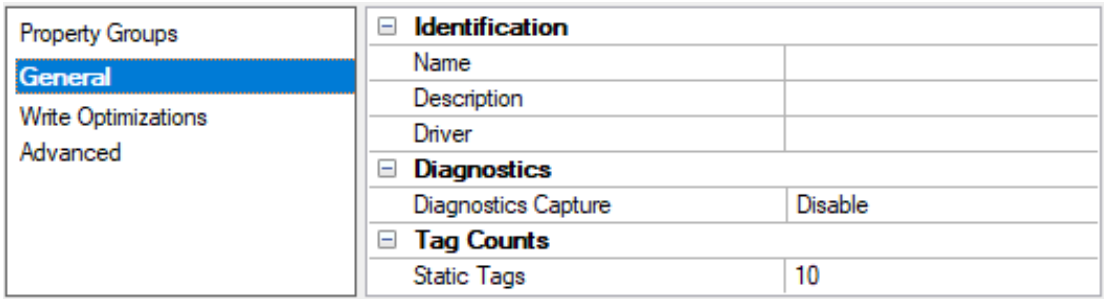
See Also: **Block Sizes** and **Device Settings**.

RS-485 to RS-232 Hardware Setup

The UDC communicates via RS-485, and the PC via RS-232. An RS-232 to RS-485 converter must be purchased separately.

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.



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.

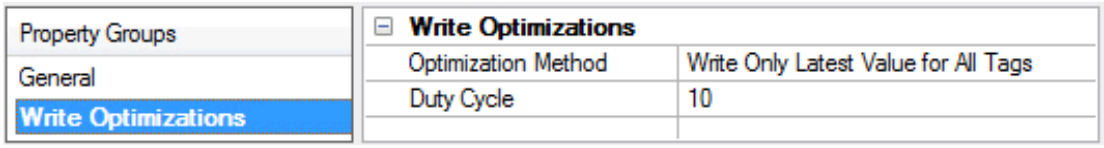
For more information, refer to "Communication Diagnostics" and "Statistics Tags" in the server help.

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 — 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.



Write Optimizations

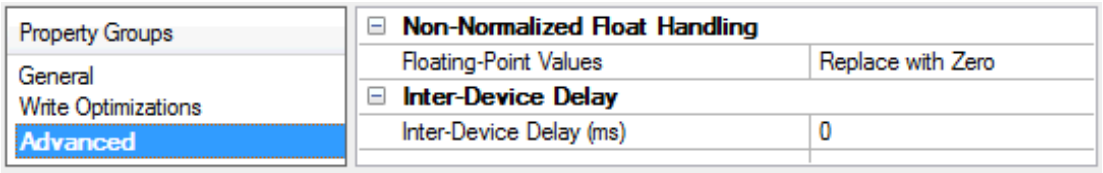
Optimization Method: Controls how write data is passed to the underlying communications driver. The options are:

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

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

Channel Properties — Advanced

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



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

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

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

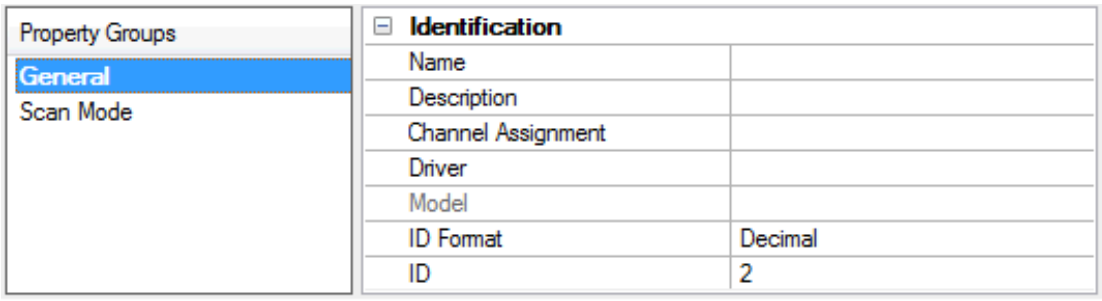
For more information on the floating-point values, refer to "How To ... Work with Non-Normalized Floating-Point Values" in the server help.

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

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

Device Properties — General

A device represents a single target on a communications channel. If the driver supports multiple controllers, users must enter a device ID for each controller.



Identification

Name: Specify the name of the device. It is a logical user-defined name that can be up to 256 characters long and may be used on multiple channels.

Note: Although descriptive names are generally a good idea, some OPC client applications may have a limited display window when browsing the OPC server's tag space. The device name and channel name become part of the browse tree information as well. Within an OPC client, the combination of channel name and device name would appear as "ChannelName.DeviceName".
For more information, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in server help.

Description: Specify the user-defined information about this device.
Many of these properties, including Description, have an associated system tag.

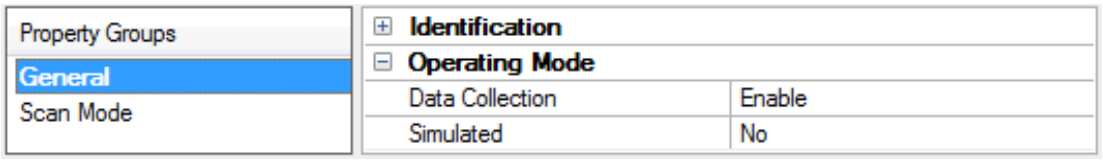
Channel Assignment: Specify the user-defined name of the channel to which this device currently belongs.

Driver: Selected protocol driver for this device.

Model: Specify the type of device that is associated with this ID. The contents of the drop-down menu depend on the type of communications driver being used. Models that are not supported by a driver are disabled. If the communications driver supports multiple device models, the model selection can only be changed when there are no client applications connected to the device.
Note: If the communication driver supports multiple models, users should try to match the model selection to the physical device. If the device is not represented in the drop-down menu, select a model that conforms closest to the target device. Some drivers support a model selection called "Open," which allows users to communicate without knowing the specific details of the target device. *For more information, refer to the driver documentation.*

ID: Specify the device's driver-specific station or node. The type of ID entered depends on the communications driver being used. For many communication drivers, the ID is a numeric value. Drivers that support a Numeric ID provide users with the option to enter a numeric value whose format can be changed to suit the needs of the application or the characteristics of the selected communications driver. The format is set by the driver by default. Options include Decimal, Octal, and Hexadecimal.

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.

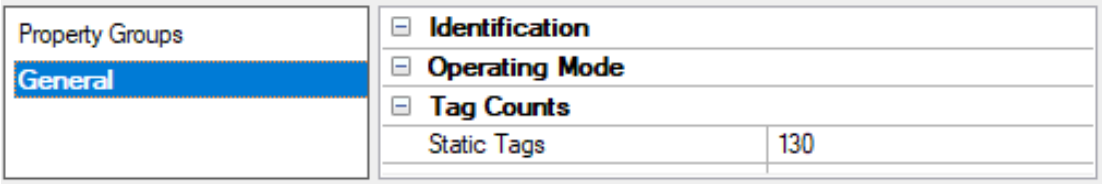
Simulated: Place the device into or out of Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client 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 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. When a device is simulated, updates may not appear faster than one (1) second in the client.

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

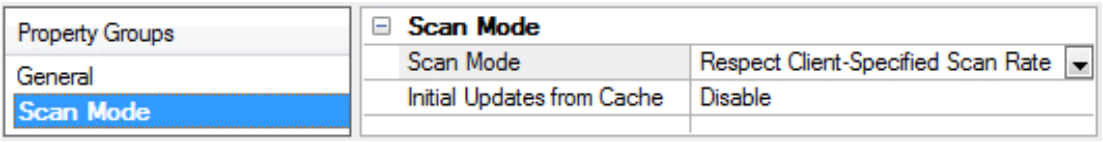
Tag Counts



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.

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.



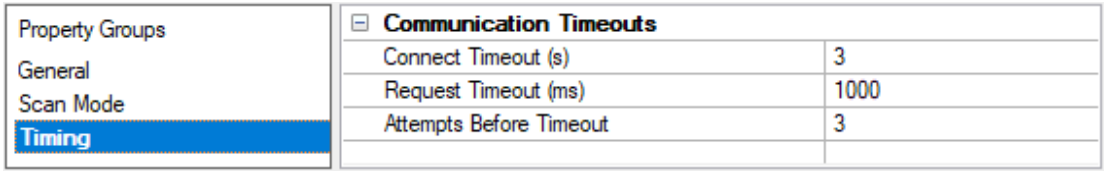
Scan Mode: Specify how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

- **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.



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.

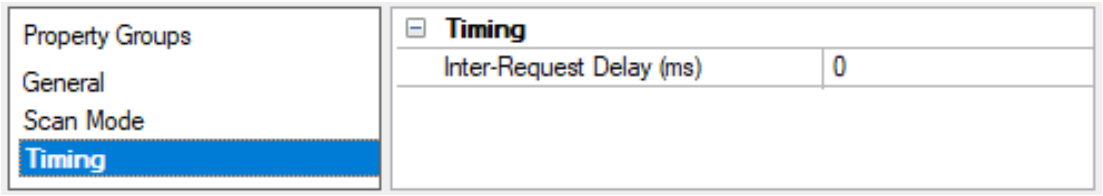
Request Timeout: Specify an interval used by all drivers to determine how long the driver waits for a response from the target device to complete. The valid range is 50 to 9999999 milliseconds (167 minutes). The default is usually 1000 milliseconds, but can vary depending on the driver. The default timeout for most serial drivers is based on a baud rate of 9600 baud or better. When using a driver at lower baud rates, increase the timeout to compensate for the increased time required to acquire data.

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

Timing

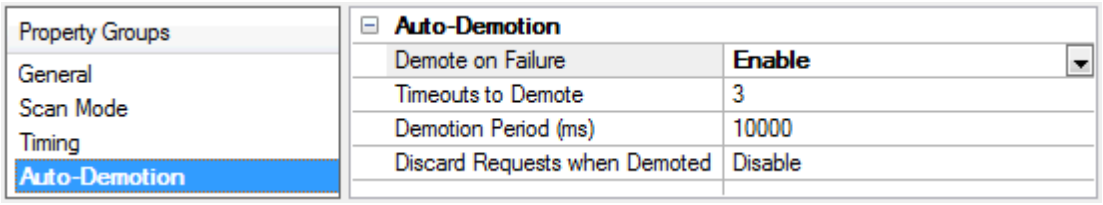
Inter-Request Delay: Specify how long the driver waits before sending the next request to the target device. It overrides the normal polling frequency of tags associated with the device, as well as one-time reads and writes. This delay can be useful when dealing with devices with slow turnaround times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

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



Device Properties — Auto-Demotion

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



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

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

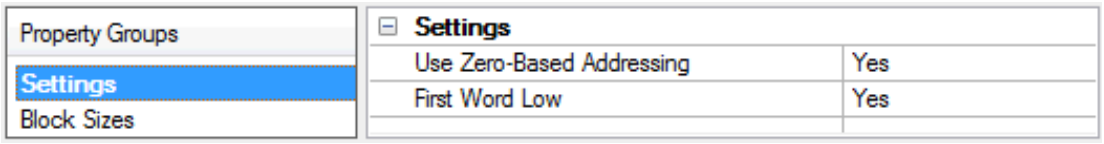
Timeouts to Demote: Specify how many successive cycles of request timeouts and retries occur before the device is placed off-scan. The valid range is 1 to 30 successive failures. The default is 3.

Demotion Period: Indicate how long the device should be placed off-scan when the timeouts value is reached. During this period, no read requests are sent to the device and all data associated with the read requests are set to bad quality. When this period expires, the driver places the device on-scan and allows for

another attempt at communications. The valid range is 100 to 3600000 milliseconds. The default is 10000 milliseconds.

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

Device Properties — Settings



Settings

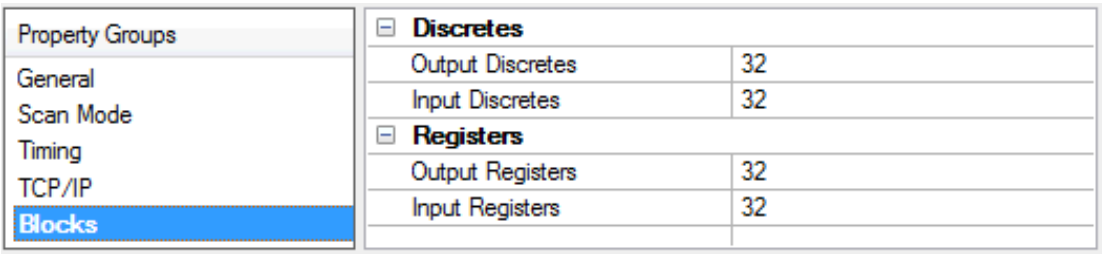
Use Zero-Based Addressing: If the device's address numbering convention starts at zero, specify Yes. When frames are constructed to communicate with a Honeywell UDC Serial device, user-defined addresses have one subtracted by default. This behavior follows the convention of the Honeywell UDC Serial devices.

First Word Low: Two consecutive registers addresses in a Honeywell UDC Serial device are used for 32-bit data types. Specify whether the driver should assume the first word is the low or the high word of the 32-bit value. The default, first word high, follows the convention of the Honeywell UDC Serial.

For the best communication behavior, try the following UDC settings:

ComSTATE = MODB3K
ComADDR =2
SHEDENAB= DISABLE
BAUD = 9600
DUPLEX = HALF
WS FLOAT = FP B
TX DELAY = 100
UNITS = ENG
CSP RATO = 1.00
CSP BIAS = 0.0
LOOPBACK = DISABLE

Device Properties — Block Sizes



Coils

Output Coils: Coils can be read from 8 to 800 points (bits) at a time. A higher block size means more points will be read from the device in a single request. If data needs to be read from non-contiguous locations within the device, the block size can be reduced.

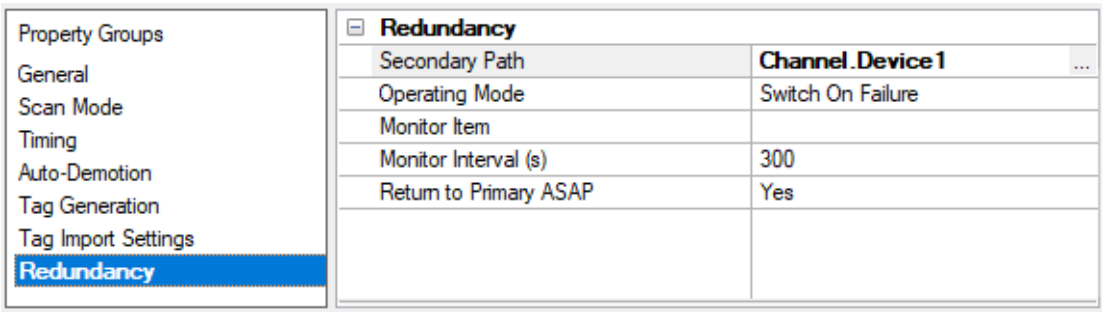
Input Coils: Coils can be read from 8 to 800 points (bits) at a time. A higher block size means more points will be read from the device in a single request. If data needs to be read from non-contiguous locations within the device, the block size can be reduced.

Registers

Internal Registers: Internal registers can be read from 1 to 125 locations (words) at a time. A higher block size means more register values will be read from the device in a single request. If the data needs to be read from non-contiguous locations within the device, the block size can be reduced.

Holding Registers: Holding registers can be read from 1 to 125 locations (words) at a time. A higher block size means more register values will be read from the device in a single request. If the data needs to be read from non-contiguous locations within the device, the block size can be reduced.

Device Properties — Redundancy



Redundancy is available with the Media-Level Redundancy Plug-In.
*Consult the website, a sales representative, or the **user manual** for more information.*

Data Types Description

The descriptions below assume first word low data handling of 32-bit data types.

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
DWord	Unsigned 32-bit value bit 0 is the low bit bit 31 is the high bit
Long	Signed 32-bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
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.
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 40001 is specified as a float, bit 0 of register 40001 would be bit 0 of the 32-bit word, and bit 15 of register 40002 would be bit 31 of the 32-bit word.

Honeywell UDC Serial 3000/3300 Address Descriptions

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

Honeywell UDC Serial Addressing Decimal Format

Address	Range	Data Type	Access
Output Coils [Function Codes (decimal): 01, 05, 15]	000001-065536	Boolean	Read/Write
Input Coils [Function Code (decimal): 02]	100001-165536	Boolean	Read Only
Internal Registers [Function Code (decimal): 04]	300001-365536 300001-365535 300001.0-365535.15	**Short**, Word, BCD Float, DWord, Long, LBCD Boolean	Read Only
Holding Registers [Function Codes (decimal): 03, 06, 16]	400001-465536 400001-465535 400001.0-465535.15	**Short**, Word, BCD Float, DWord, Long, LBCD Boolean	Read/Write

Honeywell UDC Serial Configuration ID Tags Decimal Format

Address	Range	Data Type	Access
Loop 1 Confguration Parameters (Floating Point)	GR0:0-GR0:127	**Float**	Read/Write*
Loop1 Configuration Parameters (Integer)	GR0:128-GR0:255	**Short**, Word, BCD	Read/Write*
Loop 2 Confguration Parameters (Floating Point)	GR1:0-GR1:127	**Float**	Read/Write*
Loop 2 Configuration Parameters (Integer)	GR1:128-GR1:255	**Short**, Word, BCD	Read/Write*

*Some Configuration ID Tags may be Read Only. For more information, refer to the UDC 3000/3300 documentation.

Honeywell UDC Serial Addressing Hexadecimal Format

Address	Range	Data Type	Access
Output Coils [Function Codes (decimal): 01, 05, 15]	H000001-H0FFFF	Boolean	Read/Write
Input Coils [Function Code (decimal): 02]	H100001-H1FFFF	Boolean	Read Only
Internal Registers [Function Code (decimal): 04]	H300001-H310000 H300001-H3FFFF H30001.0-H3FFFF.F	**Short**, Word, BCD Float, DWord, Long, LBCD Boolean	Read Only
Holding Registers [Function Codes (decimal): 03, 06, 16]	H400001-H410000 H400001-H4FFFF H40000.0-H4FFFF.F	**Short**, Word, BCD Float, DWord, Long, LBCD Boolean	Read/Write

Honeywell UDC Serial Configuration ID Tags Hexadecimal Format

Address	Range	Data Type	Access
Loop 1 Confguration Parameters (Floating Point)	HGR0:0-HGR0:7F	**Float**	Read/Write*
Loop1 Configuration Parameters (Integer)	HGR0:80-HGR0:FF	**Short**, Word, BCD	Read/Write*
Loop 2 Confguration Parameters (Floating Point)	HGR1:0-HGR1:7F	**Float**	Read/Write*
Loop2 Configuration Parameters (Integer)	HGR1:80-HGR1:FF	**Short**, Word, BCD	Read/Write*

*Some Configuration ID Tags may be Read Only. For more information, refer to the UDC 3000/3300 documentation.

Examples

- Address 40001 will access the PV or Process Variable scaled by a factor of 10.
- Address 40006 will access the PB Proportional Band (Gain).
- Address GR0:120 will access the PV Process Variable in float format.
- Address GR0:255 will access UDC error status.

Arrays

Arrays are supported for internal and holding register locations for all data types except Booleans or the Configuration ID Tags. There are two methods of addressing an array. Examples are given using holding register locations.

4xxxx [rows] [cols]
4xxxx [cols] this method assumes rows is equal to one

Rows multiplied by cols cannot exceed the block size that has been assigned to the device for the register type. For arrays of 32-bit data types, rows multiplied by cols multiplied by 2 cannot exceed the block size.

Error Descriptions

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

Address Validation

Serial Communications

Device Status Messages

Honeywell UDC Serial Device Specific Messages

Missing address

Error Type:

Warning

Possible Cause:

A tag address that has been specified statically has no length.

Solution:

Re-enter the address in the client application.

Device address <address> contains a syntax error

Error Type:

Warning

Possible Cause:

A tag address that has been specified statically contains one or more invalid characters.

Solution:

Re-enter the address in the client application.

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

Error Type:
Warning

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

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

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

Error Type:
Warning

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

Solution:
Modify the requested data type in the client application.

Device address <address> is Read Only

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically has a requested access mode that is not compatible with what the device supports for that address.

Solution:
Change the access mode in the client application.

Array size is out of range for address <address>

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically is requesting an array size that is too large for the address type or block size of the driver.

Solution:
Re-enter the address in the client application to specify a smaller value for the array or a different starting point.

Array support is not available for the specified address: <address>

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically contains an array reference for an address type that doesn't support arrays.

Solution:
Re-enter the address in the client application to remove the array reference or correct the address type.

COMn does not exist

Error Type:
Fatal

Possible Cause:
The specified COM port is not present on the target computer.

Solution:
Verify that the proper COM port has been selected in the Channel Properties.

Error opening COMn

Error Type:
Fatal

Possible Cause:
The specified COM port could not be opened due to an internal hardware or software problem on the target computer.

Solution:
Verify that the COM port is functional and may be accessed by other Windows applications.

COMn is in use by another application

Error Type:
Fatal

Possible Cause:
The specified COM port is not present on the target computer.

Solution:
Verify that the proper COM port has been selected in the Channel Properties.

Unable to set comm parameters on COMn

Error Type:
Fatal

Possible Cause:
The serial parameters for the specified COM port are not valid.

Solution:
Verify the serial parameters and make any necessary changes.

Communications error on <channel name> [<error mask>]

Error Type:
Serious

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.

Possible Cause:
1. The serial connection between the device and the Host PC is bad.
2. The communications parameters for the serial connection are incorrect.

Solution:
1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communications parameters match those of the device.

Device <device name> is not responding

Error Type:
Serious

Possible Cause:
1. The serial connection between the device and the Host PC is broken.
2. The communications parameters for the serial connection are incorrect.
3. The named device may have been assigned an incorrect Network ID.
4. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device property.

Solution:
1. Verify the cabling between the PC and the PLC device.
2. Verify the specified communications parameters match those of the device.

3. Verify the Network ID given to the named device matches that of the actual device.
4. Increase the Request Timeout property so that the entire response can be handled.

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

Error Type:
Serious

Possible Cause:

1. The serial connection between the device and the Host PC is broken.
2. The communications parameters for the serial connection are incorrect.
3. The named device may have been assigned an incorrect Network ID.

Solution:

1. Verify the cabling between the PC and the PLC device.
2. Verify the specified communications parameters match those of the device.
3. Verify the Network ID given to the named device matches that of the actual device.

Bad address in block [<start address> to <end address>] on device <device name>

Error Type:
Serious

Possible Cause:
An attempt has been made to reference a nonexistent location in the specified device.

Solution:
Verify the tags assigned to addresses in the specified range on the device and eliminate ones that reference invalid locations.

Index

E

F

G

H

I

L

M

N

O

P

R

S

T

U

W