Simulator Driver Help

Table of Contents

Simulator Driver Help

Help version 1.027

CONTENTS

Overview

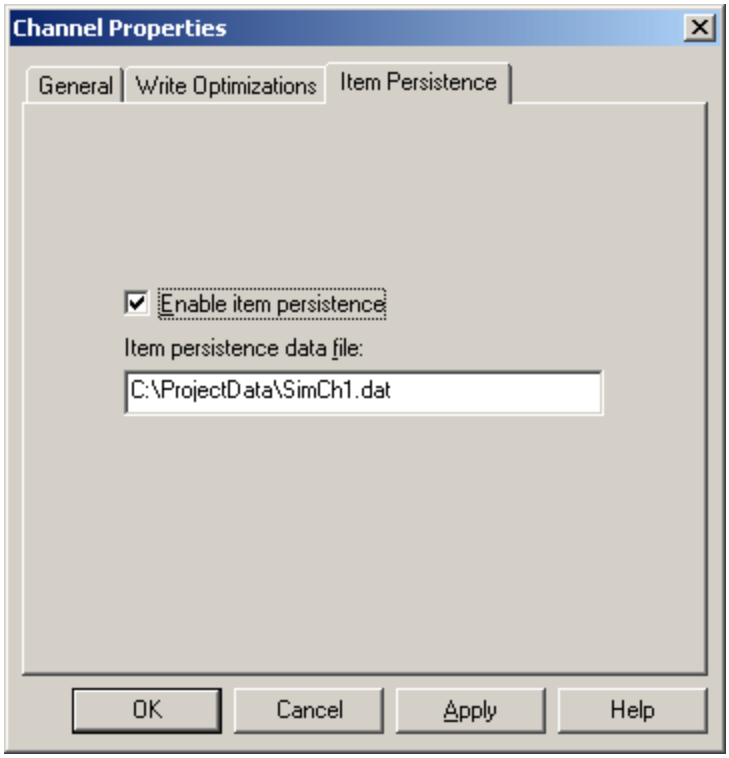
The Simulator Driver provides an easy and reliable way to connect Simulator devices to OPC Client applications, including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is provided for testing the OPC Server software product without requiring an external device.

Channel Setup

Item Persistence

The Simulator Driver may be configured to retain its values of K (constant), R (register), and S (string) addresses between runs. Item persistence for simulation functions is not currently supported. When enabled, the values stored in all K, R, and S addresses of each device configured on the channel are saved to file when the server project is closed (or the server is shut down). These values are restored from file the next time the server project is opened. A separate file is used for each Simulator Driver channel using this feature. For more information, refer to **Simulation Functions**.

Note: This feature may be configured through the **Item Persistence** tab in Channel Properties.



Descriptions of the parameters are as follows:

- **Enable Item Persistence:** This parameter controls the status of the feature. The default setting is disabled.
- **Item Persistence Data File:** This parameter specifies where the data should be stored for devices on this channel. A fully qualified path and file name is required. The driver creates the file and folders in its path, but users must use this feature to specify a unique file for each Simulator channel.

Device Setup

Supported Devices

8-Bit
16-Bit

Note: Each device supports up to 10000 addressable Read/Write register and constant locations, in addition to 1000 variable length, Read/Write string locations. For more information, refer to **Address Descriptions**.

Maximum Number of Channels and Devices

The maximum number of supported channels is 100. The maximum number of devices supported per channel is 999.

Device ID

Simulator devices can be assigned Device IDs in the range of 1 to 999. When using different model types within the same channel, unique Device IDs are required.

Live Data Simulation

The driver simulates live data by incrementing register data each time it is read as an integer data type. In addition to simulating the simple register-based memory of a PLC-like device, the Simulator Driver also supports four high-level simulation functions. These new simulation functions include RAMP, SINE, RANDOM, and USER Defined.

Each new simulation tag is structured like a function call would be in a programming language. Using each function requires that the appropriate parameters be applied to determine the desired simulation effect. With RAMP, there is the option to set the rate of change, the low limit, the high limit, and the increment value. With SINE, there is the option to set the rate of change, the low limit, the high limit, the frequency, and the phase. With RANDOM, users have the option to set the rate of change, the low limit, and the high limit. The most creative of the new simulation functions, however, is the USER Defined function.

The USER Defined function similarly provides the option to specify a rate of change. Unlike the preset simulation outputs of the RAMP, RANDOM, and SINE functions, the USER Defined function is used to create sequences of data. In the place of the high limits or low limits, the USER Defined function accepts a comma separated list of items. The list of items can be either numeric data or string data. Once a list is established, the USER Defined function cycles through the elements of the list at the rate specified. The USER Defined function can create complex demos that can be used to mirror real world outputs and results.

Note: For more information, refer to **Simulation Functions**.

Data Types Description

Each address that can be accessed must be assigned a data type. Simulator devices support the following data types.

Data Type	Description
Boolean	Single bit
Byte	Unsigned 8-bit value bit 0 is the low bit bit 7 is the high bit
Char	Signed 8-bit value bit 0 is the low bit bit 6 is the high bit bit 7 is the sign 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
QWord	Unsigned 64-bit value bit 0 is the low bit bit 63 is the high bit
LLong	Signed 64-bit value bit 0 is the low bit bit 62 is the high bit bit 63 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.
String	Null-terminated ASCII character array
Float*	32-bit floating point value The driver interprets two consecutive registers as a single precision value by making the last 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 data type and bit 15 of register 40002 would be bit 31 of the 32-bit data type.
Double*	64-bit floating point value The driver interprets four consecutive registers as a double precision value by making the last two registers the high DWord and the first two registers the low DWord.
Double example	If register 40001 is specified as a double, bit 0 of register 40001 would be bit 0 of the 64-bit data type and bit 15 of register 40004 would be bit 63 of the 64-bit data type.

*The descriptions assume the default; that is, first DWord low data handling of 64-bit data types and first word low data handling of 32-bit data types.

Address Descriptions

The Simulator Driver supports three types of addresses: R registers, K registers, and S registers. The R and K registers are numeric data. S registers are variable length string data locations.

The R registers simulate changing data by incrementing by one on each read when referenced as type Char, Byte, Word, Short, BCD, Long, DWord, LLong, QWord, or LBCD. Arrays of these types increment by one on each read. When R registers are referenced as type Float or Double, the value is incremented by 1.25 on each read. Arrays of type Float or Double do not increment unless there are individual tags for the addresses in the array. Furthermore, each type has a range over which the incrementing occurs. For type Float, the range is 0 to 32767. For type Double, the range is 0 to 65535.

The K and R registers have an initial value of zero. S registers have an initial value of 'String data S*n*' where *n* is the register number.

Address range and data type specifications vary depending on the model in use. The Simulator Driver also supports new simulation functions, which include RAMP, SINE, RANDOM and USER Defined. For more information, select a link from the list below.

8-Bit Device Addresses
16-Bit Device Addresses

8-Bit Device Addresses

The memory configuration for the 8-bit Device is simulated as a block of byte locations numbered from 0 to 9999. Each byte can be addressed as an offset from the start of the block. The default data types for each format are shown in **bold**.

Device Type	Range	Data Type	Access
Registers	R0000-R9999 R0000-R9998 R0000-R9996 R0000-R9992	**Byte**, Char Word, Short, BCD DWord, Long, LBCD, Float LLong, QWord, Double	Read/Write
Constants	K0000-K9999 K0000-K9998 K0000-K9996 K0000-K9992	**Byte**, Char Word, Short, BCD DWord, Long, LBCD, Float LLong, QWord, Double	Read/Write
Bits	R0000.0-R9999.7 K0000.0-K9999.7	**Boolean**	Read/Write
Strings	S000-S999	**String**	Read/Write

Note 1: All data types except Boolean and String support arrays by appending the [r] or [r][c] notation to the address.

Note 2: The address specified for a data type must allow for the full size of the data type. This means that users cannot write past the end of the data range.

See Also: **Simulation Functions**

16-Bit Device Addresses

The memory configuration for the 16-Bit Device is simulated as a block of word locations numbered from 0 to 9999. Each word can be addressed as an offset from the start of the block. The default data types for each format are shown in **bold**.

Device Type	Range	Data Type	Access
Registers	R0000-R9999 R0000-R9998 R0000-R9996	**Word**, Short, BCD DWord, Long, LBCD, Float LLong, QWord, Double	Read/Write
Constants	K0000-K9999 K0000-K9998 K0000-K9996	**Word**, Short, BCD DWord, Long, LBCD, Float LLong, QWord, Double	Read/Write
Bits	R0000.00-R9999.15 K0000.00-K9999.15	**Boolean**	Read/Write
Strings	S000-S999	**String**	Read/Write

Note 1: All data types (except Boolean and String) support arrays by appending the [r] or [r][c] notation to the address.

Note 2: The address specified for a data type must allow for the full size of the data type. This means that users cannot write past the end of the data range.

See Also: Simulation Functions

Simulation Functions

Simulation functions can be used to create OPC items that mimic many real world data sources. While each of the simulation functions provides different outputs, they have many common parameters such as **Rate**, **Low Limit**, and **High Limit**. The Rate (which is also called Rate of Change) is used to specify how often the simulation value changes states. The Rate value is entered as a count of milliseconds with the value range being 10 to 3600000. This rate of change is completely independent of how often the client application may be reading data. Like a PLC, the simulation functions are always running in the background.

The Low Limit and High Limit

The Low Limit and High Limit parameters can be used to specify the range or span of data generated by the simulation function. By using the Low Limit and High Limit setting, users can produce simulation values that are offset by simply adjusting the span of the data. For the simulation functions that support limits, the format in which the range is entered is used to determine the data type of the simulation value. If, for example, a RAMP tag is entered with a Low Limit of 75 and a High Limit of 3000 the resulting OPC item is considered to be a Long data type. If the same RAMP is entered with a Low Limit of 75.123 and a High Limit of 3000.567 the resulting OPC item is considered to be Float data type. In these two examples, the default data format was either Float or Long, but any of the available data types can be chosen as the output format for any simulation function.The range specified by the Low and High Limits, however, must fit within the desired data type selection.

Unlike the register-based address above, there is no limit to the number of simulation functions that can be entered. Each simulation function that has unique parameters is considered a new simulation value. Thus, when creating addressing simulation functions with the intent of reading the same value in multiple locations in the client application, it is important that the simulation function is entered the same way in each case.

Simulation Functions are Read Only Objects

Simulation functions are Read Only objects. Once a client application begins reading data from a simulation function, the function continues to generate data until the item is deleted by a client application. When entering floating point parameters, the simulation functions do not accept the entry of numeric values in exponential form.

Functions Definitions

Ramp Function
Random Function
Sine Function
User Function

Ramp Function

RAMP(Rate, Low Limit, High Limit, Increment)

The RAMP function can be used to create a value that increments or decrements through a numeric range. The low limit and h igh limit should be used to set the desired range. The low or high limits can be adjusted to apply an offset to the data generated. The increment value can be either a positive or negative value. If the increment value is positive, the value generated ramps from the low limit to the high limit at the desired rate. If the increment value is negative, the value generated ramps from the high limit to the low limit at the desired rate. The values of low limit, high limit, and increment can be entered either as whole numbers or in floating-point format.

Supported Data Types

Byte, Char, Word, Short, DWord, **Long**, Float, Double

Examples

RAMP(120, 35, 100, 4)
This creates a value that ramps up from 35 to 100 incremented by 4 every 120 milliseconds.

RAMP(300, 150.75, 200.50, -0.25)
This creates a value that ramps down from 200.50 to 150.75 decremented by 0.25 every 300 milliseconds.

Random Function

RANDOM(Rate, Low Limit, High Limit)

The RANDOM function can be used to create an item that changes randomly within a specific numeric range. The Low Limit and High Limit should be used to set the desired range. The Low or High limits can be adjusted in order to apply an offset to the data generated.

Supported Data Types

Byte, Char, Word, Short, DWord, **Long**, Float, Double

Example

RANDOM(30, -20, 75)
This creates a value that randomly changes within the range of -20 to 75 at a rate of 30 milliseconds.

Sine Function

SINE(Rate, Low Limit, High Limit, Frequency, Phase)

SINE function can be used to create an item that follows a sinusoidal change of value. The Low Limit and High Limit should be used to set the desired range. The Low or High limits can be adjusted in order to apply an offset to the data generated. The Frequency parameter can be used to specify the desired waveform in Hertz. The maximum effective frequency is about 5 Hertz. The valid range for the Frequency parameter is 0.001 to 5 Hertz. The Phase parameter can be used to offset the sine wave generated by a specific angle. Phase should be entered with a range of 0.0 to 360.0. The Rate parameter in this case plays a key role in how this simulation function operates. In order to get a good sinusoidal output from this function, the Rate must at least twice as fast as the desired Frequency. For example, if users desire a sine wave of 5 Hertz which changes at about a 200 millisecond rate, the Rate parameter should be set to 100 milliseconds at maximum. For the best Sine wave results setting the Rate to either 10 or 20 milliseconds is recommended. The valid range of Rate for a SINE function is 10-1000 milliseconds.

Supported Data Types

Float, Double

Example

SINE(10, -40, 40, 2, 0)
This creates a sinusoidal value with a frequency of 2 Hertz that ranges from -40 to 40 with no phase shift.

User Function

USER(Rate, User Value1, User Value2, User Value3, ...User ValueN)

The USER function provides the most flexibility in defining what type of data the simulation function returns. Unlike the other functions that operate over a specified range, the USER function can be used to specify a set of numeric or string values to be cycled through at the specified rate. The values entered are used to determine data type of this item. For example, if a value of 100.45 is entered as one of the user values, the output of the simulation object would be considered to be floating point data. If one of the user values entered was "Hello World" the output of the simulation object would be considered to be string data. These default selections can be overridden by specifying the desired data type when the item is defined.

Note: When entering user values as strings the comma can be entered within a string value by first prefixing it with the backslash "\,".

Supported Data Types

Bool, Byte, Char, Word, Short, DWord, Long, Float, Double
Note: The values entered in the USER simulation function are used to determine the default data type.

Examples

USER(250, Hello, World, this, is, a, test)
This creates a value of data type string that changes from one text word in the sequence to the next at a rate of 250 milliseconds.

USER(20, 1.25, 100.56, 200.11,75.1)
This creates a value of data type float that changes from one floating-point value in the sequence to the next at a rate of 20 milliseconds.

USER(50, 1,1,0,1,0,1,0,0,1,1,1,0,0,0)
This generates a value of type Boolean that changes from one Boolean state in the sequence to the next at a rate of 50 milliseconds. This can be used to create very complex bit patterns.

USER(1000, In this case\, , I needed to use a \, in , my text)
The "\," in these text fragments were needed to allow a comma to be placed within a text value. Additionally the text for each value can be a sentence or sentence fragment if needed.

Error Descriptions

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

Address Validation

Item Persistence

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 that the address is correct; if it is not, re-enter it 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.

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> 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.

Device address <address> is not supported by model <model name>

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically references a location that is valid for the communications protocol but not supported by the target device.

Solution:
Verify that the address is correct; if it is not, re-enter it in the client application. Verify also that the selected model name for the device is correct.

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.

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.

Could not <load/save> item state data. Reason: <reason>

Error Type:
Serious

Possible Cause:
1. The driver could not load or save item state data for the specified reason.
2. Corrupt data files.
3. Inadequate disk space.
4. Invalid drive in path.
5. Deleted or renamed data files.

Solution:
Solution depends upon the reason given in the error message. In the case of file corruption or deletion, previous state data is lost.

Could not allocate memory for simulated device

Error Type:

Serious

Possible Cause:
The driver could not acquire memory resources needed for simulated device.

Solution:
Close all unnecessary applications. Increase physical or virtual memory in computer.

Index

U

W