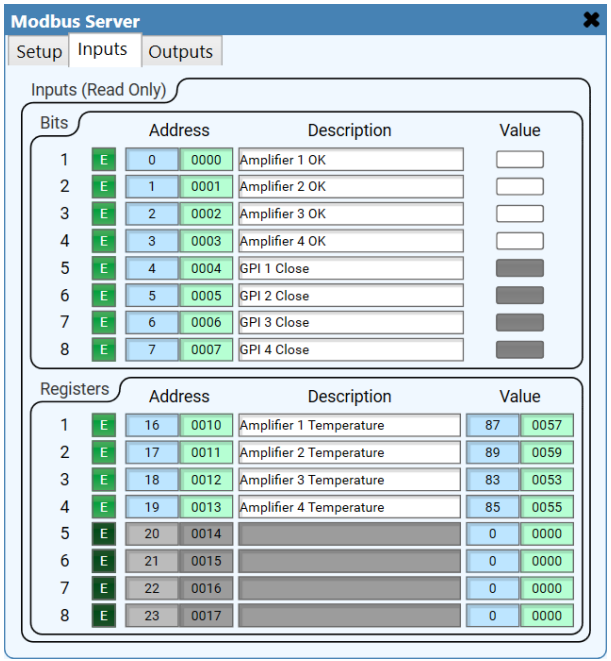
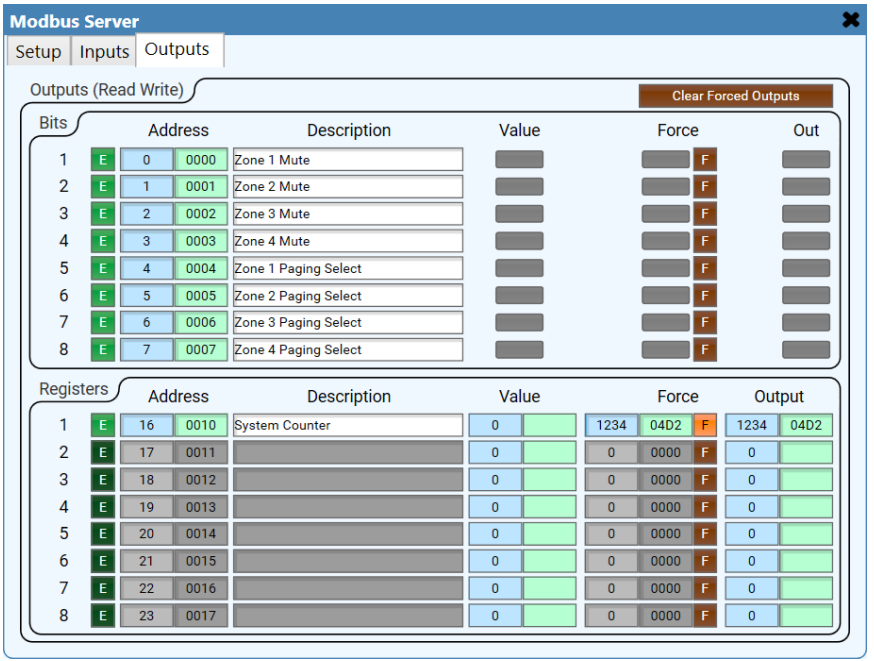
Plugin Guide

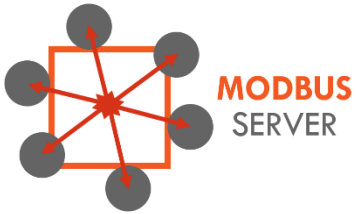
MODBUS SERVER

Version 1.0.0
November 28, 2020





Programation.io



MODBUS SERVER

offers a flexible set of controls to be external read from and written to via the Modbus TCP and JBus protocols. Now the Q-SYS platform can be much more easily leveraged as a critical subsystem for paging and alarm purposes as the Modbus Server plugin offers easy integration with most Building Management System (BMS) software.

The plugin offers up to 999 each of Input Bits, Input Registers, Output Bits, and Output Registers offering a staggering number of control IO points – hundreds more than a typical control system. Inputs are read-only from a Modbus Client and represent system status or feedback information. General purpose inputs can be passed directly via control pins to allow Q-SYS hardware to act as real digital discrete inputs. Outputs are read-write and allow elements to be controlled from a Modbus Client. Both Input and Output Bits are represented as toggle buttons for easy control pin wiring. Registers are represented as integer fields with possible values from 0 to 65,535, and as hexadecimal string fields with values from 0x0000 to 0xFFFF.

Each Input and Output in the plugin may be configured for any IO address and do not have to be sequential. Unused IO can be disabled to reduce clutter and confusion. IO may also be temporarily disabled while testing or commissioning is ongoing. This provides maximum flexibility for working with larger, more complex systems.

Outputs offer a “Force” feature which allows manual override of data being written from the Modbus Client. When a Force is active, the provided value will be applied to the output allowing for easier system testing, commissioning, and temporary bypass.

Refer to the Plugin Guide Notes for information regarding protocol compatibility.

Properties

Input Bits

an `integer` between `0` and 999 that defines the number of input bits available for configuration

Input Registers

an `integer` between `0` and 999 that defines the number of input registers available for configuration

Output Bits

an `integer` between `0` and 999 that defines the number of output bits available for configuration

Output Registers

an `integer` between `0` and 999 that defines the number of output registers available for configuration

Registered Owner

a `string` that lists the license holder name of the plugin registration for this installation

Registered Project

a `string` that lists the licensed project name of the plugin registration for this installation

Registered Key

a `string` that lists the license key code of the plugin registration for this installation

Controls

Status

displays various status messages to aid in monitoring the connections to external devices. The number of connected clients is listed, when available. Communication errors will show as a fault. When a force is present on an output, the status will show as Compromised as a reminder that the system has been overridden.

TCP Port

sets the TCP communication port number that clients should connect to

Bus Address

sets the Modbus Bus Address (or Device ID) of this server. This is useful if more than one Server exists at the same IP address. Refer to the Notes section for more details.

Incoming Counter

displays the number of incoming messages that the plugin has received from the connected Client.

Successful Counter

displays the number of incoming messages that were successfully acted upon. If this number does not increase when the Incoming Counter increases, this means the Client has made a request that does not match the configuration of the IO. For example, an IO could be disabled, or a requested IO address does not exist in the configuration.

Counter Reset

a trigger that resets the Incoming Counter and Successful Counter to 0.

Controls

Input Enable

a toggle button that sets whether this Input should be allowed for external read

Input Dec Address

an integer field that ranges from 0 to 65,536 that sets the memory address of the input in decimal

Input Hex Address

a string field that ranges from 0x0000 to 0xFFFF that sets the memory address of the input in hexadecimal

Input Description

a text field that provides human-readable labeling of each input

Input Bit Value

a toggle button that sets the bit high or low

Input Register Dec Value

an integer field that ranges from 0 to 65,535 that sets the value of the register in decimal

Input Register Hex Value

a string field that represents the register value in hexadecimal, ranging from 0x000 to 0xFFFF

Controls

Output Enable

a toggle button that sets whether this Output should be allowed for external read or write

Output Dec Address

an integer field that ranges from 0 to 65,536 that sets the memory address of the output in decimal

Output Hex Address

a string field that ranges from 0x0000 to 0xFFFF that sets the memory address of the output in hexadecimal

Output Description

a text field that provides human-readable labeling of each output

Output Bit Value

a toggle button that indicates the state of the output as set by the Modbus Client

Output Bit Force Value

a toggle button that sets the value of the output in the forced state

Output Bit Force Enable

a toggle button that overrides the output bit's value with the value specified by the Bit Force Value toggle button. When active, the output will only follow the forced value and ignore changes made by the Modbus Client.

Output Bit Output

a toggle button that represents the actual output state, after considering the forced state

Output Register Dec Value

an integer field ranging from 0 to 65,535 that indicates the decimal value of the output as set by the Client

Output Register Hex Value

a string field ranging from 0x0000 to 0xFFFF that indicates the hexadecimal value of the output as set by the Client

Output Register Dec Force Value

an integer field ranging from 0 to 65,535 that sets the decimal value of the output in a forced state

Output Register Hex Force Value

a string field ranging from 0x0000 to 0xFFFF that sets the hexadecimal value of the output in a forced state

Output Register Force Enable

a toggle button that overrides the output register's value with the value specified by the Register Force Dec/Hex Value fields. When active, the output will only follow the forced values and ignore changes made by the Modbus Client.

Output Register Dec Output

an integer field ranging from 0 to 65,535 that represents the actual output value in decimal, after considering the force state

Output Register Hex Output

a string field ranging from 0x0000 to 0xFFFF that represents the actual output value in hexadecimal, after considering the force state

Notes

Supported Modbus Functions

The Modbus Server plugin is compliant with the official Modbus TCP protocol and supports the methods:

- 0x01 (Read Coils)
- 0x02 (Read Discrete Inputs)
- 0x03 (Read Holding Registers)
- 0x04 (Read Input Registers)
- 0x05 (Write Single Coil)
- 0x06 (Write Single Register)
- 0x15 (Write Multiple Coils)
- 0x016 (Write Multiple Registers)
- 0x23 (Read/Write Multiple Registers)

I/O Addressing – Supporting both Modbus TCP and JBus

Modbus TCP is somewhat modified version of the original Modbus Serial communication protocol. Modbus TCP has certain limitations to the number of addresses available and the quantity of coils and registers that can be communicated in a single request. This is to keep a TCP message under the default MTU size. JBus, on the other hand, is more relaxed with these requirements. The most noticeable difference between the two protocols for system programmers is that Modbus TCP is 0-based in its memory addressing while JBus is 1-based.

Notes

The Modbus Server plugin is designed to not be concerned with whether the requests are 0-based or 1-based. It is up to the Q-SYS programmer to define the memory addresses for their use-case appropriately. The plugin does not apply any offset to the address numbers listed. If no I/O are addressed as 0, and a request is issued for memory address 0, then the request will fail.

Consequently, the plugin allows I/O to be addressed as high as 65,536 or 0x10000. This is to maintain compliancy with JBus, however it is an illegal data address for the Modbus protocol. Since all hexadecimal values are limited to 4 digits (2 bytes), the plugin will show a decimal address of 65,536 but a hexadecimal address of 0xFFFF.

Hexadecimal readouts of address and value fields are just for the programmer's convenience and do not have any practical effect on the data transmission. The plugin only considers decimal values for requests and transmissions. When a hexadecimal value is entered or is changed, however, the corresponding decimal field will also be updated to the correct value.

While the plugin will attempt to receive either Modbus TCP or JBus style messages without additional user configuration, it is recommended that messages stay under the default MTU size of 1500 bytes, otherwise the message may be split into multiple packets and the request may fail to be processed correctly.

I/O Priority – Non-Sequential and Non-Contiguous Addresses

As the Modbus Server plugin allows any Input or Output to have any address, addressing may be non-sequential and non-contiguous. This allows flexibility in only creating I/O that is needed at any address. Several installations

Notes

typically prefer to use logical numbering conventions for various groups of I/O which often skips over blocks of addresses. The plugin supports this without having to have a large number of unused controls.

When a request is received, the plugin searches in a top-down fashion from the lowest position I/O (that is, top of the list) and attempts to search for the first lowest address in the request. Once the item has been matched, the data is transferred and the pattern continues for the next address in the request. In the event that more than one I/O has the same address, the lowest position will be used and the higher positions will be ignored. If an I/O is not updating or seemingly not transmitting appropriately, verify that all addresses are set as intended.

Since reading input bits, input registers, output bits, and output registers are each a different Modbus method, addresses may be reused between the four sections without issue. Address matching only pertains to the type of I/O that corresponds to the read (or write) operation requested.

Bus Address

Traditional Modbus is a serial line protocol that supported multiple devices to be connected to the same line. Therefore, the protocol includes a "Bus Address" field which allows requests to target a specific device on the communication bus. Modbus TCP offers the same functionality for cases where multiple servers share the same IP address. When multiple instances of the Modbus Server plugin are to be used in the same Q-Sys design, it may be necessary for each plugin to use a different Bus Address (Device ID) so the Client can differentiate requests. In addition, only one plugin may bind to a TCP port at a time so each port number should also be unique. Failure to do so will only allow on Modbus Server plugin to work at a time. The default Modbus TCP port number is 502.

Notes

Error Handling

The Modbus Server plugin complies with the built-in Modbus protocol error messages for when requests are invalid or cannot be completed. The plugin may send one of the following Modbus Exception Codes for various situations:

Implemented Modbus Exception Codes

Code	Name	Cause
01	ILLEGAL FUNCTION	The Client issued a request using a command that the plugin cannot process. See the list of Supported Modbus Functions above.
02	ILLEGAL DATA ADDRESS	The Client issued a request for an I/O address that is not configured, or is one address that is not configured within a multi-read or multi-write request, or the address is configured on an I/O that is not enabled.
03	ILLEGAL DATA VALUE	The Client issued a request to write a data value that is larger than the maximum supported value.

Change Log

V1.0.0 – NOVEMBER 28, 2020

Original release