INDUSTRIAL CONTROL COMMUNICATIONS, INC.

Modbus/TCP Client Driver Manual

October 30, 2014

TABLE OF CONTENTS

1 Modbus/TCP Client

1.1 Overview

The Modbus/TCP client driver can be used to access information on any device which supports the Modbus/TCP server protocol. This driver also supports the 32-bit extension to the Modbus standard (commonly referred to as the “Enron/Daniel” extension).

Other notes of interest are:

- Service objects can access holding registers (4X references), input registers (3X references), coils (0X references) and discrete inputs (1X references).
- The driver is conformance class 0 and partial class 1 and class 2 compliant. Supported Modbus/TCP client functions are indicated in Table 1.

Table 1: Supported Modbus/TCP Client Functions

Function Code	Function	Modbus/TCP Class
01	Read coils	1
02	Read discrete inputs	1
03	Read holding registers	0
04	Read input registers	1
05	Write single coil	1
06	Write single register	1
15	Write multiple coils	2
16	Write multiple registers	0

- Supports up to 16 simultaneous connections (defined by the connection objects) and up to 64 total service objects.
- 32-bit register accesses are supported in a variety of options and formats.
- Standard 16-bit Modbus registers are mapped in the database as 2-byte values. This means that each register in a service object takes up two database addresses. For example, if a service object’s starting register is “1”, the number of registers is “5”, and the database address is “100”, then registers 1 through 5 will be mapped at database addresses 100 through 109 (register 1 mapped at address 100 and 101, register 2 mapped at address 102 and 103 and so on).
- Coils and Discrete Inputs (from here on collectively referred to as “discretes”) are mapped on a bit-by-bit basis in the database starting with the least significant bit of the database byte. For example, if a service object’s starting discrete is “1”, the number of discretes is “19”, and the database address is “320”, then discrete 1 through 8 will be mapped to bit 0 through 7, respectively, at address 320, discrete 9 through 16 will be mapped to bit 0 through 7, respectively, at address 321, and discrete 17 through 19 will be mapped to bit 0 through 2, respectively, at address 322. The remaining 5 bits in the byte at database address 322 are unused.

1.2 Client Settings

Scan Rate

This is the time in milliseconds the driver will wait between sending requests. This is a useful feature to reduce overall network utilization, or for certain devices or infrastructure components (such as radio modems) that may not be capable of sustaining the maximum packet rates that the driver is capable of producing. The start time for this delay is taken with respect to the moment at which the driver is capable of sending the next packet (due to either reception or timeout of the previous request). If no additional time is required, setting this field to 0 instructs the driver to send its next request packet as soon as possible.

1.3 Connection Object Settings

The Modbus/TCP client driver uses a construct called a connection object in order to target Modbus/TCP server devices. A connection object can be thought of as a communication channel or “pipe” which is created between the driver and the server device, independent of the service objects that may later make use of that communication channel to transfer service object requests. A connection object defines a connection to a specific endpoint (IP address).

Name

A unique name used for identifying the connection object. Enter a string of up to 16 characters in length.

IP Address

Defines the IP address of the server device to be targeted by the connection object. All connection object endpoints (IP addresses) must be unique.

1.4 Service Object Settings

The Modbus/TCP client driver uses service objects to describe what services the driver should perform. For each service object, the driver will continually read the registers or discretes defined within the service object from the designated server, storing the value(s) in the database (if the read function is enabled). When data in the database changes where the registers or discretes are mapped, a write request is generated to the designated server notifying it of the changed value(s) (if the write function is available and enabled).

1.4.1 Holding Register Service Object Settings

Description

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Unit ID

This field is used primarily for intra-system routing purposes. Many Modbus/TCP servers ignore this field.

Start Register

Defines the starting register number (1…65535) for a range of registers associated with this service object.

Number of Registers

Defines the number of registers (1…123 for standard 16-bit accesses or 1…61 for 32-bit accesses) to be targeted by this service object.

Database Address

Defines the database address where the first register of this service object will be mapped. The configuration studio will not allow entry of a starting database address that will cause the service object to run past the end of the database. The highest valid database address, therefore, will depend on the targeted data type, as well as the number of items to be accessed.

Multiplier

The amount that associated network values are scaled by prior to being stored into the database or after being retrieved from the database. Upon retrieval from the database, raw data is multiplied by the multiplier to produce a network value (to be sent to the device). Similarly, network values (read from the device) are divided by the multiplier before being stored into the database.

Read Function

Select whether or not to allow the service object to issue read requests. When enabled, the service object will continuously read from the server unless a pending write exists.

Write Function

Select whether or not to allow the service object to issue write requests, and the function code to use when enabled. When values encompassed by this service object change in the internal database and writes are enabled, these changes will be written down to the targeted server.

Data Type

Available only when the "32-Bit Registers" checkbox is checked. Defaults to standard Modbus "16-Bit Unsigned" when the "32-Bit Registers" checkbox is unchecked. Specifies how the value will be stored in the database for each register (or register pair) in this service object. This defines how many bytes will be allocated, whether the value should be treated as signed or unsigned, and whether the value should be interpreted as an integer or a floating point number. Select the desired data type from this dropdown menu.

1.4.1.1 32-Bit Options Settings

32-Bit Registers

Check this checkbox if the target registers are associated with the Enron/Daniel extension to the Modbus specification, or are represented by 32-bit values.

Floating Point

Available only when the "32-Bit Registers" checkbox is checked. Check this checkbox if the register values are encoded in IEEE 754 floating point format.

Big Endian

Available only when the "32-Bit Registers" checkbox is checked. Check this checkbox if the register values are encoded in big-endian 16-bit word order, i.e. the most significant 16-bit word is before the least significant 16-bit word.

Word-Size Register

Available only when the "32-Bit Registers" checkbox is checked. Check this checkbox if target registers are only 16-bits wide, but two 16-bit registers comprise one 32-bit value. If unchecked, target registers are assumed to be 32-bits wide.

Note that when this checkbox is checked, the "Number of Registers" field indicates the number of 16-bit register pairs. Each register pair will use two register addresses and the selected Data Type will be applicable for the register pair, not each individual register. For example, if Start Register is set to 100, Number of Registers is set to 2, and Data Type is set to 32-bit Unsigned, then register numbers 100…103 will be accessed by the service object, with registers 100 and 101 stored as the first 32-bit unsigned value and registers 102 and 103 stored as the next 32-bit unsigned value in the internal database.

Word Count

Available only when the "32-Bit Registers" checkbox is checked. Check this checkbox to interpret the Modbus "quantity of registers" field as the number of 16-bit words to be transferred. If unchecked, the Modbus "quantity of registers" field is interpreted as the number of 32-bit registers.

1.4.2 Input Register Service Object Settings

Description

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Unit ID

This field is used primarily for intra-system routing purposes. Many Modbus/TCP servers ignore this field.

Start Register

Defines the starting register number (1…65535) for a range of registers associated with this service object.

Number of Registers

Defines the number of registers (1…123 for standard 16-bit accesses or 1…61 for 32-bit accesses) to be targeted by this service object.

Database Address

Defines the database address where the first register of this service object will be mapped. The configuration studio will not allow entry of a starting database address that will cause the service object to run past the end of the database. The highest valid database address, therefore, will depend on the targeted data type, as well as the number of items to be accessed.

Multiplier

The amount that associated network values are scaled by prior to being stored into the database. Network values (read from the device) are divided by the multiplier before being stored into the database.

Read Function

Fixed at “4 (Read Input Registers)”.

Data Type

Available only when the “32-Bit Registers” checkbox is checked. Defaults to standard Modbus “16-Bit Unsigned” when the “32-Bit Registers” checkbox is unchecked. Specifies how the value will be stored in the database for each register (or register pair) in this service object. This defines how many bytes will be allocated, whether the value should be treated as signed or unsigned, and whether the value should be interpreted as an integer or a floating point number. Select the desired data type from this dropdown menu.

1.4.2.1 32-Bit Options Settings

32-Bit Registers

Check this checkbox if the target registers are associated with the Enron/Daniel extension to the Modbus specification, or are represented by 32-bit values.

Floating Point

Available only when the “32-Bit Registers” checkbox is checked. Check this checkbox if the register values are encoded in IEEE 754 floating point format.

Big Endian

Available only when the “32-Bit Registers” checkbox is checked. Check this checkbox if the register values are encoded in big-endian 16-bit word order, i.e. the most significant 16-bit word is before the least significant 16-bit word.

Word-Size Register

Available only when the “32-Bit Registers” checkbox is checked. Check this checkbox if target registers are only 16-bits wide, but two 16-bit registers comprise one 32-bit value. If unchecked, target registers are assumed to be 32-bits wide.

Note that when this checkbox is checked, the “Number of Registers” field indicates the number of 16-bit register pairs. Each register pair will use two register addresses and the selected Data Type will be applicable for the register pair, not each individual register. For example, if Start Register is set to 100, Number of Registers is set to 2, and Data Type is set to 32-bit Unsigned, then register numbers 100…103 will be accessed by the service object, with registers 100 and 101 stored as the first 32-bit unsigned value and registers 102 and 103 stored as the next 32-bit unsigned value in the internal database.

Word Count

Available only when the “32-Bit Registers” checkbox is checked. Check this checkbox to interpret the Modbus “quantity of registers” field as the number of 16-bit words to be transferred. If unchecked, the Modbus “quantity of registers” field is interpreted as the number of 32-bit registers.

1.4.3 Coil Service Object Settings

Description

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Unit ID

This field is used primarily for intra-system routing purposes. Many Modbus/TCP servers ignore this field.

Start Coil

Defines the starting coil number (1…65535) for a range of coils associated with this service object.

Number of Coils

Defines the number of coils (1…1968) to be targeted by this service object.

Database Address

Defines the database address where the first coil of this service object will be mapped. The configuration studio will not allow entry of a starting database address that will cause the service object to run past the end of the database. The highest valid database address, therefore, will depend on the number of items to be accessed.

Read Function

Select whether or not to allow the service object to issue read requests. When enabled, the service object will continuously read from the server unless a pending write exists.

Write Function

Select whether or not to allow the service object to issue write requests, and the function code to use when enabled. When values encompassed by this service object change in the internal database and writes are enabled, these changes will be written down to the targeted server.

Data Type

Fixed at “1 Bit”.

1.4.4 Discrete Input Service Object Settings

Description

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

Unit ID

This field is used primarily for intra-system routing purposes. Many Modbus/TCP servers ignore this field.

Start Input

Defines the starting input (1…65535) for a range of inputs associated with this service object.

Number of Inputs

Defines the number of inputs (1…1968) to be targeted by this service object.

Database Address

Defines the database address where the first input of this service object will be mapped. The configuration studio will not allow entry of a starting database address that will cause the service object to run past the end of the database. The highest valid database address, therefore, will depend on the number of items to be accessed.

Read Function

Fixed at "2 (Read Input Status)".

Data Type

Fixed at "1 Bit".

1.5 Diagnostics Object

Each service object can optionally include a diagnostics object for debugging and diagnostics.

Diagnostics Database Address

Enter the database address at which to store the diagnostics information.