



Communication Processor Driver Documentation Modbus RTU/ASCII Module

Author: Joe Teixeira
Date: 8-Jun-04
Revision: 0.3





Table of Contents

Error! Bookmark not defined.

OVERVIEW.....	1
FUNCTIONALITY	1
Transmission Modes.....	1
Registers	1
Extended Registers	1
Command Set	2
Communication Interface.....	2
Pass-Through.....	2
NETWORK CONFIGURATION	2
Mode	3
MDRTU.....	3
MBRTU	3
MDASCII	3
MBASCII.....	3
Errors	4
MAPPING CONFIGURATION	4
Errors	4
STATUS COUNTERS.....	5
OTHER MESSAGES	6



Overview

This document details the configuration interface for implementing a Pronghorn Modbus module driver installation. This document assumes the reader is familiar with the Liaison's Pronghorn product. Refer to the **Pronghorn Users Guide** for more detailed information.

The main purpose of this module driver, named *modbus*, is to provide connectivity to a Modicon PLC (or compatible device) via the Modbus protocol. This adheres to the protocol specification as detailed in the **Modicon Modbus Protocol Reference Guide**, publication PI-MBUS-300 Rev. E.

Functionality

The module driver's main purpose is to obtain or provide data contained in any of the Modbus *registers*, to or from a compatible device on the Modbus network. The module driver can emulate a Master or Slave device on the network. This is controlled via the configuration parameter *Station* identification. A value of zero automatically identifies it as a Master device while a non-zero values identifies it as Slave device.

Transmission Modes

There are two basic modes available in the Modbus protocol, RTU and ASCII. The module supports both with an appropriate designation of which mode is being used. See "Network Configuration".

Registers

There are four major types of data identified in the modbus protocol, which are all supported by this module driver. In addition there are two extended data types that are widely supported by many devices, which are also supported by this module driver.

	Type	Prefix	Length
Standard	Coil Status	0xxxxx	1 bit
	Input Status	1xxxxx	1 bit
	Status Registers	3xxxxx	16 bits
	Holding Registers	4xxxxx	16 bits
Extended	Long Registers	6xxxxx	32 bits
	Float Registers	7xxxxx	32 bit IEEE Floating Point

The xxxxx represents the address offset in the range of 1 - 65535.

Extended Registers

Since these extended registers are not part of the Modbus protocol special handling is required. Although a prefix of 6xxxxx and 7xxxxx is used, internally these registers are handled as a Holding register (4xxxxx). Therefore they exist within the same register buffer area and care must be taken to not overlap them. Also the same holding registers should be set aside to handle the extended registers on all devices on the network. This will keep all devices in sync and avoid confusion. Keep in mind that holding registers are typically 16 bits therefore the each extended register will occupy two consecutive holding registers. I.e. a float 70101 is really 40101 and 40102.

Command Set

In total the Modbus protocol contains 24 commands (referred to as Function Codes) that can be supported by any device. However this module driver only supports a sub-set of these:

Code	Description
1	Read Coil Status
2	Read Input Status
3	Read Holding Registers
4	Read Input Registers
5	Force Single Coil
6	Preset Single Register
15	Force Multiple Coils
16	Preset Multiple Registers

Communication Interface

The Modbus protocol requires that a serial interface be used for the physical communication layer. This layer is independent of the Modbus application layer. Therefore configuration of the physical layer interface will greatly depend on the hardware being used. Options like RS-232, RS-485, RS-422 etc. will be configured via jumpers. Consult the hardware documentation for configuration options of the hardware being used.

Pass-Through

This is the Modbus router or pass-through functionality. In this mode the module driver does not parse any of the Modbus commands, it merely passes the entire packet to the peer Modbus Network (i.e. Modbus TCP or Modbus Plus). In slave mode (unsolicited), a request from the master contains a station number which has a corresponding mapping. A lookup, in the mapping table, is done for the station number and the packet is sent to the corresponding mapping on the peer network. Each port (for point-to-point connections) or station (in multidrop configuration), has one map in each direction (read,write) as required

Note: This mode is automatically selected if the either of “up” (Unsolicited Packet) or “wp” (Write Packet) poll command types are used. Please refer to section 6.4 in the configuration manual.

Network Configuration

The configuration of this module and its network interface is accomplished through the network comma separated values file (.csv) using software such as Excel or gnumeric. Some of the parameters are standard parameters used to configure the serial interface while others are particular to this module. Most parameters have a default value that will be used if the parameter does not exist. Additionally the parameters can be listed in any order.

Name	Valid Range	Default	Description
Station	0 - 255	0	The device (or Node) number that identifies the module on the Modbus network.
Mode	MDRTU, MBRTU, MDASCII or MBASCII	MDRTU	Identifies the different variants of the Modbus protocol being supported
Reg_Start	0 - 2	1	This identifies the first address offset in each of the register types. Typically this is set to 1. However some Modbus variants use a different starting point.
Long_Start	0 - 65535	Calculated	This is the starting Holding register where Long registers will start at
Long_Len	0 - 65535	Calculated	The number for Holding registers being used for Long registers
Float_Start	0 - 65535	Calculated	This is the starting Holding register where Float registers will start at
Float_Len	0 - 65535	Calculated	The number for Holding registers being used for Float registers
Retries	0 - 9	3	This the number of times a transaction will re-transmitted upon the detection of a timeout
Timeout	0 - 360000	1000	The number of milliseconds to wait for a response

Name	Valid Range	Default	Description
Com_Port	1 - 16	1	The serial port interface to be used
Baud	50 - 460800 ⁱ	9600	The serial line speed being used.
Stop_Bits	1 or 2	1	The number of stop-bits per transmitted character
Data_Bits	5 - 8	8	The number of data-bits per transmitted character
Parity	None, Even, Odd	None	The type of parity checking performed
Tx_Delay	0 - 65535	0	The number milliseconds to wait before transmitting
Handshake	Yes or No	No	Specifies if hardware (CTS/RTS) handshaking is to be used
IRQ	0 - 16 ⁱⁱ	0	The IRQ used to access the serial hardware.
IO_Port	0 - 0xffff ⁱⁱⁱ	0	The port address used to access the serial hardware.

Mode

This parameter is used to specify which different variant of the Modbus protocol to emulate. Currently there are two different modes.

Mode	Description
MDRTU	Modicon RTU
MBRTU	Gould Modbus RTU
MDASCII	Modicon ASCII
MBASCII	Gould Modbus ASCII

MDRTU

This is the Modicon RTU emulation. Each extended register, internally, are treated as two Holding registers. Therefore, a request for a single extended register results in a packet request for two holding registers. In addition, each byte, within each 16-bit word, is swapped.

MBRTU

This is the Gould Modbus RTU emulation. Each extended register, internally, are treated as two Holding registers. However, a request for a single extended register results in a packet request for only one register but two 16-bit words are returned. In addition, each byte, within each 16-bit word, is swapped and each adjacent 16-bit word is swapped.

MDASCII

This is the same as MDRTU accept that the transmission of data is done using displayable ASCII characters. Typically using only 7 data bits instead of 8.

MBASCII

This is the same as MBRTU accept that the transmission of data is done using displayable ASCII characters. Typically using only 7 data bits instead of 8.

ⁱValid values are: 50, 75, 134, 200, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 56700, 115200, 230400, 460800. This depends on support of the hardware being used.

ⁱⁱA value of zero specifies to use default values for the given serial port being used. Typically COM1=IRQ3, COM2=IRQ4, COM3=IRQ3, COM4=IRQ4.

ⁱⁱⁱA value of zero specifies to use default values for the given serial port being used. Typically COM1=0x3e0, COM2=0x2e0, COM3=0x3f0, COM4=0x2f0.

Errors

If any of the validation, for the network configuration parameters, fails a critical error message will be placed in the logs. In addition the module driver will not start. The following is a list of critical message that might be generated.

<i>Msg ID</i>	<i>Mnemonic</i>	<i>Description</i>
1023	CRT_BAD_STATION	Invalid station number (%d) must be less than %d - The specified station number greater than the maximum of 255
95	CRT_BAD_COMPORT	Invalid COM port (%d) Must be between 1 and %d - The communication port specified is not in the proper range. Must be numeric in the range of 1 - 16
96	CRT_BAD_BAUD	Invalid BAUD rate (%d) - The communication speed specified is not one of the predefined values of: 50, 75, 134, 200, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 56700, 115200, 230400, 460800
97	CRT_BAD_STOPBITS	Invalid number of stop bits (%d) must be 1 or 2 - The number of stop-bits specified is not one of 1 or 2. Typically RTU modes use 1 stop bit
98	CRT_BAD_PARITY	Invalid parity (%s) must be 'None', 'Even' or 'Odd' - The parity check specified is not one of the predefined mnemonics. Typically RTU modes use 'Even' parity
99	CRT_DATABITS	Invalid number of data bits (%d) must be between 5 and 8 - The number of data-bits specified is out of range. Typically RTU modes use 8 data bits
100	CRT_BAD_TIMEOUT	Invalid timeout value (%d) must be less than %d ms - The timeout value specified is too large, it should be less than 360000 milliseconds

Mapping Configuration

The configuration of the module mappings is accomplished through the network comma separated values file (.csv) using software such as Excel or gnumeric. The parameters listed only pertain to this module. Refer to the **Pronghorn Users Guide** for a list of the other common parameters used to configure a mapping. Most parameters have a default value that will be used if the parameter does not exist. Additionally the parameters can be listed in any order.

<i>Name</i>	<i>Valid Range</i>	<i>Default</i>	<i>Description</i>
Station	0 - 255	0	The device (or Node) number that identifies the slave to be accessed. This is only used when the module is emulating a master station.
Address	Refer to Register description	0	Identifies the data address to access, within the slave device.

Errors

If any of the validation, for the mapping configuration parameters, fails an error message will be placed in the logs. In addition that particular mapping item will not be added to the internal mapping list. However the module continues with any remaining mappings.

<i>Msg ID</i>	<i>Mnemonic</i>	<i>Description</i>
1001	ERR_ADDRESS_RANGE	Address is out of range (addr=%c%04d, len=%d) - The specified Register offset value is out of range (in combination with the mapping length).
1002	ERR_BAD_ADDR_TYPE	The address type is not valid (0x%02x) - The specified address does not point to a valid Register type.
1009	ERR_WRITE_ON_INPUTS	Action not allowed (Writing to 'Input Status' or 'Input Registers') - An attempt to perform a write operation on read-only Register address

Status Counters

This module driver maintains a set of status counter for recording the operational status of the module and the network.

<i>Msg ID</i>	<i>Mnemonic</i>	<i>Description</i>
1013	CNT_MSG_SENT	Number of Messages transmitted - Keeps track of the number of transactions transmitted from this module.
1014	CNT_MSG_RECV	Number of Messages received - Keeps track of the number of transaction received for this module.
1015	CNT_CMD_SENT	Number of commands transmitted - Keeps track of the number of command requests transmitted (Master mode)
1016	CNT_CMD_RECV	Number of commands received - Keeps track of the number of command requests received (Slave mode)
1017	CNT_REPLY_SENT	Number of replies transmitted - Keeps track of the number of response packets transmitted (Slave mode)
1018	CNT_REPLY_RECV	Number of replies received - Keeps track of the number of response packets received (Master mode)
1019	CNT_ERR_SENT	Number of errors sent - Keeps track of the number of error response packets transmitted (Slave Mode)
1020	CNT_ERR_RECV	Number of errors received - Keeps track of the number of error response packets received (Master mode)
1021	CNT_RETRIES	Number of retries - Keeps track of the number of times a re-transmit had to be performed
1022	CNT_REPLY_TIMEOUTS	Response timeouts - Keeps track of the number of times no response was received from a slave
1026	CRD_CHAR_SENT	Number of characters transmitted - The number of characters sent out the serial port (maintained by the hardware interface).
1027	CRD_CHAR_RECV	Number of characters received - The number of characters received from the serial port (maintained by the hardware interface).

Other Messages

In the normal operation of this module certain types of message may appear in the logs. The messages specific to this module are listed below. Refer to the *Pronghorn Users Guide* for a list of the common messages that can be seen in the logs.

<i>Msg ID</i>	<i>Mnemonic</i>	<i>Description</i>
75	ERR_WRONG_RESPONSE	Unexpected Response 0x%x s/b 0x%x - Received a response from an unexpected station number or - Received a response for an unexpected function code or - Received a response with incorrect register offset
59	ERR_BAD_READ_LENGTH	Read length not what expected (actual=%d, expected=%d) - A response packet was received with an unexpected data length
60	ERR_BAD_WRITE_LENGTH	Write length not what expected (actual=%d, expected=%d) - A response packet was received with an unexpected data length
73	ERR_NOT_HANDLED	Command/Type 0x%x not handled in this context - A function code is not supported
81	ERR_INP_TIMEOUT	Timed out waiting for input - No response was received, in the allotted timeout/retires
1007	ERR_BAD_MOD_RESPONSE	Response with error (stn=%d, func=%d, error=%d) - A error response packet was received
1008	ERR_BAD_CRC	Response with wrong CRC (actual=0x%02x%02x, expected=0x%02x%02x) - A packet was received with a CRC that differs from the calculated CRC
1010	ERR_UNREAD	Error processing unsolicited Read (func=%d, addr=%c%04d, len=%d) - An unexpected error occurred while processing a read request (Slave mode)
1011	ERR_UNWRITE	Error processing unsolicited Write (func=%d, addr=%c%04d, len=%d) - An unexpected error occurred while processing a write request (Slave mode)