



Pronghorn

MBPlus Module

Author: Joe Teixeira

Date: 28-Apr-04

Revision: 0.2

Table of Contents

OVERVIEW	1
FUNCTIONALITY	1
NETWORK CONFIGURATION	2
MAPPING CONFIGURATION	3
STATUS COUNTERS	4
OTHER MESSAGES	5

Overview

This document details the configuration interface for implementing a Pronghorn Modbus PLUS 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 *mbplus*, is to provide connectivity to a Modicon PLC (or compatible device) via the Modbus PLUS 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 Plus network. The module driver can emulate a Master and Slave device on the network at the same time (a peer-to-peer network).

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.

	<i>Type</i>	<i>Prefix</i>	<i>Length</i>
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:

<i>Code</i>	<i>Description</i>
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 Plus protocol uses the Modicon PCI85 interface card for the physical communication layer. The Modbus PLUS protocol uses a token passing architecture as its main transport method, allowing each device an opportunity to transmit data. Consult the hardware documentation for configuration options of the PCI85 interface hardware. The interface card has no jumpers or settings; therefore no configuration of the interface card/hardware is required.

Pass-Through

This is the Modbus 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 Serial). In slave mode (unsolicited) one of the route codes identifies the peer device that the packet should be sent to. In the route code, the byte immediately following the S5's node number identifies the station for which the packet is intended. 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.

Note: This mode is automatically selected if the either of “*up*” (Unsolicited Packet) or “*wp*” (Write Packet) poll command types are used.

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.

<i>Name</i>	<i>Valid Range</i>	<i>Default</i>	<i>Description</i>
Station	1 - 64	0	The device (or Node) number that identifies the PCI85 interface card on the Modbus Plus network.
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. I.e. zero
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
Card	1 - 4	1	The PCI85 interface card being used. Typically only 1 interface card is installed.

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>
8025	CRT_BAD_STATION	Invalid station number (%d) must be less than %d - The specified station address is out of range
101	CRT_BAD_CARDNO	Invalid card number (%d) Must be between 1 and %d - The specified card number must be 1 or 2

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>
Route	1 - 64	0	The device (or Node) route code that identifies the slave to be accessed. This can be up to 5 different route code using a '.' (dot) notation to separate routes. The first number should always be the destination station number.
Station	1 – 255	0	Used for Pass-through mode, this is used as the station lookup to identify the peer Modbus network mapping.
Address	Refer to Register description	0	Identifies the data address to access, within the slave device. Ignored for pass-through mode.

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>
8001	ERR_NO_ROUTES	Item %d, A valid route must be specified (format #.#.#.#) - The station/route was not specified or entered incorrectly.
8002	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).
8003	ERR_BAD_ADDR_TYPE	The address type is not valid (0x%02x) - The specified address does not point to a valid Register type.
8010	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 counters for recording the operational status of the module and the network.

<i>Msg ID</i>	<i>Mnemonic</i>	<i>Description</i>
8015	CNT_MSG_SENT	Number of Messages transmitted - Keeps track of the number of transactions transmitted from this module.
8016	CNT_MSG_RECV	Number of Messages received - Keeps track of the number of transaction received for this module.
8017	CNT_CMD_SENT	Number of commands transmitted - Keeps track of the number of command requests transmitted (Master mode)
8018	CNT_CMD_RECV	Number of commands received - Keeps track of the number of command requests received (Slave mode)
8019	CNT_REPLY_SENT	Number of replies transmitted - Keeps track of the number of response packets transmitted (Slave mode)
8020	CNT_REPLY_RECV	Number of replies received - Keeps track of the number of response packets received (Master mode)
8021	CNT_ERR_SENT	Number of errors sent - Keeps track of the number of error response packets transmitted (Slave Mode)
8022	CNT_ERR_RECV	Number of errors received - Keeps track of the number of error response packets received (Master mode)
8023	CNT_RETRIES	Number of retries - Keeps track of the number of times a re-transmit had to be performed
8024	CNT_REPLY_TIMEOUTS	Response timeouts - Keeps track of the number of times no response was received from a slave
8026	CRD_NODE_TYPE	Number of characters transmitted - The number of characters sent out the serial port (maintained by the hardware interface).
8027	CRD_CARD_VERSION	Number of characters received - The number of characters received from the serial port (maintained by the hardware interface).
8028	CRD_NODE_NUMBER	The node type of this Modbus Plus card (always = 3)
8029	CRD_MAC_STATE	MAC State
8030	CRD_NODE_STATUS	Node Status
8031	CRD_TOKEN_PASS	Number of token passes
8032	CRD_TOKEN_TIME	Token rotation time (ms)
8033	CRD_PRE_TX_DEFERS	Pre-transmit deferral errors
8034	CRD_RX_DMA_OVERRUNS	Receive DMA overrun errors
8035	CRD_REPEAT_CM_RX	Repeat command received counter
8036	CRD_NODE_ABSENT	Destination Node absent counter
8037	CRD_RX_COLLISIONS	Receiver collisions errors (or Cable A framing errors)
8038	CRD_RX_ALIGNMENTS	Receiver alignment errors (or Cable B framing errors)
8039	CRD_RX_CRC_ERRORS	Receiver CRC errors
8040	CRD_BAD_PACKET_LEN	Bad packet length errors
8041	CRD_BAD_LINK_ADDR	Bad link address errors
8042	CRD_TX_DMA_UNDERRUNS	Transmit buffer DMA under-run errors
8043	CRD_INT_BAD_PACKET_LEN	Bad internal packet length errors

<i>Msg ID</i>	<i>Mnemonic</i>	<i>Description</i>
8044	CRD_BAD_MAC_FUNCTION	Bad MAC function code errors
8045	CRD_COMM_RETRIES	Communication retry errors
8046	CRD_COMM_FAILED	Communication failed errors
8047	CRD_GOOD_RX_PACKETS	Good receive packets
8048	CRD_NO_RESP_RX	No response received counter
8049	CRD_EXCEPTION_RESP	Exception response received counter
8050	CRD_PATH_ERRORS	Unexpected path errors
8051	CRD_UNEXPECTED_RESP	Unexpected response errors
8052	CRD_FORGOT_TRANS	Forgotten transaction errors
8053	CRD_ACTIVE_STN_1TO32	Active stations (1 - 32)
8054	CRD_ACTIVE_STN_33TO64	Active stations (33 - 64)
8055	CRD_TOKEN_STN_1TO32	Token stations (1 - 32)
8056	CRD_TOKEN_STN_33TO64	Token stations (33 - 64)
8057	CRD_GLOBAL_STN_1TO32	Global data stations (1 - 32)
8058	CRD_GLOBAL_STN_33TO64	Global data stations (33 - 64)
8059	CRD_RX_BUF_1TO32	RX buffers in use (1 - 32)
8060	CRD_RX_BUF_33TO34	RX buffers in use (33 - 34)
8061	CRD_STN_MANAGMENT	Station management command processed counter
8062	CRD_DM_PATH_1_TX	Data master output path 1 TX count
8063	CRD_DM_PATH_2_TX	Data master output path 2 TX count
8064	CRD_DM_PATH_3_TX	Data master output path 3 TX count
8065	CRD_DM_PATH_4_TX	Data master output path 4 TX count
8066	CRD_DM_PATH_5_TX	Data master output path 5 TX count
8067	CRD_DM_PATH_6_TX	Data master output path 6 TX count
8068	CRD_DM_PATH_7_TX	Data master output path 7 TX count
8069	CRD_DM_PATH_8_TX	Data master output path 8 TX count
8070	CRD_DS_PATH_1_RX	Data slave input path 1 TX count
8071	CRD_DS_PATH_2_RX	Data slave input path 2 TX count
8072	CRD_DS_PATH_3_RX	Data slave input path 3 TX count
8073	CRD_DS_PATH_4_RX	Data slave input path 4 TX count
8074	CRD_DS_PATH_5_RX	Data slave input path 5 TX count
8075	CRD_DS_PATH_6_RX	Data slave input path 6 TX count
8076	CRD_DS_PATH_7_RX	Data slave input path 7 TX count
8077	CRD_DS_PATH_8_RX	Data slave input path 8 TX count

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>
111	ERR_NOT_OPEN	Resource %s not open - The specified IP/Hostname is not yet opened or accessible
124	ERR_OBJ_NOT_FOUND	Object PathList[%#x] not found - The original message was not found in the queue
124	ERR_OBJ_NOT_FOUND	Object Pending[%#x] not found - The original message was not found in the queue
52	ERR_NOT_FOUND	Did Not Find 'Stn xref' in ' route' - In pass-through mode, the incoming route did not contain a non-zero value following the S5's node number
73	ERR_NOT_HANDLED	Command/Type 0x%x not handled in this context - A function code is not supported
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)

<i>Msg ID</i>	<i>Mnemonic</i>	<i>Description</i>
		- 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
6508	ERR_BAD_MOD_RESPONSE	Response with error (stn=%d, func=%d, error=%d) - A error response packet was received
113	ERR_CHECKSUM_FAILED	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
6511	ERR_UNREAD	Error processing unsolicited Read (func=%d, addr=%c%04d, len=%d) - An unexpected error occurred while processing a read request (Slave mode)
6512	ERR_UNWRITE	Error processing unsolicited Write (func=%d, addr=%c%04d, len=%d) - An unexpected error occurred while processing a write request (Slave mode)
81	PH_ERR_INP_TIMEOUT	Timed out waiting for input - The time was expired waiting for a response