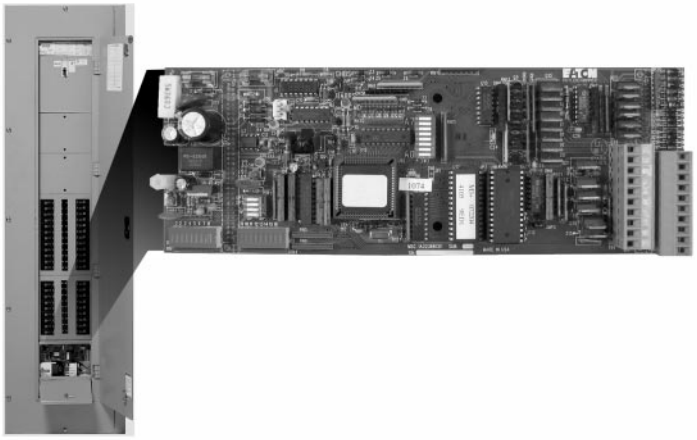
EATON | Cutler-Hammer

Pow-R-Command™ PRC100
MOD100 Modbus® Controller
Installation and Programming Reference

Instruction Leaflet IL01412015E

Effective September 2008



Contents

⚠ NOTICE

THE INSTRUCTIONS AND PRESCRIBED PROCEDURES IN THIS PROGRAMMING AND USE MANUAL DO NOT PURPORT TO COVER ALL POSSIBLE CONTINGENCIES THAT MAY ARISE DURING INSTALLATION, PROGRAMMING, OPERATION, OR MAINTENANCE, OR ALL DETAILS AND VARIATIONS OF THE SYSTEM. IF FURTHER INFORMATION IS REQUIRED REGARDING A PARTICULAR INSTALLATION, PROGRAMMING, OPERATION, OR MAINTENANCE, PLEASE CONTACT YOUR LOCAL EATON REPRESENTATIVE.

Introduction

This document describes the hardware configuration of the Pow-R-Command PRC100MOD Modbus System Controller (MSC). The MSC can be placed into any lighting control panel designed to accept the PRC100USC. This design similarity allows the MSC to be used in a Cutler-Hammer® Smart Panelboard, Relay Panel and Control Cabinet, providing the same flexibility of application. The MSC however, does NOT interface with other Pow-R-Command controllers. The MSC conforms to the "Modbus Over Serial Specification" whereas the Pow-R-Command PRC100 standard products follow a Cutler-Hammer proprietary serial network.

The MSC provides the following features:

- Easy integration to vendor-specific control systems through common, off-the-shelf, Modbus Gateway products.
- Sixteen internal zones of control (Groups).
- Switch inputs providing up to 16 zones of control.
- Switches are configurable to accept maintained, momentary and paired inputs.
- RS-485 compliant communication network.
- Support for both Modbus ASCII and RTU implementations.
- Diagnostics port for offline programming of in-system controllers.

The MSC follows the Modbus Specification for memory partitioning into regions of registers. Each region is selected based on the type of data stored and function. The regions supported by this release of the MSC are the 0x Coils, 1x Discrete Inputs, and 4x Holding registers. In the MSC, these regions are divided primarily as command and control (0x), status (1x), and configuration (4x). To query and alter these regions, the MSC supports a subset of the full complement of Modbus function codes.

The following sections describe the regions (registers) in detail. Where appropriate, the interaction between the regional values is discussed and finally some operation examples are given.

Hardware Configuration

The MSC can be installed into any panel or cabinet designed to accept the Pow-R-Command USC100. For installation of the controller into the panel or cabinet, please refer to the PRC100 Installation Guide (IL01412003E). Also, refer to the same document for Power and Network wiring assistance.

Figure 1 shows a graphic representation of the MSC Circuit Board. There are labeled areas on the graphic representing the topics of discussion.

Address Switch

The MSC requires a unique address for communication. The controller is addressed by setting eight switches. The switches represent the binary values from 0 to 255. The Modbus protocol specification explicitly excludes addressed 0 and 248 through 255. In other words, the address of the controller must be set from 1 to 247.

The address is set by summing the weighted switch values according to the following table.

SWITCH NUMBER	ADDER VALUE
8	128
7	64
6	32
5	16
4	8
3	4
2	2
1	1

For example, to set address 75; we first subtract the largest number from the table that will result with a positive remainder. In this example 64; so, turn switch 7 ON. The remainder is 75 – 64, or 11. Again, subtract the largest possible number from the table, in this case that would be 8. Turn switch 4 ON and the remainder is 11 – 8, or 3. It's now plain to see that switches 1 and 2 add up to 3, so switches 1 and 2 need to be ON as well. Therefore, address 75 is switches 7, 4, 2 and 1 ON, all other switches should be OFF. Setting the address of the controller to 75 should look like **Figure 2**.

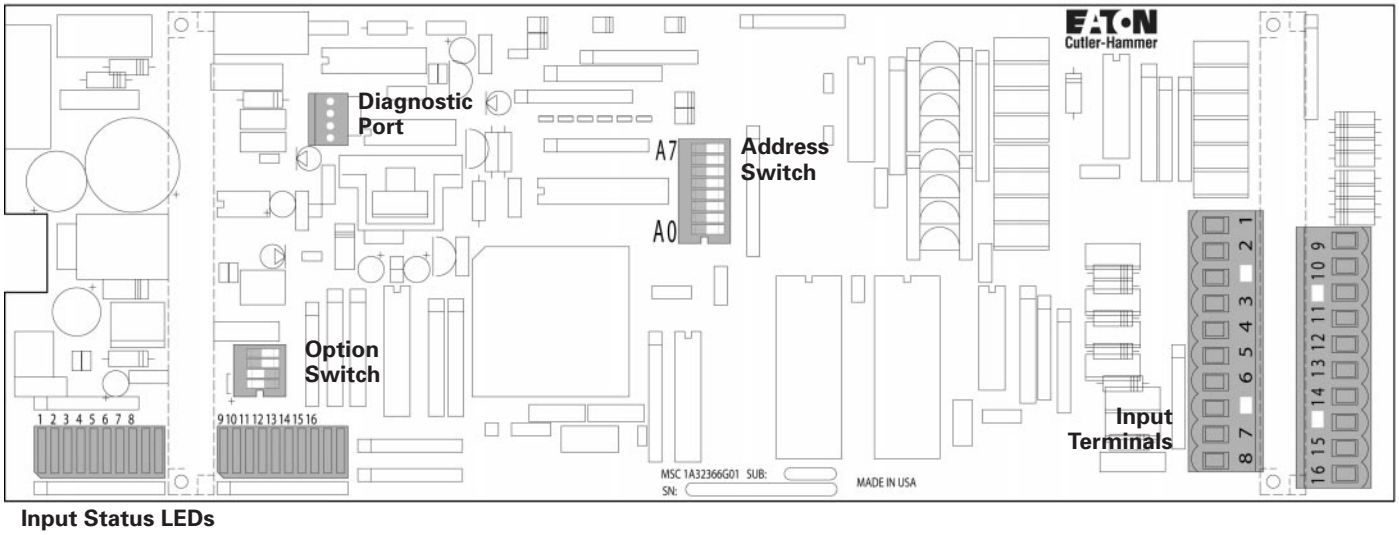


FIGURE 1. LOCATING THE DISCUSSED COMPONENTS

Option Switch

The Diagnostic and Options Switches shown in **Figure 1** are primarily for local configuration use. **Figure 3** shows the two bottom switches labeled "NET" used for disconnecting the controller from the building network. This will allow a local configuration tool to be connected to the Diagnostic/Programming Port on the controller without interfering with the building control system network. REMEMBER to turn the switches to the ON position when local communication is completed. Failure to turn ON the NET switches will require a return trip to enable communication with the host system.

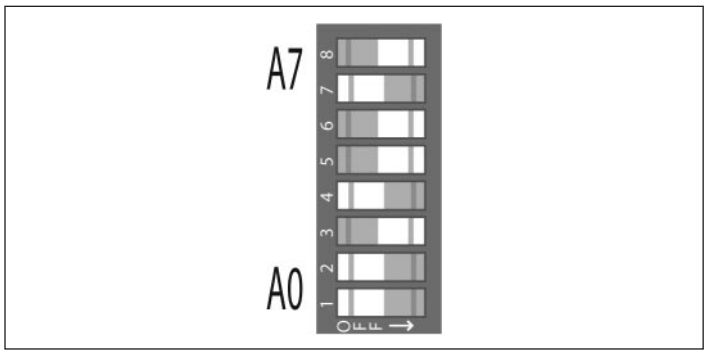


FIGURE 2. SETTING ADDRESS 75

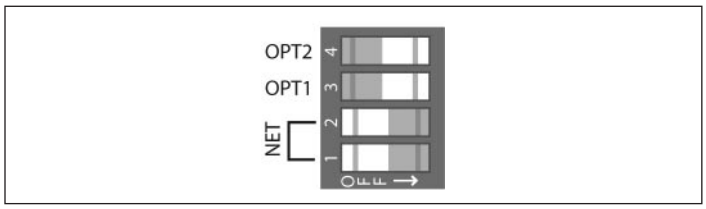


FIGURE 3. OPTIONS AND NET SWITCHES

The controller communicates with both ports at 9600 baud with 8 data bits, No Parity, and 1 Stop bit. Changing the "NET" switches has no effect on the communications speed and format.

The remaining two switches labeled "ASCII" and "OPT1" have no function in this application and are reserved for future use.

Diagnostic Port

The Diagnostic/Programming Port shown in **Figure 1** accepts the Cutler-Hammer pendant adaptor cable. This is a 4-pin device with n integral RS-485 converter providing direct connection between a laptop PC serial port and the MSC. Third-part RS-485 converters can also be used. Consult the Appendix for information regarding these adaptors.

Input Terminals

Switches are connected to the MSC at the two terminal blocks located along the right side of the controller board. These terminal blocks accept dry-contact closures only. The connections are such that a switch position is wired to the ground terminal (white block). The controller recognizes the input as active when a closure to ground is detected (active low) however, internal logic is provided to invert this action (see later sections). Whenever a switch closure occurs, the associated Switch Status LED will illuminate. The Switch Status LEDs are located along the lower-left edge of the MSC. These are hard-wired to reflect a closure and cannot be "inverted" through internal logic. **Figure 4** shows a few examples of typical input wiring.

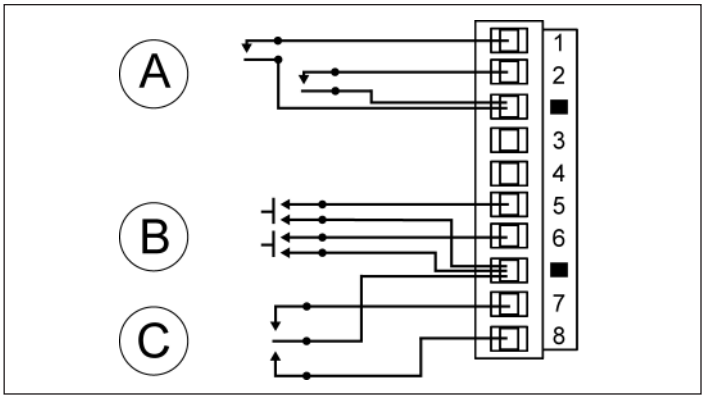


FIGURE 4. INPUT WIRING EXAMPLES

The example at "A" in **Figure 4** shows a typical application of a normally-open maintained closure switch. "B" shows a pair of normally-open, momentary push-button switches connected to the controller, while Example "C" shows a momentary 3-wire, center-off switch. Each of the above examples need to have the inputs configured in the controller database either by using the Cutler-Hammer Modbus Configuration Software, or by using the standard Modbus register write commands.

Software Configuration

The remaining topics deal with internal organization and configuration of the Cutler-Hammer MSC. It should be noted that the MSC conforms to the Modbus Application Protocol Specification V1.1. This document makes no effort to explain the protocol. Though some examples are given for sending packets to the controller, please refer to the aforementioned document for interpreting the responses.

Modbus Register Assignments

Below begins a discussion of the MSC Modbus register assignments. The accompanying table details the register assignments, and may be helpful for reference during the following discussions.

Discrete Coils [0x: Registers]

The 0x region of the controller is primarily for direct control of the controlled loads in the panel. There are three Modbus commands allowing interaction with these controls from the Operators Console.

RD_0X_MULTI (Code 1)

Reads one or more registers from the 0x memory region. This region contains the coil control points used to turn on and off the attached controllable circuit breakers.

WR_0X_SINGLE (Code 5)

This command is used to set the state of individual coils in the 0x region of the controller. The register written can have varying effects on the controlled devices.

WR_0X_MULTI (Code 15)

This command can be used to set the state of multiple coils with one write operation. Typically this command is used in lieu of streaming multiple WR_0X_SINGLE commands.

For each circuit breaker, there are actually three sets of device flags for direct control. The three sets interact to allow remote override of local control. This is accomplished by manipulation of the PEER and PRIORITY coils.

Peer Coils

Forcing a single PEER coil will affect the associated circuit. Forcing a PEER coil will only affect the circuit until some other equal (PEER) event occurs such as a switch input or other force command. Hence the PEER name, implying that all actions at this level are equal. Writing a one to a PEER coil will turn the associated device on, while a zero will turn off the device.

Priority Coils

To provide specific ability to override internal or local control, the PRIORITY coils were added. Write (force) a single PRIORITY coil to inhibit any internal switch event, or external PEER event from affecting this coil. While a circuit is commanded via a PRIORITY coil, it is important to mention that the underlying PEER coils will still change according to local control events (switch) but will have no effect on the output.

When a PRIORITY coil is written (either on or off) the associated SERVICE coil is also written with a one. The SERVICE coils act as a flag to keep the PEER coils from affecting the attached load while it is OUT-OF-SERVICE (SERVICE = 1). For example: it is desired to prevent circuit 5 from being affected by any switch or group controls. Forcing coil 68 (Priority #5) to one will turn on the circuit. It will also set coil 132 (Service #5) to one flagging the load OUT-OF-SERVICE. As long as this circuit remains in this state, the PEER coils can toggle and change without any effect on the output.

Service Coils

To place a circuit back IN-SERVICE, it is only necessary to write access (1 or 0) the SERVICE coil (132 in this example) to restore the circuit to the current PEER state.

From this discussion, it may seem unclear exactly how to tell the state of the load. Remember that these coils are for command and control of the loads. Interrogating the actual state is discussed in the next section.

Group Coils

In addition to single circuit commands, it is also possible to force multiple circuits with a single coil. The 16 GROUP coils can be written to force multiple circuits on or off from a single coil. These flags are the same flags used by the switch inputs to control assigned loads. For example, if breakers 1, 10 and 22 are assigned to Group A, then changing the state of the Group A flag will also change the state of circuits 1, 10 and 22 provided they are IN-SERVICE.

There are no Group PRIORITY or SERVICE flags. If it is desirable, to place several circuits OUT-OF-SERVICE then it is necessary to use multiple PRIORITY force commands.

Input Changed Monitors

The INPUT CHANGED MONITORS are included exclusively for integration interface systems. Whenever input action occurs (e.g., press, release, on, off) the associated bit in this field will toggle. This function allows for polling of inputs. If it is desired to detect a change by seeing a one in the bit position, then after reading the action and seeing a one, clear the associated bit (zero). When the input is activated again, the corresponding bit will be toggled to a one. If the bit is not reset after reading, it should not be assumed that zero is open (off) and one is closed (on). These coils are included merely as an integrators tool; how they are implemented and processed is the responsibility of the Modbus Master.

⚠ NOTICE

THESE COILS DO NOT FOLLOW NORMAL MODBUS CONVENTION. BEST RESULTS ARE OBTAINED BY READING THE PAIR OF COILS AND FORMING AN INTEGER. THEN BIT 0 OF THE INTEGER WILL CORRESPOND TO SWITCH 1 AND BIT 15 TO SWITCH 16.

⚠ APPLICATION NOTE

SOME BAS SYSTEMS MAY WANT TO POLL THE INPUTS AND THEN USE THE WRITE MULTI COIL COMMANDS TO SET THE STATE OF THE CIRCUITS ACCORDINGLY. IT IS POSSIBLE TO USE THE GROUP FLAGS TO DETERMINE WHEN A CHANGE IS MADE FROM THE INPUTS. CONFIGURING THE INPUT AS MOMENTARY OR MAINTAINED CHANGE (DISCUSSED LATER) THE BAS SYSTEM COULD READ THE GROUP FLAG, AND THEN RESET IT TO ZERO TO DETECT THE NEXT ACTIVATION. IF THIS METHOD IS USED, THERE CANNOT BE ANY CIRCUITS ASSIGNED TO THE MONITORED GROUP FLAGS.

It is better to use the multiple coil versions of the following commands where possible. This is due to the fact that a single command can turn multiple breakers on or off quicker than spooling individual coil commands. When spooling individual commands, it is important to pace the commands by allowing a minimum of 200 milliseconds between commands. This limitation is due to the fact that a breaker attempting a retry cycle will cause communications disruption. The MSC will acknowledge the receipt of the command and then proceed to process the command. There is currently no way to send a device busy notification if the breaker enters a retry cycle; conversely waiting until the breaker completes a full cycle, including retries, would violate the Modbus response timing requirements.

Below are some examples of writing coils. Assume the controller address is 0x55 (85 decimal).

Turn on circuit one:

- 55: address
- 05: function WR_0X_SINGLE (force a single coil)
- 0000: register address for PEER Coil Circuit 1
- FF00: format to turn ON a coil
- XX: two-digit LRC
- "55 05 0000 FF00 XX"

The result of issuing the above command will be to turn on circuit 1. The result of reading the PEER, PRIORITY and SERVICE flags would most likely be:

- PEER = 1
- PRIORITY = 0
- SERVICE = 0

This next example will turn the circuit off and place it OUT-OF-SERVICE:

- 55: address
- 05: function WR_0X_SINGLE
- 0040: coil 64 (the first PRIORITY coil)
- 0000: format of coil off command
- XX: two-digit LRC
- "55 05 0000 0000 XX"

The result of issuing this command following the previous example will be:

- PEER = 1
- PRIORITY = 0
- SERVICE = 1
  - The load if now OFF and OUT-OF-SERVICE!
  - Use the WR_0X_SINGLE to restore the load to its PEER state.
- 55: address
- 05:
- 0080: coil 128 (the first SERVICE coil)
- 0000 or FF00: state does not matter
- XX: the LRC
- "55 05 0080 FF00 XX"

Notice this command only needs to write to the SERVICE coil. The actual state of the command is ignored. The result of issuing this command following the previous examples would be:

- PEER = 1
- PRIORITY = 0
- SERVICE = 0

This example turns the load back ON according to the PEER state and places the load back in service.

Discrete Inputs [1x: Registers]

As discussed in the previous section, 0x coils are used to interact with the controller. The 1x coils are where the real status of actions can be determined. By looking at the 0x PEER and PRIORITY coils, it is not possible to accurately determine the circuit's state. It can tell the desired state but the actual state can only be determined by interrogating the 1x registers.

RD_1X_MULTI (Code 2)

This is the only command supported for reading the Discrete Input Registers of the Modbus Controller.

Like the 0x space, the 1x registers are grouped per each coil. There are three such coils for each circuit. Unlike the 0x PEER, PRIORITY and SERVICE coils however, the 1x coils act independent of each other recording different state data for each coil. The 1x coils are, by definition, read-only. The following discusses each coil group in turn.

Feedback Coils

Represent the true state of the attached device (assuming good working order of the device). When a circuit is commanded on via a 0x coil, these coils will change to reflect the current image of the entire 48 (max.) circuits. A one in the feedback status coil indicates a closure, or the circuit breaker is on.

Command Coils

Reflect the last commanded state of the circuit breaker. This will reflect the last PEER or PRIORITY command and thus is the definitive command result.

Alarm Coils

Represent a fault between the FEEDBACK and COMMAND status. Alarms can be present when a defective circuit breaker fails to change to the commanded state. In addition, a circuit breaker trip will change the FEEDBACK from on to off and thus will not match the commanded state hence an alarm. Manually operating the circuit breaker's override handle can generate an alarm as well. In simplest terms, when the commanded state does not match the feedback-state, the alarm status for that breaker will be set to one.

Switch Input Log

There is additional state monitoring associated with the switch inputs. The Switch Input Logs show the debounced raw state of the input activity. While they have no practical use in operation, they are provided as a possible troubleshooting diagnostic point.

Reserved
Reserved
Condition Code

This bit-mapped register is intended to provide fast-status of the controller's general state. There are currently only two bits defined in this register. Bit-0 will tell if there are any OUT-OF-SERVICE (overrides) present in this panel. Bit-1 will reflect any active Alarm coils. Reading the Condition Code can be a fast means to determine if further coil reads are required. Both of these bits are active high, meaning that a one will indicate that an override or alarm is active. These bits are self-clearing when the source is resolved.

Schedule Active

These 25 one-bit registers reflect the active status of the 25 internal time schedules. When a schedule indicates active, that implies the time-of-day is between the From and Until times inclusively. It should be noted that one-shot (event) schedules do not have an active state as they are instantaneous (only a From time) rather than range functions.

Input Registers [3x: Registers]

There are two functions represented in the 3x Register space. There is only one command that can be issued to interrogate these registers.

RD_3X_MULTI (Code 4)

This command can be used to retrieve one or more Input Register values as described below. This is the only command since this region is, by definition, read-only.

Override Timers

Sixteen 16-bit registers represent any remaining time of a timed override set from the associated input. The values set in these counters are programmed in the input attributes (see below). These timers decrement in one-minute intervals and can have a maximum value of 1440 minutes (24 hours). If a zero is present in these registers, it means there is no timing activity set, or the timing activity is complete. These registers will be refreshed from the input whenever the input is activated.

Load Wink Timers

These registers are comprised of two 8-bit values. The first byte read contains the off timer. This timer will typically be zero as the time is only 1 second. The second represents the remaining egress (on time), in minutes * 4. The lower 2 bits are used for internal state machine control.

The information presented in the 3x Register space has limited function capability. Certainly, a Master Controller can monitor these timers and act accordingly. Additionally, these timers can be used for troubleshooting possible unexpected programming results.

Holding Registers [4x: Registers]

The 4x, or holding, registers are used to maintain panel configuration data. These registers are used to assign attributes to loads, inputs, groups and schedules, as well as the general panel configuration. In an attempt to simplify the interface to the MSC, the 4x registers are arranged in a record format. For this reason, items may span more than one register, or conversely, there may be more than one item per register. An example of the former is a Name field; many are 20 characters long. Since the 4x register space is divided into 16-bit registers, a 20-character name must span 10 registers. In the case of the latter would be when an 8-bit-mapped field is concatenated with an 8-bit counter to fill one register. Great care is taken to keep related items together and when inappropriate to combine values, a "pad" definition is added.

There are three commands that can be used to read and alter information in these registers:

RD_4X_MULTI (Code 3)

Reads one or more registers from the 4x memory region. This command can be used to interrogate the controller's configuration settings.

WR_0X_SINGLE (Code 6)

This command is used to set the state of individual registers in the 4x region of the controller. The register written to can have varying effects on the controlled devices.

WR_0X_MULTI (Code 16)

This command can be used to set the state of multiple registers with one write operation. Typically this command is used in lieu of streaming multiple WR_0X_SINGLE commands.

In the following discussions of the 4x Registers, the default values are shown in **BOLD** letters or numbers where appropriate.

Controller Attributes

The first five registers are concatenated to form a 10-character panel name. This field can contain any valid ASCII printable character. The controller does not use this field; it is provided for use by host applications.

- **Panel Type** — This byte contains a 4-bit integer in the lower 4 bits to designate the panel type. For this application, the defined (supported) values are:
  - 0 – Circuit Breaker Panel
  - 1 – Relay Panel

  These two values are used by the configuration software to properly display the mapping of circuits in the Breaker Display Window.

  Values other than those described above should not be used as future ROM versions and may indeed include additional definitions.

  Note: The upper four bits are reserved for internal use and should not be used except at your own risk.

Reserved

- **Fail Mode** — The fail mode selects how the controller should act when communications with the Master Controller is lost. The MSC supports three such modes:
  - **00** = Keep running. When the fail mode is set to zero, the controller essentially ignores any communications timeout function and keeps running. This is the intended setting for a stand-alone panel application.
  - 01 = HALT. When halt is chosen for the failure mode, the controller freezes the current state of the loads. This is done by placing every load Out-Of-Service. It is important to note that when communications are restored, the loads will restore to their current active state. This means that a time schedule or switch may have requested a change-of-state while the controller was halted.
  - 02 = ALL ON. This mode places all loads on and Out-Of-Service. When communications are restored, the loads will restore to their current active state based on input and schedule actions.
- **Timeout** — The timeout value represents the number of minutes to wait after communications are lost before activating the failure mode. This value is a signed 8-bit value having a maximum time of 120 minutes. A value of **zero** means this function is disabled (default).

Load Attributes

The load records provide "personality" programming for each specific circuit breaker or relay in the panel. There are 48 total records, one per load, and each is identical.

- **Attributes** — The attributes described below are bit-mapped options and are written to the registers as integer values.
  - **B7** — Load present: Clear this bit for any position not containing a controllable circuit breaker or relay. Leaving this bit set could result in false alarms if the point is commanded on by mistake. Also clearing this bit tells the controller to ignore any commands to this load, which will eliminate retries when commanding the breaker. Clearing this bit for non-controllable breakers provides improved performance due to retry cycles.
  - **B6** — Load enabled: Set this bit to one to allow internal logic to control the breaker. An "enabled" load allows it to be influenced by external force-coil commands as well as internal group commands. If it is desired to not have local control, clear this bit.
  - B5 — Always: The always bit tells the controller what to do if the load is present but NOT enabled (B6 = 0). Setting this bit to one will force the controller to evaluate any command attempt to this load on; while clearing this bit would result in any command turning it off. The combination of bits 6 and 5 can be used to disallow control of a load yet provide alarm reporting of trip, or manual activations.
  - B4 — Wink: The Wink bit allows the selected function to act on this load or not. As wink is a group-wide function, (see Group Attributes below) it may be desirable to have a load ignore this request. Possible reasons for clearing this bit would be to prevent HID lights, or heavy draw loads, from winking.
  - B3 — B0 NOT DEFINED.
- **On-Delay** — **Zero** by default; this value is the number of ten-millisecond ticks to wait before allowing any subsequent loads to turn ON. Legal values are zero to disable this function and any number between 10 and 250. Numbers less that 10 may not have a noticeable effect depending on the controlled device.
- **Groups A-P** — Each load can be assigned to one or more groups. If a group is not assigned to a load, it cannot be controlled by switches. The Groups parameter is a 16-bit bit field where the msb represents group P and the lsb group A. For a load to actually turn off, it is necessary that all group flags assigned this load must be off. If any one flag is on the load will turn on, but all assigned groups must be off before the load will turn off.
- **Description** — The description is a 20-character name for the load. This field is actually 10 4x registers concatenated to hold the ASCII printable character load name. The controller makes no use of the data contained within this field.

Group Attributes

- **Attributes** — **Zero** by default. There are no configuration options in this application.
- **Wink Time** — **Zero** by default; this value can be set to a value between 1 and 40 minutes to enable winking for this group. Winking occurs when the group is turned off. The load goes immediately off then after a brief delay (~ 1 sec.) turns back on for the duration (in minutes) of this timer. (Refer to 3x definitions above for further explanation.)
- **Description** — The description is a 20-character name for the group. This field is actually 10 4x registers concatenated to hold the ASCII printable character group name. The controller makes no use of the data contained within this field.

Input Attributes

- **Attributes** — Is a bit mapped field for which the following definitions apply.
  - B7 — Reserved (always **zero**).
  - B6 — Input Enabled: Obsolete (see switch type below).
  - B5 — Unused (always **zero**).
  - B4 — The closure type can be defined to **normally open (zero)** or normally open (one). The default setting causes the input to activate when a closure to ground occurs.
  - B3 — Honor Wink: **Zero** by default, setting this bit to one will cause the input to honor winking when turning off. This feature, when enabled, will wink when a switch is used to turn off the lights. Clearing this bit will force the lights to turn off and stay off.
  - B2 — Off Enable: **Zero** by default, setting this bit to one will cause the off-action of the switch to immediately turn the lights off and cancel any override timers associated with this group.
  - B1 & B0 — NOT DEFINED.
- **Switch Type** — This field is used to define the type of input connected. The MSC supports four distinct input types.
  - 0 — None. An input type of none is disabled.
  - **1** — **Maintained** Contact switches are typical in lighting applications. These switches are usually wired so that up is on and down is off.
  - 2 — Maintained Change switches are a hybrid of the Maintained Contact. This switch type toggles the action upon every change of state. Using this type can alleviate the need to turn a switch off then back on to get the lights on.
  - 3 — Momentary (On|Off) is a simple push-button switch. Each press will alternate the resulting action (push-on, push-off).
  - 4 — Momentary (3-wire) is a hybrid of the Momentary on|off. This switch type goes one step further in that it pairs two adjacent inputs to work with one 3-wire switch. This switch typically remains in the center position and requires a press upward to turn the lights on and downward for off. When connecting this switch to a controller, it is necessary to maintain the proper relationship. Wire the common connection to one of the white blocks, the ON connection to the odd terminal, and the OFF wire to the next higher even terminal. For example, 1-2, 3-4, 5-6 and 7-8 are legitimate pairs.

    The rules for inputs are generally Input 1 controls Group 1, Input 2 – Group 2, etc. The distinction being the 3-wire switch which only controls the odd group of the pair.
- **Override Time** — **Zero** by default, setting this to any value between 1 and 1440 minutes (1 minute to 24 hours) will enable the timed override function of the input. Unless the switch has the Off Enable feature set, (see attrib B2) turning off the switch will have no effect. The load(s) will remain on until the Override Time reaches zero. If the switch is turned on before the current override expires, it will be re-loaded with this value. That is the timer is refreshed not additive.
- **OPTION 1 & 2** — For future consideration.
- **Description** — The description is a 20-character name for the group. This field is actually 10 4x registers concatenated to hold the ASCII printable character group name. The controller makes no use of the data contained within this field.

Schedule Attributes

- **Attributes** — Is a bit mapped field for which the following definitions apply.
  - B7 — B2 Unused (always **zero**).
  - B1 — Present active state: This is in reality a status flag used by the schedule program. It is necessary to write a zero to this bit when a schedule is changed.
  - B0 — New active state: Again primarily a status flag but writing a zero to this bit when the schedule is changed is required.
- **Reserved** — Currently a pad to make the attributes 16 bits.
- **Day** — This field has dual purposes. It holds the bit-mapped mask for the day of the week or a hard date for one-time schedules.
  - Day-of-week scheduling is set by sending data in the following format: 0x0000 0000 0SaFrTh WeTuMoSu.
  - Placing a one in the relative bit positions will enable the schedule to run on that weekday. For example, setting to 0x0000 0000 0100 0001 will enable the schedule for Saturday (the leftmost one) and Sunday (the rightmost one). Another means of determining the value is to calculate the sum of the weighted bit values. For example, the above would become (1 * 1) + (1 * 64) = 65 or HEX 41. Therefore, the register value would be HEX 0041 for a weekend-only schedule.
  - If a fixed-date schedule is desired then the most significant bit should be a one and the month (1 – 12) added to the upper byte. The lower byte contains the day. For example, May 15 would be 0x1000 0101 0000 1111 or HEX 850F.

    Note: The fixed date schedules do not incorporate a year, they are; however, persistent. This means that July 4 schedules will execute every July 4.

From Time

Until Time

These two entries are identical in format. Writing all ones HEX FFFF will disable this schedule. To set a valid time, the value is computed by the formula.

- Time = (HOURS * 60) + MINUTES
- To set a schedule for between 8:30 AM and 5:00 PM the values would be:
  - From Time = (8 * 60) + 30 = 510 or HEX 01FE
  - Until Time = (17 * 60) + 0 = 1020 or HEX 03FC
- Notice that the math is done using 24-hour format time. Also, notice that these times end up simply being the number of minutes since midnight.

Groups

The groups consist of a bit field. The rightmost, or least significant, bit corresponds to Group-A while the leftmost is Group-P. The value in this field can be calculated by summing the weighted values for each bit position set to one (see the Appendix).

- **Description** — The description is a 20-character name for the group. This field is actually 10 4x registers concatenated to hold the ASCII printable character group name. The controller makes no use of the data contained within this field.

Special Topics

In addition to the standard Modbus interface through register reads and writes, the MSC also provides some special functions specific to this controller.

Controller Functions

There are three controller specific commands that the MSC supports. These special functions don't use the Modbus register interface and are comprised of special case messages and formats. Each is described below.

Report Slave ID

This command responds to the standard Modbus Function Code 0x11 (17) and simply returns an identifier string and some run-time status. The reply is:

- 0x11 – Function Code
- 0x0D – the Byte count of the following data
- 0x50, 0x52, 0x43, 0x31, 0x30, 0x30, 0x20 – the string "PRC100"
- 0x4D, 0x3v, 0x3v, 0x20(or 0x2b) – the string Mvv, where 'vv' is the two-digit version number, followed by a space or "+". The plus ("+") is added for partial revisions not warranting a full version increase.
- 0x0N, where "N" represents the condition_code register value. This register has bit 0 set if an override is currently active for any one, or more, breakers. Bit 1 is set if there are any alarms (feedback errors) currently active.
- 0xFF, the last byte of data is the run-time indicator. Since the act of communicating with the device implies it is running, this value will always be 0xFF.

Restart Controller

The restart controller uses the Modbus Function Code 0x41 (65) and with a sub-code and qualification key can be used to restart the controller from a remote location.

The command format for a restart is:

- 0x41 – Function Code
- 0x04 – Restart warm sub-code, use 0x0C for restart default (see below)
- 0xNN – (NN = 0 for cold, or the number of loads to default as described below)
- 0x52, 0x50, 0x43, 0x31, 0x30, 0x30 – the string "PRC100"
- The 0x04 Cold sub-command can be used to force re-initialization of certain hardware and software parameters. Using this command is the same as pressing the reset button on the local controller.

The 0x0C Default sub-command is a special case that clears all programming and restores specific default conditions. The restart default will:

1. Clear ALL 3x, and 4x Register space
2. Sets all loads to present and enabled
3. Sets all loads as members of Group-P
4. Sets all inputs to type 1 (Maintained) with OFF enabled
5. All other features are turned off

This function was included with intent to allow temporary operation by a contractor. When a MSC is installed into a panel and shipped to a construction site, the contractor can connect a switch across input 16 and one of the ground points to temporarily operate all the loads in the panel. If it is desired to not have all loads added to Group P, then adding 128 to the number of default loads will inhibit the addition to Group P. For example, a 42-circuit panelboard with no breaker assigned to Group P would use a load count of 128 + 42 = 170 (0xBA).

Clock Functions

Not described in any of the register definitions is the Time-Of-Day data. This data is written or read through a special case Modbus command. The data format in both directions is the same, and is described below. The difference between the read and write packets is in the clock sub-command. Time packets are formatted as follow:

Setting the time:

	HEX	DECIMAL
Function code	0x42	66
Byte count	0x0D	13
Identifier	0x65	101
Sub-command	0xFC	252
RTC data	0x00	0
RTC data	0x00	0
RTC minutes	0x00 to 0x3B	00 to 59
RTC hours	0x00 to 0x17	0 to 23
RTC d.o.w.	0x01 to 0x07	01 -Sun to 07 -Sat
RTC day	0x01 to 0x1F	01 to 31
RTC month	0x01 to 0x0C	01 -Jan to 12 -DEC
RTC year	0x03 to 0x63	03 to 99
DST start (N.A. Std.)	0x4000 (see text)	16384
DST end (N.A. Std.)	0x0000 (see text)	00

- For NO DST support:
  - DST start = 0x0000 (0)
  - DST end = 0x0000 (0)
- For other than North America:
  - DST date = (month * 32) + day
  - For DST start time, add 0x4000 (16384) to enable DST
  - Use DST date to calculate DST end
  - For example, Daylight Saving Time starts May 10 and ends Sep 3; calculate the two values
  - May 10 = (5*32)+10 = 170
  - Sep 3 = (9*32)+3 = 291
  - DST start = 16384 + 170 = 16554 (0x40AA)
  - Dst end = 291 = (0x0123)

Getting Time

Getting the time is much the same as setting the time. The command (request) is shown below and the data returned is exactly like that sent when setting the time. The request packet format is:

	HEX	FIXED VALUE
Function code	0x42	66
Byte count	0x0D	13
Identifier	0x65	101
Sub-command	0x7C	124

Note: Notice the only difference is the sub-command.

Appendix A: Modbus Command Reference

TABLE 1. POW-R-COMMAND PRC1000 MOD100 INSTALLATION AND PROGRAMMING REFERENCE

COMMAND	REGION	NOTE	FUNCTION	PDU2	PDU3	PDU4	PDU5	PDU6	PDU7	PDU8	PDU9 ...
RD_0X_MULTI	0x	1 = On	0x01	R_addr_Hi	R_addr_Lo	R_cnt_Hi	R_cnt_Lo				
WR_0X_SINGL	0x	On Off	0x05 0x05	R_addr_Hi R_addr_Hi	R_addr_Lo R_addr_Lo	0xFF 0x00	0x00 0x00				
WR_0X_MULTI	0x	1 = On	0x0F	R_addr_Hi	R_addr_Lo	R_cnt_Hi	R_cnt_Lo	Byte_cnt	DATA …		
RD_1X_MULTI	1x	1 = On	0x02	R_addr_Hi	R_addr_Lo	R_cnt_Hi	R_cnt_Lo				
RD_3X_MULTI	3x		0x04	R_addr_Hi	R_addr_Lo	R_cnt_Hi	R_cnt_Lo				
RD_4X_MULTI	4x		0x03	R_addr_Hi	R_addr_Lo	R_cnt_Hi	R_cnt_Lo				
WR_4X_SINGL	4x		0x06	R_addr_Hi	R_addr_Lo	value_Hi	value_Lo				
WR_4X_MULTI	4x		0x10	R_addr_Hi	R_addr_Lo	R_cnt_Hi	R_cnt_Lo	byte_cnt	DATA …		
RD_SLAVEID	ROM		0x11								
RESTART	ROM	Cold Default	0x41 0x41	0x04 0x0C	0x00 LD_cnt	0x50 ("P") 0x50 ("P")	0x52 ("R") 0x52 ("R")	0x43 ("C") 0x43 ("C")	0x31 ("1") 0x31 ("1")	0x30 ("0") 0x30 ("0")	0x30 ("0") 0x30 ("0")
RDWR_CLOCK	ROM	SET	0x42	0x0D	0x65	0xFC	Time Data <see text>				
		GET	0x42	0x0D	0x65	0x7C					

TABLE 2. POW-R-COMMAND FUNCTIONS

FUNCTION	CODE	FUNCTION NAME	REFERENCE	POW-R-COMMAND IMPLEMENTATION
0x01	1	Read Coil Status	0x	Read Control Point Command States and Input Monitors.
0x02	2	Read Discrete Inputs	1x	Read Control Point Status.
0x03	3	Read Holding Registers	4x	Read one, or more, 16-bit parameter registers.
0x04	4	Read Input Registers	3x	Read Wink and Override Timer Values.
0x05	5	Write Single Coil	0x	Force a single breaker or group.
0x06	6	Write Single Holding Reg.	4x	Write a single 16-bit parameter.
0x0F	15	Write Multiple Coils	0x	Force contiguous breakers or groups.
0x10	16	Write Multiple Holding Reg.	4x	Write contiguous 16-bit parameter registers.
0x11	17	Request Slave ID	n/a	Request Product Identifier, version, and condition code.
0x41	65	User Defined Codes	n/a	Restart the controller.
0x42	66	User Defined Codes	n/a	Read / Set the real-time-clock.

Appendix B: Modbus Register Reference

TABLE 3. MODBUS REGISTER REFERENCE — 0X (MB COILS)

REGISTERS				ENTITY SIZE			
HEX START	ADDRESS		DEVICE/RECORD	REGS	BYTES	DESCRIPTION	ACCESS
Circuit Breaker/Relay Standard Commands							
0x0000	0	7	1	8	8	1	Peer Coils (Breakers 1 – 8 Control)	R/W
0x0008	8	15	9	16	8	1	Peer Coils (Breakers 9 – 16 Control)	R/W
0x0010	16	23	17	24	8	1	Peer Coils (Breakers 17 – 24 Control)	R/W
0x0018	24	31	25	32	8	1	Peer Coils (Breakers 25 – 32 Control)	R/W
0x0020	32	39	33	40	8	1	Peer Coils (Breakers 33 – 40 Control)	R/W
0x0028	40	47	41	48	8	1	Peer Coils (Breakers 41 – 48 Control)	R/W
0x0030	48	55	49	57	8	1	Peer Flags (General Purpose Flags 1 – 8)	R/W
0x0038	56	63	58	64	8	1	Peer Flags (General Purpose Flags 9 – 16)	R/W
Circuit Breaker/Relay Override Commands							
0x0040	64	71	1	8	8	1	Priority Coils (Breakers 1 – 8 Control)	R/W
0x0048	72	79	9	16	8	1	Priority Coils (Breakers 9 – 16 Control)	R/W
0x0050	80	87	17	24	8	1	Priority Coils (Breakers 17 – 24 Control)	R/W
0x0058	88	95	25	32	8	1	Priority Coils (Breakers 25 – 32 Control)	R/W
0x0060	96	103	33	40	8	1	Priority Coils (Breakers 33 – 40 Control)	R/W
0x0068	104	111	41	48	8	1	Priority Coils (Breakers 41 – 48 Control)	R/W
0x0070	112	119	49	57	8	1	RESERVED	
0x0078	120	127	58	64	8	1	RESERVED	
Circuit Breaker/Relay Override Status							
0x0080	128	135	1	8	8	1	Service Coils (Breakers 1 – 8 Control)	R/W
0x0088	136	143	9	16	8	1	Service Coils (Breakers 9 – 16 Control)	R/W
0x0090	144	151	17	24	8	1	Service Coils (Breakers 17 – 24 Control)	R/W
0x0098	152	159	25	32	8	1	Service Coils (Breakers 25 – 32 Control)	R/W
0x00A0	160	167	33	40	8	1	Service Coils (Breakers 33 – 40 Control)	R/W
0x00A8	168	175	41	48	8	1	Service Coils (Breakers 41 – 48 Control)	R/W
0x00B0	176	183	49	57	8	1	RESERVED	
0x00B8	184	191	58	64	8	1	RESERVED	
Group Control/Status							
0x00C0	192	199	1	8	8	1	Group Coils (Groups A through H Control)	R/W
0x00C8	200	207	9	16	8	1	Group Coils (Groups I through P Control)	R/W
Input Changed Monitors							
0x00D0	208	215	1	8	8	1	Sw 9 – 16 Changed (16 – msb)	R/W
0x00D8	216	223	9	16	8	1	Sw 1 – 8 Changed (8 – msb)	R/W

TABLE 4. MODBUS REGISTER REFERENCE — 1X (MB DISCRETE INPUTS)

REGISTERS				ENTITY SIZE			
HEX START	ADDRESS		DEVICE/RECORD	REGS	BYTES	DESCRIPTION	ACCESS
Circuit Breaker/Relay Feedback Status							
0x0000	0	7	1 8	8	1	Feedback Coils (Breakers 1 – 8 Control)	RO
0x0008	8	15	9 16	8	1	Feedback Coils (Breakers 9 – 16 Control)	RO
0x0010	16	23	17 24	8	1	Feedback Coils (Breakers 17 – 24 Control)	RO
0x0018	24	31	25 32	8	1	Feedback Coils (Breakers 25 – 32 Control)	RO
0x0020	32	39	33 40	8	1	Feedback Coils (Breakers 33 – 40 Control)	RO
0x0028	40	47	41 48	8	1	Feedback Coils (Breakers 41 – 48 Control)	RO
0x0030	48	55	49 57	8	1	RESERVED	RO
0x0038	56	63	58 64	8	1	RESERVED	RO
Circuit Breaker/Relay Commanded State							
0x0040	64	71	1 8	8	1	Command Coils (Breakers 1 – 8 Control)	RO
0x0048	72	79	9 16	8	1	Command Coils (Breakers 9 – 16 Control)	RO
0x0050	80	87	17 24	8	1	Command Coils (Breakers 17 – 24 Control)	RO
0x0058	88	95	25 32	8	1	Command Coils (Breakers 25 – 32 Control)	RO
0x0060	96	103	33 40	8	1	Command Coils (Breakers 33 – 40 Control)	RO
0x0068	104	111	41 48	8	1	Command Coils (Breakers 41 – 48 Control)	RO
0x0070	112	119	49 57	8	1	RESERVED	
0x0078	120	127	58 64	8	1	RESERVED	
Circuit Breaker/Relay Alarm Status							
0x0080	128	135	1 8	8	1	Alarm Coils (Breakers 1 – 8 Control)	RO
0x0088	136	143	9 16	8	1	Alarm Coils (Breakers 9 – 16 Control)	RO
0x0090	144	151	17 24	8	1	Alarm Coils (Breakers 17 – 24 Control)	RO
0x0098	152	159	25 32	8	1	Alarm Coils (Breakers 25 – 32 Control)	RO
0x00A0	160	167	33 40	8	1	Alarm Coils (Breakers 33 – 40 Control)	RO
0x00A8	168	175	41 48	8	1	Alarm Coils (Breakers 41 – 48 Control)	RO
0x00B0	176	183	49 57	8	1	RESERVED	
0x00B8	184	191	58 64	8	1	RESERVED	
Switch Input Monitors							
0x00C0	192	199	1 8	8	1	Switches 9 – 16 Log (16 – msb)	RO
0x00C8	200	217	9 16	8	1	Switches 1 – 8 Log (8 – msb0)	RO
Reserved							
0x00D0	208	215	1 8	8	1	RESERVED	
0x00D8	216	223	9 16	8	1	RESERVED	
Condition Codes							
0x00E0	224	231	1 8	8	1	Pad	RO
0x00E8	232	239	9 16	8	1	Condition Code	RO
Schedule Status							
0x00F0	240	247	1 8	8	1	Schedule 1 – 8 Active	RO
0x00F8	248	255	9 16	8	1	Schedule 9 – 16 Active	RO
0x0100	256	263	17 24	8	1	Schedule 17 – 24 Active	RO
0x0108	264	271	25	1	1	Schedule 25 Active	RO

TABLE 5. MODBUS REGISTER REFERENCE — 3X (MB INPUTS REGS)

REGISTERS				ENTITY SIZE			
HEX START	ADDRESS		DEVICE/RECORD	REGS	BYTES	DESCRIPTION	ACCESS
Override Timers							
0x0000	0		1 0 ①	1	2	Input 1 Override Timer	RO
0x0001	1	15	2	15	30	Input 2 – 16 Override Timer	RO
Load Wink Timers							
0x0010	16		1 0 ①	0.5	1	Load 1 Wink Delay (Off Timer) Load 1 Egress Timer (Time remaining in Wink Cycle)	RO RO
0x0011	17	63		47	94	Loads 2 – 48 Egress Timers	RO

① Offset

TABLE 6. MODBUS REGISTER REFERENCE — 4X (MB HOLDING REGS)

REGISTERS					ENTITY SIZE			
HEX START	ADDRESS		DEVICE/RECORD		REGS	BYTES	DESCRIPTION	ACCESS
Controller Attributes								
0x0000	0	4	1	0 ①	5	10	Controller Name	R/W
0x0005	5			5 ①	0.5	1	Panel Type Definition	R/W
					0.5	1	[RESERVED]	R/W
0x0006	6			6 ①	0.5	1	Fail Mode	R/W
					0.5	1	Fail Timeout	R/W
Load Attributes								
0x0007	7		1	0 ①	0.5	1	Load 1 Attributes	R/W
					0.5	1	On Delay	R/W
0x0008	8			1 ①	0.5	1	Groups I - P (P - msb)	R/W
					0.5	1	Groups A - H (H - msb)	R/W
0x0009	9	18		2 ①	10	20	Description	R/W
0x0013	19	30	2		12	24	Load 2 Record	R/W
0x001F	31	42	3		12	24	Load 3 Record	R/W
0x002B	43	54	4		12	24	Load 4 Record	R/W
0x0037	55	66	5		12	24	Load 5 Record	R/W
0x0043	67	78	6		12	24	Load 6 Record	R/W
0x004F	79	90	7		12	24	Load 7 Record	R/W
0x005B	91	102	8		12	24	Load 8 Record	R/W
0x0067	103	114	9		12	24	Load 9 Record	R/W
0x0073	115	126	10		12	24	Load 10 Record	R/W
0x007F	127	138	11		12	24	Load 11 Record	R/W
0x008B	139	150	12		12	24	Load 12 Record	R/W
0x0097	151	162	13		12	24	Load 13 Record	R/W
0x00A3	163	174	14		12	24	Load 14 Record	R/W
0x00AF	175	186	15		12	24	Load 15 Record	R/W
0x00BB	187	198	16		12	24	Load 16 Record	R/W
0x00C7	199	210	17		12	24	Load 17 Record	R/W
0x00D3	211	222	18		12	24	Load 18 Record	R/W
0x00DF	223	234	19		12	24	Load 19 Record	R/W
0x00EB	235	246	20		12	24	Load 20 Record	R/W
0x00F7	247	258	21		12	24	Load 21 Record	R/W
0x0103	259	270	22		12	24	Load 22 Record	R/W
0x010F	271	282	23		12	24	Load 23 Record	R/W
0x011B	283	294	24		12	24	Load 24 Record	R/W
0x0127	295	306	25		12	24	Load 25 Record	R/W
0x0133	307	318	26		12	24	Load 26 Record	R/W
0x013F	319	330	27		12	24	Load 27 Record	R/W
0x014B	331	342	28		12	24	Load 28 Record	R/W
0x0157	343	354	29		12	24	Load 29 Record	R/W
0x0163	355	366	30		12	24	Load 30 Record	R/W
0x016F	367	378	31		12	24	Load 31 Record	R/W
0x017B	379	390	32		12	24	Load 32 Record	R/W
0x0187	391	402	33		12	24	Load 33 Record	R/W
0x0193	403	414	34		12	24	Load 34 Record	R/W
0x019F	415	426	35		12	24	Load 35 Record	R/W
0x01AB	427	438	36		12	24	Load 36 Record	R/W
0x01B7	439	450	37		12	24	Load 37 Record	R/W
0x01C3	451	462	38		12	24	Load 38 Record	R/W
0x01CF	463	474	39		12	24	Load 39 Record	R/W
0x01DB	475	486	40		12	24	Load 40 Record	R/W
0x01E7	487	498	41		12	24	Load 41 Record	R/W
0x01F3	499	510	42		12	24	Load 42 Record	R/W
0x01FF	511	522	43		12	24	Load 43 Record	R/W
0x020B	523	534	44		12	24	Load 44 Record	R/W
0x0217	535	546	45		12	24	Load 45 Record	R/W
0x0223	547	558	46		12	24	Load 46 Record	R/W
0x022F	559	570	47		12	24	Load 47 Record	R/W
0x023B	571	582	48		12	24	Load 48 Record	R/W

① Offset

TABLE 6. MODBUS REGISTER REFERENCE — 4X (MB HOLDING REGS) CONTINUED

REGISTERS					ENTITY SIZE			
HEX START	**ADDRESS**		**DEVICE/RECORD**		**REGS**	**BYTES**	**DESCRIPTION**	**ACCESS**
Group Attributes								
0x0247	583		1	0	0.5 0.5	1 1	Group 1 Attributes Wink Time	R/W R/W
0x0248	584	593		1	10	20	Description	R/W
0x0252	594	604	2		11	22	Group 2 Configuration Options	R/W
0x025D	605	615	3		11	22	Group 3 Configuration Options	R/W
0x0268	616	626	4		11	22	Group 4 Configuration Options	R/W
0x0273	627	637	5		11	22	Group 5 Configuration Options	R/W
0x027E	638	648	6		11	22	Group 6 Configuration Options	R/W
0x0289	649	659	7		11	22	Group 7 Configuration Options	R/W
0x0294	660	670	8		11	22	Group 8 Configuration Options	R/W
0x029F	671	681	9		11	22	Group 9 Configuration Options	R/W
0x02AA	682	692	10		11	22	Group 10 Configuration Options	R/W
0x02B5	693	703	11		11	22	Group 11 Configuration Options	R/W
0x02C0	704	714	12		11	22	Group 12 Configuration Options	R/W
0x02CB	715	725	13		11	22	Group 13 Configuration Options	R/W
0x02D6	726	736	14		11	22	Group 14 Configuration Options	R/W
0x02E1	737	747	15		11	22	Group 15 Configuration Options	R/W
0x02EC	748	758	16		11	22	Group 16 Configuration Options	R/W
0x02F7	759		1	0	0.5 0.5	1 1	Input 1 Attributes Switch Type	R/W R/W
0x2F8	760			1	1	2	Override Time	R/W
0x2F9	761			2	0.5 0.5	1 1	OPTION-1 [FUTURE] OPTION-2 [FUTURE]	R/W R/W
0x2FA	762	771	3		10	20	Description	R/W
0x0304	772	784	2		13	26	Input 2 Record	R/W
0x0311	785	797	3		13	26	Input 3 Record	R/W
0x031E	798	810	4		13	26	Input 4 Record	R/W
0x032B	811	823	5		13	26	Input 5 Record	R/W
0x0338	824	836	6		13	26	Input 6 Record	R/W
0x0345	837	849	7		13	26	Input 7 Record	R/W
0x0352	850	862	8		13	26	Input 8 Record	R/W
0x035F	863	875	9		13	26	Input 9 Record	R/W
0x036C	876	888	10		13	26	Input 10 Record	R/W
0x0379	889	901	11		13	26	Input 11 Record	R/W
0x0386	902	914	12		13	26	Input 12 Record	R/W
0x0393	915	927	13		13	26	Input 13 Record	R/W
0x03A0	928	940	14		13	26	Input 14 Record	R/W
0x03AD	941	953	15		13	26	Input 15 Record	R/W
0x03BA	954	966	16		13	26	Input 16 Record	R/W

TABLE 6. MODBUS REGISTER REFERENCE — 4X (MB HOLDING REGS) CONTINUED

REGISTERS				ENTITY SIZE			
HEX START	**ADDRESS**	**DEVICE/RECORD**		**REGS**	**BYTES**	**DESCRIPTION**	**ACCESS**
Schedule Attributes							
0x03C7	967	1	0	0.5 0.5	1 1	Schedule 1 Attributes [RESERVED]	R/W R/W
0x03C8	968		1	1	2	Sched. DAY	R/W
0x03C9	969		2	1	2	From Time	R/W
0x03CA	970		3	1	2	Until Time	R/W
0x03CB	971		4	0.5 0.5	1 1	Groups I – P (P – msb) Groups A – H (H – msb)	R/W R/W
0x03CC	972	981	5	10	20	Description	R/W
0x03D6	982	996	2	15	30	Schedule 2 Record	R/W
0x03E5	997	1011	3	15	30	Schedule 3 Record	R/W
0x03F4	1012	1026	4	15	30	Schedule 4 Record	R/W
0x0403	1027	1041	5	15	30	Schedule 5 Record	R/W
0x0412	1042	1056	6	15	30	Schedule 6 Record	R/W
0x0421	1057	1071	7	15	30	Schedule 7 Record	R/W
0x0430	1072	1086	8	15	30	Schedule 8 Record	R/W
0x043F	1087	1101	9	15	30	Schedule 9 Record	R/W
0x044E	1102	1116	10	15	30	Schedule 10 Record	R/W
0x045D	1117	1131	11	15	30	Schedule 11 Record	R/W
0x046C	1132	1146	12	15	30	Schedule 12 Record	R/W
0x047B	1147	1161	13	15	30	Schedule 13 Record	R/W
0x048A	1162	1176	14	15	30	Schedule 14 Record	R/W
0x0499	1177	1191	15	15	30	Schedule 15 Record	R/W
0x04A8	1192	1206	16	15	30	Schedule 16 Record	R/W
0x04B7	1207	1221	17	15	30	Schedule 17 Record	R/W
0x04C6	1222	1236	18	15	30	Schedule 18 Record	R/W
0x04D5	1237	1251	19	15	30	Schedule 19 Record	R/W
0x04E4	1252	1266	20	15	30	Schedule 20 Record	R/W
0x04F3	1267	1281	21	15	30	Schedule 21 Record	R/W
0x0502	1282	1296	22	15	30	Schedule 22 Record	R/W
0x0511	1297	1311	23	15	30	Schedule 23 Record	R/W
0x0520	1312	1326	24	15	30	Schedule 24 Record	R/W
0x052F	1327	1341	25	15	30	Schedule 25 Record	R/W

Appendix C: Modbus Interface Devices

The PRC100MSC Modbus System Controller requires a RS-485 adaptor to interface the PC-based Configuration Software to the MSC Controller. Modern-day PCs are beginning to replace the older RS-232 Serial Ports with the USB communications choice. These ports are physically different and have unique adaptors for each host type.

Documented herein are two adaptors, one each RS-232 to RS-485 and USB to RS-485. The following text discusses the wiring and configuration for each type.

RS-232 to RS-485 Adaptor

B&B Electronics Model Number 485TBLED

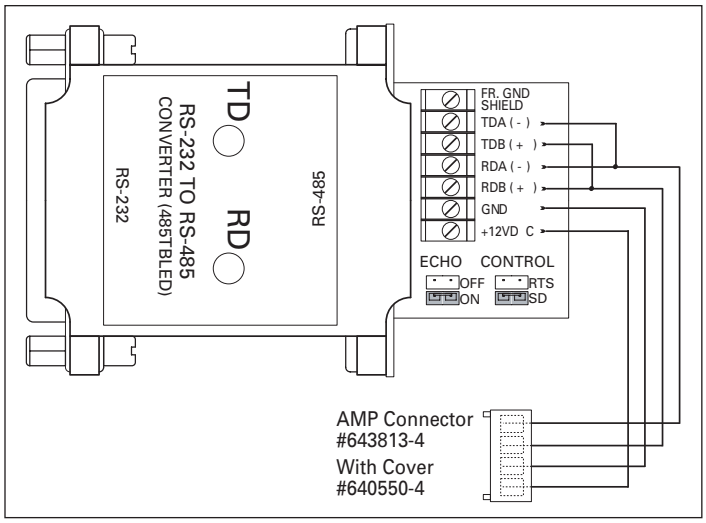


FIGURE 5. MODBUS SERIAL 485 CONNECTION

This device is powered from the diagnostic port on the MSC, because of this it is necessary to connect all four wires from the diagnostic port to the 485TBLED terminals. In addition to wiring the power and network, the 485TBLED must have the TDA and RDA terminals connected. This makes the device a single-pair Tx/Rx device. As shown, the BLACK (network -) wire is connected to "TDA(-)" and also to "RDA(-)". Likewise the RED (network +) wire is connected to "TDB(+)" and "RDB(+)". The WHITE and GREEN wires connect to power the 485TBLED converter. The WHITE wire connects to the "GND" terminal and the GREEN wire connects to the "+12VDC" terminal.

In addition to the wiring, there are also two jumpers that need to be configured. The two jumper blocks are labeled "ECHO" and "CONTROL" and are described below.

Echo Jumpers

Place the supplied jumper across the "ON" pair. The Modbus configuration software relies on the echoed characters for communication integrity checks. If this jumper is not set for ECHO = ON, then the software will report communication errors.

Control Jumpers

The "CONTROL" jumpers need to be configured to facilitate the converter's transition between transmit and receive modes. This is accomplished by setting the jumper across the "SD" pins. This allows the converter to automatically sense data direction from the PC and selects the transmitter or receiver accordingly.

USB to RS-485 Adaptor

B&B Electronics Model Number USTL4

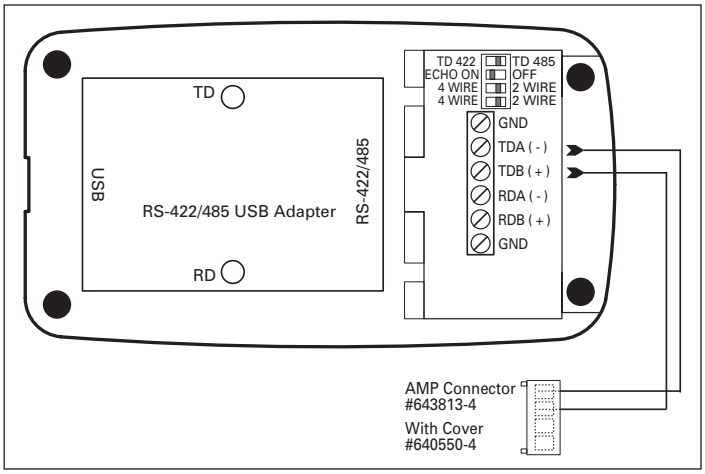


FIGURE 6. MODBUS USB 485 CONNECTION

This converter is powered from the USB port and thus may require a powered USB hub depending on your individual requirements. The Photograph shows the USB cable connected into the left side of the device and shows the cable connected to the terminals on the right side. Above the terminal strip is a four-position DIP switch that is used to configure how the device will communicate.

Wiring this device is much simpler than the 485TBLED above. The USTL4 requires only two connections. The RED (network +) wire is to be connected to the "TDB(+)" and the BLACK (network -) wire to the "TDA(-)" terminal. DO NOT CONNECT THE WHITE ("GND") OR GREEN (+15 VDC) WIRES TO THE USTL4. DAMAGE TO THE CONVERTER AND OR CONNECTED EQUIPMENT MAY RESULT.

In addition to these two connections, the DIP switch must also be configured as shown below.

DESC	OFF	ON
TD 422		X
ECHO ON	X	
4 WIRE		X
4 WIRE	X	X

Note: As with other USB devices connected to a computer running the Windows Operating System, a USB device will be assigned a COMM PORT number. This number can be found by looking at the system properties (Device Manager) and matching the assigned Port to the "USB RS-485 Port (COM#) where "#" will be the com port number assigned. It is necessary to configure the appropriate COM port in the Settings menu of the configuration software.

Using Other Converters

While this document outlines the two most popular converters, it in no way implies that only these converters will work. Several models and vendors are available from which to obtain RS-485 converters. If you choose to use a converter other than the two described herein, then you do so at your own risk. Be aware than the major criteria that MUST be met before any converter can work are the ability to support, transmit and receive over a single pair of wires (2-wire operation), and the ability to echo the PC transmitted characters back to the PC. The ability to switch between transmit and receive can be accomplished by either the data direction sensing or the "RTS" hand-shake line handshake the RS-232 port. It is generally preferable to use the "SD" (Send Data) option as this requires only the Transmit, Receive and Signal Ground wires be connected from the PC to the device.

Appendix D: Header for M3+ ROM Memory Map

```
/*  physical constants */
#define MAXDIO 64       /* number of digital I/O points */
#define NUMDO 48        /*number of actual control points (breakers)*/
#define GRPINP 16       /* Group Control Inputs */
#define MEBINP 48       /* Expansion Board inputs (excludes group ins) */
#define SW_OPT1 0x04    /* Option switch SW2.3 Port 3.2 */
#define SW_OPT2 0x08    /* Option switch SW2.4 Port 3.3 (MEB present) */

/* max values */
#define NUMGRP 16       /* max number of groups (A-P)*/
#define NUMSCH 25       /* max number of schedules */

/* Arithmetic constants */
#define PEER 0          /* offset added to a load number for set/get coil ops */
#define PRIORITY 64
#define SERVICE  64

#pragma EJECT

/*         data structures    */
/* data structure bit defines   */
//Condition Codes (bit addresses)
#define CCOVR  0x00     /* an override is active (out of service exists) */
#define CCALRM 0x01     /* Feedback error alarm exists */
#define CC_CLK 0x02     /* Real Time Clock failure */
#define CCWDOG 0x03     /* Watchdog reset occurred */
//
#define CC_YY  0x04     /* unused */
#define CC_ZZ  0x05     /* unused */
#define CC_OPT1     0x06     /* from P3.2 */
#define CC_OPT2     0x07     /* from P3.3 */

/*-----------------------------*/
// Load configuration
#define LDPRES 0x80     /* Breaker present (programed) */
#define LDENBL 0x40     /* Breaker control Enabled */
#define LDOVR  0x20     /* Breaker always on (1) off (0) if not enabled */
#define LDWINK 0x10     /* Breaker Wink (1) no Wink (0) */
#define WINKOFF    100      /* 100 10ms ticks or 1 second */

// Input configurations
#define PRI_SW 0x01     /* b0 = 0 = peer input, 1 = priority */
#define RELEASE     0x02     /* b1 = 1 = restore, 0 = override off */
#define OFFENBL 0x04    /* b2 = 1 = clr ovr timer on off */
#define EGRESS 0x08     /* b3 = 1 = wink on off    */
#define INVERT 0x10     /* b4 = 1 = invert (n.c. operation)*/
#define ENABLE 0x40     /* b6 = 1 = enable load control */

//Schedule flags
#define SCH_ACTV 0x02  /* Schedule active flag */
/*-----------------------------*/
// Controller Options
// uart_cfg
#define UCBAUD 0x01     /* 1 = 19200, 0 = 9600 */
#define UCWORD 0x02     /* 1 = 8 bits, 1 Stop, 0 = 7 bits 2 stop. */
#define UCPAREN     0x04     /* 1 = Parity enabled, 0 = no parity */
#define UCPAR   0x08     /* 1 = Odd parity, 0 = even */
```

```
// failmode
#define FMRUN   0x00   /* Keep running with backup schedules if present */
#define FMHALT   0x01   /* stop prossing all schedules and inputs */
#define FMALLON     0x02   /* turn all loads on, cease all processing load requests */
#define FMALLOFF 0x03  /* turn all loads off, cease processing load requests */

// Miscelaneous constants
//FORCE used by eval _ set _ grp to clear the override and  wink timers
#define FORCE _ NONE 0
#define FORCE _ NO _ TIMER  1
#define FORCE _ NO _ WINK 2
#define FORCE _ ALL  FORCE _ NO _ TIMER + FORCE _ NO _ WINK

// Function code modifiers
#define BROADCAST 0x80 /* Modbus function code received as broadcast */


/*--------------------------------------------**
** ------------------------                    **
**  MODBUS 8-bit Registers                     **
** ------------------------                    **
**  MODBUS 0X  Coils Structure Definitions     **
**--------------------------------------------*/
struct MB0 _ REGS {
    // Breaker control and service coils
    unsigned char peer _ coils[MAXDIO/8];         // 8 bytes 64 bits 1 = on
    unsigned char priority _ coils[MAXDIO/8];     // 8 bytes 64 bits 1 = on
    unsigned char service _ coils[MAXDIO/8];      // 8 bytes 64 bits 1 = on
    // Group control coils
    unsigned char group _ coils[NUMGRP/8];       // 2 bytes 16 bits 1 = on
    // resettable integrators input flags
    unsigned int input _ changed;                // 2 bytes 16 bits, toggle
    unsigned char meb _ changed[MEBINP/8];      // 6 bytes 48 bits, toggle  (excluding group inputs
49-64)
    //ADD ADDITIONAL REGISTER DEFS HERE           34 bytes TOTAL
};

//MUST UPDATE RAW[##] TO REFLECT CHANGES TO STRUCT MB0 _ REGS
union UREG _ 0 {
    unsigned char raw[34]; // raw, or register num access
    struct MB0 _ REGS regs;  // variable name access
};


/** ------------------------                    **
**  MODBUS 1X  Coils Structure Definitions     **
**--------------------------------------------*/
struct MB1 _ REGS {
    unsigned char feedback _ coils[MAXDIO/8];     // 8 bytes 64 bits 1 = on
    unsigned char command _ coils[MAXDIO/8];     // 8 bytes 64 bits 1 = on
    unsigned char alarm _ coils[MAXDIO/8];       // 8 bytes 64 bits 1 = on
    unsigned int sw _ inp _ log;                 // 2 bytes 16 bits 1 = 0n
    unsigned int reserved;                       // 2 bytes 16 bits 1 = active
//    unsigned char pad;                         // 1 byte
    unsigned char features;            /* feature map 0 = enable, 1 = exclude    */
                                          // bit 0: 1 = NO Schedule Button
                                          // bit 1: 1 = NO MEB button
                                          // bit 2: 1 = NO Wink or Egress controls
                                          // bit 3: 1 = NO Override Timer controls
                                          // bit 4: 1 = NO MEB PEER/PRIORITY controls.
```

```
                                        // bit 5: 1 = NO edit failmode params
                                        // bit 6: undefined
                                        // bit 7: undefined
    unsigned char condition_code;          // 1 byte
    unsigned char schedule_active[4];      // 4 bytes
    unsigned char meb_inp_log[MEBINP/8];   // 6 bytes      (excluding group inputs 49-64)
    //ADD ADDITIONAL REGISTER DEFS HERE          40 bytes TOTAL
};

//MUST UPDATE RAW[##] TO REFLECT CHANGES TO STRUCT MB1_REGS
union UREG_1 {
    unsigned char raw[40]; // raw, or register num access
    struct MB1_REGS regs;  //variable name access
};


/*--------------------**
** ------------------ **
**  16-bit Registers  **
** ------------------ **
**--------------------*/
/** ------------------------                    **
**  MODBUS 3X Input Structure Definitions      **
**-------------------------------------------*/
struct OVERRIDES {
    int    ld_timer;               /*  Load Override Timer (1440 minutes) */
};

struct WINK {
    char wink_delay;       /*  off-time timer 250 MAX = 1.2 seconds */
    unsigned char egress_timer;                /*  time remaining in wink cycle ( 120 minutes)
*/
};

struct MB3_REGS {
    struct OVERRIDES ovr_times[GRPINP];   /* 2 *  16 =  32 */
    struct WINK winking[NUMDO];           /* 2 *  48 =  96 */
    unsigned int romchk;                  /* 1 *   2 =   2 */
};

// ADD ADDITIONAL REGISTER DEFS HERE     total    = 130 */


//MUST UPDATE RAW[##] TO REFLECT CHANGES TO STRUCT MB3_REGS
union UREG_3 {
    unsigned char raw[130];
    struct MB3_REGS regs;
};


/** ------------------------                    **
**  MODBUS 4X Input Structure Definitions      **
**-------------------------------------------*/
struct OPTIONS {
    unsigned char ctlr_name[10];       /* panel name */
    unsigned char type_cfg;            /* bitmap bit 0 = 0 for Breaker = 1 for Relay Panel */
    unsigned char reserved;           /* unused byte for integer boundary */
    unsigned char failmode;           /* how to act after comm loss with master */
    char timeout;                     /* loss of master comms before fail (minutes) */
};  // 14 RAW BYTES
```

```
struct LOADREC {
    unsigned char attrib;  /* configuration bitmap */
                                    /* bit 7 load present */
                                    /* bit 6 load enabled */
                                    /* bit 5 if NOT enabled then always this state (1=ON) */
                                    /* bit 4 blink 0=NO blink  */
                                    /* bit 3, 2, 1, 0 UNUSED   */
    unsigned char on _ delay;       /* delay between on commands (10millisecond count) */
    unsigned int group _ ap;        /* bitmap b0 = group A, b15 = group P    */
    unsigned char description[20];
};  //24 RAW BYTES

struct GROUPREC {
    char config;           /* possible bit-map field for group options */
    char wink _ time;               /* 0 = disabled, 1 to 40 Wink (minutes) */
    char description[20];
};  // 22 RAW BYTES

struct INPUTREC {
    unsigned char attrib;  /* configuration bitmap */
                                    /* b7 unused */
                                    /* b6 enabled 1 = input recognized and processed */
                                    /* b5 always 1 = on or 0=off if enabled = 0 */
                                    /* b4 0 = norm open, 1 = norm closed */
                                    /* b3 1 = wink honored */
                                    /* b2 OFF Enabled  */
                                    /* b1 unused */
                                    /* b0 unused  */
    unsigned char inp _ type;    /* contact performance type */
                                    /* 1 = maint. 2-wire, 2 = maint. change 2-wire */
                                    /* 3 = momentary single 2-wire, 4 = momentary double 3-
wire */
    int ovr _ time;                 /* override time 0 ignored 1 -> 1440 minutes */
    char option _ 1;                /* Reserved for future possibilities */
    char option _ 2;                /* Reserved for future possibilities */
    unsigned char description[20];
};  // 26 RAW BYTES

struct SCHEDULE {
    unsigned char flag;             /* bit0 - new active state      1=active */
                                    /* bit1 - present active state  1=active */
                                    /* bit4 - unused */
                                    /* bit5 - unused */
                                    /* bit6 - unused */
                                    /* bit7 - unused */

    unsigned char pad;              /* future use */
    unsigned int day;               /* 0x8000+date or day mask  */
    unsigned int from;
    unsigned int til;
                                    //from and til times (unitary)
                                    //  FFFF  no time
                                    //  0xxx  xxx is minutes since midnight
                                    //  ryyy  r is relative flags (for future ASTRO
implementation)
                                    //         bit 15   1 - relative   0 - absolute
                                    //         bit 14   1 - sunrise/set  0 - from/til
dependent
                                    //         bit 13   1 - til or set   0 - from or rise
```

```
                                        //          bit 12   1 - minus        0 - plus
                                        //          yyy is relative minutes
    unsigned int action;
    unsigned char description[20];      /* added for ModBUS HIC */
} ; // 30 RAW BYTES

struct MEBINREC {
    unsigned char attrib;  /* configuration bitmap */
                                        /* b7 unused */
                                        /* b6 enabled 1 = input controls breaker
                                        /* b5 unused */
                                        /* b4 0 = norm open, 1 = norm closed */
                                        /* b3 unused */
                                        /* b2 unused */
                                        /* b1 unused */
                                        /* b0 0 = PEER, 1 = PRIORITY */
    unsigned char inp_type;             /* contact performance type */
                                        /* 1 = maint. 2-wire, 3 = momentary (toggle) */
};  // 2 RAW BYTES

/*struct LOCATION astro {
};
struct CLOCK dates {
};*/

struct MB4_REGS {
    struct OPTIONS params;              // 14 *  1 =   14 bytes
    struct LOADREC relay[NUMDO];        // 24 * 48 = 1152 bytes
    struct GROUPREC group[NUMGRP];      // 22 * 16 =  352 bytes
    struct INPUTREC input[GRPINP];      // 26 * 16 =  416 bytes
    struct SCHEDULE sch[NUMSCH];        // 30 * 25 =  750 bytes
    struct MEBINREC meb_in[NUMDO];      //  2 * 48 =   96 bytes
}
;                                       // TOTAL     2780 bytes

//MUST UPDATE RAW[##] TO REFLECT CHANGES TO STRUCT MB4_REGS
union UREG_4 {
    unsigned char raw[2780];
    struct MB4_REGS regs;
};
```

Addendum: Modbus Expansion Board Installation

Features added to the Pow-R-Command PRC100MSC Installation Reference Guide

The Pow-R-Command PRC100MEB Modbus Expansion Board provides 48 inputs in addition to the 16 already provided on the MSC. Relocating the Group Inputs to the MEB and adding 48 additional inputs yielding a total of 64 inputs provides additional capacity. The relocated Group Inputs have the same function and flexibility as originally supplied and are programmed exactly as described in the MSC manual. The 48 additional inputs (Discrete Inputs) are provided primarily for discrete control of individual circuit breakers. While the MEB has only terminations for the Group Inputs onboard, it picks up the additional 48 inputs from the backplane of the Expansion Controller or the RTB II, 7X Expansion Chassis. The MEB must be wired via a cable harness to the MSC, as this is how input data is communicated to the MSC.

Installation

The following procedures should be followed with power off to both the MSC and MEB. It will become apparent why as the procedure unfolds.

The MEB should be installed into the same panel as the parent MSC. If the panel is an Expansion Controller or Relay Cabinet, then the MEB is placed into the bottom pair pf connectors (P7 and P8 just above the screw terminals). If the MEB is installed into a Pow-R-Command smart panelboard, then it should be placed into the provided 7X Expansion Chassis. The 7X Expansion Chassis has only one pair of connectors to accept the MEB. Refer to the Pow-R-Command System Installation Guide for details of component placement and field wiring termination.

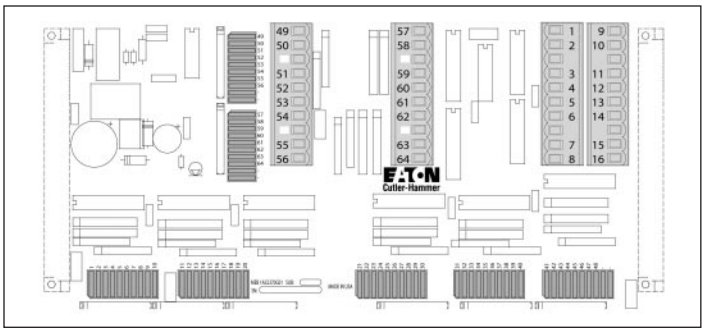


FIGURE 7. MODBUS EXPANSION BOARD LAYOUT

It is possible to remotely mount the MEB. If this is desired, consult your Cutler-Hammer Representative for custom ordering an enclosure, and special wiring harness for power.

The MEB gets its power from the system backplane. It runs from the 28 VAC system supply. The MEB gets its additional inputs from the RTB, or similar, backplane and communicates commands to the MSC via the MSC's original Group Input terminals.

Configure the MSC

For the MSC to properly interpret the input signals, it must be configured to recognize MEB input data patterns. Failure to configure the MSC will result in unpredictable and undesirable results. In the lower-left quadrant of the MSC is a four-position DIP switch. The switches are labeled sequentially in descending order from top to bottom. The top position of this switch (Labeled OPT2) is the switch controlling the input configuration. The default position set at the factory is OFF. This position configures the MSC to only recognize the onboard 16 inputs. Moving this switch to the left will prepare the MSC to recognize the expanded inputs provided by the MEB.

Note: Just changing the switch position will have no apparent effect on the operation of the MSC. The MSC will only recognize the change after power-on or reset. For this reason it is absolutely necessary that this switch be left in the appropriate position at all times. Failure to observe this simple precaution could result in unexpected behavior after a power failure.

Installing the MEB

MEB Input List

Input	Description	Status	Type	NormClosed	Enable	Priority
01	Breaker 1 Desc.	ON	Maintained		Yes	peer
02	Breaker 2 Desc.	ON	Momentary (ON|OFF)		Yes	peer
03		OFF	Maintained			peer
04	Breaker 4 Desc.	OFF	Maintained Change		Yes	PRIORITY
05		OFF	Maintained		Yes	peer
06		OFF	Maintained		Yes	peer
07		OFF	Maintained		Yes	peer
08		OFF	Maintained		Yes	peer

FIGURE 8. MEB INPUT LIST

The MEB is plugged onto the RTB, or similar backplane, completing the power connection. The field wiring removed from the MSC (if any was previously installed) is relocated to the provisions on the MEB. Group Inputs 1-8 (MSC-TB1) is moved to the MEB TB1 (SW49-56) socket. Similarly the Group Inputs 9-16 (MSC-TB2) is moved to the MEB TB2 (SW57-64) socket (See **Figure 8**).

A wiring harness between the MSC and MEB completes the installation. The MSC-TB1 connector is wired from the MEB-TB3 (TO MSC 1-8) terminal and the MSC-TB2 connector is wired from the MEB-TB4 (TO MSC 9-16) terminal.

Connecting the MEB to the MSC

Theory of Operation

Connecting the MEB to the MSC provides additional inputs for discrete breaker control. The switches wired to the RTB, or similar backplane, provide a one-to-one relationship with the installed circuit breakers. Each switch can be individually configured as a dissociated input or one that effects control over the breaker. Different input configurations are possible to accommodate maintained or momentary contact switches (see the OPERATION section following).

The MEB encodes these 64 inputs into 8 groups of 8 switches. The actual switch data is communicated to the MSC on its inputs 1-8 terminals while the group number is encoded on the inputs 9-16. The MSC reads these input signals and interprets them according to the setting of the OPT2 switch. Is it is configured for a MEB input, then it reads the switch data and uses the group data to generate a switch number.

Perhaps this operation is best described by example. Switch 17 of the RTB is closed. While the MEB scans the 64 inputs (8 at a time) it places the switch data onto the 1 – 8 pins and then the group onto the 9 – 16 pins. Switch 17 is in the third group (group 1 is 1 – 8, group 2 is 9 – 16, and group 3 is 17 – 24). This implies that the MSC will see a pulse on the input 1 terminal AND a pulse on the input 11 terminal (9+ group number). This strobing runs continuously at approximately 140 Hz. The MEB free-runs the scanning operation and the MSC samples the presented data approximately every 10 ms. Because of this scanning and parsing and a certain amount of random error, it can take up to 240 to 300 ms for a switch to be recognized. This delay is due in part to the MSC internal debounce of the inputs. Each input is sampled every 10 ms but there must be 4 consecutive successful reads to be valid. A debounce time of 40 ms and a sampling of 8 groups can mean a maximum delay of 320 ms debounce time.

⚠ WARNING

BECAUSE OF THIS STROBING AND ENCODING OF DATA CONNECTING THE MEB WITHOUT FIRST CONFIGURING THE MSC TO PROPERLY DECODE WILL RESULT IN WILD AND UNCONTROLLABLE INPUT PARSING BY THE MEC. FOR THIS REASON IT IS ALWAYS ADVISED TO FIRST CONFIGURE THE MSC BEFORE CONNECTING THE MEB. FAILURE TO RECOGNIZE THIS SIMPLE FACT WILL RESULT IN ERRATIC BEHAVIOR AND MAY RESULT IN PRODUCT DAMAGE, OR PERSONAL INJURY.

Operation

The following text describes the configuration options for the MEB discrete inputs (MEB 1-48). You must be connected to a MSC configured with the MEB. If the OPT2 switch is off, then access to the MEB configuration is restricted. Please refer to this Addendum, Section 1 for help configuring the MSC.

Viewing Discrete Inputs

The MEB Configuration is accessed through a button on the Panel Details screen of the CHmbTool program. Launch the software and open the desired panel. If the MSC just opened, was correctly configured with the MEB, a button labeled "MEB Cfg" will be visible in the upper pane of the Panel Properties window. Clicking this button will open the "Discrete Inputs List" window and the attributes of each input numbered 1 – 48 will be displayed in a list.

The Discrete Inputs List presents the data in a spreadsheet-like arrangement (see **Figure 8**). The input numbers appear down the left column and each column across is discussed in detail below.

Description

If the switch has "Yes" in the ENABLE column, this column will show the description entered into the Breaker record. If the switch is not enabled for breaker control then "???" will be displayed in this column.

Status

The status column will display "ON" for any switch that is closed (ignoring any invert option). "OFF" is displayed for any open switch. It is important to realize the status is derived from the contact closure state and not the logical result of the process. The status of momentary contacts will only display "ON" while the contacts are closed.

Type

Each discrete input can be configured to recognize two different contact types, with one additional hybrid type. If "<disabled>" appears in this column, then no processing of this input will be performed. The other choices are: "Maintained", "Maintained Change", and "Momentary ON|OFF". These switch types are detailed in the MSC installation and programming guide, the MEB inputs act as described there.

Normally Closed

If this column is blank, then the switch is configured as having normally open contacts. A "Yes" in this column indicates that the input is wired with normally closed contacts.

Enable

This column displays "Yes" to indicate that this input has been enabled to issue commands to the associated circuit. A blank entry in this column indicates that it is just a digital input with no control association. It is possible to use a switch for general-purpose input functions. The Integrators Flags can be used to interact with the MEB/MSC.

Priority

When "PRIORITY" is displayed in the priority column, this switch can override any other action or state affecting this circuit. A PRIORITY switch acts as an override switch. Overrides are discussed in greater detail in the following Section. If this column displays "peer" then this switch has the same level of influence as do group controls. Please refer to the MSC Installation and Programming manual for additional information regarding PEER and PRIORITY concepts.

Configuring Discrete Inputs

Double-clicking any record in the Discrete Inputs List window will open the associated configuration dialog (see **Figure 9**). This dialog presents some information already presented above, but adds some additional control options.

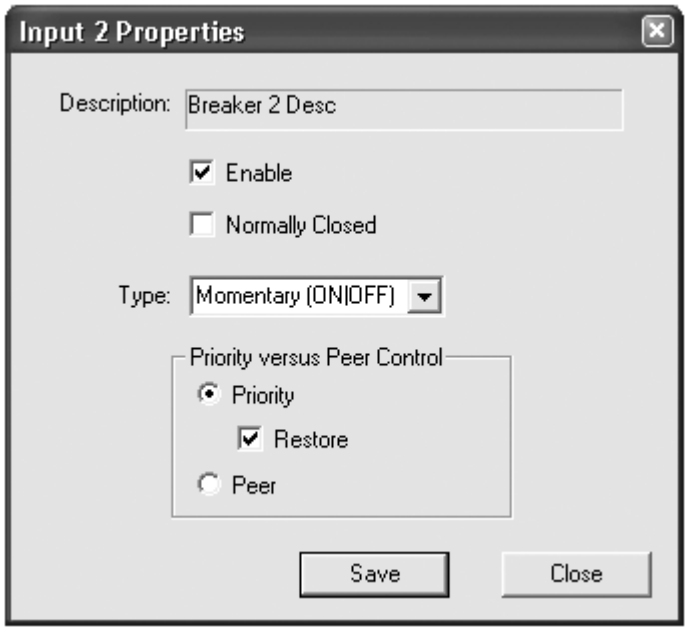


FIGURE 9. DISCRETE INPUT CONFIGURATION

Description — This un-editable field is as described above.

Enable — The enable, when checked, allows this input to control the associated circuit breaker or relay.

Switch Type — Clicking on the down-pointing arrow will drop a list of available choices. The choices act the same as those described in the MSC manual; please refer to that manual for additional information.

Normally Closed — Placing a check in this box will allow the MSC/MEB to recognize a switch opening as the actionable event. In other words, this check will change the contact performance from Normally Open, to Normally Closed.

Priority — Clicking in this radio button will make the switch write commands to the Priority Coils of the circuit breaker.

TABLE 7. MODBUS REGISTER REFERENCE — 0X

REGISTERS				ENTITY SIZE			
HEX START	ADDRESS		DEVICE/RECORD	REGS	BYTES	DESCRIPTION	ACCESS
Expansion Input Changed Monitors (Discrete Control Inputs)							
0x00E0	224	231	1	8	8	1	RTB/RTB II Sw 1 – 8	R/W
0x00E8	232	239	9	16	8	1	RTB/RTB II Sw 9 – 16	R/W
0x00F0	240	247	17	24	8	1	RTB/RTB II Sw 17 – 24	R/W
0x00F8	248	255	25	32	8	1	RTB/RTB II Sw 25 – 32	R/W
0x0100	256	263	33	40	8	1	RTB/RTB II Sw 33 – 40	R/W
0x0108	264	271	41	48	8	1	RTB/RTB II Sw 41 – 48	R/W

TABLE 8. MODBUS REGISTER REFERENCE — 1X

REGISTERS				ENTITY SIZE			
HEX START	ADDRESS		DEVICE/RECORD	REGS	BYTES	DESCRIPTION	ACCESS
Expansion Input Monitors							
0x0110	272	279	1	8	8	1	RTB/RTB II Switches 1 – 8 log	RO
0x0118	280	287	9	16	8	1	RTB/RTB II Switches 9 – 16 log	RO
0x0120	288	295	17	24	8	1	RTB/RTB II Switches 17 – 24 log	RO
0x0128	296	303	25	32	8	1	RTB/RTB II Switches 25 – 32 log	RO
0x0130	304	311	33	40	8	1	RTB/RTB II Switches 33 – 40 log	RO
0x0138	312	319	41	48	8	1	RTB/RTB II Switches 41 – 48 log	RO

TABLE 9. MODBUS REGISTER REFERENCE — 4X

REGISTERS				ENTITY SIZE			
HEX START	ADDRESS		DEVICE/RECORD	REGS	BYTES	DESCRIPTION	ACCESS
Expansion Switch Attributes							
0x053E	1342	1	0 ①	0.5 0.5	1 1	RTB Switch 1 attributes Switch Type	R/W R/W
0x053F	1343	2		1	2	RTB Switch 2 Record	R/W
0x0540	1344	3		1	2	RTB Switch 3 Record	R/W
0x0541	1345	4		1	2	RTB Switch 4 Record	R/W
0x0542	1346	5		1	2	RTB Switch 5 Record	R/W
0x0543	1347	6		1	2	RTB Switch 6 Record	R/W
0x0544	1348	7		1	2	RTB Switch 7 Record	R/W
0x0545	1349	8		1	2	RTB Switch 8 Record	R/W
0x0546	1350	9		1	2	RTB Switch 9 Record	R/W
0x0547	1351	10		1	2	RTB Switch 10 Record	R/W
0x0548	1352	11		1	2	RTB Switch 11 Record	R/W
0x0549	1353	12		1	2	RTB Switch 12 Record	R/W
0x054A	1354	13		1	2	RTB Switch 13 Record	R/W
0x054B	1355	14		1	2	RTB Switch 14 Record	R/W
0x054C	1356	15		1	2	RTB Switch 15 Record	R/W
0x054D	1357	16		1	2	RTB Switch 16 Record	R/W
0x054E	1358	17		1	2	RTB Switch 17 Record	R/W
0x054F	1359	18		1	2	RTB Switch 18 Record	R/W
0x0550	1360	19		1	2	RTB Switch 19 Record	R/W
0x0551	1361	20		1	2	RTB Switch 20 Record	R/W
0x0552	1362	21		1	2	RTB Switch 21 Record	R/W
0x0553	1363	22		1	2	RTB Switch 22 Record	R/W
0x0554	1364	23		1	2	RTB Switch 23 Record	R/W
0x0555	1365	24		1	2	RTB Switch 24 Record	R/W
0x0556	1366	25		1	2	RTB Switch 25 Record	R/W
0x0557	1367	26		1	2	RTB Switch 26 Record	R/W
0x0558	1368	27		1	2	RTB Switch 27 Record	R/W
0x0559	1369	28		1	2	RTB Switch 28 Record	R/W
0x055A	1370	29		1	2	RTB Switch 29 Record	R/W
0x055B	1371	30		1	2	RTB Switch 30 Record	R/W
0x055C	1378	31		1	2	RTB Switch 31 Record	R/W
0x055D	1379	32		1	2	RTB Switch 32 Record	R/W
0x055E	1380	33		1	2	RTB Switch 33 Record	R/W
0x055F	1381	34		1	2	RTB Switch 34 Record	R/W
0x0560	1382	35		1	2	RTB Switch 35 Record	R/W
0x0561	1383	36		1	2	RTB Switch 36 Record	R/W
0x0562	1384	37		1	2	RTB Switch 37 Record	R/W
0x0563	1385	38		1	2	RTB Switch 38 Record	R/W
0x0564	1386	39		1	2	RTB Switch 39 Record	R/W
0x0565	1387	40		1	2	RTB Switch 40 Record	R/W
0x0566	1389	41		1	2	RTB Switch 41 Record	R/W
0x0567	1401	42		1	2	RTB Switch 42 Record	R/W
0x0568	1413	43		1	2	RTB Switch 43 Record	R/W
0x0569	1425	44		1	2	RTB Switch 44 Record	R/W
0x056A	1437	45		1	2	RTB Switch 45 Record	R/W
0x056B	1449	46		1	2	RTB Switch 46 Record	R/W
0x056C	1461	47		1	2	RTB Switch 47 Record	R/W
0x056D	1473	48		1	2	RTB Switch 48 Record	R/W

① Offset

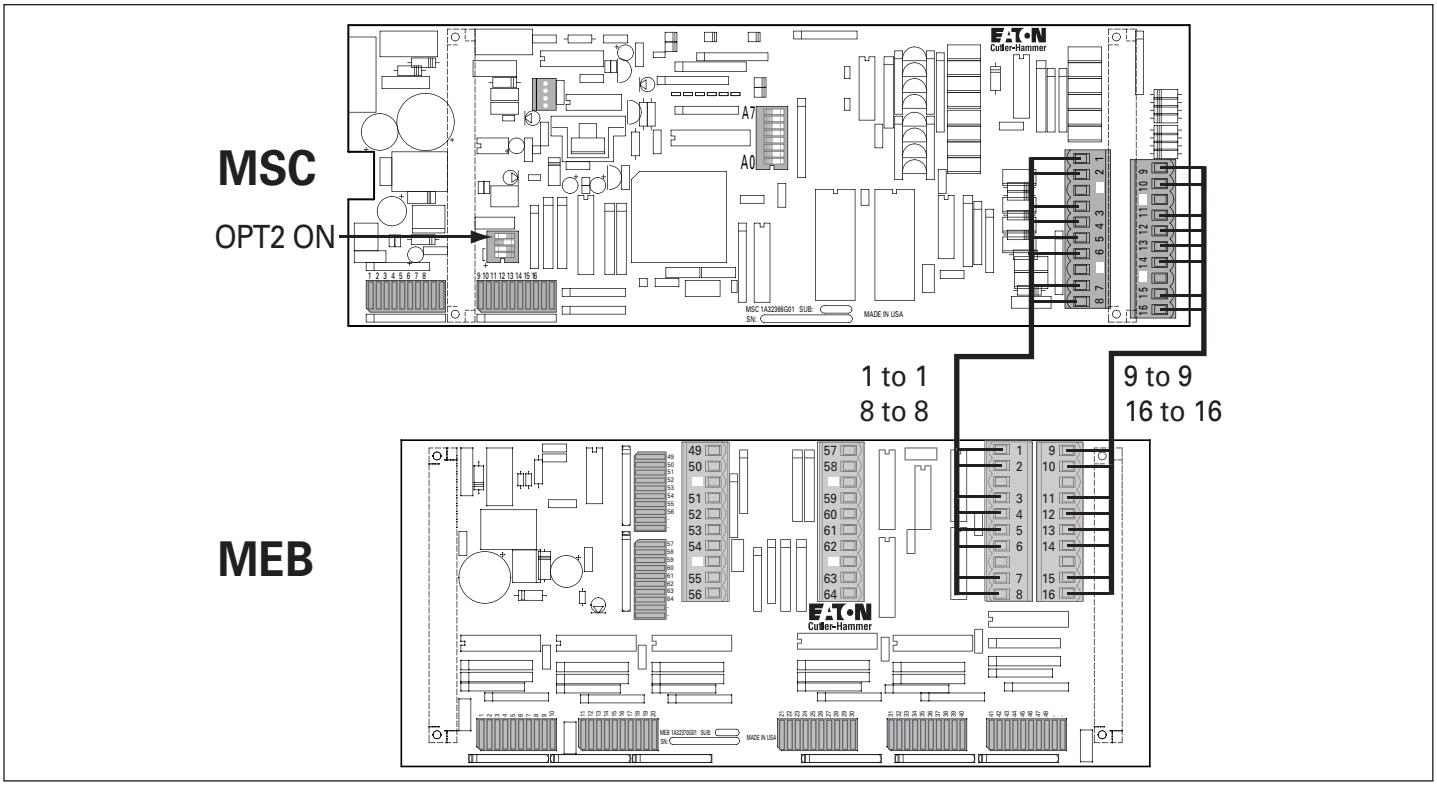
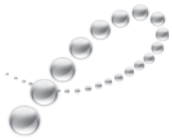


FIGURE 10. WIRING THE MEB TO THE MSC

Notes

Notes

Eaton Electrical Inc.
Electrical Group
1000 Cherrington Parkway
Moon Township, PA 15108
United States
877-ETN-CARE (877-386-2273)
Eaton.com



PowerChain Management®

Printed in USA
Publication No. IL01412015E / Z6900
September 2008

EATON | Cutler-Hammer