MC³ Configuration and Troubleshooting for Modbus

Serial port Comm 1 in an MC³ can be configured to support Modbus ASCII or Modbus RTU. This allows you to establish direct communications to almost any industrial network, either directly or using versatile converters. Merrick currently support interfacing to Profibus DP, EtherNet/IP, Modbus/TCP, Modbus Plus, DeviceNet or ControlNet using this method.

The network host can take control over and/or monitor the MC³, using CIT data table exchange. For details of the CIT structure, see http://www2.merrick-inc.com/mct/CIT.pdf

System Requirements

Later versions of the Merrick's MC³ controllers support the Mobus ASCII and Modbus RTU communications protocols. They will act as slaves, supporting the Modbus functions 3 (Read Holding Registers) and 16 (Preset Multiple Registers). Function 8, sub function 0 (Return Query Data) is also supported.

The information in this document applies to the following MC³ firmware versions:

Firmware	Used for	Released	Comm Ver
20.10.EX.F	Belt Feeder	03/28/02	1
20.20.EX (All)	Belt feeder	04/17/03	2
24.10.EX.H	Pressurized Coal Feeder	08/02/02	1
24.10.EX.I and later	Pressurized Coal Feeder	08/12/03	1
30.00.EX.C and later	Loss-In-Weight	04/25/02	1
30.10.EX.E and later	Enhanced Loss-In-Weight	06/06/04	2
40.10.EX.A and later	Impact Flow Meter	04/14/03	2
90.10.EX.Y and later	MasterSet	01/02/03	2

Other Merrick firmware releases may also support ModBus RTU communications.

The physical interface is 4-wire RS-485. The master is assumed to maintain a cyclic conversation with the controllers, which exposes a Common Interface Table (CIT), making it possible monitor and/or supervise the MC³ completely.

Different data types are used for control/status bits, integer numbers and floating-point numbers. Control/Status bits and Integer numbers are organized in 16 bit words. Parameters are organized as IEEE 32 bit floating point numbers, located in two consecutive 16 bit words. For PLCs that don't support floating point numbers, it is possible to split parameters into two 16 bit integers, one containing the integer part and one containing the fractional part multiplied by 10,000 (four implied decimal places).

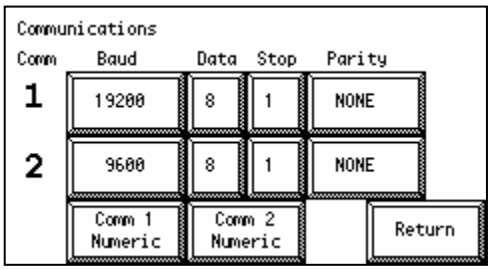
Configuring the MC³

Configuring the MC³ controllers include setting up communications parameters, register tags, warnings, faults and external inputs and outputs. The register tags are set to make any internal MC³ parameter appear in Tag 1 R to Tag 5 R values. Standard feeder parameters are always available. Warnings and faults are user preference qualified, associated with any logical I/O point in the MC³. This configuration is done regardless if communications is used or not. Any logical inputs you want to control from the PLC must be mapped to an external input. In the same way, logical outputs must be mapped to external outputs for monitoring purposes.

The following menu references and screen shots were taken using the MC³ 20.20.EX.B Belt Feeder Controller application. Operation and Maintenance Manuals as well as register specifications are available at the Merrick Web Site:

http://www2.merrick-inc.com/mct/MC3Apps/MC3Apps.htm

Setting the MC³ communications parameters



To get to the **Communications** screen from the main screen, touch **Action Menu**, **Settings Menu**, enter the password, **Inputs & Outputs** and finally **Comm Settings**. Modbus RTU runs on COM 1. In this example, we are using, 19200 baud, 8 data bits, 1 stop bit and No parity. These parameters must agree with the settings in the Modbus Master.

Touching the **Comm 1 Numeric** button takes you to the **Comm Numeric Params** screen. Set the parameters as follows:

Parameter	Value	Comment
Controller #	any	This is the Slave Address. They must be different for each controller.
Start Char	10	Has no meaning for Modbus RTU
End Char	13	Has no meaning for Modbus RTU
Comm Timeout	5.0	If the MC³ does not receive any valid Mofbus telegrams for this time, the logical output "**Ser Comm Lost**" will go ON.
Comm Protocol	2	Enter 1 for ASCII or 2 for RTU
Write Protect	3071	BFF hex. All registers write protected except Primary Setpoint
Word Order	0	All registers have normal word order. This has to be found by trial and error. The Siemens S7 PLC, for example, has the word order for a floating point value reversed (as compared with a Modicon PLC).
Int/Frac FP	0	Floating Point transfer is supported. Integer/Fraction is not needed.
Tag Reg 1 - 5	MC³ reg #	Internal MC³ Register Number you want to monitor. The corresponding register value will appear in the "Tag 1 R value" position.

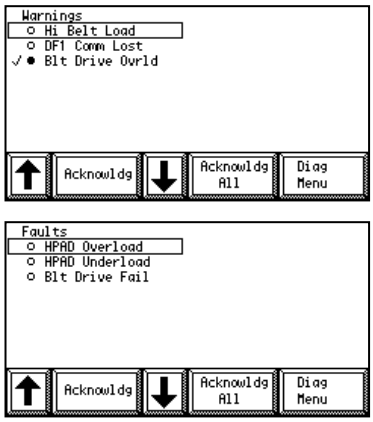
If you extend the size of the write table, you will overwrite "Write Protect", "Word Order", "Int/Frac FP" and "Tag Reg 1-5" parameters from the PLC. This allows for multiplexing of the parameters in the "Tag x R Value" positions.

Configuring Warnings and Faults

Warnings and Faults are qualifiers to logical inputs and outputs, normally set by the user. Warnings are considered to require attention. Faults are considered to be fatal for the feeder operation, and the controller will attempt to stop the feeder. See O&M, page 56. In this example the warnings and faults are set up according to the following table:

Logical I/O	**Qualifier**	**Comment**
HPAD Overload	Fault	Invalid Load Cell Signal
HPAD Underload	Fault	Invalid Load Cell Signal
Blt Drive Fail	Fault	Signal from the belt motor VFD, connected to an MC³ input.
High Belt Load	Warning	Too much material on the belt
Blt Drive Ovrld	Warning	Signal from the belt motor drive, connected to an MC³ input.

The qualifiers are set up in the Digital Inputs (O&M Page 56) and Digital Outputs (O&M Page 60) screens. With the settings above, the Warnings and Faults screens look like this.



The state of the checkmark is transferred to the Warnings [18] and Faults [19] word in the CIT. The bit order is the same as the displayed order on the screen. It is important to note that the bits in the Warnings and Faults registers reflect the state of the checkmark, not the dot. In this warning screen, both are on for the logical input **Blt Drive Ovrld**. Bit 2 of the Warnings word is on. If the **Blt Drive Ovrld** input is turned off, then the dot goes away, but the checkmark stays until the warning is acknowledged, either on this screen or by the "Clear Warnings Command" bit [8] in the Control [44] register.

Note that the bit order in the CIT words for Warnings and Faults are not configurable. It is derived from the order of logical outputs and logical inputs in the Digital Outputs and Digital Inputs screens. If you add or remove a warning or fault qualifier to a logical output or input, the bit order changes.

Configuring External Inputs and Outputs

Logical inputs and outputs can be mapped in three ways:

1. To a physical input or output. In this example the logical input **Belt Drive Ovrld** is mapped to **Rack 1 Input 2**, which, in turn, is connected to the Overload output of the belt motor VFD. The physical output **Rack 1 Output 1** is mapped to the logical output **Drive Enable**. The output is then connected to the Start input of the belt motor VFD.
2. To an external input or output. In this example, the **Run Permission** logical input is mapped to **External Input 1**. This allows the PLC to start and stop the feeder through the External Inputs register, CIT Word 45, bit 0.
3. Unused Logical Inputs are typically connected to the Physical Input **Always On** or **Always Off**.

The PLC controls inputs to the MC³ as bits in the External Inputs register, CIT Word 45. They are then mapped to Logical Inputs in the MC³. Note that the External Inputs are numbered 1 - 16. Bits in the External Inputs word are typically numbered 0 - 15 in the PLC.

It is possible to have a physical input wrapped around to an external output (for monitoring purposes) by first map the input to an **Available I/O** point, and then map the same **Available I/O** point to an external output.

In this example, we use 4 inputs. Two are physical connections from the VFD to the MC³, one is a physical connection to the emergency stop circuit (**Feeder Block**), and one input is controlled from the PLC (**Run Permission**).

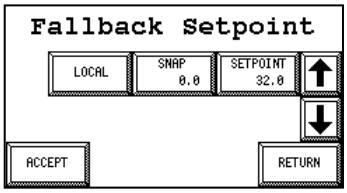
This is how the digital inputs were mapped in the MC³ for this example:

Logical	Physical	Warning/Fault	Word:Bit in the CIT
Run Permission	External Input 1		45:0 to MC³
Feeder Block	Rack 1 Input 1		41:0 from MC³
Belt Drive Ovrld	Rack 1 Input 2	Warning	41:1 from MC³
Belt Drive Fail	Rack 1 Input 3	Fault	41:2 from MC³

Digital output mapping:

Physical	Logical	Warning/Fault	Word:Bit in the CIT
Rack 1 Output 1	Fdr Drv Enable		42:0 from MC³
External Output 1	Fault		17:0
External Output 2	Warning		17:1
External Output 3	Ready		17:2
External Output 4	Hi Belt Load	Warning	17:3
External Output 5	HPAD Overload	Fault	17:4
External Output 6	HPAD Underload	Fault	17:5

Setting up the setpoint source



The MC³ Setpoint Method should be set to **Serial**. See O&M, Page 27. The setpoint is taken from CIT word 46 and 47.There is a way to make the MC³ fall back to another Setpoint method in case of communications failure. By mapping the Logical Input **Frc Comm Setpt** to **Always On**, Serial Setpoint will be used regardless of the setpoint method, until serial communication fails, setting the **Comm Timeout** logical output. When this scheme in effect, the Setpoint Screen looks like this. If the controller has not received a telegram, addressed to it, for 5 seconds the setpoint method and value will fall back to the settings in this screen. Furthermore, all External Inputs will be set to zero.

Connect the host or interface to the MC³s and check communications

Use a cable designed for RS-422. There should be two pairs, individually shielded. Belden 9368 or equivalent is a good alternative. Connect the shields only at the Modbus master. At the MC³s, connect the incoming and outgoing shields together only. Add terminating resistors at the last MC³, 121Ω. Connect one between terminals 1 and 2 and one between terminals 3 and 4. The pin numbering on the MC³ is left to right. See the picture on the next page.

Pair	Part	MC³ Terminals	RS-485 Name	Color in picture
1	1	1	Tx +	White
1	2	2	Tx -	Green
2	1	3	Rx +	Red
2	2	4	Rx -	Black

The MC³ receives data on 3 and 4 (Rx+ and Rx-), and transmits, after being correctly addressed, on terminals 1 and 2 (Tx+ and Tx-).

A “New” CPU board, Rev 11 or later, have two RS-485 ports built in. You connect to the right port on the bottom board. The DB9-P (Pins) connector to the right is the RS232 port for Comm 1. The two connectors on the left are for Comm 2.

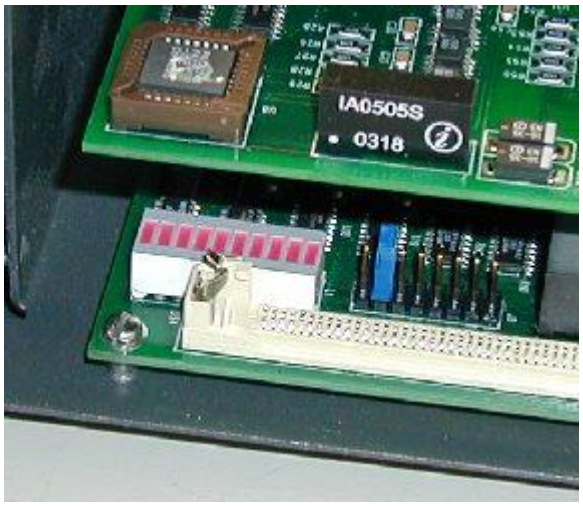
If you are using one of the recommended protocol converters from Anybus, Merrick has an adapter board (Part No M22787-1) that connects to the DB9-S connector and offers screw terminals for the cable.

When you have connected and powered up, check that all looks OK in the ModBus Diagnostic screen. See page 6.

Troubleshooting tips

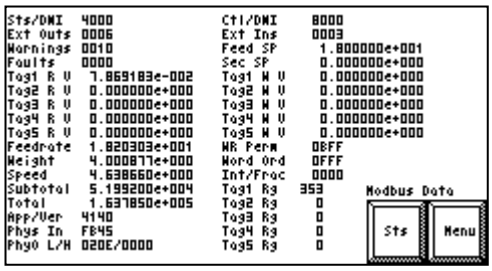
Setting up industrial networks can sometimes be a daunting task. In this example, you may have to deal with several mapping layers and communication protocols. Fortunately, there are excellent troubleshooting tools available.

Look at the LED's on the MC³ CPU board.



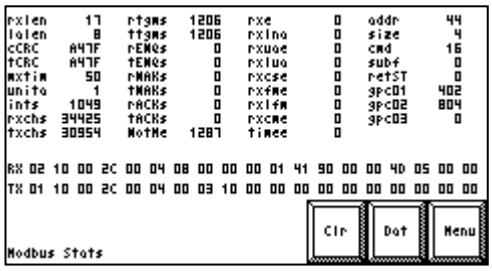
The LED indicators 3 and 4 on the MC³ CPU board are connected to serial Receive Data and Transmit Data, respectively. LED1 is to the left in this picture.

Check data in the CIT data screen.



The actual values in the CIT exposed to communications can be inspected in the MC³ by touching **Action Menu**, **Diag Display**, **Modbus Diag**, **Dat**. Note that the values are only updated on valid Modbus telegrams. If no telegrams have been received, most values are zero. As you can see, the layout follows the CIT exactly. All integer values are presented in hexadecimal format except the Tag register numbers. The 'e' format for the floating points can help troubleshooting FP transfers. You are reading from the left column, and writing to the right. If you succeed with the integrity bit, you should see bit 7 in the Sts/DNI and Ctl/DNI toggle.

Check error counters in the Communication Diagnostic screen.



Communications status and statistics can be inspected in the MC³, by touching **Action Menu**, **Diag Display**, **Modbus Diag**. The screen looks like this.

In this shot, out of 1206 successful exchanges, there was none lost to errors. There were 1287 telegrams addressed to some other Modbus device. The last command received was 16 (Preset Multiple Regs), starting at word 44, 4 words long.

Label	Meaning
rxlen	Length, in bytes, of the last incoming telegram
lalen	Length, in bytes, of the last outgoing telegram
cCRC	CRC16 value calculated out of the incoming telegram. Hex.
tCRC	CRC16 value received in the incoming telegram. Hex. Should be the same as cCRC
mxtim	Communications timeout in 100 ms ticks – "Comm Timeout" in the Com/Num menu
unita	"Slave address" for this controller number. "Controller Number" in Com/Num menu
ints	Comm events counter. Counts all incoming and outgoing bytes
rxchs	Received bytes counter.
txchs	Transmitted bytes counter
rtgms	Received, complete telegrams to this slave counter
ttgms	Transmitted telegrams from this slave counter
rENQs	Not used for Modbus.
tENQs	Not used for Modbus
rNAKs	Not used for Modbus
tNAKs	Transmitted NAK counter. Telegrams to this node with badly formatted data, requesting non-existing registers or writing to write-protected registers
rACKs	Not used in Modbus
tACKs	Not used in Modbus
NotMe	Counter for Received telegrams intended for other slaves
rxe	Received telegrams in error counter
rxuae	Received bytes with UART errors counter
rxlua	Last encountered UART error. See note 1.
rxcse	Received telegrams with CRC16 error counter
rxfme	Received telegrams with format error counter

Label	Meaning
rxlfm	Last format error encountered. See note 2.
txcme	Non-supported command received counter
timee	Comm timeouts counter
addr	First register in received command. Should toggle between 16 and 44
size	Number of registers in received command. Should toggle between 28 and 4
cmd	Modbus command received. Should toggle between 3 and 16
subf	Subfunction in diagnostics command. Only 0, (Return Query Data) supported.
retST	Exception status of an received command causing a NAK
RX	Received telegram. First 19 bytes in HEX format
TX	Transmitted telegram from this node. First 19 bytes in hex format.
Gpc0x	Internal troubleshooting counters. No useful information.

Note 1 This is the UART status register, bit encoded. Bit 0: Not Used. Bit 1: Overrun error. Bit 2: Parity error. Bit 3: Framing error. Bit 4: Break detected.

Note 2 Format errors have a decimal numerical value:

1. Unsupported Modbus command
2. Read Holding Register telegram not 6 bytes long
3. Trying to read from non-existing registers
4. Read Diagnostic telegram not 8 byres long
5. Unsupported subfunction in Read Diagnostics telegram
6. Trying to write to non-existing registers
7. Byte count field disagrees with length field in Preset Multiple Registers command
8. Telegram length disagrees with length field in Preset Multiple Registers command
9. Trying to write to read-only registers
10. Unknown Modbus command
11. MC³ Reciever buffer overrun - more than 255 bytes in telegram.
12. Linefeed not following Carriage Return in Modbus ASCII telegram
13. Bytes received after complete telegram, before telegram interpretation (too fast).
14. Should never happen… Unknown receiver state.
15. Should never happen… Transmitter buffer overrun.

Use the integrity bit.

In the PLC, toggle the integrity bits (Word [44], bit 7) every 2 seconds. Monitor the integrity echo bit (Word [16] bit 7). If they stop toggling, communications has failed, and appropriate steps can be taken. The integrity bit can be monitored in the MC³ Data Table screen.