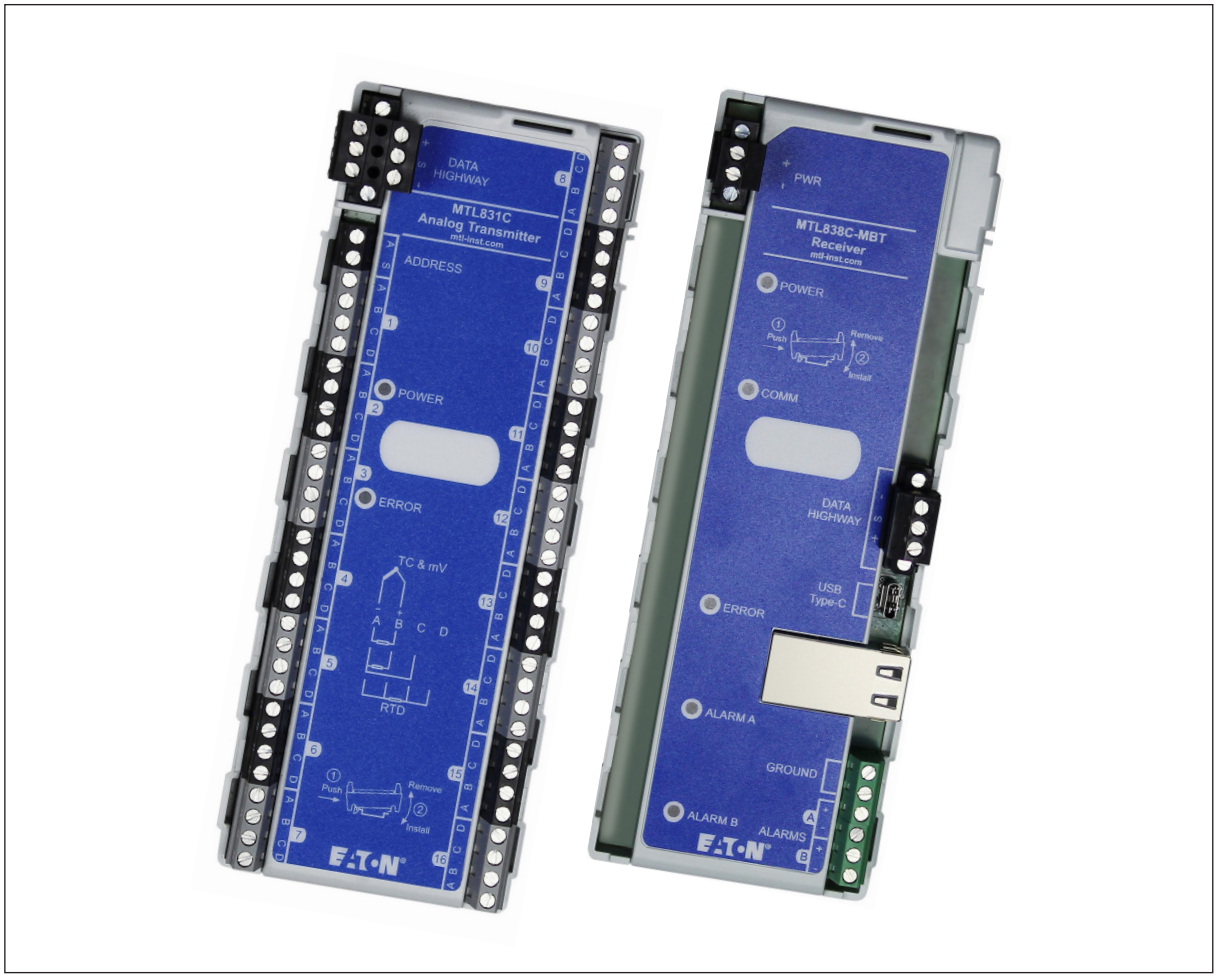
Instruction manual
MTL fieldbus networks

February 2021
INM MTL838C-MBF Rev 4

CROUSE-HINDS SERIES

MTL838C

Modbus Implementation



Powering Business Worldwide

This page left intentionally blank

CONTENTS

1 INTRODUCTION

This manual is principally intended for instrumentation engineers and technicians who need to configure the communications between Modbus system hosts and MTL838C or MTL838C-MBT multiplexer receivers. Since both receivers use the Modbus protocol, MTL838C will be used to refer to both models unless otherwise stated.

The manual provides comprehensive information on the Modbus® protocol, describes the communication between the MTL838C and the host, and provides detailed information relating to the functions of the MTL838C. No previous knowledge of Modbus is assumed.

The JBUS® protocol is also supported by the MTL838C. JBUS is virtually identical to Modbus apart from a slight difference in the addressing of slaves, and this manual may be used for both protocols. The difference in slave addressing is explained in the relevant section.

The manual is divided into chapters which can be summarized as follows:

QuickStart Guide

This describes the commissioning of a simple system with the most commonly used settings.

Background to the MTL838C

Essential information for configuration and maintenance of MTL830 Multiplexer Systems.

Modbus functions supported by the MTL838C

A detailed description of the Modbus functions recognized by the MTL838C. This is to enable users to select the most appropriate function for the Modbus master.

Exception responses supported by the MTL838C

This covers a range of diagnostics for the more advanced user.

Input Status Flags and Registers

Input status flag and register location required for configuration of Modbus master.

Coil Status Flags

Mainly for advance users considering configuration of the MTL838C from the Modbus master- a method that is not really recommended.

Holding Registers

Also for advance users considering configuration of the MTL838C from the Modbus master- a method that is not really recommended.

MTL838C Exception Responses

Interpretation of exception responses for advanced users.

Scaling

Points to consider for selecting scaling parameters within the MTL838C.

2 QUICKSTART GUIDE

This quickstart guide is written for an MTL830C system based on an MTL831C temperature input multiplexer transmitter with an MTL838C multiplexer receiver.

Before actual installation, it is recommended that new users initially set up a simple system on the bench to become familiar with the MTL830C system. The minimum hardware required for a test system is as follows:

MTL831C	Multiplexer transmitter.
MTL838C	Multiplexer receiver.
MTL5553	Isolator (for hazardous area installations only).

In order to run a test the following equipment will be required:

A PC loaded with the MTL838C Configuration Software.

Power supply 24V @ 200mA, together with suitable cabling for the following requirements:

Data highway connections (see INM831C / INM838C).

Power supply connections.

USB cable.

The user will also need the following documentation for wiring information:

INM831C MTL831C installation manual.

INM838C MTL838C installation manual.

INM838C-MBT MTL838C installation manual

Connect at least one sensor to an MTL831C.

Refer to the MTL838C Configuration Software Manual to configure and test the system.

3 BACKGROUND TO THE MTL838C

3.1 The analog-input multiplexer system

The MTL838C is an analog multiplexer receiver that is used with the MTL831C hazardous area millivolt input multiplexer transmitter. The status of up to 32 analog inputs may be communicated from the hazardous area to the safe area via a data highway, comprised of a simple twisted pair- over distances up to 2km.

Each data highway must be protected by an MTL5553 digital isolator when the inputs are located in a Zone 0 or 1 hazardous area. The MTL831C is typically used with thermocouple and RTD inputs and is intrinsically safe. It can be mounted in a Zone 0 or 1 hazardous area and will accept 16 inputs. For systems that do not require Zone 0 or Zone 1 installation, the MTL5553 can be eliminated.

Up to two MTL831C transmitters can be combined on a single MTL838C receiver input- up to a total of 32 analog inputs- as shown in Figure 1.

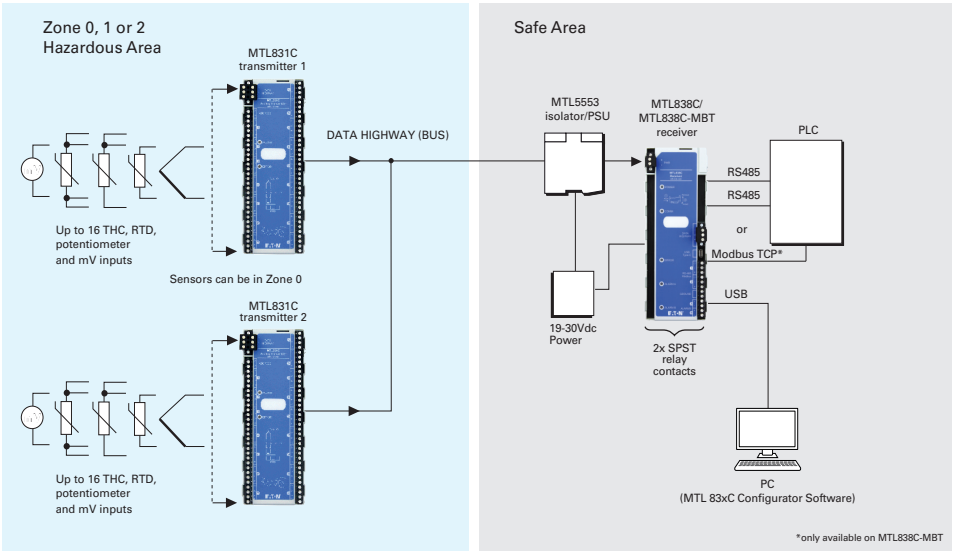


Figure 1- MTL838C/MTL831C System Diagram

The MTL838C acts as a Modbus slave. It may be connected into any standard Modbus network, with up to 31 MTL838C slaves on each network. If each unit has its full complement of 32 analog inputs, the status of a total of 992 analog inputs may be passed to a Modbus master using a single RS485 network. For the MTL838C-MBT the single Ethernet connection allows for up to 32 analog inputs.

3.2 Configuring the MTL838C

The MTL838C must first be configured using software on a PC and the USB connection. This configures things such as the slave address and communication parameters. After the initial configuration, the MTL838C is ready to communicate with the Modbus host. At this point, the remaining configuration may be done in one of two ways:

- on-line via the Modbus link, direct from the host
- off-line using the PC software and USB connection

Using the PC software is required for initial configuration and recommended for first time configuration of the measuring channels.

3.3 On-line Configuration

Configuring the unit via the Modbus master and the network might seem to be the simplest method at first sight, but there are a number of practical difficulties with this configuration technique. This approach means that the user must deal with a number of complex aspects which require a significant investment of the configurer's time before they are understood fully. A further difficulty may be a lack of the necessary memory space within the Modbus master. If the configuration is likely to be changed frequently it could even be necessary for the system designer to design specific 'user interface' screens, such as those used by the PC software, to allow changes to be made by operators. This would be a time consuming and costly task.

For most users, the attraction of being able to use the Modbus master to configure the unit is that the configuration can be re-sent if the slave's memory becomes corrupted. Whilst this is true, it is not possible to avoid the difficulties (and costs) outlined earlier and the decision to adopt a strategy of configuring via the Modbus master should be arrived at only after due consideration.

A cost effective compromise would be to perform the initial configuration via the PC software, and then read the configuration parameters stored in the MTL838C via the host. The stored parameters could then be re-written to the MTL838C should the configuration database ever become corrupt.

If a user intends to adopt the on-line configuration method, the calculation of configuration parameters for storage in the master can be simplified, and the possibility of 'human error' reduced, by using the PC software to input the required data and data format, and then reading the stored values (encoded correctly in the required data format) back from the MTL838C via Modbus. The user should still realize that any subsequent alterations of the parameters will require further use of the PC software.

3.4 Off-line Configuration

Off-line configuration requires the use of the PC software briefly described below. Once configured, the configuration parameters are stored in non-volatile memory within the MTL838C.

3.5 The PC software

By far the simplest method of configuring the MTL838C is using the PC software. This software has been specifically designed to perform all of the complex calculations that must be carried out, in order to configure the unit. These calculations are transparent to the user, and this method provides a convenient and time efficient method.

Alternatively, as explained before, the master could read the configured parameters after initial off-line configuration and these may then be stored within the host for use in the event of a database failure.

3.6 Interconnection of the MTL838C

The MTL838C may be connected to a Modbus host in a number of ways—as was mentioned earlier it may be connected for multi-drop or point-to-point operation.

Two RS485 ports, 1 and 2, are provided on the MTL838C. As there are two ports the unit can either be connected to a single Modbus master, with dual redundancy, or connected to two separate Modbus hosts.

The MTL838C will respond on whichever RS485 connection the query is received, and there is no restriction placed on the simultaneous use of both interfaces. The slave address for each RS-485 port is set using the PC Software.

For the MTL838C-MBT there is only one Ethernet connection over which all Modbus TCP traffic flows to/from the MTL838C-MBT at it's given IP address. Section 7 will cover the configuration of the Ethernet module.Connect the Ethernet port to your Modbus TCP network using a standard Ethernet cable (RJ45 connections). Configuration of the Ethernet port on the MTL838C-MBT is covered in Section 7 of the INM PC Modbus manual.

3.7 Initialization mode

The MTL838C has two distinct modes of operation - normal and initialization.

It will always enter initialization mode during power-up. It can also be triggered by the detection of internal hardware or software faults, or after receiving an instruction from the host to reset some or all of the configuration registers.

During initialization, the unit will ignore all commands from the master.

The initialization period will take 1 or 2 seconds to complete all the necessary operations and calculations. Following successful initialization, the unit will automatically enter, or return to, normal operation mode.

If a corrupted configuration database is detected during initialization the unit will revert to a set of default values, and on entering normal operation mode, will issue exception responses when requested by the host to read input values. Exception responses will continue to be issued until the unit is re-configured. The need to re-configure the unit will remain even if the MTL838C is powered down and back up.

If a corrupted configuration is detected, the slave address may be reset. If this occurs, the user must use the PC software to set the slave address.

3.8 Slave, Transmitter and Input addressing

The following discusses the allocation of addresses to the slaves on the Modbus network- including the MTL838C - and the allocation of addresses for the transmitters and inputs connected to each MTL838C.

3.8.1 Addressing MTL838C slaves

Modbus allows slave addresses in the range 1 to 247. JBUS allows slave addresses in the range 1 to 255. This is the only difference between the two protocols. Since the MTL838C can only have addresses in the range 1 to 31, it will work equally well with either protocol.

The Modbus address for each MTL838C slave is set via the PC software. For reasons of security, it is not possible to set the address of the slave via the Modbus host.

The address for each RS485 port on the MTL838C may be set from 1 to 255. This facility allows the MTL838C to be connected to the same master twice or to two different masters independently. There is no restriction regarding simultaneous communication on both ports. The unit will respond via the port on which it received the query.

Modbus TCP (Ethernet) does not allow multiple slaves on the same IP address. For this reason the address becomes somewhat unimportant except that it is still used. The address on the MTL838C-MBT which is set using the PC software must match what the Master is using.

3.8.2 Addressing the transmitters of each MTL838C

Each MTL831C transmitter accepts up to 16 sensor inputs and there can be one or two MTL831C transmitters connected to a single MTL838C. The address of the MTL831C as seen by the MTL838C is determined by whether a jumper wire is installed on the MTL831C. A jumper wire not installed gives the MTL831C an address of '1' and a jumper installed gives it an address of '2'. The following shows the sensor numbers used by the MTL838C for a given MTL831C address.

MTL831C Address	Sensor Numbers
1	0 - 15
1	32 CJC
2	16- 31
2	33 CJC

Addressing of the MTL831C transmitters affects which sensor is given which address in the MTL838C. For example, if only one MTL831C is connected to the MTL838C but its address jumper is installed, it will be at transmitter address '2' and the sensor range will be 16 – 3, 1, 33.

It is also important that with two MTL831C's connected to a single MTL838C, that one and only one of them has the jumper installed. Otherwise they will both be at the same address and communication between the MTL831's and the MTL838C will fail. Likewise, there can be no more than two MTL831C's connected to a single MTL838C.

CJC stands for Cold Junction Compensation and reports the average temperature of the MTL831C circuit board. This is an average of two temperature sensors and should not be used by the Modbus Host to cold junction compensate thermocouples. Instead the thermocouple channels should be compensated by the MTL831C by selecting the correct Input Type.

4 MODBUS FUNCTIONS SUPPORTED BY THE MTL838C

The following section describes the Modbus functions supported by the MTL838C:

CODE	DESCRIPTION
01	Read coil status
02	Read input status
03	Read holding registers
04	Read input registers
05	Force single coil
06	Preset single register
07	Read exception status
08	Diagnostics
16	Preset multiple registers

All other functions in the range 0 to 127 will not be acted upon or will be ignored. In some cases, when functions that are not supported, the MTL838C will respond with an appropriate exception response.

Important Note:

This chapter contains a number of detailed tables that demonstrate the construction of messages passed along the Modbus network. However, most Modbus masters will have a user-interface that "shelters" the user from most of these details, and will only require the slave address, the function code, the initial coil or register location and the number of coils or registers to be read. The reader need not concern themselves with much of the detail presented here.

Some of the values are shown as hex and some as binary. The hex values are given to describe the code or value that must be sent in the query and the response.

For ASCII mode, the communication is shown as hex coded ASCII, which demonstrates the additional level of character transmission required in this mode- actual communication is, of course, as a binary signal. The binary code for transmission in RTU mode is given directly, and it will be seen that this is a simple encoding of the equivalent hex value. Note that the ASCII is shown for completeness but the MTL838C does not support the ASCII mode.

In the body of the text, decimal values are used, so as to be consistent with the numbering of the function codes in Revision 'E' of the Modbus specification. Where the encoding of these decimal values in to hex makes them appear differently in the table, the hex value is given in parenthesis- e.g. function code 10 ('0A' in hex). For simplicity, start, stop and parity bits are ignored throughout.

4.1 READ COIL STATUS (function 01)

The READ COIL STATUS function requests that the slave reads the status of a specified range of its single bit input/output flags and returns these to the master. The range of flags to be read is given in the query, by the master indicating the address of the first flag to be read and then total number of subsequent flags- including the first.

The example in the following table shows the query required to read the status of flags 00001 to 00009 of slave number 20 (14 in hex.). The start address and number of flags to be read are always transmitted as two bytes- most significant bits (MSB) first, followed by the least significant bits (LSB):

FIELD NAME	HEX	ASCII	RTU
Slave address	14	31 34	14
Function	01	30 31	01
Starting address MSB	00	30 30	00
Starting address LSB	00	30 30	00
No. of locations MSB	00	30 30	00
No. of locations LSB	09	30 39	09
Error check	-	LRC	CRC

Note:

Due to the anomaly in the address and flag locations in Modbus, the address is always 1 less than the flag location. Thus flag '10001' is addressed by the hex value '00 00'.

The normal response to a READ COIL STATUS query contains the slave address, the repeated function code, the number of data bytes that are being transmitted in the response, the data bytes themselves and the error check.

The data bytes encode the status of the flags so that the status of the first flag to be read forms the LSB of the first data byte. Subsequent flag states form the next most significant bits of the first byte - thus if the master had requested the status of 8 flags, the data would be transmitted in a single data byte, with the LSB being the status of the first flag and the MSB being the status of the eighth. This is continued so that the status of the ninth flag requested forms the LSB of the second data byte.

If the master requests the status of a number of flags so that it is not possible to return 'complete' 8-bit data bytes (e.g. if the master requests the status of 9 flags, as above, which would require one complete 8-bit byte and a single bit), then the last data byte to be transmitted is 'packed' with '0's in its MSBs.

The convention followed for the status is: 1 = ON; 0 = OFF.

A response to the query above would have the following format:

FIELD NAME	HEX	ASCII	RTU
Slave address	14	31 34	14
Function	01	30 31	01
No. of Data Bytes	02	30 32	02
First Data Byte	XX	XX XX	XX
Second Data Byte	0X	30 XX	0X
Error check	-	LRC	CRC

Notes:

1. The seven most significant bits of the second data byte in RTU mode are zero, as the query only requested the status of 9 inputs. The seven zeros were packed in to the response to allow the slave to return complete 8-bit data bytes. The same packing of zeros takes place in ASCII mode, which will result in the ASCII characters returned being 30 XX.
2. The 'byte count' in the data field of the response shows the number of bytes returned in RTU mode, and half the number returned in ASCII.
3. The possible ranges for the elements of the query and response for the MTL838C are:

   slave address 1 to 63

   number of locations that may be read 1 to 512

   number of data bytes returned 1 to 64

4.2 READ INPUT STATUS (function 02)

The READ INPUT STATUS function requests that the slave reads the status of a specified range of its single bit output flags and returns these to the master. The range of inputs to be read is given in the query, by the master indicating the address of the first input to be read and then total number of subsequent flags - including the first.

The example below shows the query required to read the status of flags 10001 to 10030 of slave number 17 (11 in hex). The start address and number of flags to be read are always transmitted as two bytes - most significant bits (MSB) first, followed by the least significant bits (LSB):

FIELD NAME	HEX	ASCII	RTU
Slave address	11	31 31	11
Function	02	30 32	02
Starting address MSB	00	30 30	00
Starting address LSB	00	30 30	00
No. of locations MSB	00	30 30	00
No. of locations LSB	1E	30 45	1E
Error check	-	LRC	CRC

Note:

due to the anomaly in the addresses and input status locations in Modbus, the address is always 1 less than the status location. Thus input '10001' is addressed by the hex value '00 00'.

The normal response to a READ INPUT STATUS comprises the slave address, the repeated function code, the number of data bytes that are being transmitted in the response, the data bytes themselves and the error check.

The data bytes encode the status of the inputs so that the status of the first input to be read forms the LSB of the first data byte. Subsequent input states form the next most significant bits of the first byte- thus if the master had requested the status of 8 inputs, the data would be transmitted in a single data byte, with the LSB being the status of the first input and the MSB being the status of the eighth. This is continued so that the status of the ninth input requested forms the LSB of the second data byte.

If the master requests the status of a number of inputs so that it is not possible to return 'complete' 8-bit data bytes (e.g. if the master requests the status of 9 inputs, which would require one complete 8-bit byte and a single bit), then the last data byte to be transmitted is 'packed' with '0's in its MSBs.

The convention followed for the status is: 1 = ON; 0 = OFF.

A response to the query above would have the following format:

FIELD NAME	HEX	ASCII	RTU
Address	11	31 31	11
Function	02	30 32	02
Number of bytes Slave returned	04	30 34	04
First data byte	XX	XX XX	XX
Second data byte	XX	XX XX	XX
Third data byte	XX	XX XX	XX
Fourth data byte	XX	XX XX	00XX XXXX
Error check	-	LRC	CRC

Notes:

1. The two most significant bits of the fourth data byte in RTU mode are zero, as the query only requested the status of 30 inputs. The two zeros were packed in to the response to allow the slave to return complete 8-bit data bytes. The same packing of zeros takes place in ASCII mode, but the value returned in ASCII cannot be determined without knowing the status of the last few data bits.
2. The 'byte count' in the data field of the response shows the number of bytes returned in RTU mode, and half the number returned in ASCII.
3. The possible ranges for the elements of the query and the response from the MTL838C are:

slave address	1 to 63
number of locations that may be read	1 to 512
number of data bytes returned	1 to 64

4.3 READ HOLDING REGISTERS (function 03)

The READ HOLDING REGISTERS function requests that the slave reads the binary contents of a specified range of its 16-bit holding registers and returns the values to the master. The range of registers to be read is given in the query, by the master indicating the address of the first register and the total number of subsequent registers to be read- including the first register.

The example below shows the query required to read the values held in holding registers 40108 to 40110 from slave 17 (108 and 17 are 6C and 11 in hex).

FIELD NAME	HEX	ASCII	RTU
Slave address	11	31 31	11
Function	03	30 33	03
Starting address MSB	00	30 30	00
Starting address LSB	6B	36 4B	6B
No. of registers MSB	00	30 30	00
No. of registers LSB	03	30 33	03
Error check	-	LRC	CRC

Note:

Due to the anomaly in the addresses and register locations in Modbus, the address is always 1 less than the register location. Thus register '40108' is addressed as 0107 (hex value '6B').

The normal response to a READ HOLDING REGISTERS query comprises the slave address, the repeated function code, the number of data bytes that are being transmitted in the response, the data bytes themselves and the error check.

The data bytes encode the contents of the holding registers as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second byte the low order bits.

A response to the query above would have the following format:

FIELD NAME	HEX	ASCII	RTU
Slave address	11	31 31	11
Function	03	30 33	03
Number of bytes returned	06	30 36	06
First data byte (MSB 40108)	XX	XX XX	XX
Second data byte (LSB 40108)	XX	XX XX	XX
Third data byte (MSB 40109)	XX	XX XX	XX
Fourth data byte (LSB 40109)	XX	XX XX	XX
Fifth data byte (MSB 40110)	XX	XX XX	XX
Sixth data byte (LSB 40110)	XX	XX XX	XX
Error check	-	LRC	CRC

Notes:

1. The 'byte count' in the data field of the response shows the number of bytes returned in RTU mode, and half the number returned in ASCII.
2. The possible ranges for the elements of the query and the response from the MTL838C are:

slave address	1 to 63
number of registers that may be read	1 to 60
number of data bytes returned (2x the number of registers)	2 to 120

4.4 READ INPUT REGISTERS (function 04)

The READ INPUT REGISTERS function requests that the slave reads the binary contents of a specified range of its 16-bit input registers and returns the values to the master. The range of inputs to be read is given in the query, by the master indicating the address of the first register and the total number of subsequent registers to be read- including the first register.

The example below shows the query required to read the values held in input register 30009 from slave 31 (1F in hex).

FIELD NAME	HEX	ASCII	RTU
Slave address	1F	31 46	1F
Function	04	30 34	04
Starting address MSB	00	30 30	00
Starting address LSB	08	30 38	08
No. of points MSB	00	30 30	00
No. of points LSB	01	30 31	01
Error check	-	LRC	CRC

Note:

Due to the anomaly in the addresses and register locations in Modbus, the address is always 1 less than the register location. Thus register '30009' is addressed by the hex. value '00 08'.

The normal response to a READ INPUT REGISTERS query comprises the slave address, the repeated function code, the number of data bytes that are being transmitted in the response, the data bytes themselves and the error check.

The data bytes encode the contents of the input registers as two bytes per register. For each register, the first byte contains the high order bits and the second byte the low order bits.

A response to the query above would have the following format:

FIELD NAME	HEX	ASCII	RTU
Slave address	1F	31 46	1F
Function	04	30 34	04
Number of bytes returned	02	30 32	02
First data byte (MSB 30009)	XX	XX XX	XX
Second data byte (LSB 30009)	XX	XX XX	XX
Error check	-	LRC	CRC

Notes:

1. The 'byte count' in the data field of the response shows the number of bytes returned in RTU mode, and half the number returned in ASCII. .
2. The possible ranges for the elements of the query and the response from the MTL838C are:

slave address	1 to 63
number of registers that may be read	1 to 60
number of data bytes returned (2x the number of registers)	2 to 120

4.5 FORCE SINGLE COIL (function 05)

The FORCE SINGLE COIL function requests that the slave sets a specified input/output flag to a particular status. The address of the flag to be set is given in the query. The status to which the flag must be set is provided by two data bytes. If the flag is to be set to '1', then the data bytes sent are FF 00. If the flag is to be set to '0' the data bytes are 00 00.

The example below shows the query required to force the status of a flag with address 10065 of slave 18 to '1'. (65 and 18 are '41' and '12' in hex).

FIELD NAME	HEX	ASCII	RTU
Slave address	12	31 32	12
Function	05	30 35	05
Flag address MSB	00	30 30	00
Flag address LSB	40	34 30	40
Force data MSB	FF	46 46	FF
Force data LSB	00	30 30	00
Error check	-	LRC	CRC

The normal response to a FORCE SINGLE COIL query comprises the slave address, an echo of the function code, echoes of the flag address and status request, and an error check.

A response to the query above would have the following format:

FIELD NAME	HEX	ASCII	RTU
Slave address	12	31 32	12
Function	05	30 35	05
Flag address MSB	00	30 30	00
Flag address LSB	40	34 30	40
Force data MSB	FF	46 46	FF
Force data LSB	00	30 30	00
Error check	-	LRC	CRC

The possible ranges for the elements of the query and the response from the MTL838C are:

slave address	1 to 63
coil address	0000 to 65535
data bytes returned	FF00 or 0000 - as query

4.6 PRESET SINGLE REGISTER (function 06)

The PRESET SINGLE REGISTER function requests that the slave writes specified data in to a particular register. The address of the register to be written to is given in the query. The data to be written is provided by two data bytes.

The example below shows the query required to 'pre-set' or write a register so that it holds the value 'FF FF'. The register location is 40003 of slave 1.

FIELD NAME	HEX	ASCII	RTU
Slave address	01	30 31	01
Function	06	30 36	06
Register address MSB	00	30 30	00
Register address LSB	02	30 32	02
Pre-set data MSB	FF	46 46	FF
Pre-set data LSB	FF	46 46	FF
Error check	-	LRC	CRC

The normal response to a PRESET SINGLE REGISTER query comprises the slave address, an echo of the function code, echoes of the register address and the pre-set data, and an error check.

A response to the query above would have the following format:

FIELD NAME	HEX	ASCII	RTU
Slave address	01	30 31	01
Function	06	30 36	06
Register address MSB	00	30 30	00
Register address LSB	02	30 32	02
Pre-set data MSB	FF	46 46	FF
Pre-set data LSB	FF	46 46	FF
Error check	-	LRC	CRC

The possible ranges for the elements of the query and the response from the MTL838C are:

slave address	1 to 63
coil address	0 to 65535
data bytes returned	0 to 65535 - as query

4.7 READ EXCEPTION STATUS (function 07)

The READ EXCEPTION STATUS function requests that the slave reads the contents of eight status bits within the slave and returns their values to the master. The registers that are used to store these exception status bits are pre-defined, so that the command itself is sufficient to locate the required locations.

In the MTL838C, the eight bits that are read correspond to the least significant bits of the STATUS input register 30005.

The example below shows the query required to read the exception status values for slave 10 (0A in hex).

FIELD NAME	HEX	ASCII	RTU
Slave address	0A	30 41	0A
Function	07	30 37	07
Error check	-	LRC	CRC

The normal response to a READ EXCEPTION STATUS query comprises the slave address, the repeated function code, a single data byte and the error check.

The data byte encodes the contents of the exception status bits in binary format, with the status of the lowest bit as the LSB of the byte.

A response to the query above would have the following format:

FIELD NAME	HEX	ASCII	RTU
Slave address	0A	30 41	0A
Function	07	30 37	07
Exception status data	XX	XX XX	XX
Error check	-	LRC	CRC

4.8 DIAGNOSTICS (function 08)

The DIAGNOSTICS function has a number of tests to check the communication link between the master and the slave. The subfunction code is transmitted as the first two bytes of data following a '08' function code in the query.

Some of the tests specified by the subfunctions require the slave to return data in the response to the query, others only require the slave to acknowledge receipt of the response in the normal way. Responses to diagnostic functions will return a repetition of the subfunction code as well as the '08' function.

Most of the diagnostic subfunctions supported by the MTL838C are defined so that the query must include two data bytes packed with zeros, immediately following the subfunction code.

A large number of diagnostic codes are specified in revision 'E' of Modbus, not all of which are supported by the MTL838C. The ones supported are listed below:

CODE	DIAGNOSTIC SUBFUNCTION
00 00	Return query data
00 02	Return diagnostic register
00 10	Clear contents and diagnostic registers
00 11	Return bus message count
00 12	Return bus comms error count
00 13	Return bus exception count

The application of each of these subfunctions is discussed in detail in the following sections.

4.9 RETURN QUERY DATA (subfunction 00 00)

The diagnostic subfunction RETURN QUERY DATA requests the addressed slave to return (loop back) an exact copy of the data contained in the query, to the master, via the response.

An example of a query and response with this subfunction is given below. The master requests slave number 4 to return the hexadecimal data 'AA BB'.

The query:

FIELD NAME	HEX	ASCII	RTU
Slave address	04	30 34	04
Function	09	30 38	08
Subfunction MSB	00	30 30	00
Subfunction LSB	00	30 30	00
Data MSB	AA	41 41	AA
Data LSB	BB	42 42	BB
Error check	-	LRC	CRC

The response:

FIELD NAME	HEX	ASCII	RTU
Slave address	04	30 34	04
Function	08	30 38	08
Subfunction MSB	00	30 30	00
Subfunction LSB	00	30 30	00
Data MSB	AA	41 41	AA
Data LSB	BB	42 42	BB
Error check	-	LRC	CRC

4.10 RETURN DIAGNOSTIC REGISTER (subfunction 00 02)

The diagnostic subfunction RETURN DIAGNOSTIC REGISTER requests that the slave reads the contents of the diagnostic register and returns the binary data values to the master.

The query sends two zero data bytes, following the data bytes containing the subfunction code. The response returns two 8-bit data bytes containing the register data.

The contents of the diagnostic register may be defined by the manufacturer, according to the needs of each Modbus slave. For the MTL838C, the response to this query returns the content one of the registers that contain the STATUS data. The register returned is number 30006.

The following example shows the query and response generated when the master requests diagnostic subfunction 00 02 from the slave with address 12 ('0C' in hex).

The query:

FIELD NAME	HEX	ASCII	RTU
Slave address	0C	30 43	0C
Function	08	30 38	08
Subfunction MSB	00	30 30	00
Subfunction LSB	02	30 32	02
Data MSB	00	30 30	XX
Data LSB	00	30 30	XX
Error check	-	LRC	CRC

The response:

FIELD NAME	HEX	ASCII	RTU
Slave address	0C	30 43	0C
Function	08	30 38	08
Subfunction MSB	00	30 30	00
Subfunction LSB	02	30 32	02
Data MSB	XX	XX XX	XX
Data LSB	XX	XX XX	XX
Error check	-	LRC	CRC

4.11 CLEAR COUNTERS AND DIAGNOSTIC REGISTERS (subfunction 00 10)

The diagnostic subfunction CLEAR COUNTERS AND DIAGNOSTIC REGISTERS requests that the slave clears a number of registers of their current values. In some slaves the function is as expected, and both counter and diagnostic registers are cleared. In some slaves (and the MTL838C is one of these) this subfunction only clears the counter registers and leaves the diagnostic registers untouched.

The query sends two zero data bytes, following the data bytes containing the subfunction code, and this is echoed in the response.

The following example shows the query and response generated when the master requests diagnostic subfunction 00 10 ('00 0A' in hex.) from the slave with address 02.

The query:

FIELD NAME	HEX	ASCII	RTU
Slave address	02	30 32	02
Function	08	30 38	08
Subfunction MSB	00	30 30	00
Subfunction LSB	0A	30 41	0A
Data MSB	00	30 30	00
Data LSB	00	30 30	00
Error check	-	LRC	CRC

The response:

FIELD NAME	HEX	ASCII	RTU
Slave address	02	30 32	02
Function	08	30 38	08
Subfunction MSB	00	30 30	00
Subfunction LSB	0A	30 41	0A
Data MSB	00	30 30	00
Data LSB	00	30 30	00
Error check	-	LRC	CRC

4.12 RETURN BUS MESSAGE COUNT (subfunction 00 11)

The diagnostic subfunction RETURN BUS MESSAGE COUNT requests that the slave returns to the master the contents of a register that is used to count the number of messages that the slave has detected on the system since its last restart, its last clear counters instruction, or since power-up - whichever was the most recent.

The query sends two zero data bytes, following the data bytes containing the subfunction code. The response returns two 8-bit data bytes containing the register data.

The following example shows the query and response generated when the master requests diagnostic subfunction 00 11 ('00 0B' in hex.) from the slave with address 13 ('0D' in hex.).

The query:

FIELD NAME	HEX	ASCII	RTU
Slave address	0D	30 44	0D
Function	08	30 38	08
Subfunction MSB	00	30 30	00
Subfunction LSB	0B	30 42	0B
Data MSB	00	30 30	00
Data LSB	00	30 30	00
Error check	-	LRC	CRC

The response:

FIELD NAME	HEX	ASCII	RTU
Slave address	0D	30 44	0D
Function	08	30 38	08
Subfunction MSB	00	30 30	00
Subfunction LSB	0B	30 42	0B
Message count MSB	XX	XX XX	XX
Message count LSB	XX	XX XX	XX
Error check	-	LRC	CRC

4.13 RETURN BUS COMMUNICATION ERROR COUNT (subfunction 00 12)

The diagnostic subfunction RETURN BUS COMMUNICATION ERROR COUNT requests that the slave returns to the master the contents of a register that is used to count the number of CRC (or LRC) errors that the slave has detected on the system since its last restart, its last clear counters instruction, or since power-up- whichever was the most recent.

The query sends two zero data bytes, following the data bytes containing the subfunction code. The response returns two 8-bit data bytes containing the register data.

The following example shows the query and response generated when the master requests diagnostic subfunction 00 12 ('00 0C' in hex.) from the slave with address 08.

The query:

FIELD NAME	HEX	ASCII	RTU
Slave address	08	30 38	08
Function	08	30 38	08
Subfunction MSB	00	30 30	00
Subfunction LSB	0C	30 43	0C
Data MSB	00	30 30	00
Data LSB	00	30 30	00
Error check	-	LRC	CRC

The response:

FIELD NAME	HEX	ASCII	RTU
Slave address	08	30 38	08
Function	08	30 38	08
Subfunction MSB	00	30 30	00
Subfunction LSB	0C	30 43	0C
Data MSB	XX	XX XX	XX
Data LSB	XX	XX XX	XX
Error check	-	LRC	CRC

4.14 RETURN BUS EXCEPTION ERROR COUNT (subfunction 00 13)

The diagnostic subfunction RETURN BUS EXCEPTION ERROR COUNT requests that the slave returns to the master the contents of a register that is used to count the number of exception errors (i.e. the number of times the slave has issued exception responses) that the slave has returned since its last restart, its last clear counters instruction, or since power-up- whichever was the most recent.

The query sends two zero data bytes, following the data bytes containing the subfunction code. The response returns two 8-bit data bytes containing the register data.

The following example shows the query and response generated when the master requests diagnostic subfunction 00 13 ('00 0D' in hex.) from the slave with address 06.

The query:

FIELD NAME	HEX	ASCII	RTU
Slave address	06	30 36	06
Function	08	30 38	08
Subfunction MSB	00	30 30	00
Subfunction LSB	0D	30 44	0D
Data MSB	00	30 30	00
Data LSB	00	30 30	00
Error check	-	LRC	CRC

The response:

FIELD NAME	HEX	ASCII	RTU
Slave address	06	30 36	06
Function	08	30 38	08
Subfunction MSB	00	30 30	00
Subfunction LSB	0D	30 44	0D
Data MSB	XX	XX XX	XX
Data LSB	XX	XX XX	XX
Error check	-	LRC	CRC

4.15 PRESET MULTIPLE REGISTERS (function 16)

The PRESET MULTIPLE REGISTERS function requests that the slave writes specified data in to a range of registers. The range of registers to be written is identified by the master which indicates the location of the first register and then the total number of registers to be written - including the first.

The example below shows the query required to 'pre-set' or write two registers so that they both contain the value 'FF FF'. The first register location is 40003 of slave 1. Function 16 is '10' in hex.

FIELD NAME	HEX	ASCII	RTU
Slave address	01	30 31	01
Function	10	31 30	10
Register address MSB	00	30 30	00
Register address LSB	02	30 32	02
Register number MSB	00	30 30	00
Register number LSB	02	30 32	02
Number of bytes to follow	04	30 34	04
1st preset data MSB	FF	46 46	FF
1st preset data LSB	FF	46 46	FF
2nd preset data MSB	FF	46 46	FF
2nd preset data LSB	FF	46 46	FF
Error check	-	LRC	CRC

The normal response to a PRESET MULTIPLE REGISTERS query comprises the slave address, an echo of the function code, echoes of the register address and number of registers written, and an error check.

A response to the query above would have the following format:

FIELD NAME	HEX	ASCII	RTU
Slave address	01	30 31	01
Function	10	31 30	10
Register address MSB	00	30 30	00
Register address LSB	02	30 32	02
Register number MSB	00	30 30	00
Register number LSB	02	30 32	02
Error check	-	LRC	CRC

The possible ranges for the elements of the response from an MTL838C are:

slave address	1 to 63
number of registers	0 to 60

5 EXCEPTION RESPONSES SUPPORTED BY THE MTL838C

An MTL838C slave will issue one of five available exception responses if a message is received correctly (i.e. it passes the error checking), but the slave then finds it is unable to perform the required operation. The following section describes the construction of exception responses in general, and describes in detail those exception responses that are supported by the MTL838C.

The following exception responses are supported by the MTL838C:

ILLEGAL FUNCTION	Response 01
ILLEGAL DATA ADDRESS	Response 02
ILLEGAL DATA VALUE	Response 03
SLAVE DEVICE FAILURE	Response 04
NEGATIVE ACKNOWLEDGE	Response 07

Note that if a slave receives a message which does not pass the error checking employed, it will discard the message and will not issue a response. This prevents a slave from carrying out operations that have either not been translated correctly or which were intended for another slave. The master employs a 'time-out' check, and if it has not received a response after a given time period, it will re-try or take other appropriate action.

5.1 Construction of exception responses

In a normal response, the slave exactly echoes the function code received from the master. In an exception response the slave returns to the master an echo of the function code received, but with its MSB set to '1'. The master can therefore identify that an exception response is being returned, and identify the function code that was received by the slave. This is possible as there are less than 128 (or 80 hex) function codes defined which, as binary 8-bit numbers, must always have a '0' as their MSB.

The example below shows the first few bytes of an exception response issued after slave 9 correctly received a function code '01' that it was then unable to perform.

FIELD NAME	HEX	ASCII	RTU
Slave address	09	30 39	09
Function (an exception for 01)	81	38 31	81

The exception response to function code '01' is perhaps most easily understood when examining the result in RTU mode, where it is clear that the MSB has been set to '1'.

Further data is passed to the master via the first bytes of the response's data field. The bytes returned are referred to as the 'exception code' and these are used to provide the master with additional information regarding the nature of the exception.

The exception code that is generated by the slave for any particular event can be determined by the manufacturer of the device. It is normal, however, to try and use the exception codes so that the code name is as near as possible, in meaning, to the event that has caused the exception.

The example below shows the full exception response for the example used earlier- with the reason for the exception being identified as exception code '02'. This is the 'ILLEGAL DATA ADDRESS', which would typically be used if the master had requested the slave to read a non-existent status location.

FIELD NAME	HEX	ASCII	RTU
Slave address	09	30 39	09
Function (an exception for 01)	81	38 31	81
Exception code	02	30 32	02
Error check	-	LRC	CRC

The sections below describe the exception codes supported by the MTL838C in detail.

5.2 ILLEGAL FUNCTION (exception code 01)

The ILLEGAL FUNCTION exception code is used to inform the master that the function code received by the slave is not an allowable function for that slave.

An example of such a request would be the function code FORCE MULTIPLE COILS (function 15) sent to an MTL838C. This function is not supported by the MTL838C (because the design of the device does not require groups of coils to be set at a given moment). If the master were to send a request containing such a function code, an exception code '01' would be returned. The example below shows the exception response returned by such a slave, with address '04':

FIELD NAME	HEX	ASCII	RTU
Slave address	04	30 34	04
Function (an exception for 01)	95	39 35	95
Exception code	01	30 31	01
Error check	-	LRC	CRC

5.3 ILLEGAL DATA ADDRESS (exception code 02)

The ILLEGAL DATA ADDRESS exception code is used to inform the master that an address used in the query is not available within the slave.

An example of such a request would be the function code READ INPUT REGISTERS ('04') with the number of registers to be read given as 61, sent to an MTL838C. The MTL838C has a communication buffer that is only capable of containing sixty input registers, and a request to read more than this number could not be handled by the unit. The example below shows the exception response returned by such a slave, with address '09':

FIELD NAME	HEX	ASCII	RTU
Slave address	09	30 39	09
Function (an exception for 01)	84	38 34	84
Exception code	02	30 32	02
Error check	-	LRC	CRC

5.4 ILLEGAL DATA VALUE (exception code 03)

The ILLEGAL DATA VALUE exception code is used to inform the master that a value used in the query is not valid for the function requested form that slave.

An example of such a request would be the function code for DIAGNOSTICS with a diagnostic code of '01', sent to an MTL838C. The MTL838C does not support this diagnostic code, and would return the exception code above. The example below shows the exception response returned by such a slave, with address '06':

FIELD NAME	HEX	ASCII	RTU
Slave address	06	30 36	06
Function (an exception for 01)	88	38 38	88
Exception code	03	30 33	03
Error check	-	LRC	CRC

5.5 SLAVE DEVICE FAILURE (exception code 04)

The SLAVE DEVICE FAILURE exception code is used to inform the master that an error occurred in the slave while it was attempting to carry out the action required by the query.

An example of such a failure would be corruption of the configuration data stored by the MTL838C. The MTL838C would return a 'SLAVE DEVICE FAILURE' exception code if the configuration data was found to be corrupted and the master issued a request to READ INPUT REGISTERS '04' for registers that contained input status data. The example below shows the exception response returned by such a slave, with address '03':

FIELD NAME	HEX	ASCII	RTU
Slave address	03	30 33	03
Function (an exception for 01)	84	38 34	84
Exception code	04	30 34	04
Error check	-	LRC	CRC

5.6 NEGATIVE ACKNOWLEDGE (exception code 07)

The NEGATIVE ACKNOWLEDGE exception code is used to inform the master that the slave cannot perform the requested function.

An example of such a failure would be attempting to write data in to a register that was 'write disabled'. The MTL838C has a facility whereby the configuration data may be protected against over-writing. If this facility is used and the Modbus master attempts to write in to the configuration registers, then the exception code '07' will be returned. The following table shows the exception code returned by such a slave, with address 04.

FIELD NAME	HEX	ASCII	RTU
Slave address	04	30 34	04
Function (an exception for 01)	86	38 36	86
Exception code	07	30 37	07
Error check	-	LRC	CRC

6 INPUT STATUS FLAGS AND REGISTERS

The input status flags and input registers are used to store information that the master will want to read from the MTL838C. The data stored by the MTL838C is mapped twice. Once to the input status flags and again to the input registers. The user can choose which of the two data stores is the simplest to read from, given the application in question.

6.1 Mapping of input status flags and input registers

The tables below show the mapping of the flag and register locations used by the MTL838C. The tables show the mappings with IEEE data format selected and with non-IEEE.

Mapping for IEEE data format

INPUT STATUS FLAG LOCATION	INPUT REGISTER LOCATION	NAME	DATA TYPE
10001- 10032	30001- 30002	838_REV	ASCII
10033- 10064	30003- 30004	831_REV	ASCII
10065- 10096	30005- 30006	STATUS	binary
10097- 10128	30007- 30008	Not Used	binary
10129-10160	30009- 30010	HIGH_ALARM	binary
10161- 10192	30011- 30012	LOW_ALARM	binary
10193- 10224	30013- 00014	OPEN_ALARM	binary
10225- 10256	30015- 30016	INPUT_1	IEEE
10257- 10288	30017- 30018	INPUT_2	IEEE
10289- 10320	30019- 30020	INPUT_3	IEEE
10321- 10352	30021- 30022	INPUT_4	IEEE
10353- 10384	30023- 30024	INPUT_5	IEEE
10385- 10416	30025- 30026	INPUT_6	IEEE
10417- 10448	30027- 30028	INPUT_7	IEEE
10449- 10480	30029- 30030	INPUT_8	IEEE
10481- 10512	30031- 30032	INPUT_9	IEEE
10513- 10544	30033- 30034	INPUT_10	IEEE
10545- 10576	30035- 30036	INPUT_11	IEEE
10576- 10608	30037- 30038	INPUT_12	IEEE
10609- 10640	30039- 30040	INPUT_13	IEEE
10640- 10671	30041- 30042	INPUT_14	IEEE
10673- 10704	30043- 30044	INPUT_15	IEEE
10705- 10736	30045- 30046	INPUT_16	IEEE
10737- 10768	30047- 30048	INPUT_17	IEEE
10769- 10800	30049- 30050	INPUT_18	IEEE
10801- 10832	30051- 30052	INPUT_19	IEEE
10833- 10864	30053- 30054	INPUT_20	IEEE
10865- 10896	30055- 30056	INPUT_21	IEEE
10897- 10928	30057- 30058	INPUT_22	IEEE

continued

INPUT STATUS FLAG LOCATION	INPUT REGISTER LOCATION	NAME	DATA TYPE
10929- 10960	30059- 30060	INPUT_23	IEEE
10961- 10992	30061- 30062	INPUT_24	IEEE
10993- 11024	30063- 30064	INPUT_25	IEEE
11025- 11056	30065- 30066	INPUT_26	IEEE
11057- 11088	30067- 30068	INPUT_27	IEEE
11089- 11120	30069- 30070	INPUT_28	IEEE
11121- 11152	30071- 30072	INPUT_29	IEEE
11153- 11184	30073- 30074	INPUT_30	IEEE
11185- 11216	30075- 30076	INPUT_31	IEEE
11217- 11248	30077- 30078	INPUT_32	IEEE
11249- 11280	30079- 30080	CJ1	IEEE
11281- 11312	30081- 30082	CJ2	IEEE

Note:

When addressing the locations of the data given above, remember the anomaly that exists in Modbus, between the address passed by the function and the location within the slave. Thus locations 10001 - 10032 are addressed by the READ INPUT STATUS function with addresses 0000 - 0031, etc.

The contents of each location are explained more fully in the sections below.

Mapping for non-IEEE data format

INPUT STATUS FLAG LOCATION	INPUT REGISTER LOCATION	NAME	DATA TYPE
10001- 10032	30001-30002	838_REV	ASCII
10033- 10064	30003- 30004	831_REV	ASCII
10065- 10096	30005- 30006	STATUS	binary
10097- 10128	30007- 30008	Not Used	binary
10129-10160	30009- 30010	HIGH_ALARM	binary
10161- 10192	30011- 30012	LOW_ALARM	binary
10193- 10224	30013- 30014	OPEN_ALARM	binary
10225- 10240	30015	INPUT_1	non-IEEE
10241- 10256	30016	INPUT_2	non-IEEE
10257- 10272	30017	INPUT_3	non-IEEE
10273-10288	30018	INPUT_4	non-IEEE
10289- 10304	30019	INPUT_5	non-IEEE
10305- 10320	30020	INPUT_6	non-IEEE
10321- 10336	30021	INPUT_7	non-IEEE
10337- 10352	30022	INPUT_8	non-IEEE
10353- 10368	30023	INPUT_9	non-IEEE
10369- 10384	30024	INPUT_10	non-IEEE
10385- 10400	30025	INPUT_11	non-IEEE
10401- 10416	30026	INPUT_12	non-IEEE
10417- 10432	30027	INPUT_13	non-IEEE
10433- 10448	30028	INPUT_14	non-IEEE

continued

10449- 10464	30029	INPUT_15	non-IEEE
10465- 10480	30030	INPUT_16	non-IEEE
10481- 10496	30031	INPUT_17	non-IEEE
10497- 10512	30032	INPUT_18	non-IEEE
10513- 10528	30033	INPUT_19	non-IEEE
10529- 10544	30034	INPUT_20	non-IEEE
10545- 10560	30035	INPUT_21	non-IEEE
10561- 10576	30036	INPUT_22	non-IEEE
10577- 10592	30037	INPUT_23	non-IEEE
10593- 10608	30038	INPUT_24	non-IEEE
10609- 10624	30039	INPUT_25	non-IEEE
10625- 10640	30040	INPUT_26	non-IEEE
10641- 10656	30041	INPUT_27	non-IEEE
10657- 10672	30042	INPUT_28	non-IEEE
10673- 10688	30043	INPUT_29	non-IEEE
10689- 10704	30044	INPUT_30	non-IEEE
10705- 10720	30045	INPUT_31	non-IEEE
10721- 10736	30046	INPUT_32	non-IEEE
10737- 10752	30047	CJ1	non-IEEE
10753- 10768	30048	CJ2	non-IEEE

NOTE

When addressing the locations of the data given above, remember the anomaly that exists in Modbus, between the address passed by the function and the location within the slave. Thus locations 10001- 10032 are addressed by the READ INPUT STATUS function with addresses 0000- 0031, etc.

The contents of each location are explained more fully in the sections below.

6.2 Revision number of 838 software

INPUT STATUS FLAG LOCATIONS: 10001-10032

INPUT REGISTER LOCATIONS: 30001-30002

Four ASCII characters are provided which identify the firmware revision number:

'838_REV' of the firmware running on the MTL838C. This firmware controls the entire MTL838C. The revision is stored in register location 30001 and is a letter (high byte) followed by a number (low byte). Register 30002 is always filled with blanks.

6.3 Revision number of 831 software

INPUT STATUS FLAG LOCATIONS: 10033-10064

INPUT REGISTER LOCATIONS: 30003-30004

Four ASCII characters are provided which identify the firmware revision number:

'831_REV' of the firmware running on the transmitter(s). This firmware is running on the processor in the MTL831C that communicates with the MTL838C. There are two other processors within the MTL831C that make the sensor measurements. The firmware revision for these processors is available only by using the PC software and a USB cable connected to the MTL838C. The first 2 ASCII characters is the revision number for transmitter '1' (30003) and the second two characters are the revision number for transmitter '2' (30004). The revision is a letter (high byte) followed by a number (low byte).

6.4 MTL838C status information

INPUT STATUS FLAG LOCATIONS: 10065- 10096

INPUT REGISTER LOCATIONS: 30005- 30006

A total of 16 STATUS bits are provided which inform the master of the overall status of the MTL838C. The table below shows the meaning of each bit, with the input status flag and the input register locations shown for each bit.

INPUT STATUS FLAG LOCATION	INPUT REGISTER LOCATION	NAME
10096	30006: bit 0	Error flag- set when any of the status bits '8' to '15' are set to '1'
10095	30006: bit 1	Unused – 0
10094	30006: bit 2	Unused – 0
10093	30006: bit 3	Invalid database detected
10092	30006: bit 4	Unused – 0
10091	30006: bit 5	Unused – 0
10090	30006: bit 6	Highway OK
10089	30006: bit 7	Unused – 0
10088	30006: bit 8	Transmitter 1 failed
10087	30006: bit 9	Transmitter 2 failed
10086	30006: bit 10	Unused – 0
10085	30006: bit 11	CJC Range Error
10084	30006: bit 12	CJC Delta Error
10083	30006: bit 13	Open circuit detected on any input
10082	30006: bit 14	Low alarm detected on any input
10081	30006: bit 15	High alarm detected on any input

All status bits are set to logic '1' for the 'true' condition.

The functions 'READ EXCEPTION STATUS' and 'RETURN DIAGNOSTICS REGISTER' are defined so that they read part of the STATUS register. These functions may be more convenient methods of accessing status data.

6.5 'Error flag'

INPUT STATUS FLAG LOCATION: 10096

INPUT REGISTER LOCATION: 30006: bit 0

The 'Error flag' is set when any of the bits '8' to '15' are set to '1'. This single bit then shows that there is a fault with some part of the system. Monitoring this bit alone can allow the Modbus master to maintain a check on the correct operation of the slave and to monitor for any alarms on the slaves inputs without absorbing a significant amount of communication time. If this 'Error flag' is monitored in this way, once an error has been detected, the master can quickly establish which areas to investigate by examining the whole of the 30006 status register. The master can then take appropriate action according to which of the other status bits has caused the 'Error flag' to be set.

6.6 'Invalid database'

INPUT STATUS FLAG LOCATION: 10093

INPUT REGISTER LOCATION: 30006: bit 3

The 'Invalid database' flag is set when a fault is detected in the configuration database.

6.7 'Highway OK'

INPUT STATUS FLAG LOCATIONS: 10090

INPUT REGISTER LOCATIONS: 30006: bits 6

The 'Highway OK' bit indicates that communication is taking place successfully on the highway between the MTL838C and the MTL831C(s). This corresponds to the illumination of the "Comm" LED of the unit.

6.8 'Transmitter failed'

INPUT STATUS FLAG LOCATIONS: 10087 - 10088

INPUT REGISTER LOCATIONS: 30006: bits 8 - 9

The 'Transmitter failed' bits are set when the MTL838C is not receiving data from a particular transmitter. The bits will only be set when a transmitter is identified as being present by the configuration parameter N_TY_MPX (see pages 40&43). This defines the number of Tx devices connected to the MTL838C.

6.9 'CJC Range Error'

INPUT STATUS FLAG LOCATIONS: 10085

INPUT REGISTER LOCATIONS: 30006: bits 11

The 'CJC Range Error' bit indicates that one of the MTL831Cs appears to be installed in a location outside of its specified operating temperature – or something has failed in the CJC measuring electronics. Use the PC Software to determine which MTL831C is in error.

6.10 'CJC Delta Error'

INPUT STATUS FLAG LOCATIONS: 10084

INPUT REGISTER LOCATIONS: 30006: bits 12

The 'CJC Delta Error' bit indicates that there is an unexpected differential between the two CJC temperature measuring devices on one of the MTL831Cs – use the PC Software to determine which unit is in error.

6.11 'Open circuit -', 'Low alarm -' and 'High alarm detected on any input'

INPUT STATUS FLAG LOCATIONS: 10081 - 10083

INPUT REGISTER LOCATIONS: 30006: bits 13 - 15

The 'Open circuit-', 'Low alarm-' and 'High alarm detected on any circuit' flags will be set if any of the inputs from the field are, respectively, open circuit or showing low or high alarms.

6.12 Not Used

INPUT STATUS FLAG LOCATIONS: 10097 - 10128

INPUT REGISTER LOCATIONS: 30007 - 30008

These registers are not used and will always return zeros.

6.13 High alarm status register

INPUT STATUS FLAG LOCATIONS: 10129 - 10160

INPUT REGISTER LOCATIONS: 30009 - 30010

The 'HIGH_ALARM' flags and register bits indicate the presence of a high alarm on the inputs to the field transmitters. Each of the possible 32 inputs is allocated a bit within the 32 bits and two registers.

The bits are arranged so that a high alarm on input 1 will set the input status flag at location 10129 and the most significant bit of register 30009 and so on through the 32 inputs, with the most significant bit of register 30010 (and location 10145) corresponding to input 17.

If any of the above bits are set to '1', then the associated bit in the STATUS register will also be set to '1'.

6.14 Low alarm status register

INPUT STATUS FLAG LOCATIONS: 10161 - 10192

INPUT REGISTER LOCATIONS: 30011 - 30012

The 'LOW_ALARM' flags and register bits indicate the presence of a low alarm on the inputs to the field transmitters. Each of the possible 32 inputs is allocated a bit within the 32 bits and two registers.

The bits are arranged so that a low alarm on input 1 will set the input status flag at location 10161 and the most significant bit of register 30011 and so on through the 32 inputs, with the most significant bit of register 30012 (and location 10167) corresponding to input 17.

If any of the above bits are set to '1', then the associated bit in the STATUS register will also be set to '1'.

6.15 Open alarm status register

INPUT STATUS FLAG LOCATIONS: 10193 - 10224

INPUT REGISTER LOCATIONS: 30013 - 30014

The 'OPEN_ALARM' flags and register bits indicate the presence of an open circuit alarm on the inputs to the field transmitters. Each of the possible 32 inputs is allocated a bit within the 32 bits and two registers.

The bits are arranged so that an open alarm on input 1 will set the input status flag at location 10193 and the most significant bit of register 30013 and so on through the 32 inputs, with the most significant bit of register 30014 (and location 10209) corresponding to input 17.

If any of the above bits are set to '1', then the associated bit in the STATUS register will also be set to '1'.

Once an open alarm has been detected by the MTL838C, the safety drive for each input will be engaged to drive the input high or low.

6.16 Scaled analog input value

With IEEE format data:

INPUT STATUS FLAG LOCATIONS: 10225 - 11248

INPUT REGISTER LOCATIONS: 30015 - 30078

With non-IEEE format data:

INPUT STATUS FLAG LOCATIONS: 10225 - 10736

INPUT REGISTER LOCATIONS: 30015 - 30046

The scaled analog values of each input are stored in the registers 'INPUT_1' to 'INPUT_32'.

If an IEEE data format is chosen, the value of each input is stored in 32 sequential flag locations and is also mapped to two input registers.

If a non-IEEE format is chosen, then the system uses 16 sequential flag locations and a single input register to store the value of each input.

The convention adopted for mapping of data in to the flags and registers is a function of the data format selected. Data format '1' maps the most significant bits of the data value in to the least significant register location (and the least significant bit in to the highest register location). All other data formats map the most significant bit of the data value in to the lowest flag location (and the most significant bit of the lowest register location).

6.17 Cold junction temperature of MTL831C's

With IEEE format data:

INPUT STATUS FLAG LOCATIONS: 11249 - 11312

INPUT REGISTER LOCATIONS: 30079 - 30082

With non-IEEE format data:

INPUT STATUS FLAG LOCATIONS: 10737 - 10768

INPUT REGISTER LOCATIONS: 30047 - 30048

The temperature of the cold junction (CJ) in each of the MTL831Cs connected to the MTL838C is stored - in either IEEE or non-IEEE data format - in the flag and register locations shown above. The first flag location contains the most significant bit of the temperature of the first MTL831C (CJ1) and so on.

If a non-IEEE data format is selected, the CJ temperature is stored in tenths of degrees. Further, if the data format chosen is unsigned, an offset of 40° is applied by the MTL838C, so that CJ temperatures down to-40° can be reported. Hence, for non-IEEE unsigned data format, a stored value of 678 corresponds to a temperature of:

678 = 10 ((I/P + 40) - 0) + 0

67.8 = I/P + 40

I/P = 67.8 -40 = 27.8°

NOTE

The 'degrees' refer to whichever unit of temperature (°C, °F or °K) is specified in Holding Register 40030.

7 COIL STATUS FLAGS

A small number of single bit coil status flags are set aside for the Modbus master to read from and write to. The facility to write to these flags is not disabled by the internal settings which disable the configuration parameters write facility.

The coil status flags are only of use when the configuration of the MTL838C is being done via the Modbus host.

NOTE

The flags that reset the unit to factory default values cause the unit to perform a significant number of internal operations. This process can take several seconds, and during this time the unit is unable to communicate with the master.

7.1 Mapping of coil status flags

The mapping of the six coil status flags within the MTL838C is shown below:

COIL STATUS FLAG LOCATION	NAME	FUNCTION
00001	CSTORE	Not used
00002	DFT831	configure to factory defaults, mV inputs
00003	DFT832	Not used
00004	CONFIRM	confirms configuration completed correctly
00005	FMT831	as DFT831, without re¬setting data format

7.2 Set factory defaults for mV inputs

COIL STATUS FLAG LOCATION: 00002

Writing a '1' into the coil status flag location DFT831 will cause the MTL838C to re-set itself and to install factory default values in to its configuration database, assuming that the field inputs are mV inputs to MTL831C units.

On receiving the instruction to force the coil status flag DFT831 to a logic '1', the MTL838C will issue a response confirming receipt of the instruction and then enter 'initialization mode' for several seconds. During this time, the unit is unable to communicate with the master and any queries addressed to the unit will be ignored.

The default parameters are:

1 MTL831C transmitter

Data format type IEEE754 (type 0)

All inputs mV type

IPZERO and OPZERO scaling parameters 0

GAIN parameters set to '1', i.e. values read directly as mV and degrees C.

7.3 Confirm database correctly configured

COIL STATUS FLAG LOCATION: 00004

A hazard exists with the MTL838C, whereby it would be possible for the unit to become re-configured, and for the master to be unaware that this had taken place. This could arise following a 'power-up' sequence in which the MTL838C detects that its stored CONFIGURATION DATABASE has become corrupted (so that the factory default values for configuration are used instead).

To protect against this risk, once such a re-configuration has occurred, the slave will respond to any READ DATA requests by issuing an EXCEPTION response. Only when the master writes a logic '1' to the CONFIRM flag location will the slave allow data to be read.

The requirement to write to the CONFIRM flag location CONFIRM remains, even if the unit is subjected to further power-down and power-up cycles.

A similar precaution must be taken to prevent the master reading data when it has instructed the slave to use a new DATAFORMAT, but before the CONFIGURATION DATABASE has been re-written in the new DATAFORMAT.

Again, to prevent the master reading data that is not configured correctly, any READ DATA queries will give rise to EXCEPTION responses, until the CONFIRM flag is set to '1'. The requirement to write to the CONFIRM flag location remains, even if the unit is subjected to further power-down and power-up cycles.

NOTE

Confirmation of a change in configuration database can also be achieved with the PC software. The 'sign-off' operation in PC software, issues an instruction equivalent to 'CONFIRM'.

7.4 Set factory defaults for mV inputs, leaving DATAFORMAT unchanged

COIL STATUS FLAG LOCATION: 00005

Writing a logic '1' to status flag FMT831 performs the same operation as DFT831, but leaves the DATAFORMAT register unaltered. This allows the unit to be reset to factory default values, and then allows the master to write a known DATABASE CONFIGURATION in the required format of data.

8 HOLDING REGISTERS

The holding registers of the MTL838C are used almost exclusively to hold data regarding the configuration of the unit. A few unused registers are available for retaining other data if required. All configuration database parameters are stored in non-volatile memory.

The layout of the holding registers is summarized in the table below:

HOLDING REGISTERS	NAME	DATA TYPE	DESCRIPTION
40001 - 40002	CSUMREF	B	Not Used
40003 - 40015	SPARE	A	unused registers
40016	DATAFMT	B	output data format
40017 - 40028	TAG	A	tag string - defined by user
40029	N_TY_MPX	B	number and type of transmitters
40030	UNIT	B	units of temperature to use
40031	POWER	B	frequency of power supply
40032	CFGTEST	B	Not Used
40033	IPTYSF_1	DF	input type and safety drive, input 1
:	:	:	:
40064	IPTYSF_32	DF	input type and safety drive, input 32
40065 - 40066	IPZERO_1	DF	zero for input 1
:	:	:	:
40127 - 40128	IPZERO_32	DF	zero for input 32
40129 - 40130	GAIN_1	DF	gain for input 1
:	:	:	:
40191 - 40192	GAIN_32	DF	gain for input 32
40193 - 40194	HA_1	DF	high alarm for input 1
:	:	:	:
40255 - 40256	HA_32	DF	high alarm for input 32
40257 - 40258	LA_1	DF	low alarm for input 1
:	:	:	:
40319 - 40320	LA_32	DF	low alarm for input 32
40321 - 40322	OPZERO_1	DF	output zero for input 1
:	:	:	:
40383 - 40384	OPZERO_32	DF	output zero for input 32

8.1 Configuration checksum reference

HOLDING REGISTER LOCATION: 40001 - 40002

These holding registers have no defined use. ASCII data can be written to and read from these registers, according to the needs of each user.

8.2 Unused holding registers

HOLDING REGISTER LOCATIONS: 40003 - 40015

A number of holding registers are provided that have no defined use. ASCII data can be written to and read from these registers, according to the needs of each user. A typical example of the use of these registers would be to store the last date of calibration check.

8.3 Data format selection

HOLDING REGISTER LOCATION: 40016

The DATAFMT register is used to select the format of the data stored by the MTL838C in those holding registers identified by 'DF' in the tables and all of its sensor input registers. (In the tables showing the contents of each register, those which are governed by DATAFMT are marked 'DF').

When a new value is written to the DATAFMT register, the scaling parameters become invalid as they conformed to the previously set data format. Any attempts to read data from these registers will cause the unit to issue an EXCEPTION response, until the CONFIRM flag is set to '1' (after the master has re-written the scaling parameters in the new data format).

The table below shows the decimal values that must be written (in binary) to the DATAFMT register to select each of the defined data formats:

DATA FMT	DESCRIPTION OF FORMAT	value stored in register	decimal value
0	IEEE single precision, floating point. Most significant data in lowest register address	0 to FFFFH	-3.4×10^{38} to $+3.4 \times 10^{38}$
1	IEEE single precision, floating point. Most significant data in highest register address	0 to FFFFH	-3.4×10^{38} to $+3.4 \times 10^{38}$
4	Unsigned 16-bit binary	0 to FFFFH	0 to 65535
5	Offset 16-bit binary	0 to FFFFH	-32768 to +32767
6	2's complement 16-bit binary	0 to FFFFH	-32768 to +32767
7	Signed 16-bit binary	0 to FFFFH	-32768 to +32767
8	Unsigned 12-bit binary	0 to FFFH	0 to 4095
9	Offset 12-bit binary	0 to FFFH	-2048 to +2047
10	2's complement 12-bit binary	0 to FFFH	-2048 to +2047
11	Signed 12-bit binary	0 to FFFH	-2048 to +2047
12	Unsigned 4-decade BCD	0 to 9999 (BCD)	0 to 9999
13	Offset 4-decade BCD	0 to 9999 (BCD)	-5000 to +4999
14	10's complement 4-decade BCD	0 to 9999 (BCD)	-5000 to +4999
16	Unsigned 3-decade BCD	0 to 999 (BCD)	0 to 999
17	Offset 3-decade BCD	0 to 999 (BCD)	-500 to +499
18	Offset 10's comp. 3-decade BCD	0 to 999 (BCD)	-500 to +499

In many of the non-IEEE data formats specified in the table above, the encoding of the value in to the chosen format is not immediately apparent. The table below explains the encoding of each format. The table shows the decimal value of the binary, hexadecimal, or BCD content of the register, and for each range of values for each data type, the formula for finding the 'represented value' is given.

DATA FMT	RANGE OF VALUES (dec. equivalent)	FORMULA FOR REPRESENTED VALUE
4	0 to 65535	RV = REG
5	0 to 65535	RV = REG- 32768
6	0 to +32767	RV = REG
	32768 to 65535	RV = REG- 65536
7	0 to +32767	RV =- REG
	32768 to 65535	RV = REG- 32768
8	0 to 4095	RV = REG
9	0 to 4095	RV = REG- 2048
10	0 to 2047	RV = REG
	2048 to 4095	RV = REG- 4096
11	0 to 2047	RV =- REG
	2048 to 4095	RV = REG- 2048
12	0 to 9999	RV = REG
13	0 to 9999	RV = REG- 5000
14	0 to 4999	RV = REG
	5000 to 9999	RV = REG- 10000
16	0 to 999	RV = REG
17	0 to 999	RV = REG- 500
18	0 to 499	RV = REG
	500 to 999	RV = REG- 1000

NOTES

1. The conventions used in the table are that 'RV' is the represented value and 'REG' is the decimal equivalent of the registers contents (which will actually be in hexadecimal or binary).
2. The encoding of some parameters in non-IEEE format require further manipulation to be expressed as 'numerand and exponent'. See Appendix B.
3. The encoding of IEEE data is described in Appendix A.

8.4 Tag field

HOLDING REGISTER LOCATION: 40017- 40028

The TAG holding register will contain a blank ASCII string as a default, which may be modified to a more suitable TAG by the master. Each TAG register will hold two ASCII characters, giving a maximum of 24 characters stored.

8.5 Number and type of transmitters

HOLDING REGISTER LOCATION: 40029

The holding register N_TY_MPX contains a binary value that is encoded to describe the number and type of transmitters connected to the data highway of any given MTL838C receiver. The information is encoded in to the binary value by a series of multiplications and additions. The result is a binary value that uniquely describes the number and type of receivers. The calculation of the value is shown below:

N_TY_MPX = MPX1_TYPE
+ (8 x MPX2_TYPE)
+ (64 X MPX3_TYPE)
+ (512 x MPX4_TYPE)
+ (4096 x NUM_MPX)

where: 'MPXn_TYPE' defines the type of transmitter 'n', and:

MPXn_TYPE = 3 for MTL831C (lower numbers for previous versions)

'NUM_MPX' defines the number of transmitters. Valid values are 1 or 2.

Currently, 'n' can only be 1 or 2 as we only support up to 2 transmitters on the Data Highway.

8.6 Units of temperature

HOLDING REGISTER LOCATION: 40030

The binary value stored in the UNIT register defines the units that are used for the temperature readings made by the MTL831C multiplexer receiver for thermocouples, RTDs and cold junctions. The value is stored as the binary equivalent of the decimal values:

1: degrees Centigrade (°C)

2: degrees Fahrenheit (°F)

3: Kelvin (K)

8.7 Line frequency of power supply

HOLDING REGISTER LOCATION: 40031

The value that is placed in the holding register POWER identifies the frequency of the local AC power supply. This is then used to establish the line frequency which should be rejected. The frequency is identified by writing a the following decimal values in to the register:

0: 50Hz supply 1: 60Hz supply

8.8 Input type and safety drive

HOLDING REGISTER LOCATION: 40033 - 40064

The holding registers INPTYSF_1 to INPTYSF_32 are used to store information regarding the type of field input to the multiplexer transmitter and the safety drive that is specified for each input. The actual contents of the register are binary encoded values that uniquely describe the input type and safety drive selected for each input.

The input type selected can be one of a wide range of inputs. The value that is used here is used in conjunction with the value calculated for N_TY_MPX, which defines the number and type of transmitters connected to each MTL838C.

The safety drive comes in to action on detection of an open circuit sensor (if open sensor detection is selected) or if a transmitter is found to have failed. The appropriate OPEN_ALARM and/or transmitter failed STATUS bit will be set and the input will be driven to its full scale or lowest value (depending on the selection of safety drive). If HIGH_AL or LOW_AL are selected, these will also be triggered by the safety drive.

If no safety drive is selected, if a transmitter fails, or an input becomes open circuit the MTL838C will continue to supply the most up-to-date information it has received. This then allows the host to read values that may differ widely from the actual measured value in the field. It is the users responsibility to ensure that the data read from the MTL838C is valid, either by the judicious use of safety drives (which is recommended) and/or by continually monitoring that the unit and its inputs are giving valid readings by way of the STATUS information.

By default, the upscale safety drive will be selected.

The value is calculated as follows: INPTYSF_n = IPTYPE_n + (256 x SAFETY_n)

'IPTYPE_n' is the type of input connected to input 'n', as shown in the table below:

IP TYPE_n	MTL831C
0	mV voltage input (scalable)
1	E-type THC temp without CJ comp.
2	J-type THC temp without CJ comp.
3	K-type THC temp without CJ comp.
4	N-type THC temp without CJ comp.
5	R-type THC temp without CJ comp.
6	T-type THC temp without CJ comp
7	E-type THC temp with CJ comp.
8	J-type THC temp with CJ comp.
9	K-type THC temp with CJ comp.
10	N-type THC temp with CJ comp.
11	R-type THC temp with CJ comp.
12	T-type THC temp with CJ comp.
13	3-wire RTD Resistance
14	3-wire RTD PT100 Temperature
15	Not Used
16	S-type THC temp without CJ comp
17	S-type THC temp with CJ comp

continued

IP TYPE_n	MTL831C
18	E-type THC mV with CJ comp.
19	J-type THC mV with CJ comp.
20	K-type THC mV with CJ comp.
21	N-type THC mV with CJ comp.
22	R-type THC mV with CJ comp.
23	T-type THC mV with CJ comp.
24	S-type THC mV with CJ comp.
25	B-type THC mV with CJ comp.
26	B-type THC temp without CJ comp
27	B-type THC temp with CJ comp
28	4-wire RTD Resistance
29	2-wire RTD Resistance
30	4-wire PT100 RTD Temperature
31	2-wire PT100 RTD Temperature
32	C-type THC mV with CJ comp.
33	C-type THC temp without CJ comp
34	C-type THC temp with CJ comp
35	XK-type THC mV with CJ comp.
36	XK-type THC temp without CJ comp
37	XK-type THC temp with CJ comp
38	2-wire Cu50 RTD Temperature
39	3-wire Cu50 RTD Temperature
40	4-wire Cu50 RTD Temperature
41	2-wire Cu53 RTD Temperature
42	3-wire Cu53 RTD Temperature
43	4-wire Cu53 RTD Temperature
44	2-wire Ni100 RTD Temperature
45	3-wire Ni100 RTD Temperature
46	4-wire Ni100 RTD Temperature

'SAFETY_n' is the type of safety drive selected, as shown in the following table:

SAFETY_n	SAFETY SELECTION
0	no safety drive selected
1	upscale safety drive selected
2	downscale safety drive selected

The open sensor detection and the output value that will be given by the upscale and downscale drives depend on the input type selected. The table below shows the values that will be read on the outputs when driven upscale or downscale after the detection of an open circuit input. These values will be reached if the full scale values of the selected data format are sufficiently wide to include these values.

SENSOR TYPE	DOWNSCALE LIMIT	UPSCALE LIMIT
millivolt	-120mV	+120mV
thermocouple	-120mV	+120mV
resistance	0 Ω	1200 Ω
Type E THC	-300°C	1200°C
Type J THC	-250°C	1300°C
Type K THC	-300°C	1400°C
Type N THC	-300°C	1400°C
Type R THC	-100°C	1800°C
Type T THC	-300°C	500°C
Type S THC	-100°C	1800°C
Type B THC	-50°C	1900°C
Type C THC	-50°C	2400°C
Type XK THC	-250°C	900°C
PT100	-250°C	900°C
Cu50	-250°C	250°C
Cu53	-100°C	250°C
Ni100	-100°C	300°C

8.9 Input zero with offset - for scaling output measurements

HOLDING REGISTER LOCATION: 40065 - 40128

The two IPZERO_n registers for each input are used to hold the value of the input zero after the offset has been applied. Each IPZERO_n will be found from:

IPZERO_n = input zero + offset, where:

input zero: is the lowest input value that may be recorded by the field input

offset: is the value included by the MTL838C to allow negative numbers to be represented by unsigned data formats.

Calculation of scaled output values is discussed in detail on page 51.

NOTE

As with many other database configuration parameters, the values stored in IPZERO are of the data format selected by the user. When non-IEEE data is stored, the two registers hold a 'numerand' and an 'exponent' of the required value. This allows the selected data format to provide a broader range of values than would otherwise be possible. The calculation of such values is discussed in Appendix B.

8.10 Gain - for scaling output measurements

HOLDING REGISTER LOCATIONS: 40129 - 40192

The two GAIN_n registers are used to hold a value termed 'gain' for each of the n inputs. This value is used in conjunction with the IPZERO, to give the required range (or span) of measurements. Gain for each input is normally calculated as below:

GAIN = (Output FSD - Output zero) / (Input FSD - Input zero)

Calculation of scaled output values is discussed in detail on page 51.

NOTE

As with IPZERO_n, for non-IEEE data formats, the values for GAIN_n are stored as 'numerand' and 'exponent' according to the data format chosen. This is discussed further in Appendix B.

8.11 High alarm level

HOLDING REGISTER LOCATIONS: 40193 - 40256

The HA_n registers are used to store the level which should not be exceeded by the scaled output value. If the scaled output does exceed this level, the appropriate bit within the HIGH_ALARM input register will be set, and the 15th bit of the STATUS register will also be set.

NOTE

For non-IEEE data, the value for HA_n will be stored as a 'numerand' and 'exponent'. See Appendix B.

8.12 Low alarm level

The LA_n registers are used to store the level below which the scaled output value should not go. If the scaled output does fall below this level, the appropriate bit within the LOW_ALARM input register will be set, and the 16th bit of the STATUS register will also be set.

NOTE

For non-IEEE data, the value for LA_n will be stored as a 'numerand' and 'exponent'. See Appendix B.

8.13 Output zero offset

The value stored in the two OPZERO_n registers matches the lowest value of output that is required from the nth output - corresponding to the lowest value of the nth input.

Calculation of scaled output values is discussed in detail on page 51.

NOTE

As with IPZERO_n, for non-IEEE data formats, the values for OPZERO_n are stored as 'numerand' and 'exponent' according to the data format chosen. This is discussed further in Appendix B.

9 MTL838C EXCEPTION RESPONSES

The following section describes the exception responses that may be given to each type of query that may be received by the MTL838C.

Exception responses are only issued by Modbus slaves if a query that is received correctly (i.e. passes the error and parity checks) cannot be carried out by the slave. An exception response is constructed by returning the received function code to the master with its MSB set to '1', followed by an exception code, passed back to the master as the first byte of the data field. See pages 7 and 25 for more detail.

9.1 Following 'READ COIL STATUS' queries

EXCEPTION	EXCEPTION RESPONSE
Address of first status to be read is outside the range 0000- 0005	Code 02 (Only coil addresses 0000 to 0005 contain defined information)
Number of locations to be read is outside the range 1- 5	

9.2 Following 'READ INPUT STATUS' queries

EXCEPTION	EXCEPTION RESPONSE
Address of first location to be read is outside the range 0000- 1311	Code 02 (Only status flag addresses 0000 to 1311 for IEEE, 0000 to 0767 for non-IEEE contain defined information)
Number of locations to be read is outside the range 1-512	
Configuration database is not yet confirmed	Code 04

9.3 Following 'READ HOLDING REGISTERS' query

EXCEPTION	EXCEPTION RESPONSE
Address of first register to be read is outside the range 0000- 0383	Code 02 (Only register addresses 0000 to 0383 contain defined information)
Number of registers to be read is outside the range 1- 60	

9.4 Following 'READ INPUT REGISTERS' query

EXCEPTION	EXCEPTION RESPONSE
Address of first register to be read is outside the range 0000- 0081	Code 02 (Only register addresses 0000 to 0081 contain defined information)
Number of registers to be read is outside the range 1- 60	
Configuration database is not yet confirmed	Code 04

9.5 Following 'FORCE SINGLE COIL' queries

EXCEPTION	EXCEPTION RESPONSE
Address of coil to be forced is outside the range 0000- 0005	Code 02 (only addresses 0000 to 0005 are defined)
Data value is neither FF00 hex ('1') nor 0000 hex ('0')	Code 03
Coil that was to be forced is 'write disabled'	Code 01

9.6 Following 'PRESET SINGLE REGISTER' queries

EXCEPTION	EXCEPTION RESPONSE
Address of register to preset is outside the range 0000- 0383	Code 02 (only addresses 0000 to 0383 are defined)
Data value is outside range 0- 65535	Code 03
Register that was to preset is 'write disabled'	Code 01

9.7 Following 'READ EXCEPTION STATUS' queries

No exception responses can be generated by the MTL838C on correctly receiving a READ EXCEPTION STATUS query.

9.8 Following 'DIAGNOSTICS' queries

EXCEPTION	EXCEPTION RESPONSE
Diagnostic code not supported by the MTL838C	Code 03

9.9 Following 'PRESET MULTIPLE REGISTERS' queries

EXCEPTION	EXCEPTION RESPONSE
Address of first register to be preset is outside the range 0000- 0383	Code 02 (only addresses 0000 to 0383 are defined)
Number of registers to be preset is outside the range 1 to 60	
Number of data bytes is outside the range 2 to 120	
Register data values are outside the range 0 to 65535	Code 03
Register that was to be forced is 'write disabled'	Code 01

NOTE

Within the limits of the 'address of register to be preset' given above, it is possible for the MTL838C to accept a query that requires an undefined register to be preset. The MTL838C will accept the query and issue a confirming response, but it will not modify any registers.

9.10 Following queries not supported by the MTL838C

EXCEPTION	EXCEPTION RESPONSE
Function code is not supported by the MTL838C	Code 01

NOTE

The 'broadcast' function is not supported by the MTL838C. The unit does not decode messages issued with the broadcast slave address '0', so that it does not subsequently issue an exception response.

10 SCALING

The inputs that are received by the MTL831C transmitter are processed by the MTL838C according to the type of input, and depending on the scaling parameters that have been selected by the user. The processing and scaling of input data is discussed here in detail. In practice, most users have a standard data format and standard zero and FSD values for their control system. This is easily accommodated using the PC configuration software.

The timing of responses to requests and the speed of response of the overall system is also covered.

10.1 Background to scaling input data

This section describes the fundamentals of scaling input data that will need to be understood by those configuring the MTL838C via Modbus. If configuration via the PC software is to be used, this section need not be understood in detail. The following section on practical calculations will be of more relevance when using the PC software.

In general terms, each input to the MTL838C will have an output given by:

output = gain x (input - input zero) + output zero

Where:

output:	is a digital value. This output value is shown on the PC screen as the "Reading" and is also available in the associated Modbus register.
gain:	is a value provides the required output range for the specified input range. This must be calculated by the user and written to the unit.
input:	is the value of the field input (mV, °C, etc.)
input zero:	is the lowest value that the input will be expected to record.
output zero:	is the output value that corresponds to the lowest input value.

NOTE

It is important to understand the limitations of the output range that is defined by the data format that has been chosen. With IEEE format there is little need for concern because of the enormous range that is available (> 10^{38}) but with non-IEEE data formats the output zero and FSD values should be chosen to give the maximum resolution. As mentioned earlier, many users have site standards for zero and FSD values that have been selected for optimum performance.

In practice, the previous equation must be modified. This is because the MTL838C must be able to represent negative numbers when using unsigned data formats. This means that the whole scale must be lifted, using an 'offset', to allow negative values to be represented by a 'positive' value. Specifically then, the equation must be:

output = GAIN_n x (input + offset - IPZERO_n) + OPZERO_n

Where:

IPZERO_n = input zero + offset

All the parameters in upper case represent values written to the Holding Registers for each input number 'n'.

The level of offset varies according to the type of input selected:

INPUT TYPE	DATA FORMAT	OFFSET VALUE
All	signed	0
CJ temperature (0.1)	unsigned non-IEEE	40°C
mV	unsigned	100mV
Temperature	unsigned	500° (C, F, or K)

10.2 Calculation of scaling parameters - in practice

Scaling parameters must be calculated for each input to the multiplexer system. This can be done by first completing a table as shown below:

ZERO VALUES	FULL SCALE VALUES	RANGES	GAIN (GAIN_n)
Input zero	Input FSD	(input FSD)-(input zero)	(output range)

OPZERO_n	output FSD	(output FSD) – (OPZERO_n)	(input range)

The table gives parameters that need to be entered in to the PC software. Those wishing to configure via Modbus, however, must follow the further working below.

The OPZERO_n and GAIN_n values are identical to the values written to the MTL838C. However, the 'input zero' value will need to be modified as described in the last section:

IPZERO_n = input zero + offset

As an example, consider an application using a thermocouple to measure temperature in the range -10°C to +40°C. The output data format will be unsigned 3 decade BCD, with a (decimal) range of 100 to 600. These values can be put in the table and the gain calculated:

ZERO VALUE	FULL SCALE VALUE	RANGE	GAIN
-10°C	+40°C	+40°C – -10°C = 50°C	500 / 50 =10
100	600	600 - 100 = 500	

The data format selected will require an offset of 500 C and based on this therefore, the values written to the MTL838C would be:

IP_ZERO : 490 (found from-10°C + 500°C = 490°C)

OP_ZERO: 100

GAIN: 10

As an example, the output can be calculated for an input of +40°C:

output = GAIN x (input + offset- IPZERO) + OPZERO

= 10 x (40 + 500- 490) + 100 = 500 + 100 = 600

10.3 Sensor Input Processing

The inputs to the transmitters are measured and processed in a number of different ways according to the type of input selected and a number of other factors. The processing of each input type is discussed in detail in the sections below:

10.3.1 Thermocouple inputs

The message received from the MTL831C is decoded to a mV measurement for each thermocouple input. Each measurement is then corrected according to the latest figures for calibration for that input.

If CJ compensation is selected, the mV measurement value is further corrected according to the CJ temperature of the associated transmitter.

Linearization and conversion to a temperature reading is carried out by comparing the corrected mV value with the linearization tables that are stored within the MTL838C. The result is a temperature measurement expressed in Kelvin, degrees Fahrenheit, or degrees Centigrade according to the units selected.

The temperature value is converted to the required output according to the equation below, and depending on the scaling parameters selected:

output = GAIN_n x (temperature + offset- IPZERO_n) + OPZERO_n

The output is expressed in the required data format and is written to the input registers for 'INPUT_n', from where it may be read by the Modbus host.

10.3.2 Resistance inputs

The message received from the MTL831C is decoded to a resistance measurement for each resistance input. Each measurement is then corrected according to the latest figures for calibration for that input.

The resistance value for each input is converted to the required output according to the equation below, and depending on the scaling parameters selected:

output = GAIN_n x (m resistance + offset- IPZERO_n) + OPZERO_n

The output is expressed in the required data format and is written to the input registers for 'INPUT_n', from where it may be read by the Modbus host.

10.3.3 RTD inputs

The message received from the MTL831C is decoded to a resistance measurement for each RTD input. Each measurement is then corrected according to the latest figures for calibration for that input.

Linearization and conversion to a temperature reading is carried out by comparing the corrected resistance value with the linearization tables that are stored within the MTL838C. The result is a temperature measurement expressed in Kelvin, degrees Fahrenheit, or degrees Centigrade according to the units selected.

The temperature value is converted to the required output according to the equation below, and depending on the scaling parameters selected:

output = GAIN_n x (temperature + offset- IPZERO_n) + OPZERO_n

The output is expressed in the required data format and is written to the input registers for 'INPUT_n', from where it may be read by the Modbus host.

10.3.4 mV inputs

The message received from the MTL831C is decoded to a mV measurement for each resistance input. Each measurement is then corrected according to the latest figures for calibration for that input.

The mV value for each input is converted to the required output according to the equation below, and depending on the scaling parameters selected:

output = GAIN_n x (mV input + offset- IPZERO_n) + OPZERO_n

The output is expressed in the required data format and is written to the input registers for 'INPUT_n', from where it may be read by the Modbus host.

10.3.5 Data timing

The MTL838C typically receives updated measurement data from each MTL831C every 500ms.

The delay between receiving a Modbus request and issuing a response will be approximately 3.5 character periods.

The overall response time is largely dependent on the choice of baud rate and communication mode. The time taken to transmit each query and each response can be easily calculated by multiplying the chosen baud rate by the number of bits that are transmitted in each message.

11 APPENDIX A

11.1 IEEE single precision data format

This appendix describes the encoding of IEEE single precision data. The table below shows the composition of the four 8-bit bytes required to describe a value in IEEE format. (Strictly, the data format is termed the IEEE754 single precision data format.) The most significant byte is transmitted first.

ZERO VALUE	ZERO VALUE							
	7	6	5	4	3	2	1	0
most significant byte	s	e7	e6	e5	e4	e3	e2	e1
2nd most significant byte	e0	f22	f21	f20	f19	f18	f17	f16
3rd most significant byte	f15	f14	f13	f12	f11	f10	f9	f8
least significant byte	f7	f6	f5	f4	f3	f2	f1	f0

where:

s = sign bit, e = exponent, f = significand

A sign bit (s) of 0 indicates a positive number, and a 1 is negative

The value to be encoded is given by the relevant entry in the table below:

e	f	v
0 < e < 255	all	v = +/- 2^{e-127} x1.f
e = 0	f <> 0	v = +/- 2^{e-126} x0.f
e = 0	f = 0	v = 0
e = 255	f = 0	v = +/- infinity
e = 255	f <>0	v = non-allowed number

For example:

s = 0, e = 128, f = 5

v = + 2^{e-127} x1.f

= $2^{128-127}$ x 1.5

= 2 x 1.5

= 3

12 APPENDIX B

12.1 Non-IEEE data format

When non-IEEE data formats are used, some of the scaling parameter values used by the MTL838C must be expressed as a numerand and an exponent. This gives much greater flexibility to the values that may be used- especially when very large or very small numbers are required. This issue need only be considered if the user intends to configure the MTL838C via the Modbus host. If the PCS83 is used to configure the unit there is no need to consider the encoding of data in this way, as both configuration tools make these calculations automatically. The process becomes totally transparent to the user.
The parameters that must be expressed as numerand and exponent are GAIN_n, OPZERO_n, IPZERO_n, HA_n and LA_n.

12.2 Numerand and exponent

The exact process for expressing a value as a numerand and exponent will vary with the type of data format selected, but the overall principle behind the expression remains the same irrespective of the data format selected.

The value must be expressed in the general form:

$$V = n \times 10^{e}$$

where:

V = the value to be expressed
n = the 'numerand', with $-1 < n < 1$
e = the 'exponent', with $e < 6$

Expressing 'n' and 'e' with non-IEEE data formats.

The equation for the calculation of 'n' and 'e' is modified slightly from that shown above, by the introduction of another factor:

$$V = (x / N) \times 10^{e}$$

where:

x = an integer value expressed in the chosen data format
N = the maximum value that can be expressed in the chosen data format

with:

$|x / N| < 1$

The value of N varies with the chosen data format as shown in the table below:

DATA FORMAT	EXPRESSION
unsigned 16-bit binary	$V = (x / 65535) \times 10^{e}$
other 16-bit formats	$V = (x / 32768) \times 10^{e}$
unsigned 12-bit binary	$V = (x / 4095) \times 10^{e}$
other 12-bit formats	$V = (x / 2048) \times 10^{e}$
unsigned 4-decade BCD	$V = (x / 9999) \times 10^{e}$
other 4-decade BCD formats	$V = (x / 5000) \times 10^{e}$
unsigned 3-decade BCD	$V = (x / 999) \times 10^{e}$
other 3-decade BCD	$V = (x / 500) \times 10^{e}$

Once the values of 'x' and 'e' have been calculated, they must be adjusted by the offset for each data format given in the table of section 10.1.

The values of 'x' and 'e', after the offset has been applied, are written to the two registers that contain the required scaling value. The upper register contains the numerand and the lower register the exponent.

Example:

To represent-399.9 in offset 16-bit format. (Note that it would not be possible to encode the value directly in the chosen data format, the closest value that the format could represent is-400).

First 'normalise' the numerand to give a value of $n < 1$:

$$V = n \times 10^{e} = -0.3999 \times 10^{3} = -399.9$$

For this data format, the value of 'N' is 32768, thus:

$$V = (x / N) \times 10^{e} = x / 32768 \times 10^{3} = -13104/32768 \times 10^{3} = -399.9$$

Thus:

$$x = -13104 \text{ and } e = 3$$

In the offset 16-bit format, the figure to be written to the register is found from:

$$RV = REG - 32768$$

thus:

$$REGx = x + 32768 = -13104 + 32768 = 19664 \quad REGe = e + 32768 = 3 + 32768 = 32771$$

These two values REGx and REGe can then be written to the two registers for the scaling parameter-399.9 with 'Offset 16-bit' data format selected. The value for REGx is written to the upper of the two registers, REGe is written to the lower.

13 APPENDIX C

13.1 Faultfinding on the MTL830C System

This section will focus on possible issues with Modbus communication. For other system issues, please see the relevant manual below. Also, connecting to the MTL838C using a USB cable, PC, and the PC software will allow validation of the configuration in the unit as well as verification of operation and much diagnostic information.

- INM838C MTL838C Installation Manual
- INM831C MTL831C Installation Manual
- INM838C-MBT MT838C Installation Manual

13.1.1 Host cannot communicate with the MTL838C

- Verify that the RS485 wiring is correct (also check for proper line termi nation) – see INM838C. For the MTL838C-MBT verify the Ethernet wiring and that the unit has the correct IP address- see section 7 of the INM 838C PC Modbus manual.
- Make sure the MTL838C is properly powered (POWER LED is ON)
- Use the PC Software to verify the configured Modbus Address. For the RTU version (model MTL838C) verify that the Baud Rate and Parity are set correctly.
- Make sure PC Software is NOT communicating with the MTL838C as this disables Modbus communication

13.1.2 Host cannot read Input Status Flags and Registers

- The MTL838C will reject a request to read data if it has not received a confirmation from either the PC Software or the Modbus Host that the configuration is correct. In the PC Software this is called "Sign Off". For Modbus it requires a write to the Coil Status Register CONFIRM (00004).

THIS PAGE IS LEFT INTENTIONALLY BLANK

AUSTRALIA
Eaton Electrical (Australia) Pty Ltd,
10 Kent Road, Mascot, New South Wales, 2020, Australia

Tel: +61 1300 308 374 Fax: +61 1300 308 463
E-mail: mtlsalesanz@eaton.com

BeNeLux
MTL Instruments BV
Ambacht 6, 5301 KW Zaltbommel
The Netherlands

Tel: +31 (0) 418 570290 Fax: +31 (0) 418 541044
E-mail: mtl.benelux@eaton.com

CHINA
Cooper Electric (Shanghai) Co. Ltd
955 Shengli Road, Heqing Industrial Park
Pudong New Area, Shanghai 201201

Tel: +86 21 2899 3817 Fax: +86 21 2899 3992
E-mail: mtl-cn@eaton.com

FRANCE
MTL Instruments sarl,
7 rue des Rosiéristes, 69410 Champagne au Mont d'Or
France

Tel: +33 (0)4 37 46 16 53 Fax: +33 (0)4 37 46 17 20
E-mail: mtlfrance@eaton.com

GERMANY
MTL Instruments GmbH,
Heinrich-Hertz-Str. 12, 50170 Kerpen, Germany

Tel: +49 (0)22 73 98 12- 0 Fax: +49 (0)22 73 98 12- 2 00
E-mail: csckerpen@eaton.com

INDIA
MTL India,
No.36, Nehru Street, Off Old Mahabalipuram Road
Sholinganallur, Chennai- 600 119, India

Tel: +91 (0) 44 24501660 /24501857 Fax: +91 (0) 44 24501463
E-mail: mtlindiasales@eaton.com

ITALY
MTL Italia srl,
Via San Bovio, 3, 20090 Segrate, Milano, Italy

Tel: +39 02 959501 Fax: +39 02 95950759
E-mail: chmninfo@eaton.com

JAPAN
Cooper Industries Japan K.K.
MT Building 3F, 2-7-5 Shiba Diamon, Minato-ku
Tokyo, Japan 102-0012

Tel: +81 (0)3 6430 3128 Fax:+81 (0)3 6430 3129
E-mail: mtl-jp@eaton.com

NORWAY
Norex AS
Fekjan 7c, Postboks 147,
N-1378 Nesbru, Norway

Tel: +47 66 77 43 80 Fax: +47 66 84 55 33
E-mail: info@norex.no

RUSSIA
Cooper Industries Russia LLC
Elektrozavodskaya Str 33
Building 4
Moscow 107076, Russia

Tel: +7 (495) 981 3770 Fax: +7 (495) 981 3771
E-mail: mtlrussia@eaton.com

SINGAPORE
Eaton Electric (Singapore) Pte Ltd
100G Pasir Panjang Road
Interlocal Centre
#07-08 Singapore 118523
#02-09 to #02-12 (Warehouse and Workshop)

Tel: +65 6 645 9888 ext 9864/9865
Fax: 65 6 645 9811
E-mail: sales.mtlsing@eaton.com

SOUTH KOREA
Cooper Crouse-Hinds Korea
7F. Parkland Building 237-11 Nonhyun-dong Gangnam-gu,
Seoul 135-546, South Korea.

Tel: +82 6380 4805 Fax: +82 6380 4839
E-mail: mtl-korea@eaton.com

UNITED ARAB EMIRATES
Cooper Industries/Eaton Corporation
Office 205/206, 2nd Floor SJ Towers, off. Old Airport Road,
Abu Dhabi, United Arab Emirates

Tel: +971 2 44 66 840 Fax: +971 2 44 66 841
E-mail: mtlgulf@eaton.com

UNITED KINGDOM
Eaton Electric Limited,
Great Marlings, Butterfield, Luton
Beds LU2 8DL

Tel: +44 (0)1582 723633 Fax: +44 (0)1582 422283
E-mail: mtlenquiry@eaton.com

AMERICAS
Cooper Crouse-Hinds MTL Inc.
3413 N. Sam Houston Parkway W.
Suite 200, Houston TX 77086, USA

Tel: +1 800-835-7075 Fax: +1 866-298-2468
E-mail: mtl-us-info@eaton.com

Powering Business Worldwide

Eaton Electric Limited,
Great Marlings, Butterfield, Luton
Beds, LU2 8DL, UK.
Tel: + 44 (0)1582 723633 Fax: + 44 (0)1582 422283
E-mail: mtlenquiry@eaton.com
www.mtl-inst.com

Publication No. INM MTL838C- MBF Rev 4 190221
February 2021

EUROPE (EMEA):
+44 (0)1582 723633
mtlenquiry@eaton.com

THE AMERICAS:
+1 800 835 7075
mtl-us-info@eaton.com

ASIA-PACIFIC:
+65 6645 9888
sales.mtlsing@eaton.com