CHAPTER 4 COMMAND SET

The M1000 modules operate with a simple command/response protocol to control all module functions. A command must be transmitted to the module by the host computer or terminal before the module will respond with useful data. A module can never initiate a communications sequence. A variety of commands exists to exploit the full functionality of the modules. A list of available commands and a sample format for each command is listed in Table 4.1.

Command Structure

Each command message from the host must begin with a command prompt character to signal to the modules that a command message is to follow. There are two valid prompt characters; a dollar sign character ($) is used to generate a short response message from the module. A short response is the minimum amount of data necessary to complete the command. The second prompt character is the pound sign character (#) which generates long responses (the long response format will be covered a little later).

The prompt character must be followed by a single address character identifying the module to which the command is directed. Each module attached to a common communications port must be setup with its own unique address so that commands may be directed to the proper unit. Module addresses are assigned by the user with the SetUp (SU) command. For best results, use printable ASCII characters such as '1' (ASCII $31) or 'A' (ASCII $41) are the best choices for address characters.

The address character is followed by a two-character command which identifies the function to be performed by the module. All of the available commands are listed in Table 4.1 along with a short function definition. All commands are described in full later in this section. Commands must be transmitted as uppercase characters.

A two-character checksum may be appended to any command message as a user option. See 'Checksum' section below.

All commands must be terminated by a Carriage Return character (ASCII $0D). (In all command examples in this text the Carriage Return is either implied or denoted by the symbol 'CR'.)

Data Structure

Many commands require additional data values to complete the command definition as shown in the example commands in Table 4.1. The particular data necessary for these commands is described in full in the complete command descriptions.

The most common type of data used in commands and responses is analog data. Analog data is always represented in the same format for all models in the M1000 series. Analog data is represented as a nine-character string consisting of a sign, five digits, decimal point, and two additional digits. The string represents a decimal value in engineering units.

Examples:	+12345.68
	+00100.00
	-00072.10
	-00000.00

When using commands that require analog data as an argument, the full nine-character string must be specified, even though some digits may not be significant. Failure to do this will result in a SYNTAX ERROR.

Analog data responses from the module will always be transmitted in the nine-character format. This greatly simplifies software parsing routines since all analog data is in the same format for all module types.

In many cases, some of the digits in the analog data may not be significant. For instance, the M1300 thermocouple input modules feature 1 degree output resolution. A typical analog data value from this type of module could be +00123.00 . The two digits to the right of the decimal point have no significance in this particular model. However, the data format is always adhered to in order to maintain compatability with other module types.

The maximum computational resolution of the module is 16 bits, which is less than the resolution that may be represented by an analog data variable. This may lead to round-off errors in some cases. For example, an alarm value may be stored in a M1000 module using the 'HI' command:

Command: $1HI+12345.67M
Response: *

The alarm value may then be read back with the Read High (RH) command:

Command: $1RH
Response: *+12345.60M

It appears that the data read back does not match the value that was originally saved. The error is caused by the fact that the value saved exceeds the computational resolution of the module. This type of round-off error only appears when large data values saved in the module's EEPROM are read back. In most practical applications, the problem is nonexistent.

Overload values of analog data are represented by +99999.99 and -99999.99 .

Data read back from the Event Counter with the Read Events (RE) command is in the form of a seven-digit decimal number with no sign or decimal point. Round-off errors do not occur on the event counter. For example:

Command: $1RE
Response: *0000123

The Digital Input, Digital Output, and Setup commands use hexadecimal representations of data. The data structures for these commands are detailed in the command descriptions.

Write Protection

Many of the commands listed in Table 4.1 are under the heading of 'Write Protected Commands'. These commands are used to alter setup data in the module's EEPROM. These commands are write protected to guard against accidental loss of setup data. All write-protected commands must be preceded by a Write Enable (WE) command before the protected command may be executed.

Miscellaneous Protocol Notes

The address character must transmitted immediately after the command prompt character. After the address character the module will ignore any character below ASCII $23 (except, of course, CR). This allows the use of spaces (ASCII $20) within the command message for better readability if desired.

The length of a command message is limited to 20 printable characters. If a properly addressed module receives a command message of more than 20 characters the module will abort the whole command sequence and no response will result.

If a properly addressed module receives a second command prompt before it receives a CR, the command will be aborted and no response will result.

Response Structure

Response messages from the M1000 module begin with either an asterisk ' * ' (ASCII $2A) or a question mark ' ? ' (ASCII $3F) prompt. The ' * ' prompt indicates acknowledgment of a valid command. The ' ? ' prompt precedes an error message. All response messages are terminated with a CR. Many commands simply return a single ' * ' character to acknowledge that the command has been executed by the module. Other commands send data information following the ' * ' prompt. The response format of all commands may be found in the detailed command description.

The maximum response message length is 20 characters.

A command/response sequence is not complete until a valid response is received. The host may not initiate a new command until the response from a previous command is complete. Failure to observe this rule will result in communications collisions. A valid response can be in one of three forms:

1) a normal response indicated by a ' * ' prompt
2) an error message indicated by a ' ? ' prompt
3) a communications time-out error

When a module receives a valid command, it must interpret the command, perform the desired function, and the communicate the response back to the host. Each command has an associated delay time in which the module is busy calculating the response. If the host does not receive a response in an appropriate amount of time specified in Table 4.1, a communications time-out error has occured. After the communications time-out it is assumed that no response data is forthcoming. This error usually results when an improper command prompt or address is transmitted.

Long Form Responses

When the pound sign ' # ' command prompt is used, the module will respond with a 'long form' response. This type of response will echo the command message, supply the necessary response data, and will add a two-character checksum to the end of the message. Long form responses are used in cases where the host wishes to verify the command received by the module. The checksum is included to verify the integrity of the response data. The ' # ' command prompt may be used with any command. For example:

Command: $1RD (short form)
Response: *+00072.10

Command: #1RD (long form)
Response: *1RD+00072.10A4 (A4=checksum)

Checksum

Checksum, a two character hexadecimal value added to the end of a message, verifies that the message received is exactly the same as the message sent. The checksum ensures the integrity of the information communicated.

Command Checksum

A two-character checksum may be appended to any command to the M1000 module as a user option. When a module interprets a command, it looks for the two extra characters and assumes that it is a checksum. If the checksum is not present, the module will perform the command normally. If the two extra characters are present, the module will calculate the checksum for the message. If the calculated checksum does not agree with the transmitted checksum, the module will respond with a 'BAD CHECKSUM' error message and the command will be aborted. If the checksums agree, the command will be executed. If the module receives a single extra character, it will respond with a 'SYNTAX ERROR' and the command will be aborted. For example:

```
Command:  $1RD          (no checksum)
Response: *+00072.10

Command:  $1RDEB        (with checksum)
Response: *+00072.10

Command:  $1RDAB        (incorrect checksum)
Response: ?1 BAD CHECKSUM

Command:  $1RDE         (one extra character)
Response: ?1 SYNTAX ERROR
```

Response Checksums

If the long form ' # ' version of a command is transmitted to a module, a checksum will be appended to the end of the response. For example:

```
Command:  $1RD              (short form)
Response: *+00072.10

Command:  #1RD              (long form)
Response: *1RD+00072.10A4 (A4=checksum)
```

Checksum Calculation

The checksum is calculated by summing the hexadecimal values of all the ASCII characters in the message. The lowest order two hex digits of the sum are used as the checksum. These two digits are then converted to their ASCII character equivalents and appended to the message. This ensures that the checksum is in the form of printable characters.

Example: Append a checksum to the command #1DOFF

Characters:	#	1	D	O	F	F
ASCII hex values:	23	31	44	4F	46	46

Sum (hex addition) 23 + 31 + 44 + 4F + 46 + 46 = 173

The checksum is 73 (hex). Append the characters 7 and 3 to the end of the message: #1DOFF73

Example: Verify the checksum of a module response *1RD+00072.10A4

The checksum is the two characters preceding the CR: A4

Add the remaining character values:

*	1	R	D	+	0	0	0	7	2	.	1	0
2A+	31+	52+	44+	2B+	30+	30+	30+	37+	32+	2E+	31+	30 = 2A4

The two lowest-order hex digits of the sum are A4 which agrees with the transmitted checksum.

Note that the transmitted checksum is the character string equivalent to the calculated hex integer. The variables must be converted to like types in the host software to determine equivalency.

If checksums do not agree, a communications error has occurred.

If a module is setup to provide linefeeds, the linefeed characters are not included in the checksum calculation.

Parity bits are never included in the checksum calculation.

Table 4.1 M1000 Command Set

Command and Definition		Typical Command Message ($ prompt)	Typical Response Message
DI	Read Alarms/Digital Inputs	$1DI	*0003
DO	Set Digital Outputs	$1DOFF	*
ND	New Data	$1ND	*+00072.00
RD	Read Data	$1RD	*+00072.00
RE	Read Event Counter	$1RE	*0000107
RL	Read Low Alarm Value	$1RL	*+00000.00 L
RH	Read High Alarm Value	$1RH	*+00510.00 L
RS	Read Setup	$1RS	*31070142
RZ	Read Zero	$1RZ	*+00000.00
WE	Write Enable	$1WE	*
Write Protected Commands.			
CA	Clear Alarms	$1CA	*
CE	Clear Events	$1CE	*
CZ	Clear Zero	$1CZ	*
DA	Disable Alarms	$1DA	*
EA	Enable Alarms	$1EA	*
HI	Set High Alarm Limit	$1HI+12345.67L	*
LO	Set Low Alarm Limit	$1LO+12345.67L	*
RR	Remote Reset	$1RR	*
SU	Setup Module	$1SU31070142	*
SP	Set Setpoint	$1SP+00600.00	*
TS	Trim Span	$1TS+00600.00	*
TZ	Trim Zero	$1TZ+00000.00	*

M1000 User Commands

Note that in all command and response examples given below, a carriage return is implied after every character string.

Clear Alarms (CA)

The clear alarms command turns both the HI and LO alarms OFF. This command does not affect the enable/disable or momentary/latching alarm conditions. The alarms will continue to be compared to the input data after the CA command is given. In cases where the alarm condition persists, the alarms will be set at the end of the next input data conversion. The primary purpose of the CA command is to clear latching alarms. See the Alarms section for more information.

Command: $1CA
Response: *

Command: #1CA
Response: *1CADF

Clear Events (CE)

The Clear Events command clears the events counter to 0000000.

Command: $1CE
Response: *

Command: #1
Response: *1CEE3

Note: When the events counter reaches 9999999, it stops counting. A CE command must be sent to resume counting.

Clear Zero (CZ)

The Clear Zero command clears the output offset register value to +00000.00. This command clears any data resulting from a Trim Zero (TZ) or SetPoint (SP) command.

Command: $1CZ
Response: *

Command: #1CZ
Response: *1CZF8

Disable Alarms (DA)

Most models in the M1000 series feature LO/DO0 and HI/DO1 pins on the module connector. These pins serve a dual function and can be used to output either the alarm outputs or digital outputs 0 and 1. The Disable Alarms command is used to connect the digital outputs 0 and 1 to the connector pins. The alarm settings are not affected in any way except that the alarm outputs are disconnected from the module connector. The alarm status can still be read with the Digital Input (DI) command. The complement to the DA command is the Enable Alarms (EA) command.

Command: $1DA
Response: *

Command: #1DA
Response: *1DAE0

Digital Input (DI)

The DI command reads the status of the digital inputs and the alarms. The response to the DI command is four hex characters representing two bytes of data. The first byte contains the alarm status. The second byte contains the digital input data.

Command: $1DI
Response: *0003

Command: #1DI
Response: *1DI0003AB

Listed below are the four possible alarm states in the first digital input byte and their hex values.

00	Both HI and LO alarms off.
01	HI alarm off. LO alarm on.
02	HI alarm on. LO alarm off.
03	Both HI and LO alarms on.

The second byte displays the hex value of the digital input status. The number of digital inputs varies depending on module type.

Digital Inputs	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
Data Bits	7	6	5	4	3	2	1	0

For example: A typical response from a $1DI command could be: *01FE. This response indicates that the HI alarm is off, the LO alarm is on, DI0 = 0 and all other digital inputs are = 1.

All digital inputs that are not implemented or left unconnected are read as '1'.

Digital input 0 serves a dual function. It is both a digital input and the Event Counter input.

When reading digital inputs with a checksum, be sure not to confuse the checksum with the data.

Digital Output (DO)

The DO command controls eight bits of digital outputs on the module connector. The number of digital outputs implemented depends on the model used. The digital outputs allow the module to control external circuits under host command. The DO command requires an argument of two hex characters specifying the eight bits of output data.

Digital Outputs	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
Data Bits	7	6	5	4	3	2	1	0

The electrical implementation of the digital output consists of open-collector transistors wired to the module connector. If a digital output is set to '1' the corresponding transistor is turned on and sinks current. Note that when a digital output bit is set to '1' the electrical output is near 0 volts. If a digital output is set to '0' the corresponding transistor is turned off and sinks no current.

Assume a module has two digital outputs, and you wish to turn both outputs on (sinking current). Set data bit 0 and data bit 1 to '1'. Since the module has only two digital outputs, all the other bits are 'don't cares'. For example, this command will turn both outputs 'on':

Command: $1DOFF
Response: *

To turn both outputs off you could use the command:

Command: $1DO00
Response: *

Digital outputs 0 and 1 share connector pins with the HI and LO alarms. The Disable Alarms (DA) command is used to configure these pins as digital outputs.

Digital output settings are not stored in nonvolatile memory. If a power failure occurs, all digital outputs will be set to 0 upon power up.

The DO command is the only means of changing digital outputs. There is no software provision to read the state of the digital outputs.

Enable Alarms (EA)

Digital outputs DO0/LO and DO1/HI serve a dual purpose as digital outputs and alarms. Digital output 0 is shared with the LO alarm and digital output 1 is shared with the HI alarm. The Enable Alarms (EA) command configures the shared outputs to indicate alarm conditions and disconnects digital outputs 0 and 1. The EA command only affects the electrical output of the alarms to the pins. The alarm status can be read at any time with the Digital Input (DI) command. The complement to the EA command is the Disable Alarms (DA) command.

Command: $1EA
Response: *

Command: #1EA
Response: *1EAE1

High Alarm Limit (HI)

The high alarm command sets the value and type of the high alarm. The data specified by the HI command is stored in nonvolatile memory and compared with the sensor data after every A/D conversion. The high alarm is activated if the input data is greater than the value stored by the HI command. The high alarm status may be read using the Digital Input (DI) command. The alarm may be used to activate a digital output by using the Enable Alarms (EA) command. The HI command also specifies whether the high alarm is momentary or latching. A letter indicating the alarm type, "L" for latching or "M" for momentary, must follow the alarm value. For example:

Command: $1HI+00100.00M
Response: *

Command: #1HI+00100.00M
Response: *1HI+00100.00ME3

The alarm limit should be set within the output range of the module. If the alarm limit is set beyond the output range, the alarm will be activated only on an overload condition.

The high alarm value may be read back with the Read High Alarm (RH) command.

A latched alarm may be cleared with the Clear Alarms (CA) command. More information on alarms may be found in the Digital I/O section.

Low Alarm Limit (LO)

The low alarm command sets the value and type of the low alarm. The data specified with the LO command is stored in nonvolatile memory and compared with the sensor data after every A/D conversion. If the input data is less than the low limit, the low alarm is activated. The low alarm status may be read using the Digital Input (DI) command. The alarm may be used to activate a digital output by using the Enable Alarms (EA) command. A letter indicating the alarm type, "L" for latching or "M" for momentary, must follow the alarm value. For example:

```
Command:  $1LO+00000.00M
Response: *

Command:  #1LO+00000.00M
Response: *1LO+00000.00MEC
```

The alarm limit should be set within the output range of the module. If the alarm limit is set beyond the output range, the alarm will be activated only on an overload condition.

The low limit value may be read back with the Read Low Limit (RL) command.

More information on alarms may be found in the Digital I/O section.

New Data Command (ND)

The New Data (ND) command is a variation of the Read Data (RD) command used to read sensor data from the module. The ND command guarantees that the output data has not been previously read.

The M1000 module acquires analog input data eight times a second and stores the result in the output buffer (see Figure 2.1). The Read Data (RD) command simply reads the results stored in the output buffer. A fast host communicating at a high baud rate could possibly read the output buffer several times before the information is updated with a new A/D conversion. This results in redundant information which may be confusing or may be a waste of host processor time.

Associated with the output buffer is the New Data Flag (see Figure 2.1). This flag is cleared each time an RD or ND command is performed. The flag is set when the module's microprocessor loads the output buffer with the result of the most recent A/D conversion. The ND command will output data only when the New Data Flag is set. If the flag is cleared when an ND command is received, the module will wait until new data is present in the output buffer before responding to the command. Thus, the output data obtained with an ND command is always the result of a new A/D conversion.

The ND command is especially useful with computers that handle communications on an interrupt basis. The ND command may be used to get the maximum possible throughput without producing redundant data.

```
Command:  $1ND
Response: *+00072.00

Command:  #1ND
Response: *1ND+00072.009F
```

A special condition exists when using the ND command with the M1600 frequency/ pulse modules. These modules differ from the other sensor input modules in that they require an input trigger signal to obtain new data. If no signal exists on the input of the M1600, an ND command will wait indefinitely for new data and the module will not respond.

In order to escape this condition, a single control-C ($03) may be issued by the host to abort the ND command. The aborted ND command will respond with the data value currently stored in the output buffer. Be aware that on an RS-485 system, the control-C character may interfere with the ND output data, causing a communications collision.

Read Data (RD)

The read data command is the basic command used to read the buffered sensor data. The output buffer (Figure 2.1) allows the data to be read immediately without waiting for an input A/D conversion. For example:

```
Command:  $1RD
Response: *+00072.00

Command:  #1RD
Response: *1RD+00072.10A4
```

Since the RD command is the most frequently used command in normal operation, a special shortened version of the command is available. If a module is addressed without a two-letter command, the module interprets the string as an RD command.

```
Command:  $1
Response: *+00072.10

Command:  #1
Response: *1RD+00072.10A4
```

READ EVENTS (RE)

The Read Events command reads the number of events that have been accumulated in the Events Counter. The output is a seven-digit decimal number. For example:

Command: $1RE
Response: *0000107

Command: #1RE
Response: *1RE00001074A

The maximum accumulated count is 9999999. When this count is reached, the Events Counter stops counting. The counter may be cleared at any time with the Clear Events (CE) command.

The Event Counter count is not stored in nonvolatile memory. If power is removed, the Event Counter will reset to all 0's upon power up.

The Remote Reset (RR) command or a line break does not effect the value of the Event Counter.

When reading the Event Counter with a checksum, be sure not to confuse the checksum with the data.

Read High Alarm (RH)

The Read High alarm command reads the value and type of the high alarm previously loaded by the HI command. The alarm type can be either latching or momentary. A letter indicating the alarm type, "L" for latching or "M" for momentary, will follow the alarm value. For example:

Command: $1RH
Response: *+00510.00L

Command: #1RH
Response: *1RH+00510.00LF0

The RH command may be used to verify the data loaded into nonvolatile memory by the HI command.

Read Low Alarm (RL)

The Read Low alarm command reads the value and type of the low alarm. The alarm type can be either latching or momentary. A letter indicating the alarm type, "L" for latching or "M" for momentary, will follow the alarm value. For example:

Command: $1RL
Response: *+00000.00L

Command: #1RL
Response: *1RL+00000.00LEE

The RL command may be used to verify data loaded into the nonvolatile memory with the LO command.

Remote Reset (RR)

The reset command allows the host to perform a program reset on the module's microprocessor. This may be necessary if the module's internal program is disrupted by static or other electrical disturbances. Once a reset command is received, the module will recalibrate itself. The calibration process takes approximately 2 seconds. For example:

Command: $1RR
Response: *

Command: #1RR
Response: *1RRFF

In general, the state of the digital outputs and the event counter will not be affected by the RR command. However, if data in the microprocessor's RAM (Random Access Memory) has been lost, the RR command will result in a full power-up reset.

Any commands sent to the module during the self-calibration sequence will result in a NOT READY error.

Read Setup (RS)

The read setup command reads back the setup information loaded into the module's nonvolatile memory with the SetUp (SU) command. The response to the RS command is four bytes of information formatted as eight hex characters.

Command: $1RS
Response: *31070142

Command: #1RS
Response: *1RS3107014292

The response contains the module's channel address, baud rate, averaging constants, °C/°F, and other parameters. Refer to the setup command (SU), and the Setup section of this manual for a full list of parameters contained within the setup information.

When reading the setup with a checksum, be sure not to confuse the checksum with the setup information.

Read Zero (RZ)

The Read Zero command reads back the value stored in the Output Offset Register (Figure 2.1).

Command: $1RZ
Response: *+00000.00

Command: #1RZ
Response: *1RZ+00000.00B0

The data read back from the Output Offset Register may be interpreted in several ways. The commands that affect this value are the Trim Zero (TZ), SetPoint (SP), and Clear Zero (CZ).

Setpoint (SP)

The data specified by the setpoint command is multiplied by -1 and loaded into the Output Offset Register (Figure 2.1). The SP command is useful in on-off controller applications and is described in detail in the Digital I/O section of this manual. The SP command may be used to null out sensor data to obtain a deviation output when the RD or ND commands are used.

Command: $1SP+00450.00
Response: *

Command: #1SP+00450.00
Response: *1SP+00450.00B0

It is possible to load setpoint data that is beyond the output range of the sensor. In this case, the setpoint can never be reached by the sensor data unless an overload is present.

To clear a setpoint, use the Clear Zero (CZ) command.

The SP command will write over any data written into the Output Offset Register by the Trim Zero (TZ) command. If the Output Offset Register is used as a trim value, this must be accounted for by the host before using the SP command. The value stored in the offset register may be read back using the Read Zero (RZ) command.

The setpoint data or trim data in the Output Offset Register is saved in nonvolatile memory.

Setup Command (SU)

Each M1000 module contains an EEPROM (Electrically Erasable Programmable Read Only Memory) which is used to store module setup information such as address, baud rate, parity, etc. The EEPROM is a special type of memory that will retain information even if power is removed from the module. The EEPROM is used to replace the usual array of DIP switches normally used to configure electronic equipment.

The SetUp command is used to modify the user-specified parameters contained in the EEPROM to taylor the module to your application. Since the SetUp command is so important to the proper operation of a module, a whole section of this manual has been devoted to its description. See Section 5.

The SU command requires an argument of eight hexadecimal digits to describe four bytes of setup information:

Command: $1SU31070182
Response: *

Command: #1SU31070182
Response: *1SU3107018299

Trim Span (TS)

The trim span command is the basic means of trimming the accuracy of a M1000 sensor module. The TS command loads a calibration factor into nonvolatile memory to trim the full-scale output of the signal conditioning circuitry. It is intended only to compensate for long-term drifts due to aging of the analog circuits, and has a useful trim value of ±10% of the nominal calibration set at the factory. It is not intended to be used to change the basic transfer function of the module. Full information on the use of the TS command may be found in the Calibration section.

Command: $1TS+00500.00
Response: *

Command: #1TS+00500.00
Response: *1TS+00500.00B0

Caution! TS is the only command associated with the span trim. There is no provision to read back or clear erroneous information loaded by the TS command. Unwarranted use of the TS command may destroy the calibration of the unit which can only be restored by using laboratory calibration instruments in a controlled environment.

Trim Zero (TZ)

The Trim Zero command is used to load a value into the Output Offset Register (Figure 2.1) to null out an undesirable offset in the output data. It may be used to trim offsets created by sensors such as strain gages. It may also be used to null out data to create a deviation output.

Example: Assume a M1511 bridge input module is being used with a load delb r weight measurement. An inital reading of the load cell with no weight applied may reveal an initial offset error:

Command: $1RD
Response: *+00005.00

With no weight applied, we would like to trim the output to read zero. To trim the system, use the TZ command and specify the desired output reading:

Command: $1TZ+00000.00 (zero output)
Response: *

The TZ command will load a data value into the Output Offset Register to force the output to read zero. The module will compensate for any previous value loaded into the Output Offset Register. If another output reading is taken, it will show that the offset has been eliminated:

Command: $1RD
Response: *+00000.00

Although the TZ command is most commonly used to null an output to zero, it may be used to offset the output to any specified value. Assume thatwith the previously nulled load cell system we performed this command:

Command: $1TZ-000100.00
Response: *

The new data output with no load applied would be:

Command: $1RD
Response: *-000100.00

The load cell output is now offset by -100.

The offset value stored by the TZ command is stored in nonvolatile memory and may be read back with the Read Zero (RZ) command and cleared with the Clear Zero (CZ) command.

The SetPoint (SP) command will write over any value loaded by the TZ command.

Write Enable (WE)

Each MetraByte module is write protected against accidental changing of alarms, limits, setup, or span and zero trims. To change any of these write protected parameters, the WE command must precede the write-protected command. The response to the WE command is an asterisk indicating that the module is ready to accept a write protected command. After the write-protected command is successfully completed, the module becomes automatically write disabled. Each write-protected command must be preceded individually with a WE command. For example:

Command: $1WE
Response: *

Command: #1WE
Response: *1WEF7

If a module is write enabled and the execution of a command results in an error message other than WRITE PROTECTED, the module remains write enabled until a command is successfully completed resulting in an ' * ' prompt. This lets the user correct the command error without having to execute another WE command.

ERROR MESSAGES

The M1000 modules feature extensive error checking on input commands to avoid erroneous operation. Any errors detected will result in an error message and the command will be aborted.

All error messages begin with "?", followed by the channel address, a space, and the error description. The error messages have the same format for either the ' $ ' or ' # ' command prompts. For example:

?1 SYNTAX ERROR

There are eight possible error messages, and each error message description begins with a different character. This makes it easy for a computer program to identify the error without having to read the entire string.

ADDRESS ERROR

There are four ASCII values that are illegal for use as a module address: NULL ($00), CR ($0D), $ ($24), and # ($23). The ADDRESS ERROR will occur when an attempt is made to load an illegal address into a module with the SetUp (SU) command. An attempt to load an address greater than $7F will also produce an error.

BAD CHECKSUM

This error is caused by an incorrect checksum included in the command string. The module recognizes any two hex characters appended to a command string as a checksum. Usually a BAD CHECKSUM error occurs due to noise or interference on the communications line. In many cases, repeating the command will solve the problem. If the error persists, either the checksum is being calculated incorrectly or there is a problem with the communications channel. In some cases, more reliable transmissions may be obtained by using a lower baud rate.

COMMAND ERROR

This error occurs when the two-character command is not recognized by the module. Often this error results when the command is sent with lower-case letters. All valid commands are upper-case.

NOT READY

If a module is reset, it performs a self-calibration routine which takes 2-3 seconds to complete. Any commands sent to the module during the self-calibration period will result in a NOT READY error. When this occurs, simply wait a couple seconds and repeat the command.

The module may be reset in four ways: a power-up reset, a Remote Reset (RR) command, a line break, or an internal reset. All modules contain a 'watchdog' timer to ensure proper operation of the microprocessor. The timer may be tripped if the microprocessor is executing its program improperly due to power transients or static discharge.

If the NOT READY error persists for more than 30 seconds, check the power supply to be sure it is within specifications.

PARITY ERROR

A parity error can only occur if the module is setup with parity on (see Setup). Usually a parity error results from a bit error caused by interference on the communications line. Random parity errors are usually overcome by simply repeating the command. If too many errors occur, the communications channel may have to be improved or a slower baud rate may be used.

A consistent parity error will result if the host parity does not match the module parity. In this situation, the easiest solution may be to change the parity in the host to obtain communication. At this point the parity in the module may be changed to the desired value with the SetUp (SU) command.

The parity may also be changed or turned off by using the Default Mode.

SYNTAX ERROR

A SYNTAX ERROR will result if the structure of the command is not correct. This is caused by having too few or too many characters, signs or decimal points missing or in the wrong place. Table 4.1 lists the correct syntax for all the commands.

VALUE ERROR

This error results when an incorrect character is used as a numerical value. Data values can only contain decimal digits 0-9. Hex values used in the SetUp (SU) and Digital Output (DO) commands can range from 0-F.

WRITE PROTECTED

All commands that write data into nonvolatile memory are write-protected to prevent accidental erasures. These commands must be preceded with a Write Enable (WE) command or else a WRITE PROTECTED error will result.

CHAPTER 5
SETUP INFORMATION & COMMAND

The MetraByte M1000 modules feature a wide choice of user configurable options which gives them the flexibility to operate on virtually any computer or terminal based system. The user options include a choice of baud rate, parity, address, and many other parameters. The particular choice of options for a module is referred to as the setup information.

The setup information is loaded into the module using the SetUp (SU) command. The SU command stores 4 bytes (32 bits) of setup information into a nonvolatile memory contained in the MetraByte module. Once the information is stored, the module can be powered down indefinitely (10 years minimum) without losing the setup data. The nonvolatile memory is implemented with EEPROM so there are no batteries to replace.

The EEPROM has many advantages over DIP switches or jumpers normally used for option selection. The module never has to be opened because all of the options are selected through the communications port. This allows the setup to be changed at any time even though the module may be located thousands of feet away from the host computer or terminal. The setup information stored in a module may be read back at any time using the Read Setup command (RS).

The following options can be specified by the SetUp command:

Channel address (124 values)
Linefeeds
Parity (odd, even, none)
Baud rate (300 to 38,400)
Alarm enable/disable
Alarm momentary/latching
CJC disable (M1300 series)
RTD 3/4 wire (M1400 series)
Positive/negative edge trigger (M1600 series)
Fahrenheit/Celsius
Echo
Communication delay (0-6 characters)
Number of displayed digits
Large-signal filter constant
Small-signal filter constant

Each of these options will be described in detail below. For a quick look-up chart on all options, refer to Tables 5.1-4. Once you are completely familar with the setups, you can refer to Table 5.6 for a summary of all of the setup information.

Command Syntax

The general format for the SetUp (SU) command is:

$1SU[byte1][byte 2][byte 3][byte 4]

A typical SetUp command would look like: $1SU31070182.

Notice that each byte is represented by its two-character ASCII equivalent. In this example, byte 1 is described by the ASCII characters '31' which is the equivalent of binary 0011 0001 (31 hex). The operand of a SU command must contain exactly 8 hex (0-F) characters. Any deviation from this format will result in a SYNTAX ERROR. The Appendix contains a convenient hex-to-binary conversion chart.

For the purposes of describing the SetUp command, 'bit 7' refers to the highest-order bit of a byte of data. 'Bit 0' refers to lowest-order bit:

```
'bit number':   7 6 5 4   3 2 1 0
binary data:    0 0 1 1   0 0 0 1  = $31 (hex)
```

The SU command is write protected to guard against erroneous changes in the setup data; therefore each SU command must be preceded by a Write Enable (WE) command. To abort an SU command in progress, simply send a non-hex character (an 'X' for example) to generate a SYNTAX ERROR, and try again.

caution: Care must be exercised in using the SU command. Improper use may result in changing communications parameters (address, baud rate, parity) which will result in a loss of communications between the host and the module. In some cases the user may have to resort to using Default Mode to restore the proper setups. The recommended procedure is to first use the Read Setup (RS) command to to examine the existing setup data before proceeding with the SU command.

Byte 1

Byte 1 contains the module (channel) address. The address is stored as the ASCII code for the string character used to address the module. In our example command $1SU31070080 , the first byte '31' is the ASCII code for the character '1'. If our sample command is sent to a module, the EEPROM will be loaded with the address '1', which in this particular case remains unchanged. To change the module address to '2' , byte 1 of the SetUp command becomes '32', which is the ASCII code for the character '2'. Now the command will look like this: $1SU32070080. When this command is sent, the module address is changed from '1' to '2'.

The module will no longer respond to address '1'..

When using the SU command to change the address of a module, be sure to record the new address in a place that is easily retrievable. The only way to communicate with a module with an unknown address is with the Default Mode.

The most significant bit of byte 1 (bit 7) must be set to '0'. In addition, there are four ASCII codes that are illegal for use as an address. These codes are $00, $0D, $24, $23 which are ASCII codes for the characters NUL, CR, $, and #. Using these codes for an address will cause an ADDRESS ERROR and the setup data will remain unchanged. This leaves a total of 124 possible addresses that can be loaded with the SU command. It is highly recommended that only ASCII codes for printable characters be used ($21 to $7E) which greatly simplifies system debugging with a dumb terminal. Refer to Appendix A for a list of ASCII codes. Table 5.1 lists the printable ASCII codes that may be used as addresses.

Table 5.1 Byte 1 ASCII Printable Characters.

HEX	ASCII	HEX	ASCII	HEX	ASCII	HEX	ASCII
21	!	3A	:	51	Q	68	h
22	"	3B	;	52	R	69	i
25	%	3C	<	53	S	6A	j
26	&	3D	=	54	T	6B	k
27	'	3E	>	55	U	6C	l
28	(	3F	?	56	V	6D	m
29	)	40	@	57	W	6E	n
2A	*	41	A	58	X	6F	o
2B	+	42	B	59	Y	70	p
2C	,	43	C	5A	Z	71	q
2D	-	44	D	5B	[	72	r
2E	.	45	E	5C	\	73	s
2F	/	46	F	5D	]	74	t
30	0	47	G	5E	^	75	u
31	1	48	H	5F	_	76	v
32	2	49	I	60	`	77	w
33	3	4A	J	61	a	78	x
34	4	4B	K	62	b	79	y
35	5	4C	L	63	c	7A	z
36	6	4D	M	64	d	7B	{
37	7	4E	N	65	e	7C	|
38	8	4F	O	66	f	7D	}
39	9	50	P	67	g	7E	~

Byte 2

Byte 2 is used to configure some of the characteristics of the communications channel; linefeeds, parity, and baud rate.

Linefeeds

The most significant bit of byte 2 (bit 7) controls linefeed generation by the module. This option can be useful when using the module with a dumb terminal. All responses from the M1000 are terminated with a carriage return (ASCII $0D). Most terminals will generate a automatic linefeed when a carriage return is detected. However, for terminals that do not have this capability, the MetraByte module can generate the linefeed if desired. By setting bit 7 to '1' the module will send a linefeed (ASCII $0A) before and after each response. If bit 7 is cleared (0), no linefeeds are transmitted.

When using the '#' command prompt, the linefeed characters are not included in the checksum calculation.

Parity

Bits 5 and 6 select the parity to be used by the module. Bit 5 turns the parity on and off. If bit 5 is '0', the parity of the command string is ignored and the parity bit of characters transmitted by the module is set to '1'.

If bit 5 is '1', the parity of command strings is checked and the parity of characters output by the module is calculated as specified by bit 6.

If bit 6 is '0', parity is even; if bit 6 is '1', parity is odd.

If a parity error is detected by the module, it will respond with a PARITY ERROR message. This is usually caused by noise on the communications line.

If parity setup values are changed with the SU command, the response to the SU command will be transmitted with the old parity setup. The new parity setup becomes effective immediately after the response message from the SU command.

Baud Rate

Bits 0-2 specify the communications baud rate. The baud rate can be selected from eight values between 300 and 38400 baud. Refer to Table 5.2 for the desired code.

The baud rate selection is the only setup data that is not implemented directly after an SU command. In order for the baud rate to be actually changed, a module reset must occur. A reset is performed by sending a Remote Reset (RR) command, performing a line break (see Communications), or powering down. This extra level of write protection is necessary to ensure that communications to the module is not accidently lost. This is very important when changing the baud rate of an RS-232C string. For more information on changing baud rate, refer to the Communications section.

Let's run through an example of changing the baud rate. Assume our sample module contains the setup data value of '31070080'. Byte 2 is '07'. By referring to the SU command chart we can determine that the module is set for no linefeeds, no parity, and baud rate 300. If we perform the Read Setup command with this module we would get:

```
Command:   $1RS
Response:  *31070080
```

Let's say we wish to change the baud rate to 9600 baud. The code for 9600 baud is '010' (from Table 5.2). This would change byte 2 to '02'. To perform the SU command we must first send a Write Enable command because SU is write protected:

```
Command:   $1WE
Response:  *

Command:   $1SU31020080
Response:  *
```

This sequence of messages is done in 300 baud because that was the original baud rate of the module. The module remains in 300 baud after this sequence. We can use the Read Setup (RS) command to check the setup data:

```
Command:   $1RS
Response:  *31020080
```

Notice that although the module is communicating in 300 baud, the setup data indicates a baud rate of 9600 (byte 2 = '02'). To actually change the baud rate to 9600, send a Remote Reset (RR) command (RR is write protected):

```
Command:   $1WE
Response:  *

Command:   $1RR
Response:  *
```

Up to this point all communications have been sent at 300 baud. The module will not respond to any further communications at 300 baud because it is now running at 9600 baud. At this point the host computer or terminal must be set to 9600 baud to continue operation.

If the module does not respond to the new baud rate, most likely the setup data is incorrect. Try various baud rates from the host until the module responds. The last resort is to set the module to Default Mode where the baud rate is always 300.

Setting a string of RS-232C modules to a new baud rate requires special consideration. Refer to the Communications section for instructions.

Bits 3 and 4

These two bits of byte 2 are not used by the M1000 series and should be set to '0'.

Table 5.2 Byte 2: Linefeed, Parity and Baud Rate.

BYTE2

FUNCTION	DATA BIT							
	7	6	5	4	3	2	1	0
LINEFEED	1							
NO LINEFEED	0							
NO PARITY		0	0					
NO PARITY		1	0					
EVEN PARITY		0	1					
ODD PARITY		1	1					
NOT USED				X	X			
38400 BAUD						0	0	0
19200 BAUD						0	0	1
9600 BAUD						0	1	0
4800 BAUD						0	1	1
2400 BAUD						1	0	0
1200 BAUD						1	0	1
600 BAUD						1	1	0
300 BAUD						1	1	1

Byte 3

This byte contains the setup information for several seldom-used options. The default value for this byte is '01'.

Alarm Enable

Bit 7 determines if the outputs from the LO and HI alarms are connected to module terminal block. If the value is '0' the alarms are not connected to the terminal block. In this condition the outputs are controlled by the Digital Output (DO) command. If bit 7 is '1' the alarms are connected to the terminal block. This bit is also controlled by the Enable Alarms (EA) command which sets the bit to '1'. The Disable Alarms (DA) command clears the bit to '0'.

Low Alarm Latch

Bit 6 determines whether the LO Alarm is latching or momentary. A '1' indicates that the alarm is latching; '0' indicates a momentary alarm. Bit 6 is also controlled by the LO Alarm (LO) command.

High Alarm Latch

Bit 5 determines whether the HI Alarm is latching or momentary. A'1' indicates latching. Bit 5 is also controlled individually by the HI Alarm (HI) command.

Disable CJC
RTD 3/4 Wire
Trigger Edge Select

The setup information stored in bit 4 has different meanings depending on the M1000 model number.

Disable CJC; this functions pertains only to the M1300 series of thermocouple input modules. If the bit is set to '1' the Cold Junction Compensation is disabled. The module calculates the temperature output with a fixed cold junction temperature of 0°C. This setup is useful for calibrating the module or in cases where remote CJC is used. Normally this bit is cleared to '0'.

RTD 3/4 Wire; this functions pertains only to the M1400 series of RTD input modules. If the bit is set to '1', the module provides the correct lead-compensation calculation for 4-wire RTD's. If the bit is cleared to '0', the module calculates the correct lead compensation for 3-wire RTD's. Measurement errors may result if the module is not set to the correct sensor type.

Trigger Edge Select; this function pertains only to the M1600 series frequency and pulse modules. Bit 4 determines the polarity of the edge used to trigger the measurement cycle. If bit 4 is '1' then the measurement cycle is started on a positive-going edge. The measurement cycle is started on the negative-going edge if bit 4 is '0'. In general, this setup has very little effect on frequency inputs. It is primarily used with pulse inputs where the pulse train deviates from 50% duty cycle.

Celsius/Fahrenheit

The default scaling for temperature output modules is Celsius which is selected by making bit 3 = 0. To change the scaling to Fahrenheit, set bit 3 to '1'. All modules that do not have temperature output must have bit 3 cleared to zero. The scaling factors are operative only on the sensor data; HI and LO limits and setpoints must be modified by appropriate commands to reflect a scaling change (see Figure 2.1).

Echo

When bit 2 is set to '1', the M1000 module will retransmit any characters it has received on the communications line. This option is necessary to 'daisy-chain' multiple RS-232C modules. Echo is optional for systems with a single RS-232C module. Bit 2 must be cleared to '0' on RS-485 models. See the Communications section for a more complete description.

Delay

Bits 0 and 1 specify a minimum turn-around delay between a command and the module response. This delay time is useful on host systems that are not fast enough to capture data from quick-responding commands such as RD. This is particularly true for systems that use software UART's. The specified delay is added to the typical command delays listed in the Communications section. Each unit of delay specified by bits 0 and 1 is equal to time required to transmit one character with the baud rate specified in byte 2. For example, one unit of delay at 300 baud is 33.3 ms; for 38.4 kilobaud the delay is 0.26 ms. The number of delay units is selectable from 0 to 6 as shown in Table 5.3.

In some systems, such as IBM BASIC, a carriage return (CR) is always followed by a linefeed (LF). The M1000 modules will respond immediately after a command terminated by a CR and will ignore the linefeed. To avoid a communications collision between the linefeed and the module response, the module should be setup to delay by 2 units.

Table 5.3 Byte 3 Options.

BYTE 3

FUNCTION	DATA BIT							
	7	6	5	4	3	2	1	0
ALARMS OFF	0							
ALARMS ON	1							
HIGH ALARM MOMENTARY		0						
HIGH ALARM LATCHING		1						
LOW ALARM MOMENTARY			0					
LOW ALARM LATCHING			1					
CJC (D1300'S)				0				
NO CJC (D1300'S)				1				
3 WIRE (D1400'S)				0				
4 WIRE (D1400'S)				1				
- STARTING EDGE (D1600'S)				0				
+ STARTING EDGE (D1600'S)				1				
CELSIUS					0			
FAHRENHEIT					1			
NO ECHO						0		
ECHO						1		
NO DELAYS							0	0
2 BYTE TIME DELAYS							0	1
4 BYTE TIME DELAYS							1	0
6 BYTE TIME DELAYS							1	1

Byte 4

This setup byte specifies the number of displayed digits and the digital filter time constants.

Number of displayed digits

For ease of use, the data outputs of all M1000 modules are standardized to a common 7-digit output consisting of sign, 5 digits, decimal point, and two more digits. Typical output data looks like: +00100.00. However, best-case resolution of the A/D converter is 1 part in 50,000 or about 4 1/2 digits. In some cases, the resolution of the output format is much greater than the resolution of the measurement system. In such cases, the trailing digits of the response would display meaningless information. Bits 6 and 7 are used to insert trailing zeros into the output data to limit the output resolution and mask-off meaningless digits.

Bit 7	**Bit 6**		
0	**0**	**XXXX0.00**	**(4 displayed digits)**
0	**1**	**XXXXX.00**	**(5 displayed digits)**
1	**0**	**XXXXX.X0**	**(6 displayed digits)**
1	**1**	**XXXXX.XX**	**(7 displayed digits)**

For example, the M1411 model for RTD's has .1 degree output resolution. The appropriate number of digits for this module is 6, to mask off the .01 digit which has no meaningful data. In some cases, the user may want to limit the output resolution to 1 degree. To do this, select bits 6 and 7 to display 5 digits. With this selection, the rightmost-two digits will always be set to '0'.

The number of displayed digits affects only data received from an RD or ND command.

Large Signal Filter, Bits 3,4,5
Small Signal Filter, Bits 0,1,2

The M1000 series modules contain a versatile single-pole, low-pass digital filter to smooth out unwanted noise caused by interference or small signal variations. The digital filter offers many advantages over traditional analog filters. The filtering action is done completely in firmware and is not affected by component drifts, offsets, and circuit noise typically found in analog filters. The filter time constant is programmable through the SetUp (SU) command and can be changed at any time, even if the module is remote from the host.

The digital filter features separate time constants for large and small signal variations. The Large Signal Filter time constant is controlled by bits 3,4,5. This time constant is used when large signal variations are present on the input. The

Small Signal Filter time constant is controlled by bits 0,1,2. This filter time constant is automatically selected when input signal variations are small. The microprocessor in the MetraByte module automatically selects the correct filter constant after every A/D conversion. The constant selected depends on the magnitude of the change of the input signal and the setup for the number of digits displayed. The microprocessor always keeps the value of the last calculated output to compare to a new data conversion. If the new data differs from the last output by more than ten counts of the last displayed digit, the large signal time constant is used in the digital filter. If the result of the most recent A/D conversion differs from the last output value by less than ten counts of the last displayed digit, the small signal time constant is used. Let's look at an example:

The M1411 module for RTD's has a standard output resolution of .1 degrees. The standard number-of -displayed-digits setup for this module is 6 digits, specified by byte 4 of the setup data. Therefore, the large signal filter will be selected if a new input conversion differs from the previous value by more than 1.0 degree:

Previous data	**New data**	**Filter selected (large/small)**
+00100.00	+00100.50	small
+00100.00	+00101.50	large
+00100.00	+00099.90	small
+00100.00	+00098.90	large
-00050.50	-00050.00	small
-00050.50	-00060.00	small

If the number of displayed digits is changed to reduce output resolution, the filter selection is also affected. If the number of displayed digits in the previous example is changed to 5, the output resolution becomes 1.0 degree.

n this case the large signal time constant is used if the new reading differs from the old by more than 10.0 degrees:

Previous data	**New data**	**Filter selected (large/small)**
+00100.00	+00105.00	small
+00100.00	+00111.00	large
+00100.00	+00091.00	small
+00100.00	+00085.00	large
-00050.00	-00045.00	small
-00050.00	-00039.00	large

Large Signal Time Constant

The large signal filter time constant is specified by bits 3,4,5 of byte 4. It may be specified from 0 (no filter) to 16 seconds. The time constant for a first-order filter is the time required for the output to reach 63% of its final value for a step input.

Small Signal Time Constant

Bits 0,1,2 specify the filter time constant for small signals. Its values are similar to the ones for the large signal filter. Most sensors can benefit from a small amount of small signal filtering such as T = 0.5 sec. In most applications, the small signal time constant should be larger than the large signal time constant. This gives stable readings for steady-state inputs while providing fast response to large signal changes.

Table 5.4 Byte 4 Displayed Digits and Filter Time Constants.

BYTE 4

FUNCTION	DATA BIT							
	7	6	5	4	3	2	1	0
+XXXX0.00 DISPLAYED DIGITS	0	0						
+XXXXX.00 DISPLAYED DIGITS	0	1						
+XXXXX.X0 DISPLAYED DIGITS	1	0						
+XXXXX.XX DISPLAYED DIGITS	1	1						
NO LARGE SIGNAL FILTERING			0	0	0			
0.25 SECOND TIME CONSTANT			0	0	1			
0.5 SECOND TIME CONSTANT			0	1	0			
1.0 SECOND TIME CONSTANT			0	1	1			
2.0 SECOND TIME CONSTANT			1	0	0			
4.0 SECOND TIME CONSTANT			1	0	1			
8.0 SECOND TIME CONSTANT			1	1	0			
16.0 SECOND TIME CONSTANT			1	1	1			
NO SMALL SIGNAL FILTERING						0	0	0
0.25 SECOND TIME CONSTANT						0	0	1
0.5 SECOND TIME CONSTANT						0	1	0
1.0 SECOND TIME CONSTANT						0	1	1
2.0 SECOND TIME CONSTANT						1	0	0
4.0 SECOND TIME CONSTANT						1	0	1
8.0 SECOND TIME CONSTANT						1	1	0
16.0 SECOND TIME CONSTANT						1	1	1

Setup Hints

Until you become completely familiar with the SetUp command, the best method of changing setups is to change one parameter at a time and to verify that the change has been made correctly. Attempting to modify all the setups at once can often lead to confusion. If you reach a state of total confusion, the best recourse is to reload the factory setup as shown in Table 5.5 and try again, changing one parameter at a time. Use the Read Setup (RS) command to examine the setup information currently in the module as a basis for creating a new setup. For example:

Assume you have a M1111 unit and you wish to set the unit to echo so that it may be used in a daisy-chain (See Communications). Read out the current setup with the Read Setup command:

Command: $1RS
Response: *310701C2

By referring to Table 5.3, we find that the echo is controlled by bit 2 of byte 3. From the RS command we see that byte 3 is currently set to 01. This is the hexadecimal representation of binary 0000 0001. To set echo, bit 2 must be set to '1'. This results in binary 0000 0101. The new hexadecimal value of byte 3 is 05. To perform the SU command, use the data read out with the RS command, changing only byte 3:

Command: $1WE (SU is write-protected)
Response: *

Command: $1SU310705C2
Response: *

Verify that the module is echoing characters and the setup is correct.

By using the RS command and changing one setup parameter at a time, any problems associated with incorrect setups may be identified immediately. Once a satisfactory setup has been developed, record the setup value and use it to configure similar modules.

If you commit an error in using the SetUp command, it is possible to lose communications with the module. In this case, it may be necessary to use the Default Mode to re-establish communications.

The DA, EA, HI, and LO commands affect some of the bits of the setup data that are associated with alarms. If these commands are performed, the setup data read back with the Read Setup (RS) command may not correspond exactly with the data previously written with the SetUp (SU) command.

Table 5.5 Factory Setups by Model.

(All modules from the factory are set for address '1', 300 baud, no parity)

Model	Setup Message
M111X, M121X, M123X, M125X	310701C2
M112X, M124X	31070182
M113X, M114X	31070142
M13XX	31070142
M14XX	31070182
M15XX	310701C2
M16XX	310701C0
M170X	31070100