REVISIONS						
REV.	DESCRIPTION	EO#	DATE	ENG	QLTY	MFG
A	Production Release	3505	09-09-96	R.L.D.	W.C.	G.N.
B	Revised	3554	10-31-96	M.M.	M.P.I.	G.N.
C	Revised for v3.17 .. 3.21 code	4224	06-05-98	T.M.	M.P.I.	K.P.

NOTICE

UNIT
INSTRUMENTS, INC. 22600 Savi Ranch Parkway, Yorba Linda, California 92887

DWN:	J.L.	DATE:	08-30-96
ENG:	R.L.D.		09-09-96
QLTY:	W.C.		09-09-96
MFG:	G.N.		09-09-96

SYSTEM PERFORMANCE SPECIFICATION

Digitally Compensated Mass Flow Controller Firmware Specification Excerpts

DOC. # **SPS-004-0004** **REV C**

1. Purpose

This document describes the firmware embedded in the Digitally Compensated Mass Flow Controller (DCMFC), Versions 3.14 and following, based on the Unit Instruments 203 motherboard, Rev. A or later.

Note that this document contains excerpts from the original firmware specification, as well as sections written specifically for it. These excerpts follow the numbering of the original firmware specification. As a result some sections will have appeared to be omitted. These are sections that, in order to simplify this document and reduce the scope of the original document, have been omitted intentionally.

This document is intended for those wishing to configure or control Unit Instruments DCMFCs via the included RS-485 interface. It does **NOT** include sufficient detail to re-calibrate a unit, or to generate new calibration table entries from an existing entry. If such information is needed, please contact Unit Instruments for further details.

The DCMFC contains many attributes not documented here, but needed for calibration & service. Such attributes should **NOT** be changed without understanding the consequences. Please contact Unit Instruments if you need further details on such undocumented attributes.

1.1 Hardware versions

At the time of writing, the 203 PCB has gone through 7 revisions. For purposes of this spec, the only significant changes occurred with revision G. Rev. A through F were produced with only 2 EEPROMs, while most Rev. G and later were produced with all 4.

1.2 Firmware versions

At the time of writing, 5 versions of the firmware are outstanding:

3.14 Production code from 4/97 to 2/98. Supports 1 or 2 EEPROMs.

3.15 Production code from 2/98 to 4/98. Backwards-compatible update, with provision for sensor cooling recovery and flow over-range recovery.

3.17 Never used as production code. Backwards-compatible update, with provision to support 4 EEPROMs (20 Calibration & 9 SurrogateGas instances), the **OverFlowTimeout** and **TurnOnDelay** attributes, and **SensorCoolingRecovery*** attributes in mS.

3.20 Production code from 4/98. Updated for faster processing and a more compact EEPROM memory layout, thus supporting more gasses (26

Calibration and 9 SurrogateGas instances). Supports all enhancements of v3.17 except for the **TurnOnDelay** attribute.

3.21 Production code from 5/98. Minor bug fix, and adds **TurnOnDelay** attribute.

This specification does not address firmware prior to v3.14.

Version 3.17 can directly replace v3.14 through v3.15 in the field, without any re-calibration.

Versions 3.18 and 3.19 are reserved for EEPROM-compatible upgrades to v3.17 and earlier firmware. They have not been released as of this writing.

Version 3.20 changed the EEPROM memory allocation significantly, and cannot directly replace earlier versions. Updating to this version requires complete re-programming at a Unit Instruments service center, or transfer of all attributes using special software.

Version 3.21 can directly replace v3.20 in the field, without any re-calibration.

All versions of firmware are designed to work on all versions of the motherboard. However, firmware v3.17 or later is required to make use of more than 2 EEPROMs.

Where different versions of the firmware define attributes differently, such differences are noted here.

1.3 Future versions

Future Unit Instruments DCMFCs are likely to use different Class and Attribute IDs for some of the attributes described here, or to change the format or allowable range of such attributes. Programmers writing drivers to talk to the DCMFC over the 485 bus would be well advised to provide some sort of readily changeable mapping between an attribute name & value and the actual packet contents on the bus.

2. Introduction

The DCMFC is a direct replacement for several current Unit Instruments MFCs. It provides both improved operational characteristics (such as linearity), and improved manufacturability.

The DCMFC provides the following features:

A. Auto Zero Disable input

B. Alarm output

C. Configurable self-calibration (Sensor Zero)

D. Configurable range

E. Configurable low-setpoint speed-up

F. Configurable auto shut-off

G. Configurable auto zero

H. Configurable soft start

I. Configurable Override pin for either Unit Instruments (J-pin) or SEMI mode

J. Setpoint from 0 to 120% of rated flow

K. Auto Purge for setpoint >= 122% of rated flow

L. Non-volatile storage for up to 9 sets of calibration gas data (Calibration Gas, Conversion Factor, Range, Calibration constants, etc.) (up to 4 sets for firmware versions earlier than v3.17)

M. Non-volatile storage for up to 26 sets of process gas data (Calibration Gas selection, Process Gas, Range, Conversion Factor, etc.) (up to 9 sets for firmware earlier than v3.17, and up to 20 sets for firmware v3.17 .. v3.19)

N. Configuration, Control, & Monitoring via RS-485 interface from Unit Instruments software (Datacal or laptop)

O. Better tolerance than analog controllers for abuse by the tool, such as putting the controller in flow mode before turning on the gas, etc.

3. Quick references:

3.1 Digitally controlled set points

The controller must be set to a digital command mode in order to act upon a set point sent digitally. This is controlled by the variable **ControlMode**. **ControlMode** must be set to 1 to allow the flow controller to respond to digital set points. The controller may be set to power up in digital command mode (**ControlMode** = 1) by setting **DefaultControlMode** to 1.

Set points are sent using the **NewSetpoint** attribute.

The are 2 modes of digital setpoint control available in the DMFC. One mode allows a new set point to be executed as the set point command is received. The other mode allows the next set point to be stored until a message is received which commands the controller to act upon the previously sent and stored set point.

The variable that controls the set point control modes is named **FreezeFollow**. When **FreezeFollow** is set to 1 the **NewSetpoint** command is acted upon immediately when received. If **FreezeFollow** is set to 0 the new set point is stored but not executed until **FreezeFollow** is set to 1.

The **NewSetpoint** request takes values in the range of 0x4000 to 0xC000 which represent set points between 0% and 100% full scale. The linear relationship between Full Scale set points and **NewSetpoint** is demonstrated in the following table.

Full Scale % set point	NewSetpoint value (Hex)
0.0	4000
25.0	6000
50.0	8000

Full Scale % set point	NewSetpoint value (Hex)
75.0	A000
99.0	BEB8
100.0	C000

The “NewSetpoint” value may be calculated from the full scale percent value by:

“NewSetpoint” = (327.68 * full scale %) + 16,384

For example, for a 50% of Full Scale Setpoint:

327.68 * 50 = 16,384	Scale the message units to the total percentage
16,384 + 16,384 = 32,768	Add the offset

hex(32,768) = 0x8000 Convert to hexadecimal

Note that at the communications level all values are sent in binary. The decimal and hexadecimal formats shown are for convenience.

3.1.1 Digitally controlled setpoints example

For the purpose of sending the **ControlMode**, **FreezeFollow**, and **NewSetpoint** requests to the flow controller a packet follows specific formats and numerical examples as shown in the following tables. Note that the format will change in length and content accord to the type of command. These examples are intended to show the format for the **ControlMode**, **FreezeFollow**, and **NewSetpoint** messages explicitly.

	Request			
Field name	**DefaultControlMode = 1**	**ControlMode = 1**	**FreezeFollow = 0**	**FreezeFollow = 1**
MacId	**FF**	**FF**	**FF**	**FF**
STX	**02**	**02**	**02**	**02**
Command code	**81**	**81**	**81**	**81**
Packet Length - 6	**04**	**04**	**04**	**04**
Class ID	**69**	**69**	**69**	**69**
Instance ID	**01**	**01**	**01**	**01**
Attribute ID	**04**	**03**	**05**	**05**
Data	**01**	**01**	**00**	**01**
Pad	**00**	**00**	**00**	**00**
Checksum	**F6**	**F5**	**F6**	**F7**

Field name	Request			
	NewSetpoint = 0.25 (0% FS)	NewSetpoint = 0.50 (50% FS)	NewSetpoint = 0.745 (99% FS)	NewSetpoint = 0.75 (100% FS)
MacId	**FF**	**FF**	**FF**	**FF**
STX	**02**	**02**	**02**	**02**
Command code	**81**	**81**	**81**	**81**
Packet Length - 6	**05**	**05**	**05**	**05**
Class ID	**69**	**69**	**69**	**69**
Instance ID	**01**	**01**	**01**	**01**
Attribute ID	**A4**	**A4**	**A4**	**A4**
Data (lsb)	**00**	**00**	**B8**	**00**
Data (msb)	**40**	**80**	**BE**	**C0**
Pad	**00**	**00**	**00**	**00**
Checksum	**D6**	**16**	**0C**	**56**

Notes for both tables:

- Each column shows one request packet, with one byte of packet information in hexadecimal format per table cell.
- The **MacId** shown (FF) is a global address and should be replaced with the **MacId** for the individual flow controller. The **Checksum** must be changed accordingly.
- (lsb) = least significant byte.
- (msb) = most significant byte.

3.2 Most Frequently Used Attributes

ATTRIBUTE DESCRIPTION	REQUEST ATTRIBUTE	CHANGE ATTRIBUTE	NOMINAL
MacId (for Terminal program)	MFCAddr	MFCAddr ###	32-96,255
MacId of DMFC	MacId?	MacId=00	32-96
Process gas #	CurrentCalibration?	CurrentCalibration=#	1-9
Next setpoint to 0%	CurrentSetpoint?	NewSetpoint=.25	0.25 - 0.75
Next setpoint to 50%	CurrentSetpoint?	NewSetpoint=.50	0.25 - 0.75
Next setpoint to 100%	CurrentSetpoint?	NewSetpoint=.75	0.25 - 0.75
NewSetpoint effective immediately	FreezeFollow?	FreezeFollow=1	1 = immediate
Indicated flow	NormalizedFlow?	N/A	0.25 - 0.75
Filtered indicated flow	MFCOut?	(mfcFilterGain must be set)	0.25 - 0.75
mfcOut filtering (integration)	MFCFiltergain?	mfcFilterGain=0.200	0.200
Valve Voltage	ValveVoltage?	N/A	
Digital Control Mode	ControlMode?	ControlMode=1	Digital control = 1 Analog control = 2
AutoZero Function	AutoZeroEnable?	AutoZeroEnable=1	0 = off, 1 = on
Auto Zero enable delay	AutoZeroDelay?	AutoZeroDelay=90	90 (seconds)
Auto Zero Rate of operation	AutoZeroRate?	AutoZeroRate=0.000122	0.000122 (819 sec)
Auto shutoff threshold	AutoShutoffThreshold?	AutoShutoffThreshold=.25	.2575 = 1.5%
SoftStartRate SoftStartRate = .005 / Seconds to FS Flow	SoftStartRate?	SoftStartRate=0.5	0.5 = fast 0.002 = 2.5 sec 0.001 = 5.0 sec
User calibration trim	UserCalSpanMult(page) ?	UserCalSpanMult(page)=	1.0 (range 0.9-3.0)
Control Pin 2 state or 3 state	OverrideMode?	OverrideMode=0	0 = 2 state (on/off) 1 = 3 state (purge)
Valve Control Mode	ValveOverride?	ValveOverride=0	0 = Control,1 = Off, 2 = Purge
Exceptions	Exceptions?	Exceptions=0xffff (Reset)	0
Firmware Revision Level	FirmwareRevisionLevel?	FirmwareRevisionLevel=	XX.XX
Shutdown time after sensor cooling detection.	RecoveryTimeout?	RecoveryTimeout=	2000
mS of flow over-range before shutdown recovery.	OverFlowTimeout?	OverFlowTimeout=	750

"CurrentSetpoint" to %Setpoint relationships:

% = (**CurrentSetpoint** - .25) * 200 **CurrentSetpoint** = (% / 200) + 0.25

3.3 Alarms & Diagnostics

This list of diagnostics are all reported under the **Exceptions** attribute

Exception Name	Description	Bit #
Reset	Device has automatically reset after any error	0
azWarning	Device zero has drifted beyond preset warning level	1
azLimit	Device zero has drifted beyond preset limit level	2
BadTable	Downloaded process gas table is invalid, default table is active	3
Init	EEPROM contents are invalid, EEPROM has been reset	16
Comm	Communications routines have internally timed out	17
BadConfig	One or more configuration blocks have a CRC error	21,22,23
eeRead	There was an EEPROM read failure	24
eeWrite	There was an EEPROM write failure	25
eeVerify	There was an EEPROM verify failure	26
Timer	There was a reliability timer CRC error	27

Notes:

Diagnostic Initiation:

Complete memory CRC diagnostics are performed upon power up. Exceptions continue to be generated and reported during operation.

Alarms reporting mechanism:

All alarms are reported by polling the “exceptions” attribute. As a result all alarms may be monitored by polling the device once. A return of 0 indicates no alarms. Alarms do not broadcast onto the network spontaneously. Such “peer to peer” or “peer to master” network activity shall be supported when the broadcast conditions are defined by a customer and/or a standards committee.

Valve Voltage Alarms:

The valve voltage may be polled by the process tool to detect any changes. (Alarms based only upon valve voltage variations are not supported due to the erroneous alarms caused by the normal effects that pressure fluctuations have on valve voltages).

Set point to flow point Alarms:

“CurrentSetpoint” and “NormalizedFlow” attributes may be compared to generate flow point alarms within the process tool.

Manufacturing QA and exceptions prevention:

All manufacturing software constantly scans for and warns of any exceptions and does not allow the operator to proceed without first correcting the problem causing exception reporting. Exceptions are tested for during final QA of all digital devices. No devices leave the factory with exceptions.

4. Unit Instruments DMFC Electrical Connections

Connection is EIA-485 half duplex connected VIA a top mounted RJ12 Connector.

EIA-485 Function	RJ12 Pin #
Ground	1
DX - (minus)	3
DX+ (plus)	4

All other pins of the RJ12 connector are unused.

All data is transmitted in standard asynchronous serial format, 8 bits per character, no parity, 1 stop bit, at 9600 baud.

DO NOT USE STANDARD TELEPHONE STYLE RJ11 TO RJ11 CABLES.

When the EIA-485 connections are daisy chained by connecting from controller to controller all pin connections must be in parallel. Unit DMFC product provides two RJ12 connectors which may make it appear that you could use common RJ11 telephone style cables. These cables will not work because they reverse the tip and ring of the telephone line from end to end.

5. Device Modeling

The DCMFC is modeled as a collection of Objects. An Object provides an abstract representation of a particular component of the DCMFC.

A Class is the set of Objects representing the same kind of system component. An Object Instance (or Instance) is a specific (physical) occurrence of an Object within a Class. Each Instance of a Class has the same set of attributes, but its own particular set of attribute values.

A Class or an Object Instance has Attributes, provides Services, and implements a Behavior.

Attributes are characteristics of a Class or of an Object Instance. Typically, Attributes provide status information or govern the operation of an Object. Services are invoked to trigger the Object or Class to perform a task. Services correspond to messages which can be sent to an Object, causing it to either change state or to provide information about its current state. The Behavior of an Object indicates how it responds to particular events.

Within a network of DCMFCs, individual attributes are accessed via a hierarchical address consisting of:

MacID	An 8-bit identifier assigned to a particular MFC on the network. This value is assigned by the network administrator for the process tool containing the DCMFC.
ClassID	An 8-bit identifier assigned to a particular Class of objects. This value is assigned by the DCMFC architect, and is documented in this specification.
InstanceID	An 8-bit identifier which distinguishes the Class and individual Instances of the Class from each other. A value of 0 indicates the reference is to an attribute of the Class rather than an attribute of a particular Instance. Any other value indicates the reference is to an attribute of the specified Instance. The maximum number of Instances permitted of each class is determined by the DCMFC architect, and documented in this specification.
AttributeID	An 8-bit identifier assigned to a particular Attribute of the Class or Instance. This value is assigned by the DCMFC architect, and documented in this specification.
Service Code	An 8-bit identifier specifying the particular service desired from the DCMFC. The services available for a particular attribute are defined by the DCMFC architect, and are documented in this specification.

It is possible to direct a request to an attribute of the Class itself, rather than to an attribute of a specific instance within the class. This is done by using the **InstanceID** special value 0. **InstanceID** 0 is reserved to indicate a reference to the Class rather than to a specific instance.

5.1 Common Services

All communication with the DCMFC is done by service requests, each addressed to a specific **MacID, ClassID, InstanceID, & AttributeID**. The DCMFC provides only 3 services -- Read, Write, and Monitor. Read and Write are directly comparable to DeviceNet's "Get Attribute Single" and "Set Attribute Single" services. Monitor is (very) roughly comparable to a DeviceNet I/O Connection.

There are no special-purpose service requests, such as "Init" or "select a gas table". This will permit reasonably direct migration to DeviceNet and SEMI compatible objects for Datacal or other software needing network access to the DCMFC.

5.2 Access Rights

Each Attribute definition lists the types of access permitted, selected from the following:

R	**Read**	Attributes for which a single value can be obtained over the network via the Read service.
M	**Monitor**	Attributes for which a stream of successive values, at the rate specified by the DeviceManagement class **SampleRate** attribute, can be obtained over the network via the Monitor service.
W	**Write**	Attributes which can be changed over the network at any time via the Write service.
C	**Calibrate**	Attributes which can be changed over the network at any time via the Write service, but which should not be changed except during calibration. Most users should treat such attributes as read-only. Changing them without a thorough understanding of the consequences can cause significant problems.
I	**Initialize**	Attributes which are password protected, and which should never be changed outside a Unit Instruments service facility.

5.3 Attribute Formats

The DCMFC communicates and stores attributes in several different formats. Each format is described separately below.

Several attributes are stored internally in a different format from that used for network access. This is indicated in the "Format (Internal)" column of an attribute description table. When this column is blank, the attribute is stored and communicated in the same format.

Format Name	Description
Text **(TextN)**	Text attributes are sequences of up to N printable ASCII characters. Within the DCMFC, strings shorter than the specified number of characters are indicated by a terminating ASCII NUL character appended to the string, though that character is not considered part of the string. When transmitted, only the number of characters actually present are sent. The terminal NUL (if present) is not sent. Note that this format differs from that specified by DeviceNet, which includes the string length as a discrete value within the structure. They are transmitted left-most character first.
Integer **(IntN or UIntN)**	Integers (**IntN**) are two's complement binary numbers with N bits to the left of the decimal point, and 0 bits to the right. N must equal 8, 16, 24, or 32 externally. They are transmitted least-significant byte first. Unsigned integers (**UIntN**) are unsigned binary numbers with N bits to the left of the decimal point, and 0 bits to the right. N must equal 8, 16, 24, or 32 externally. They are stored and transmitted least-significant byte first.
Fraction **(UFractN)**	Fractions are unsigned binary numbers, with 0 bits to the left of the decimal point, and N bits to the right of the decimal point. N must be 8, 16, 24, or 32. They are transmitted least-significant byte first. Since fractions have a decimal point just to the left of the most-significant bit, they can express values in the range [0 .. 1). For example, the number 0.875, expressed as a **Fract8** number, would be (in binary) 11100000.
Fixed Point **(UFixedN.M)**	Fixed point values are unsigned binary numbers, with N bits to the left of the decimal point, and M bits to the right of the decimal point. N + M must equal 8, 16, 24, or 32. They are transmitted least-significant byte first. For example, the number 1.875, expressed as a **UFixed2.6** number, would be (in binary) 01111000.
Real **(Real32 or UReal32)**	Real values are IEEE-format 32-bit floating point numbers. They are transmitted least-significant byte of the mantissa first. They are never used directly, but may be converted to or from the internal floating point (**FloatN** or **UFloatN**) or other formats as needed. They are used for storage of information not used directly by the DCMFC, and as the network representation of a variety of attributes. A value specified as **URealN** is restricted to be equal to or greater than 0.

5.4 Attribute Storage

Attributes are stored and calculated in a variety of ways, depending on what causes a change in value. The class descriptions to follow use several storage classes:

Storage Class	Description
ROM	Attributes with a fixed value, stored in read-only memory. These values cannot be changed without replacing the microprocessor.
EEPROM	Attributes stored in non-volatile memory. These values are initialized by the DCMFC during EEPROM initialization. They retain their value even if power is removed from the DCMFC. Except where also labeled Volatile, they are updated only in response to network service requests.
RAM	Attributes stored in volatile memory. These values are initialized by the DCMFC following a Reset, and updated only in response to network service requests.
Volatile	Attributes typically stored in volatile memory, and updated by the DCMFC during normal operation. Most cannot be changed over the network. Many can be monitored, providing a near-continuous stream of values calculated at the **SampleRate**.
Temporary	Attributes stored only in registers. These values are calculated as needed by the DCMFC during normal operation, and discarded after use. They cannot be changed over the network, but can be monitored.

6. Object Classes

Conceptually, the DCMFC is composed of 11 public classes:

Class Name	Description
Identity	Identifies the device to other devices on the network
NetworkManagement	Responsible for all communication over the network interface
DeviceManagement	Provides general information about the DCMFC's current status
ExtendedConfiguration	Provides information about the DCMFC's construction, capabilities, and manufacture
ReliabilityTimer	Maintains timers to track reliability & failure-analysis related cumulative time since manufacture, and maintains EEPROM copies of critical RAM variables.
Exception	Processes error flags from the other functional classes, and generates appropriate alarm/warning indicators for both the network and the **Alarm** output signal.
Calibration	Provides information for each Process gas for which the DCMFC is calibrated.
SurrogateGas	Provides information for each Surrogate gas with which the DCMFC has been calibrated.
FlowMeter	Converts A/D converter readings to a standardized indicated flow measurement.
FlowController	Converts Setpoint A/D converter readings, Valve Mode from the Valve Driver, and a variety of network variables, into a setpoint for the Valve Driver.
ValveDriver	Combines valve override requests from several sources to select the valve operating mode, and generates the signals actually required by the valve driver hardware.

This section describes the major attributes and operation of each class.

Private variables (those not accessible over the network) are described only where necessary to define or understand operation of the DCMFC.

6.1 Identity Class (Class ID = 1)

The Identity class reports the type of device. It is used by external devices to select appropriate calibration algorithms, constants, etc. as needed for a particular device type. It is provided as a courtesy to external devices, and provides identifying information compatible with that provided by DeviceNet devices.

The DCMFC maintains exactly one instance of the Identity class.

6.1.1 Class Attributes

The Identity class provides no class attributes.

6.1.2 Instance Attributes

Attribute Name	Attribute ID	Access	Format	Storage
Vendor	**1**	**R I**	**UInt16**	**EEPROM**
Manufacturer ID code as assigned by the DeviceNet Developer's Group. Until Unit Instruments receives an assigned value, the value 0 is used to indicate "unknown".				
ProductType	**2**	**R I**	**UInt16**	**EEPROM**
Product Type code as assigned by the DeviceNet Developer's Group. Until the Developer's Group assigns a value for MFC/MFM devices, the value 0 is used to indicate "Generic Device".				
ProductCode	**3**	**R I**	**UInt16**	**EEPROM**
Product code as assigned by Unit Instruments to differentiate the 203-based DCMFC from other Unit Instruments devices with the same Product Type. Typical value: 203				
Revision	**4**	**R I**	**UInt16**	**EEPROM**
Revision code as assigned by Unit Instruments. The least-significant byte is the major revision, and is limited to the range [0 .. 127]. The most-significant byte is the minor revision. Typical value: 0				
ProductName	**7**	**R I**	**Text25**	**EEPROM**
Human-readable identification string. Typical value: "UII DCMFC 203"				

6.1.3 Behavior

The Identity class stores and reports instance attribute values as defined above. Specific instance attribute values do not affect operation of the DCMFC in any way.

6.2 NetworkManagement Class (Class ID = 3)

The NetworkManagement class is responsible for all communication over the network. The protocol used is described in section 8 below.

The DCMFC maintains exactly one instance of the NetworkManagement class.

6.2.1 Class Attributes

The NetworkManagement class provides no class attributes.

6.2.2 Instance Attributes

Attribute Name	Attribute ID	Access	Format	Range Min	Range Max	Storage
MacID	**1**	**R W**	**UInt8**	**0x20**	**0xEF**	**EEPROM / RAM**
Address of this device on the network. Initial value: 0x3F Any Write with a value outside the valid range [0x20 .. 0xEF] will be rejected. Most installations should restrict this attribute to the range [0x20 .. 0x5F]						

6.2.3 Behavior

The DCMFC responds to properly-formatted messages addressed to the specified **MacID**, or to **MacID** 0xFF. The DCMFC ignores messages addressed to any other **MacID**.

If two units on the RS-485 bus have the same **MacID** value, proper communication with either is likely to be impossible.

External devices should use caution when changing the **MacID** attribute value. If the external device misses the final ACK response from the DCMFC, it may be unaware whether or not the DCMFC has in fact changed its internal **MacID**. Further messages to the old **MacID** may go unanswered, because the DCMFC is waiting for messages addressed to the new **MacID**.

6.3 DeviceManagement Class (Class ID = 0x64)

The DeviceManagement class provides services common to MFC/MFM devices, as well as services which do not fit readily within one of the other available classes.

The DCMFC maintains exactly one instance of the DeviceManagement class.

6.3.1 Class Attributes

The DeviceManagement class provides no class attributes.

6.3.2 Instance Attributes

Variable Name	Attribute ID	Access	Format External	Format Internal	Range Min	Range Max	Storage
DeviceType	**1**	**R I**	**Text10**				**EEPROM**
Device Type This value indicates whether the device is a mass flow controller (value "MFC") or a mass flow meter (value "MFM").							
StandardRevLevel	**2**	**R I**	**Text9**				**EEPROM**
This value indicates which version of SEMI E54 the device meets. Since the SEMI specs require a DeviceNet interface, this attribute should normally have the value "00".							
DeviceManufacturer Identifier	**3**	**R**	**Text30**	**Text21**			**ROM**
Manufacturer's name Fixed value: "Unit Instruments Inc."							
ManufacturerModel Number	**4**	**R I**	**Text15**				**EEPROM**
Manufacturer’s device model number							
FirmwareRevision Level	**5**	**R**	**Text5**				**ROM**
Device firmware revision level -- of the form "3.20" for production firmware, or "3x20" for developmental versions.							
HardwareRevision Level	**6**	**R I**	**Text5**				**EEPROM**
Device hardware revision level – of the form “G”.							
SerialNumber	**7**	**R I**	**Text15**				**EEPROM**
Device serial number as assigned by Unit Instruments, and shown on the unit's label.							

DeviceConfiguration	**8**	**R I**	**Text30**				**EEPROM**
This value represents the basic device configuration in a compact form. Additional details required for calibration, etc. are available in the "Extended Configuration" classes described below.							
FirmwareID	**0xA0**	**R**	**Text15**				**ROM**
Indicates the firmware installed in the device. Fixed value: “MFC203”							
SampleRate	**0xA2**	**R**	**Real4**				**ROM**
This value indicates the nominal analog sample rate of the device in Hz. This is also the rate at which Monitor samples are produced. It is needed for calculating digital filter parameters. It is a fixed value of approximately 100.0, calculated from the design frequency of the crystal oscillator, and ignores any deviation of the oscillator frequency from its nominal value. **Note:** This value is likely to change in future devices and/or versions of the firmware. Calibration equipment or process tools should always use the actual value read from the unit in appropriate calculations, rather than simply assuming any particular value.							
Plus15	**0xA4**	**R M**	**UFract16**				**Temp**
Voltage at the +15V power input pin, represented as an offset fraction in the range [0 .. 1). A value of 0 corresponds to approximately -17.875V. A value of 1 corresponds to approximately +22.875V.							
Minus15	**0xA5**	**R M**	**UFract16**				**Temp**
Voltage at the -15V power input pin, represented as an offset fraction in the range [0 .. 1). A value of 0 corresponds to approximately -17.875V. A value of 1 corresponds to approximately +22.875V.							
PetName	**0xA6**	**R W**	**Text30**				**EEPROM**
User-provided name for the unit, for the convenience of application programs.							

6.3.3 Behavior

The DeviceManagement class stores and reports instance attribute values as defined above. Specific instance attribute values do not affect operation of the DCMFC in any way.

6.4 ExtendedConfiguration Class (Class ID = 0xA0)

The ExtendedConfiguration class provides storage for config data for use by external applications. It is intended to record information about original manufacture & calibration of the device, which may be needed during maintenance or re-calibration.

The DCMFC supports 147 instances of the class. Each instance has only 1 attribute. Unlike all other classes, the format of instance attributes varies from one instance to another.

6.4.1 Class Attributes

Attribute Name	Attribute ID	Access	Format	Storage
ecVariableSet	**1**	**R I**	**UInt16**	**EEPROM**
Specifies the contents of the ecValue fields. A value of 0 specifies the parameter set described in Appendix A of this document. Other values indicate later revisions.				

6.4.2 Instance Attributes for ecVariableSet = 0

Attribute Name	InstanceID	AttributeID	Access	Format	Storage
CustomerNumber	**121**	**1**	**R I**	**Text12**	**EEPROM**
Customer's part number for the unit					
SalesOrderNumber	**122**	**1**	**R I**	**Text12**	**EEPROM**
Sales order number for the unit					
CsrIdentifier[4]	**97 .. 100**	**1**	**R I**	**Text8**	**EEPROM**
Text strings identifying any CSRs incorporated in this unit.					
SealMaterial	**123**	**1**	**R I**	**Text12**	**EEPROM**
Material used for the unit's seals, such as "Teflon"					
SeatMaterial	**124**	**1**	**R I**	**Text12**	**EEPROM**
Material used for the valve seat, such as "Teflon"					
SurfaceFinish	**9**	**1**	**R I**	**Float**	**EEPROM**
Interior surface finish of the block, as originally manufactured, in microinches.					

Attribute Name	InstanceID	AttributeID	Access	Format	Storage
Note: this value is NOT guaranteed or updated on refurbished or repaired units.					
ElectronicsType	**58**	**1**	**R I**	**Text4**	**EEPROM**
Type and revision of the unit's electronics, such as "203A"					
Jet	**59**	**1**	**R I**	**Text4**	**EEPROM**
Identifier of the unit's valve jet, such as "2"					
Inlet	**125**	**1**	**R I**	**Text12**	**EEPROM**
Inlet pressure range, PSI, typically "10-30 PSIG"					
Exhaust	**126**	**1**	**R I**	**Text12**	**EEPROM**
Nominal exhaust pressure, typically "Atm"					
Attitude	**60**	**1**	**R I**	**Text4**	**EEPROM**
Mounting attitude for which the unit is designed, such as "VIU"					
ScreenMesh	**25**	**1**	**R I**	**Ureal32**	**EEPROM**
Filter mesh (in Wires/Inch) used at the MFC input.					

6.4.3 Behavior

The ExtendedConfiguration class stores and reports instance attribute values as defined above. Specific instance attribute values do not directly affect operation of the DCMFC.

6.5 ReliabilityTimer Class (Class ID = 0xA1)

The ReliabilityTimer class provides timers used to track total time since the device was manufactured of various conditions. Each instance of the class maintains one, 32-bit timer. The timers are initialized to zero when the firmware first initializes the EEPROM, incremented as needed, and (absent an exceptionally unlucky EEPROM or processor failure) never reset or decremented. Their primary use is to provide reliability data.

The DCMFC maintains 2 instances of the class.

Instance 1 tracks total time the unit has been powered up, and is incremented once per second whenever the unit is powered up.

Instance 2 tracks time the unit has been flowing a controlled amount of gas, and is incremented once per second whenever the valve is attempting to flow a controlled amount of gas -- i.e. is not in Off or Purge mode.

6.5.1 Class Attributes

The ReliabilityTimer class provides no class attributes.

6.5.2 Instance Attributes

Attribute Name	Attribute ID	Access	Format	Storage
CumulativeTime	**1**	**R**	**UInt32**	**EEPROM / Volatile**
Accumulated time, in seconds, from manufacture to the present. Sum of the private variables **eeTimerSum** and **TimerDelta** below.				
TimerName	**2**	**R**	**Text30**	**ROM**
Name of this timer instance. In the DCMFC, Instance 1: "Power On Time" In the DCMFC, Instance 2: "Controlled Flow Time"				

6.6 Exception Class (Class ID = 0x65)

The Exception class reports error conditions over the network and via the Alarm output signal, and shuts the MFC down when specified errors are detected.

The DCMFC maintains one instance of the class.

6.6.1 Class Attributes.

The Exception class provides no class attributes.

6.6.2 Instance Attributes

Variable Name	Attribute ID	Access	Format	Storage
Exceptions	**0xA0**	**R W**	**UInt32**	**See description**
This attribute consists of 32 error flags, each indicating a particular condition has been detected. Individual flags are described under Exception Flag Descriptions below. A bit is set whenever the uP detects the corresponding error condition. A Write to this attribute of a value with specific bits set will clear the corresponding bits of **Exceptions** -- i.e., if **Exceptions** reports the value 0x0003, and the value 0x0001 is written, **Exceptions** will return the value 0x0002 the next time it is read (assuming no additional exceptions are detected in the meantime). Note, however, that some errors are checked for only during initialization, and the corresponding bits may remain cleared even though the condition causing the exception is still present. The high-order 16 bits of this value are also maintained in EEPROM, and are preserved through power failures.				
AlarmMask	**0xA1**	**R W**	**UInt32**	**EEPROM**
This value specifies which conditions reported in **Exceptions** shall be considered "Alarms" and shut down the DCMFC, as described under “Behavior” below.				
WarningMask	**0xA2**	**R W**	**UInt32**	**EEPROM**
This value specifies which conditions reported in **Exceptions** shall be considered "Warnings" and activate the **Alarm** output, as described under “Behavior” below.				

6.6.3 Exception Flags:

Exception	Bit	Description
Reset	**0**	Device has automatically reset after any error
azWarning	**1**	**sensorCurrentZero** has reached **AutoZeroWarning**. Device should be zeroed or re-calibrated.
azLimit	**2**	**sensorCurrentZero** has reached **AutoZeroLimit**. Device must be zeroed or re-calibrated.
BadTable	**3**	The selected Calibration instance linearization table is invalid. A default table, probably not good enough for any serious use of the DCMFC, is now in effect (generally indicates an EEPROM failure, an external application error, or a power failure during calibration).
Init	**16**	EEPROM contents were determined to be invalid (see Timer Set Selection), so the EEPROM has been completely initialized (see EEPROM Initialization) (generally indicates an EEPROM failure).
Comm	**17**	Communication routines have timed out waiting for a response from the service request processor (generally indicates an EEPROM failure).
BadConfig	**21**	One or more Configuration blocks in EEPROM has a CRC error (generally indicates an EEPROM failure, or a power failure during configuration at the factory)
BadCalParm	**22**	One or more Calibration blocks in EEPROM has a CRC error (generally indicates an EEPROM failure, a power failure during calibration, or a change in **nSurrogateGasses** without updating the new Calibration or SurrogateGas instances).
BadParm	**23**	Some EEPROM parameter is invalid (generally indicates an external application error, an EEPROM failure, or a power failure while changing an EEPROM parameter).
eeRead	**24**	An EEPROM read operation failed (generally indicates an EEPROM failure).
eeWrite	**25**	An EEPROM write operation failed (generally indicates an EEPROM failure).
eeVerify	**26**	An EEPROM verify operation failed (generally indicates an EEPROM failure).
Timer	**27**	A ReliabilityTimer block had a bad CRC (typically due to premature power failure during a timer block write. Generally indicates an EEPROM or switching power supply failure).

6.6.4 Behavior

The Exception class reports various abnormal conditions. **Exceptions** is a 32-bit value in which each bit represents a particular condition, as described above.

Exceptions is combined with two other attributes, **AlarmMask** and **WarningMask**, to determine how the DCMFC will behave when an exception is detected. **Exceptions** can cause activation of the Alarm output, shut down the MFC, or both.

If both an **Exceptions** bit and the corresponding **AlarmMask** bit are set, the DCMFC immediately returns the valve to its power-off condition.

Any bit set in **AlarmMask** should normally also be set in **WarningMask**.

If both an **Exceptions** bit and the corresponding **WarningMask** bit are set, the DCMFC activates the **Alarm** output signal, but otherwise continues to function normally.

The high-order 16 bits of **Exceptions**, used to report possible firmware or hardware problems, are maintained in EEPROM.

6.7 Calibration Class (Class ID = 066h)

The Calibration class contains information needed by other classes for accurate flow measurement and control.

The number of instances available in a particular unit is **nProcessGasses**, described below.

Each Calibration instance contains values needed by the FlowMeter class to calculate the actual gas flow rate for a particular process gas and flow range from sensor readings. It also contains information about the both the process gas and surrogate gas for use during future re-calibration, and a reference to the appropriate SurrogateGas instance from which the operational parameters were calculated. Calibration instance attribute values are calculated from the corresponding SurrogateGas attributes, not measured directly.

6.7.1 Class Attributes

Attribute Name	Attribute ID	Services	Format External	Format Internal	Range Min	Storage
CurrentCalibration	**0x65**	**R W**	**UInt16**	**UInt8**	**1**	**EEPROM**
This attribute selects which Calibration instance is to be used for flow metering. If the specified instance does not exist, or has its **GasIdentifier** empty, writing to this variable has no effect.						
nProcessGasses	**0xA0**	**R**	**UInt8**		**1**	**EEPROM**
This attribute indicates the available number of Calibration instances. It is calculated as needed from the amount of memory available and **nSurrogateGasses**. See the SurrogateGas class Behavior description below for further details.						

6.7.2 Instance Attributes

Attribute Name	Attribute ID	Services	Format External	Format Internal	Range Min	Range Max	Storage
GasIdentifier	**1**	**R C**	**Text15**				**EEPROM**
When the process gas is one of those listed in SEMI E52-95, this attribute is the Symbol from E52-95 for that gas. For example, for Bromine Trifluoride, this attribute would be "BrF3". When the process gas is not one of those listed in SEMI E52-95, or is a mixture, this attribute is a name chosen to uniquely identify that gas or mixture. If this value is empty (""), the instance is considered to be available, other attributes of the instance will not be accessible, and it will not be possible to set **CurrentCalibration** to refer to the instance. It is not possible to set this value to empty if **CurrentCalibration** refers to							

Attribute Name	Attribute ID	Services	Format External	Format Internal	Range Min	Range Max	Storage
the instance.							
FullScaleRange	**2**	**R C**	**UInt16**		**1**		**EEPROM**
Full-scale range of the process gas. LSB = 0.1 **FlowUnits**.							
FlowUnits	**3**	**R C**	**Text8**				**EEPROM**
Engineering units used to specify **FullScaleRange** -- typically "slm" or "sccm".							
ReferenceTemperature	**4**	**R C**	**Int16**				**EEPROM**
Indicates the reference temperature used in defining **FlowUnits**. It is expressed as temperature in degrees Celsius, with an LSB of 0.1 C. For the semiconductor industry, this value is defined by SEMI E12 as 0 C, or a value of 0.							
ReferencePressure	**5**	**R C**	**Int16**				**EEPROM**
Indicates the reference pressure used in defining **FlowUnits**. It is expressed as pressure minus 25 Pascal, with an LSB of 100 Pa. For the semiconductor industry, this value is defined by SEMI E12 as 101.325 kPa, or a value of 1013.							
UserCalSpanMult	**6**	**R W**	**Ufixed 2.14**		**0.9**	**3.0**	**EEPROM**
This is a user-supplied multiplier to be applied to the normalized flow reading as part of the Span Adjustment algorithm. The DCMFC may reject any value outside the range (0.90 .. 3.0) [14746 .. 49152].							
UserCalZeroOffset	**7**	**R W**	**Int16**		**-1,600**	**1,600**	**EEPROM**
This is a user-supplied offset to be added to the normalized flow reading as part of the Span Adjustment algorithm. It is expressed as a percentage of full scale, with an LSB of 0.025%. Thus, the normal value of 0 corresponds to no user adjustment to the normal calibration data. The DCMFC may reject any value outside the range [-1600 .. +1600]. **Use of any value other than 0 is strongly discouraged.** This attribute will almost certainly be removed from a future version of the DCMFC. If the sensor has drifted, the unit should be re-zeroed through use of the **Calibrate** attribute or switch, or by enabling auto zero. If the flow output D/A or amplifier has drifted, that drift should be corrected by re-calibrating the Flow output circuitry.							
SurrogateGasID	**8**	**R**	**Text15**				**EEPROM**
Value, from the SurrogateGas instance specified by **SurrogateGasInstance** below, of **sgGasIdentifier**.							
DateCalibrated	**9**	**R**	**Text8**				**EEPROM**
Value, from the SurrogateGas instance specified by **SurrogateGasInstance** below, of **sgDateCalibratedd**.							
TimeCalibrated	**0x0A**	**R**	**Text6**				**EEPROM**
Value, from the SurrogateGas instance specified by **SurrogateGasInstance** below, of **sgTimeCalibratedd**.							
DateCalibrationDue	**0x0B**	**R**	**Text8**				**EEPROM**
Value, from the SurrogateGas instance specified by **SurrogateGasInstance** below, of							

Attribute Name	Attribute ID	Services	Format External	Format Internal	Range Min	Range Max	Storage
sgDateCalibrationDue.							
CalibratedBy	**0x0C**	**R**	**Text15**				**EEPROM**
Value, from the SurrogateGas instance specified by **SurrogateGasInstance** below, of **sgCalibratedBy**.							
CalibratedAt	**0x0D**	**R**	**Text15**				**EEPROM**
Value, from the SurrogateGas instance specified by **SurrogateGasInstance** below, of **sgCalibratedAt**.							
SurrogateGasInstance	**0x16**	**R C**	**UInt8**		**1**		**EEPROM**
InstanceID of the SurrogateGas instance used to derive the calibration parameters for this Calibration instance.							
ConversionFactorN2	**0x17**	**R C**	**UReal32**				**EEPROM**
Nominal conversion factor of the process gas to Nitrogen. For example, if the Process gas is Arsenic Pentafluoride, this attribute is approximately 0.332.							
ConversionFactor	**0x18**	**R C**	**UReal32**				**EEPROM**
Nominal conversion factor of the process gas to the specified surrogate gas. For example, if the Process gas is Arsenic Pentafluoride, and the Surrogate gas is SF6, this attribute is approximately 1.287.							

6.7.3 Behavior

The class attribute **CurrentCalibration** (defined above) selects a Calibration instance for use by the DCMFC.

6.8 SurrogateGas Class (Class ID = 0xA3)

The SurrogateGas class contains information needed for maintenance of the Calibration and SurrogateGas classes, and calibration data from actual measurements with specified surrogate gasses flowing through the device.

Each instance of the SurrogateGas class contains information about one gas with which the DCMFC has been calibrated, and which is needed to calculate attributes for any derived Calibration instances. Values are either properties of the gas, or have been calculated from direct measurements with the specified surrogate gas flowing. Instance attribute values are not used directly by the DCMFC.

The number of instances available in a particular unit is **nSurrogateGasses**, described below.

Other than the class attribute **nSurrogateGasses**, the SurrogateGas class is not used directly by the DCMFC; rather it provides default values for corresponding attributes in the Calibration class.

6.8.1 Class Attributes

Attribute Name	Attribute ID	Access	Format	Range (Min)	Storage
nSurrogateGasses	**0xA0**	**R C**	**UInt8**	**1**	**EEPROM**
This attribute indicates the available number of SurrogateGas instances. Changing it will affect **nProcessGasses**. Increasing it may delete high-numbered instances from the Calibration class. Decreasing it may delete high-numbered instances from the SurrogateGas class. See Behavior (below) for further details. If the specified value is outside the allowable range, writing to this variable has no effect.					

6.8.2 Instance Attributes

Attribute Name	Attribute ID	Access	External	Internal	Max	Storage
sgGasIdentifier	**1**	**R C**	**Text15**			**EEPROM**
When the surrogate gas is one of those listed in SEMI E52-95, this attribute is the Symbol from E52-95 for that gas. For example, for Bromine Trifluoride, this attribute would be "BrF3". When the surrogate gas is not one of those listed in SEMI E52-95, or is a mixture, this attribute is a name chosen to uniquely identify that gas or mixture. If this value is empty (""), the instance is considered to be available, and other attributes of the instance will not be accessible.						

Attribute Name	Attribute ID	Access	External	Internal	Max	Storage
sgLinearization Range	**2**	**R C**	**UReal32**			**EEPROM**
Linearization range of the surrogate gas in sccm.						
sgReference Temperature	**4**	**R C**	**Int16**			**EEPROM**
Indicates the reference temperature used in defining sccm. It is expressed as temperature in degrees Celsius, with an LSB of 0.1 C. For the semiconductor industry, this value is defined by SEMI E12 as 0 C, or a value of 0.						
sgReferencePressure	**5**	**R C**	**Int16**			**EEPROM**
The second indicates the reference pressure used in defining sccm. It is expressed as pressure minus 25 Pascal, with an LSB of 100 Pa. For the semiconductor industry, this value is defined by SEMI E12 as 101.325 kPa, or a value of 1013.						
sgDateCalibrated	**9**	**R C**	**Text8**			**EEPROM**
Date the MFC was calibrated, in MMDDYY or MMDDYYYY format. Use of the 8-character date format is strongly recommended for all dates, and is mandatory for any date January 1, 2000 or later.						
sgTimeCalibrated	**10**	**R C**	**Text6**			**EEPROM**
Time the MFC was calibrated, in HHMMSS (24-hour) format.						
sgDateCalibrationDue	**11**	**R C**	**Text8**			**EEPROM**
Date the MFC should be re-calibrated, in MMDDYY or MMDDYYYY format. Use of the 8-character date format is strongly recommended for all dates, and is mandatory for any date January 1, 2000 or later.						
sgCalibratedBy	**12**	**R C**	**Text15**			**EEPROM**
Name of the individual who calibrated the MFC.						
sgCalibratedAt	**13**	**R C**	**Text15**			**EEPROM**
Name of the facility at which the MFC was calibrated.						
sgStandardID	**0x1F**	**R C**	**Text15**			**EEPROM**
Identifier of the ROR or other reference used to calibrate the DCMFC.						
sgInletPressure	**0x20**	**R C**	**Real32**			**EEPROM**
Average inlet pressure during calibration, in PSI (Gauge).						
sgOutletPressure	**0x21**	**R C**	**Real32**			**EEPROM**
Average outlet pressure during calibration, in PSI (Gauge).						

6.8.3 Behavior

A fixed area of EEPROM is shared between Calibration and SurrogateGas instances. The size of this area is dependent on the amount of memory installed, as is the possible number of instances in each class. Since the memory is shared between classes, changing the number of available

instances in one will also change the number of available instances in the other.

Table 6.8.3.1. **nProcessGasses** for versions prior to 3.20:

		Number of EEPROMs			
		1	2	3*	4*
nSurrogateGasses	1	4	12	20	28
	2	3	11	19	27
	3	2	10	18	26
	4	1	9	17	25
	5		8	16	24
	6		7	15	23
	7		6	14	22
	8		5	13	21
	9		4	12	20

* Not permitted prior to v3.17

Table 6.8.3.2. **nProcessGasses** for versions 3.20 & later:

		Number of EEPROMs			
		1	2	3*	4*
nSurrogateGasses	1	5	15	26	37
	2	4	14	25	35
	3	2	13	23	34
	4	1	12	22	33
	5		10	21	31
	6		9	20	30
	7		8	18	29
	8		7	17	28
	9		5	16	26

6.9 FlowMeter Class (Class ID = 0x68)

The FlowMeter class is responsible for converting sensor measurements into true flow measurements, given a set of calibration parameters provided by the Calibration class.

The DCMFC maintains 1 instance of the class.

6.9.1 Class Attributes

The FlowMeter class provides no class attributes.

6.9.2 Instance Attributes

Attribute Name	Attribute ID	Access	Format External	Format Internal	Range Min	Range Max	Storage
AutoZeroEnable	**0xA5**	**R W**	**UInt8**				**EEPROM / RAM**
If non-zero, enables the Auto Zero feature.							
AutoZeroDelay	**0xA6**	**R W**	**UInt8**		**1**		**EEPROM / RAM**
Represents the delay (in seconds) between valve closure and the first Auto Zero update. Typically 90.							
AutoZeroRate	**0xA7**	**R W**	**UReal32**	**UFloat8**	**1E-5**	**.01**	**EEPROM**
Rate multiplier constant R needed for the sensor zero filter during an auto-zero operation. Note that the zero filter is applied to ten samples per second during auto-zero. Given the desired time constant in seconds, it is calculated as: **AutoZeroRate** = 1 / (10 * desired time constant) Typically 0.000122 (2^-13), => time constant = 819 seconds.							
RequestedZeroRate	**0xA8**	**R W**	**UReal32**	**UFloat8**	**1E-3**	**.1**	**EEPROM**
Rate multiplier constant R needed for the sensor zero filter during a requested zero operation. Given the desired time constant in seconds, it is calculated as: **RequestedZeroRate** = 1 / (**SampleRate** * desired time constant) Typically .00195 (2^-9), => time constant = 5.12 seconds).							
SensorCurrentZero	**0xA9**	**R**	**Ufract32**				**Volatile**
Equivalent to **SensorReferenceZero**, but is updated periodically by the auto-zero routines. It is subtracted from each Sensor measurement before use. A copy of **SensorCurrentZero** is maintained in EEPROM by the ReliabilityTimer class (see							

Attribute Name	Attribute ID	Access	Format External	Format Internal	Range Min	Range Max	Storage
the **eeSensorCurrentZero** private variable).							
SensorReferenceZero	**0xAA**	**R W**	**Ufract16**				**EEPROM / Volatile**
The Sensor signal measured by the DCMFC A/D during a Zero operation (initiated either by the Calibrate* pin being grounded, or by the **Calibrate** attribute, **calSensorZero** bit being set). It is not updated during auto-zero operation. It represents the initial sensor offset voltage, and is the reference value for **AutoZeroWarning** and **AutoZeroLimit** operation.							
AutoZeroWarning	**0xAB**	**R W**	**Ufract16**				**EEPROM / RAM**
Represents the magnitude of **SensorCurrentZero** minus **SensorReferenceZero**, above which the DCMFC will indicate an **AutoZeroAlarm** exception.							
AutoZeroLimit	**0xAC**	**R W**	**Ufract16**				**EEPROM / RAM**
Represents the maximum permissible magnitude of **SensorCurrentZero** minus **SensorReferenceZero**.							
NormalizedFlow	**0xB9**	**R M**	**Ufract16**				**Volatile**
This is the normalized flow signal. It is a fraction equal to 0.25 + Flow / (2 * **FullScaleRange**) It is a fraction nominally in the range [0.25 .. 0.75], where 0.25 represents 0 flow, and 0.75 represents the rated flow of the process gas.							
Calibrate	**0xBA**	**R W**	**UInt8**				**Volatile**
Indicates which internal calibration operations are currently scheduled. Setting a bit in this variable initiates one of the DCMFC's internal calibration operations. The bit will be automatically reset when the operation completes. Unused bits should be written as "0". The bits are assigned as follows: bit 0: **calSensorZero** Initiates the Requested Zero operation							
AutoCalibrate	**0xBB**	**R W**	**UInt8**				**EEPROM**
Indicates which internal calibration operations are to be performed when the user presses the Calibrate switch. When that occurs, this value is loaded into **Calibrate** and begins the scheduled calibration operations.							
RecoveryTimeout	**0xC0**	**R W**	**UInt16**		**1**	**25549**	**EEPROM**
The sensor cooling recovery routines basically work by turning off the valve for a pre-set time to allow the sensors to warm up. **RecoveryTimeout** specifies the amount of time (in mS) the valve should be shut down upon detection of sensor cooling. The value specified is rounded to the nearest 100 mS, except that values from 1 to 49 mS are rounded up to 100 mS. If this value is 0, the DCMFC will behave as though it were 2000. This attribute did not exist prior to firmware version 3.15. In firmware versions 3.15 and 3.16, this attribute is a UInt8, and is specified in tenths of a							

Attribute Name	Attribute ID	Access	Format External	Format Internal	Range Min	Range Max	Storage
second.							
OverFlowTimeout	**0xC2**	**R W**	**UInt16**		**0**	**2554**	**RAM**
The time delay in mS before triggering an OverFlow recovery cycle. If the measured flow is significantly and continuously above normal range for this time, **RecoveryTimer** will be set to a value corresponding to 300 mS. The value specified is rounded to the nearest sample interval, except that non-zero values less than one sample interval are rounded up to one sample interval. If this value is 0, the DCMFC will behave as though it were 750. This attribute did not exist prior to firmware version 3.17.							

6.9.4 Behavior

6.9.4.3.1 Sensor Zero Filter

The DCMFC is able to correct for reasonably stable offsets in the gas flow sensor. It does this by subtracting a value, known as **SensorCurrentZero**, from each sensor measurement.

The DCMFC updates **SensorCurrentZero** under two conditions:

1. Requested Zero:

   When the **Calibrate** attribute's **calSensorZero** bit is set (either via request from an external application, or by closing the "Zero" switch on the DCMFC circuit board), the FlowMeter switches the ValveDriver to the Off state, and waits until **AutoZeroDelay** seconds have elapsed.

   The FlowMeter updates **SensorCurrentZero** using **RequestedZeroRate** for each succeeding sample for 30 seconds.

   The FlowMeter sets **SensorReferenceZero** equal to **SensorCurrentZero**, and saves both in non-volatile memory (updating CRCs A/R) for later use. **SensorCurrentZero** is used for further operation, and both can be retrieved by external applications as needed. It then clears the **calSensorZero** bit to indicate the operation has been completed.

   Note that the actual gas flow must have been truly zero long enough for the sensor output to stabilize (preferably 90 seconds or longer) BEFORE beginning the requested zero operation, and for the entire duration of the operation. IT IS THE USER'S

RESPONSIBILITY to ensure this condition is met. Failure to do so may lead to radical flow measurement errors.

2. Auto Zero:

The DCMFC begins counting time when all of the following apply:

a. **AutoZeroEnable** is non-zero

b. **AutoZeroDisable*** input is not active

c. The DCMFC is in Off mode

After all pre-conditions have been true for a specified delay (**AutoZeroDelay**), the DCMFC sets **AutoZero** to initiate a single auto zero operation ten times per second. The DCMFC clears **AutoZero** after each auto zero filter operation has occurred.

So long as **AutoZero** remains set, the DCMFC updates **SensorCurrentZero** using **AutoZeroRate** for each sample.

SensorCurrentZero is a 32-bit fraction, updated as needed using the equation:

SensorCurrentZero **=** SensorCurrentZero **+** (SensorScaled **-** SensorCurrentZero) ***** Zrate

where

Zrate is RequestedZeroRate (typically 2^-9) for commanded zero,

or AutoZeroRate (typically 2^-13) for auto-zero.

During auto-zero calculations, **SensorCurrentZero** is strictly limited to the range [**SensorReferenceZero** - **AutoZeroLimit** .. **SensorReferenceZero** + **AutoZeroLimit**]. If **SensorCurrentZero** reaches either limit during auto-zero calculations, the DCMFC will declare an **AutoZeroLimit** exception, typically activating the **Alarm** output. This limit does not apply to zero measurements made as a result of a requested-zero operation.

If | **SensorCurrentZero** - **SensorReferenceZero** | exceeds **AutoZeroWarning**, the DCMFC will declare an **AutoZeroAlarm**

exception, typically activating the **Alarm** output. This limit applies to both auto-zero and requested-zero operations.

The ReliabilityTimer class saves a copy of **SensorCurrentZero** in non-volatile memory (see the ReliabilityTimer class **eeSensorCurrentZero** private attribute).

6.9.4.5 Sensor Cooling Detection

The thermal mass flow detector used with the 203 board becomes very non-linear at high flow rates, with output actually decreasing with increasing flow, because of sensor cooling. When this happens it is not possible to make any rational measurement of true gas flow rate.

Typically this happens when the valve is wide open and an adequate pressure differential exists across the MFC. This can happen when the unit is put into purge mode, or when the unit is in flow mode and there is initially not enough pressure differential to provide the requested amount of gas, and then pressure increases suddenly. This can lead to flow rates orders of magnitude beyond the measurement capability of the sensor. The normal control loop will not recover from this situation unaided.

Code was added in v3.15 to detect this situation, and automatically initiate a recovery cycle at the appropriate time.

Prior to firmware v3.17, **RecoveryTimeout** was a UInt8 in tenths of a second, rather than a UInt16 in mS.

6.9.4.6 Flow Over-Range Detection

The DCMFC monitors the indicated flow for out-of-range conditions which can lead to poor recovery from sudden pressure increases, etc. It does this by monitoring both **LinearizedFlow** and **NormalizedFlow**. If either signal continuously exceeds a value ~ 145% of rated flow for the time specified by **OverFlowTimeout**, it triggers a recovery cycle of 200 mS minimum.

This capability did not exist prior to firmware v3.17.

6.10 FlowController Class (Class ID = 0x69)

The FlowController class is responsible for generating a single **CurrentSetpoint** value acceptable to the ValveDriver class. **CurrentSetpoint** is dependent on **RawSetpoint** from the A/D converter, and on several instance attributes which can be set digitally by the process controller.

The DCMFC maintains 1 instance of the class.

6.10.1 Class Attributes

The FlowController class provides no class attributes.

6.10.2 Instance Attributes

Attribute Name	Attribute ID	Access	Format External / Internal		Range Min	Range Max	Storage
ControlMode	**3**	**R W**	**UInt8**	**UInt2**	**1**	**2**	**RAM**
Determines whether the DCMFC is to accept setpoints from the A/D converter, or from the network. A value of 1 (**CM_Digital**) indicates **NewSetpoint** values received over the network are to be used. A value of 2 (**CM_Analog**), indicates **CalibratedSetpoint** values from the Shared A/D converter are to be used.							
DefaultControlMode	**4**	**R W**	**UInt8**	**UInt2**	**1**	**2**	**EEPROM**
Value to be used for **ControlMode** after any reset.							
FreezeFollow	**5**	**R W**	**UInt8**	**UInt1**			**RAM**
This variable permits the master to send setpoints to several DCMFCs, then have all controllers change to their new setpoints at the same time. A value of 0 (**FF_Freeze**) indicates **CurrentSetpoint** is to be frozen, and any new setpoints from the network will only affect **NewSetpoint**. A value of 1 (**FF_Follow**) indicates **CurrentSetpoint** is to track **NewSetpoint**. Any new setpoints from the network are to immediately update **CurrentSetpoint**. Unless **ControlMode** = **CM_Digital**, the controller ignores **FreezeFollow**.							
CalibratedSetpoint	**0xA3**	**R M**	**UFract16**				**Temp**
Setpoint from the A/D converter, adjusted for circuitry gain & offset errors. It is a fraction, nominally in the range [0.25 .. 0.85], where 0.25 represents 0 flow, and 0.75 represents 100% of full scale.							
NewSetpoint	**0xA4**	**R W**	**UFract16**				**RAM**
Represents a setpoint received from the network, to be used as described under “Digital Setpoint Algorithm” below, and under the **ValveOverride** instance attribute of the **ValveDriver** class below. It is an offset binary fraction nominally in the range [0.25 ..							

Attribute Name	Attribute ID	Access	Format External Internal		Range Min	Range Max	Storage
0.85], where a value of 0.25 represents 0 flow, and 0.75 represents 100% of rated flow.							
CurrentSetpoint	**0xA5**	**R**	**UFract16**				**RAM**
Represents the current setpoint, either from the A/D converter or from the network, before application of any soft-start or soft-stop ramps. It is an offset binary fraction nominally in the range [0.25 .. 0.85], where a value of 0.25 represents 0 flow, and 0.75 represents 100% of rated flow.							
TurnOnDelay	**0xA6**	**R W**	**UInt16**		**1**	**25549**	**EEPROM**
TurnOnDelay specifies the amount of time (in mS) the valve should wait before switching from Off to Flow. The value specified is rounded to the nearest 100 mS, except that values from 1 to 49 mS are rounded up to 100 mS. This permits graceful turn-on in tools which do not wait for supply gas pressure to stabilize before attempting to flow gas. **TurnOnDelay** should be long enough for the supply gas to reach perhaps 90% of final pressure. This attribute did not exist prior to firmware version 3.17, or in firmware version 3.20.							

6.10.3 Behavior

The DCMFC algorithms all use a setpoint represented by an unsigned fraction where 0.25 represents 0% flow (0.000 V at the analog Setpoint input), and 0.75 represents 100% of rated flow (5.000 V at the analog Setpoint input).

6.10.3.2 Digital Setpoint Algorithm

The MFC setpoint can come from either the analog Setpoint input, or from a request over the serial port. When **ControlMode** is 1 (**CM_Digital**), the DCMFC ignores the analog setpoint input in favor of the digital input. **ControlMode** defaults to **DefaultControlMode** after a reset. It can be changed to **CM_Digital** by a specific request over the network. It will also be changed automatically to **CM_Digital** whenever the DCMFC receives a **NewSetpoint** value via the network.

The DCMFC maintains 2 setpoint values in RAM. One is **NewSetpoint**, which can be set and read over the network. The other is **CurrentSetpoint**, which represents the value in use at any given moment.

When **ControlMode** is set to **CM_Analog**, **CurrentSetpoint** follows **CalibratedSetpoint** calculated from the A/D converter value.

When **ControlMode** is set to **CM_Digital** and **FreezeFollow** is set to **FF_Freeze**, **CurrentSetpoint** does not change.

When **ControlMode** is set to **CM_Digital** and **FreezeFollow** is set to **FF_Follow**, **CurrentSetpoint** follows **NewSetpoint**.

Under some conditions, the ValveDriver class uses **NewSetpoint** independently from **CurrentSetpoint**.

6.10.3.3 Turn-On Delay Algorithm

TurnOnDelay specifies the amount of time (in mS) the valve should wait before switching from Off to Flow. The value specified is rounded to the nearest 100 mS, except that values from 1 to 49 mS are rounded up to 100 mS.

This permits graceful turn-on in tools which do not wait for supply gas pressure to stabilize before attempting to flow gas. **TurnOnDelay** should be long enough for the supply gas to reach perhaps 90% of final pressure.

Whenever the unit is in Off mode, except when due solely to a recovery cycle in progress, the DCMFC triggers a recovery cycle at least equal to **TurnOnDelay** minus 100 mS.

This attribute did not exist prior to firmware version 3.17, or in firmware version 3.20.

6.11 ValveDriver Class (class ID = 0x6A)

The ValveDriver class is responsible for all hardware involved in controlling the valve driver.

The DCMFC maintains 1 instance of the class.

6.11.1 Class Attributes

The ValveDriver class provides no class attributes.

6.11.2 Instance Attributes

Attribute Name	Attribute ID	Access	Format External	Format Internal	Storage
ValveOverride	**1**	**R W**	**UInt8**	**UInt4**	**RAM**
This is the valve operating mode requested via the network. It has the following values: **OR_Flow** = 0 => Flow control mode **OR_Off** = 1 => Valve should be closed **OR_Purge** = 2 => Valve should be completely open **OR_PwrOff** = 3 => Set to power off state **OR_PwrOff** is a transient condition -- it is immediately translated to **OR_Off** in a normally closed valve, or to **OR_Purge** in a normally open valve.					
ValveType	**0xA0**	**R I**	**UInt8**	**UInt2**	**EEPROM / RAM**
ValveType indicates the type of valve used in this MFC. A value of 1 (**VT_Closed**) indicates a Normally-Closed valve is attached. A value of 2 (**VT_Open**) indicates a Normally-Open valve is attached. A value of 0 (**VT_None**) indicates no valve is attached.					
OverrideMode	**0xA1**	**R W**	**UInt8**	**UInt1**	**EEPROM**
OverrideMode determines how the Override (J) pin is to be interpreted. A value of 1 (**OR_Semi**) means the Override is to be interpreted per SEMI 2253 specs. A value of 0 (**OR_Unit**) means the Override pin is to be interpreted per traditional Unit Instruments definition.					
AutoShutOffThreshold	**0xA2**	**R W**	**UFract16**		**EEPROM / RAM**
This value, in the range [0.25 .. 0.5), represents the lowest setpoint which the DCMFC will treat as valid. Any setpoint below this value will be treated as though the J-pin had been grounded. Threshold is in the same units as the DCMFC's internal **CalibratedSetpoint** signal. Thus, a Threshold of 0.2575 corresponds to an external setpoint of 1.5%.					

Attribute Name	Attribute ID	Access	Format External	Format Internal	Storage
SoftStartRate	**0xA4**	**R W**	**UFract16**		**EEPROM / RAM**
Represents the maximum increase permitted in **FilteredSetpoint** per sample interval. It is equal to half the desired increase in % of full scale -- i. e., a value of 0.01 represents a change in setpoint of 2% per sample interva. A value greater than 0.5 effectively disables this feature. It is calculated as: 1 / (2 * **SampleRate** * desired ramp time in seconds)					
FilteredSetpoint	**0xA6**	**R M**	**UFract16**		**Volatile**
This is the **CurrentSetpoint**, after any **SoftStartRate**, **SoftPurgeRate**, or other limits have been applied. It is a fraction nominally in the range [0.25 .. 0.75], where 0.25 represents 0 flow, and 0.75 represents rated flow.					
LssEnable	**0xA7**	**R W**	**UInt8**	**UInt1**	**EEPROM / RAM**
If non-zero, enables the Low Setpoint Speed-Up feature.					
mfcFilterGain	**0xA8**	**R W**	**UFract16**		**EEPROM / RAM**
Filter constant for normalized flow to be used for display purposes only - typically 0.200. This value is the reciprocal of the time constant (in sample intervals) applied to mfcOut. It should be calculated as: **mfcFilterGain** = 1 / (Tc * **SampleRate**) where Tc is the desired time constant.					
mfcOut	**0xA9**	**R**	**UFract16**		**RAM**
Filtered signal of normalized flow.					
ValveVoltage	**0xB6**	**R M**	**UFract16**		**Temp**
This value is the valve drive current, as measured by the Shared A/D. It is a fraction, nominally in the range [0 .. 0.5).					

6.11.3 ValveDriver Private Variables

Variable Name	Storage	Format
ValveMode	**RAM**	**UInt**
Represents the mode in which the valve is currently operating. It can take on 5 values: **VM_Flow, VM_Off, VM_Purge, VM_DaCalibrate,** and **VM_SensorTune**		

6.11.4 Behavior

6.11.4.1 Auto Shut-Off Algorithm

Auto Shut-Off prevents the DCMFC from continuously attempting to pass a small amount of gas because of minor uncorrected ground differences between the process controller and the DCMFC, and from attempting to control gas flow outside its designed control range. It does this by comparing **CurrentSetpoint** with a programmable **AutoShutOffThreshold** (typically about 1.5% of rated flow). If the **CurrentSetpoint** is below the threshold, the DCMFC assumes the process controller wants the valve completely closed.

6.11.4.3 Override Pin Algorithm

The Override pin (or J-pin) can be used in 2 modes -- Unit Instruments and SEMI -- selected by the **OverrideMode** attribute.

In Unit Instruments mode, when the Override pin is left open or pulled high the valve should control flow normally. When the Override pin is pulled low, the valve should be fully closed.

In SEMI mode, when the Override pin is left open or pulled to ground, the valve should control flow normally. When the Override pin is pulled to +15V, the valve should be fully open. When the Override pin is pulled to -15V, the valve should be fully closed.

Override Pin Voltage	OrMode = OR_Unit	OrMode = OR_SEMI
-16.5 .. -4.85	Closed	Closed
-4.85 .. +1.67	Closed	Control Flow
+1.67 .. +5	Control Flow	Control Flow
+5 .. +16.5	Control Flow	Purge

6.11.4.4 Valve Mode Algorithm

The valve driver module operates in several modes:

ValveMode	Description
VM_Flow	Setpoint and Flow D/As directly represent desired setpoint and measured flow values.
VM_Off	Valve is fully closed. Flow D/A represents measured flow, limited to [-3 .. 120%] of full-scale. Setpoint D/A represents

	valve current a little before the valve will begin to open, calculated by the Low Setpoint Speed-Up algorithm.
VM_Purge	Valve is fully open. Flow D/A represents measured flow, limited to [-3 .. 120%] of full-scale. Setpoint D/A is at a value which drives the valve fully open, dependent on both measured flow and the **ValveOverdrive** attribute.

The mode selected depends on several factors, applied in the following order:

Factor	**ValveMode**
ValveType = **VT_None**	**VM_Flow**
Exceptions::**Alarm** = 1 and **ValveType** = **VT_Closed**	**VM_Off**
Exceptions::**Alarm** = 1 and **ValveType** = **VT_Open**	**VM_Purge**
FlowMeter::**Calibrate.calSensorZero** = 1	**VM_Off**
Override pin => "Closed"	**VM_Off**
Override pin => "Open"	**VM_Purge**
ValveOverride = **OR_Off**	**VM_Off**
ValveOverride = **OR_Purge**	**VM_Purge**
CurrentSetpoint < **AutoShutOffThreshold**	**VM_Off**
CurrentSetpoint >= 122%	**VM_Purge**
Recovery cycle in progress	**VM_Off**
otherwise	**VM_Flow**

8. Communication Protocol

The DMFC acts as a slave device on an RS-485 multidrop bus. It continually listens for requests from the master device, processes requests addressed to it, and sends replies as needed.

The protocol described here is designed to provide a programming interface as similar as practical to that which will be required by the SEMI E54.4-0997 Sensor/Actuator Communication Standard. It is intended as an interim step on the way to SEMI E54.4-0997, and is NOT intended to coexist with that protocol or electrical interface on the same bus.

Data on the bus is carried as a series of 8-bit bytes, each transmitted in standard asynchronous serial format with a leading start bit, 8 data bits (LSB first), and a single trailing stop bit (no parity) at 9600 baud.

Each device on the bus has a unique address. By definition, the bus master is address 0. DCMFCs have addresses in the range 0x20 (32) through 0xEF (239)., though most systems should restrict the assigned addresses to the range 0x20 through 0x5F. Address 0x00 is reserved for the bus master. Address 0xFF (255) is the broadcast address for requests addressed to all slaves on the bus. Addresses 0x01 through 0x1F are reserved for bus control characters, such as ACK (0x06) and ENQ (0x05). Addresses 0xF0 through 0xFE are reserved for future use.

The broadcast address cannot be used if there is more than one slave device on the bus. It is only necessary when changing the address of a device, or when querying a device with an unknown address.

Messages on the bus are sent as packets with a fixed format. Each packet begins with the target device address, an STX character (0x02), a service code (0x80 to 0x82), and a data byte count between 2 and 33 (inclusive). Each packet ends with a pad byte of 0, and a 1-byte checksum which is the sum of all of the bytes in the communication packet, other than the device address byte, modulo 256. The checksum calculation discards the carry from the byte summation calculation.

Note: A future version of the DCMFC firmware is likely to permit much longer packets. Anyone writing drivers or application software for the DCMFC, particularly anyone using the Monitor service, should allow buffer space for data byte counts of up to 255.

Devices on the bus distinguish addresses from other characters by maintaining an idle timer. This timer is started at the end of each received character, and expires if another character does not arrive within two character times (20 bit times). If the timer expires, the device assumes the message has ended, and the next character received will be either a target address, or a response indicating acceptance of the previous message. Thus, it is vital that devices on the bus not insert idle gaps of 1

character time or more within a packet, as the following character could then be misinterpreted as a target address. A target device can also assume an error has occurred if a new character is expected and does not arrive within 2 character times of the preceding character's arrival.

Each transaction on the bus begins when the master transmits a request packet on the bus, following an idle gap of at least 1 character time. Slave devices ignore the packet unless all of the following are true:

A. The first byte of the packet either matches the slave's address, or is 0xFF.

B. The second byte of the packet is STX (0x02).

C. The third byte of the packet is a valid service code.

D. The fourth byte of the packet, plus 6, is equal to the total number of bytes received.

E. The Pad byte is 0.

F. The Checksum byte matches the checksum calculated for the received packet.

The specified slave device replies quickly with an ACK character (0x06) to indicate it has received the packet correctly. This response will begin no less than 2 nor more than 4 character times after the end of the packet.

If the master receives no response within 5 character times after the end of the message, the master should assume that the slave will never respond, and either re-send the packet, or abort the operation, as appropriate.

If the master receives any initial response other than an ACK character, it should wait for either some further response (which should be discarded) followed by a gap of at least 2 character times, or no further response for at least 100 mS. In either case, it should then either re-send the packet, or abort the operation, as appropriate.

If the master receives an initial response of ACK, it should wait for up to 100 mS for the remainder of the response to begin. If it receives no further response, or the additional response appears invalid (see below), it should either re-send the packet, or abort the operation, as appropriate..

After the message has been processed, the slave sends an additional ACK or NAK (0x16) character, or one or more reply packets, as required for the specific service request.

Reply packets are structured identically to the request packets, except that the target address is 0 (since they are destined for the bus master). The service code is the same as the service code in the original request.

The bus master responds to each reply packet with either an ACK if the service was not Monitor or if it wishes no more monitor data, or an ENQ (0x05) character if it wishes further Monitor data.

If the slave does not receive any response within a reasonable time (currently 18 to 20 character times) after sending a message, it behaves as though it received an ACK.

If a slave receives an invalid response, it behaves as though it received an ACK, and assumes the invalid character is the target address for a new message.

The master waits for at least 2 character times following the final response packet, then begins the next transaction on the bus.

Binary values are always transmitted LSB first. Text values are transmitted left-most character first.

8.1 Service requests

The DCMFC understands 3 service requests, received over the RS-485 bus:

- Read
- Write
- Monitor

Each service is described separately below.

Each service interaction begins with a request packet containing the Service Code in byte 3.

8.1.1 Read

The Read request is a standard 9-byte packet containing:

byte 3 = Service Code (0x80)

byte 5 = Class ID

byte 6 = Instance ID

byte 7 = Attribute ID

DCMFC responds immediately with an ACK character indicating it has correctly received the message, and will process it as appropriate.

When the requested value becomes available (after reading from EEPROM, etc.) the DCMFC sends a standard N-byte reply packet containing:

byte 3 = Service Code (0x80)

byte 5 = Class ID

byte 6 = Instance ID

byte 7 = Attribute ID

bytes 8 .. N-2 = variable value

The master responds with an ACK to indicate it has correctly received the packet.

If the Class, Instance, or Attribute ID is invalid, the DCMFC responds with a NAK character rather than the normal response packet.

8.1.2 Write

The Write request is a standard N-byte packet containing:

byte 3 = Service Code (0x81)

byte 5 = Class ID

byte 6 = Instance ID

byte 7 = Attribute ID

bytes 8 .. N-2 = variable value

The DCMFC responds with an ACK character indicating it has correctly received the message, and will process it as appropriate.

After the DCMFC has finished processing the message, it sends either an ACK or NAK character to indicate success or failure. A NAK response may indicate an invalid Class, Instance, or Attribute ID, an invalid value, or an error while attempting to save a value in EEPROM.

8.1.3 Monitor

The Monitor request is a standard 9-byte packet containing:

byte 3 = Service Code (0x82)

byte 5 = Class ID

byte 6 = Instance ID

byte 7 = Attribute ID

The DMFC responds immediately with an ACK character indicating it has correctly received the message, and will process it as appropriate.

If the Class, Instance or Attribute ID is invalid, the DMFC responds with a NAK character rather than the normal response packet.

When the requested value becomes available, the DMFC sends a standard N-byte reply packet containing:

byte 3 = Service Code (0x82)

byte 5 = Class ID

byte 6 = Instance ID

byte 7 = Attribute ID

bytes 8 .. N-2 = variable values

Each response message contains one to 15 samples of the requested variable, as the DCMFC sees fit.

The master responds with an ENQ to indicate it has correctly received the packet and wishes additional samples. The master responds with an ACK if it wishes no additional samples.