3D Series MicroDisplay Module

Programmers Manual

Grayhill PN: 3DUM1271-1

Revisions

A	Original Release	03/08/2010
B	Added the ability to control the individual colors of the LCD backlight	05/17/2010
C	Added appendix that includes the pinout for the main connector. Added Width parameter to the line, circle and box. Indicates which Control commands need to come from Source Address 249	08/11/2010
D	Added new features for firmware II	10/07/2011
E	Removed bad reference.	3/29/2012
F	Corrected bit length range lower limit from 1 to 2	8/02/2012
G	Sec 7.6, Sub commands 4 and 5 were swapped	10/04/2012
H	Sec 8.9, Corrected error in the Format example	11/26/2012

Table of Contents

1. Overview

This document describes the functionality and communication of the Grayhill MicroDisplay unit

1.1. Reference Documents

The following documents are referenced within this document.

- SAE-J1939
- SAE-J1939/11
- SAE-J1939/21
- SAE-J1939/71
- SAE-J1939/81

2. Functionality

The MicroDisplay is a small display unit designed to display vehicle information and simple graphics. Display screens are comprised of objects (gauges, text, graphics, etc.) that are created and downloaded with the configuration tool. Objects are instantiated either automatically at boot, with a special CAN message, or using a menu, which itself is an object. The four buttons can be used to control a menu system and/or can be sent in a message for processing by the main ECU device. The MicroDisplay is equipped with two outputs capable of sourcing 200mA and three inputs that are read and sent to the ECU via a standard message. The I/O also have objects that, for inputs can instantiate other objects and for outputs, control the outputs using objects.

The MicroDisplay can display so called Monitoring Objects or Non-monitoring Objects. Non-monitoring object, such as simple icons or text, are displayed once and remains on the display until it is overwritten or erased either explicitly or via a Clear Screen command. Monitoring objects, such as a Dynamic Text and Custom Gauges, are loaded into internal memory and bound to a specific bit field, PGN, and source address during configuration. The MicroDisplay then listens for the specific message and updates the monitoring objects accordingly without further assistance from the ECU. Monitoring objects must be removed with a Kill command. This will remove the object from internal memory and also erase the graphic object from the display.

The objects can also be either non volatile objects or volatile objects. Non volatile objects are stored locally in flash. One or more non volatile objects can be displayed by simply referencing an objects ID number with one J1939 message. Volatile objects are objects that are displayed at run time and are sent to the display with multiple J1939 messages where parameters are read directly from the corresponding J1939 messages giving the MicroDisplay the flexibility of a simple terminal device. Only non-monitoring objects (boxes, text, lines, etc.) are volatile and can be created on the fly at run time.

2.1. Power Up Sequence

Upon first power up, a splash screen, if used, is displayed for the programmed amount of time. The device will then display the last object instantiated via the menu if the feature is enabled. If not enabled or if the object cannot be found the Default Object is displayed. If the default object is unused or can't be found the display is simply left blank.

The MicroDisplay module sends out an Address Claimed message. If there is a Name contention and the MicroDisplay module loses arbitration, it will either send another Address Claimed message with a new source address if the MicroDisplay module is using

Dynamic Addressing otherwise it will send out the Cannot Claim Address message. If the MicroDisplay module is using Dynamic Addressing and cannot find an unused source address it will then send the Cannot Claim Address message. If a MicroDisplay module sends out the Cannot Claim Address message it will not enter Run Mode (it will not transmit or act upon any messages). Refer to SAE-J1939/81.

2.2. Run Mode

2.2.1. Key Data and Auxiliary Input - Transmit

The key and the auxiliary input information are sent on separate messages every 100ms or upon a change in status with a minimum period of 20ms. The MicroDisplay's PGN, priority and transmission period are configurable at runtime.

The transmission of either the key data or the auxiliary input messages has the ability to be disabled.

2.2.2. Monitored Messages - Receive

The MicroDisplay module constantly monitors the J1939 messages assigned to monitoring objects and updates the display accordingly. It also listens for the Configuration and Control messages and the Auxiliary Output message. The communication watchdog timeout flag can be configured to display a simple message "NO DATA" if the MicroDisplay does not detect any CAN message. It will revert back to the last group object displayed. Volatile objects displayed in real time will not be restored.

3. Non Volatile Objects

Non volatile graphic objects are objects that are created with an external PC application and transferred to the MicroDisplay's flash using J1939 messages. These Objects are assigned a unique ID and are displayed when commanded by using the Display Object command described in Section 8.7. An object can be removed from the display by either sending a Kill Object or a Clear Screen command.

All objects begin with three 16 bit fields that give information about the object. These include the size, unique ID, and the type. The remaining fields hold information pertaining to the specific type of object, such as color, length, font, placement, etc.

3.1. Static Non Volatile Objects

Static non volatile objects are simple objects such as bitmaps, text, lines, etc. that are painted on the display from flash using the Display Object command. Once displayed no additional resources are required to maintain the object. These objects can overlap each other. The most recent object will be displayed as the full object. However, if an underlying object is removed with the Kill Object command it will remove the overlapped portion of the most recent object. In this event the most recent object needs to be refreshed by another Display Object command.

3.1.1. Circle Object

The circle object is stored with the specified ID, location, radius, grayscale color, and fill.

3.1.2. Box Object

The box object is stored with the specified ID, location, size, grayscale color, and fill.

3.1.3. Line Object

The line object is stored with the specified ID, location, length and grayscale color.

3.1.4. Bitmap Object

The bimap object is stored with the specified ID, location, and data.

3.1.5. Static Text Object

The text object is stored with the specified ID, location, color, font and text string.

3.1.6. Group Object

The group object is stored with the specified ID, member object IDs, and a flag to indicate if the group is the default screen.

3.1.7. Bootup Object

The Bootup Objects are objects that are not invoked over CAN but display automatically. Two different Bootup Objects exists.

3.1.7.1. Splash Screen

The Splash Screen object, if used, displays the object just after boot for the duration of time specified in Param1 of the structure in units of 1ms. After this time expires the object is killed.

3.1.7.2. Default Screen

The Default Screen, if used, is displayed after the Splash Screen has expired or immediately after boot if the Splash Screen is not used.

3.1.8. Backlight Object (FW-II)

This object controls the backlighting of the LCD and/or the keypad when called. The intent is that the Backlight object is included in a Group Object to set the screen color while the Group object is displayed. When the Backlight object is removed the backlighting will revert back to the previous settings. Ex. A group object may contain a “Warning” static text or bitmap along with a Backlight Object that turns the LCD backlights to red. When the group object is removed the Backlight Object is also removed that would revert the LCD backlight to the original setting. If a value of larger than the max of 255 is used for any one of the three colors for the LCD, or for the keypad, it will be treated as a ‘don’t care’ condition. This allows controlling of the backlight for the LCD/keypad without affecting the setting of the keypad/LCD respectively.

3.1.9. GPIO Output Object (FW-II)

This GPIO output object controls the two outputs according to the value in the variable bound to each. The values can be zero or one to drive the output high or low respectively. Any other value will be treated as a ‘don’t care’. This allows more than one GPIO Out object to be called and would prevent inadvertent control of the output not of interest. A flag exists to revert the selected output(s) to zero if the object is killed.

3.1.10. J1939 Transmission Object (FW-II)

This object is used to transmit a full CAN message when instantiated. The number of times the message is transmitted is determined by the Send Count parameter. The Send Period parameter defines the time between consecutive message transfers in milliseconds. The user can specify the message's full CAN ID, the data length from zero to eight, and the content of the data. A flag can be set to instruct the unit to monitor for a corresponding acknowledgement message. The ack message is specified by a PGN, source address, eight data and mask bytes. the CAN Object is successfully acknowledged when a message is received that matches the PGN, source address and non masked off bits within the data bytes. When this happens the CAN Object ceases to transmit the remaining count of the transmit message. The TX OK Object will be displayed upon successful transmission of the message and, if enabled, successful reception of an acknowledgement message. The Fault Object is only displayed if the ack flag is enabled and a proper ack message is not received. The Null Object can be assigned to either or both if no action is desired.

3.1.11. Overloaded Bitmap Object (FW-II)

This object is used to display an existing bitmap object stored in the AppData section of flash with new coordinates and, for monochrome bitmaps, a new color.

3.1.12. Screen List Object (FW-II)

This object allows the use of the left and right buttons on the display to scroll through a list of object defined. Each object can be a group object that populates the entire screen with other objects.

3.2. PGN-Driven Non Volatile Objects

Dynamic non volatile objects are objects that are configured to respond to a specific bit field within a message from a specified PGN and source address. The dynamic object is effectively bound to the SPN associated with the PGN, source address and bit field without the mention of the SPN. The object parses the J1939 message and updates the output according to the bit field value. In the event of overlapping objects the dynamic objects will always display as the top object when the associated message is received causing an object update.

3.2.1. Bar Graph Object (Depricated)

The bar graph object is stored with the specified ID, location, color and dimensions along with information pertaining to the J1939 parameter the bar graph will be bound to. These include the source address, PGN, bit start and bit length of the J1939 SPN. This will extract the raw bit data from the message. In order to use this raw data, other information is stored, which include the resolution per bit, offset and the min and max values associated with 0% and 100% of the bar graph value respectively

This object does not allow for multiplexed PGN's. If this is desired use Bar Graph 2 object.

3.2.2. Bar Graph 2 Object (FW-II)

This is the same as the Bar Graph object with the inclusion of three more parameters that parse the data field of the CAN messages containing the data to be displayed on the bar graph. This allows for multiplexed PGN's were a data value depends on control bits in the same message. The bar graph is only updated when the value specified by the control bit

field is equal to the compare value. If both the bit start and length of the control bit field is zero then the Bar Graph 2 object behaves exactly like the original Bar Graph object.

3.2.3. Dynamic Text Object (Depricated)

This object is stored with the specified ID, location and color along with information pertaining to the J1939 parameter the dynamic text will be bound to. These include the source address, PGN, bit start and bit length of the J1939 SPN. This will extract the raw bit data from the message. In order to use this raw data, other information is stored, which includes the resolution per bit, offset and prefix and suffix text that may serve to display the units of the SPN.

This object does not allow for multiplexed PGN's. If this is desired use the Dynamic Text 2 object.

3.2.4. Dynamic Text 2 Object (FW-II)

This is the same as the Dynamic Text object with the inclusion of three more parameters that parse the data field of the CAN messages containing the data to be displayed on the bar graph. This allows for multiplexed PGN's were a data value depends on control bits in the same message. The text is only updated when the value specified by the control bit field is equal to the compare value. If both the bit start and length of the control bit field is zero then the Dynamic Text 2 object behaves exactly like the original Dynamic Text object.

3.2.5. SPN Value Object (FW-II)

The Value object is effectively a table that associates values of an SPN to objects. The user specifies the PGN, Source Address and bitfield the map object will be bound to. A table is also specified that binds unique bitfield values to objects. Each unique value is assigned one object. The object can be a group object in order to bind multiple objects to the unique value. When the bitfield value changes the previous value's corresponding object is killed and the new bitfield value object, if it exists, will then be activated. The null object can be associated with a bitfield value if no action is desired. This is capable of handling multiplexed PGN's

3.2.6. SPN Range Object (FW-II)

This Range object displays one of three objects based on a range defined by a min and a max value. If the value is less than the min, object 1 is activated, if the value is greater than or equal to the min and is less than or equal to the max, object 2 is activated. If the value is greater than the max, object 3 is activated. The null object can be associated with a range if no action is desired. This is capable of handling multiplexed PGN's

3.2.7. Custom Gauge Object (FW-II)

This object allows for the creation of custom gauges where the indicator segments are made up of individual objects. A custom gauge can be made up of up to 50 segments. Each segment has two objects bound to it along with a check value used to compare to the new incoming data. The mode of the gauge can be one of two ways: standard, or pointer. In standard mode, all segments who's check value is lower than the incoming data value will display object one. All others will display object two. In pointer mode, only one segment shows its object 1 at a time. This is the segment where the data value is between the check value and the check value of the next segment in the index. All others will display object 2. When the gauge is updated with a new value the array is traversed comparing the stored values then display/kills the bound objects for each segment

accordingly. If it is desired that no object be shown, most likely this may be the case for object 2, then assign the object the Null Object (zero). This is capable of handling multiplexed PGN's

3.2.8. PGN Watchdog Object (FW-II)

This object is used to indicate if a specified CAN message has been received by the unit according to the specified PGN and source address. A timeout period is specified in milliseconds. When the specified message is received the timer is reset. During this time the normal object is displayed. If the specified message is not received within the timeout period the normal object is killed and the timeout object is displayed. If the message is received after this point the timeout object is killed and the normal object is redisplayed. Killing the PGN Watchdog Object will also kill the associated object called out as parameters.

3.3. GPIO-Driven Input Object (FW-II)

The GPIO Input object simply displays one of two different objects based on the state of a specified input. All three inputs can be read with one instantiation of this object or up to three different objects can be instantiated where the Null object would be assigned to the inputs that are not of interest. Ex. The first instance may monitor the first output where the Null would be assigned for the objects bound to inputs two and three. A second instance may monitor input two where the Null object is assigned to the object bound to one and three. It is possible to have more than one GPIO In objects monitor the same input, but a better way would be to have only one and use a Group object to display every thing needed.

3.4. Timer-Driven Object (FW-II)

The timer object toggles between two objects according to the timing parameters. The two timing parameters, period 1 and period 2, are bound to object 1 and object 2 respectively. Each object is displayed for the length of time in milliseconds defined by the period. The null object can be assigned to either one if no action is desired for that time slice. If more than two states are desired, a Timer Object can call other Timer Objects.

Example: (Brian insert work here).

3.5. Keypad-Driven Menu Object (FW-II)

The Menu object is used to display a menu that can be navigated by the keypad. Two different implementations exist.

1. Pressing the left most button will launch a Root Menu. The left and right arrow buttons navigate through the different menu items. the enter button selects the current menu item. This can either launch a submenu or an object, most likely a group object that will populate the entire screen.
2. Each button is bound to an object. Pressing the button will display the corresponding object, most likely a Group object that populates the screen.

Only one Root Menu and/or Button Bound menu should exist in the AppData at a time.

4. Volatile Graphic Objects

Volatile graphic objects are objects that can be displayed in real time by sending the object type along with its associated parameters with multiple J1939 messages. Refer to Section 8. Volatile graphic objects give the MicroDisplay the flexibility of a simple terminal device that can display text and simple graphics at run time requiring no knowledge of the non volatile objects stored in flash. It is possible to use the MicroDisplay only in this fashion without the need of application objects and the PC configuration tool.

the available volatile graphic objects follow:

4.1.1. Circle Object

The circle object is drawn with the specified ID, location, radius, grayscale color, and fill.

4.1.2. Box Object

The box object is drawn with the specified ID, location, size, grayscale color, and fill.

4.1.3. Line Object

The line object is drawn with the specified ID, location, length and grayscale color.

4.1.4. Static Text Object

The text object is stored with the specified ID, location, color, font and text string.

5. Embedded Objects.

Embedded objects are objects that are located within the application code itself and not in the application data section. These objects are tools for the configuration of the MicroDisplay embedded parameters such as contrast adjustment and backlighting. The object IDs start at 32,000.

5.1. Backlight Control Object 32,000 (7D00h)

This object displays a menu like screen for controlling the intensity of the three different colors of for the LCD backlighting and the intensity of the keypad backlighting. When 'Save' is selected the values are stored in non-volatile memory and will remain until it is changed either using this control or through the Backlight Intensity command under Configuration and Control

5.2. Contrast Control Object 32,001 (7D01h)

This object displays a menu like screen for controlling the contrast of the display. When 'Save' is selected the value is stored in non-volatile memory.

6. Communications

6.1. Message Header Description

The following illustrates the format of the CAN message ID. A brief description of each field follows.

SOF	Identifier 11 Bits											SRR	IDE	Identifier Extension 18 Bits																		RTR
SOF	Priority			R	DP	PDU Format (PF) 6 Bits (MSB)						SRR	IDE	PF (cont.)		PDU Specific (PS) (Destination Address, Group Extension or Proprietary)								Source Address								RTR
	3	2	1			8	7	6	5	4	3			2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	23	25	26	27	28	29	30	31	32	33
	28	27	26	25	24	23	22	21	20	19	18			17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Figure 2

6.1.1. Priority

This 3-bit field is used to define the priority during arbitration. '000' is the highest priority and is usually associated with high-speed control messages. Low priority is used for non-critical configuration and information messages.

6.1.2. Data Page

This 1-bit field defines on which data page (0 or 1) the message is defined in the J1939 specification. Page 0 contains the messages that are presently defined, while Page 1 is for future expansion according to J1939.

6.1.3. Protocol Data Unit (PDU), PDU Format (PF), PDU Specific (PS)

This 8-bit field determines the format of the message and is one of the fields that determine the Parameter Group Number of the message (see Figure 2). If the value is between 0 and 239, the message is a PDU 1 Format message. These messages are sent to specific addresses. The PDU Specific (PS) field is the Destination Address (DA). If the value is between 240 and 255, the message is a PDU 2 Format message. These messages are not sent to a specific address, but are instead broadcast to the entire network. The PS then becomes the Group Extension (GE) field.

6.1.4. Source Address

This 8-bit field is the source address of the device that sent the message.

6.1.5. Data Length

The number of data bytes in the message.

6.1.6. Data

Up to 8 bytes of data. The first five items are placed into the CAN 29-bit extended identifier in the format shown in figure 2. Most messages are intended to be broadcast messages, or PDU 2 Format, where the message is not sent to a particular address. The J1939

specification defines PDU Format and PDU Specific values for many messages by specifying the message Parameter Group Numbers.

6.1.7. Parameter Group Number

J1939 defines allowable messages by their Parameter Group Number (PGN). The Parameter Group Number is a 3-byte value that uniquely defines the message purpose. A PGN has the following format: If the PDU Format value for a message is less than 240, then the last 8 bits of the PGN are set to '0'. The specification gives the decimal equivalent of the PGNs. To obtain the PF and PS values to use for a specific message, convert the decimal value from the specification to hexadecimal and use the last two bytes. These values can then be used to either send messages on the network or to request messages from other source addresses.

6.2. Bitfield Location and Byte Ordering

The byte and bit ordering and location within the data field are the same as to what's called out in J1939. The first data byte is sent first and is referenced as Byte 1, 0x01 in the example below. The LSB of the data bytes are on the right and are referenced as Bit 1.

The convention used to locate a parameter in the data field is the same as specified in SAE-J1939/71. The format used is "R.x" where R is the byte number and x is the starting bit number within the byte. The length is the number of bits starting at this point.

In the event of a byte or a multi-byte length parameter the format becomes "R-S" where R is the start byte location and the data occupies bytes R through S.

Examples:

Start	Length	Description
4.3	3 bits	Ex. Parameter between 0..6, where 7 is 'don't care"

Location 4.3 with a length of 3 bits would have the value of 1 as illustrated below using the above data frame example.

Byte 4 = 0x67 = 0b011**001**11. The bold value is the three bit field holding a value of 0b001

A 16 bit word starting at byte location 2 would be represented by the following example.

Start	Length	Description
2-3	2 bytes	Ex. Parameter between 0..65534, where 65535 is 'don't care'

6.3. MicroDisplay Source Address

The source address of the Grayhill standard MicroDisplay module shall default to 84 (54h). This may be modified either dynamically if the MicroDisplay module is Self Configurable and with the Commanded Address message in accordance with J1939-81. The new source address value shall be stored in non-volatile memory. The ability to change the source address will allow multiple standard MicroDisplay modules to coexist in the same system.

6.4. Physical Layer

The bit rate and signal lines shall comply with J1939/11 with a bit rate of 250kbps using an 82C251 or equivalent CAN transceiver.

6.5. Standard Messages

The Key Press Data PGN uses the Proprietary B PDU2 format (PF = 255) that broadcasts to no specific address the status of the keys, and the Control Data PGN uses Proprietary A PDU1 format (PF = 239).

The MicroDisplay module's configuration and control is done using the Proprietary A J1939 messages. The I/O is controlled using the J1939 Auxiliary I/O #2 PGN where each two bit field controls the respective input/output. This is a test

6.5.1. Key Press Data

PGN – 65282 (FF02h), Proprietary B PDU2 Format
Direction - Transmit
Priority – 6
Data Length - 8
Transmission Rate – 100ms (programmable)
***The Key Press PGN number can be reassigned using a configuration command.**

Start	Length	Description	Value
1.1	2 bits	Far Left Push Button	00 – Not pushed down 01 – Pushed down 10 - Unused 11 - Not available
1.3	2 bits	Second Push Button	Same
1.5	2 bits	Third Push Button	Same
1.7	2 bits	Far Right push button	Same
2-8	7 bytes	Unused	All bits set

Bits of unused buttons shall default to 0b11.

Example: Pressing the first key will cause the following message to be transmitted.

ID=0x18FF0254, LEN=8, DATA=0x01,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF

Example: Pressing the 3rd and 4th keys will cause the following message to be transmitted.

ID=0x18FF0254, LEN=8, DATA=0x50,0xFF,0xFf,0xFF,0xFF,0xFF,0xFF,0xFF

6.5.2. Backlight Intensity

PF – 239, Proprietary A PDU1 Format
PS – DA, The Source Address of the MicroDisplay module. Default value: 84 (54h)
PGN = (PF * 256) + PS = 61255
Direction - Receive
Data Length – 8

Start	Length	Desc.	Value
1.1	1 Byte	Backlight Intensity Control Byte	128 (80h)
2.1	1 Byte	Unused	255 – Not Available
3.1	1 Byte	Backlight Intensity	0 – Off 177 – Brightest* 255 – Not Available
4.1	1 Byte	LCD Blue	Same
5.1	1 Byte	LCD Red	Same
6.1	1 Byte	LCD Green	Same
7-8	2 Bytes	Unused	0xFFFF

***Values between 177 and 254 revert to 177**

6.5.3. Discrete Output

PF – 167 (A7h)
PS – DA, The Source Address of the MicroDisplay. Default value: 84 (54h)
PGN – 42752 (A700h), Auxilary I/O #2, PDU1 Format
Direction - Receive
Data Length – 8

Start	Length	Desc.	Values
1.1	2 bits	Output Control #1	0b00-Output Inactive (Off or not sourcing current) 0b01-Output Active (On or sourcing current) 0b10-Not used 0b11-No Change
1.3	2 bits	Output Control #2	Same
1.5	4 bits	Unused	0b1111
2-8	7 Bytes	Unused	0xFF

6.5.4. Discrete Input

PF – 254 (FEh)
PS – 217 (D9h)
PGN – 65241 (FED9h), Auxilary I/O #1, PDU2 Format
Direction - Transmit
Data Length – 8

Start	Length	Desc.	Values
1.1	2 bits	Input Status #1	0b00-Input Low 0b01-Input High 0b10-Not used 0b11-No Change
1.3	2 bits	Input Status #2	Same
1.5	2 bits	Input Status #3	Same
1.7	2 bits	Unused	0b11
2-8	7 Bytes	Unused	0xFF

6.5.5. Configuration and Control

PF – 239, Proprietary A PDU1 Format
PS – DA, The Source Address of the MicroDisplay module. Default value: 84 (54h)
PGN = (PF * 256) = 61184
Direction - Receive
Data Length – 8

Start	Length	Desc.	Values
1.1	1 Byte	Configuration and Control Command	224 – 254 Control Byte Command as described in Sec. 4
2.1	7 Bytes	Configuration and Control Data	0 – 255

***Note: Some commands need to be sent from a Source Address of 249 (F9h)**

6.5.6. Proprietary ID Message

PGN – 65408 (FF80h), Proprietary B PDU2 Format
Priority – 7
Data Length - 8
Transmission Rate – Upon request

Start	Length	Desc.	Values
1.1	8 Byte	Proprietary Unit ID Message	0 – 255. If all eight bytes are 255 then the ID is considered unprogrammed.

7. Configuration and Control

Changing the configuration and how the MicroDisplay module behaves is done with the Configuration and Control message described in Sec 6.5.5. The first byte serves as the command byte. Where applicable, changes take effect immediately and are stored in non-volatile memory unless otherwise noted.

All data indicated as 'xx' is 'Don't Care'

*Asterisk indicates the command must be sent from a Source Address of F9h

7.1. Backlight Intensity 128 (80h)

Data Field

Start	Length	Desc.	Value
1.1	1 Byte	Backlight Intensity	128 (80h)
2.1	1 Byte	Unused	255 – Not Available
3.1	1 Byte	Backlight Intensity Keypad	0 – Off 177 – Brightest 255 – Not Available
4.1	1 bytes	Backlight Intensity Blue	0 – Off 177 – Brightest 255 – Not Available
5.1	1 bytes	Backlight Intensity Red	0 – Off 177 – Brightest 255 – Not Available
6.1	1 bytes	Backlight Intensity Green	0 – Off 177 – Brightest 255 – Not Available

Example: Sending the following message to a module having an address of 0x54 will set the backlight brightness to approximately 50% of full brightness for the keypad backlights and the LCD green backlights while turning the LCD red and blue colors off.

ID=18EF5421, LEN=8, Data=0x80, 0xFF, 0x80, 0x00, 0x00, 0x80, 0xFF, 0xFF

7.2. Set Source Address 224 (E0h)

Data field

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
E0h	dd	xx	xx	xx	xx	55h	AAh

dd - Has a value between 0 and 255 and is the new source address
xx – Don't Care. Should be FFh following J1939 convention
55h – Low byte of 16 bit key
AAh – High byte of 16 bit key

7.3. Set Key Press Data PGN 225 (E1h)

Data field

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
E1h	aa	bb	xx	xx	xx	55h	AAh

aa – The least significant byte of the new PGN. Valid Range: 0..255
bb - The most significant byte of the new PGN. Valid Range: 0..255
xx – Don't Care. Should be FFh following J1939 convention
55h – Low byte of 16 bit key
AAh – High byte of 16 bit key

7.4. Set Transmit Data Priority 226 (E2h)

Data field

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
E2h	Key	Aux	xx	xx	xx	55h	AAh

Key – The new priority for the key message. Valid Range: 0..7
Aux – The new priority for the Auxliary I/O message. Valid Range: 0..7
xx – Don't Care. Should be FFh following J1939 convention
55h – Low byte of 16 bit key
AAh – High byte of 16 bit key

7.5. Set Transmit Data Transmission Period 227 (E3h)

Data field

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
E3h	dd	Event	dd	Event	xx	55h	AAh

Bytes 2 and 3 control the keypad transmission and 4 and 5 control the Auxiliary I/O transmission

dd – The value multiplied by 10ms. Valid range: 1..255 yielding between 10ms to 2.54 seconds. Minimum transmission period is limited to 20ms. Settings lower than two, other than zero result in a 20ms minimum.
Event – Valid settings is 0 or 1. A value of one sends the key message upon change in key information. Upon transmission the timer is reset. A value of zero will cause the message to be transmitted at the specified time interval.
xx – Don't Care. Should be FFh following J1939 convention
55h – Low byte of 16 bit key
AAh – High byte of 16 bit key

7.6. Changing J1939 NAME Fields* 228 (E4h)

***Note: Must be sent from source address of 249 (0xF9)**

Data field

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
E4h	subcmd	db0	db1	db2	xx	55h	AAh

subcmd – Represents the field within the name to change.
db0, db1, db2 – Data bytes associated with the sub command, LSB to MSB respectively.
xx – Don't Care. Should be FFh following J1939 convention
55h – Low byte of 16 bit key
AAh – High byte of 16 bit key

Sub Commands

Sub Cmd	Field	Description
0	ID	21 bits of db0..2
1	ECU Instance	Lowest 3 bits of db0
2	Function Instance	Lowest 5 bits of db0
3	Function	All 8 bits of db0
4	Vehicle System	Lowest 7 bits of db0
5	Vehicle System Instance	Lowest 4 bits of db0
6	Industry Group	Lowest 3 bits of db0
7	Arbitrary Addr. Capable	Lowest 1 bit of db0

Refer to J1939 base document for field value ranges and relationships.

7.7. Change ECUID Command* 229 (E5)

***Note: Must be sent from source address of 249 (0xF9)**

Data field

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
E5h	0..2	dd	xx	xx	xx	55h	AAh

0 – Selects ECUID Part Number to change
1 – Selects ECUID Location to change
2 – Selects ECUID Type to change
dd – Number of ASCII characters in the field, max of 64

7.8. Change ECUID Field Data* 230 (E6h)

***Note: Must be sent from source address of 249 (0xF9)**

Data field

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
E6h	1 to 7 bytes of ASCII Data						

* No Key used in bytes 7 and 8

7.9. Set Communication Watchdog Timeout Period* 231 (E7h)

***Note: Must be sent from source address of 249 (0xF9)**

Data field

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
E7h	dd	xx	xx	xx	xx	55h	AAh

dd – This holds the timeout value in units of 100ms that controls when and if a message will display on the screen indicating CAN messages that control the device have not been received within the timeout period. A value of zero disables this feature. A value of FFh results in no change.

Bytes 3 through 6 should be set to FFh for possible future expandability.

7.10. Graphic Object Command 233 (E9h)

This command works in conjunction with the Graphic Object Data message if applicable. These messages are used to create and display graphic objects.

Command with six bytes of parameters or less:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
E9	CMD	Six bytes of parameter Data					

Command with more than six bytes of parameters:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
E9	CMD	# of packets		Size of Data in Bytes			

- **CMD** – This 8-bit field holds the graphics sub command
- **# of packets** – This 16-bit field indicates the number of data packets that follows
- **Size of Data in Bytes** – This 32-bit field hold the total number of bytes in the data packets. Unused locations are included in the count. The Configuration and Control command byte and the Graphic Object subcommand byte are not included in the size count. Just the raw data itself is included.

7.11. Graphic Object Data 234 (EAh)

This command is the data command that works in conjunction with the Graphic Object Command for command sets that require more than six bytes of data.

Graphics Object Data:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
EA	Seq	Six bytes of parameter Data					

- **Seq –** The sequence number of the transmitted packet starting with 1. Used for verification of correct delivery of the data. In the event of more than 255 packets the sequence rolls over to zero then continues.
- **Data –** Up to six bytes of data.

7.12. Miscellaneous Settings 241 (F1h)

This command enables various settings.

Format:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
F1	Btn Ctrl	Aux Ctrl	Menu Save	xx	xx	KEY1	KEY2

- **Btn Ctrl:** A non zero value allows the transmission of the button message. A zero disables the message if button message transmission is not desired.
- **Aux Ctrl:** A non zero value allows the transmission of the AUXIO message. A zero disables the message if AUXIO message transmission is not desired.
- **Menu Save:** A non zero value causes the last selected object with the menu system to be displayed instead of the Default Object, if it exists, after boot.

8. Graphic Object Command Subset

The Graphic Object command subset uses the Graphic Object Command along with the Graphic Object Data for commands with larger than six bytes of data.

8.1. Clear Screen (CMD = 0)

Clears the MicroDisplay of all graphics objects and text. Does not change the mode (normal or inverse)

Format:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
E9	CMD	na					

8.2. Mode Select (CMD = 1)

Select the mode of the MicroDisplay

Format:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
E9	CMD	Mode	N/A				

- **MODE:** 0 – normal, 1 - inverted.

8.3. Draw Line (CMD = 2)

Draws a line on the display using a start X,Y coordinate and an end X,Y coordinate in pixels. The origin is located in the top, left of the display and has positive growth downward and to the right.

Format:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
E9	CMD	# Packets = 2		Size = 12			

- **Packets:** The number of data packets that follow.
- **Size:** The size in bytes of the parameters

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
EA	1	X1		Y1			
EA	2	X2		Y2		Color	Width

- **X1:** The column number of endpoint one of the line
- **Y1:** The row number of endpoint one of the line
- **X2;** The column number of endpoint two of the line
- **Y2:** The row number of endpoint two of the line
- **Color:** The color of the line.
- **Width:** The width of the line in pixels

8.4. Draw Box (CMD = 3)

Draws a box on the display using X,Y coordinates of the upper left corner and X,Y coordinates of the lower right corner in pixels. The origin is located in the top, left of the display and has positive growth downward and to the right.

Format:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
E9	CMD	# Packets = 2		Size = 12			

- **Packets:** The number of data packets that follow.
- **Size:** The size in bytes of the parameters

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
EA	1	X1		Y1		Width	
EA	2	X2		Y2		Color	Fill

- **X1:** The column number of the upper left corner
- **Y1:** The row number of the upper left corner
- **Width:** The line width of the box in pixels.
- **X2;** The column number of the lower right corner
- **Y2:** The row number of the lower right corner
- **Color:** The color of the box.
- **Fill:** If non zero the box is filled in according to the Color, hollow otherwise

8.5. Draw Circle (CMD = 4)

Draws a circle on the display using the X,Y coordinates of the radix and the radius in pixels.

Format:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
E9	CMD	# Packets = 2		Size = 12			

- **Packets:** The number of data packets that follow.
- **Size:** The size in bytes of the parameters

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
EA	1	X1		Y1		Rad	
EA	2	Color	Fill	Width			

- **X1:** The column number of the radix
- **Y1:** The row number of the radix
- **Rad;** The radius of the circle in pixels
- **Color:** The color of the box.
- **Fill:** If non zero the box is filled in according to the Color, hollow otherwise
- **Width:** The line width of the circle in pixels.

8.6. Draw Text (CMD = 5)

This command places text on the screen at the indicated location. The X,Y coordinates is the upper, left corner of the first character displayed. The ASCII string is delimited with the NULL character and is included in the size.

Format:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
E9	CMD	# Packets = (Size/6+1)		Size = various			

- **Packets:** The number of data packets that follow.
- **Size:** The size in bytes of the parameters

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
EA	1	X1		Y1		Font	color
EA	2..(#pkts)	ASCII character string.					

- **X1:** The column number of the upper left corner of the first character to print
- **Y1:** The row number of the upper left corner of the first character to print
- **Font:** A possible font size and style

8.7. Display Object (CMD = 6)

Displays an object, either static or dynamic, that is stored in non volatile memory according to the object's ID.

Format:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
E9	CMD	ID					

- **ID:** The object ID assigned with the Grayhill PC application. Serves as the handle with other functions.

8.8. Display Object Group (CMD = 7)

Displays a group of objects, either static or dynamic, that is stored in non volatile memory according to the list of objects in the following data packets.

Format:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
E9	CMD	# Packets = int((Size-1)/6)+1		Size			

- **Size:** The size in bytes of the parameters (1 to 65534 bytes, 65535 is reserved)
- **Packet Num:** Payload packet number. 1 to (Size-1)/6+1.

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
EA	Packet Num = 1	# of objects		Object 1		Object 2	
EA	Packet Num = Packet Num + 1	Object 3		Object 4		Object 5	

8.9. Kill Object (CMD = 8)

Kills a displayed Non volatile. In the case of a static image it simply erases the image. In the case of a dynamic image it erases the object from the display

Format:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
E9	CMD	ID					

- **ID:** The ID of the displayed object.

8.10. Overload Object (CMD = 17)

This command gives the ability to display static objects residing in the AppData section of flash with new parameters, such as the location on the screen, a new color for monochrome bitmaps and static text, and a new font size for static text.

Format:

DB 0	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7
E9	CMD	ID		X	Y	Color	Font

- **ID:** ID of the static object to display with new parameters
- **X:** x coord of the new location
- **Y:** y coord of the new location
- **Color:** New color the object should be displayed with, if applicable

- **Font:** New font the text object should be drawn with, if applicable.

The only objects that support overloading are the Bitmap and Static Text objects.

9. J1939 Messages

The following messages are defined in the J1939 documents and are implemented in the MicroDisplay module.

9.1. Address Claimed

PF – 238, Address Claimed
PS – 255, The destination address should always be the Global Address
PGN = (PF * 256) = 60928
Direction - Transmit
Data Length – 8
Transmission Rate – Upon boot or whenever requested

Start	Length	Desc.	Values
1.1	21 Bits	Identity Number	0 to 2^{21}-1
3.6	11 Bits	Manufacturers Code	294 (Assigned to Grayhill by SAE)
5.1	3 Bits	ECU Instance	0 (Default)
5.4	5 Bits	Function Instance	0 (Default)
6.1	8 Bits	Function	135 (MicroDisplay module, Default) *
7.1	1 Bit	Reserved	0 (Defined by SAE)
7.2	7 Bits	Vehicle System	0 (Default) *
8.1	4 Bits	Vehicle System Instance	0 (Default)
8.5	3 Bits	Industry Group	0 = Global (Default) * 1 = On-Highway Equipment 2 = Agricultural and Forestry Equipment 3 = Construction Equipment 4 = Marine 5 = Industrial-Process Control-Stationary 6 & 7 = Reserved
8.8	1 Bit	Arbitrary Address Capable	0 = Not Capable 1 = Capable (Default)

*Refer to J1939 base document for the Function value based on the Industry Group and Vehicle System combinations

9.2. PGN Request

PF – 234, PGN Request
PS – DA, The Source Address of the MicroDisplay module to respond or the Global Address
PGN = (PF * 256) = 59904
Direction - Receive
Data Length – 3

Start	Length	Desc.	Values
1.1	1 Byte	Byte 1 of PGN being requested (LSB)	0 to 255
2.1	1 Byte	Byte 2 of PGN being requested	0 to 255
3.1	1 Byte	Byte 3 of PGN being requested (MSB)	0

The following are the supported PGN's that can be requested from the MicroDisplay module. If the request is unsupported the MicroDisplay module shall respond with a NACK (Refer to J1939-21). Only Broadcast Announce Message (BAM) are supported for the Software and ECU identification. Therefore, the PGN request must be made to the global source address 255

9.2.1. ECU Identification Information

PGN = 64965
Direction - Transmit
Data Length – Variable
Transmission Rate – Upon Request
Multi Packet Transerred – Yes

Start	Length	Desc.	Values *
A	<=200	ECU Part Number	Ex. "3DYY1001-1"
B	<=200	ECU Serial Number	Ex. "123456"
C	<=200	ECU Location	Ex. "CAB"
D	<=200	ECU Type	"MicroDisplay"

***All fields asterisk delimited**

9.2.2. Software Identification

PGN = 65242 (0xFEDA)
Direction - Transmit
Data Length – Variable
Transmission Rate – Upon Request
Multi Packet Transerred – Yes

Start	Length	Desc.	Values
1	1 Byte	Number of software fields	1 to 125
2-N	Variable	Software ID field	ASCII characters. Each field delimited with an asterisk and up to 200 characters

9.3. Acknowledgement Message

PF = 232 (0xE8)
PS = DA
PGN = 64965 (0xE800)
Direction - Transmit
Data Length – 8
Transmission Rate – Upon appropriate response

Start	Length	Desc.	Values *
1	1 Byte	Control Byte	0 = Postive Acknowledgement 1 = Negative Acknowledgement 2 = Access Denied 3 = Cannot Respond
2	1 Byte	Group Function	Refer to SAE-J1939-21
3-5	3 Bytes	Reserved by SAE	
6-8	3 Bytes	Parameter Group being Acknowledged	

This message is send in response to a PGN Request of an unsupported PGN with the Control Byte having a value of one.

10. Object Usage

Each object is assigned an object ID. This is a unique 16 bit number assigned to each object in the application's data section of flash. Bits 14..0 define the ID value and has a range between 1 and 31,999. Object ID of zero is referred to as the Null Object and is used to assigned to parameters that take an object ID if no action is desired. . Bit 15 is used as the kill, or unload, flag. When set, instead of instantiating the object it will kill or remove the object. The Null Object with the most significant bit set will kill all objects and clear the display when instantiated. Objects are either instantiated or destroyed by referencing its ID.

Visible objects are placed on the display according to their coordinates. On the display, the origin is the upper, left corner and point (255, 127) is at the bottom right. The coordinates for bitmaps and text objects reference the top, left corner of the object. This is similar to how graphics object in the Windows operating system are.

The three possible colors an object can be displayed with are as follows:

- White – value 0, pixel Off
- Light Gray – value 1, pixel 1/3 on
- Dark Gray – value 2, pixel 2/3 on
- Black – value 3, pixel full on

The color supports two possible flags occupying the forth and fifth bits of the color variable. The values are

- 0x10 – Pixels are OR'd with the background. For monochrome bitmaps, only the ON pixels are written to the display leaving the pixel where a OFF pixel would be written alone.
- 0x20 – Pixels are inverted. For Text the OFF and ON pixels are inverted. This is good for indicating the selected item in a menu system.

To use the flags the color is simply added to the flag value. ex. to display fully dark text inverted set the color to 0x23, or 35 decimal.

10.1. Circle Object

Displays a circle on the LCD. The parameters are as follows:

- **X:** The x coordinate of the radix.
- **Y:** The y coordinate of the radix.
- **Radius:** The radius.
- **Color:** One of four possible colors: white, light gray, dark gray, black
- **Fill:** A non-zero value specifies a solid circle, hollow otherwise.
- **Width:** The thickness of the circle.

10.2. Box Object:

Displays a box on the LCD. The parameters are as follows:

- **X0:** The x coordinate of the upper, left corner of the box.
- **Y0:** The y coordinate of the upper, left corner of the box.
- **X1:** The x coordinate of the lower, right corner of the box.
- **Y1:** The y coordinate of the lower, right corner of the box.
- **Color:** One of four possible colors: white, light gray, dark gray, black
- **Fill:** A non-zero value specifies a solid circle, hollow otherwise.
- **Width:** The thickness of the sides.

10.3. Line Object:

Displays a line on the LCD. The parameters are as follows:

- **X0:** The x coordinate of point 1.
- **Y0:** The y coordinate of point 1.
- **X1:** The x coordinate of point 2.
- **Y1:** The y coordinate of point 2.
- **Color:** One of four possible colors: white, light gray, dark gray, black
- **Width:** The thickness of the sides.

10.4. Bitmap Object

Displays a bitmap on the LCD. The parameters are as follows:

- **X:** The x coordinate of the upper left corner of the bitmap.
- **Y:** The y coordinate of the upper left corner of the bitmap.
- **Height:** The height of the bitmap. This parameter is set by the Config Tool.
- **Width:** The width of the bitmap. This parameter is set by the Config Tool.
- **Format:** Specifies the number of bits per pixel, either 1 or 2. This parameter is set by the Config Tool.
- **Color:** One of four possible colors: white, light gray, dark gray, black. Only applicable for monochrome bitmaps
- **ORed:** References the most significant bit of the Color parameter. When set off pixels do not modify the pixels on the display. This allows

10.5. Static Text Object

Displays static text on the LCD. The parameters are as follows:

- **X:** The x coordinate of the upper left corner of the bitmap.
- **Y:** The y coordinate of the upper left corner of the bitmap.
- **Font:** The size of the desired font. Possible values are: 6x8, 8x12, 12x16, 24,32
- **Color:** One of four possible colors: white, light gray, dark gray, black. Supports color inversion
- **Invert:** References the most significant bit of the Color attribute. If set the foreground and background colors of the text is inverted.
- **Text:** The actual text that is to be displayed when the object is instantiated.

10.6. Group Object

This object holds a collection of other objects as a group. This allows multiple objects to be instantiated or killed as one when the parent group object ID is referenced. A group object can contain other group objects. The objects within a group object are instantiated in order. This is important to keep in mind when objects overlap each other. A group object item can hold an ID of another with the kill bit set. This allows automatic unloading of previously loaded group objects without the need to first unload the object before instantiating the group object. One example is to use the Null Object with the kill bit set as the first group object item. This will automatically kill all objects and clear the display before instantiating the remaining objects within the item list.

The parameters are as follows:

- **Object Count:** The total count of objects the group object contains. This parameter is set by the Config Tool.
- **Object List[]:** The list of objects instantiated/killed when the group object is instantiated/killed.

10.7. Special Objects

10.7.1.Default Object

This object references an object that is to be instantiated just after boot up or a reset or after the Splash Screen Object has timed out, if used.

10.7.2.Splash Object

This object refernces an object to be displayed as a splash screen for the specified amount of time. Once the time is expired, the Default Object is displayed, if used.

The parameters are as follows:

- **Object ID:** The object to be displayed.
- **Time:** When the special object is a splash screen object, this parameters specifies the period in milliseconds the referenced object is to be displayed. The object is automatically removed when the time expires.

10.8. Bargraph Object

This object monitors an SPN Range and adjusts the bargraph value accordingly. The parameters are as follows:

- **PGN:** This is the PGN that contains the SPN parameter to display and monitor
- **Source Address:** The source address of the sending device.
- **Bit Start:** The bit position within the data field of the CAN message where the variable lies. This value can range between 1 and 64
- **Bit Length:** This indicates the size of the monitored parameter in bits. The range is between 2 and 32
- **Control Bit Start:** For multiplexed PGN's this is the bit position of the control field that determines what the parameter is at Bit Start. Set to zero no control field exists. The range is between 1 and 64.
- **Control Bit Length:** This indicates the size of the control bit field. Set to zero if no control field exists. The range is between 2 and 32.
- **Control Compare Value:** This is the value that the bit field defined by the Control Bit Start and Control Bit Length must equal in order for the object to update itself with the data found at Bit Start.
- **MinVal:** This is the value of the measured parameter that is associated with the bargraph at zero percent.
- **MaxVal:** This is the value of the measured parameter that is associated with a bargraph of 100 percent.
- **Orientation:** Determines if the bargraph grows horizontally (0) or vertically (1)
- **X:** The x coordinate of the upper, left corner of the bargraph rectangle.
- **Y:** The y coordinate of the upper, left corner of the bargraph rectangle.
- **Length:** The length of the bargraph in the orientation it grows in.
- **Width:** The width of the bargraph.
- **Color:** One of four possible colors: white, light gray, dark gray, black.

10.9. Dynamic Text Object

This object monitors an SPN Range and displays the results as a text field on the display The parameters are as follows:

- **PGN:** This is the PGN that contains the SPN parameter to display and monitor
- **Source Address:** The source address of the sending device.
- **Bit Start:** The bit position within the data field of the CAN message where the variable lies. This value can range between 1 and 64
- **Bit Length:** This indicates the size of the monitored parameter in bits. The range is between 2 and 32
- **Control Bit Start:** For multiplexed PGN's this is the bit position of the control field that determines what the parameter is at Bit Start. Set to zero no control field exists. The range is between 1 and 64.
- **Control Bit Length:** This indicates the size of the control bit field. Set to zero if no control field exists. The range is between 2 and 32.
- **Control Compare Value:** This is the value that the bit field defined by the Control Bit Start and Control Bit Length must equal in order for the object to update itself with the data found at Bit Start.
- **Resolution:** This is the scalar the measured value is to be multiplied by.
- **Offset:** This is the offset added to the result of the measured value after being multiplied by the resolution.
- **Orientation:** Determines if the bargraph grows horizontally (0) or vertically (1)
- **X:** The x coordinate of the upper, left corner of the text.
- **Y:** The y coordinate of the upper, left corner of the text.

- **Font:** The size of the desired font. Possible values are: 6x8, 8x12, 12x16, 24,32
- **Color:** One of four possible colors: white, light gray, dark gray, black.
- **Text:** The actual text to be displayed with the formatting parameters.

The text is printed on the display using the C programming language's printf statement. The formatting string for variables of type 'float' is used. This has the following:
%[flags][field width][.precision]specifier.
Where:

- **Flags:** One or more of the following characters: +, (space),-,0,#
- **field width:** The minimum length in characters of the formatted field.
- **Precision:** The number of decimal places printed.
- **Specifier:** Always the lower case letter 'f'

ex. for a text string of "PI = %+06.2f rads" and a value of 3.14159, the displayed string will be:

PI = +03.14 rads

10.10. Menu Object

The Menu Object has the following flags:

- **Menu Option Flags:** Used to select the type of menu.
  - **Bit 1:** Specifies the object as the Root Menu object. Pressing the Menu button will display the first menu object in the AppData section when this parameter is equal to one.
  - **Bit 2:** Button Bound menu where each button is assigned to a static object. Each button is effectively bound to the first four menu items. When a button is pressed the Object ID To Display object is instantiated. Pressing another button will unload the previously instantiated object and display the buttons associated object.
  - **Bit 3:** When this flag is set the menu is a loop menu where pressing the left/right buttons will display a complete page of objects as opposed to displaying a list of choices to select. Most likely the Selected Item Object is a group object comprised of one or more monitoring objects along with other objects.
  - **Bit 4:** When this flag is set it inhibits the menu from unloading. Select this option if the menu item is the root menu and it is desired that pressing the menu button has no effect. This is useful when the root menu is in loop mode and the menu should never exit. The Selected Item object should be a group object that contains monitoring objects, graphics, or another menu object to enter a submenu.
- **Menu Item Count:** This is the total number of menu items in this menu object. This does not include any sub menus that an item may refer to.
- **Timeout Inhibit:** This flag is used to disable the timeout feature of the menu. By default, if the menu system is entered but an item is not selected by pressing the Enter key within about 5 seconds then the menu will unload itself and the previously selected object will be loaded.
- **Background Object:** This is the object that gets loaded once when the menu object is displayed. It is intended to load background objects that remain static while menu items are being navigated. If not background object is desired then assign the Null Object to this parameter.
- **Array of Menu Items**: Array size is specified by the Menu Item Count.

The Menu Items contain the following parameters:

- **Object ID To Display:** When the Enter button is pressed, the menu unloads itself and displays the object specified by this parameter. In the event that this

parameter holds the ID of a Menu Object the new menu is displayed with its own menu items.

- **Selected Item Object:** Pressing the left and right arrows selects a new menu item. This parameter displays the object that indicates a selected item. The object can be any object but most likely it will be a Static Text object or a Bitmap object.
- **Unselected Item Object:** This is the object that is displayed when the item is not selected when navigating through the menu. Typically the selected and unselected objects occupy the same LCD space since they are only displayed one at a time. This parameter can be set to the Null Object. When this happens the object pointed to by the Selected Item Object is redisplayed but with different parameters associated with the object.
- **Selected Color:** When the Unselected Item is the Null Object the selected item is displayed but with different attributes specified by this parameter. The color is a 16 bit value where the lowest two bits specifies the color of the Selected Item to be displayed. If the most significant bit of this parameter is set then the object is displayed in inverse mode.
- **Unselected Color:** When the Unselected Item is the Null Object the selected item is displayed with this parameter when the menu item is not selected.

Note: In order for bitmap object to be displayed with a different color it must be in monochorome (1 bit per pixel) format.

Menus with multiple sub menus can be realized by assigning the Object ID To Display object to another menu object. Pressing the Enter button will enter into a sub menu. Pressing the Menu button will back out of a sub menu and display the previously displayed menu object. In the event that the root menu object is displayed then pressing the menu button will unload the menu and display the previously loaded object.

10.11. Custom Gauge Object

The Custom Gauge Object gives the user the ability to create custom gauges that displays objects base on a value. This object uses an array of boundary segments composed of a compare value, and active object, and an inactive object. Either the active or inactive object is displayed depending on the comparison between the segment's compare value and the measured value. The segment's compare values must be an ascending order. This object has the following parameters:

- **PGN:** This is the PGN that contains the SPN parameter to display and monitor
- **Source Address:** The source address of the sending device.
- **Bit Start:** The bit position within the data field of the CAN message where the variable lies. This value can range between 1 and 64
- **Bit Length:** This indicates the size of the monitored parameter in bits. The range is between 2 and 32
- **Control Bit Start:** For multiplexed PGN's this is the bit position of the control field that determines what the parameter is at Bit Start. Set to zero no control field exists. The range is between 1 and 64.
- **Control Bit Length:** This indicates the size of the control bit field. Set to zero if no control field exists. The range is between 2 and 32.
- **Control Compare Value:** This is the value that the bit field defined by the Control Bit Start and Control Bit Length must equal in order for the object to update itself with the data found at Bit Start.
- **Option:** This value specifies the type gauge.
  - **Value 0:** Standard. If the measured value is greater than or equal to a boundary segment object's compare value then the corresponding Active Object is displayed otherwise the Inactive object is displayed

    - **Value 1:** Pointer. If the measured value is greater than or equal to a boundary segment's compare value and is less than the next boundary segment's compare value then the corresponding segment's active object is displayed. Otherwise the inactive segment is displayed. Only one boundary object is displayed at a time with this type.
- **Segment Count:** Holds the number of boundary segments in the array described below. The maximum number of boundary segments that can be defined is 50.
- **Array of Boundary Segments:** This array holds the following parameters
    - **Boundary Value:** This is what the measured value is compared to.
    - **Active Object:** This is the object that is displayed if the boundary value is less than or equal to the measured value for the standard option or if the measured value is between two adjacent segment compare values.
    - **Inactive Object:** The object that is displayed if the active object is not. Can be assigned the Null Object.

10.12. SPN Value Object

This object maps an exact value to a specific object. This is intended for SPN's where the measured data is a state rather than an analog value. It has the following parameters:

- **PGN:** This is the PGN that contains the SPN parameter to display and monitor
- **Source Address:** The source address of the sending device.
- **Bit Start:** The bit position within the data field of the CAN message where the variable lies. This value can range between 1 and 64
- **Bit Length:** This indicates the size of the monitored parameter in bits. The range is between 2 and 32
- **Control Bit Start:** For multiplexed PGN's this is the bit position of the control field that determines what the parameter is at Bit Start. Set to zero no control field exists. The range is between 1 and 64.
- **Control Bit Length:** This indicates the size of the control bit field. Set to zero if no control field exists. The range is between 2 and 32.
- **Control Compare Value:** This is the value that the bit field defined by the Control Bit Start and Control Bit Length must equal in order for the object to update itself with the data found at Bit Start.
- **Count:** This is the total number of elements in the lookup table.
- **SPN Value Table:** The table that associates a specific value to a object.
    - **Value:** The value to compare against the measured value.
    - **Object:** The object that is displayed if the Value matches the measured value exactly.

10.13. SPN Range Object

This object is used to display an object if the measured value falls within a specified range. The parameters are as follows:

- **PGN:** This is the PGN that contains the SPN parameter to display and monitor
- **Source Address:** The source address of the sending device.
- **Bit Start:** The bit position within the data field of the CAN message where the variable lies. This value can range between 1 and 64
- **Bit Length:** This indicates the size of the monitored parameter in bits. The range is between 2 and 32
- **Control Bit Start:** For multiplexed PGN's this is the bit position of the control field that determines what the parameter is at Bit Start. Set to zero no control field exists. The range is between 1 and 64.
- **Control Bit Length:** This indicates the size of the control bit field. Set to zero if no control field exists. The range is between 2 and 32.

- **Control Compare Value:** This is the value that the bit field defined by the Control Bit Start and Control Bit Length must equal in order for the object to update itself with the data found at Bit Start.
- **Min Val:** This holds the minimum value the measured value is compared to.
- **Max Val:** This holds the maximum value the measured value is compared to.
- **Below Range Object:** This is the object that is displayed if the measured value is less than the value stored in Min Val. This parameter can be assigned the Null Object.
- **In Range Object:** This is the object that is displayed if the measured value is between the Min Value and Max Value including both values. This parameter can be assigned the Null Object.
- **Above Range Object:** This is the object that is displayed if the measured value is greater than the value stored in Max Val. This parameter can be assigned the Null Object.

10.14. Bitmap Overload Object

This object is used to display an existing bitmap within the AppData section with new coordinates and/or, in the event of a monochrome bitmap, a new color. The parameters follow:

- **Overloaded Bitmap Object:** The existing bitmap object that is to be displayed with new parameters.
- **X:** The new x coordinate of the bitmap to be displayed.
- **Y:** The new y coordinate of the bitmap to be displayed.
- **Color:** If the object defined in Overloaded Bitmap Object is a monochrome bitmap, it will be displayed with the specified color.

10.15. CAN Object

This object is used to transmit a CAN message when instantiated. The full CAN ID, DLC, and all data byes must be known at design time. An option is available to detect an acknowledgement message. Once an acknowledgement message is received, the transmission sequence terminates and either the Fault Object or the Tx OK Object is displayed if they are not the Null Object. The parameters are as follows:

- **Send Count:** This is the number of times the CAN message will be transmitted.
- **Send Period:** This specifies the time in milliseconds between multiple transmission if Send Count is greater than one.
- **Option:**
  - **Option 0:** No acknowledgement message used.
  - **Option 1:** Monitor for an acknowledgement message.
- **CAN ID:** The full 29 bit identifier of the CAN message to be transmitted.
- **CAN DLC:** The data length of the transmitted CAN message
- **CAN Data[]:** Data bytes of the CAN message.
- **Ack PGN:** The PGN value of the acknowledgement message.
- **Ack SA:** The source address of the acknowledgement message.
- **Ack DLC:** The data length of the acknowledgement message
- **Ack Data[]:** The data of the acknowledgement message.
- **Ack Data Mask[]:** The mask values of the received acknowledgement message to check against the Ack Data[]. Cleared bits will be treated as ‘don’t care’
- **Fault Object:** When using an acknowledgement option, this is the object to be displayed if a valid acknowledgement message is not received during the duration of the transmission sequence. Can be assigned the Null Object.

- **Tx OK Object:** The object to be displayed when either the transmission is complete for the non-acknowledgement option or when a valid acknowledgement message was received. Can be assigned the Null Object.

10.16. Timeout Object

This object is used to monitor CAN traffic for a specific PGN and source address. If received within the specified time period the Default Object is displayed. If the message is not received within the time specified the Timeout Object is displayed. If the object times out then the specified message is received then the Timeout Object is removed and the Default Object redisplayed. The parameters are as follows:

- **PGN:** This is the PGN that contains the SPN parameter to display and monitor
- **Source Address:** The source address of the sending device.
- **Timeout Period:** The maximum amount of time that passes without receiving the expected CAN message before displaying the Timeout Object.
- **Default Object:** The object that is displayed when this object is first instantiated and when receiving the expected message after a timeout event.

10.17. Timer Object

This object is used to periodically toggle between two objects. The parameters are as follows:

- **Time On:** The time period in milliseconds to display the first object.
- **Time Off:** The time period in milliseconds to display the second object.
- **First Object:** The object displayed during the first time slice.
- **Second Object:** The object displayed during the second time slice.

Either of the two objects can be assigned the Null Object.

10.18. Backlight Object

This object is used. to control the backlighting of either or both the LCD and the keypads. When this object is instantiated the backlights are immediately modified. When this object is removed or killed the previously set backlighting values are restored. The parameters are as follow:

- **Red:** The red component of the LCD backlighting.
- **Green:** The green component of the LCD backlighting.
- **Blue:** The blue component of the LCD Backlihting.
- **Keypad:** The keypad backlighting.
- **Set As Default:** When non zero the values specified get stored as the devices default values.

The valid range for any of the backlight values is 0 to 254. Assigning the value of 255 to the any color for the LCD backlight will be interpreted as a ‘don’t care’. Same is true if 255 is assigned for the Keypad backlight. This allows the ability to change either the LCD or the keypad backlighting without affecting the other.

10.19. GPIO Out Object

This object allows the control of the two GPIO outputs when this object is instantiated either jointly or independently. The parameters are as follows:

- **Output 1:** The value to set the output to. 0 – Off, 1 – On, anything else is 'No change'
- **Output 2:** The value to set the output to. 0 – Off, 1 – On, anything else is 'No change'

When killing this object the output will turn off only if the value is either zero or one. Otherwise it will leave it alone. This allows the control of one output without affecting the state of the other.

10.20. GPIO In Object

This object monitors the three inputs and displays objects according to the state if the input. The parameters are as follows

- **Input x Low Object:** The object that is instantiated if the corresponding input is low.
- **Input x High Object:** The object that is instantiated if the corresponding input is high.

Where 'x' can be either 1, 2, or 3. If it is desired to only monitor one output simply assign the Null Object to the others. This allows up to three independently instantiated objects to monitor its own input without caring about the others.

10.21. Screen List Object

This object scrolls through a defined list of objects using the left and right buttons on the display. The parameters are as follows:

- **Flags:** TBD. Default to zero
- **Object Count:** The number of objects in the object array to loop through
- **Object List[]:** The array containing the objects. Size can vary.

10.22. Configuration Object

This object allows eeprom settings to be embedded into the project and also reports a revision of the PC Tool used to download the object file.

- **Revision:** 32 bit word containing revision information.
- **Flags:** General purpose flags, if needed.
- **Object Count:** The number of eeprom items in the array.
- **Item Array:** The array of items to be stored in the eeprom.

10.23. Backlight Control Object (Embedded)

This is an object that is embedded in the application as opposed to being located in the application data section. This is intended to be loaded via a Menu Object for the purposes of adjusting the backlights for both the LCD and the keypad. It is permanently assigned an object ID of 32,000

When using this object in a menu system, it is recommended to create its own menu object with only one menu item and assign the Selected Item Object parameter to this ID as opposed to the Object ID To Display parameter. This will allow the object to be displayed and still remain within the menu system. If no keys are pressed and if the Timeout Inhibit flag is not set then the previously displayed object before the menu was displayed will be redisplayed.

10.24. Contrast Control Object (Embedded)

This is an object that is embedded in the application as opposed to being located in the application data section. It is intended to be loaded via a Menu Object for the purposes of adjusting the LCD Contrast. It is permanently assigned an object ID of 32,001.

When using this object in a menu system, it is recommended to create its own menu object with only one menu item and assign the Selected Item Object parameter to this ID as opposed to the Object ID To Display parameter. This will allow the object to be displayed and still remain within the menu system. If no keys are pressed and if the Timeout Inhibit flag is not set then the previously displayed object before the menu was displayed will be redisplayed.

11. Complete Example

11.1. Circle Object Example

The following example illustrates how to draw a volatile circle object. The circle's parameters are as follows (all values are in decimal unless followed by an h):

- **Size** 16, Total size in bytes of the object
- **ID** 33, Arbitrary ID given to the object
- **Type** 1, Indicates a Circle type.
- **xLoc** 80, distance in pixels from the left of the display
- **yLoc** 40, distance in pixels from the top of the display
- **Radius** 10, in pixels
- **Color** 3, Black
- **Fill** 1, Non-zero means fill the circle
- **Width** 8, Irrelevant if Fill is 1

The following Graphic Object Command and Data messages would be sent to the MicroDisplay with its default source address of 84 (54h). Note that all bytes are left as decimal and are stored across multiple bytes in Little Endian format:

The CAN ID = 18EF54F9h, DLC = 8 for all messages.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
E9	4	2		12			
EA	1	80		40		10	
EA	2	3	1	8	xx	xx	xx

Upon successful transmission the MicroDisplay will display the object and respond with an ACK message.

The object is then displayed by sending a message with the same ID and DLC but with the following data:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
E9	6	33		xx	xx	xx	xx

12. Appendix A

12.1. Mating Connector.

Deutsch DT06-12SA.

12.2. Pinout

Pin #	Description
1	Vin Positive
2	Vin Return
3	RS485+
4	RS485-
5	Digital In 1
6	Digital In 2
7	Digital In 3
8	Digital Out 1
9	Digital Out 2
10	CAN Shield
11	CAN HI
12	CAN LOW