GLK24064-25
Technical Manual

Revision: 1.0

Contents

1 Introduction

The GLK24064-25 comes equipped with the following features;

- 240 x 64 pixel graphics display
- Text display using built in or user supplied fonts
- Adjustable contrast
- Backlighting
- Keypad interface
- RS-232 or I^2C communications

1.1 What to Expect From the GLK24064-25

The GLK24064-25 is designed as the display unit for an associated controller. The controller may be anything from a single board, special purpose micro-controller to a PC, depending on the application. This controller is responsible for what is displayed on the screen of the display

The display provides a simple command structure to allow both text and graphics to be transferred to the screen. Text fonts and graphics, if desired, are stored in the display's flash ROM and may be regarded as 'permanent' in that they survive power-off periods and don't change until explicitly reprogrammed.

The screen is backlit for low-light situations. Backlighting may be turned on or off under program control. Contrast is adjustable to compensate for differing lighting conditions and viewing angles.

1.2 What Not to Expect From the GLK24064-25

Since the display is intended to be used with a controller, it does not have any built in text editing functions. If a stream of ASCII characters is inputed they will be displayed, but the CR, LF, backspace, etc., will be ignored. If the application requires these functions, they must be provided by the software in the controller, which can issue the appropriate positioning commands to the display.

1.3 Keypad Interface

The keypad interface takes row / column input and converts it to ASCII characters, which are delivered out the RS-232 or I^2C port to the associated controller. Note that the keypad is not used to directly control any aspect of the operation of the display, which acts simply as a matrix to serial converter. If use of the keypad to control the display is required, the controller must then be programmed accordingly.

1.4 mogd.exe

Matrix Orbital has developed an interface program which exercises all the features of the display. It is also used to manage font and graphics downloads. The program, called "mogd.exe", is provided on the Matrix Orbital Cd and website.

To install mogd.exe follow these steps;

1. Insert the Matrix Orbital Cd-ROM into the Cd drive.
2. Locate the file "mogd.zip". It should be in the “Download” directory.
3. Unzip mogd.zip to a temporary directory, using a program such as Winzip, Pkzip, etc.
4. Double click on "setup.exe".
5. Follow the instructions on the screen to complete the installation.

After installation is complete there will be a Matrix Orbital entry under “Programs” in the “Start Menu”. Click on this entry to run mogd.exe.

The first time mogd.exe is run, some information will be required;

- The port number to be used. (Usually COM1 or COM2)
- The baud rate for the connection. (Use 19,200 for initial startup of the display)
- The type of display unit. (Set to 240 x 64 for the display)

Once this information is entered the program can be used to control all functions of the display.

1.5 Trying Out the GLK24064-25

Matrix Orbital suggests testing the display out before setting it up for any application. This is easily done with a PC. The following will be required;

- A 5V power supply. (8 to 30 VDC for Efficient Switching Supply (VPT) models)
- A power connector. The type used for 3.5” floppy drives works fine
- A PC with a spare RS-232 port (COM1 or COM2)
- The mogd.exe program, installed as in the previous section
- A 9 or 25 conductor RS-232 serial cable. If using a 25 conductor cable, a 25 to 9 pin adapter will also be required

The back view of the GLK24064-25 is shown below for reference.

Figure 1: Rear View of GLK24064-25

1.5.1 Here’s what to do:

1. Refer to the Figure above for the following steps.

2. Wire the connector to the power supply. On most connectors the RED lead will go to +5V and the BLACK lead to GND. **Do not connect the GLK24064-25 to a spare floppy drive power lead in the PC; the wiring is not correct and the unit will be damaged.**

> **NOTE** The Manufacturer's Warranty becomes void if the unit is subjected to over-voltage or reversed polarity.

3. Connect the display to the PC using the serial cable and adapter if required.
4. Connect the power connector, making sure that the +5V goes to V+ as shown in the diagram. Turn on the power: the LCD backlight should come on.
5. Use the mogd.exe program to exercise some of the features of the display to make sure everything works properly.
6. To experiment with typing text, run a PC terminal program, such as Hyperterm. Make sure it's configured to use the correct port. Set the baud rate to 19,200.

If characters are typed on the keyboard, they should now appear on the display screen. Note that CR, backspace etc., will not have any effect. Text will wrap around to the next line when the end of a line has been reached.

1.6 Trying out a Keypad

Since a number of different keypad types can be connected to the display, the results may be a little unpredictable. At this point all we need to do is make sure that the keypad and interface work, and possibly generate an ASCII map for any programming needs.

The keypad interface on the display converts a row / column connection to an ASCII character. By default, a keypress is transmitted as serial data immediately. Keypad buffering can be selected using the appropriate commands.

1.6.1 Here's what to do:

1. The PC should be running a terminal program, such as Hyperterm (as in the previous section).
2. With the display connected to the PC, plug in the keypad. If the connector has fewer pins than the one on the display, center it as well as possible.

> **NOTES**
>
> - The keypad connector must be wired with columns on one side and rows on the other side of the center of the connector. If the keypad isn't wired this way an adapter must be made or the connector must be rewired to meet this requirement.
> - The connector is reversible. Reversing the connector will not damage the keypad or the display, but will however, change the ASCII character map.

3. Press a key on the keypad. An upper case ASCII character (A-Y) should appear on the PC screen. Different keys should generate different characters.

To experiment, reverse the connector and see if it generates a more logical set of characters. Ultimately, the program in the micro-controller will have to 'map' these characters to the ones marked on the keypad, which will likely be different.

1.7 Manual Over-ride

Manual over-ride should only be required in one instance. If for some reason the module is set at a baud rate which cannot be produced by the host system and all communication to the display is lost, then the user should follow this simple procedure;

1. Turn off the display.
2. Put a jumper on pins 5 and 6 of the keypad connector.
3. Power up the display. The baud rate is now set to 19,200.
4. Remove the jumper and change the RS-232 port settings to the desired baud rate.
5. Turn off the display.
6. Power up the display.

Refer to the "Set RS-232 port speed" command for acceptable baud rates. This procedure does not change settings in the memory chip, it uses default settings stored in the main processor. This allows the user to communicate with the display when all other communications are lost. Once able to communicate with the display, the user may then change the default settings in the memory chip.

Please note, with the manual over-ride jumper in place, the unit will receive and perform commands such as turning on and off the backlight, but will not output text to the LCD.

1.8 Memory Chip Lock Down

The display uses a memory chip to store speed, startup screen contrast, fonts, bitmaps and I^2C settings. When everything has been changed to the desired settings and the unit is in a finished product or in the field, locking down the memory chip so no settings can be changed becomes very useful. This is only to be done by knowledgeable people. Any damage to the display by this procedure resulting from user error will not be covered under the Manufacturer's Warranty.

Figure 2: Lockdown

To lock down the memory chip, solder the jumper in the 'red box' and cut the trace where the red 'X' is. This will lock down the memory chip, preventing anything from being changed inside it until the track is restored and the solder jumper is removed.

2 Connections

2.1 Power

WARNINGS

- Do not apply any power with reversed polarization.
- Do not apply any voltage other than the specified voltage.
- Do not use any cables other than the cables supplied by Matrix Orbital, unless aware of the modifications required.
- Do not apply voltage to the DB-9 connector AND power connector
- Do not apply more then +5Vdc to pin #9 on the DB-9 connector.

Refer to the Figure below for this chapter.

Figure 3: Electrical Connections

Table 1: Connectors and Functions

Connector	Function
10 pin header	Keypad header, 5x5 matrix
4 pin	Power (Vdc) and I^2C communications or RS-232
DB-9F	RS-232/power
2 pin header	GPO header
3x2 header	I^2C and RS-232 select

Table 2: Connector Pinout

Pin 4	Ground
Pin 3	SDA (I^2C data) / Rx
Pin 2	SCL (I^2C clock) / Tx
Pin 1	Vdc

2.1.1 Power Connection (4 pin header)

Power is applied via pins 1 and 4.
Power requirement is;

- +5Vdc $\pm$0.25V on standard voltage units
- +7Vdc to +30Vdc with Wide Voltage and Efficient Switching Power supply (-VPT).

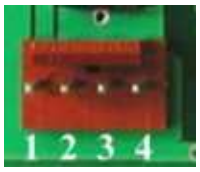

Figure 4: Power Connector

2.1.2 Five Volt Power Cable

If a display module is used in a PC it becomes tempting to plug a spare power connector into the unit. **Don't do this!** Wiring for the PC power connector and that required for the module are different as shown in the Figure below.

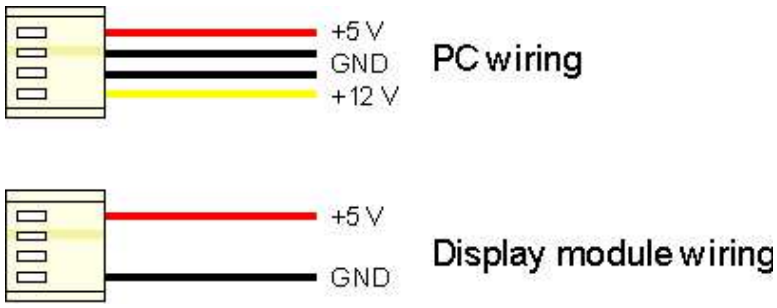

Figure 5: Wiring for Five Volt Modules

Matrix Orbital can supply an adapter cable designed to use with the display module when it's installed in a PC. The cable is wired as shown in the Figure below.

Figure 6: Five Volt Cable

Simply insert the splitter cable in series with a 'large' power connector (i.e., one going to a hard drive) and plug the small connector into the display module. The connector is 'keyed' and will only fit one way.

NOTE The connector provided does not allow access to the middle two pins, which are used for I^2C communications. If this functionality is required, Matrix Orbital suggests wiring a suitable connector.

2.1.3 Power Connection (DB-9 Connector)

The display can be powered via pin 9 on the DB-9 connector. No matter what voltage option the display is +5Vdc may only be used through pin 9, as it bypasses any voltage regulator on the display. These modifications must be done. However, improper modifications will damage the display and will result in a void of the Manufacturer's Warranty.

For standard power displays, two modifications have to be done;

1. Remove the solder jumper from the bottom pad and solder the top one.

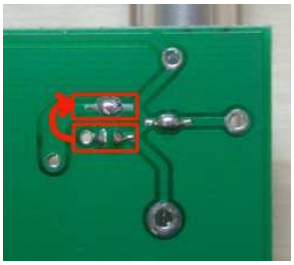

Figure 7: DB-9 Power Conversion

2. Solder the jumper in the 'red square'. It's located right of the DB-9 connector, in the middle.

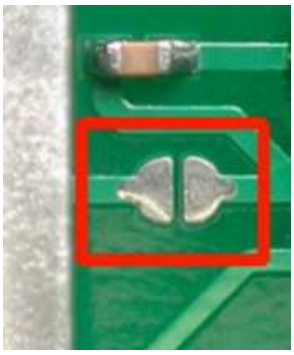

Figure 8: DB-9 Solder Jumper

WARNING For users with wide voltage units, please note; power may only be delivered by +5Vdc through the DB-9 connector, regardless of VPT.

Only one modification must to be done.

1. Solder the jumper in the 'red square'. It's located right of the DB-9 connector, in the middle.

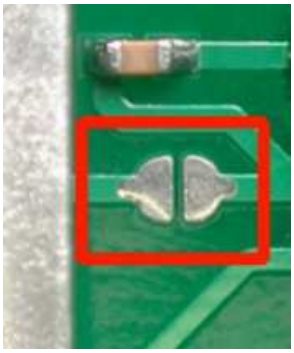

Figure 9: DB-9 Solder Jumper

2.2 Communications

2.2.1 RS-232 Communications and DB-9 Connector Pinout

A standard DB-9F is provided for RS-232 communications. Power may also be supplied via this connector if desired. As well, the two middle pins of the power connector may also be used for serial communications.

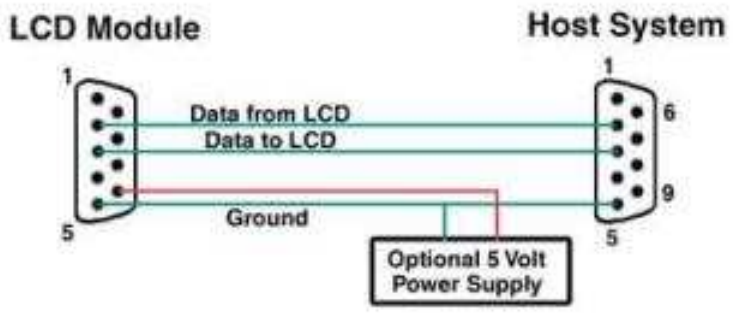

Figure 10: RS-232 and Power Connector

The RS-232 connector on the serial cable is wired so that a standard 'straight through' 9 pin D-sub cable may be used to connect the modules to a standard serial port such as COM ports on PCs. Note that this device complies with the EIA232 standard in that it uses signal levels from +/-12V to +/- 12V. It will can also operate correctly at TTL (0 to +5V) levels. To use standard RS-232 no modifications are required. For TTL, please see below.

Table 3: RS-232 Pinout

Pin Number	Direction	Description	LCD	Host
2	Data from LCD	Data Out (LCD)	Tx	Rx
3	Data to LCD	Data In (LCD)	Rx	Tx
5	-	Ground	gnd	gnd

2.2.2 TTL Communication

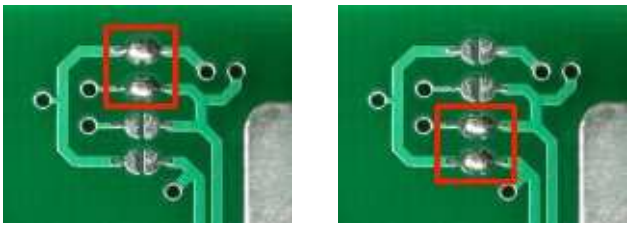

Figure 11: Standard and TTL Configuration

Standard: (+/-12V to +/- 12V) Jumper 1 and 3 soldered
TTL: (0 to +5V) Jumper 2 and 4 soldered

This will allow TTL RS-232 levels of communication through the DB-9 connector or the power connector.

2.2.3 RS-232 Communication through the Power Connector

RS-232 communication can also be achieved through the middle two pins of the power connector.

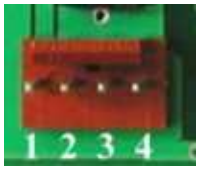

Figure 12: Power Connector

Table 4: Connector Pinout

Pin 4	Ground
Pin 3	Rx
Pin 2	Tx
Pin 1	Vdc

To allow this, the jumpers must be set as indicated below;

Figure 13: RS-232

2.2.4 I²C Communications

The display has I²C communications running at 100 Kbps and up to 127 units can be on a single communications line. The display's I²C communication lines are the two middle pins of the power connector. They are shared with RS-232 and have to be turned on. The I²C data line operates on 5 volt CMOS levels. The default I²C address is 0x50.

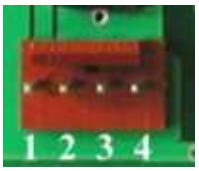

Figure 14: Power Connector

Table 5: Connector Pinout

Pin 4	Ground
Pin 3	SDA (I^2C data)
Pin 2	SCL (I^2C clock)
Pin 1	Vdc

The display does not work on I^2C by default. The modification shown below must be done;

Figure 15: I^2C

2.2.5 ACK

The idea of ACK is to indicate when the data has been received correctly. ACK does not indicate data incorrectly received. ACK simply fails to indicate when data is correctly received. Clearly, this is of limited usefulness and even less so with Matrix Orbital modules. Matrix orbital modules are not capable of failing to acknowledge and incorrectly received byte in response to that bytes transition. They are only capable of failing to acknowledge the bytes following the byte, which was not received. To fully understand the reasons for this one needs to understand something about how a Matrix Orbital module processes data. Basically the reason why a Matrix Orbital module might fail to receive a byte correctly is that it was unable to process the byte previous before the failed byte was transmitted. Because the module cannot possibly know that it would be unable to store the byte before the next byte was received it cannot know to not ACK. The reason for this situation in deference to situations one might be familiar with (i.e., memory chips, etc) is that the Matrix Orbital module employs a micro-processor to perform these data storage functions. A memory chip takes

care of these things entirely within hardware subsystems which operate at the same speed as the transmission themselves.

The display uses a standard Phillips 7bit address as defined by Phillips. However, Matrix Orbital specifies I^2C address in 8bits. The 8th bit, least significant bit (LSB or Low Order Bit) of the 8bit address is a read / write bit. If we take a standard Phillips 7bit address of 45hex this would be in binary 1000101. This is 7bits. Matrix Orbital would describe the Phillips I^2C address of 45hex as 8Ahex. The read address would be 8Bhex.

For more information on Phillips I^2C please visit;

`http://www.ping.be/~ping0751/i2cfaq/i2cindex.htm`

2.3 General Purpose Output

The general purpose output is provided to control relays or other devices via the display. This allows an external device to be turned on or off using the controller and software commands.

The GPO is meant to be used as a pair. The positive side of the GPO is connected to a power source of +5Vdc supplied by the module. This connection is via a 240 ohm resistor which limits the maximum current to 20 mA. The negative side of the GPO is connected to ground.

If the device which is being driven by a GPO requires a relatively high current (such as a relay) and has an internal resistance of its own greater than 250 ohms, then the 240 ohm resistor may be removed and replaced with a jumper. This resistor can be located directly to the left of the positive pin of the general purpose output.

NOTE This operation requires soldering. With the resistor removed the GPO does not have any over current or over / under voltage protection so care must be taken when using the GPO. For instance, if the external device is a relay it must be fully clamped (using a diode and capacitor) to absorb any generated back electro-motive force (EMF).

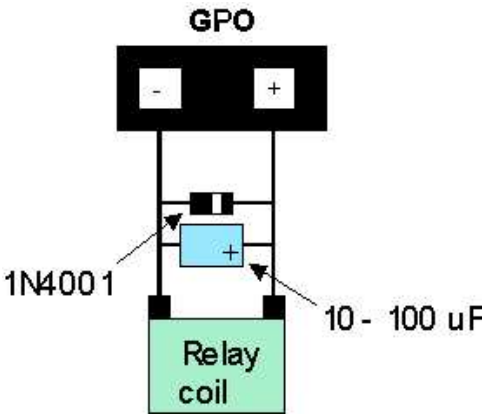

Figure 16: Clamping a Relay

3 Displaying Text

This chapter describes the various text display commands in detail.

3.1 General

Text is displayed on the display using fonts saved in its internal flash memory. The display is supplied with a 5x7 font installed. If this is suitable, there is no need to install any other fonts.

3.2 Writing Text to the Display

When the display receives a character, it displays that character at the position currently defined. The next character sent to the module then advances to the following position on the display. Characters are drawn using the currently selected font, and only characters defined in the current font are actually processed. Characters which are not defined by the current font are ignored, and the positioning is not advanced for the next character.

The position where text is to be displayed is a single pixel location stored in the display's volatile memory and maintained internally by the display's firmware. This position is manipulated by the commands shown in the following section.

3.3 Text Commands

In this section commands are identified by their names and decimal values.

3.3.1 Auto scroll on (254 81)

When auto scrolling is on, it causes the display to shift the entire display's contents up to make room for a new line of text when the text reaches the scroll position defined by the "Set font metrics" command in the display memory (normally the bottom right character position - default value for the GLK24064-25 is 64).

3.3.2 Auto scroll off (254 82)

When auto scrolling is disabled, text will wrap to the top left corner of the display area. Existing graphics or text in the display area are not erased before text is placed. When using proportional fonts without auto scrolling, care should be taken to clear areas where text is being written, particularly when wrapping occurs. This may be done using the "Draw solid rectangle" command with the colour set to white.

3.3.3 Set text insertion point (254 71 [col] [row])

This command sets the insertion point to the [column] and [row] specified. The insertion point is positioned using the base size of the current font (this command does not position the insertion point at a specific pixel). The pixel column used is determined by multiplying the width of the widest character in the font by [column]. The pixel row used is determined by multiplying the height of the font by [row + interline spacing].

3.3.4 Set current text insertion point to top left (254 72)

This command moves the text insertion point to the top left of the display area, based on the metrics of the current font. Refer to the "Set font metrics" command below for more details.

3.3.5 Set text insertion point using pixel values (254 121 [x][y])

This command sets the next position for text placement to an individual pixel location. The coordinate ([x position],[y position]) defines a pixel on the screen where the top left corner of the screen is defined as (0,0). This pixel location will be used as the top left corner of the next character of text which is sent to the module without any regard to 'font metrics' like character spacing or line spacing.

3.3.6 Set current font (254 49 [font ID])

This command instructs the display to use the font specified by [font identifier] as the default font. The value specified should refer to a font already present in the display's memory.

NOTE The font ID is established when the font is saved to the display, normally using the mogd.exe program. The installed 5x7 font ID is 0x01, unless changed by user.

3.3.7 Set font metrics (254 50 [metrics])

Where [metrics] = [left margin][top margin][x space][y space][scroll row]
This command defines the metrics of a font already present in the display's memory.

- [left margin] specifies the first pixel column to use for the first character in a row. In some instances, a font may not evenly fit on the screen and dividing the extra space between the margins will improve the overall appearance of the font.
- [top margin] specifies the top pixel row to begin drawing the first row of text on the display area.
- [x space] specifies the number of pixels to place between characters (i.e., character spacing).
- [y space] specifies the number of pixels to place between rows of text (i.e., line spacing).
- [scroll row] specifies the pixel row where scrolling should start (or, if auto scrolling is off, where wrapping should occur). Typically, this value should be set to the first pixel row immediately below the last row of text which will fit the display.

4 Displaying Graphics

This chapter describes the various graphics display commands in detail. Uploading a bitmap to position 1 will result in it being displayed on power up.

4.1 General

Since the display is a bit mapped device, it may be used to display graphics. Graphic images may be created by means of a pixel oriented graphics program, saved as bitmaps, and loaded into the display using the mogd.exe program. Images may be saved in the display's memory, and displayed upon command or they may be downloaded 'on the fly' (inline) during display operation.

Please note that 'saved' and 'on the fly' graphics images are processed differently. These differences must be taken into account when processing graphics.

1. **Saved bitmaps:** These use each byte (8 bits) to represent a vertical column of 8 pixels. The next byte represents the next column to the right. If the graphic is 'taller' than 8 pixels, the LSB of the next data byte will be the next pixel. Orientation is top to bottom - LSB to MSB. Pixels / bits are 'packed' - that is, if the height of the graphic is not an even multiple of 8, the leftover bits go on the next X column to the right, etc.
2. **Inline bitmaps:** These are processed horizontally, and each byte represents a horizontal row of 8 bits, with the next byte representing the next 8 bits to the right. Orientation is left to right - MSB to LSB, which is the opposite to the serial transmission sequence (bytes are sent LSB first).

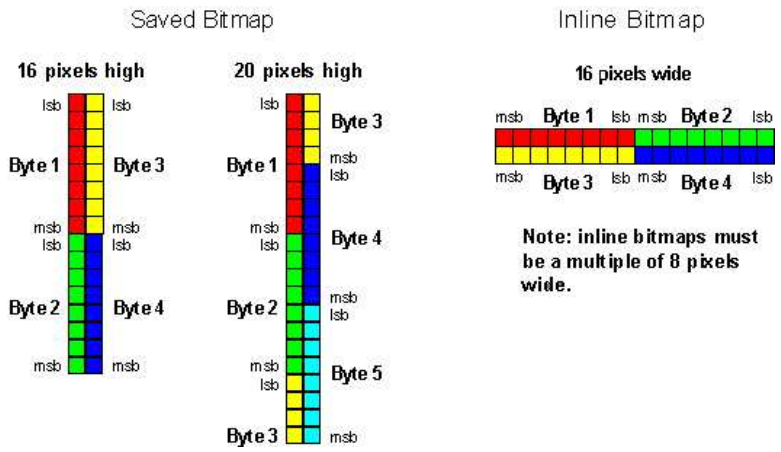

Figure 17: Graphic Bitmaps

Each pixel in a bitmap is described by a single bit, and may only have the values ON or OFF. For example, shades of grey are not supported.

4.2 Graphics Commands

In this section commands are identified by their names and decimal values.

The coordinate origin (0,0) is at the top left corner of the display. X values go from 0 to 239 (increasing toward the right) and Y values go from 0 to 63 (increasing toward the bottom).

4.2.1 Set drawing color (254 99 [color])

This command sets the drawing color for subsequent graphic commands that do not have the drawing color passed as a parameter. The parameter [color] is the value of the color where white 0 Hex, and black is 255 Hex.

NOTE All non-zero values will display as black.

4.2.2 Draw line (254 108 [x1][y1][x2][y2])

This command will draw a line from (x1,y1) to (x2,y2) using the current drawing color. Lines may be drawn from any part of the display to any other part. It may be important to note that the line may interpolate differently right to left, or left to right. This means that a line drawn in white from right to left may not fully erase the same line drawn in black from left to right.

4.2.3 Continue line (254 101 [x][y])

This command will draw a line with the current drawing color from the last line end (x2,y2) to (x,y). This command uses the global drawing color so the "Set drawing color" command should be used before the first line segment if required.

4.2.4 Put pixel (254 112 [x][y])

This command will draw a pixel at (x,y) using the current drawing color. The unit processes these requests fast enough to keep up with a steady stream at 115 kbaud, so flow control is not required.

4.2.5 Draw outline rectangle (254 114 [color][x1][y1][x2][y2])

This command draws a rectangular box in the specified color (0 = white, non-zero = black). The top left corner is specified by (x1,y1) and the bottom right corner by (x2,y2).

4.2.6 Draw solid rectangle (254 120 [color][x1][y1][x2][y2])

This command draws a solid rectangle in the specified color (0 = white, non-zero = black). The top left corner is specified by (x1,y1) and the bottom right corner by (x2,y2). Since this command involves considerable processing overhead, we **strongly recommend** the use of flow control, particularly if the command is to be repeated frequently.

This procedure is common for monitoring applications where there is a 'field' on the display that is constantly being updated from, say, a temperature sensor.

4.2.7 Initialize bar graph (254 103 [ref][type][x1][y1][x2][y2])

This command initializes a bar graph referred to by number [reference number] of type [type] with size from (x1,y1) (top left) to (x2,y2) (bottom right). A maximum of 16 bar graphs with reference numbers from 0 to 15 can be initialized as;

- [type = 0] Vertical, bottom referenced
- [type = 1] Horizontal left referenced
- [type = 2] Vertical top referenced
- [type = 3] Horizontal right referenced

The bar graphs may be located anywhere on the display, but if they overlap, they will not display properly.

NOTE It is important that [x1] is less than [x2], and [y1] is less than [y2].

This command doesn't actually draw the graph, it must be 'filled in' using the "Write to bar graph" command, described below. The unit saves time by only drawing that part of the bar graph which has changed from the last write, so the representation on the screen may not survive a screen clear or other corruptive action. A write of value zero, followed by new values will restore the proper look of the bar graph.

4.2.8 Write to bar graph (254 105 [reference number][value])

Once the bar graph has been initialized it can be 'filled in' using this command. This command sets the bar graph [reference number] to value [value]. [value] is given in pixels and should not exceed the available height / width of the graph. (If it does, the graph will simply be written to its maximum size.)

4.2.9 Display saved bitmap (254 98 [reference number][x][y])

This command causes a previously stored bitmap referenced by [reference number] to be displayed to the screen at pixel location (x, y) where this location defines the top left corner of the bitmap.

NOTE The reference number is established when the bitmap is saved, normally using mogd.exe. Bitmaps and fonts may use the same reference numbers. For example, it is possible to have both a bitmap 1 and a font 1.

4.2.10 Direct screen write bitmap (254 100 [x1][y1][x2][y2][data]{[data]})

This command is used to draw a bitmap to the screen directly without first storing it in the file space. This is a far quicker method of drawing a bitmap to the screen than using individual pixels. There are some irregularities and limitations to this command.

Unlike the format of bitmap download, this command expects data to be sent horizontally instead of vertically. The first byte of data is located in the top left of the area defined by (x1,y1) and the MSB of that byte is the left-most pixel. The second byte is the next 8 pixels to the right of the first byte. While each individual byte is oriented as one would expect, with LSB toward the right and the bytes moving from left to right across the screen, it's not a 'bit-stream' as might be expected. After the first pixel row of data is sent, data continues on the next pixel row.

The definitions of [x1] and [x2] must lie on 'byte boundaries'. That is, [x1] and [x2] must be defined as 0x00, 0x08, 0x10, etc. It is not possible to write a 10x10 bitmap to the screen at pixel location (13,13). It is however, possible to write a 16x10 bitmap to the screen at pixel location (24,13).

Use flow control during direct screen bitmap writes to avoid possible buffer overflow.

4.2.11 Initialize strip chart (254 106 [ref][x1][y1][x2][y2])

A 'strip chart' is an area of the screen reserved for horizontal scrolling. This is normally used as follows;

1. Initialize the strip chart. This reserves the appropriate area of the screen.
2. Draw a line segment at the right or left side of the strip chart.
3. Shift the strip chart to the right or left.
4. Draw the next line segment.

Used this way, the strip chart can produce a graph which scrolls smoothly horizontally in either direction. With text the strip chart can produce a 'marquis' effect.

NOTE If the strip chart is used with text we recommend the use of a 6 or 7 pixel wide fixed width character set, with each character placed 8 pixels from the start of the previous one.

Up to 7 strip charts ([ref] = 0 - 6) may be defined. To initialize a strip chart the user must define an area on the display in which to place the strip chart. (x1,y1) is the top left corner of the area to be used, where [x1] is the placement of the column where the strip chart is to begin and [y1] is the row. The user must then define [x2] as the bottom right column of the area to be utilized and [y2] as the bottom right row.

The definition of x must lie on 'byte boundaries'. That is, x must be defined as 0x00, 0x08, 0x10, etc. This restriction does not apply to y values.

4.2.12 Shift strip chart (254 107 [ref])

This command shifts the strip chart left or right. [ref] determines both which strip chart is used and which direction it will shift. The direction is selected by the most significant bit (MSB);

MSB = 0 shifts left

MSB = 1 shifts right

For example, if [ref] is 1;

254 107 1 (hex FE 6B 01) shifts left

254 107 129 (hex FE 6B 81) shifts right

This command shifts the contents of the area defined in the "Initialize strip chart" command 8 pixels at a time.

4.3 Flow Control

The display has built in flow control which is very useful during direct bitmap display and multiple pixel placement. Flow control is enabled or disabled by two commands. If flow control is enabled, the display will return an "almost full" message (0xFE) to the micro-controller when its internal buffer fills to a defined level, and an "almost empty" message (0xFF) when the buffer contents drop to a defined level.

4.3.1 Enter Flow Control Mode (254 58 [full][empty])

NOTE Flow control applies only to the RS-232 interface. It is not available for I^2C.

This command enables flow control. When the buffer fills so that only [full] bytes are available the display will return an "almost full" message (0xFE) to the micro-controller. When the buffer empties so that only [empty] bytes remain the display will return an "almost empty" message (0xFF) to the micro-controller.

The display will return the "almost full" message for every byte sent to the display until the used buffer space once more drops below the [full] level.

Whether the user is in 'flow control mode' or not, the module will ignore display or command bytes which would over-run the buffer. While in 'flow control mode' the unit will return 0xFE when the buffer is almost full even though it may have already thrown rejected data away. The buffer size for the display is 96 bytes.

When using this command in an application, selection of the value for the buffer almost full should be considered very carefully. This is a critical aspect of using this command to it's full potential. When using a host system or PC which contains a FIFO, the user should set the value of equal to or greater than the size of the FIFO. The reason for this is that the FIFO may be full when the host system receives 0xFE. In the case of 16550 UART the size at its maximum is 16, therefore the value of should be set to 16 or greater.

WARNING This mode must not be used during loading of fonts and bitmaps. It is highly recommended for use with direct screen write and multiple pixel placements.

4.3.2 Exit Flow Control Mode (254 59)

This command turns off flow control. Bytes may overflow the buffer without warning.

5 Keypad Interface

This chapter describes the keypad interface and associated commands in detail.

5.1 General

The display keypad interface processes the keypad row / column matrix into a serial (RS-232 or I^2C) data byte stream. Aside from this processing, the keypad has no effect on the display. If the user requires keystrokes to be sent to the display, they must then be routed through the micro-controller.

5.2 Connections

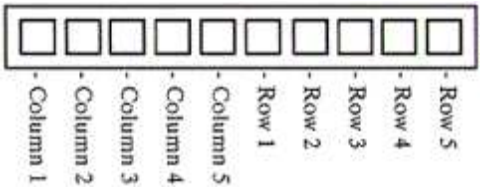

Figure 18: Keypad Connector

The connector is not 'keyed' so the keypad will probably plug in either of two ways. The display will not be damaged by reversing the connector. However, the keypad will generate a different ASCII character mapping for each position. If the connector has fewer than 10 pins it should be centered on the display connector.

The diagram shows the logical layout (row 1, column 1 in upper left). The connector for the keypad is a 10 pin 0.1" spacing male header. Pins 1 through 5 are columns and pins 6 through 10 are rows. The keypad is scanned whenever a key is pressed. There is no continuous key scan. This means that key presses are dealt with immediately without any appreciable latency. This also prevents electrical noise which is often caused by continuous key scans.

NOTE Please note that keypads may be laid out in a different pattern. If this is the case, the user will need to interpret the key codes differently.

Table 6: Keypad Layout

	Columns					
		1	2	3	4	5
	1	T	S	R	Q	P
Rows	2	O	N	M	L	K
	3	J	I	H	G	F
	4	E	D	C	B	A
	5	Y	X	W	V	U

NOTE The keypad connector must be wired with columns on one side and rows on the other side of the center of the connector. If the keypad isn't wired this way an adapter must be made or the connector must be rewired to meet this requirement.

5.3 I²C Interface

The keypad is read by I²C master read. In short, this means that a read of the module will always return the first unread key press. A read is initiated by writing to the module with its base address plus 1, then clocking the module's return byte after the module releases the SDA line. Much more detail on this basic I²C function can be found in the I²C specification by Phillips. A good reference is also available at;

`http://www.ping.be/~ping0751/i2cfaq/i2cindex.htm`

The module contains a ten key press buffer so that it can be polled for key presses at an infrequent rate (every 500 to 1000 mS is typical). All returned key presses indicate the presence or absence of additional logged key presses by the most significant bit (MSB - bit 7). If the user has pressed two keys since the last poll of the keypad interface, the first read will return the key code with bit 7 set and the second read will return the key code with bit 7 clear. The application must take into account this bit to keep up with user key presses. If there are no keypresses detected, the module will return zero (0x00).

5.4 RS-232 Interface

By default on any press of a key, the module will immediately send out the key code at the selected baud rate. This behavior can be modified using commands found in the next section.

5.5 Commands

5.5.1 Auto repeat mode on (254 126 [mode])

[mode] = 0x00 gives Resend Key Code mode
[mode] = 0x01 gives Key Down / Key Up code mode
Two Modes of auto repeat are available and are set via the same command.

1. **Resend Key Code:** This mode is similar to the action of a keyboard on a PC. In this mode, when a key is held down, the key code is transmitted immediately followed by a 1/2 second delay. After this delay, key codes will be sent via the RS-232 interface at a rate of about 5 codes per second. This mode has no effect if polling or if using the I^2C interface.
2. **Key Down / Key Up codes:** This mode may be used when the typematic parameters of the "Resend Key Code" mode are unacceptable or if the unit is being operated in polled mode. The host system detects the press of a key and simulates an auto repeat inside the host system until the key release is detected.

In this mode, when a key is held down, the key code is transmitted immediately and no other codes will be sent until the key is released. On the release of the key, the key release code transmitted will be a value equal to the key down code plus 20 hex. For example, the key code associated with key 'P' (0x50) is pressed, the release code is 'p' (0x70).

In RS-232 polled mode or via the I^2C interface, the "Key Down / Key Up" codes are used. However, the user should be careful of timing details. If the poll rate is slower than the simulated auto-repeat it is possible that polling for a key up code will be delayed long enough for an unwanted key repeat to be generated.

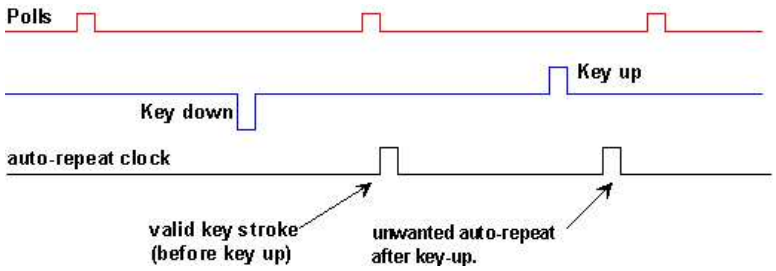

Figure 19: Poll Timing

5.5.2 Auto repeat mode off (254 96)

This command turns off auto repeat mode.

5.5.3 Auto transmit keypresses on (254 65)

In this mode, all keypresses are sent immediately to the host system without the use of poll keypad command. This is the default mode on power up.

5.5.4 Auto transmit keypresses off (254 79)

In this mode, up to 10 keypresses are buffered until the unit is polled by the host system via the poll keypad command. Issuing this command places the unit in polled mode.

5.5.5 Clear key buffer (254 69)

This command clears any unread keypresses. In a menuing application, if the user presses a key which changes the menu context, any following key presses may be inaccurate and can be cleared out of the buffer between menu changes to prevent jumping around the menu tree. It may also be used to, in effect, reset the keypad in case the host application resets for whatever reason.

5.5.6 Poll keypad (254 38)

This command returns any unbuffered keypresses via the RS-232 interface. The host system must be set up to receive the key codes. When the display receives this command it will immediately return any unbuffered keypresses which may have not been read already. If there is more than one keypress buffered the high order bit (MSB) of this returned keycode will be set (1). If this is the only buffered keypress, the MSB will be reset (0). If there are no buffered keypresses the returned code will be 0x00. Please note, to make use of this command the "Auto transmit keypress" mode should be off.

5.5.7 Set debounce time (254 85 [time])

[time] is in increments of 6554 microseconds.

This command sets the time between key press and key read. All key types with the exception of latched piezo switches will 'bounce' for a varying time, depending on their physical characteristics. The default debounce time for the module is about 52 mS, which is adequate for most membrane keypads.

6 Fonts and Graphics Files

6.1 General

Matrix Orbital graphic modules contain a sophisticated file system for storing and retrieving font information, bitmaps and system parameters; not unlike the way that a computer deals with files on a hard drive. However, the modules use no moving parts, therefore, data is stored far more reliably than data on a home PC.

Operationally, there is one important difference between the Matrix Orbital file system and that of a PC. While a PC will allow fragmentation of its files across the available file space, the Matrix Orbital file system takes great care to ensure that all parts of a file are stored together. This system works well to maximize storage space and operational efficiency. However, during file downloads, the modules may need to spend considerable time moving files to make room for the new file. This delay during download can be as much as a minute, but generally it will not exceed 10 seconds.

When a file is being downloaded with the same 'name' or reference number as a previously existing file, the old file needs to be deleted first. We cannot know if the new file is exactly the same size as the old file since the space vacated by the old files are filled by moving previously existing files down to fill up the vacated space. This ensures that no file space is wasted.

Of course, the average module will simply have files loaded into it and will then get to work without ever having to perform this file reorganization task. The file space may be rewritten up to 100 000 times, but most users will simply load in their fonts and bitmaps once and that will be it.

6.2 Using mogd.exe

The Matrix Orbital interface program "mogd.exe", which is provided on the disk and the website, generates and saves fonts larger than 14 pixels in height. It is also used to save graphic images (bitmaps) to the display.

To make use of smaller fonts it is recommended that a pre-generated font be used. These fonts can be located on the disk or the website. Unfortunately, integrating these fonts is not as straight forward as generating the fonts. To make use of these fonts the user must place the font files in their font directory as defined in the interface program. This directory can be found under "settings".

A font file consists of a single file with an extension.mgf and a directory which contains bitmaps for every character. All .mgf files are contained within the font directory and all bitmap directories are sub directories of the font directory. After download of a font file use a "Zip" program to "UnZip" the .mgf file and bitmap sub-directory into the font directory. Start or restart mogd.exe and click on the font tab. The font list of mogd should now display the new pre-generated font list.

6.3 Commands

In addition to the commands listed below, the mogd.exe program saves fonts and bitmaps to the display's flash memory.

6.3.1 Erase file (254 173 [type] [ref])

This command erases a file within the display memory, and will erase a single file at a time.

The command must be given two parameters: [type] and [ref]. The file type and reference number are defined when the file is saved to the display using mogd.exe. Since there is no command to list files in memory, the user must keep track of the memory contents.

- [type] = 1 is a font file
- [type] = 5 is a bitmap

Once this command is completed all files 'move up' and recover the empty space for efficient memory management.

6.3.2 Purge memory (254 33 89 33)

This command completely erases the display's non volatile memory. This removes all fonts, font metrics, bitmaps, and settings (current font, cursor position, communication speed, etc.). It is an 'odd' command in that it is three bytes in length. This is to prevent accidental execution.

6.3.3 Upload Font (254 36 [ref] [file size] [file data])

This command begins a font upload to the display's non-volatile memory. [ref] is the reference number to be used for this font. File size is a 2 byte value that must be calculated by the host before the transfer takes place.

6.3.4 Upload Bitmap (254 94[ref] [file size] [file data])

This command begins a bitmap upload to the display's non volatile memory. [ref] is the reference number to be used for this bitmap. File size is a 2 byte value that must be calculated by the host before the transfer takes place.

6.4 Working with Font Files

A font file consists of a header, a character list, and character bitmaps.

The header consists of;

- Placeholder for actual EOF (2 bytes, use 0xFF 0xFF - these bytes will be set to their final value by the module)
- Nominal character width (1 byte)
- Absolute font height (1 byte)
- ASCII value of first character defined in this file (1 byte)
- ASCII value of last character defined in this file (1 byte)

The character list consists of groups of 3 bytes per character;

- Offset to character bitmap (2 bytes)
- Actual width of this character (1 byte)

6.4.1 Font File in Table Form

The Table below shows the layout of a font file in table form.

Table 7: File Format

0xFF	0xFF	X size	Y size	Start	End	O-High	O-Low
Width	O-High	O-Low	Width	O-High	O-Low	Width	O-High
O-Low	Width	O-High	O-Low	Width	O-High	O-Low	Width
O-High	O-Low	Width	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data	Data	Data	Data
Data	Data	Data	Data	Data			

6.4.2 Uploading the File to the Module

The “Upload font” command is used to actually upload the font file. Recall that the syntax for this command is;

0xFE 0x24 [ref] [file size]

[file data]

In this example the file size is 94 bytes (0x5E) and the reference number is 2. The communications exchange between the host and the module looks like this;

Table 8: Uploading the File to the Module

Host sends	Module sends
0xfe	
'$' (command)	
'2' (reference)	
	'2' (echo reference)
0x01 (host confirms echo)	
0x5e (low size)	
	0x5e (echo)
0x01 (host confirms echo)	
0x00 (high size)	
	0x00 (echo)
	0x01 (file fits)*
0xFF (first byte of data)	
	0xFF (echo)
0x01 (host confirms echo)	
0xFF (second byte of data)	
	0xFF (echo)
0x01 (host confirms echo)	
0x20 (third byte of data)	
	0x20 (echo)
0x01 (host confirms echo)	
etc	

- If the module detects that the file will not fit in the available memory when the file size has been transmitted, it will send 0x08 instead of 0x01. In this case, the host should cease transmission. The module will return to a ready state.

From this point, the module treats all data as raw and just stores it away. The module will store the data, then read it back from memory and send the read value back to the host. If the host system receives an incorrect echo, it should send status as 0x08 instead of 0x01. This will terminate the transfer. Upon termination, the module will delete the partially completed file and return to a ready state.

6.4.3 A Sample Font File

Let's look at a short sample font file containing only the letters "h", "i" and "j". First we need to define the font size. For this example we'll use a 5x7 pixel font. Next, we have to draw the bitmaps for each of the characters.

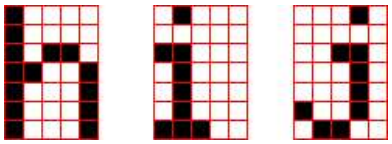

Figure 20: Bitmaps for h, i, and j

Now the bitmaps have to be converted to bytes. If the font is 8 bits high, it becomes a fairly simple job because each vertical column is simply one byte (lsb at the top). In this case, however, the font is only 7 bits high so the bytes 'wrap around'.

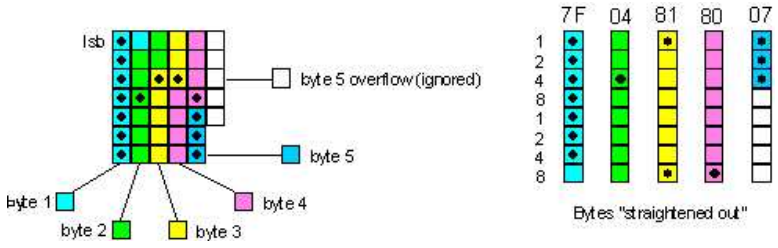

Figure 21: Bytes for a 7 Bit High Font

We've marked in the bits that are set for the letter "h". Remember that the bytes are 'inverted', i.e., the LSB is at the top. Each byte is shown in a different colour in the diagram. When the bytes are straightened out, it's simple enough to find their hex values, which are shown in the diagram above each byte. Trailing zero bytes at the end of narrow characters are not included in the file.

Table 9: Example of a Font File

0xFF	0xFF	0X05	0x07	0x68	0x6A	0x00	0x0F
0x05	0x00	0x14	0x03	0x00	0x17	0x04	0x7F
0x04	0x81	0x80	0x07	0xC4	0x3E	0x10	0x02
0x20	0xB1	0x07					

The colours refer to: Font information header, character 'h', character 'i', character 'j'.

Table 10: Explanation of Bytes in the File

FF FF	placeholders for actual EOF
05	font width
07	font height
68	first ASCII character defined
70	last ASCII character defined
00 0F	offset to definition of first character (h)
05	number of bytes in definition of first character
00 14	offset to definition of second character (i)
03	number of bytes in definition of second character
00 017	offset to definition of third character (j)
04	number of bytes in definition of third character
7F 04 81 80 07	definition of first character
C4 3E 10	definition of second character
02 20 B1 07	definition of third character

6.5 Working with Bitmap Files

Uploading a bitmap is the same as uploading a font file except that the character header information is not required.

The bitmap file consists of a header followed by the bitmap data. The header format is as follows;

- Placeholder for actual EOF (2 bytes, use 0xFF 0xFF - these bytes will be set to their final value by the module)
- x size of bitmap (1 byte)
- y size of bitmap (1 byte)

Bitmap data follows with the bits organized vertically from the top left. The last byte may be padded with zeros.

7 Miscellaneous Commands

7.1 General

The commands listed in this chapter don't readily fit in any of the other categories, or are used in more than one category.

7.1.1 Clear display (254 88)

This command clears the display and resets the text write position to the top left of the screen.

7.1.2 Set contrast (254 80 [contrast])

This command sets the display's contrast to [contrast], where [contrast] is a value between 0x00 and 0xFF (between 0 and 255). Lower values cause 'on' elements in the display area to appear lighter, while higher values cause 'on' elements to appear darker.

Lighting conditions will affect the actual value used for optimal viewing. Individual display modules will also differ slightly from each other in appearance. In addition, values for optimal viewing while the display backlight is on may differ from values used when backlight is off.

7.1.3 Set contrast and save (254 145 [contrast])

This command works in exactly the same way as the "Set contrast" command. The only difference is it saves the contrast value in the memory of the module, whereas, the previous command only changes the value for the duration of use.

7.1.4 Backlight on (254 66 [minutes])

This command turns on the backlight for a time of [minutes] minutes. (This specifies how long the backlight will remain on after reception of the command). If [minutes] is zero (0), the backlight will remain on indefinitely.

NOTE Backlight is always on by default on power up.

7.1.5 Backlight off (254 70)

This command turns the backlight of the display off.

7.1.6 General purpose output on (254 86)

This command turns ON the general purpose output. In the 'on' state the GPO provides 5 volts at a maximum current of 20 mA to operate external devices.

7.1.7 General purpose output off (254 87)

This command turns OFF the general purpose output.

7.1.8 Set I²C address 254 51 [address])

This command sets the I^2C write address of the module. This value must be an even number and the read address is one higher. For example if the I^2C write address is set to 0x50, then the read address is 0x51. The change in address is immediate. This address is 0x50 by default, and is reset temporarily back to that value when the 'manual over-ride' jumper is used on power up.

7.1.9 Read module type (254 55)

This command will return, over the RS-232 interface, the model type value of the module. Values for various modules at the time of this publication are as follows;

Table 11: Module Values

LCD0821 - 0x01	LCD2021 - 0x03	LCD2041 - 0x05
LCD4021 - 0x06	LCD4041 - 0x07	LK202-25 - 0x08
LK204-25 - 0x09	LK404-55 - 0x0A	VFD2021 - 0x0B
VFD2041 - 0x0C	VFD4021 - 0x0D	VK202-25 - 0x0E
VK204-25 - 0x0F	GLC12232 - 0x10	**GLK24064-25 - 0x15**
GLK12232-25- 0x22	GLK12232-25-SM - 0x24	LK404-AT - 0x31
LK402-12 - 0x33	LK162-12 - 0x34	LK204-25PC - 0x35

7.1.10 Set RS-232 port speed (254 57 [speed])

This command sets the display's RS-232 port to the specified [speed]. The change takes place immediately. [speed] is a single byte specifying the desired port speed. Valid speeds are shown in the Table below. The display can be manually reset to 19,200 baud in the event of an error during transmission (including transmitting a value not listed below) by setting the 'manual over-ride' jumper on the display controller board during power up. This command is ignored until this jumper is removed again.

Table 12: Speed Settings

Speed Value	Speed
20 Hex	9600 baud
0F Hex	**19200 baud**
95 Hex	57600 baud
03 Hex	76800 baud
8A Hex	115000 baud

7.1.11 Set Serial Number (254 52 [byte1] [byte2]

Modules may be delivered with the serial number blank. In this case the user may set the desired 2 byte serial number using this **one time only** command.

Upon the execution of this command, the module will echo these two bytes back over the RS-232 interface. The serial number may be set only once. Any future attempt to execute this command will result in no change and the module will return to the originally set serial number.

7.1.12 Read Serial Number (254 53)

This command will return, over the RS-232 interface, the serial number of the module as it was previously stored.

7.1.13 Read Version Number 254 54)

This command will return the firmware version number of the display.

8 Appendix: Command Summary

8.1 General

The operation of the display is controlled by a simple and consistent command set. Commands control;

- Text display
- Graphics display
- Keypad interface
- The display file system
- Miscellaneous operating parameters

This chapter includes summary tables of all commands.

8.2 Issuing Commands

Commands are issued to the display by the micro-controller. In a test setup, commands can be issued to the display by means of a BASIC program using the chr$( ) function. In the tables below, we've shown commands in hex, ASCII and decimal form. All commands begin with the prefix character 0xFE (254 decimal). These commands are issued on the serial communications link (I^2C or RS-232) at the currently defined baud rate.

For example (using BASIC in a test setup), the user could issue the command to clear the screen on the display by including the line:

```
PRINT#1,chr$(254);chr$(88)
```

in the BASIC program.

Or, with C the user could (using Zcomm serial library)

```
ZComm1->WriteCommByte(0xfe);
ZComm1->WriteCommByte('X');
```

8.3 On Numbers

Like all computerized devices, the display operates with commands and values in the form of binary numbers. These binary numbers are arranged in 8 digit (i.e. 8 bit) groups called bytes. The decimal value of a byte may have any value from 0 to 255.

Bytes are usually specified in either decimal or hexadecimal (base 16) form for convenience, since binary numbers are confusing to deal with directly. Hexadecimal (hex) numbers are particularly convenient because exactly two hexadecimal digits make up one byte, each hex digit representing 4 binary digits (4 bits) as shown here;

Table 13: Hex Value Table

Binary	Hex	Decimal	Binary	Hex	Decimal
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	A	10
0011	3	3	1011	B	11
0100	4	4	1100	C	12
0101	5	5	1101	D	13
0110	6	6	1110	E	14
0111	7	7	1111	F	15

Based on the table, the byte 01001011 can be represented in hex as 4B, which is usually written as any of 4Bh, 4BH, 4B hex or 0x4B. The numbers can also be expressed in decimal form if preferred.

8.3.1 ASCII Characters

Since computers deal internally with numbers only, but externally with both letters and numbers, several schemes were developed to 'map' written characters to numeric values. One such scheme has become universal, the American Standard Code for Information Interchange, or ASCII. ASCII tables are readily available from a number of sources. A few examples will do here;

Table 14: Example of an ASCII Table

The letter	A	has a value of	65 decimal or	41 hex
The letter	a	has a value of	97 decimal or	61 hex
The number	0	has a value of	48 decimal or	30 hex
The number	9	has a value of	57 decimal or	39 hex

This gives rise to the possibility of confusion when parameters are being set on the display. For example, some commands use a [type] parameter to indicate a file type. We're told that acceptable values are 0 and 5. All parameters must use numeric values (i.e., the actual byte values). If we send the ASCII number 0 by mistake it will actually give the value 48 decimal (30 hex) to the parameter, which is wrong. In the tables given in the following sections ASCII characters are shown as 'A', with single quotes.

8.4 Text Commands

Table 15: Text Commands

Command	**Syntax**	**Default**	**Notes**
Auto scroll on	FE 51 254 81 254 'Q'	off	Enables scroll at bottom of screen. Text will push display up one line to make room for new line.
Auto scroll off	FE 52 254 82 254 'R'	off	Disables auto scroll. Text will wrap to top left and overwrite existing text.
Set text insertion point	FE 47 [col] [row] 254 71 [col] [row] 254 'G' [col] [row]	n/a	Sets text insertion point using the base size of the current font.

Command	Syntax	Default	Notes
Set text insertion point to top left	FE 48 254 72 254 'H'		This command moves the text insertion point to the top left of the display area, based on the metrics of the current font. See "Set font metrics", for more details.
Set text insertion point using pixel values	FE 79 [x][y] 254 121 [x][y] 254 'y' [x][y]	n/a	Sets text insertion point to position (x,y), where x and y are in pixels. Value is top left corner of next text character.
Set current font	FE 31 [font id] 254 49 254 '1'	n/a	Sets font to [font id]. Font must be in memory.
Set font metrics	FE 32 [metrics] 254 50 [metrics] 254 '2' [metrics]	n/a	For definition of [metrics], please see "Set font metrics".

8.5 Graphics Commands

Table 17: Graphics Commands

Command	Syntax	Notes
Set drawing color	FE 63 [color] 254 99 [color] 254 'c' [color]	Sets color (0 = white, 255 = black) for the various drawing commands.
Draw line	FE 6C [x1][y1][x2][y2] 254 108 [x1][y1][x2][y2] 254 'l' [x1][y1][x2][y2]	Draws a line from x1,y1 to x2, y2. x values are from 0 - 63 (decimal) and y values from 0 - 239 (decimal).
Continue line	FE 65 [x][y] 254 101 [x][y] 254 'e' [x][y]	Continues line from last line end (x2,y2) to (x,y). Uses current drawing color.

Command	Syntax	Notes
Put pixel	FE 70 [x][y] 254 112 [x][y] 254 'p' [x][y]	Puts pixel in position (x,y). Uses current drawing color.
Draw outline rectangle	FE 72 [color][x1][y1][x2][y2] 254 114 [color][x1][y1][x2][y2] 254 'r' [color][x1][y1][x2][y2]	Draws a rectangular outline using color [color].
Draw solid rectangle	FE 78 [color][x1][y1][x2][y2] 254 120 [color][x1][y1][x2][y2] 254 'x' [color][x1][y1][x2][y2]	Draws a solid rectangle using color [color].
Initialize bar graph	FE 67[ref][type][x1][y1][x2][y2] 254 103 [ref][type][x1][y1][x2][y2] 254 'g' [ref][type][x1][y1][x2][y2]	Sets aside space for a bar graph. [ref] is reference number (0-15) for use by the Write to Bar Graph command. [type] has values: 0 = vertical, starting from bottom 1 = horizontal, starting from left 2 = vertical, starting from top 3 = horizontal, starting from right
Write to bar graph	FE 69 [ref][value] 254 105 [ref][value] 254 'I' [ref][value]	Fills the bar graph referred to as [ref] from start to [value]. [value] is in pixels.
Display saved bitmap	FE 62 [ref][x][y] 254 98 [ref][x][y] 254 'b' [ref][x][y]	Causes bitmap [ref] to be displayed with its top left corner starting at position (x,y).
Initialize Strip Chart	FE 6A [ref][x1][y1][x2][y2] 254 106 [ref][x1][y1][x2][y2] 254 'j' [ref][x1][y1][x2][y2]	[ref] is a reference number from 0 - 6, allowing 7 strip charts to be defined.
Shift Strip Chart	FE 6B [ref] 254 107 [ref] 254 'k' [ref]	[ref] selects the strip chart and the direction in which it moves. Movement is always 8 pixels at a time. MSB = 0 shifts left MSB = 1 shifts right.

8.6 Keypad Interface Commands

Table 19: Keypad Interface Commands

Command	Syntax	Default	Notes
Auto repeat mode on	FE 7E [0|1] 254 126 [0|1] 254 '~' [0|1]	off	Applies to keypad only. 0 = 200 ms typematic, 1 = key down/key up codes sent.
Auto repeat mode off	FE 60 254 96 254 '‘'	off	Applies to keypad only.
Auto transmit keypresses on	FE 41 254 65 254 'A'	on	Sets auto transmit mode for keypad. Keypresses are transmitted to host without polling.
Auto transmit keypresses off	FE 4F 254 79 254 'O'	off	Up to 10 keypresses buffered until polled.
Clear key buffer	FE 45 254 69 254 'E'	n/a	Clear unread keypresses.
Poll keypad	FE 26 254 38 254 '&'	n/a	Returns buffered keypresses to application.
Set debounce time	FE 55 [time] 254 85 [time] 254 'U' [time]	52 ms	Resolution: 1 = 0.6554 ms

8.7 File System Commands

In addition to these commands, the mogd.exe program is used to download fonts and graphics to the display.

Table 21: File System Commands

Command	Syntax	Default	Notes
Erase file	FE B0 [type] [ref] 254 173 [type] [ref]	n/a	Erases file in memory. Type = 1 is font, type = 5 is bitmap. [ref] is reference number.
Purge memory	FE 21 59 21 254 33 89 33	n/a	Removes all fonts, font metrics, bitmaps and settings from memory.
Upload bitmap	FE ^E [ref] [size] [data] 254 94[ref] [size] [data] 254 '^' [ref] [size] [data]	n/a	Uploads a bitmap to the memory of the display.
Upload font	FE 24 [ref] [size] [data] 254 36 [ref] [size] [data] 254 '$' [ref] [size] [data]	n/a	Uploads a font to the memory of the display.

8.8 Miscellaneous Commands

Table 23: Miscellaneous Commands

Command	Syntax	Default	Notes
Clear display	FE 58 254 88 254 'X'	n/a	Clears screen of text and graphics, places text cursor at top left.
Set contrast	FE 50 [contrast] 254 80 [contrast] 254 'P' [contrast]	128	Sets display contrast. Compensates for viewing angle. Contrast is a value between 0 and 255 (hex 0 to FF). Larger = darker.
Set contrast and save	FE 91 [contrast] 254 145 [contrast]	128	Same as "set contrast" but saves [contrast] as default.

Command	Syntax	Default	Notes
Backlight on	FE 42 [minutes] 254 66 [minutes] 254 'B' [minutes]	on	Backlight will stay on for [minutes]. If [minutes] = 0 backlight will stay on permanently.
Backlight off	FE 46 254 70 254 'F'	on	Turns off backlight.
General purpose output on	FE 56 254 86 254 'V'	off	Turns the general purpose output ON.
General purpose output off	FE 57 254 87 254 'W'	off	Turns the general purpose output OFF.
Set I^2C address	FE 33 [address] 254 51 [address] 254 '3' [address]	ox50	Value is write address and must be even, read address is 1 higher.
Read module type	FE 37 254 55 254 '7'	see table	Reads the module type. Returns a 1-byte hex value.
Set RS-232 port speed	FE 39 [speed] 254 57 [speed] 254 '9' [speed]	19,200	Sets RS-232 speed.
Enter flow control mode	FE 3A [full] [empty] 254 58 [full] [empty] 254 ':' [full] [empty]	off	Sets "full" and "empty" marks for the 96 byte display buffer. When buffer reaches [full] display will return 0xFE to host. When buffer reaches [empty] display will return 0xFF.
Exit flow control mode	FE 3B 254 59 254 ';'		Turns off flow control (buffer handshaking).

Command	Syntax	Default	Notes
Set Serial Number	FE 34 [byte1][byte2] 254 52 [byte1][byte2] 254 '4' [byte1][byte2]		This is a one-time-use command which works only on units without factory set serial numbers.
Read Serial Number	FE 35 254 53 254 '5'		Reads the two byte serial number of the module.
Read Version Number	FE 36 254 54 254 '6'		Reads the firmware version number of the module. Returns a 1-byte hex value.

9 Appendix: Specifications

Table 25: Environmental Specifications

	Standard Temperature	Extended Temperature
Operating Temperature	0°C to +50°C	-20°C to +70°C
Storage Temperature	-20°C to +70°C	-30°C to +80°C
Operating Relative Humidity	90% max non-condensing	90% max non-condensing
Vibration (Operating)	4.9 m/s^2 XYZ directions	
Vibration (Non-Operating)	19.6 m/s^2 XYZ directions	
Shock (Operating)	29.4 m/s^2 XYZ directions	
Shock (Non-Operating)	490 m/s^2 XYZ directions	

Table 26: Electrical Specifications

Supply Voltage	4.75 - 5.25 Vdc (Optional 7 - 30 Vdc)
Supply Current	31 mA typical
Supply Backlight Current	160 mA typical

Table 27: Optical Characteristics

Pixel Layout	240 x 64 pixels XxY
Number of Characters	320 (maximum 40 characters x 8 Lines with 5x7 font)
Display Area	127.16 x 33.88mm XxY
Dot Size	0.49 x 0.49mm (XxY)
Dot Pitch	0.53 x 0.53mm (XxY)
LED/CCFL Backlight Life	100, 000 hours typical
Color of Illumination	Yellow Green (LED), Light Blue (CCFL)

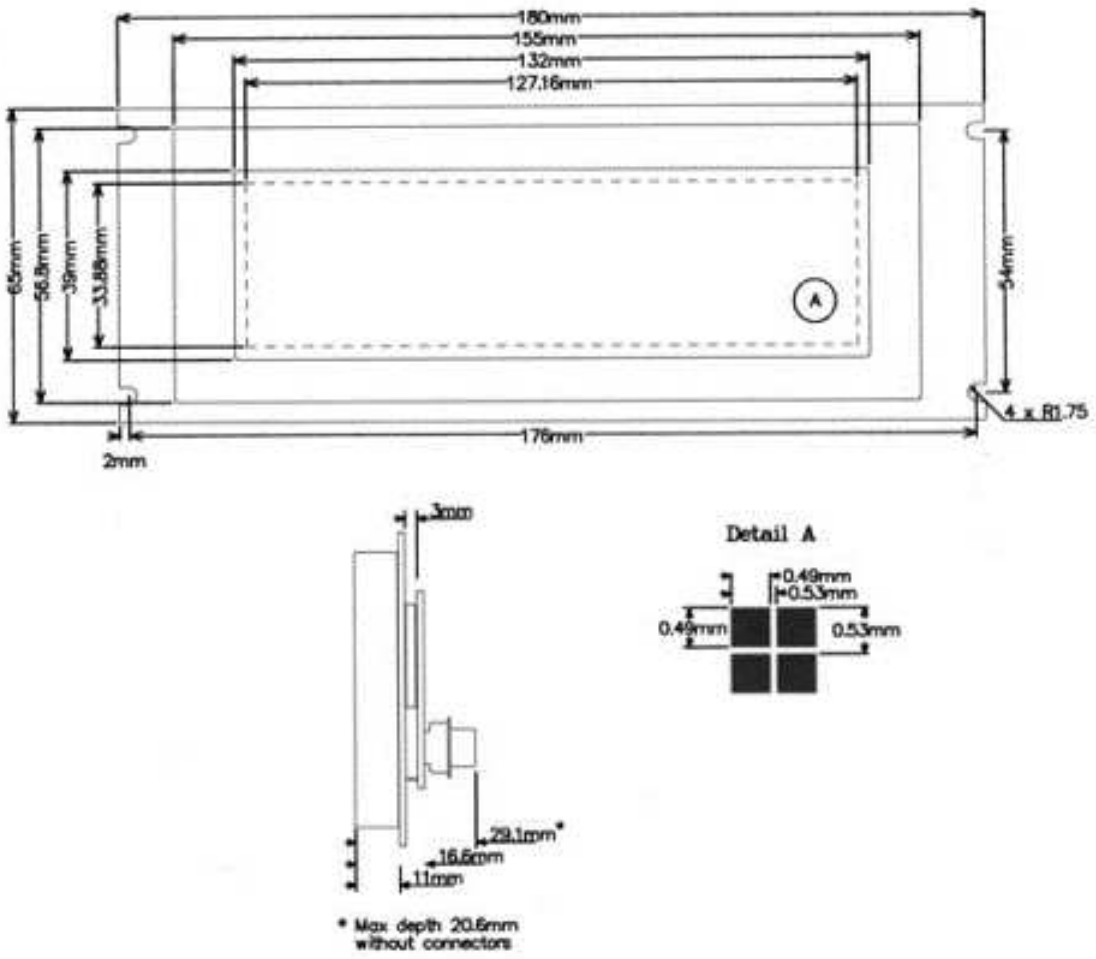

Figure 22: Physical Layout

Table 28: Display Options and Availability

Model	E	V	VPT	EL	FL
LCD0821	a	a	n/a	n/a	n/a
LCD2041	a	a	n/a	n/a	n/a
LCD4021	a	a	n/a	n/a	n/a
LCD4041	a	n/a	a	a	n/a
LK162-12	a	a	n/a	n/a	n/a

Model	E	V	VPT	EL	FL
LK202-25	a	a	n/a	n/a	n/a
LK204-25	a	a	a	n/a	n/a
LK402-12	a	a	a	n/a	n/a
LK404-55	a	n/a	a	a	n/a
LK404-AT	a	n/a	a	a	n/a
BLC2041	a	a	n/a	n/a	n/a
VFD2041	a	a	n/a	n/a	n/a
VK202-25	a	a	n/a	n/a	n/a
VK204-25	a	a	a	n/a	n/a
BVF2041	a	a	n/a	n/a	n/a
GLC24064	a	n/a	a	a	n/a
GLK12232-25	n/a	n/a	n/a	n/a	n/a
GLK12232-25-SM	n/a	n/a	n/a	n/a	n/a
GLK24064-25	**a**	**n/a**	**a**	**n/a**	**a**