

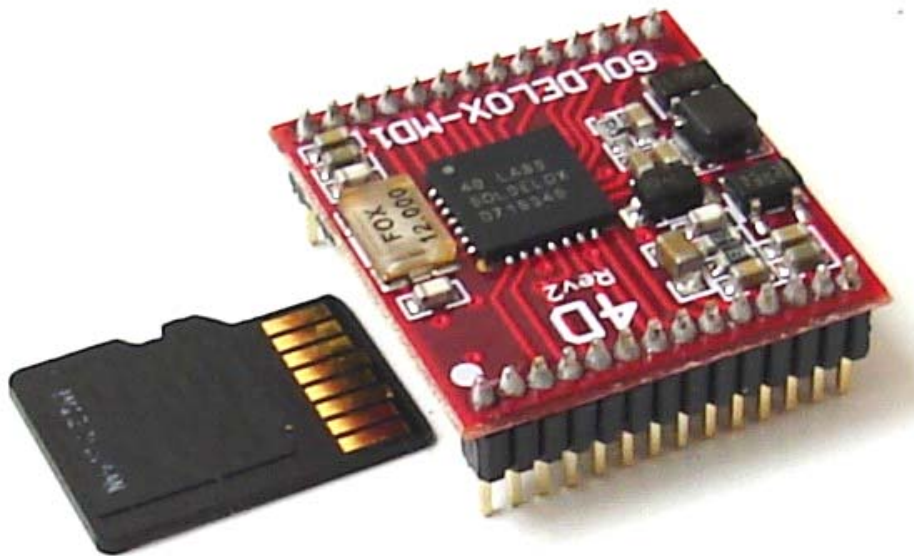
GOLDELOX

USERS MANUAL

GOLDELOX MD1
 μ SD

Tiny OLED-LCD Graphics Controller
with integrated micro-SD memory card support

Revision 1.0



4D Systems



GOLDELOX MDI

PROPRIETARY INFORMATION

The information contained in this document is the property of [4D Systems Pty. Ltd.](#), and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission. It should not be used for commercial purposes without prior agreement in writing.

[4D Systems Pty. Ltd.](#) Endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of [4D Systems](#) products and services is continuous and published information may not be up to date. It is important to check the current position with [4D Systems](#).

Contact details are available from the company web site at www.4dsystems.com.au

All trademarks recognised and acknowledged.

Copyright [4D Systems Pty. Ltd.](#) 2000-2007

DISCLAIMER OF WARRANTIES & LIMITATION OF LIABILITY

4D Systems Pty. Ltd. makes no warranty, either express or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose. 4d systems' sole obligation and liability for product defects shall be, at 4d systems' option, to replace such defective product or refund to buyer the amount paid by buyer therefore. In no event shall 4D Systems' liability exceed the buyer's purchase price.

The foregoing remedy shall be subject to buyer's written notification of defect and return of the defective product within ninety (90) days of purchase. The foregoing remedy does not apply to products that have been subjected to misuse (including without limitation static discharge), neglect, accident or modification, or to products that have been soldered or altered during assembly, or are otherwise not capable of being tested, or if damage occurs as a result of the failure of buyer to follow specific instructions.

In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.



GOLDELOX MDI

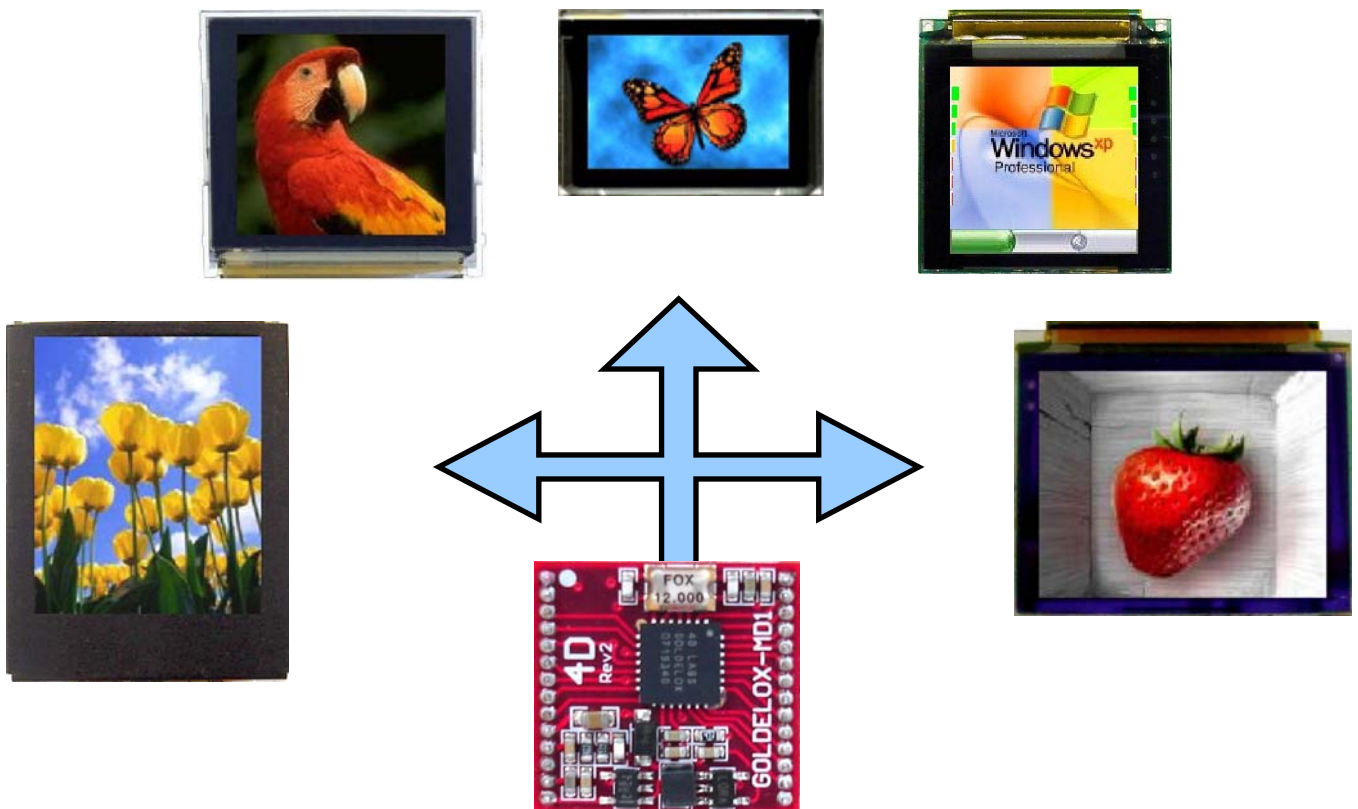
Table of contents

1	Introduction	4
2	Features	5
3	Pin Description	6
4	Reference Designs	7
5	Personality Module Micro Code (PmmC)	11
6	Graphics Command Set	12
	6.1 General Command Set	13
	6.2 Display Specific Command Set	37
	6.2 Extended Command Set	41
7	Serial Interface (TTL Level)	60
8	USB Interface	62
9	Specifications	63
10	Mechanical Details	64
11	Colour Organistaion	65
	11.1 65,536 Colour Bitmap Organisation	65
	11.2 256 Colour Bitmap Organisation	67
12	Power-Up Reset	68
13	Related Products	69
14	Appendix	70
	14.1 OLED Display Precautions	70
	14.2 Help and Other Information	70

1 Introduction

The **GOLDELOX-MD1** is a compact & cost effective all in one 'SMART' OLED and LCD panel controller module with embedded graphics that will deliver 'stand-alone' functionality to your project. The 'simple to use' embedded commands not only control background colour but can produce text in a variety of sizes as well as draw shapes (which can include user definable bitmapped characters such as logos) in 256 or 65,536 colours whilst freeing up the host processor from the 'processor hungry' screen control functions. This means a simple micro-controller with a standard serial or USB interface can drive the module with total ease.

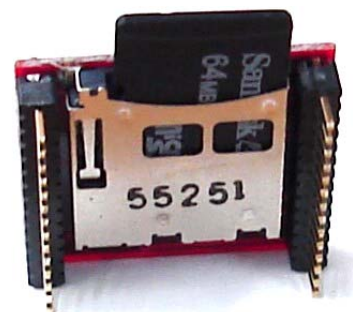
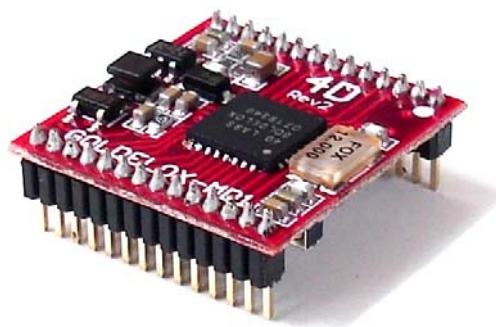
Figures below show some of the graphics capability of the **GOLDELOX-MD1** with various OLED and LCD display panels.



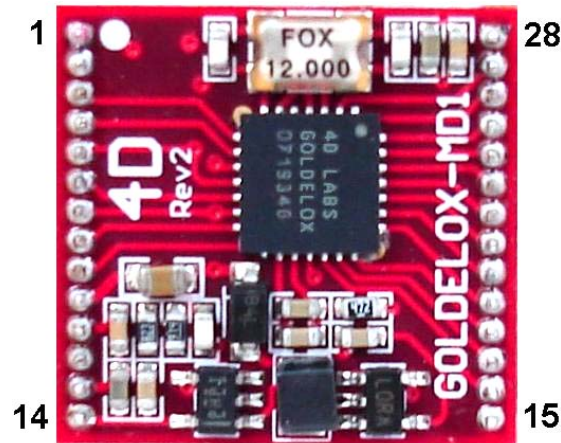
2 Features

The GOLDELOX-MD1 module is aimed at being integrated into a variety of different applications via a wealth of features designed to facilitate any given functionality quickly and cost effectively and thus reduce 'time to market'. These features are as follows:

- Easily interface to any small to medium size monochrome or full colour graphics OLED or LCD panel that has a CPU interface with these signals directly from the module:
 - D0-D7, RD, WR, CS, RESET, D/C
- Directly drives displays with resolutions from 96x64, 128x64, 128x128, 128x160, 160x128 up to 174x220.
- 28 pin module with a tiny footprint measuring only 19.2(L) x 20.3(W) x 5.5(H)mm
- Choose any one of the 4D range of LCD or OLED displays and with a few external components interface it directly to the GOLDELOX-MD1 module on the user application board. Then download via the serial port the specific display Personality Module Micro-Code (**PmmC**) file into the GOLDELOX-MD1 module.
- Voltage supply from 3.6V to 6.0V, current @ 15mA nominal when using a 5.0V supply source.
- Serial RS-232 (0V to 3.3V) with auto-baud feature (300 to 256K baud).
- Onboard micro-SD (**μSD**) memory card adaptor for storing of icons, images, animations, etc. 64Mb to 1Gig **μSD** memory cards can be purchased separately.
- Three selectable font sizes (5x7, 8x8 and 8x12) for ASCII characters as well as user-defined bitmapped characters (32 @ 8x8)
- Built in graphics commands such as: LINE, CIRCLE, RECTANGLE, TEXT, USER BITMAP, BACKGROUND COLOUR, PUT PIXEL, IMAGE, etc. just to name a few



3 Pin Description



PIN	FUNCTION	DESCRIPTION
1	3.3Vout	Regulated 3.3 Volts Supply to the OLED or LCD panel @150mA max
2, 10, 13	GND	GND
23	RESET	Module Hardware Reset. Active Low > 10us
14	VCC	Module Voltage Input. 3.6V to 6.0V range. Nominal 5Volts.
11	VHI OUT	Hi Voltage output for OLED driver or LCD backlight LED voltage supply @14.9 Volts up to 150mA.
12	VHI SET	Connecting a resistor from this pin to VHI OUT will reduce VHI OUT and connecting a resistor to GND will increase VHI OUT. Normally unconnected.
3	CS_SPI	SPI interface. If internal micro-SD memory card is not used the SPI signals can be connected to an external serial flash chip or an external SD card.
5	SCK	
6	SDI	
7	SDO	
8	TX	Serial interface for host processor commands. Also used for uploading PmmC files into the GOLDELOX module from the PC for different displays as well as for future upgrades.
9	RX	
15	D0	OLED-LCD Data Bus.
16	D1	
17	D2	
18	D3	
19	D4	
20	D5	
21	D6	
22	D7	
25	RD	OLED-LCD Read signal, Active Low.
26	WR	OLED-LCD Write signal, Active Low.
27	D/C or RS	OLED-LCD Data/Command signal.
28	DISP_RESET	OLED-LCD Reset signal, Active Low.
4	DISP_CS	OLED-LCD Chip Enable signal, Active Low.
24	RUN	Internal/External memory slide show playback when Active Low on Power-Up. Also runs 4DGL user programs from power-up.

4 Reference Designs

The following reference designs show how easily the GOLDELOX-MD1 module can be interfaced to a wide range of OLED and LCD panels with minimal external circuitry. Please check the following link for more reference designs using different displays as they come to hand.

www.4dsystems.com.au/downloads/Modules/GOLDELOX-MD1/Reference-Designs/

To evaluate the GOLDELOX-MD1 module with the displays shown please refer to the following products:

For OLED-128:

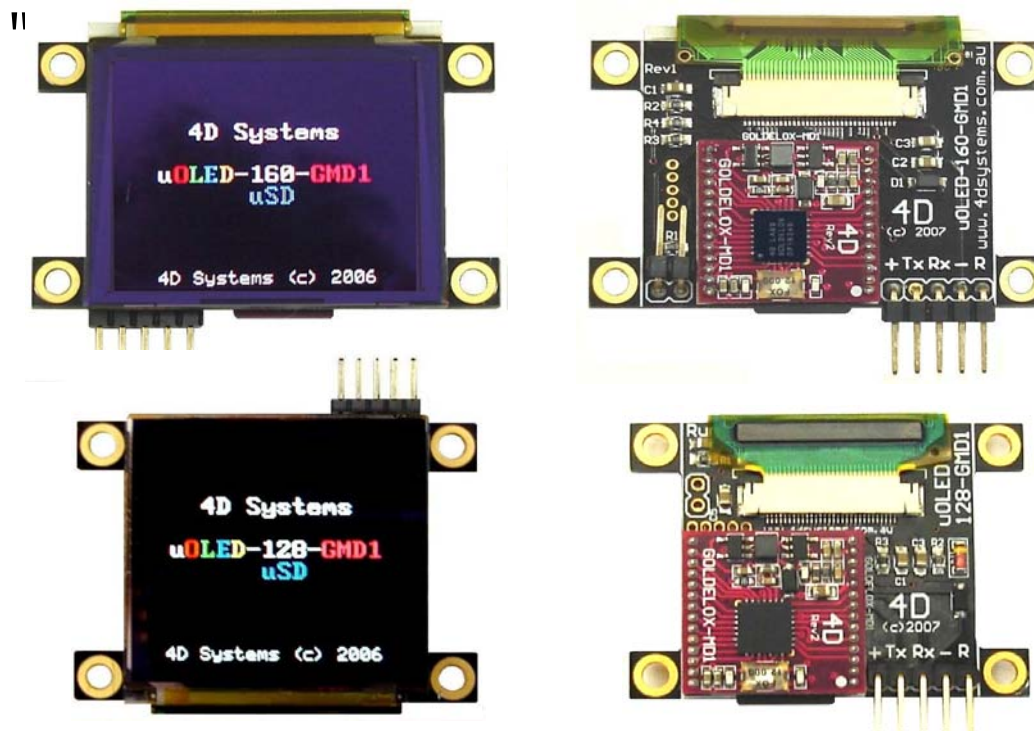
www.4dsystems.com.au/downloads/micro-OLED/uOLED-128-GMD1/Docs/Pdf/

For OLED-160:

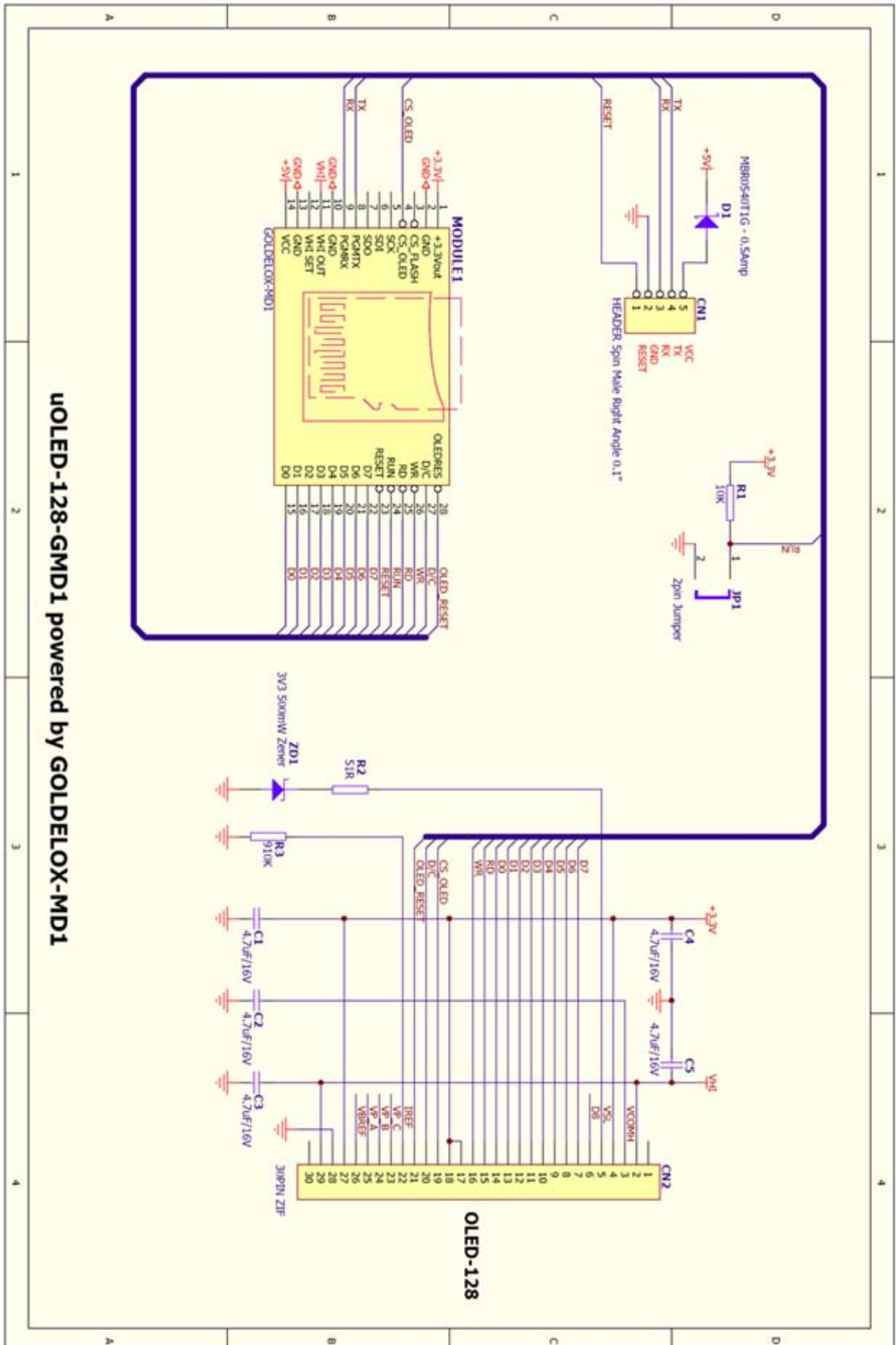
www.4dsystems.com.au/downloads/micro-OLED/uOLED-160-GMD1/Docs/Pdf/

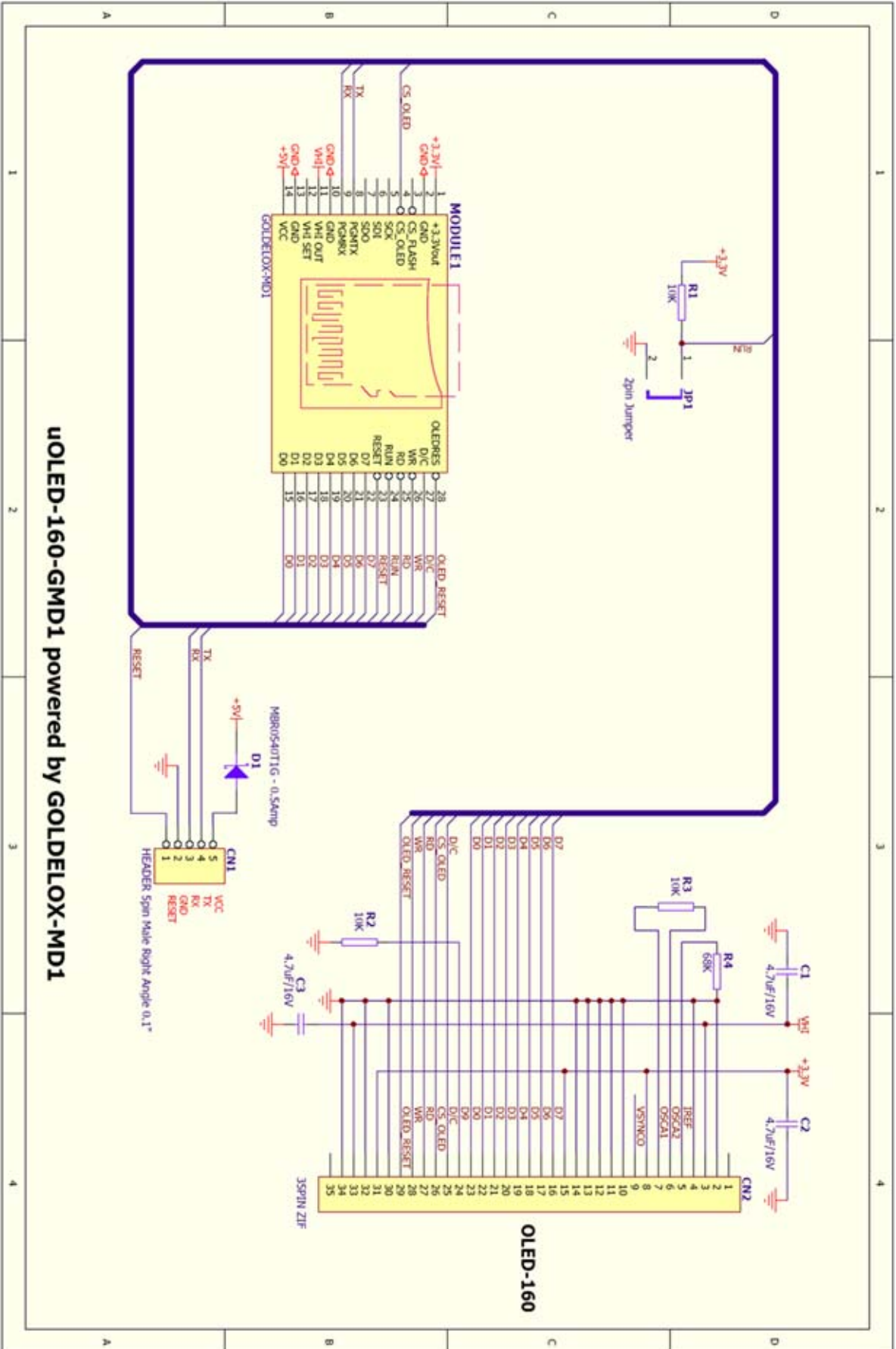
For LCD-128:

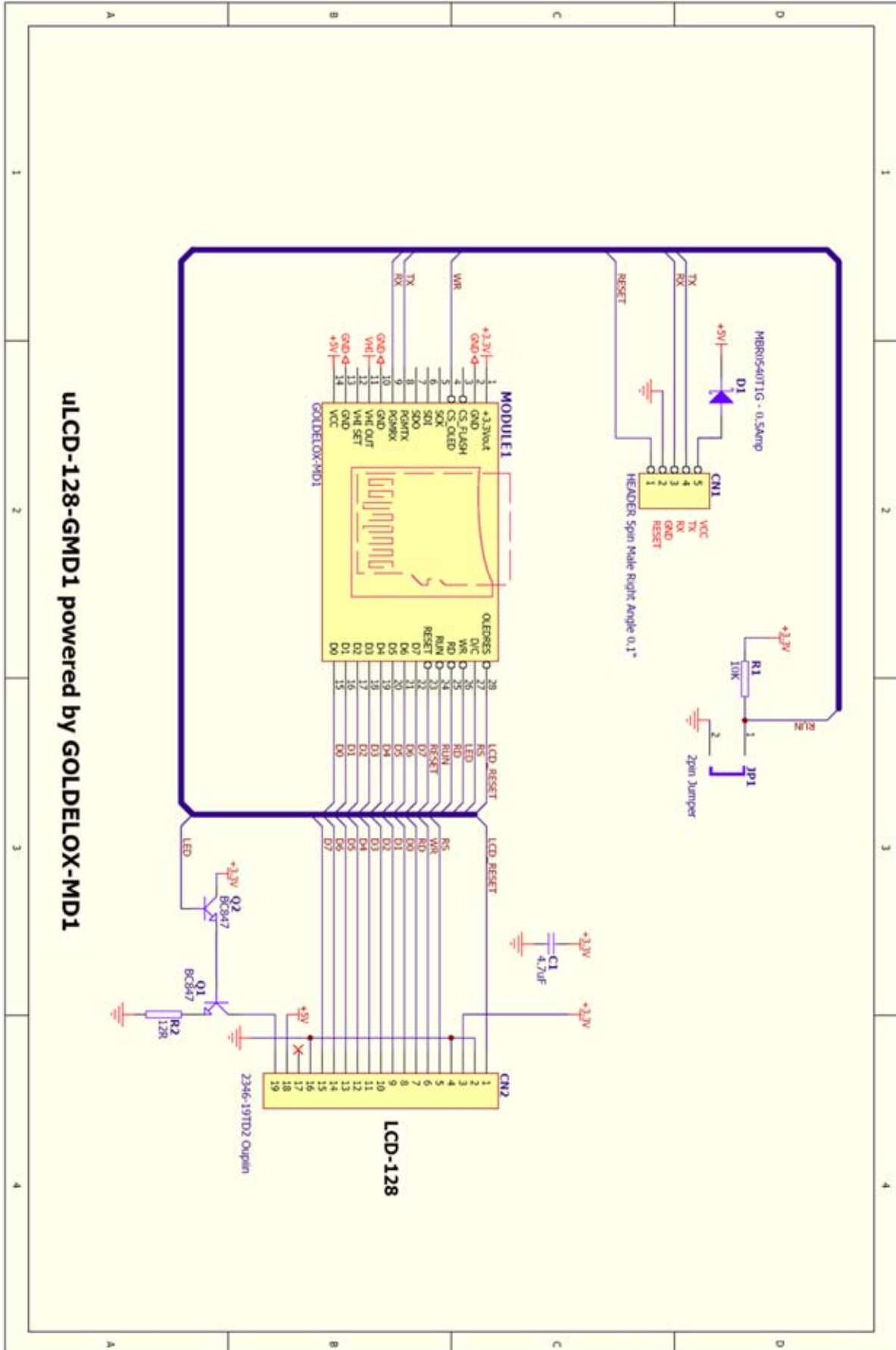
www.4dsystems.com.au/downloads/micro-LCD/uLCD-128-GMD1/Docs/Pdf/



Example Products designed with the GOLDELOX module









5 Personality Module Micro Code (PmmC)

One of the important features of the GOLDELOX module is the ability to upload its onboard GOLDELOX processor with a micro-Code firmware which allows the module to take on a new personality. This is referred to as Personality Module Micro Code (**PmmC**). The benefits of this are as follows:

- Allows the module to be easily upgraded by the user at any time with display specific PmmC file.
- As further enhancements are made in the future, allows the user to benefit from those latest features.
- Allows the GOLDELOX-MD1 module to be used across a wide range of displays, as long as the hardware platform supports the specific display.
- Allows the user to upload a new Operating System to change the device from a serial command driven platform into a high level language platform such as 4DGL. A built-in higher level language such as 4DGL allows user programs to be run directly within the GOLDELOX processor where the graphics operations can be performed much faster than sending the commands serially. This avoids serial bottle necks for those graphics intensive applications as well as allows the user to take complete control of all of the internal resources of the module.
- Allows OEM customers to design their application with their own custom display. 4D Systems will provide a custom **PmmC** file which supports drivers for the display.

6 Graphics Command Set

The heart of the GOLDELOX module is its rich graphics oriented command set. This comprises of powerful instructions that can draw lines, circles, rectangles, text, images, etc. to provide a full graphical user interface. The commands can be either sent via the serial link or called as functions from the built-in library if the platform is running a higher level language such as 4DGL. The command set is divided into 3 groups:

- **General Command Set**
 - These commands are generic and display independent. These are standard commands that apply across all applications.
- **Display Specific Command Set**
 - These commands are display specific. Certain displays may have hardware functions that allow operations to be performed much faster than otherwise possible if they were to be implemented in firmware. These commands utilise those features, if available.
- **Extended Command Set**
 - The GOLDELOX module supports a built in micro-SD (μ SD) memory card. It can also support an external SD or MMC type card. These commands are specific to the memory card related operations. Graphics commands in the form of objects can be stored into the memory card and later recalled under user control.

Each Command set is described in detail in the following sections.

6.1 General Command Set

General Command Set	Serial	4DGL	Object	Memory
(A) Add User Bitmapped Character	✓			
(B) Set Background Colour	✓	✓	✓	✓
(b) Place Text button	✓	✓	✓	✓
(C) Draw Circle	✓	✓	✓	✓
(c) Block copy and Paste (bitmap copy)	✓	✓		
(D) Display User Bitmapped Character	✓	✓		
(E) Erase Screen	✓	✓	✓	✓
(F) Font Size	✓	✓	✓	✓
(G) Draw Triangle	✓	✓	✓	✓
(g) Draw Polygon	✓	✓	✓	✓
(I) Display Image	✓	✓		
(L) Draw Line	✓	✓	✓	✓
(O) Opaque or Transparent Text	✓	✓	✓	✓
(P) Put Pixel	✓	✓		
(p) Set pen Size	✓	✓	✓	✓
(R) Read Pixel	✓	✓		
(r) Draw rectangle	✓	✓	✓	✓
(S) Place String of ASCII Text (unformatted)	✓	✓	✓	✓
(s) Place string of ASCII Text (formatted)	✓	✓	✓	✓
(T) Place Text Character (formatted)	✓	✓	✓	✓
(t) Place text Character (unformatted)	✓	✓	✓	✓
(V) Version/Device Info Request	✓			
(Y) Display Control functions	✓	✓	✓	✓

NOTES:

Serial : Those commands that can be sent via the serial link and executed by the uOLED module.

4DGL : Those commands that can be executed from user application code written in 4DGL via function calls within the built in 4DGL library.

Object : Those commands that can be recalled from the memory card at any time by the host and displayed on the screen using the “Display Object from Memory Card” command.

Memory: Those commands that can reside and be executed from inside the memory card.

6.1.1 Add User Bitmapped Character (A)

Serial Syntax : cmd, char#, data1, data2,, dataN

4DGL Syntax : N/A

cmd : 41hex, Aascii

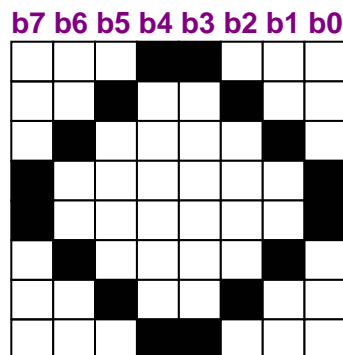
char# : bitmap character number to add to memory:
0 to 31 (00h to 1Fh), 32 characters of 8x8 format.

data1 to dataN : number of data bytes that make up the composition and format of the bitmapped character. The 8x8 bitmap composition is 1 byte wide (8bits) by 8 bytes deep which makes $N = 1 \times 8 = 8$.

Description : This command will add a user defined bitmapped character into the internal memory.

Example1: 41hex, 01hex, 18hex, 24hex, 42hex, 81hex, 81hex, 42hex, 24hex, 18hex

This adds and saves user defined 8x8 bitmap as character number 1 into memory as seen below.



data1 (hex = 18h)
 data2 (hex = 24h)
 data3 (hex = 42h)
 data4 (hex = 81h)
 data5 (hex = 81h)
 data6 (hex = 42h)
 data7 (hex = 24h)
 data8 (hex = 18h)

Example of a 8x8 user defined bitmap



6.1.2 Set Background Colour (B)

Serial Syntax : cmd, colour(msb:lsb)

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 42hex, Bascii

colour(msb:lsb) : pixel colour value: 2 bytes (16 bits) msb:lsb
65,536 colours to choose from
Black = 0000hex, 0dec
White = FFFFhex, 65,535dec, 1111111111111111bin

Description : This command sets the current background colour. Once this command is sent, only the background colour will change. Any other object on the screen with a different colour value will not be affected.

Example : 42hex, FFFFhex

Set the background colour to value 65,535 (white).

6.1.3 Text button (b)

Serial Syntax : cmd, state, x, y, buttonColour(msb:lsb), font, textColour(msb:lsb), textWidth, textHeight, char1, .., charN, terminator

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 62hex, bascii

state : Specifies whether the displayed button is drawn as **UP** (not pressed) or **DOWN** (pressed).
 0 = Button Down (pressed)
 1 = Button Up (not pressed)

x : top left horizontal start position of the button

y : top left vertical start position of the button

buttonColour(msb:lsb) : 2 byte button colour value

font : 0 = 5x7 font, 1 = 8x8 font, 2 = 8x12 font. This has precedence and does not affect the Font command.

textColour(msb:lsb) : 2 byte text colour value

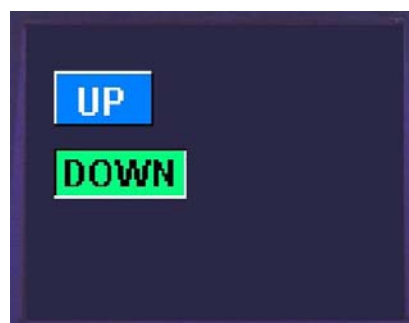
textWidth : horizontal size of the character

textHeight : vertical size of the character

char1..charN : string of ASCII characters (limit the string to line width)

terminator : the string must be terminated with 00hex

Description : This command will place a Text button similar to the ones used in a PC Windows environment. (x, y) refers to the top left corner of the button and the size of the button is automatically calculated and drawn on the screen with the text relatively justified inside the button box. The button can be displayed in an UP (button not pressed) or DOWN (button pressed) position by specifying the appropriate value in the **state** byte. Separate button and text colours provide many variations in appearance and format.



6.1.4 Draw Circle (C)

Serial Syntax : cmd, x, y, rad, colour(msb:lsb)

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 43hex, Cascii

x : circle centre horizontal position. Range depends on display resolution.

y : circle centre vertical position. Range depends on display resolution.

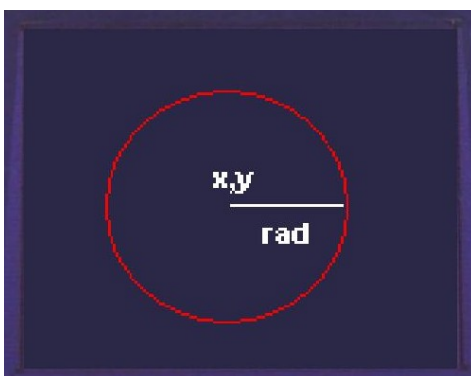
rad : radius size of the circle. Range depends on display resolution.

colour(msb:lsb) : 2 byte circle colour value

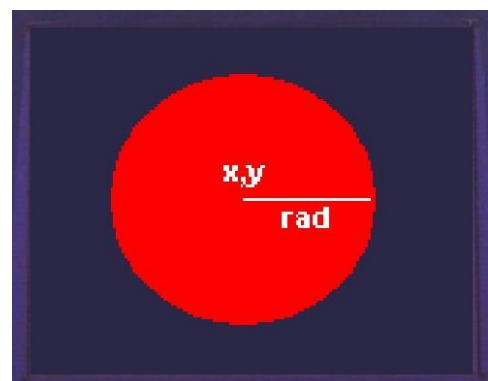
Description : This command will draw a coloured circle centred at (x, y) with a radius determined by the value of rad. The circle can be either solid or wire frame (empty) depending on the value of the Pen Size (see **Set Pen Size** command). When Pen Size = 0 circle is solid, Pen Size = 1 circle is wire frame.

Example : 43hex, 3Fhex, 3Fhex, 22hex, 00hex, 1Fhex

Draws a RED circle (001Fhex) centred at x = 63dec (3Fhex) and y = 63dec (3Fhex) with a radius of 34dec (22hex).



When Pen Size = 1



When Pen Size = 0

6.1.5 Block copy & Paste (Screen Bitmap Copy) (c)

Serial Syntax : cmd, xs, ys, xd, yd, width, height

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 63hex, cascii

xs: top left horizontal start position of block to be copied (source)

ys: top left vertical start position of block to be copied (source)

xd: top left horizontal start position of where copied block is to be pasted (destination)

yd: top left vertical start position of where the copied block is to be pasted (destination)

width: width of block to be copied (source)

height: height of block to be copied (source)

Description : This command copies an area of a bitmap block of specified size. The start location of the block to be copied is represented by **xs, ys** (top left corner) and the size of the area to be copied is represented by **width** and **height** parameters. The start location of where the block is to be pasted (destination) is represented by **xd, yd** (top left corner).

This is a very powerful feature for animating objects, smooth scrolling, implementing a windowing system or copying patterns across the screen to make borders or tiles.

6.1.6 Display User Bitmapped Character (D)

Serial Syntax : cmd, char#, x, y, colour(msb:lsb)

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 44hex, Dascii

char# : which user defined character number to display from the selected group. 0dec to 31dec (00hex to 1Fhex), of 8x8 format.

x : horizontal display position of the character. Range depends on display resolution.

y : vertical display position of the character. Range depends on display resolution.

colour(msb:lsb) : 2 byte bitmap colour value

Description : This command displays the previously defined user bitmapped character at location (x, y) on the screen. User defined bitmaps allow drawing & displaying unlimited graphic patterns quickly & effectively.

Example 1: 44hex, 01hex, 00hex, 00hex, F8hex, 00hex

Display 8x8 bitmap character number 1 at x = 0, y = 0, colour = red

Example 2: 44hex, 01hex, 08hex, 00hex, 07hex, E0hex

Display 8x8 bitmap character number 1 at x = 8, y = 0, colour = green

Example 3: 44hex, 01hex, 10hex, 00hex, 00hex, 1Fhex

Display 8x8 bitmap character number 1 at x = 16, y = 0, colour = blue





GOLDELOX MDI

6.1.7 Erase Screen (E)

Serial Syntax : cmd

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 45hex, Eascii

Description : This command clears the entire screen using the current background colour.

Example : 45hex

Clear the screen.

6.1.8 Set Font Size (F)

Serial Syntax : cmd, size

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 46hex, Fascii

size : = 00hex : 5x7 small size font
= 01hex : 8x8 medium size font
= 02hex : 8x12 large size font

Description : This command will change the size of the font according to the value set by **size**. Changes take place after the command is issued. Any character on the screen with the old font size will remain as it was.

Example1: 46hex, 00hex Select small 5x7 fonts
Example1: 46hex, 01hex Select medium 8x8 fonts
Example1: 46hex, 02hex Select large 8x12 fonts

6.1.9 Draw TrianGle (G)

Serial Syntax : cmd, x1, y1, x2, y2, x3, y3, colour(msb:lsb)

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 47hex, Gascii

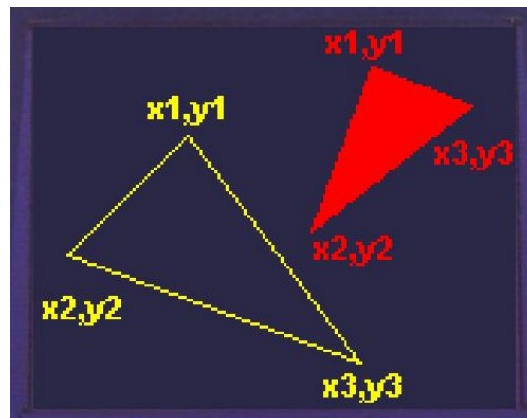
x1, y1, x2, y2, x3, y3 : 3 vertices of the triangle. These must be specified in an anti-clockwise fashion.

colour(msb:lsb) : 2 byte triangle colour value

Description : This command draws a Solid/Empty triangle. The vertices must be specified in an anti-clock wise manner, i.e.

$x2 < x1$, $x3 > x2$, $y2 > y1$, $y3 > y1$.

A solid or a wire frame triangle is determined by the value of the Pen Size setting, i.e. 0 = solid, 1 = wire frame.



6.1.10 Draw Polygon (g)

Serial Syntax : cmd, vertices, x1, y1, xn, yn, colour(msb:lsb)

4DGL Syntax : (refer to 4DGL programmers manual)

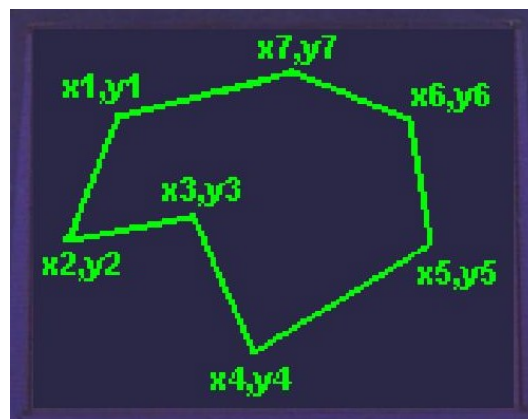
cmd : 67hex, g ascii

vertices : number of vertices from 3 to 7. Specifies the number of vertices of the polygon.

(x1, y1) (xn, yn) : vertices of the polygon. These can be specified in any fashion.

colour(msb:lsb) : 2 byte polygon colour value

Description : This command draws an Empty/Wire Frame polygon. Up to 7 vertices can be specified in any manner. Currently only a wire frame polygon is supported.



6.1.11 Display Image (I)

Serial Syntax : cmd, x, y, width, height, colourMode, pixel1, .. pixelN

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 49hex, lascii

x : Image horizontal start position (top left corner)

y : Image vertical start position (top left corner)

width : horizontal size of the image

height : vertical size of the image

colourMode : 8dec = 256 colour mode, 8bits/1byte per pixel

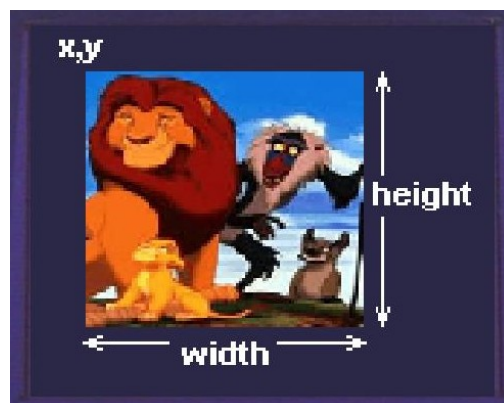
16dec = 65K colour mode, 16bits/2bytes per pixel (msb:lsb)

pixel1..pixelN : image pixel data and N is the total number of pixels

N = height x width when colourMode = 8

N = height x width x 2 when colourMode = 16

Description : This command displays a bitmap image on to the screen with the top left corner specified by (x, y) and size of the image specified by **width** and **height** parameters. This command is more effective than using the "Put Pixel" command, where there are no overheads in specifying the x, y location of each pixel.



6.1.12 Draw Line (L)

Serial Syntax : cmd, x1, y1, x2, y2, colour(msb:lsb)

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 4Chex, Lascii

x1 : horizontal position of line start. Range depends on display resolution.

y1 : vertical position of line start. Range depends on display resolution.

x2 : horizontal position of line end. Range depends on display resolution.

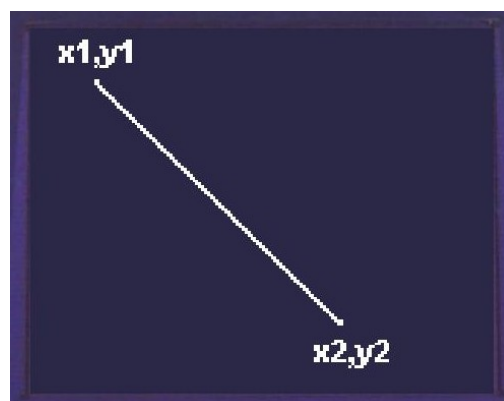
y2 : vertical position of line end. Range depends on display resolution.

colour(msb:lsb) : 2 byte line colour value

Description : This command will draw a coloured line from point (x1, y1) to point (x2, y2) on the screen.

Example : 4Chex, 00hex, 00hex, 7Fhex, 7Fhex, FFhex, FFhex

Draws a white line from (x1=0, y1=0) to (x2=127, y2=127).



6.1.13 Opaque / Transparent Text (O)

Serial Syntax : cmd, mode

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 4Fhex, Oascii

mode : = 00hex : Transparent Text, objects behind the text can be seen.

= 01hex: Opaque Text, objects behind text is blocked by background

Description : This command will change the attribute of the text so that an object behind the text can either be blocked or transparent. Changes take place after the command is sent.

This command will change the attribute so that when a character is written, it will either write just the character alone (Transparent Mode) so any original character will be seen as well as the new, or overwrite any existing data with the new character.

Example1: 4Fhex, 00hex Transparent Text Mode

Example2: 4Fhex, 01hex Opaque Text Mode



6.1.14 Put Pixel (P)

Serial Syntax : cmd, x, y, colour(msb:lsb)

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 50hex, Pascii

x : horizontal pixel position. Range depends on display resolution.

y : vertical pixel position. Range depends on display resolution.

colour : pixel colour value: 2 bytes (16 bits) msb, lsb
65,536 colours to choose from

Black = 0000hex, 0dec

White = FFFFhex, 65,535dec, 1111111111111111bin

Description : This command will put a coloured pixel at location (x, y) on the screen.

Example : 50hex, 01hex, 0Ahex, FFhex, FFhex

Puts a white (FFFFhex) pixel at location x = 01dec (01hex) and y = 10dec (0Ahex).





6.1.15 Set pen Size (p)

Serial Syntax : cmd, size

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 70hex, p ascii

size : = 00hex : All objects such as circles, rectangles, triangles, etc are solid
= 01hex : All objects are wire frame (empty)

Description : This command determines if certain graphics objects are drawn in solid or wire frame fashion.

Example1: 70hex, 00hex All objects will be drawn solid

Example1: 70hex, 01hex All objects will be drawn wire frame.



6.1.16 Read Pixel (R)

Serial Syntax : cmd, x, y

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 52hex, Rascii

x : horizontal pixel position. Range depends on display resolution.

y : vertical pixel position. Range depends on display resolution.

Description : This command will read the colour value of pixel at location (x, y) on the screen and return it to the host. This is a useful command when for example a white pointer is moved across the screen and the host can read the colour on the screen and switch the colour of the pointer when it's on top of a light coloured area.

Example : 52hex, 01hex, 01hex
GOLDELOX reply : 00hex, 1Fhex

Reads a blue (001Fhex) pixel at location x = 1dec (01hex) and y = 1dec (01hex).

6.1.17 Draw rectangle (r)

Serial Syntax : cmd, x1, y1, x2, y2, colour(msb:lsb)

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 72hex, r ascii

x1 : top left horizontal start position. Range depends on display resolution.

y1 : top left vertical start position. Range depends on display resolution.

x2 : bottom right horizontal end position. Range depends on display resolution.

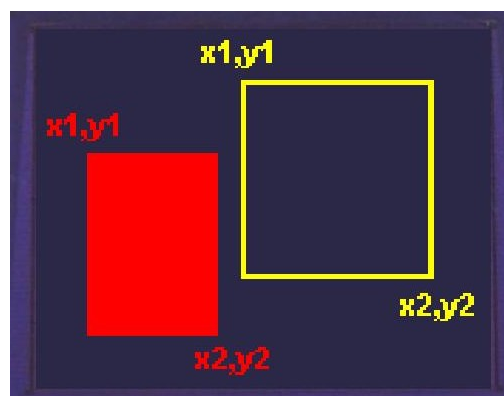
y2 : bottom right vertical end position. Range depends on display resolution.

colour(msb:lsb) : 2 byte rectangle colour value

Description : This command will draw a rectangle of specified area on the screen. **x1, y1** refers to the top left corner of the area and **x2, y2** refers to the bottom right hand corner of the rectangle on the screen. If colour is chosen to be that of the background then the effect will be erasure. If Pen Size value was previously set to 0 rectangle will be solid, otherwise wire frame if value was 1.

Example : 70hex, 00hex, 00hex, 10hex, 10hex, 00hex, 1Fhex

Draws a RED (001Fhex) rectangle that has its top left corner at x1=0, y1=0 and its bottom right corner at x2=16, y2=16.



6.1.18 Place **S**tring of Ascii Text (unformatted) (**S**)

Serial Syntax : `cmd, x, y, font, colour(msb:lsb), width, height, char1, .. , charN, terminator`

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 53hex, Sascii

x : the horizontal start position of string (in pixels).

y : the vertical start position of string (in pixels).

font : 0 = 5x7 font, 1 = 8x8 font, 2 = 8x12 font. This has precedence over the Font command but does not effect the previous font selection.

colour(msb:lsb) : 2 byte colour value of the string.

width : horizontal size of the string characters, n x normal size

height : vertical size of the string characters, m x normal size

char1..charN : string of ASCII characters (max 256 characters)

terminator : the string must be terminated with 00hex

Description : This command allows the display of a string of bitmapped (unformatted) ASCII characters. The horizontal start position of the string is specified by **x** and the vertical position is specified by **y**. The string must be **terminated** with 00hex. The sizes of the characters are determined by the **width** and **height** parameters. If the length of the string is longer than the maximum number of characters per line, then a wrap around will occur on to the next line. Maximum string length is **256 bytes**.

6.1.19 Place string of Ascii Text (formatted) (s)

Serial Syntax : cmd, column, row, font, colour(msb:lsb), char1,.., charN, terminator

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 73hex, sascii

column : string horizontal start position. Range depends on display resolution.

row : string vertical start position. Range depends on display resolution.

font : 0 = 5x7 font, 1 = 8x8 font, 2 = 8x12 font. This has precedence over the Font command.

colour(msb:lsb) : 2 byte colour value of the string.

char1..charN : string of ASCII characters (max 256 characters)

terminator : the string must be terminated with 00hex

Description : This command allows the display of a string of ASCII characters. The horizontal start position of the string is specified by **column** and the vertical position is specified by **row**. The string must be **terminated** with 00hex. If the length of the string is longer than the maximum number of characters per line, then a wrap around will occur on to the next line. Maximum string length is **256 bytes**.

6.1.20 Place Text Character (formatted) (T)

Serial Syntax : cmd, char, column, row, colour(msb:lsb)

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 54hex, Tascii

char : inbuilt standard ASCII character, 32dec to 127dec (20hex to 7Fhex)

column : horizontal position of character. Range depends on display resolution.

row : vertical position of character. Range depends on display resolution.

colour(msb:lsb) : 2 byte colour value of the character.

Description : This command will place a coloured ASCII character (from the ASCII chart) on the screen at a location specified by (**column, row**). The position of the character on the screen is determined by the predefined horizontal and vertical positions available.

Example : 54hex, 41hex, 00hex, 00hex, FFhex, FFhex

Place character 'A' (41hex) at column = 0, row = 0, colour = white (65,535).



6.1.21 Place text Character (unformatted) (t)

Serial Syntax : cmd, char, x, y, colour(msb:lsb), width, height

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 74hex, tascii

char : inbuilt standard ASCII character, 32dec to 127dec (20hex to 7Fhex)

x : the horizontal position of character (in pixel units).

y : the vertical position of character (in pixel units).

colour(msb:lsb) : 2 byte colour value of the character.

width : horizontal size of the character, n x normal size

height : vertical size of the character, m x normal size

Description : This command will place a coloured built in ASCII character anywhere on the screen at a location specified by (x, y). Unlike the 'T' command, this option allows text of any size (determined by width and height) to be placed at any position. The font of the character is determined by the 'Font Size' command.

6.1.22 Display Control Functions (Y)

Serial Syntax : cmd, mode, value

4DGL Syntax : (refer to 4DGL programmers manual)

cmd : 59hex, Yascii

mode : = 00hex : **BACKLIGHT CONTROL.**

value = XXhex: has no effect as there is no backlighting on the OLED display. This is only retained for legacy.

mode : = 01hex : **DISPLAY ON/OFF.**

value = 00hex: Display OFF
= 01hex: Display ON

mode : = 02hex : **DISPLAY CONTRAST.**

value = 0dec to 15dec : Contrast range (default = 15dec)

mode : = 03hex : **DISPLAY POWER-UP/POWER-DOWN.**

value = 00hex: Display Power-Down
= 01hex: Display Power-Up

Note: It is important that the display be issued with the Power-Down command before switching off the power. This command switches off the internal voltage boosters and current amplifiers and they need to be turned off before main power is removed. If the power is removed without issuing this command, the display maybe damaged (over a period of time). This command also turns off the display. This command need not only be issued to shutdown but can be issued to conserve power by turning off the display and the backlight. The Power-Up command does not need to be executed when applying power. If a Power-Down command has been issued and Power is not switched off, the Power-Up command can be sent to Power the display back up again.



6.1.23 Version/Device Info Request (V)

Serial Syntax : cmd, output

Response : device_type, hardware_rev, firmware_rev, horizontal_res, vertical_res

4DGL Syntax : N/A

cmd : 56hex, Vascii

output : 00hex : outputs the version and device info to the serial port only.
01hex : outputs the version and device info to the serial port as well as to the screen.

device_type : this response indicates the device type.

00hex = micro-OLED.

01hex = micro-LCD.

02hex = micro-VGA.

hardware_rev : this response indicates the device hardware version.

firmware_rev : this response indicates the device firmware version.

horizontal_res : this response indicates the horizontal resolution of the display.

22hex : 220 pixels

28hex : 128 pixels

60hex : 160 pixels

64hex : 64 pixels

76hex : 176 pixels

96hex : 96 pixels

vertical_res : this response indicates the vertical resolution of the display. See horizontal_res above for resolution options.

Description : This command requests all the necessary information from the module about its characteristics and capability.

6.2 Display Specific Command Set

Different display panels have certain built in features that are controlled directly by the embedded display driver. These features otherwise would be too cumbersome to implement in firmware and would require resources that are not available. The Display Specific Command set utilises these built in hardware features directly. These are detailed in this section.

Display Specific Command Set	Serial	4DGL	Object	Memory
(\$W) Write to OLED Register	✓	✓		
(\$S) Display Scroll Control	✓	✓		✓
(\$D) Dim Screen Area	✓			✓

6.2.1 Write to OLED-128 Register (**\$W**)

Serial Syntax : spCmd, cmd, reg_data, mode

4DGL Syntax : (refer to 4DGL programmers manual)

spCmd : 24hex, \$ascii

cmd : 57hex, Wascii

reg_data : This byte is either OLED-128 internal register address or data for the register once the register has been selected. Refer to mode parameter.

mode : 0x00 : reg_data is OLED-128 internal register address
0x01 : reg_data is data for the selected register

Description : This command allows direct access to all of the SSD1339 driver registers. For more detail, refer to the SSD1339 driver data sheet available from:

www.4dsystems.com.au/micro-OLED/OLED-128/data/SSD1339.pdf

6.2.2 Write to OLED-160 Register (**\$W**)

Syntax : spCmd, cmd, reg, data

4DGL Syntax : (refer to 4DGL programmers manual)

spCmd : 24hex, \$ascii

cmd : 57hex, Wascii

reg : OLED-160 internal register address. Refer to SEPS525 OLED driver data sheet for all register information.

data : OLED-160 internal register data.

Description : This command allows direct access to all of the SEP525 driver registers. For more detail, refer to the SEP525 driver data sheet available from:

www.4dsystems.com.au/micro-OLED/OLED-160/data/SEPS525.pdf



6.2.3 Display Scroll Control (\$S)

Serial Syntax : spCmd, cmd, register, data

4DGL Syntax : (refer to 4DGL programmers manual)

spCmd : 24hex, \$ascii

cmd : 57hex, Wascii

reg : Scroll Control Register.

<u>register</u>	<u>data</u>
0x00 (Scroll Enable/Disable)	0 = Disable, 1 = Enable
0x01 Reserved	XXX
0x02 (Scroll Speed)	1 = fast, 2 = normal, 3 = slow

data : Scroll register data. Refer to above for detail.

Description : This command allows control of screen scrolling.



6.2.4 Dim Screen Area OLED-128 only (**\$D**)

Serial Syntax : spCmd, cmd, x, y, width, height

4DGL Syntax : N/A

spCmd : 24hex, \$ascii

cmd : 44hex, Dascii

x : horizontal start position of screen area to dim (top left corner)

y : vertical start position of screen area to dim (top left corner)

width : horizontal size of the area to dim

height : vertical size of the area to dim

Description : This command allows a portion of the screen to be dimmed to achieve certain effects such as highlight control, etc.



6.3 Extended Command Set

The following commands are related to the **GOLDELOX-MD1** extended command set and they are described in this section. The module supports an integrated micro-SD (**μSD**) memory card adaptor. It can also support external SD or MMC memory cards. The GOLDELOX-MD1 module can accept memory cards of any size from 64Mb up to 1Gig for storing of text, images, icons, animations, movie clips and all other graphics objects. To utilise the Extended Command set, a **μSD** memory card must be inserted into the module since all of these commands are based around the memory card.

You will find references being made to “**Objects**” throughout this section. An object can be simply defined as those commands that reside inside the memory card (programmed/downloaded previously) and can be displayed on the screen by the “**Display Object from Memory Card**” command. The idea of programming objects into the memory card is so that they can be automatically played back like a slide show without any host processor intervention.

There are also some commands that can only reside inside the card and must be executed from there. These commands will return a NAK if executed live from the serial link.

Extended Command Set	Serial	4DGL	Object	Memory
(@i) initialise uSD Memory Card	✓	✓		
(@R) Read Sector	✓	✓		
(@W) Write Sector	✓	✓		
(@r) read Byte	✓	✓		
(@w) write Byte	✓	✓		
(@A) Set Address	✓	✓		
(@C) Copy Screen to Memory Card	✓	✓		
(@I) Display Image/Icon from Memory Card	✓	✓	✓	✓
(@O) Display Object from Memory Card	✓	✓		
(@P) Run Program from Memory Card	✓			
(07hex) Delay (in milliseconds)				✓
(08hex) Set Counter				✓
(09hex) Decrement Counter				✓
(0Ahex) Jump to Address if Counter not Zero				✓
(0Bhex) Jump to Address				✓
(0Chex) Exit Program from Memory Card	✓			✓

NOTES:

Serial : Those commands that can be sent via the serial link and executed by the uOLED module.

4DGL : Those commands that can be executed from user application code written in 4DGL via function calls within the built in 4DGL library.

Object : Those commands that can be recalled from the memory card at any time by the host and displayed on the screen using the “Display Object from Memory Card” command.

Memory: Those commands that can reside and be executed from inside the memory card.



GOLDELOX MDI

6.3.1 initialise Memory Card (@i)

Serial Syntax : extCmd, cmd

4DGL Syntax : (refer to 4DGL programmers manual)

extCmd : 40hex, @ascii

cmd : 69hex, i ascii

Description : This command initialises the μ SD memory card. The memory card is always initialised upon Power-Up or Reset cycle, if the card is present. If the card is inserted after the power up or a reset then this command must be used to initialise the card.



GOLDELOX MDI

6.3.2 Read Sector Data from Memory Card (@R)

Serial Syntax : extCmd, cmd, SectorAddress(hi:mid:lo)

4DGL Syntax : (refer to 4DGL programmers manual)

extCmd : 40hex, @ascii

cmd : 52hex, Rascii

SectorAddress(hi:mid:lo): A 3 byte sector address. Sector Address range from 0 to 16,777,215 depending on the capacity of the card. Each sector is 512 bytes in size. There are 2048 sectors per every 1Mb of card memory.

Description : This command provides a means of reading data back from the memory card in lengths of 512 bytes. It maybe useful in validating the data that was stored previously using the Write Sector command. Once this command is sent, GOLDELOX will return 512 bytes of data relating to that particular sector.

6.3.3 Write Sector Data to Memory Card (@W)

Serial Syntax : `extCmd, cmd, SectorAddress(hi:mid:lo), data(1), .. , data(512)`

4DGL Syntax : (refer to 4DGL programmers manual)

extCmd : `40hex, @ascii`

cmd : `57hex, Wascii`

SectorAddress(hi:mid:lo): A 3 byte sector address. Sector Address range from 0 to 16,777,215 depending on the capacity of the card. Each sector is 512 bytes in size. There are 2048 sectors per every 1Mb of card memory.

data(1), .. , data(512): 512 bytes of sector data. The data length must be 512 bytes long. Unused bytes must be padded even if not all are used.

Description : This command allows downloading of objects such as images and other commands for storage that can be retrieved and used later on. It can also be used as general purpose storage for user specific data. Downloads must always be limited to 512 bytes in length. For large objects such as images, the data must be broken up into multiple sectors (chunks of 512 bytes) and this command then maybe used many times until all of the data is written into the card. If the data block to be written is less than 512 bytes in length, then make sure the rest of the remaining data are padded with 00hex or FFhex (it can be anything).

If only few bytes of data are to be written then the **Write Byte** command can be used.

Once this command is issued, the GOLDELOX will take a few milliseconds to write the data into its memory card. If the Serial Command platform is used it will reply back with an **ACK**(06hex) if the write cycle was successful. If there was a problem in writing the data to the card a **NAK**(15hex) will be sent back without any write attempts.

Only **data(1)** to **data(512)** are stored in the card. Other bytes in the command message such as Sector Address are not stored.



GOLDELOX MDI

6.3.4 read Byte Data from Memory Card (@r)

Serial Syntax : extCmd, cmd

4DGL Syntax : (refer to 4DGL programmers manual)

extCmd : 40hex, @ascii

cmd : 72hex, r ascii

Description : This command provides a means of reading a single byte of data back from the memory card. Before this command can be used the card memory address location must be set using the **Set Memory Address** command. Once this command is issued, the GOLDELOX will return 1 byte of data relating to that memory location set by the memory Address pointer. The memory Address location pointer is automatically incremented to the next address location.



6.3.5 write Byte Data to Memory Card (@w)

Serial Syntax : extCmd, cmd, data

4DGL Syntax : (refer to 4DGL programmers manual)

extCmd : 40hex, @ascii

cmd : 77hex, w ascii

data : 1 byte of memory card data.

Description : This command allows writing single bytes of data to the memory card. This is useful for writing small chunks of data relating to graphics objects or user application specific data for general purpose storage. For large data blocks it is more efficient to use the **Write Sector Data** command described in the previous section.

Before this command can be used the card memory address location must be set using the **Set Memory Address** command. Once this command is issued, the GOLDELOX will write 1 byte of data relating to that memory location set by the memory Address pointer. The memory Address location pointer is automatically incremented to the next address location.

Only the **data** byte is stored in the card. Other bytes in the command message are not stored.



6.3.6 Set Memory Address (@A)

Serial Syntax : extCmd, cmd, Address(Umsb:Ulsb:Lmsb:Llsb)

4DGL Syntax : (refer to 4DGL programmers manual)

extCmd : 40hex, @ascii

cmd : 41hex, Aascii

Address(Umsb:Ulsb:Lmsb:Llsb): A 4 byte memory card address for byte wise access.

Description : This command sets the card memory Address pointer for byte wise reads and writes. After a byte read or write the Address pointer is automatically incremented internally to the next Address location.

6.3.7 Copy Screen to Memory Card (@C)

Serial Syntax : `extCmd, cmd, x, y, width, height, SectorAddress(hi:mid:lo)`

4DGL Syntax : (refer to 4DGL programmers manual)

extCmd : 40hex, @ascii

cmd : 43hex, Cascii

x : Screen horizontal start position (top left corner)

y : Screen vertical start position (top left corner)

width : horizontal size of the screen area to be copied

height : vertical size of the screen area to be copied

SectorAddress(hi:mid:lo): A 3 byte sector address where the copied screen area is to be stored.

Description :

This command copies an area of the screen of specified size. The start location of the block to be copied is represented by **x, y** (top left corner) and the size of the area to be copied is represented by **width** and **height** parameters. This is similar the **Block Copy and Paste** command but instead of the copied screen area being pasted to another location on the screen it is stored into the memory card. The stored screen image can then be later recalled from the memory card and redisplayed onto the screen at the same or different location by using the **Display Image/Icon from Memory Card** command.

This is a very powerful feature for animating objects, smooth scrolling, or implementing a windowing system.

6.3.8 Display Image/Icon from Memory Card (@I)

Serial Syntax : extCmd, cmd, x, y, width, height, colourMode, SectorAddress(hi:mid:lo)

4DGL Syntax : (refer to 4DGL programmers manual)

extCmd : 40hex, @ascii

cmd : 49hex, lascii

x : Screen horizontal start position (top left corner)

y : Screen vertical start position (top left corner)

width : horizontal size of the Image/Icon

height : vertical size of the Image/Icon

colourMode : 8dec = 256 colour mode, 8bits/1byte per pixel
16dec = 65K colour mode, 16bits/2bytes per pixel

SectorAddress(hi:mid:lo): A 3 byte memory card sector address of a previously stored Image or an Icon that is about to be displayed.

Description : This command displays a bitmap image or an icon on to the screen that has been previously stored at a particular sector address in the memory card. The screen position of the image to be displayed is specified by (x, y) and the size of the image by width and height parameters.

If the previously stored image was in 8 bit colour format (1 byte per pixel) or 16 bits (2 bytes per pixel) then this must be specified in the colourMode byte parameter. Do not store an image/icon in one colour format then display it in another colour format, this will result in a corrupted image display.

Notes:

- The **Copy Screen to Memory Card** command always stores that part of the screen as a 16 bit image, i.e. 2 bytes per pixel.
- The images or icons when stored into the memory card must be sector boundary aligned, i.e. the object start location must be at the start of a sector boundary.



6.3.9 Display Object from Memory Card (@O)

Syntax : extCmd, cmd, Address(Umsb:Ulsb:Lmsb:Llsb)

4DGL Syntax : (refer to 4DGL programmers manual)

extCmd : 40hex, @ascii

cmd : 4Fhex, Oascii

Address(Umsb:Ulsb:Lmsb:Llsb): A 4 byte (32 bit) memory address of a previously stored Object that is about to be displayed.

Description: Some of the commands can be stored as objects in the memory card which can be later recalled by the host on demand and displayed or executed. The user must make sure the 32 bit address of each stored command/object is known before using this feature. For example, a series of images can be stored as icons and later displayed as the application requires them. The table at the end of this section lists all of the commands that can be stored as objects within the memory card.



6.3.10 Run Program from Memory Card (@P)

Serial Syntax : extCmd, cmd, Address(Umsb:Ulsb:Lmsb:Llsb)

4DGL Syntax : N/A

extCmd : 40hex, @ascii

cmd : 50hex, Pascii

Address(Umsb:Ulsb:Lmsb:Llsb): A 4 byte memory card address for the internal command execution.

Description : The Run command forces the 32bit internal memory pointer to jump to the specified address and automatically start executing commands, from the memory card without any further interaction by the host processor. It will sequentially execute any valid memory related commands and display objects until it gets to the end of the memory. It is advisable to have the **Exit Program** or the **Jump to Address** commands at the end of the user composed program so that the pointer does not run off so to speak.



6.3.11 Delay (**07hex**) (memory card command only)

Syntax : cmd, value(msb:lsb)

4DGL Syntax : N/A

cmd : 07hex

value(msb:lsb) : A 2 byte delay value in milliseconds. Maximum value of 65,535 milliseconds or 65.5 seconds.

Description : When objects from the memory card such as images are displayed sequentially, a delay can be inserted between subsequent objects. A delay basically has the same effect as a NOP (No Operation) which can be used to determine how long the object stays on the screen before the next object is displayed.

6.3.12 Set Counter (08hex) (memory card command only)

Syntax : cmd, value

4DGL Syntax : N/A

cmd : 08hex

value : A 1 byte counter value that can be used with **Decrement Counter** and **Jump to Address If Counter Not Zero** commands to form loops. Practical values should be between 2 and 255.

Description : A series of images that might be part of an animation may need to be redisplayed over and over to achieve a lengthy viewing. This command when used in conjunction with **Decrement Counter** and **Jump to Address If Counter Not Zero** commands allow the user to determine exactly how many times the series of images are looped.

For example, we may want to animate the Globe rotating. Let's say we have 10 image slides of the Globe at different rotated positions residing in the memory card. When the images are displayed sequentially, the effective duration will only be the length of time it takes to display the 10 image frames. We can increase that length by looping through the animation a number of times depending on the value set in the counter. When the display reaches the end of the last frame and encounters the **Decrement Counter** followed by **Jump to Address If Counter Not Zero** commands, the counter will be decremented and then the internal pointer will jump to the memory Address specified in the "Jump to Address If Counter Not Zero" command. This sequence will repeat until the value in the counter reaches zero. The following demonstrates how this maybe used:

<u>Address (dec)</u>	<u>Command</u>
00000000	Set Counter (value = 25),
00000002	Display Image from Memory Card (image1),
00000012	Delay(10ms),
00000015	Display Image from Memory Card (image2),
00000025	Delay(10ms),
	...
00000119	Display Image from Memory Card (image10),
00000129	Delay(10ms),
00000132	Decrement Counter
00000134	Jump to Address if Counter Not Zero (Address = 00000002)

Note : The above example is typical of how a series of commands might be loaded into the memory card and then executed by using the Run Program from Memory Card command. The commands would ofcourse be the series of hex codes.





GOLDELOX MDI

6.3.13 Decrement Counter (**09hex**) (memory card command only)

Syntax : cmd, value

4DGL Syntax : N/A

cmd : 08hex

Description : Decrements the counter. See detailed description on how this command can be used effectively in the **Set Counter** command section.



6.3.14 Jump to Address If Counter Not Zero (**0Ahex**) (memory card command only)

Syntax : cmd, Address(Umsb:Ulsb:Lmsb:Llsb)

4DGL Syntax : N/A

cmd : 0Ahex

Address(Umsb:Ulsb:Lmsb:Llsb): A 4 byte (32 bit) memory jump address if the counter is not zero.

Description : If the internal counter is not zero the program pointer will jump to the specified address. If the counter is zero then it will continue executing the next command. Please see detailed description on how this command can be used effectively in the **Set Counter** command section.



GOLDELOX MDI

6.3.15 Jump to Address (**0Bhex**) (memory card command only)

Syntax : cmd, Address(Umsb:Ulsb:Lmsb:Llsb)

4DGL Syntax : N/A

cmd : 0Bhex

Address(Umsb:Ulsb:Lmsb:Llsb): A 4 byte (32 bit) memory jump address.

Description : This command will force the internal 32 bit program memory pointer to jump unconditionally to the specified address and start executing commands from there.

6.3.16 Exit Program from Memory Card (**0Chex**)

Syntax : cmd

4DGL Syntax : N/A

cmd : 0Chex

Description : This command forces the program to stop executing from the memory card and ready to accept and execute commands from the host via the serial interface. When the internal program memory pointer encounters this command it will force the command execution from memory card to stop. It can also be sent via the serial port while the program is running and commands are being executed from the memory card.

Summary of Commands Executable from Memory Card

Command	Memory Card
Set B ackground Colour	✓
Place Text b utton	✓
Draw C ircle	✓
E rase Screen	✓
F ont Size	✓
Draw Trian G le	✓
Draw L ine	✓
O paque or Transparent Text	✓
Set p en Size	✓
Draw r ectangle	✓
Place S tring of ASCII Text (unformatted)	✓
Place s tring of ASCII Text (formatted)	✓
Place T ext Character (formatted)	✓
Place t ext Character (unformatted)	✓
OLED Displa Y Control functions	✓
Scroll Control	✓
Display I mage/ I con from Memory Card	✓
Delay (in milliseconds)	✓
Set Counter	✓
Decrement Counter	✓
Jump to Address if Counter not Zero	✓
Jump to Address	✓
Exit Program from Memory Card	✓

7 Serial Interface (TTL Level)

The GOLDELOX-MD1 module communicates with the outside world via its serial link. The serial link is always used as a means to upload a **PmmC** file for a new platform change or to update the module with enhancements. If the module is running the “**Serial Command**” platform, it can be used by a host processor to send serial commands to the module so that characters and graphics can be displayed on the screen. If the module is running the “**4DGL High Level Language**” platform then user application code is downloaded into the GOLDELOX processor via the same serial link. The serial link is also used to upload the onboard micro-SD memory card with graphics objects such as images as well as slide shows with animations and movie clips using the Graphics Composer PC tool.

Please note that the serial connection (RX/TX) is at TTL levels (0 – 3.3V) and the logic levels are “high” = 1 = 3.3V, “low” = 0 = 0V. If interfacing to a host system running at voltage levels greater than 3.6V, then a 1K series resistor must be inserted between the Host TX and the GOLDELOX RX signal.

Auto Baud Detect:

The GOLDELOX-MD1 has an auto-baud detect function which can operate from **300 baud to 256K baud**. Prior to any graphical formatting and commands being sent, it must first be initialized by sending the ASCII character ‘**U**’ (**55h**) after power-up. This will allow the module to determine and lock on to the baud rate of the host automatically without needing any further setup. This must be done every time the module is powered up or reset.

If the module needs its baud rate changed, then it must be powered down and powered back up again or go through a reset. The “U” command cannot be used to change the baud rate during the middle of normal usage.

Serial Command Timing:

Each serial command is made up of a sequence of data bytes. Some commands are a single byte and others are multiple bytes. When a command is sent to the module and the operation is completed, it will reply back with a single acknowledge byte called the **ACK** (06h). This tells the host that the command was understood and the operation is completed. It will take anywhere between 1 to several milliseconds to reply back with an **ACK**, depending on the command and the operation the GOLDELOX-MD1 has to perform. If it receives a command that it does not understand it will reply back with a negative acknowledge called the **NAK** (15h).



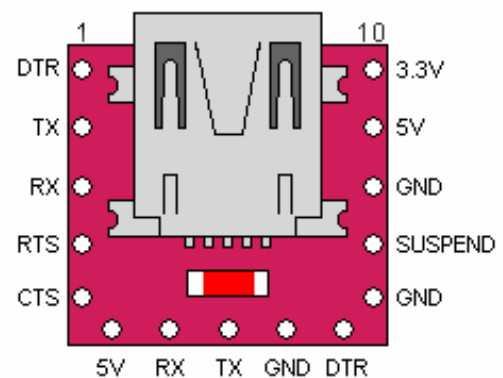
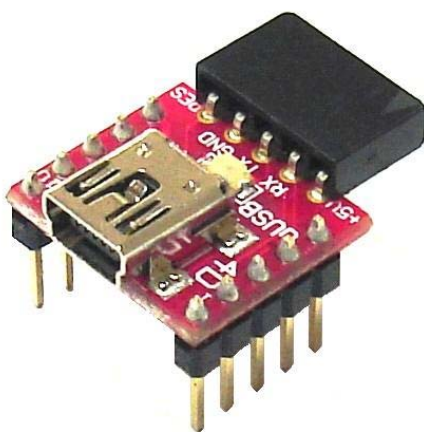
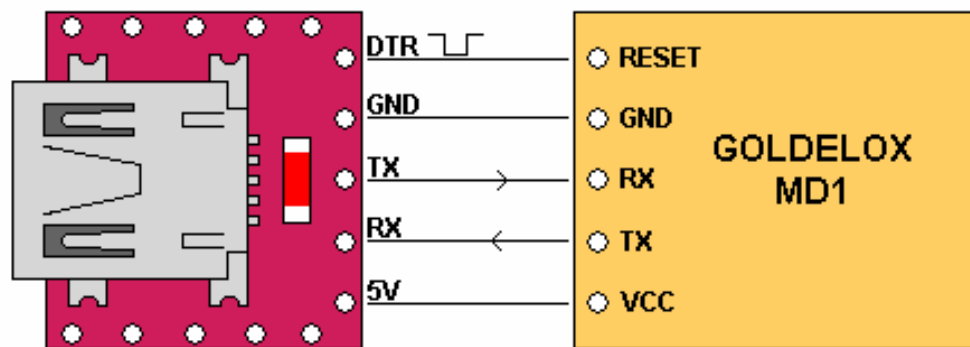
GOLDELOX MDI

For example, if a command has 5 bytes but only 4 bytes are sent, the command will not be executed and when the next following command bytes are sent the module will reply back with a **NAK** for each and every byte it receives. For correct operation make sure the command bytes are sent in the correct sequence.

Note: No termination character is to be sent at the end of the command sequence. i.e. don't send any CR, or Null, or any other end of command bytes.

8 USB Interface

The GOLDELOX-MD1 module needs to be interfaced to a PC from time to time. Using a standard USB cable and the 4D Systems micro-USB module (uUSB-MB5) as shown below a PC to GOLDELOX connection can be achieved simply. The micro-USB module (optional extra), simply connects to the GOLDELOX-MD1 module and captures the USB data and converts it into serial TTL data. The micro-USB modules and drivers are available from your local 4D distributor. This is an optional extra product and is not included with the module.





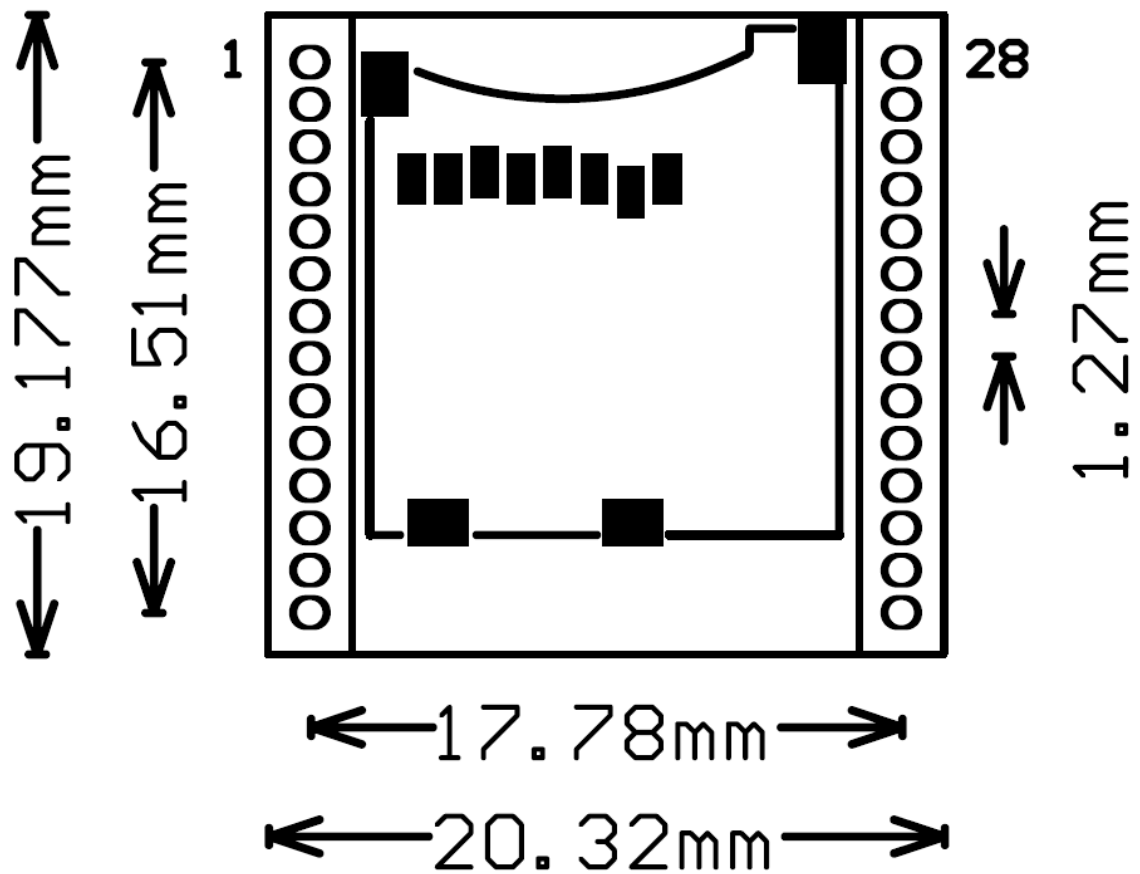
GOLDELOX MD1

9 Specifications

The GOLDELOX-MD1 module has the following electrical specifications which must be adhered to at all times to prevent damage to the device.

Symbol	Characteristic	Min	Typ	Max	Units
VCC	Supply voltage	3.6	5	6.0	V
I	Current	10	15	20	mA
Deg C	Operating temp	0	30	70	C

10 Mechanical Details



The GOLDELOX-MD1 module dimensions are **19.2mm** x **20.3mm** x **5.5mm**.

11 Colour Organisation

11.1 65,536 Colour Bitmap Organisation

The GOLDELOX-MD1 65K colour byte is organised as 5 bits for Red(D11, D12, D13, D14, D15), 6 bits for Green(D5, D6, D7, D8, D9, D10) and 5 bits for Blue(D0, D1, D2, D3, D4). This will give a combination of $32 \times 64 \times 32 = 65,536$ colours. Each colour is not limited to 32/64 shades. For example a lighter shade of Red can be obtained by adding a little bit of the Green and a little bit of the Blue. Full Red and full Green will result in Yellow. Some experimentation will be needed to obtain the desired colour.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B
E	E	E	E	E	R	R	R	R	R	R	L	L	L	L	L
D	D	D	D	D	E	E	E	E	E	E	U	U	U	U	U
					N	N	N	N	N	N	E	E	E	E	E
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Example: To Obtain the Colour Yellow

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B
E	E	E	E	E	R	R	R	R	R	R	L	L	L	L	L
D	D	D	D	D	E	E	E	E	E	E	U	U	U	U	U
					N	N	N	N	N	N	E	E	E	E	E
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0



Example: To Obtain the Colour Magenta

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B
E	E	E	E	E	R	R	R	R	R	R	L	L	L	L	L
D	D	D	D	D	E	E	E	E	E	E	U	U	U	U	U
1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1



Example: To Obtain the Colour White

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B
E	E	E	E	E	R	R	R	R	R	R	L	L	L	L	L
D	D	D	D	D	E	E	E	E	E	E	U	U	U	U	U
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1




11.2 256 Colour Bitmap Organisation

The GOLDELOX-MD1 256 colour byte is organised as 3 bits for **Red** (D5, D6, D7), 3 bits for **Green** (D2, D3, D4) and 2 bits for **Blue** (D0, D1). This will give a combination of $8 \times 8 \times 4 = 256$ colours. Each colour is not limited to 4/8 shades. For example a lighter shade of Red can be obtained by adding a little bit of the Green and a little bit of the Blue. Full Red and full Green will result in Yellow. Some experimentation will be needed to obtain the desired colour.

D7	D6	D5	D4	D3	D2	D1	D0
RED2	RED1	RED0	GREEN2	GREEN1	GREEN0	BLUE2	BLUE1


Example: To Obtain the Colour Yellow

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	1	1	1	1




Example: To Obtain the Colour Magenta

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	0	0	0	1	1



Example: To Obtain the Colour White

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1



12 Power-Up Reset

When the GOLDELOX-MD1 comes out of a power up reset it initialises the Graphics RAM and the internal Display registers. Allow up to 500ms before attempting to communicate with the module. The power up sequence of events should be as follows:

- Allow 500ms after power-up for module to settle. Do not attempt to communicate with the module during this period. The GOLDELOX-MD1 may send garbage on its Tx Data line during this period, the host should disable its Rx Data reception.
- Within 100ms of powering up, the host should make sure it has its Tx line pulled HIGH. If the host Tx (GOLDELOX-MD1 Rx) is LOW or floating after the 100ms period, the module may misinterpret this as the START bit and lock onto some unknown Baud Rate. If the host has a slow wake up time, i.e. less than 100ms, its Tx line maybe floating. This can be easily resolved by adding a pull up resistor on the host Tx line which will ensure the module does not encounter a false START bit. The pull up resistor can be any value within 10K to 100K.
- The host transmits the ASCII 'U' (capital **U**, **55hex**) as the first command so the module can lock onto the host's serial baud rate. This is called "**Auto Baud**". The module will respond with an 'ACK' (06h). See section 2.5
- The GOLDELOX-MD1 module is now ready to accept screen function commands from the host.
- Note: The module will wait up to 5 seconds with its screen blank for the host to transmit the Auto-Baud character. If the host has not transmitted the Auto Baud character by the end of this period the GOLDELOX-MD1 will then display its splash screen. If the host has transmitted the Auto Baud character the screen will remain blank. This wait period is for those customer specific applications where the splash screen is undesired.

13 Related Products

▪ uUSB-MB5

- microUSB module, USB to Serial Bridge
- Standard USB miniB connector
- 10 pin header provides the following signals:
 - 5V, 3.3V, GND, Tx, Rx, Suspend,
 - DTR, CTS, RTS, GND
- 5 Volts supply @ 500mA, 3.3 Volts supply @ 100mA
- Additional flow control signals, DTR, CTS, RTS



▪ uSD-64Mb

- 64Mb micro-SD Memory Card
- Extremely small footprint.
- Measuring only 15mm x 11mm x 0.8mm
- The uSD-64Mb memory card can be used to store images, animations, text or any graphics objects.



▪ Software Utility Tools (free download)

- Range of PC based software utility tools for Windows
- Download PmmC files into the GOLDELOX processor on the module
- Download 4DGL user programs
- Download images/text/animations into the GOLDELOX-MD1 memory card.
- For available software tools please visit the **GOLDELOX-MD1** web-page of your local distributor or visit the 4D Systems website www.4dsystems.com.au

▪ Display Panels

- **OLED-96:** 96x64 65K colour 0.96" OLED Display
- **OLED-2896:** 128x96 262K colour 1.2" OLED Display
- **OLED-128:** 128x128 262K colour 1.5" OLED Display
- **OLED-160:** 160x128 262K colour 1.96" OLED Display
- **LCD-128:** 128x128 65K colour 1.5" LCD Display



14 Appendix

14.1 OLED Display Precautions

- Avoid having a White Background. The more pixels that are lit up, the more the OLED panel will consume current. A full white screen will have the highest power consumption.
- Avoid displaying objects or text on White Backgrounds. This will cause a smearing effect which is inherent to all OLED displays. Instead try a shaded mixed colour as the background or better still a black background. Ideally have mixed coloured objects/text/icons on a black background.
- Avoid having to display the same image/object on the screen for lengthy periods of time. This will cause a burn-in which is a common problem with all types display technologies. Blank the screen after a while or dim it very low by adjusting the contrast. This can be achieved via the “**Display Control Functions**” command (section 6.1.22). Better still, implement a screen saver.
- Observe the Power-Down procedure (section 6.1.22). The GOLDELOX-MD1 module automatically takes care of the proper Power-Up sequence.

14.2 Help and Other Information:

- Assistance with latest information and downloads visit the **GOLDELOX-MD1** product page of your distributor.
- Questions and technical support please email support@4dsystems.com.au
- All related product information can be downloaded from www.4dsystems.com.au/downloads/Modules/GOLDELOX-MD1/