Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

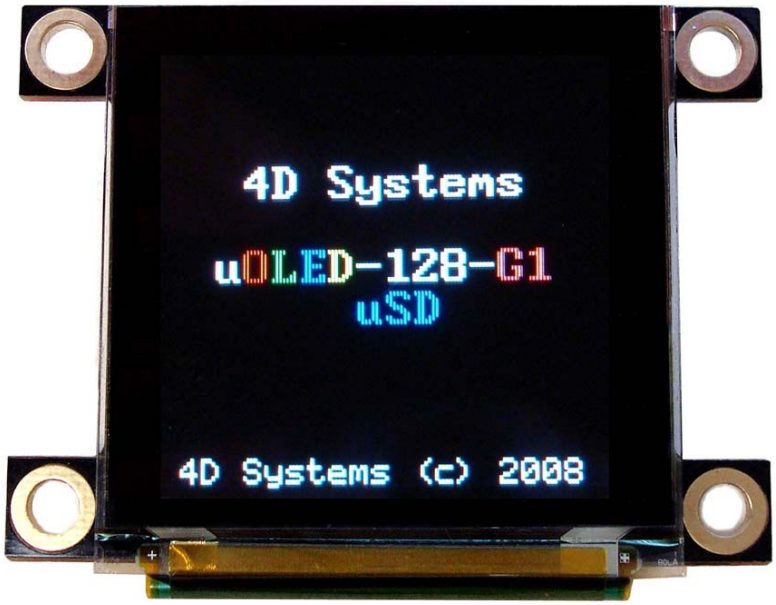
µOLED-128-G1

USERS MANUAL

Intelligent OLED Display Module

For embedded applications with integrated micro-SD card support

Document Revision 1.0 (April 10th 2008)



4D Systems

PROPRIETORY 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 with out 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

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.

Table of contents

1 Introduction

The **µOLED-128-G1** is a compact and cost effective all in one 'SMART" display module using the latest state of the art Passive Matrix OLED (PMOLED) technology with an embedded GOLDELOX graphics controller that delivers 'stand-alone' functionality to any project. The module is designed to operate under 2 different software platforms; the **Serial Command** platform or the **4DGL** (4D Graphics Language) platform.

Serial Command Platform:

The serial command platform allows the **µOLED-128-G1** module to be used as slave device connected to an external host. The host can be any controller such as a PIC, AVR, ARM, STAMP, etc. or even a PC where all screen related functions are sent using a simple protocol via the serial interface. Serial commands may comprise of a single byte or multiple bytes of data depending on the command type. The serial platform allows users to develop their application using their favourite microcontroller and software development tools.

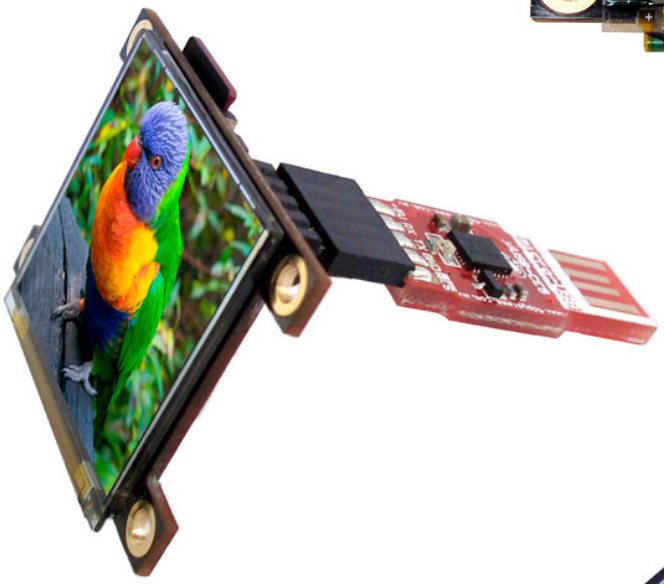
Note: The **µOLED-128-G1** is preloaded with the serial command software platform as the factory default.

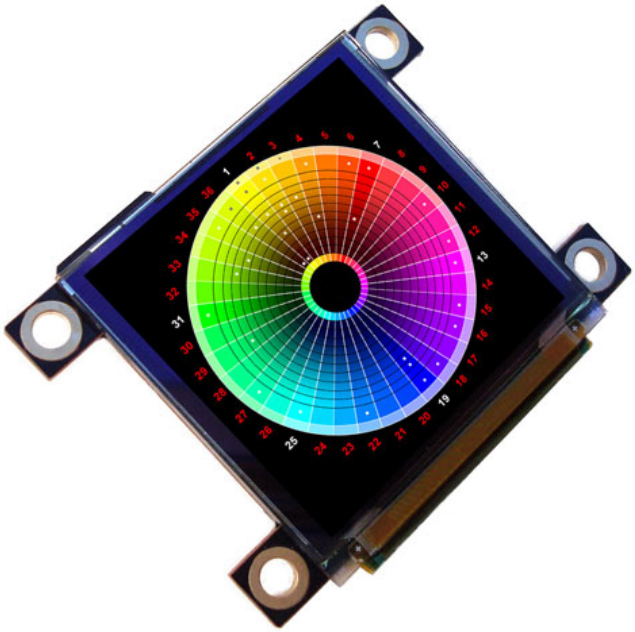
4DGL Platform:

4DGL is a graphics oriented language allowing the developer to write applications in a high level language (syntax similar to popular languages such as BASIC, C, Pascal, etc.) and run it directly on the GOLDELOX processor embedded in the **µOLED-128-G1** module.
The rich set of built in library functions and the high level syntax allows the user to take complete control of all available hardware resources such as the Serial Port, Graphics Display, micro-SD card, I/O pins, etc. This eliminates the need for an external host microcontroller to drive the **µOLED-128-G1** module via serial commands. It provides the user complete independence to quickly develop powerful applications.

Note: The 4DGL Platform will need to be uploaded into the module using the relevant **PmmC** file. You will need the **PmmC-Loader** software tool to assist in the process. The links to these are provided on the **µOLED-128-G1** product page.

Figures below show some of the graphics capability of the **µOLED-128-G1**.

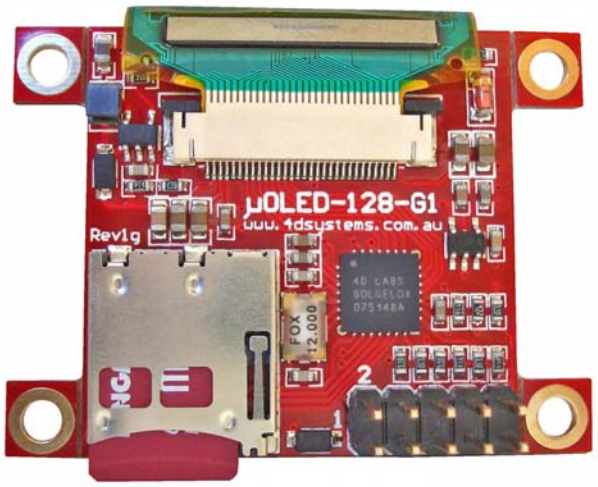


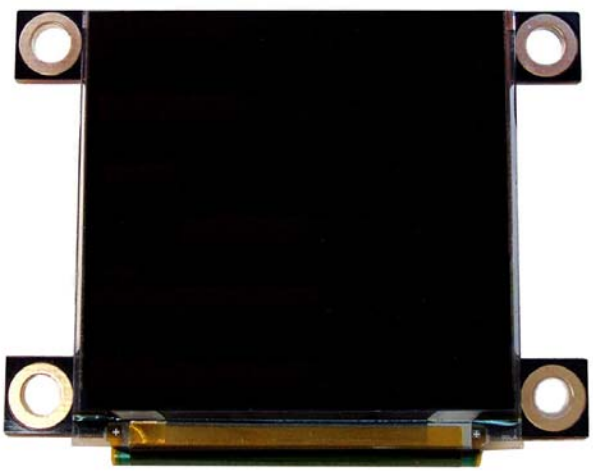


1.1 Features

The **µOLED-128-G1** 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:

- 128 x 128 resolution, 256/65K true to life colours, PMOLED screen.
- 1.5" diagonal size, 45.5 x 33.5 x 6.3mm. Active Area: 27mm x 27mm.
- No backlighting with near 180° viewing angle.
- Easy 5 pin interface to any host device: VCC, TX, RX, GND, RESET
- Voltage supply from 3.6V to 6.0V, current @40mA nominal when using a 5.0V supply. Note: The module may need to be supplied with a voltage greater than 4.0 volts when using it with a SD memory card.
- Serial RS-232 (0V to 3.3V) with auto-baud feature (300 to 256K baud). Rx line has built in series current limit resistor and a pull-up resistor.
- Powered by the 4D Labs **GOLDELOX** processor (also available as separate OEM IC for volume users).
- 2 different operating platforms; the **Serial Command** platform (factory default) or the **4DGL** (4D Graphics Language) platform.
- Optional USB to Serial interface via the 4D micro-USB (**µUSB-MB5** or **µUSB-CE5**) modules.
- Onboard micro-SD (**µSD**) memory card adaptor for storing of icons, images, animations, etc. 64Mb to 2Gig µSD memory cards can be purchased separately.
- Rich set of built in graphics commands and functions.





2 Serial Command Platform

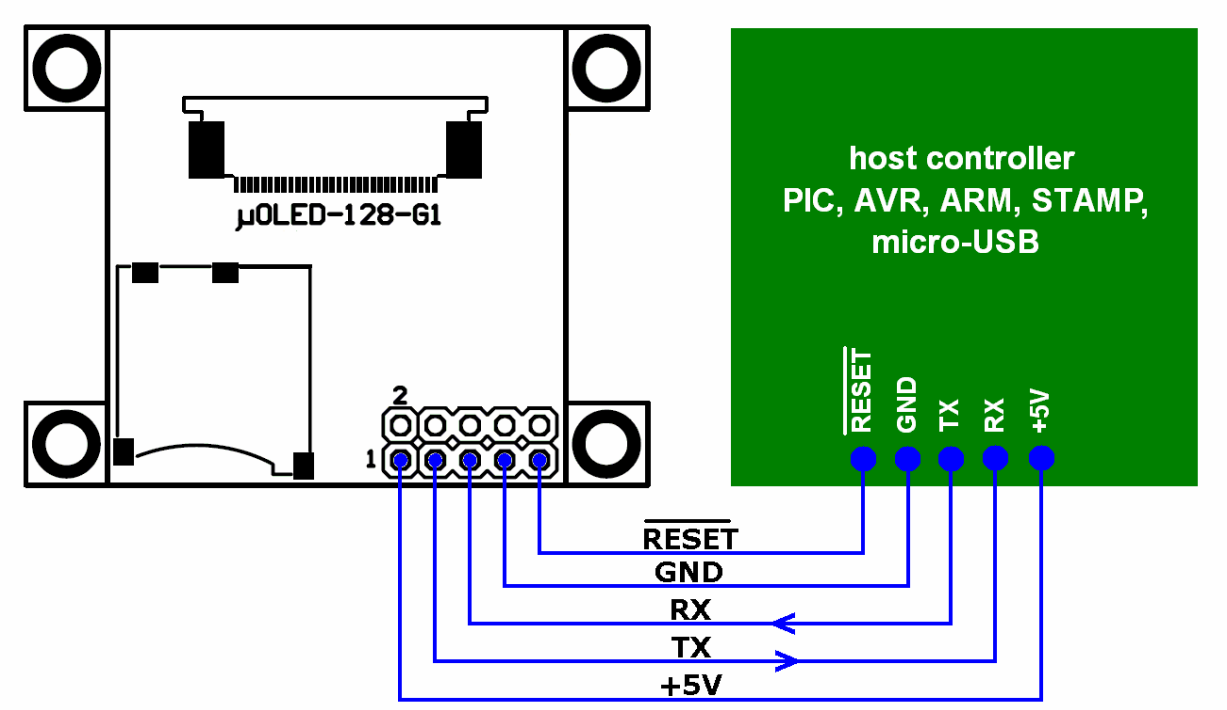
The heart of the Serial Platform is the easy to understand command set. This comprises of easy to learn instructions that provide a full text and graphical user interface. The commands are sent to the **µOLED-128-G1** via its serial connection. The command set is grouped into 3 sections:

- General Command Set
- Display Specific Command Set
- Extended Command Set (uSD Memory Card commands)

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

NOTE!

Serial Data Format: 8 Bits, No Parity, 1 Stop Bit.



2.1 Serial Command Protocol

The following applies only to the Serial Platform and each serial command is described in detail and how it can be used. Please note that all command examples listed in this section are in hex (**00**hex). Due to the high colour depth of the display module, a pixel colour value will not fit into a single byte, a byte can only hold a maximum value of 255. Therefore the colour is represented as a 2 byte value, **colour**(msb:lsb). The most significant byte (msb) is transmitted first followed by the least significant byte (lsb). This format is called the big endian. So for a 2 byte colour value of **013F**hex the byte order can be shown as (**01**hex),(**3F**hex).

NOTE: When transmitting the command and data bytes, do not include any separators such as commas ',' or spaces ' ' or brackets '(' ')' between the bytes. The examples show these separators purely for legibility; these must not be included when transmitting data to the **µOLED-128-G1** module.

Serial Interface:

The **µOLED-128-G1** needs to be connected via a serial link to a host system. The host uses this serial link to send commands to the module so that characters and graphics can be displayed on the screen. Use the signal pin-outs as well as the application example shown in the **"User Interface"** section for correct connection to the host.

Auto Baud Detect:

As previously mentioned, the module has an auto-baud detect feature which can operate from **300 baud to 256K baud**. Prior to any commands being sent to the module, 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 host needs to change the baud rate, the module must be powered down and powered back up again or reset. The "U" command cannot be used to change the baud rate during the middle of normal usage.

Serial Timing:

Each serial command is made up of a sequence of data bytes. Some commands are single byte and others are multiple bytes. When a command is sent, the module will reply back with a single acknowledge byte called the **ACK** (**06**hex). This tells the host controller that the command was understood and the operation is completed. It will take the module anywhere between 1 to several milliseconds to reply back with an **ACK**, depending on the command and the operation it has to perform.

If the module receives a command that it does not understand it will reply back with a negative acknowledge called the **NAK** (**15**hex).

For example, if a command has 5 bytes but only 4 bytes are sent, the command will not be executed and the **µOLED-128-G1** will wait until another byte is sent before trying to execute the command. There is no timeout when incomplete commands are sent. The module will reply back with a **NAK** for each invalid command it receives. For correct operation make sure the command bytes are sent in the correct sequence.

Power-Up Reset:

When the **µOLED-128-G1** comes out of a power up reset, it initialises the Graphics RAM and the internal Display registers. Allow up to 1 second before attempting to communicate with the module. The power up sequence of events should be as follows:

- Allow up to 1000ms after power-up for voltages to settle and internal initialisations to complete. Do not attempt to communicate with the module during this period. The module may send garbage on its Tx Data line during this period; the host should disregard any data.
- Within 100ms of powering up, the host should make sure it has its transmit (TX) line pulled HIGH. If the host TX (module Rx) is LOW after the 100ms period, it may misinterpret this as the START bit and lock onto some unknown Baud Rate.
- The host must transmit the ASCII '**U**' (capital **U**, **55**hex) as the first command so the module can lock onto the host's serial baud rate. This is called "**Auto Bauding**". The module will respond with an '**ACK**' (**06**hex). See previous section.
- The module is now ready to accept screen function commands from the host.

Splash Screen on Power Up:

The **µOLED-128-G1** will wait up to 5 seconds with its screen blank for the host to transmit the Auto-Baud character ('U'). If the host has not transmitted the Auto Baud character by the end of this period the module will 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.

Auto Run uSD Card Slide Show:

The **µOLED-128-G1** module is equipped to accept memory cards. There is a **10 way header** at the back of the unit (on the component side). Upon power-up, if a jumper shunt is inserted across **pins 6 and 8** and there is a preloaded slide show in the µSD memory card, the module will automatically play/display these. The memory cards are supplied as blank separate products and as such the user will have to upload a slide show composition to the card to benefit from this auto play feature. For normal usage this jumper must be **removed**.

See section 4 "**User Interface**" for further details.

2.2 General Command Set

General Command Set	Live	Object	µSD Card
(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) OLED DisplaY Control functions	√	√	√

NOTES:

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

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.**

µSD Card : **Those commands that can reside and be executed from inside the memory card.**

2.2.1 Add User Bitmapped Character (A)

Syntax : cmd, char#, data1, data2, ……., data8

cmd : 41hex, **A**ascii

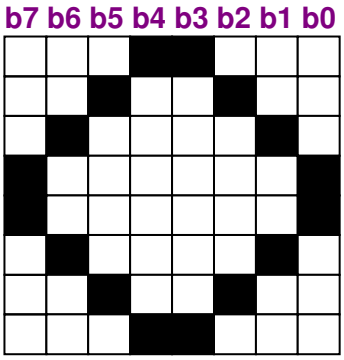
char# : bitmap character number to add to memory:
range is 0 to 31 (**00**h to **1F**h), 32 characters of 8x8 format.

data1 to data8 : 8 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.

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

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

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

2.2.2 Set Background Colour (B)

Syntax : **cmd, colour**(msb:lsb)

cmd : **42**hex**, B**ascii

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

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 : **42**hex**, FFFF**hex
Set the background colour to value 65,535 (**white**).

2.2.3 Text button (b)

Syntax : cmd, state, x, y, buttonColour(msb:lsb)**, font, textColour**(msb:lsb)**, width, height, "string", terminator**

cmd : 62hex**, b**ascii

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

width : text width or horizontal size of the characters in the string, effects the width of the button.

height : text height or vertical size of the characters in the string, effects the height of the button.

"string" : string of ASCII characters (limit the string to line width)

terminator : the string must be terminated with **00**hex

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.

2.2.4 Draw Circle (C)

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

cmd : 43hex**, C**ascii

x : circle centre horizontal position.

y : circle centre vertical position.

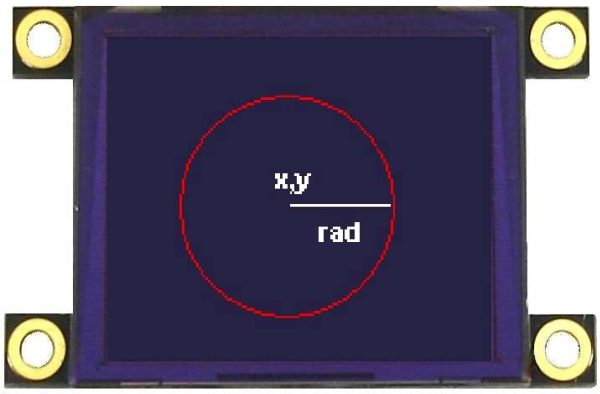
rad : radius size of the circle.

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

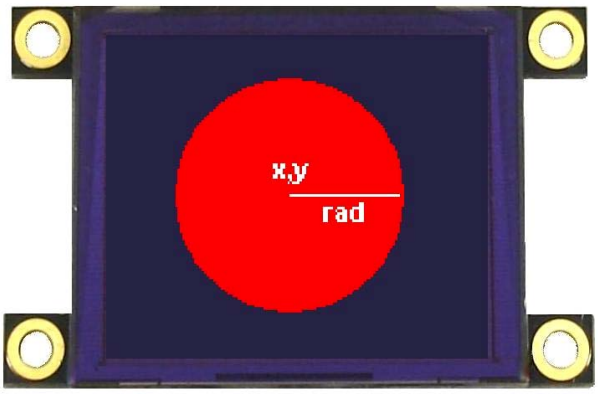
Description : This command will draw a coloured circle centred at **(x, y)** with a radius determined by the value of **rad**. The circle can be either solid or wire frame (empty) depending on the value of the Pen Size (see **Set Pen Size** command). When Pen Size = 0 circle is solid, Pen Size = 1 circle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

Example : 43hex**, 3F**hex**, 3F**hex**, 22**hex, **00**hex**, 1F**hex

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



When Pen Size = 1



When Pen Size = 0

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

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

cmd : 63hex**, c**ascii

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.

2.2.6 Display User Bitmapped Character (D)

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

cmd : 44hex**, D**ascii

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

x : horizontal display position of the character.

y : vertical display position of the character.

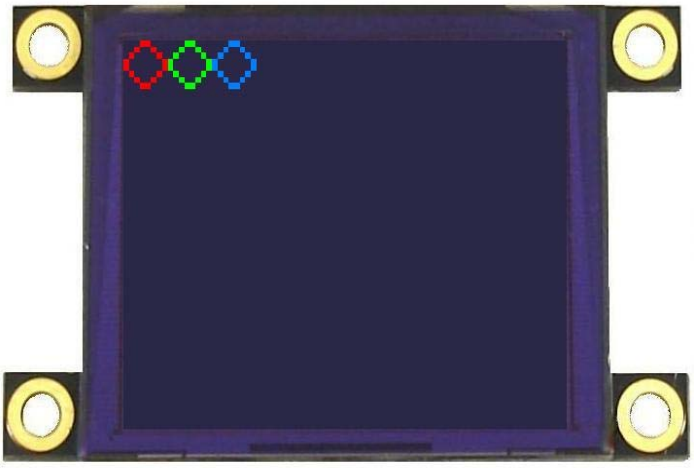
colour(msb:lsb) **:** 2 byte bitmap colour value.

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**, 01**hex **, 00**hex**, 00**hex**, F8**hex**, 00**hex
Display 8x8 bitmap character number 1 at x=0, y=0, colour=red

Example 2: 44hex**, 01**hex**, 08**hex**, 00**hex**, 07**hex**, E0**hex
Display 8x8 bitmap character number 1 at x=8, y=0, colour=green

Example 3: 44hex **, 01**hex**, 10**hex**, 00**hex**, 00**hex**, 1F**hex
Display 8x8 bitmap character number 1 at x=16, y=0, colour=blue



2.2.7 Erase Screen (E)

Syntax : cmd

cmd : 45hex**, E**ascii

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

Example : 45hex
Clear the screen.

2.2.8 Set Font Size (F)

Syntax : cmd, size

cmd : 46hex**, F**ascii

size : = **00**hex : 5x7 small size font
= **01**hex : 8x8 medium size font
= **02**hex : 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 sent. Any character on the screen with the old font size will remain as it was.

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

2.2.9 Draw TrianGle (G)

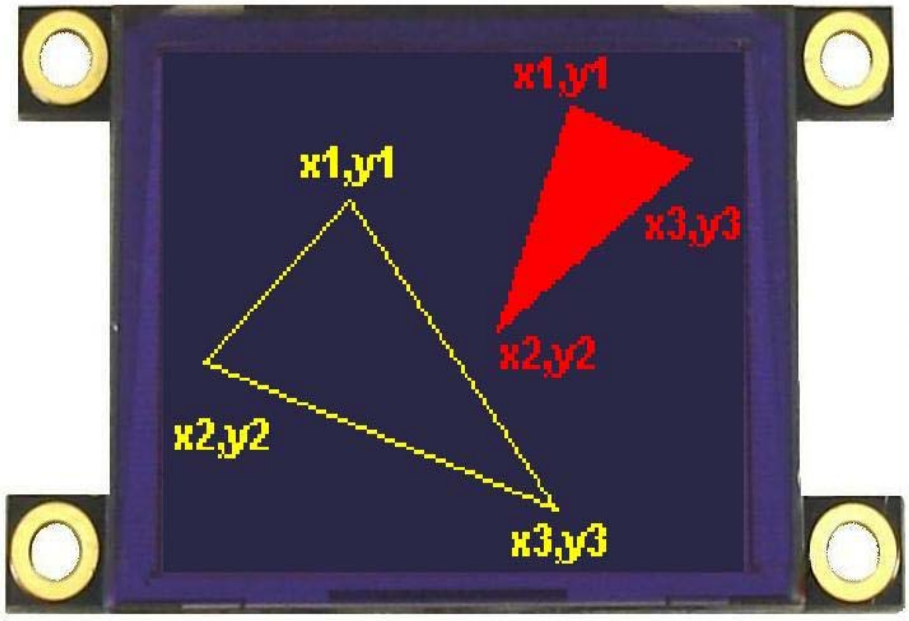
Syntax : **cmd, x1**, **y1**, **x2**, **y2**, **x3**, **y3**, **colour**(msb:lsb)

cmd : 47hex**, G**ascii

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**.



2.2.10 Draw Polygon (g)

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

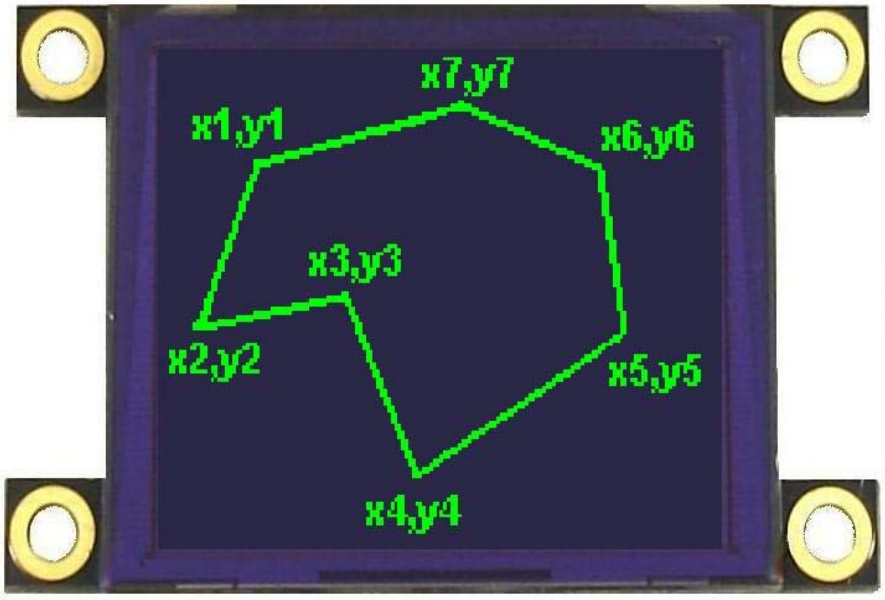
cmd : 67hex**, g** ascii

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.



2.2.11 Display Image (I)

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

cmd : **49**hex**, I**ascii

x : Image horizontal start position (top left corner)

y : Image vertical start position (top left corner)

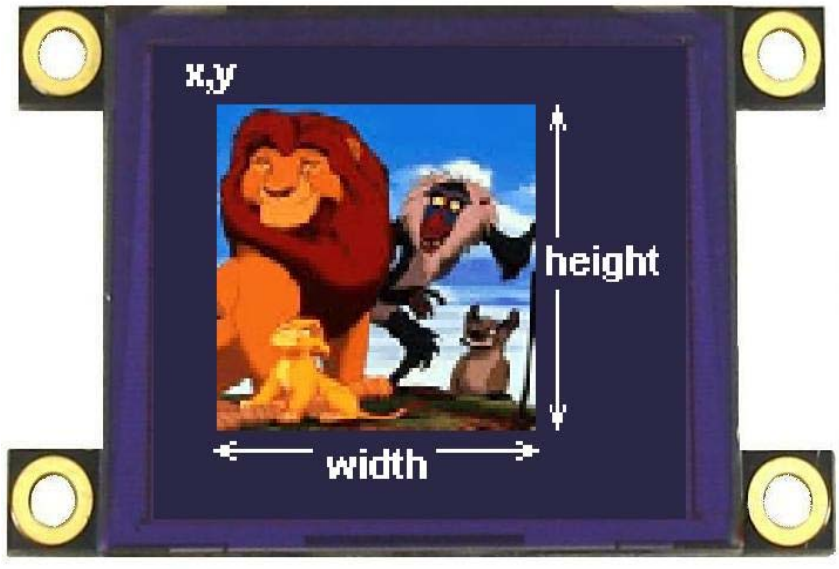
width : horizontal size of the image

height : vertical size of the image

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

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.



2.2.12 Draw Line (L)

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

cmd : **4C**hex, **L**ascii

x1 : horizontal position of line start.

y1 : vertical position of line start.

x2 : horizontal position of line end.

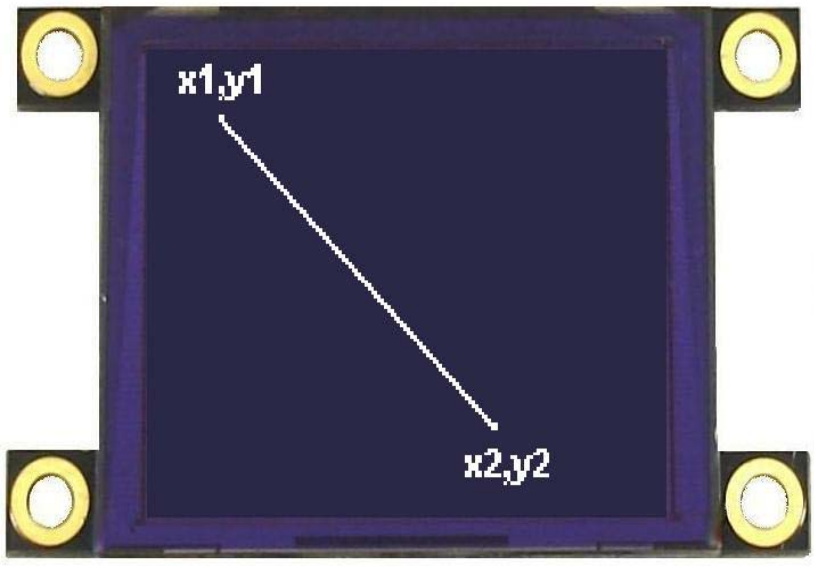
y2 : vertical position of line end.

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 : **4C**hex, **00**hex, **00**hex, **7F**hex, **7F**hex, **FF**hex, **FF**hex

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



2.2.13 Opaque / Transparent Text (O)

Syntax : cmd, mode

cmd : 4Fhex**, O**ascii

mode: = **00**hex: Transparent, objects behind text are visible.
= **01**hex: Opaque, objects behind text 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: **4F**hex**, 00**hex Transparent Text Mode
Example2: **4F**hex**, 01**hex Opaque Text Mode