µLCD MkII Command Set (additional to MkI)
(Draft Only)

The following commands are related to the µLCD MkII only. This document is a supplement to the original "µLCD Users Manual" (referred to as MkI) and it must be read in conjunction with it. The µLCD MkII incorporates all of the µLCD MkI commands plus the additional commands outlined here. The features and commands set out in this document help differentiate MkII from the MkI model.

You will find references being made to "**Objects**" throughout the document. An object can be simply defined as those commands that reside inside the flash memory (programmed/downloaded) and can be displayed on the screen by the **"Display Object from Flash Memory"** command.

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

Tables below list all of the commands for both MkI and MkII.

Command Set Summary (MkI & MkII)

	Live	Object	Flash
Erase Screen	x		x
Background Colour	x		x
Put Pixel	x		
Read Pixel	x		
Draw Circle	x	x	x
Draw Line	x	x	x
Font Size	x		x
Opaque or Transparent Text	x		
Place Text Character (formatted)	x	x	x
LCD Display Control Functions	x		x
Add User Bitmapped Character	x		
Display User Bitmapped Character	x		
Paint Area	x	x	x

Command Set Summary (MkII only)

	Live	Object	Flash
Draw Circle with Fill	x	x	x
Place Text Character (unformatted)	x	x	x
Place String of ASCII Text (formatted)	x	x	x
Display Image	x	x	x
Set Transparent Colour (for Images only)	x		x
Enable / Disable Transparent colour	x		x
Place Button Icon	x	x	x
Block Copy & Paste (Screen Bitmap Copy)	x		
Display Object from Flash Memory	x		
Delay			x
Loop Start			x
Loop End			x
Run	x		
Exit	x		x
Restart			x
Download Data to Flash Memory	x		
Read Data from Flash Memory	x		
Erase Flash Memory	x		

NOTES:

Live :	Those commands that can be sent via the serial link and executed by the host.
Object :	Those commands that can be recalled from the flash memory at any time by the host and displayed on the screen using the "Display Object from Flash Memory" command.
Flash :	Those commands that can reside and be executed from inside the flash memory.

Draw Circle with Fill:

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

cmd : 69hex**, i**ascii

x : circle horizontal centre position

y : circle vertical centre position

rad : length of circle radius (in pixel units)

colour(msb), **colour**(lsb) **:** 2 byte fill colour of the circle

Description : This command will draw a filled circle centred at **(x, y)** with a radius determined by the value of **rad**.

Place Text Character (unformatted)

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

cmd : 74hex**, t**ascii

char : Ascii character

x : the horizontal location (in pixels units)

y : the vertical location (in pixels units)

colour(msb), **colour**(lsb) **:** 2 byte colour 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.

Place String of Ascii Text (formatted):

Syntax : cmd, column, **row**, **font_size**, **colour**(msb), **colour**(lsb), **char1**, .. **charN, terminator**

cmd : 73hex**, s**ascii

char : Ascii character

column : the horizontal start position of string

row : the vertical start position of string

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

colour(msb), **colour**(lsb) **:** 2 byte colour of the string

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

terminator : string terminator, must be **00**hex.

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 **00**hex. 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**.

Display Image

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

cmd : 49hex**, I**ascii

x : Image horizontal start position (top left corner)

y : Image vertical start position (top left corner)

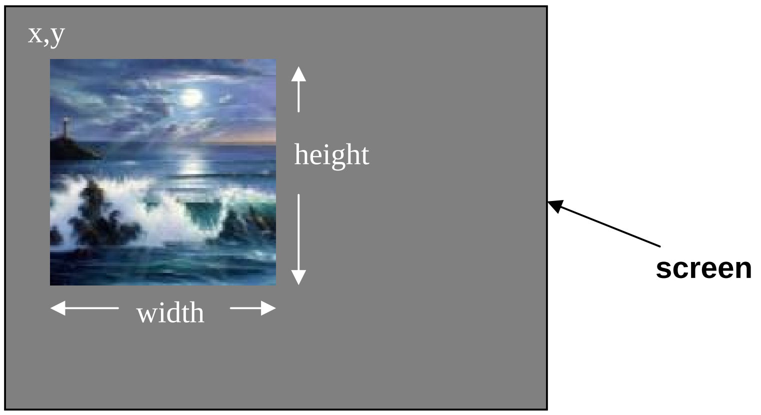
width : horizontal size of the image

height : vertical size of the image

colour_mode : 0 = 256 colour mode, 8bits/1byte per pixel
1 = 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 colour_mode = 0
N = height x width x 2 when colour_mode = 1

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.



Set Transparent Colour (for Images only)

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

cmd : 4Ehex**, N**ascii

colour(msb), **colour**(lsb) **:** 2 byte transparency colour

Description: When a 2 byte transparent colour is specified, it inhibits those matching colour values of the bitmap image from being displayed or painted on to the screen. For example, a small icon image maybe surrounded by an area of a certain background colour. If we know the colour value of the background, we can stop it from being displayed on the screen and only display the image of interest.
For example, the smiley on the left has a red background colour and if we set our transparent colour value to match the same red colour value, the resultant display on the screen will be the smiley on the right.
The power up default value = 0xFFFF (White)

Enable/Disable Transparent colour

Syntax : cmd, mode

cmd : 65hex**, e**ascii

mode : 0 = Disable Transparent colour (default)
1 = Enable Transparent colour

Description : The Transparent Colour function can be enabled or disabled by executing this command. See “Set Transparent Colour” command for a detailed explanation.

Place Button Icon

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

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)

x1 : top left horizontal start position of the button

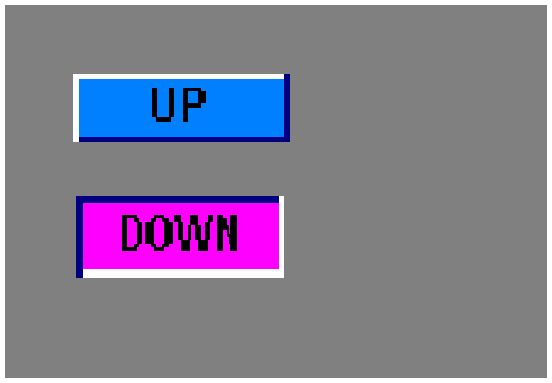
y1 : top left vertical start position of the button

x1 : bottom right horizontal end position of the button

y1 : bottom right vertical end position of the button

colour(msb), **colour**(lsb) **:** 2 byte button colour value

Description : This command will place a Button similar to the ones used in a PC Windows environment on the screen. **(x1, y1)** refers to the top left corner of the button and **(x2, y2)** refers to the bottom right hand corner of the button on the screen. 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. Text can be placed inside the button, using the 't' **"Place Text Character"** (unformatted) command, describing the button function.



Block Copy & Paste (Screen Bitmap Copy)

Syntax : cmd, x_source, y_source, x_dest, y_dest, width, height

cmd : 63hex**, c**ascii

x_source: top left horizontal start position of block to be copied

y_source: top left vertical start position of block to be copied

x_dest: top left horizontal start position of where the copied block is to be pasted

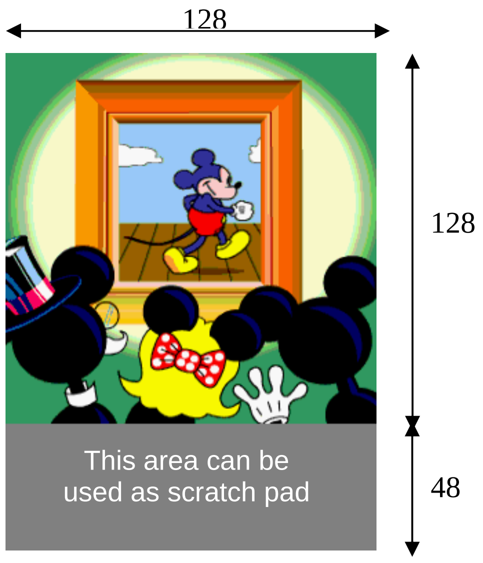
y_dest: top left vertical start position of where the copied block is to be pasted

width : horizontal size of the block to be copied

height : vertical size of the block to be copied

Description : This command copies an area of a bitmap block of specified size defined by the **width** and the **height** parameters. The start location of the block to be copied is represented by **x_source, y_source** (top left corner) and the start location of where the block is to be pasted to, is represented by **x_dest, y_dest** (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.
The Graphics RAM or the Screen Memory is made up of 128 x 176 pixels. Only the 128 x 128 portion is used for the display. Therefore there is a 128 x 48 area of memory that can be used as a temporary scratch pad to copy and paste blocks of bitmap data. This is ideal for animations, where the area under the object can be copied over to the scratch pad before it's displayed and then re-pasted back once the object has been moved to a different screen location. The following indicates how the graphics RAM is utilised:

Graphics RAM	Comment
x = 0, y = 0 to x = 127, y = 127	Used for the Screen Display
x = 0, y = 128 to x = 127, y = 175	Scratch Pad area



Display Object from Flash Memory

Syntax : cmd, address(hi), **address**(mid), **address**(lo)

cmd : 66hex**, f**ascii
address : 24bit (3 bytes) address of the object in flash ROM
(hi) = Address high byte
(mid) = Address middle byte
(lo) = Address low byte

Description: Some of the commands can be stored as objects in the internal flash which can be later recalled by the host on demand and displayed or executed. The user must make sure the 24 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 following is a list of all the commands that can be stored as objects within the internal flash ROM:
Circle (empty), Circle (fill), Line, Text Character (formatted), Text Character (unformatted), Radio Button, Paint Area, Display String, Display Image.

Delay (must reside inside the flash)

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

cmd : 04hex

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 flash memory 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.

Loop Start (must reside inside the flash)

Syntax : cmd, counter

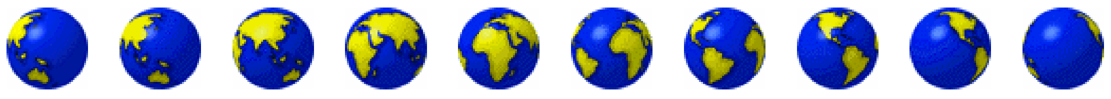
cmd : 0Dhex

counter : A 1 byte counter that determines how many times the loop occurs between the "Loop Start" and "Loop End" commands. 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 allows the user to determine exactly how many times the series of images are looped. This command must always be terminated with the "Loop End" command.

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 flash memory. When the images are displayed sequentially, the effective duration will only be the length of time it takes to display the 10 image frames. With this command, we can increase that length by looping thru the animation a number of times depending on the value set in the **counter** byte. When the display reaches the end of the last frame and encounters the "Loop End" command, the counter will be decremented and then the internal pointer will jump back to the object just after the "Loop Begin" command. This sequence will then repeat until the value in the counter reaches zero. The following demonstrates how this maybe used:

```
Loop_Start, (counter = 25),
Image1,
Delay(10ms),
Image2,
Delay(10ms),
…,
…,
Image10,
Delay(10ms),
Loop_End (decrements counter then jumps to Image1)
```



Loop End (must reside inside the flash)

Syntax : **cmd**

cmd : 0Chex

Description : Decrements the “Loop Start” counter and forces the internal flash pointer to jump to that command/object, which resides in the flash memory, just after the “Loop Start” command. When the “Loop Start” counter becomes zero, the “Loop End” command is bypassed and the command/object following it is executed. See “Loop Start” command for a detailed explanation.

Run

Syntax : cmd

cmd : 06hex

Description : The Run command forces the 24bit internal flash pointer to reset to zero (000000hex) and automatically start executing commands, from the flash memory, without any further interaction by the host processor. It will sequentially execute any valid flash related commands and display objects until it gets to the end of the 1Mb flash memory. It is advisable to have the “Exit” or the “Restart” command at the end of the user composed slide show so that the pointer does not run off so to speak.

Exit

Syntax : cmd

cmd : 07hex

Description : This command forces the program to stop executing from the internal flash memory and ready to accept and execute commands from the host via the serial interface. It can reside in the flash which will force the slide show to stop or it can be sent via the serial port while the program is running from the internal flash.

Restart (must reside inside the flash)

Syntax : cmd

cmd : 05hex

Description : The Restart command forces the 24bit internal flash pointer to reset to zero (000000hex). If a certain slide show is composed inside the flash memory and the "Restart" command is encountered at the end of it, the program will then jump back to the start of the flash and begin executing again.

Download Data to Flash Memory

Syntax : **cmd, pageNum**(msb), **pageNum**(lsb), **data**(1), **.. , data**(256)**, chkSum**(msb), **chkSum**(lsb)

cmd : 08hex

pageNumb(msb, lsb) **:** A 2 byte page number from 0 to 4095. The internal 1Mb flash memory has 4096 pages and each page is 256 bytes long.

data(1 to 256) : 256 bytes of data. The data length must be 256 bytes long. If not all used then the rest must be padded.

chkSum(msb, lsb) **:** A 2 byte checksum value. The checksum is calculated by taking a binary sum of all bytes in the command frame and forming a 2 byte value, from (and including) the pageNum(msb) byte to the last data byte data(256). Then, the two's complement is taken (i.e., invert all bits of the result and then add 1) to yield the checksum. **Note**: If the command message is correct, adding the checksum word to the 2 byte sum of all the other bytes (excluding cmd) in the message will give a result of zero.

Description : This command allows downloading of objects such as images and other commands for storage that can be retrieved and used later on. The μLCD-MkII has a 1Mbyte of flash memory which is divided into 4096 pages and each page is 256 bytes long. Downloads must always be limited to 256 bytes in length. For large objects such as images, the data must be broken up into multiple pages (chunks of 256 bytes) and this command then maybe used many times until all of the data is downloaded. There is no provision to download a single byte or few bytes and the data length must always be 256 bytes long. If only few bytes of data are to be downloaded, then make sure the rest of the remaining data are padded with 00hex or FFhex (it can be anything).

Once the command message is sent and the checksum is correct, the μLCD will take a few milliseconds to write the data into its flash memory and at the end of which it will reply back with an **ACK**(06hex). If the checksum is incorrect a **NAK**(15hex) will be sent back without any write attempts.

Only **data**(1) to **data**(256) are stored in the flash. Other bytes in the command message such as page number and checksum are not stored.

Read Data from Flash Memory

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

cmd : 09hex

pageNumb(msb, lsb) **:** A 2 byte page number from 0 to 4095. The internal 1Mb flash memory has 4096 pages and each page is 256 bytes long.

Description : This command provides a means of reading data back from the flash memory in lengths of 256 bytes. It maybe useful in validating the data that was stored previously using the download command. Once this command is sent, the µLCD will return 256 bytes of data relating to that particular page.

Erase Flash Memory

Syntax : cmd,

cmd : 0Ahex

Description : Before any data can be downloaded and stored in the flash memory, it must first be erased. There is no provision to erase individual bytes or pages of memory. Once this command is executed, all of the 1Mbytes (4096 pages) of memory will be erased. It will take around 10 seconds to completely erase the flash at the end of which an **ACK**(06hex) is returned.