EB043 Graphical LCD Programming Strategy

The EB043 Graphical LCD E-Block comes complete with test code and sample code to help get you started with programming your own graphical LCD operations.

This document will explain how to use the sample C functions to allow you to make your own graphical LCD programs and routines.

Contents of the Graphical LCD programmer support package

Here is a list of the files that you receive to support your MMC E-Block.

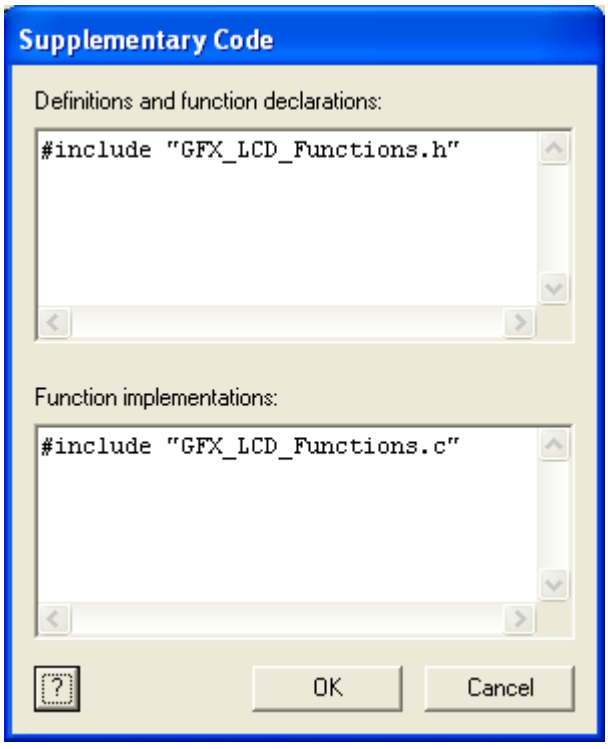
- GFX_LCD_Functions.c – C code file with simple driver functions.
- GFX_LCD_Functions.h – Header file with the definitions of the C functions.
- E-Blocks.fcf – Flowcode file demonstrating most Graphical LCD functions.
- Text.fcf – Flowcode file demonstrating Graphical LCD text functions.
- Plot.fcf – Flowcode file demonstrating Graphical LCD image functions.

Including the C functions in your program

To include the C functions into your programs you will need to copy the GFX_LCD_Functions.c and the GFX_LCD_Functions.h files into the directory containing your program and then do the following.

Flowcode Users:

Add the following lines into the supplementary code window available from the Edit menu.



BoostC Users:

Add the following lines to the beginning of your program.
#include “GFX_LCD_Functions.h”
#include “GFX_LCD_Functions.c”

Other PIC C Users:

The Functions and header files will need to be ported into your version of C. This should be a relatively easy job and should be fairly self evident if you a familiar with your particular version of C.

Graphical LCD Commands

Here is a brief example of some of the more common MMC commands.

Commands and parameters are sent to the graphical LCD by first sending the instruction type bit and then the data byte. Commands are specified as logic 1 and parameters are represented by logic 0.

Command	Type	Instruction
Reset	Command	0x01
Sleep Off	Command	0x11
Display On	Command	0x29
Interface Pixel Format	Command	0x3A
Data Format 8 Bit	Parameter	0x02

For a full list of commands and their recommended order see the init and sendByte functions in the GFX_LCD_Functions.c file.

Description of C functions

Here is the list of functions with details of their operation, listings of their inputs and expected responses.

SPI Bus and Graphical LCD Initialise Function

The Graphical LCD is interfaced via a protocol named SPI. This protocol uses 4 I/O pins to represent data in, data out, data clock and chip select. The example Graphical LCD functions use a software bit banging method to create the SPI bus. The bit banging approach was used so that PICmicros that don’t have the hardware SPI module can still access the E-Block. The digital I/O pins that are used to control the Graphical LCD E-Block are defined in the GFX_LCD_Functions.h file.

```
#define LCD_PORT   portc
#define LCD_TRIS   trisc
#define RS         3
#define CS         2
#define SCLK       1
#define SDATA      0
```

To initialise the SPI bus and the Graphical LCD, use the Lcd_init function. This function configures the direction of data on the I/O pins and then sends commands to start up the Graphical LCD. The Lcd_init function is designed for use with 16F877A and other pin compatible PICmicros. If using a PICmicro that uses different pins or ports to access the Graphical LCD then the pin definitions in GFX_LCD_Functions.h will need to be edited accordingly.

Example Lcd_init function call:

```
Lcd_init();
```

Clear Function

To start working with the Graphical LCD the first thing that needs to be done is to clear the display so it is ready for any other graphical data. The display is cleared using the function Lcd_clear.

Example Lcd_clear function call:

```
Lcd_clear();
```

Colours

In the example GFX_LCD_Functions.h file there is a list of predefined colours available. To use the colour when calling a function you can simply pass the colour name. E.g. (BLUE). These colours are all 8-bit and are created by a look up table available in the GFX_LCD_Functions.c file. The display can also be set into a 16-bit colour mode but this requires a lot more processing from the PICmicro, so it won't be covered in this programming strategy.

```
#define BLUE            0x03
#define YELLOW          0xFC
#define RED             0XE0
#define GREEN           0X1C
#define BLACK           0X00
#define WHITE           0XFF
#define BRIGHTGREEN     0X3D
#define DARKGREEN       0X14
#define DARKRED         0XA0
#define DARKBLUE        0X02
#define BRIGHTBLUE      0X1F
#define ORANGE          0XF8
```

Graphical LCD Pixels

The Graphical LCD displays have 132 by 132 pixels, which are indexed 0 – 131. You can output a single pixel onto the display by using the Lcd_plot function.

Example Lcd_plot function call

```
Lcd_plot ( x1, y1, Colour );        //Outputs a colour pixel at location X1, Y1
Lcd_plot ( 35,  35, BLUE);          //Outputs a blue pixel at location 35, 35
```

Drawing Lines

The Lcd_drawline function simply uses the Lcd_plot function to set in pixels between coordinates X1, Y1 and X2, Y2.

Example Lcd_drwaline function call

```
Lcd_drawline (x1, y1, x2, y2, Colour);  //Draws a coloured line X1, Y1 and X2, Y2
Lcd_drawline (0, 0, 131, 131, BLUE);      //Draws a blue line top left to bottom right
```

Drawing Boxes

To draw a square or rectangle on the Graphical LCD you can use the Lcd_box command. Similarly you can use this command to clear the LCD with a square box of any colour. Set coordinates x1, y1 to be the top left hand corner and coordinates x2, y2 to be the bottom right hand corner.

Example Lcd_box function call

```
Lcd_box (x1, y1, x2, y2, Colour);      //Draws a coloured box from x1, y1 to x2, y2
Lcd_box (35, 35, 42, 50, BLACK);   //Draws a black box from 35, 35 to 42, 50
Lcd_box (0, 0, 131, 131, BLUE);      //Fills the entire screen with solid blue
```

Printing Text

The Graphical LCD does not include a alphanumeric lookup table so any text that is displayed has to be sent at the pixel level. Luckily there is the Lcd_print function included in the GFX_LCD_Functions.c file, which has pixel data for almost all of the ASCII characters. This obviously takes up a large chunk of the program memory so if you are struggling with fitting your program onto the PICmicro then some of the characters can be deleted from the GFX_LCD_Functions.c file. However be aware that removing data from the ASCII lookup table will cause problems in the Lcd_print function if you are not careful. The Lcd_print function uses the pre stored lookup table of ASCII characters to output a string at coordinates X1, Y1 in one of three font sizes with a colour for the text and a colour for the background. The output coordinates differ from the pixel coordinates used in the previous functions. The coordinates used in the Lcd_print function signify column address and line number in relation to the current font size.

The font sizes that are allowed are:
0 – Standard font 5 x 8 pixels
1 – Tall font 5 x 16 pixels
2 – Large font 10 x 16 pixels

Coordinates example: **xa** = x1 x (fontx + 1) **ya** = y1 x (fonty + 1)
x1 = 1, y1 = 2, Font = 0 : Actual top left co-ord of character : xa = 6, ya = 18
x1 = 1, y1 = 2, Font = 1 : Actual top left co-ord of character : xa = 6, ya = 34
x1 = 1, y1 = 2, Font = 2 : Actual top left co-ord of character : xa = 11, ya = 34

Example Lcd_print function call

```
Lcd_print ( String, x1, y1, Font, F_Colour, B_Colour );
Lcd_print ( “Hello”, 1, 1, 0, BLACK, WHITE );
```

Plotting Data

There is a sample program to demonstrate data plotting called Plot.fcf. The program takes its input from analogue channel 0 and then draws a chart of the voltage against time. The analogue signal is converted from 0 – 1024 to 0 –102 by diving the signal by 10 so that the full input range will fit onto the display. The data point is plotted using the Lcd_plot function. For a slightly clearer output, the line function could be used to draw a line between the previous ADC value and the current ADC value.

Drawing Images

Displaying images is similar to printing text, as both of them have to be sent one pixel at a time. This means that to output an image you first must have the image stored into the memory of the PICmicro or a storage device that the PICmicro can interface. Once the image is stored on or made available to the PICmicro, the Graphical LCD has to be started in 16-bit colour mode to allow colours to look correct. You can also change the image to use 8-bit colour but as a general rule most picture files will be at least 16-bit by default.

There is a partial example for outputting an image onto the Graphical LCD. The Lcd_drawimage function creates a window, 16 by 16 pixels wide and then fills it with colour values 0x00 through to 0xFF. This function can easily be edited to create your own image size and to send image data instead of a count value. The variables x1 and y1 stand for the top left pixel of the image area.

Example of Lcd_drawimage function call
Lcd_drawimage(x1, y1);

Example Flowcode Programs

There is an example Flowcode program designed to test your MMC E-Block and provide an example of how to utilize the MMC functions provided.

E-Blocks.fcf

A demonstration of the how to use text, lines and boxes with the Graphical LCD E-Block.

Text.fcf

A demonstration of the different font sizes and text coordinates.

Plot.fcf

A demonstration of how to use the Graphical LCD to plot an analogue value.