Laboratory #6
Primitive Layer of the Microchip Graphics Library

Objective: To get familiarization with the Primitive Layer of the Microchip Graphics Library.

Introduction:

Graphic displays are becoming very common in increasing number of embedded applications such as mobile communication and navigation devices, point-of-sales, home appliances, medical devices, etc. Among the main benefits of adding graphic functions to display applications are: they offer greater enhancement to the user experience and they deliver more accurate and detailed information with sharper images. Also, they allow more sophisticated graphic rendering capability for certain markets, and finally, they add technologies such as touch screen to the display, leading to more effective and efficient usages in the application. All these are made possible by a significant price drop in LCD displays over the last few years.

Microchip provides a rich set of resources for graphics application developments. These include free Microchip Graphics Library, with source code, API documentation, touch screen support, and a number of low-cost, full-featured development environments with integrated graphics displays. Microchip also enables 3rd party graphic support to provide faster time-to-market software development. The Microchip Graphics Library is available for 16-bit and 32-bit PIC MCUs and DSCs. The library integrates some device specific, some application specific, and mostly device and application independent modules that provide a solution that is simple to use, and easily portable.

The Microchip Graphics solution supports up to 320 x 240 QVGA (quarter VGA) resolution with up to 16-bit or 65K colors. The Microchip Graphics Library is designed in a layered architecture (see Figure 1). At the bottom level of this architecture is the Device Driver Layer, in which basic functions (such as PutPixel, GetPixel, SetColor, etc.) that are needed to initialize and use the display device are implemented. These operations are carried out through a hardware specific display device controller. In some cases, a Graphical Accelerator is implemented in the display controller which performs a basic rendering function efficiently. This may be enabled by writing to some special registers in the device controller.

Above the Device Driver Layer is the Graphics Primitive Layer. This layer calls the basic functions in the Device Driver Layer to render primitive shapes. For example, a Line function in the Graphics Primitive Layer calls a PutPixel function of the Device Driver Layer repeatedly with the controlled changes to the x and y positions. Each pixel identified by the x and y positions gets set to the color chosen for the line.

The next layer is the Graphics Object Layer. In this layer, advanced objects such as Widgets are implemented. Examples of Widgets currently available include button, slider, meter, check box, window, progress bar, etc. This layer also automatically handles management of objects from creation to destruction.

The library assumes the application layer to contain the modules that initialize and control the user interface modules (such as keypad, touchscreen). Through the message interface, the application can communicate with the Objects. Together with the Graphics Object Layer API, applications can now fully integrate the Objects into the application functions. Furthermore, the architecture offers the facility to allow the application directly communicate with any layer of the library.

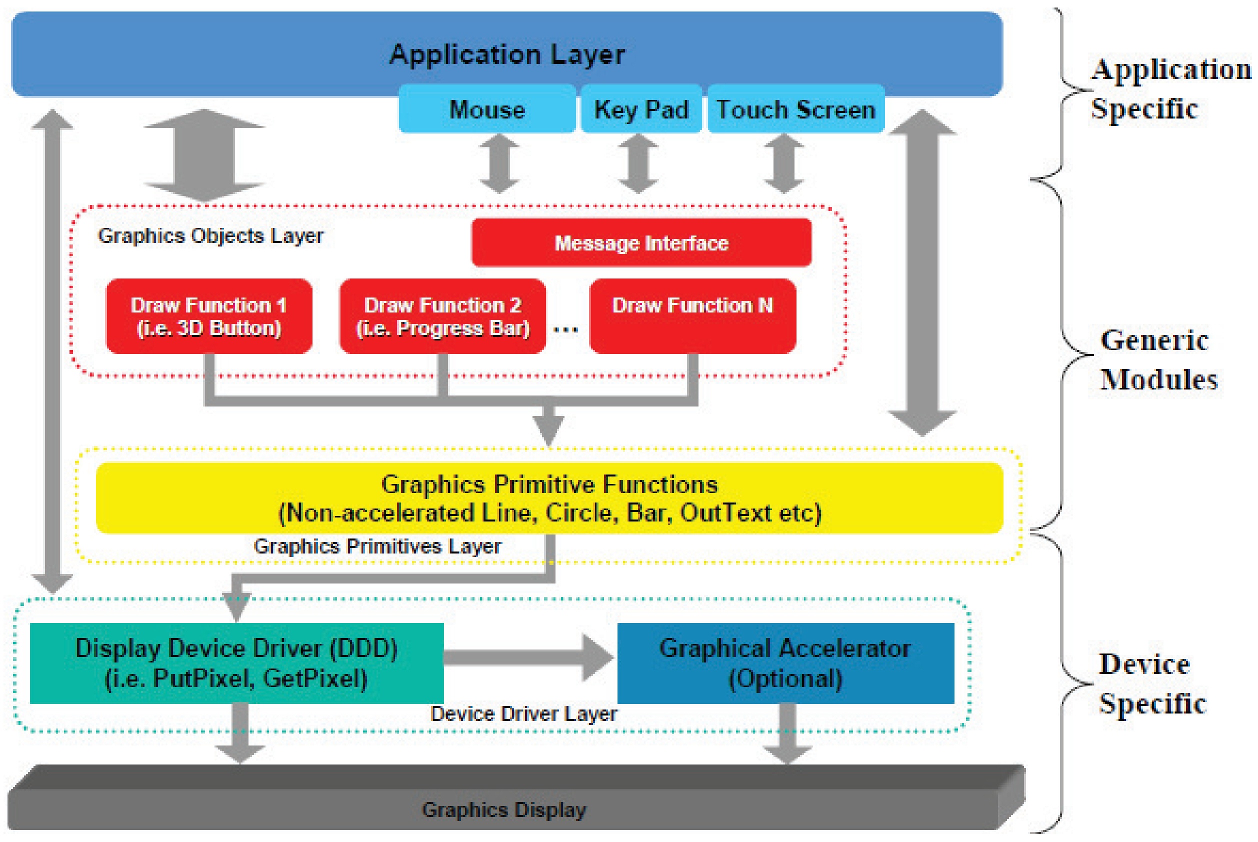
Application Layer
Mouse
Key Pad
Touch Screen
Graphics Objects Layer
Message Interface
Draw Function 1 (i.e. 3D Button)
Draw Function 2 (i.e. Progress Bar)
...
Draw Function N
Graphics Primitive Functions (Non-accelerated Line, Circle, Bar, OutText etc)
Graphics Primitives Layer
Display Device Driver (DDD) (i.e. PutPixel, GetPixel)
Graphical Accelerator (Optional)
Device Driver Layer
Graphics Display
Application Specific
Generic Modules
Device Specific

Figure 1: Layered architecture of the Microchip Graphics Library

Task:

The task in this week's lab is to make use of the basic functions in the Primitive Layer of the Microchip Graphics Library for implementing a simple application described as follows. The program starts by first displaying a "Welcome to ECE" message overlaid on a background graphics picture of the Kettering University logo. Then it displays animated geometrical objects (such as triangle, rectangle, circle, etc.) that are centered at the middle of the graphics display. The shapes start out with small sizes and gradually grow in slow animation to the maximum size limited by the display dimensions. In the middle of the drawings the application prints the name of the geometrical object being displayed. The program should cycle through all the geometrical shapes and eventually repeat the whole process starting with the Welcome screen.

To illustrate the expected output I have provided on Blackboard an executable file (in HEX format) with the filename ‘PrimitiveLayerLab.hex’. Program your MCU on the Explorer 16 board using this executable file and observe the output. Your program should behave exactly the same way.

Helpful Notes:

1) You will find the demo programs that come with the Microchip Graphics Library very helpful. Feel free to use them as templates for building your applications. In particular the “Graphics Primitive Layer Demo” provides a good example for this week’s lab. Carefully go through the program files, making sure that you understand the different pieces of the design.
2) If you would like to test the graphics demos first make a copy of the demo folder into your flash drive. Before opening the project in MPLAB follow the following steps to make sure that the project has the correct paths for each of the source and library include files.
   a) Edit the project file (with the file type “Microchip MPLAB.Project” or file extension of .mcp) that corresponds to the target hardware platform for your project by opening the file in a text editor (such as Notepad). Modify the reference to the root directory of the library include files by searching for the old folder name and replacing it with the correct name, such as:

      Search for ".." and replace with "C:\Microchip Solutions"

   b) Edit the "HardwareProfile.h" file located in the root folder of the project by appropriately indicating the type of graphics hardware your target platform has, i.e. Graphics PICtail Plus Board version 2 or version 3, on either a PIC24 or PIC32MX, on the Explorer 16 or other demo boards, etc. We are currently developing the programs for the Explorer 16 target board with a “PIC32MX” MCU and the “Graphics PICTail Version 2” graphics board. Therefore, you need to identify this hardware configuration in the “HardwareProfile.h” file. This file needs to be edited just once and could simply be copied and pasted as new projects are created, as long as the hardware configuration remains the same.

3) To directly download hex files into a device, do the following:
   a) Close all open workspaces of MPLAB
   b) Configure your programmer by selecting the right programming interface
   c) In MPLAB, click on 'File' and select 'Import…' and select the hex file to import
   d) It seems that nothing is happening, but the hex file is imported
   e) Click on the programmer icon or select the ‘Program’ command under the ‘Programmer’ menu. The output window should state that the device is programmed. The program will then start executing right away.

On the other hand, if you would like to generate the executable program of your project in a hex format, you can simply use the 'Export…' command under the 'File' menu.

4) How to convert images and font files into files the Graphics Library can use?

   Microchip provides a utility that allows converting images in bitmap (BMP extension) format and JPEG (JPG or JPEG extension) as well as MS Window's installed fonts or True Type fonts (TTF extension) directly from files into formats to be used with Microchip Graphics Library. Bitmap and fonts are converted to a new optimized encoding for PIC microcontroller usage while the JPEG encoding of JPEG images are maintained. Fonts maybe copyrighted material so please ensure that you have the rights to use. You may find free fonts distributed under Open Font License (OFL) agreement.

   Launch the tool using:

   "Start -> Programs -> Microchip -> Graphics Library v2.10 ->Graphics Resource converter"

   - For converting image files click on "Add Images" in the tool. Once the file is loaded, verify the image dimension and the color depth.
   - For PIC24 targets, set the build option to C30. For PIC32 targets, set the build option to C32.
   - Click "Convert". Then type the file name for the converted file, and save the file as type "Array in Internal Flash (*.c)"

   The next step is to add the converted files into your application project. In the MPLAB project window, add the files to the project by right clicking source files, then select "Add Files…" from the drop down menu. Alternatively, you may select "Project -> Add Files to Project ..." from the menu bar.

<u>Checkout and report</u>:

Demonstrate your working program to the lab instructor and hand in a lab report that includes the following:

- Title page: course #, course title, Lab# & title, date, professor's name, names of group members
- This lab handout.
- Printout of the C program files of your main project source files and other relevant files you have created or modified from the demo project.