Laboratory #7
Graphics Object Layer of the Microchip Graphics Library

Objective: To get familiarization with the Graphics Object Layer of the Microchip Graphics Library. You will learn to use graphics widgets, interface user input devices such as touch screen, and process messages using callback functions.

Introduction:

The Microchip Graphics Library offers an Application Programming Interface (API) that performs rendering of primitive graphics objects as well as advanced widget-like objects. The library also facilitates easy integration of input devices through a messaging interface. The layered architectural design of the library makes it possible to easily change display devices if the need arises.

The Graphics Object Layer (GOL) renders the widgets, such as Button, Slider, Window, Check Box, Radio Button, Edit Box, List Box, Group Box, Horizontal/Vertical Scroll Bars, Progress Bar, Static Text, Picture, Dial, and Meter. To control these Objects, the Graphics Object Layer (GOL) has a message interface which accepts messages from the Application Layer. This interface supports a variety of input devices, such as keyboards, side buttons, touch screens, mice, etc.

Each widget has its own ***OBJCreate(…)*** function, where OBJ is object abbreviation (such as Btn for Button, St for Static Text, Pb for Progress Bar, etc). With this function the widget structure is created and populated with given parameters. The function also automatically places the widget into a global linked list and returns a pointer to the widget created. The code below shows an example on creating a Button:

Example 1:

```
#define     ID_BTN1     10
…
BtnCreate
(
        ID_BTN1,            // 2nd Button ID
        x1, y1,             // left, top
        x2, y2,             // right, bottom
        Radius,             // Rounded edges
        BTN_DRAW,           // Display button
        &arrow,             // use this bitmap
        NULL,               // no text
        altScheme           // style scheme – defines widget appearacne
);
```

You can find more information about the various functions of the library in the Graphics Library Help at ***Start -> Microchip -> Graphics Library v2.10 -> Graphics Library Help*** or at ***C:\Microchip Solutions\Microchip\Help***

The Style Scheme is a structure used by the library to define the appearance of a widget by assigning colors and fonts. The function ***GOLCreateScheme(…)*** is used to create the style scheme structure which contains 10 members. Multiple schemes may be defined. The different widgets may use the member values of the style scheme differently. If no style scheme is provides a default scheme is assigned upon widget creation.

To render the objects, the application should call a draw manager, ***GOLDraw()***.The function parses the active linked list of widgets and redraws the objects with the drawing states set. When the rendering is completed, drawing states of the objects are cleared automatically. The first created object will be drawn first. After all objects in the current linked list are drawn, the ***GOLDraw()*** calls the ***GOLDrawCallback()*** function. Custom drawing can be implemented in this function.

A callback function allows a lower level software layer to call a subroutine defined in a higher layer. It is used to add system or widget response to user inputs.

To handle the user interaction, the Microchip Graphics Library provides an interface to accept messages from input devices. Any input device event is sent to the library following the GOL message structure. The structure contains fields that identify the ID of the input device, the type of event (such as press or release of a button), and additional parameters that hold information that depends on the input device and type of event. For example, for events from touch screen the parameters contain the x and y coordinate position of the touch.

The ***GOLMsg(&msg)*** library message handler function is used to translate messages, find the affected widget in the linked list, modify widget state bits & perform some action. When it completes its operation the function returns TRUE. The user must supply a pointer to a message structure. Every widget has its own default action based on the translated message. These are described in the library help file. Custom actions on the input device events can be done in the ***GOLMsgCallback(...)*** function. This function is called by ***GOLMsg()*** each time a valid message for some object is received. The library then automatically redraws the object to show the change in state.

GOLMsgCallback(…) takes three parameters. The first parameter, *objMsg*, is translated message of the widget. The second parameter, *pObj*, is pointer to the object, and the third parameter, *pMsg*, is pointer to message structure. Return TRUE (a non-zero value) to perform default actions too. Return FALSE (a zero value) to skip default actions.

```
WORD GOLMsgCallback(WORD objMsg, OBJ_HEADER * pObj, GOL_MSG * pMsg)
```

Figure 1 shows a simple flow to use the Graphics Library. It is assumed that the user interface module and display drivers are chosen and added. For more information and examples on the use of the Graphics Object Layer (GOL) please refer to the help files and the Microchip Application Note AN1136 (How to Use Widgets in Microchip Graphics Library) available on Microchip website and on Blackboard.

The Example 2 code below shows the basic usage of the Microchip Graphics Library. The function ***GOLDraw()*** returns TRUE (a non-zero value) if the active linked list drawing is

complete. Do not call functions that capture and process input messages until the ***GOLDraw()*** function completes.

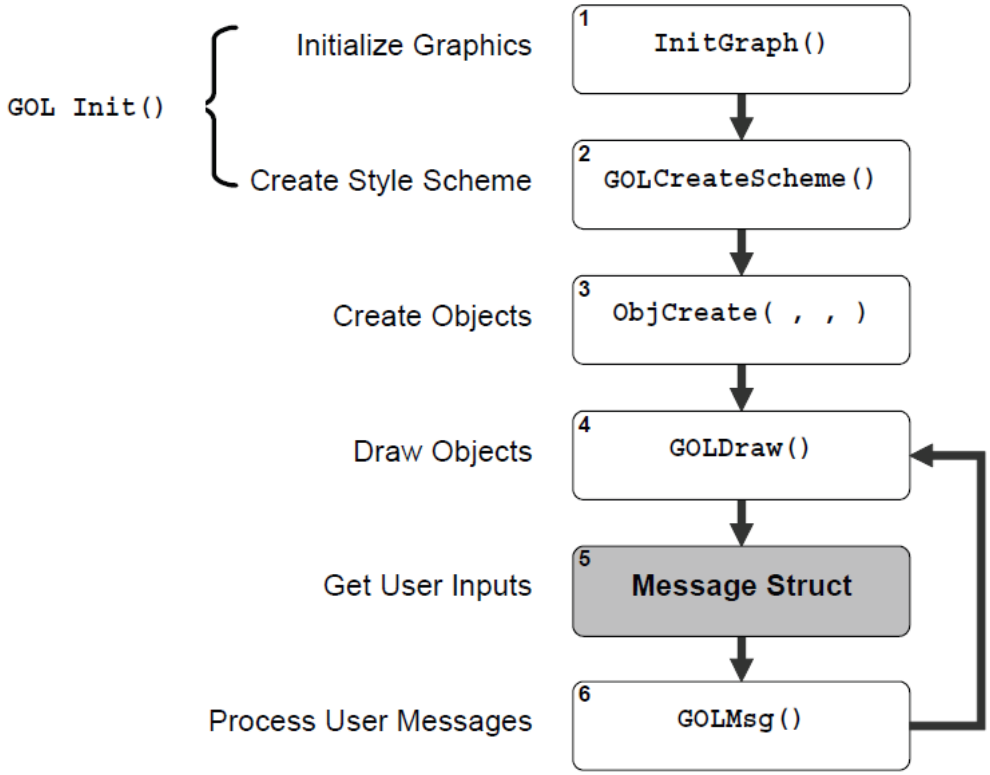

Figure 1: Basic Microchip Graphics Library usage flow

Example 2:

```
// Assume objects are created & states are set to draw objects
while(1){
   if(GOLDraw()){              // parse active list and redraw objects that need to be redrawn
      // here GOL drawing is completed
      // it is safe to modify objects states and linked list
      TouchGetMsg(&msg);     // evaluate messages from touch screen device
      GOLMsg(&msg);          // evaluate each object affected by the message
   }
}
```

For complete usage examples of the Graphics Library you are strongly advised to refer to the AN1136 document, the help files, and the demo programs that come with the installation of the library. Specifically refer to the "Graphics Object Layer Demo", "Graphics AN1136", and "Graphics AN1182" demo projects.

Task:

Create an application that implements a simple calculator. The calculator should support operations that can perform addition (+), subtraction (-), multiplication (x) and division (/) on two integer numbers of up to 10 digits each. The touch screen graphics display is used for rendering a picture of the calculator and accepting user inputs for its operation. Numbers as they get entered and results of operations will appear in the *edit box* on the top row of the calculator. The keys for entering numbers, operator signs, a backspace key and the equal symbol appear on a 4x4 keypad underneath the edit box.

Since the goal of this exercise is primarily to learn how to use the object layer of the graphics library for creating widgets on the graphics display and interacting with users using the touch screen interface, you should not care too much about handling errors in the syntax of the calculator input or overflow conditions. Assume the user enters correct syntax for the calculator.

Hint: Refer to the source code of the Graphics Object Layer Demo which demonstrates the use of Edit Box and Button objects in a phone pad application.

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

Checkout and report:

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.