19039 GFX2

Creating Graphical Applications Using MPLAB® Harmony

Hands–On

Lab Manual

Instructors: Ryan Rogerson & Bill Li
Microchip Technology Inc.

MICROCHIP

MASTERs Conference

Creating Graphical Applications Using MPLAB® Harmony

Table of Contents

This page intentionally blank

Hardware

Special Instructions for Hardware Setup

Hardware Description

All of the labs included in this manual are designed to use the PIC32MZ EF Starter Kit and the Multimedia Expansion Board II (MEB-II). This section gives a brief description of those development boards and how to set them up for use in this class.

1 The PIC32MZ EF Starter Kit

The PIC32MZ EF Starter Kit has a 200MHz PIC32MZ2048EFx144 microcontroller on-board with 2MB Flash, 512 KB RAM and Floating Point Unit. The board features two Mini-B USB ports, one for debugging and one for USB-to-UART or I2C™ communications, a Standard-A USB port for the embedded host, and a Micro-AB USB Host/Device/OTG port. The PHY daughter board features an RJ-45 Ethernet port for network connectivity.
The PIC32MZ EF Starter Kit uses the LCC driver (included in the GFX library) to drive the MEB II display without the need for a separate graphics controller.

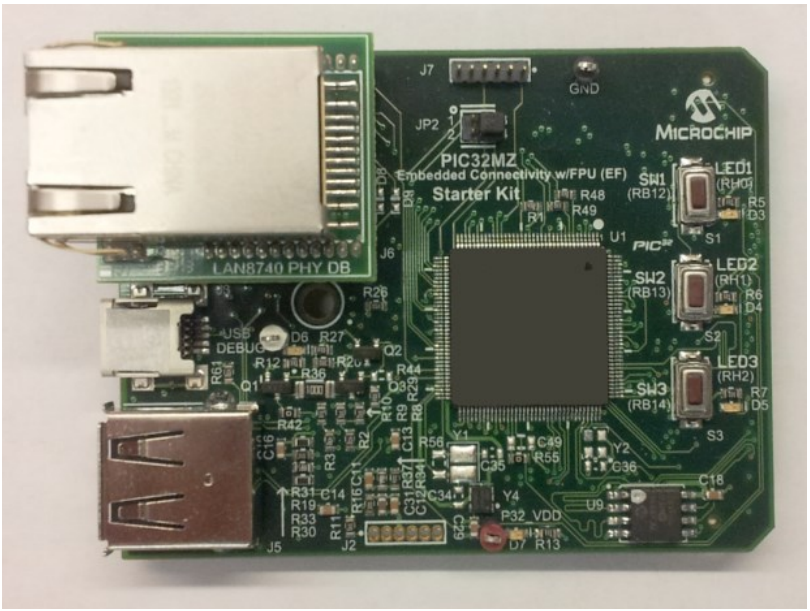


Figure H.1 PIC32MZ EF Starter Kit *(DM320007)*

2 The MEB-II display board

The MEB-II Multimedia Expansion Board includes a 24-bit stereo audio codec, integrated 802.11 b/g wireless module, low-cost Bluetooth HCI transceiver, optional EBI SRAM memory, 4.3” WQVGA PCAP touch display daughter board, microSD slot, mTouch sensing solutions buttons, analog temperature sensor and a VGA camera.

Figure H.2
MEB-II (DM320005-2)

3 Set jumper J9 on MEB II board

The MEB II has two memory options External or Internal. For this lab, we will setup the MEB II to run in Internal mode. Make sure EBIWE and LCD_PCLK are short-circuited with J9 jumper

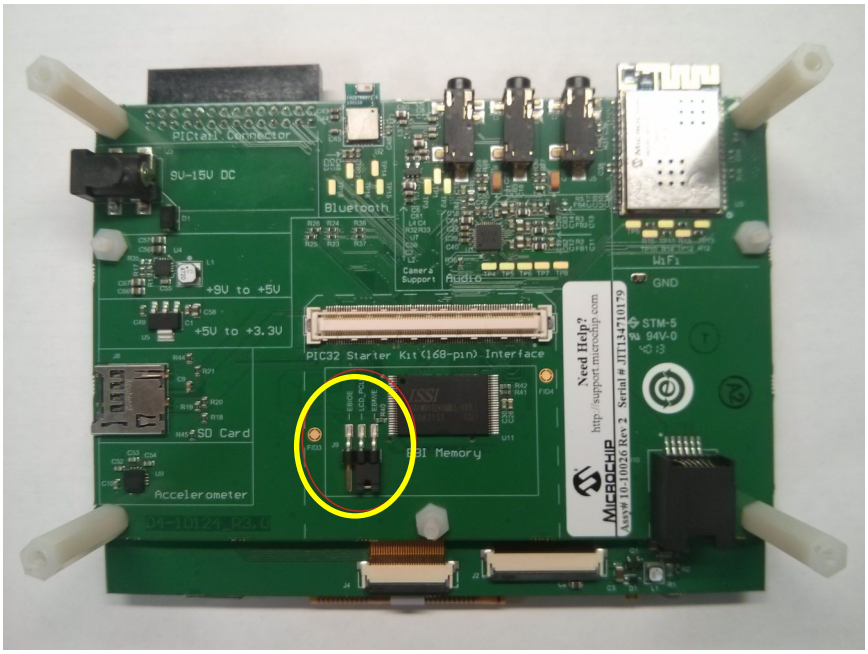


Figure H.3
Jumper J9 at the back of MEB-II

4 Set up the hardware development platform

As shown in Figure H.4, on the back of the MEB-II:
1) Plug the PIC32MZ EF Starter kit onto the 168-pin Hirose receptacle
2) Connect the power supply to the 9V-15V DC receptacle
3) Attach the Real ICE RJ-12 connector

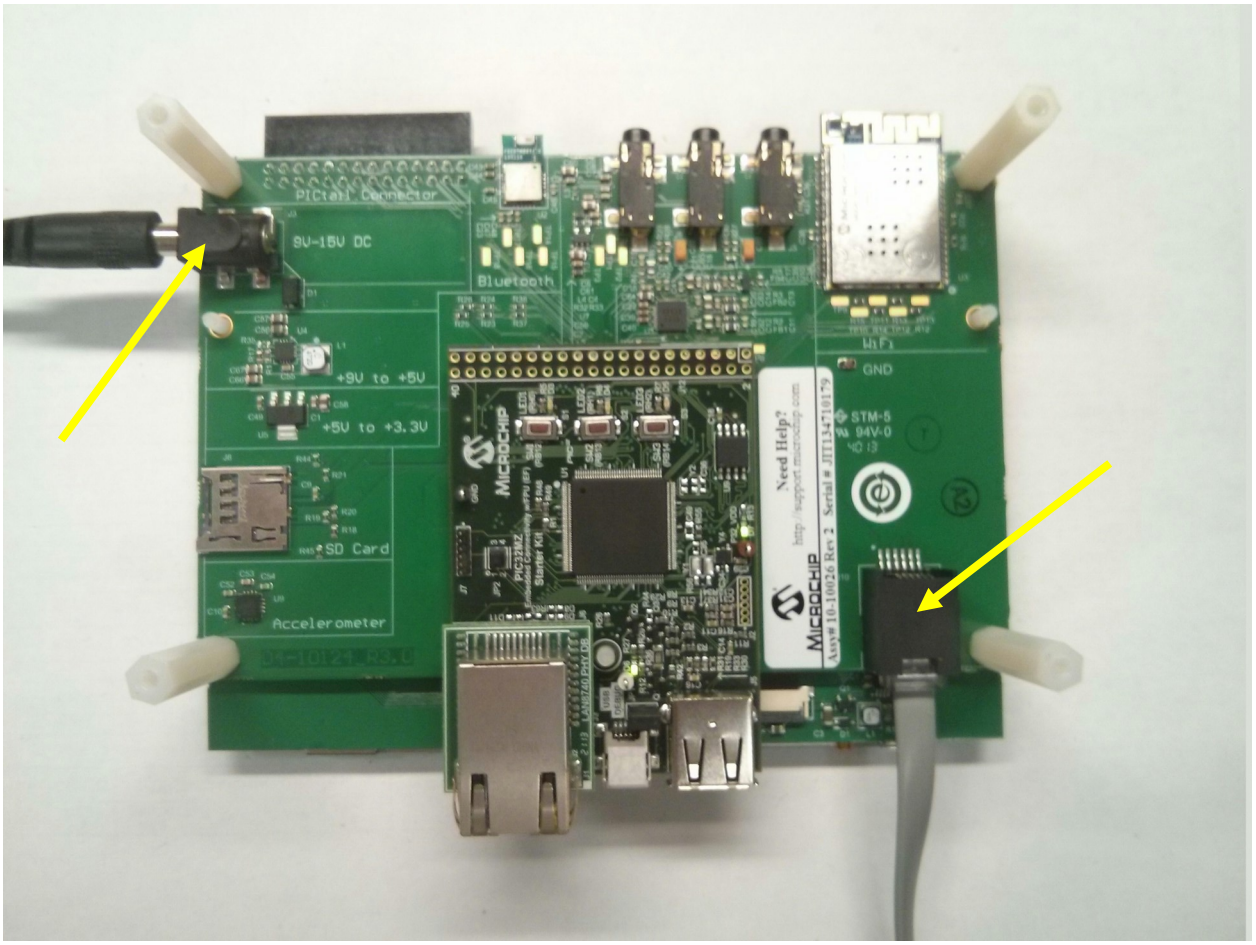


Figure H.4
MEB-II with PIC32MZ EF Starter Kit, power supply and Real ICE attached

Software

Description of Software Setup

Software Description

All of the labs included in this manual are designed to using MPLAB®X IDE v3.06 with MPLAB® Harmony Configurator (MHC) Plug-in 1.06.12 installed

Harmony framework version 1.06 is installed at **C:\microchip\harmony\v1_06**

Lab 1

Getting Started

Objective

Using the MPLAB® Harmony Configurator, you will create and configure a new project from scratch that will display a background color on the MEB-II LCD display.

Solution files may be found in:
C:\Microchip\harmony\v1_06\apps\masters\solutions\19039_GFX2

Tools

- MPLAB®X IDE v3.06 or higher with MPLAB® Harmony Configurator (MHC) Plug-in 1.06.12 installed
- PIC32MZ EF Starter Kit
- Multimedia Expansion Board II
- Real-Ice debugger
- 9V to 15V DC power supply
- XC32 Compiler v1.40

Procedure

1) Create a new Harmony project for the PIC32MZ2048EFM144 in MPLAB®X IDE
2) Use the MPLAB® Harmony Configurator to configure the project to for displaying graphics on the MEB-II LCD display
3) Use the MPLAB® Harmony Graphics Composer to display one screen with one background color.

Expected Results

At the completion of this lab, you are expected to see the LCD display of the MEB-II displaying a color you have picked as the background in the MPLAB® Harmony Graphics Composer.

Step-by-Step

1. **Launch MPLAB® X. Go to File > New Project. You will see the following window. Select Microchip Embedded > MPLAB Harmony Project, press Next**

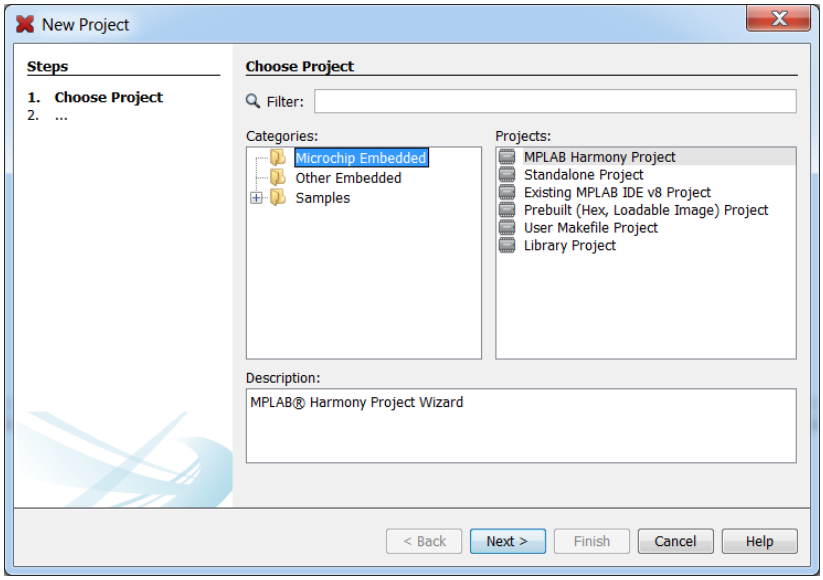


Figure 1.1
MPLAB® X New Project Window: Choose Project

2. **At the New Project window, make sure the following fields are as follows :**
   a) **Harmony Path: c:\microchip\harmony\v1_06**
   b) **Project Locations: c:\microchip\harmony\v1_06\apps\masters**
   c) **Project Name: 19039_GFX2**
   d) **Configuration Name: default**
   e) **Target Device: PIC32MZ2048EFM144**

Press Finish

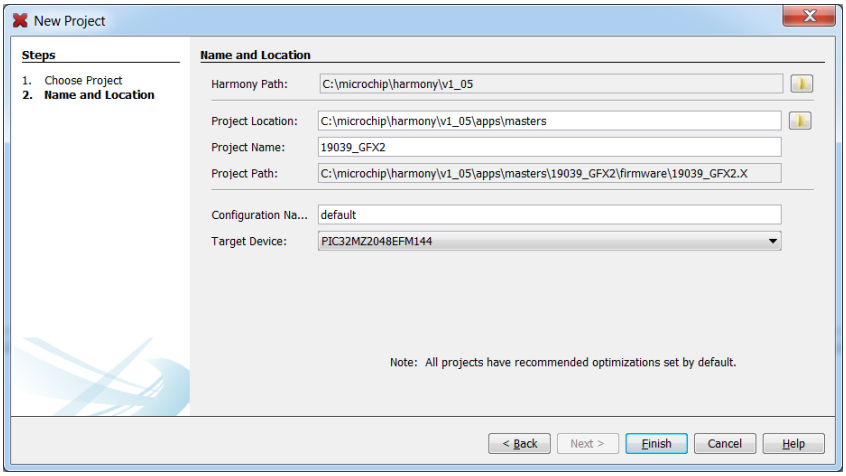


Figure 1.2
MPLAB® X New Project Window: Name and Location

3 **MPLAB® X will proceed to create the project, set the project as Main Project and Launch MPLAB® Harmony Configurator. This automated step will finish when you see the following in MPLAB® X.**

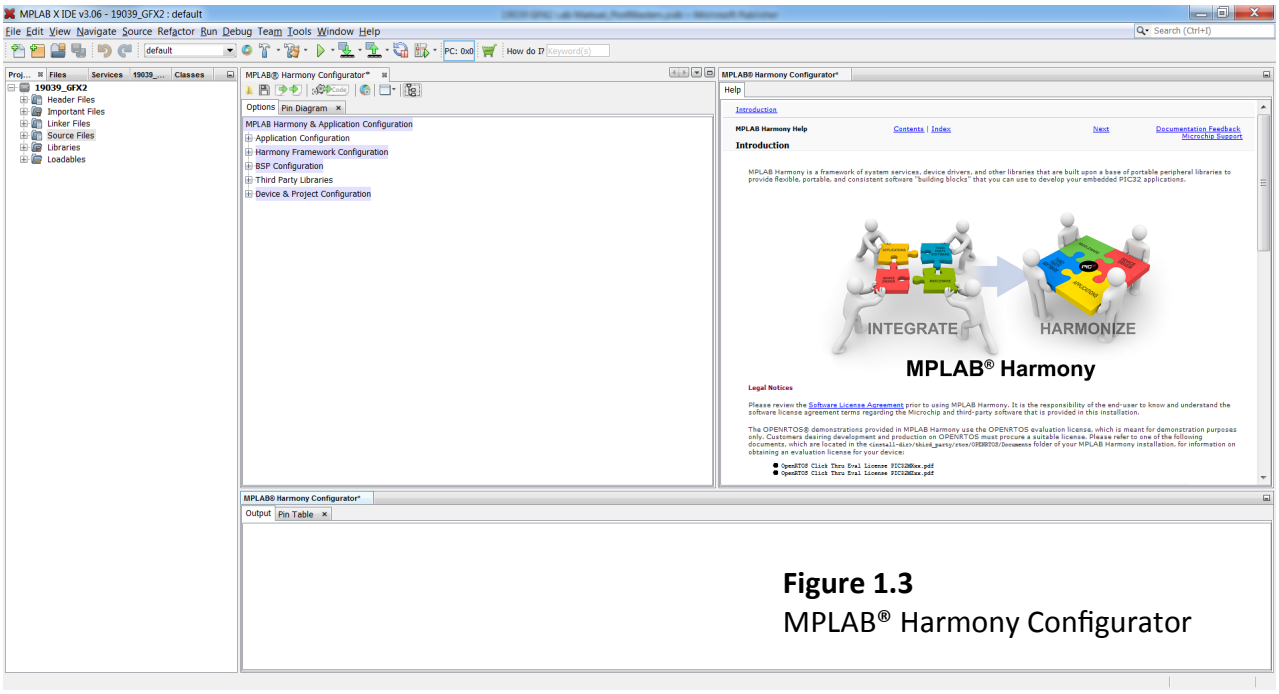


Figure 1.3
MPLAB® Harmony Configurator

4 **First, we will set the project properties. Right-click on the project name 19039_GFX2 and select Properties in the click menu.**

This will bring up the Project Properties pull-down menu

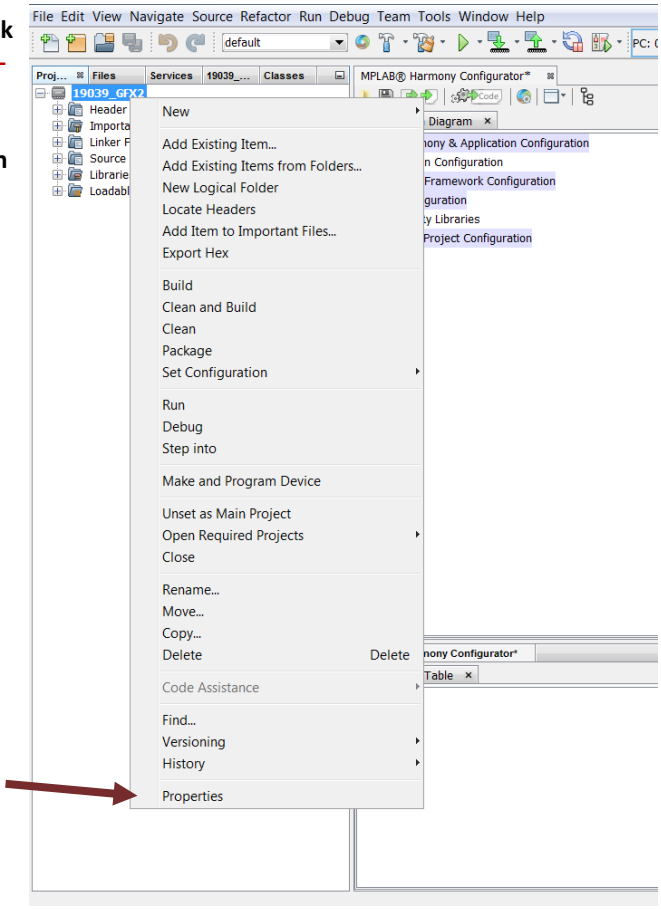


Figure 1.4
MPLAB® X Project Properties

5 **In the Project Properties window, make sure to select Real ICE as hardware tool and make sure the X32 compiler version selected is v1.40 or newer. Press OK.**

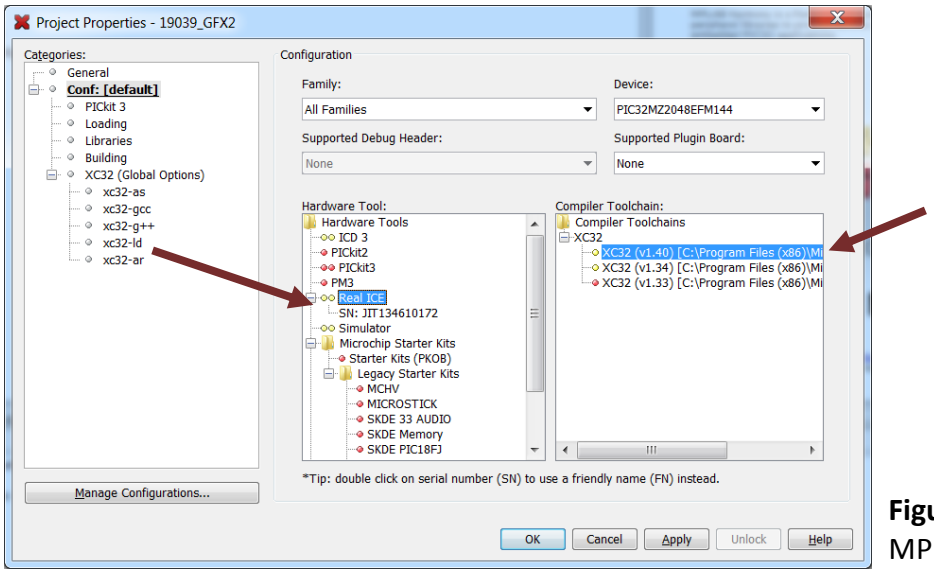


Figure 1.5
MPLAB® X Project Properties

6 **Next, we will use the MPLAB® Harmony Configurator to configure the project.**

In the tree view of the MPLAB® Harmony Configurator window tab, expand Device & Project Configuration and then expand PIC32MZ2048EFM144 Device Configuration, and select the following settings :

1) **DEVCFG1 -> Watchdog Timer Enable (FWDTEN): OFF**
2) **DEVCFG1 -> Deadman Timer Enable (FDMTEN): OFF**
3) **DEVCFG0 -> ICE\ICD Comm Channel: ICS_PGx2**

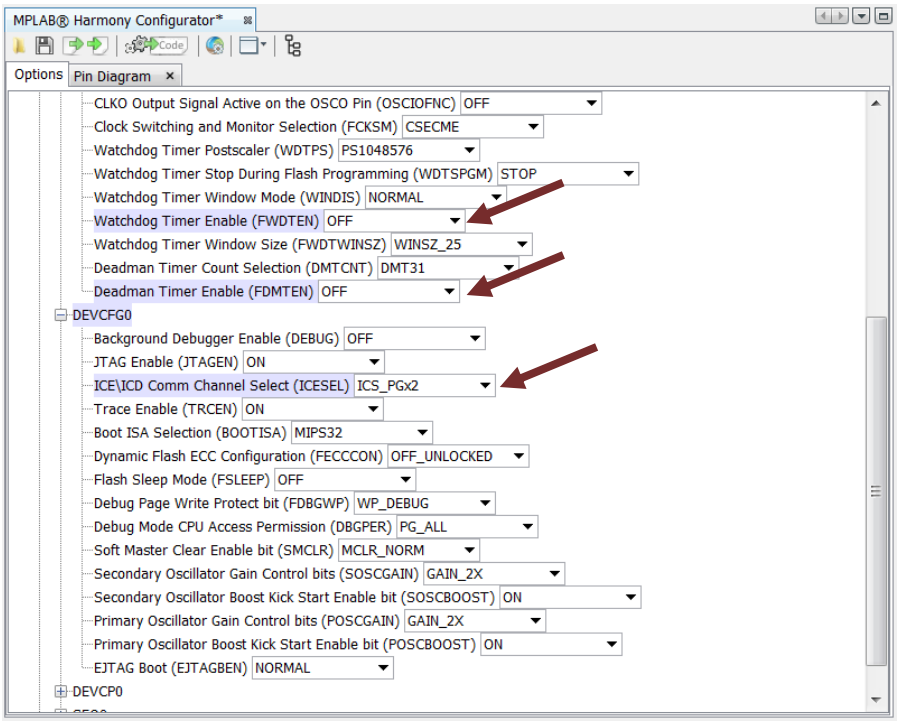


Figure 1.6
MHC DEVCFG settings

7 **Now we will configure the Oscillator module, using the Clock Diagram.**
Click on the Clock Diagram tab of the MPLAB® Harmony Configurator we select the following settings:

1) **FNOSC: SPLL**
2) **POSCMOD: EC**
3) **FPLLICLK: POSC**

Finally, we will configure the System Clock PLL using the Auto-Calculate button.

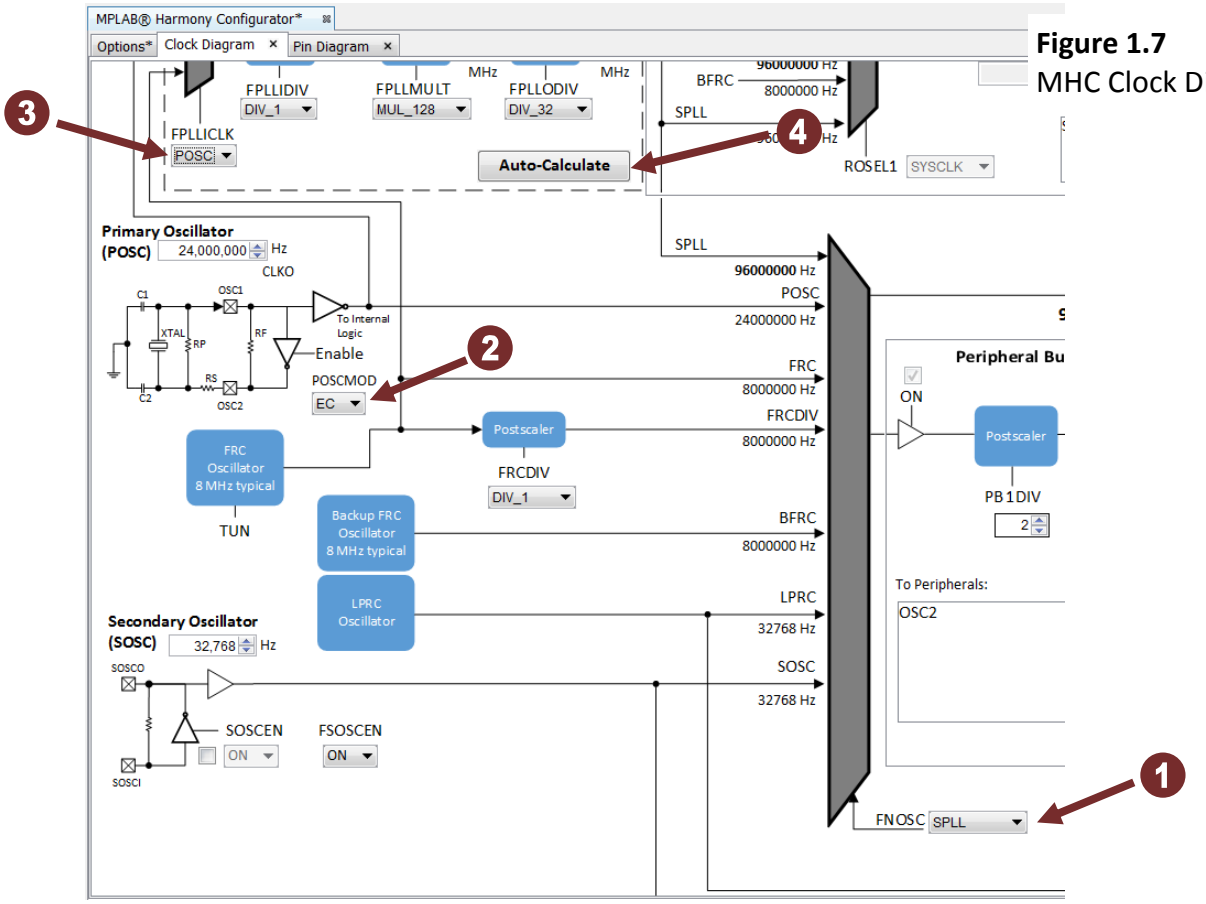
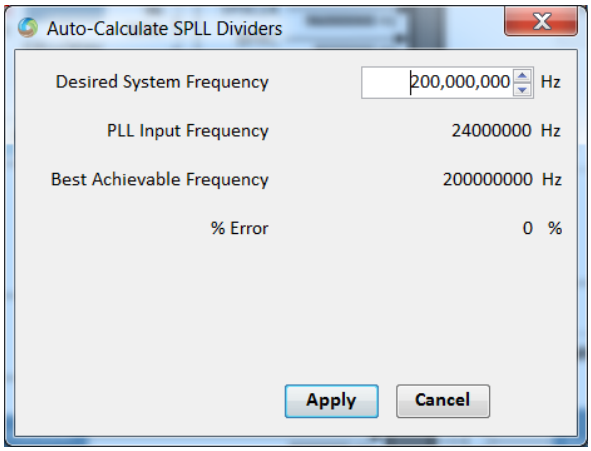


Figure 1.7
MHC Clock Diagram

8



The System Clock PLL (SPLL) Auto-Calculate feature takes the form of a pop-up window.

Confirm that the Desired System Frequency is set to 200 MHz. This will automatically sets the SPLL multiplier and divider to generate a System Clock of 200 MHz from the default input XTAL frequency of 24MHz.

Figure 1.8
MHC Clock SPLL Auto-Calculator

At this stage you may want to collapse the Device & Project Configuration tree under the Options tab.

9 **Back to the tree view (under Options tab), expand BSP Configuration, select Use BSP? and select the MEB-II for our Board Support Package (BSP) by checking the box PIC32MZ EF Starter Kit w\Multimedia Expansion Board**

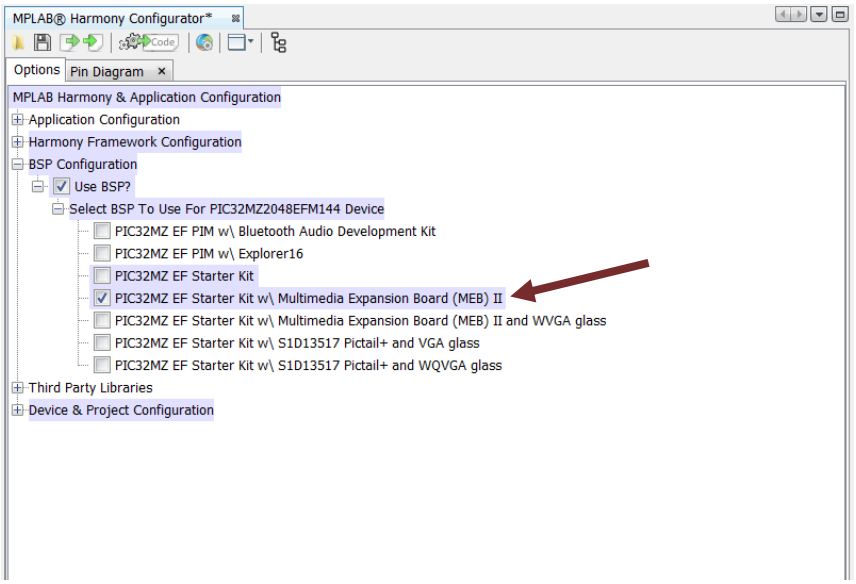


Figure 1.9 : MHC BSP Selection

At this stage you may want to collapse the BSP Configuration tree.

10 **In order to start MPLAB® Harmony Graphics Composer (MHGC) graphics design environment, press button to bring up the submenu. Select Graphics Composer.**

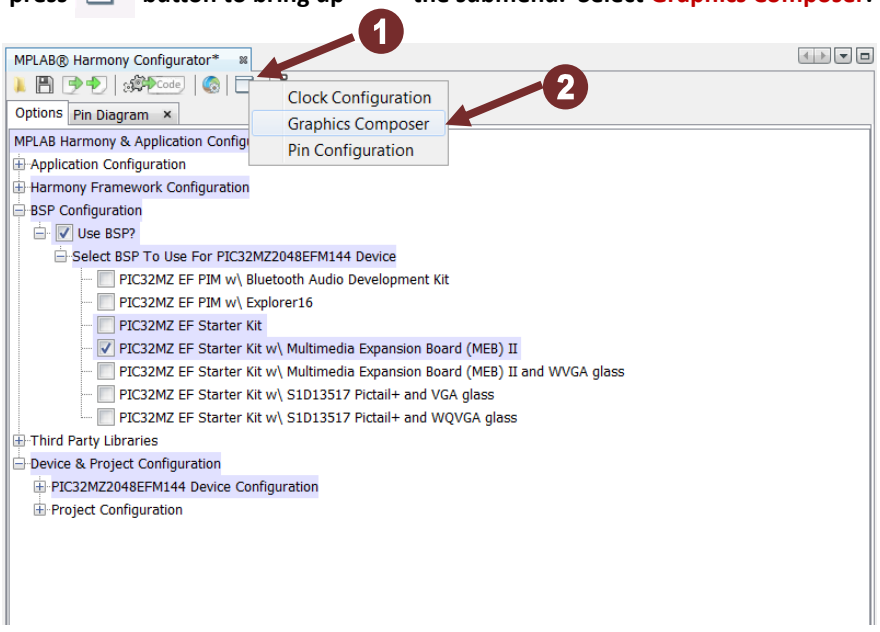


Figure 1.10 : MHC Select Graphics Library and Activate MHGC

11 **The MHGC is now launched in the Graphics Composer Screen window tab. When initially launched, the MHGC automatically creates an empty initial screen. You can change the color of the background by changing the RGB values in Graphics Composer Properties window, or you can click on the button to bring up the Color Picker Dialog (you may need to enlarge the Graphics Composer Properties window to see this button).**

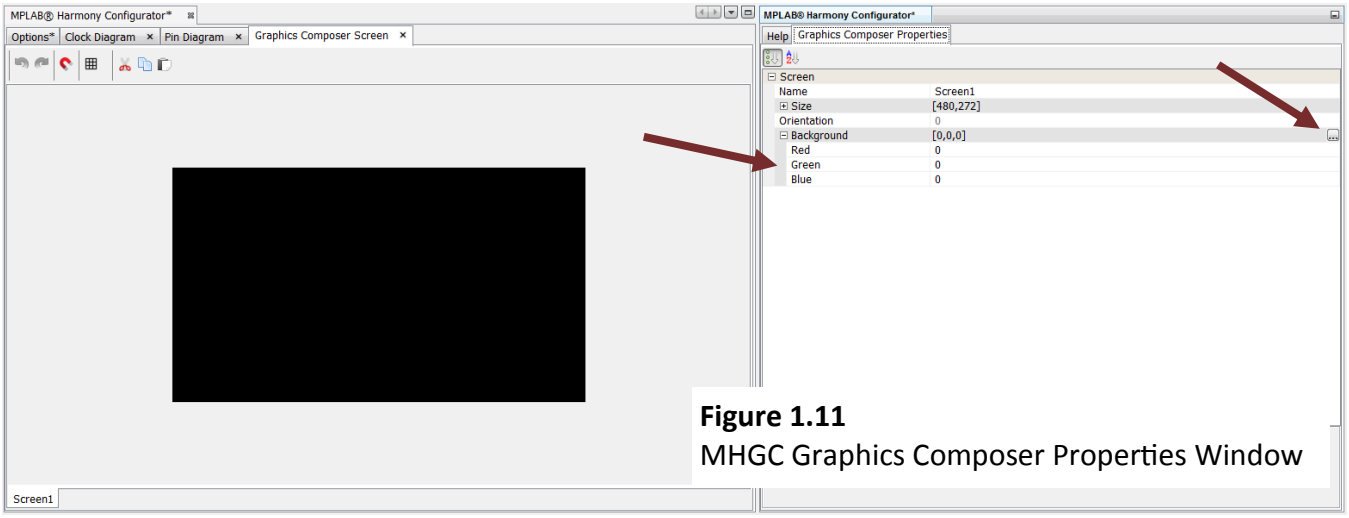


Figure 1.11
MHGC Graphics Composer Properties Window

12 **The Color Picker can be used to select a color either via the Color Wheel, one of the predetermined colors on the lower right or the manually enter RGB values.**
The Color Wheel is used in combination with the Brightness bar in the middle to determine the exact color to select.
The preview window on the top right gives a simulated preview of the color based on the color depth setting selection. This color depth selection can be found in the MHC tree.

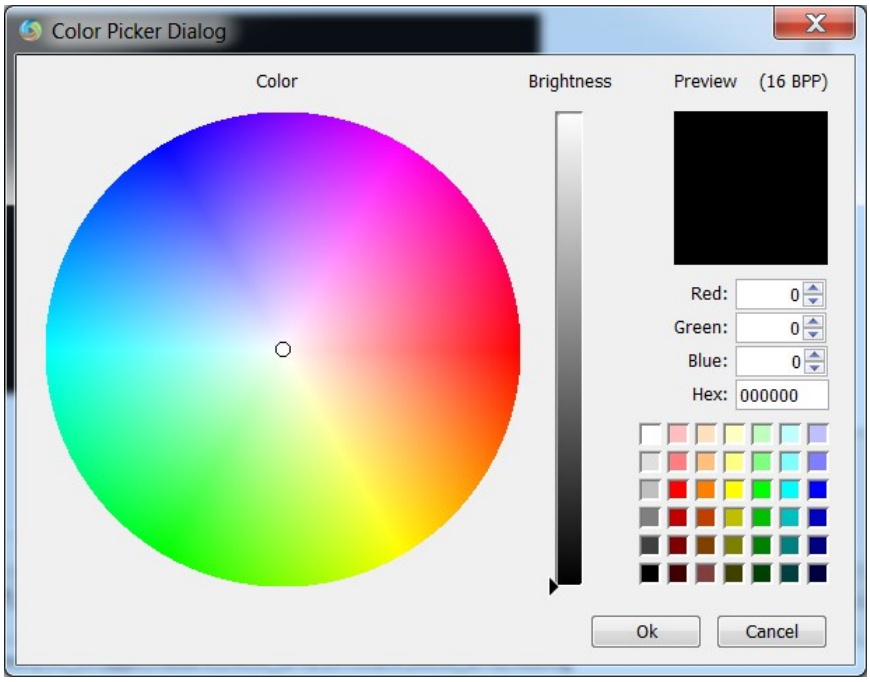


Figure 1.12
MHGC Color Picker Dialog

13 **Once you have selected a color and see the color change on screen preview, we can now generate the project.**

Go back to Options tab, press Generate, Save, Generate and finally compile and flash the program by pressing 

You should see the LCD display on the MEB-II displaying the color you had selected in the MHGC.

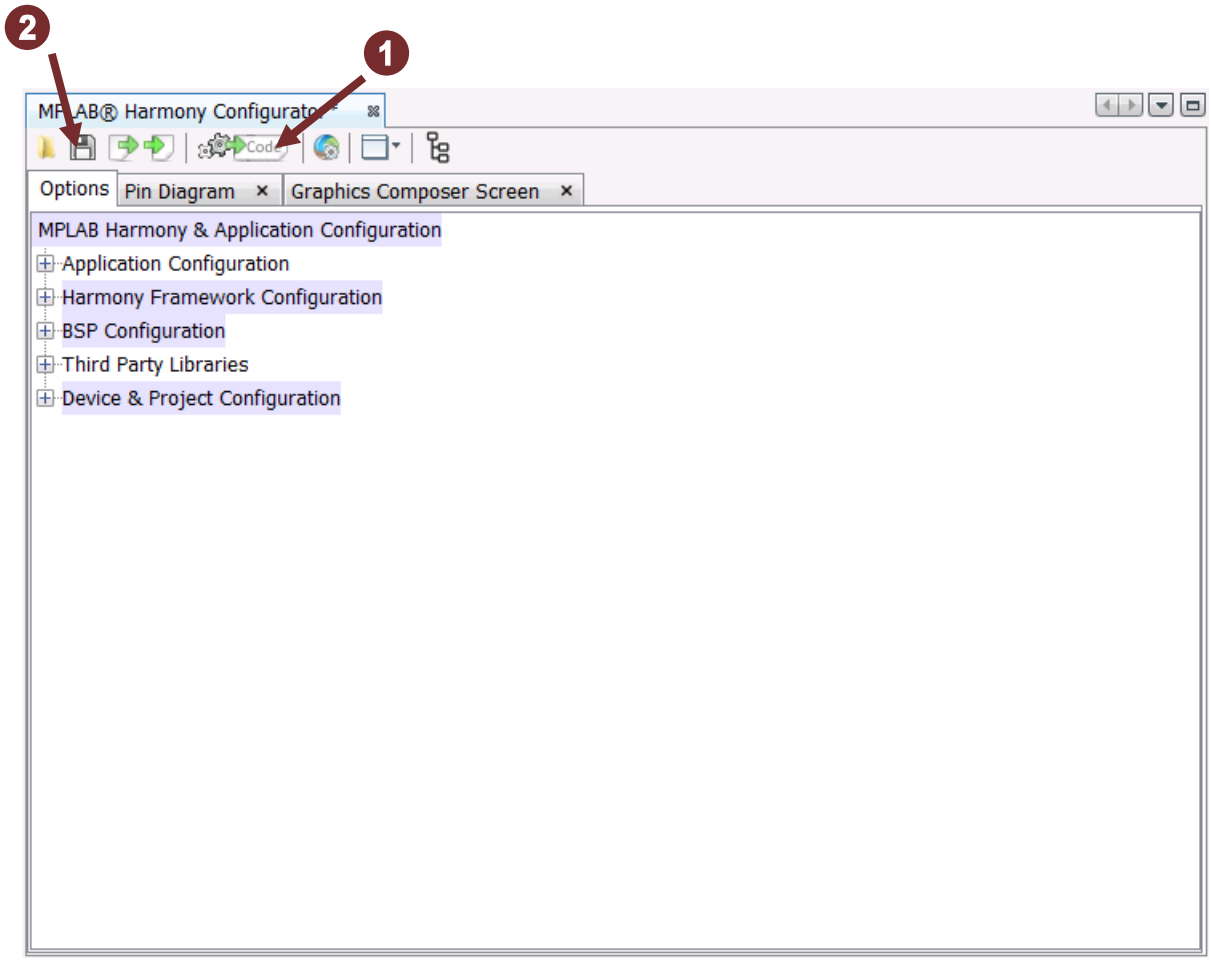


Figure 1.13
Generate, Build and Program

Results

You have just learned how to use MPLAB® Harmony Configurator and the MPLAB® Harmony Graphics Composer to create an project from scratch, configure it to the hardware, and enable the graphics library to render a color on the LCD display.

Conclusions

Having creating a basic graphics project, it is time to import some images and fonts to build the splash screen.

THIS PAGE INTENTIONALLY LEFT BLANK

Lab 2

Creating the Splash Screen

Purpose

Building on the project you have created in Lab 1, you will create a splash screen with images and font using the MPLAB® Harmony Graphics Composer.

Solution files may be found in:
C:\Microchip\harmony\v1_06 \apps\masters\solutions\19039_GFX2

Tools

- MPLAB®X IDE v3.06 or higher with MPLAB® Harmony Configurator (MHC) Plug-in 1.06.12 installed
- PIC32MZ EF Starter Kit
- Multimedia Expansion Board II
- Real-Ice
- 9V to 15V DC power supply
- XC32 Compiler v1.40

Procedure

1) Use the MPLAB® Harmony Graphics Composer tool to incorporate font and image source files.
2) Use the MPLAB® Harmony Graphics Composer tool, design the splash screen, including images, and text strings.

Expected Results

Figure 2.1 *When the lab is completed, you should see something like this on your LCD panel.*

Information

Font Files:
All fonts used in this lab have been preinstalled on the lab computers
Image Files:
C:\microchip\harmony\v1_06\apps\masters\resources

Step-by-step

In lab 2, we will create the splash screen shown in figure 2.1. The information box to the left is provided to help you navigate the lab directory.

Continuing from Lab 1, you will use the project you have created:
C:\Microchip\harmony\v1_06\apps\masters\19039_GFX2\firmware\19039_GFX2.X

1. **Launch MPLAB® X and open the 19039_GFX2.X project. Verify that the configuration selected is default in the configuration pulldown menu.**

   Launch MHC by press the MHC button. Another way to launch the MHC is at the menu bar, under Tools -> Embedded -> MPLAB® Harmony Configurator.

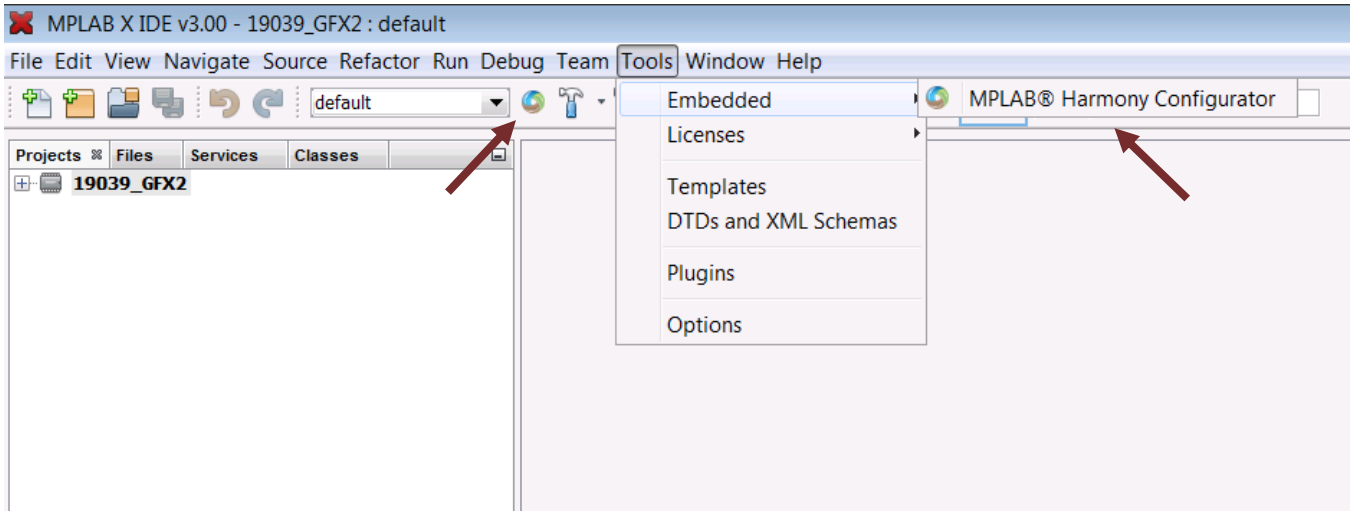


Figure 2.2 *Two ways to launch MHC*

2 **Launch MHGC : Press**  **button to bring up the submenu. Select Graphics Composer.**

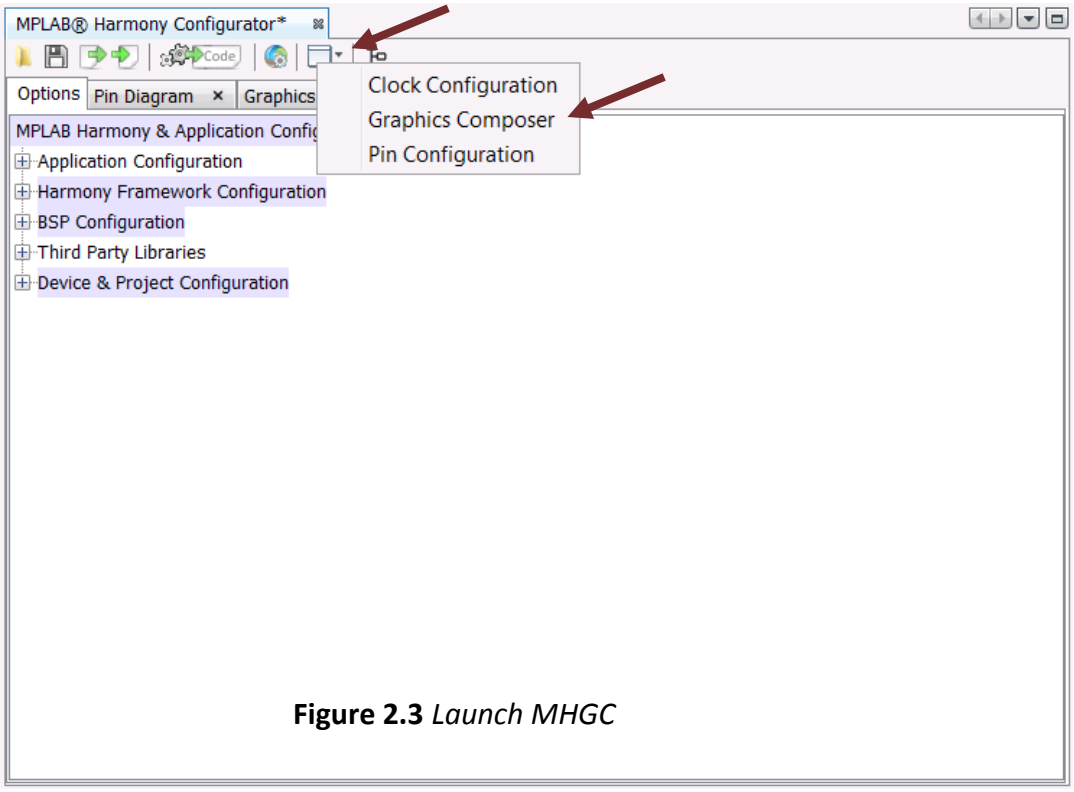


Figure 2.3 *Launch MHGC*

3 **In the MHGC GUI, in the Graphics Composer Properties window , expand the Background tree and set the background color to white by entering the values (255, 255, 255) to the RGB Properties**

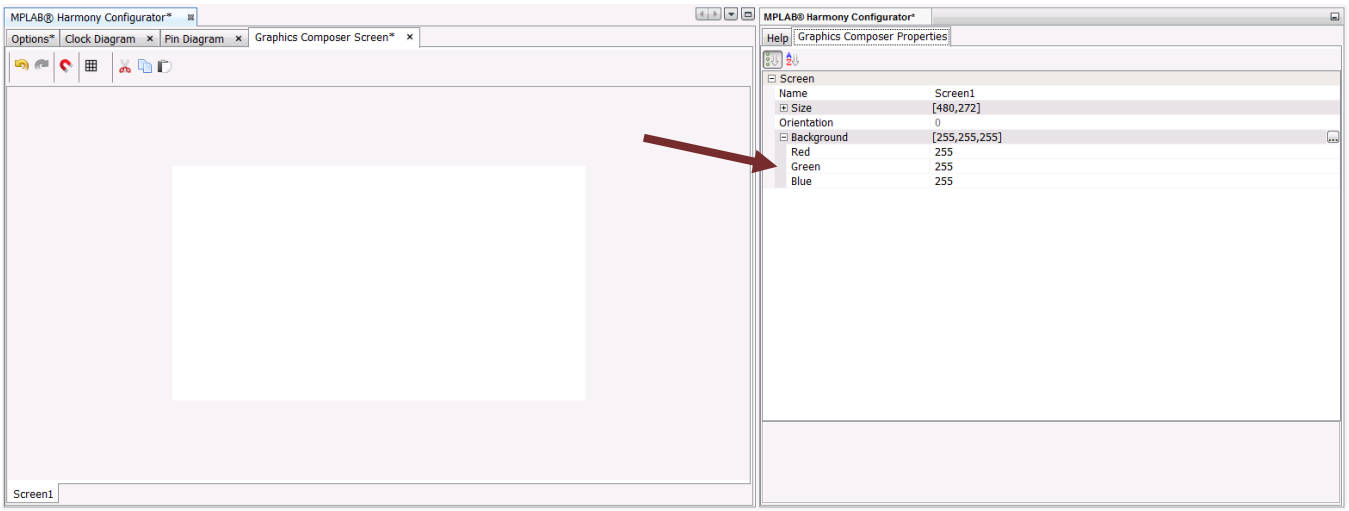


Figure 2.4 *Set background Colour to white*

4 **Next, we will import the image resources.**
Locate the Graphics Composer Management window at the bottom link corner of MPLAB X IDE.

1) **Select the Asset tab at the bottom of the Graphics Composer Management window, click on the Image button. This will bring up the Import Image dialog window.**
2) **Use the Browse button to navigate to C:\microchip\harmony\v1_06\apps\masters\resources\ to import the 10 image files in that folder. Once an image is selected, the Import Image dialog window will provide a preview of the image.**
3) **Press Import and the image will be added to an asset cache to be generated into the project.**

Repeat these steps until all 10 images are added to the project.

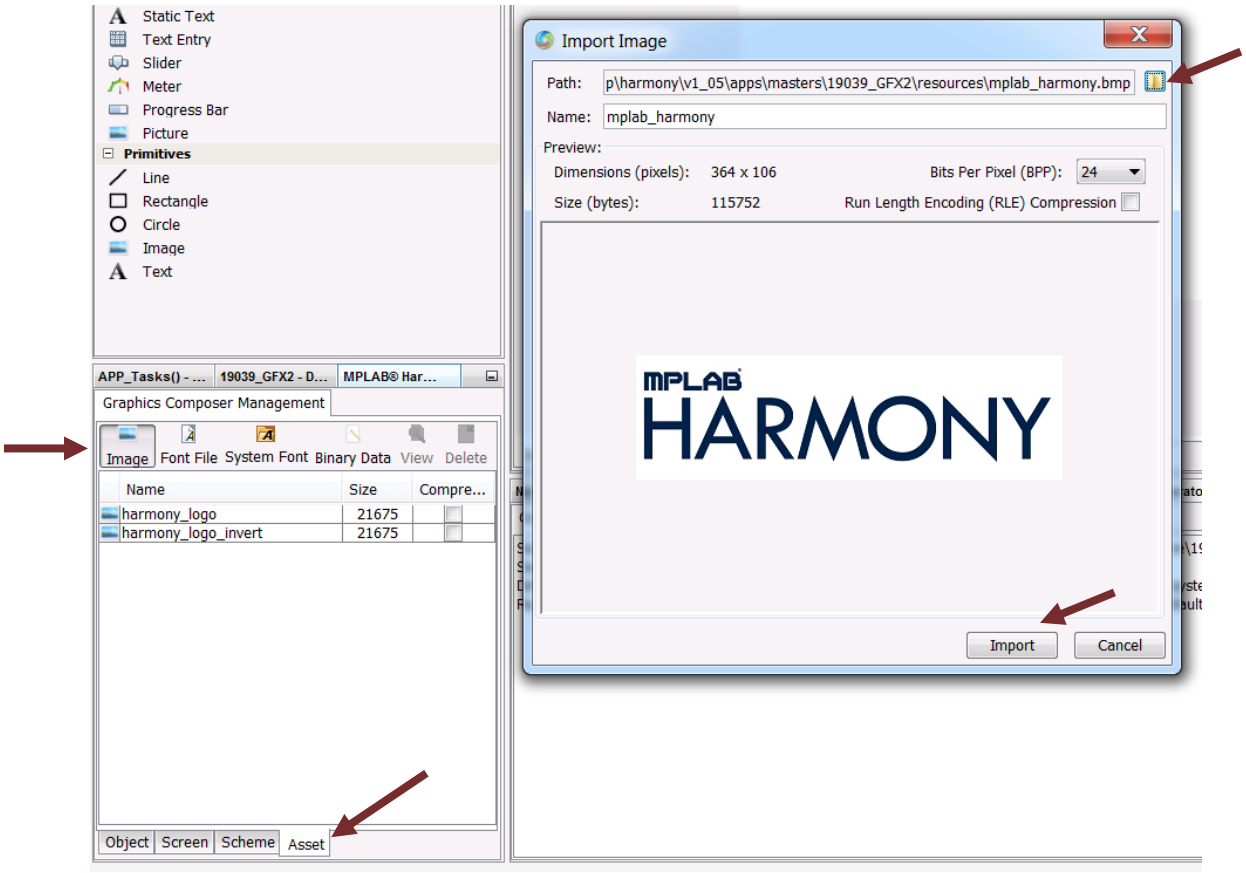


Figure 2.5 *Importing images*

5 **Next, we will import the font resources. There are two options for font resources. The MHGC can import fonts in .ttf file format or import fonts that are already installed in the operating system. For this lab, we will choose the latter.**

1) **Above Asset tab in the Graphics Composer Management window, click on the System Font button. This will bring up the Import System Font dialog window.**
2) **Choose the Arial font from the dropdown. Choose size 26. Note that the imported font is locked to a specific size. The dialog offers a preview of the font.**
3) **The name you choose for the font in the Name: field will be the name used in the generated C-code. Make sure to use alphanumeric characters with no spaces.**
4) **Press OK and the font will be added to an asset cache to be generated into the project.**

Repeat these step to import the Impact font at size 20.

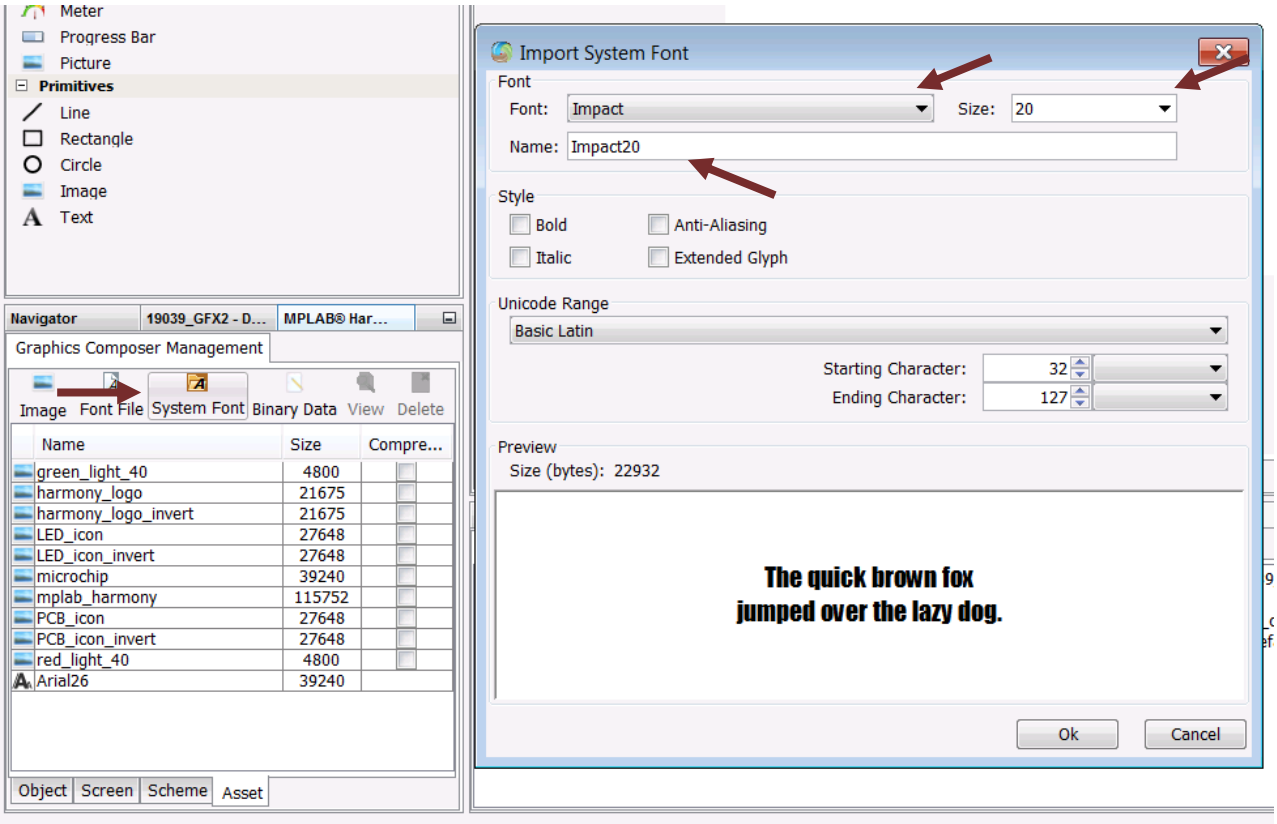


Figure 2.6 *Importing fonts*

6 **Now we are ready to build the splash screen. Let's add an image.**

1) **In the Graphics Composer Tool Box window, under Primitives tree, drag the Image icon onto the screen within the white area under the Graphics Composer Screen tab**
2) **At this point, you have only dragged in a box with dotted-line borders. Under the Graphics Composer Properties tab window, expand the Image tree, and select the mplab_harmony image in the dropdown. You can see that the image is now rendered on the middle screen window.**

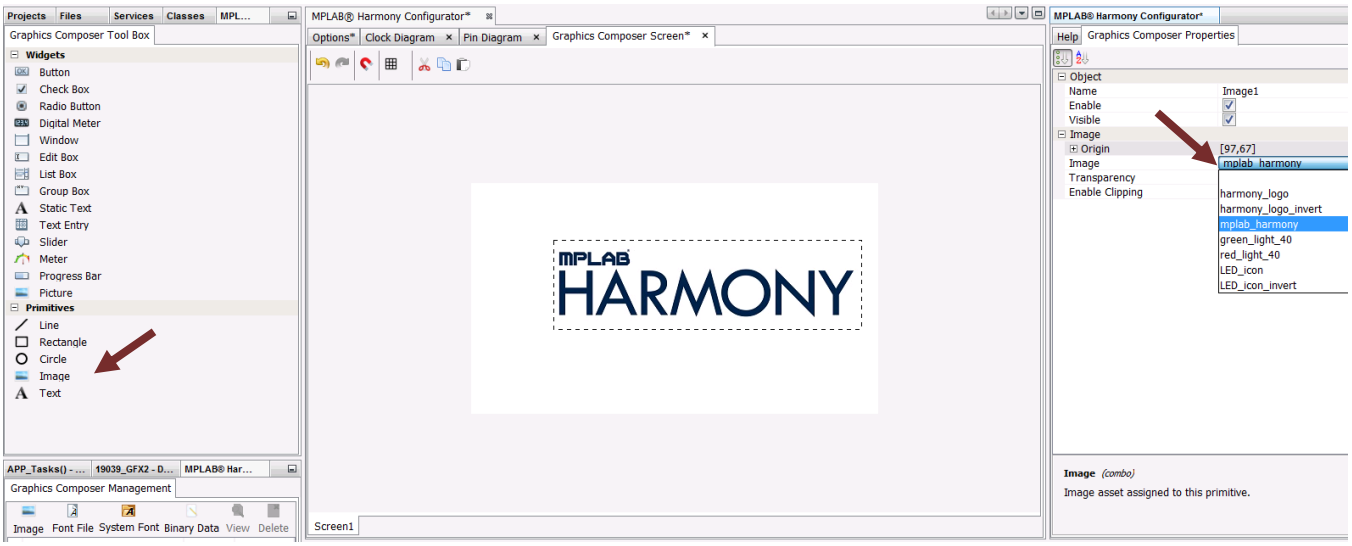


Figure 2.7 *Add an image*

7 **Next, we will put some text on the screen**

1) **In the Graphics Composer Tool Box window, under Primitives tree, drag the Text icon onto the white screen.**
2) **In the Graphics Composer Properties window, under Text tree > Text, enter "19039 GFX2" in the Text field**
3) **Under Text tree > Font, select the Arial26 font in the dropdown box**

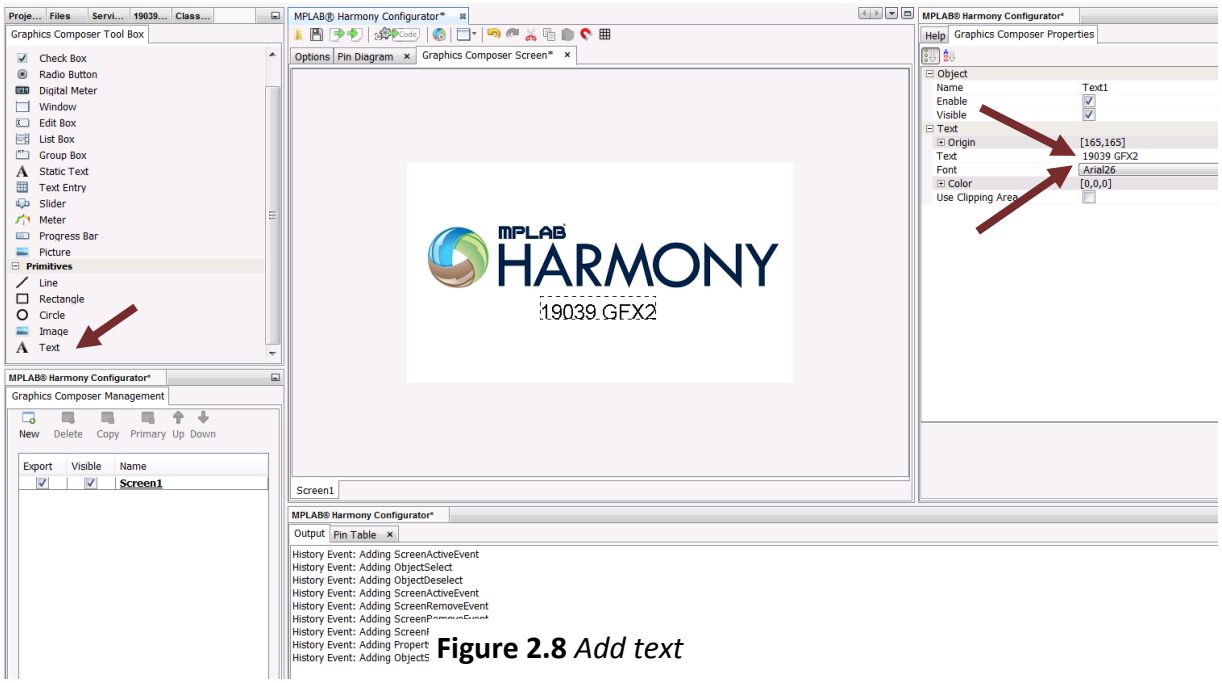


Figure 2.8 *Add text*

8 **Finally, let's add the Harmony logo. We will be using the logo both as an image and as an entry point to the next screen. So we will do something a little different. We will use a button widget to handle the logo.**

1) **In the Graphics Composer Tool Box window, under Widgets tree, drag the Button icon onto the screen. To the left of the HARMONY text.**
2) **In the Graphics Composer Properties window, under the Button tree, remove Button text from the Text field**
3) **Select Button Type to be Nopanel in the pulldown menu.**
4) **For Released Image, select "harmony_logo_invert" image**
5) **For Pressed Image, select "harmony_logo"**
6) **Resize the button so that the entire Harmony logo is visible**

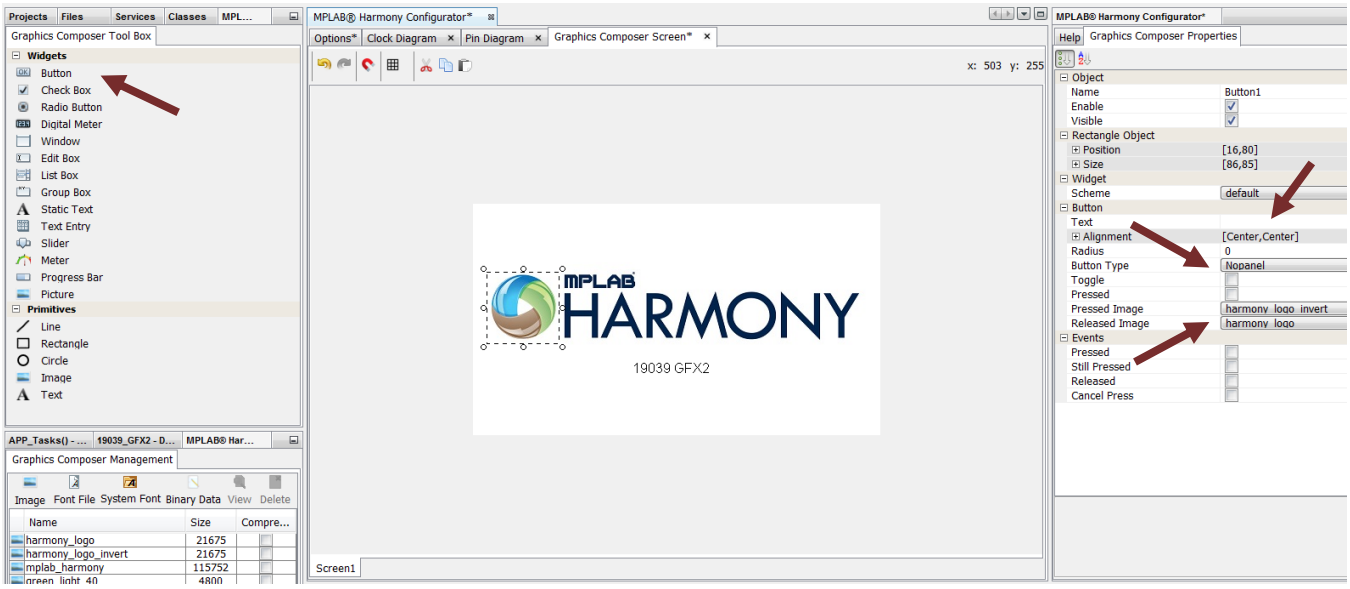


Figure 2.9 Adding the logo

9 **Now you are ready to generate, compile and flash the project :**

Go back to Options tab, press Generate, Save, Generate and finally compile and flash the program by pressing 

Note : Since we have not enabled touch drivers yet, the screen button is not interactive at this point. We will be enabling touch in the next lab.

Results

You have just learned how to use the MPLAB® Harmony Graphics Composer to import images and fonts. You have also learned how to use the MHGC to build a splash screen and render it on the LCD display.

Conclusions

Now that we have a splash screen, let's add touch and build an interactive menu.

THIS PAGE INTENTIONALLY LEFT BLANK

Lab Exercise 3

Creating the Interactive Menu Screen

Purpose

In this lab, you will enable touch to your project, create a menu screen with widgets and add events to the widgets.

Solution files may be found at:
C:\Microchip\harmony\v1_06\apps\masters\solutions\19039_GFX2

Tools

- MPLAB®X IDE v3.06 or higher with MPLAB® Harmony Configurator (MHC) Plug-in 1.06.12 installed
- PIC32MZ EF Starter Kit
- Multimedia Expansion Board II
- Real-Ice debugger
- 9V to 15V DC power supply
- XC32 Compiler v1.40

Objectives

1) Using the MHC, configure the project to enable touch system service and drivers
2) Use MPLAB® Harmony Graphics Composer (MHGC) event action feature to add a screen change event
3) Use MPLAB® Harmony Graphics Composer to create an Interactive Menu screen

Expected Results

When this lab is complete, you will be able to:

- Press the Harmony logo in the splash screen to move from to the menu screen
- The two buttons in the menu screen will be responsive to touch, but will not perform additional function

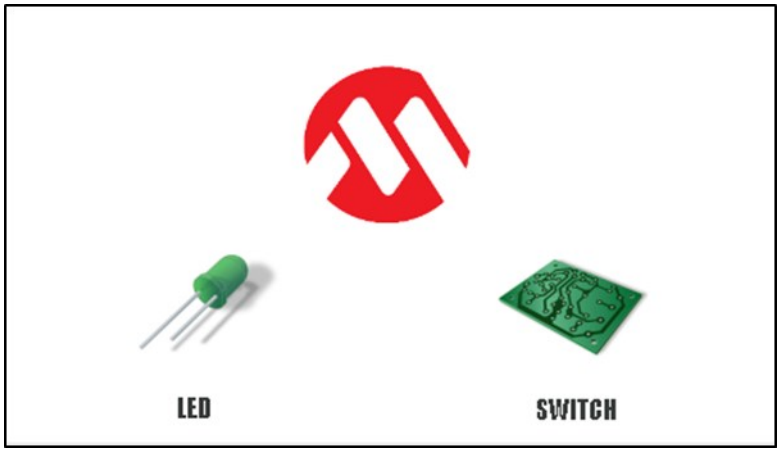


Figure 3.1 *When the lab is fully completed, you should see something like this on your LCD panel.*

1 **First, we need to enable the touch driver and system services.**

1) **Figure 3.2: under Options tab window, expand Harmony Framework Configuration tree, expand Use Graphics Library tree, check Use Input Devices checkbox and check Enable Touchscreen Support checkbox.**
2) **Figure 3.3: Expand Drivers tree, expand I2C tree, expand I2C Driver tree, check Interrupt Mode checkbox.**
3) **Figure 3.4: in the Harmony Framework Configuration tree, expand System Services tree, expand Interrupts tree, expand Use External Interrupts tree, expand External Interrupt Instance 0, for External Interrupt module ID, select INT_EXTERNAL_INT_SOURCE1 from pull-down menu.**
4) **Also in Figure 3.4: Set Interrupt Priority to INT_PRIORITY_LEVEL_5.**

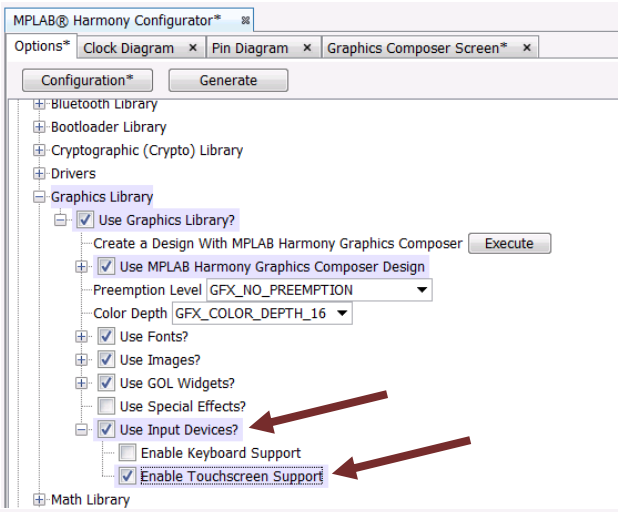
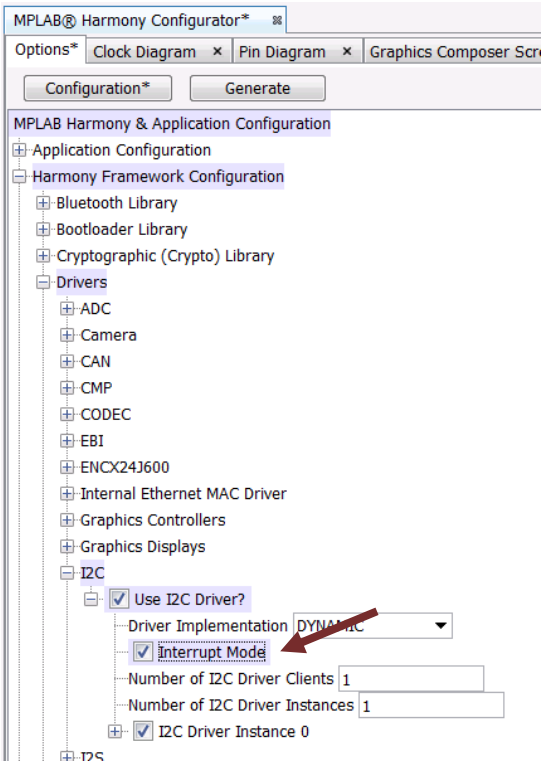
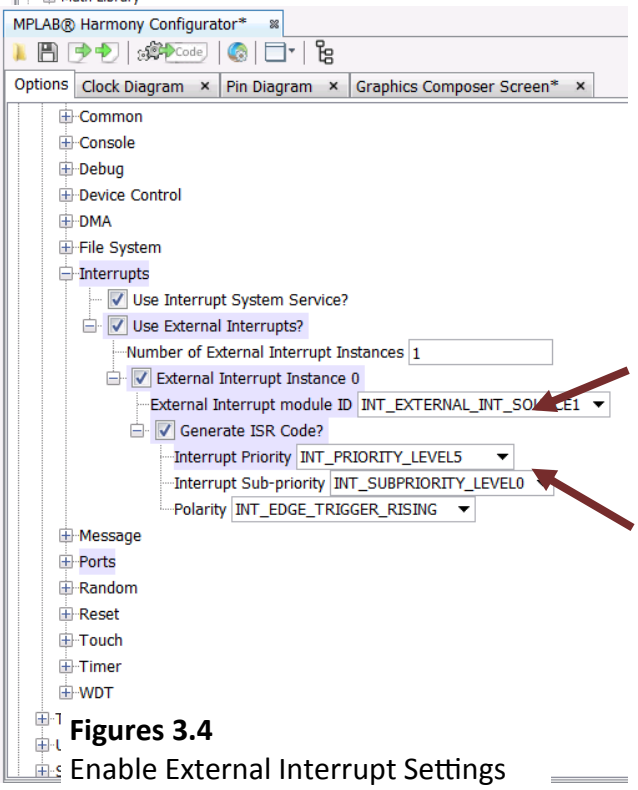


Figure 3.2
Enable Touchscreen Support



Figures 3.3
Enable I2C Interrupt Mode



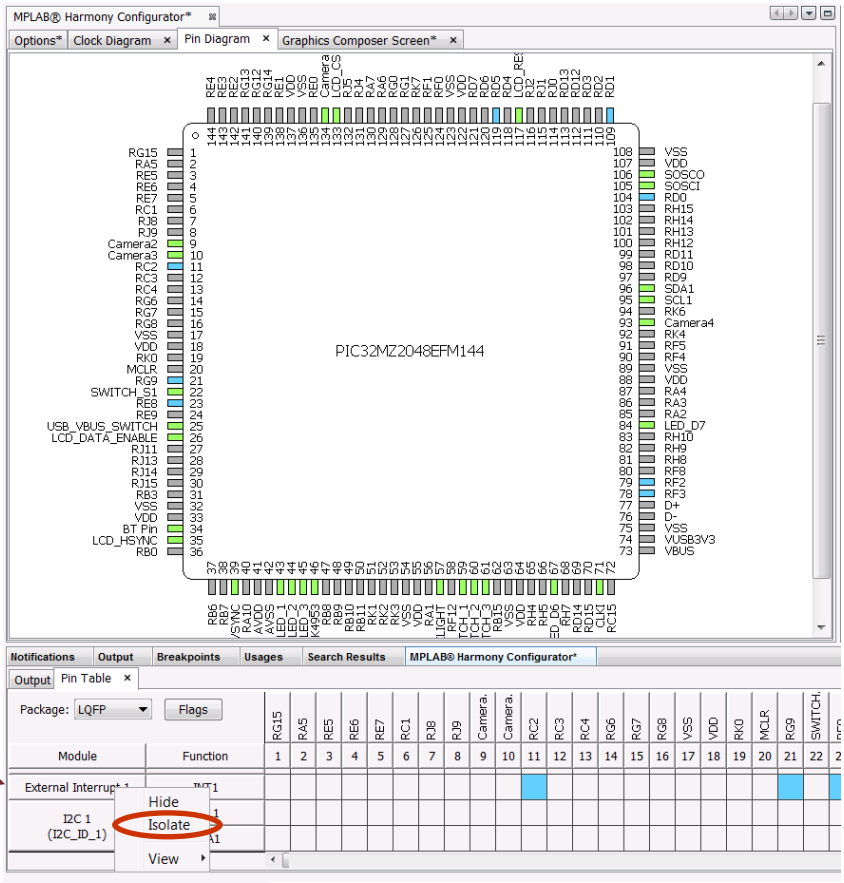
Figures 3.4
Enable External Interrupt Settings

2 **We need to map the external interrupt pin using the PPS.**

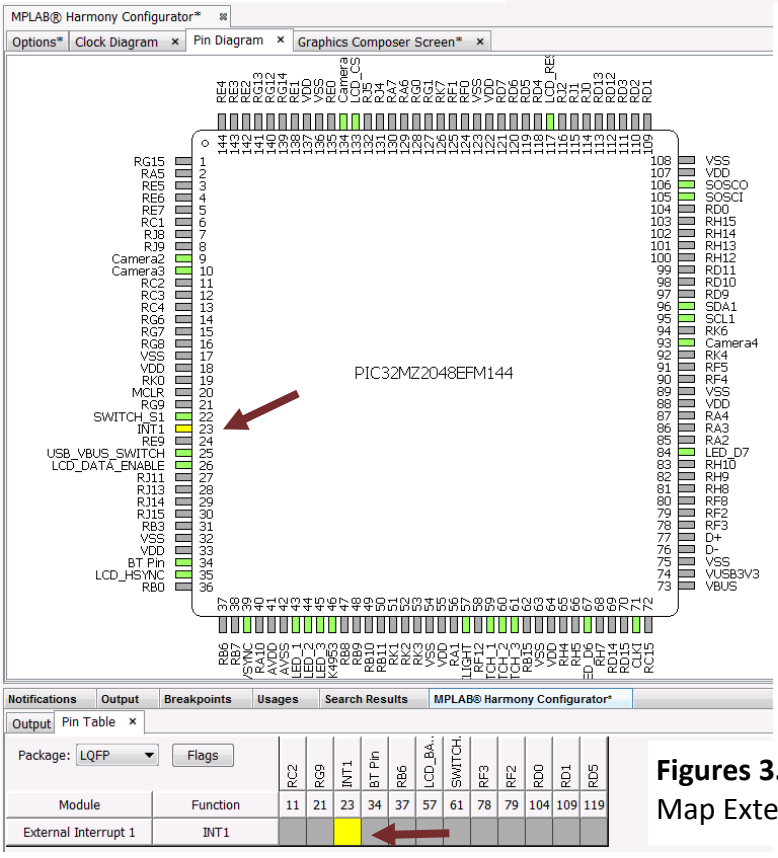
As shown in Figure 3.5, select the Pin Diagram tab and the Pin Table tab in their respective panels to see their windows

In the Pin Table window , note that there are blue squares in the row corresponding to the External Interrupt 1 module. These blue squares indicate all the viable pins, identically shown in blue in the Pin Diagram window.

Right-click External Interrupt and select Isolate.



Figures 3.5
Pin Diagram and Pin Table

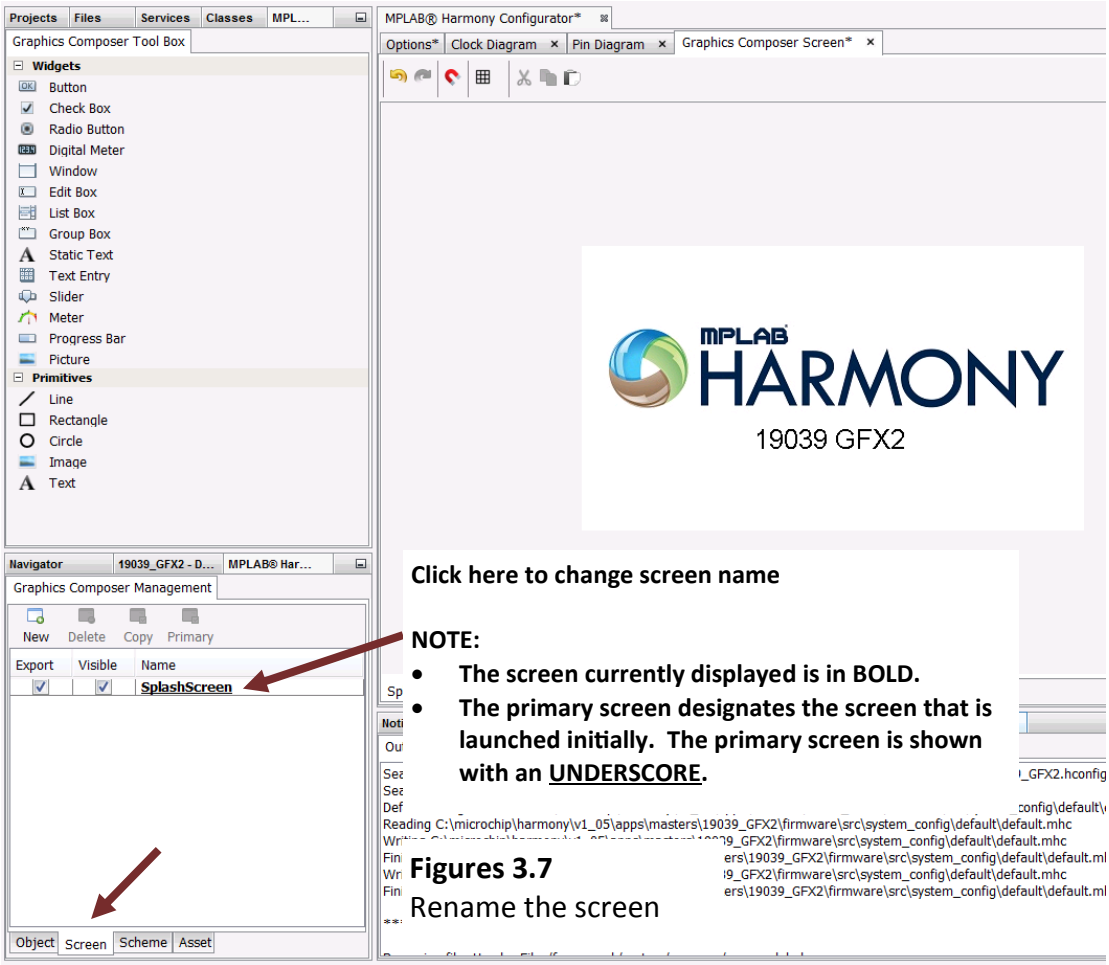


3 **Using the Isolate option, the Pin Table filters out all the pins not mappable by the PPS.**

The pin we want is RE8. Simply click on the RE8 square in the Pin Table and External Interrupt 1 signal will be mapped to that pin. The yellow square will turn green if you move away Note that the pin is renamed

Figures 3.6
Map External INT1 to RE8

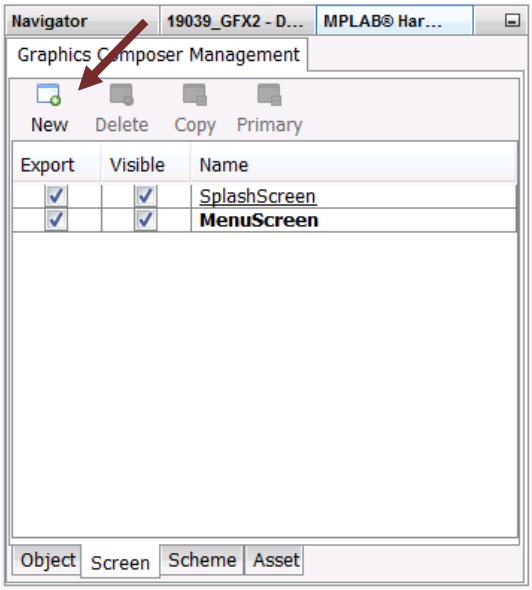
4 **Now, we can go back to working in the MHGC. We will need to create a new screen. First, bring up the MHGC windows (you may need to relaunch it by pushing Execute button as shown in Lab1 step10. Let's take the opportunity to rename the first screen. In the Graphics Composer Management window, click on the bottom Screen tab. Rename "Untitled" to "SplashScreen" by clicking on the name and changing it.**



Figures 3.7
Rename the screen

5 **Again In the Graphics Composer Management window, click on the New icon to create a new screen.**

We will rename the screen from "screen1" to "MenuScreen".

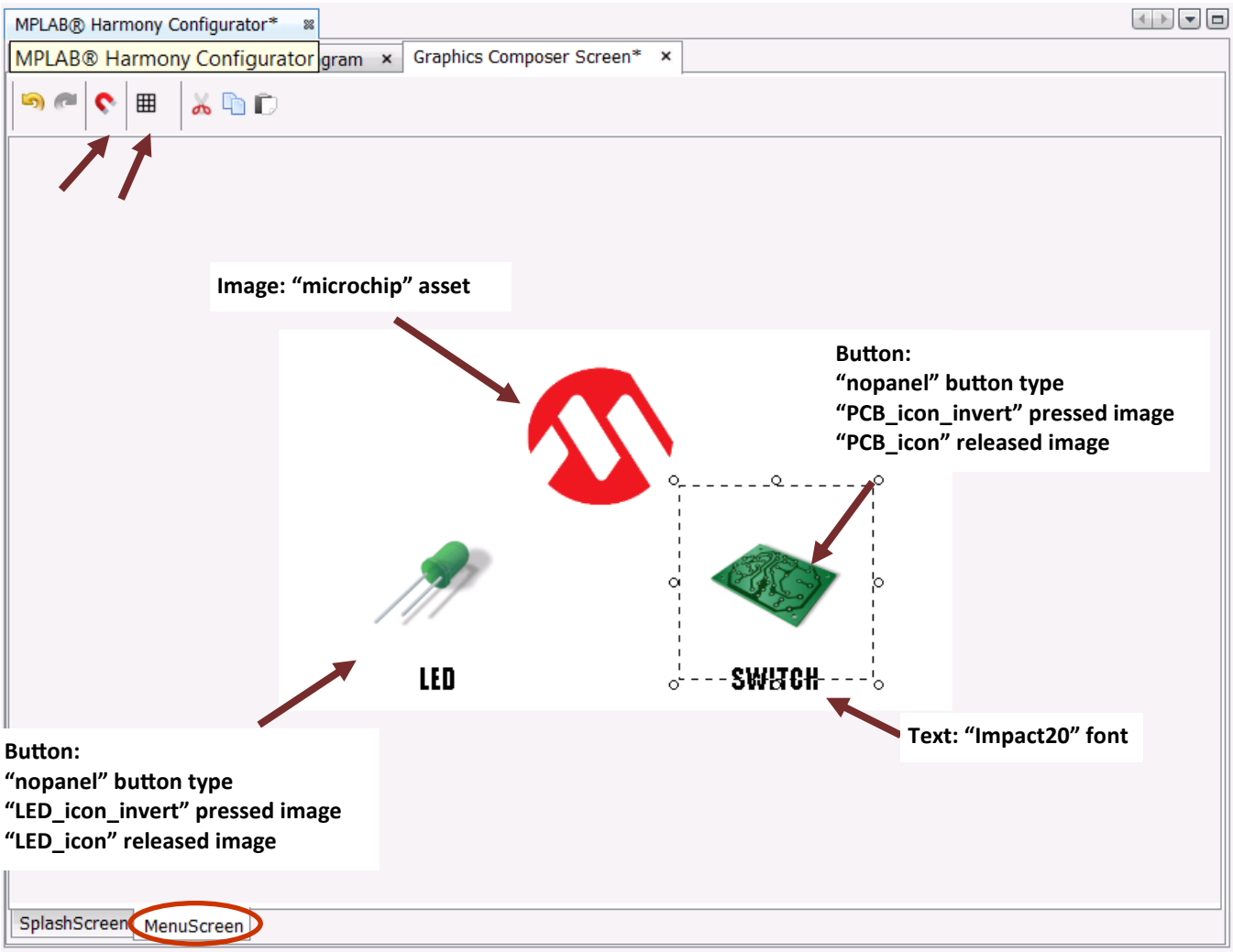


Figures 3.8
Rename the screen

6 **Using the techniques discussed in Lab 2, construct the Menu screen. To improve on aesthetics, you may want to try the Snap and Grid features.**

Note: The dotted box outline of the button dictates the touch-detection area. Make the box as large as possible to minimize the chance of a missed detection, often a primary source of user-frustration with regards to UI design.

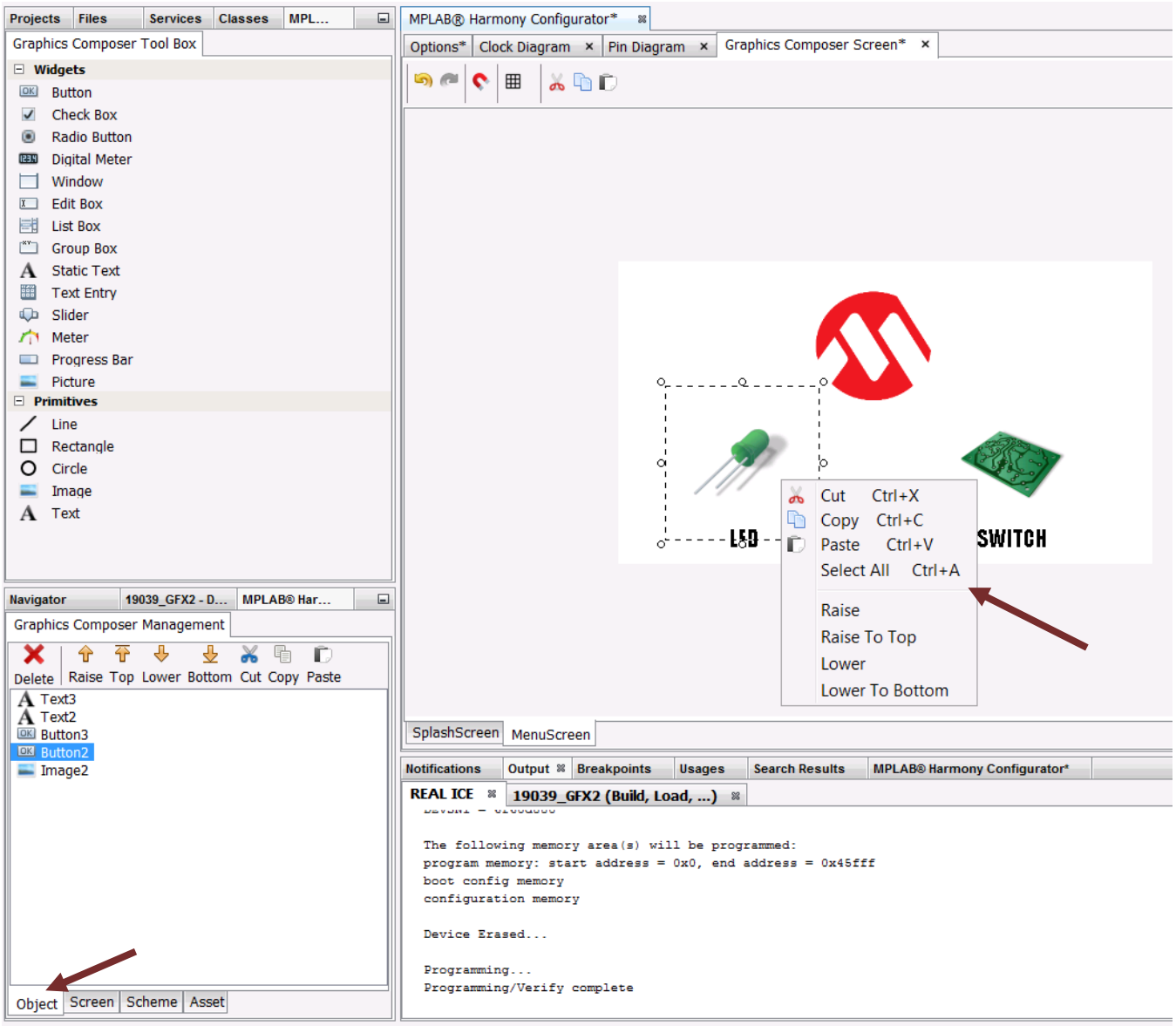
As shown in Figure 3.9, the button area can be set as large as possible, encompassing the text. As long as the images used for the button press and release do not overlap the text , this will not be a problem.



Figures 3.9
Construct the menu screen

7 **The draw-order of the various items on the application screen can be managed in the Graphics Composer Management window chosing bottom Object tab . Another way to manage draw-order is via the click menu by right-clicking on items already on the screen.**

Note: There is one exception to draw-order. Draw-order can be sorted among widgets and among primitives. However, primitives will always be drawn in front of widgets.



Figures 3.10
Draw-order management

8 **Now that we have the Menu Screen created, we need to find a way for the user to navigate from the Splash Screen to the Menu Screen when the application is running. We will do so by adding a Screen Change event to the button we had previously added in the SplashScreen.**

At the bottom of the Graphics Composer Screen window, click on the SplashScreen tab to go back to edit the SplashScreen. Once there, click on the button that is the Harmony logo.

In the Graphics Composer Properties window, under Events tree, enable the Released event.

To add actions to the event, click on the "..." button for the event. This will bring up the Event Editor dialog window.

In the Event Editor window, click on the Add button to bring up the Create New Action dialog window

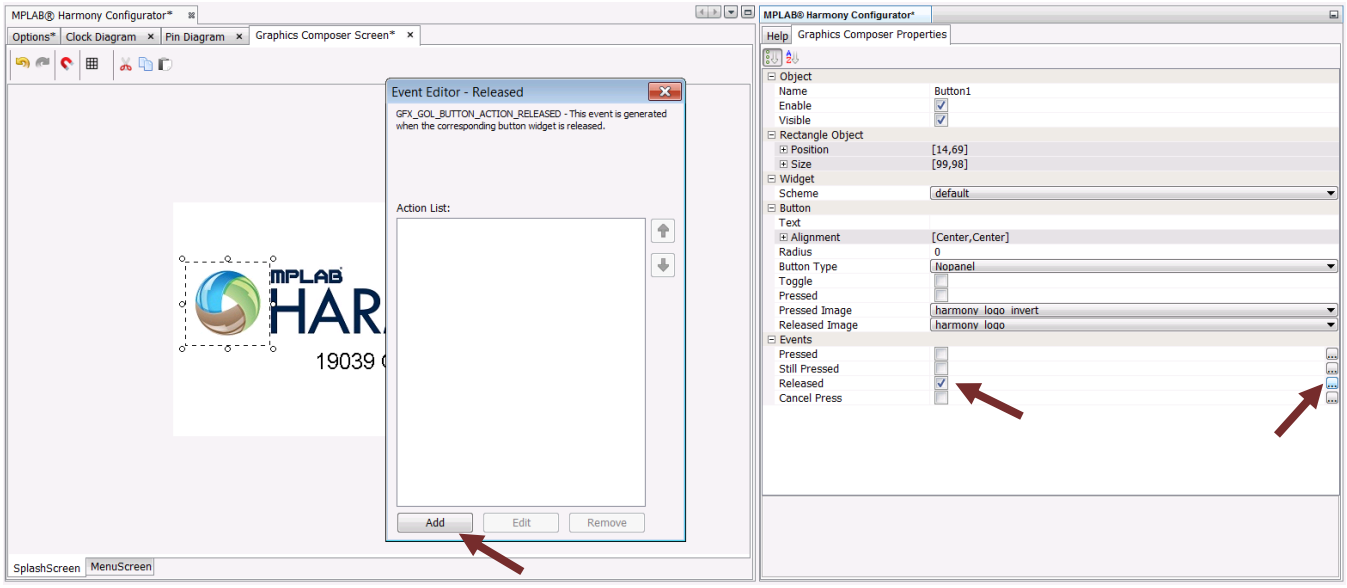


Figure 3.11
Editor Event dialog

9 **In the Create New Action dialog, we can add either Template or Custom action (from the Type dropdown selection). We will look at Custom actions later.**

At first we create an Activate Screen template action by selecting MenuScreen in the first and second columns. Then, select Activate Screen in the third column.

To help with future ease of maintenance, you may want to change the name of the action to be more descriptive. Click Create to add the action and take you back to the Event Editor dialog.

Click the X in the Event Editor dialog to close it.

Now you are ready to generate, compile and flash the project :

Go back to Options tab, press Generate, Save, Generate and finally compile and flash the program by pressing 

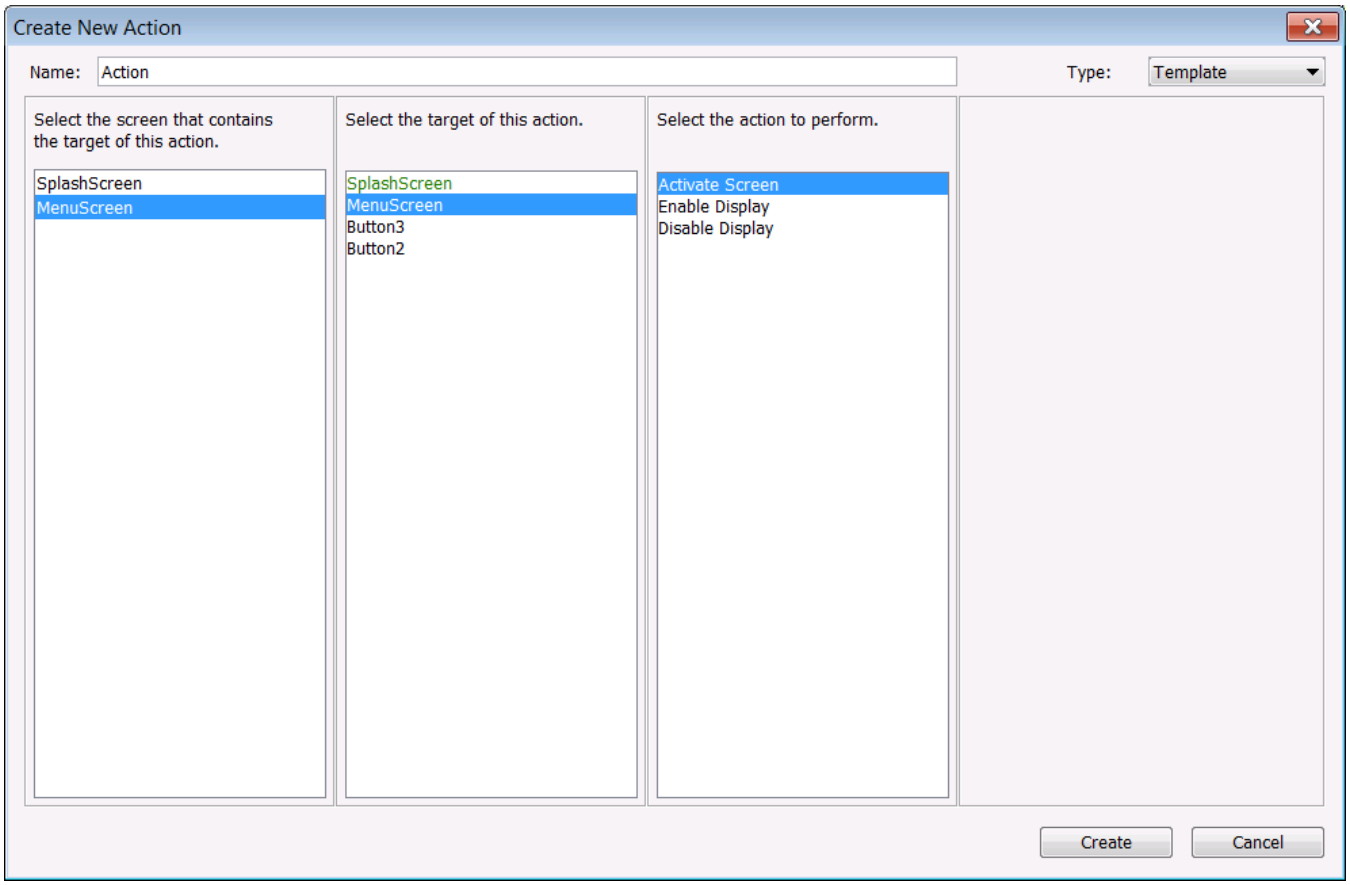


Figure 3.12
Create New Action dialog

Results

You have just learned how to add touchscreen capability to your project and added an event action to handle a touch event.

Conclusions

With the interactive menu, we will next create the LED screen. This screen will allow us to send output signs to the LEDs on the MEB-II.

THIS PAGE INTENTIONALLY LEFT BLANK

Lab Exercise 4

Creating the LED Control Screen

Purpose

In this lab, you will create an LED Control screen. It will contain widgets to control the LEDs on the MEB II development board.

Solution files may be found in:
C:\Microchip\harmony\v1_06 \apps\masters\solutions\19039_GFX2

Tools

- MPLAB®X IDE v3.06 or higher with MPLAB® Harmony Configurator (MHC) Plug-in 1.06.12 installed
- PIC32MZ EF Starter Kit
- Multimedia Expansion Board II
- Real-Ice
- 9V to 15V DC power supply
- XC32 Compiler v1.40

Objectives

1) Use MPLAB® Harmony Graphics Composer event action feature to add custom action to access LEDs on the MEB-II
2) Add code to the MPLABX project to refresh the widgets on the LCD.

Expected Results

When this lab is complete, you will be able to:

- Press the LED button to navigate from the Menu Screen to an LED Control screen.
- Pressing any of the LED buttons in the LED Control screen will toggle the corresponding LED on the MEB-II.

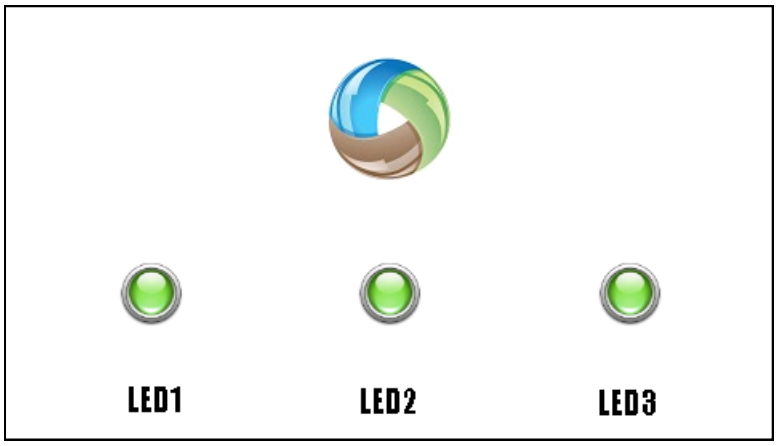


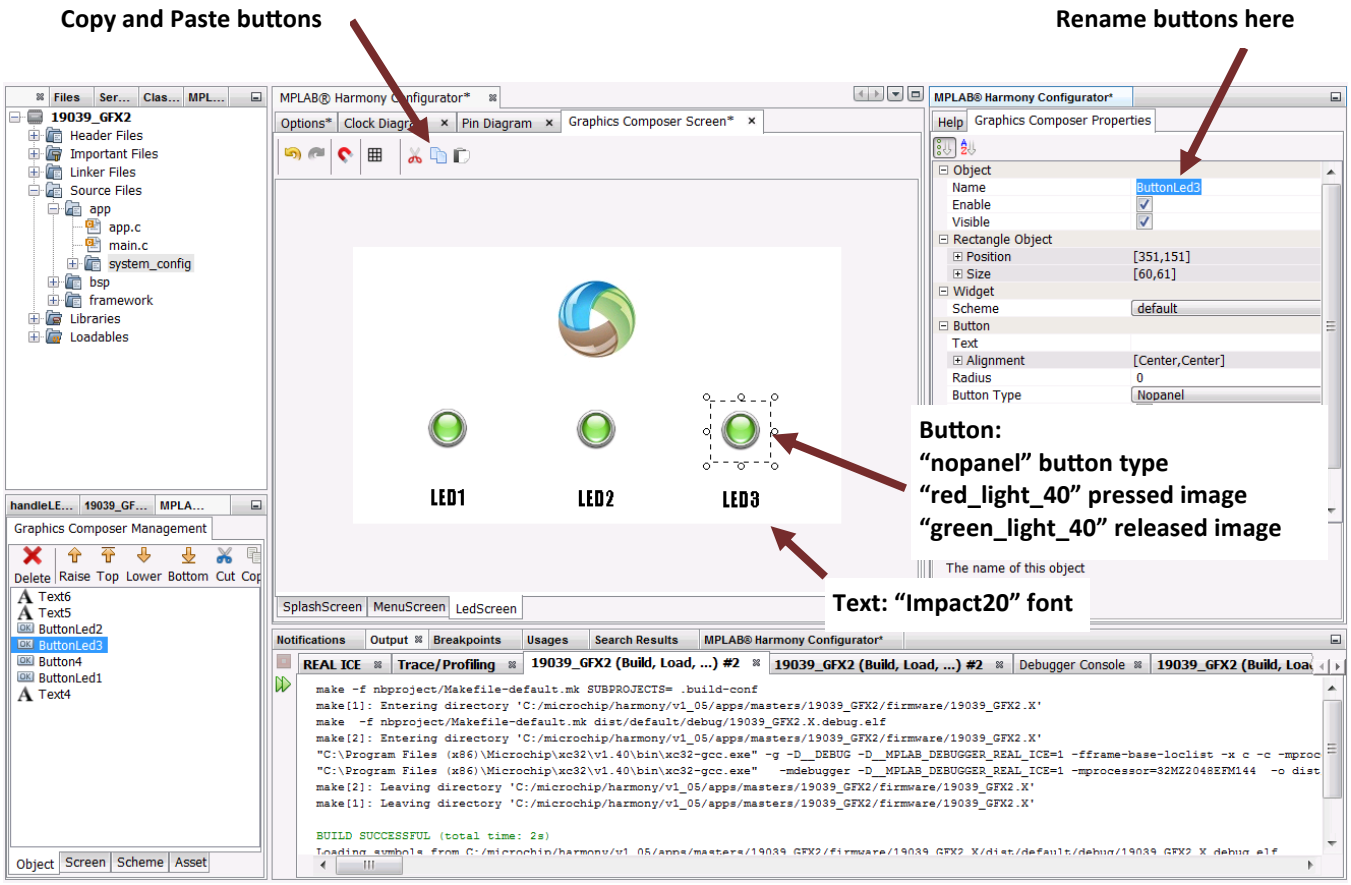
Figure 4.1 *When the lab is completed, the LCD Control screen should look something like this.*

1 **Using the techniques discussed in Lab 2 & 3, construct the LCD Control screen, name it LedScreen.**

To speed up your development time, you may want to try the Copy and Paste features. They are accessible as buttons in the sub-menu Graphics Composer Screen window, via hotkeys (Ctrl-C and Ctrl-V), or on the right-click menu.

The Harmony logo in the center of the screen doubles as the button to navigate back to MenuScreen. You can copy the button from the SplashScreen and paste it here.

Create 3 LED buttons and 3 text labels as described in Figure 4.2. To make things easier later, make sure to re-name the buttons to "ButtonLed1", "ButtonLed2", and "ButtonLed3". Case-sensitivity is important.



Figures 4.2
Construct the LED Control screen

2 **Now we need to add custom event actions to the buttons. This should be the very first time in this class you having to touch C-code.**

Starting with ButtonLed1, we will enable the Release event. This time, instead of a template action, we will add a custom action.

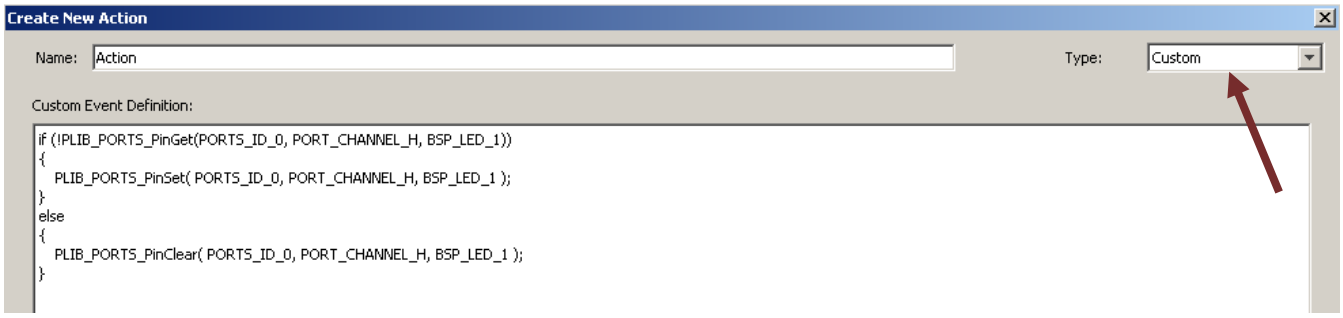
We will enter the PLIB calls that will read the LED pins, and toggle the pin base on the status of the LED.

1. **In the Graphics Composer Screen, select highlight Led1 object**
2. **In the Graphics Composer Properties, validate the Released checkbox and push the "…" button**
3. **In the Event Editor - Released dialog box, press the Add button at the bottom**
4. **In the Create New Action dialog box, change the Type pull-down menu to Custom**
5. **Add below source code under the Custom Event Definition field and push the Create button**

```
if (!PLIB_PORTS_PinGet(PORTS_ID_0, PORT_CHANNEL_H, BSP_LED_1))
{
    PLIB_PORTS_PinSet( PORTS_ID_0, PORT_CHANNEL_H, BSP_LED_1 );
}
else
{
    PLIB_PORTS_PinClear( PORTS_ID_0, PORT_CHANNEL_H, BSP_LED_1 );
}
```

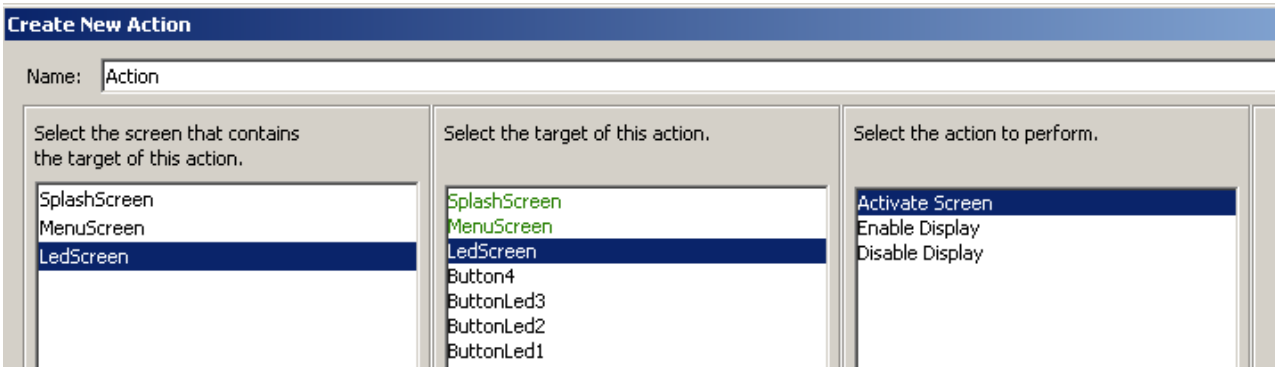
Repeat this for ButtonLed2 and ButtonLed3.
Be sure to use `BSP_LED_2` **and** `BSP_LED_3` **for the other custom actions.**

You can copy and paste the code from ButtonLed1.txt, ButtonLed2.txt, and ButtonLed3.txt in C:\Microchip\harmony\v1_06\apps\masters\code\ directory.

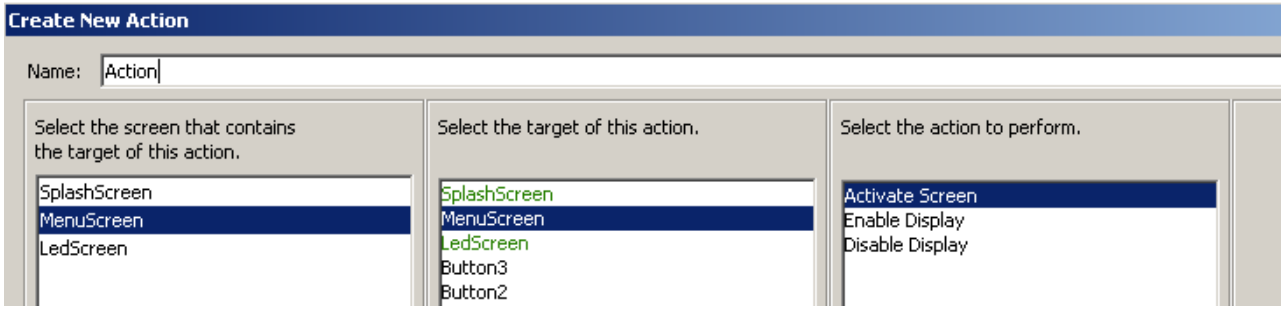


Figures 4.3 : Toggle LED Custom Action

6. In the MenuScreen menu, we need to assign the LED button to transition to the LedScreen menu :
    a. In Graphics Composer Screen window, highlight the LED object in the MenuScreen menu
    b. In Graphics Composer Properties window, under Events tree, validate the Released checkbox
    c. Push the "…" button on the right to add the LED button action
    d. In Event Editor window, click on the Add button to bring up the Create New Action dialog window
    e. In Create New Action dialog window, select LedScreen in the first and second columns and select Activate Screen in the third column.
    f. Push Create button and X in Event Editor dialog box to finish



7. In the LedScreen menu, we need to assign the Harmony logo button to transition to the MenuScreen menu :
    a. In Graphics Composer Screen window, highlight the Harmony logo object in the LedScreen menu
    b. In Graphics Composer Properties window, under Events tree, validate the Released checkbox
    c. Push the "…" button on the right to add the LED button action
    d. In Event Editor window, click on the Add button to bring up the Create New Action dialog window
    e. In Create New Action dialog window, select MenuScreen in the first and second columns and select Activate Screen in the third column.
    f. Push Create button and X in Event Editor dialog box to finish

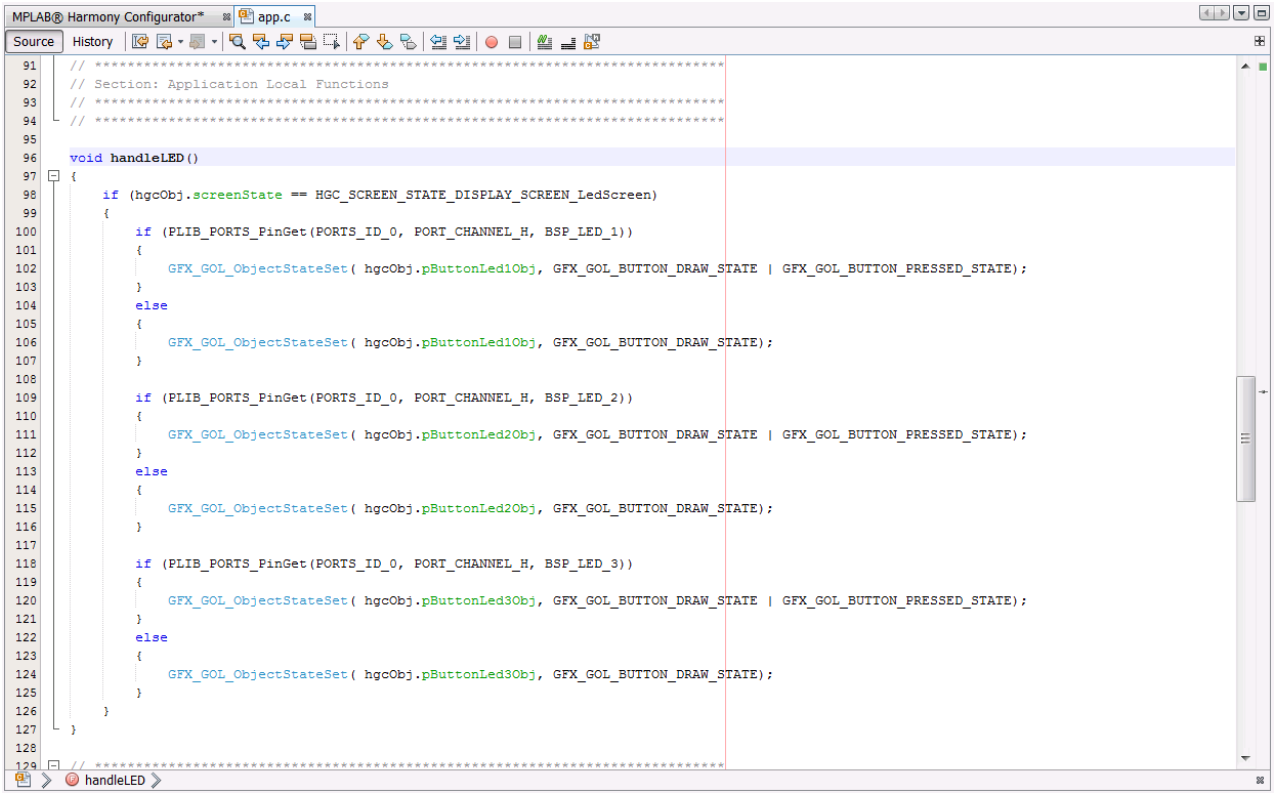


3 **Lastly, to close the loop with the LED control system, we will add code to make sure the LED buttons are drawn matched to the actual status of the LEDs.**

We will add the follow local function into app.c.

```
void handleLED()
{
    if (hgcObj.screenState == HGC_SCREEN_STATE_DISPLAY_SCREEN_LedScreen)
    {
        if (PLIB_PORTS_PinGet(PORTS_ID_0, PORT_CHANNEL_H, BSP_LED_1))
        {
            GFX_GOL_ObjectStateSet( hgcObj.pButtonLed1Obj, GFX_GOL_BUTTON_DRAW_STATE | GFX_GOL_BUTTON_PRESSED_STATE);
        }
        else
        {
            GFX_GOL_ObjectStateSet( hgcObj.pButtonLed1Obj, GFX_GOL_BUTTON_DRAW_STATE);
        }

        if (PLIB_PORTS_PinGet(PORTS_ID_0, PORT_CHANNEL_H, BSP_LED_2))
        {
            GFX_GOL_ObjectStateSet( hgcObj.pButtonLed2Obj, GFX_GOL_BUTTON_DRAW_STATE | GFX_GOL_BUTTON_PRESSED_STATE);
        }
        else
        {
            GFX_GOL_ObjectStateSet( hgcObj.pButtonLed2Obj, GFX_GOL_BUTTON_DRAW_STATE);
        }

        if (PLIB_PORTS_PinGet(PORTS_ID_0, PORT_CHANNEL_H, BSP_LED_3))
        {
            GFX_GOL_ObjectStateSet( hgcObj.pButtonLed3Obj, GFX_GOL_BUTTON_DRAW_STATE | GFX_GOL_BUTTON_PRESSED_STATE);
        }
        else
        {
            GFX_GOL_ObjectStateSet( hgcObj.pButtonLed3Obj, GFX_GOL_BUTTON_DRAW_STATE);
        }
    }
}
```

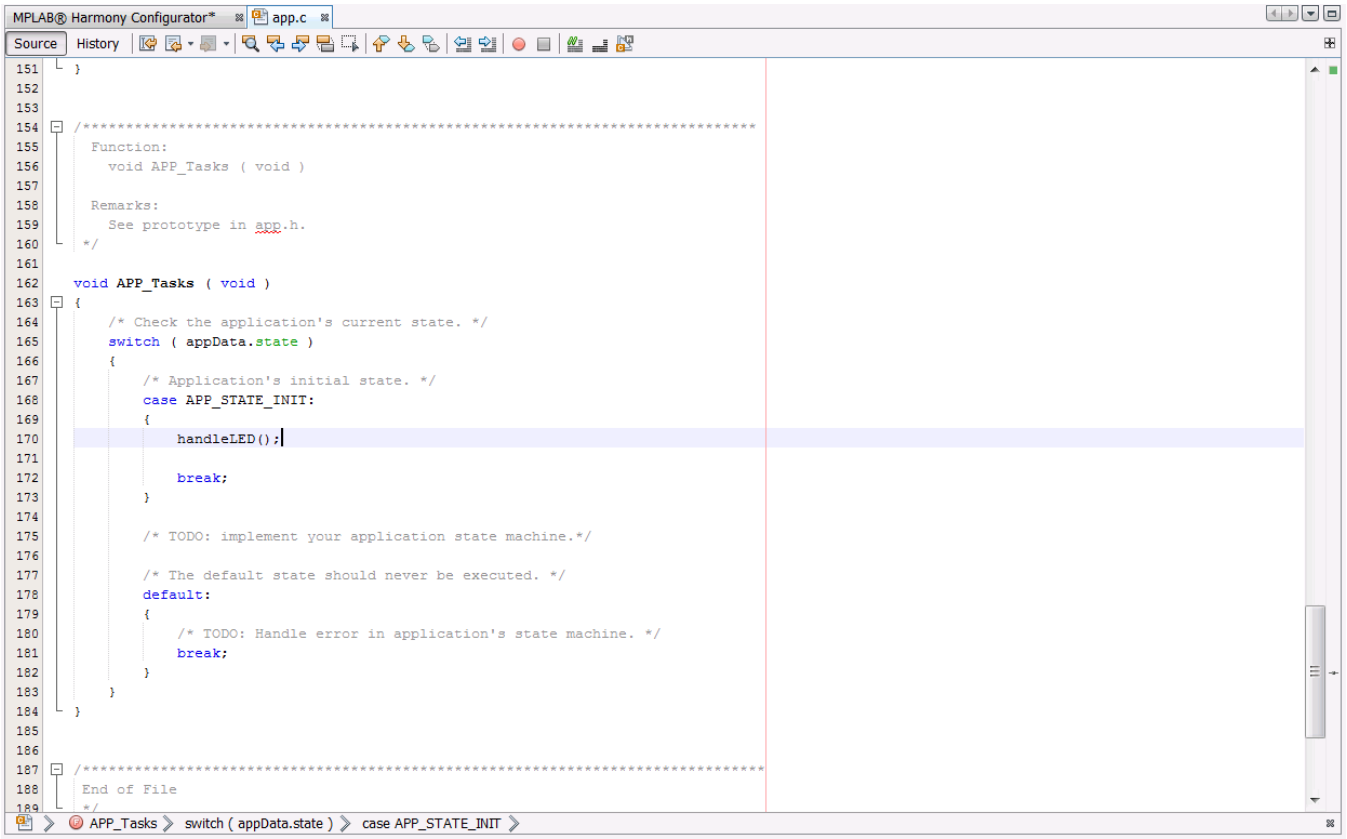
You can copy and paste this code from handleLED.txt in C:\Microchip\harmony\v1_06\apps\masters\code



Figures 4.4 : Add handleLED() to app.c

4 **The very last step is to add handleLED() into the APP_Task() function.**

Generate, build, and deploy the project.



Figures 4.5
Add handleLED() to APP_STATE_INIT

Results

Using MPLAB® Harmony Graphics Composer, we have added custom event action to interactive with LEDs on the MEB-II. This demonstrates the ease to add output signal to peripherals via the MHGC.

Code Analysis

At the conclusion to this lab, the instructor will walk you through the application specific portions of the code used to handle the LEDs.

Conclusions

Now that we have shown how easy it is to add output signaling, in the next lab we will try to read an input and display it on the LCD.

BONUS

There is a way to do this lab with a lot less code. You may want to get this a try.

THIS PAGE INTENTIONALLY LEFT BLANK

Lab Exercise 5

Creating the Switch Input Screen

Purpose

In this lab, you will create a Switch Input screen. It will contain a progress bar widget to demonstrate reading a push button input from the MEB II development board.

Solution files may be found in:
C:\Microchip\harmony\v1_06\apps\masters\solutions\19039_GFX2

Tools

- MPLAB®X IDE v3.06 or higher with MPLAB® Harmony Configurator (MHC) Plug-in 1.06.12 installed
- PIC32MZ EF Starter Kit
- Multimedia Expansion Board II
- Real-Ice
- 9V to 15V DC power supply
- XC32 Compiler v1.40

Objectives

1) Use MPLAB® Harmony Graphics Composer to create a Switch Input screen
2) Add code to the MPLABX project to update the progress bar widget on the LCD.

Expected Results

When this lab is complete, you will be able to:

- Navigate to the Switch Input screen from the Menu screen
- Press the push button on the MEB-II to fill a progress bar in the Switch Input screen
- Press the Clear button on the LCD Display will set the progress bar back to 0%

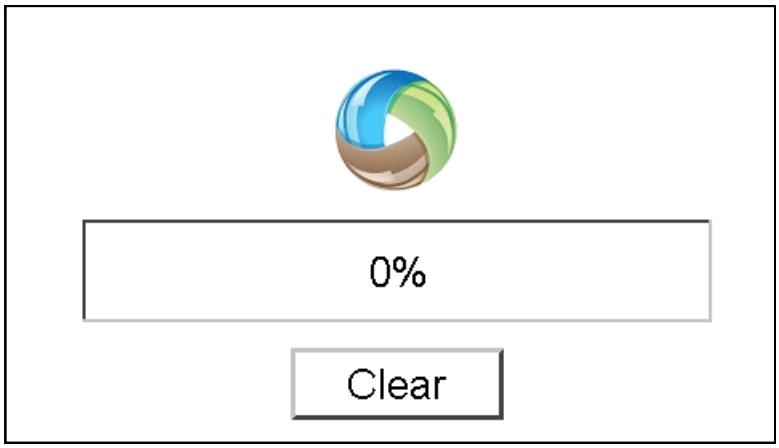
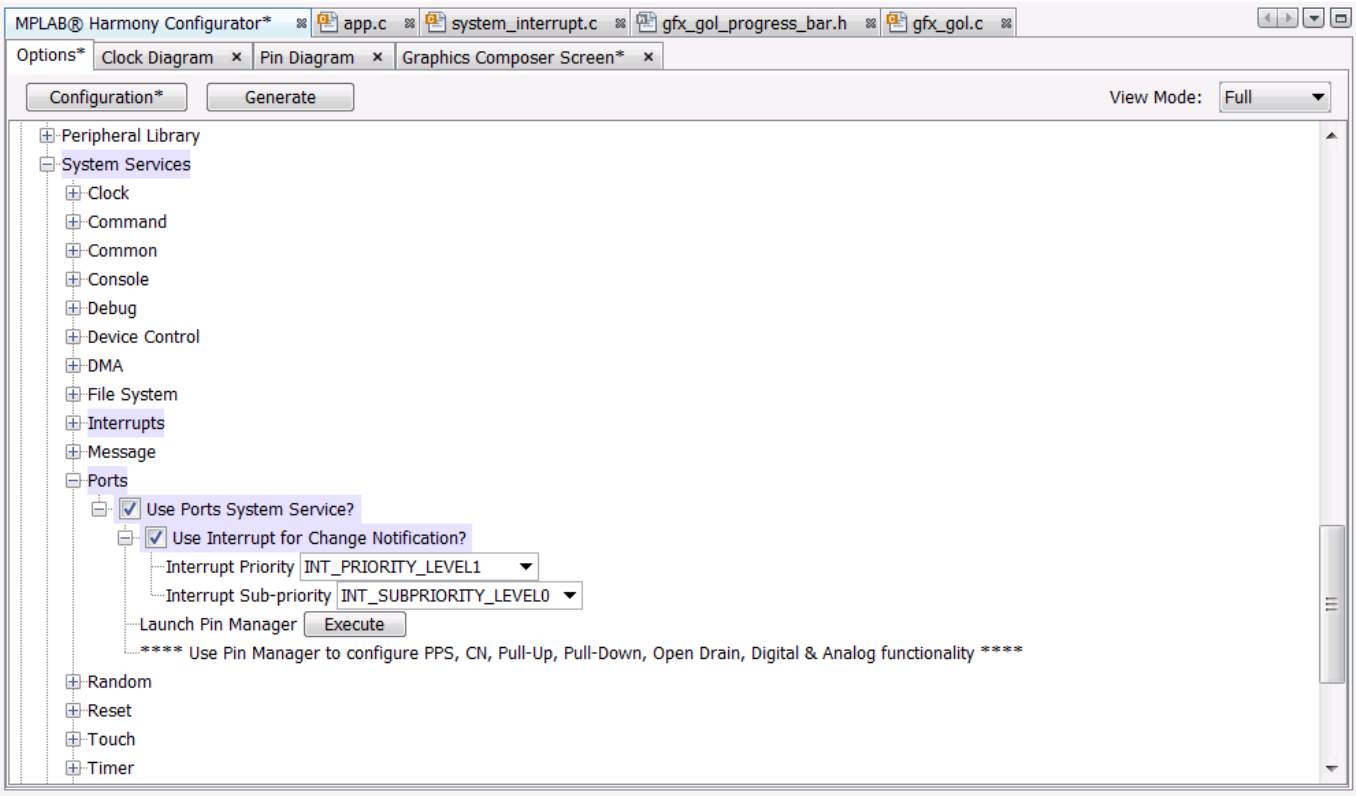


Figure 5.1 *When the lab is completed, the Switch Input screen should look something like this.*

1 **First, we will need the switch push to generate an interrupt. To do this, we need Change Notification.**

Back in the tree view, under System Services -> Ports, we will enable Use Interrupt for Change Notification.



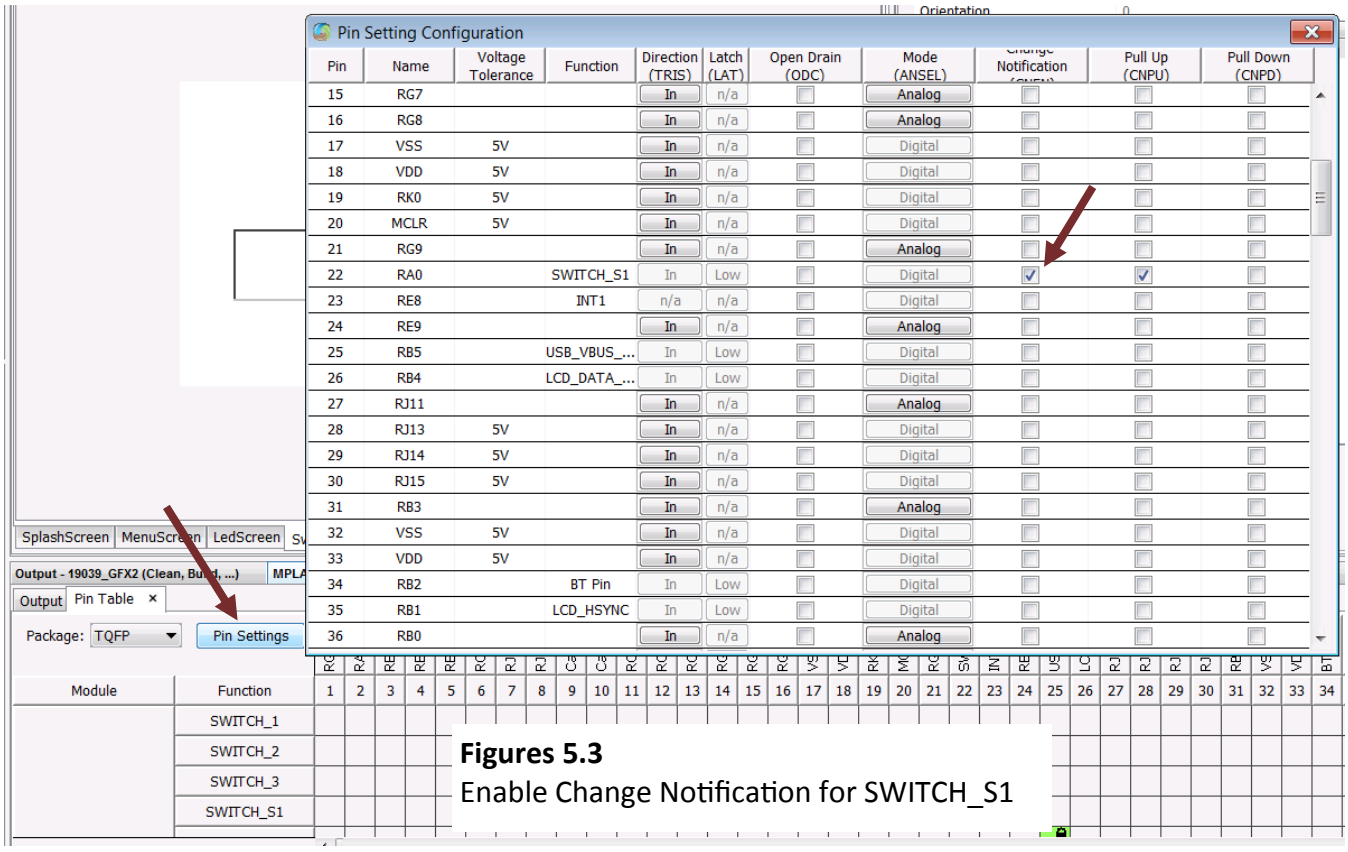
Figures 5.2
Enable Change Notification

2 **Next, we will enable the Change Notification for the switch pin.**

In the Pin Table view, click on the Pin Settings button. This will bring up the Pin Setting Configuration Dialog.

In this dialog, find pin 22, which is already mapped to RA0 by the BSP settings. Note it already has the function label SWITCH_S1.

Close the dialog and the setting is applied.

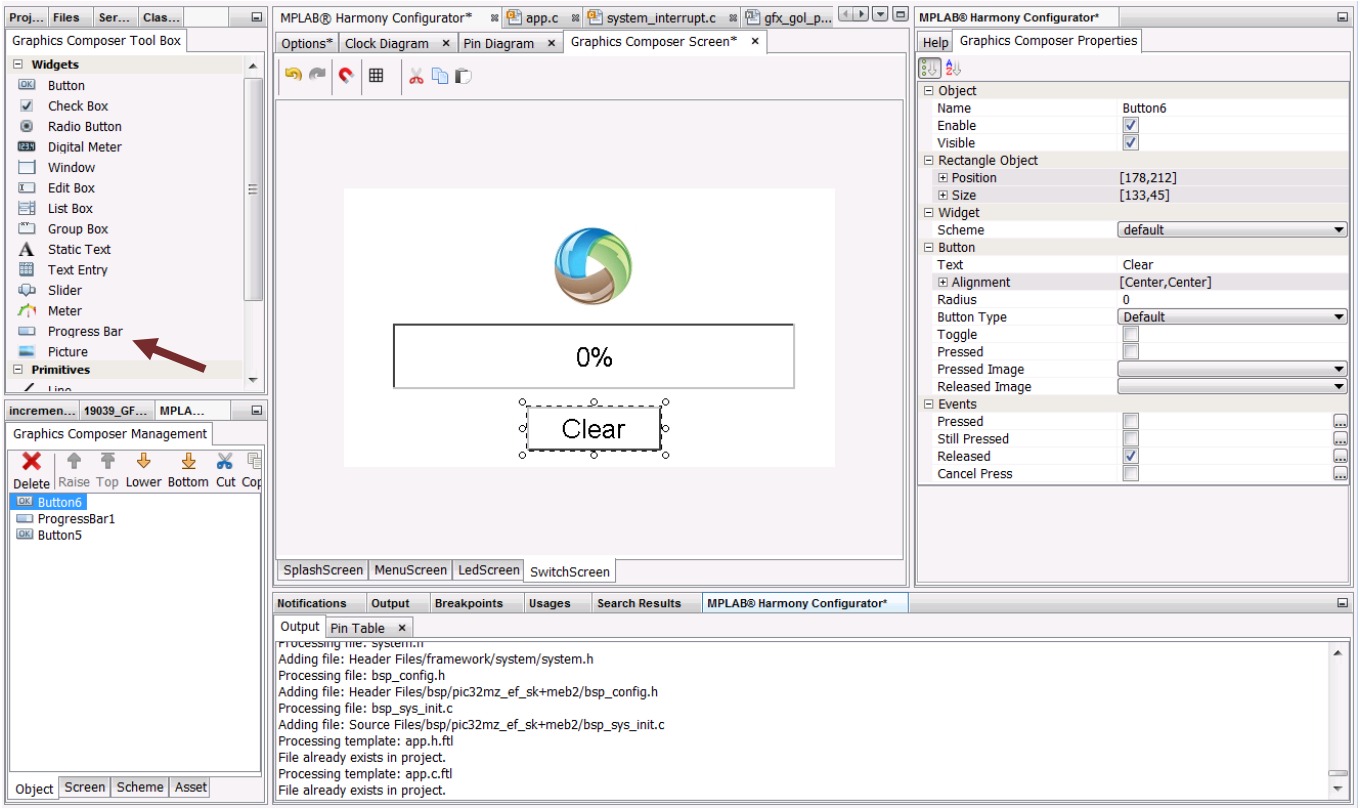


Figures 5.3
Enable Change Notification for SWITCH_S1

3 **Now, we are back in the MHGC. Let's construct the Switch Input screen.**

Similar to the LED Control Screen. It will have a Harmony logo button to go back to the Menu screen.

We will also drag in a Progress Bar and a Button from under Widgets.



Figures 5.4
Build Switch Input Screen

4 **Let's create a unique scheme for the progress bar.**

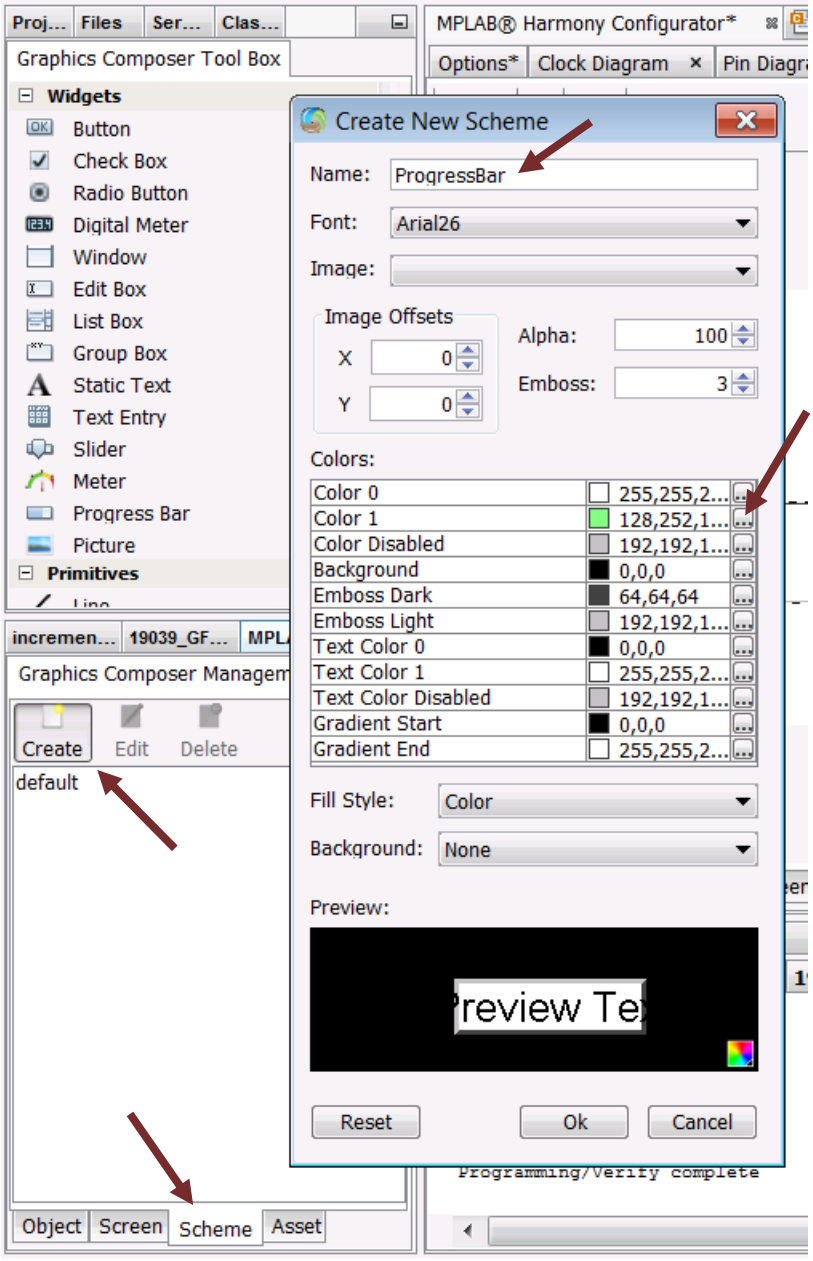
Click on the Scheme tab in the Graphics Composer Management window.

Click Create to bring up the Create New Scheme dialog.

Change the default name to a meaningful name like "ProgressBar".

Change Color 1 to a color that contrasts well with the color for Text Color 0.

Press OK to create scheme.

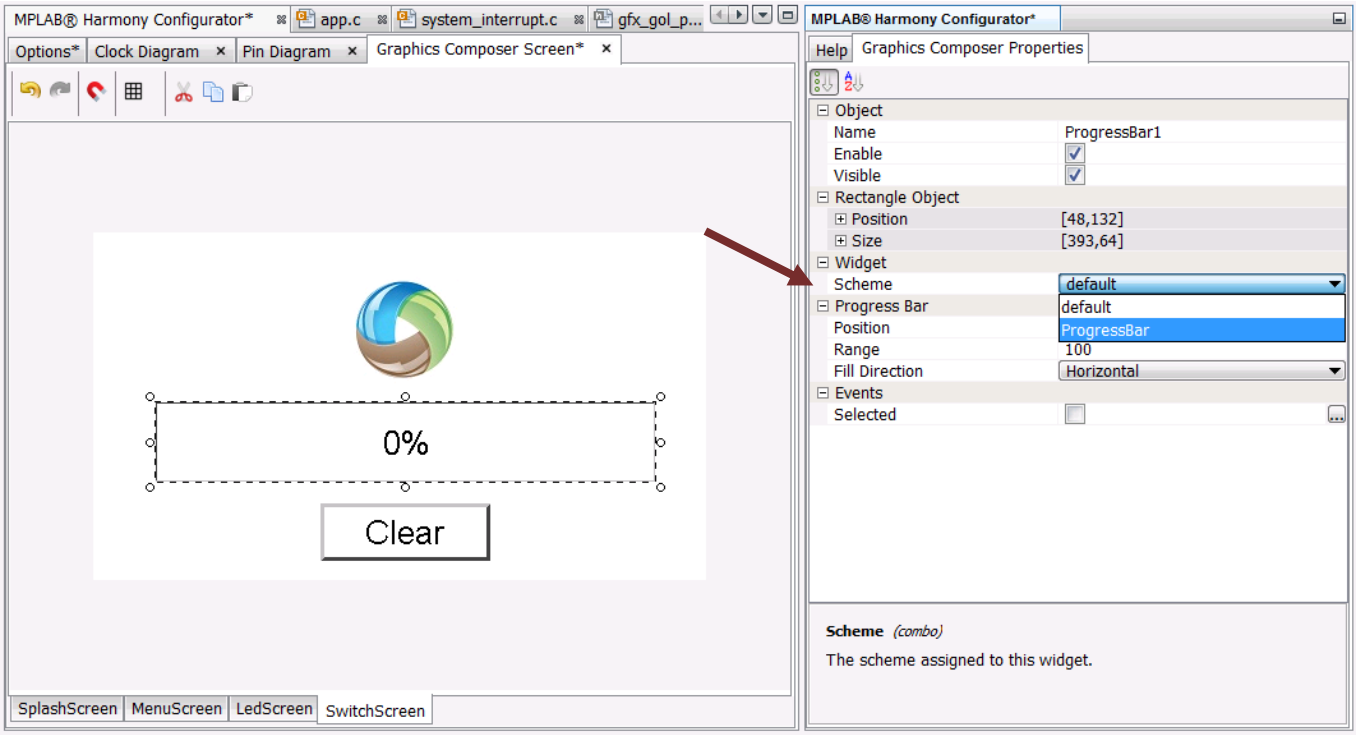


Figures 5.5
Edit Scheme Dialog

5 **Now we need to apply the newly created scheme.**

Click on the progress bar object in the screen. This will bring up the its properties in the Graphics Composer Properties window.

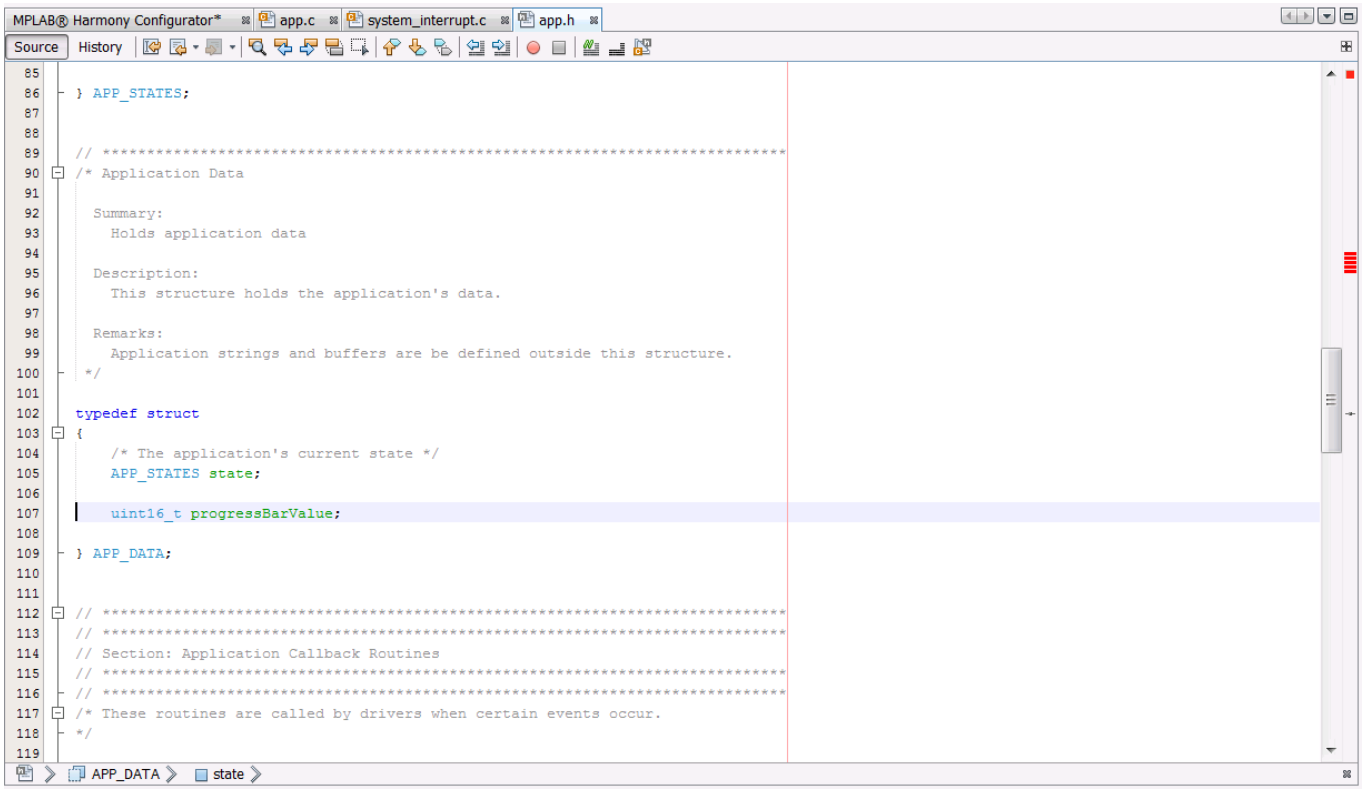
Under Scheme, select the newly created scheme from the dropdown box.



Figures 5.6
Associate scheme with progress bar

6 **Next, we need to add some code.**

Add a uint16_t variable progressBarValue to the appData definition in app.h.



Figures 5.7
Add uint16_t definition in app.h

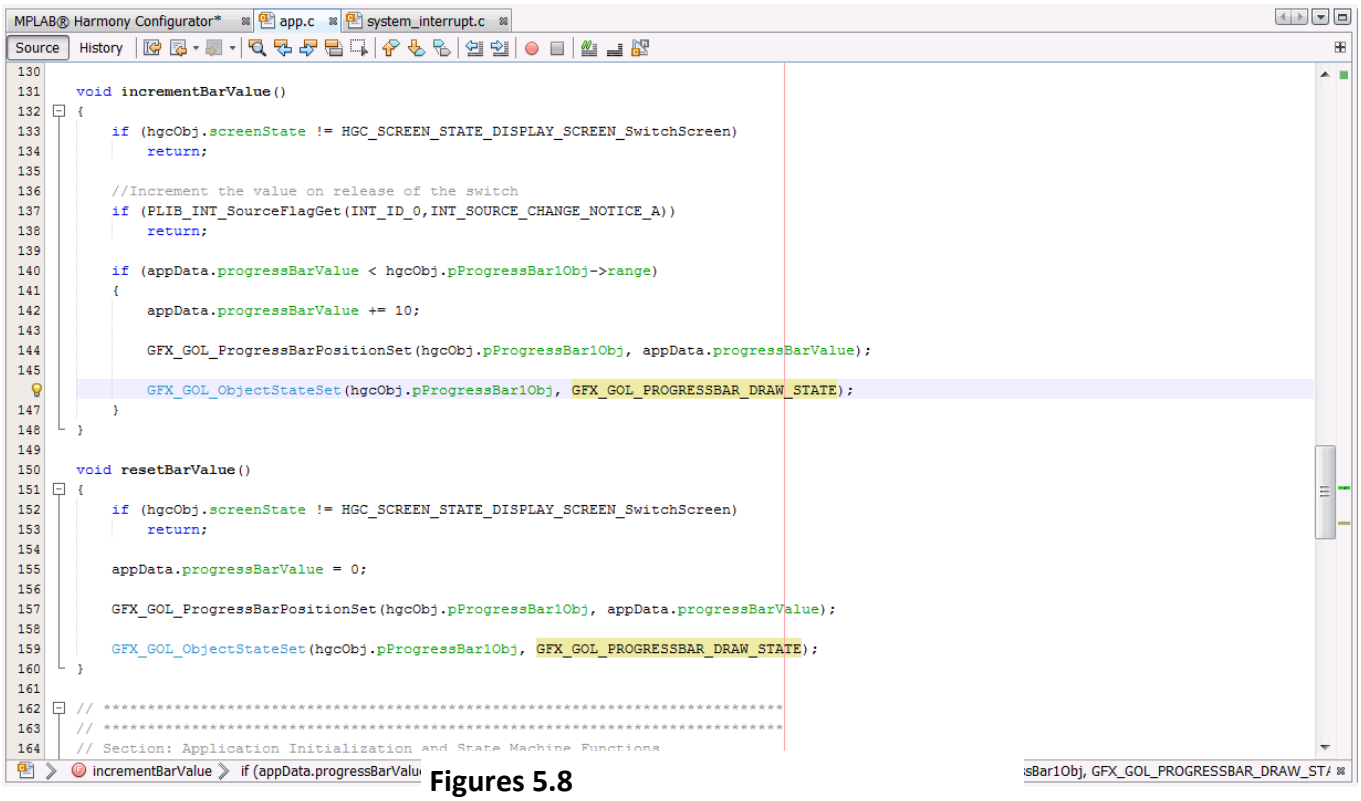
7 **Next, we need to add some code.**

Add the follow functions as local functions into app.c.

```
void incrementBarValue()
{
    if (hgcObj.screenState != HGC_SCREEN_STATE_DISPLAY_SCREEN_SwitchScreen)
        return;

    //Increment the value on release of the switch
    if (PLIB_INT_SourceFlagGet(INT_ID_0,INT_SOURCE_CHANGE_NOTICE_A))
        return;

    if (appData.progressBarValue < hgcObj.pProgressBar1Obj->range)
    {
        appData.progressBarValue += 10;

        GFX_GOL_ProgressBarPositionSet(hgcObj.pProgressBar1Obj, appData.progressBarValue);

        GFX_GOL_ObjectStateSet(hgcObj.pProgressBar1Obj, GFX_GOL_PROGRESSBAR_DRAW_STATE);
    }
}

void resetBarValue()
{
    if (hgcObj.screenState != HGC_SCREEN_STATE_DISPLAY_SCREEN_SwitchScreen)
        return;

    appData.progressBarValue = 0;

    GFX_GOL_ProgressBarPositionSet(hgcObj.pProgressBar1Obj, appData.progressBarValue);

    GFX_GOL_ObjectStateSet(hgcObj.pProgressBar1Obj, GFX_GOL_PROGRESSBAR_DRAW_STATE);
}
```

These functions will handle the incrementing and reset the progress bar.

You can copy and paste this code from progressBarCode.txt in C:\Microchip\harmony\v1_06\apps\masters\code



Figures 5.8
Add code in app.c

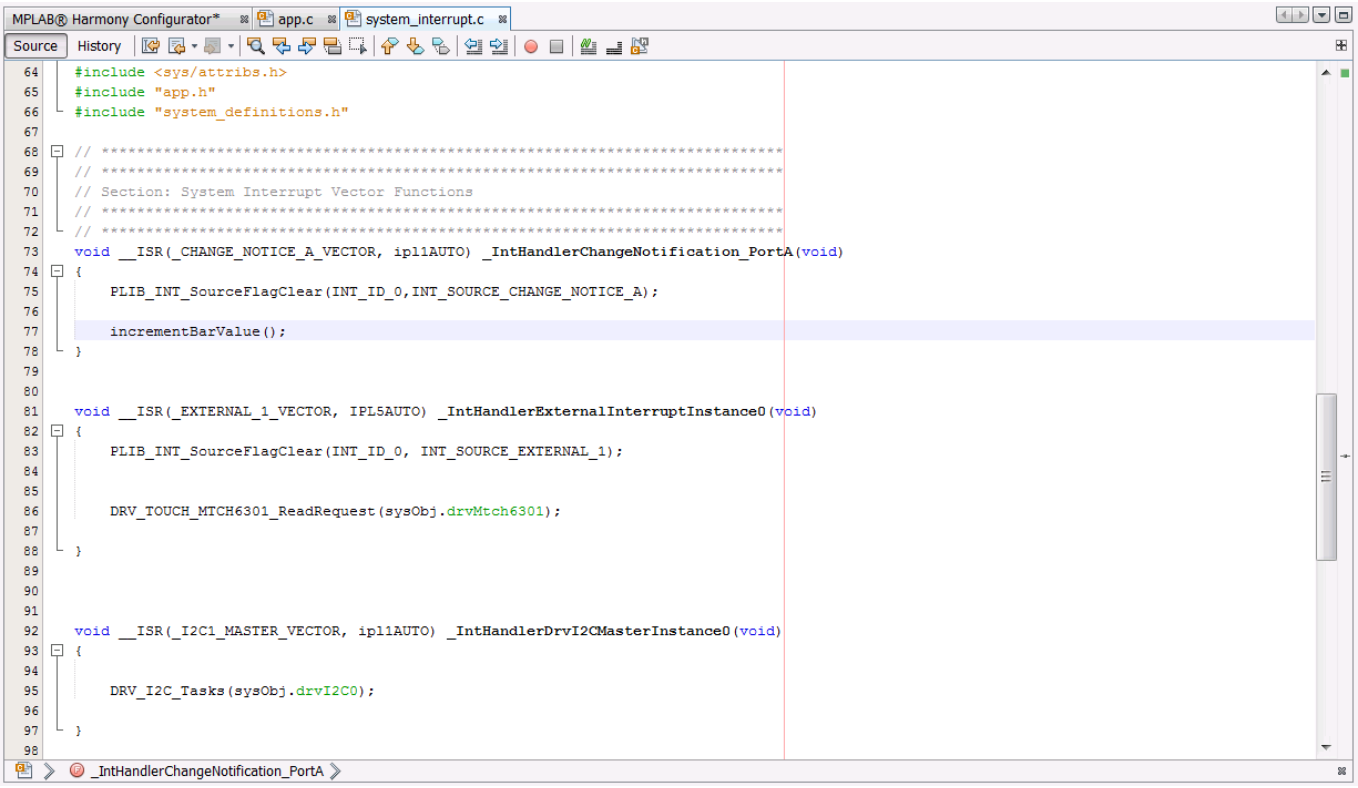
8 **As shown in the previous lab, add a custom event action for the Clear button. Have the action call resetBarValue().**

Generate the code. But we do not build yet. There is one more step.

The generate will have added a stub of an interrupt service routine in system_interrupt.c to handle Port A Change Notifications, this is the port for Switch S1 (RA0).

As shown in Figure 5.8, add a call to incrementBarValue().

Build and deploy.



Figures 5.9
Add function call to system_interrupt.c

Results

Using the MPLAB® Harmony Configurator and the MPLAB® Harmony Graphics Composer, we have added interrupt change notification handling to map the S1 switch on the MEB-II to display some graphical result on the LCD display.

Code Analysis

At the conclusion to this lab, the instructor will walk you through the application specific portions of the code used to handle the progress bar value.

Conclusions

With very little C-coding, you have now successfully created an application that will handle graphic user input as well as peripheral I/O, using the MPLAB® Harmony Configurator and the MPLAB® Harmony Graphics Composer.

These labs are designed to demonstrate the potential of the MHC and the MHGC as tools to cut down on development time for projects using the MPLAB® Harmony framework.

THIS PAGE INTENTIONALLY LEFT BLANK

Lab Exercise 6

Adding Graphics to an Existing Project

Purpose

As a bonus lab, using everything you have learned in the previous labs, you will add a GUI to an demo project in the apps folder in Harmony.

There are no solution files for this lab.

Tools

- MPLAB®X IDE v3.06 or higher with MPLAB® Harmony Configurator (MHC) Plug-in 1.06.12 installed
- PIC32MZ EF Starter Kit
- Multimedia Expansion Board II
- Real-Ice
- 9V to 15V DC power supply
- XC32 Compiler v1.40

Objectives

1) Reconfigure the demo project C:\microchip\harmony\v1_06\apps\examples\system\debug_usb_cdc_2 to use the Graphics Library with touchscreen input.
2) Use MPLAB® Harmony Graphics Composer to create a screen with a widget
3) Add custom event action code to send a message out the USB console service

Expected Results

When this lab is complete, you will be able to:

- Launch the application with USB console terminal running
- The console will successfully output some messages in legible ASCII format
- Pressing a widget on the LCD display will cause the console to display some additional debug messages

Results

Using the MPLAB® Harmony Configurator and the MPLAB® Harmony Graphics Composer, you had, hopefully, added a GUI to an existing Harmony project, with very little development time, an almost no coding required.

Conclusions

This labs are designed to further demonstrate the potential of the MHC and the MHGC as tools to cut down on development time for projects using the MPLAB® Harmony framework.

THIS PAGE INTENTIONALLY LEFT BLANK