HIF2132A

Designing Stylish HMIs with Microchip Graphics Library and Visual Graphics Display Designer (VGDD) using PIC32 LCC Graphics

Hands –On

Lab Manual

Microchip Technology Inc.

MICROCHIP

Regional Training Centers

Designing with Microchip Graphics Library

Table of Contents

Section 1: Managing Projects

Section 2: Building Projects

Section 3: Debugging Projects

Hardware

Special Instructions for Hardware Setup

Hardware Selection

All the labs included in this manual will work with the hardware combinations discussed below. Microchip Technology offers many other development tools that may be used to develop graphical applications. If properly configured, any of Microchip's graphical development tools should work with the provided lab code. Due to time constraints, we have only tested the hardware combinations used for this class. Information provided in this section will assist you with configuring the projects provided to work with either your custom hardware or one of Microchip's other graphics development platforms.

1 Choose the processor you wish to work with today

The labs provided with this class have been tested using the following Microchip MCU on the PIC32 USB Starter Kit. Please choose one to use as you work through the hands-on portion of the class.

PIC32MX795F512L

The PIC32MX795F512L is a highly integrated, powerful 32-bit microcontroller utilizing an 80 MHz, 1.56 DMIPS\MHz, 32-bit M4K® Core. The device features 512 KB Flash Memory, 128 KB RAM, USB OTG\Host module, and a variety of other Advanced Peripherals including Ethernet. For QVGA display designs at 8bpp, the PIC32MX family may be used to direct drive the display using a recently developed controllerless interface.

Information

When working the Microchip Graphics Library in your application, any of the 16– or 32-bit MCUs may be used. Please visit www.microchip.com\graphics for further information.

References

- Please refer to the device datasheet and family reference manual for further details on how to use the selected device.
- Application Note AN1368 provides details for working with the PIC24FJ DA product family
- Application Note AN1387 provides details for working with the PIC32 in a controllerless application
- The PIC32 controllerless application is also described in a webinar available from www.microchip.com\graphics

Figure H.1
Truly 3.2" QVGA Display board (AC164127-4)

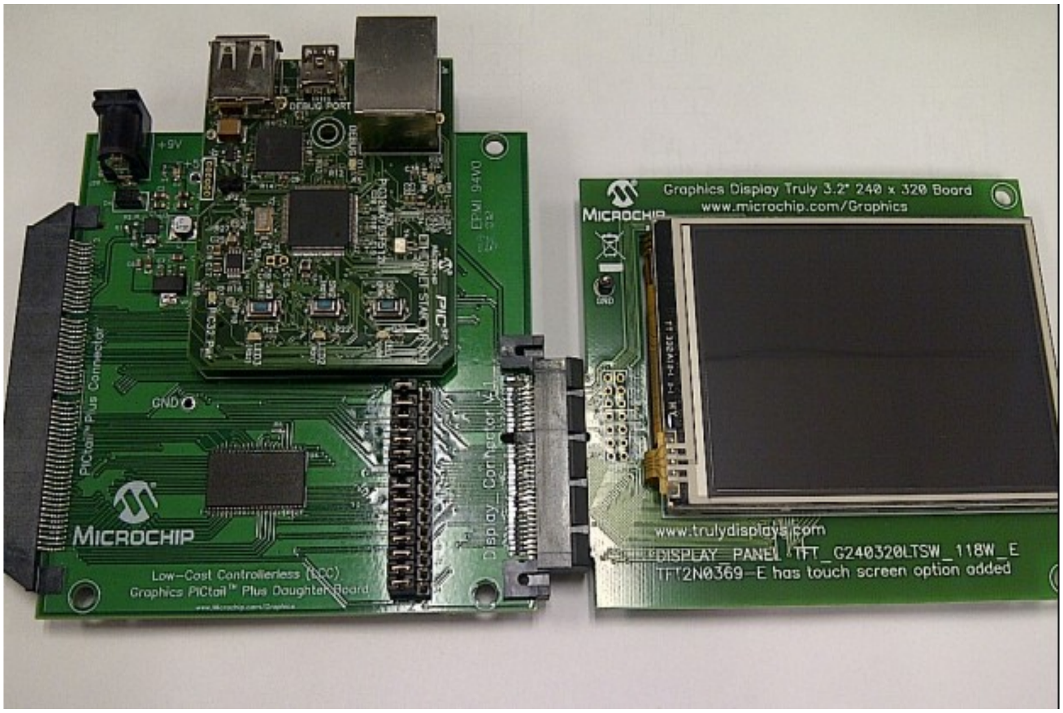
② The display board

The labs provided with this class have been tested with the 3.2" Truly QVGA display (shown in figure H.1) boards. Using the Visual Graphics Display Designer (VGDD), it is easy to migrate projects to support different screen sizes.

③ Set up the hardware development platform

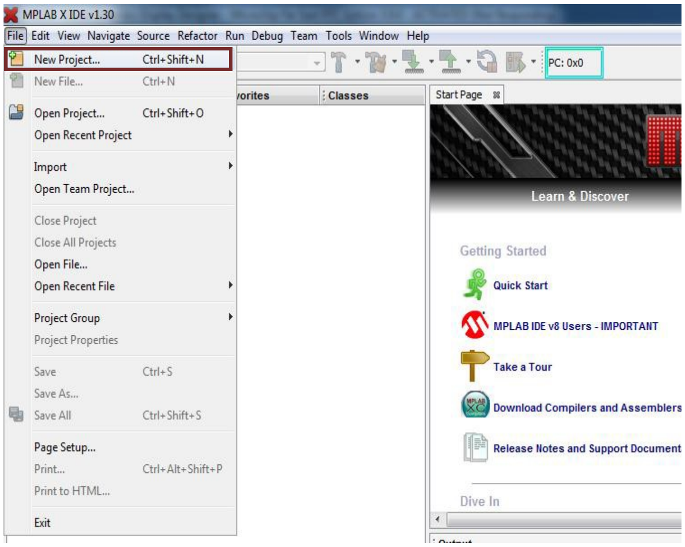
This class uses the LCC PICTail graphics board along with the PIC32 USB Starter Kit. The image below shows how the hardware should be connected. All jumpers on the PICTail should be set to positions (2-3). This sets up the PICTail to be used in external memory mode.

Hardware Tools:	PIC32 USB Starter Kit II Low Cost Controllerless (LCC) Graphics Board Graphics Display Truly 3.2" 240 x 320 Board

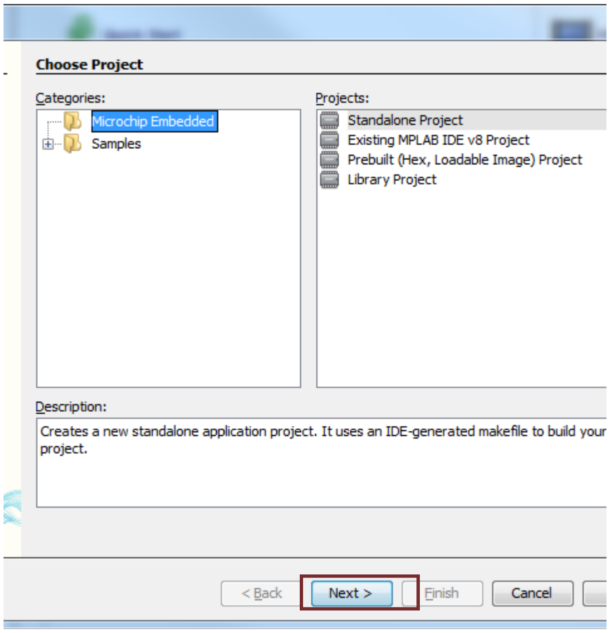

DEBUG PORT
+9V
GND
PICtail™ Plus Connector
MICROCHIP
Low-Cost Controllerless (LCC) Graphics PICtail™ Plus Daughter Board
Display_Connector
Graphics Display Truly 3.2" 240 x 320 Board
www.microchip.com/Graphics
MICROCHIP
GND
www.trulydisplays.com
DISPLAY_PANEL TFT_G240320LTSW_118W_E
TFT2N0369-E has touch screen option added

④ Working with Microchip Graphics New Projects For Labs

Step 1 – MPLAB(X) Template For New Project

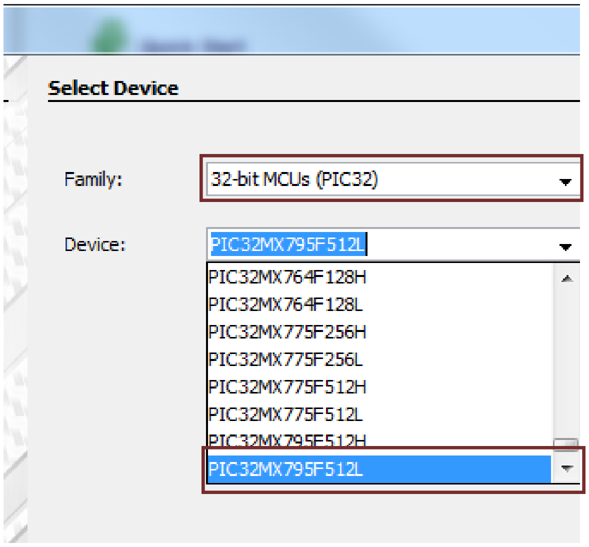


Step 2 – Choose Standard Project



Creates a new standalone application project. It uses an IDE-generated makefile to build your project.

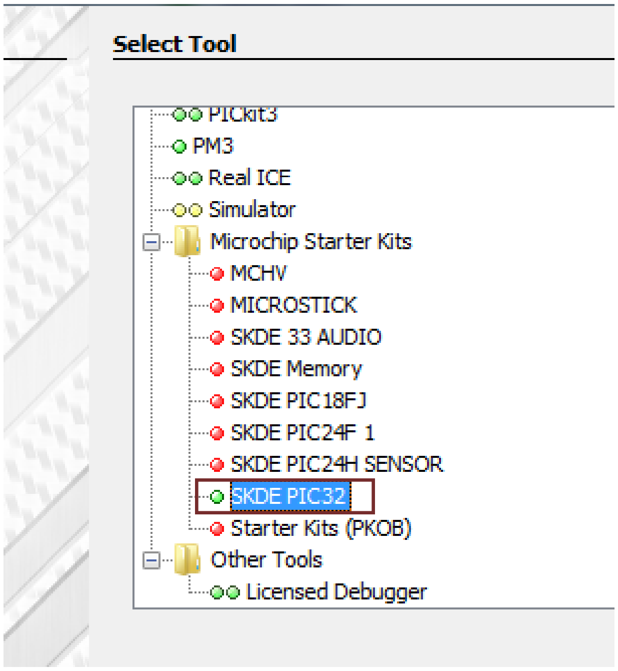
Step 3 – Select PIC32 With PIC32MX795F512L



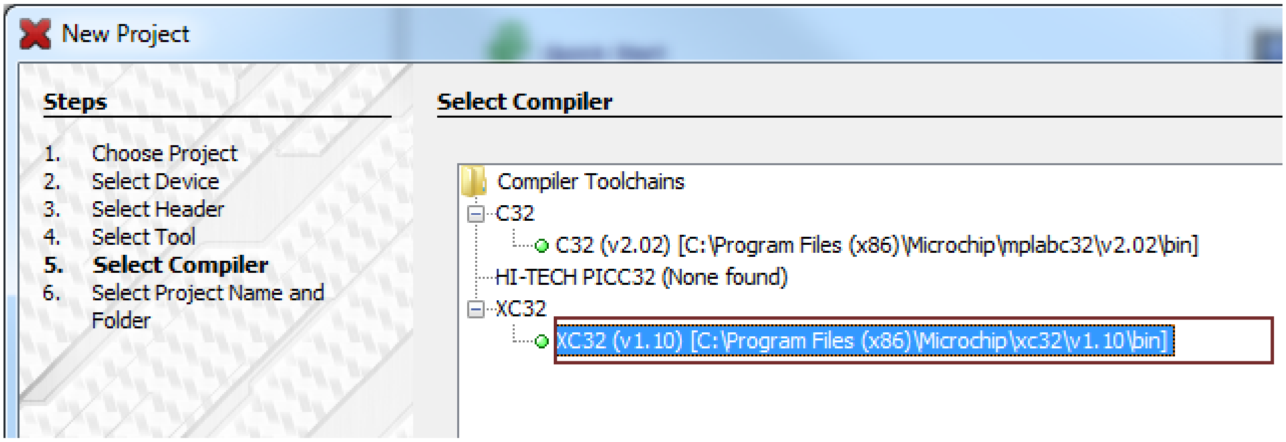
Family: 32-bit MCUs (PIC32)

Device: PIC32MX795F512L

Step 4 – Select Starter Kit (SKDE PIC32) As Debugger

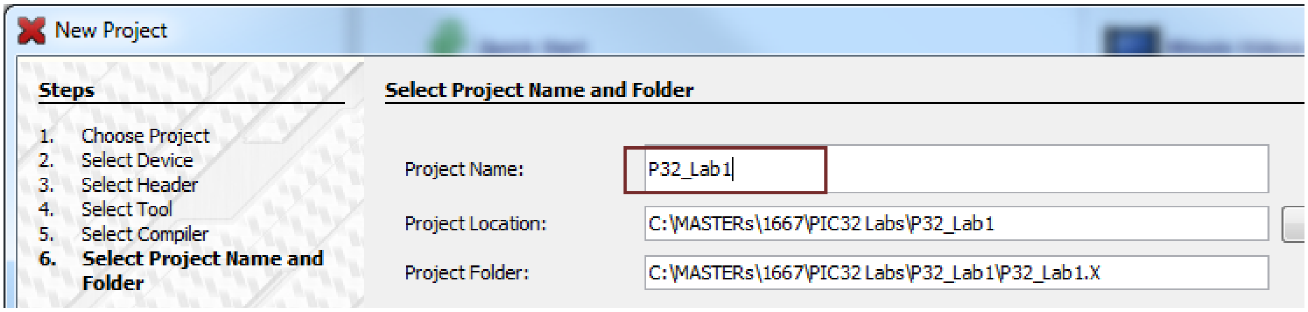


Step 5 – Select XC32 For Complier Option



Compiler Toolchains
- C32
  - C32 (v2.02) [C:\Program Files (x86)\Microchip\mplabc32\v2.02\bin]
- HI-TECH PICC32 (None found)
- XC32
  - XC32 (v1.10) [C:\Program Files (x86)\Microchip\xc32\v1.10\bin]

Step 6 – Give Project Name

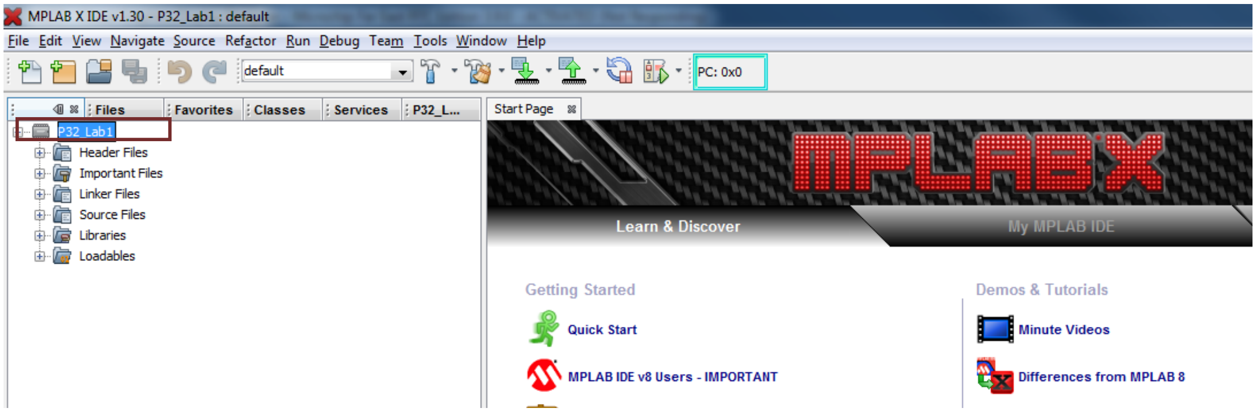


Project Name: P32_Lab1

Project Location: C:\MASTERs\1667\PIC32 Labs\P32_Lab1

Project Folder: C:\MASTERs\1667\PIC32 Labs\P32_Lab1\P32_Lab1.X

Step 7 – New Project Generation



④ Reference for Working with Microchip Graphics Demo Projects

When building applications, there will be programs that are specific to your selected hardware. In the demo code that comes with the Graphics Library, we have created generic programs that are applicable to the multiple hardware platforms supported by the demos. To link these generic programs to specific hardware, we created the HardwareProfile.h file that is specific to the selected hardware.

In the Graphics Demos that are provided with the library, the HardwareProfile.h becomes the hardware profile file selector. The file that is selected will depend on which MCU and hardware platform are used. These hardware dependent files are located in the demo and lab project folders under the **..\<project>\Config** subdirectory.

When you develop your own specific hardware profile, you may opt to directly replace the contents of the provided HardwareProfile.h file with your own hardware specific code.

HardwareProfile.h

```
#ifndef CFG_INCLUDE_MPLAB_X
…
\\ MPLAB v8 support
  …
  #if defined (__PIC24FJ256DA210__)
  \\ #include “Configs\HWP_DA210_BRD_16PMP_QVGAv1.h”
  \\ #include “Configs\HWP_DA210_BRD_16PMP_WQVGAv1.h”
  …
  #endif
…
#else  \\MPLAB®X  support
…
#if defined(CFG_INCLUDE_DA210_BRD_16PMP_QVGAv1)
  #include “..\Configs\HWP_DA210_BRD_16PMP_QVGAv1.h”
#elif defined (CFG_INCLUDE_DA210_BRD_16PMP_WQVGAv1.h”
…
```

Attention

An abbreviation guide is provided with the Microchip Applications Library installation ..\Microchip\Help\abbreviations.html

The above code snippet is taken from the HardwareProfile.h file used in the hands-on labs and shows just the portions of the code that are used to select the correct hardware profile when using a DA210 development platform.

Notice that there are 2 sections to the code. The top section is used by MPLAB 8 IDE users. The value “__PIC24FJ256DA210__” is set by the MPLAB8 IDE when a user selects that particular MCU for their project. The user must then open the HardwareProfile.h file, locate the DA210 section, then uncomment the correct include statement based on the display board used.

The second section is for MPLAB®X ™ IDE users. When using the MPLAB C Compiler for PIC24 or PIC32 in the MPLAB®X ™ environment, we are able to set preprocessor macros. The values “CFG_INCLUDE_MPLAB_X” and “CFG_INCLUDE_DA210_BRD_16PMP_QVGAv1” are set in the project build options for the compiler.

Another benefit of the MPLAB®X ™ IDE environment is the ability to establish multiple build configurations within a single project. Build configurations are used to store common build parameters such as the MCU part number, the compiler toolchain, the debugger and or programmer that will be used, linker settings, include directories, and (in the case of the 16– and 32– bit compilers), the preprocessor macros. For any given project, multiple configurations may be defined. In the graphics demo MPLAB®X projects, we take advantage of this feature by providing multiple configurations for each of the projects. The user simply has to select the correct hardware configuration.

Tools

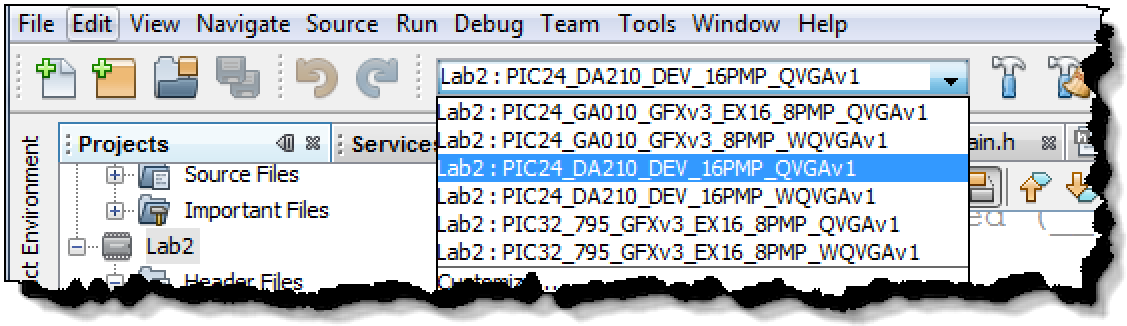
Table H.1
MPLAB®X ™ Configurations for use with Graphics Demos

Configuration	Supported MCU	Supported Display Board
PIC24_GA010_GFXv3_EX16_8PMP_QVGAv1	PIC24FJ128GA010	Truly 3.2” QVGA
PIC24_GA010_GFXv3_EX16_8PMP_WQVGAv1	PIC24FJ128GA010	Powertips 4.3” WQVGA
PIC24_DA210_DEV_16PMP_QVGAv1	PIC24FJ256DA210	Truly 3.2” QVGA
PIC24_DA210_DEV_16PMP_WQVGAv1	PIC24FJ256DA210	Powertips 4.3” WQVGA
PIC32_795_GFXv3_EX16_8PMP_QVGAv1	PIC32MX795F512L	Truly 3.2” QVGA
PIC32_795_GFXv3_EX16_8PMP_WQVGAv1	PIC32MX795F512L	Powertips 4.3” WQVGA

Using Table H.1, select the configuration for the LCC hardware. NOTE: The MPLAB®X ™ IDE appends the project name to the front of the configuration name.

a) In MPLAB®X , set the desired project as the Main Project (refer to Appendix A for details).

b) Choose the desired configuration from the drop down menu in the top toolbar

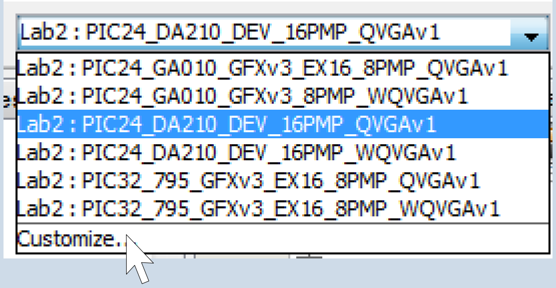

File
Edit
View
Navigate
Source
Run
Debug
Team
Tools
Window
Help
Lab2 : PIC24_DA210_DEV_16PMP_QVGAv1
Lab2 : PIC24_GA010_GFXv3_EX16_8PMP_QVGAv1
Lab2 : PIC24_GA010_GFXv3_8PMP_WQVGAv1
Lab2 : PIC24_DA210_DEV_16PMP_QVGAv1
Lab2 : PIC24_DA210_DEV_16PMP_WQVGAv1
Lab2 : PIC32_795_GFXv3_EX16_8PMP_QVGAv1
Lab2 : PIC32_795_GFXv3_EX16_8PMP_WQVGAv1
Projects
Services
ct Environment
Source Files
Important Files
Lab2
Header Files
ain.h

YOUR PROJECT IS NOW CONFIGURED AND YOU ARE READY TO BEGIN WORKING ON YOUR CODE. Happy Coding!!

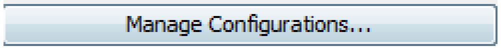
Customizing Configurations

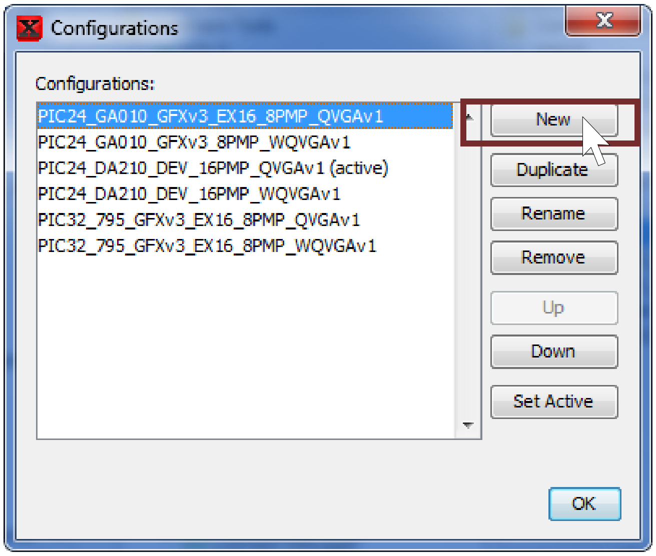
The configurations may be customized to add additional hardware platforms or to support your custom hardware.

1 Select **Customize...** from the bottom of the configuration list.


Lab2 : PIC24_DA210_DEV_16PMP_QVGAv1
Lab2 : PIC24_GA010_GFXv3_EX16_8PMP_QVGAv1
Lab2 : PIC24_GA010_GFXv3_8PMP_WQVGAv1
Lab2 : PIC24_DA210_DEV_16PMP_QVGAv1
Lab2 : PIC24_DA210_DEV_16PMP_WQVGAv1
Lab2 : PIC32_795_GFXv3_EX16_8PMP_QVGAv1
Lab2 : PIC32_795_GFXv3_EX16_8PMP_WQVGAv1
Customize...

2 When the project properties box appears, choose the **Manage Configurations** button in the lower left corner.

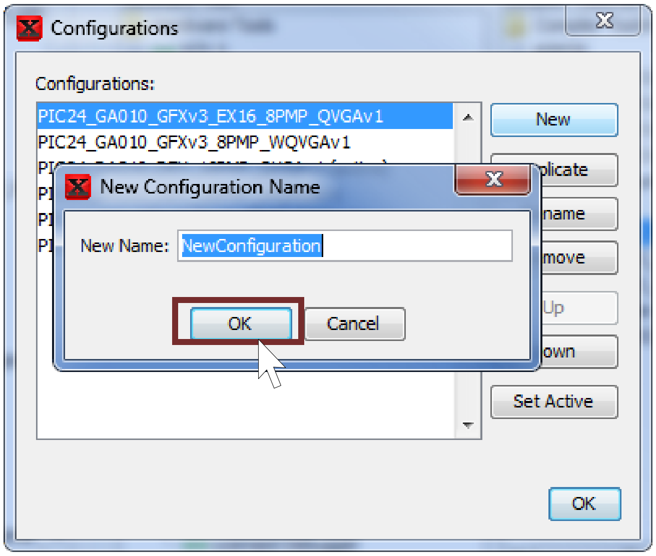



Configurations

Configurations:

PIC24_GA010_GFXv3_EX16_8PMP_QVGAv1
PIC24_GA010_GFXv3_8PMP_WQVGAv1
PIC24_DA210_DEV_16PMP_QVGAv1 (active)
PIC24_DA210_DEV_16PMP_WQVGAv1
PIC32_795_GFXv3_EX16_8PMP_QVGAv1
PIC32_795_GFXv3_EX16_8PMP_WQVGAv1

③ In the Configurations window, choose **New**


Configurations

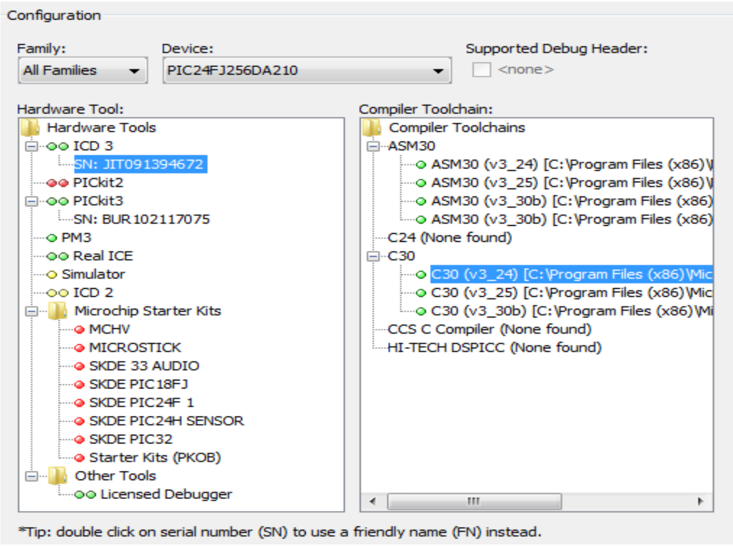
Configurations:

PIC24_GA010_GFXv3_EX16_8PMP_QVGAv1
PIC24_GA010_GFXv3_8PMP_WQVGAv1

New Configuration Name

New Name: NewConfiguration

④ Name the configuration and select OK


Configuration

Family: All Families
Device: PIC24FJ256DA210
Supported Debug Header: ☐ <none>

Hardware Tool:
Hardware Tools
ICD 3
SN: JIT091394672
PICkit2
PICkit3
SN: BUR102117075
PM3
Real ICE
Simulator
ICD 2
Microchip Starter Kits
MCHV
MICROSTICK
SKDE 33 AUDIO
SKDE PIC18FJ
SKDE PIC24F 1
SKDE PIC24H SENSOR
SKDE PIC32
Starter Kits (PKOB)
Other Tools
Licensed Debugger

Compiler Toolchain:
Compiler Toolchains
ASM30
ASM30 (v3_24) [C:\Program Files (x86)\
ASM30 (v3_25) [C:\Program Files (x86)\
ASM30 (v3_30b) [C:\Program Files (x86)
ASM30 (v3_30b) [C:\Program Files (x86)
C24 (None found)
C30
C30 (v3_24) [C:\Program Files (x86)\Mic
C30 (v3_25) [C:\Program Files (x86)\Mic
C30 (v3_30b) [C:\Program Files (x86)\Mi
CCS C Compiler (None found)
HI-TECH DSPICC (None found)

*Tip: double click on serial number (SN) to use a friendly name (FN) instead.

⑤ In the project properties window, select the desired properties. Additional build options may be set by clicking on the compiler or linker name under the new created configuration. Please see the MPLAB®X ™ help files for more information.

This page intentionally left blank

Lab 1

Creating the Splash Screen

Purpose

In this lab, you will demonstrate your ability to use VGDD to implement the library's primitive layer APIs as we create a splash screen for our HMI.

Solution files may be found at:

C:\RTC\HIF2132A\Solution Files

Tools

- MPLAB®X IDE v1.20 or higher
- Visual Graphics Display Designer (VGDD) v3.7 or higher
- REAL ICE™ or MPLAB ICD 3
- Graphics LCD Controller PICtail ™ Plus LCC Board
- PIC32 USB Starter Kit II
- Truly 3.2" QVGA LCD Display
- If using **PIC32**, you will need to use **Microchip® XC32 Compiler**

Objectives

1) Use the VGDD utility to generate font, image and application source files.
2) Via VGDD, use the Microchip Graphics Library Primitive Layer (GPL) to render images, text strings and shapes to the graphics LCD display.
3) Explore how gradients and transparency can be used to enhance the splash screen.

Expected Results

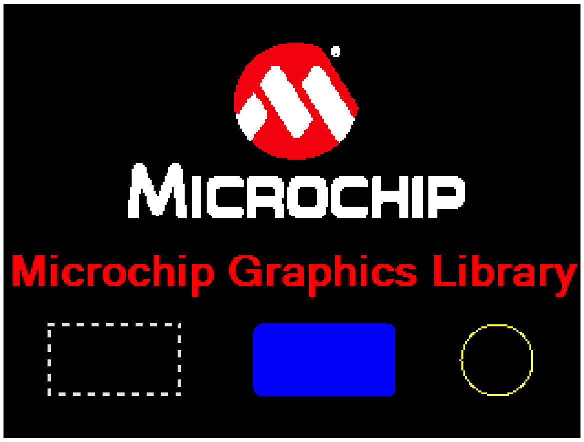


Figure 1.1
When the lab is fully completed, you should see this on your LCD panel.

Information

Hands-On Labs:
C:\RTC\HIF2132A\PIC32_Labs
Font Files:
All fonts used in this lab have been preinstalled on the lab computers
Image Files:
..\RTC\HIF2132A\Images
Visual Graphics Display Designer:
..\RTC\HIF2132A\VGDD.exe
Icon Files:
..\RTC\HIF2132A\Icons

Procedure

In lab 1, we will create the splash screen shown in figure 1.1. The information box to the left is provided to help you navigate the lab directory. For this lab you will using either, depending on the development board you have chosen:

C:\RTC\HIF2132A\PIC32_Labs\P32_Lab1\Lab1_PIC32.X

1 Launch Visual Graphics Display Designer

If it is not already open, please launch VGDD
C:\RTC\HIF2132A\VGDD.exe

VGDD

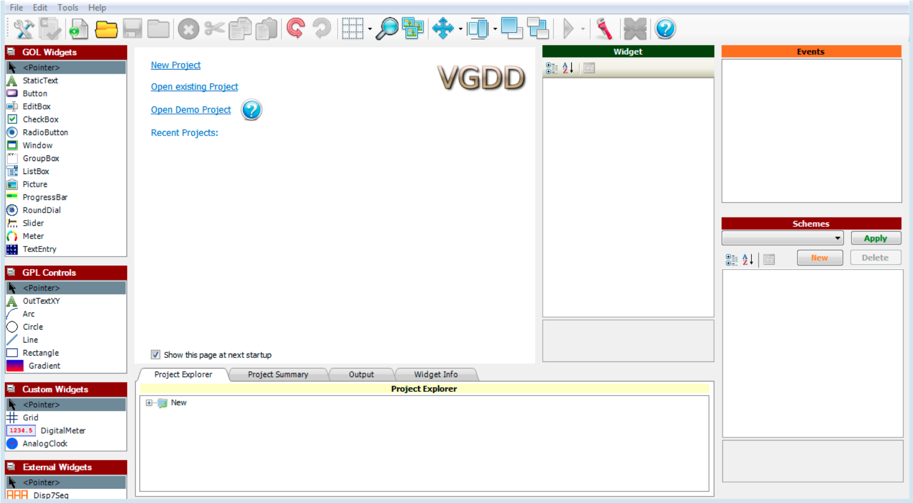
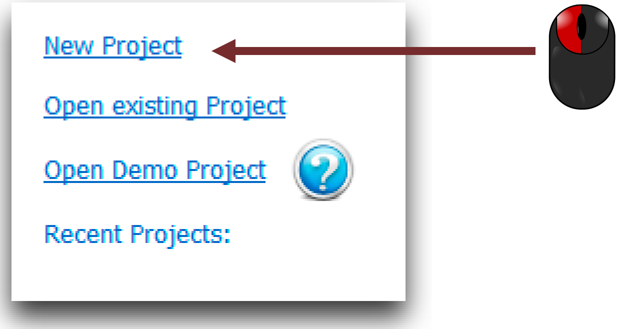
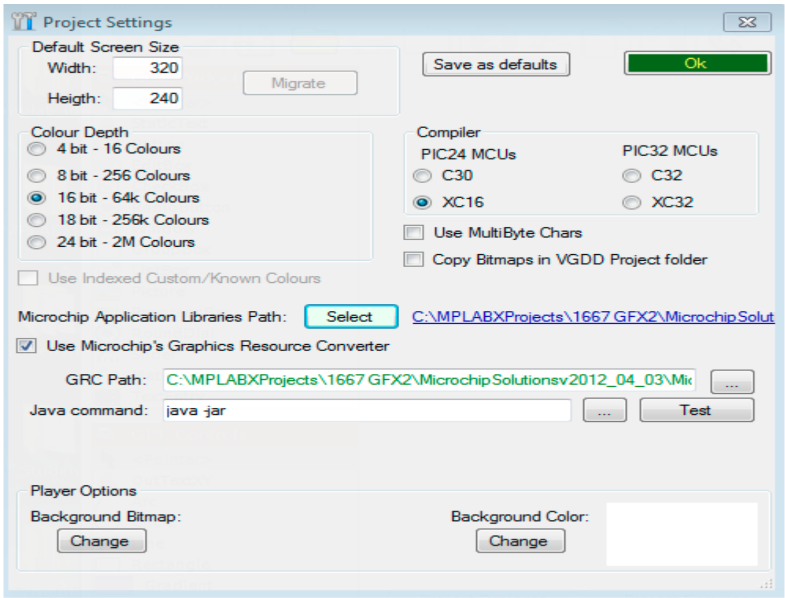


Figure 1.2
VGDD window when it is first opened.

2 Create a new project

Click on New Project in the VGDD window





Default Screen Size
Width: 320
Heigth: 240

Colour Depth
○ 4 bit - 16 Colours
○ 8 bit - 256 Colours
● 16 bit - 64k Colours
○ 18 bit - 256k Colours
○ 24 bit - 2M Colours
☐ Use Indexed Custom/Known Colours

Compiler
PIC24 MCUs
○ C30
● XC16
PIC32 MCUs
○ C32
○ XC32
☐ Use MultiByte Chars
☐ Copy Bitmaps in VGDD Project folder

Microchip Application Libraries Path: C:\MPLABXProjects\1667 GFX2\MicrochipSolut
☑ Use Microchip's Graphics Resource Converter
GRC Path: C:\MPLABXProjects\1667 GFX2\MicrochipSolutionsv2012_04_03\Mic
Java command: java -jar

Player Options
Background Bitmap:
Background Color:

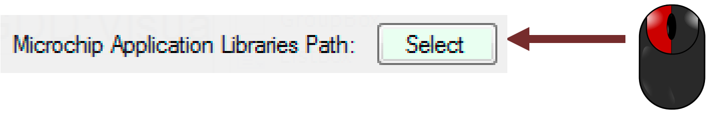
Figure 1.3
Clicking New Project launches the Project settings screen.

Information

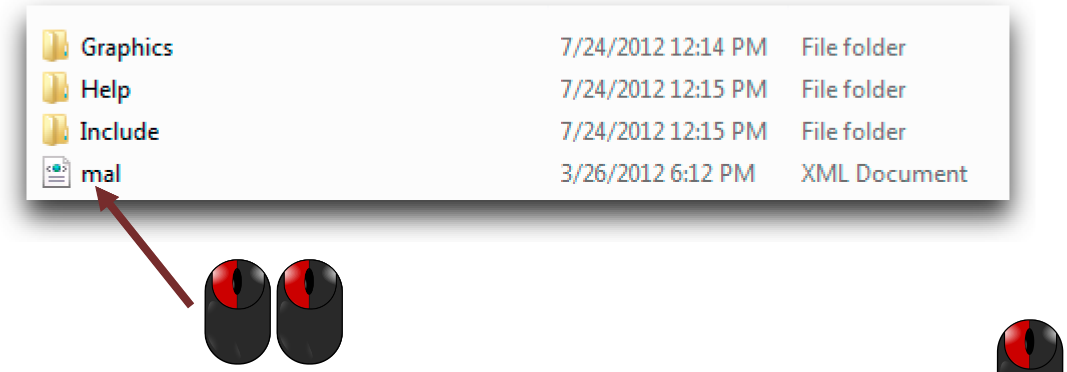
You may return to project settings at anytime by selecting the Project Settings icon in the VGDD toolbar.

3 Set up the path for Microchip's Applications Library (MLA).

Near the middle of the **Project Settings** window, locate the **Microchip Applications Library Path** option and click the **Select** button next to it.

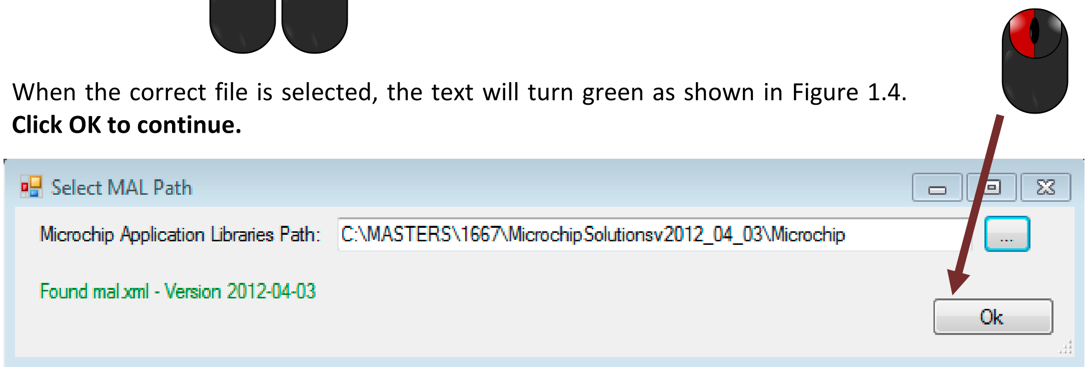


Navigate to
C:\RTC\HIF2132A\MicrochipSolutionsv2012_04_03\Microchip
And double click to select **mal.xml**



Graphics	7/24/2012 12:14 PM	File folder
Help	7/24/2012 12:15 PM	File folder
Include	7/24/2012 12:15 PM	File folder
mal	3/26/2012 6:12 PM	XML Document

When the correct file is selected, the text will turn green as shown in Figure 1.4.
Click OK to continue.



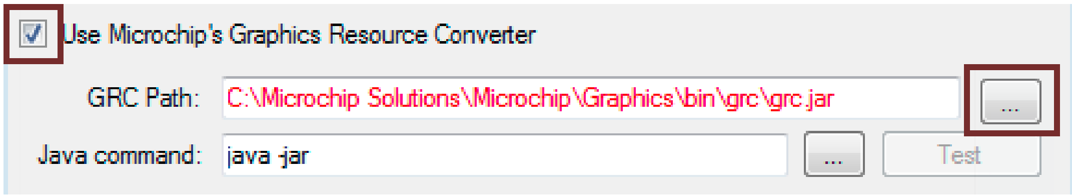
Microchip Application Libraries Path: C:\MASTERS\1667\MicrochipSolutionsv2012_04_03\Microchip

Found mal.xml - Version 2012-04-03

Figure 1.4
Mal.xml was found

④ Set up the path for Microchip's Graphics Resource Converter

Before fonts and images may be used by the Microchip Graphics Library, they must be converted into formats the library can use. VGDD has a built in conversion utility; however, we will be using Microchip's Graphics Resource Converter (GRC). If it is not checked, please check the box to "Use Microchip's Graphics Resource Converter".



☑ Use Microchip's Graphics Resource Converter

GRC Path: C:\Microchip Solutions\Microchip\Graphics\bin\grc\grc.jar

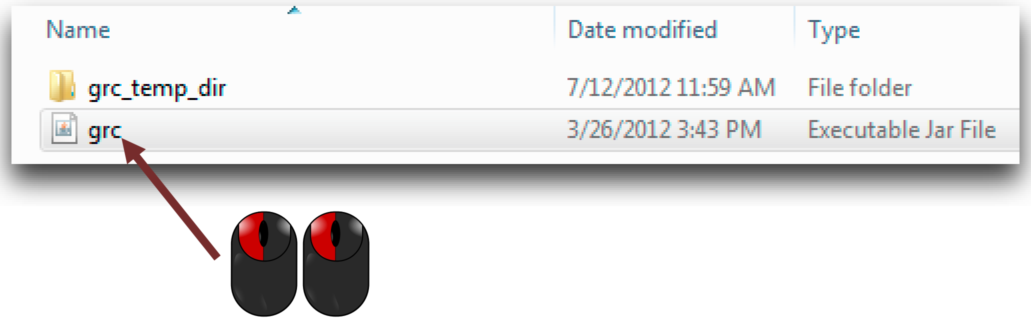
Java command: java -jar

Figure 1.5
GRC settings

The red text indicates that VGDD is not able to find Microchip's GRC application. Next, we need to set up the path where the Microchip GRC application may be found. **Select the ellipsis next to the path name and navigate to :**

C:\RTC\HIF2132A\MicrochipSolutionsv2012_04_03\Microchip\Graphics\bin\grc

Double click to select the **grc.jar** application file



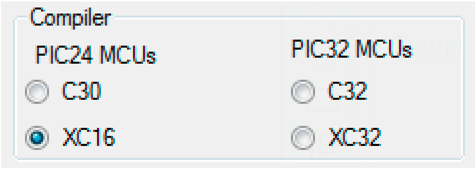
Name	Date modified	Type
grc_temp_dir	7/12/2012 11:59 AM	File folder
grc	3/26/2012 3:43 PM	Executable Jar File

⑤ Select the compiler

For the purpose of this class, the MPLAB® XC compilers will be used. If you are using a PIC24, please choose XC16. If you are using a PIC32, please choose XC32.

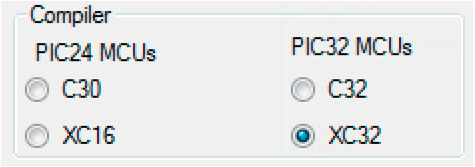
Attention

The compiler version selected will impact the format for the converted font and image files. Selecting the wrong compiler will result in compiler errors.



Compiler
PIC24 MCUs: ○ C30 ● XC16
PIC32 MCUs: ○ C32 ○ XC32

Figure 1.6
If using PIC24, choose XC16



Compiler
PIC24 MCUs: ○ C30 ○ XC16
PIC32 MCUs: ○ C32 ● XC32

Figure 1.7
If using PIC32, choose XC32

6 Set up the display parameters.

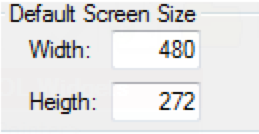
Microchip Graphics Library uses information in **HardwareProfile.h** to communicate with the display controller drivers. When VGDD generates the source code, **HardwareProfile.h** is populated with these values.

Attention

These settings will be added to the **HardwareProfile.h** file to properly orient the display. Incorrect settings will result in odd looking displays.

Change the default screen settings to match your display.

Graphics Display Powertip 3.2" display
Width = 320
Height = 240


Default Screen Size
Width: 480
Heigth: 272

Choose the color depth for the controller you will use in your application. The drivers used in this class support a maximum of 16 bpp.

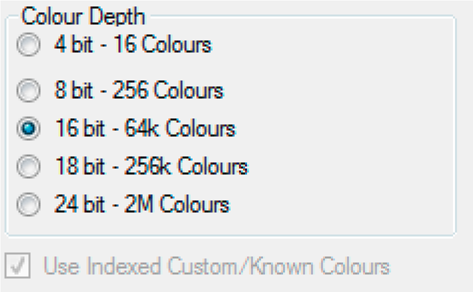

Colour Depth
○ 4 bit - 16 Colours
○ 8 bit - 256 Colours
● 16 bit - 64k Colours
○ 18 bit - 256k Colours
○ 24 bit - 2M Colours
☑ Use Indexed Custom/Known Colours

Figure 1.8
Color depth selection

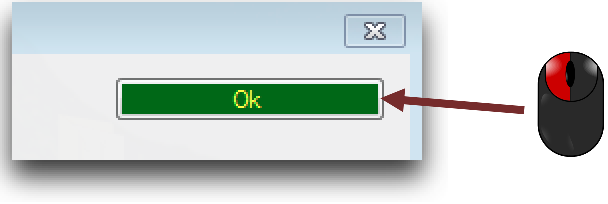
Information

When choosing 4-bit or 8-bit color, palettes (aka indexed color) are used. VGDD will set up these palettes for use in the application.

For applications that employ a graphics driver capable of 18– or 24– bit color, you may select those options here.

7 Save the settings and proceed to screen design.

When you are happy with the settings click Ok.



Information

The paths provided for GRC and the MLA will be used the next time you create a project.

The options at the bottom of the screen are provided so you may customize the VGDD player (screen simulator). For the purpose of this class, we will work with the default options.

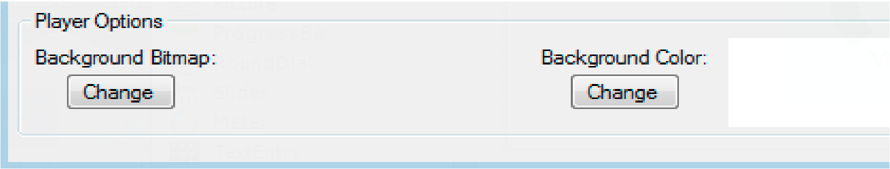
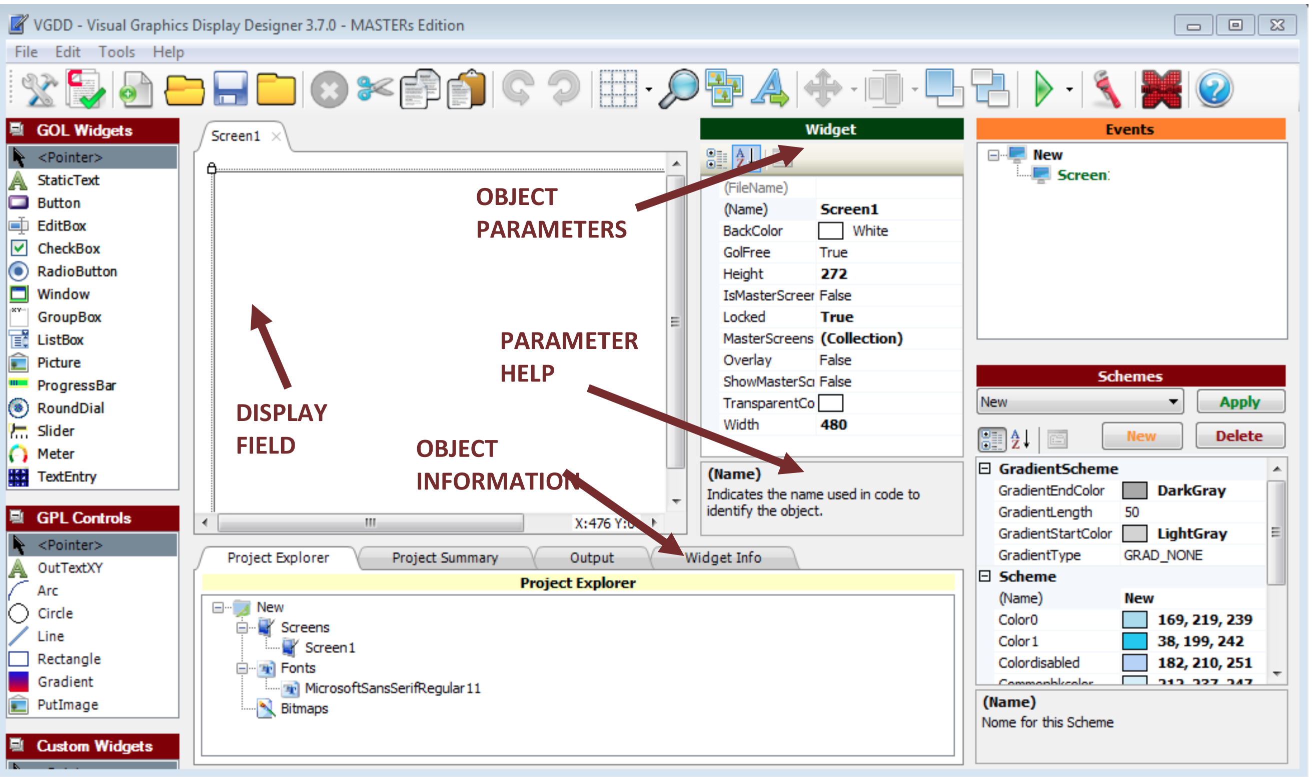

Player Options
Background Bitmap: Change
Background Color: Change

Figure 1.9
VGDD simulator options



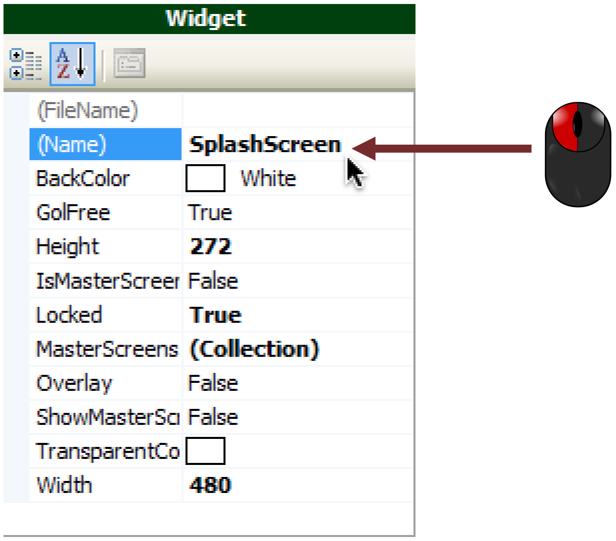
(Name)
Indicates the name used in code to identify the object.

Figure 1.10
VGDD screen design environment

8 Change the screen name

VGDD will use the screen name in the code. To help make our code more readable, we will change the name to **SplashScreen**.

In the object parameter box (labeled Widget), select the **(Name)** parameter. Left click in the in the name field. When the cursor appears, change the value to Splash Screen.



(FileName)	
(Name)	SplashScreen
BackColor	White
GolFree	True
Height	272
IsMasterScreer	False
Locked	True
MasterScreens	(Collection)
Overlay	False
ShowMasterSc	False
TransparentCo	
Width	480

Information

You will use this same method to change Widget names when we start working on the labs.

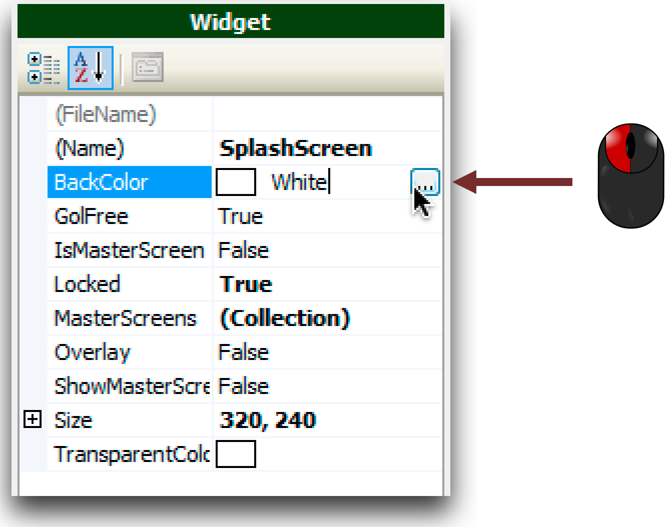
The value in name field is the value that will be used in the code.

9 Change the screen background color

In the object parameter box (labeled Widget), select the **BackColor** parameter. Left click in the in the field to the right of the parameter name, then click the ellipsis that appear to the far right of the field.

Attention

The ellipsis will not be present until the **BackColor** property has been selected.

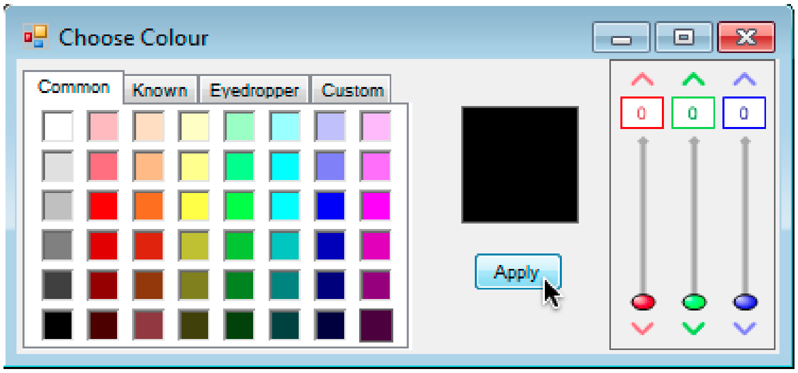


(FileName)	
(Name)	SplashScreen
BackColor	White
GolFree	True
IsMasterScreen	False
Locked	True
MasterScreens	(Collection)
Overlay	False
ShowMasterScre	False
Size	320, 240
TransparentColc	

Information

Other parameters will behave the same way. When in doubt, click the parameter name to find out what options might be available. In some cases, you'll be able to choose from a dropdown menu. In others, the ellipsis will open a selection (chooser) window.

When the **Choose Colour** window pops up, select the desired screen color then click Apply. (RGB = 0,0,0 for Black)



Information

The **Known** tab is used to store indexed color values from Microchip's color definitions file.

The **Eyedropper** can be used to select colors from anywhere on the PC monitor.

The **Custom** tab is used to load previously saved color definition files.

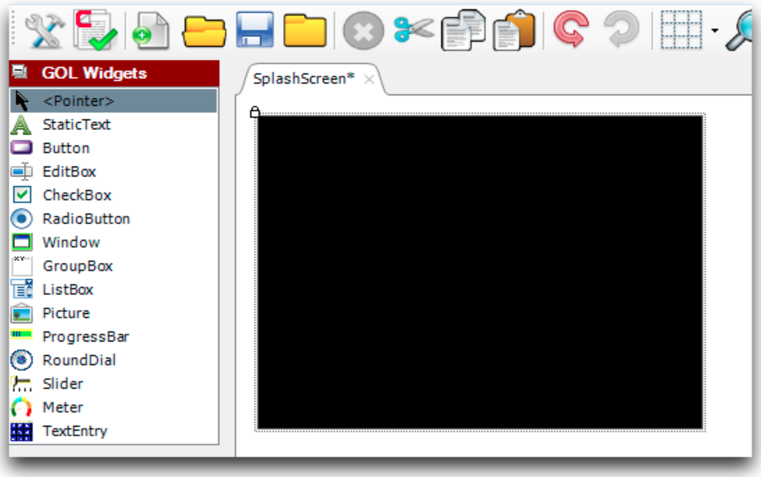


Figure 1.11
Display in VGDD after clicking Apply

⑩ Add the Microchip logo to the screen.

To add the image, we will add a **PutImage** primitive to the screen.

To add a **PutImage** primitive, **select PutImage from the GPL Controls box then double click in the display field.**

Information

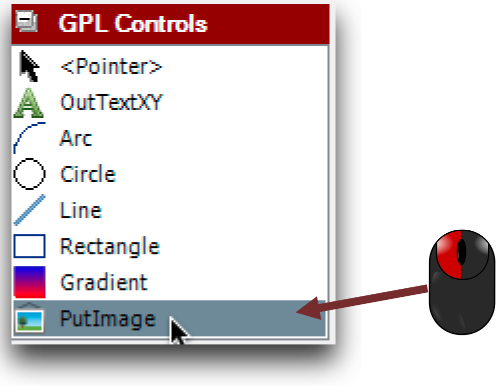
VGDD provides 3 options for adding objects to the screen.

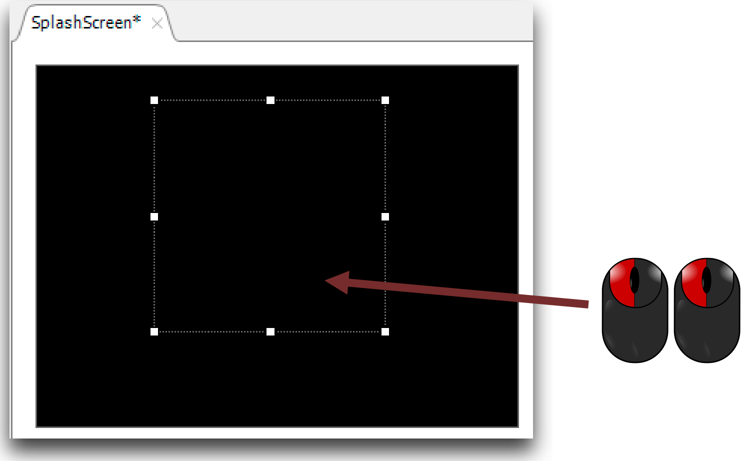
1 — Select the object then double click in the display field.

2 — Select the object, then click and drag in the display field to size the object.

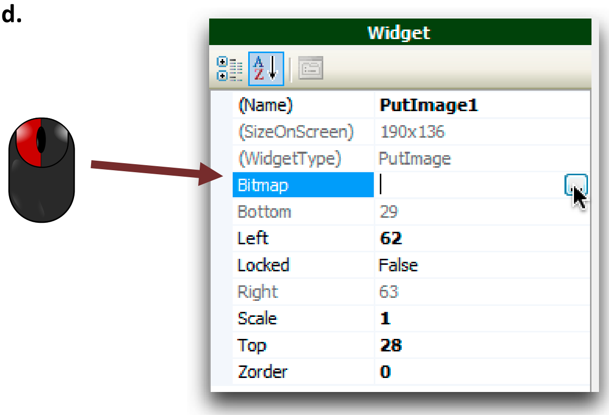
3 — Double click the object.

No matter the option you choose to add the widget, you will be able to resize it later by dragging





With the **PutImage** primitive selected, click the Bitmap field in the Widget properties box and **click on the ellipsis that appear on the far right of the field.**



(Name)	**PutImage1**
(SizeOnScreen)	190x136
(WidgetType)	PutImage
Bitmap	
Bottom	29
Left	**62**
Locked	False
Right	63
Scale	**1**
Top	**28**
Zorder	**0**

Attention

When the **PutImage** primitive is added, the Bitmap field will be highlighted and blink several times to catch your attention. If you inadvertently click somewhere else in the screen, the ellipsis to the right of the bitmap field will disappear. Clicking on the Bitmap field will cause the ellipsis to reappear.

Clicking the ellipsis will open the Bitmap Chooser window where you may select from all the images that have been added to the project. For this exercise, we will need to add an image to use on our splash screen.

Click Add Add

And navigate to **C:\RTC\HIF2132A\Images.** Select the **Logo_black_2bpp** file.

Information

You may open the Bitmap Chooser window at anytime by clicking the Bitmap Chooser icon in the toolbar

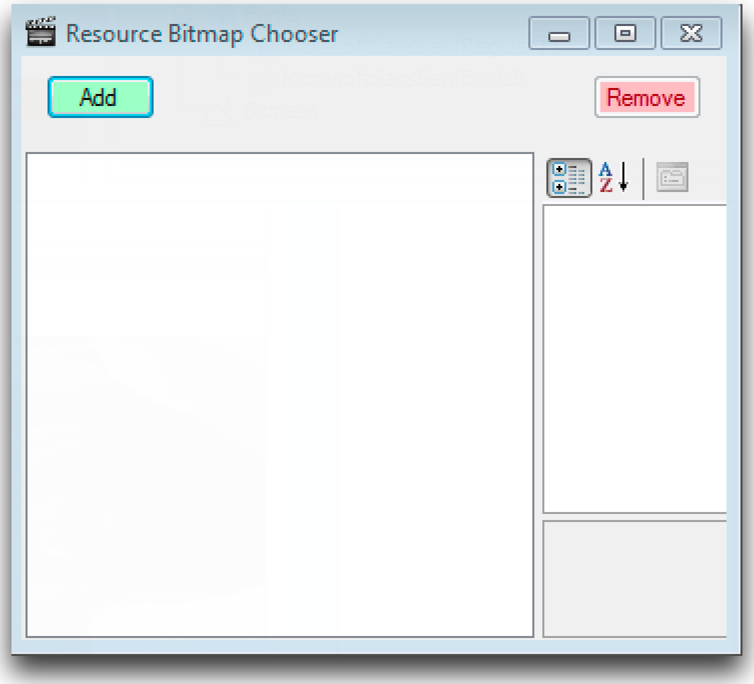


Figure 1.12
Bitmap Chooser

Information

As more images are added to the project, they will be sorted into **IN USE** and **NOT IN USE** types. To avoid wasting memory resources, images in the **NOT IN USE** type will NOT be converted for application use.

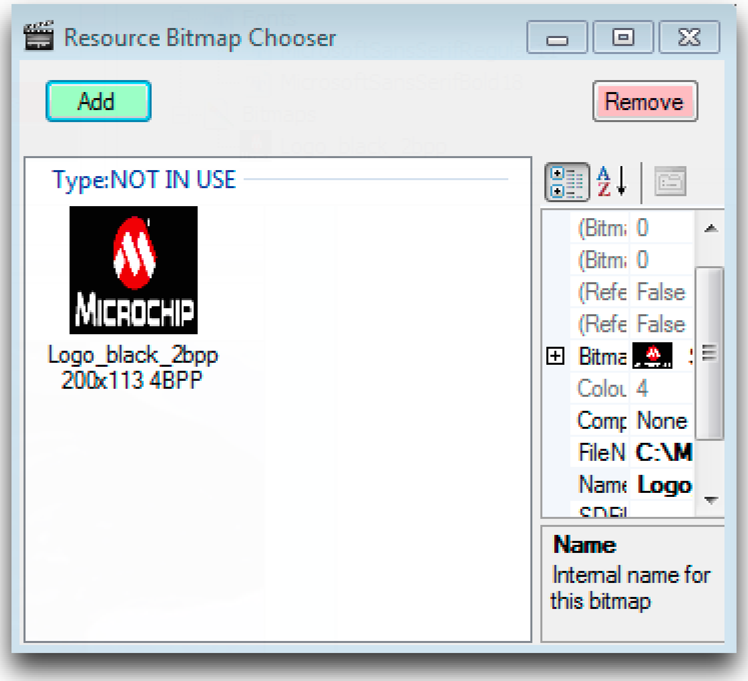
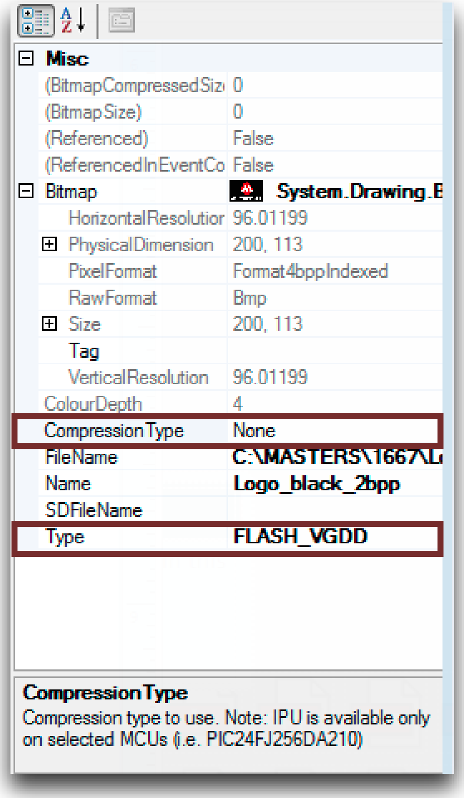


Figure 1.13
Bitmap Chooser with Logo file added



Misc	
(BitmapCompressedSiz	0
(BitmapSize)	0
(Referenced)	False
(ReferencedInEventCo	False
Bitmap	**System.Drawing.E**
HorizontalResolutior	96.01199
PhysicalDimension	200, 113
PixelFormat	Format4bppIndexed
RawFormat	Bmp
Size	200, 113
Tag	
VerticalResolution	96.01199
ColourDepth	4
CompressionType	None
FileName	**C:\MASTERS\1667\L**
Name	**Logo_black_2bpp**
SDFileName	
Type	**FLASH_VGDD**

CompressionType
Compression type to use. Note: IPU is available only on selected MCUs (i.e. PIC24FJ256DA210)

Information

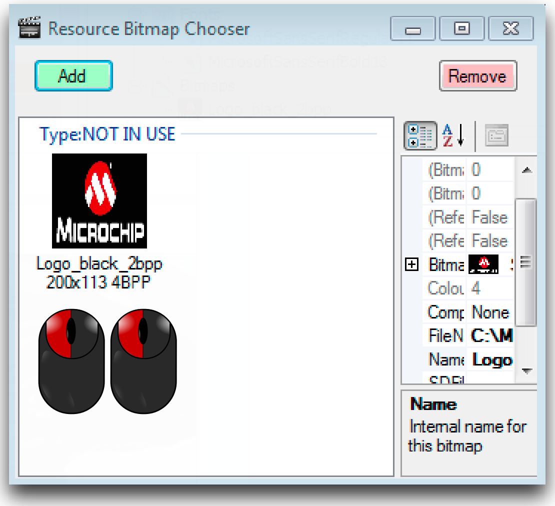
Bitmap properties for a selected image are shown in the field to the right of the image list. In this image property box, we can add a compression algorithm to the image by selecting the **Compression Type** field and choosing a value from the drop down menu.

We may also choose the storage location by selecting the Type field and choosing a value from the drop down menu. The storage location options are:

FLASH_VGDD: Fonts will be stored in internal FLASH and all declarations and definitions will be included in the generated code.
FLASH: Fonts will be stored in internal FLASH; however, the generated code will not include font definitions. Use this option only if you need to use an external font converter (e.g. Microchip Graphics Converter).
EXTERNAL: Use this option to generate a bin file so fonts may be stored in external memory. The bin file name is set in the **BinFileName** field.

Verify the settings are as shown in Figure 1.14. To add the image to the screen, double click the image in the Bitmap chooser window.

Figure 1.14
Image property box



Logo_black_2bpp
200x113 4BPP

(Bitm	0
(Bitm	0
(Refe	False
(Refe	False
Bitma	
Colou	4
Comp	None
FileN	**C:\M**
Name	**Logo**

Name
Internal name for this bitmap

Information

When you add a bitmap to an object within VGDD, the object will automatically resize to match the shape of the bitmap file selected.

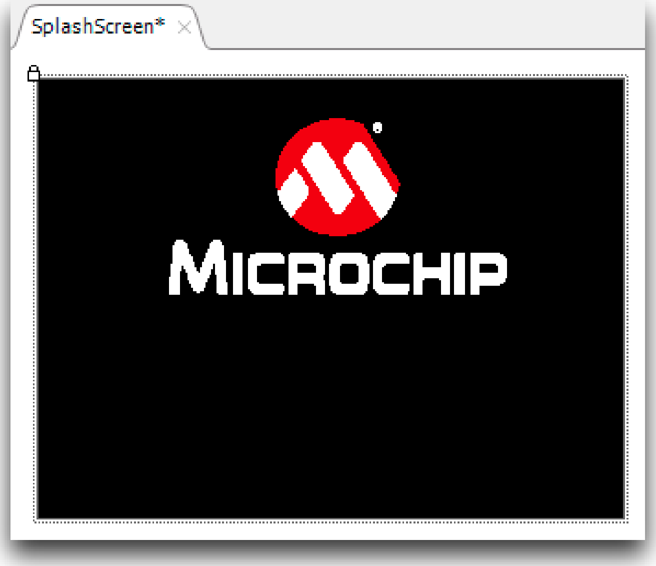
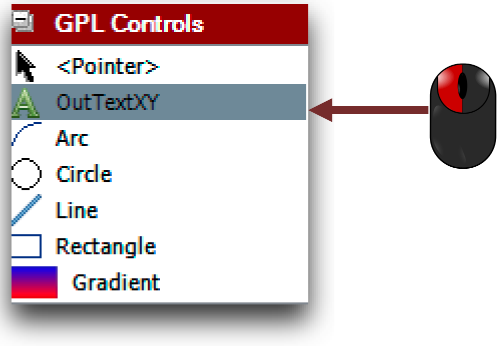


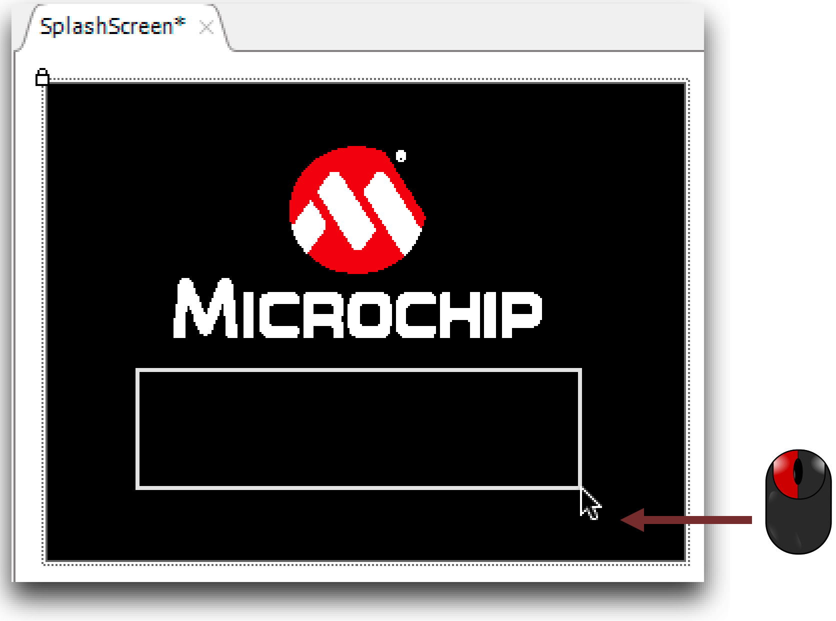
Figure 1.15
VGDD display after the image is added to the Picture widget.

⑪ **Add the text string to the screen.**

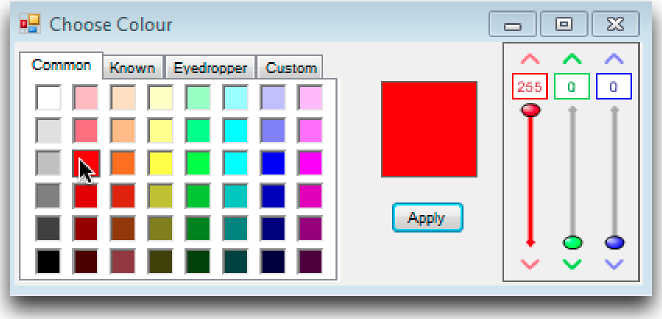
Now we will add a text string using the **OutTextXY** primitive object. In the GPL Controls box, select OutTextXY.



Left click in the display field and drag the mouse to size the text box. Don't worry if it isn't correct yet. We will size it correctly in a few moments.



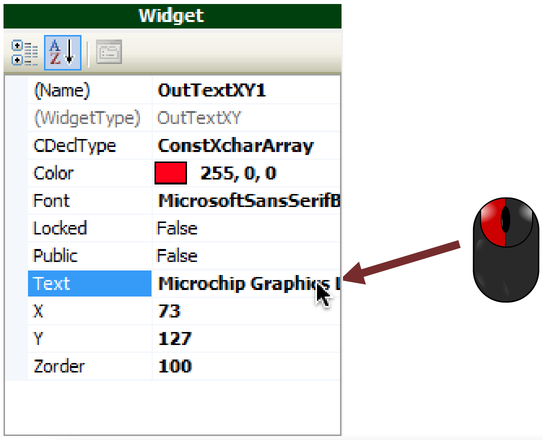
With the text box selected, **choose Color parameter in the Widget box and click the ellipsis to use the Color Chooser window to change the string color to Red (RGB = 255,0,0)** When the desired color is displayed, **click Apply**.



Attention

The ellipsis will not be present until the **Color** property has been selected.

To change the string value, click the **Text** property in the Widget box. In the field to the right, change the text to **Microchip Graphics Library**.



Widget

(Name)	OutTextXY1
(WidgetType)	OutTextXY
CDeclType	ConstXcharArray
Color	🟥 255, 0, 0
Font	MicrosoftSansSerifB
Locked	False
Public	False
Text	Microchip Graphics I
X	73
Y	127
Zorder	100

Information

If you wish to explore resizing the OutTextXY box, use the mouse to drag the handles until the object is the desired size.

To move an object, left click the mouse in the middle of the object until the cross sign appears then drag the object to the desired position. You may also "nudge" objects into position using the arrow keys on the keyboard.

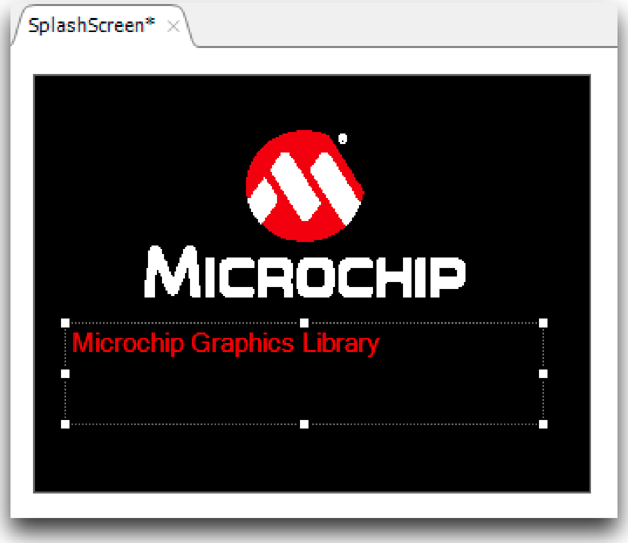


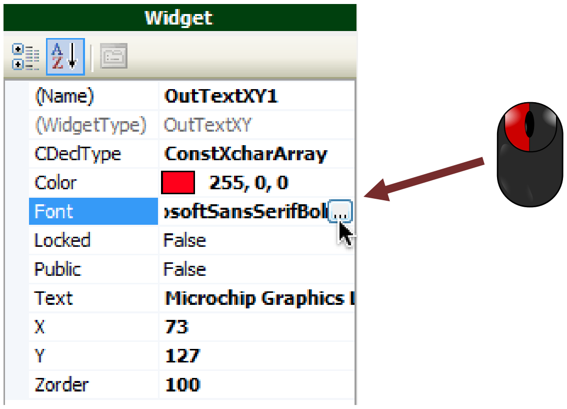
Figure 1.16
Display in VGDD with text added.

⑫ Change the font used to draw the text string

The default font is too small, so we need to change it. **Click the Font property in the Widget box; then click the ellipsis that appear to the right.**

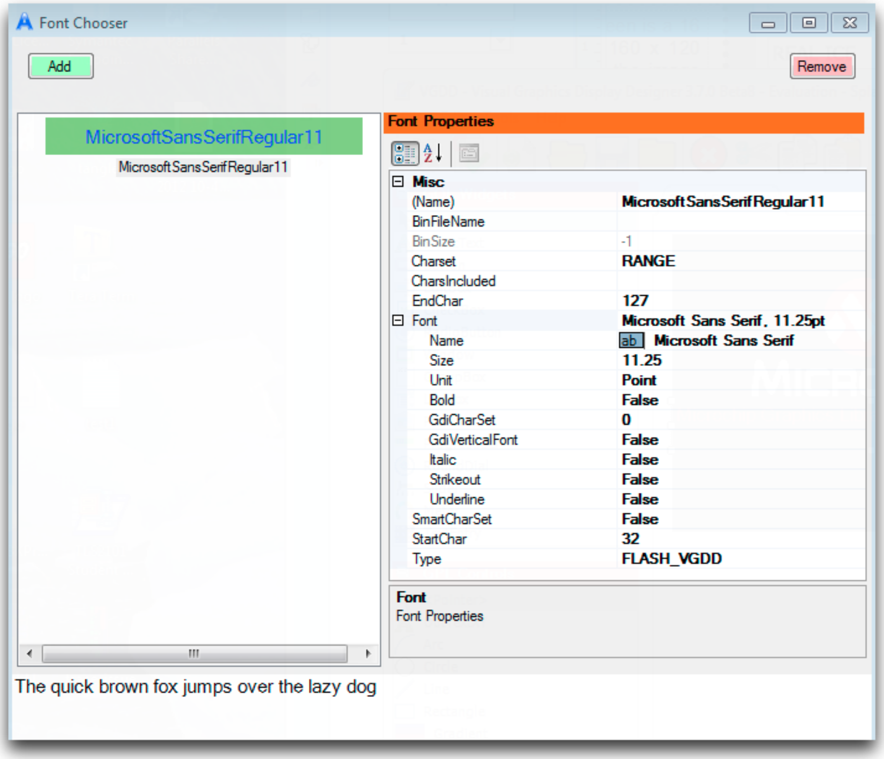
Information

The font used for Widgets (GOL, Custom, or External) is set in the Scheme box. We will explore Style Schemes in Lab 2.



Widget

(Name)	OutTextXY1
(WidgetType)	OutTextXY
CDeclType	ConstXcharArray
Color	🟥 255, 0, 0
Font	›softSansSerifBol ...
Locked	False
Public	False
Text	Microchip Graphics I
X	73
Y	127
Zorder	100



Misc

(Name)	MicrosoftSansSerifRegular11
BinFileName	
BinSize	-1
Charset	RANGE
CharsIncluded	
EndChar	127
Font	Microsoft Sans Serif, 11.25pt
Name	Microsoft Sans Serif
Size	11.25
Unit	Point
Bold	False
GdiCharSet	0
GdiVerticalFont	False
Italic	False
Strikeout	False
Underline	False
SmartCharSet	False
StartChar	32
Type	FLASH_VGDD

Font
Font Properties

The quick brown fox jumps over the lazy dog

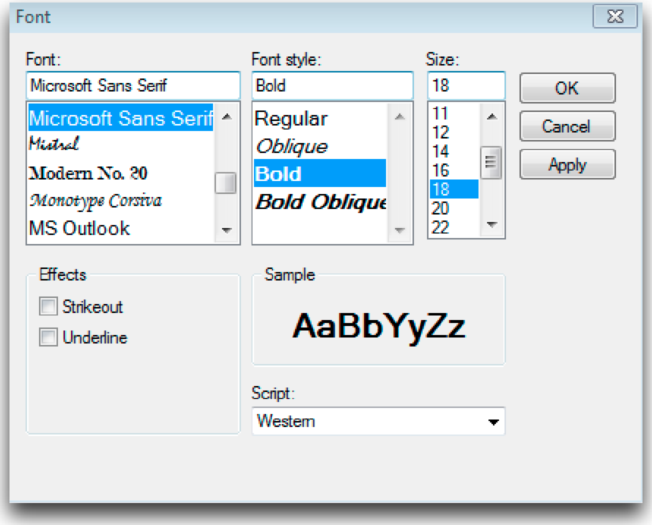
Figure 1.17
Font Chooser window

Information

You may open the **Font Chooser** window at anytime by clicking the **Font Chooser** icon in the toolbar

In the Font Chooser window, we can either add a new font or change the existing font to meet our needs. For the purpose of this exercise, we will add a new font from the available installed fonts on the computer.

Click the Add button in the Font Chooser window. Add



Font: Microsoft Sans Serif
Font style: Bold
Size: 18

Font list: Microsoft Sans Serif, Mistral, Modern No. 20, Monotype Corsiva, MS Outlook
Font style list: Regular, Oblique, Bold, Bold Oblique
Size list: 11, 12, 14, 16, 18, 20, 22

☐ Strikeout
☐ Underline

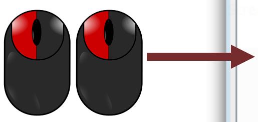
Script: Western

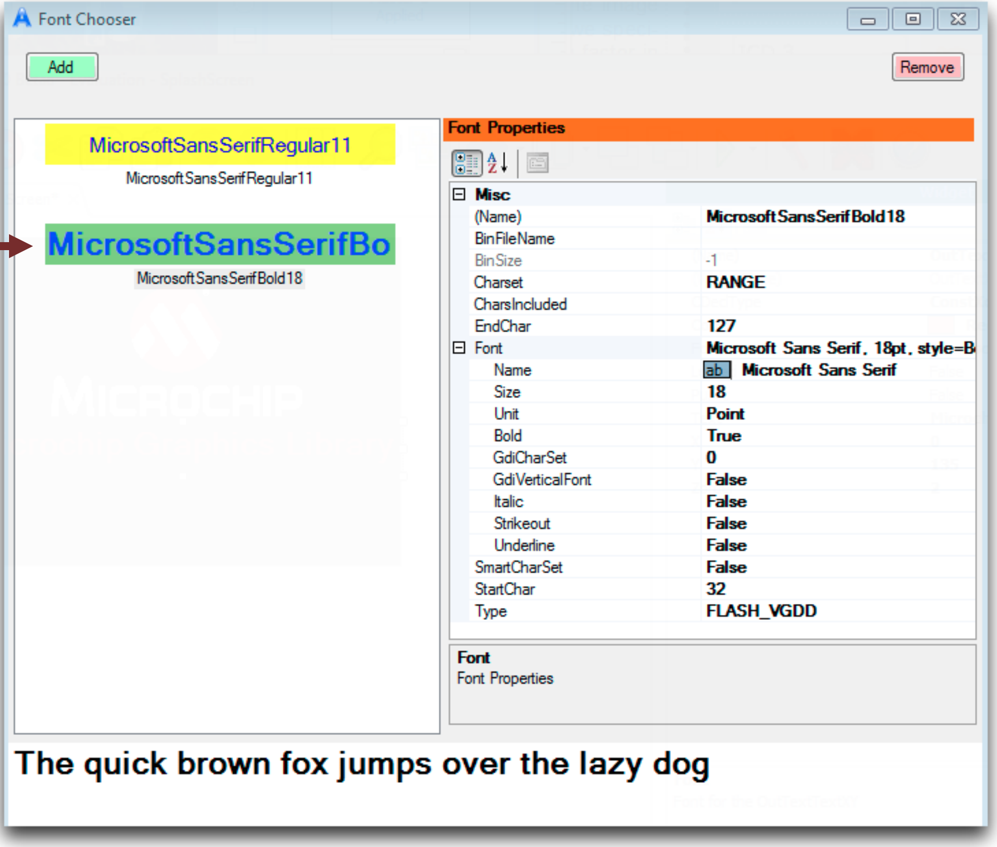
Figure 1.18
Listing of installed fonts

Attention

VGDD is only able to access pre-installed fonts. Please refer to Windows Help files for instructions on how to install a font.

In the list of installed fonts, **choose a font**, set it's style and size, then click OK. The font we need is **Microsoft Sans Serif, Bold, 18 points**.





Misc	
(Name)	MicrosoftSansSerifBold18
BinFileName	
BinSize	-1
Charset	RANGE
CharsIncluded	
EndChar	127
Font	Microsoft Sans Serif, 18pt, style=B
Name	Microsoft Sans Serif
Size	18
Unit	Point
Bold	True
GdiCharSet	0
GdiVerticalFont	False
Italic	False
Strikeout	False
Underline	False
SmartCharSet	False
StartChar	32
Type	FLASH_VGDD

Font
Font Properties

The quick brown fox jumps over the lazy dog

Figure 1.19
Font Chooser window after new font added.

Information

To change the storage location of the font table, use the Type field in the font chooser window. The available types are:

FLASH_VGDD: Fonts will be stored in internal FLASH and all declarations and definitions will be included in the generated code.
FLASH: Fonts will be stored in internal FLASH; however, the generated code will not include font definitions. Use this option only if you need to use an external font converter (e.g. Microchip Graphics Converter).
EXTERNAL: Use this option to generate a bin file so fonts may be stored in external memory. The bin file name is set in the **BinFileName** field.

Double click the newly added font to apply it to the string shown in the selected **OutTextXY** widget on the display. If needed, use the widget handles to resize the text box.

VGDD offers two options to help you reduce the memory footprint required for the font. First, you may choose **RANGE** in the **Charset** parameter. With this option, only the ASCII characters between **StartChar** and **EndChar** will be included in the generated font table. Second, you may choose **SELECTION**. This option is more complicated, but employs a font filter to limit the font table to necessary characters as determined by the strings used in the VGDD project. More details on these options are discussed in on the VGDD Wiki Help page that is accessed by clicking the help icon in the toolbar.

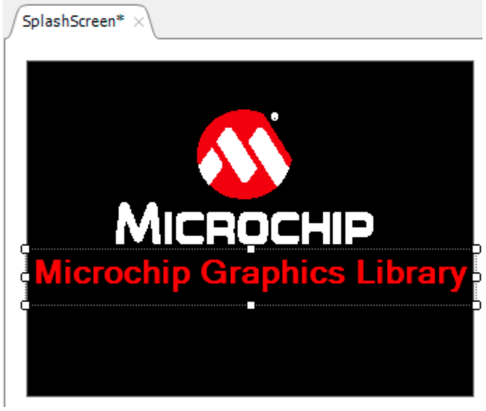
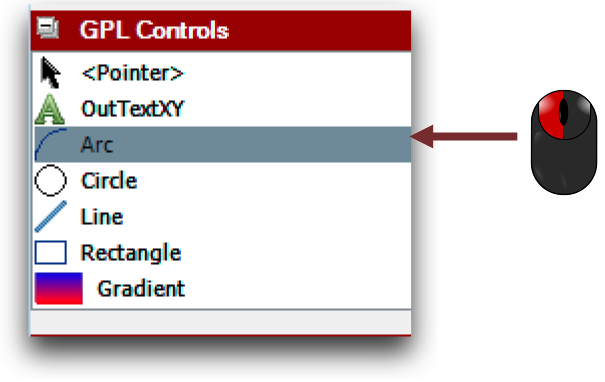
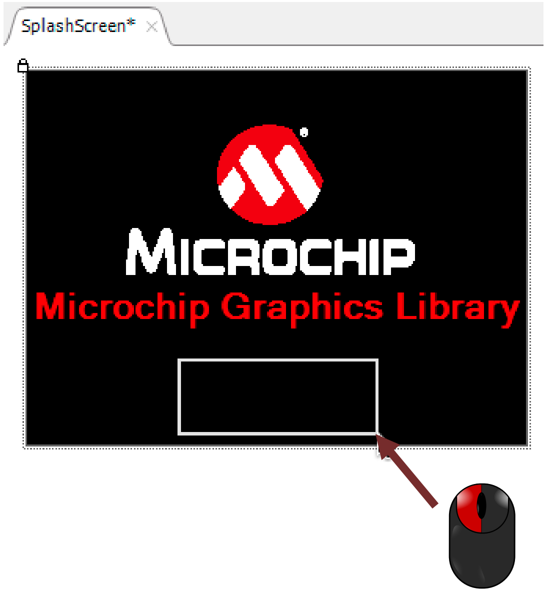


Figure 1.20
VGDD display with the new font applied

⑬ Add primitive shapes and lines.

Using the GPL controls, we will now add the primitive shapes and decorative lines to the screen. First, we will draw the filled bevel using the **Arc** control. **Select the Arc, then left click and drag in the display field to size the bevel.**





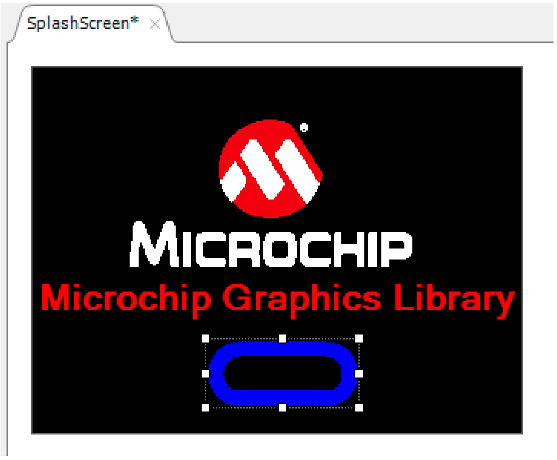
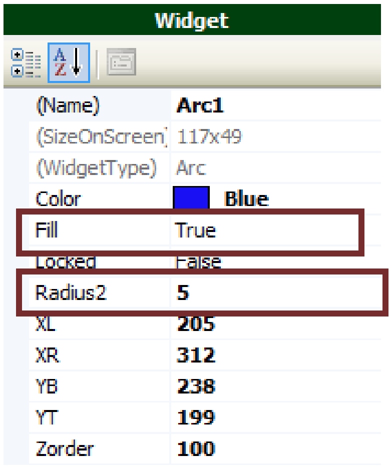


Figure 1.21
VGDD display with the arc primitive added

In the Widget properties box, adjust the arc properties so that a filled bevel will be shown on the display. **Change the Fill parameter to True, then adjust the radius to 5 (left click next to Radius2 and change the value to 5). Hit Enter when done to apply the changes.**



(Name)	Arc1
(SizeOnScreen)	117x49
(WidgetType)	Arc
Color	Blue
Fill	True
Locked	False
Radius2	5
XL	205
XR	312
YB	238
YT	199
Zorder	100

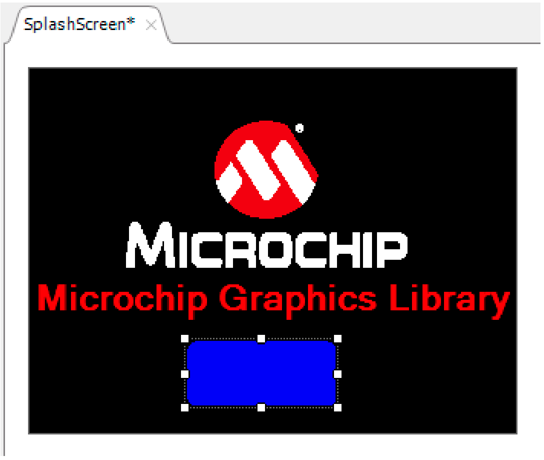
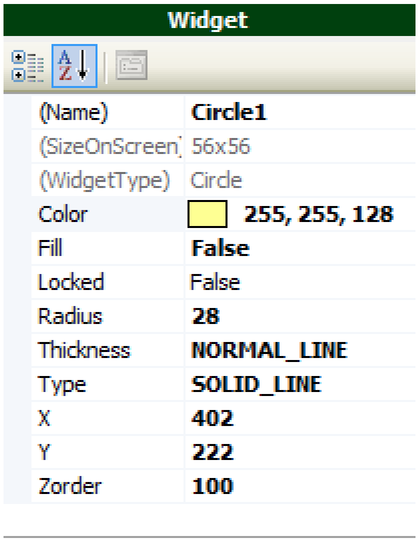


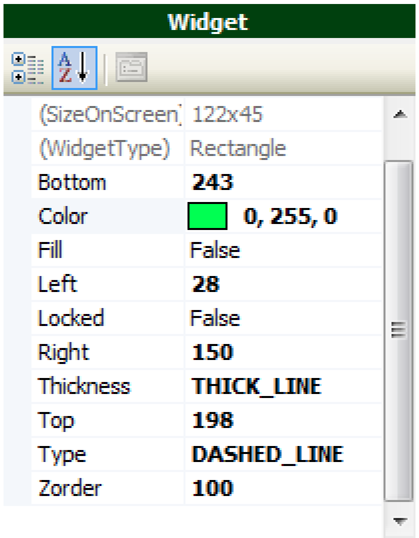
Figure 1.22
VGDD display with a filled bevel added

Using the GPL controls, add a rectangle, a circle to the screen. Using the Widget properties box, please adjust the parameters as shown below in **Figures 1.23 and 1.24.**



Widget	
(Name)	Circle1
(SizeOnScreen)	56x56
(WidgetType)	Circle
Color	🟨 255, 255, 128
Fill	False
Locked	False
Radius	28
Thickness	NORMAL_LINE
Type	SOLID_LINE
X	402
Y	222
Zorder	100

Figure 1.23
Properties for the circle primitive



Widget	
(SizeOnScreen)	122x45
(WidgetType)	Rectangle
Bottom	243
Color	🟩 0, 255, 0
Fill	False
Left	28
Locked	False
Right	150
Thickness	THICK_LINE
Top	198
Type	DASHED_LINE
Zorder	100

Figure 1.24
Properties for the rectangle primitive

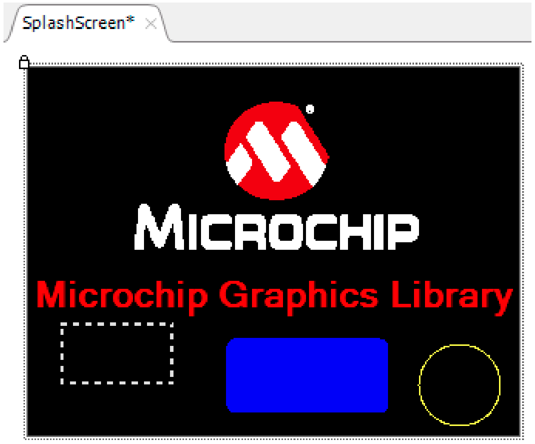


Figure 1.25
VDD display with rectangle and circle

Information

The Line, Circle and Rectangle primitives have common parameters to change the style of the line used to draw the primitive.

THICKNESS:
NORMAL_LINE (lines will be 1 pixel thick)
THICK_LINE (lines will be 3 pixels thick)

TYPE:
SOLID_LINE
DOTTED_LINE
DASHED_LINE

Tips and Tricks

Save time when drawing multiples of the same object by using copy

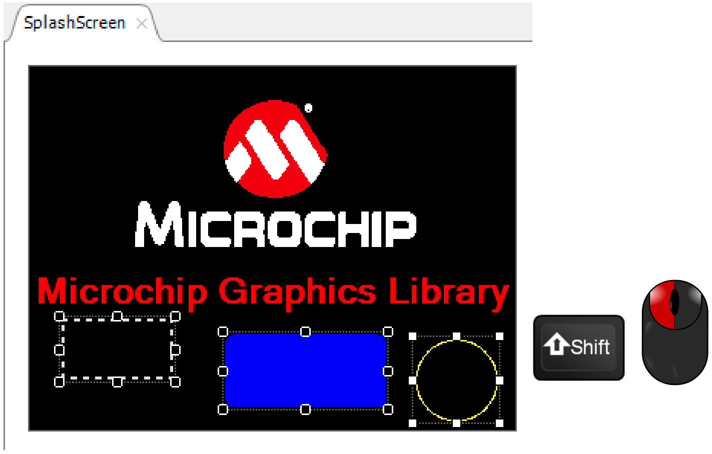
And paste

Icons on the toolbar.

You may also use the typical Windows shortcuts
CTRL+C to copy
CTRL+V to paste
CTRL+X to cut

⑭ Align the objects on the display.

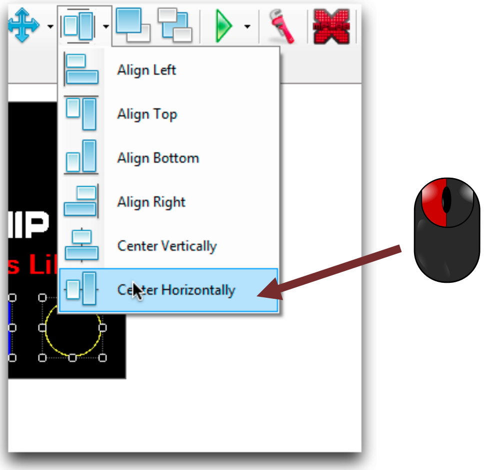
VGDD provides several editing functions to help us align objects on the display. **Select the rectangle, filled arc (bevel) and circle primitives by holding the shift key as you click on each primitive.**



Attention

Always select the widget you want to align to first, then select the other widgets.

With all the primitives selected, click the alignment icon to get a drop down menu of alignment options and choose Center Horizontally.



Tips and Tricks

Because there is not a center on screen alignment option, it is sometimes difficult to center the objects screen well. To get around this limitation, draw a rectangle the size of the screen and align the objects to that rectangle. When you are finished aligning, delete that rectangle.

Continue working with the alignment options until you are happy with the screen's appearance.

⑮ Save the project

It's a good practice to save your project prior to generating the code. **Click the Save Project icon in the toolbar.**

You will first be prompted for a project name. Save your project to
C:\RTC\HIF2132A\PIC24_Labs\P24_Lab1\VGDD Project\P24_Lab1.vdp
OR
C:\RTC\HIF2132A\PIC32_Labs\P32_Lab1\VGDD Project\P32_Lab1.vdp

Next, you will be prompted for a screen name. In projects with multiple screens, you will save each screen individually. Save the screen in the same directory as:
SplashScreen.vds

⑯ Generate C code.

Attention

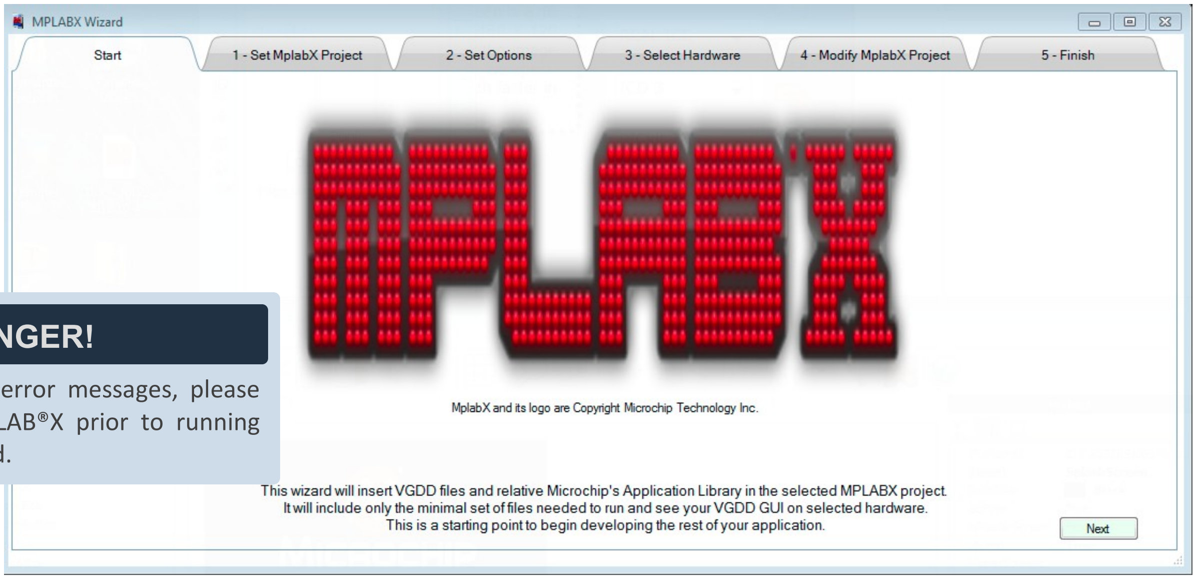
Although you have saved your project, you will be prompted to save the project again when you launch the MPLAB®X Wizard. This is by design to insure you have saved your work.

Now that we have a design for the splash screen, we are ready to generate the code we will use on our hardware. The first step in this process is to use the MPLABX wizard to set up the project. When you go through this process at home, you will need to create an MPLABX project first, close it, then run the Wizard. To save time during the labs, all of the MPLABX projects have been created for you.

Launch the MPLAB®X Wizard from the toolbar
Select OK when prompted to save your project.

DANGER!

To avoid error messages, please close MPLAB®X prior to running the wizard.



This wizard will insert VGDD files and relative Microchip's Application Library in the selected MPLABX project.
It will include only the minimal set of files needed to run and see your VGDD GUI on selected hardware.
This is a starting point to begin developing the rest of your application.

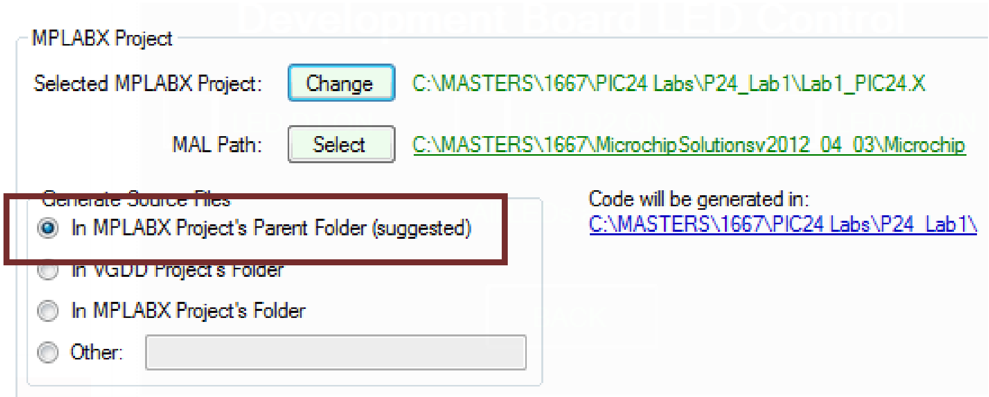
Figure 1.26
VGDD MPLAB®X Wizard

When the Wizard launches **click Next, then set up the project settings.**

If using a PIC32, use this project:
C:\RTC\HIF2132A\PIC32_Labs\P32_Lab1\Lab1_PIC32.X

In the Generate Source Files section, **check the option to generate code in MPLAB®X Project's Parent Folder as shown in Figure 1.26. Then click Next.**

Figure 1.27
Set MPLAB®X Project tab



Selected MPLABX Project: C:\MASTERS\1667\PIC24 Labs\P24_Lab1\Lab1_PIC24.X

MAL Path: C:\MASTERS\1667\MicrochipSolutionsv2012_04_03\Microchip

● In MPLABX Project's Parent Folder (suggested)
○ In VGDD Project's Folder
○ In MPLABX Project's Folder
○ Other:

Code will be generated in:
C:\MASTERS\1667\PIC24 Labs\P24_Lab1\

Since this is the first time generating code, we need to populate the MPLAB®X project and generate application specific files. You will not usually run the MPLAB®X Wizard again unless you specifically need to change the MPLAB®X project (e.g. changing the hardware).

Also in the Set Options tab, you will find options for the width of the parallel master port (PMP). This option sets the width of the data path between the PIC microcontroller and the LCD graphics controller. The Explorer 16 will only support an 8-bit data path to the LCD graphics controller board; therefore, if you will be using the Explorer 16 + PIC32 PIM hardware, you MUST check this box to avoid build errors in the generated code. Because LCC uses a virtual LCD controller, checking or not checking the PMP options will have no effect on the generated code.

Select the Parallel Port option appropriate for the hardware you are using.

After the setting are correct, **click Next.**

Attention

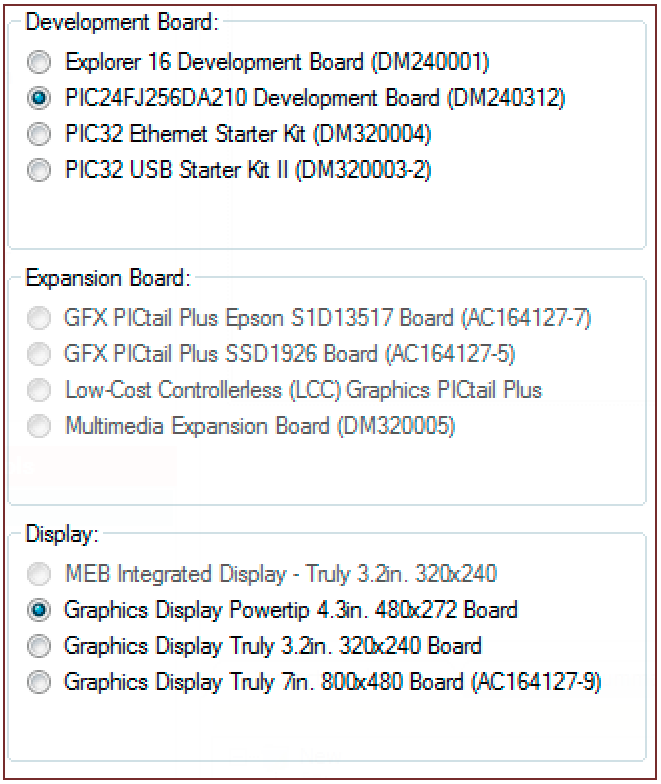
After running through the Wizard on a new project, selecting the option to **Generate Skeleton files will overwrite existing files**. If you wish to preserve existing application code in VGDDmain.c, VGDDmain.h, HardwareProfile.h and GraphicsConfig.h files, choose only the Modify Skeleton files option.

Information

When you are generating code for the hardware you have custom designed, do not generate the HardwareProfile.h file and skip this step.

In the Hardware tab, we will choose the development hardware we are using. These settings will tell VGDD which HardwareProfile.h and which graphics driver files we will need. When you are designing for your own hardware, deselect the option to generate the HardwareProfile.h file on the Set Options screen.

For these hands-on labs, we will select the development platforms. **If you are using PIC24, choose the options shown in Figure 1.30. If you are using PIC32, choose the options shown in Figure 1.31.**

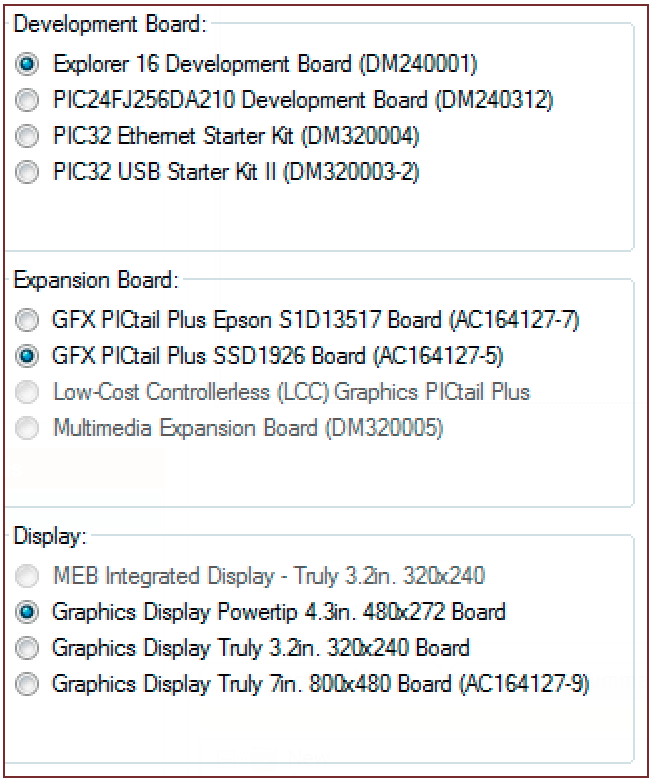


Development Board:
- ○ Explorer 16 Development Board (DM240001)
- ● PIC24FJ256DA210 Development Board (DM240312)
- ○ PIC32 Ethernet Starter Kit (DM320004)
- ○ PIC32 USB Starter Kit II (DM320003-2)

Expansion Board:
- ○ GFX PICtail Plus Epson S1D13517 Board (AC164127-7)
- ○ GFX PICtail Plus SSD1926 Board (AC164127-5)
- ○ Low-Cost Controllerless (LCC) Graphics PICtail Plus
- ○ Multimedia Expansion Board (DM320005)

Display:
- ○ MEB Integrated Display - Truly 3.2in. 320x240
- ● Graphics Display Powertip 4.3in. 480x272 Board
- ○ Graphics Display Truly 3.2in. 320x240 Board
- ○ Graphics Display Truly 7in. 800x480 Board (AC164127-9)

Figure 1.30
Hardware Options for PIC24



Development Board:
- ● Explorer 16 Development Board (DM240001)
- ○ PIC24FJ256DA210 Development Board (DM240312)
- ○ PIC32 Ethernet Starter Kit (DM320004)
- ○ PIC32 USB Starter Kit II (DM320003-2)

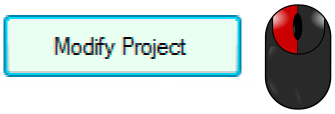
Expansion Board:
- ○ GFX PICtail Plus Epson S1D13517 Board (AC164127-7)
- ● GFX PICtail Plus SSD1926 Board (AC164127-5)
- ○ Low-Cost Controllerless (LCC) Graphics PICtail Plus
- ○ Multimedia Expansion Board (DM320005)

Display:
- ○ MEB Integrated Display - Truly 3.2in. 320x240
- ● Graphics Display Powertip 4.3in. 480x272 Board
- ○ Graphics Display Truly 3.2in. 320x240 Board
- ○ Graphics Display Truly 7in. 800x480 Board (AC164127-9)

Figure 1.31
Hardware Options for PIC32

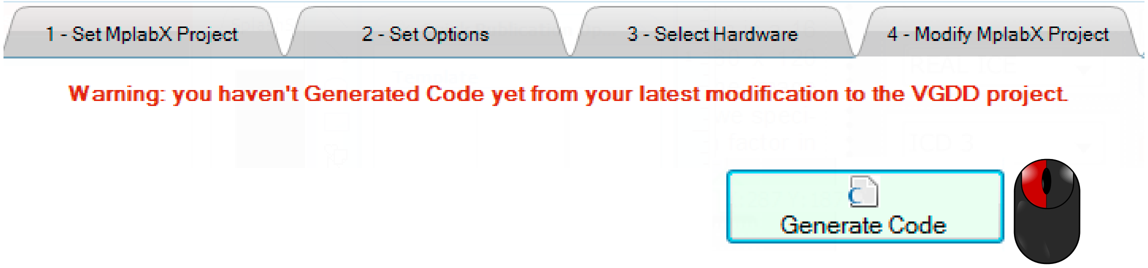
After the settings are correct, **click Next.**

In the Modify MPLABX Project tab, verify that all checks were OK, then **click the Modify button**.



A notification window will pop up when the modification is complete. Close that window by selecting OK, then click Next to continue.

In the Finish tab, you will see a warning similar to the one shown in Figure **1.31 on page 1-21** indicating that you need to generate code. **Click the Generate Code button**.

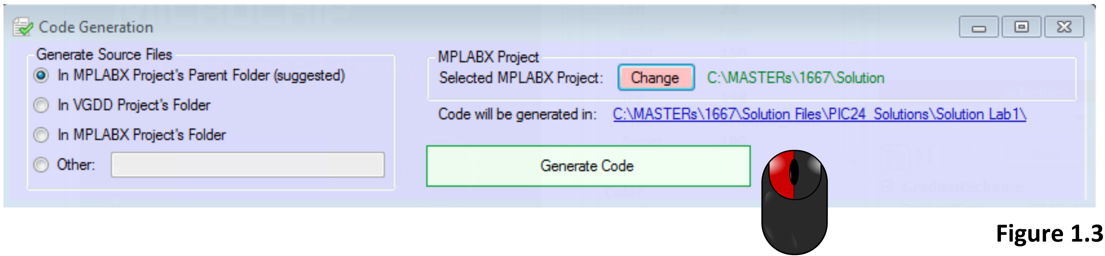


Warning: you haven't Generated Code yet from your latest modification to the VGDD project.

Figure 1.32
Finish tab

In the Code Generation pop-up window, verify the project settings are correct for the hardware you are using and that the generated source files are placed in the MPLABX Parent folder.

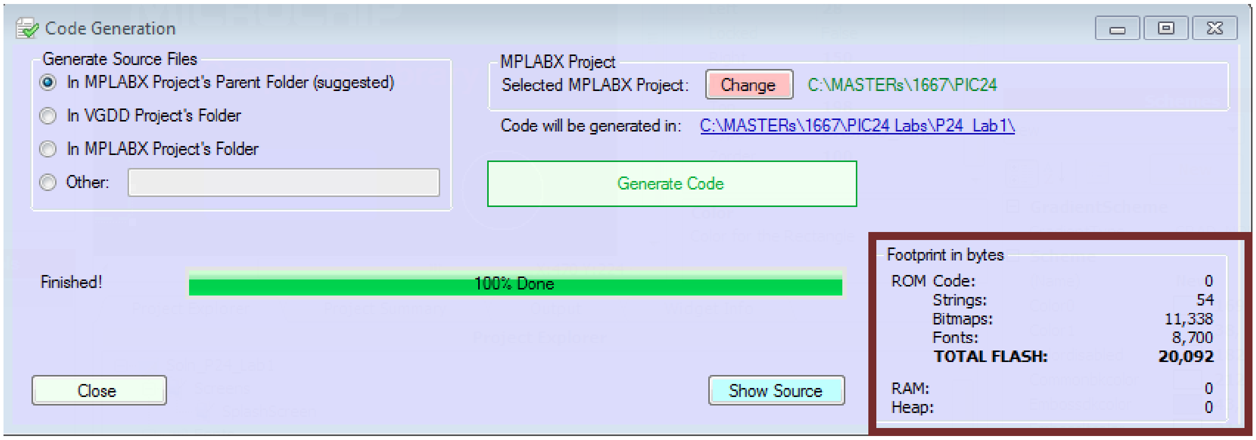
If using a PIC32, use this project:
C:\RTC\HIF2132A\PIC32 Labs\P32_Lab1\Lab1_PIC32.X



● In MPLABX Project's Parent Folder (suggested)
○ In VGDD Project's Folder
○ In MPLABX Project's Folder
○ Other:

MPLABX Project
Selected MPLABX Project: C:\MASTERs\1667\Solution
Code will be generated in: C:\MASTERs\1667\Solution Files\PIC24 Solutions\Solution Lab1\

Figure 1.33
Code Generation window



● In MPLABX Project's Parent Folder (suggested)
○ In VGDD Project's Folder
○ In MPLABX Project's Folder
○ Other:

MPLABX Project
Selected MPLABX Project: C:\MASTERs\1667\PIC24
Code will be generated in: C:\MASTERs\1667\PIC24 Labs\P24 Lab1\

Finished! 100% Done

Footprint in bytes

ROM Code:	0
Strings:	54
Bitmaps:	11,338
Fonts:	8,700
TOTAL FLASH:	**20,092**
RAM:	0
Heap:	0

Figure 1.34
Code Generation window

When the code generation is complete, the ***Footprint in bytes*** will contain information to help you estimate the amount of memory your screen design will need based on your compiler selection.

Of particular interest is the memory required for bitmaps and fonts. This number will help us determine if we need to use external memory for the application.

⑰ **Program the micro and view the results.**

Now that we have source code, let's program the boards and see how it looks on the actual display.

Launch MPLAB ® X by double clicking the icon on the desktop

MPLAB X IDE

Open the appropriate Lab1 project located at
C:\RTC\HIF2132A\PIC32 Labs\P32_Lab1\Lab1_PIC32.X
and select the correct configuration for the hardware you are using.

Using the information provided in the Hardware section at the beginning of this manual, verify you have the board properly connected, power applied and the programmer\debugger attached.

Build the code and program the device by clicking

Your display should resemble the picture in **Figure 1.35**

DANGER!

If you close VGDD, make sure you save the project and remember where you have saved it. We will use this project again in the bonus procedures section.

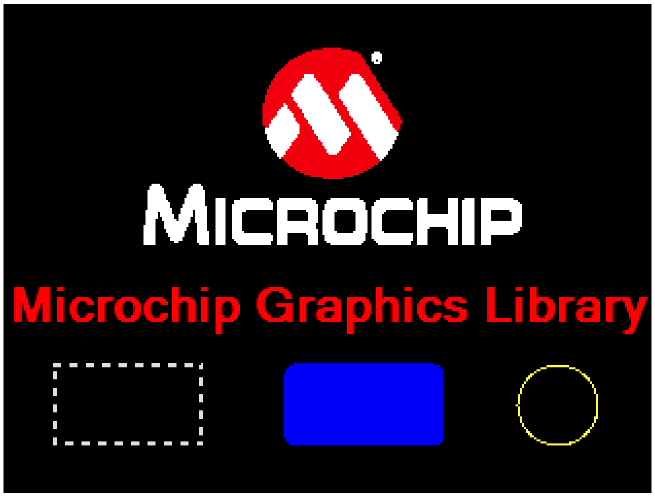


Figure 1.35
What you should see on your display

Congratulations!! You have completed Lab 1! If the allotted time has not expired, you may move on to complete the bonus procedure to explore the use of gradient backgrounds and screen transparency.

Bonus Procedure

If you have made it this far and there is still time remaining in the lab, let's examine how we might use the new gradient and transparency features to enhance our splash screen appearance. If you do not have time for this step, that's ok. The instructor will go over the solution with you or you may review the solution file at a later date.

1 Change the screen background to a gradient

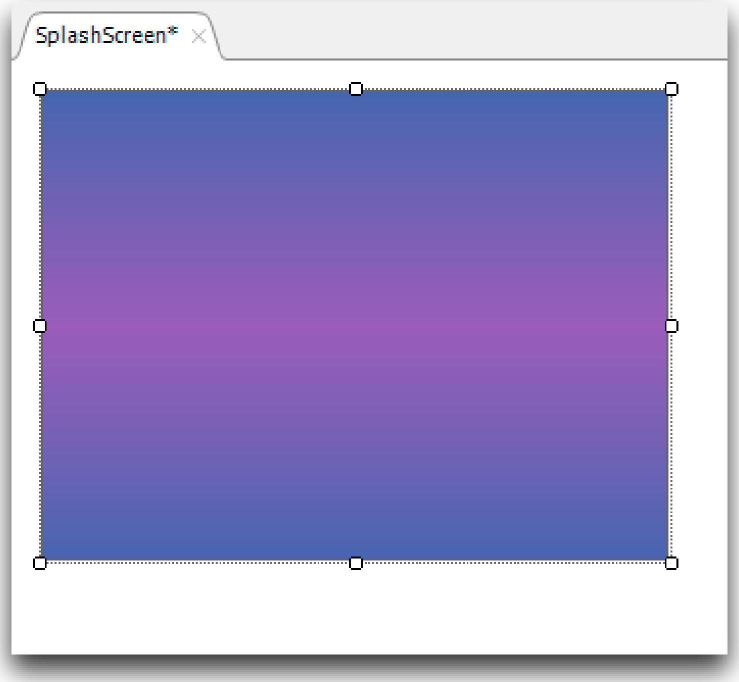
Microchip Graphics Library supports gradient backgrounds using either the **BarGradient(…)** or **BevelGradient(…)** functions. In VGDD, a **Gradient** (implemented as the **BevelGradient(…)**)is provided in the **GPL Controls** box.

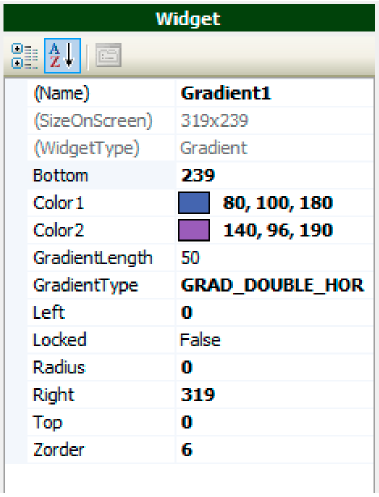
Return to the project you were working on in VGDD. **Click on the Gradient primitive in the GPL Controls box, then left click in the display to cover the entire screen.** Notice that the starting colors selected for the gradients are randomized. Will you find a color combination you hadn't considered before?

Tips and Tricks

Instead of dragging to size an object, you may enter the desired coordinates into the Widget parameters box. To fill the screen, the coordinates will be:

Top — 0
Left — 0
Bottom — max vertical-1
Right — max horizontal-1





Widget

(Name)	**Gradient1**
(SizeOnScreen)	319x239
(WidgetType)	Gradient
Bottom	**239**
Color1	🟦 **80, 100, 180**
Color2	🟪 **140, 96, 190**
GradientLength	50
GradientType	**GRAD_DOUBLE_HOR**
Left	**0**
Locked	False
Radius	**0**
Right	**319**
Top	**0**
Zorder	**6**

DON'T PANIC! Your work is not gone, it's just hidden below the large rectangle you just placed on the screen. Before we change the gradient Z-order, let's get its parameters situated. **In the Widget box, change the Radius parameter to 0 so the gradient will have sharp corners. Also adjust the coordinates if needed. Explore the options for Gradient Length, Gradient Type, Color 1 (Gradient Start Color) and Color 2 (Gradient End Color).**

When you are happy with the gradient appearance, send the gradient to the back (Z-order 0) by clicking the Send to Back icon in the toolbar.

Attention

When using thick, non-solid lines, the background will show through on the PC monitor. This is a limitation within Windows. When the code is programmed on the hardware, the rectangle will be correct.

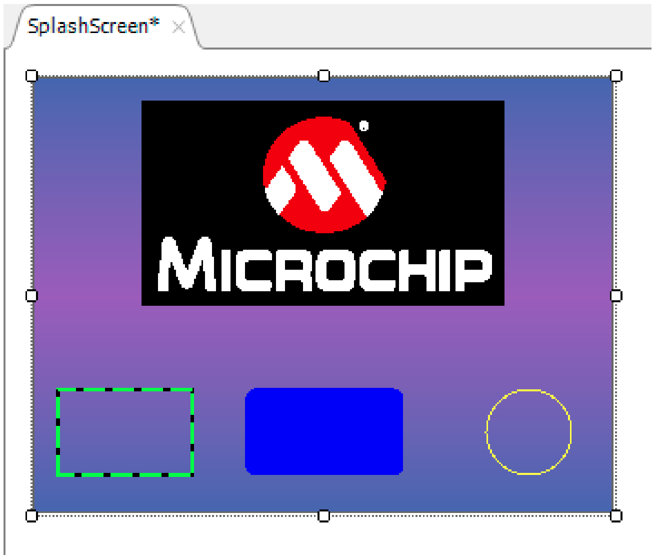
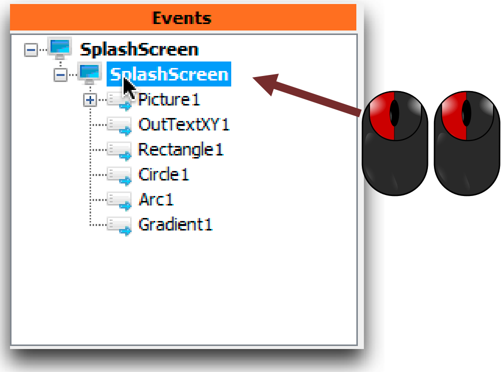


Figure 1.36
VGDD display before transparent color is set

❷ Set transparent color to hide the image background

Now we can definitely see the background color on the Microchip logo. To hide the background, we will use the transparency feature in Microchip Graphics Library. To implement this we will need to select the SplashScreen so we can modify the parameters. **In the events box (top far right side of VGDD), double click on the SplashScreen level.**

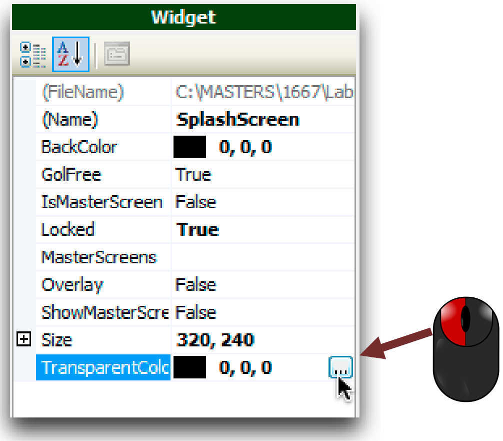


Information

When choosing a transparency color, remember that Microchip Graphics Library will ignore every pixel of that very specific color. Make sure that the transparency color is not used in the image you are trying to make "transparent".

The image used for this lab was modified so it's background color is solid black (RGB = 000). Pixels of any other color (even RGB = 001) will be drawn.

With the SplashScreen selected, **set the Transparency color in the Widget box.**



(FileName)	C:\MASTERS\1667\Lab
(Name)	SplashScreen
BackColor	■ 0, 0, 0
GolFree	True
IsMasterScreen	False
Locked	True
MasterScreens	
Overlay	False
ShowMasterScre	False
⊞ Size	320, 240
TransparentColc	■ 0, 0, 0

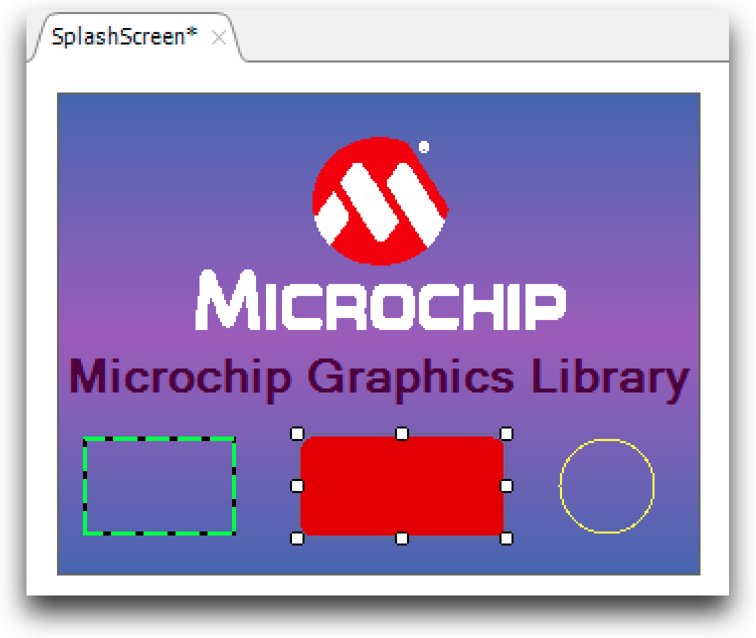


Figure 1.37
VGDD display after transparent color is set

③ **Generate code for the altered screen**

Click the Generate Code icon in the toolbar, verify the project settings, then click Generate Code in the pop up window.

④ **Launch MPLAB ® X by double clicking the icon on the desktop**

If it isn't already, open the Lab1.X project located at
..\RTC\HIF2132A\Labs\Lab1
and select the correct configuration for the hardware you are using.

Using the information provided in the Hardware section at the beginning of this manual, verify you have the board properly connected, power applied and the programmer\debugger attached.

Build the code and program the device by clicking

Your display should resemble the picture in Figure 1.37

Results

You have just learned how to use VGDD to generate Microchip Graphics Library code to generate a splash screen with images and fonts that may be easily integrated with your application. Also, you have learned that the GPL Controls generate Microchip Graphics Library primitive layer function.

Code Analysis

The simple application that you have generated implements a touch screen module and uses primitive rendering functions to draw a simple splash screen. Each primitive function is affected by the global drawing attributes such as line type, line size and color. For texts, the font and current color settings affect the way the text appears on the screen. All of this is included in the code generated by VGDD so that you may focus on application development.

Conclusions

Having learned how to use the Primitive Layer of the Graphics Library you are now ready to move upward to the next layer and learn the Graphic Object Layer that implements the Widgets. This gets you closer to fully integrating your application with a graphical solution. The next lab will give you a introduction to the Graphics Object Layer and use the Widgets that come with the library. Since the widgets in the Graphics Object layer call upon the primitive layer functions, you are now armed and ready to create your own widgets. An app note on this topic will be out soon.

Lab Exercise 2

Adding and Creating a Menu Screen

Purpose

In this lab, you will demonstrate your ability to use VGDD to create a menu screen for use in our application. You will use the style scheme box, create icons, and format widgets to meet our application needs for the HMI.

Solution files may be found at:

C:\RTC\HIF2132A\Solution Files

Tools

- MPLAB®X IDE v1.20 or higher
- Visual Graphics Display Designer (VGDD) v3.7 or higher
- PIC32 USB Starter kit II
- Truly 3.2" QVGA LCD Display
- Graphics LCD Controller PICtail ™ Plus LCC Board
- If using **PIC32**, you will need to use **Microchip® XC32 Compiler**

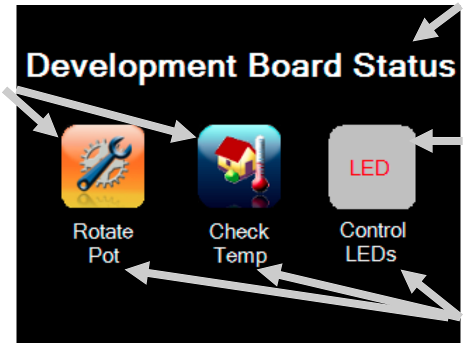
Objectives

1) Use VGDD to create a menu screen using 3 rounded buttons and 4 static texts.
2) Within VGDD, create and apply new style schemes.
3) Explore code generated by VGDD.

Expected Results

Attention

Your screen may look different than what is pictured.



Icons — Rounded Buttons with bitmaps on top

Static text without frame

Button with centered text

Static texts without frames

Figure 2.1
When the lab is fully completed (prior to completing the bonus steps), you should see this on your LCD panel.

Information

Hands-On Labs:
C:\RTC\HIF2132A\PIC32_Labs
Font Files:
All fonts used in this lab have been preinstalled on the lab computers
Image Files:
C:\RTC\HIF2132A\Images
Visual Graphics Display Designer:
C:\RTC\HIF2132A\VGDD.exe
Icon Files:
C:\RTC\HIF2132A\Icons

Attention

VGDD allows you to import screens used in other VGDD projects; however, the bitmaps and fonts used to create that screen are not stored with the screen. This means that after the screen is imported, you must add the bitmaps and fonts manually.

To import a screen choose **File –> Import –> Screen** from the top menu bar.

Procedure

In lab 2, we will add a screen to the project we worked on in Lab 1. To give us all a fresh start, we will work from a new project. If needed, please launch VGDD from:
C:\RTC\HIF2132A\VGDD.exe

If it VGDD is open from the previous lab, please close the VGDD project used in Lab 1 and open the Lab2_VGDD.vdp project found at :

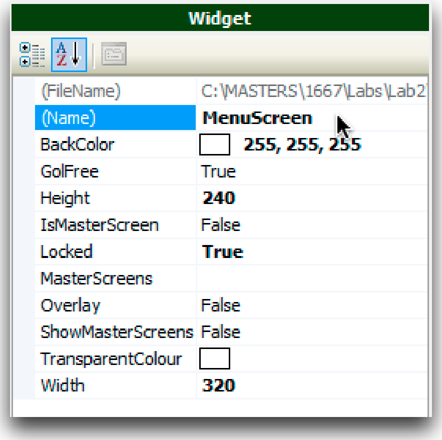
C:\RTC\HIF2132A\PIC32 Labs\P32_Lab2\VGDD Project

1 Add a new screen to the project.

To add a new screen to the project, **click the new screen icon on the toolbar**.

2 Change the Screen Name.

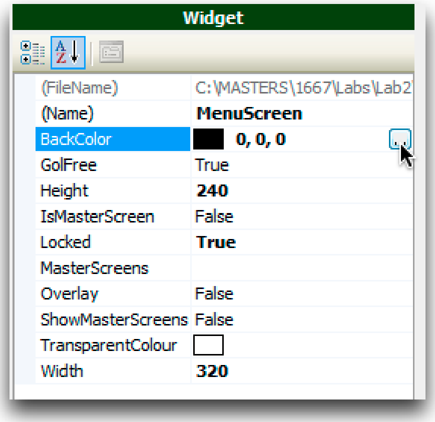
With **Screen2** selected, change the screen name in the Widget box by left clicking in the **(Name)** field and typing in **MenuScreen**.



Widget

(FileName)	C:\MASTERS\1667\Labs\Lab2
(Name)	MenuScreen
BackColor	☐ 255, 255, 255
GolFree	True
Height	240
IsMasterScreen	False
Locked	True
MasterScreens	
Overlay	False
ShowMasterScreens	False
TransparentColour	☐
Width	320

3 Change the screen background color to Black (0,0,0).

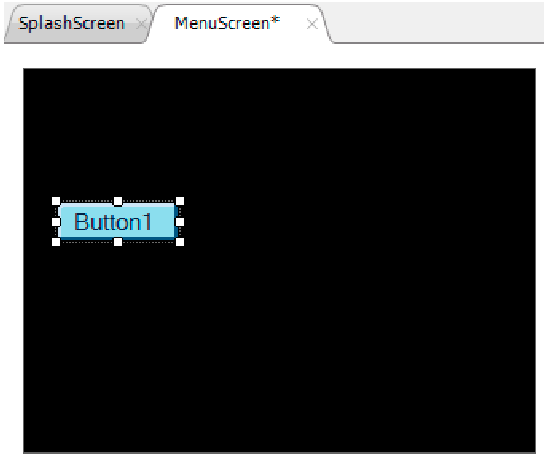


Widget

(FileName)	C:\MASTERS\1667\Labs\Lab2
(Name)	MenuScreen
BackColor	■ 0, 0, 0
GolFree	True
Height	240
IsMasterScreen	False
Locked	True
MasterScreens	
Overlay	False
ShowMasterScreens	False
TransparentColour	☐
Width	320

4 Add a button to the Menu Screen

Click **Button** in the GOL Widgets box and double click on the screen.



Notice that the button is not the shape we need for our design. **Resize the button** using the handles around the button or by changing the coordinate values in the coordinate fields of the Widget box :
top = 84, bottom = 155, right = 96, left = 21

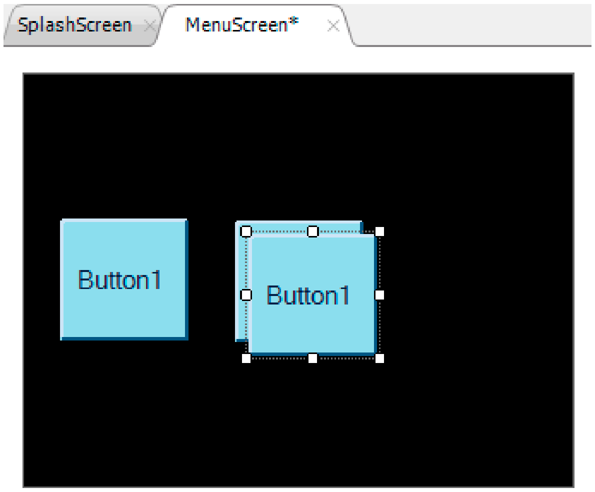
Attention

VGDD also supports Windows shortcuts for Cut, Copy, Paste and Delete.

5 Use copy and paste to add 2 more buttons to the screen

To copy an object, select the object to copy (in this case, Button1) and **click the copy icon in the toolbar**.

To paste the object, **click the paste icon in the toolbar. Do this twice to add 2 buttons.**



Information

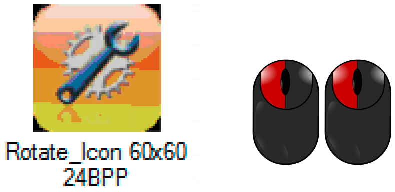
To cut an object, use the **Cut** icon

To delete an object, use the **Delete** icon

Drag the buttons into position, so that the 3 buttons are roughly centered on the screen.

⑥ Add bitmaps to the 2 buttons on the left

With the button on the far left selected, click in the **Bitmap** field in the Widget box. In the Bitmap Chooser, double click to choose the **Rotate_Icon** image.



Information

You may add as many bitmaps in the Bitmap chooser as you think you *may* need in your application. If the bitmap is not used, it will show up under the Type NOT_USED section. Bitmaps in this section are converted.

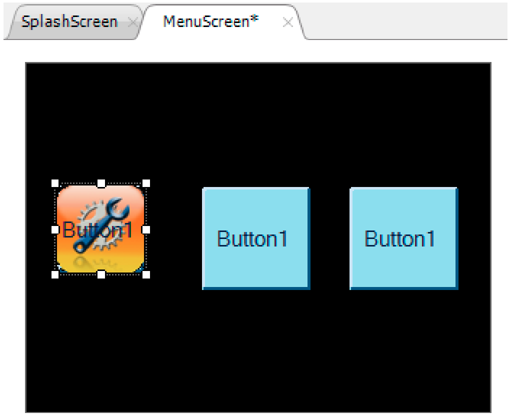
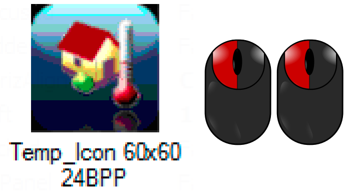


Figure 2.2
Screen after adding the bitmap to Button 1

Information

Notice that the button resizes to match the bitmap. If you need the button to be larger, you may resize it now. Because of the way bitmaps are drawn on buttons, you will not be able to size the button smaller than the bitmap. In a later step, we will show you how to hide the button panel so you should not need to size the button smaller than the bitmap.

Now, change the bitmap on the middle button. **Click on the Bitmap field in the Widget box. When the Bitmap chooser window pops open, double click the Temp_Icon file.**



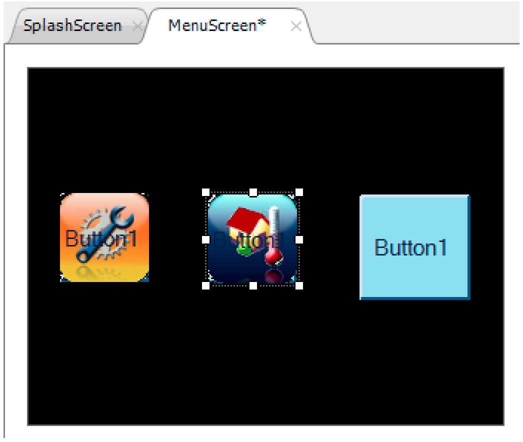
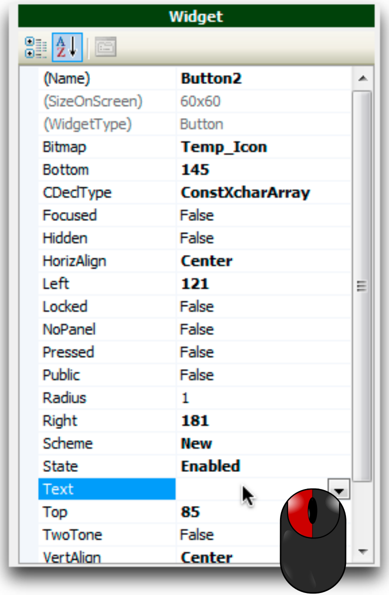


Figure 2.3
Screen after adding the bitmap to the middle button

Since we do not want text to be written atop our icons, we need to clear the text field for the two buttons to the left. **With the button selected, click in the Text field in the Widget box then delete the string.**



(Name)	**Button2**
(SizeOnScreen)	60x60
(WidgetType)	Button
Bitmap	**Temp_Icon**
Bottom	**145**
CDeclType	**ConstXcharArray**
Focused	False
Hidden	False
HorizAlign	**Center**
Left	**121**
Locked	False
NoPanel	False
Pressed	False
Public	False
Radius	1
Right	**181**
Scheme	**New**
State	**Enabled**
Text	
Top	**85**
TwoTone	False
VertAlign	**Center**

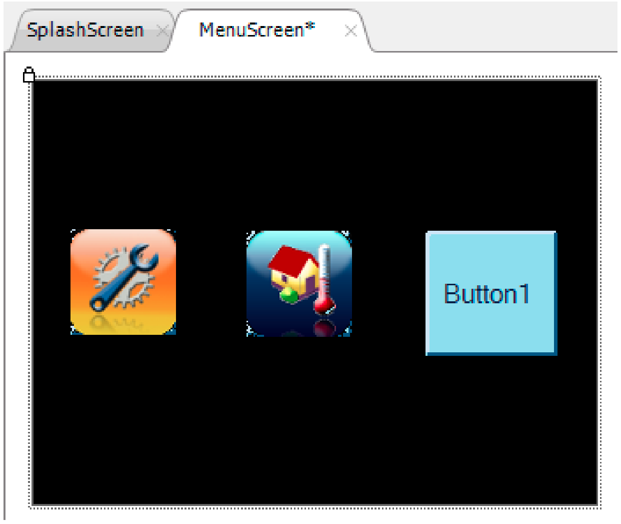
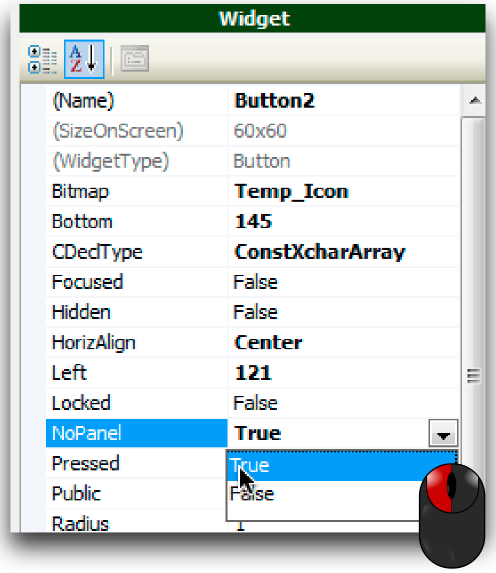


Figure 2.4
Screen after removing the text from the buttons.

Finally, notice that the corners around our buttons show the button face color. To save drawing time and to remove those unwanted colors, we will use the NO_PANEL option for the button. **With the button selected, click in the NoPanel field in the Widget box and choose True from the drop down menu.**



(Name)	**Button2**
(SizeOnScreen)	60x60
(WidgetType)	Button
Bitmap	**Temp_Icon**
Bottom	**145**
CDeclType	**ConstXcharArray**
Focused	False
Hidden	False
HorizAlign	**Center**
Left	**121**
Locked	False
NoPanel	**True**
Pressed	
Public	
Radius	1

Information

A more common use of the **BTN_NOPANEL** bit is to provide button behavior without showing the button. When using this feature, you can create a text with button properties, or if used with an image, create an icon with button properties.

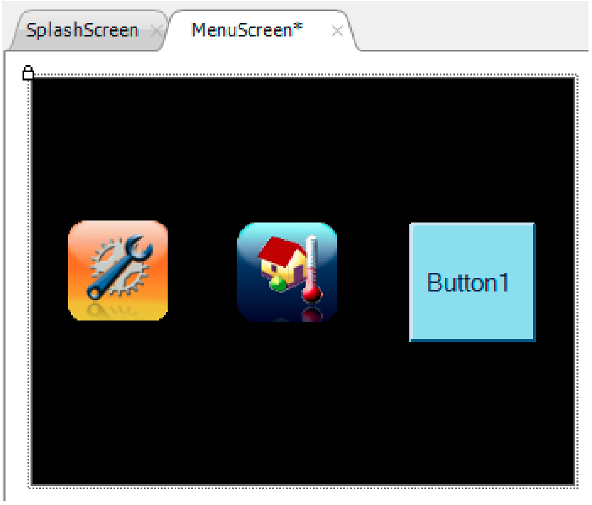


Figure 2.5
Screen after setting the NoPanel option to True

7 **Change the far right button properties to better match the two icons**

Now we need to change the far right button parameters so it better blends with our application. **With the far right button selected, click in the Radius field of the Widget box and change the value to 10**. This will round the corners to about the same curvature on the icons we used on the other 2 buttons.

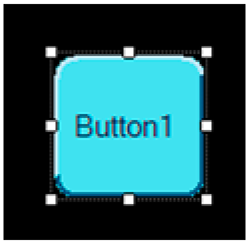


Figure 2.6
For right button with rounded corners

Using the handles on the button, resize it to match the two icons. The alignment guides may help you.

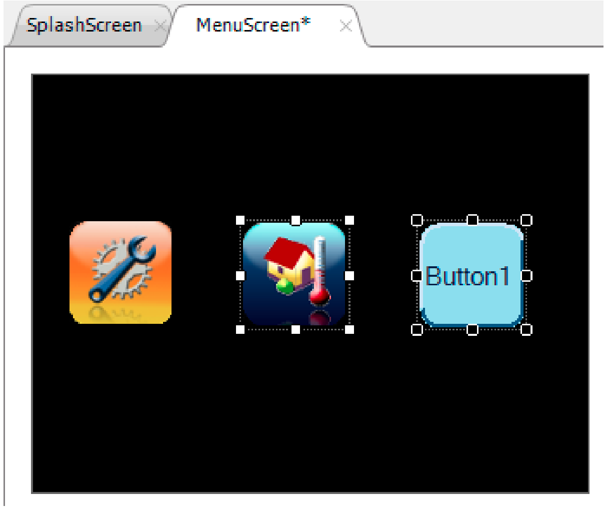


Figure 2.7
Screen after far right button parameters changed.

Tips and Tricks

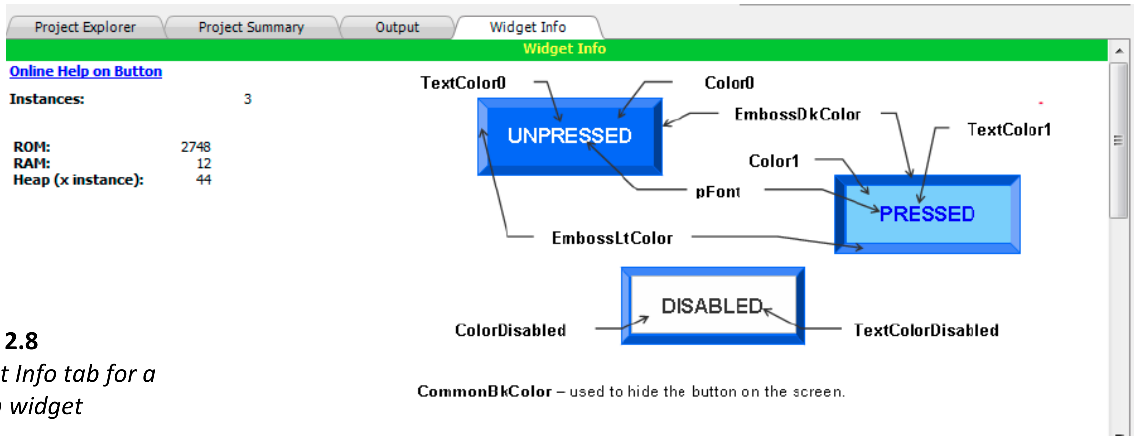
To resize an object to match another, select the object of the desired size (e.g. the middle button on the screen) hold the CTRL key, and then select the object you want to resize. On the menu tool bar, click the Size icon

And choose ***Make Same Size*** from the drop down menu.

Finally, we will change the button text to read **LED**. With the far right button selected, **click in the text field in the Widget box and change the string**.

8 Add a style scheme to give the far right button

To give the far right button a “flat” appearance (i.e. no 3D effect), we will need to create a new style scheme and modify certain parameters. With the far right button selected, click the Widget info tab in the Information box at the bottom of the VGDD window.



Instances: 3

ROM: 2748
RAM: 12
Heap (x instance): 44

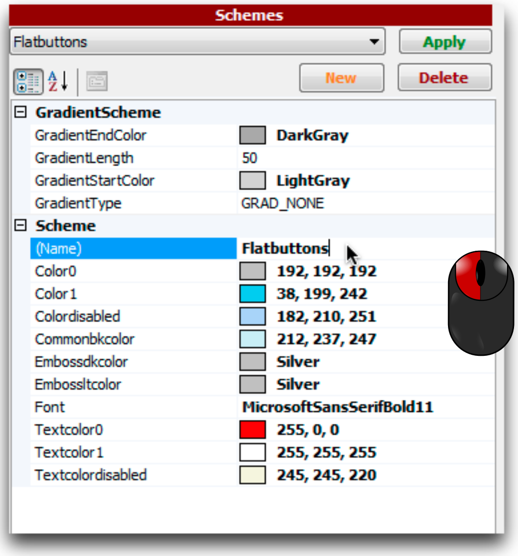
CommonBkColor – used to hide the button on the screen.

Figure 2.8
Widget Info tab for a button widget

On the right side of the Widget Info tab, we see information showing us how the style scheme parameters will affect the widget. Refer to this information as we work in the style scheme box.

To avoid impacting other screens in our project that may be using the default style scheme (named New), we will first create a new scheme. **In the Schemes box, click the New button** New

Notice that the name in the drop down menu changes to New1. **In the Name field of the Schemes box, change the scheme name from New1 to Flatbuttons.**

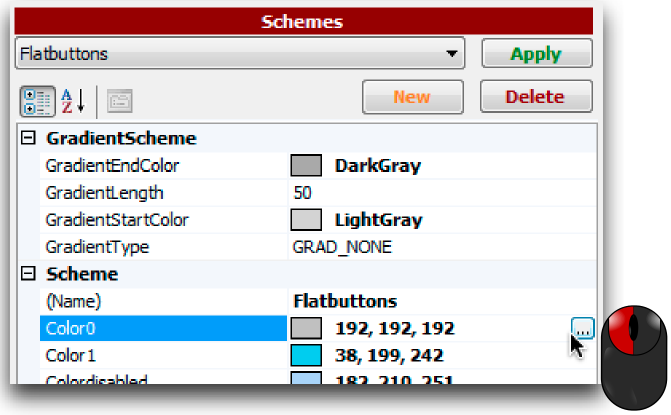


GradientScheme	
GradientEndColor	DarkGray
GradientLength	50
GradientStartColor	LightGray
GradientType	GRAD_NONE
Scheme	
(Name)	Flatbuttons
Color0	192, 192, 192
Color1	38, 199, 242
Colordisabled	182, 210, 251
Commonbkcolor	212, 237, 247
Embossdkcolor	Silver
Embossltcolor	Silver
Font	MicrosoftSansSerifBold11
Textcolor0	255, 0, 0
Textcolor1	255, 255, 255
Textcolordisabled	245, 245, 220

Information

The information to the left of in this tab lets us know how many instances of the widget are in use in the project, how much ROM, how RAM and how much Heap is needed to support the selected widget.

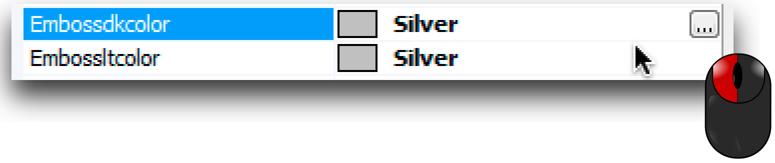
To match our application needs, we would like the face color to be gray. **In the style scheme box, change the Color0 parameter to a medium gray shade (192,192,192).** You may either click the ellipsis to bring up the color chooser window, or you may simply type the R,G,B value into the Color0 field.



Flatbuttons

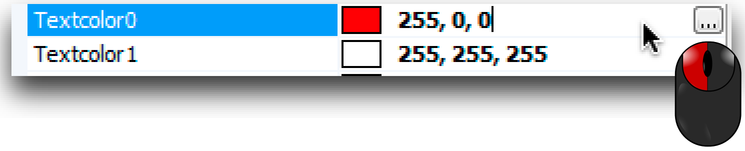
GradientScheme	
GradientEndColor	**DarkGray**
GradientLength	50
GradientStartColor	**LightGray**
GradientType	GRAD_NONE
Scheme	
(Name)	**Flatbuttons**
Color0	**192, 192, 192**
Color1	**38, 199, 242**
Colordisabled	**182, 210, 251**

To give the button a flat appearance, we now need to change the Embossdkcolor and Embossltcolor fields to match Color0. One way to do this is to **highlight the R,G,B value in the Color0 field and copy and paste it into the Embossdkcolor and Embossltcolor field using the Windows copy and paste shortcuts (Ctrl+C and Ctrl+V).** Or, you may use the color chooser window.



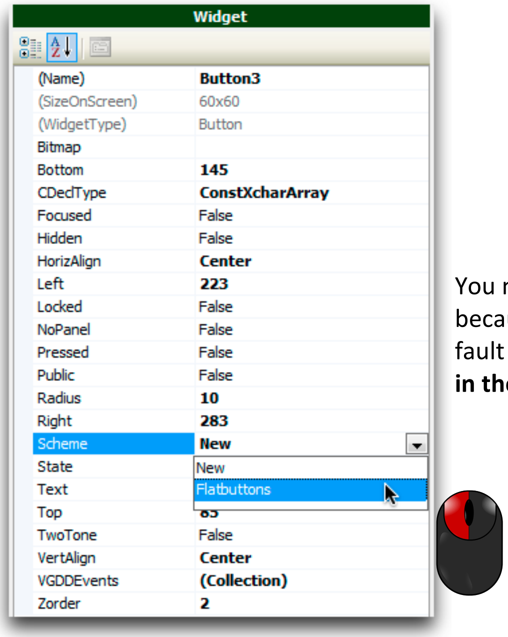
Embossdkcolor	**Silver**
Embossltcolor	**Silver**

Our application also requires that the **LED** text string appear Red. **Change the TextColor0 field to Red (255,0, 0).**



Textcolor0	**255, 0, 0**
Textcolor1	**255, 255, 255**

You may have noticed that you are not seeing the changes on the screen. This is because the button widget style scheme parameter is still pointing to the default style scheme. **With the far right button selected, click the Scheme field in the Widget box and choose Flatbuttons from the dropdown menu**



(Name)	**Button3**
(SizeOnScreen)	60x60
(WidgetType)	Button
Bitmap	
Bottom	**145**
CDeclType	**ConstXcharArray**
Focused	False
Hidden	False
HorizAlign	**Center**
Left	**223**
Locked	False
NoPanel	False
Pressed	False
Public	False
Radius	**10**
Right	**283**
Scheme	**New**
State	
Text	
Top	**85**
TwoTone	False
VertAlign	**Center**
VGDDEvents	**(Collection)**
Zorder	**2**

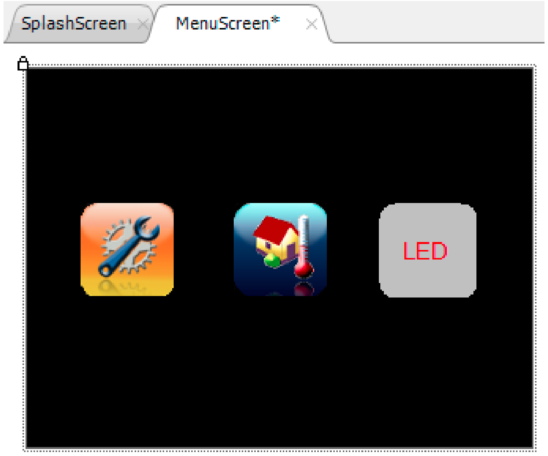


Figure 2.9
Screen with all buttons formatted and placed

9 **Add a static text box to label the menu screen.**

Now we need to let our users know what this menu screen is for. **Add a static text widget to the top center of the screen by selecting static text from the GOL Widgets box and double clicking in the screen area.**

With the static text selected, change the Text field in the Widget box to read **Development Board Status.** Using the handles, resize the static text so that all the letters in the string are visible.

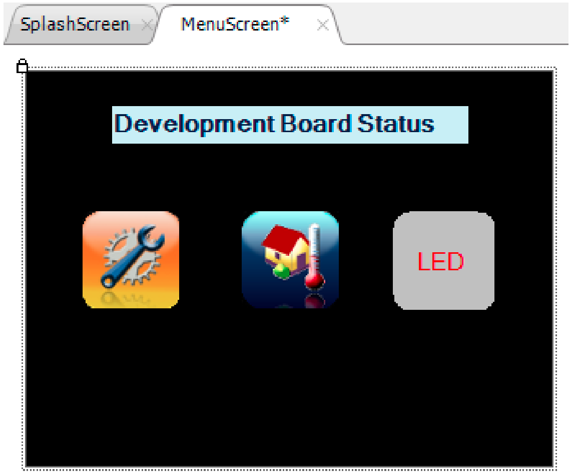
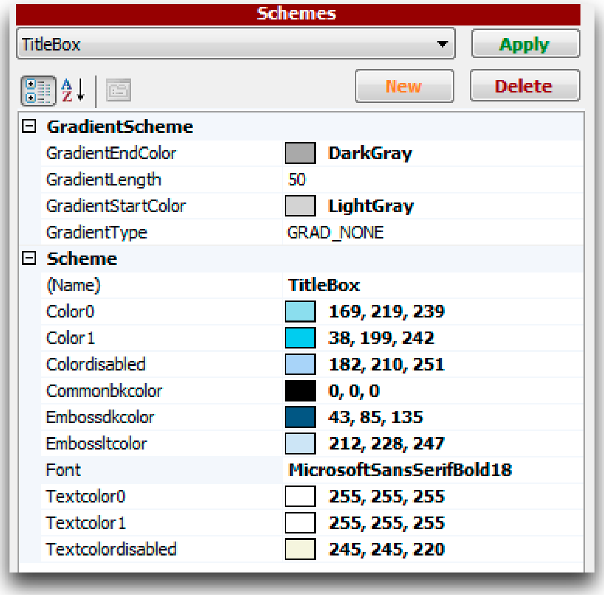


Figure 2.10
Screen with static text using default scheme added

Notice that the static text box uses the default scheme, but we would like the box to disappear and use white text in a larger font. To achieve this goal, we will need to add another style scheme. **Create a new scheme named TitleBox and change the parameters as shown in Figure 2.11**

Commonbkcolor = 0,0,0
Font = MicrosoftSansSerifBold18
Textcolor0 = 255,255,255



GradientScheme	
GradientEndColor	DarkGray
GradientLength	50
GradientStartColor	LightGray
GradientType	GRAD_NONE
Scheme	
(Name)	TitleBox
Color0	169, 219, 239
Color1	38, 199, 242
Colordisabled	182, 210, 251
Commonbkcolor	0, 0, 0
Embossdkcolor	43, 85, 135
Embossltcolor	212, 228, 247
Font	MicrosoftSansSerifBold18
Textcolor0	255, 255, 255
Textcolor1	255, 255, 255
Textcolordisabled	245, 245, 220

Figure 2.11
TitleBox scheme parameters

Attention

You may need to adjust the size and position of the title box to achieve this result.

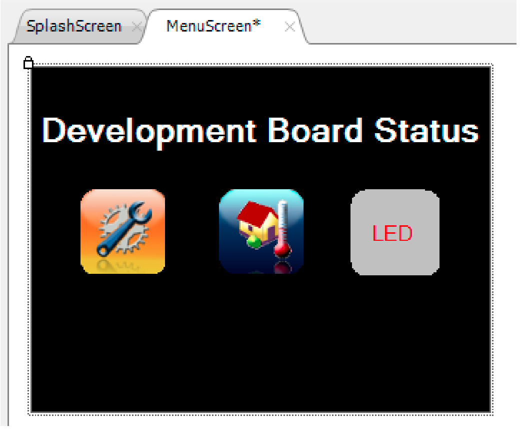


Figure 2.12
Screen with StaticText as the screen title

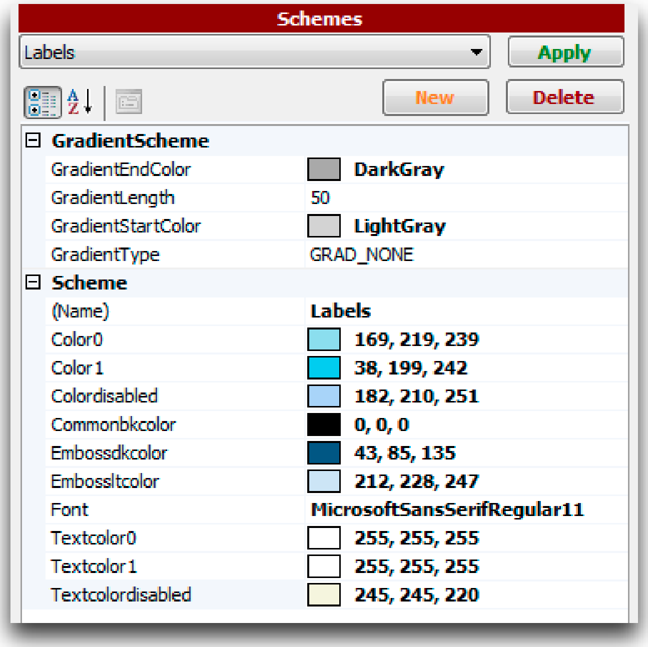
10 Add static text boxes to label the buttons

The last step in our menu screen is to add static text boxes to label the buttons. Because we need smaller font for the labels, we will need to create one more style scheme. **Create a new scheme named Labels with parameters as shown in Figure 2.13.**

References

Style Scheme Parameters

Name = Labels
Commonbkcolor = 0,0,0
Textcolor0 = 255,255,255



GradientScheme	
GradientEndColor	DarkGray
GradientLength	50
GradientStartColor	LightGray
GradientType	GRAD_NONE
Scheme	
(Name)	Labels
Color0	169, 219, 239
Color1	38, 199, 242
Colordisabled	182, 210, 251
Commonbkcolor	0, 0, 0
Embossdkcolor	43, 85, 135
Embossltcolor	212, 228, 247
Font	MicrosoftSansSerifRegular11
Textcolor0	255, 255, 255
Textcolor1	255, 255, 255
Textcolordisabled	245, 245, 220

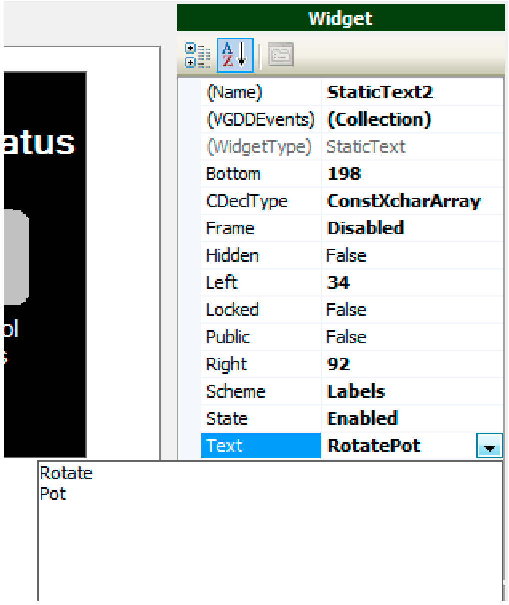
Figure 2.13
Parameters for the Labels Style Scheme

References

Static Text Parameter

Scheme = Labels
Frame = Disabled
TextAlign = Center
Text = Rotate
Pot

Add another static text box by selecting Static Text in the GOL Widgets box and double clicking in the screen area. With the static text selected, change its properties in the Widget box as shown in Figure 2.14 on page 2-11.



Widget	
(Name)	StaticText2
(VGDDEvents)	(Collection)
(WidgetType)	StaticText
Bottom	198
CDeclType	ConstXcharArray
Frame	Disabled
Hidden	False
Left	34
Locked	False
Public	False
Right	92
Scheme	Labels
State	Enabled
Text	RotatePot

Rotate
Pot

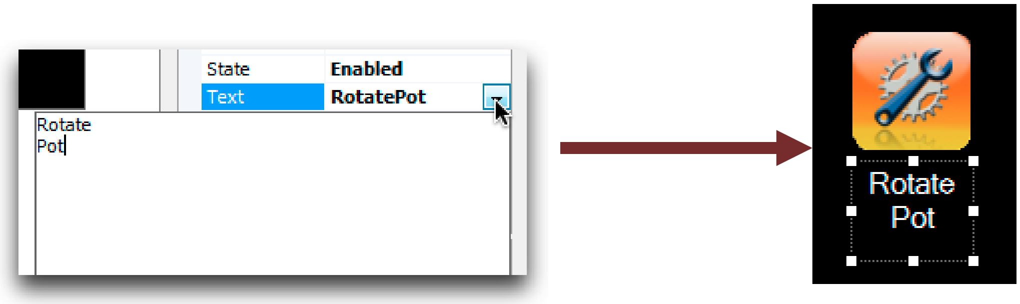
Figure 2.14
Static Text properties for the first button label

References

Static Text Parameter

Scheme = Labels
Frame = Disabled
TextAlign = Center
Text = Rotate
Pot

When adding the text string, notice the drop down box on the right side of the text field. You may type the text in this box using the Enter key to create a new line. Once all the text is added, click the down arrow again. You will see the text as a continuous string in the text field; however, the screen will show the text on multiple lines.



State	Enabled
Text	RotatePot

Rotate
Pot

When the static text and style scheme properties are correctly set, **use the copy and paste feature to add the next two labels. Position them to be centered beneath the appropriate buttons then change the text strings as shown in Figure 2.15.**

Information

You may expand the user touch space by using buttons with the NO_PANEL property set to true instead of using static texts to form the button labels.

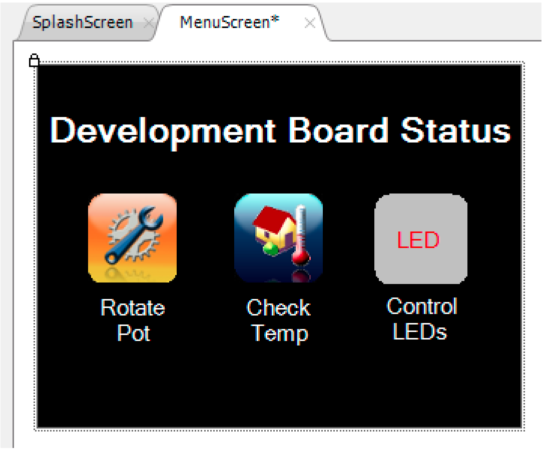


Figure 2.15
Correct text strings and positions for the button labels

⑪ **Generate the code.**

Using the MPLAB®X Wizard, generate the code and populate the MPLAB®X project for Lab 2.

If using a PIC32, use this project:
C:\RTC\HIF2132A\PIC24 Labs\P32_Lab2\P32_Lab2.X

DANGER!

To avoid error messages, please close MPLAB®X prior to running the wizard. Closing MPLAB®X is not necessary if you are only generating code.

⑫ **Program the micro and view the results.**

Now that we have source code, let's program the boards and see how it looks on the actual display.

Launch MPLAB ® X by double clicking the icon on the desktop

MPLAB X IDE

If it is still open, please close the previous MPLAB®X project (Lab1_PIC24.X or Lab1_PIC32.X). Open the appropriate Lab2 project located at :
C:\RTC\HIF2132A\PIC24 Labs\P32_Lab2\Lab2_PIC32.X
and select the correct configuration for the hardware you are using.

Using the information provided in the Hardware section at the beginning of this manual, verify you have the board properly connected, power applied and the programmer\debugger attached.

Build the code and program the device by clicking

Your display should resemble Figure 2.16. Touch anywhere on the display to move to the menu screen you just added.

Information

If you need a reminder for this step, please refer to Lab 1 steps 15 and 16 discussed on pages 1-17 thru 1-21 of this lab manual.

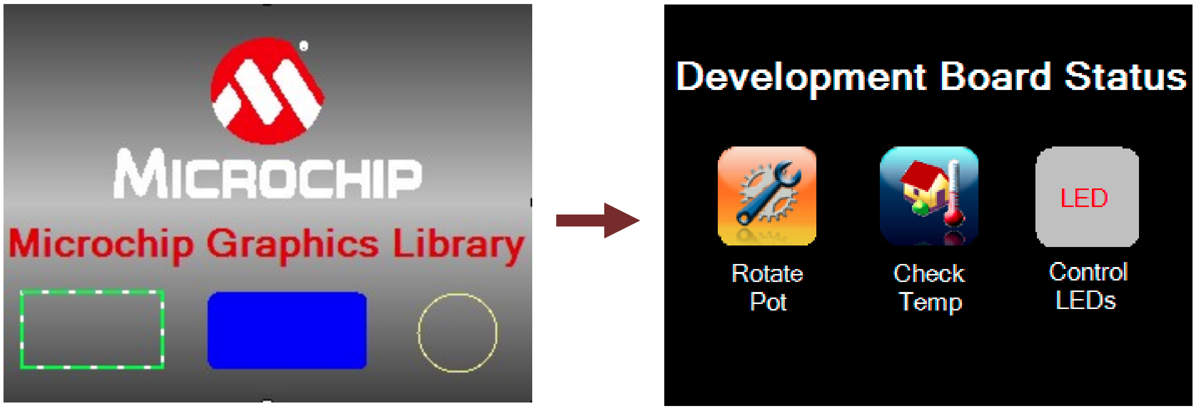


Figure 2.16
Lab 2 complete. Touch anywhere on the Splash Screen to move to the Menu Screen

Congratulations!! You have completed Lab 2!

Results

You have just learned how to use VGDD to add and style Microchip Graphics Library widgets. You have also learned that VGDD's Widget properties box makes it easy to adjust widget statebits to suit the initial appearance of the display. You have also learned how to modify the style schemes assigned to objects to change the appearance of the objects.

Code Analysis

The code that you generated created both the SplashScreen and MenuScreen screens for our application and used the GOLDraw() leave the management of rendering of objects to the library functions. Using the state bits you can modify how an object is rendered. Also with the style scheme you can easily modify the color values or font to change the style scheme assigned to the object.

Conclusions

Having learned how to create the screens for our HMI application, it is time to move forward and learn how to accept user inputs.

THIS PAGE INTENTIONALLY LEFT BLANK

Lab Exercise 3

Interfacing the User

Purpose

In this lab, you will demonstrate your ability to read the touchscreen interface and provide system and widget control in response to a user input (e.g. user touching the touchscreen). Using VGDD, you will create a simple user interface screen that implements widgets to control the LEDs on the USB Starter Kit II on the LCC Development Board.

Solution files may be found at:
C:\RTC\HIF2132A\Solution Files

Tools

- MPLAB®X IDE v1.20 or higher
- Visual Graphics Display Designer (VGDD) v3.7 or higher
- 3.2” QVGA LCD Display
- Graphics LCD Controller PICtail ™ Plus LCC Board
- PIC32 USB Starter Kit II
- If using **PIC32**, you will need to use **Microchip® XC32 Compiler**

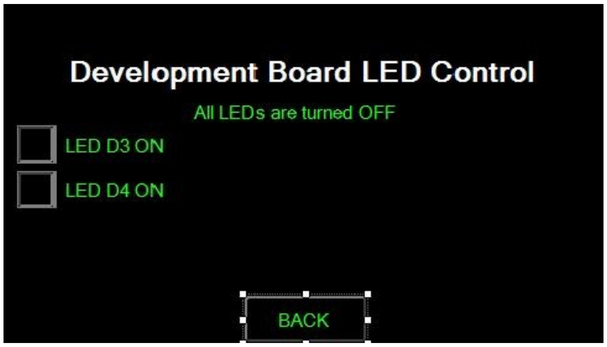
Objectives

1) Provide customized control for both the system and widgets via the GOLMsgCallback() functions.
2) Use VGDD to generate user event code.
3) Add code to the MPLABX project to control LEDs.

Expected Results

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

- See the images for Rotate Potentiometer and Check Temperature on the menu screen change with a user press.
- Press the LED button to move from the menu screen to an LED Control screen.
- Use checkbox options in the LED control screen to change LEDs on the board.



Information

Hands-On Labs:
C:\RTC\HIF2132A\PIC32_Labs
Font Files:
All fonts used in this lab have been preinstalled on the lab computers
Image Files:
C:\RTC\HIF2132A\Images
Visual Graphics Display Designer:
C:\RTC\HIF2132A\VGDD.exe
Icon Files:
C:\RTC\HIF2132A\Icons

Information

The touchscreen driver functions are application dependent. Examples have been provided for your use in the Board Support Package folder in the Microchip Applications Library installation path. Note that all the graphics demos make use of these drivers.

Information

For simplicity, we have chosen to use the ADC module with software to control the touchscreen interface. As such, the display may need to be recalibrated more frequently. To achieve more accurate touch performance that does not require frequent recalibration, please use the AR1021, AR1011, or AR1100 touch control chips from Microchip Technology. A driver for the AR1021 is provided in the Board Support Package directory.

Procedure

In lab 3, we will be using VGDD to create and test the LED Control screen for our application HMI.

To implement the user interface, we will work exclusively with the 4-wire resistive touch overlay that is provided on the display boards. The touch screen driver functions (**TouchScreen.c, TouchScreen.h, TouchScreenResistive.c, and TouchScreenResistive.h**) implement the sampling of the user touch. Sampling is done by the function **TouchProcessTouch().** This function sets up the 2 analog signals and 2 digital signals on the touch screen and uses the A/D module to obtain the actual samples. In our application, a timer interrupt calls the **TouchProcessTouch()** to update the touch positions variable in the touch screen driver.

The application will call the **TouchGetMsg()** function (located in the touchscreen driver files) to check if there is a valid touch. **TouchGetMsg()** populates the message structure **GOL_MSG**. To have the library process that message, the application will call the **GOLMsg()** function. The library evaluates the message and parses the active linked list to determine which widget, if any, was affected by the users touch.

If further processing is needed for the application to respond to the touch, the application will provide code in the **GOLMsgCallback()** function. This function is called by **GOLMsg()** and implemented in the application. Using the passed parameters, the application will be able to interpret the message. Based on that interpretation, the application can then modify widgets, update system variables, and in multiple screen applications, change the screen states.

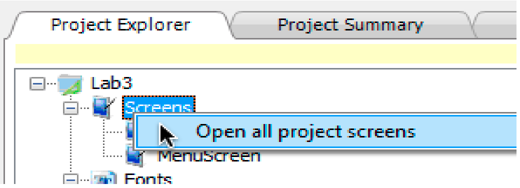
Fortunately, VGDD generates all of this code for us so that we will be able to focus on the application needs!

① Launch Visual Graphics Display Designer

If it is not already open, please launch VGDD
C:\RTC\HIF2132A\VGDD.exe
OR
Double click the VGDD icon on your desktop

And open the VGDD project located at:
C:\RTC\HIF2132A\PIC32 Labs\P32_Lab3\VGDD Project

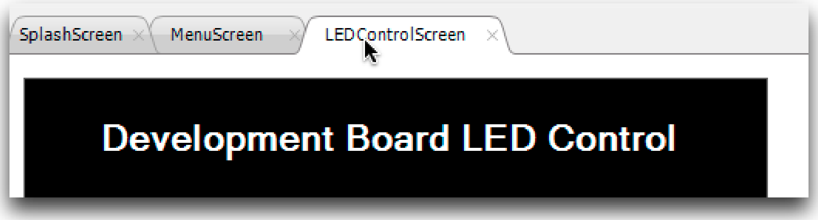
Notice that when a project with multiple screens is opened, only the first screen is shown in VGDD. You may open the other screens by double clicking the screen name in the Project Explorer window at the bottom of the screen. Alternatively, you may open all screens in the project by right clicking Screens and selecting "Open all project screens".



② Open the Event Editor for a Checkbox user event

For this lab, we are using a project that employs the screens we designed in the first two labs. To save time, the LED Control Screen has been added and designed for you.

Activate the LEDControlScreen by clicking the tab in the VGDD display box.

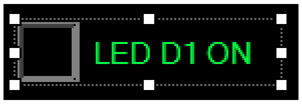


Information

If you'd like to close a screen, click the x next to the screen name in the tabs at the top of the VGDD display field.

To remove a screen from the project, right click the screen name in the Project Explorer window and select "Remove Screen from Project".

Select the top left checkbox. For the PIC32 project, this will be the checkbox we will use to control LED D3. For the PIC24 project, this will be the checkbox we will use to control LED D1.



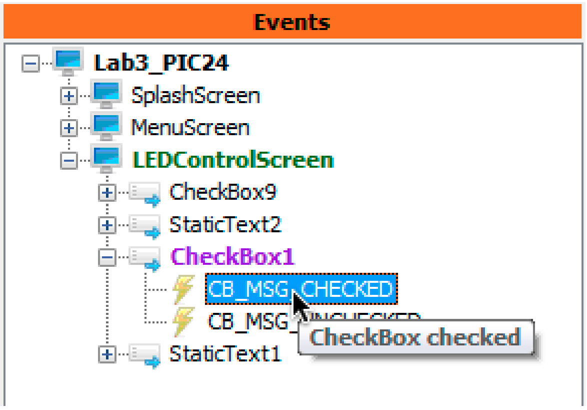


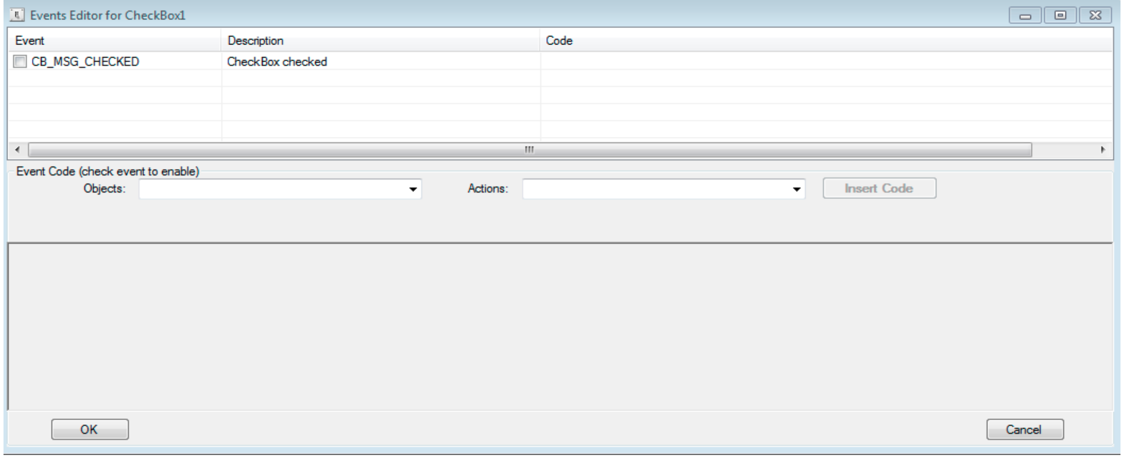
Figure 3.2
Events Box

Information

A user event is an event triggered by a user's action. For these hands-on labs, the events are triggered by touching the screen.

In the Events box (on the far right of the VGDD screen), notice that the Checkbox is highlighted an available user events are listed. Notice that when you move your mouse over an event, a brief description is displayed as shown in Figure 3.2.

Double click to select the CB_MSG_CHECKED event to open the Events editor window for that user event.


Events Editor for CheckBox1
Event Code (check event to enable)
Objects:
Actions:
Insert Code
OK
Cancel

Event	Description	Code
☐ CB_MSG_CHECKED	CheckBox checked	

Figure 3.3
Events Editor

③ Add code to handle the Check Box events

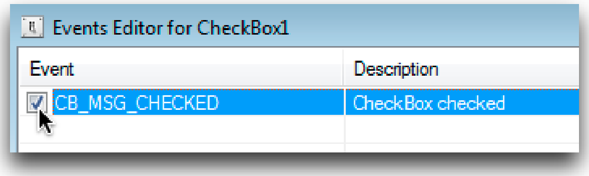
References

Although VGDD will generate much of the code for us, it is necessary to understand the available states, as well as the message defaults for the widgets. Please refer to the Graphics Library Help file for more details on the widget states.

..\..\1667\Microchip\Help

The check box widget may accept 2 messages: **CB_MSG_CHECKED** and **CB_MSG_UNCHECKED**. The Microchip Graphics Library message interface will automatically set (or clear) the **CB_CHECKED** state bit and will set **CB_DRAW_CHECK** state bit which will update the screen.

Check the box next to the CB_MSG_CHECKED event in the Events column of the window so that the event code we create will be included in the code.



Event	Description
☑ CB_MSG_CHECKED	CheckBox checked

Our application requires that we perform 3 actions when the user touches the checkbox:

1 — The checkbox must be checked or unchecked (the library will handle this by default) .
2 — The checkbox string must be changed to read "LED Dx OFF" when the check box is unchecked or "LED Dx ON" when the check box is checked.
3 — The status shown in the static text at the bottom of the screen must be changed to indicate that an LED is turned on.

Since we know that the Graphics Library messaging interface, **GOLMsg()**, will set the bits required for the checkbox to look checked, we do not need to add this code. Changing the string used to label the checkbox and changing the static text string are not handled by the messaging interface; therefore, we need to provide code to handle those tasks.

Information

The object selected in this box, is the object that we wish to be affected by the message received. For this portion of the lab, we will add code to enhance the action for (affect) **CheckBox1**.

In the Objects drop down menu (middle left of the Event Editor window), **choose the CheckBox1 widget.** It should have the words "This Widget" to the right. If it does not, you have the wrong widget selected. **Refer to Figure 3.4.**

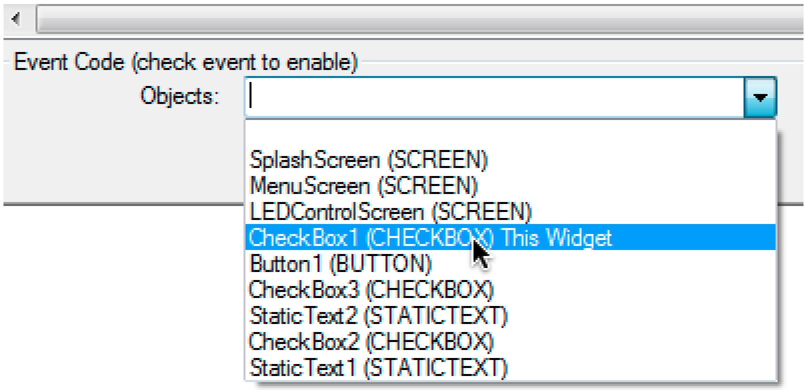


Figure 3.4
Correct object drop down menu options

Now, in the Actions drop down menu (located in the middle right of the Event Editor screen), **choose the Set CheckBox Text option.** Refer to **Figure 3.5**.

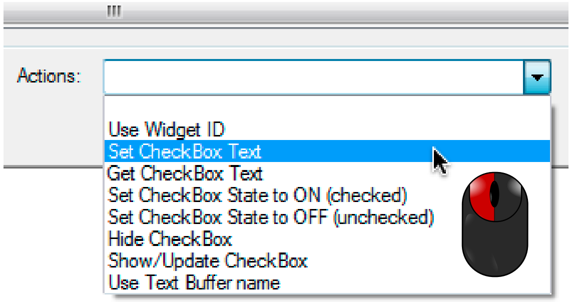
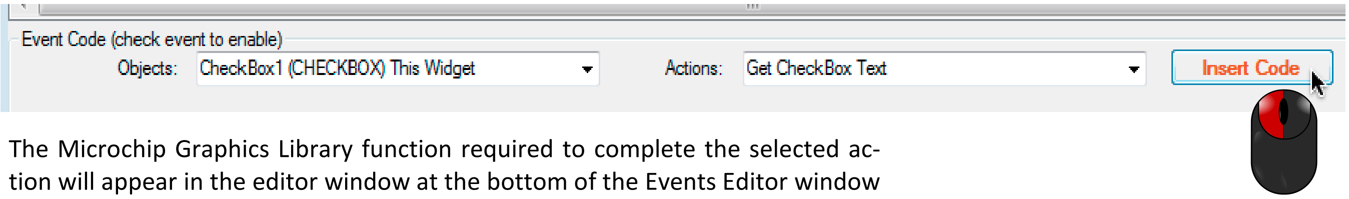


Figure 3.5
Available actions for the selected object

Information

The Actions shown represent the available GOL Widget APIs for the selected widget. Further description for these actions may be found in the graphics library help file.

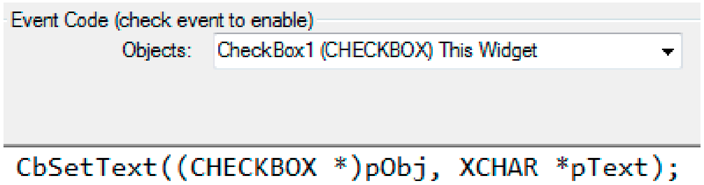
Then click the button to the right to Insert Code.



Objects: CheckBox1 (CHECKBOX) This Widget

Actions: Get CheckBox Text

The Microchip Graphics Library function required to complete the selected action will appear in the editor window at the bottom of the Events Editor window as shown in **Figure 3.6**



Objects: CheckBox1 (CHECKBOX) This Widget

```
CbSetText((CHECKBOX *)pObj, XCHAR *pText);
```

Figure 3.6
Inserted code

Attention

Code is inserted at the last location of the editor cursor. Be sure to press Enter after inserting code.

We now need to provide the new string for the Checkbox widget. Since CheckBox1 refers to LED D1 (for the PIC24) or LED D3 (for the PIC32), **change the inserted code to:**

PIC24 Users:
```
CbSetText((CHECKBOX *)pObj, (XCHAR *) “LED D1 OFF”);
```

PIC32 Users:
```
CbSetText((CHECKBOX *)pObj, (XCHAR *) “LED D3 OFF”);
```

DANGER!

If you do not change the code to provide the string, you will see build errors when you try to compile. The *pText pointer is a place holder for the new string.

As mentioned earlier, the GOL Message handler will set (or clear) the **CB_DRAW_CHECK** bit by default. Setting this bit will update the check mark, but will not update the Checkbox string. To draw the new string, we will need to add an action to “Show/Update” the widget. **In the Action drop down menu, select the Show/Update CheckBox option, then click Insert Code.**

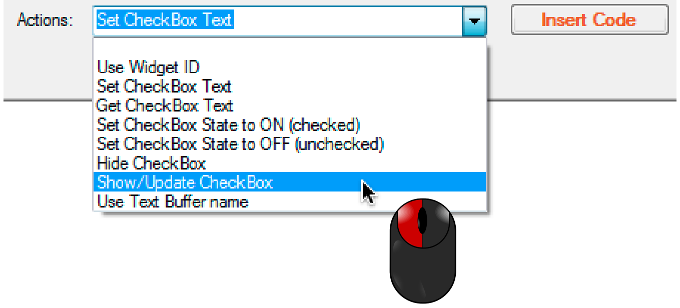
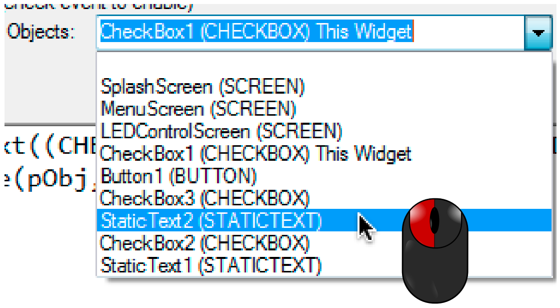
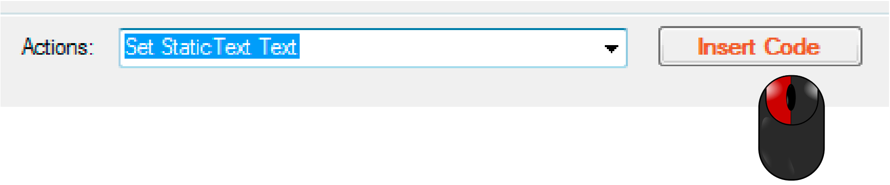


Figure 3.7
Available actions for the selected object

Using the VGDD Event Editor, we may also add code that will affect another widget on the screen when an object receives a GOL Message. For this lab, we would like to change the status shown in StaticText2 to indicate if an LED has been turned on or turned off. **In the Objects drop down menu, select the StaticText2 object.**



In the Actions drop down menu, select the Set StaticText Text option and click Insert Code.



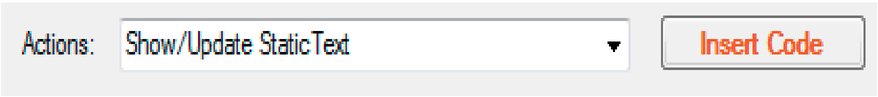
Information

When referencing a widget that is not the widget receiving the message, VGDD will use the API to find a pointer to the widget selected to receive an Action.

As with the code generated to set the CheckBox string, we will need to modify the inserted code to provide the string we'd like displayed. **Change the inserted code to replace the string pointer reference (...XCHAR *pText) ; to**

```
...(XCHAR *) “An LED has been turned ON”);
```

Recall, the GOL Message handler will not perform default actions for widgets that did not receive the message; therefore, to draw the new string, we will need to add an action to “Show/Update” the widget. **In the Action drop down menu, select the Show/Update StaticText option, then click Insert Code.**



Information

The LEDs will be turned on and off using the application provided function:

```
SetLED(status, whichOne)
```

where the status (**LED_ON** or **LED_OFF**) will turn on or off the status specified by **whichOne.** The parameter **whichOne** is the number following the LED we need to turn on or off. For example, to turn on LED D1, we will use

```
SetLED(LED_ON, 1);
```

Finally, we need to add an application specific function call to update the LEDs. For this purpose, we will use the SetLED() function explained in the information box to the left on this page. **In the Event Editor window, add the following code**:

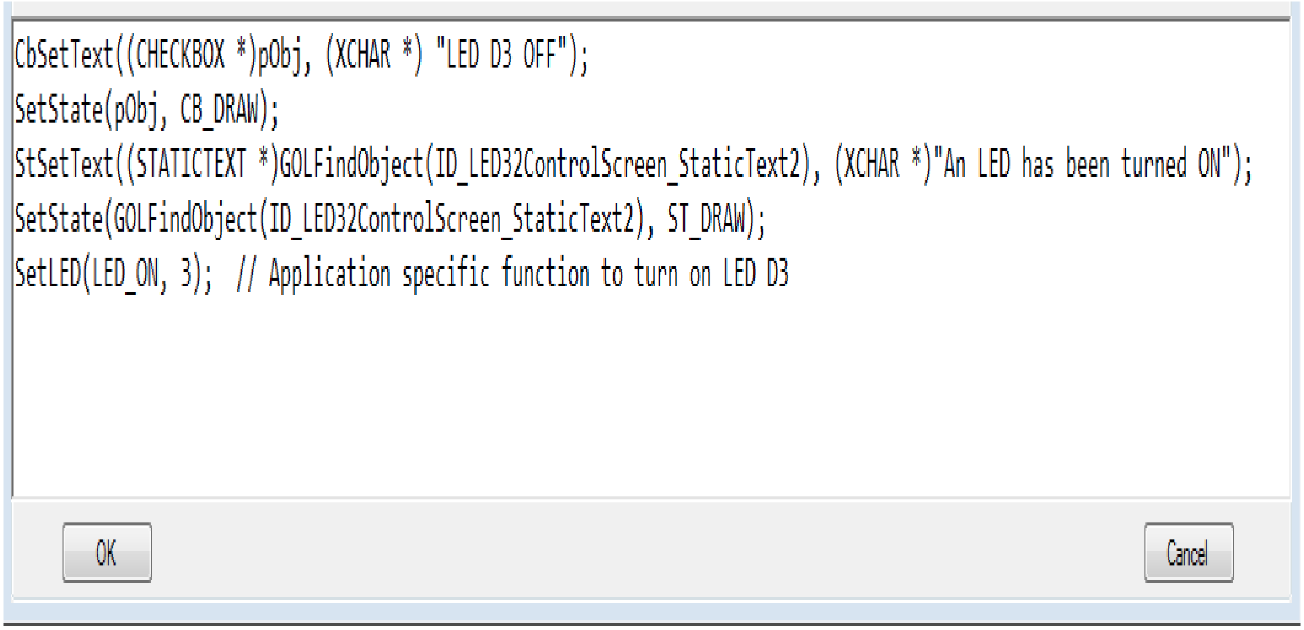
PIC24 Users:

```
SetLED(LED_ON, 1);
```

PIC32 Users:

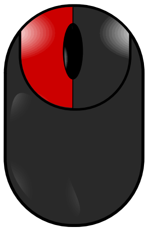
```
SetLED(LED_ON, 3);
```

When you are finished, the Event Editor window will resemble **Figure 3.8 or Figure 3.9** on page 3-7 of this manual.



```
CbSetText((CHECKBOX *)pObj, (XCHAR *) "LED D3 OFF");
SetState(pObj, CB_DRAW);
StSetText((STATICTEXT *)GOLFindObject(ID_LED32ControlScreen_StaticText2), (XCHAR *)"An LED has been turned ON");
SetState(GOLFindObject(ID_LED32ControlScreen_StaticText2), ST_DRAW);
SetLED(LED_ON, 3);  // Application specific function to turn on LED D3
```

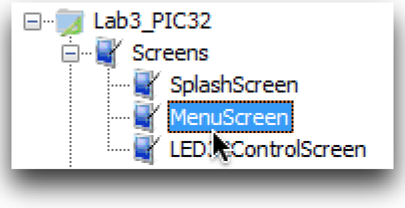
Figure 3.9
Event Editor with correct code inserted for PIC32 Users



When you are finished editing the code as shown in **Figure 3.8** or **Figure 3.9**, **click OK.** To save time, the appropriate code has already been added for the other checkbox events. **Please take the time to use the Events window to examine the other CheckBox messages.** Recall, double clicking on an event will open the Event Editor window.

4 On the MenuScreen, add the code to change the buttons on a press

In applications that use icons, changing the bitmaps on a user press will help the user see the press. Using VGDD, **select the MenuScreen by double clicking the MenuScreen in the Project Explorer window at the bottom of VGDD.**

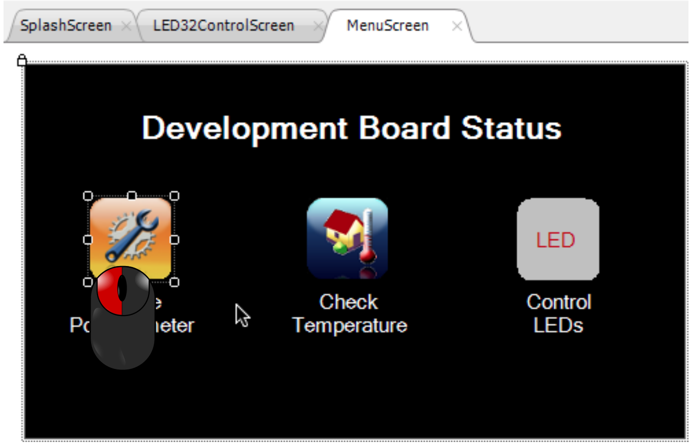


Information

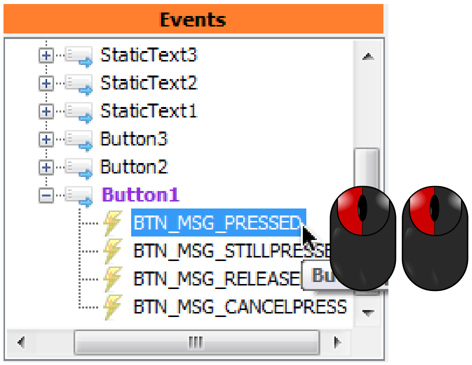
The messaging interface within the Microchip Graphics Library supports the following messages for buttons:

- BTN_PRESSED
- BTN_STILLPRESSED
- BTN_RELEASED
- BTN_CANCELPRESS

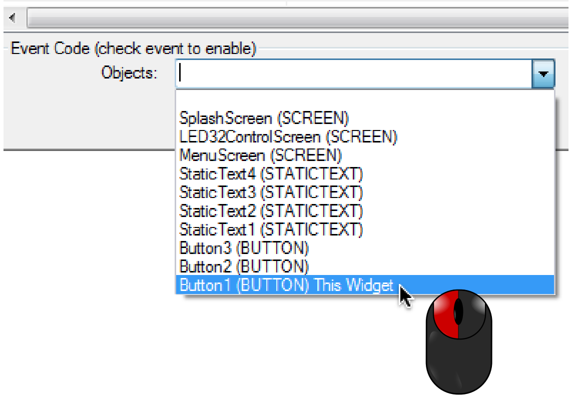
On the **MenuScreen**, use the mouse to select the **Rotate Potentiometer** button.



In the Events box, **double click the BTN_MSG_PRESSED under Button1** to open the Events Editor window.



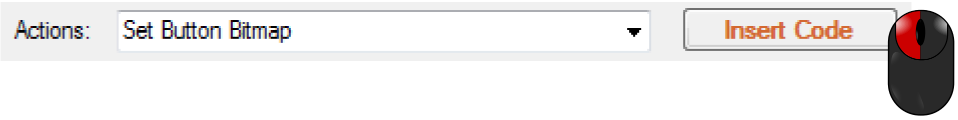
At the top of the Events Editor window, **check the box next to BTN_MSG_PRESSED.** In the Objects drop down menu, choose the **Button1 (BUTTON) This Widget** option.



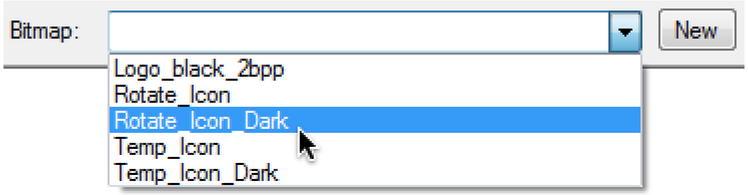
Attention

If you do not, see "**This Widget**" next to **Button1,** you have the wrong widget selected. Close the Events Editor window and go back to the Events screen as shown in **Figure 3.x.**

In the Actions drop down menu, select the Set Button Bitmap option then click the Insert Code button.



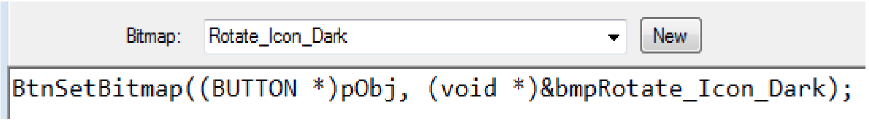
Notice that a new drop down menu appears called Bitmap. **Use the Bitmap drop down menu to select the Rotate_Icon_Dark image.**



Information

The **New** button next to the Bitmap drop down menu may be used to add images that are not already in the project.

With the image name selected, click the Insert Code button.



```
BtnSetBitmap((BUTTON *)pObj, (void *)&bmpRotate_Icon_Dark);
```

Attention

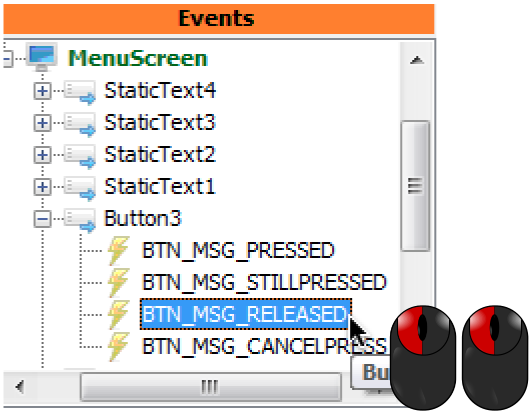
The default action performed by the GOL Messaging interface is to set the BTN_DRAW bit. As such, we do not need to select the Show/Update option for the button as we did for the CheckBox widget earlier in this lab.

Select OK to close the Event Editor window. To save time, the events for **BTN_STILLPRESSED** and **BTN_CANCELPRESS** have been populated for you. The **BTN_STILLPRESSED** event is set up to also show the dark icons while the button is pressed. The **BTN_CANCELPRESS** is set up to return the button to show the **Rotate_Icon** image when the user slides their finger off the button. We have also set up the events for the Read Temperature option for you.

5 Add an event so that pressing the LED button will change screens

Since changing screens is usually triggered by a single user event (e.g. button press), we will set the event to move the screen state machine as part of the GOL_MsgCallback() function. We will examine this state machine as well as the details on how the screen changes in the next lab.

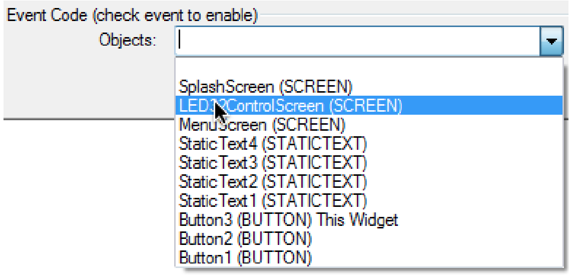
In the MenuScreen, **select the LED button** to the far right of the screen. In the Events box, **double click the BTN_MSG_RELEASED under Button3** to open the Events Editor window.



Attention

We are choosing to change the screen after the user releases the button so the user will have a chance to perceive the button press before a change occurs. When changing screens on BTN_MSG_PRESSED, the screen change happens immediately.

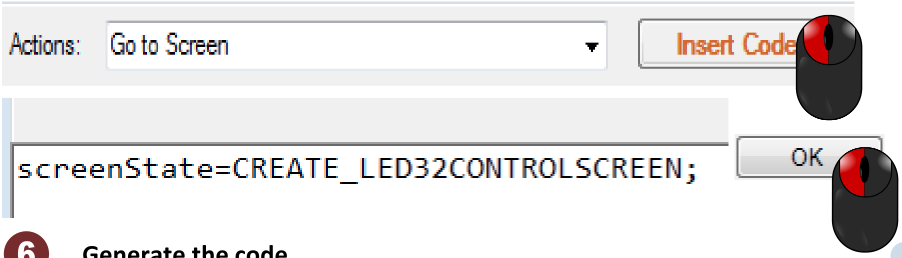
At the top of the Events Editor window, **check the box next to BTN_MSG_RELEASED.** In the Objects drop down menu, choose the **LED-ConrolScreen (or LED32ControlScreen for PIC32 users).**



Attention

Recall, we will always choose the object we want affected by the message. In this case, we want to switch to the LED Control screen, so we choose that screen in the Objects drop down menu.

In the Actions drop down menu, **choose Go to Screen, then click the Insert Code button.**



```
screenState=CREATE_LED32CONTROLSCREEN;
```

6 Generate the code.

For this lab, the MPLAB®X Wizard has been run for you; however, you will still need to generate the code to update the files with changes you just made. Click the Generate Code icon and verify the project settings.

If using a PIC32, use this project:
C:\RTC\HIF2132A\PIC32 Labs\Lab3\Lab3_PIC32.X

DANGER!

When VGDD generates code, the *<projectname>Screens.c* and *<projectname>Resources.c* files, along with their associated header files are always over written; therefore, it is important you do not place application code directly in those files. That is reason we placed the SetLED() function calls in the Events Editor window instead of the MPLABX project. Since the VGDDmain.c and VGDDmain.h files are only overwritten if they are selected for overwrite in the MPLAB®X Wizard, the application specific code needed to support the LEDs was added to those files.

7 Program the micro and view the results.

Now that we have source code, let's program the boards and see how it looks on the actual display.

Launch MPLAB ® X by double clicking the icon on the desktop

MPLAB X IDE

Using the information provided in the Hardware section at the beginning of this manual, verify you have the board properly connected, power applied and the programmer\debugger attached. Then open the appropriate MPLAB ® X project.

Build the code and program the device by clicking

Touch anywhere on the display to move to the menu screen. Observe the changes to the button images on press and release. Press the LED button to move to LED Control screen. Touch the checkboxes and observe the LEDs turning on and off.

Results

Using Visual Graphics Display Designer (VGDD) we have created a fully functional HMI with limited system interaction, based on single event user actions. The VGDD events handler allows us to fully populate the GOL_MsgCallback() functions. We have also learned where we may place our application code so it will not be overwritten by the VGDD code generator.

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. In general, code was added in the board initialization section in VGDDmain.c to set up the appropriate I/O for use with an LED. We also added a function called SetLED() to the VGDDmain.c file so that all we need to add to the VGDD event editor code is a single function call.

Conclusions

The GOL_MsgCallback() is used to control both the HMI and the system in reaction to a single user event (e.g. a button press). Using the VGDD events editor, we are quickly able to populate the GOL_MsgCallback() functions with minimal code addition.

THIS PAGE INTENTIONALLY LEFT BLANK

Lab Exercise 4

Primitive Layer Demo

Purpose

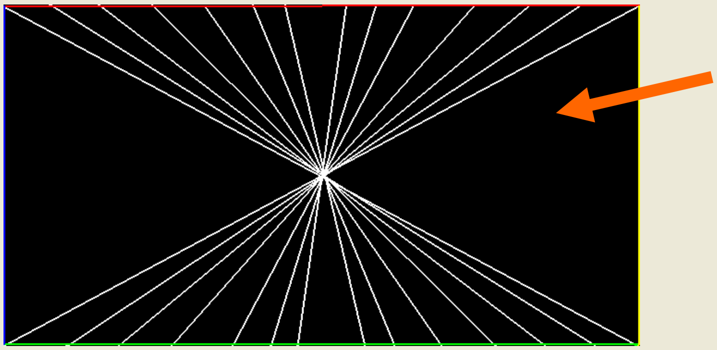
By the end of this lab you will have an LCD panel displaying a rainbow without using a graphics controller. You will see how to setup a PMP and DMA to act as a "virtual" graphics controller. Additionally, you will learn how to communicate to a display controller. We will be working in the Lab1.c file included in the Lab 1 project. For your convenience, a full solution is provided in **C:\RTC\GFX5 1670\Lab1\Lab1 Solution**.

Requirements

• **Development Environment:**	MPLAB®X IDE v1.20 or higher
Software:	
C Compiler:	MPLAB® XC32 Compiler for PIC32 v1.00
Hardware Tools:	PIC32 USB Starter Kit II Low Cost Controllerless (LCC) Graphics Board Graphics Display Truly 3.2" 240 x 320 Board

Objectives

1) Create a frame buffer
2) Initialize DMA and PMP peripherals.
3) Run the Primitive demo and see it appear on the LCD.



Microchip's Primitive is running

Procedure

1

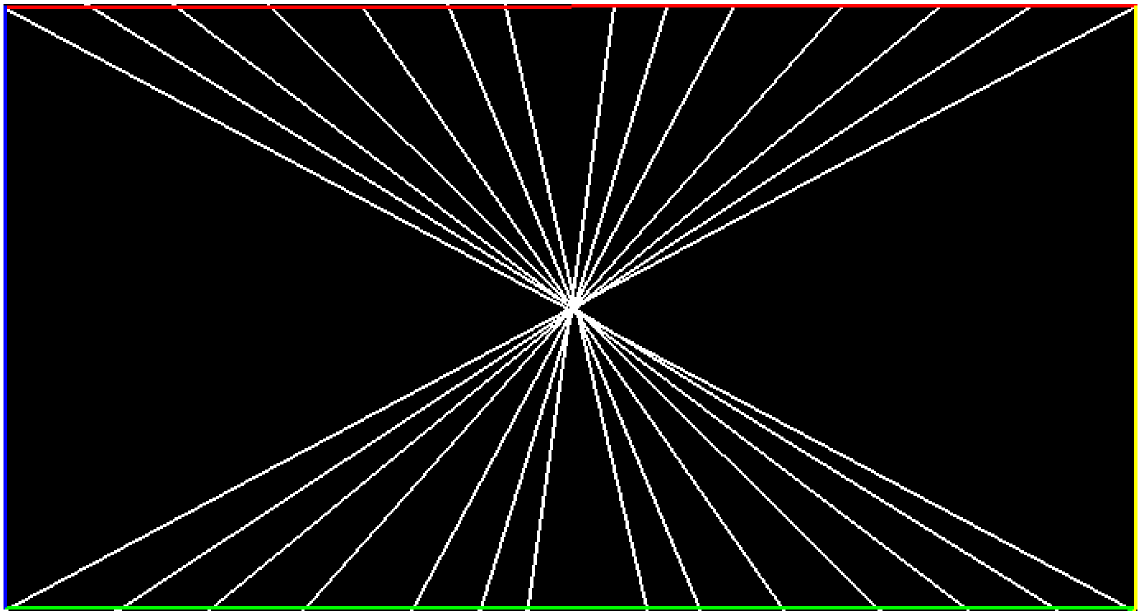
If you still have a previous project open, you can close it by selecting from the menu:
File ▸ Close

Then, open Lab 4 by selecting from the menu:
File ▸ Open Project... and opening the workspace file appropriate for you development board set

:Choose
The MPLABX project for the LCC board with the PIC32 USB starter kit II (PIC32MX795F512L)

2

Click Build in MPLAB, then program and run the device. If everything is correctly configured you should see screens showing various object (primitives) such as rainbow objects and embedded images.



3 Observe that the only time the CPU is interrupted from performing tasks in an LCC environment is during the DMA ISR. The ISR only takes on average around 60-70 clock cycles. Also observe how easy of an interrupt routine it is for handling such a complex task.

4 Open maindemo.c which can be found in the Lab 1 Sol folder. Try changing some of the

```
SetColor(XXXXX); macros to another color such as:

BRIGHTBLUE
BRIGHTRED
BRIGHTGREEN
WHITE
BLACK

And see how this effects the demo.
```

5 Compile the project and program the device. Hit run
Try and see the change you made to the demo..

Did the Primitive Layer Demo appear on your LCD display? If it did, you have now completed Lab 1. **Great Job!**

Results

You have just learned how to setup a "virtual" display controller and the steps that are involved. This method takes a seemingly complex problem, breaks it down into its proper steps, and displays the image on an LCD display panel.

Code Analysis

The simple initiation routine that you created does many things. First off it creates a communication channel from the PIC to the display. Next, it sets up the graphics frame starts the continuous DMA transfer. It then runs the Primitive Layer Demo found in Microchips MAL to ensure the controller is running properly.

Conclusions

Having learned how to setup a DMA and PMP peripheral to act as a display controller, you are now ready to start creating applications. This cost effective solution helps keep all tasks on one processor while still supplying most of the needs of a graphics controller. It is hard to complain when having a graphics controller with plenty of MIPs left for other purposeful application space.

Appendix A

MPLAB® X IDE Quick Reference Guide

Table of Contents

1. Managing Projects

2. Building Projects

3. Debugging Projects

GFX

This appendix is intended to be a supplement to lab manuals supplied with a Microchip Technical Training class. Although it may be useful on its own, it is not intended to provide complete instructions for using all aspects of the MPLAB X Integrated Development Environment. For more detailed information on the use of MPLAB X IDE, please consult one of the following Microchip Technical Training classes (for additional details, see **http:\\www.microchip.com\RTC**):
TLS0101—Getting Started with MPLAB X IDE
TLS0999—Transitioning to MPLAB X IDE for users of MPLAB IDE version 8
Or consult the Microchip Developer's Help Center at **http:\\microchip.wikidot.com\mplab:_start**

Trademarks

1. Managing Projects

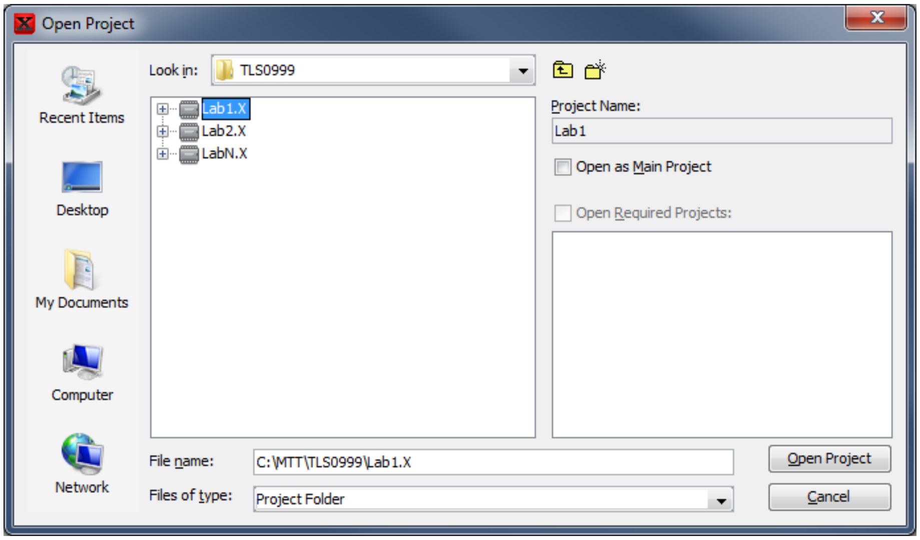
Section 1.1

How to open a project

Unlike some other IDEs, there is no single icon you can double click on from your operating system's file manager. MPLAB X Projects must be opened from within the IDE.

1. There are several methods you can use to launch the **Open Project** dialog:
   Method 1:
   Click on the **Open Project** icon on the main toolbar:
   Method 2:
   From the menu, select **File** ▶ **Open Project**
   Method 3:
   Using the keyboard: Ctrl ⇧ Shift O

2. Navigate to the project's directory and select the directory itself, which is represented by a chip icon instead of the usual folder. In MPLAB X IDE, there is no single project document, so the project's directory is used to represent the project in the **Open Project** dialog.



Look in: TLS0999
Project Name: Lab1
☐ Open as Main Project
☐ Open Required Projects:
File name: C:\MTT\TLS0999\Lab1.X
Files of type: Project Folder

Figure 1.1.1
An Open Project dialog showing three projects in the TLS0999 directory

Information

In most Microchip Technical Training classes, projects are stored in the following directory:

..\RTC\*classcode*

where *classcode* is specified in the presentation or lab manual.

3. Click on the **Open Project** button. You should now see a populated project tree in the IDE (you may need to click on the '+' next to the chip icon to expand the subfolders.

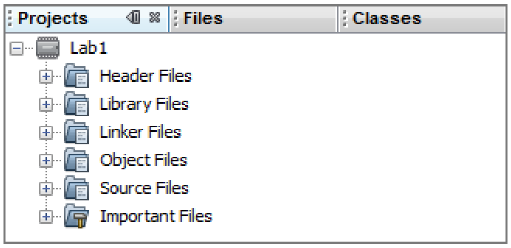


Figure 1.1.2
A populated project tree after opening a project

Section 1.2

How to add existing files to a project

Attention

Do not select **Add Existing Items from Folders…** as this will dump everything from the directory you choose into the selected logical folder.

① Right click on the logical folder in the project tree (e.g. Source Files, Header Files, etc.) where you wish to add the file(s) and select **Add Existing Item…** from the popup menu.

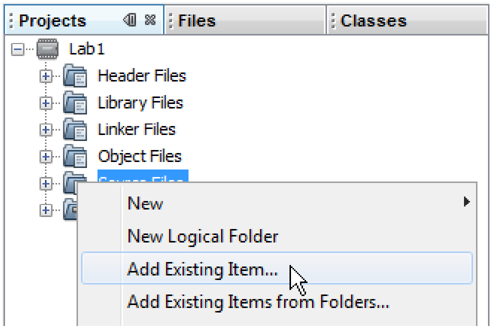
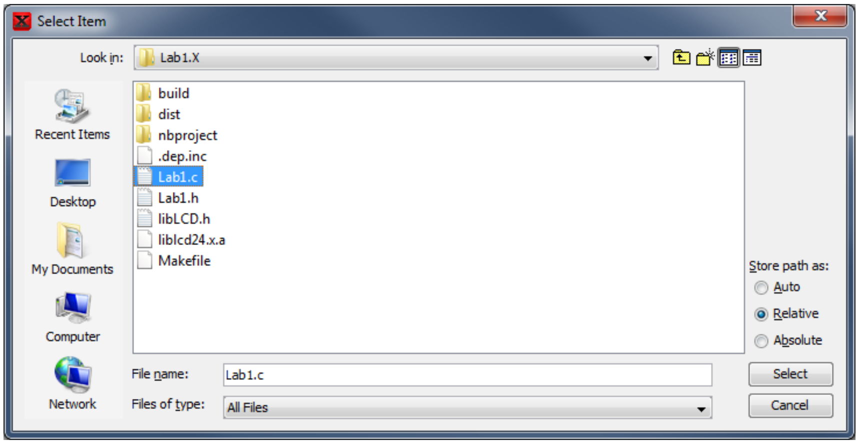


Figure 1.2.1
Popup menu displayed after right clicking on the Source Files logical folder in the project tree

② The Select Item dialog box will open and display the contents of your project directory. Select one or more files (Ctrl + click to select additional files).



Look in: Lab1.X

File name: Lab1.c

Files of type: All Files

Store path as: ● Relative

Figure 1.2.2
The Select Item dialog

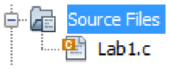
In most cases you can leave the **Store path as:** radio buttons set to **Relative**, but you may select **Auto** or **Absolute** if required for your project.

Relative: Stores paths to files relative to the project directory. For example: **\Lab1.c**, or **..\OtherDirectory\Somefile.c**
Relative is usually the best choice for files inside your project directory.

Absolute: Stores paths to files with full path from root directory. For example: **..\RTC\TLS0999\Lab1\Lab1.c**
Absolute is the best choice for files outside your project directory that won't be moved such as code shared among several projects or libraries.

Auto: Automatically uses Relative for files inside the project directory and Absolute for files outside of the project directory.

③ Click the **Select** button. You should now be able to expand the selected folder in the project tree and see that the files have been added to your project.



Section 1.3

How to create new files in a project

1 There are three methods you may use to launch the new file wizard:

Method 1:

Right click in the Projects window and select **New** ▸ ***file-type*** from the popup menu:

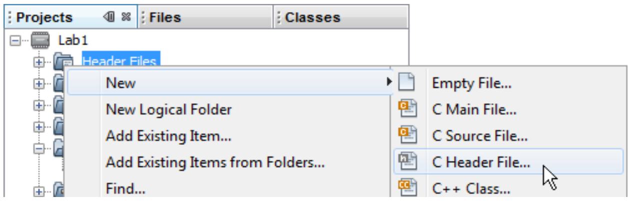


Figure 1.3.1
New file popup menu

Method 2:

Click on the New File icon on the main toolbar:

Method 3:

From the main menu select **File** ▸ **New File...**

Method 4:

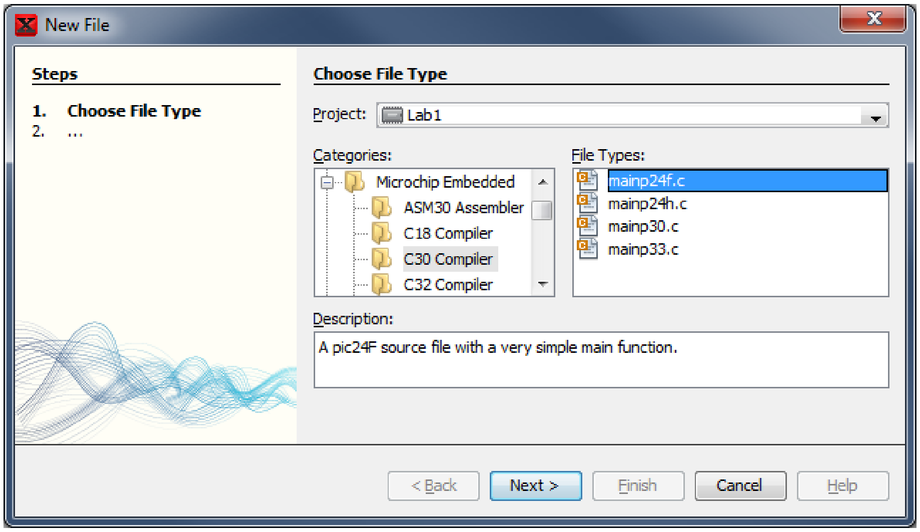
Using the keyboard: Ctrl N

2 If you chose method 1 above, skip to step 3. If you chose one of the other methods above, you will be presented with the screen shown in Figure A.6 below. In the **New File** dialog, select the type of file you wish to create. Any file created under the Microchip Embedded category or an Empty File from the Other category will be automatically added to the project tree. Other file types may need to be added manually.

Click the **Next >** button after you have made your selection.

Information

You can create a "main" file anytime you like and just delete the code that is automatically inserted to make it a regular C file.



Project: Lab1

Description:
A pic24F source file with a very simple main function.

Figure 1.3.2
New File wizard—Choose File Type

3 The dialog now prompts you for a file name and potentially a file type depending on your initial selection. It also prompts you for a folder. You can leave this blank and the IDE will automatically create the file inside your project directory. If you wish to locate the file elsewhere, click on the **Browse...** button and choose a different location.

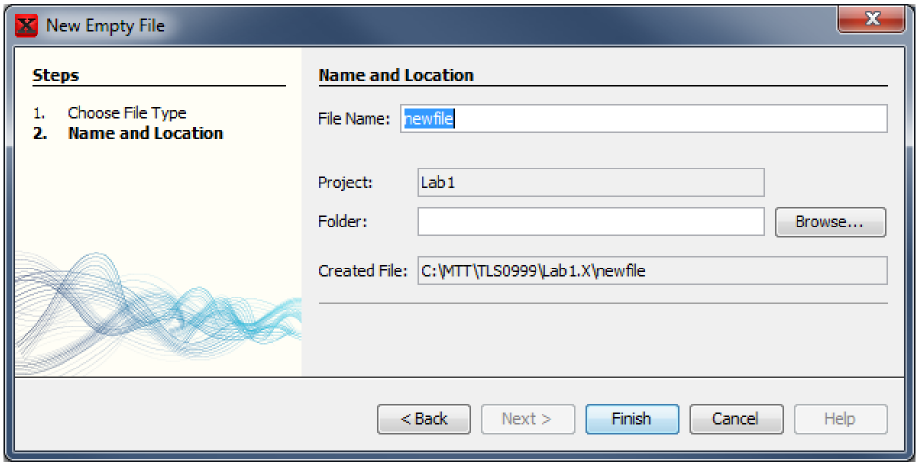
If you chose **Empty File** as the new file type, you will see the dialog in Figure A.7 on the next page.

If you chose one of the types under the **Microchip Embedded** category, you will see a dialog like the one in Figure A.8 on the next page.

Figure 1.3.3
New Empty File Dialog

Attention

You must specify a file extension (e.g. myNewFile.c) as part of the File Name if you want the file to be added to the project tree automatically into the correct logical folder.



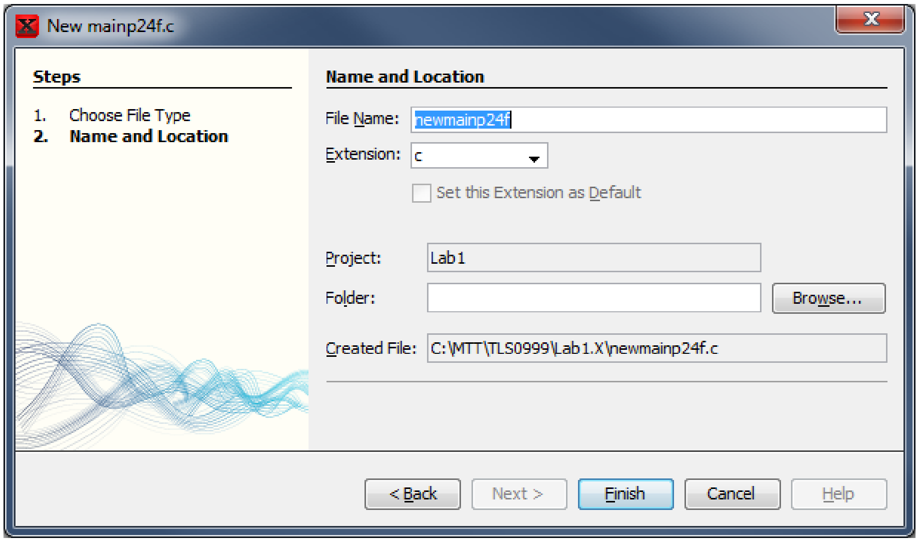
1. Choose File Type
2. **Name and Location**

File Name: newfile
Project: Lab1
Folder:
Created File: C:\MTT\TLS0999\Lab1.X\newfile

Figure 1.3.4
New Main File Dialog (Microchip Embedded category)

Information

It is not necessary to specify the file extension as part of the filename as long as the correct extension is selected below.

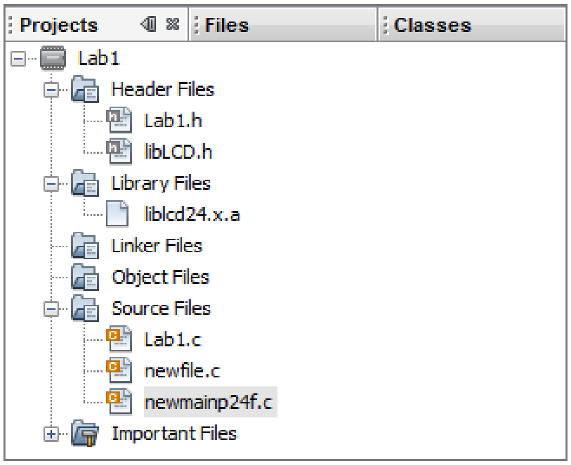


1. Choose File Type
2. **Name and Location**

File Name: newmainp24f
Extension: c
☐ Set this Extension as Default
Project: Lab1
Folder:
Created File: C:\MTT\TLS0999\Lab1.X\newmainp24f.c

4 After providing all of the required information, click the **Finish** button. You should now see the new file in the project tree. If you don't see it, you may need to add the file to the project (see page A-4)

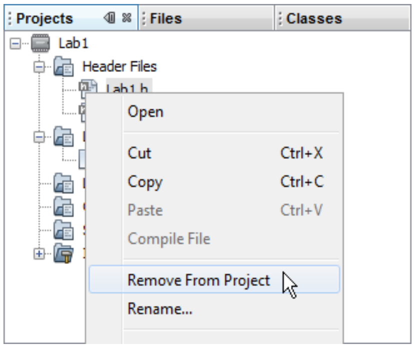
Figure 1.3.5
Project tree with newfile.c and newmainp24f.c added to the project.



- Lab1
  - Header Files
    - Lab1.h
    - libLCD.h
  - Library Files
    - liblcd24.x.a
  - Linker Files
  - Object Files
  - Source Files
    - Lab1.c
    - newfile.c
    - newmainp24f.c
  - Important Files

Section 1.4
How to remove a file from a project

1. Right click on the file you wish to remove and select from the popup menu **Remove From Project**



DANGER!

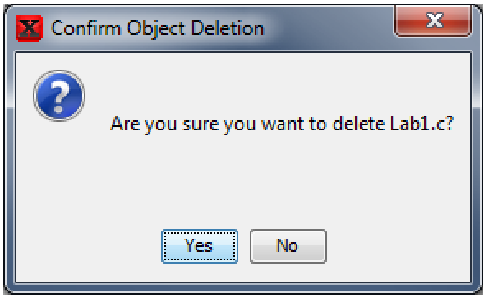
DO NOT use the **Delete** key to simply remove a file from your project. This will also ***PERMANENTLY DELETE*** the file from your hard drive. It will not be recoverable from the Trash folder.

Figure 1.4.1
Project right click menu with Remove From Project selected

Section 1.5
How to *permanently* delete a file

1. Select a file in the project tree and press Delete

This will permanently delete the file from you system in addition to removing it from the project. The file will not be recoverable from the Trash.



Are you sure you want to delete Lab1.c?

Figure 1.5.1
Confirm delete dialog. This will be your only warning.

Section 1.6
How to save a file or project

1. The entire project along with all of its files are saved automatically each time you build your code. However you may explicitly save the project and all its files using one of two methods.
Method 1:
Click the “double floppy disk” icon in the toolbar:
Method 2:
From the main menu select **File ▸ Save All**

Though it is not usually necessary to just save a single file, you can do so by selecting the file in the editor and then from the main menu **File ▸ Save**.

Section 1.7

How to close a project

1 There are two methods you can use to close a project:

Method 1:

Right click on the top node of the project in the project tree (the chip icon) and select **Close** from the popup menu (about 2\3 of the way down).

Method 2:

From the main menu, select **File ▸ Close Project (*project name*)** where ***project name*** is the name of the project you wish to close—there may be multiple similar menu items if you have more than one project open in the IDE.

Section 1.8

How to modify project settings

1 There are three ways to access a project's settings:

Method 1:

Right click on the top node (chip icon) of a project in the project tree and select **Properties** at the very bottom of the long popup menu.

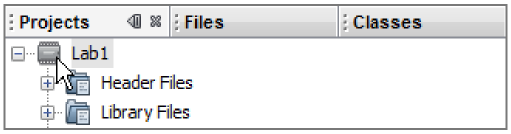


Figure 1.8.1

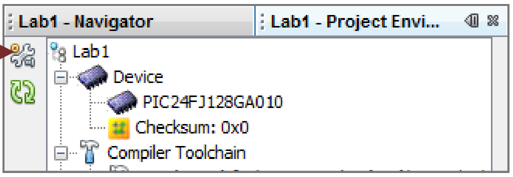
The top node of a project in the project tree

Method 2:

From the main menu select **File ▸ Project Properties (*project name*)**

Method 3:

If the **Project Environment** window is open (bottom left corner by default), you can click on the "wrench and bolt" icon in its left margin.

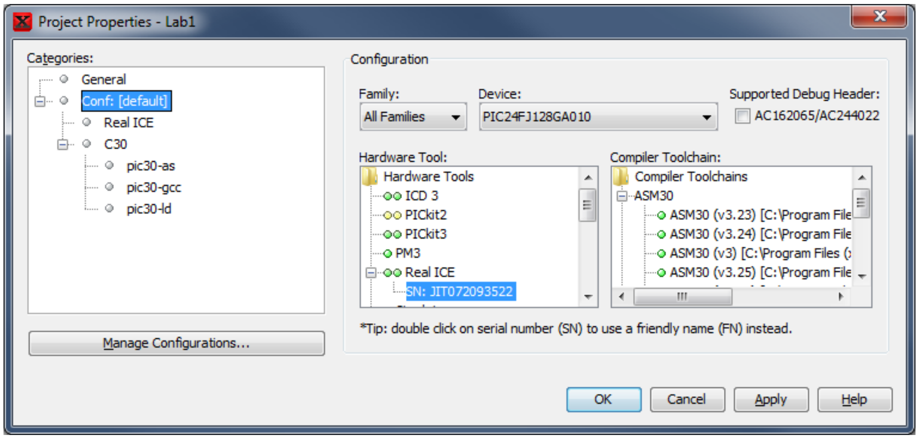


Project Properties

Figure 1.8.2

Project Properties button in the Project Environment window.

2 From here you can select a different device, debug tool or build tool and you can modify any of their settings. When choosing a new tool, click the Apply button to make it show up in the tree on the left side.



Family: All Families

Device: PIC24FJ128GA010

Supported Debug Header: ☐ AC162065/AC244022

*Tip: double click on serial number (SN) to use a friendly name (FN) instead.

2. Building Projects

Section 2.1

How to build a project

There are several ways to build a project in MPLAB X depending on what you intend to do with the results. This method is only used to see the results of a build or to produce a release mode hex file.

There are two different types of build you can do:
Build:
This will build only the files in your project that have changed since the last build or it will build everything if nothing has been built previously. It will generally be faster to use this type of build, especially for larger projects.
Clean and Build:
This will remove any intermediate files generated by the previous build and will build every file in your project regardless of whether or not it has changed since the last build to ensure a full, clean build.

Information

The **Build** and **Clean and Build** functions are not intended for use before **Run Project**, **Debug Project** or **Make and Program Target**. All three of those functions automatically do a build before performing further steps. No harm will be done by using **Build** or **Clean and Build** first, but it will be a duplication of effort and will waste time.

①

There are two ways to access these two build types:
From the Main Toolbar:

Build	***Make*** in MPLAB IDE 8
Clean and Build	***Build All*** in MPLAB IDE 8

Alternatively, you can right click on the top node of a project (chip icon) in the project tree and select either **Build** or **Clean and Build** from the popup menu.

Section 2.2

How to build and run a project with a debugger

When you want to build a project for the purpose of programming a target to run with a debugger like the MPLAB® ICD 3 or REAL ICE, this is the method to use.

Information

It is not necessary to do a **Build** or **Clean and Build** before doing a **Debug Project** because a build will be done automatically.

①

There are three ways to build and run your code through a debugger:
Method 1:
Click on the **Debug Project** button on the main toolbar
Method 2:
Right click on the top node of the project (chip icon) in the project tree and select **Debug** from the popup menu.
Method 3:
From the main menu, select **Debug** ▸ **Debug Project (*project name*)**

This will perform the following tasks automatically:

a. a. Build (make) project in debug mode
b. b. Program target (unless using simulator)
c. c. Run code on target

Section 2.3

How to build and run a project without a debugger

When you want to build a project for the purpose of programming a target to run without a debugger, this is the method to use.

① There are three ways to build and run your code on a target:

Method 1:

Click on the **Run Project** button on the main toolbar ▷

Method 2:

Right click on the top node of the project (chip icon) in the project tree and select **Run** from the popup menu.

Method 3:

From the main menu, select **Run ▸ Run Project (*project name*)**

This will perform the following tasks automatically:

a. a. Build (make) project in release mode
b. b. Program target
c. c. Run code on target

Information

It is not necessary to do a **Build** or **Clean and Build** before doing a **Run Project** because a build will be done automatically.

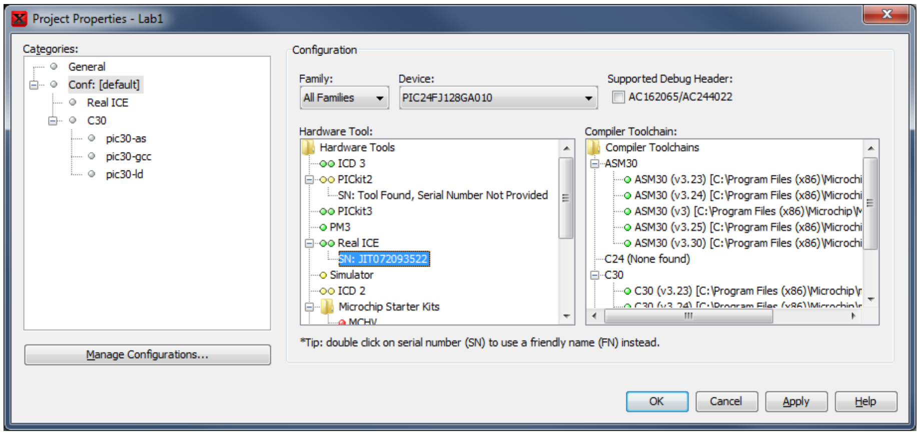
3. Debugging Projects

Section 3.1

How to set or change the debugger

① Open the project properties window (see page A-8 “How to modify a project’s settings” for details)

② In the center column under “Hardware Tool”, click on the serial number under the name of the tool you wish to use. If choosing the simulator, just click on “Simulator” since no serial number is associated with it. Some tools do not provide a serial number. In that case, click on the text right below the name of the tool (see PICkit 2 in the figure below for an example).

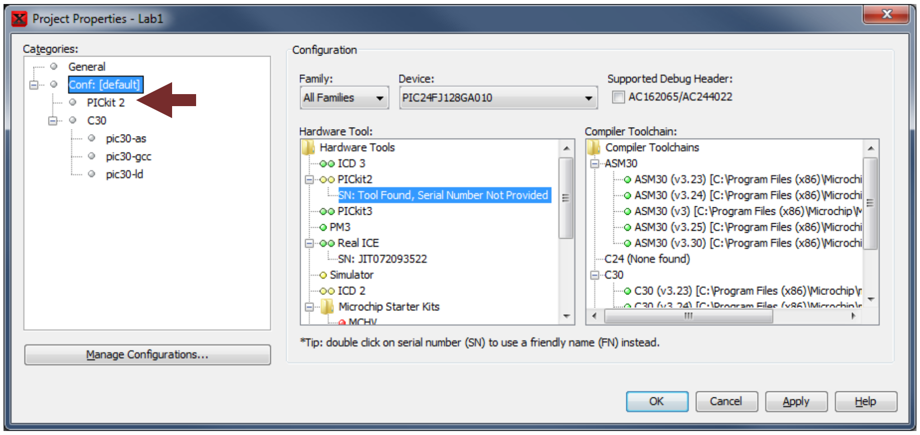

Project Properties - Lab1
Categories:
General
Conf: [default]
Real ICE
C30
pic30-as
pic30-gcc
pic30-ld
Manage Configurations...
Configuration
Hardware Tool:
Hardware Tools
ICD 3
PICkit2
SN: Tool Found, Serial Number Not Provided
PICkit3
PM3
Real ICE
SN: JIT072093522
Simulator
ICD 2
Microchip Starter Kits
MCHV
Compiler Toolchain:
Compiler Toolchains
ASM30
ASM30 (v3.23) [C:\Program Files (x86)\Microchi
ASM30 (v3.24) [C:\Program Files (x86)\Microchi
ASM30 (v3) [C:\Program Files (x86)\Microchip\
ASM30 (v3.25) [C:\Program Files (x86)\Microchi
ASM30 (v3.30) [C:\Program Files (x86)\Microchi
C24 (None found)
C30
C30 (v3.23) [C:\Program Files (x86)\Microchip\
OK
Cancel
Apply
Help

Family: All Families
Device: PIC24FJ128GA010
Supported Debug Header: ☐ AC162065/AC244022

*Tip: double click on serial number (SN) to use a friendly name (FN) instead.

Figure 3.1.1
Selecting a hardware tool in the Project Properties window

③ Click on the **Apply** button after you make your selection and you should see the selected tool in the tree of the left column.


Project Properties - Lab1
Categories:
General
Conf: [default]
PICkit 2
C30
pic30-as
pic30-gcc
pic30-ld
Manage Configurations...
Configuration
Hardware Tool:
Hardware Tools
ICD 3
PICkit2
SN: Tool Found, Serial Number Not Provided
PICkit3
PM3
Real ICE
SN: JIT072093522
Simulator
ICD 2
Microchip Starter Kits
MCHV
Compiler Toolchain:
Compiler Toolchains
ASM30
ASM30 (v3.23) [C:\Program Files (x86)\Microchi
ASM30 (v3.24) [C:\Program Files (x86)\Microchi
ASM30 (v3) [C:\Program Files (x86)\Microchip\
ASM30 (v3.25) [C:\Program Files (x86)\Microchi
ASM30 (v3.30) [C:\Program Files (x86)\Microchi
C24 (None found)
C30
C30 (v3.23) [C:\Program Files (x86)\Microchip\
OK
Cancel
Apply
Help

Family: All Families
Device: PIC24FJ128GA010
Supported Debug Header: ☐ AC162065/AC244022

*Tip: double click on serial number (SN) to use a friendly name (FN) instead.

Figure 3.1.2
Changing a hardware tool in the Project Properties window

④ Clicking on the tool in the tree of the left column will display the tool’s properties in the right side of the window, where you can modify them to suit your project’s needs. Click **OK** when finished.

Section 3.2
How to control program execution when debugging

Debug Toolbar Buttons

Finish Debug Session (Shift + F5)
This is required before you make any changes to your project settings or source code.

Pause

Reset

Continue (F5)

Step Over (F8) - Execute each line without stepping into functions (functions are executed without stepping through each line)

Step Into (F7) - Execute each line and step into functions

Run to Cursor (F4)

Set PC at Cursor

Focus Cursor at PC

Additional functions may be found in the **Debug** menu.

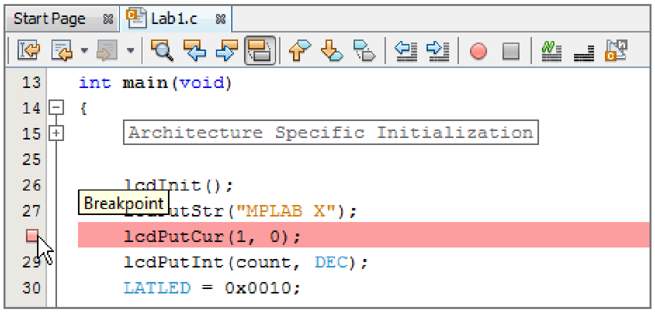
Information

When you hover the mouse pointer over a toolbar button a tool tip will be displayed explaining the function of the button.

Section 3.3
How to set and clear breakpoints

① Standard line breakpoints may be set or cleared by clicking on the line number in the glyph margin.

Figure 3.3.1
Setting \ clearing a breakpoint



```
int main(void)
{
    Architecture Specific Initialization

    lcdInit();
    lcdPutStr("MPLAB X");
    lcdPutCur(1, 0);
    lcdPutInt(count, DEC);
    LATLED = 0x0010;
```

More advanced breakpoint features may be accessed by opening the breakpoints window. From the main menu select **Window ▶ Debugging ▶ Breakpoints**.
In the breakpoint window, right click on a breakpoint in the list and select **Customize** or **Complex Breakpoint** from the popup menu for advanced options.

Section 3.4

How to use the stopwatch

1. From the main menu select **Window ▸ Debugging ▸ Stopwatch**

2. In the stopwatch window, click on the Properties button in its margin.

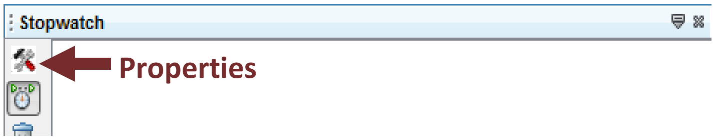
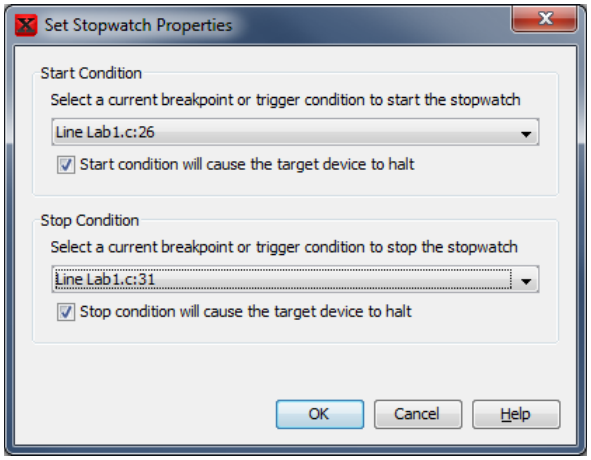


Figure 3.4.1
Stopwatch properties button

3. Select existing breakpoints for the start and stop conditions



Set Stopwatch Properties

Start Condition

Select a current breakpoint or trigger condition to start the stopwatch

Line Lab1.c:26

☑ Start condition will cause the target device to halt

Stop Condition

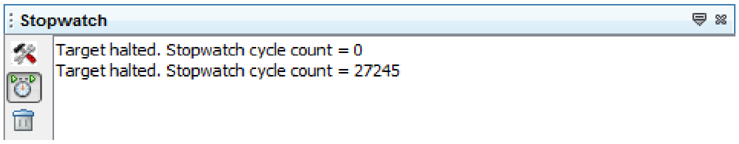
Select a current breakpoint or trigger condition to stop the stopwatch

Line Lab1.c:31

☑ Stop condition will cause the target device to halt

Figure 3.4.2
Stopwatch properties window

4. Run your code. The cycle count will be displayed each time you hit one of the selected breakpoints. The Trash icon in the margin will reset the cycle count.



```
Target halted. Stopwatch cycle count = 0
Target halted. Stopwatch cycle count = 27245
```

Figure 3.4.3
Stopwatch displaying results from the starting and ending breakpoints

Section 3.5

How to display and use Watches

① From the main menu select **Window ▸ Debugging ▸ Watches** or from the keyboard press Alt ⇧Shift 2

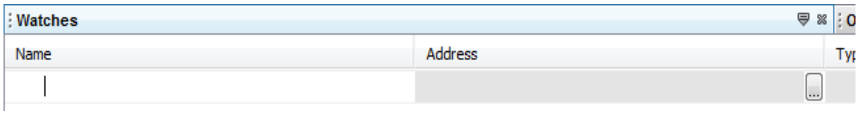
② The **Watches** window should appear as a new tab near the **Output** window in the bottom center of the IDE. There are three ways to add a watch value:

Method 1:

Double click on the first empty line in the **Name** column and type in the name of the variable\register

Figure 3.5.1

Adding a watch variable directly to the watches window

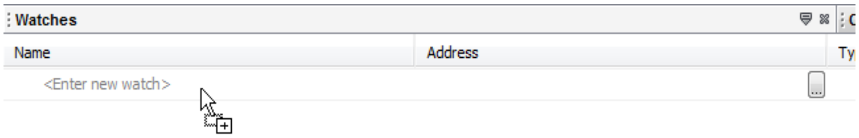


Method 2:

Double click on a variable or register name in the editor to select it. Then left click and drag it to the Watches window.

Figure 3.5.2

Dragging a variable from the editor to the watches window

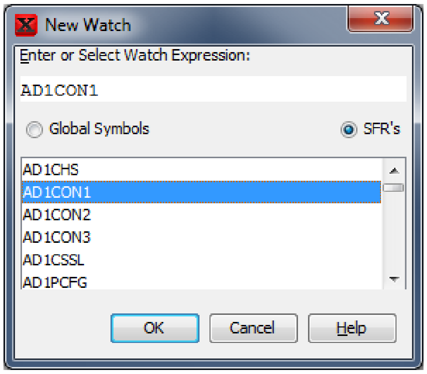


Method 3:

Right click on a variable\register name in the editor or on a new line in the **Watches** window and from the popup menu select **New Watch...** , then in the window that opens, enter the desired variable\register name in the text box or select it from the list by choosing **Global Symbols** or **SFRs**

Figure 3.5.3

New Watch dialog box



Enter or Select Watch Expression: AD1CON1

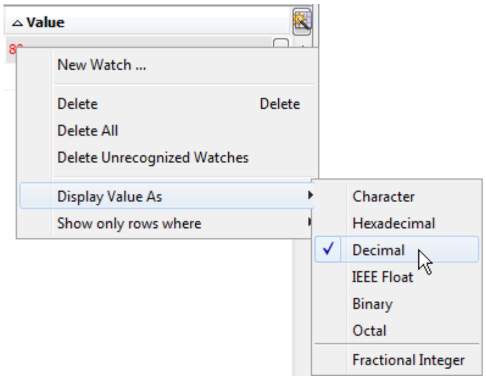
○ Global Symbols ● SFR's

Information

Watch variables may be sorted by clicking on the **Name** column.

Figure 3.5.4

Changing the Value display format



Information

To change the display value, right click on an entry in the Value column and select **Display Value As** from the popup menu.

Section 3.6

How to view Embedded Memory

(RAM, SFRs, Flash, EEPROM or Configuration Bits)

1. From the main menu select **Window ▶ PIC Memory Views ▶ Memory View *n*** where ***n*** is a value from 1 through 4. It doesn't matter which one you choose initially as they are all identical and configurable.

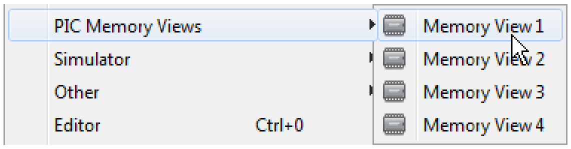
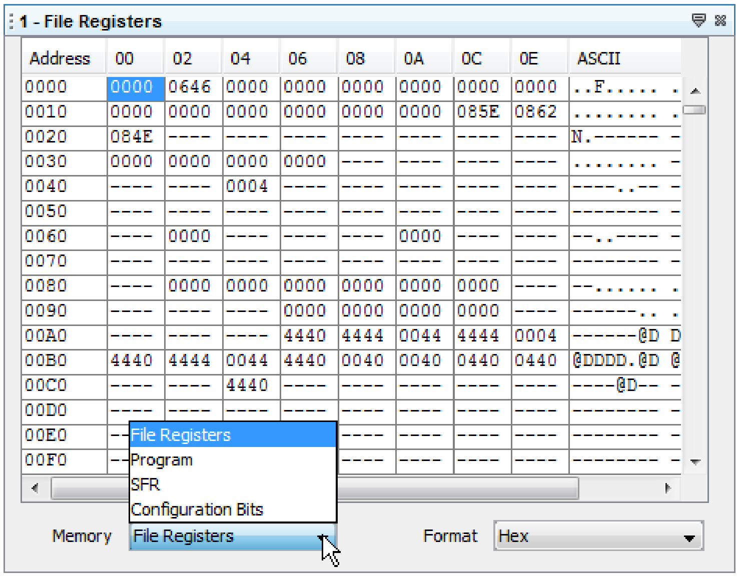


Figure 3.6.1
PIC Memory Views menu item

2. A new tab will open by default in the bottom center part of the IDE. At the bottom left of this window is a combo box labeled "Memory" that is used to configure this window to display any valid memory type for the currently selected device.



```
Address  00    02    04    06    08    0A    0C    0E    ASCII
0000     0000  0646  0000  0000  0000  0000  0000  0000  ..F..... .
0010     0000  0000  0000  0000  0000  0000  085E  0862  ........ .
0020     084E  ----  ----  ----  ----  ----  ----  ----  N.------ -
0030     0000  0000  0000  0000  ----  ----  ----  ----  ........ -
0040     ----  ----  0004  ----  ----  ----  ----  ----  ----..-- -
0050     ----  ----  ----  ----  ----  ----  ----  ----  -------- -
0060     ----  0000  ----  ----  ----  0000  ----  ----  --..---- -
0070     ----  ----  ----  ----  ----  ----  ----  ----  -------- -
0080     ----  0000  0000  0000  0000  0000  0000  ----  --...... .
0090     ----  ----  ----  0000  0000  0000  0000  ----  ------.. .
00A0     ----  ----  ----  4440  4444  0044  4444  0004  ------@D D
00B0     4440  4444  0044  4440  0040  0040  0440  0440  @DDDD.@D @
00C0     ----  ----  4440  ----  ----  ----  ----  ----  ----@D-- -
00D0     ----  ----  ----  ----  ----  ----  ----  ----  -------- -
00E0     --                      ----  ----  ----  ----  -------- -
00F0     --                      ----  ----  ----  ----  -------- -
```

Memory: File Registers
Format: Hex

Figure 3.6.2
PIC Memory View showing File Registers with other options in the Memory combo box

Information

The **Format** combo box at the bottom right of the memory view window is different for each memory type, but will configure the display in a variety of numeric and symbolic formats.