

**PIC32MZ DA Radial Graphics
Development Board and
MPLAB® Harmony Graphics Suite (MHGS)
PIC3001GFX v1.00**



A Leading Provider of Smart, Connected and Secure Embedded Control Solutions



SMART | CONNECTED | SECURE

**CAE Taiwan
Microchip Technology Inc.**
February 2023

Index

- [Microchip Graphics Portfolio and PIC32MZ DA Brief](#)
- [PIC32MZ DA Radial Graphics Development Board Introduction](#)
- [Software Preparation and Design Reference](#)
- [A New Harmony Project](#)
- [Board Support Packages \(BSP\)](#)
- [Development Board Template](#)
- [Legato Graphic Composer](#)
- [Schemes](#)
- [Event Handling](#)
- [Button Widget Event](#)
- [Boundary Circle Widget](#)
- [Graphics Assets](#)
- [Image Assets Manager](#)
 - [Create Your Design](#)
 - [Generate Transparent Background Image](#)

Index

- [Custom Button Widget](#)
- [Deducing Image Storage Space with Optimum Color Palette](#)
- [Font Assets Manager](#)
- [String Assets Manager with Multi-Lingual Label](#)
- [Label Widget with Dynamic Text](#)
- [External Graphics Resources - Font](#)
- [Multi-Screen and Splash Screen](#)
- [Multi-Layering](#)
- [Image Preprocess](#)
- [Image Preprocess without using Graphics Composer](#)
- [GPU Bit Blit Stretch Image](#)
- [CANVAS Virtual Display](#)
- [CANVAS Animation Effect](#)

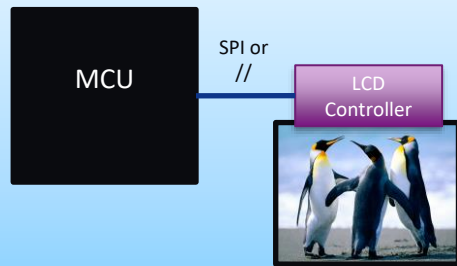
32-bit MCU/MPU for Graphics Application

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

32-bit MCU/MPU for Graphics Application

Hardware Overview

MCU + External LCD Controller



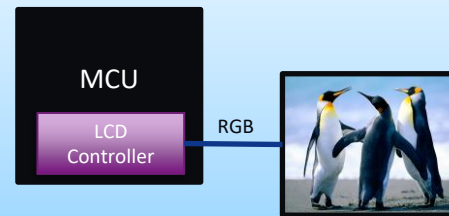
- ✓ Suitable for all display sizes
- ✓ Dedicated Display Controller
- ✓ Reduced overhead on MCU
- ✓ Advanced Graphics Capabilities
- ✓ Up to 384KB integrated display buffer

MCU w/ LCD Controller Emulation Low-Cost Controller-Less (LCC)



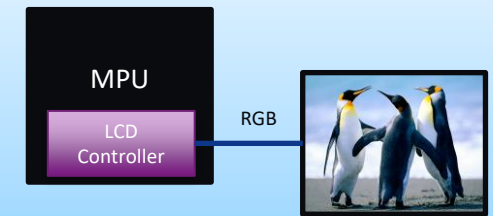
- ✓ Displays up to 800 x 480
- ✓ Requires no LCD Controller
- ✓ Lowers system cost
- ✓ Suitable for highest integration and simple systems
- ✓ Up to 512KB integrated display buffer

MCU w/ integrated LCD Controller & GPU



- ✓ Displays up to 800 x 480
- ✓ Integrated LCD Controller w/ 2D GPU
- ✓ Reduced overhead on MCU
- ✓ Advanced Graphics Capabilities
- ✓ Up to 32MB DDR2 Integrated for display buffer

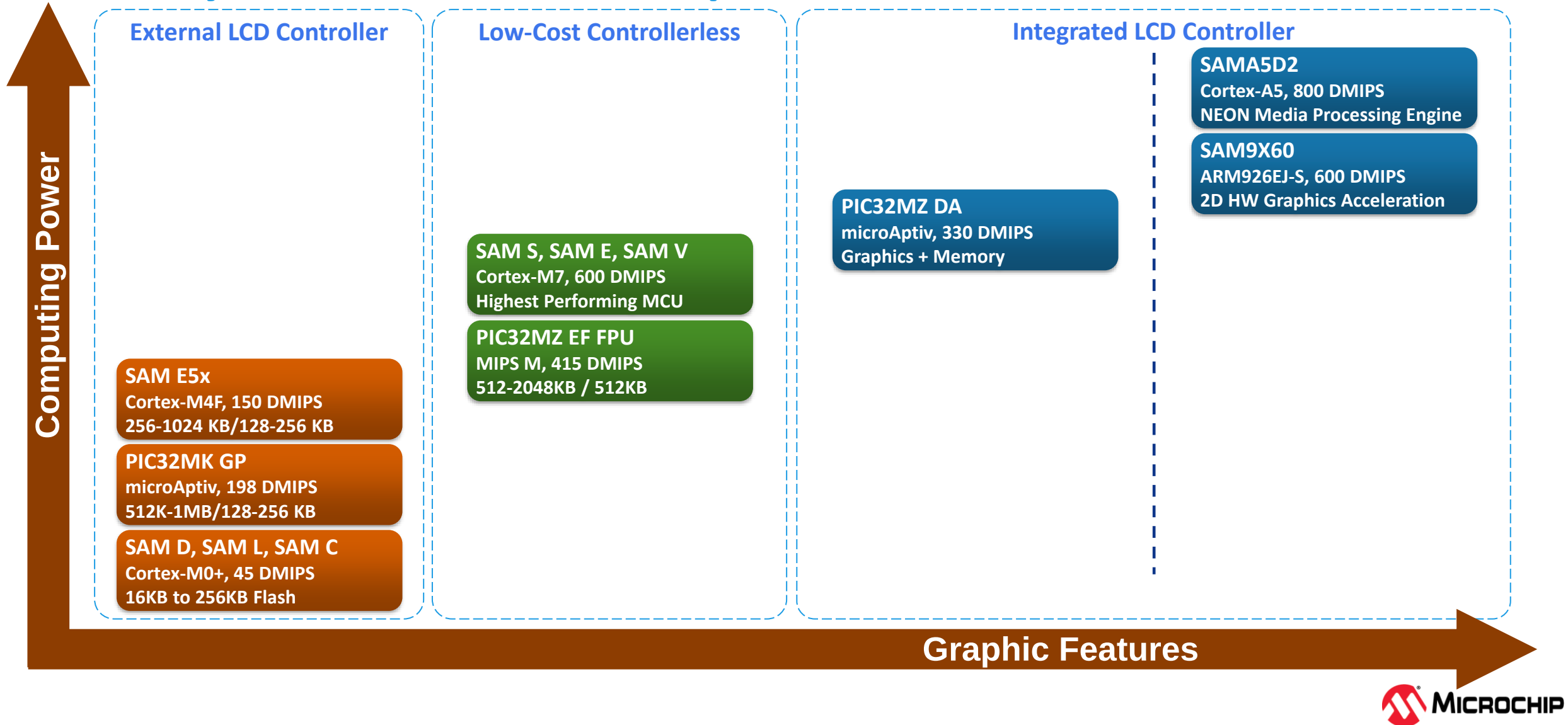
MPU w/ integrated LCD Controller



- ✓ Displays up to 1024 x 768
- ✓ Standard Audio/Video formats supported
- ✓ Camera capabilities
- ✓ Integrated LCD Controller w/ 2D HW Acceleration
- ✓ Linux or RTOS/bare-metal
- ✓ Up to 256MB LPDDR2 Integrated

32-bit MCU/MPU for Graphics Application

Industry's Broadest 32-bit Graphics Portfolio



32-bit MCU/MPU for Graphics Application

Scalable Support Across Platforms

Display

External LCD Controller



SAM E54
Cortex-M4



SAMD21
Cortex-M0+



Low-Cost Controllerless

PIC32MZ EF
200MHz MIPs



SAM E70
300MHz CM7



Integrated LCD Controller

PIC32MZDA
200MHz MIPs



SAM9X60
600MHz ARM926



SAMA5D2
500MHz Cortex-A



ENSEMBLE
GRAPHICS TOOLKIT



Ensemble Graphics Toolkit – Linux®

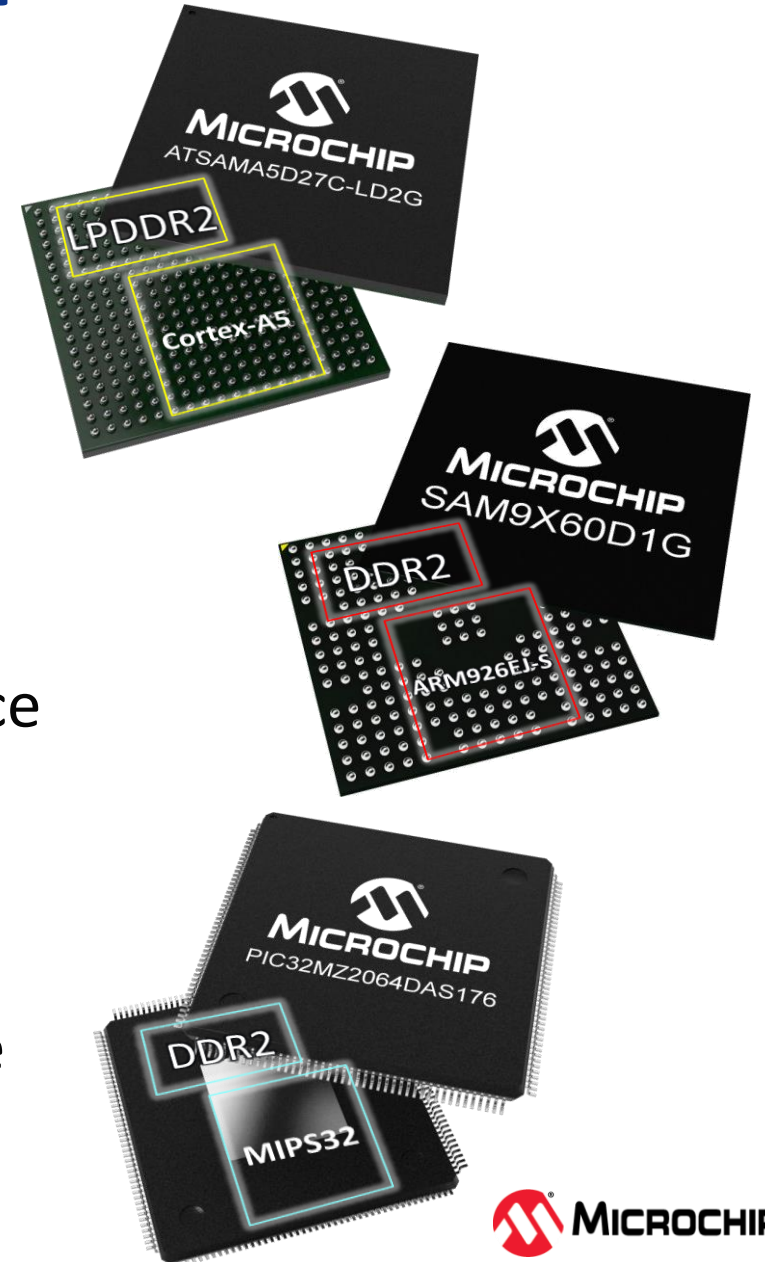
MPLAB® Harmony Graphics Suite – Bare-Metal / RTOS

Performance

32-bit MCU/MPU for Graphics Application

System in Package w/ Integrated DRAM

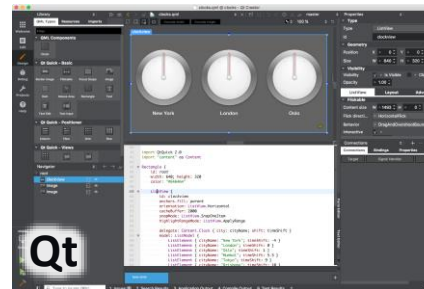
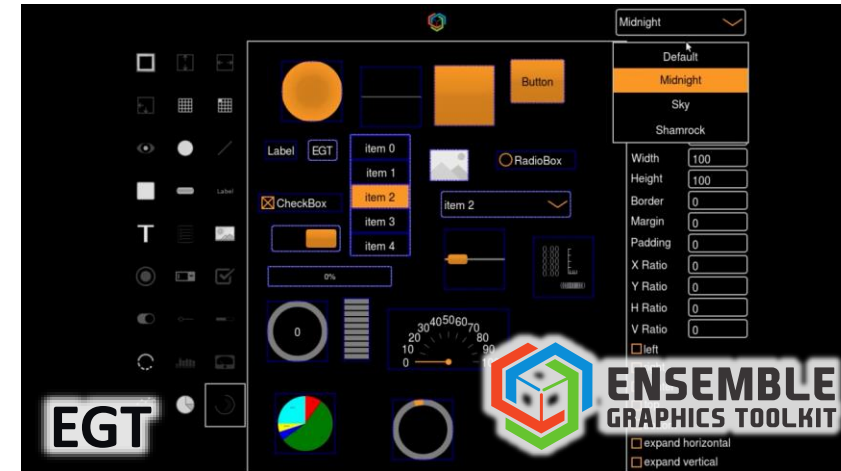
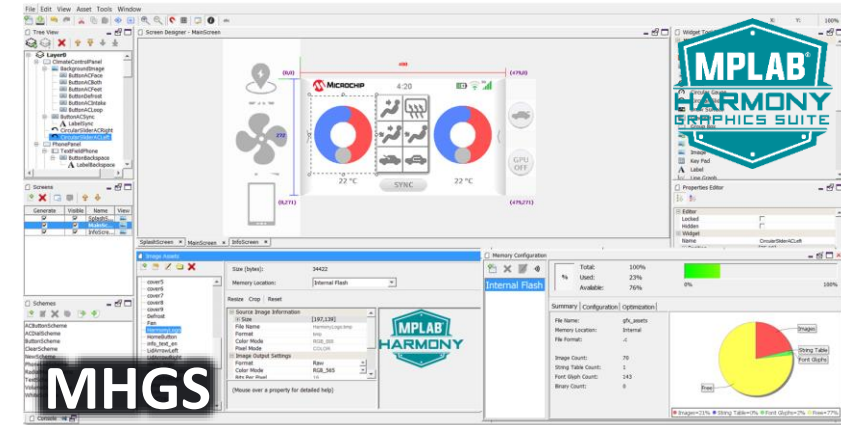
- **Larger On-Chip Display Buffer**
 - MIPS based MCUs w/ up to 32MB integrated DDR2
 - ARM MPUs w/ up to 128MB DDR2 or 256MB LPDDR2
 - Industrial grade (-40/+85C)
- **Dramatically reduced design complexity**
 - Smaller footprint and simplified PCB design
 - Lower-cost 4-layer PCB and Improved EMI performance
- **No DRAM supply issues!**
 - Single supplier for processor and DRAM
 - No EOLs or software updates to shipping product
 - Included in Microchip's customer-driven obsolescence



GUI Design Tools

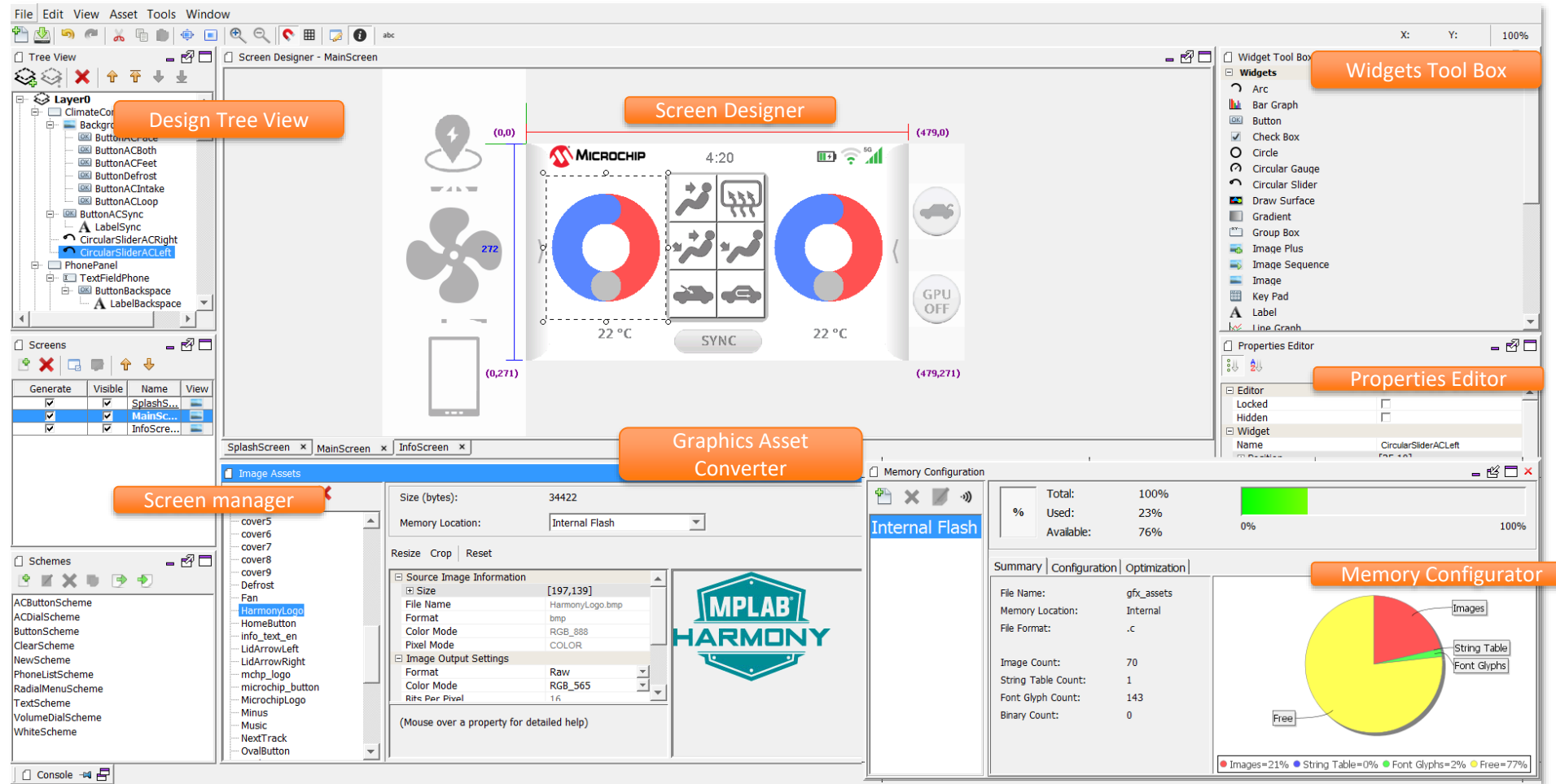
Scalable Graphics Development Software

- **Free! Developed and Supported by Microchip**
 - MPLAB Harmony Graphics Suite (MHGS)
 - Bare-metal and RTOS
 - MPLAB X IDE and MPLAB Harmony
 - Software Framework for Microchip MCUs and MPUs
 - Ensemble Graphics Toolkit (EGT) - Linux
 - Free, Open-Sourced
 - Linux Graphics Toolkit for Microchip MPUs
 - Complete Linux BSPs for Yocto, Buildroot and OpenWRT at www.linux4sam.com
- **3rd party solutions**
 - Qt - MPU
 - Crank - MPU
 - Embedded Wizard – MCU & MPU



Intuitive Design Tool

“What you see is what you get” Composition Canvas



Designer – GUI composition

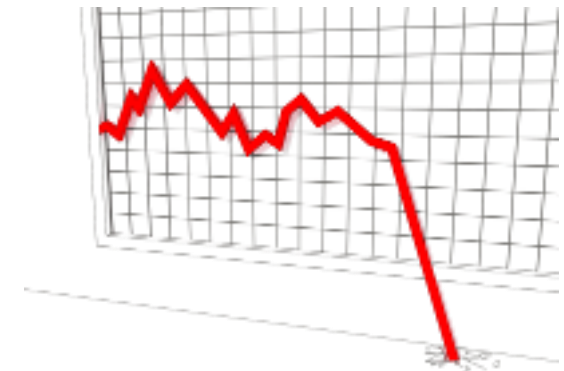
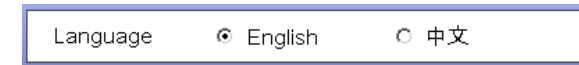
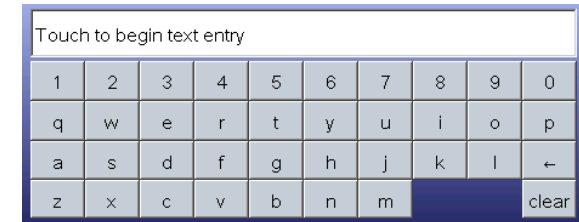
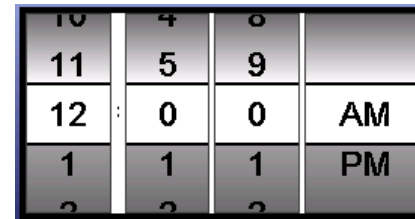


Complete Graphics Libraries

Modern graphical user interface designs



- Images (8-bit LUT, 16-bit, 32-bit color support)
- Text (Strings, Fonts, Multi-lingual Support)
- Touch and external event Handling
- Extended suite of widgets

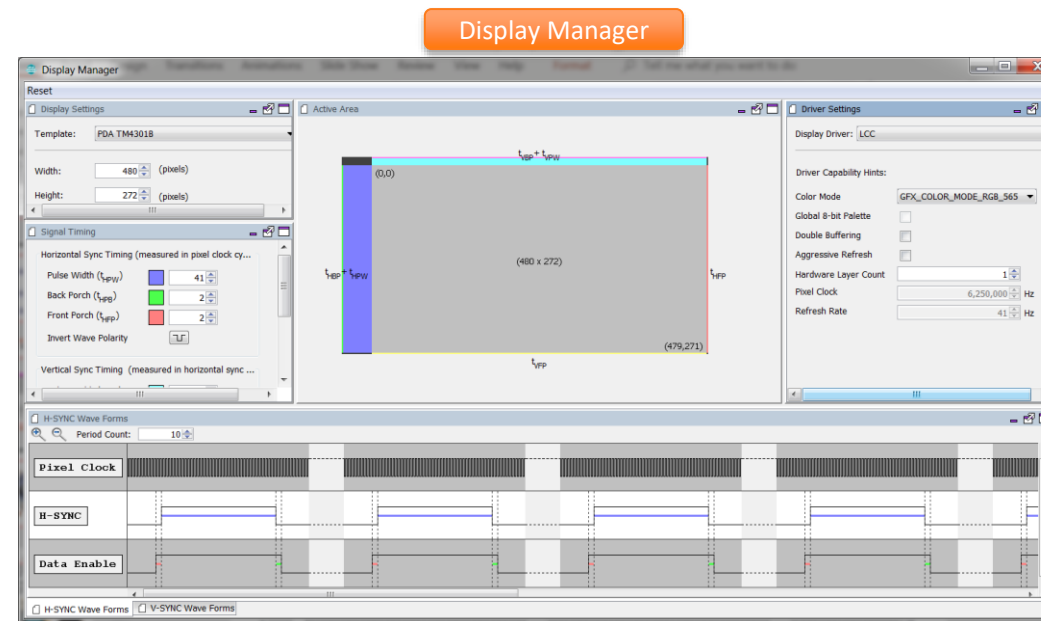
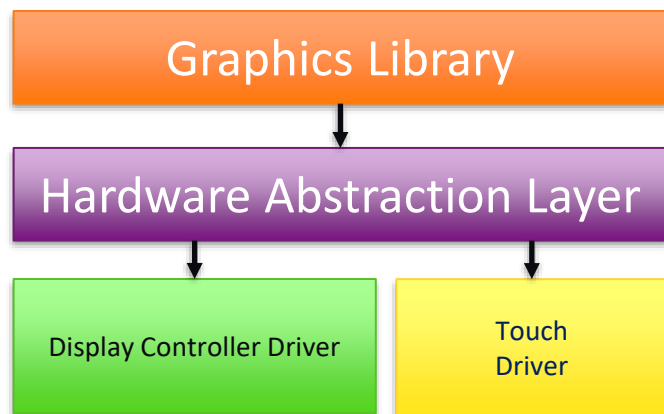


Create Drivers Without Handwritten Code

Drivers Hardware Abstraction Layer



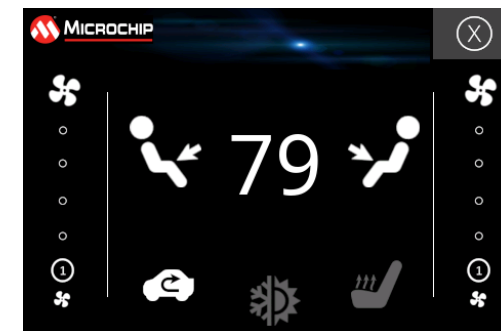
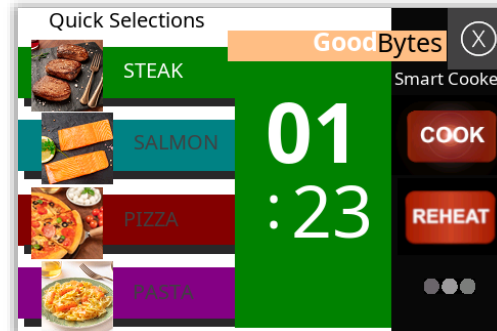
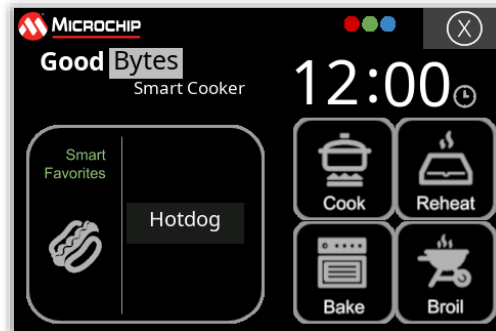
- Frame Buffer Management
- Generated Display Drivers (LCC, Internal and External Controllers)
- Display Timing configured in Display Manager
- Seamlessly supports 3rd Party Library



Target Markets and Applications

Mid-range Touchscreen Applications

- White Goods
- Small Appliances
- Vehicle Climate Control
- Industrial Control requiring CAN, Ethernet, or USB
- Vehicle Digital Cluster
- Motor Bike Gauge TFT Display
- Marine Gauge
- HMI Control Knob



PIC32MZ DA Brief

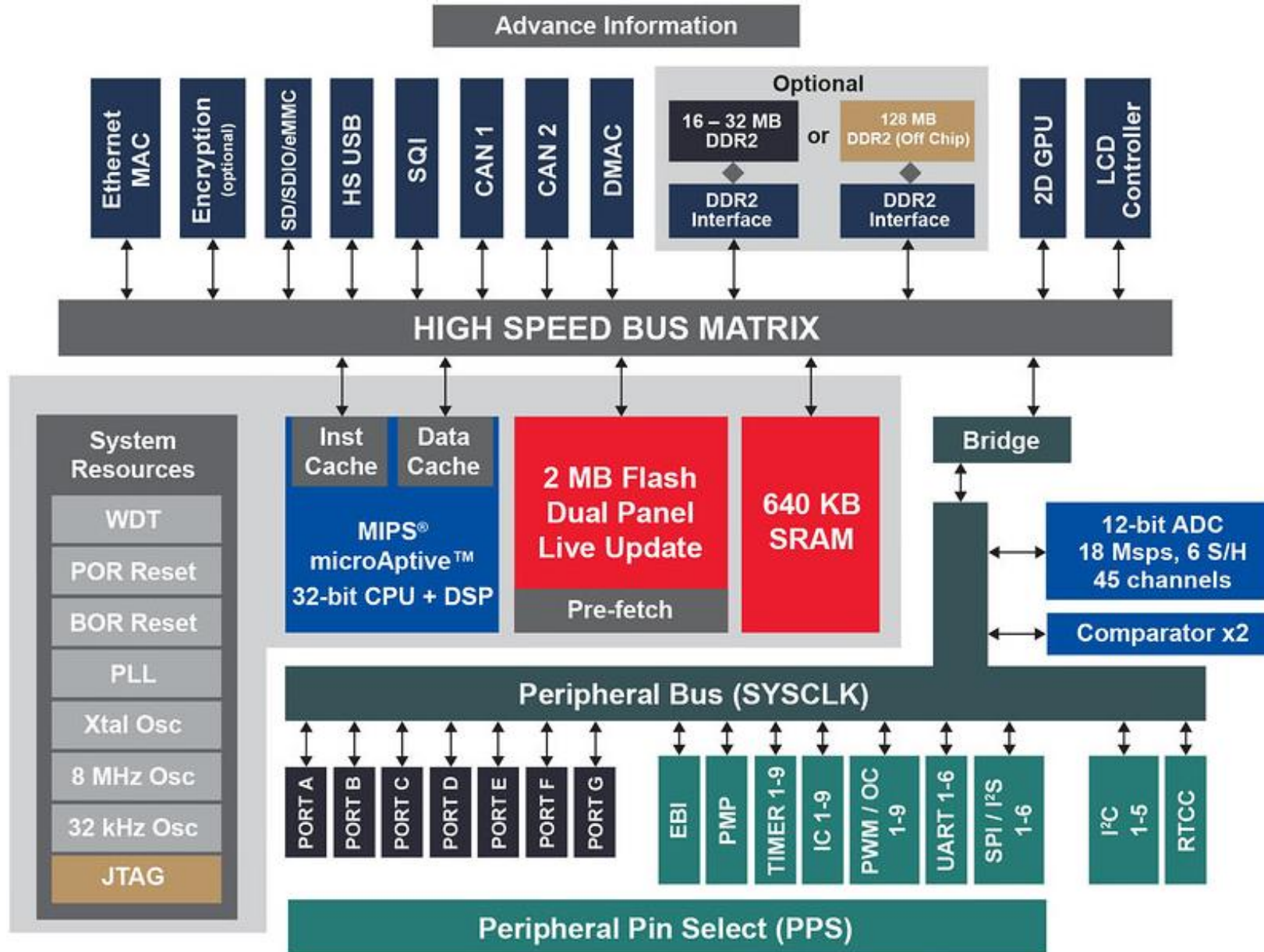
PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

PIC32MZ DA Key Features

- Core: 200 MHz/330 DMIPS MIPS32® microAptiv™
- 24-bit parallel interface supports displays up to WVGA (800 × 480)
- Three-layer graphics controller capable of driving 24-bit color WVGA
- High-performance 2D Graphics Processing Unit (GPU)
- 32 MB integrated DDR2 DRAM or 128 MB externally addressable
- Up to 2 MB Flash and 640 KB RAM
- 12-bit Analog-to-Digital Converter (ADC) throughput at 18 Msps
- Full-featured hardware crypto engine with Random Number Generator (RNG)

Boot Flash Memory (KB)	Remappable Peripherals						12-bit ADC Channels	Analog Comparators	CTMU	USB 2.0 HS OTG	I ² C	GLCD	GPU	EBI	PMP	SQI	SDHC	RTCC	Ethernet	I/O Pins	JTAG	Trace
	Remappable Pins	Timers ⁽¹⁾ /Capture/Compare	UART	SPI/I ² S	CAN 2.0B	External Interrupts ⁽²⁾																
160	47	9/9/9	6	6	2	5	45	2	Y	Y	5	Y	Y	Y	Y	Y	Y	Y	Y	120	Y	Y

PIC32MZ DA Block Diagram



PIC32MZ DA Family

Devices	Program Memory (KB)	Data Memory (KB)	DDR2 Controller Interface (Internal)	DDR2 SDRAM Size (MB)	Crypto/RNG	DMA Channels (Programmable/ Dedicated)	Package			
PIC32MZ1025DAK169	1024	256	No	1	N	8/24	HF			
PIC32MZ1025DAL169					Y	8/26				
PIC32MZ1064DAK169		640			N	8/24				
PIC32MZ1064DAL169					Y	8/26				
PIC32MZ2025DAK169	2048	256			Yes	32		N	8/24	6JX
PIC32MZ2025DAL169								Y	8/26	
PIC32MZ2064DAK169		640						N	8/24	
PIC32MZ2064DAL169								Y	8/26	
PIC32MZ1025DAR169	1024	256					N	8/24		
PIC32MZ1025DAS169							Y	8/26		
PIC32MZ1064DAR169		640					N	8/24		
PIC32MZ1064DAS169							Y	8/26		
PIC32MZ2025DAR169	2048	256					N	8/24		
PIC32MZ2025DAS169							Y	8/26		
PIC32MZ2064DAR169		640					N	8/24		
PIC32MZ2064DAS169							Y	8/26		

169-PIN LFBGA

Devices	Program Memory (KB)	Data Memory (KB)	DDR2 Controller Interface (Internal)	DDR2 SDRAM Size (MB)	Crypto/RNG	DMA Channels (Programmable/ Dedicated)	Package		
PIC32MZ1025DAK176	1024	256	No	—	N	8/24	2J		
PIC32MZ1025DAL176					Y	8/26			
PIC32MZ1064DAK176		640			N	8/24			
PIC32MZ1064DAL176					Y	8/26			
PIC32MZ2025DAK176	2048	256			N	8/24			
PIC32MZ2025DAL176					Y	8/26			
PIC32MZ2064DAK176		640			N	8/24			
PIC32MZ2064DAL176					Y	8/26			
PIC32MZ1025DAR176	1024	256	Yes	32	N	8/24			
PIC32MZ1025DAS176					Y	8/26			
PIC32MZ1064DAR176		640			N	8/24			
PIC32MZ1064DAS176					Y	8/26			
PIC32MZ2025DAR176	2048	256			N	8/24			
PIC32MZ2025DAS176					Y	8/26			
PIC32MZ2064DAR176		640			N	8/24			
PIC32MZ2064DAS176					Y	8/26			

176-PIN LQFP

PIC32MZ DA Radial Graphics Development Board Introduction

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

PIC32MZ DA Radial Graphics Development Board

- **Microchip Direct Part No.**
 - EV50P56A
- **PIC32MZ2064DAR176 Graphics MCU**
 - 2MB Flash; 640K SRAM; 32MB DDR2 DRAM
- **WINC1510 Wi-Fi Module**
 - 802.11 b/g/n; OTA firmware upgradeable
- **SAMR30M18A Mi-Wi Module**
 - A Cortex M0+ MCU + Sub-GHz Transceiver in a single package
- **On-Board Analog and Linear Product Solutions**
 - ATA6563 CAN FD Transceiver
 - MCP2221 USB to I2C/UART converter IC
 - MIC2289 white LED driver IC
 - MIC45116, MIC5528 Power Management ICs
 - MCP9808 Temperature Sensor IC
 - DSC612 MEMS CLK IC
- **Optional LCD Modules**
 - Radial Display Module from Data Image.
 - Industrial grade support
 - 3" round display 432*432, 1000 nits , part number : LCF0300633GGU00
 - Rectangular Display Module from Promate.
 - MCHP maXTouch touch panel
 - 4.3" LCD display 480*272, 420 nits , part number: 97P043C1A0F-PM



PIC32MZ DA Radial Graphics Development Board

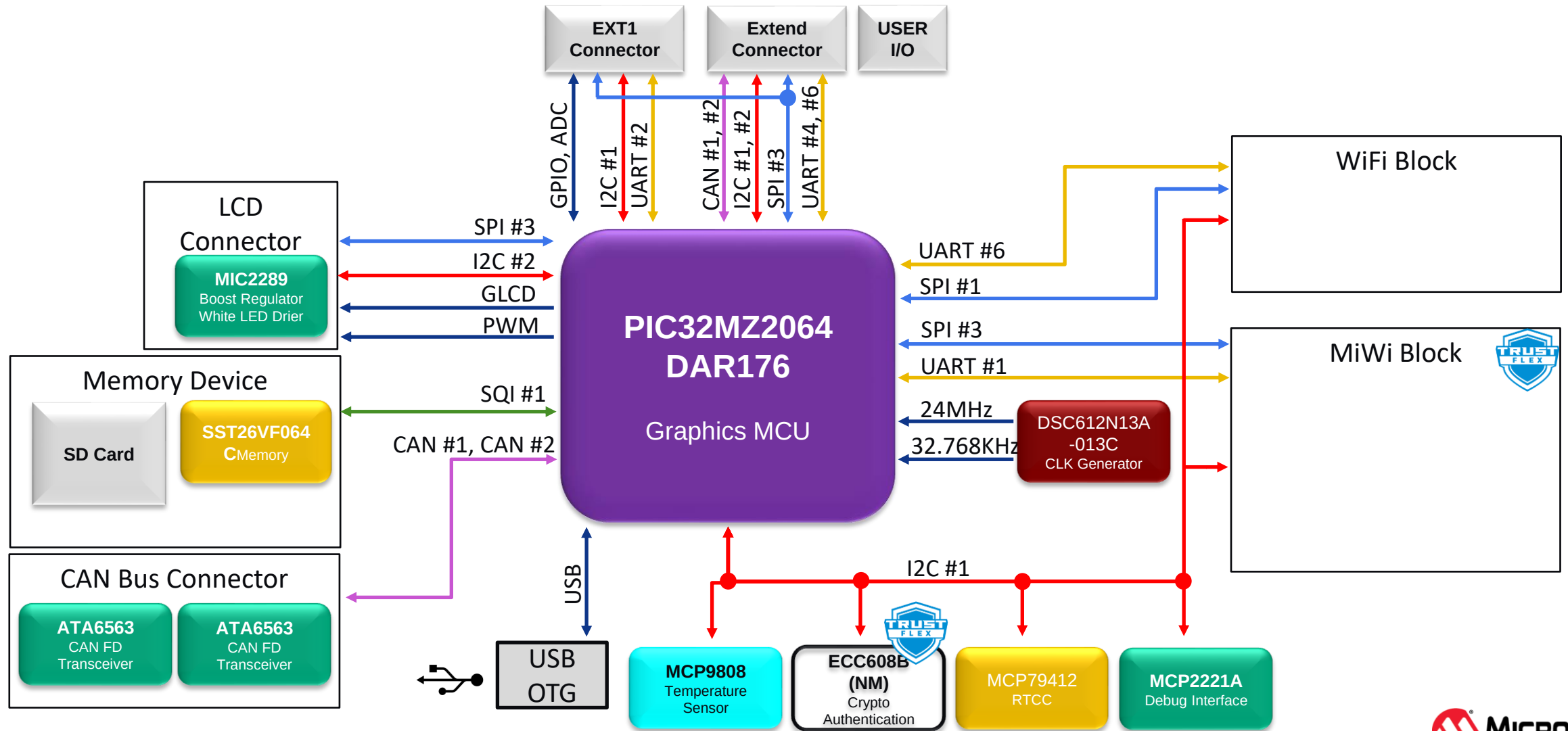
Unique Features

- Round display and touchscreen support for a stylish/modern product user interface and form-factor
- On-board rotary knob for intuitive user input
- On-board wireless and HW security modules for secure IoT and Cloud Connectivity
- On-board CAN Transceiver for vehicle display applications



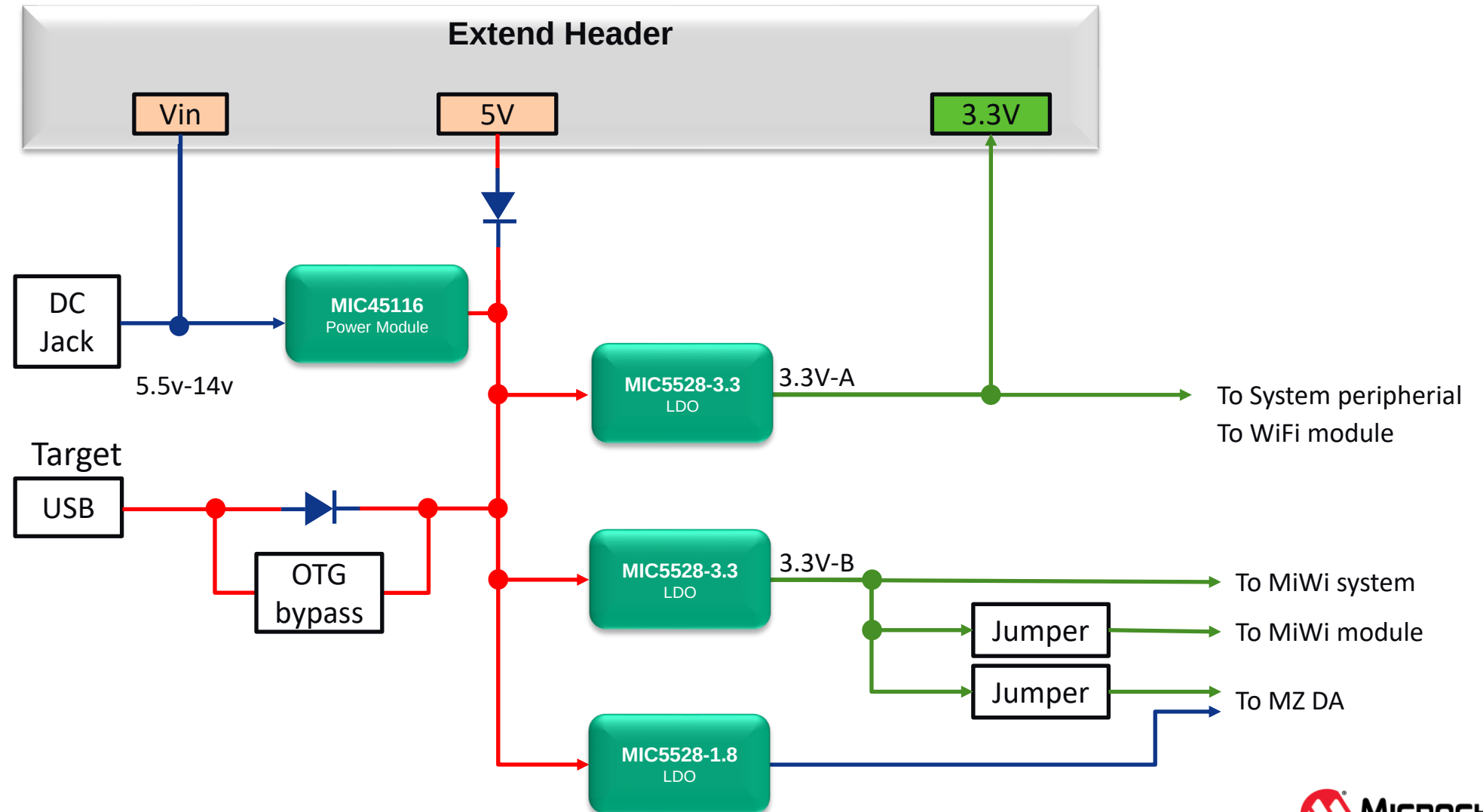
PIC32MZ DA Radial Graphics Development Board

System Block Diagram – Microchip Direct ver.



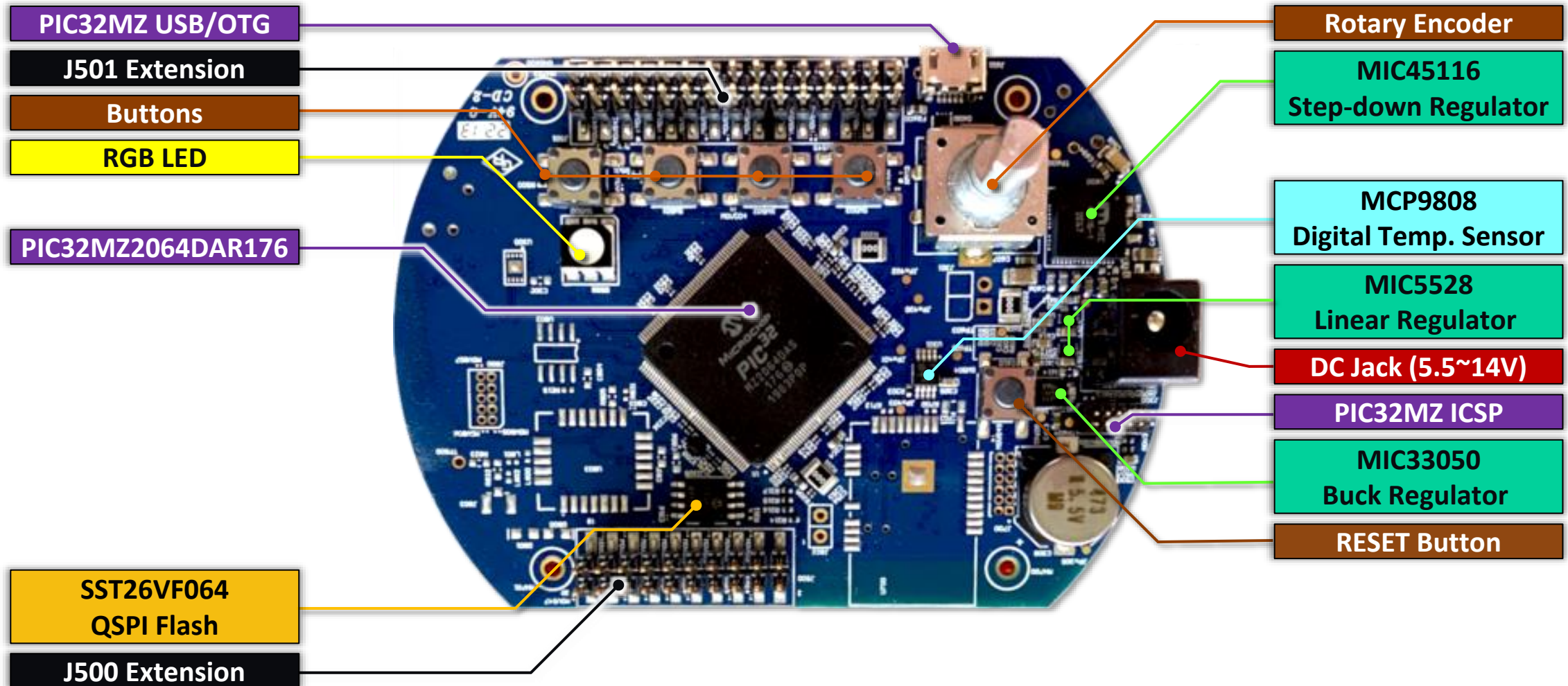
PIC32MZ DA Radial Graphics Development Board

Power Tree Block – Microchip Direct ver.



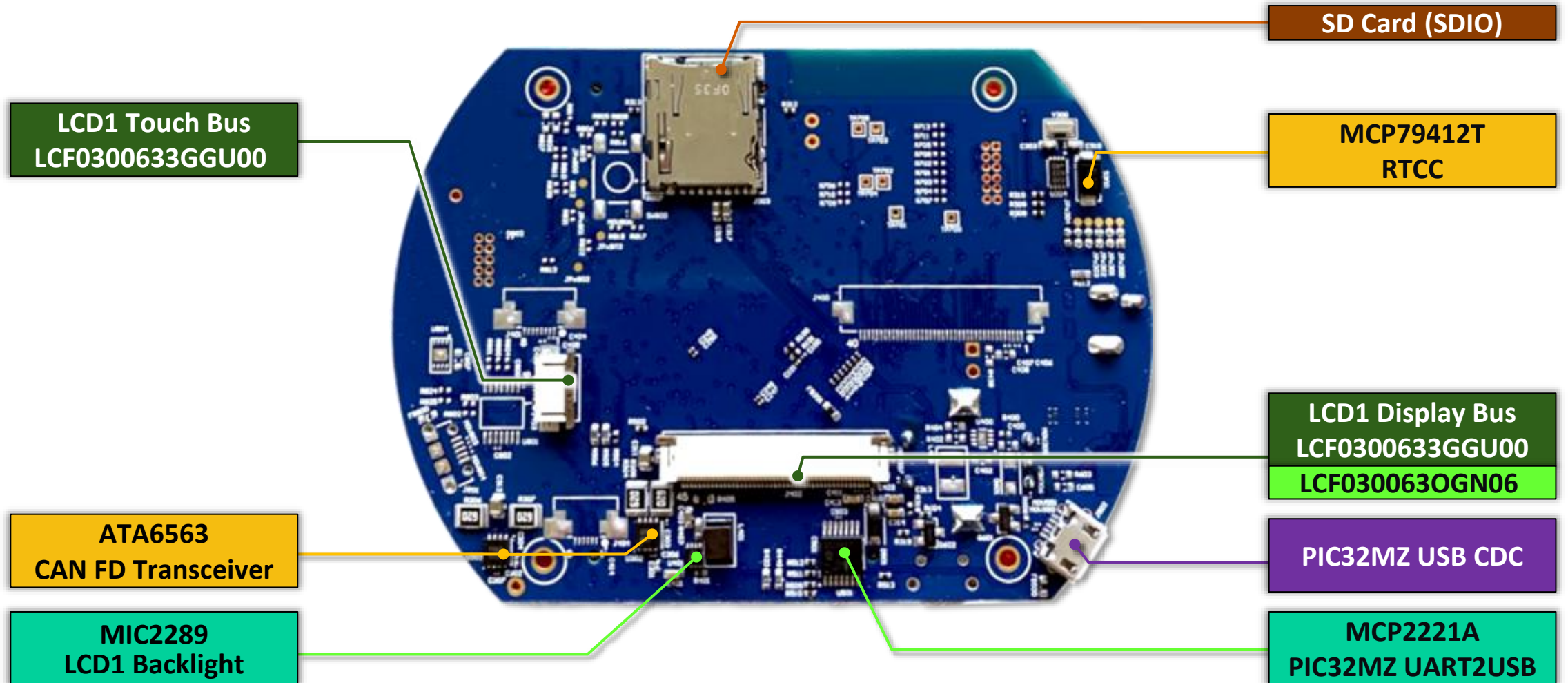
PIC32MZ DA Radial Graphics Development Board

Placement (Top) – Microchip Direct ver.



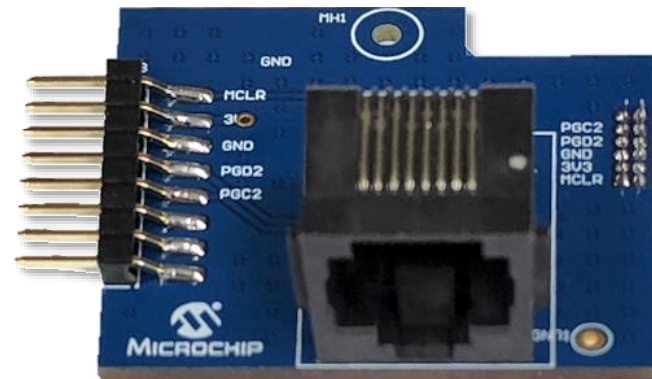
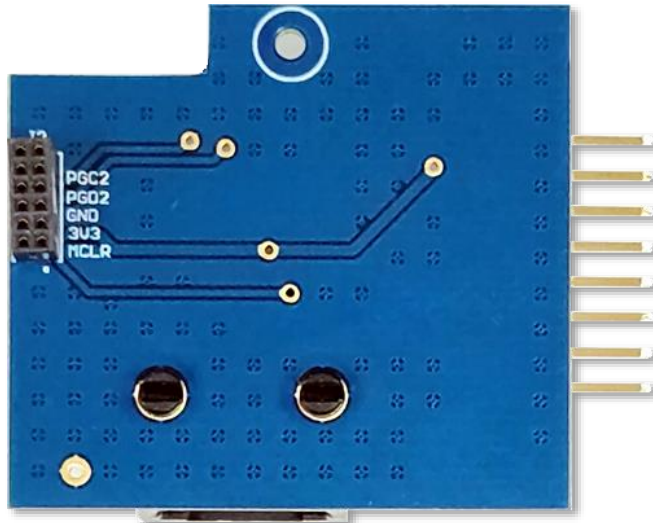
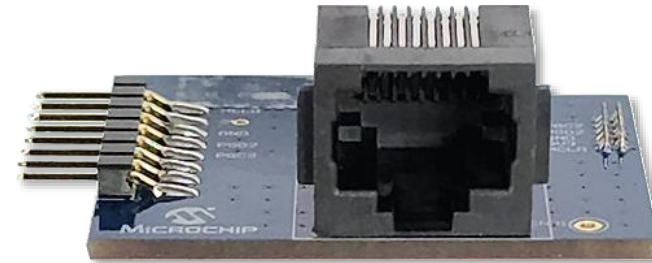
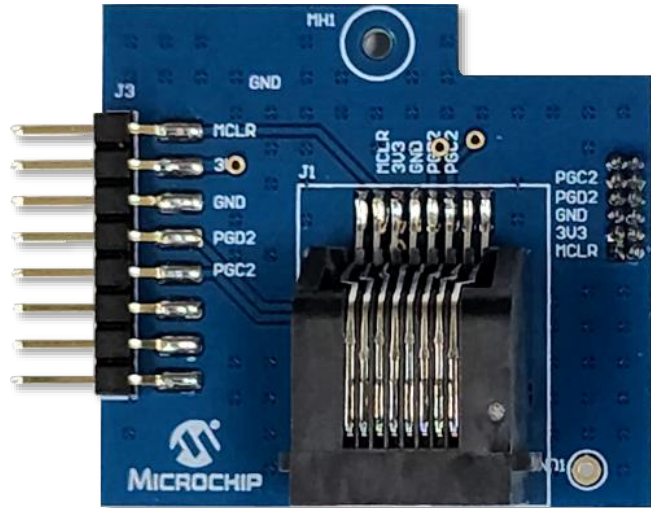
PIC32MZ DA Radial Graphics Development Board

Placement (Bottom) – Microchip Direct ver.



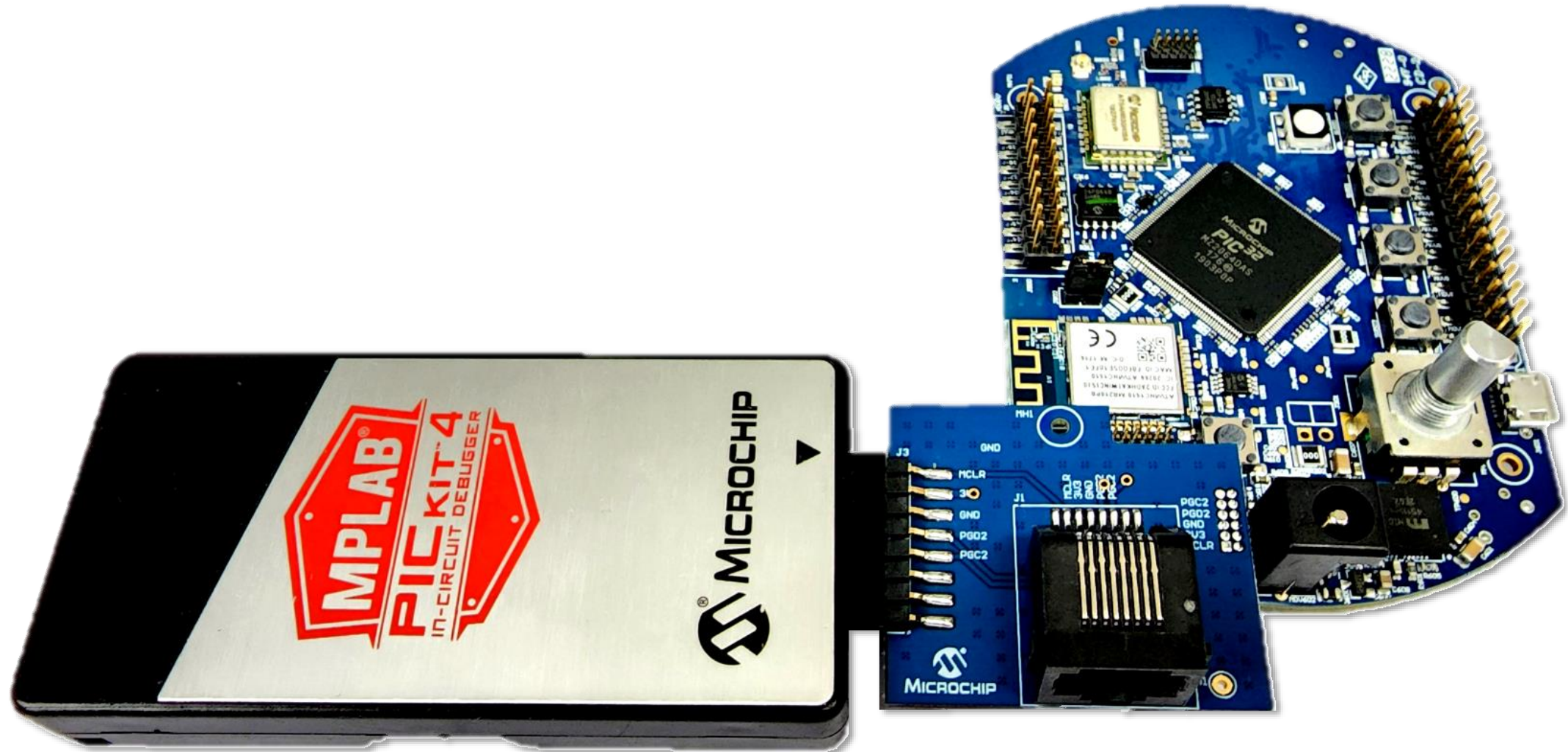
PIC32MZ DA Radial Graphics Development Board

ICSP Adapter Board



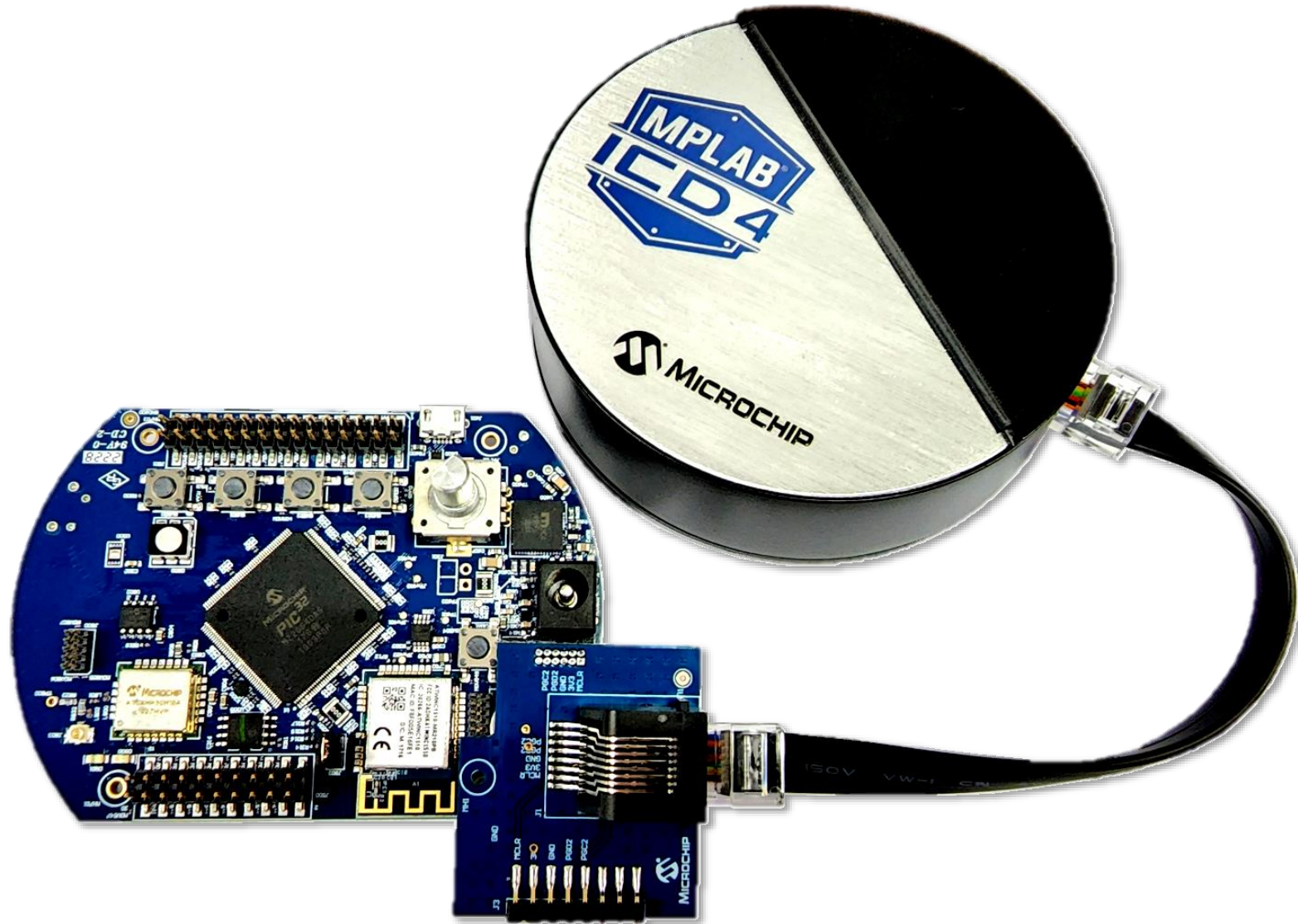
PIC32MZ DA Radial Graphics Development Board

Debugger Connection



PIC32MZ DA Radial Graphics Development Board

Debugger Connection

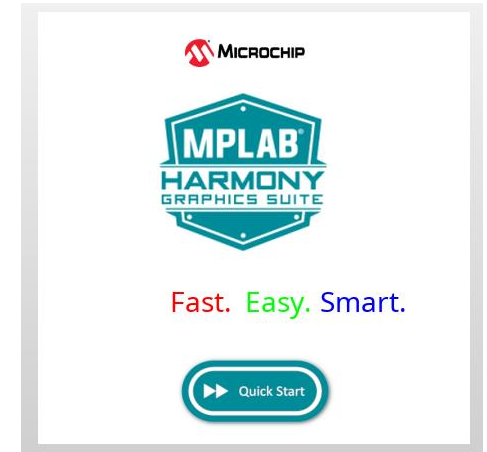
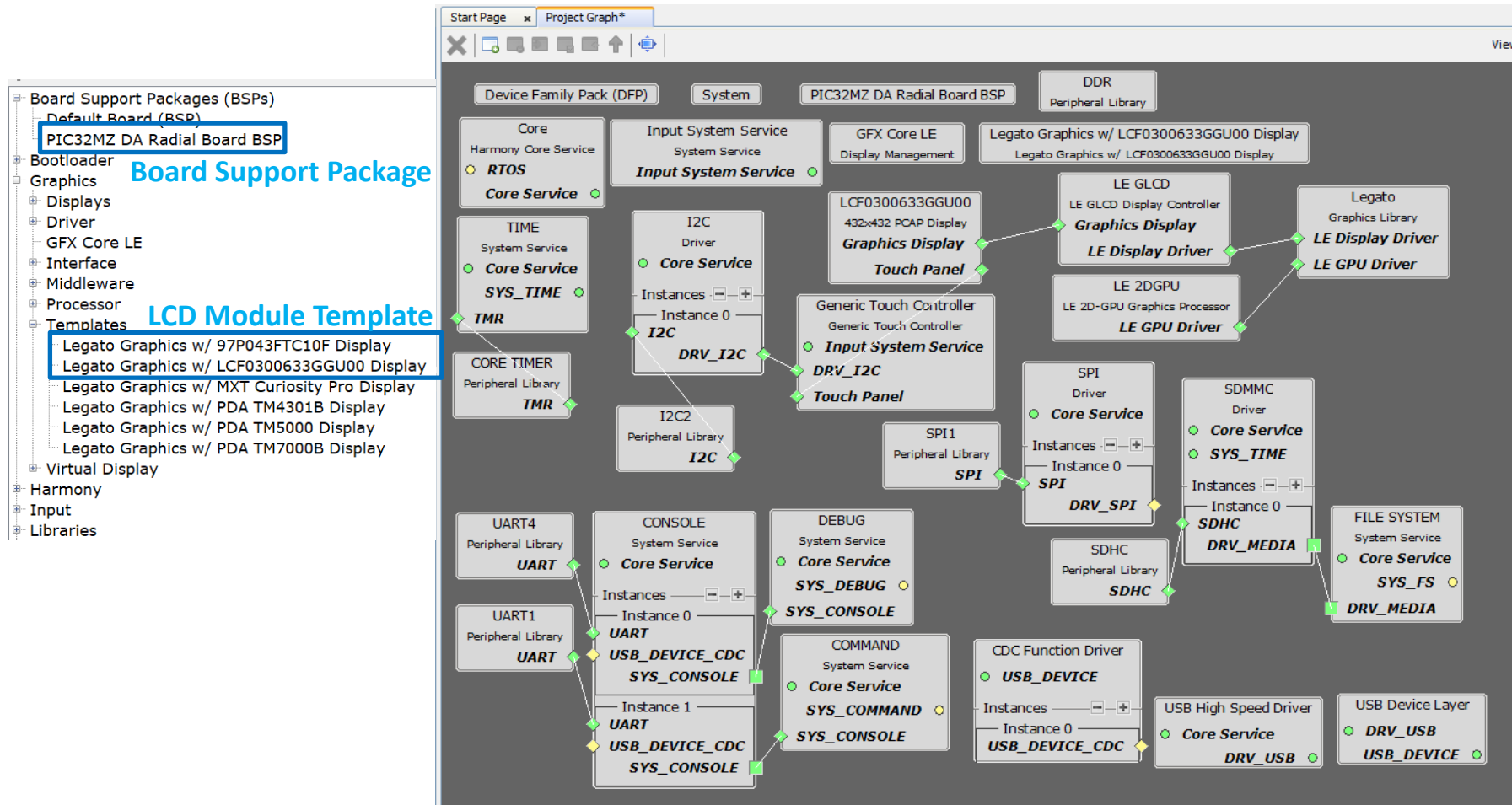


Demo Applications

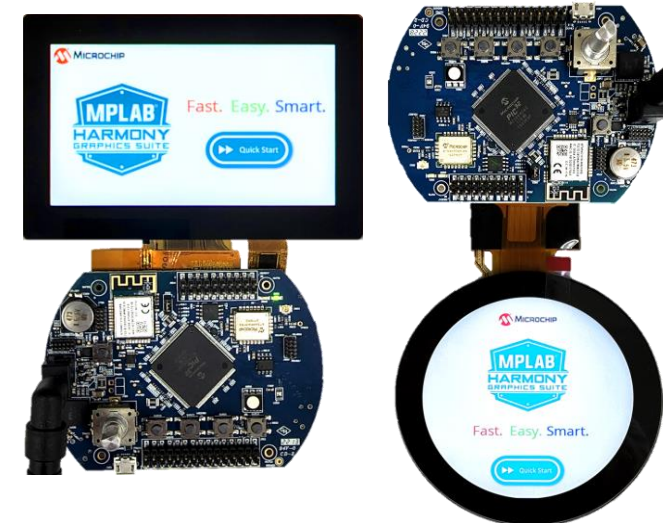
PIC32MZ DA Radial Graphics Development Board

Quick Start Project and Templets

MPLAB Harmony Configurator (MHC)



Quick Start Demo Project



PIC32MZ DA Radial Graphics Development Board

Dashboard Application Start-up

- Application boots to a splash screen and then to the RPM gauge screen
- Backlight dims and brightens between screen transitions



Splash Screen



RPM Gauge Screen

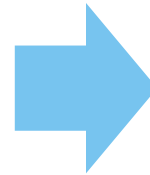
PIC32MZ DA Radial Graphics Development Board

Showcase Application Start-up

- Application boots to a splash screen and then to the Main Menu screen
- Backlight dims and brightens between screen transitions
- Brief animation introduces the Main Menu screen



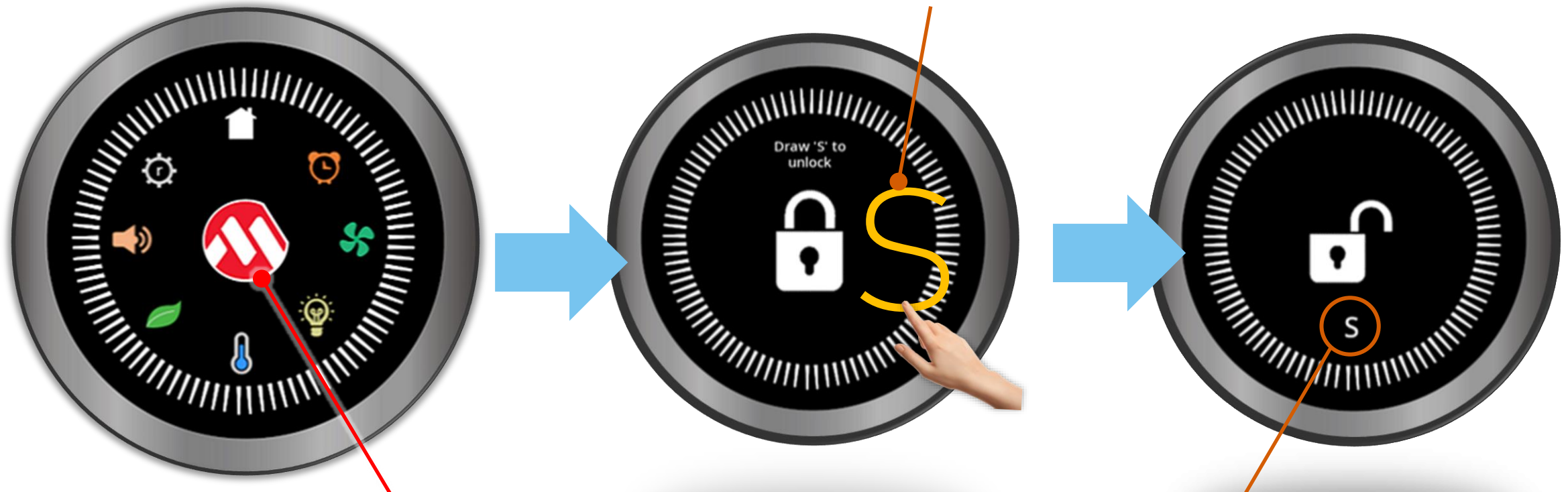
Splash Screen



Main Menu Screen

PIC32MZ DA Radial Graphics Development Board

Lock/Unlock Screen with Motion Gestures* Advanced Gesture Recognition



**Touch and Hold Microchip Logo
for 2 seconds to Lock the Screen**

* Usage outside this demo requires a Software License Agreement
with Motion Gestures (www.motiongestures.com)

PIC32MZ DA Radial Graphics Development Board

Demo Video



Software Preparation

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Software Preparation

- **MPLAB® X IDE v6.0** <https://www.microchip.com/mplab/mplab-x-ide>
- **MPLAB® XC32 v4.10** <https://www.microchip.com/mplab/compilers>
- **MPLAB® Harmony 3 Launcher v3.6.4** (From X IDE plugin)
- **MPLAB® Harmony Framework** (From X IDE Harmony 3 Content Manager)
 - bsp v3.15.0
 - core v3.11.1
 - csp v3.15.0
 - mhc v3.8.5
 - dev_packs v3.15.1
 - usb v3.10.0
 - gfx v3.12.2
- **PIC32MZ-DA_DFP v1.4.118 (Device Firmware Pack)** (From X IDE \Tools\Packs)

(Lasted Update : 2023/02/13)

Design Reference

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Harmony Graphic Design Reference

- **MPLAB® Harmony 3 Graphics Help**
<https://microchip-mplab-harmony.github.io/gfx/>
- **MPLAB® Harmony 3 Graphics Wiki**
<https://github.com/Microchip-MPLAB-Harmony/gfx/wiki>
- **MPLAB® Harmony 3 Graphics Videos**
<https://www.youtube.com/playlist?list=PL9B4edd-p2ag5xslIHhja-caKYY7AKPxe>
- **MPLAB® Harmony Graphics Suite Applications**
https://microchip-mplab-harmony.github.io/gfx_apps/frames.html?frmname=topic&frmfile=index.html
- **Graphics Libraries Help (Aria Graphic Library)**
<https://www.microchip.com/content/dam/mchp/documents/parked-documents/Graphics%20Libraries%20Help%20v2.06.pdf>
- **MPLAB® Harmony Graphics Composer User's Guide (Aria Graphic Composer)**
<https://www.microchip.com/content/dam/mchp/documents/parked-documents/MPLAB%20Harmony%20Graphics%20Composer%20Users%20Guide%20v2.06.pdf>

Legato Essential Design Reference

- **MPLAB® Harmony Graphics Suite Legato Wiki**
<https://github.com/mchpgfx/legato.docs/wiki>
- **Graphic Application Fundamentals**
<https://github.com/mchpgfx/legato.docs/wiki/Application-Essentials>
- **Legato Graphics Composer User Guide**
<https://github.com/mchpgfx/legato.docs/wiki/UserGuide-LegatoComposer>
- **Understanding Legato Composer**
<https://github.com/mchpgfx/legato.docs/wiki/Understanding-Composer>
- **Legato Composer Tutorials**
<https://github.com/mchpgfx/legato.docs/wiki/Legato-Composer-Tutorials>
- **Legato Graphic Library On-Line document**
<https://mchpgfx.github.io/legato.docs/html/index.html>
- **Legato Graphic Library Off-Line document**
<https://github.com/mchpgfx/legato.docs/archive/master.zip>
- **MPLAB® Discover (Legato + PIC32 MZ DA)**
<https://mplab-discover.microchip.com/v1?s0=legato&s1=%22pic32mz+da%22>
- **QuickStart PIC32MZ DA Curiosity Development Kit**
<https://github.com/mchpgfx/legato.docs/wiki/QuickStart-PIC32MZ-DA-Curiosity-Development-Kit>

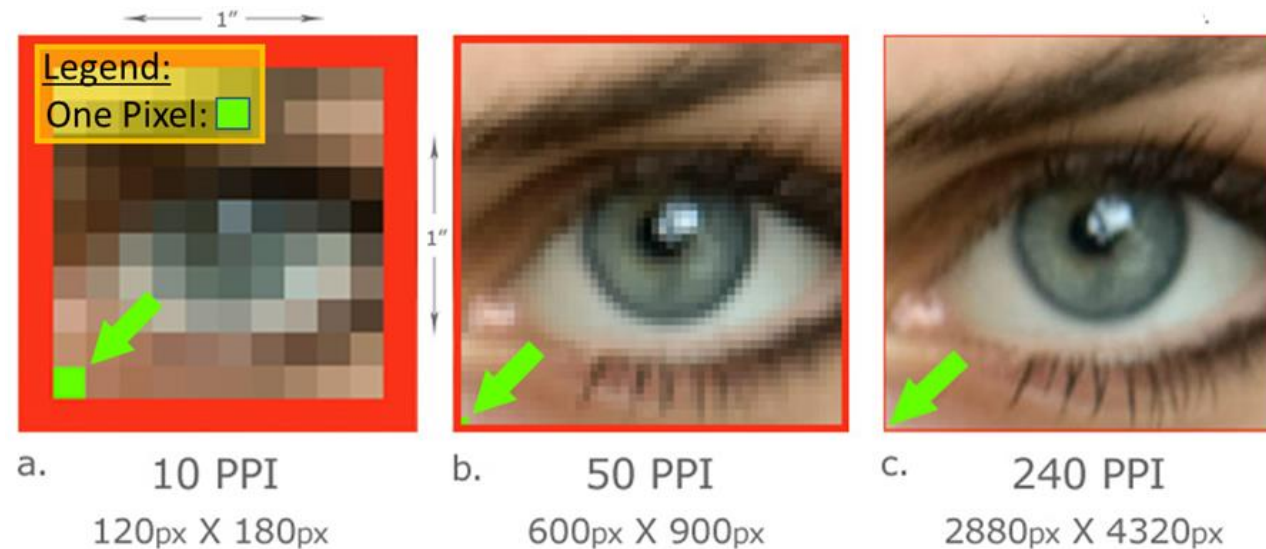
Graphics Basics

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Graphics Basics

- **Pixel:**

- A pixel is a physical point in a raster image, or the smallest addressable element in an all-points addressable display device. The number of Pixels Per Inch (PPI) is a metric for how finely the image is displayed.



- **Primitive:**

- A primitive is a basic building block for drawing to the screen:
e.g., point, line, arc, circle, bar.

- **Widget:**

- A widget is a graphical object that is used to convey information or provide user input to the system. Widgets are created by assembling multiple primitives.

Graphics Basics

- **Font:**

- A set of type or characters of one style, weight and size for a given typeface, for example:

Arial-bold

新細明體

Courier New

標楷體

Time New Roman

微軟正黑體

- **Glyph:**

- A pixel representation of an individual character in a given typeface.

The Glyph types of character 'A' and Chinese '龍' [tà]

A (Arial-Blod 28pt)

A (Courier New 32pt)

A (Times New Roman 28pt)

龍龍 (新細明體 36pt)

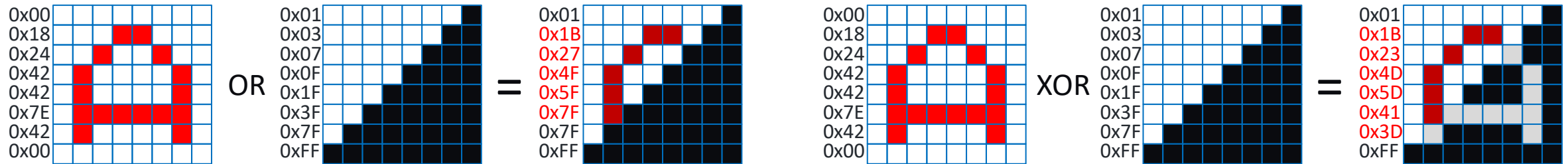
龍龍 (標楷體 36pt)

龍龍 (正黑體 36pt)

Graphics Basics

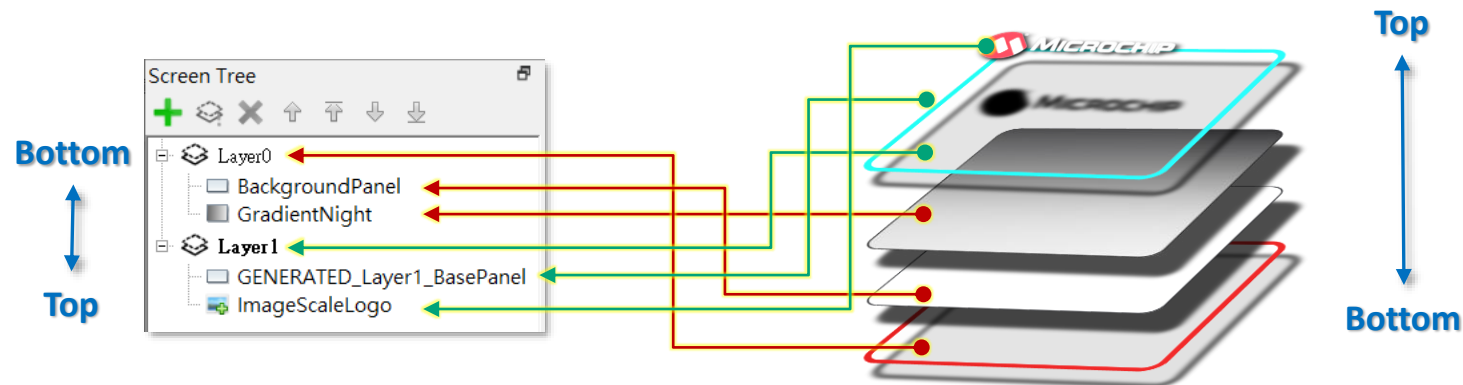
- **Bit Block Transfer (Bit BLIT):**

- A data operation that combines two or more bitmaps using Boolean functions. An example of a BLIT would be adding an animated object (sprite) on top of a background image while ignoring the background (black) pixels in the sprite's bitmap.



- **Layer:**

- The objects of a graphics design can be organized into a hierarchy with child objects which inherit properties from a parent object, which can be a child of another parent object higher in the hierarchy.



Graphics Basics

- **Sprite:**

- A small two-dimensional bit map that is added to a larger scene. Typically, sprites are used to simulate movement by quickly erasing a prior drawing and redrawing the sprite in a slightly different position.



- **Display Resolution:**

- The number of distinct pixels in each dimension that can be displayed. It is usually quoted in width x height in units of pixels.

	QVGA	VGA	WVGA	SVGA	XGA	WXGA	SXGA	HD	Full HD	2K	4K
Width	320	640	800	800	1024	1280	1280	1280	1920	2048	4096
Height	240	480	480	600	768	800	1024	720	1080	1080	2160
Pixels	76,800	307,200	384,000	480,000	786,432	1,024,000	1,310,720	921,600	2,073,600	2,211,840	8,847,360

Graphics Basics

- **Alpha Blending:**

- A graphical technique which blends pixels of overlapping widgets together instead of overwriting one with another. Alpha blending ranges from fully transparent to fully visible (blocking all beneath).



- **Color Depth and RGB Color Model :**

- The number of bits required to represent the color of a single pixel. It is expressed as **bits per pixel (bpp)**. Typical bits per pixel are 8, 16, 24, and 32 bpp.
- The RGB Color Model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. Colors are expressed as a triplet (RGB) (Source: Wikipedia).

R R R R R R R R R G G G G G G G G G B B B B B B B B B A A A A A A A A A RGBA 8888 (32 bpp, R/G/B: 8 bits, Alpha: 8 bits)

R R R R R R R R R G G G G G G G G G B B B B B B B B B RGB 888 (24 bpp, R: 8 bits, G: 8 bits, B: 8 bits)

R R R R R G G G G G G G B B B B B B RGB 565 (16 bpp, R: 5 bits, G: 6 bits, B: 5 bits)

Y Y Y Y CbCb CrCr YCbCr 422 (8 bpp, Y: 4 bits, Cb: 2 bits, Cr: 2 bits)

Graphics Basics

- **Frame Buffer:**

- The block of memory that stores the pixel data to be transferred to the screen. A frame buffer can be thought of as a copy of the display's pixel values.

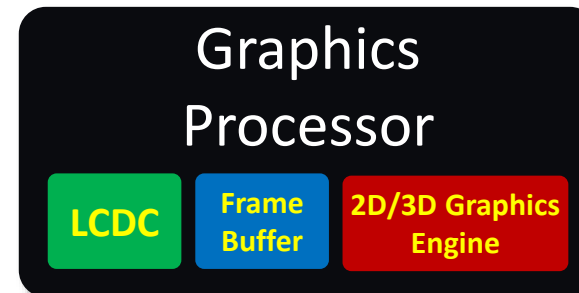
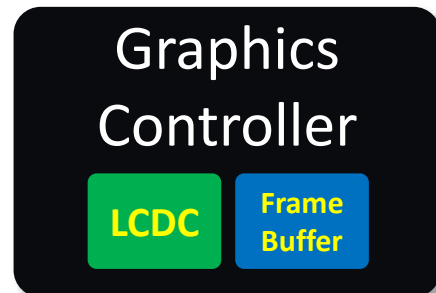


- **Graphics Controller:**

- The peripheral of a processor or separate device responsible for driving the pixel data to the display.

- **Graphics Processor:**

- Dedicated hardware within a microcontroller designed to accelerate updating the frame buffer for primitives such as lines and rectangles and for bit block transfers.



Graphics Basics

- **Question :**

How many full frames could be stored in RGDB DDR2 with color model RGBA8888 ?

- Hint : RGDB round LCD resolution is Width 432 and Height 432.
- Hint : PIC32MZ2064DAR176 has 32MByte DDR2 SDRAM inside.

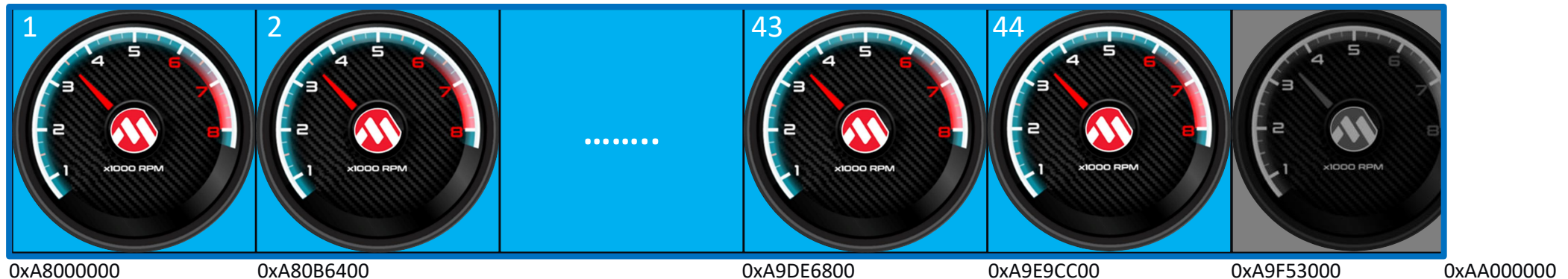
Total Pixels : $432 \times 432 = 186,624$ pixels

Pixel color depth : $\text{RGBA8888} = 8+8+8+8 = 32$ bpp (bits per pixel)

Frame Buffer size : $(432 \times 432 \times 32) / 8 = 746,496$ bytes (729 KB)

Stored full frames : $32\text{MB} / 729\text{KB} = 44$ Full Frames

DDR2 SDRAM Memory (32MBytes)

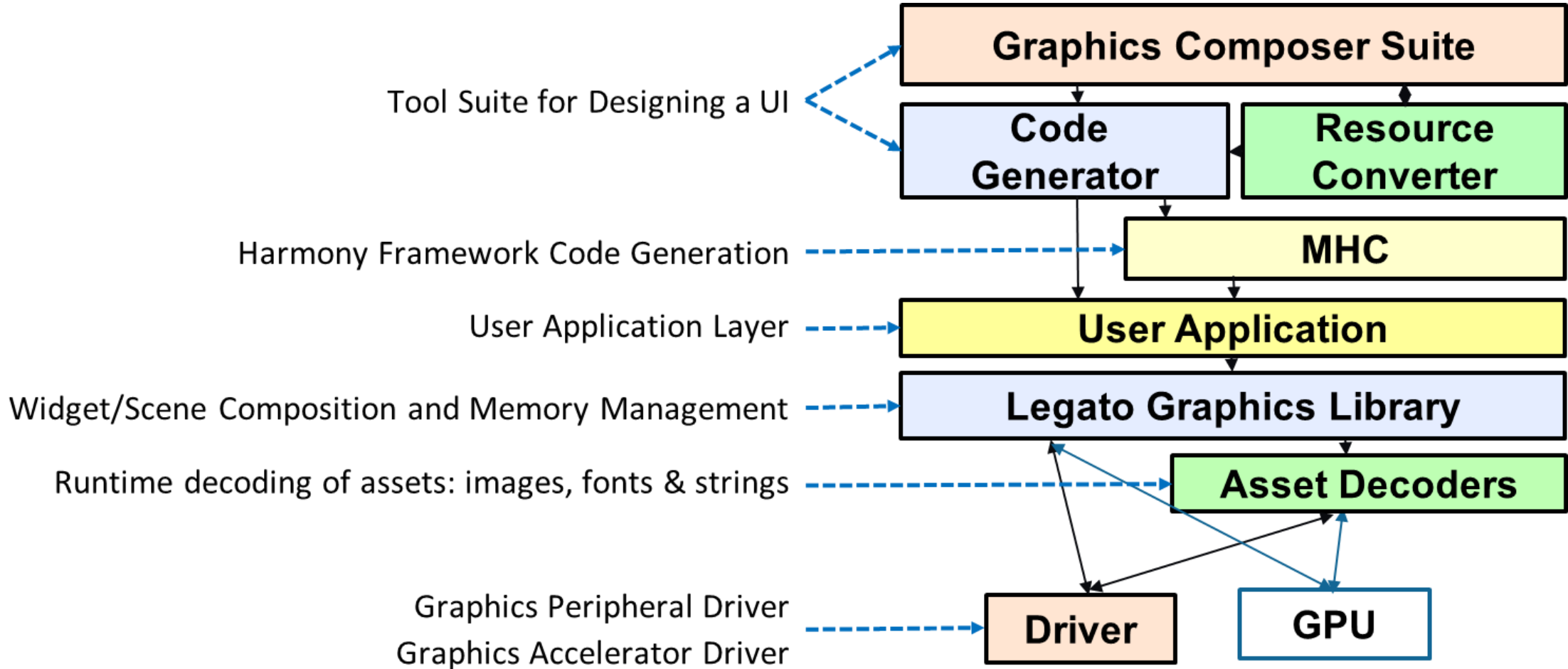


MPLAB® Harmony Graphics Suite and Legato Graphic Library

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

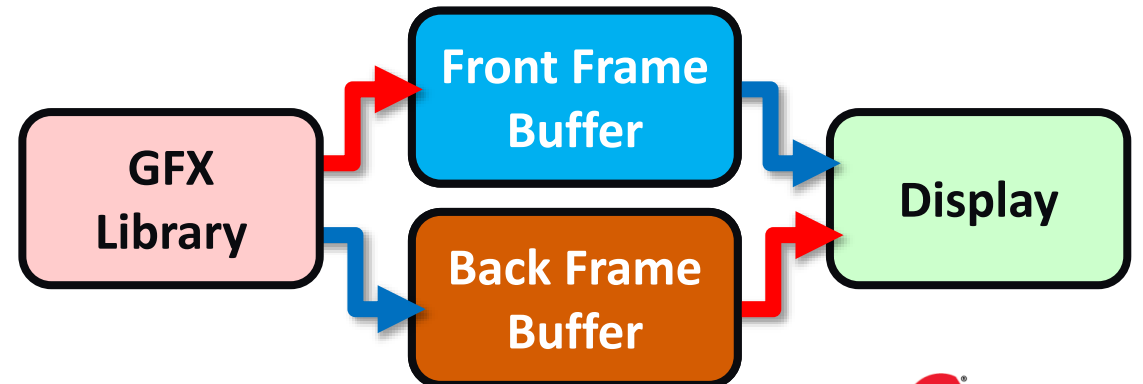
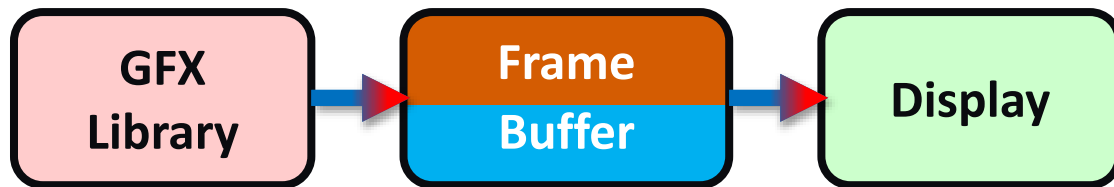
MPLAB Harmony Graphics Suite

- MPLAB® Harmony Graphics Suite is free to design with and use with our 32-bit microcontrollers.



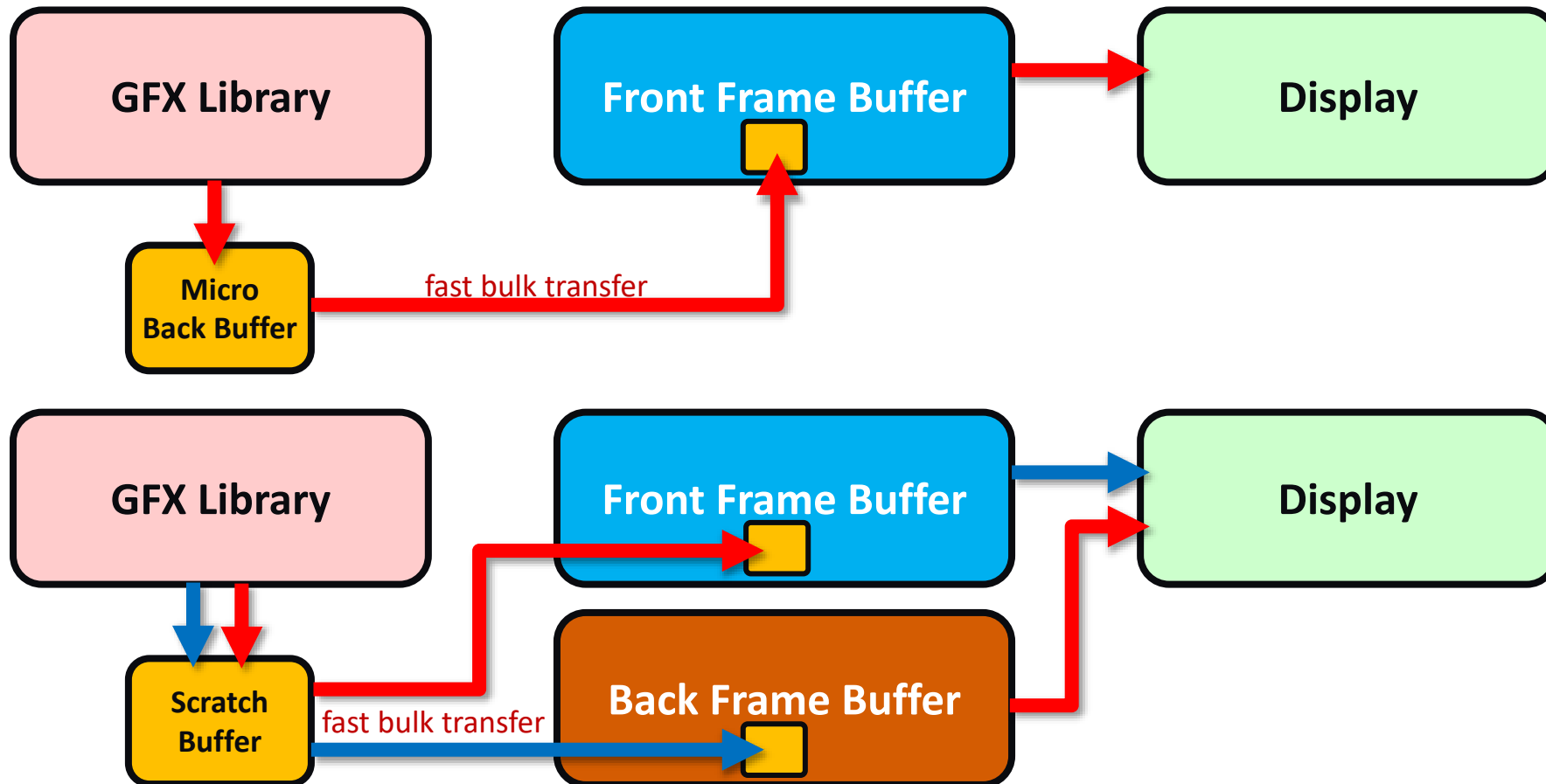
Double Buffering

- Screen tearing can be eliminated by using two frame buffers. This scheme doubles the memory requirement.



Double Buffering - Legato

- To help minimize screen tearing, Legato's draw algorithms paint to a tiny scratch/back buffer. The scratch/back buffer is then copied into the frame buffer using fast bulk transfer techniques.



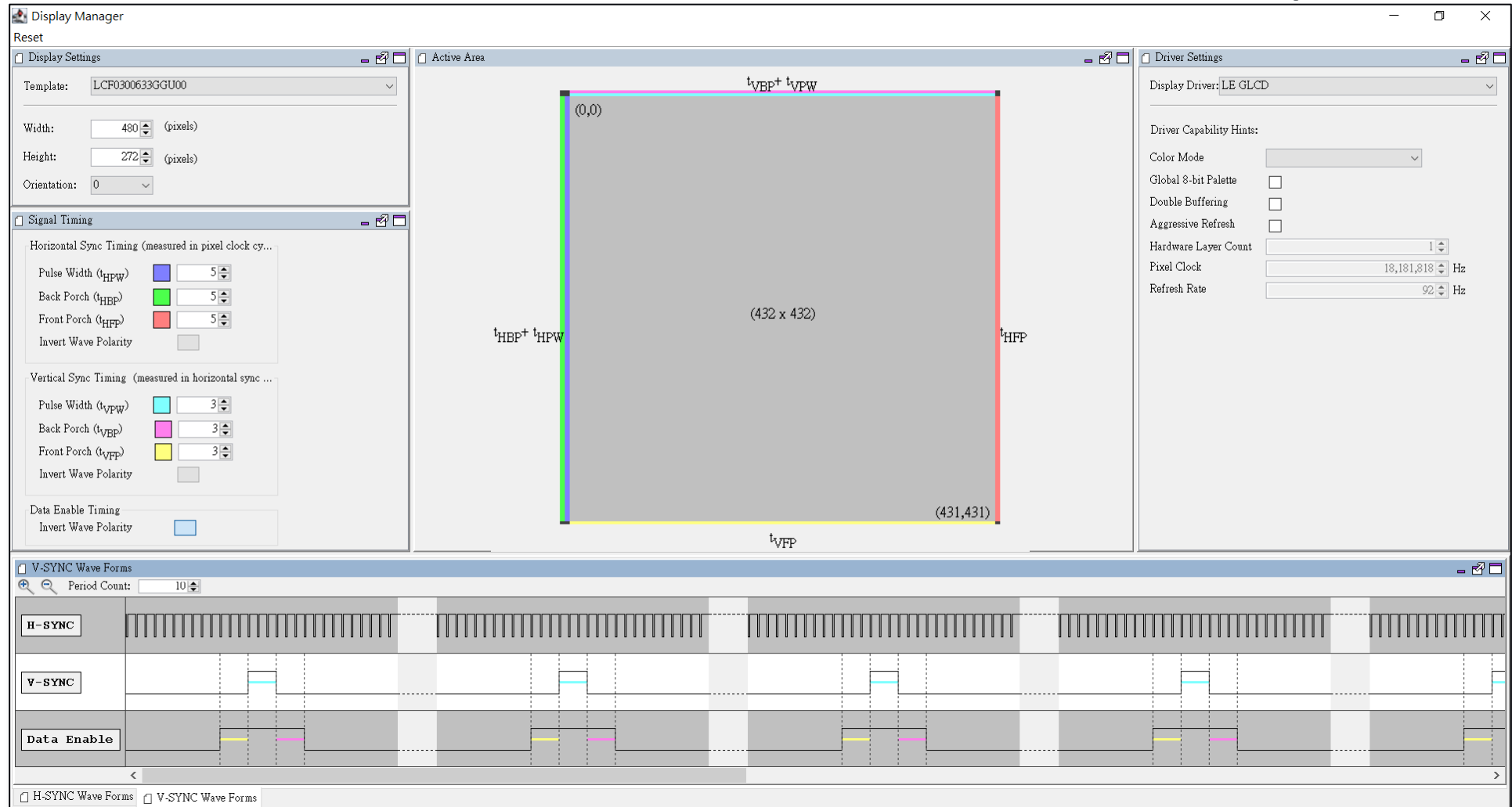
- This course will not involve it, we will use RGDDB template.

- This course will not involve it, we will use RGDDB template.



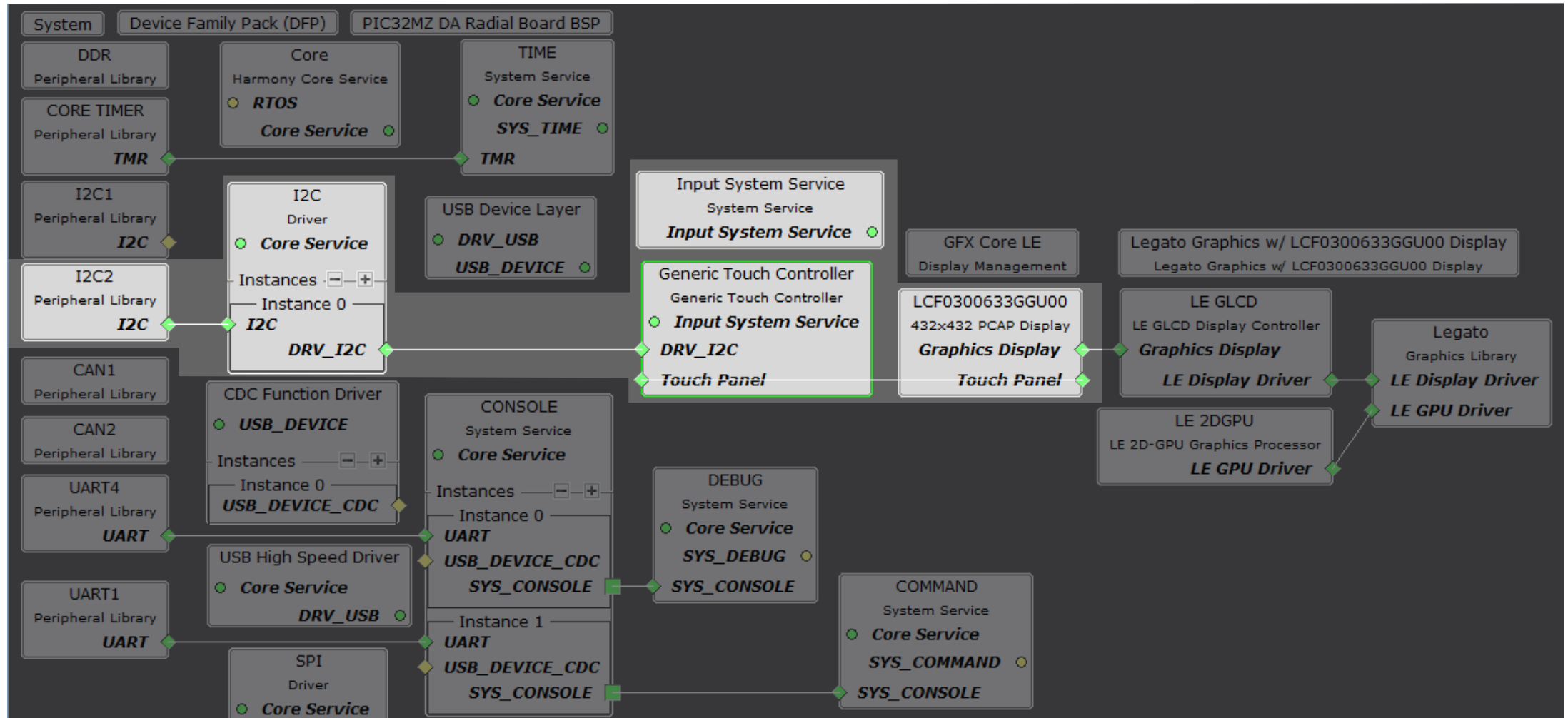
Display Manager

- This course will not involve it, we will use RGDDB template.



Touch Driver and Input System Service

- This course will not involve it, we will use RGDB template.

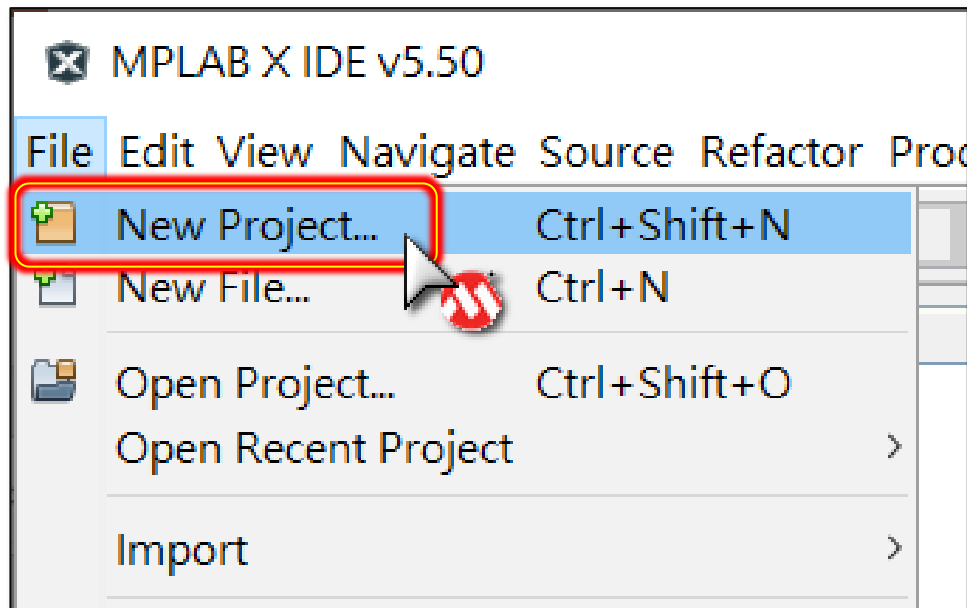


A New Harmony Project

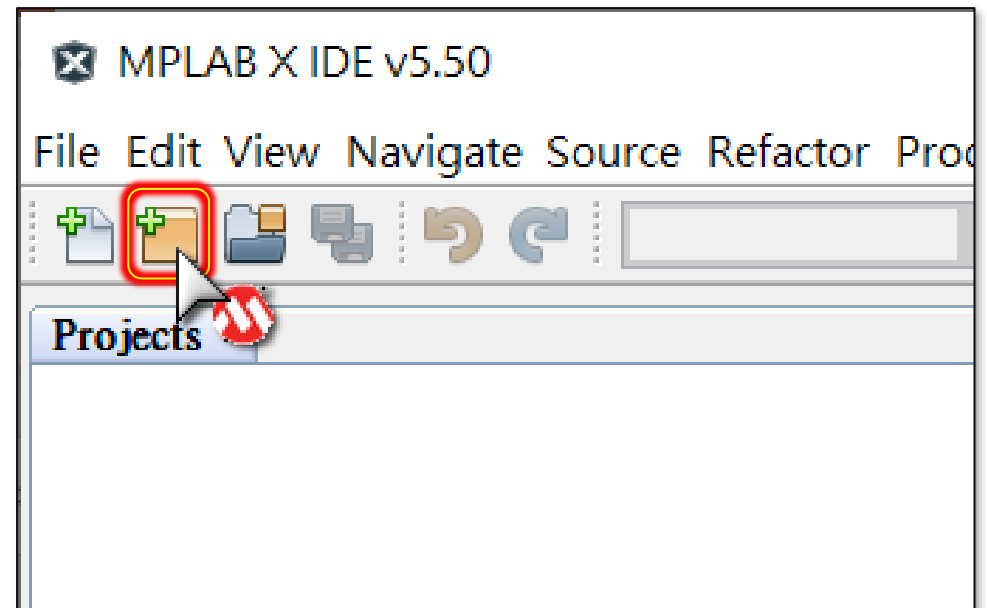
PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

A New Harmony Project

- Create a new 32 bits Harmony Project for PIC32MZ-DA

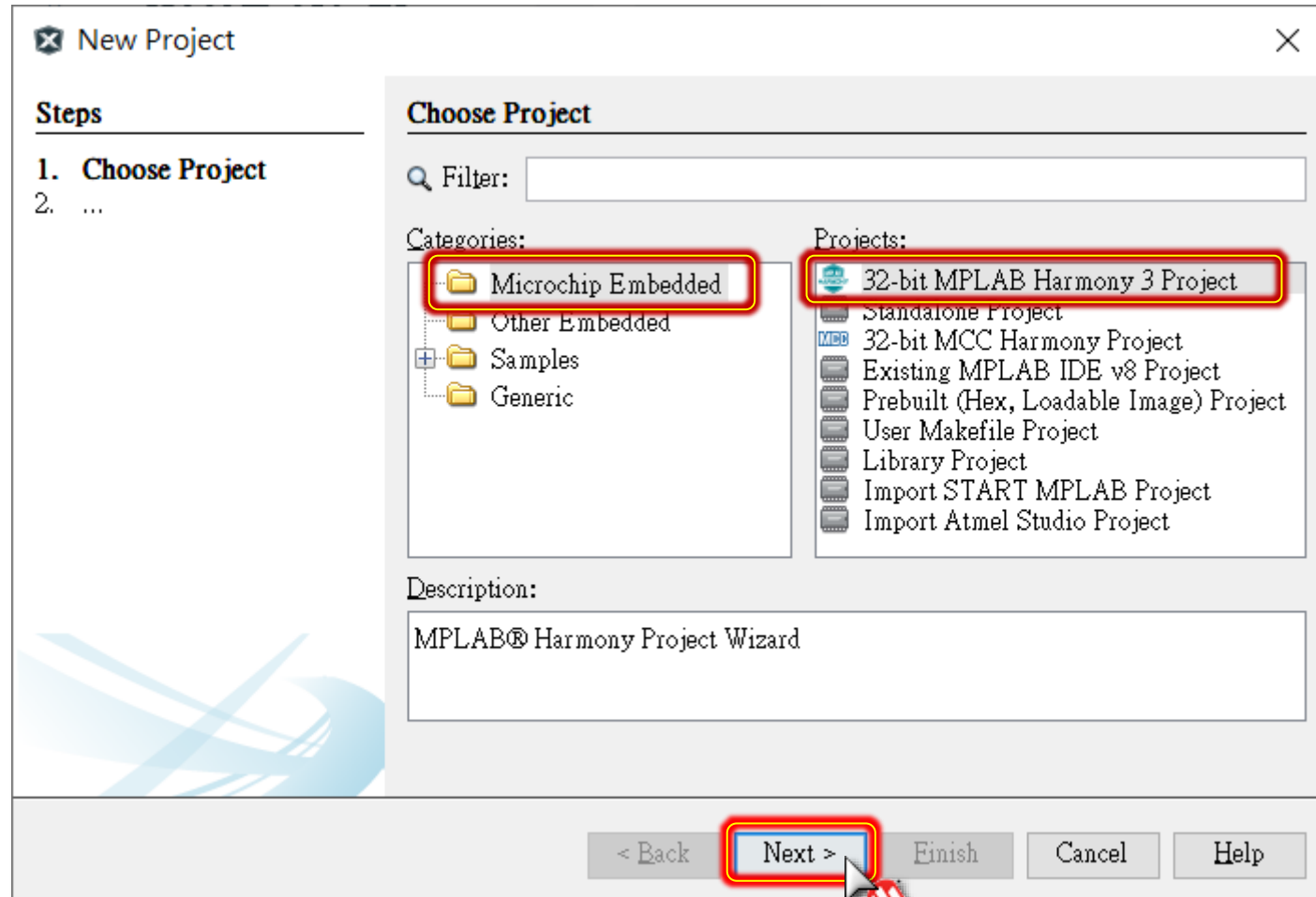


OR



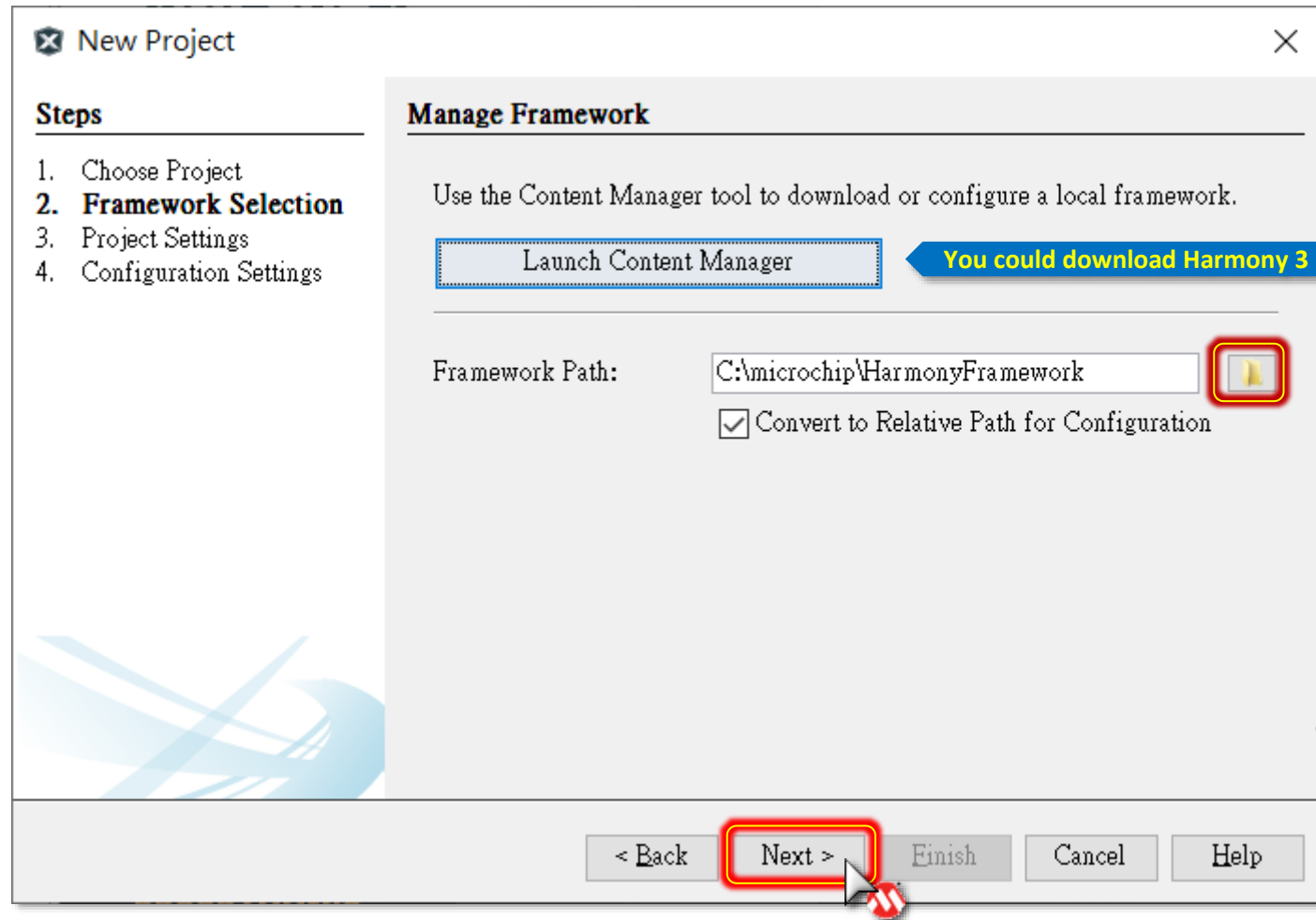
A New Harmony Project

- Categories : **Microchip Embedded**
- Projects : **32-bit MPLAB Harmony 3 Project**



A New Harmony Project

- Given the directory path of your **Harmony 3 Framework**.
- You should download **Harmony 3 Framework** individually.



Learning about
Harmony Content Manager
Please click here

A New Harmony Project

- Given the Project Folder Location.
- Given the Project Folder Name. (It will be your Project Name)

New Project

Steps

1. Choose Project
2. Framework Selection
- 3. Project Settings**
4. Configuration Settings

Name and Location

Location: C:\Exercises_PIC3001GFX\Lab0_First_Project

Folder: Lab0_First_Project **Folder name will be the Project Name**

Name: Lab0_First_Project

Path: \\Lab0_First_Project\\firmware\\ \\Lab0_First_Project.X

Show Visual Help

< Back **Next >** Finish Cancel Help

A New Harmony Project

- Device Filter : **PIC32MZ2064DAS176**
- **Please do not miss any character of your part number.**

New Project

Steps

1. Choose Project
2. Framework Selection
3. Project Settings
- 4. Configuration Settings**

Configuration Settings

Name: default

Device Family: All Target Device: PIC32MZ2064DAS176

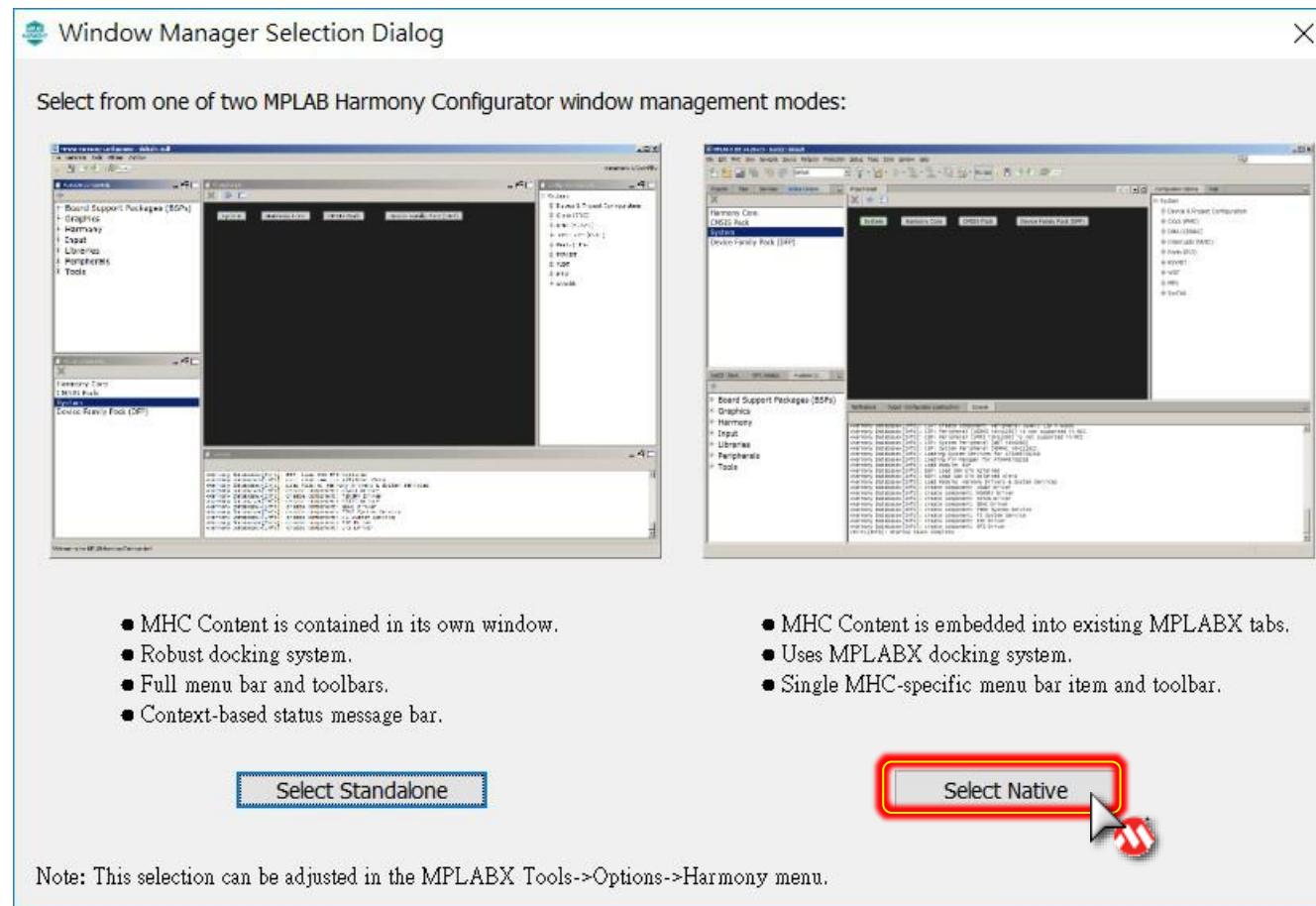
Device Filter: PIC32MZ2064DAS176 **Type in correct part number**

Show Visual Help

< Back Next > **Finish** Cancel Help

A New Harmony Project

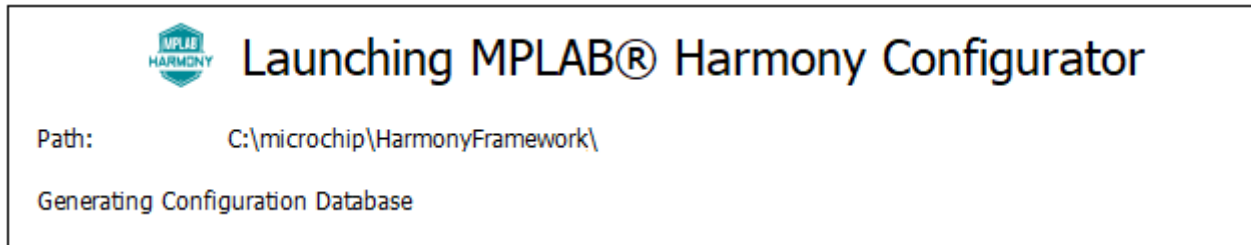
- Select Harmony Configurator window management mode, if it's your **first time** to entering Harmony after the install.



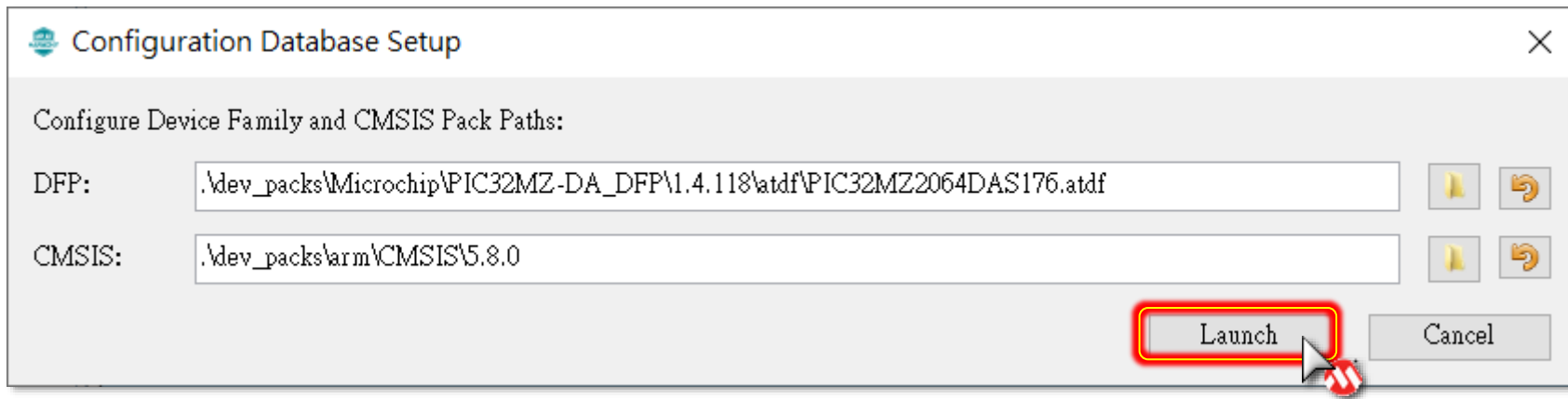
Learning about
To change the MHC Mode
Please click here

A New Harmony Project

- **Launching the MPLAB Harmony Configurator (MHC)**

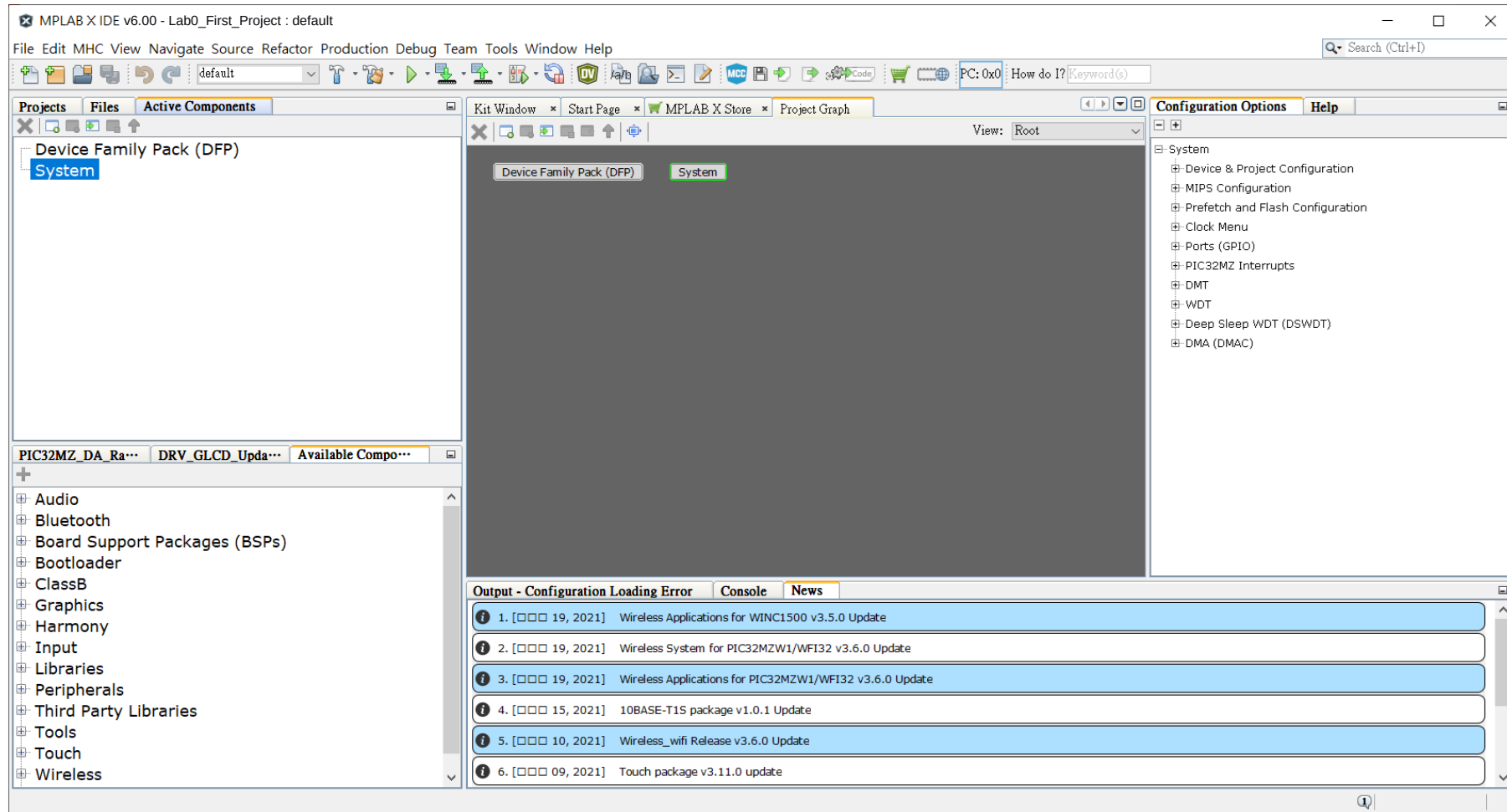


- **Configure Device Family and CMSIS Pack Path. (Keep default)**



A New Harmony Project

- **MPLAB Harmony Configurator (MHC)** in Native Window Mode



Learning about
MHC GUI Interface
Please click here

A New Harmony Project

- Try Generate Code for a new Harmony Project.

The screenshot illustrates the process of generating code for a new Harmony Project in the MPLAB X IDE. The main window shows the 'Generate Code' button in the toolbar, which is highlighted with a red box. A blue arrow points from this button to the 'Generate Project' dialog box. The 'Generate Project' dialog box is open, showing the 'Generate Settings' section with the 'Merge Strategy' set to 'USER_ALL'. A blue arrow points from the 'Generate' button in this dialog to the 'Generating Project...' dialog box. The 'Generating Project...' dialog box shows a progress table and a 100% completion bar.

Task Type	Remaining	Total
File Markup	0	16
File Copy	0	16
Libraries	0	0
Settings	0	7
Source Paths	0	0

Project generation complete!

100% Please Wait...

Generate Project

1. Configure Generation Settings

☒ Generate Settings

Merge Strategy: USER_ALL

(Mouse over a property for detailed help)

2. View Warnings

Type	Description
------	-------------

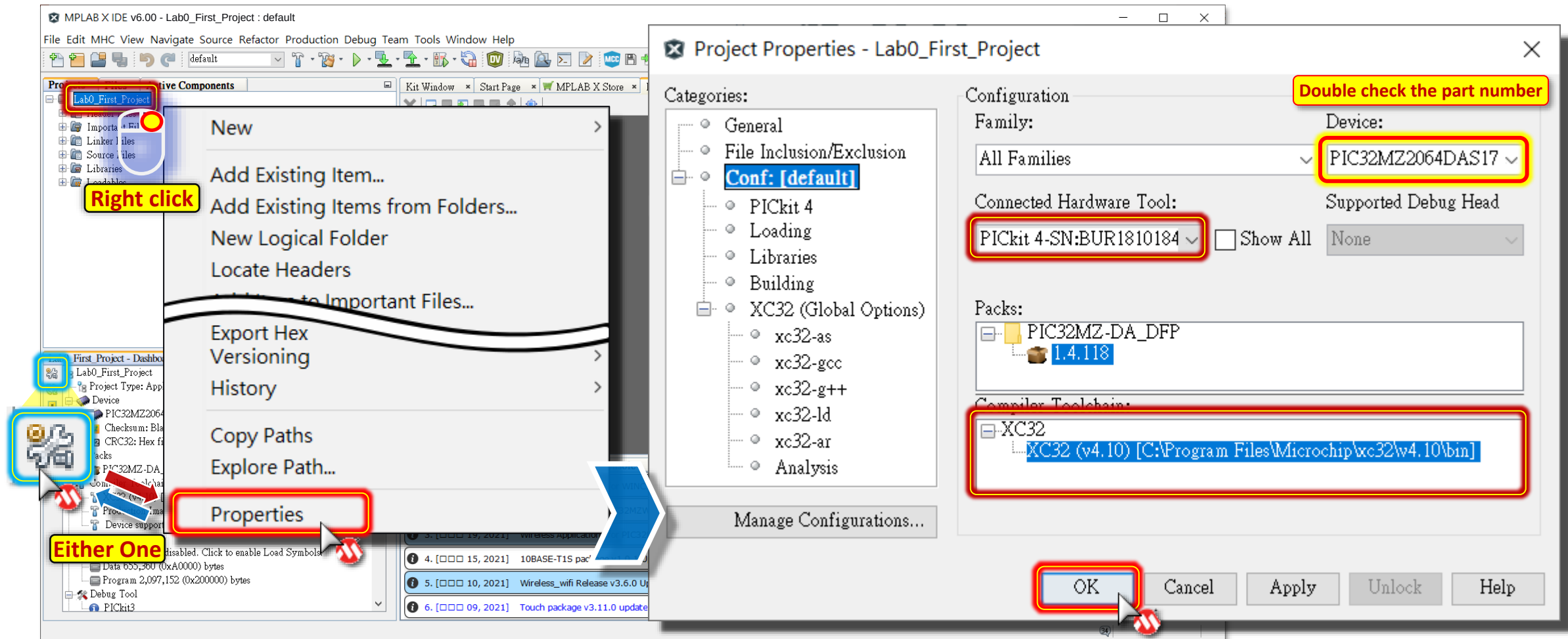
3. Click Generate

Default linker script is added to the project.
Disable it for the project that require custom linker script.

Generate Cancel

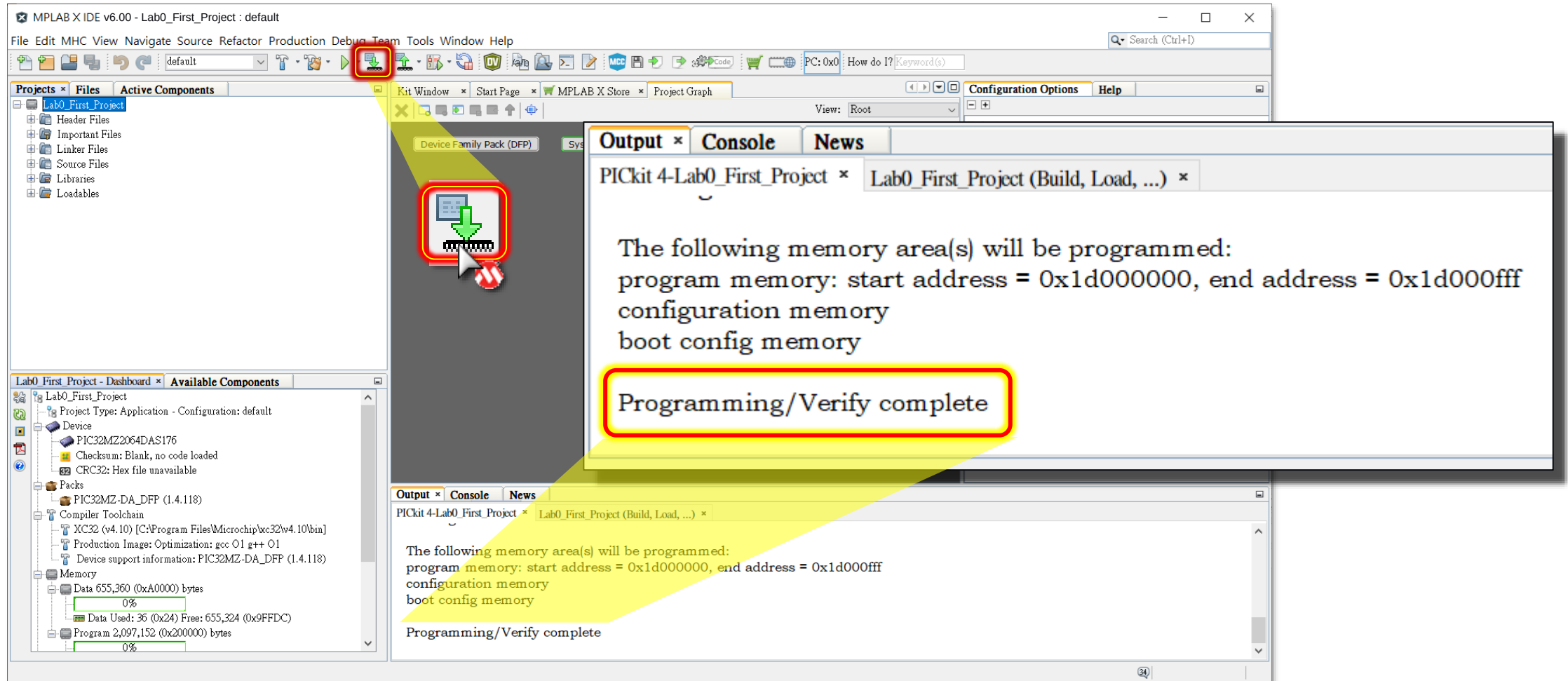
A New Harmony Project

- **Setup Project Properties for programming tool and compiler.**



A New Harmony Project

- Try program to make sure the connection and check result.

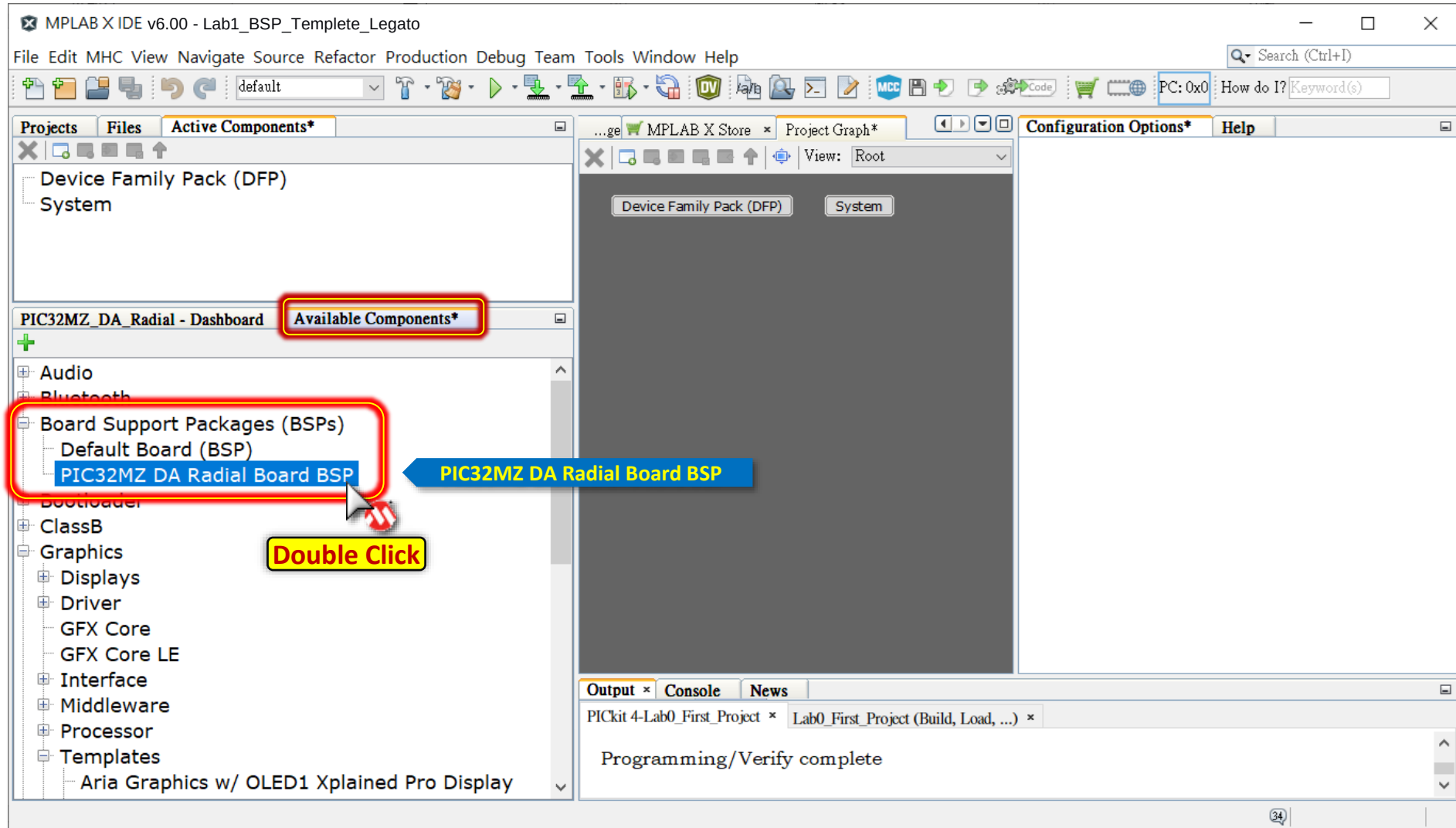


Board Support Packages (BSP)

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

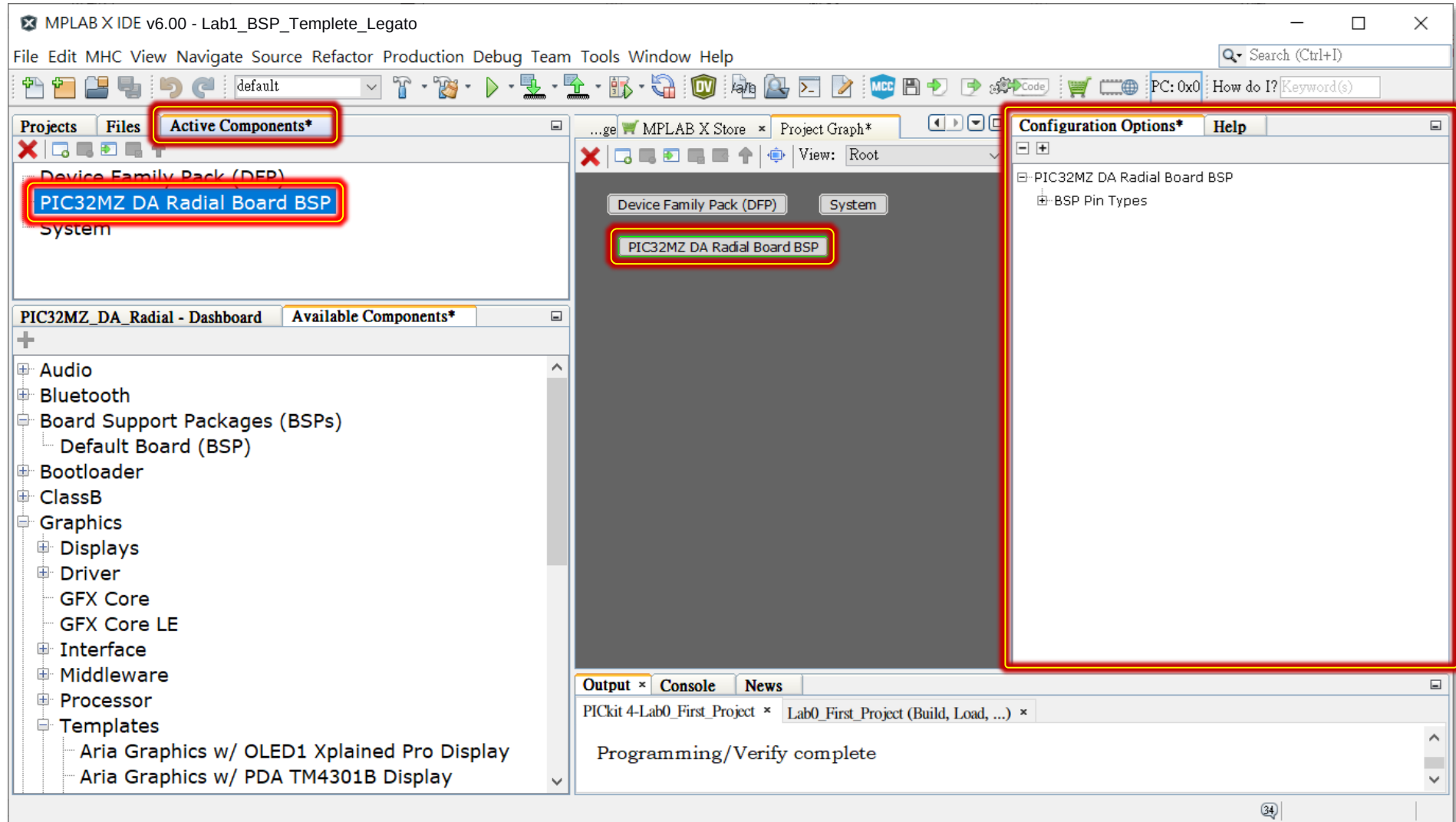
Board Support Packages (BSP)

- Find **PIC32MZ DA Radial Board BSP** in Available Components.



Board Support Packages (BSP)

- **PIC32MZ DA Radial Board BSP** been added to Project Graph.



Development Board Template

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Development Board Template

- Find **Legato Graphic w/ LCF0300633GGU00 Display** Template.

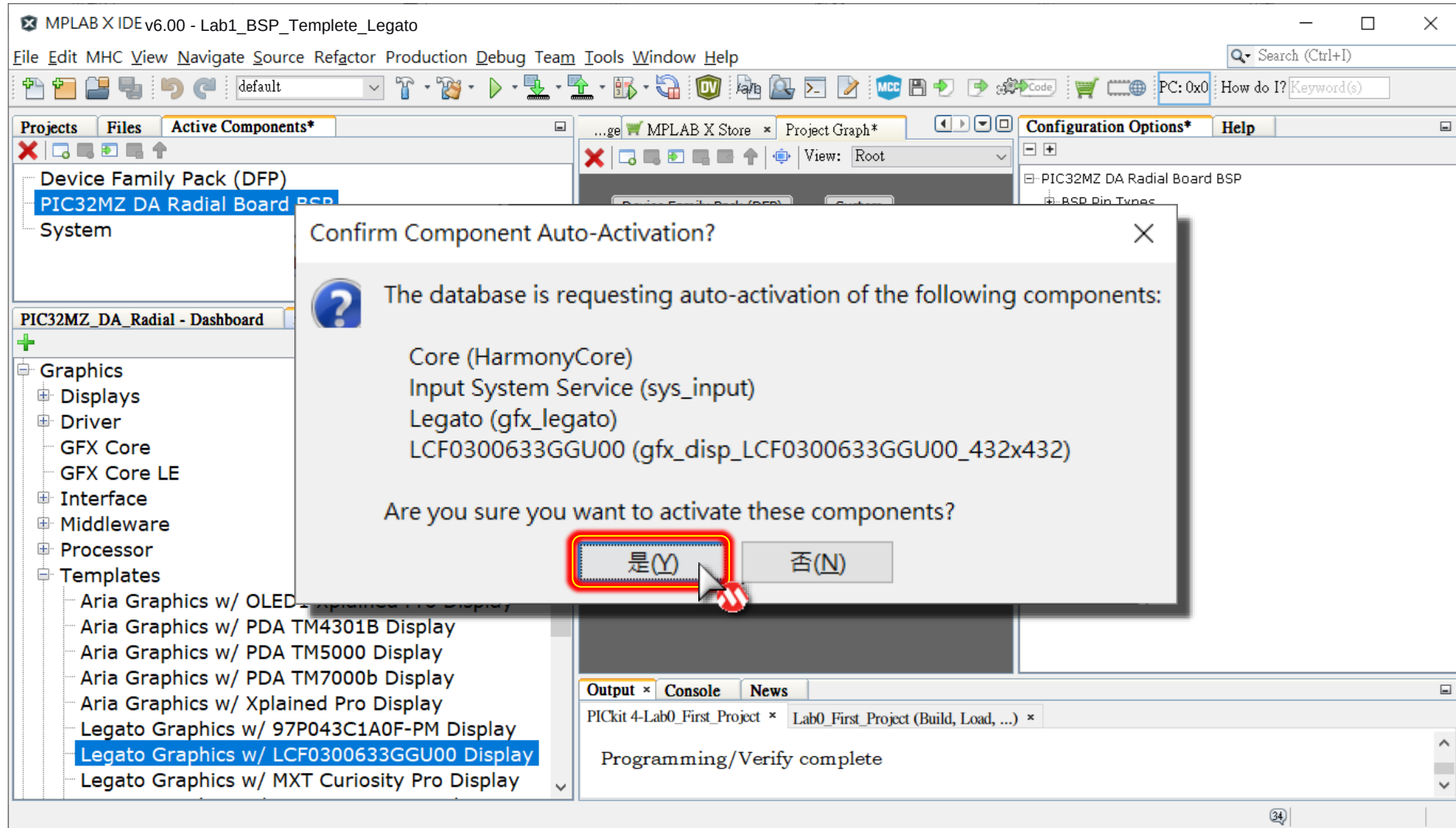
The screenshot shows the MPLAB X IDE v6.00 interface with the 'Lab1_BSP_Template_Legato' project open. The 'Active Components*' window on the left lists the 'Device Family Pack (DFP)' and 'PIC32MZ DA Radial Board BSP'. The 'Available Components*' window is open, showing a tree structure under 'Graphics' and 'Templates'. The 'Legato Graphics w/ LCF0300633GGU00 Display' component is highlighted in blue. A red box highlights the 'Graphics' category, and another red box highlights the 'Templates' category. A red box also highlights the selected component. A red callout box with yellow text says: 'Note that !!! It will spend a while to add components, make activation and connected'. A yellow box with black text says 'Double Click'. The 'Output' window at the bottom shows the project name and a status message.

Note that !!!
It will spend a while to add components,
make activation and connected

Double Click

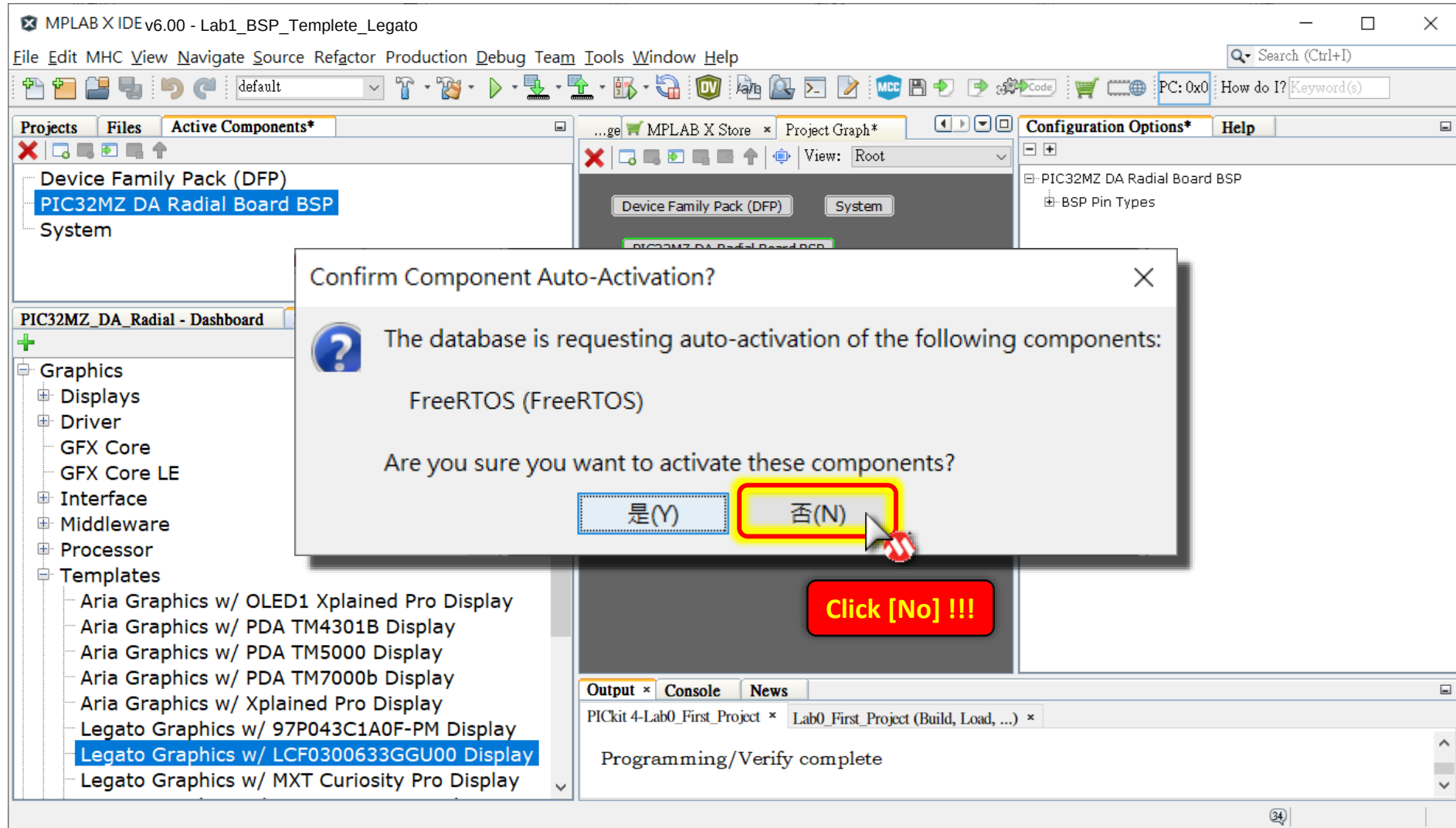
Development Board Template

- Click on **[Yes]** to auto-activation all the requested components.



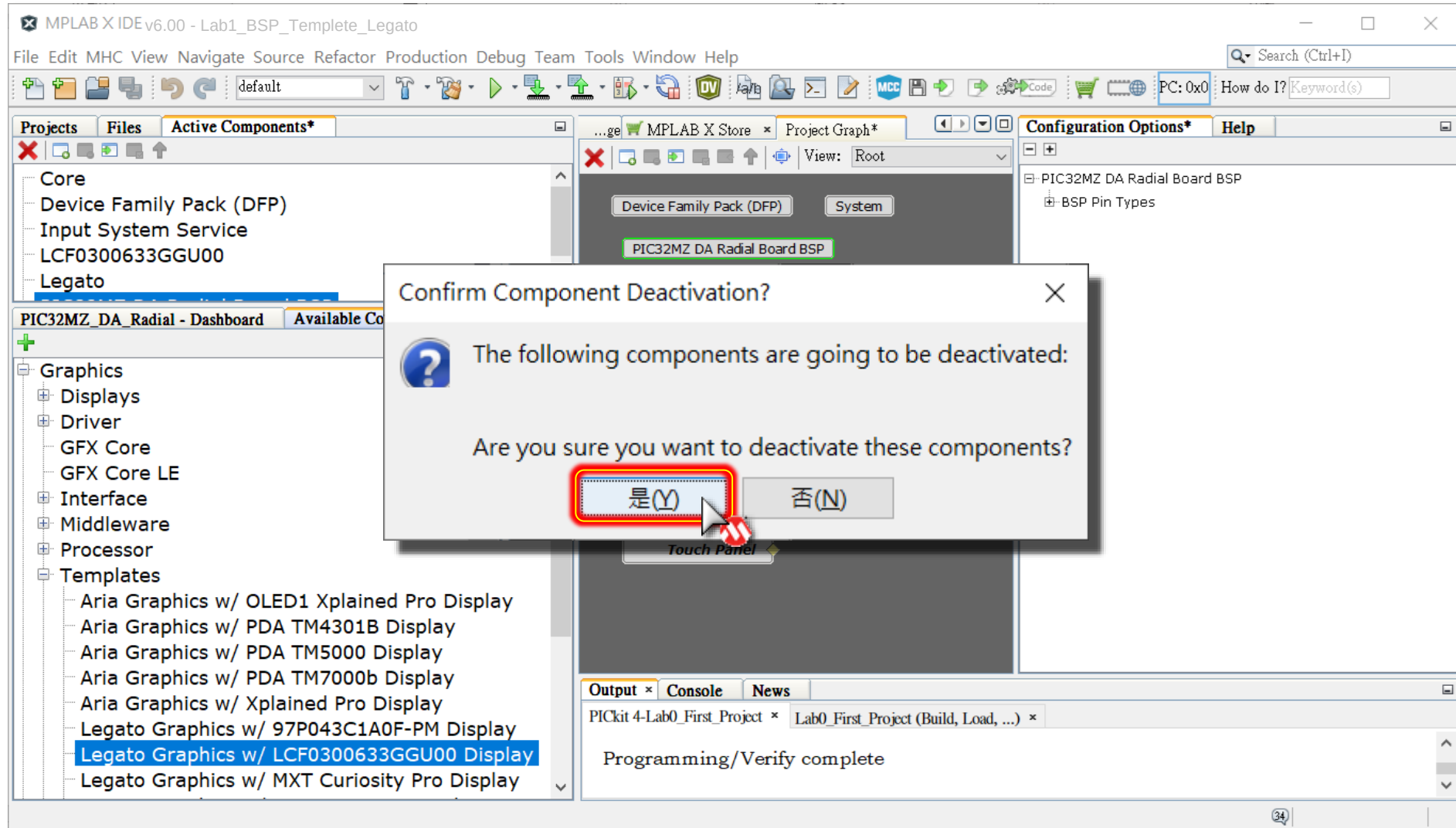
Development Board Template

- Click on **[No]** to not-activate the FreeRTOS.



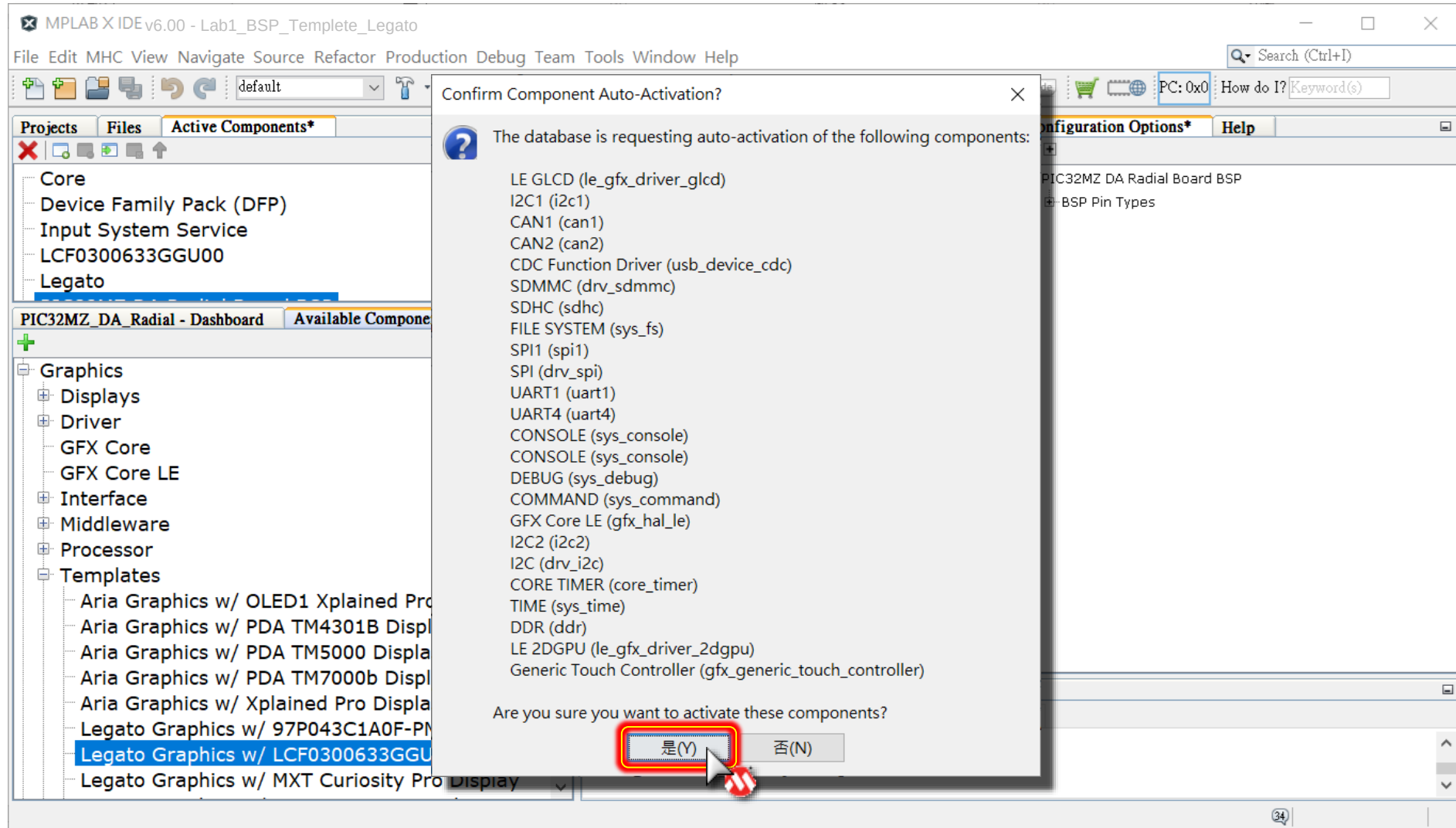
Development Board Template

- Click on **[Yes]** to deactivate components. (Not thing will be deactivated for a new Project)



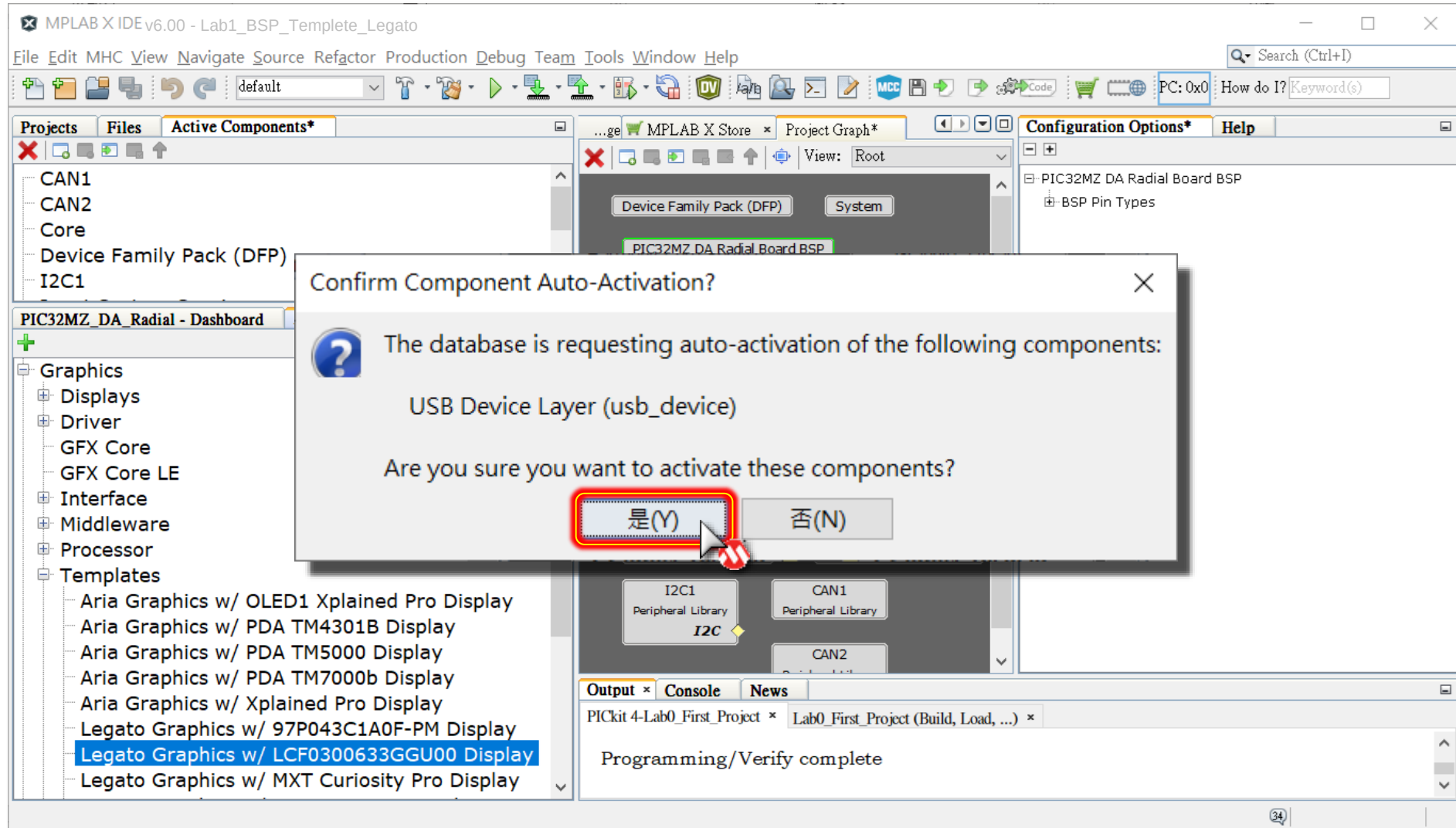
Development Board Template

- Click on **[Yes]** to auto-activation all the requested components.



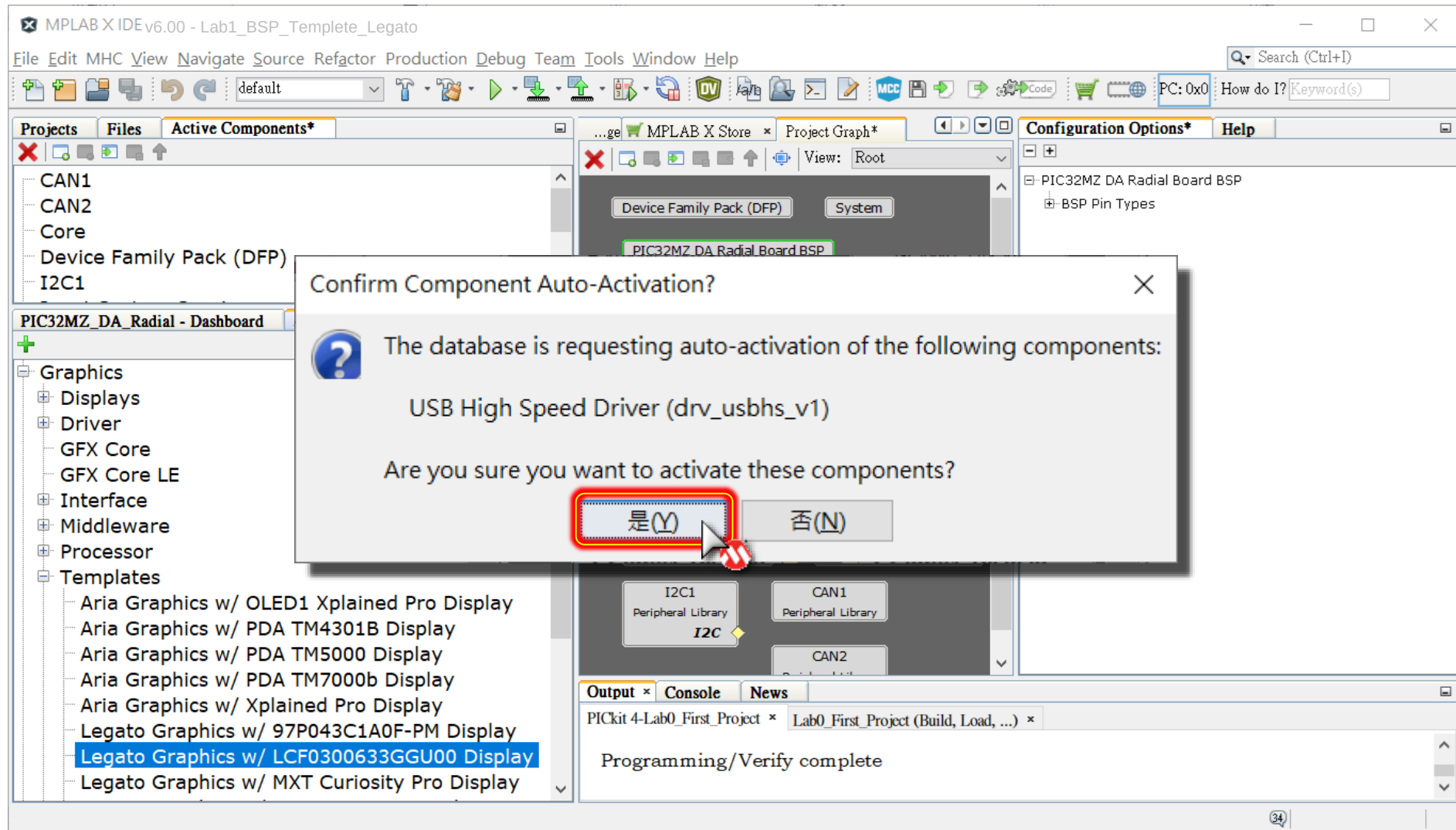
Development Board Template

- Click on **[Yes]** to auto-activation **USB Device Layer (usb_device)**



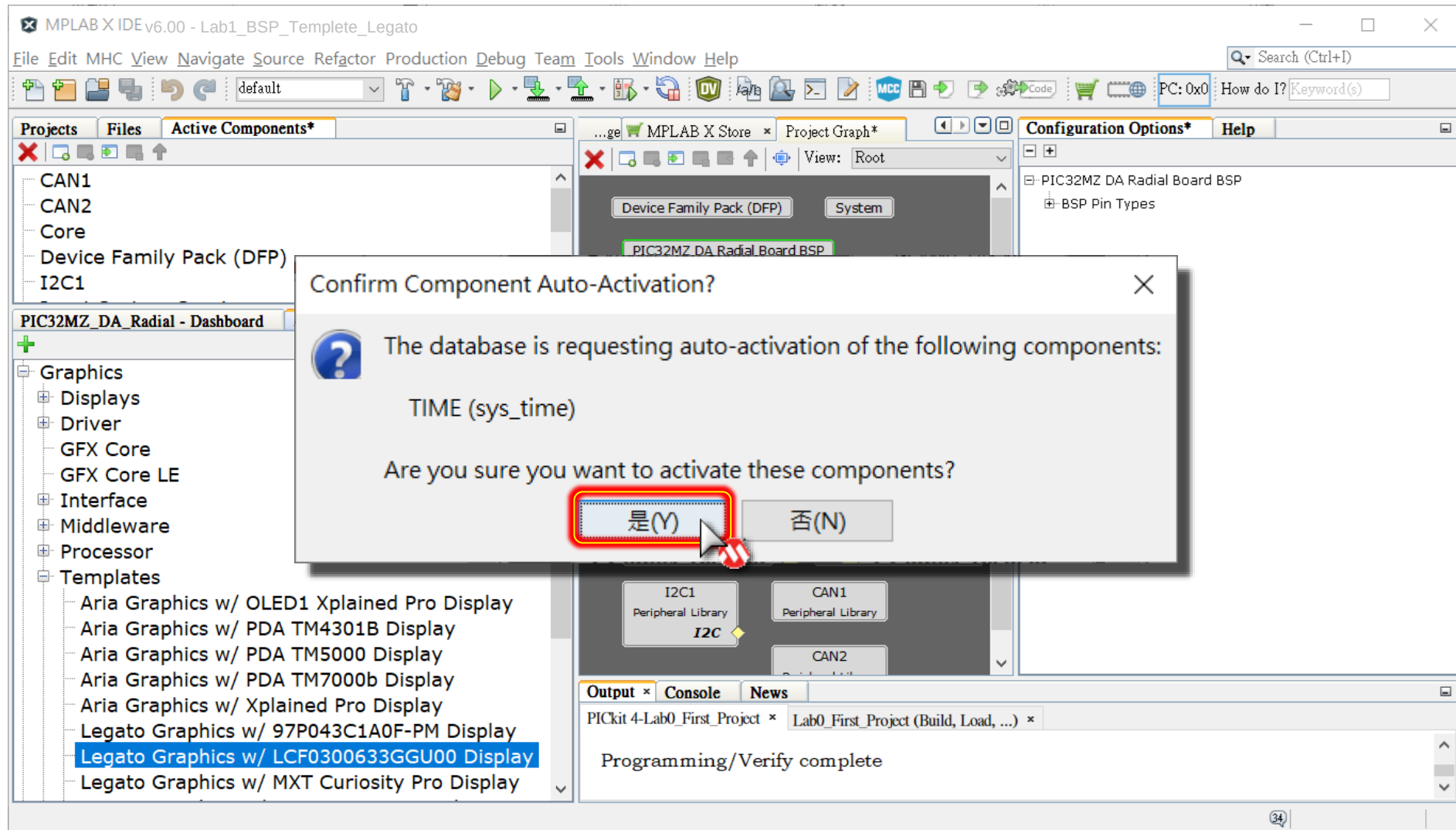
Development Board Template

- Click on **[Yes]** to auto-activation **USB HS Driver (drv_usbhs_v1)**



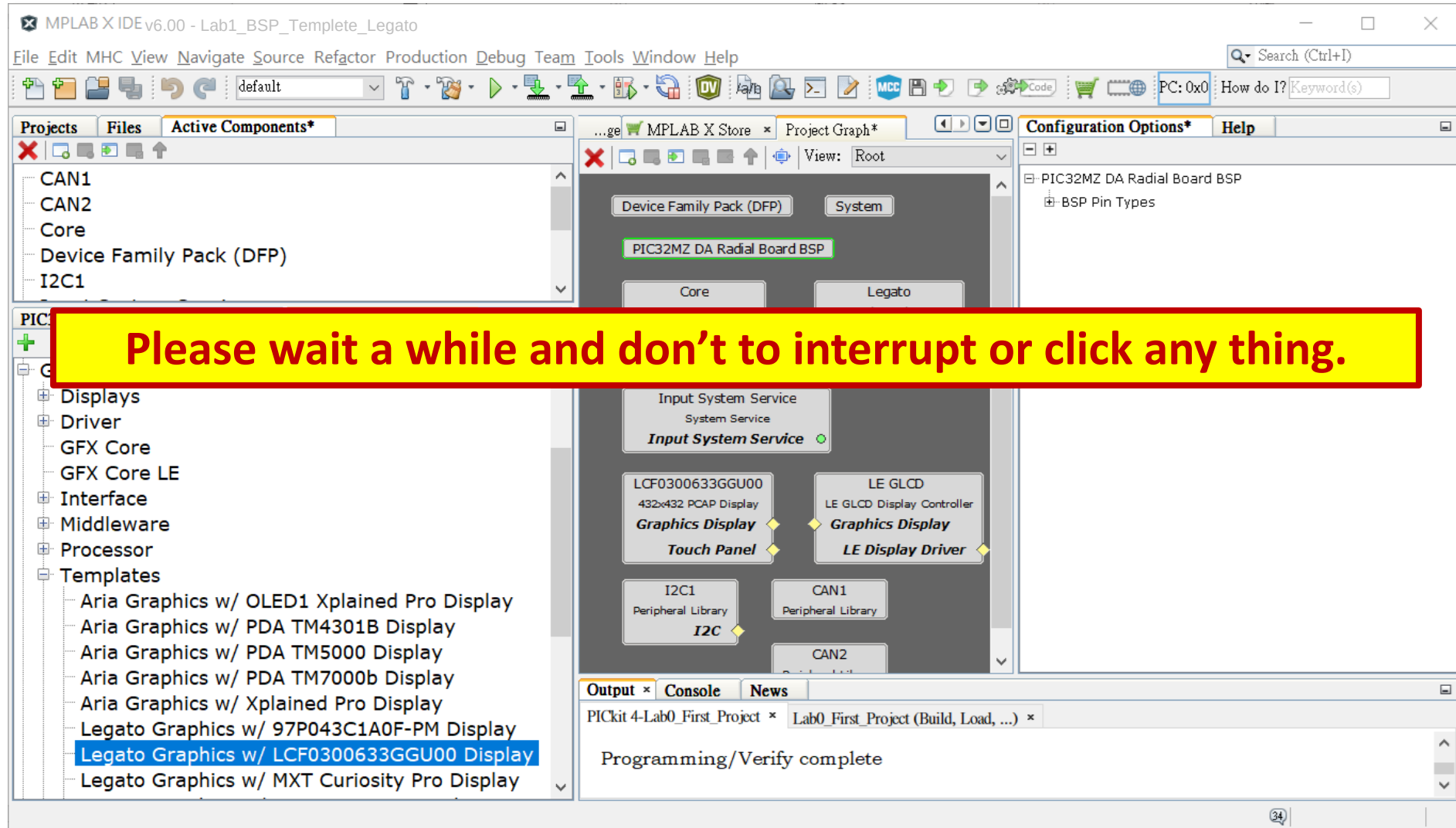
Development Board Template

- Click on **[Yes]** to auto-activation System Service **TIME (sys_time)**



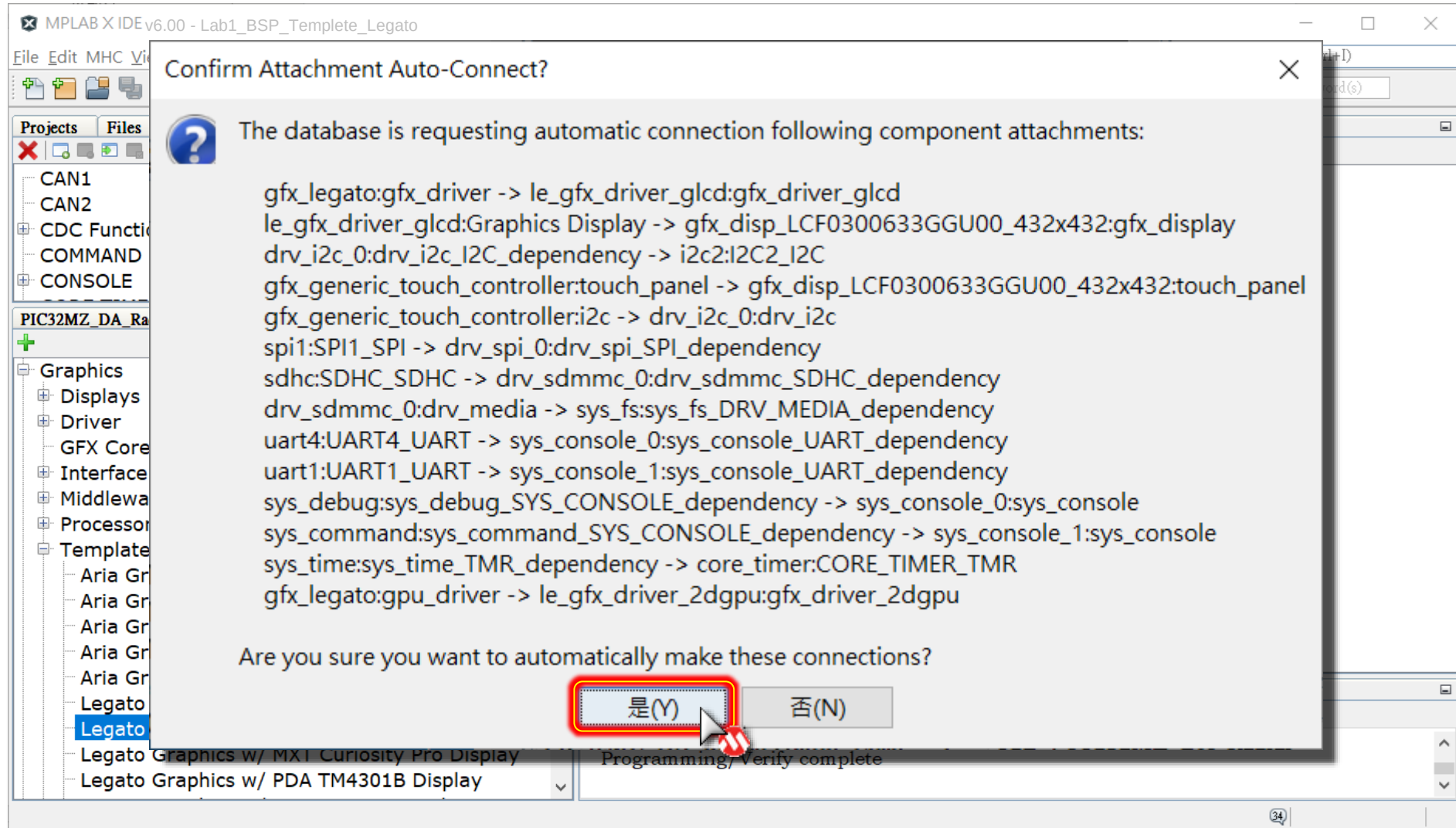
Development Board Template

- Please wait a while here (10 to 30 seconds).



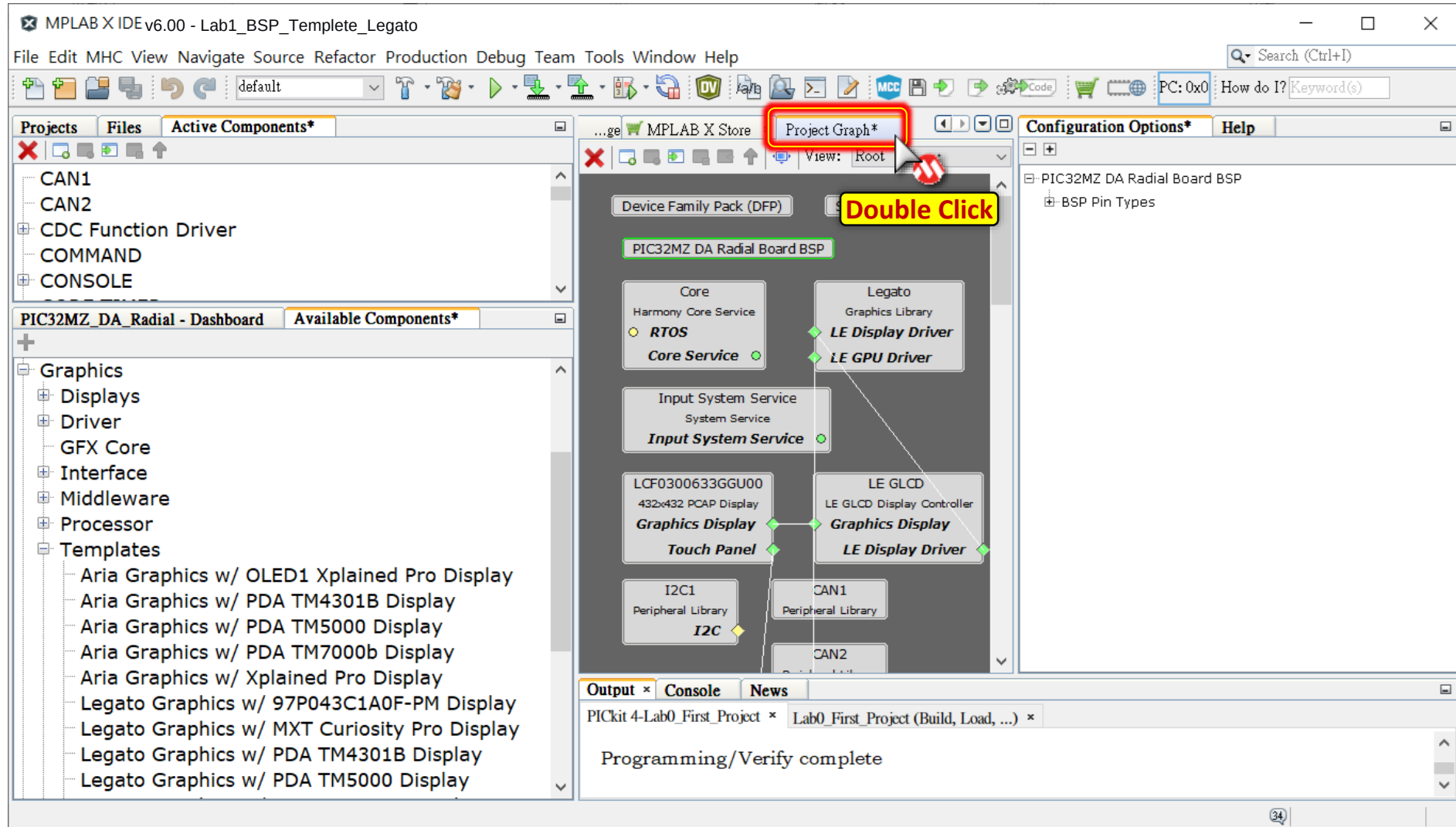
Development Board Template

- Click on **[Yes]** to automatic connection all the components.

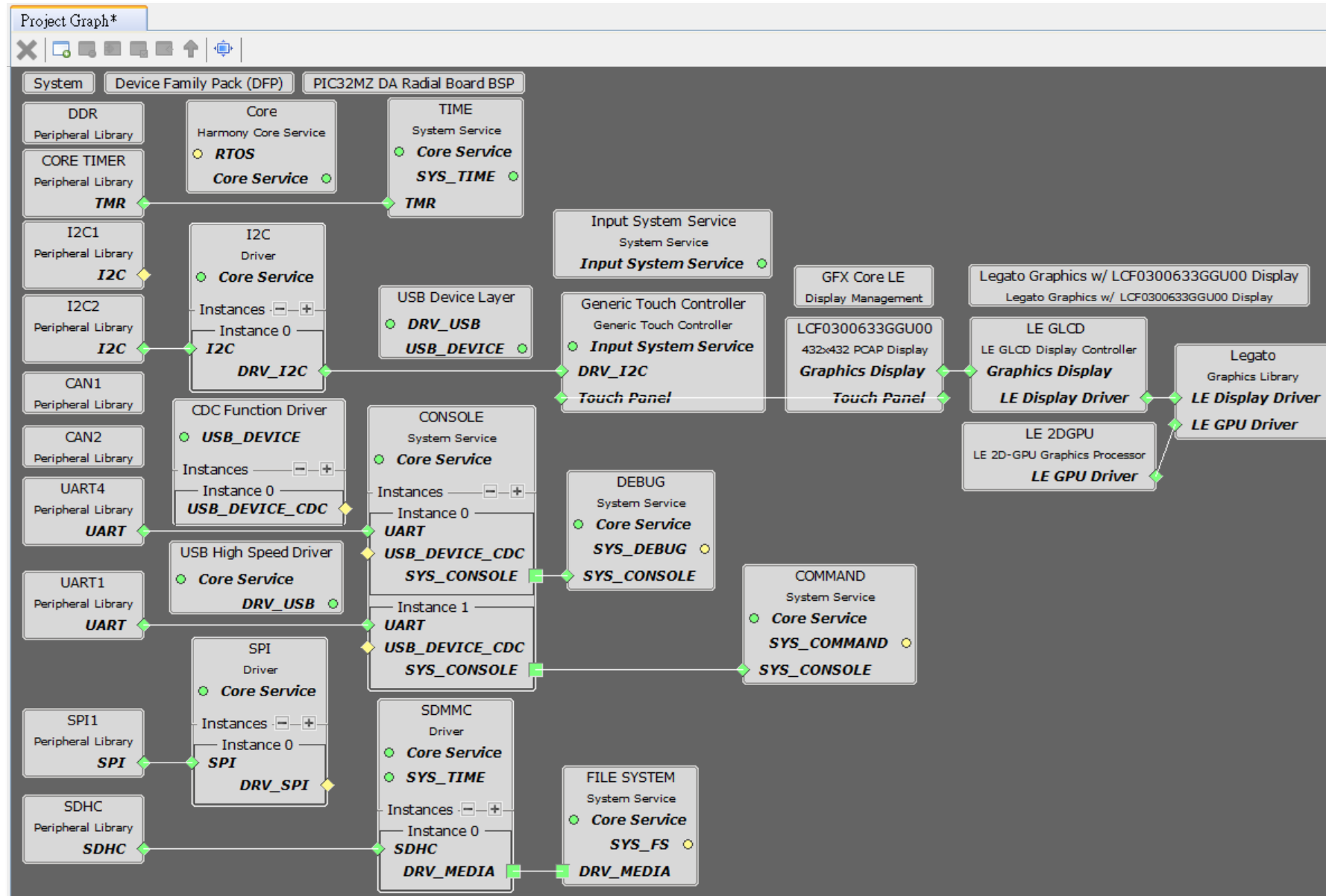


Development Board Template

- Double Click on **[Project Graph]** tab to enlarge the window.



Development Board Template

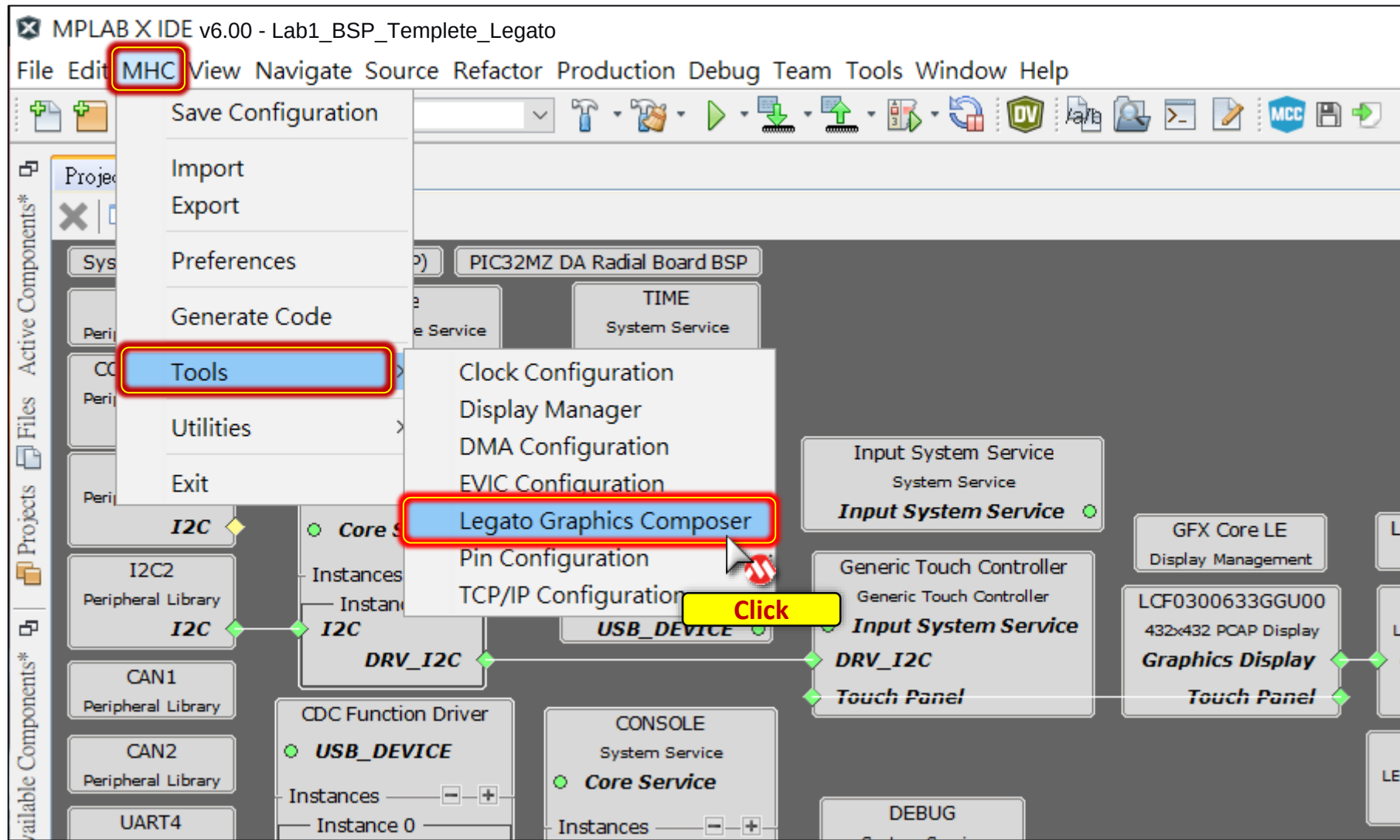


Legato Graphic Composer

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Legato Graphic Composer

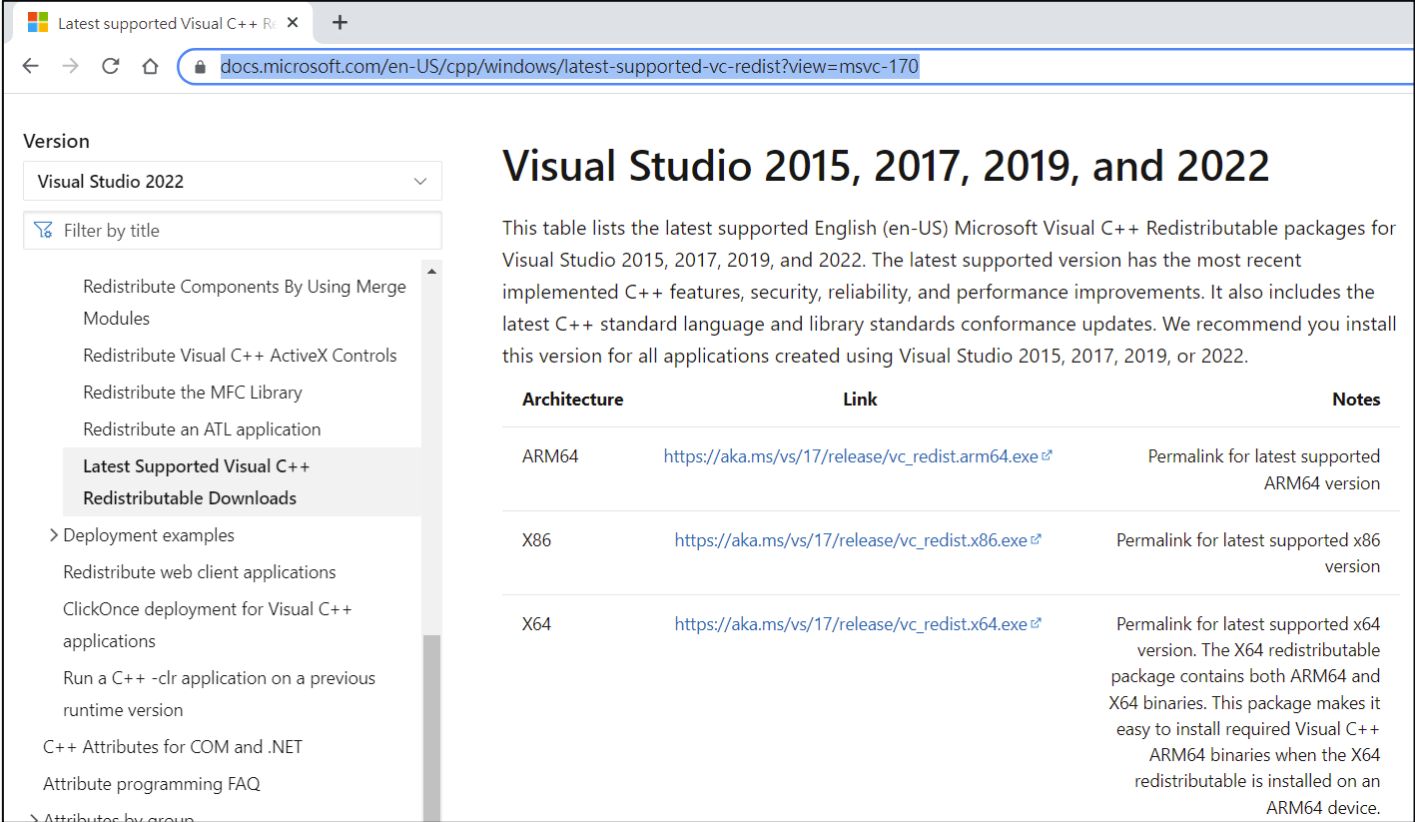
- Click [MHC] > [Tools] > [Legato Graphic Composer]



Legato Graphic Composer

- If problem to open(execute) Legato Graphic Composer, please install latest **Microsoft C and C++ (MSVC) runtime libraries** from below link.

<https://docs.microsoft.com/en-US/cpp/windows/latest-supported-vc-redist?view=msvc-170>

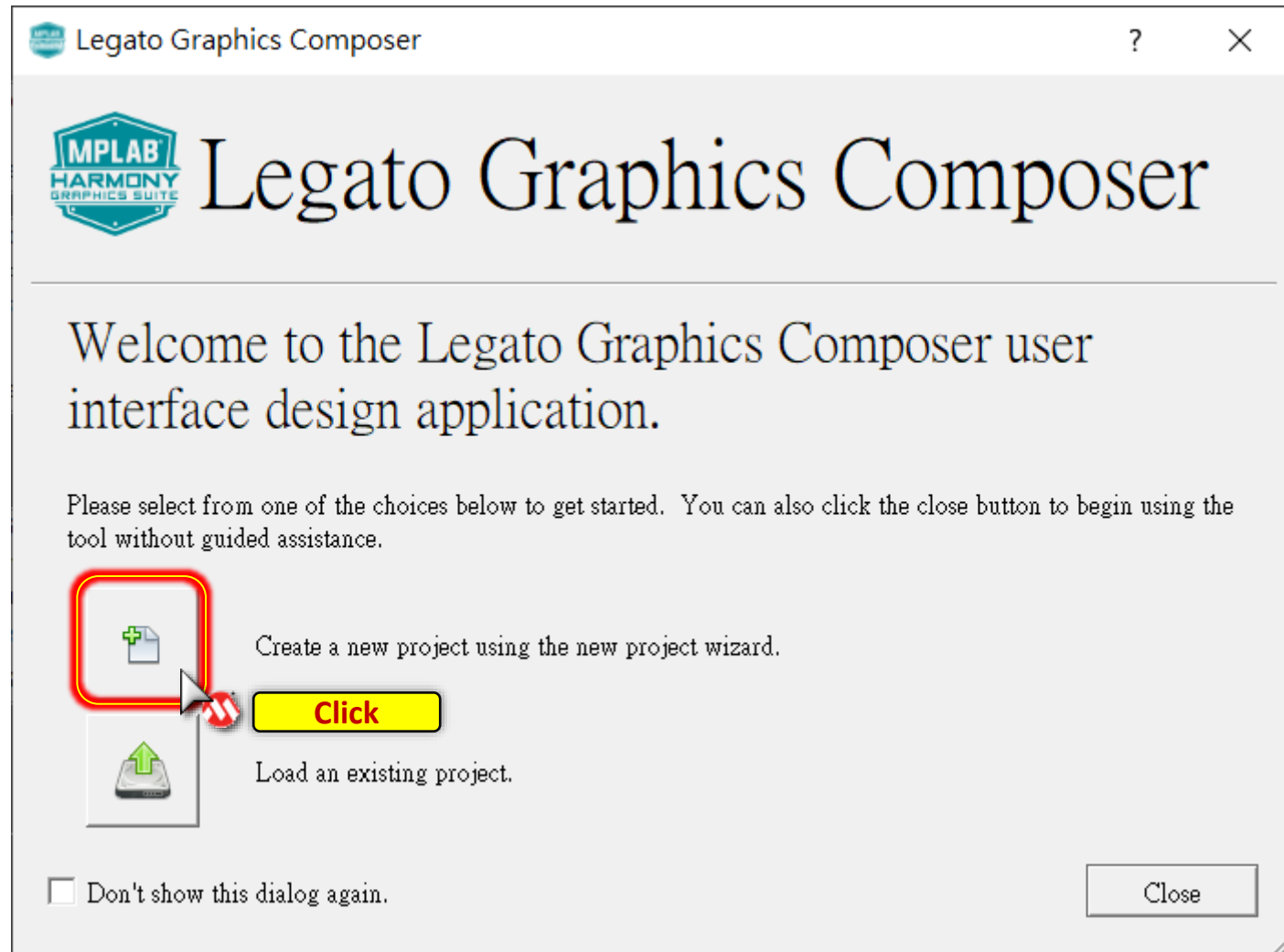


The screenshot shows a web browser window with the URL `docs.microsoft.com/en-US/cpp/windows/latest-supported-vc-redist?view=msvc-170`. The page title is "Visual Studio 2015, 2017, 2019, and 2022". The left sidebar contains a navigation menu with the following items: "Redistribute Components By Using Merge Modules", "Redistribute Visual C++ ActiveX Controls", "Redistribute the MFC Library", "Redistribute an ATL application", "Latest Supported Visual C++ Redistributable Downloads" (highlighted), "> Deployment examples", "Redistribute web client applications", "ClickOnce deployment for Visual C++ applications", "Run a C++ -clr application on a previous runtime version", "C++ Attributes for COM and .NET", and "Attribute programming FAQ". The main content area has the heading "Visual Studio 2015, 2017, 2019, and 2022" and a paragraph explaining that the table lists the latest supported English (en-US) Microsoft Visual C++ Redistributable packages. Below this is a table with three columns: "Architecture", "Link", and "Notes".

Architecture	Link	Notes
ARM64	https://aka.ms/vs/17/release/vc_redist.arm64.exe	Permalink for latest supported ARM64 version
X86	https://aka.ms/vs/17/release/vc_redist.x86.exe	Permalink for latest supported x86 version
X64	https://aka.ms/vs/17/release/vc_redist.x64.exe	Permalink for latest supported x64 version. The X64 redistributable package contains both ARM64 and X64 binaries. This package makes it easy to install required Visual C++ ARM64 binaries when the X64 redistributable is installed on an ARM64 device.

Legato Graphic Composer

- Create a new project using the new project wizard.



Legato Graphic Composer

- Display Configuration of Width and Height set to **432** x **432** pixel.

Legato Graphics Composer - New Project Wizard

Steps

1. Display Configuration
2. Color Mode
3. Memory Profile
4. Project Template
5. Finish

Display Configuration

Width: 432 432

Height: 432 432

Presets: HVGA (480x320) Apply

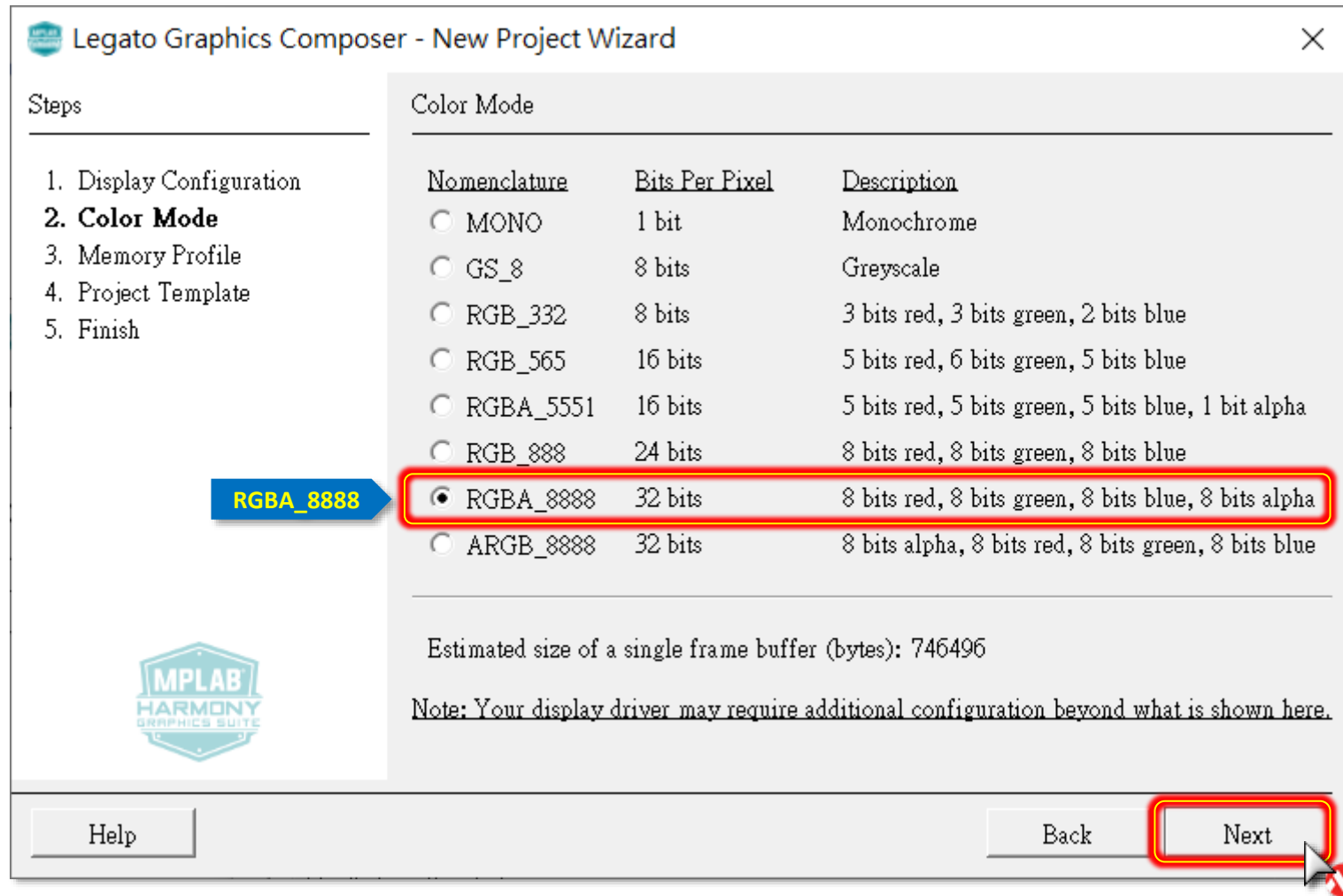
Total pixel count for a single frame buffer: 186624

Note: Your display driver may require additional configuration beyond what is shown here.

Help Back Next

Legato Graphic Composer

- Color Mode select to **RGBA_8888**



Legato Graphics Composer - New Project Wizard

Steps

1. Display Configuration
- 2. Color Mode**
3. Memory Profile
4. Project Template
5. Finish

Color Mode

Nomenclature	Bits Per Pixel	Description
☐ MONO	1 bit	Monochrome
☐ GS_8	8 bits	Greyscale
☐ RGB_332	8 bits	3 bits red, 3 bits green, 2 bits blue
☐ RGB_565	16 bits	5 bits red, 6 bits green, 5 bits blue
☐ RGBA_5551	16 bits	5 bits red, 5 bits green, 5 bits blue, 1 bit alpha
☐ RGB_888	24 bits	8 bits red, 8 bits green, 8 bits blue
☒ RGBA_8888	32 bits	8 bits red, 8 bits green, 8 bits blue, 8 bits alpha
☐ ARGB_8888	32 bits	8 bits alpha, 8 bits red, 8 bits green, 8 bits blue

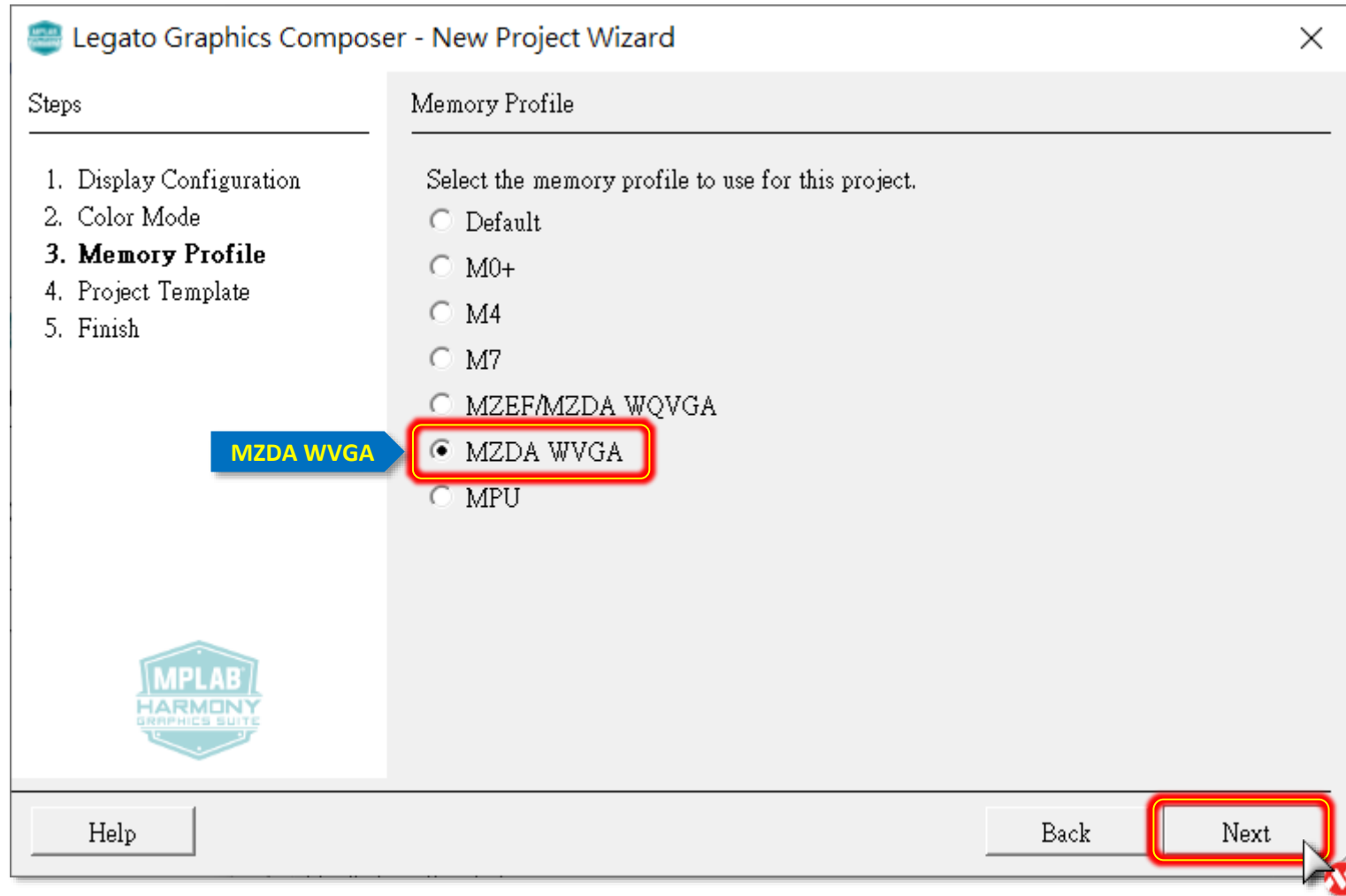
Estimated size of a single frame buffer (bytes): 746496

Note: Your display driver may require additional configuration beyond what is shown here.

Help Back **Next**

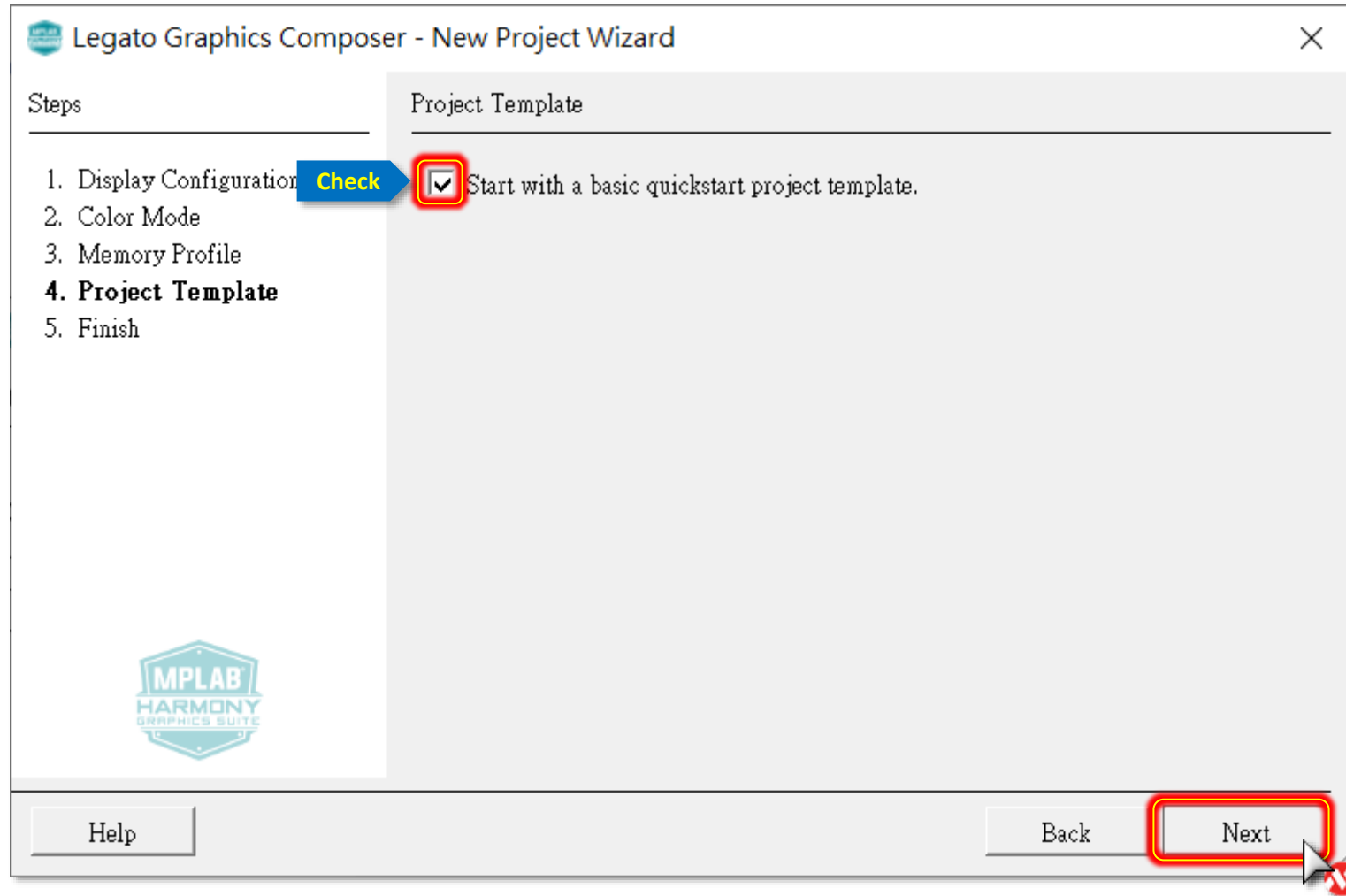
Legato Graphic Composer

- Memory Profile select to **MZDA WVGA**



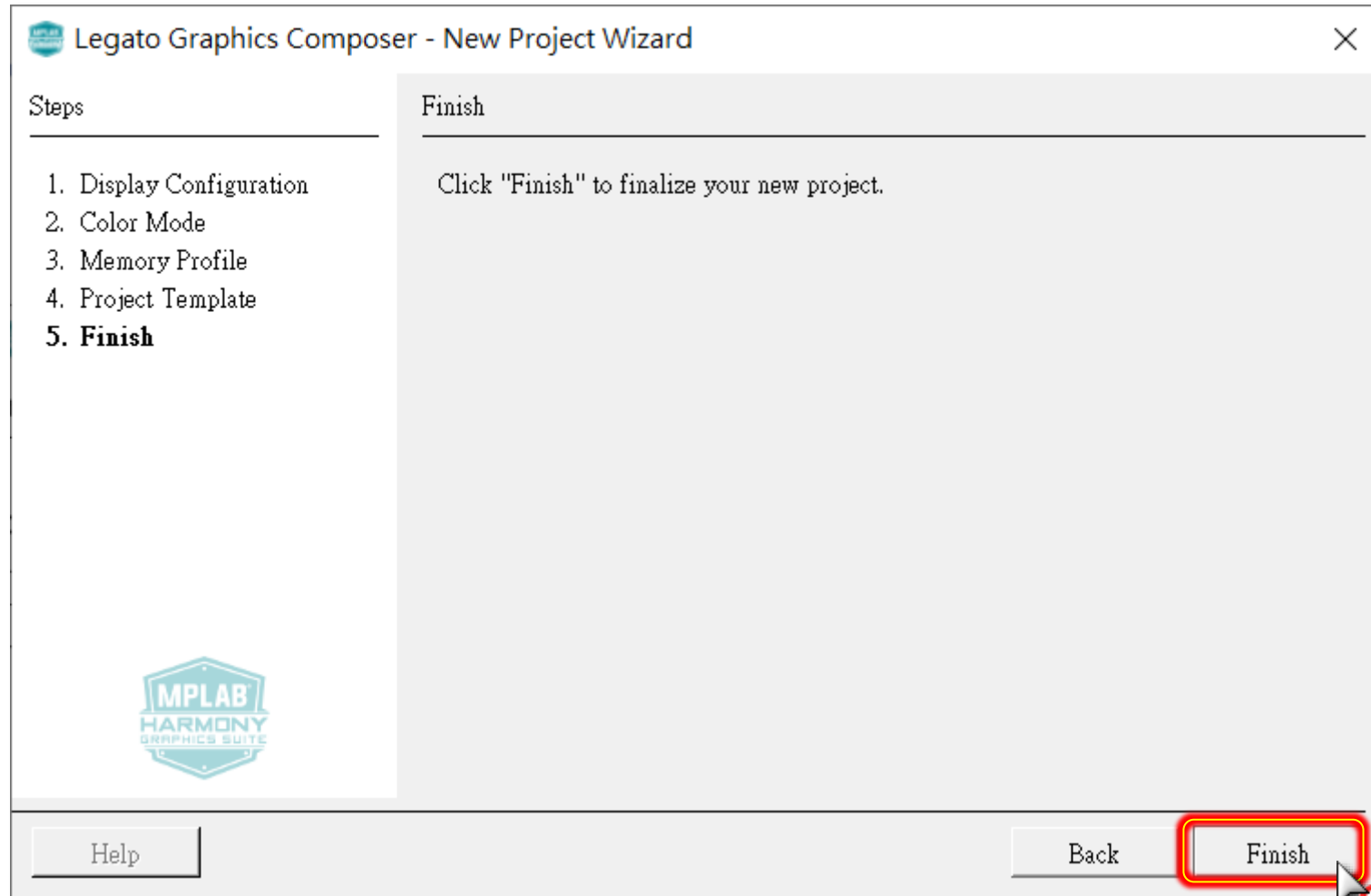
Legato Graphic Composer

- Start with a basic quick start project template.



Legato Graphic Composer

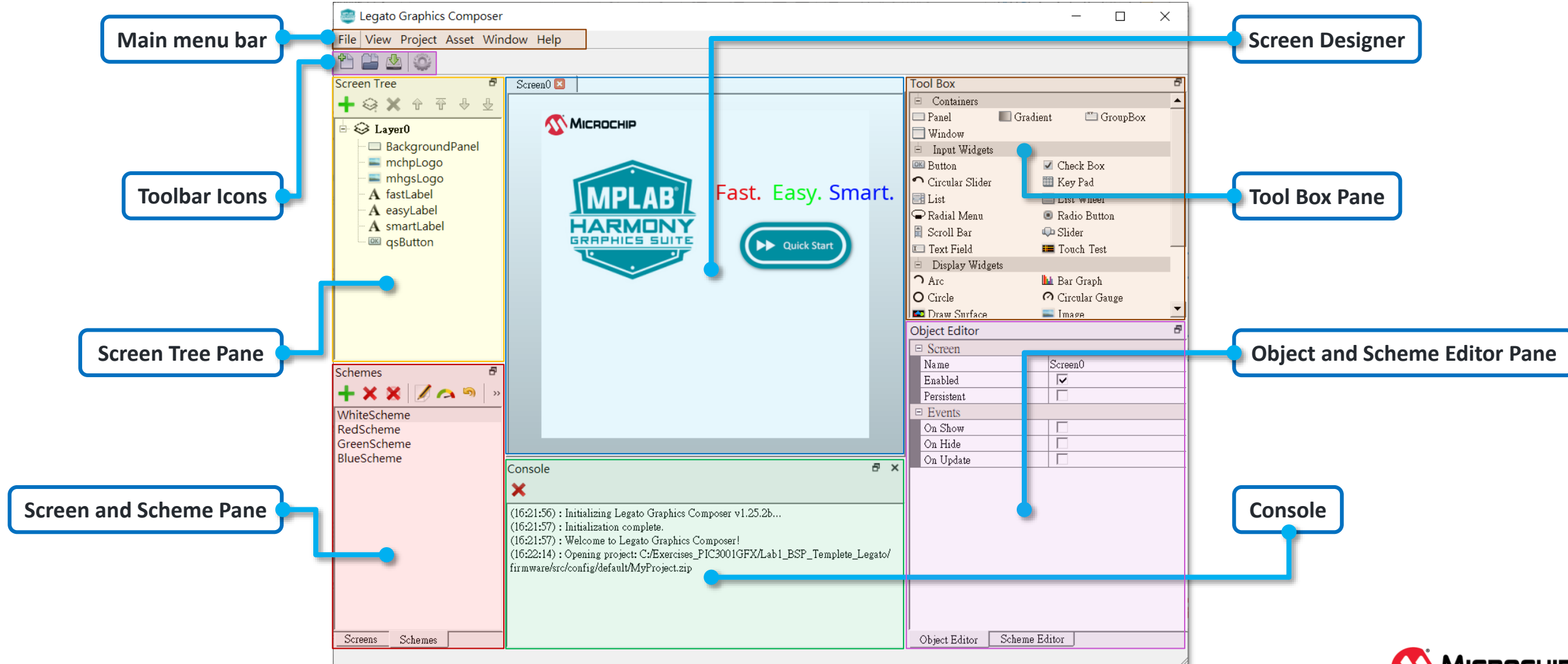
- Finish a new Legato Graphic Composer project.



Click

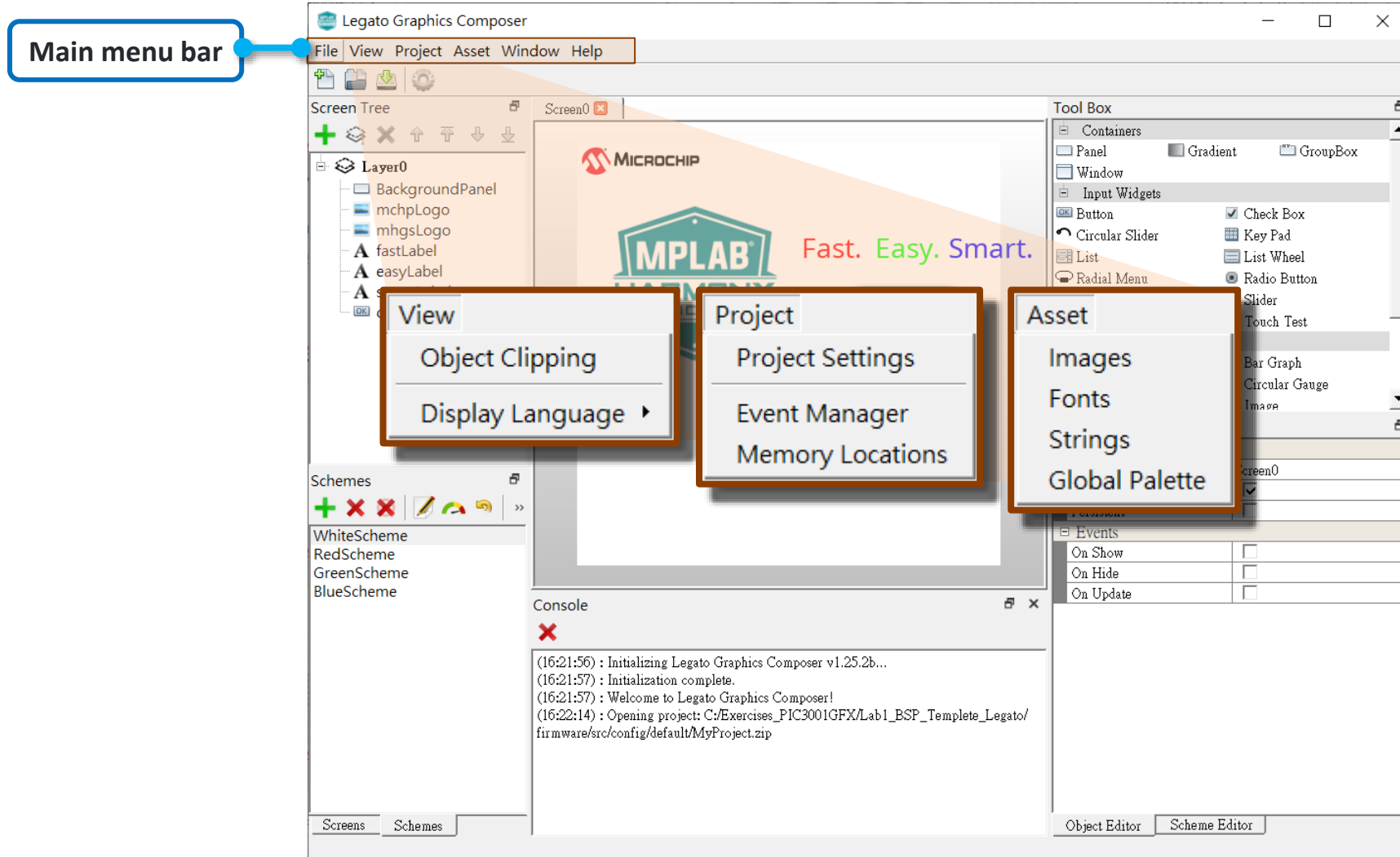
Legato Graphic Composer

- Legato Graphic Composer GUI.



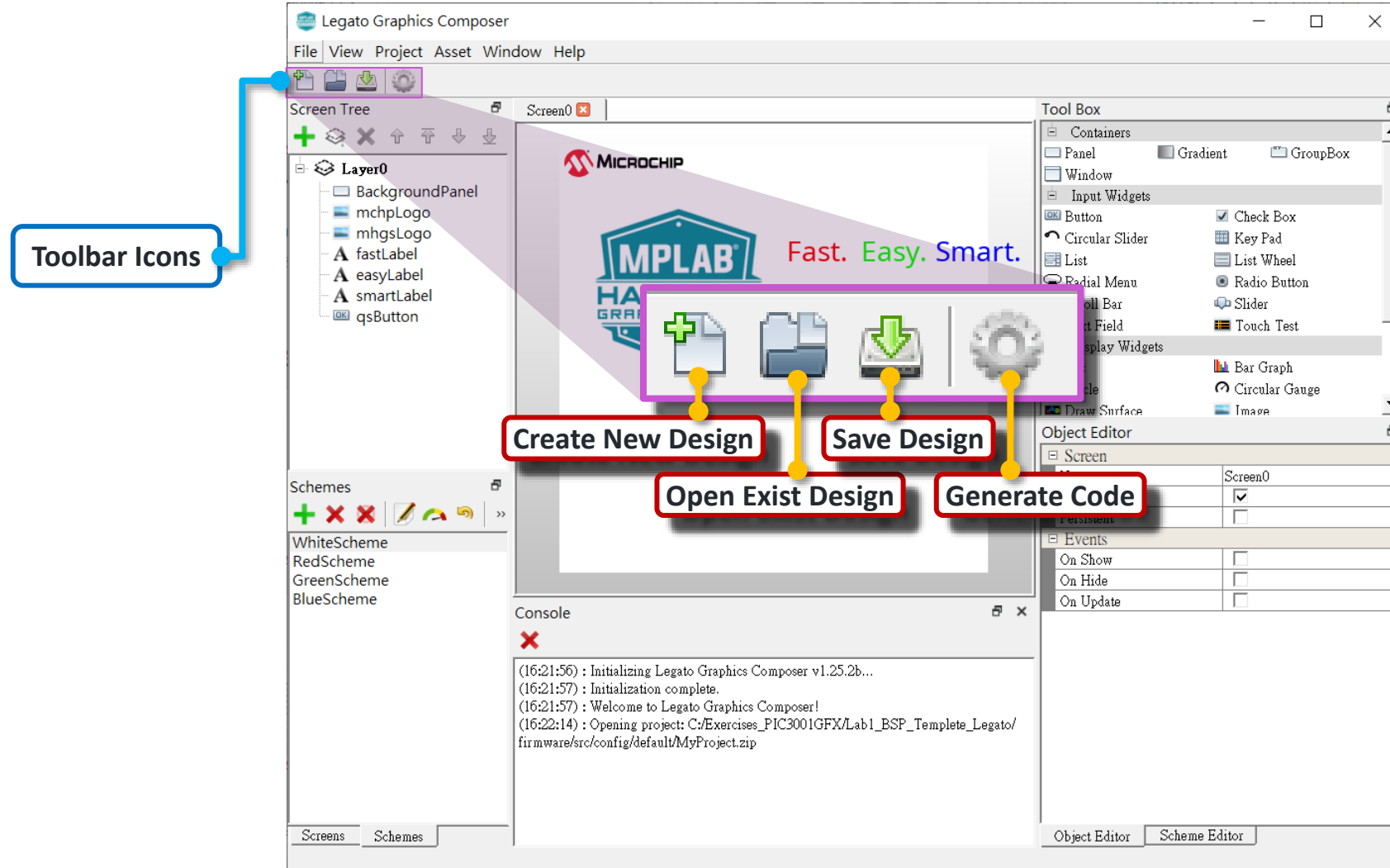
Legato Graphic Composer

- Legato Graphic Composer GUI.



Legato Graphic Composer

- Legato Graphic Composer GUI.

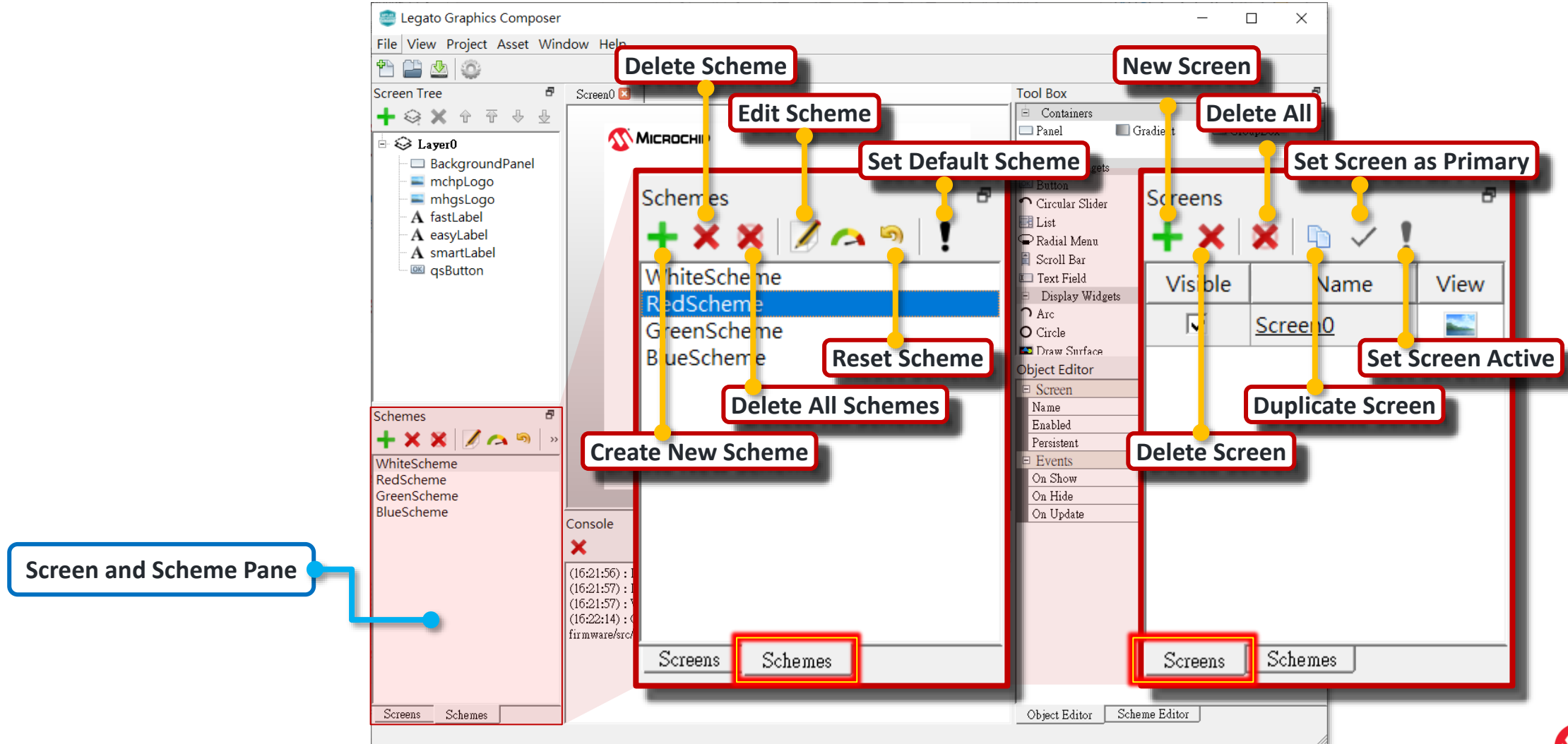


- **Legato Graphic Composer GUI.**



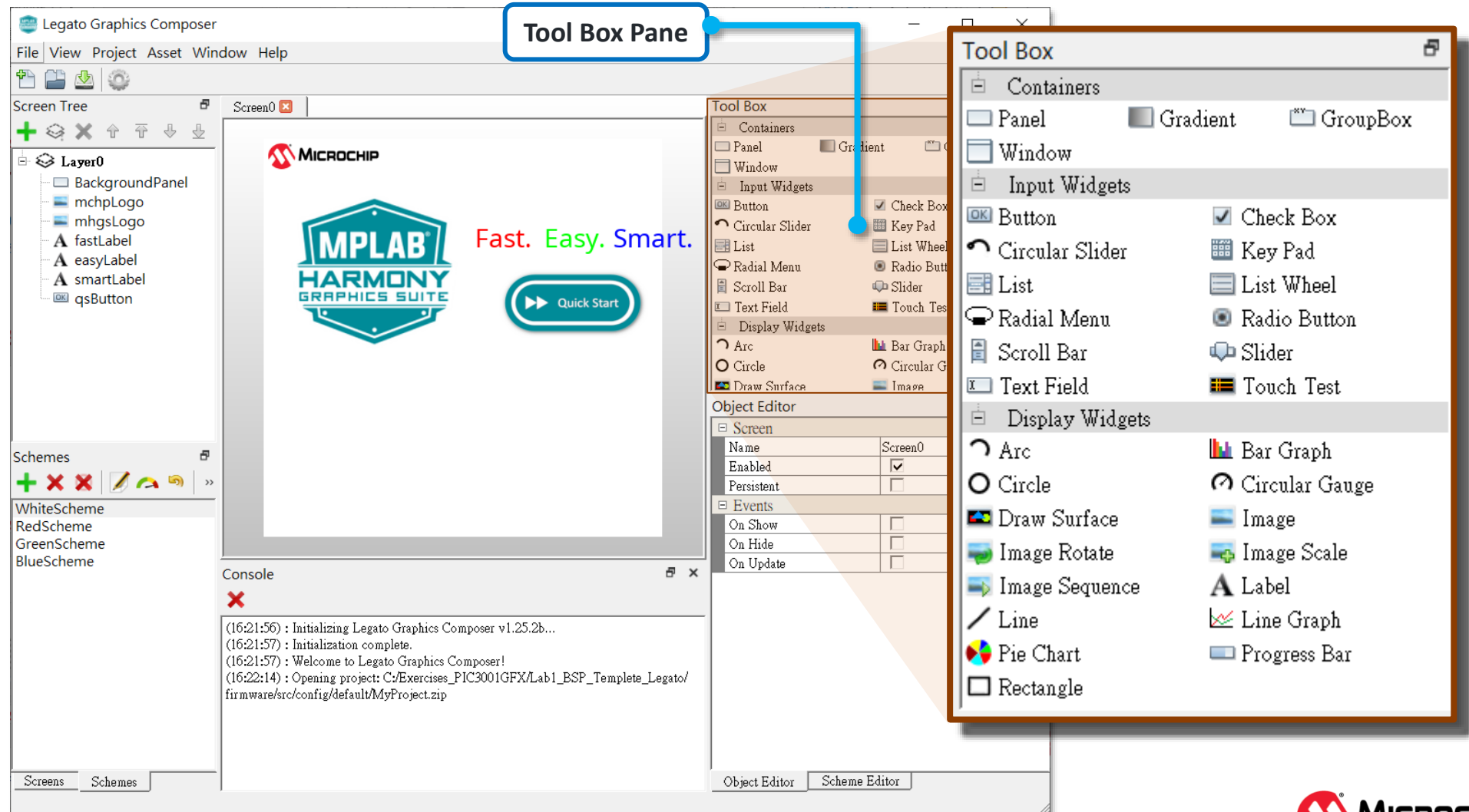
Legato Graphic Composer

- Legato Graphic Composer GUI.



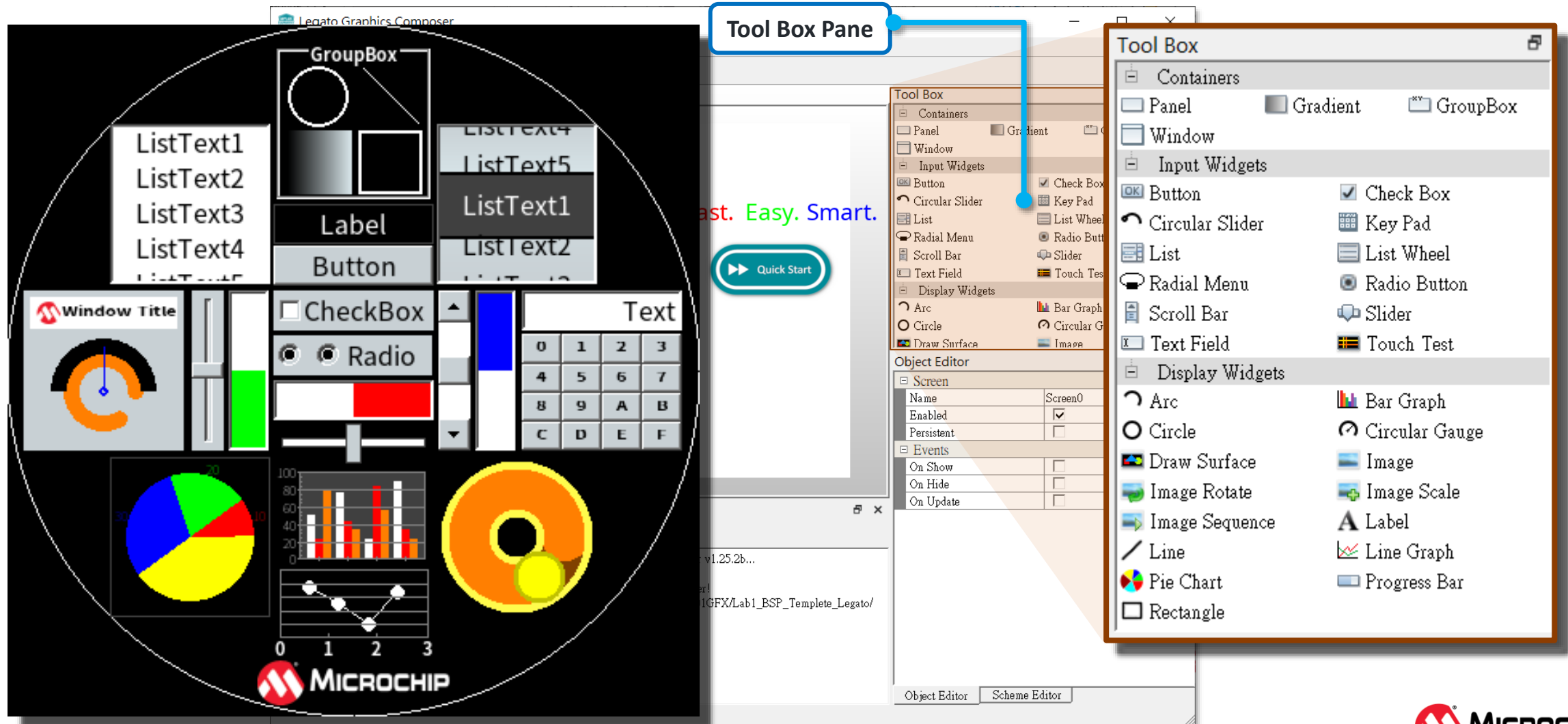
Legato Graphic Composer

- Legato Graphic Composer GUI.



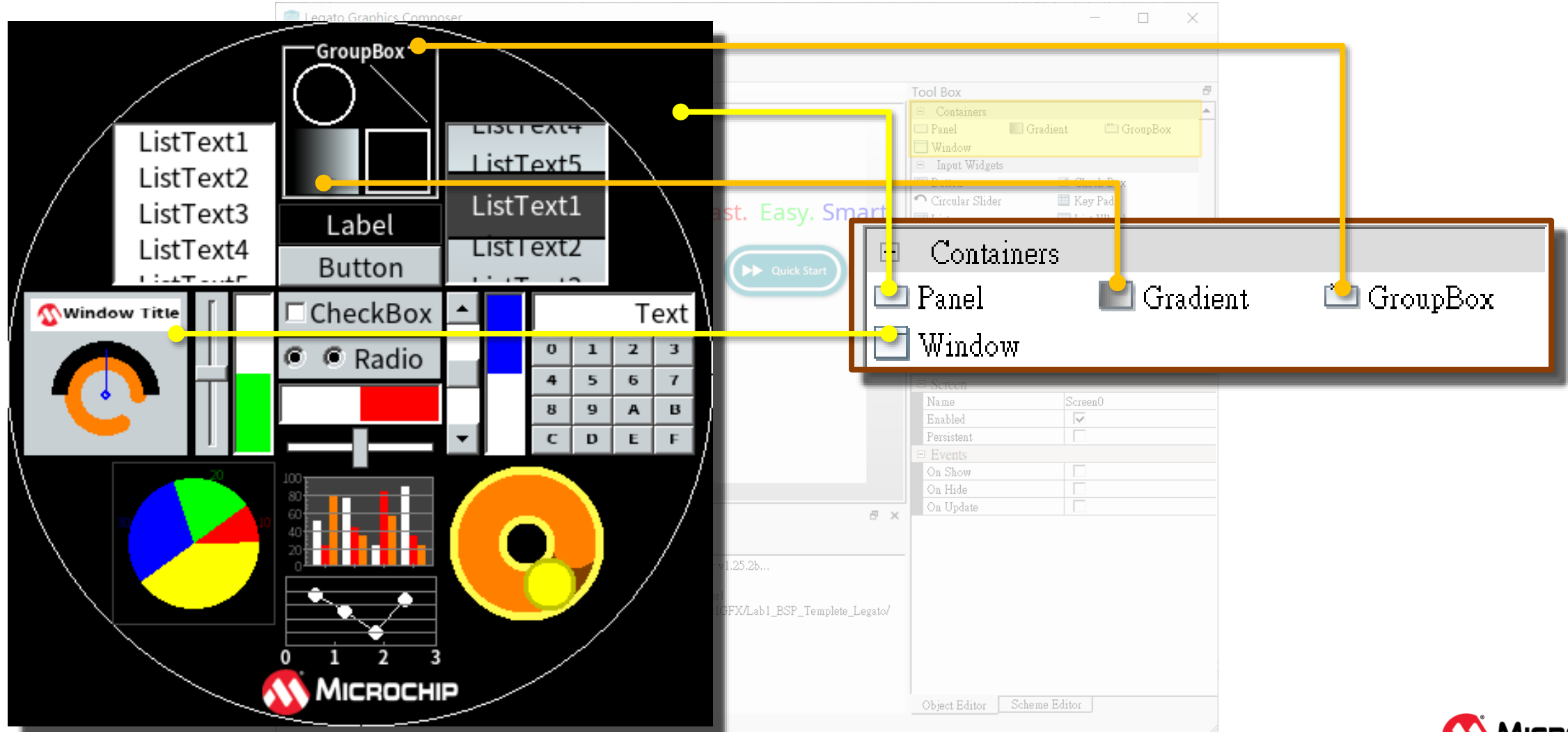
Legato Graphic Composer

- Legato Graphic Composer GUI.



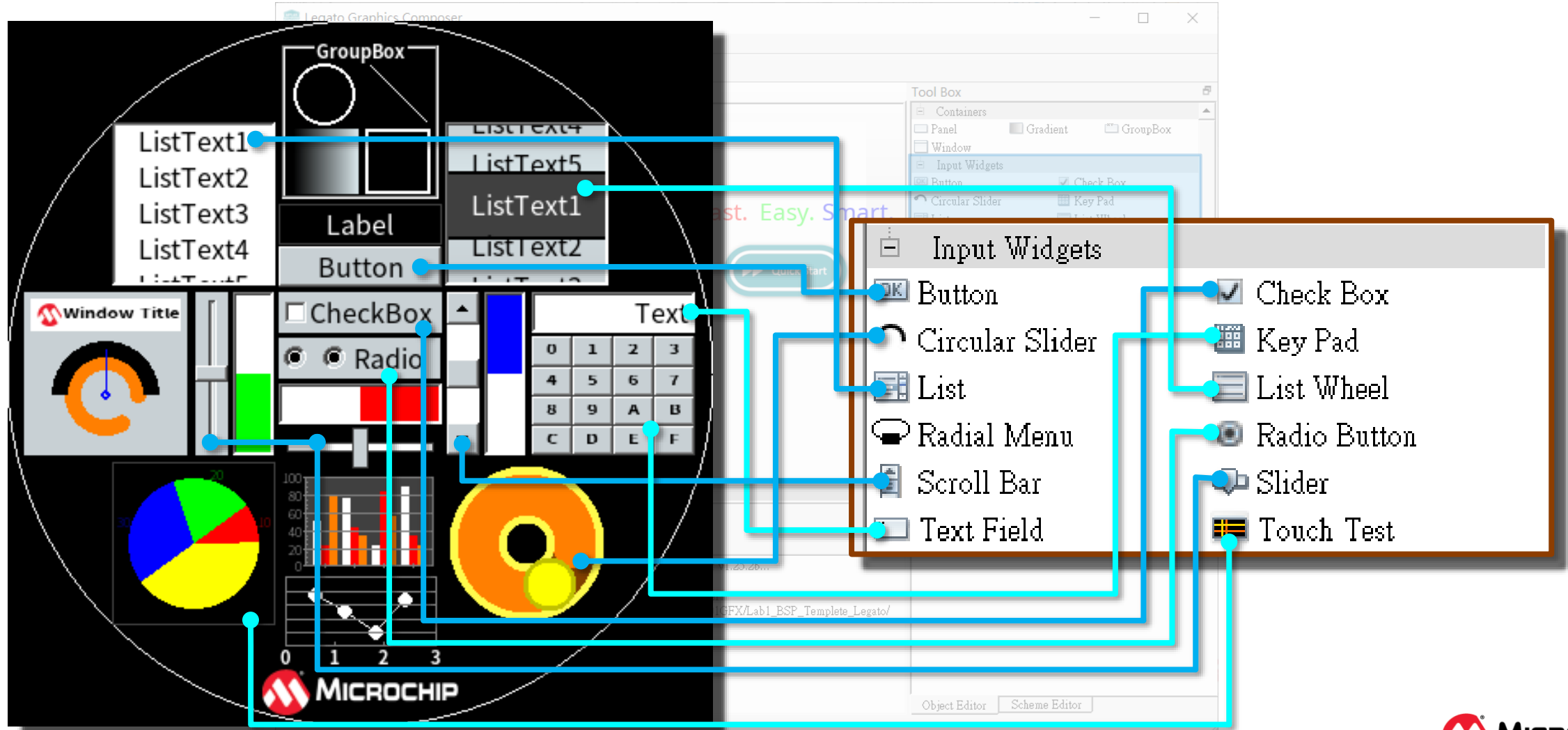
Legato Graphic Composer

- Legato Graphic Composer GUI.



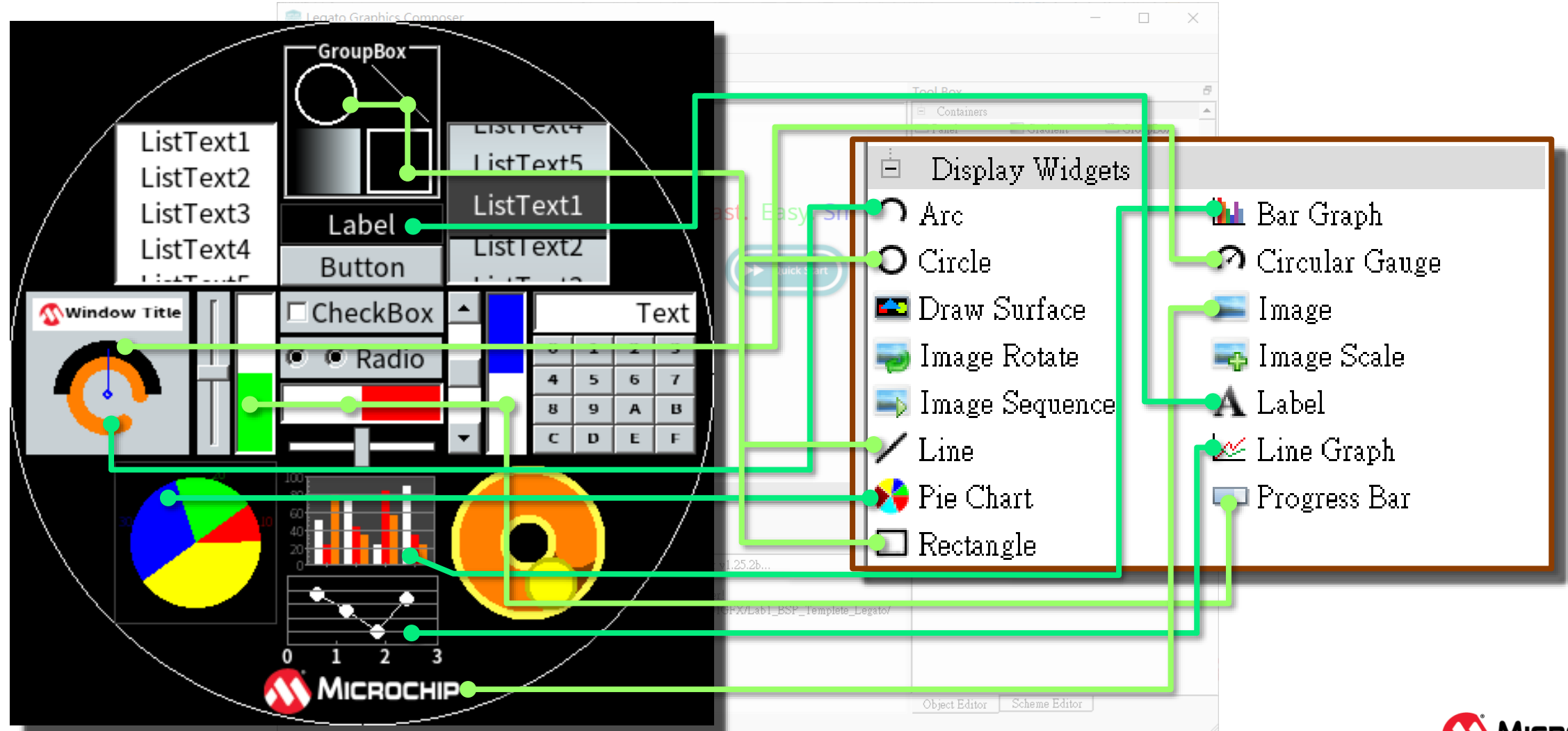
Legato Graphic Composer

- Legato Graphic Composer GUI.



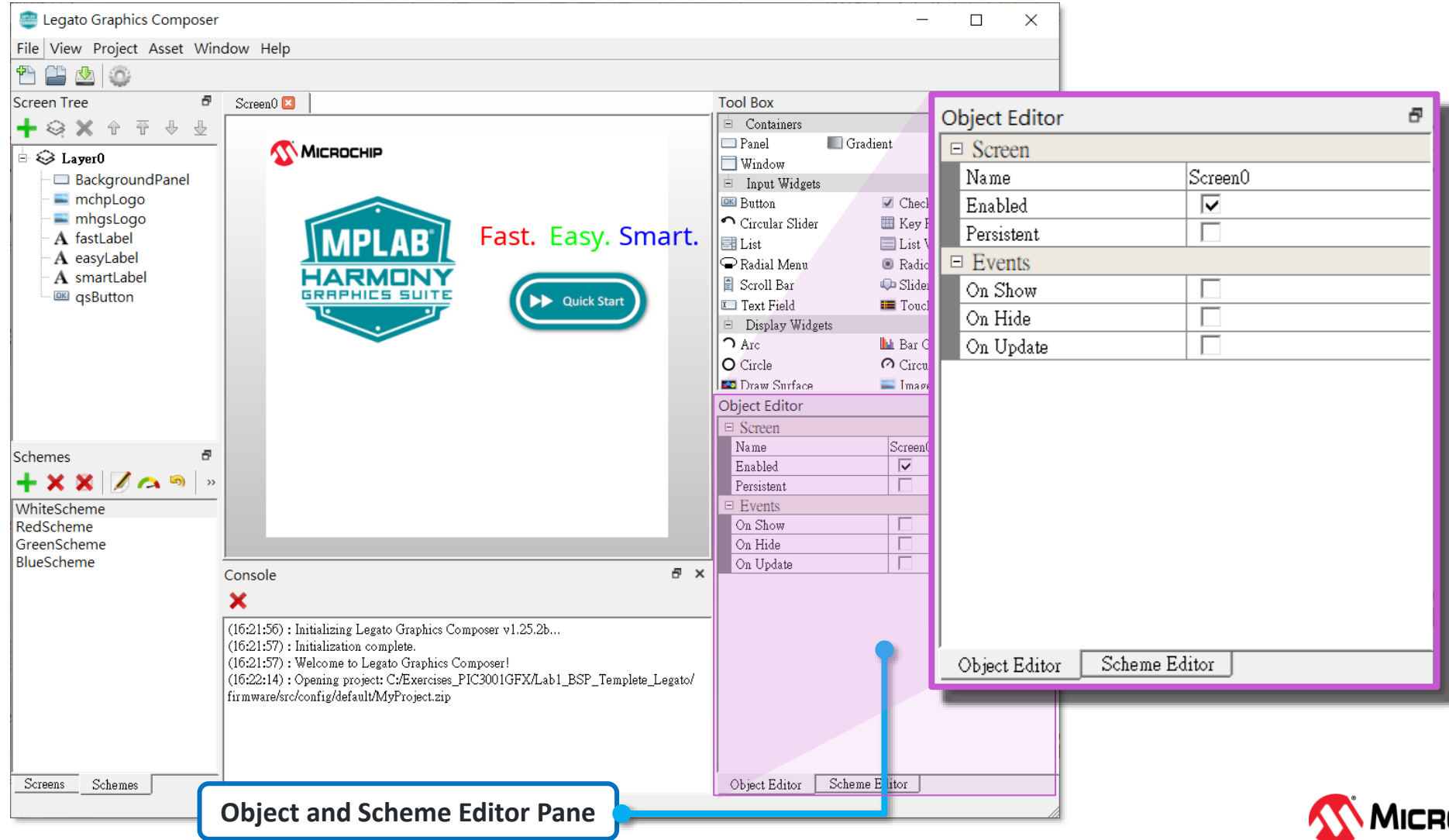
Legato Graphic Composer

- Legato Graphic Composer GUI.



Legato Graphic Composer

- Legato Graphic Composer GUI.



Legato Graphic Composer

- Legato Graphic Composer GUI.

The screenshot shows the Legato Graphics Composer interface. The main window displays a design canvas with a Microchip logo and the text 'Fast. Easy. Smart.'. On the left, the 'Screen Tree' shows a hierarchy: Layer0 > BackgroundPanel > mchpLogo > mhgsLogo > fastLabel. On the right, the 'Tool Box' lists various UI components like Containers, Panel, Gradient, Window, Input Widgets, Button, Circular Slider, List, Radial Menu, Scroll Bar, Text Field, Display Widgets, Arc, Circle, Draw Surface, Bar Chart, and Image. The 'Object Editor' panel is open, showing properties for 'Screen0'.

Screen :
Basic Screen control.

Event:
Screen Event Callback generation.
OnShow(), OnHide() and OnUpdate().

Screen	
Name	Screen0
Enabled	☒
Persistent	☐

Events	
On Show	☐
On Hide	☐
On Update	☐

Object Editor | Scheme Editor

Log messages at the bottom: (16:21:56) : Initializing Legato Graphics Composer v1.25.2b... (16:21:57) : Initialization complete. (16:21:57) : Welcome to Legato Graphics Composer! (16:22:14) : Opening project: C:/Exercises_PIC3001GFX/Lab1_BSP_Template_Legato/firmware/src/config/default/MyProject.zip

Legato Graphic Composer

- Legato Graphic Composer GUI.

Widget :

Base Widget Properties.

Object Name, position, size, scheme, visible control and alpha blending etc.

Button (Widget Type):

Widget specific properties.

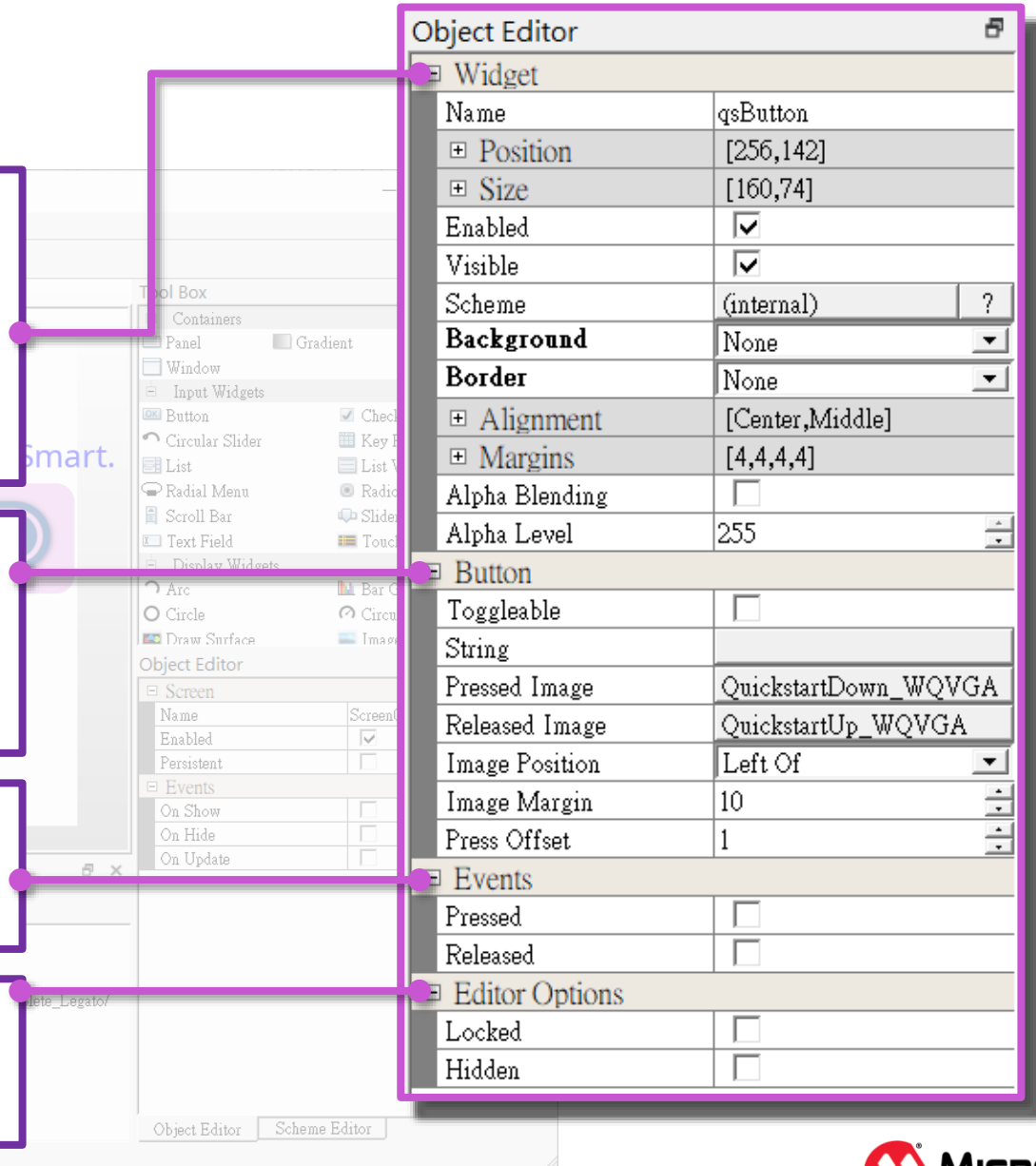
Operation, Text/String, images etc,

Event:

Widget event callback generation.

Editor Options:

Composer editor options.



Legato Graphic Composer

- Legato Graphic Composer GUI.

The screenshot displays the Legato Graphic Composer interface. The main workspace shows a 'Quick Start' button with a dashed bounding box. Dimensions are indicated: (256,142) for position and 160x74 for size. A console window at the bottom shows the following log messages:

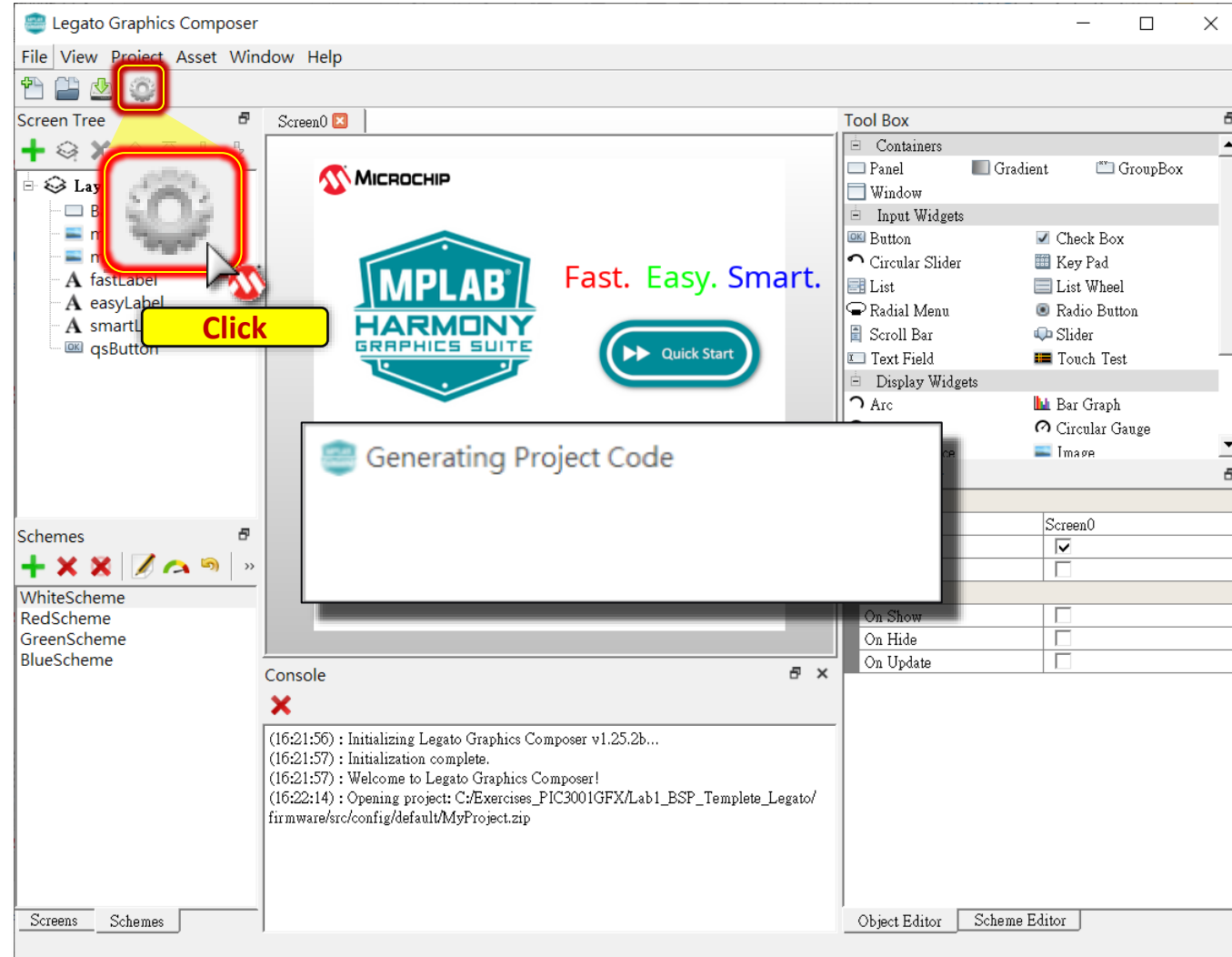
```
(16:21:56) : Initializing Legato  
(16:21:57) : Initialization complete  
(16:21:57) : Welcome to Legato  
(16:22:14) : Opening project: C:\firmware\src\config\default\MyP
```

The 'Object Editor' panel on the right shows the properties for the 'qsButton' widget:

Object Editor	
Widget	
Name	qsButton
Position	[256,142]
Size	[160,74]
Enabled	☒
Visible	☒
Scheme	(internal) ?
Background	None
Border	None
Alignment	[Center,Middle]
Margins	[4,4,4,4]
Alpha Blending	☐
Alpha Level	255
Button	
Toggleable	☐
String	
Pressed Image	QuickstartDown_WQVGA
Released Image	QuickstartUp_WQVGA
Image Position	Left Of
Image Margin	10
Press Offset	1
Events	
Pressed	☐
Released	☐
Editor Options	
Locked	☐
Hidden	☐

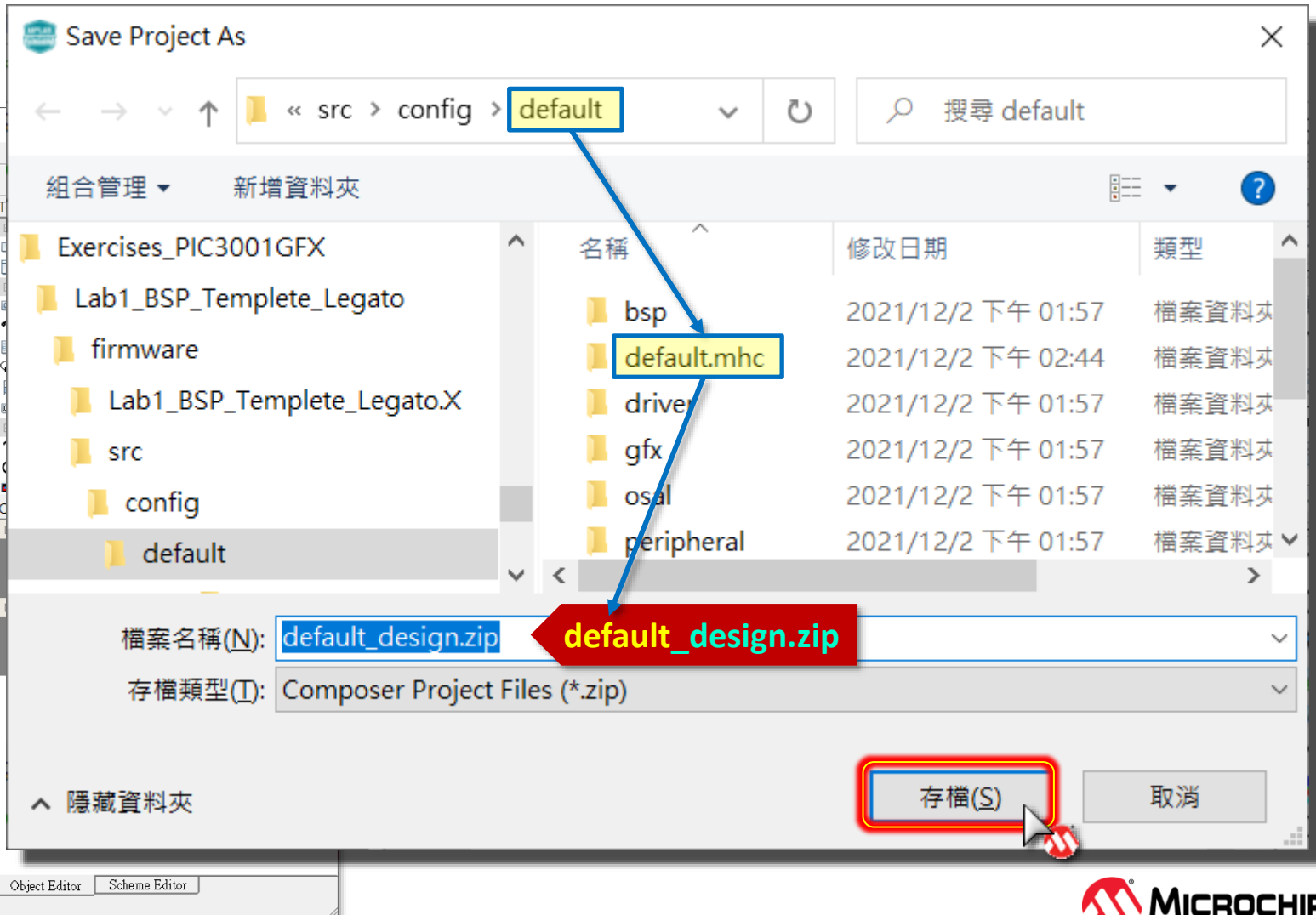
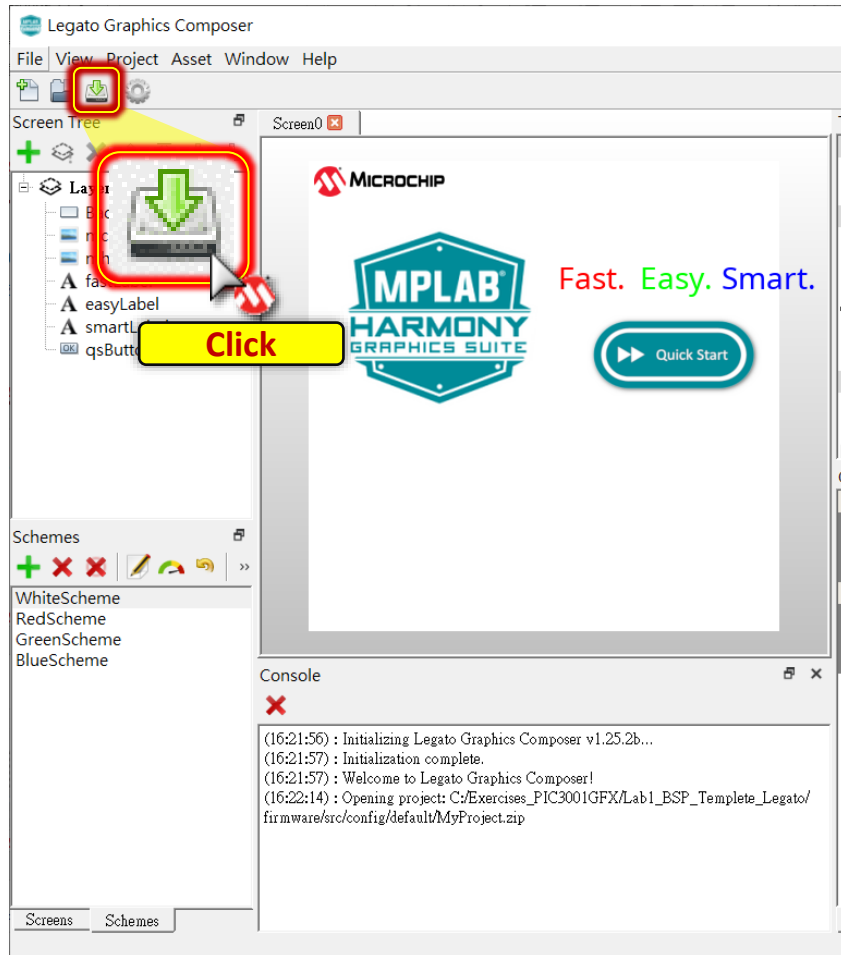
Legato Graphic Composer

- Generate Graphic Composer Design code to current project.



Legato Graphic Composer

- Save Graphic Composer Design to **[Config Name]_design.zip**
Ex. **default_design.zip**



Legato Graphic Composer

- Return to Harmony and Generate Code for Template Project.

Please remember this step and will never remind again later

MPLAB X IDE v6.00 - Lab1

File Edit MHC View Navigate Source Refactor Production Debug Team Tools Window Help

Projects Files Active Components*

Project Graph*

View: Root

System Device Family Pack (DFP) PIC32M7 DA Board BSP

TIME

Term Serv

Core Serv

S_TIME

Peripheral Library I2C

Driver Core Service

Service Layer

USB DEVICE

SOLE Service

Service

0

Build, Load, Run

hammer

0000, end address

Generating Project...

Task Type	Remaining	Total
File Markup	0	16
File Copy	0	16
Libraries	0	0
Settings	0	7
Source Paths	0	0

Project generation complete!

100%

Please Wait...

Generate Project

1. Configure Generation Settings

Generate Settings

Merge Strategy USER_ALL

(Mouse over a property for detailed help)

2. View Warnings

Type	Description
------	-------------

3. Click Generate

Default linker script is added to the project.
Disable it for the project that require custom linker script.

Generate Cancel

Legato Graphic Composer

- Try program to make sure the connection and check result.

Please remember this step and will never remind again later

The screenshot displays the MPLAB X IDE v6.00 interface. The 'Projects' pane on the left shows the project structure for 'PIC32MZ_DA_Radial'. The 'Available Components' pane shows various components like Graphics, Displays, Driver, GFX Core, Interface, Middleware, Processor, and Templates. The 'Project Graph' in the center shows the system components and their connections. The 'Output' pane on the right shows the programming progress and completion message.

Output x Console News

PIC32MZ_DA_Radial (Build, Load, ...) x ICD 4-PIC32MZ_DA_Radial x

The following memory area(s) will be programmed:
program memory: start address = 0x1d000000, end address = 0x1d000fff
configuration memory
boot config memory

Programming/Verify complete

Output x Console News

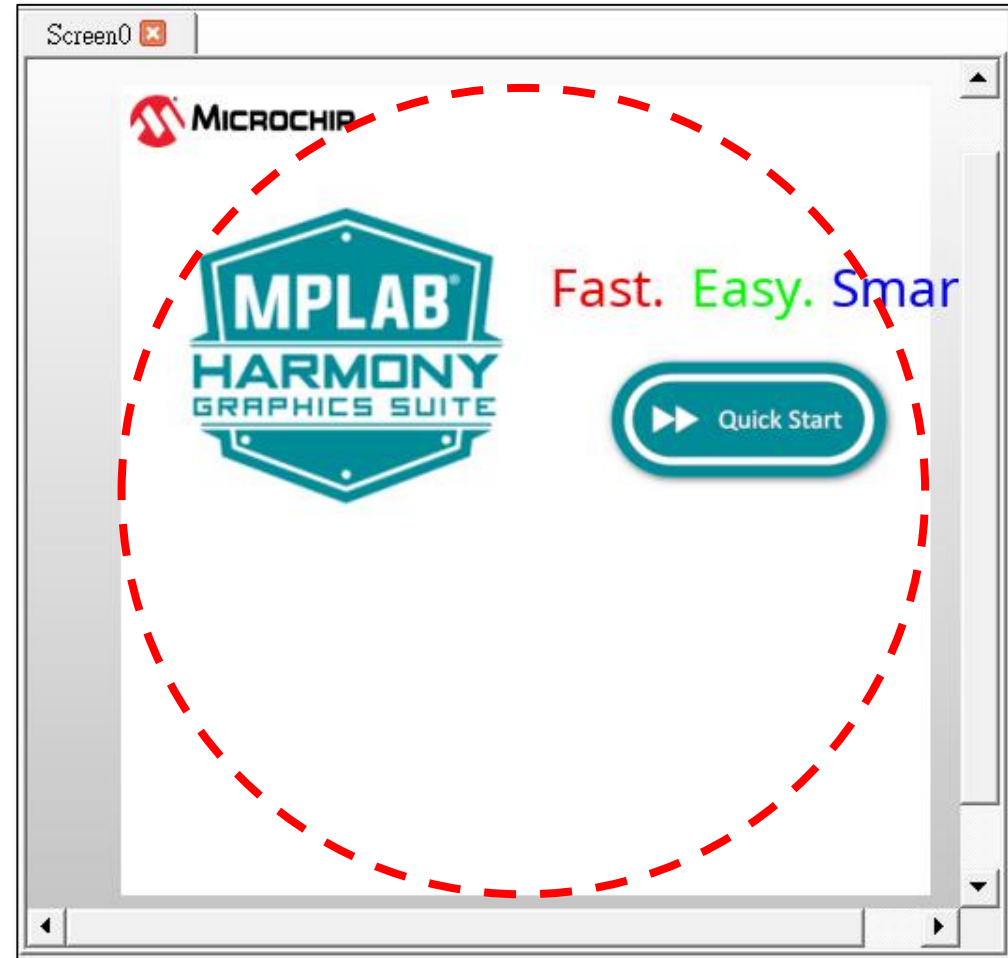
ICD 4-PIC32MZ_DA_Radial x PIC32MZ_DA_Radial (Build, Load, ...) x

configuration memory
boot config memory

Programming/Verify complete

Legato Graphic Composer

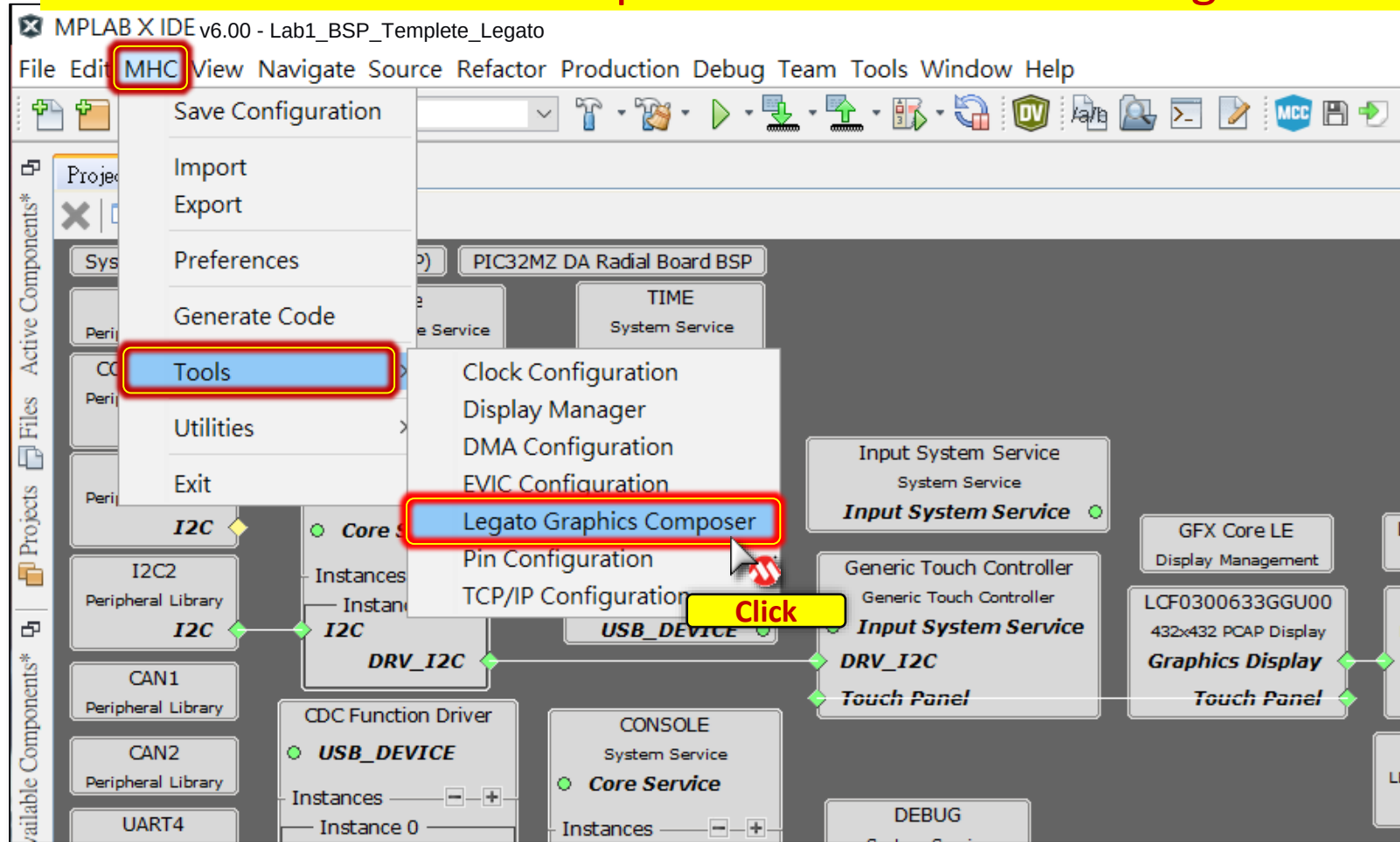
- The Template Result after program.



Legato Graphic Composer

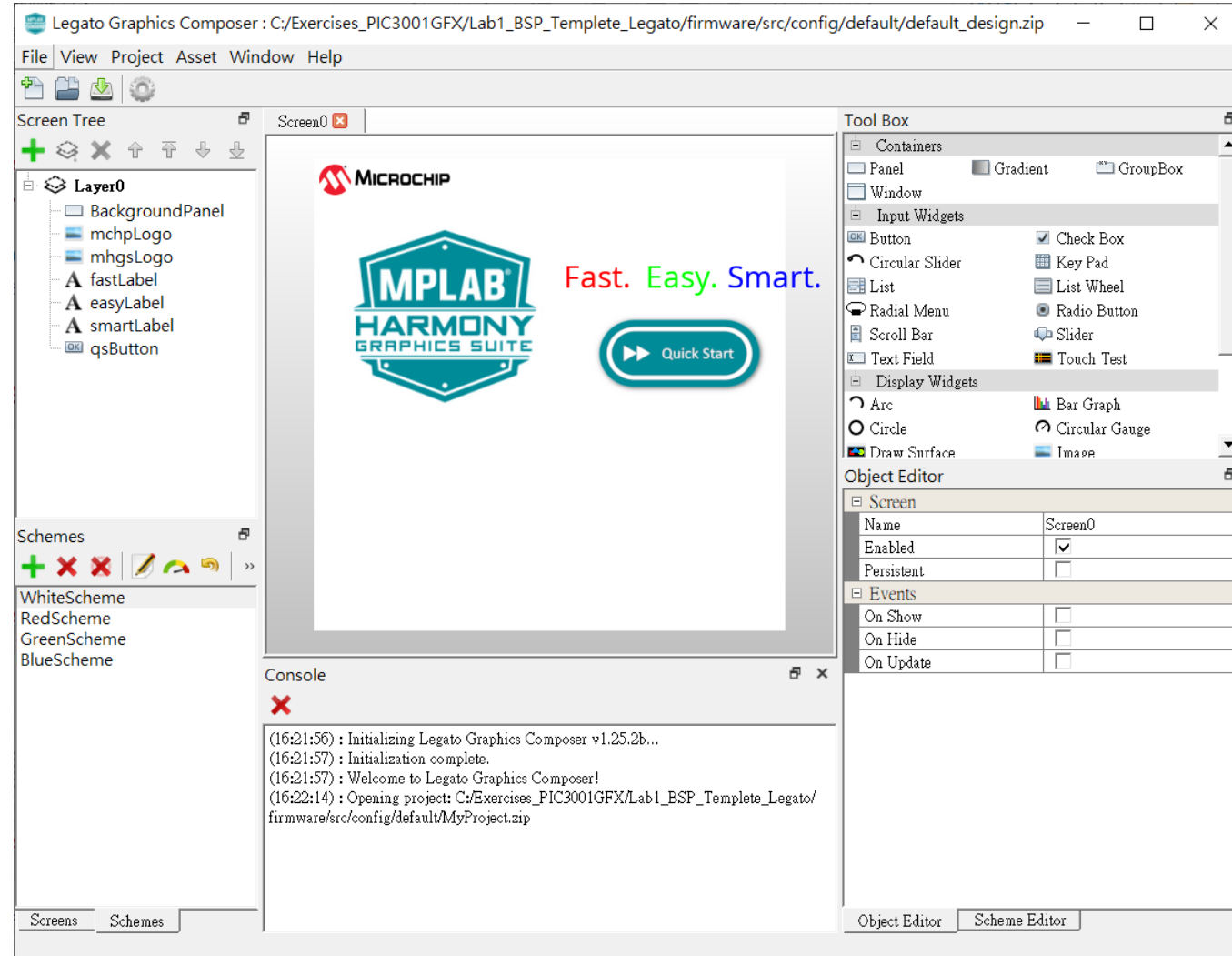
- Click [MHC] > [Tools] > [Legato Graphic Composer]

Please remember this step and will never remind again later



Legato Graphic Composer

- Return to Legato Graphic Composer.



Legato Graphic Composer

- How to execute standalone Legato Graphics Composer ?

`\HarmonyFramework\gfx\middleware\legato\composer\windows\`

The image illustrates the steps to execute the Legato Graphics Composer standalone application:

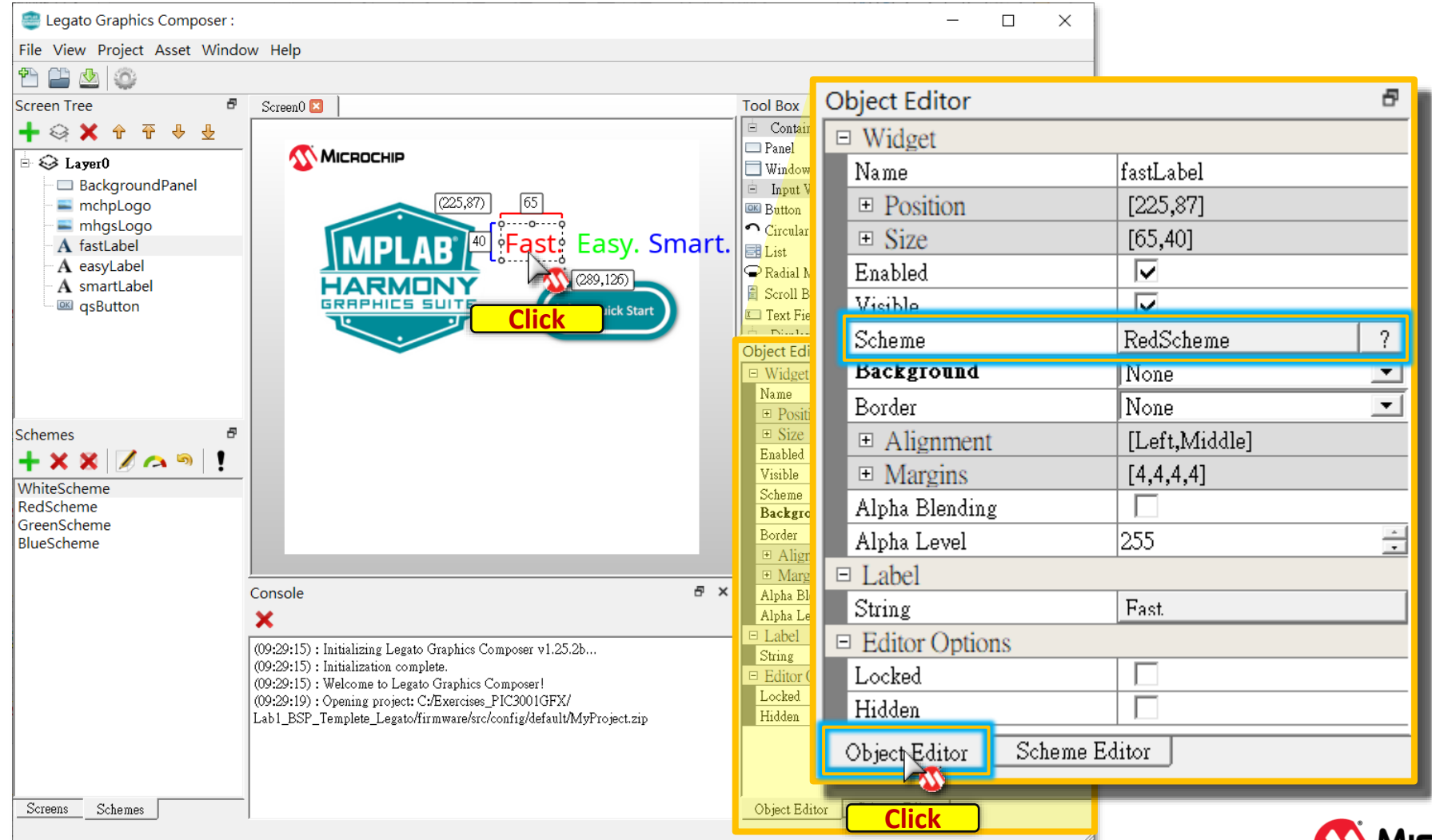
- File Explorer:** A directory listing shows files like `platforms`, `plugins`, `composer.exe` (highlighted with a red box and a "Double Click" callout), `composer.log`, and various DLL files.
- Legato Graphics Composer:** The application window is shown with the MPLAB Harmony Graphics Suite logo. It displays a "Welcome to the Legato interface design application" message and options to "Create a new project using..." or "Load an existing project..." (the latter is highlighted with a red box and a "Click" callout).
- Open Project Dialog:** A file selection dialog is open, showing a tree view of project folders. The file `default_design.zip` is selected (highlighted with a red box and a "Click" callout). The "File name" field at the bottom shows `default_design.zip`, and the "Files of type" is set to `Composer Project Files (*.zip)`. The "Open" button is highlighted with a red box and a "Click" callout.

Schemes

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Schemes

- Click on widget could set its “Scheme” in Object Editor.



Schemes

- Click on [?] of Scheme could show the help.

The screenshot displays the Legato Graphics Composer interface. A 'Label Scheme Helper' window is open, showing a diagram of a label's components: 'Outer Top/Left Border HighlightLight', 'Text', 'Background Base', 'Line Border ShadowDark', and 'Bottom/Right Border Shadow (Inner) ShadowDark (Outer)'. The 'Object Editor' window is also open, showing the 'Widget' properties for a 'fastLabel' object. The 'Scheme' dropdown is set to 'RedScheme', and a red box highlights the '?' icon next to it with a 'Click' label. The main canvas shows a 'Fast' label with a 'Quick Start' button.

Label Scheme Helper

- Outer Top/Left Border HighlightLight
- Text
- Background Base
- Line Border ShadowDark
- Bottom/Right Border Shadow (Inner) ShadowDark (Outer)

Object Editor

Widget	
Name	fastLabel
Position	[225,87]
Size	[65,40]
Enabled	☒
Visible	☒
Scheme	RedScheme ?
Background	
Border	None
Alignment	[Left,Middle]
Margins	[4,4,4,4]
Alpha Blending	☐
Alpha Level	255
Label	
String	Fast.
Editor Options	
Locked	☐
Hidden	☐

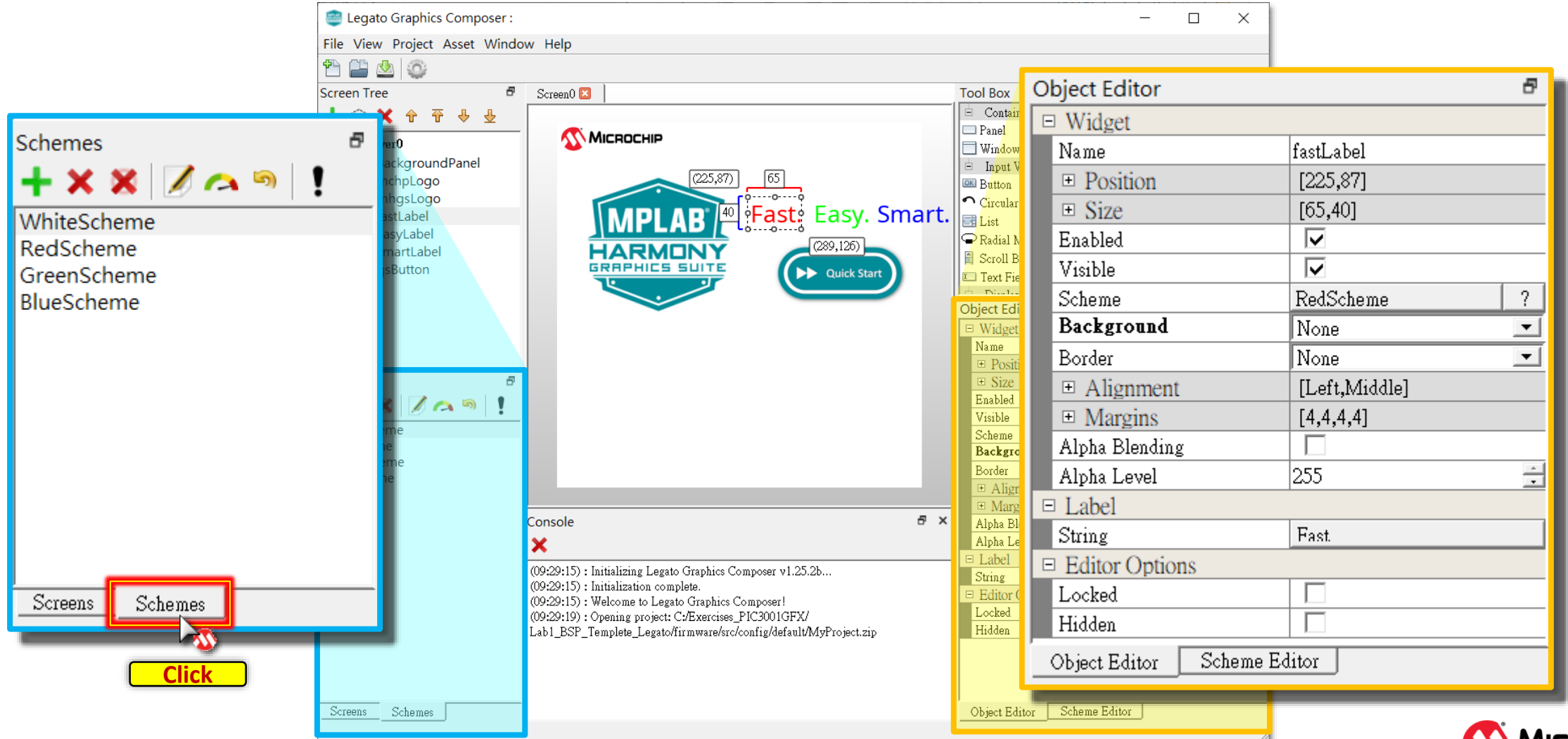
Click

Legato Graphics Composer v1.25.2b...

(09:29:15) : Welcome to Legato Graphics Composer!
(09:29:19) : Opening project: C:/Exercises_PIC3001GFX/
Lab1_BSP_Template_Legato/firmware/src/config/default/MyProject.zip

Schemes

- Check “Screen and Scheme Pane” and select to “Schemes”.



Schemes

- Select “RedScheme” in Schemes Pane to show Scheme Editor.

The screenshot displays the Legato Graphics Composer interface. The Schemes Pane on the left lists four schemes: WhiteScheme, RedScheme (highlighted with a yellow border and a red mouse cursor), GreenScheme, and BlueScheme. A yellow button labeled "Click" is positioned below the RedScheme entry. The main canvas shows a project preview with a Microchip logo, an MPLAB Harmony Graphics Suite logo, and a "Quick Start" button. The Scheme Editor on the right is open, showing the configuration for RedScheme. It includes a table for the Color Table with various color settings.

Scheme Editor Configuration:

Scheme	
Name	RedScheme
Color Mode	RGBA 8888
Color Table	
Base	Red
Highlight	Light Gray
Highlight Light	Light Gray
Shadow	Dark Gray
Shadow Dark	Dark Gray
Foreground	Red
Foreground Inactive	Light Gray
Foreground Disabled	Dark Gray
Background	Light Gray
Background Inactive	Light Gray
Background Disabled	Dark Gray
Text	Red
Text Highlight	Blue
Text Highlight Text	Blue
Text Inactive	Light Gray
Text Disabled	Dark Gray

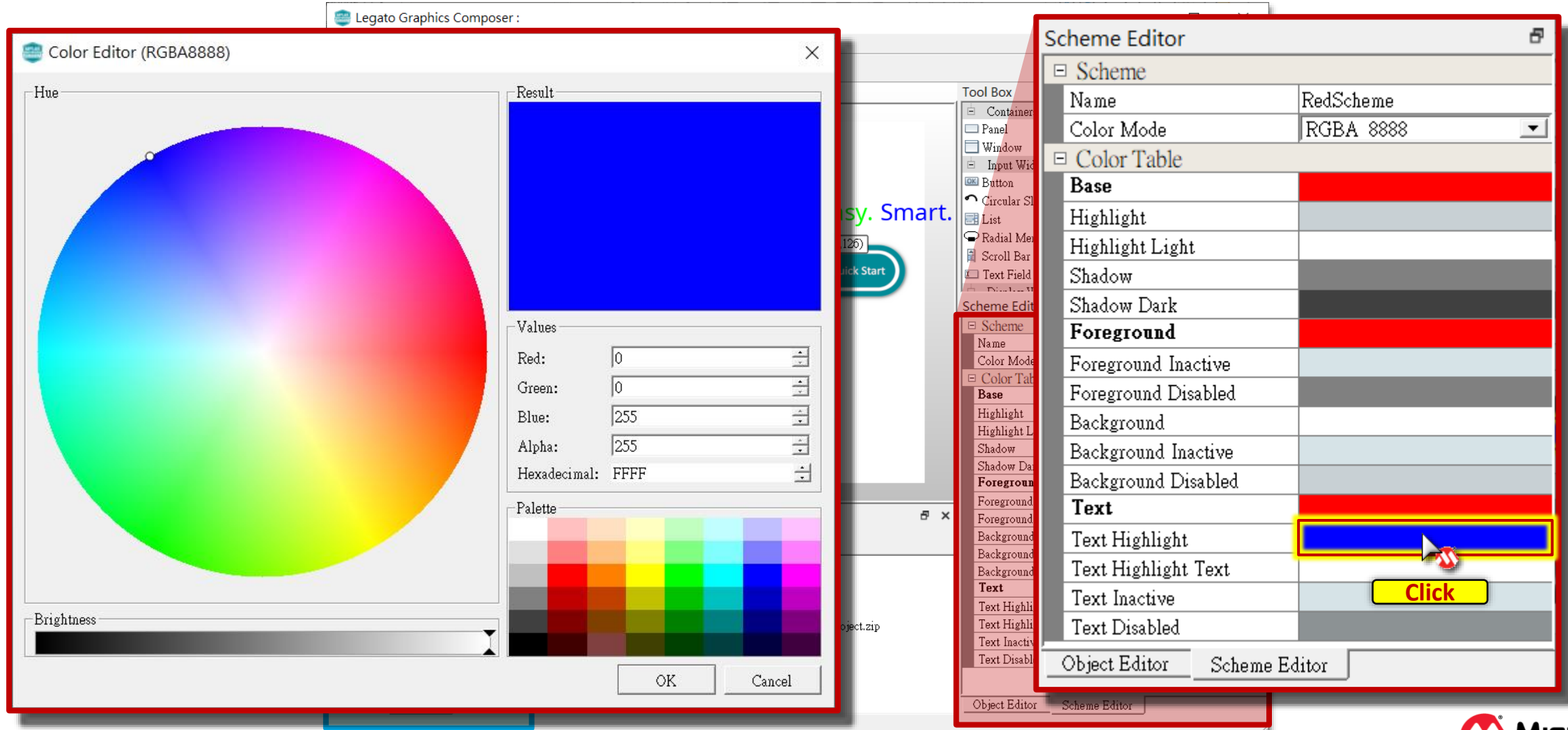
At the bottom of the Scheme Editor, there are two tabs: "Object Editor" and "Scheme Editor" (highlighted with a yellow border and a red mouse cursor). A yellow button labeled "Click" is positioned below the Scheme Editor tab.

Console Log:

```
(09:29:15) : Initializing Legato Graphics Composer v1.25.2b...
(09:29:15) : Initialization complete.
(09:29:15) : Welcome to Legato Graphics Composer!
(09:29:19) : Opening project: C:/Exercises_PIC3001GFX/
Lab1_BSP_Template_Legato/firmware/src/config/default/MyProject.zip
```

Schemes

- Click on color in Scheme Editor could show the Color Editor.



Schemes

- Manage Schemes for your Graphic Design.

The screenshot displays the Legato Graphics Composer interface. On the left, the 'Schemes' panel lists four schemes: WhiteScheme, RedScheme (selected), GreenScheme, and BlueScheme. Above this list are icons for creating, deleting, editing, and setting a default scheme. A 'Reset Scheme' button is also present. At the bottom of the panel, a 'Click' button is shown. The main canvas displays a graphic design for 'MPLAB HARMONY GRAPHICS SUITE' with a 'Quick Start' button. The 'Scheme Editor' is open on the right, showing the 'RedScheme' configuration. It includes a 'Color Table' with various color swatches and a 'Text' section with color swatches. The 'Scheme Editor' has tabs for 'Object Editor' and 'Scheme Editor'.

Delete Scheme

Edit Scheme

Set Default Scheme

Reset Scheme

Delete All Schemes

Create New Scheme

Click

Scheme Editor

Scheme	
Name	RedScheme
Color Mode	RGBA 8888
Color Table	
Base	
Highlight	
Highlight Light	
Shadow	
Shadow Dark	
Foreground	
Foreground Inactive	
Foreground Disabled	
Background	
Background Inactive	
Background Disabled	
Text	
Text Highlight	
Text Highlight Text	
Text Inactive	
Text Disabled	

Object Editor Scheme Editor

Schemes

- Schemes Acceleration for Anti-Aliased Text. (GFX v3.11.x)

The screenshot shows the Legato Graphics Composer interface. The 'Schemes' panel on the left lists 'WhiteScheme', 'RedScheme' (selected), 'GreenScheme', and 'BlueScheme'. The 'Scheme Acceleration Table Configuration (RedScheme)' dialog is open, showing a table of color names and their settings. The 'Fast.' text is displayed in a red, pixelated font. A color palette is shown in the foreground, illustrating the 'Look-up Table (Color Palette)' method.

Color Name	Color Value	Foreground	Background
Base		☐	☒
Highlight		☐	☐
Highlight Light		☐	☐
Shadow		☐	☐
Shadow Dark		☐	☐
Foreground		☐	☐
Foreground Inactive		☐	☐
Foreground Disabled		☐	☐
Background		☐	☐
Background Inactive		☐	☐
Background Disabled		☐	☐
Text		☒	☐
Text Highlight		☐	☐
Text Highlight Text		☐	☐
Text Inactive		☐	☐
Text Disabled		☐	☐

Alpha Blending

Look-up Table (Color Palette)

Schemes

- Try build a colorful gradient List Wheel widget.

The screenshot displays the Legato Graphics Composer interface with the following components:

- List Wheel Scheme Helper:** A diagram showing the widget's structure with labels for various states and areas: Outer Top/Left Border ShadowDark (Inner) Shadow (Outer), Active Area Foreground, Gradient End BackgroundDisabled, Unselected Text Text, Text, Gradient Start BackgroundInactive, Selected Text TextHighlightText, Indicator Gradient End ForegroundDisabled, Indicator Gradient Start ForegroundInactive, Indicator Lines Foreground, and Bottom/Right Border HighlightLight (Inner) HighlightLight (Outer).
- Screen Tree:** A hierarchical view of the design elements, including Layer0, BackgroundPanel, mchpLogo, mhgsLogo, fastLabel, easyLabel, smartLabel, and ListWheelWidget0 (highlighted).
- Schemes:** A list of available schemes, with ListWheelScheme (highlighted) selected.
- Screen0:** The main design canvas showing the MPLAB Harmony Graphics Suite logo and a 'Quick Start' button. A 'List Wheel' widget is placed on the canvas, displaying a gradient of colors (purple, blue, green, yellow, orange, red) with the text 'Smart.', 'Easy.', and 'Fast.' (partially visible).
- Tool Box:** A collection of widgets, including Circular Slider, List, Radial Menu, and List Wheel (highlighted).
- Scheme Editor:** A table defining the colors for the ListWheelScheme.
- Console:** A log of system messages, including 'Initializing Legato Graphics Composer v1.25.2b...' and 'Welcome to Legato Graphics Composer!'.

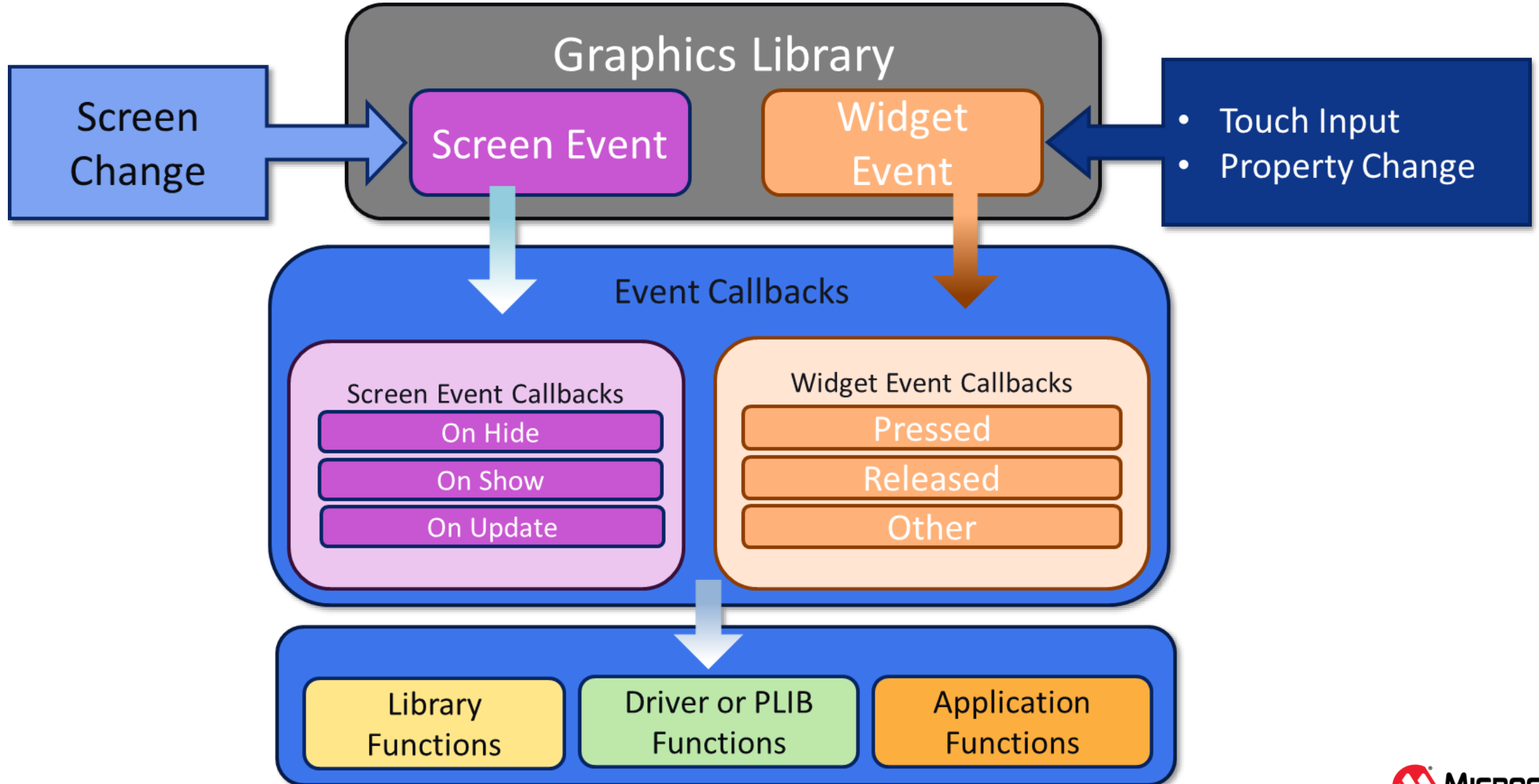
Scheme Editor	
Scheme	
Name	ListWheelScheme
Color Mode	RGBA 8888
Color Table	
Base	
Highlight	
Highlight Light	
Shadow	
Shadow Dark	
Foreground	
Foreground Inactive	
Foreground Disabled	
Background	
Background Inactive	
Background Disabled	
Text	
Text Highlight	
Text Highlight Text	
Text Inactive	
Text Disabled	

GFX v3.12.2 issues : Reopen design will miss scheme color of List Wheel Widget in Graphics Composer

Event Handling

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Event Handling



Event Handling

- The Event Manager is an interface to enable/disable event callbacks for a screen as well as for all the even-capable widgets in that screen.

The screenshot illustrates the Legato Graphics Composer interface, specifically the Event Manager window, which is used to configure event callbacks for a screen and its widgets.

Legato Graphics Composer: The top window shows the menu bar (File, View, Project, Asset, Window, Help) and the Project Settings dialog. The Event Manager button is highlighted in the Project Settings dialog, and a yellow callout box labeled "Event Manager" points to it.

Event Manager: The central window displays a tree view of the screen hierarchy. The "Screen (Screen0)" is selected, and its event callbacks are configured as follows:

- Screen (Screen0) Events:**
 - ☒ On Show
 - ☐ On Hide
 - ☒ On Update
- Layer (Layer0) and Layer (Layer1):** These layers contain various widgets, including ListWidget, ListWheelWidget, SliderWidget, ButtonWidget, CheckBoxWidget, RadioButtonWidget, ScrollBarWidget, KeyPadWidget, TouchTestWidget, and CircularSliderWidget.

Object Editor: The right-hand window shows the configuration for the selected widget, "CircularSliderWidget0".

- Screen (Screen0) Object Editor:**

Screen	
Name	Screen0
Enabled	☒
Persistent	☐
Events	
On Show	☒
On Hide	☐
On Update	☒
- Widget Object Editor:**

Widget	
Name	CircularSliderWidget0
Position	[262, 270]
Circular Slider	
Events	
Value Changed	☒
Pressed	☐
Released	☐
Editor Options	
Locked	☐
Hidden	☐

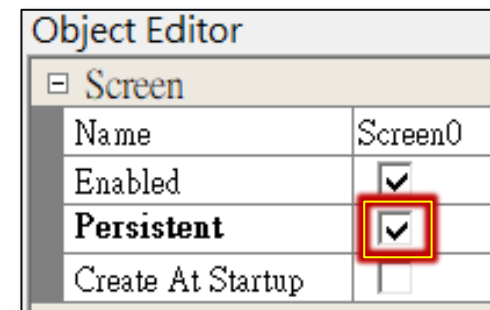
Red arrows indicate the flow of configuration from the Event Manager to the Screen Object Editor, and blue arrows indicate the flow from the Event Manager to the Widget Object Editor.

Microchip Logo: The Microchip logo is visible in the bottom right corner of the screenshot.

Event Handling

- **Screen Events Callbacks**

- **[Screen Name]_OnShow()**, ex : Screen0_OnShow()
 - Called right after the screen and children widgets are created.
 - Use to allocate and initialize run-time screen objects. (e.g., strings)
 - Use to restore widget states.
- **[Screen Name]_OnUpdate()**, ex : Screen0_OnUpdate()
 - Periodically called before the screen and children widgets are update and redrawn.
 - Use to manage screen application states.
 - Use to set widget properties.
- **[Screen Name]_OnHide()**, ex : Screen0_OnHide()
 - Called right before the screen and children widgets are destroyed/freed.
 - Use to delete and free allocations for run-time screen objects.
 - Use to save widget states.
 - Screen marked '**persistent**' are not destroyed when hidden and dynamic allocations are preserved in memory.



Event Handling

- **Widget Events Callbacks**

- Naming rule :

event_**[Screen Name]**_**[WidgetName]**_On**[Action]**(*<leWidgetType>* ptr, ...*)

- Examples :

- Button:

```
void event_Screen0_ButtonWidget0_OnPressed(leButtonWidget* btn)
void event_Screen0_ButtonWidget0_OnReleased(leButtonWidget* btn)
```

- CheckBox:

```
void event_Screen0_CheckBoxWidget0_OnChecked(leCheckBoxWidget* btn)
void event_Screen0_CheckBoxWidget0_OnUnchecked(leCheckBoxWidget* btn)
```

- Radio Button:

```
void event_Screen0_RadioButtonWidget0_OnSelected(leRadioButtonWidget* btn)
void event_Screen0_RadioButtonWidget0_OnDeselected(leRadioButtonWidget* btn)
```

- TextField:

```
void event_Screen0_TextFieldWidget0_OnTextChanged(leTextFieldWidget* btn)
void event_Screen0_TextFieldWidget0_OnFocusChanged(leTextFieldWidget* btn, leBool state)
```

- ProgressBar:

```
void event_Screen0_ProgressBarWidget0_ValueChanged(leProgressBarWidget* wgt, uint32_t val)
```

- List:

```
void event_Screen0_ListWidget0_OnSelectionChanged(leListWidget* wgt, uint32_t idx, leBool selected)
```


Event Handling

- **Widget Events Callbacks (cont.)**

- Examples :

- ListWheel:

- ```
void event_Screen0_ListWheelWidget0_OnSelectionChanged(leListWheelWidget* wgt, int32_t idx)
```

- Slider:

- ```
void event_Screen0_SliderWidget0_OnValueChanged(leSliderWidget* scr)
```

- ScrollBar:

- ```
void event_Screen0_ScrollBarWidget0_OnValueChanged(leScrollBarWidget* scr)
```

- CircularSlider:

- ```
void event_Screen0_CircularSliderWidget0_OnValueChanged(leCircularSliderWidget* sld, int32_t val)
```

- ```
void event_Screen0_CircularSliderWidget0_OnPressed(leCircularSliderWidget* sld, int32_t val)
```

- ```
void event_Screen0_CircularSliderWidget0_OnReleased(leCircularSliderWidget* sld, int32_t val)
```

- CircularGauge:

- ```
void event_Screen0_CircularGaugeWidget0_OnValueChanged(leCircularGaugeWidget* btn, int32_t val)
```

- PieChart:

- ```
void event_Screen0_PieChartWidget0_OnPressed(lePieChartWidget* wgt, uint32_t idx)
```

- KeyPad:

- ```
void event_Screen0_KeyPadWidget0_OnKeyClick(leKeyPadWidget* wgt, leButtonWidget* cell, uint32_t row, uint32_t col)
```

- TouchTest:

- ```
void event_Screen0_TouchTestWidget0_OnPointAdded(leTouchTestWidget* tch, lePoint* pnt)
```

Event Handling

- Where and when to register and execute the event callbacks ?

`\firmware\src\config\default\gfx\legato\generated\screen\le_gen_screen_[Screen Name].c`

```
leResult screenShow_[Screen Name](void)
{ ...
```

```
[Screen Name]_[WidgetName]->fn->set[Action]EventCallback( [Screen Name]_[WidgetName], [Widget Event Callback] );
```

```
...
```

```
[Screen Name]_OnShow();
```

```
return LE_SUCCESS;
```

```
}
```

```
leResult screenUpdate_[Screen Name](void)
```

```
{ ...
```

```
[Screen Name]_OnUpdate();
```

```
}
```

```
leResult screenHide_[Screen Name](void)
```

```
{
```

```
[Screen Name]_OnHide();
```

```
...
```

```
}
```

*Example : le_gen_screen_**Screen0**.c*

```
leResult screenShow_Screen0(void)
```

```
{ ...
```

```
Screen0_qsButton->fn->setPressedEventCallback( Screen0_qsButton, event_Screen0_qsButton_OnPressed );
```

```
...
```

```
Screen0_OnShow();
```

```
return LE_SUCCESS;
```

```
}
```

```
leResult screenUpdate_Screen0(void)
```

```
{ ...
```

```
Screen0_OnUpdate();
```

```
}
```

```
leResult screenHide_Screen0(void)
```

```
{ ...
```

```
Screen0_OnHide();
```

```
}
```

Event Handling

- Where the Callback functions prototype and widgets extern ?

`\firmware\src\config\default\gfx\legato\generated\screen\le_gen_screen_[Screen Name].h`

```
...
#include "gfx/legato/legato.h"

#include "gfx/legato/generated/le_gen_scheme.h"
#include "gfx/legato/generated/le_gen_assets.h"

// screen member widget declarations
```

```
extern <leWidgetType>* [Screen Name]_[WidgetName];
```

```
...
// event handlers
// !!THESE MUST BE IMPLEMENTED IN THE APPLICATION CODE!!
```

```
void event_[Screen Name]_[WidgetName]_On[Action]( <leWidgetType>* ptr, ... );
```

```
...
// Screen Events:
void [Screen Name]_OnShow(void); // called when this screen is shown
void [Screen Name]_OnUpdate(void); // called when this screen is updated
void [Screen Name]_OnHide(void); // called when this screen is hidden
```

Example : `le_gen_screen_Screen0.h`

```
extern leButtonWidget* Screen0_qsButton;
```

```
void event_Screen0_qsButton_OnPressed(leButtonWidget* btn);
```

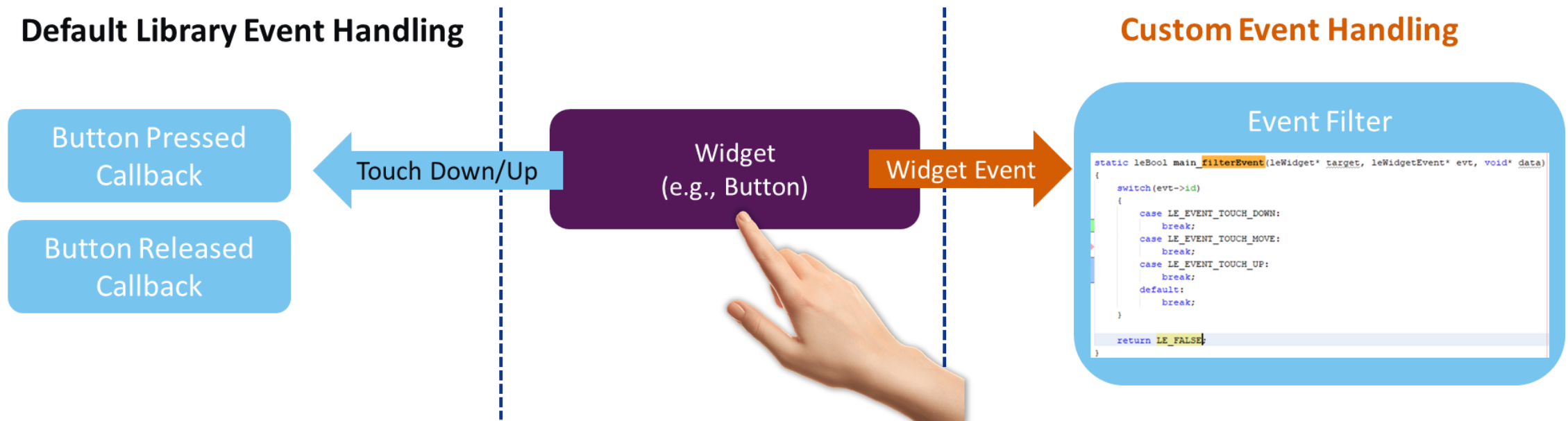
[Widget Event Callback]

```
void Screen0_OnShow(); // called when this screen is shown
void Screen0_OnUpdate(); // called when this screen is updated
void Screen0_OnHide(); // called when this screen is hidden
```

Event Handling

- **Event Filters - Custom Event Handling**

- Event filters can be used to add or 'override' default widget event handling.
- The allow events like Touch Down/Up/Move, Change Language, Widget Paint, Widget Move, Widget Resized, Widget Focus to be handled in Application Code.
- Event filters can also be used to 'mute' specific events from being handled.



Event Handling

- **Event Filters - Event ID** (legato_event.h)

```
typedef enum leEventID
```

```
{
```

```
    // internal events
```

```
    LE_EVENT_NONE,
```

```
    LE_EVENT_TOUCH_DOWN,
```

```
    LE_EVENT_TOUCH_UP,
```

```
    LE_EVENT_TOUCH_MOVE,
```

```
    LE_EVENT_LANGUAGE_CHANGED,
```

```
    // widget events
```

```
    LE_WIDGET_EVENT_PAINT = 100,
```

```
    LE_WIDGET_EVENT_MOVED,
```

```
    LE_WIDGET_EVENT_RESIZED,
```

```
    LE_WIDGET_EVENT_FOCUS_GAINED,
```

```
    LE_WIDGET_EVENT_FOCUS_LOST,
```

```
} leEventID;
```

```
/**< No event found. */
```

```
/**< Touch down event. */
```

```
/**< Touch up event. */
```

```
/**< Touch move event. */
```

```
/**< Language changed. */
```

```
/**< Paint event. */
```

```
/**< Moved event. */
```

```
/**< Resized event. */
```

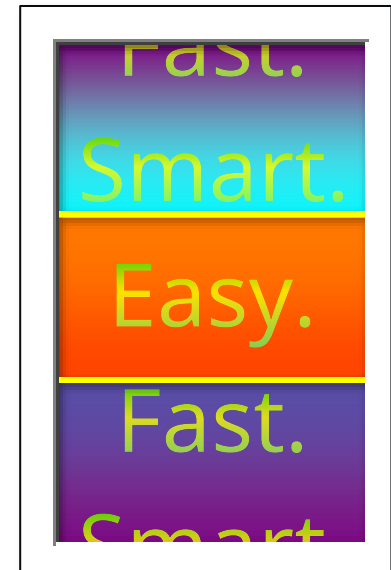
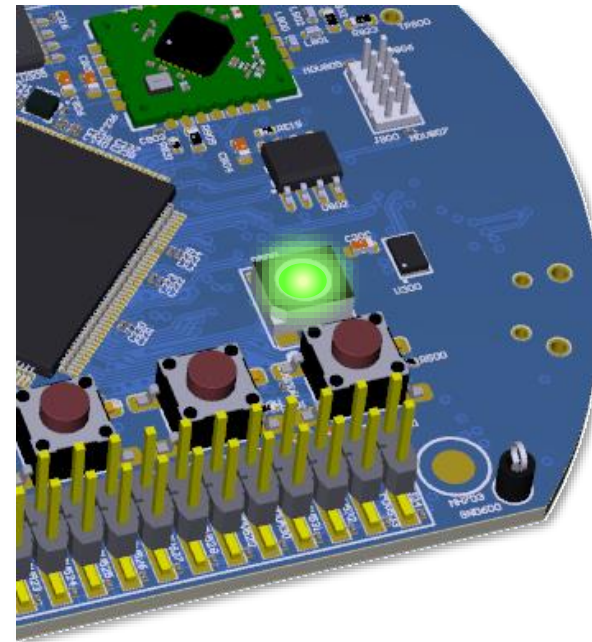
```
/**< Focus gained event. */
```

```
/**< Focus lost event. */
```

Event Handling

- Event Filters Example – Color RGB LED through List Wheel

```
leWidgetEventFilter app_WidgetEventFilter;  
  
leBool ListWheel_filterEvent(leWidget* target, leWidgetEvent* evt, void* data)  
{  
    uint32_t id = evt->id;  
    static uint32_t color = 0;  
  
    switch( id )  
    {  
    case LE_EVENT_TOUCH_DOWN:  
        RGB_LED_R_On(); RGB_LED_G_On(); RGB_LED_B_On(); break;  
    case LE_EVENT_TOUCH_UP:  
        RGB_LED_R_Off(); RGB_LED_G_Off(); RGB_LED_B_Off(); break;  
    case LE_EVENT_TOUCH_MOVE:  
    case LE_WIDGET_EVENT_PAINT:  
        (color&0x1?RGB_LED_R_On():RGB_LED_R_Off());  
        (color&0x2?RGB_LED_G_On():RGB_LED_G_Off());  
        (color&0x4?RGB_LED_B_On():RGB_LED_B_Off());  
        color = (color+1)%8; break;  
    }  
    return LE_FALSE; Must return LE_FALSE  
}  
  
void Screen0_OnShow(void)  
{  
    app_WidgetEventFilter.filterEvent = ListWheel_filterEvent;  
    app_WidgetEventFilter.data = NULL;  
    Screen0_ListWheelWidget0->fn->installEventFilter( Screen0_ListWheelWidget0, app_WidgetEventFilter );  
}
```

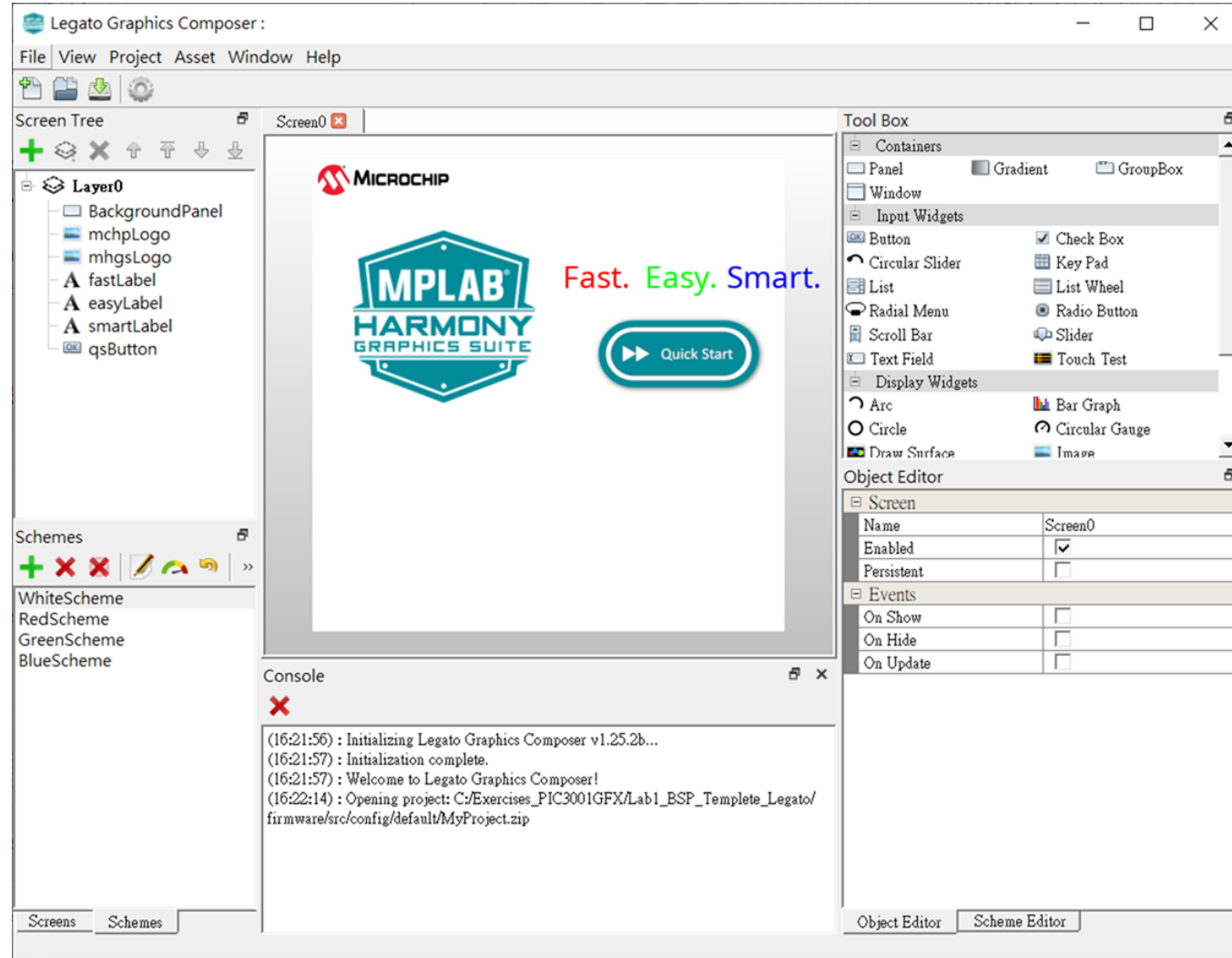


Button Widget Event

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

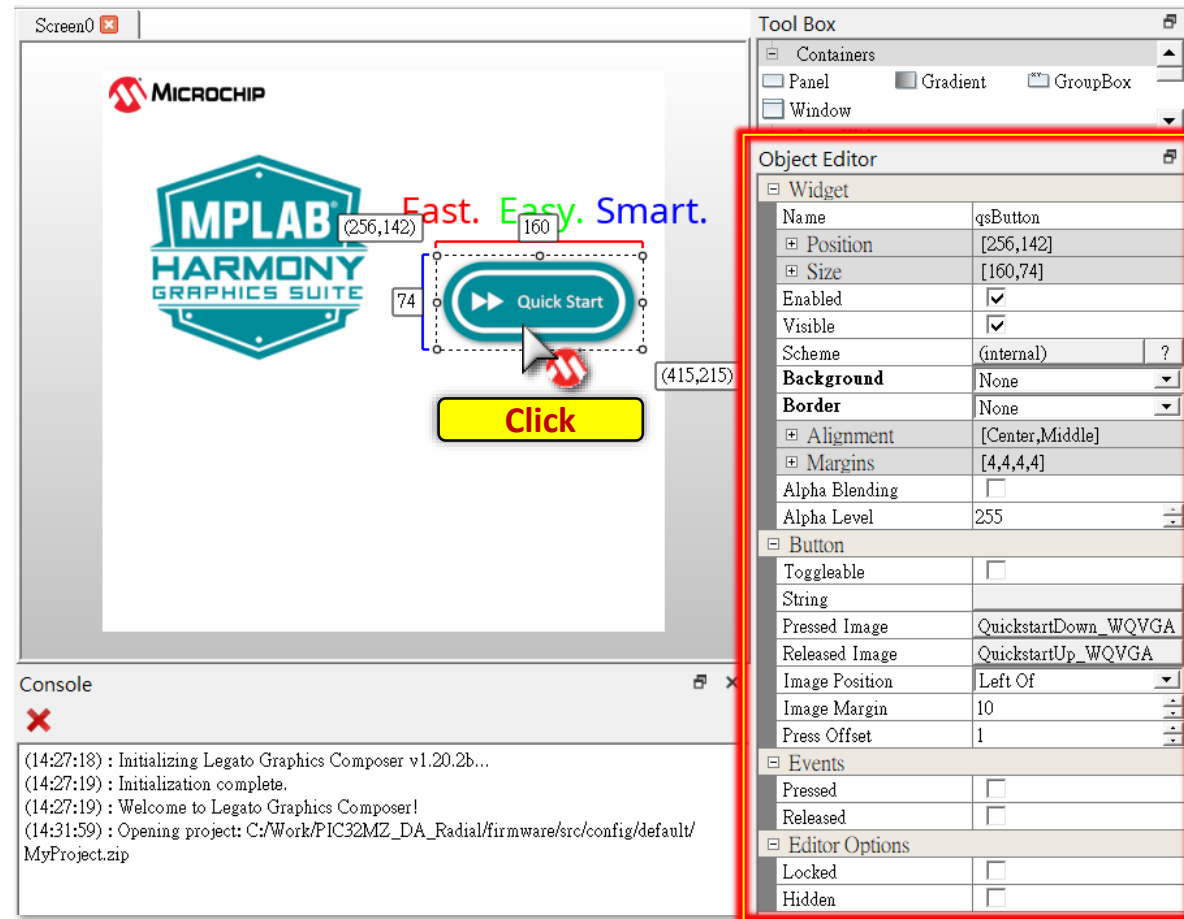
Button Widget Event

- Return to Legato Graphic Composer.



Button Widget Event

- Click on  Button Widget, the Object Editor window will show the configuration of widget.



Button Widget Event

- Enable **Pressed** and **Released** Events of Button widget.

The screenshot displays the Legato Graphics Composer interface. The main canvas shows a design with the Microchip MPLAB Harmony Graphics Suite logo and a 'Quick Start' button. The button's position is [256,142] and its size is [160,74]. The Object Editor on the right shows the widget's properties, including its name 'qsButton', position, size, and various styling options. The Events panel, highlighted with a red box, shows the 'Pressed' and 'Released' events enabled with checkmarks. A yellow callout box with the text 'Check Enable' points to the Events panel. The Console at the bottom shows the initialization logs for the Legato Graphics Composer.

Events Panel:

Event	Enabled
Pressed	☒
Released	☒

Object Editor:

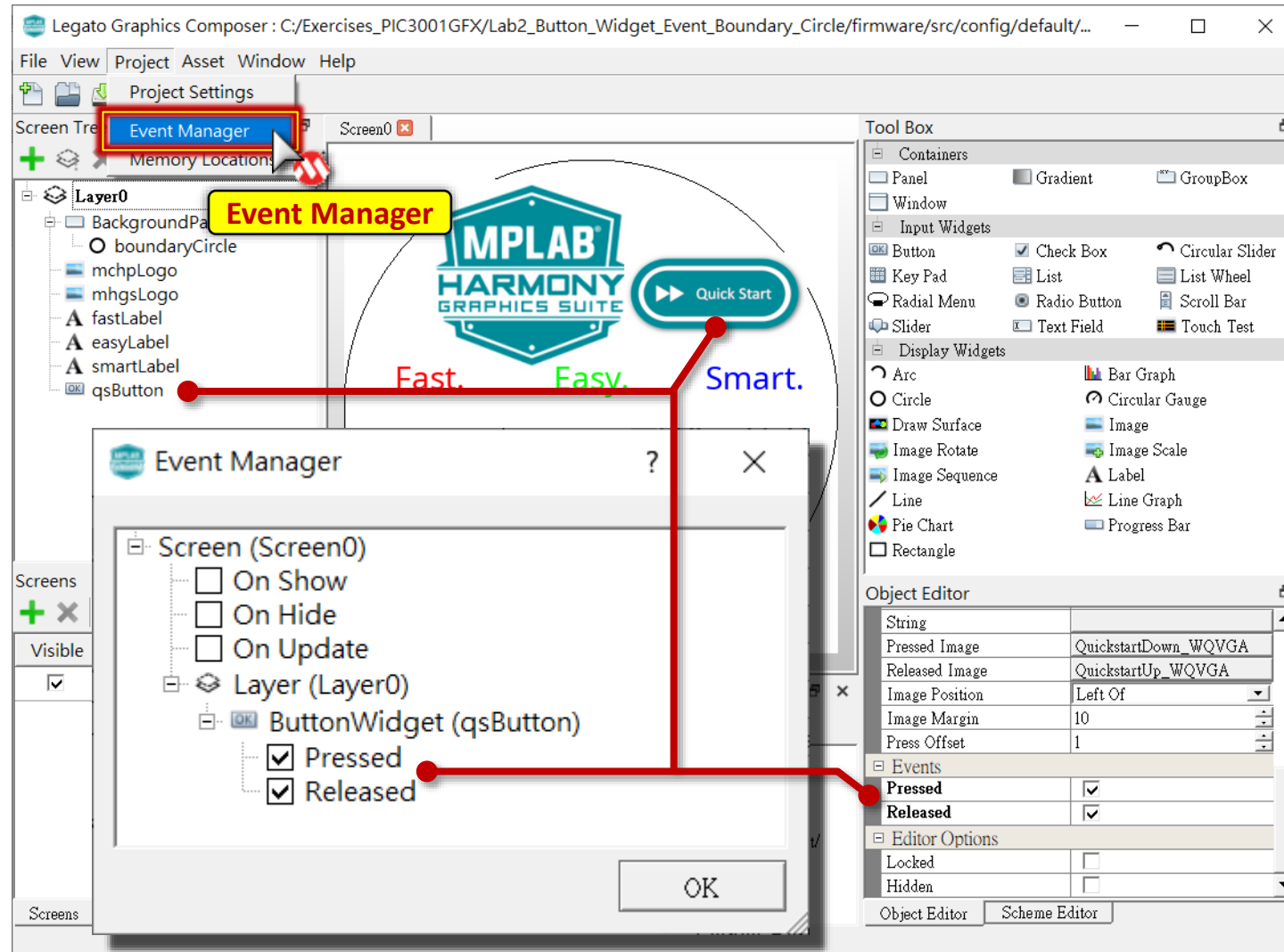
Property	Value
Name	qsButton
Position	[256,142]
Size	[160,74]
Enabled	☒
Visible	☒
Scheme	(internal)
Background	None
Border	None
Alignment	[Center,Middle]
Margins	[4,4,4,4]
Alpha Blending	☐
Alpha Level	255
Toggleable	☐
String	
Pressed Image	QuickstartDown_WQVGA
Released Image	QuickstartUp_WQVGA
Image Position	Left Of
Image Margin	10
Press Offset	1

Console:

```
(14:27:18) : Initializing Legato Graphics Composer v1.20.2b...
(14:27:19) : Initialization complete.
(14:27:19) : Welcome to Legato Graphics Composer!
(14:31:59) : Opening project: C:/Work/PIC32MZ_DA_Radial/firmware/src/config/default/MyProject.zip
```

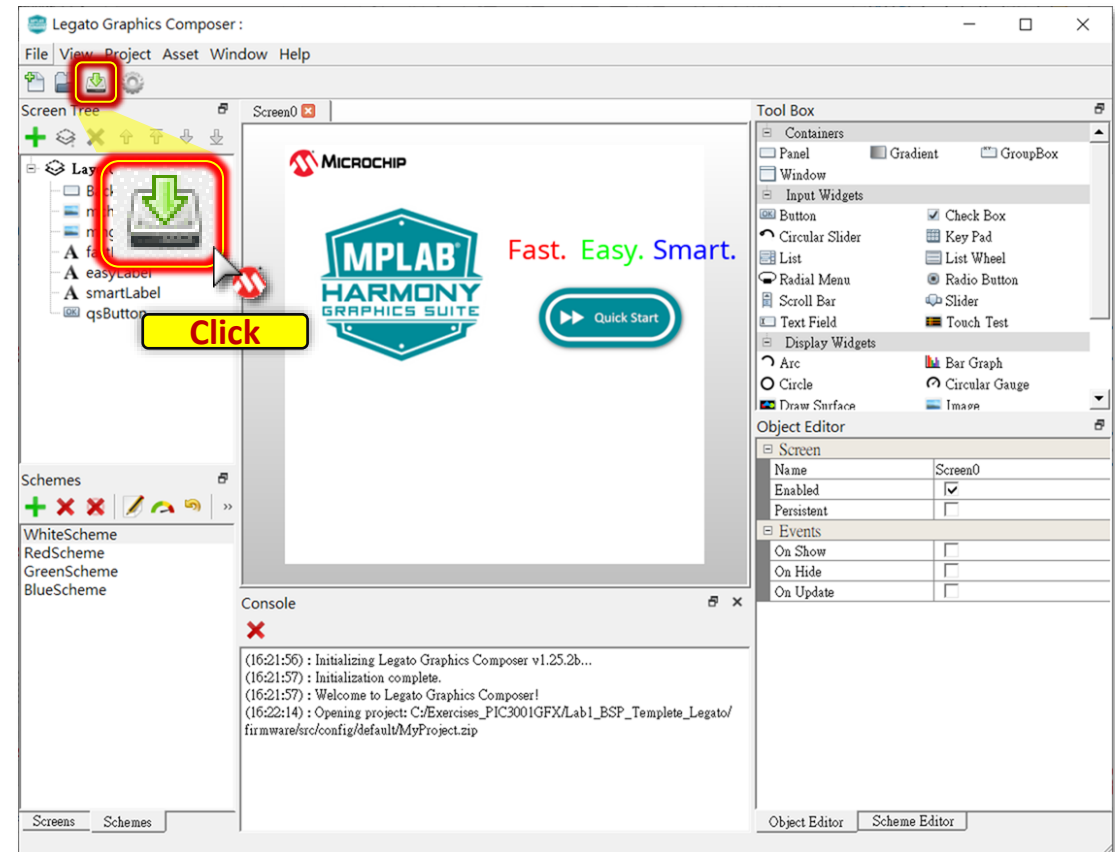
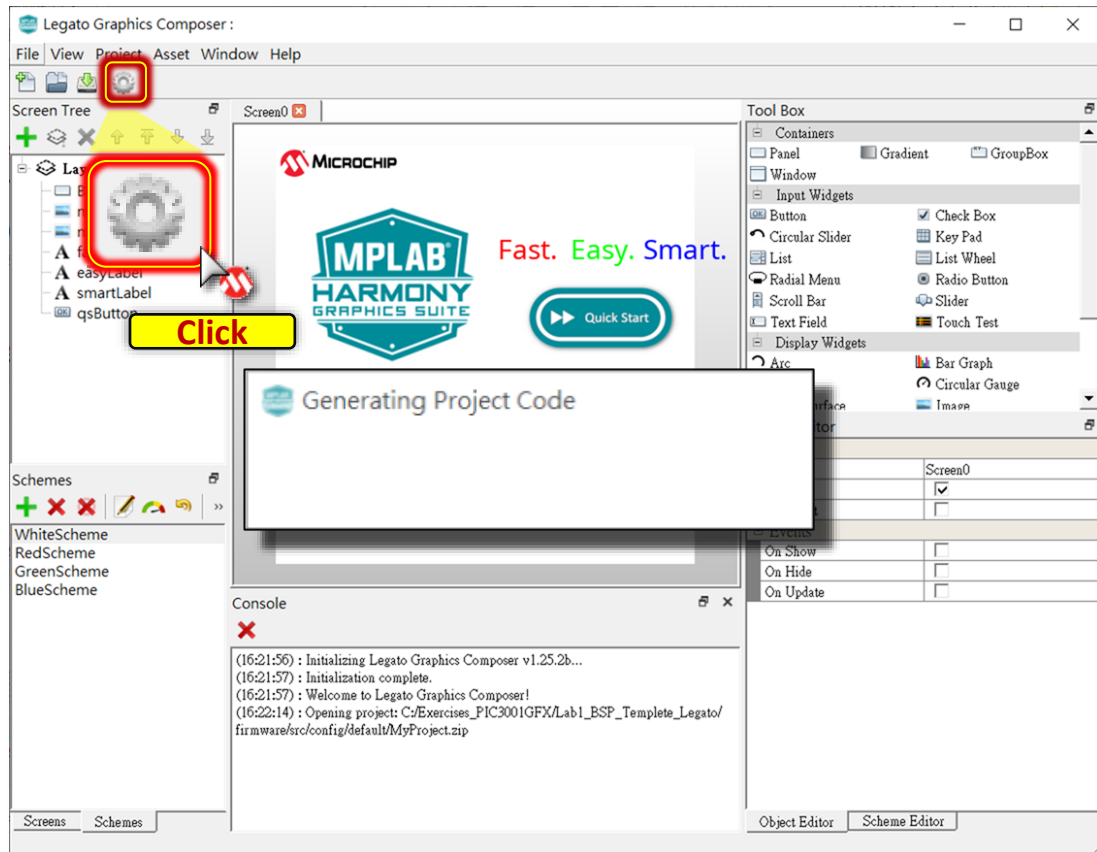
Button Widget Event

- To check all screens and widgets events in “Event Manager”.



Button Widget Event

- Generate Design Code and Save it to MyProject.zip
- **Please remember this step and will never remind again later.**



Button Widget Event

- Return to Harmony and Generate Code.
- **Please remember this step and will never remind again later.**

MPLAB X IDE v6.00 - Lab2_Button_Widget_Event_Boundary_Circle

File Edit MHC View Navigate Source Refactor Production Debug Team Tools Window Help

default

Projects Files Active Components*

CAN1
CAN2
CDC Function Driver
COMMAND
CONSOLE
CORE TIMER
Core
DDR

PIC32MZ_DA_Radial_Dashboard Available Components*

Project Graph* View: Root

Code

Peripheral Library I2C Driver Core Service

Service Layer USB DEVICE

SOLE Service Service

build, . . .
summed
000, end address

Generating Project...

Task Type	Remaining	Total
File Markup	0	16
File Copy	0	16
Libraries	0	0
Settings	0	7
Source Paths	0	0

Project generation complete!

100% Please Wait...

Generate Project

1. Configure Generation Settings

☐ Generate Settings
Merge Strategy USER_ALL

(Mouse over a property for detailed help)

2. View Warnings

Type	Description
------	-------------

3. Click Generate

Default linker script is added to the project.
Disable it for the project that require custom linker script.

Generate Cancel

Button Widget Event

- What's different between before and after we changed widget ?

`\firmware\src\config\default\gfx\legato\generated\screen\le_gen_screen_Screen0.c`

```
leResult screenShow_Screen0(void)
{
    ...
```

```
Screen0_qsButton = leButtonWidget_New();
Screen0_qsButton->fn->setPosition(Screen0_qsButton, 256, 142);
Screen0_qsButton->fn->setSize(Screen0_qsButton, 160, 74);
Screen0_qsButton->fn->setBackgroundType(Screen0_qsButton, LE_WIDGET_BACKGROUND_NONE);
Screen0_qsButton->fn->setBorderType(Screen0_qsButton, LE_WIDGET_BORDER_NONE);
Screen0_qsButton->fn->setPressedImage(Screen0_qsButton, (leImage*)&QuickstartDown_WQVGA);
Screen0_qsButton->fn->setReleasedImage(Screen0_qsButton, (leImage*)&QuickstartUp_WQVGA);
Screen0_qsButton->fn->setPressedEventCallback(Screen0_qsButton, event_Screen0_qsButton_OnPressed);
Screen0_qsButton->fn->setReleasedEventCallback(Screen0_qsButton, event_Screen0_qsButton_OnReleased);
```

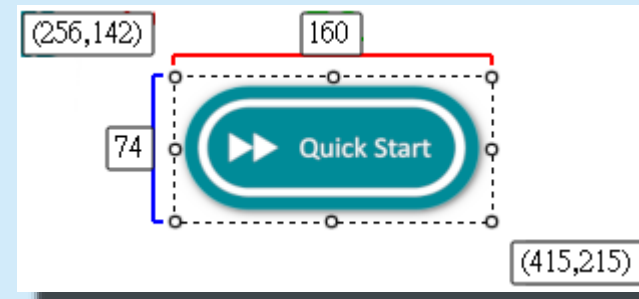
```
root0->fn->addChild(root0, (leWidget*)Screen0_qsButton);
leAddRootWidget(root0, 0);
leSetLayerColorMode(0, LE_COLOR_MODE_RGBA_8888);

showing = LE_TRUE;

return LE_SUCCESS;
}
```

qsButton Widget

Event Callback function registered



Object Editor

Widget	
Name	qsButton
Position	[256,142]
Size	[160,74]
Enabled	☒
Visible	☒
Scheme	(internal) ?
Background	None
Border	None
Alignment	[Center,Middle]
Margins	[4,4,4,4]
Alpha Blending	☐
Alpha Level	255

Button	
Toggleable	☐
String	
Pressed Image	QuickstartDown_WQVGA
Released Image	QuickstartUp_WQVGA
Image Position	Left Of
Image Margin	10
Press Offset	1

Events	
Pressed	☒
Released	☒

Editor Options	
Locked	☐
Hidden	☐

Button Widget Event

- What's different between before and after we changed widget ?

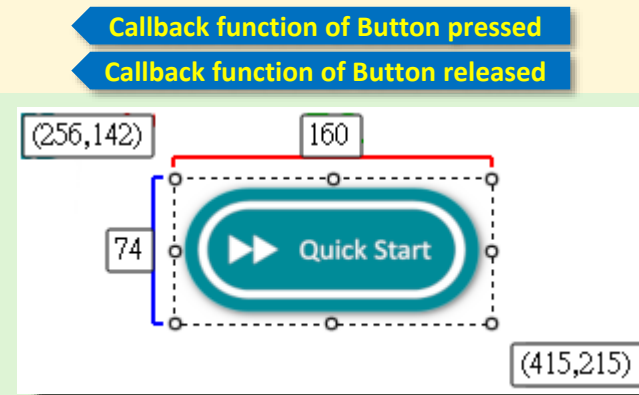
`\firmware\src\config\default\gfx\legato\generated\screen\le_gen_screen_Screen0.h`

```
...
// screen member widget declarations
extern leWidget* Screen0_BackgroundPanel;
extern leImageWidget* Screen0_mchpLogo;
extern leImageWidget* Screen0_mhgsLogo;
extern leLabelWidget* Screen0_fastLabel;
extern leLabelWidget* Screen0_easyLabel;
extern leLabelWidget* Screen0_smartLabel;
extern leButtonWidget* Screen0_qsButton;

// event handlers
// !!THESE MUST BE IMPLEMENTED IN THE APPLICATION CODE!!
void event_Screen0_qsButton_OnPressed(leButtonWidget* btn);
void event_Screen0_qsButton_OnReleased(leButtonWidget* btn);

// screen lifecycle functions
// DO NOT CALL THESE DIRECTLY
leResult screenInit_Screen0(void); // called when Legato is initialized
leResult screenShow_Screen0(void); // called when screen is shown
void screenHide_Screen0(void); // called when screen is hidden
void screenDestroy_Screen0(void); // called when Legato is destroyed
```

qsButton Widget



Object Editor	
Widget	
Name	qsButton
Position	[256,142]
Size	[160,74]
Enabled	☒
Visible	☒
Scheme	(internal) ?
Background	None
Border	None
Alignment	[Center,Middle]
Margins	[4,4,4,4]
Alpha Blending	☐
Alpha Level	255
Button	
Toggleable	☐
String	
Pressed Image	QuickstartDown_WQVGA
Released Image	QuickstartUp_WQVGA
Image Position	Left Of
Image Margin	10
Press Offset	1
Events	
Pressed	☒
Released	☒
Editor Options	
Locked	☐
Hidden	☐

Button Widget Event

- Implement Button Widget Event Callback Function.

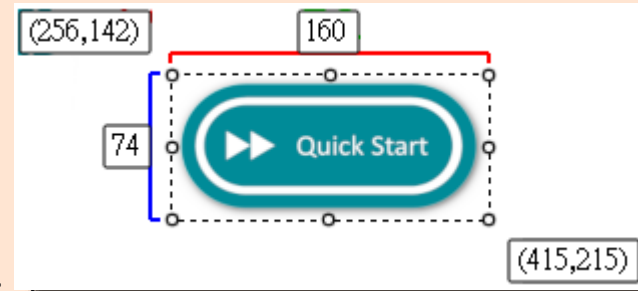
`\firmware\src\main.c`

```
...  
#include "definitions.h"      // SYS function prototypes
```

```
void event_Screen0_qsButton_OnPressed(leButtonWidget* btn) Callback function of Button pressed  
{  
    RGB_LED_B_On(); Button pressed, LED On  
}
```

```
void event_Screen0_qsButton_OnReleased(leButtonWidget* btn) Callback function of Button released  
{  
    RGB_LED_B_Off(); Button released, LED Off  
}
```

```
int main ( void )  
{  
    /* Initialize all modules */  
    SYS_Initialize ( NULL );  
  
    while ( true )  
    {  
        /* Maintain state machines of all polled MPLAB Harmony modules.
```



Object Editor

Widget	
Name	qsButton
Position	[256,142]
Size	[160,74]
Enabled	☒
Visible	☒
Scheme	(internal) ?
Background	None
Border	None
Alignment	[Center,Middle]
Margins	[4,4,4,4]
Alpha Blending	☐
Alpha Level	255

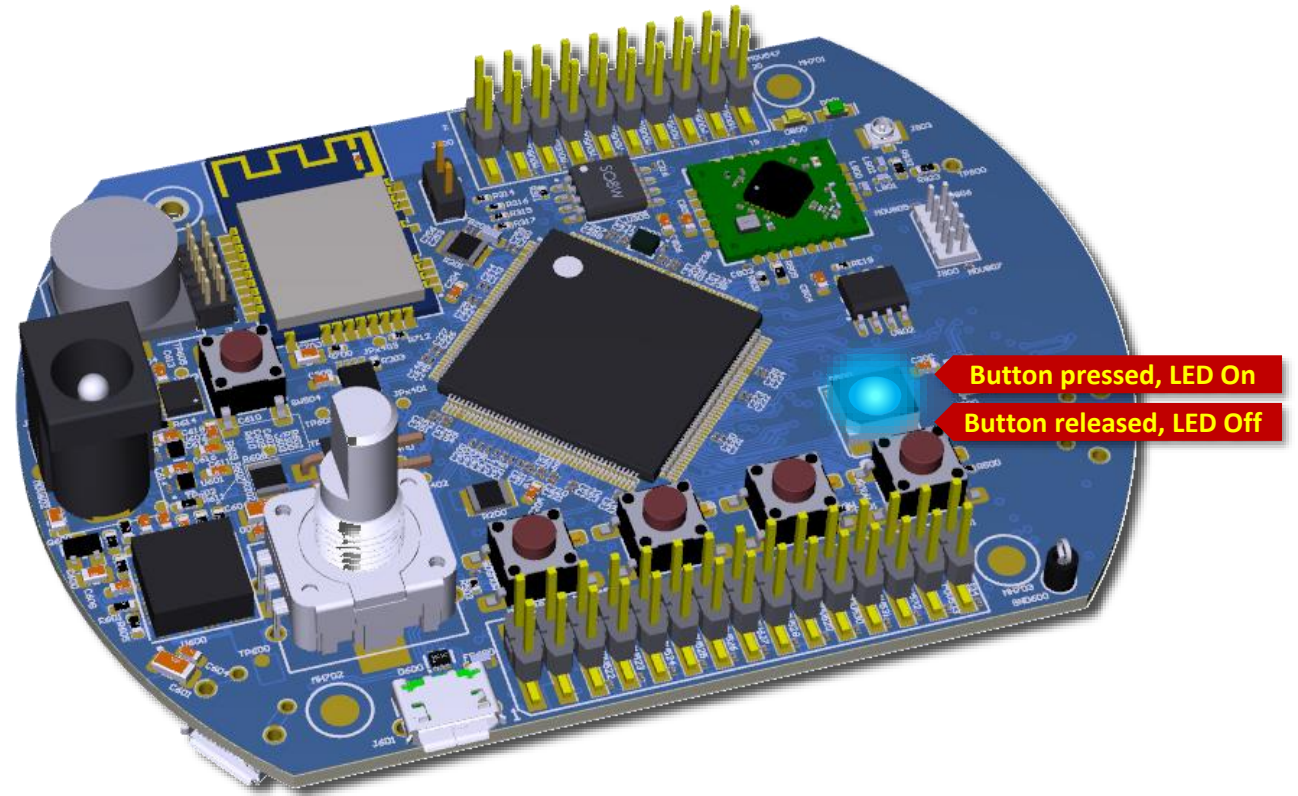
Button	
Toggleable	☐
String	
Pressed Image	QuickstartDown_WQVGA
Released Image	QuickstartUp_WQVGA
Image Position	Left Of
Image Margin	10
Press Offset	1

Events	
Pressed	☒
Released	☒

Editor Options	
Locked	☐
Hidden	☐

Button Widget Event

- The Button Widget Result after program.

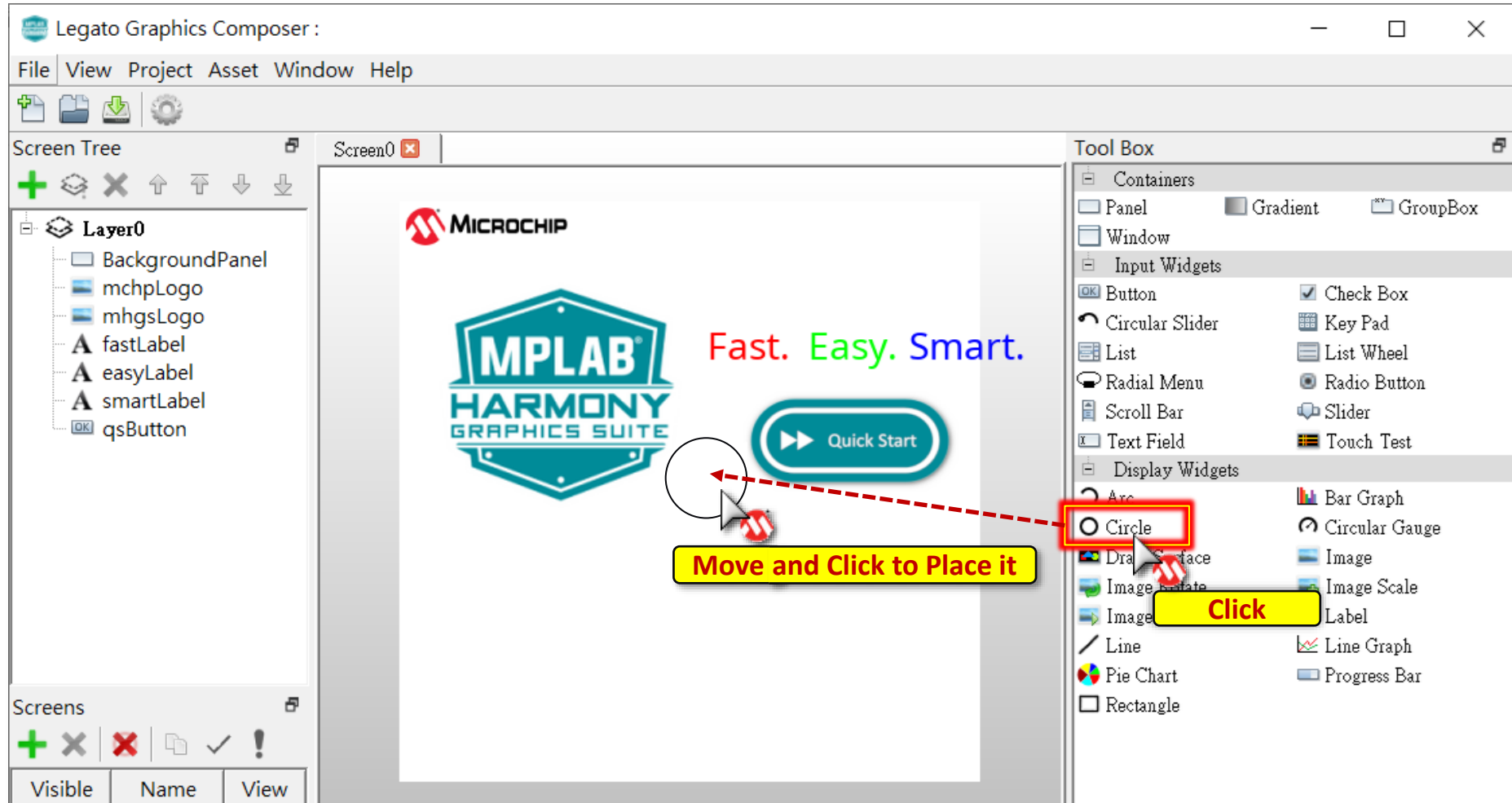


Boundary Circle Widget

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Boundary Circle Widget

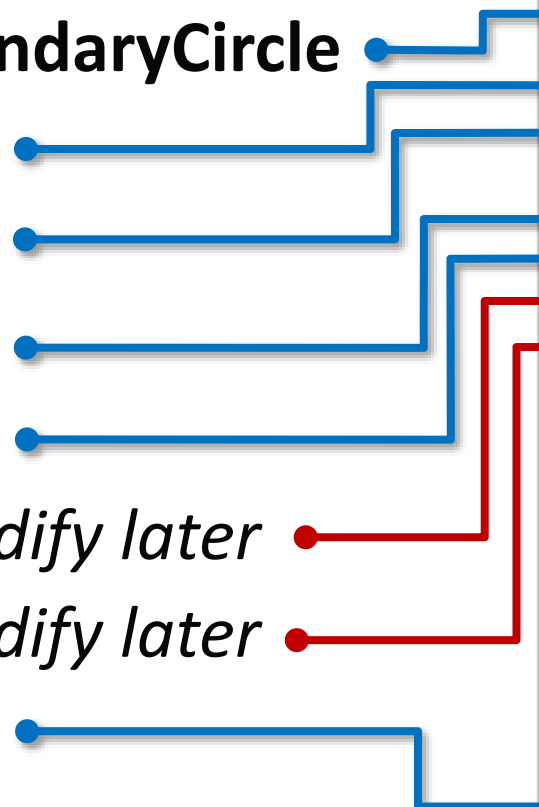
- Add a Circle Widget to indicate the display boundary.



Boundary Circle Widget

- **Configure Circle Widget as below.**

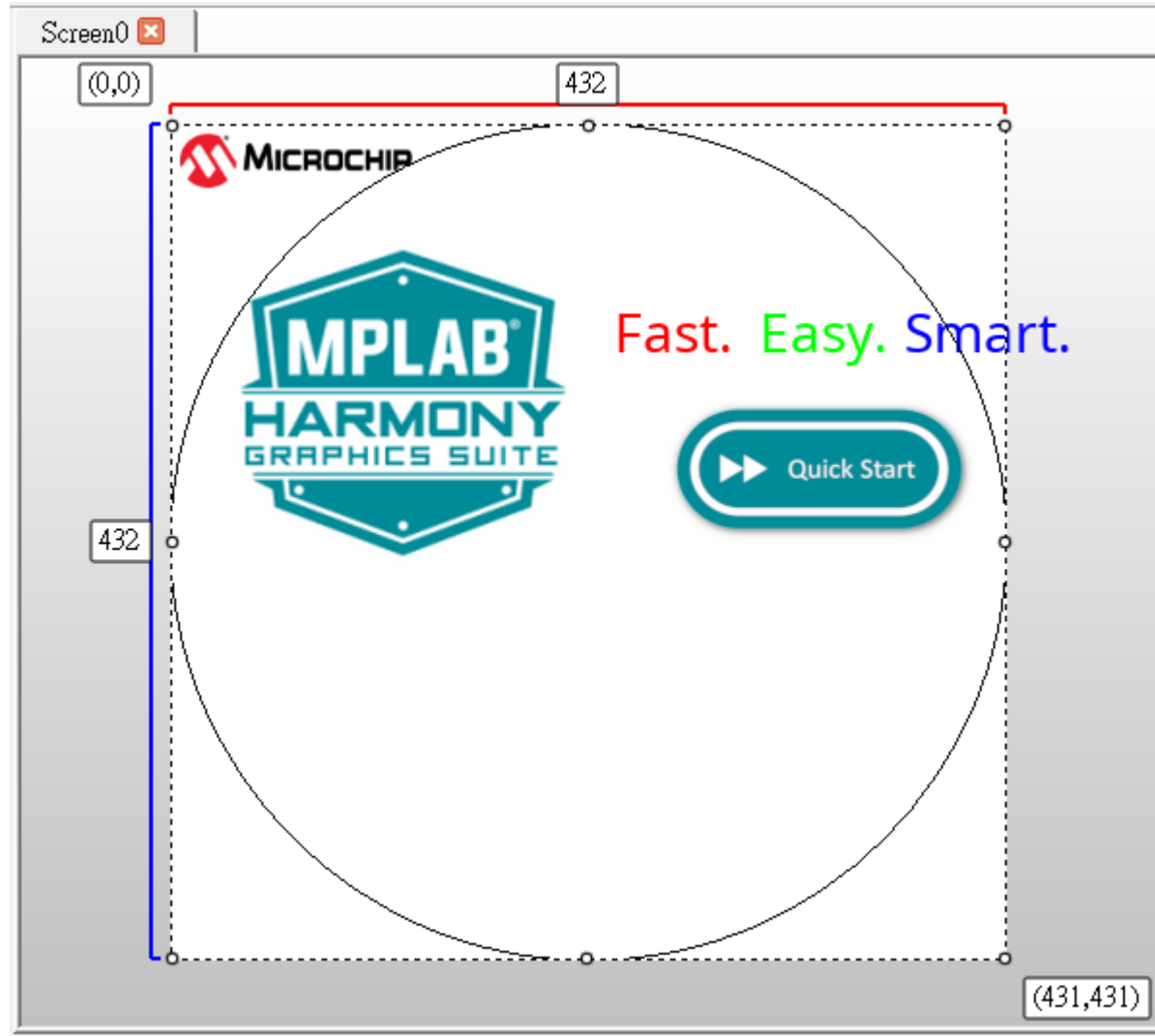
- Name : **boundaryCircle**
- Position X : **0**
- Position Y : **0**
- Size Width : **432**
- Size Height : **432**
- Enable : *Modify later*
- Visible : *Modify later*
- Radius : **216**



Object Editor	
[-] Widget	
Name	boundaryCircle
[-] Position	
X	0
Y	0
[-] Size	
Width	432
Height	432
Enabled	☒
Visible	☒
Scheme	(internal) ?
Background	None
Border	None
[+] Alignment	
[Center,Middle]	
[+] Margins	
[4,4,4,4]	
Alpha Blending	☐
Alpha Level	255
[-] Circle	
Origin X	0
Origin Y	0
Radius	216
Thickness	1
Filled	☐
[-] Editor Options	
Locked	☐
Hidden	☐

Boundary Circle Widget

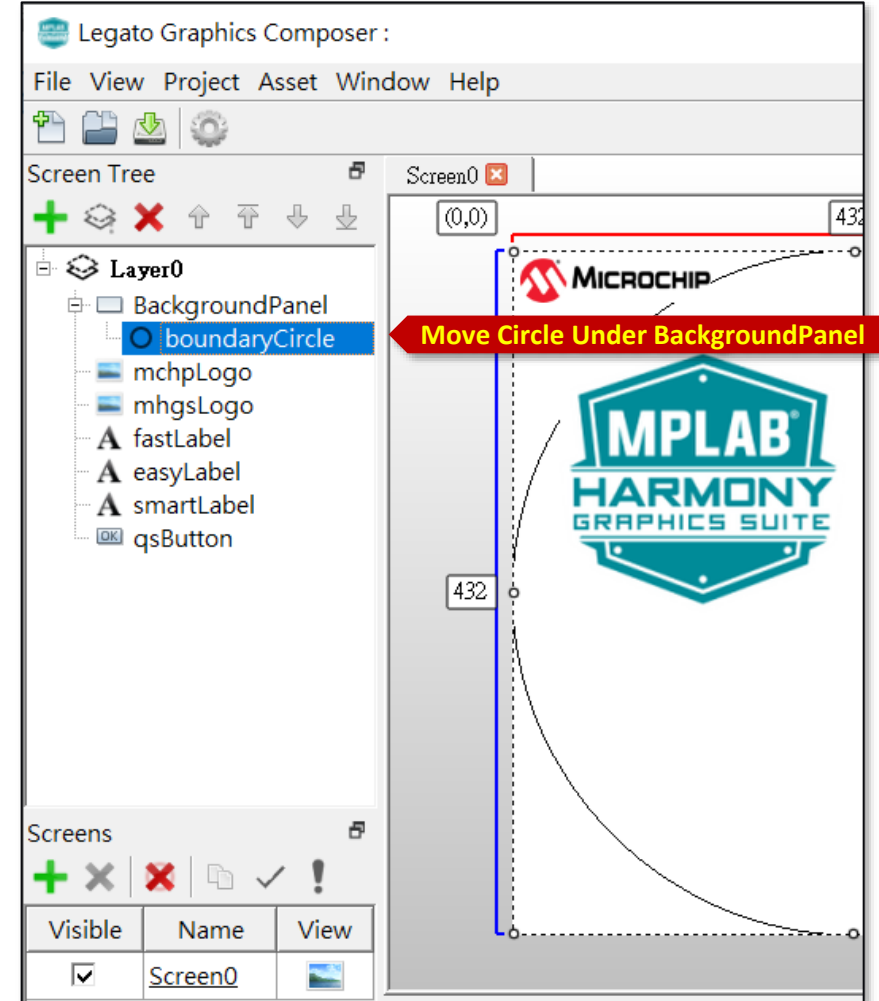
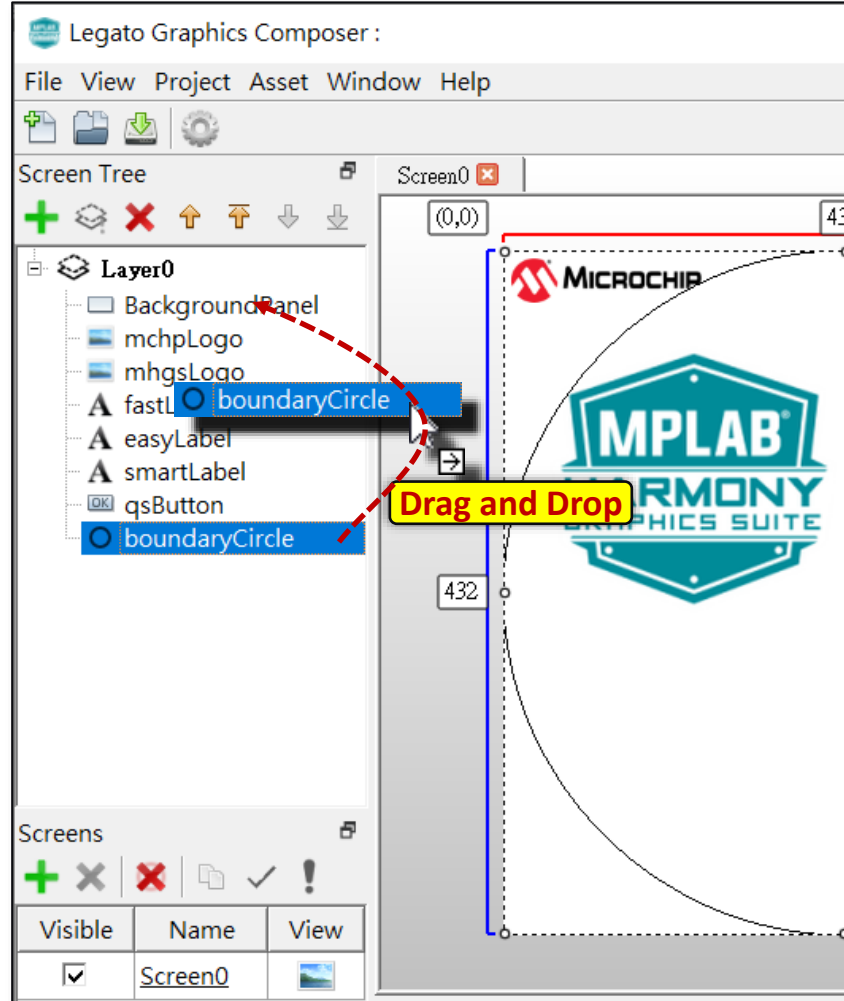
- Circle Widget after configured.



Object Editor	
Widget	
Name	boundaryCircle
Position	
X	0
Y	0
Size	
Width	432
Height	432
Enabled	☒
Visible	☒
Scheme	(internal) ?
Background	None
Border	None
Alignment	
Margins	
[4,4,4,4]	
Alpha Blending	☐
Alpha Level	255
Circle	
Origin X	0
Origin Y	0
Radius	216
Thickness	1
Filled	☐
Editor Options	
Locked	☐
Hidden	☐

Boundary Circle Widget

- Drag and Drop **Circle** Widget under the **BackgroundPanel**.

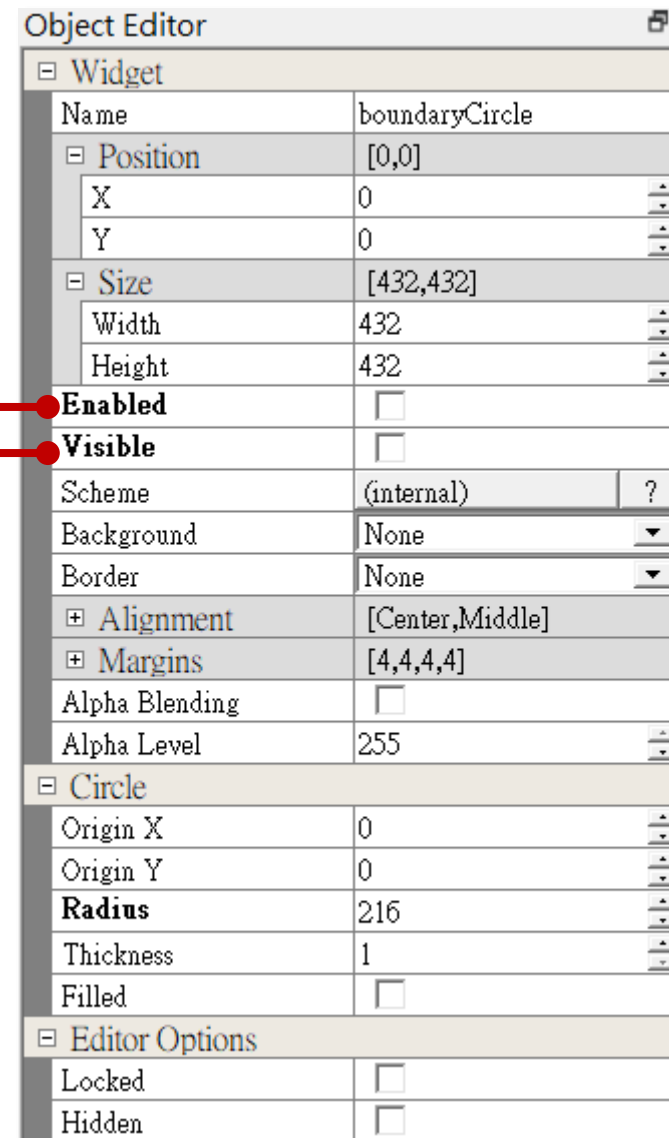


Boundary Circle Widget

- **Set Circle to Disable and Invisible**

- Name : boundaryCircle
- Position X : 0
- Position Y : 0
- Size Width : 432
- Size Height : 432
- Enable : **Disable**
- Visible : **Disable**
- Radius : 216

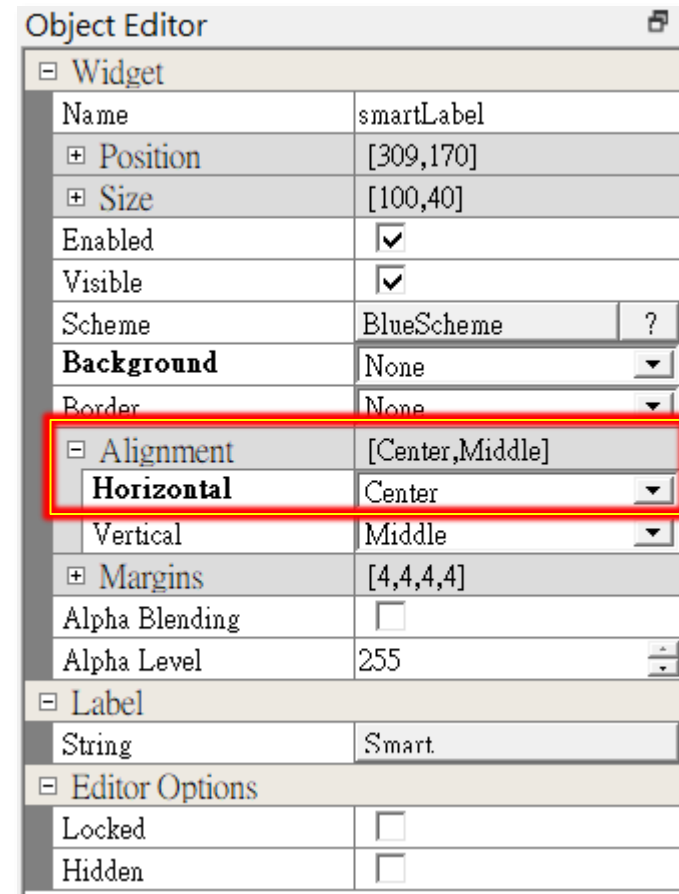
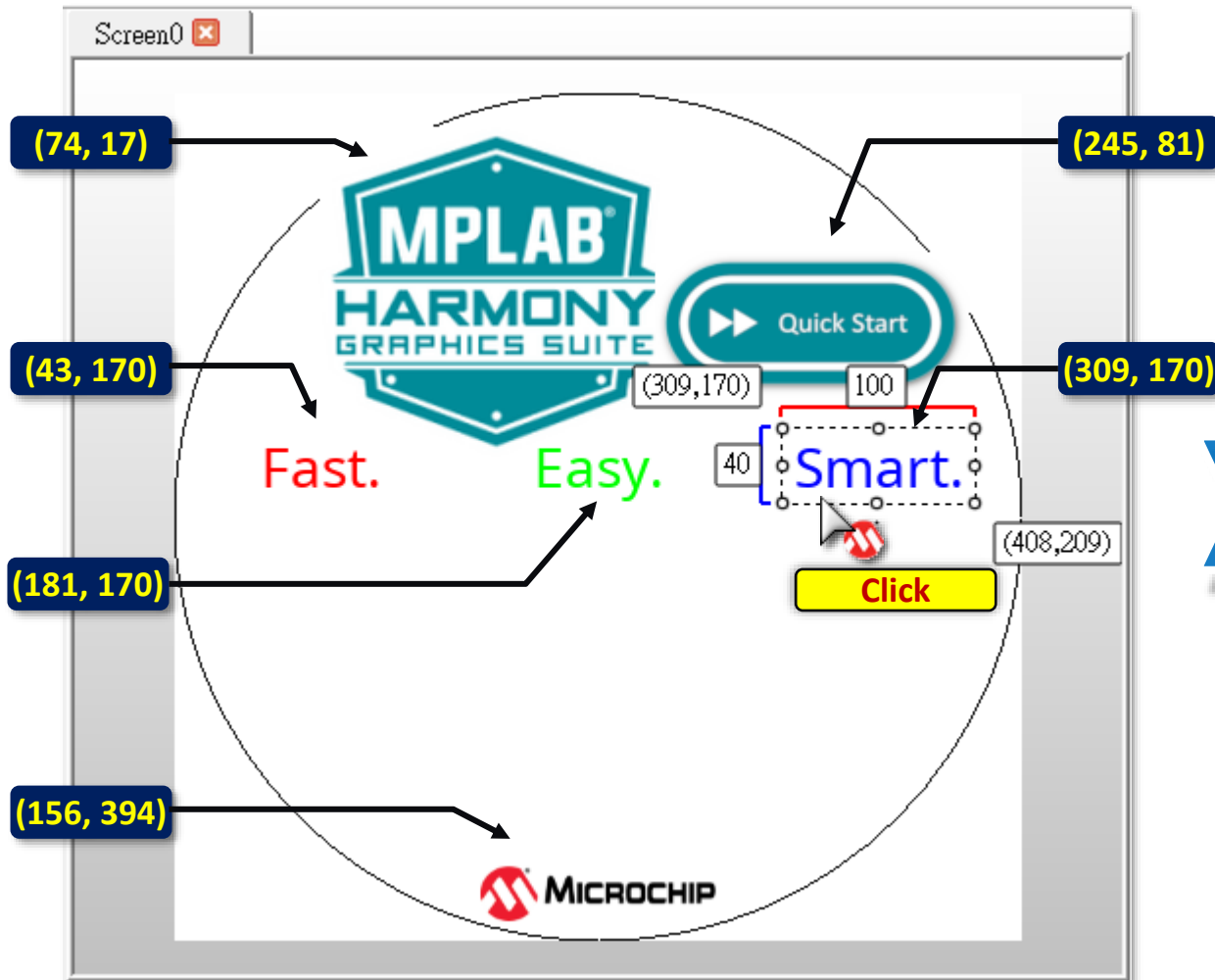
**Boundary Circle is for design reference
It should not be rendered while display.**



Object Editor	
Widget	
Name	boundaryCircle
Position	
X	0
Y	0
Size	
Width	432
Height	432
Enabled	☐
Visible	☐
Scheme	(internal) ?
Background	None
Border	None
Alignment	[Center,Middle]
Margins	[4,4,4,4]
Alpha Blending	☐
Alpha Level	255
Circle	
Origin X	0
Origin Y	0
Radius	216
Thickness	1
Filled	☐
Editor Options	
Locked	☐
Hidden	☐

Boundary Circle Widget

- Given the position (X, Y) to relocated all widgets as below and all Label Text Alignment **Horizontal** to **Center**.



Boundary Circle Widget

- Boundary circle will show in composer but not output on screen.



Graphics Assets

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Graphics Assets Converter (GAC)

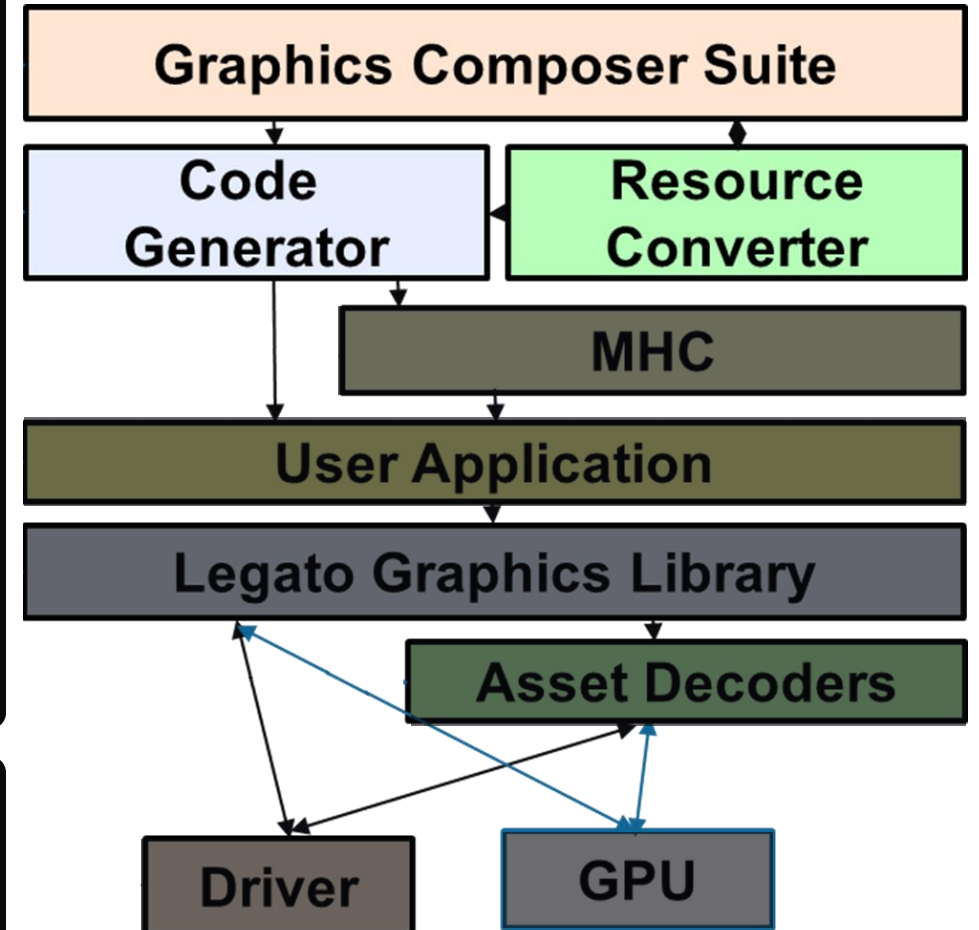
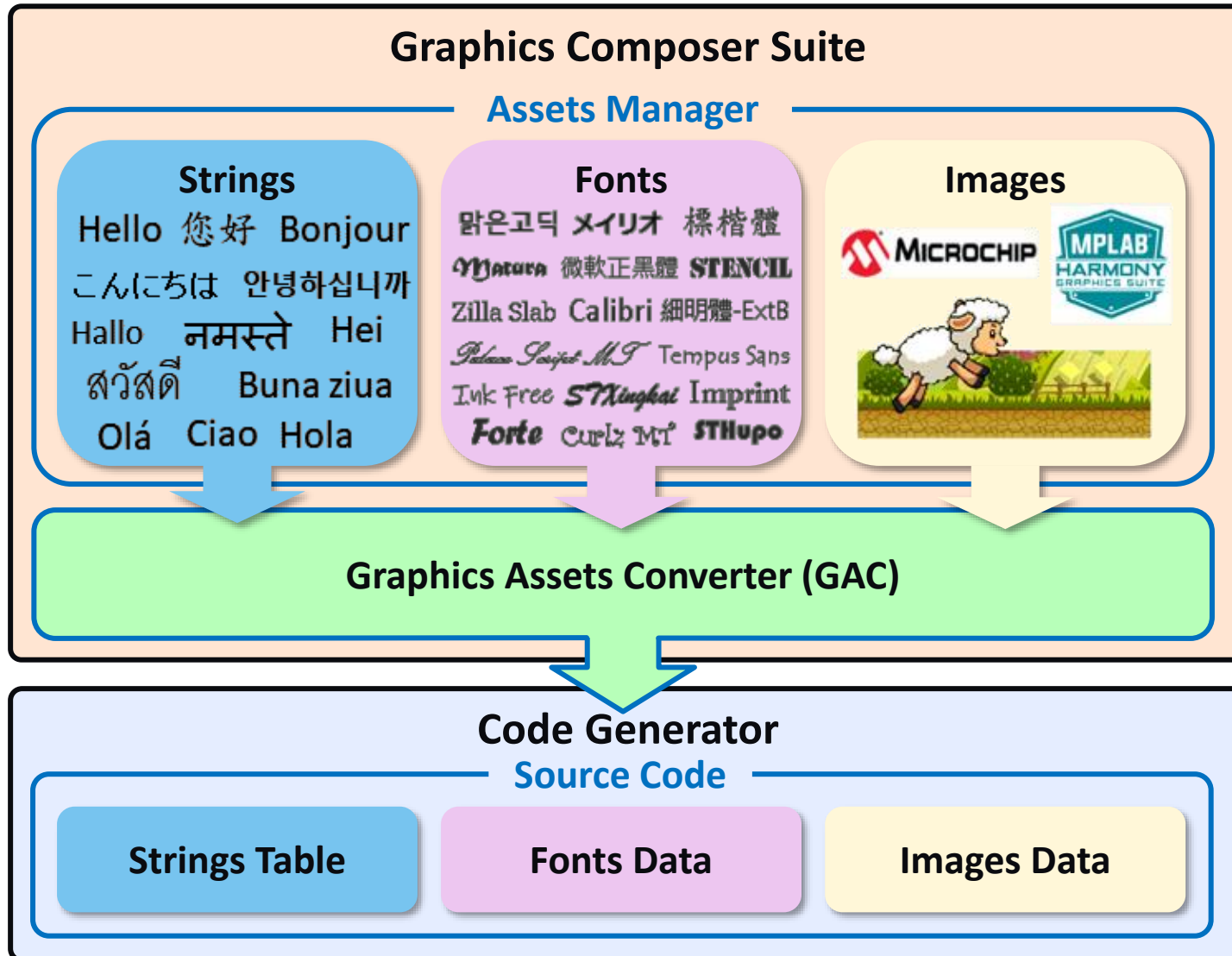
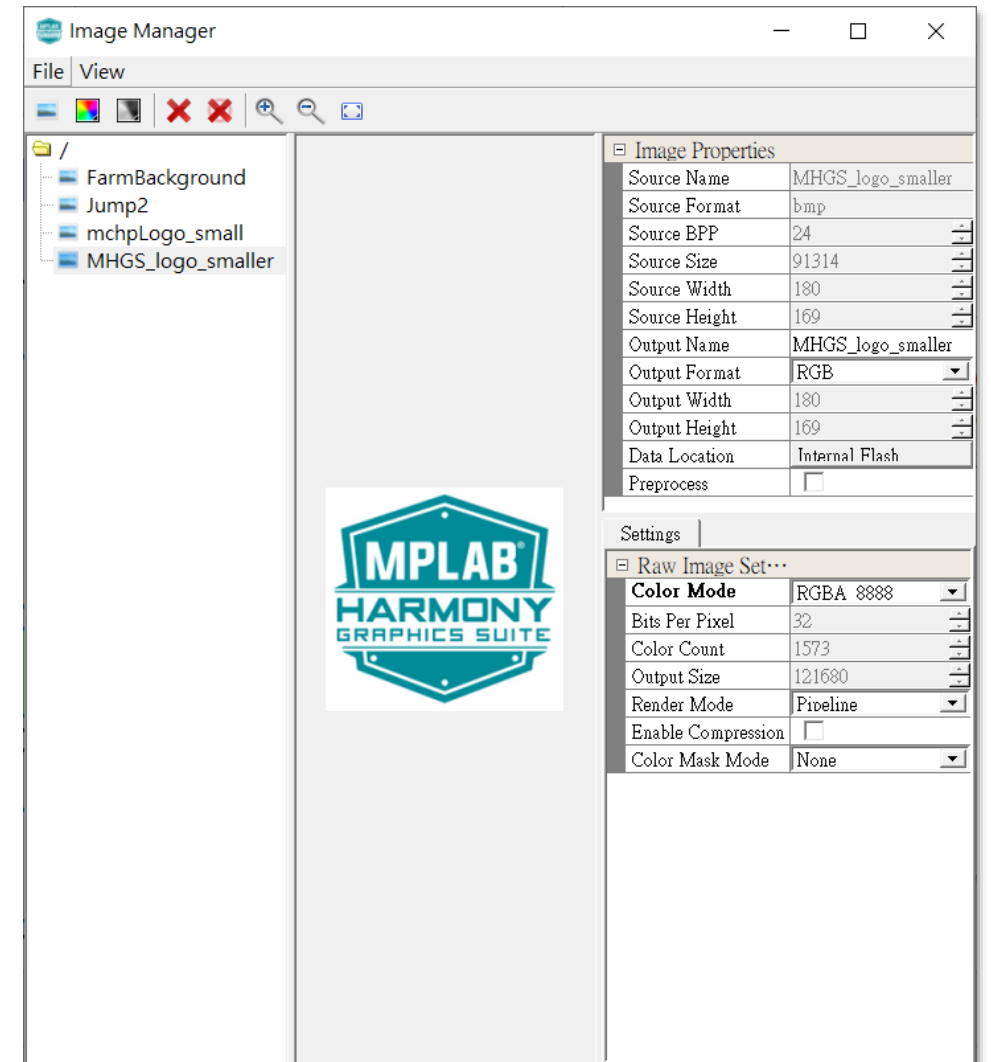
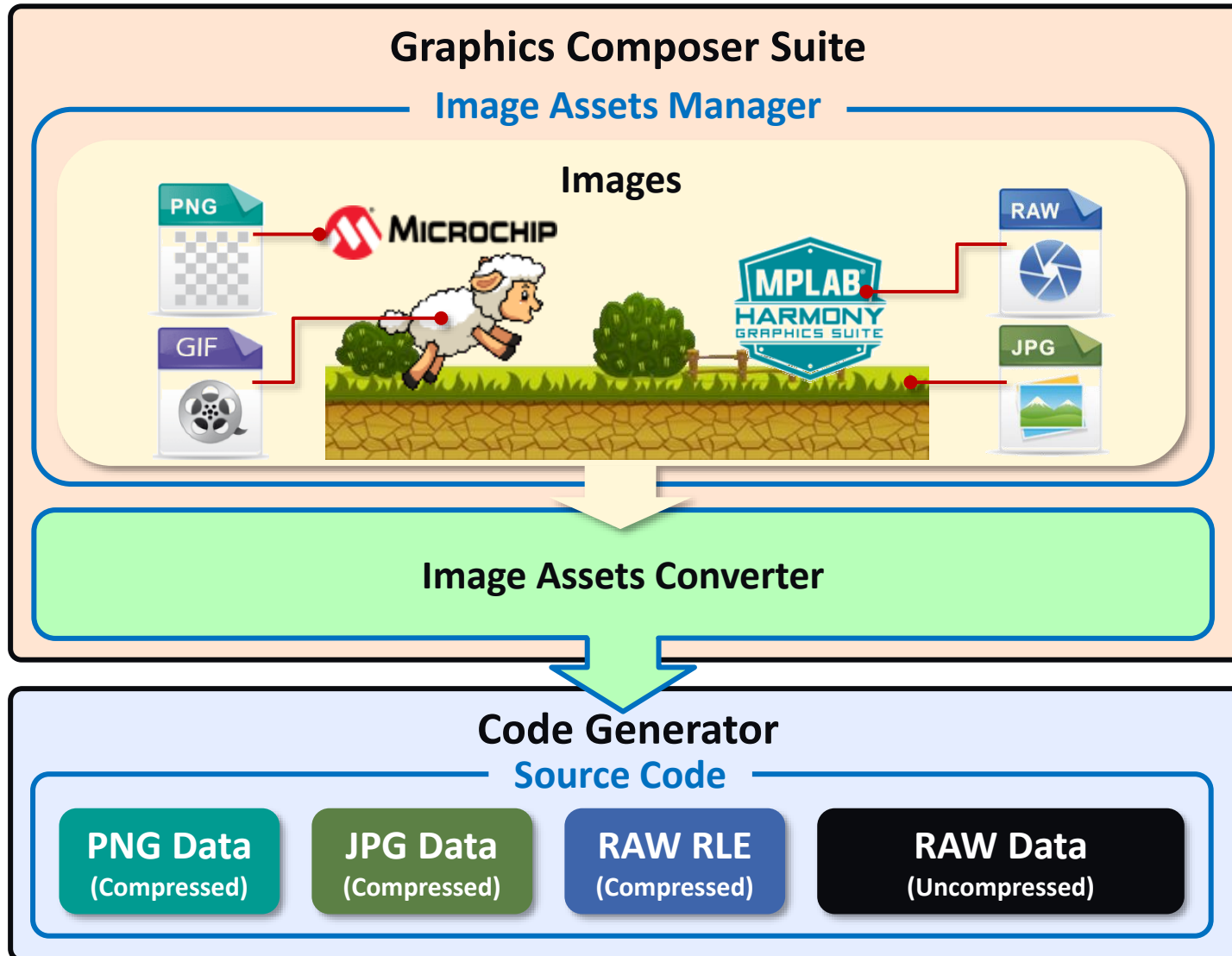


Image Assets Manager

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Image Assets Manager



Create Your Design

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Create Your Design

- Create four Start/Stop images for button animation.

StartStop_Stop

StartStop_StopPressed

StartStop_Start

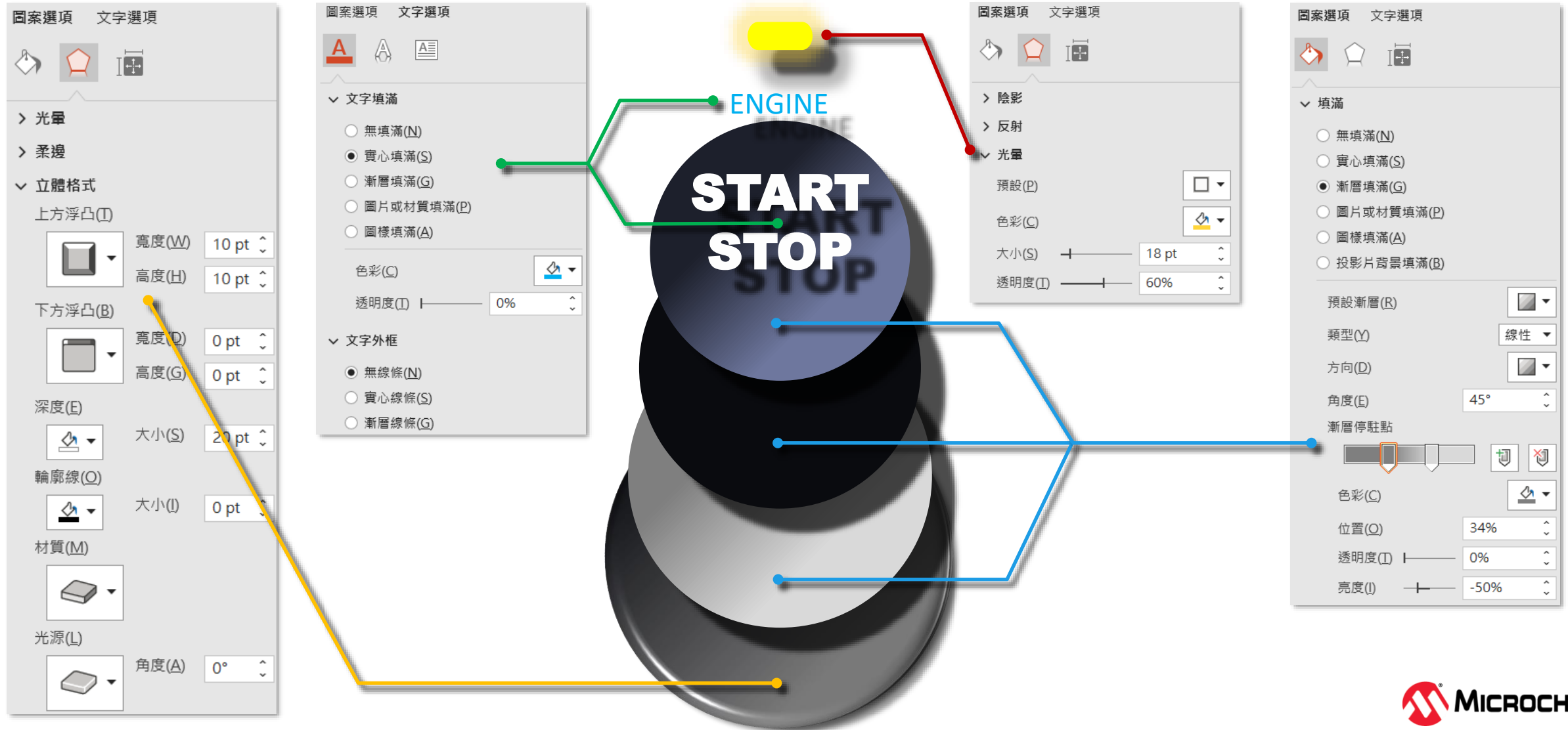
StartStop_StartPressed



Animation

Create Your Design

- Use PowerPoint  to create your fantasy design.



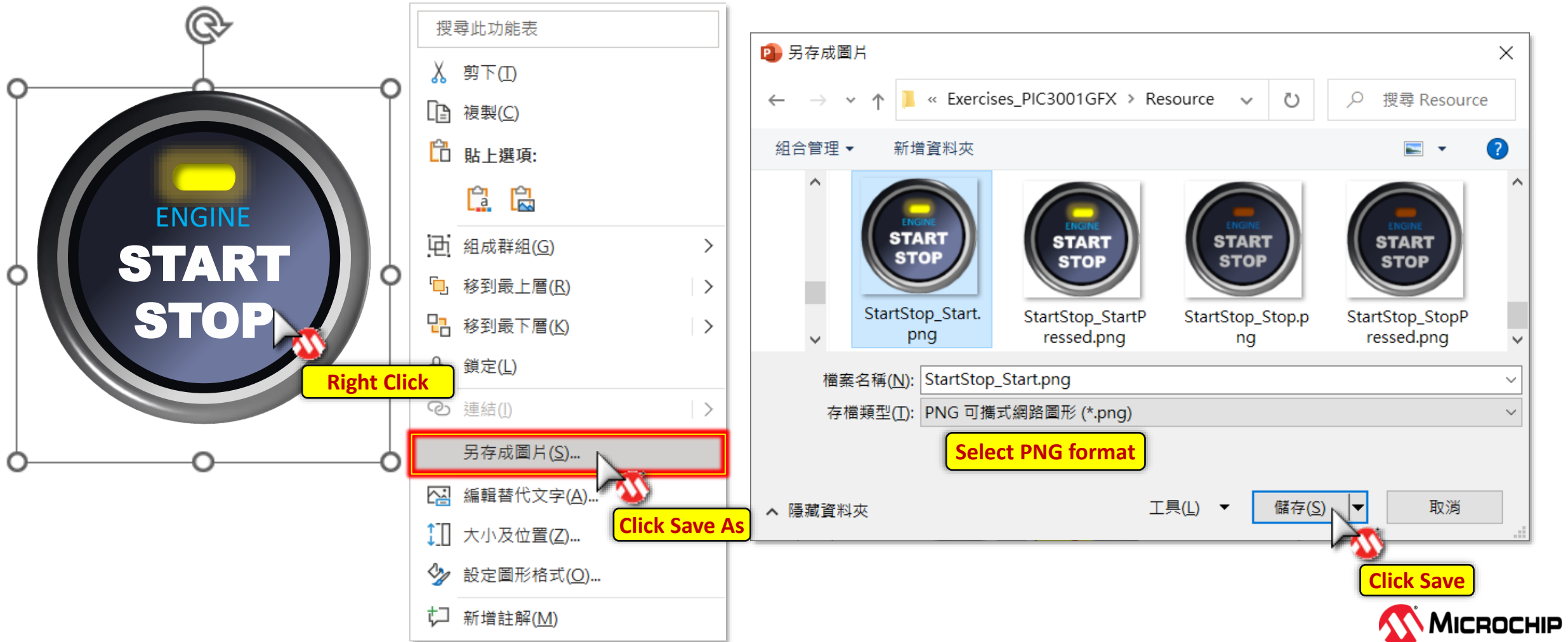
Generate Transparent Background Image

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Generate Transparent Background Image

PNG - Transparent Background Images

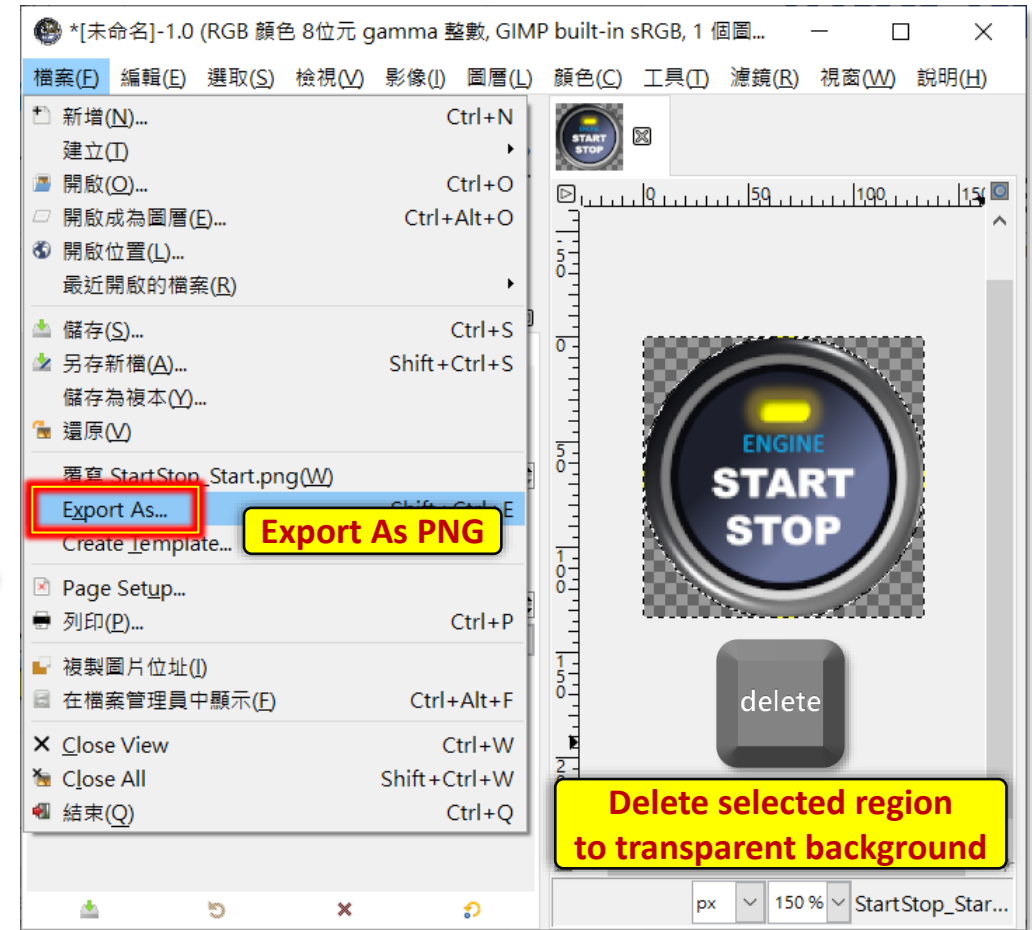
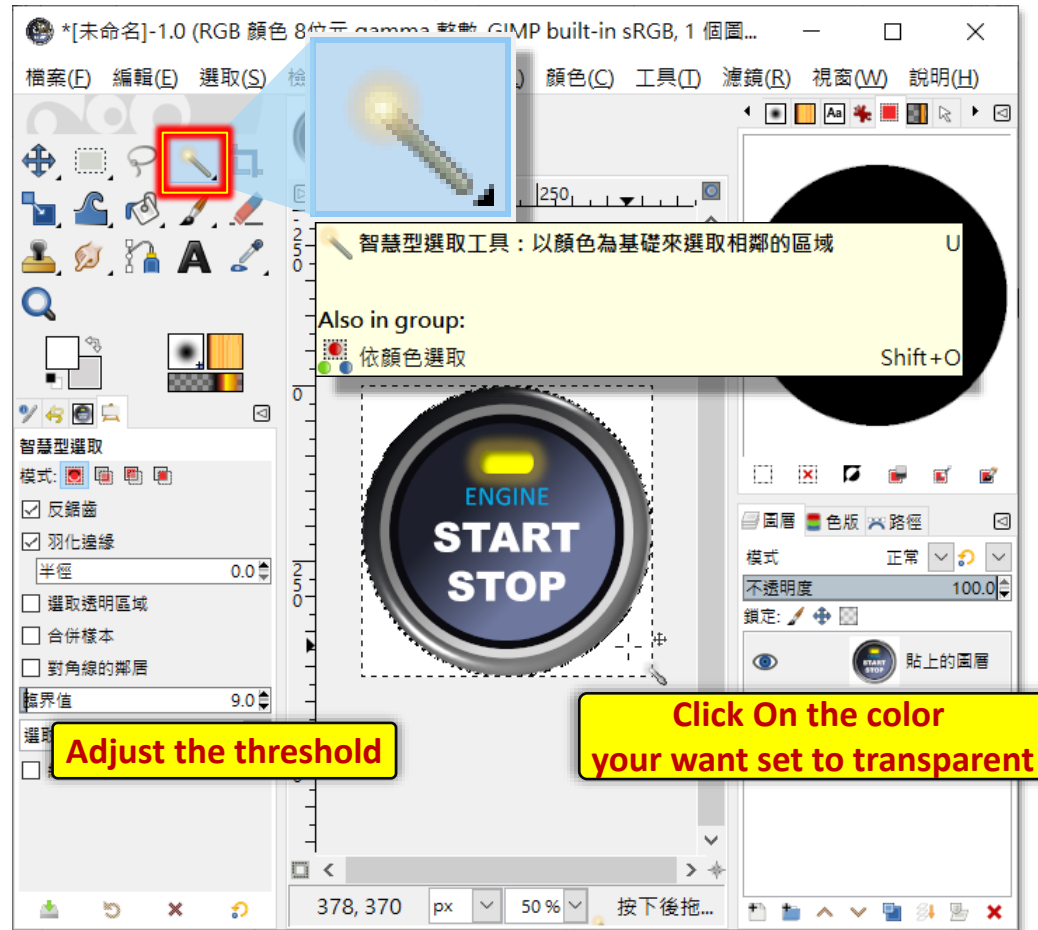
- Use PowerPoint  to save your design to PNG images.



Generate Transparent Background Image

PNG - Transparent Background Images

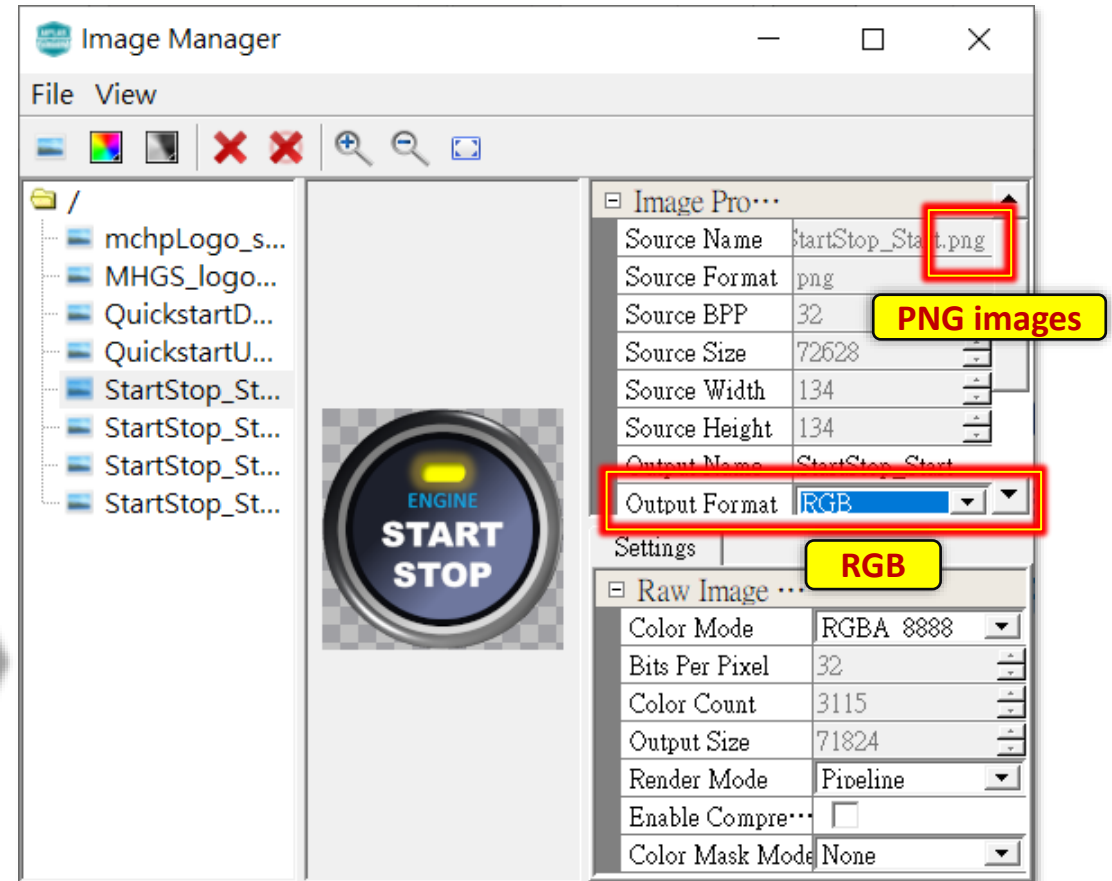
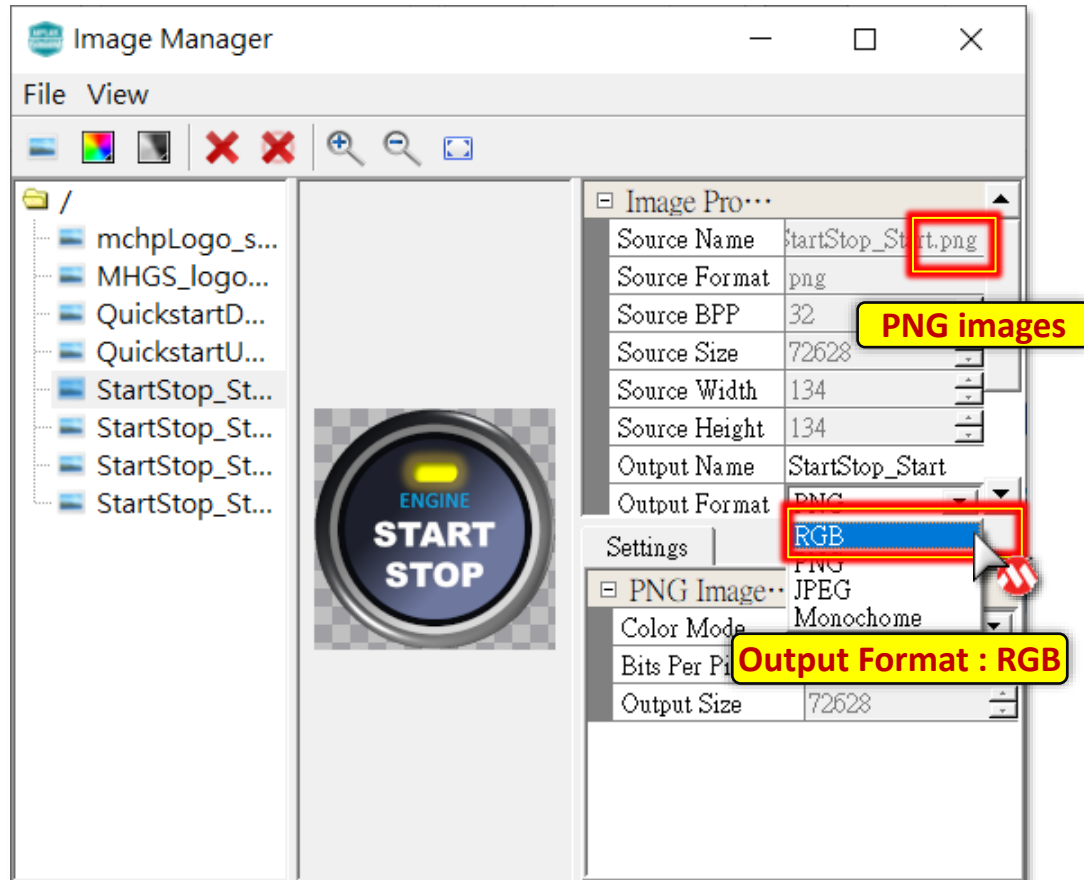
- Use GIMP  to generate transparent background PNG images.



Generate Transparent Background Image

PNG - Transparent Background Images

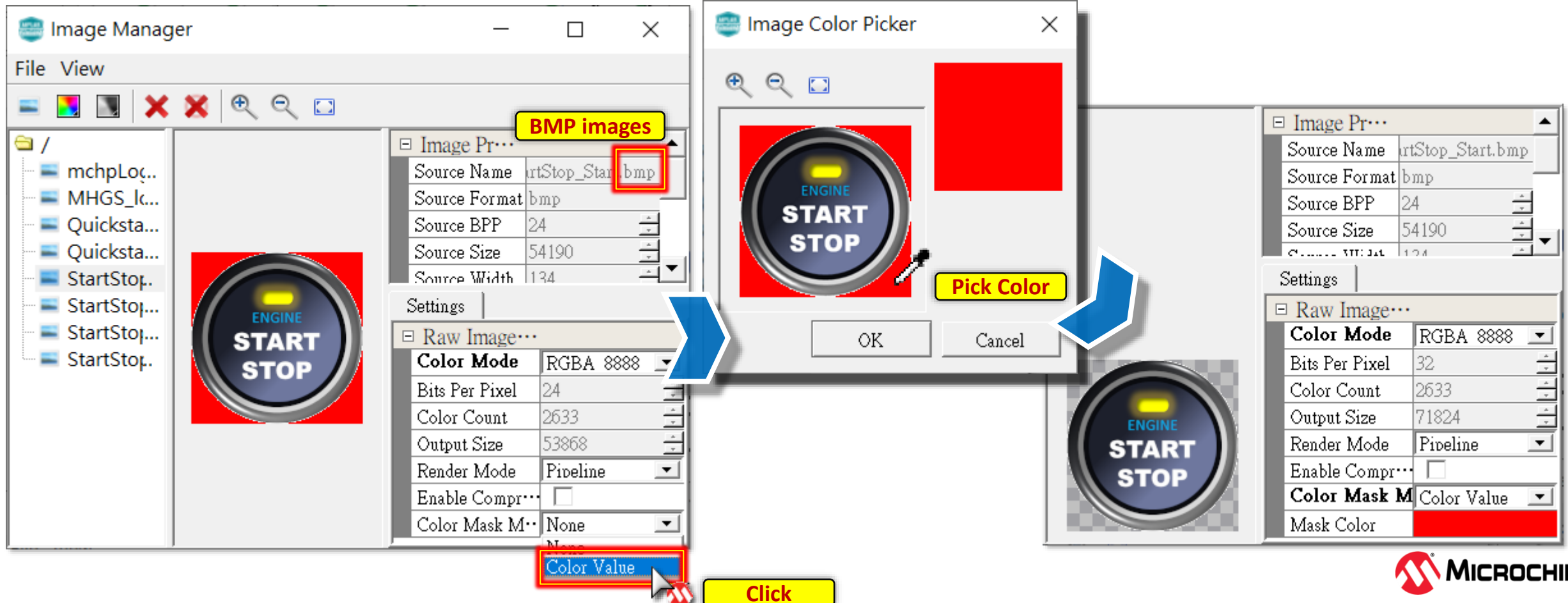
- Remember to set **Output Format** to “**RGB**” if using PNG images.



Generate Transparent Background Image

Color Mask - Transparent Background Images

- Use MHGS  and set Color Mask Mode as “Color Value” to pick-up a color for background transparent in Image Manager.



The screenshot illustrates the steps to generate a transparent background image using the MPLAB Harmony Graphics Suite. The **Image Manager** window displays a list of images, including **artStop_Start.bmp**, which is highlighted. The **Image Color Picker** window is open, showing the image and a red color swatch. The **Pick Color** button is highlighted. The **Color Value** option is selected in the **Color Mask Mode** dropdown. The **Click** button is highlighted.

Image Manager window details:

Image Properties	
Source Name	artStop_Start.bmp
Source Format	bmp
Source BPP	24
Source Size	54190
Source Width	134

Image Color Picker window details:

Image Properties	
Source Name	artStop_Start.bmp
Source Format	bmp
Source BPP	24
Source Size	54190
Source Width	134

Image Manager window details (continued):

Settings	
Color Mode	RGBA 8888
Bits Per Pixel	24
Color Count	2633
Output Size	53868
Render Mode	Pipeline
Enable Compr...	☐
Color Mask M...	None

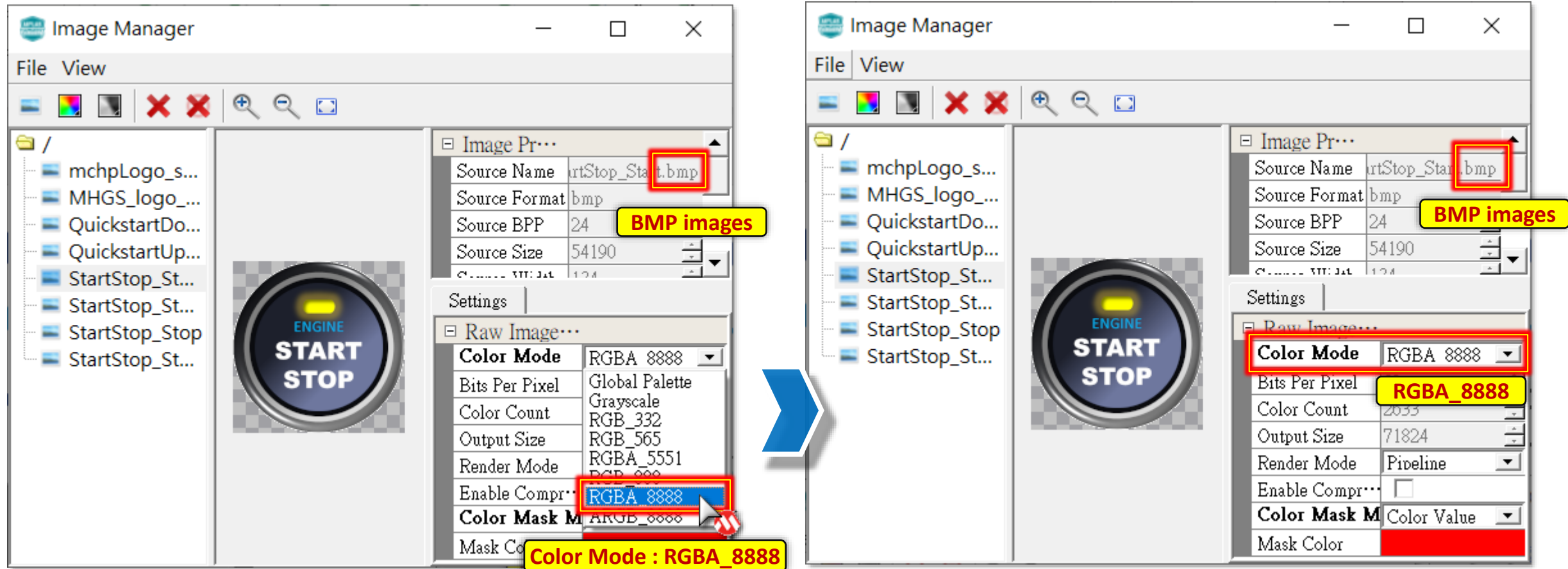
Image Color Picker window details (continued):

Settings	
Color Mode	RGBA 8888
Bits Per Pixel	32
Color Count	2633
Output Size	71824
Render Mode	Pipeline
Enable Compr...	☐
Color Mask M	Color Value
Mask Color	Red

Generate Transparent Background Image

Color Mask - Transparent Background Images

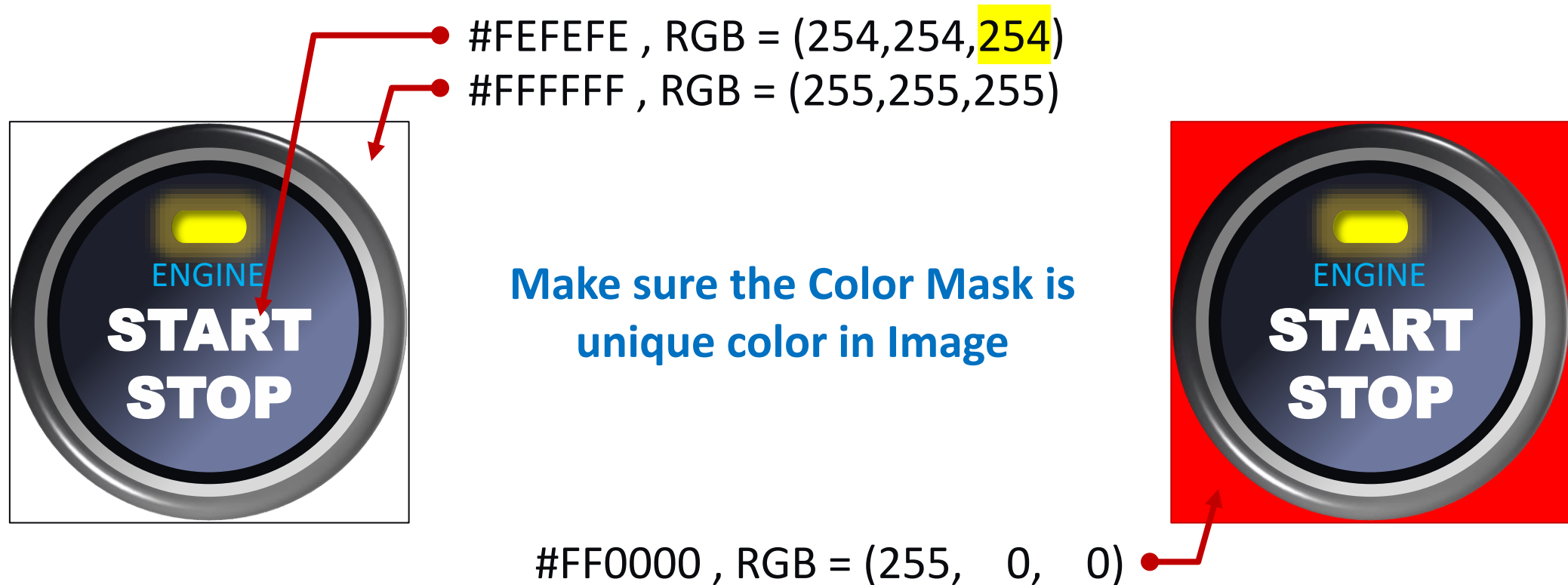
- Remember to set **Color Mode** to “**RGBA_8888**” if using BMP images.



Generate Transparent Background Image

Color Mask - Transparent Background Images

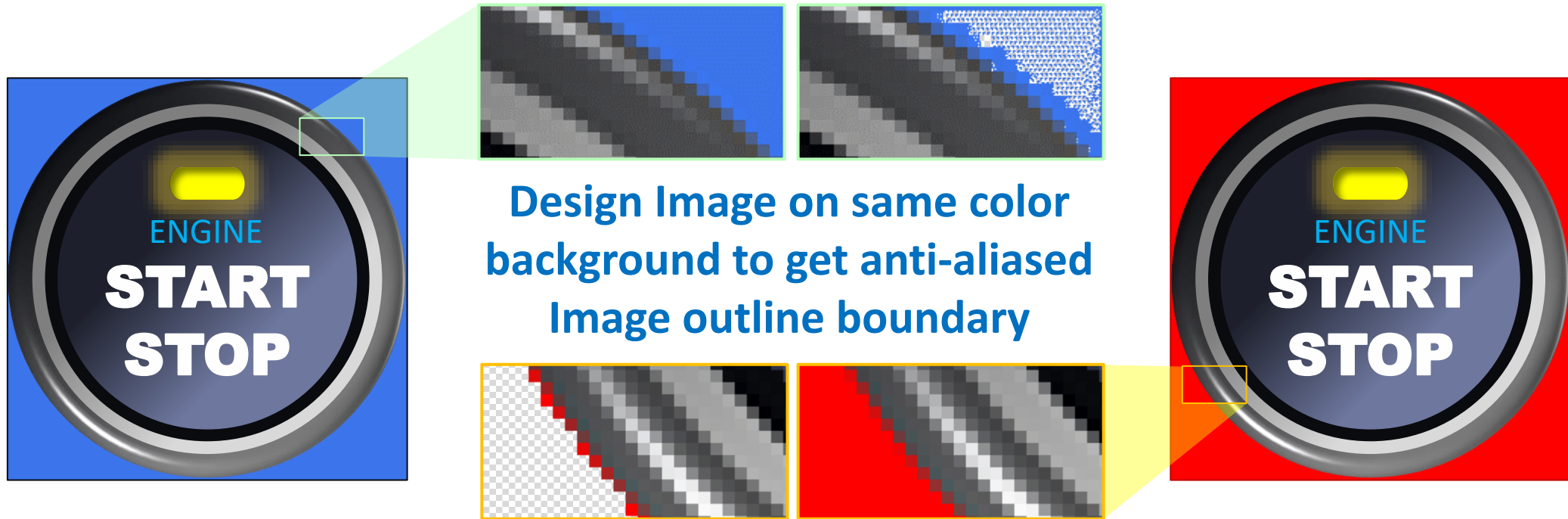
- Trick of Color Mask for transparent background.



Generate Transparent Background Image

Color Mask - Transparent Background Images

- Trick of Color Mask for transparent background.

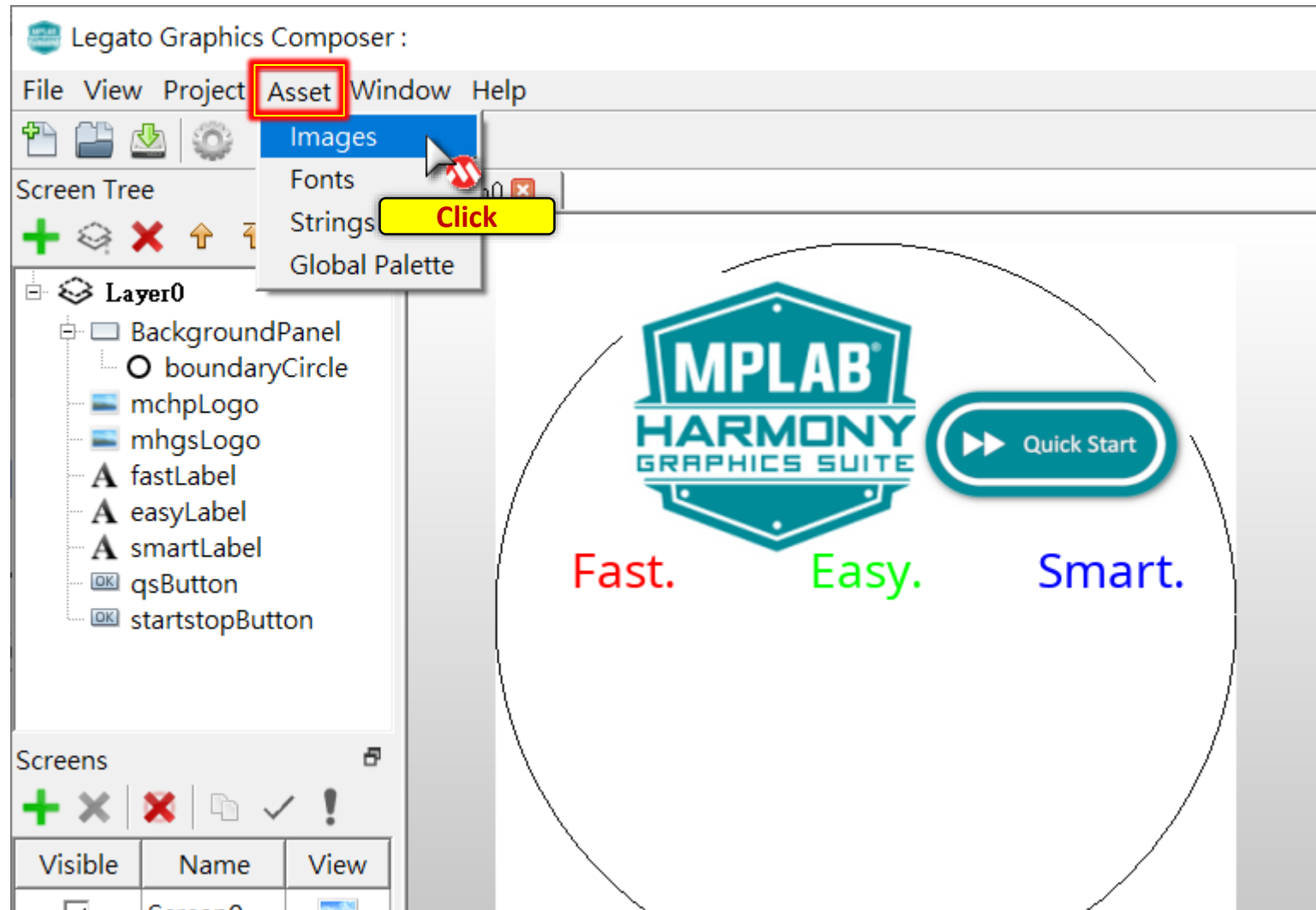


Custom Button Widget

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

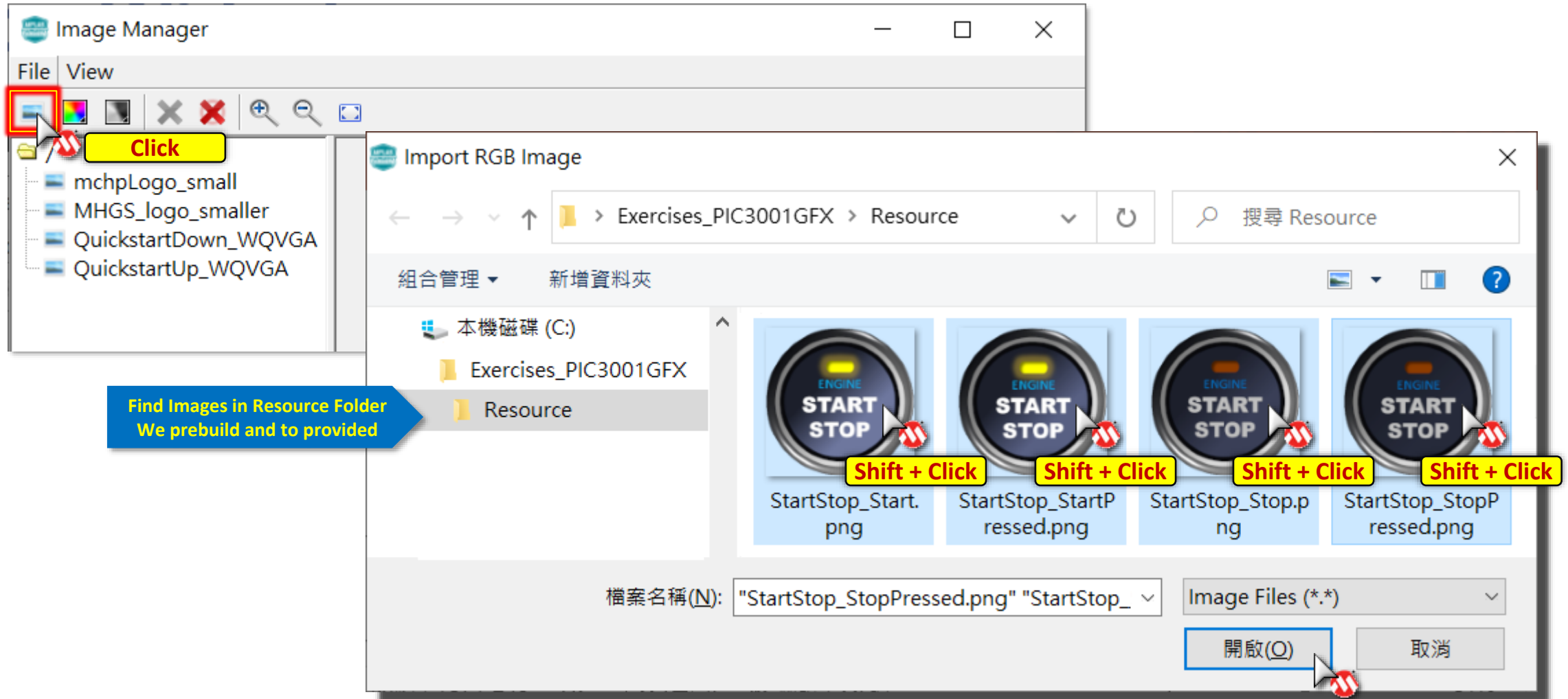
Custom Button Widget

- Add Custom Button Images through the Asset “Image Manager”.



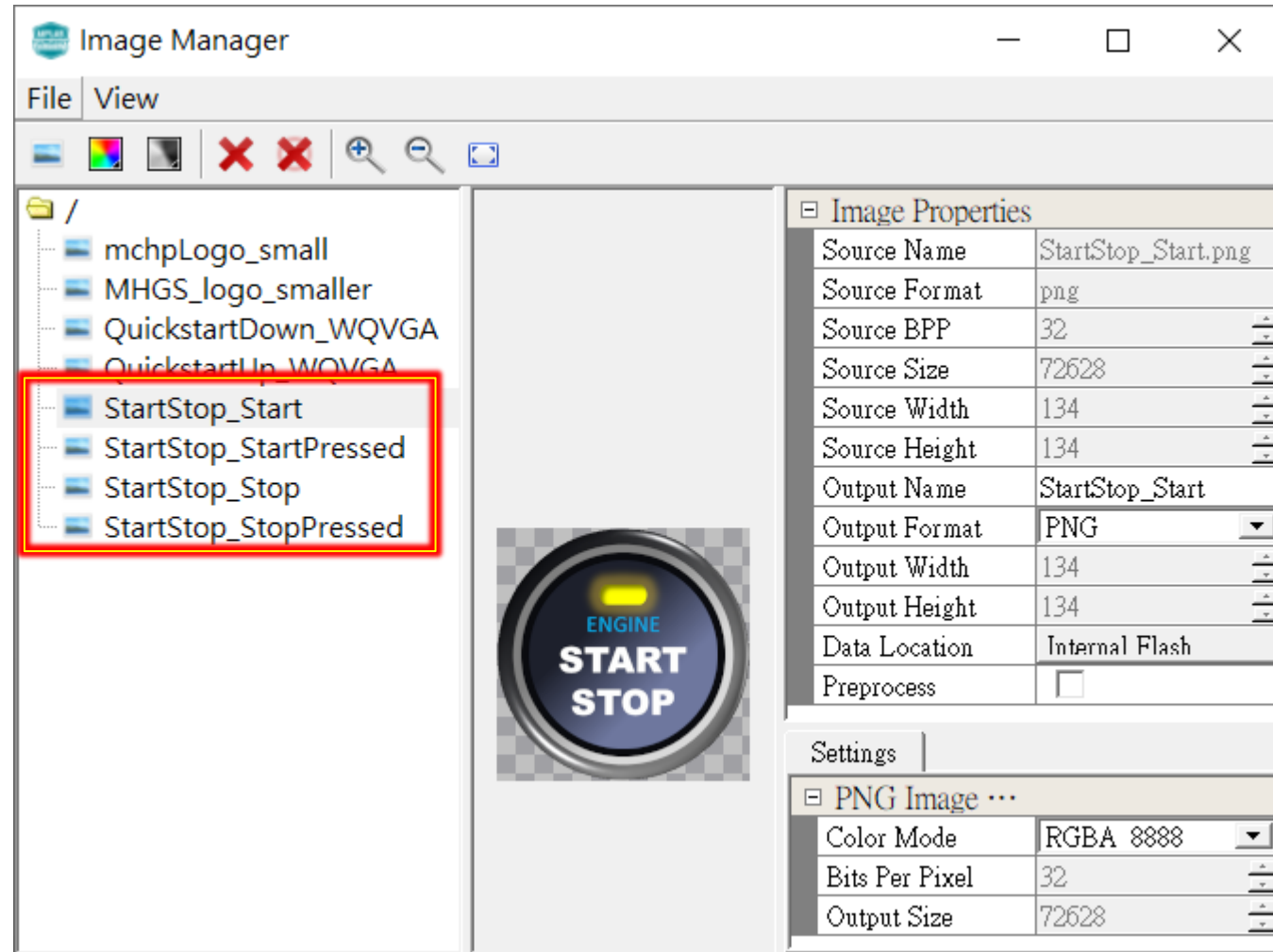
Custom Button Widget

- Add four Start/Stop PNG Images in “Image Manager”.



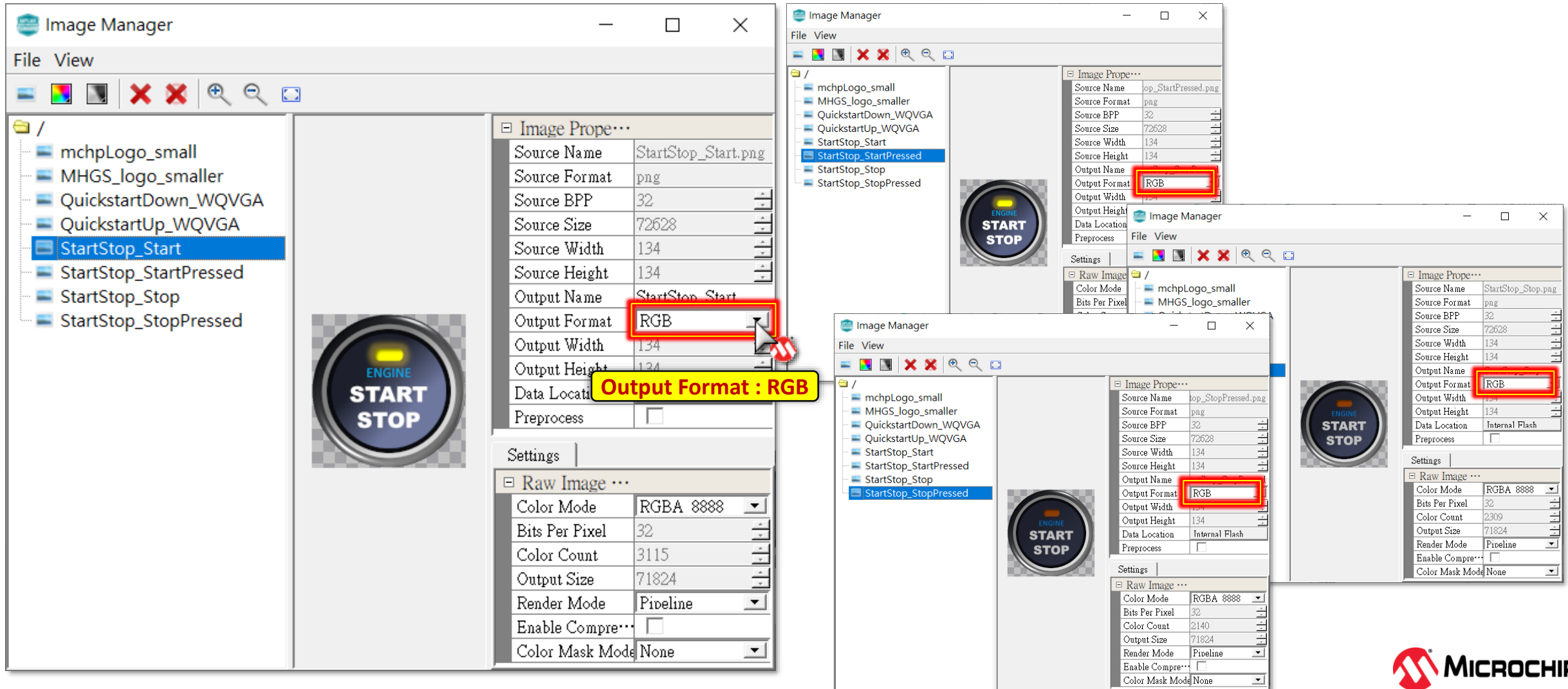
Custom Button Widget

- Imported image in “Image Manager”.



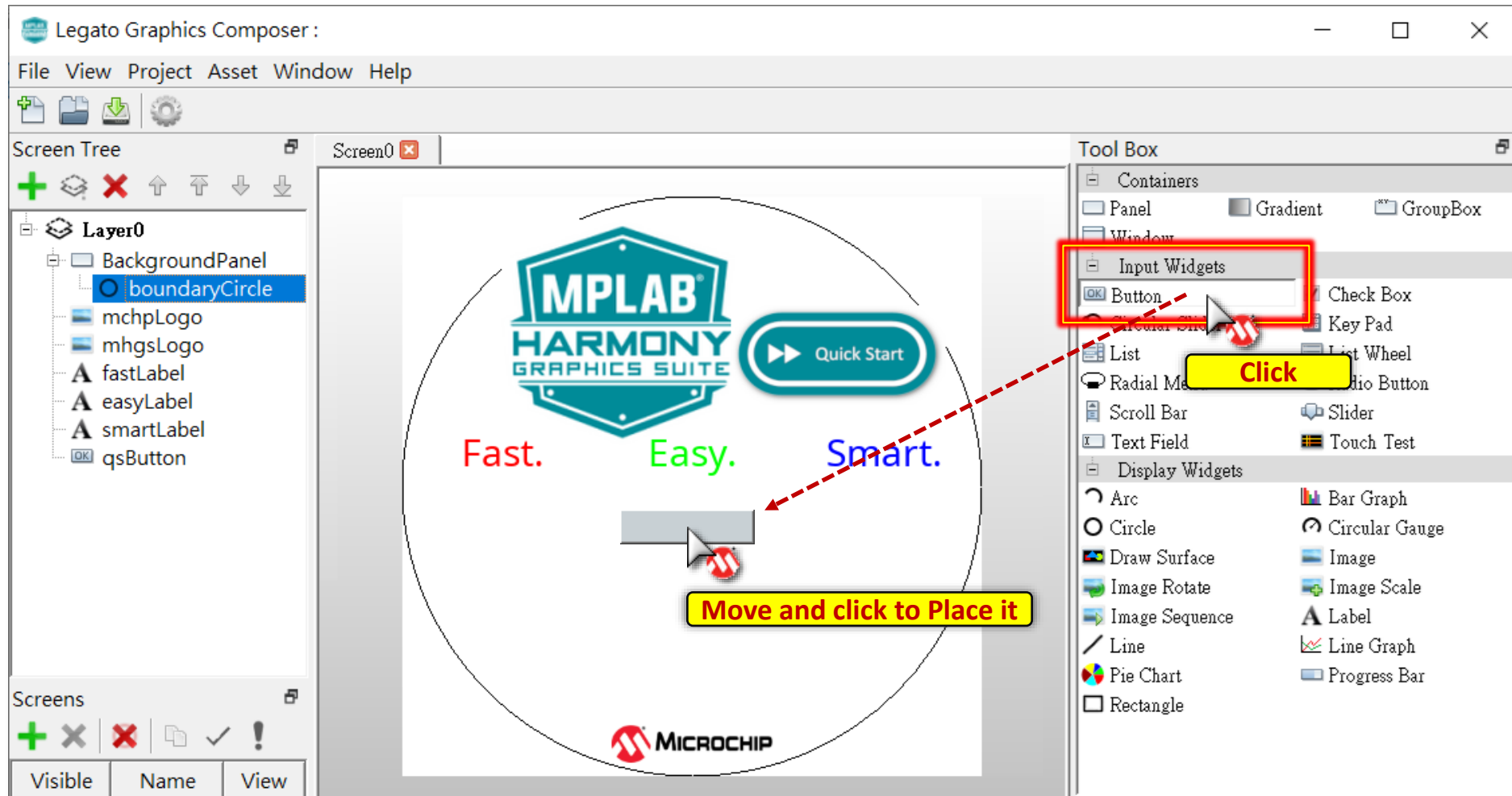
Custom Button Widget

- Select Output Format to “**RGB**” for four new **PNG** images.



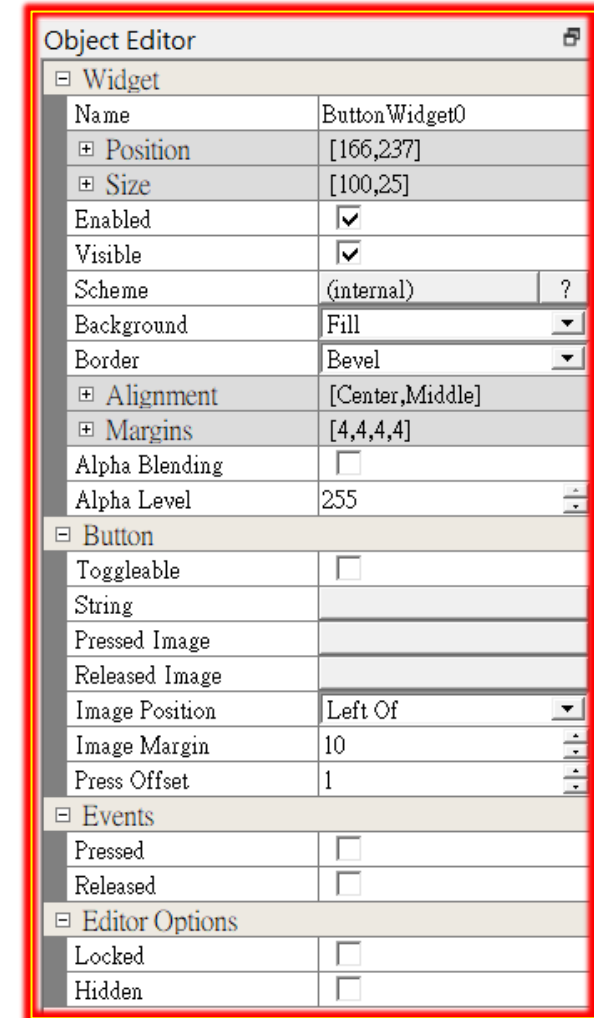
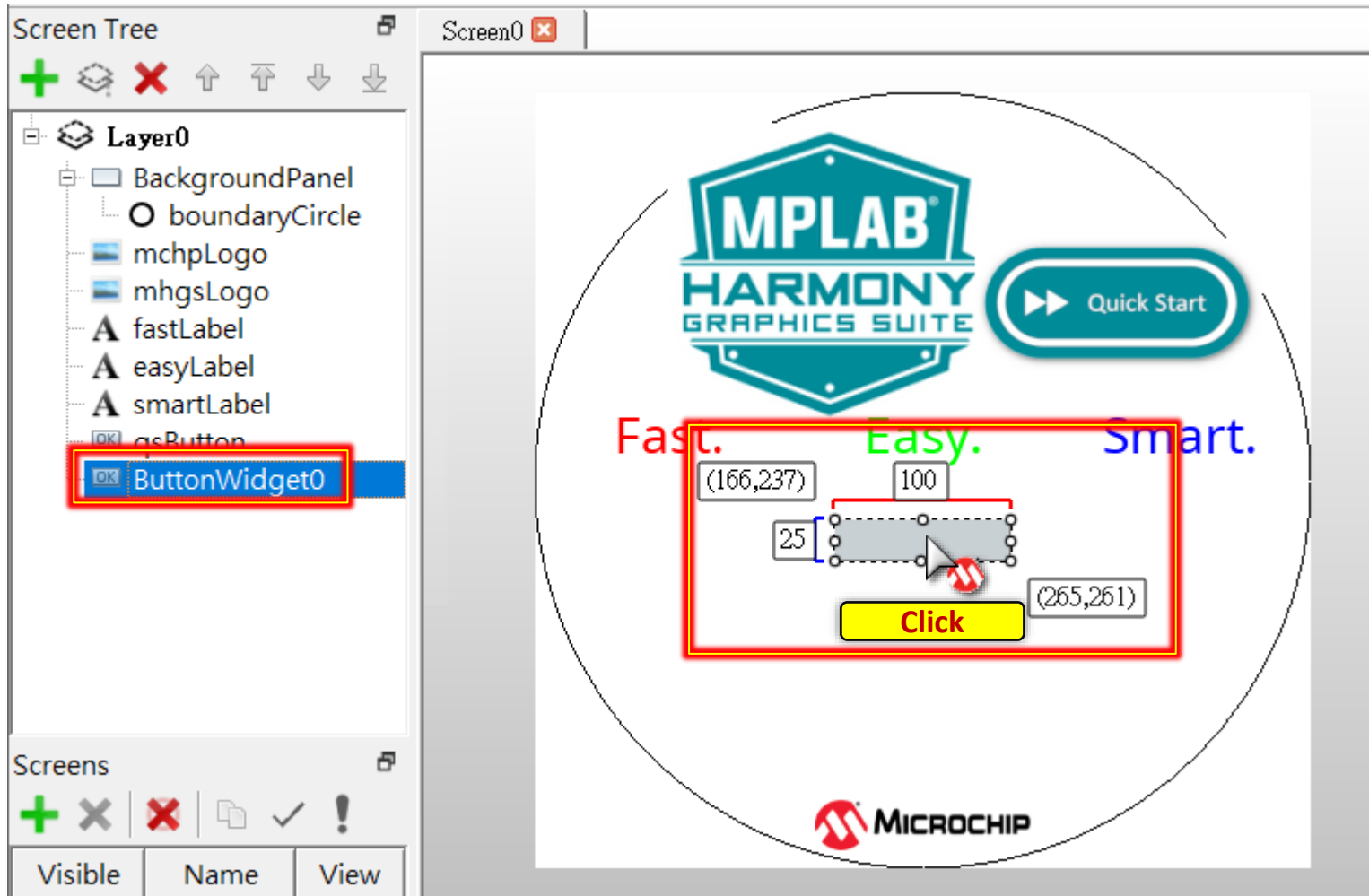
Custom Button Widget

- Click [Button] of Input Widgets window to add a button widget.



Custom Button Widget

- The new Button [ButtonWidget0] and its configuration.



Custom Button Widget

- Let's do some modify of Widget.

- Name : **startstopButton**
- Position X, Y : **149, 256**
- Size Width, Height : **134, 134**
- Background : **None**
- Border : **None**
- Pressed Image : **StartStop_StopPressed**
- Released Image : **StartStop_Stop**
- Press Offset : **0**
- Event Pressed : **Enable**
- Event Released : **Enable**

Object Editor	
Widget	
Name	startstopButton
Position	
X	149
Y	256
Size	
Width	134
Height	134
Enabled	☒
Visible	☒
Scheme	(internal) ?
Background	None
Border	None
Alignment	[Center,Middle]
Margins	[4,4,4,4]
Alpha Blending	☐
Alpha Level	255
Button	
Toggleable	☐
String	
Pressed Image	StartStop_StopPressed
Released Image	StartStop_Stop
Image Position	Left Of
Image Margin	10
Press Offset	0
Events	
Pressed	☒
Released	☒
Editor Options	
Locked	☐
Hidden	☐

Button Image added
in Image Assets Manager

Custom Button Widget

- Button Widget after configured.



Object Editor	
Widget	
Name	startstopButton
Position	
X	149
Y	256
Size	
Width	134
Height	134
Enabled	☒
Visible	☒
Scheme	(internal) ?
Background	None
Border	None
Alignment	[Center,Middle]
Margins	[4,4,4,4]
Alpha Blending	☐
Alpha Level	255
Button	
Toggleable	☐
String	
Pressed Image	StartStop_StopPressed
Released Image	StartStop_Stop
Image Position	Left Of
Image Margin	10
Press Offset	0
Events	
Pressed	☒
Released	☒
Editor Options	
Locked	☐
Hidden	☐

Custom Button Widget

- Widget Type Structure :

```
typedef struct <leWidgetType>
{
    leWidget widget;           // widget base class Structure
    const leImageWidgetVTable* fn; // widget interface function pointers (vector table) Pointer of Function
    <Other Widget depended elements>
} <leWidgetType>;
```

- Widget declarations :

```
<leWidgetType>* [Screen Name]_[Widget Name]; Pointer of Struct
```

- Understand the Widget Interface functions

```
[Screen Name]_[Widget Name]->widget.[Widget base class properties and functions];
[Screen Name]_[Widget Name]->fn->(Widget Interface Functions*)([Screen Name]_[Widget Name], ... );
```

Example:

```
Screen0_startstopButton->widget.rect.x = 100;
Screen0_startstopButton->fn->setX(Screen0_startstopButton, 100 );
Screen0_startstopButton->fn->toggleable = LE_TRUE;
```

Custom Button Widget

- Quick find Widget Interface Functions by MPLAB X IDE auto completion.

```
int main ( void )  
{  
  ...
```

Screen0_startstopButton->fn->

```
while ( true )  
{  
  /* Maintain state mach  
  SYS_Tasks ( );  
}  
  
/* Execution should not c  
  
return ( EXIT_FAILURE );  
}
```

■ _clearDirtyState	void(*) (leButtonWidget*)
■ _damageArea	void(*) (const leButtonWidget*,leRect*)
■ _destructor	void(*) (leButtonWidget*)
■ _handleEvent	void(*) (leButtonWidget*,leEvent*)
■ _increaseDirtyState	void(*) (leButtonWidget*,uint32_t)
■ _invalidateBorderAreas	void(*) (const leButtonWidget*)
■ _paint	void(*) (leButtonWidget*)
■ _setDirtyState	void(*) (leButtonWidget*,uint32_t)
■ _validateChildren	void(*) (leButtonWidget*)
■ addChild	leResult(*) (leButtonWidget*,leWidget*)
■ containsDescendant	leBool(*) (const leButtonWidget*,const leWidget*)
■ focusGainedEvent	void(*) (leButtonWidget*)
■ focusLostEvent	void(*) (leButtonWidget*)
■ getAlphaAmount	uint32_t(*) (const leButtonWidget*)
■ getAlphaEnabled	leBool(*) (const leButtonWidget*)
■ getBackgroundType	leBackgroundType(*) (const leButtonWidget*)
■ getBorderType	leBorderType(*) (const leButtonWidget*)

Custom Button Widget

- Generate Code and Implement Start/Stop Button Event Callback.

`\firmware\src\main.c`

```
volatile bool StartStop = false;
```

Start/Stop Flag

```
void event_Screen0_startstopButton_OnPressed(leButtonWidget* btn)
```

Callback function of Button pressed

```
{  
    if( StartStop )  
        Screen0_startstopButton->fn->setPressedImage(Screen0_startstopButton, (leImage*)&StartStop_StartPressed);  
    else  
        Screen0_startstopButton->fn->setPressedImage(Screen0_startstopButton, (leImage*)&StartStop_StopPressed);  
}
```

Apply Start Pressed Image

Apply Stop Pressed Image

```
void event_Screen0_startstopButton_OnReleased(leButtonWidget* btn)
```

Callback function of Button released

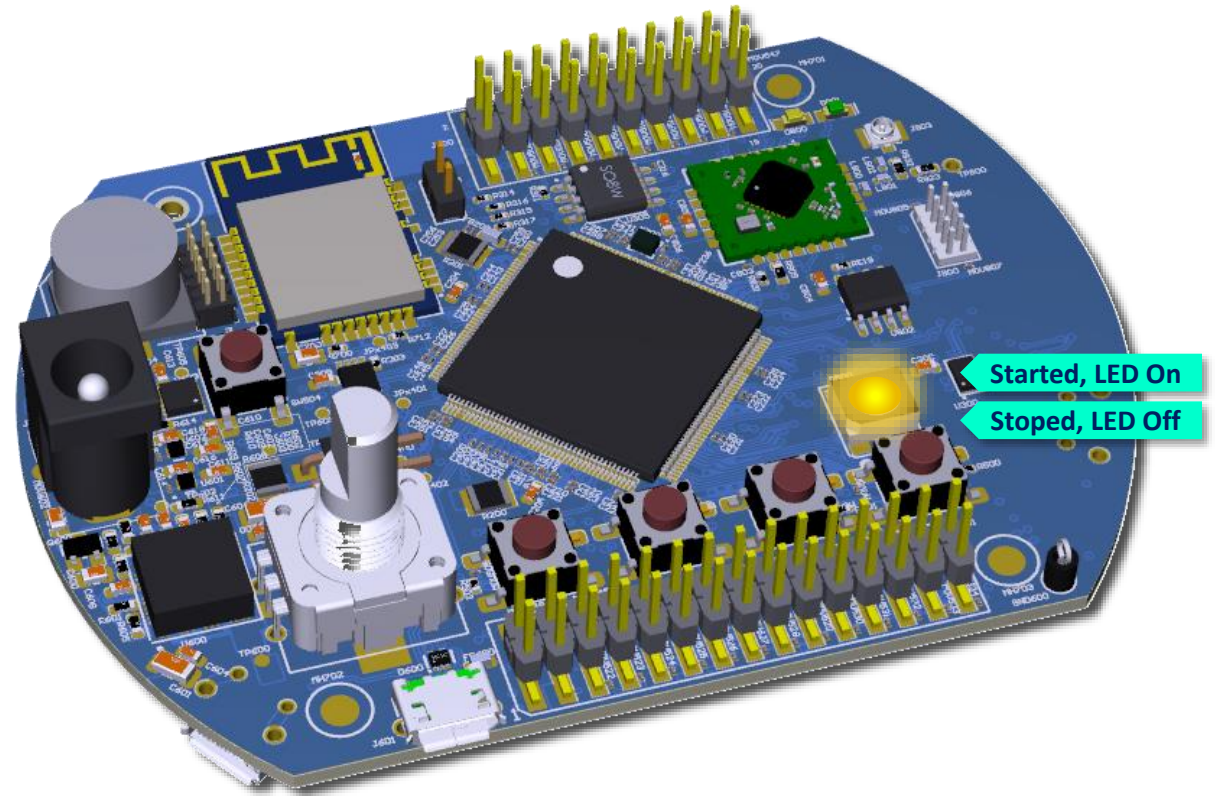
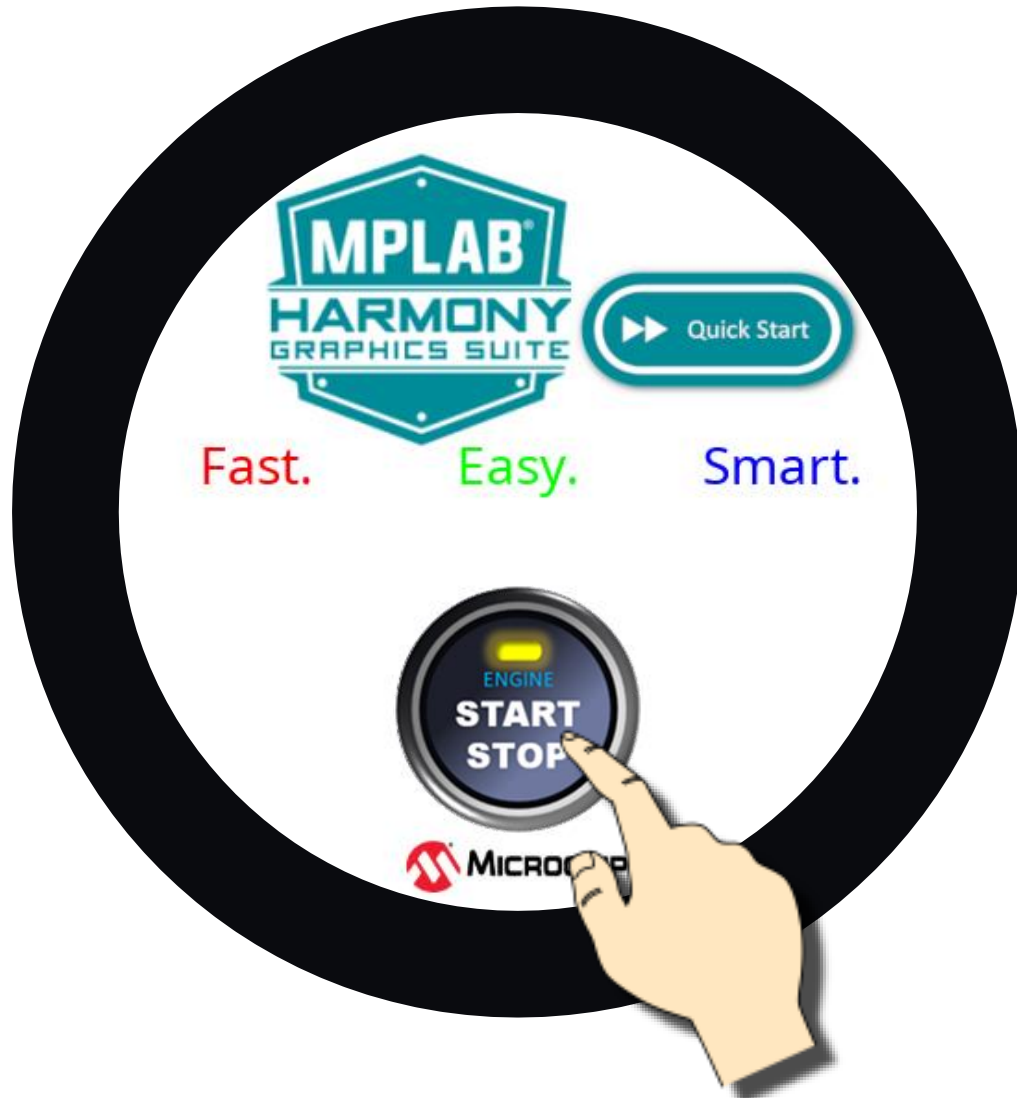
```
{  
    StartStop = !StartStop;  
    if( StartStop )  
    {  
        Screen0_startstopButton->fn->setReleasedImage(Screen0_startstopButton, (leImage*)&StartStop_Start);  
        RGB_LED_R_On(); RGB_LED_G_On();  
    }  
    else  
    {  
        Screen0_startstopButton->fn->setReleasedImage(Screen0_startstopButton, (leImage*)&StartStop_Stop);  
        RGB_LED_R_Off(); RGB_LED_G_Off();  
    }  
}
```

Apply Start Released Image
and Turned LED On

Apply Stop Released Image
and Turned LED Off

Custom Button Widget

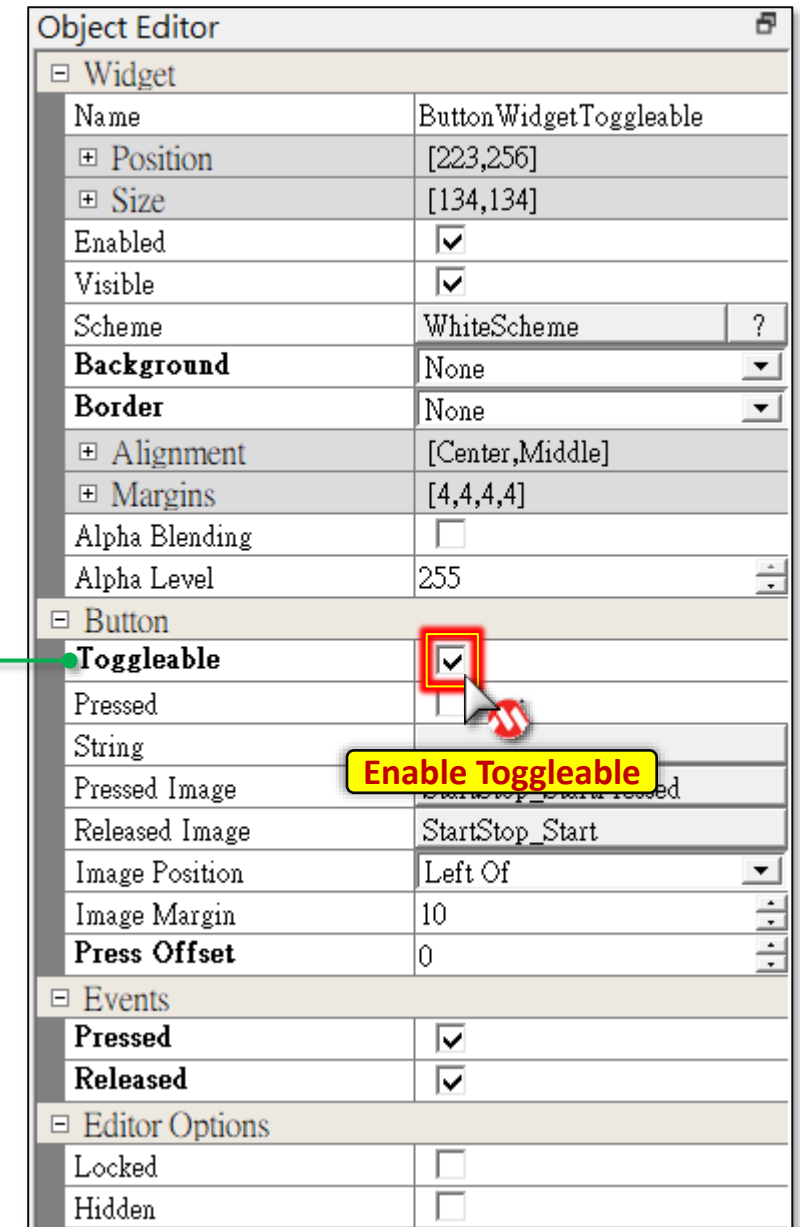
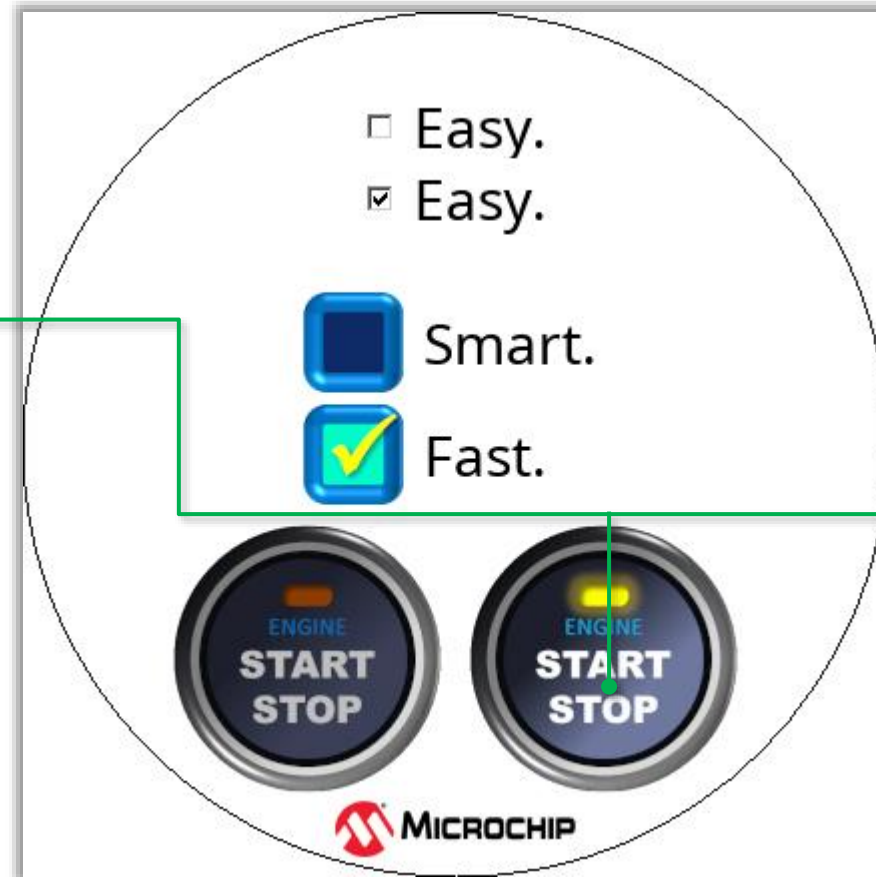
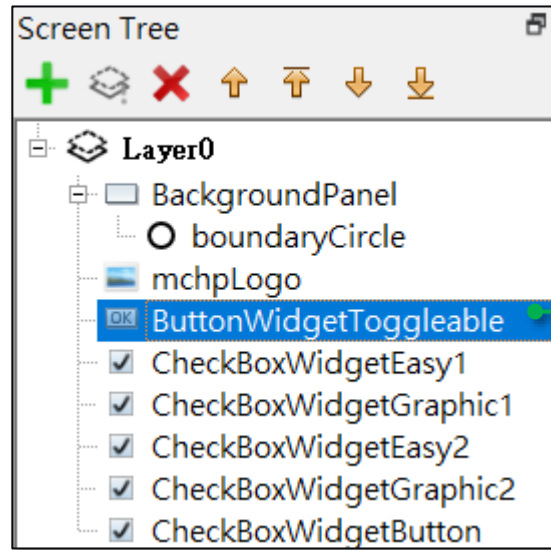
- The Start/Stop Button Widget Result after program.



Custom Button Widget

Discuss

- To enable Toggleable feature of Button.



Custom Button Widget

Discuss

- Use Check Box widgets as Button.

The screenshot illustrates the implementation of a custom button widget using checkboxes in a GUI design tool. The central canvas shows a circular panel with a Microchip logo at the bottom. Inside the panel, there are two rows of text: "Easy." and "Fast.". Each row has a checkbox to its left. The "Easy." checkbox is unchecked, and the "Fast." checkbox is checked. Below the text, there are two circular buttons labeled "ENGINE START STOP". The "Easy." button is blue with a white checkmark, and the "Fast." button is blue with a white checkmark. The "Fast." button is also highlighted with a red border.

On the left, the **Screen Tree** panel shows the hierarchy of the GUI. The **Layer0** container includes a **BackgroundPanel**, a **boundaryCircle**, a **mchpLogo**, a **ButtonWidgetToggleable**, and several **CheckBoxWidget** instances. The **CheckBoxWidgetGraphic1** is selected, and its properties are shown in the **Object Editor** panel on the right.

The **Object Editor** panel displays the properties of the selected widget:

Object Editor	
Widget	
Name	CheckBoxWidgetGraphic1
Position	[136,140]
Size	[160,50]
Enabled	☒
Visible	☒
Scheme	WhiteScheme
Background	Fill
Border	None
Alignment	[Left,Middle]
Margins	[4,4,4,4]
Alpha Blending	☐
Alpha Level	255
CheckBox	
Checked	☐
String	Smart.
Unchecked Image	Unchecked
Checked Image	Checked
Image Position	Left Of
Image Margin	10
Events	
Checked	☒
Unchecked	☒
Editor Options	
Locked	☐
Hidden	☐

At the bottom left, the **Image Manager** panel shows the **Checked** and **Unchecked** images used for the checkbox widget. The **Checked** image is a blue square with a white checkmark, and the **Unchecked** image is a blue square.

Custom Button Widget

Discuss

\firmware\src\main.c

```
volatile bool StartStop = false;
void event_Screen0_ButtonWidgetToggleable_OnPressed(lcButtonWidget* btn)
{
    if( StartStop )
        Screen0_ButtonWidgetToggleable->fn->setPressedImage(Screen0_ButtonWidgetToggleable, (lcImage*)&StartStop_StartPressed);
    else
        Screen0_ButtonWidgetToggleable->fn->setPressedImage(Screen0_ButtonWidgetToggleable, (lcImage*)&StartStop_StopPressed);
}

void event_Screen0_ButtonWidgetToggleable_OnReleased(lcButtonWidget* btn)
{
    StartStop = !StartStop;

    if( StartStop )
    {
        Screen0_ButtonWidgetToggleable->fn->setReleasedImage(Screen0_ButtonWidgetToggleable, (lcImage*)&StartStop_Start);
        RGB_LED_R_On(); RGB_LED_G_On();
    }
    else
    {
        Screen0_ButtonWidgetToggleable->fn->setReleasedImage(Screen0_ButtonWidgetToggleable, (lcImage*)&StartStop_Stop);
        RGB_LED_R_Off(); RGB_LED_G_Off();
    }
}
```


Custom Button Widget

Discuss

`\firmware\src\main.c`

```
void event_Screen0_CheckBoxWidgetEasy1_OnChecked(leCheckBoxWidget* btn) { RGB_LED_R_On(); }  
void event_Screen0_CheckBoxWidgetEasy1_OnUnchecked(leCheckBoxWidget* btn) { RGB_LED_R_Off(); }  
void event_Screen0_CheckBoxWidgetEasy2_OnChecked(leCheckBoxWidget* btn) { RGB_LED_B_Toggle(); }  
void event_Screen0_CheckBoxWidgetEasy2_OnUnchecked(leCheckBoxWidget* btn) { RGB_LED_B_Toggle(); }
```

Check Box Easy1 to control Red LED

Check Box Easy2 to toggle Blue LED

```
void event_Screen0_CheckBoxWidgetGraphic1_OnChecked(leCheckBoxWidget* btn)  
{ Screen0_CheckBoxWidgetGraphic1->fn->setString(Screen0_CheckBoxWidgetGraphic1, (leString*)&string_Fast); }
```

Check Box Graphic 1
Checked : Fast
Unchecked : Smart

```
void event_Screen0_CheckBoxWidgetGraphic1_OnUnchecked(leCheckBoxWidget* btn)  
{ Screen0_CheckBoxWidgetGraphic1->fn->setString(Screen0_CheckBoxWidgetGraphic1, (leString*)&string_Smart); }
```

```
void event_Screen0_CheckBoxWidgetGraphic2_OnChecked(leCheckBoxWidget* btn)  
{ Screen0_CheckBoxWidgetGraphic2->fn->setString(Screen0_CheckBoxWidgetGraphic2, (leString*)&string_Fast); }
```

Check Box Graphic 2
Checked : Fast
Unchecked : Smart

```
void event_Screen0_CheckBoxWidgetGraphic2_OnUnchecked(leCheckBoxWidget* btn)  
{ Screen0_CheckBoxWidgetGraphic2->fn->setString(Screen0_CheckBoxWidgetGraphic2, (leString*)&string_Smart); }
```

```
void event_Screen0_CheckBoxWidgetButton_OnChecked(leCheckBoxWidget* btn)  
{ RGB_LED_R_On(); RGB_LED_G_On(); }
```

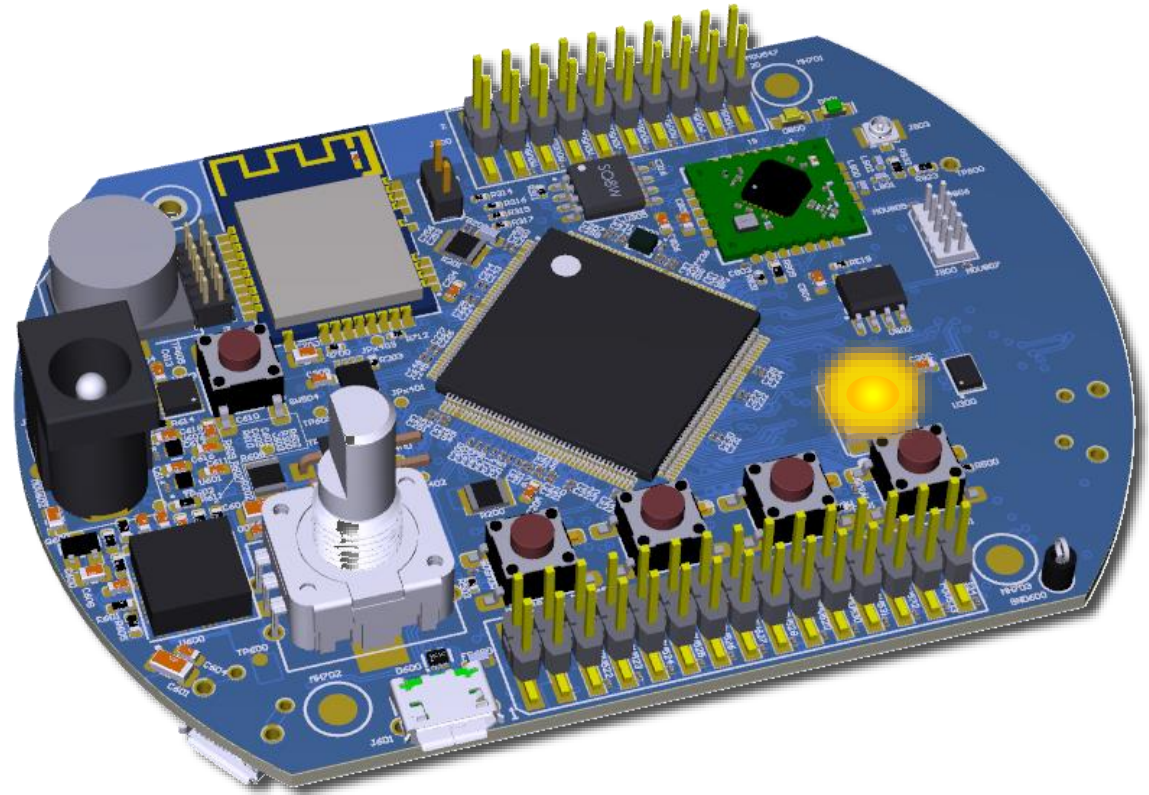
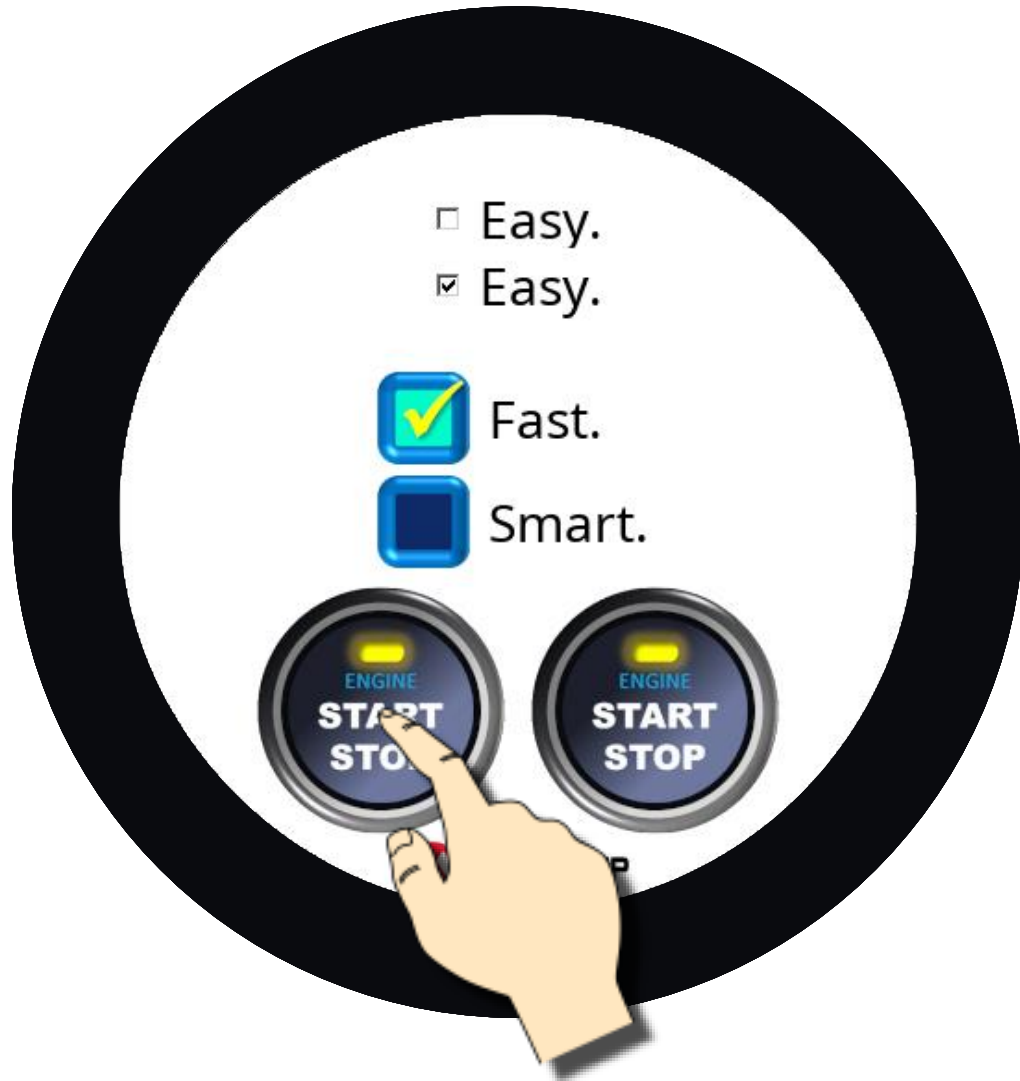
```
void event_Screen0_CheckBoxWidgetButton_OnUnchecked(leCheckBoxWidget* btn)  
{ RGB_LED_R_Off(); RGB_LED_G_Off(); }
```

Start-Stop Button
Start : Yellow LED On
Stop : Yellow LED Off

```
int main ( void ) { ... }
```

Custom Button Widget

Discuss

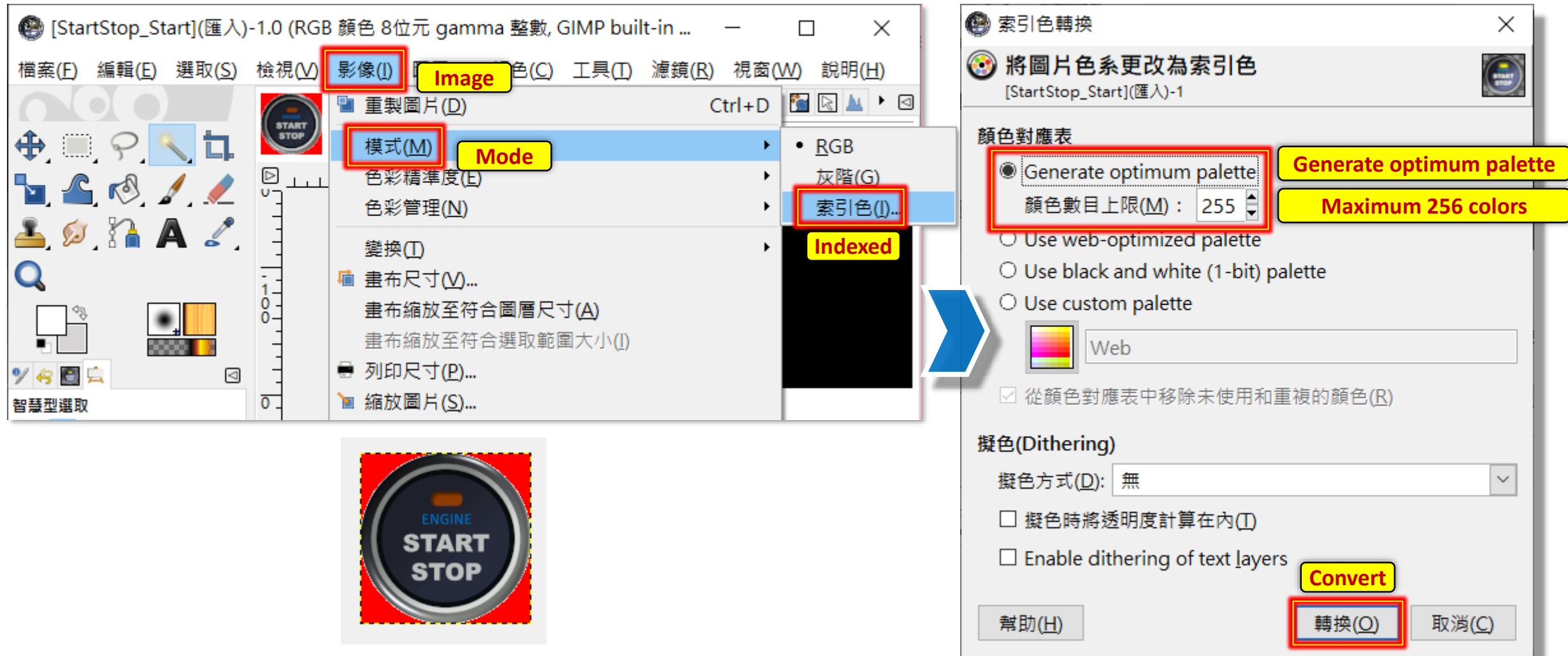


Deducing Image Storage Space with Optimum Color Palette

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Deducing Image Storage Space with Optimum Color Palette

- Use GIMP  to open BMP image file and generate optimum palette.



The screenshot illustrates the steps to generate an optimum color palette in GIMP. On the left, the GIMP main window shows the 'Image' menu open, with 'Mode' selected and 'Indexed' chosen. A blue arrow points from the 'Indexed' option to the '索引色轉換' (Index Color Conversion) dialog box on the right. In the dialog, 'Generate optimum palette' is selected, and the 'Maximum 256 colors' option is highlighted. The 'Convert' button is also highlighted. Below the main window, a small image of a 'START STOP ENGINE' button is shown with a red dashed selection box.

Image

Mode

Indexed

索引色轉換

將圖片色系更改為索引色

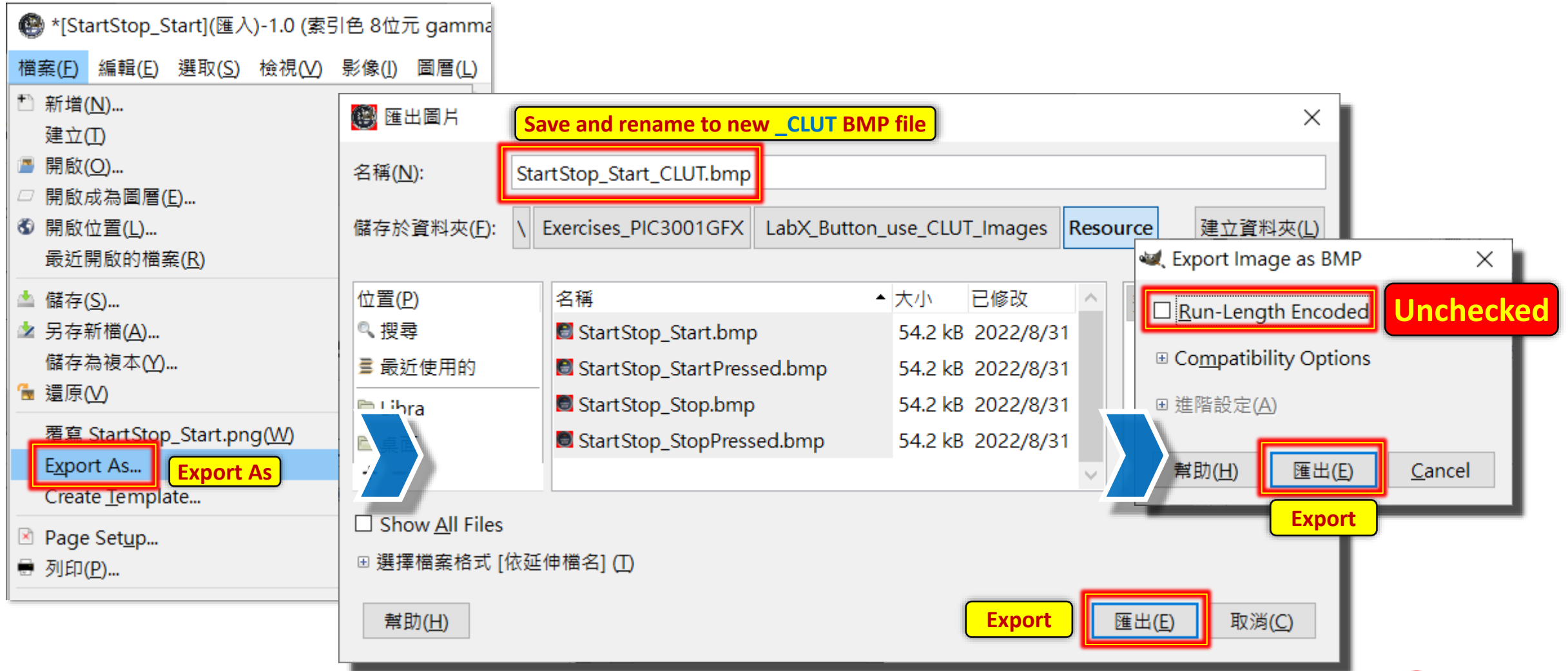
Generate optimum palette

Maximum 256 colors

Convert

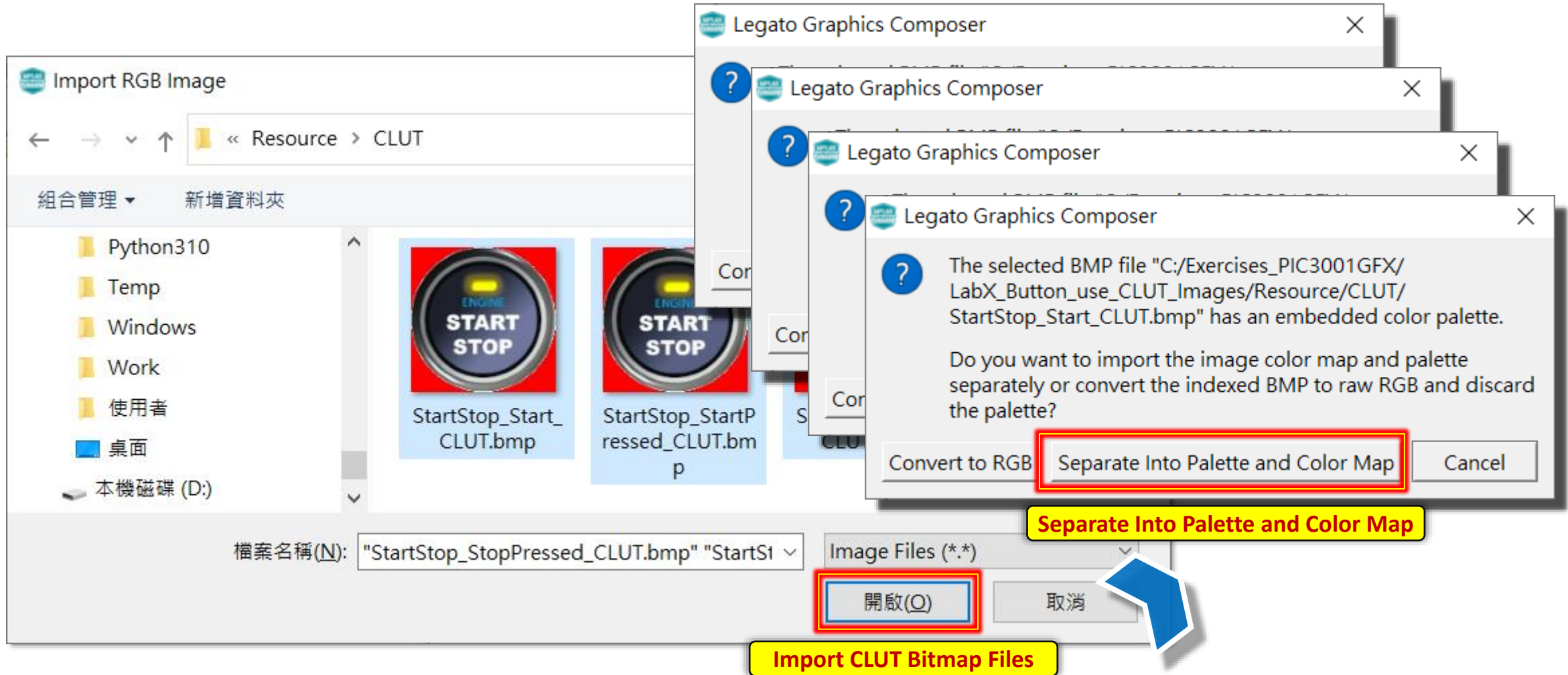
Deducing Image Storage Space with Optimum Color Palette

- Export BMP with Color Look-Up Table (CLUT) in GIMP  .



Deducing Image Storage Space with Optimum Color Palette

- Import CLUT Bitmap Images in MHGS .



Deducing Image Storage Space with Optimum Color Palette

- Each CLUT Bitmap image has an extra Color Palette with it.

The screenshot illustrates the Image Manager interface in the MPLAB IDE. It shows a file tree on the left with various image files. Two red lines connect the files `StartStop_Start_CLUT` and `StartStop_Start_CLUT_palette` to their respective property windows on the right. The `StartStop_Start_CLUT` window shows the image preview and its properties. The `StartStop_Start_CLUT_palette` window shows the palette image and its properties, including a color bar at the bottom.

Image Manager File List:

- mchpLogo_small
- MHGS_logo_smaller
- QuickstartDown_WQVGA
- QuickstartUp_WQVGA
- StartStop_Start_CLUT
- StartStop_Start_CLUT_palette
- StartStop_StartPressed_CLUT
- StartStop_StartPressed_CLUT_palette
- StartStop_Stop_CLUT
- StartStop_Stop_CLUT_palette
- StartStop_StopPressed
- StartStop_StopPressed

StartStop_Start_CLUT Image Properties:

Image Properties	
Source Name	StartStop_Start_CLUT.bmp
Source Format	bmp
Source BPP	8
Source Size	19382
Source Width	134
Source Height	134
Output Name	StartStop_Start_CLUT
Output Format	Indexed
Output Width	134
Output Height	134
Data Location	Internal Flash
Preprocess	☐

StartStop_Start_CLUT_palette Image Properties:

Image Properties	
Source Name	StartStop_Start_CLUT_palette
Source Format	bmp
Source BPP	8
Source Size	17956
Source Width	256
Source Height	1
Output Name	StartStop_Start_CLUT_palette
Output Format	RGB
Output Width	256
Output Height	1
Data Location	Internal Flash
Preprocess	☐

Color Map Settings (for StartStop_Start_CLUT):

Color Map Settings	
Palette Image	StartStop_Start_CLUT_palette
Bits Per Pixel	8
Color Count	254
Output Size	17956
Enable Compression	☐
Index Mask Mode	None

Palette Settings (for StartStop_Start_CLUT_palette):

Palette Settings	
Color Mode	RGB 888
Bits Per Pixel	24
Color Count	256
Output Size	768

Optimum 256 colors palette

Deducing Image Storage Space with Optimum Color Palette

- Use Color Mask to transparent background of images.
- The Color Mask has named “**Index Value**” to different with normal BMP.

The screenshot illustrates the process of selecting a color mask for a BMP image. The **Image Manager** window on the left shows a list of files, including **StartStop...**. The **Image Color Picker** window on the right shows a red background with a circular button labeled **ENGINE START STOP**. A yellow box labeled **CLUT BMP images** highlights the **CLUT.bmp** file in the **Image Pro...** settings. A yellow box labeled **Pick Color** points to the **Color Map ...** settings. A yellow box labeled **Index Value** points to the **Index Mask M...** dropdown menu. A yellow box labeled **Click** points to the **Index Value** dropdown menu.

Image Manager

File View

CLUT BMP images

Image Pro...

Source Name	Star CLUT.bmp
Source Format	bmp
Source BPP	8
Source Size	19382
Source Width	134

Settings

Color Map ...

Palette Image	StartStop...
Bits Per Pixel	8
Color Count	254
Output Size	17956
Enable Compr...	☐
Index Mask M...	None

Index Value

Image Color Picker

Pick Color

OK Cancel

Image Pro...

Source Name	Start_CLUT.bmp
Source Format	bmp
Source BPP	8
Source Size	19382
Source Width	134
Source Height	134

Settings

Color Map ...

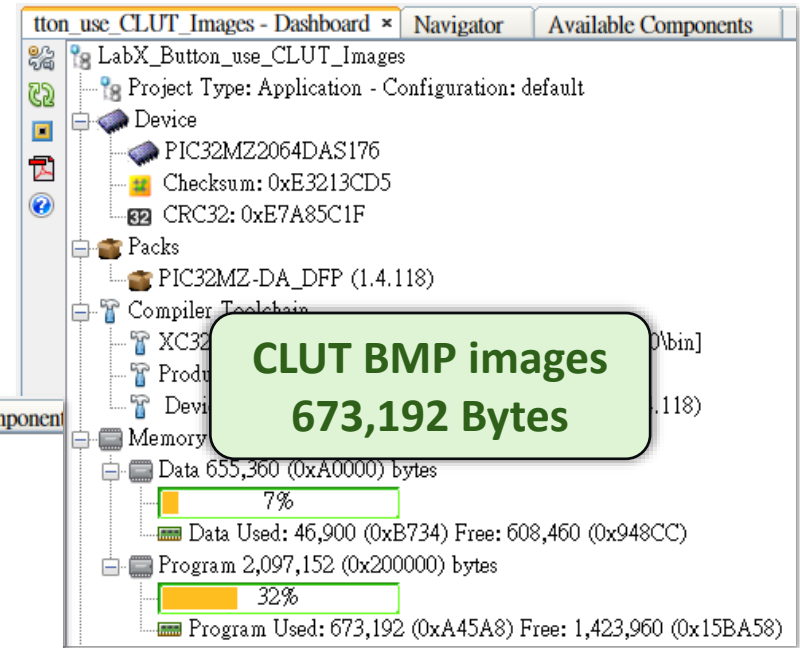
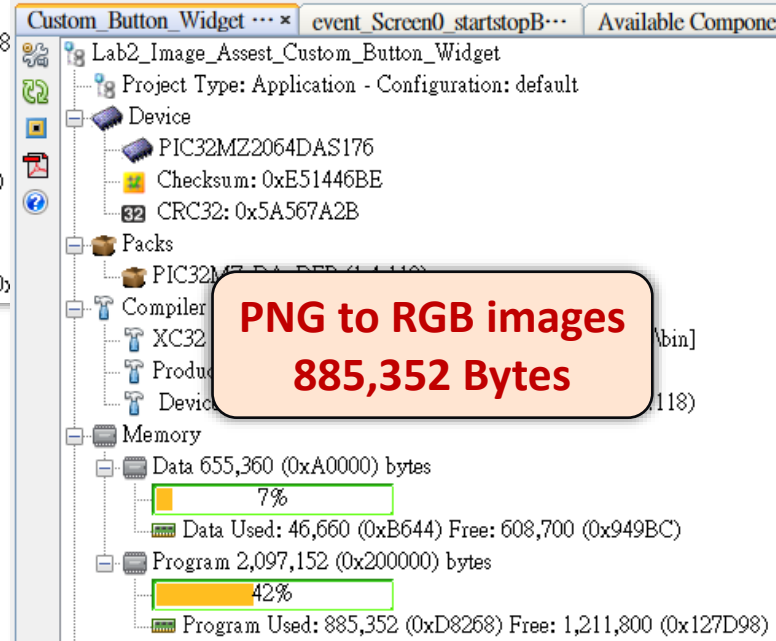
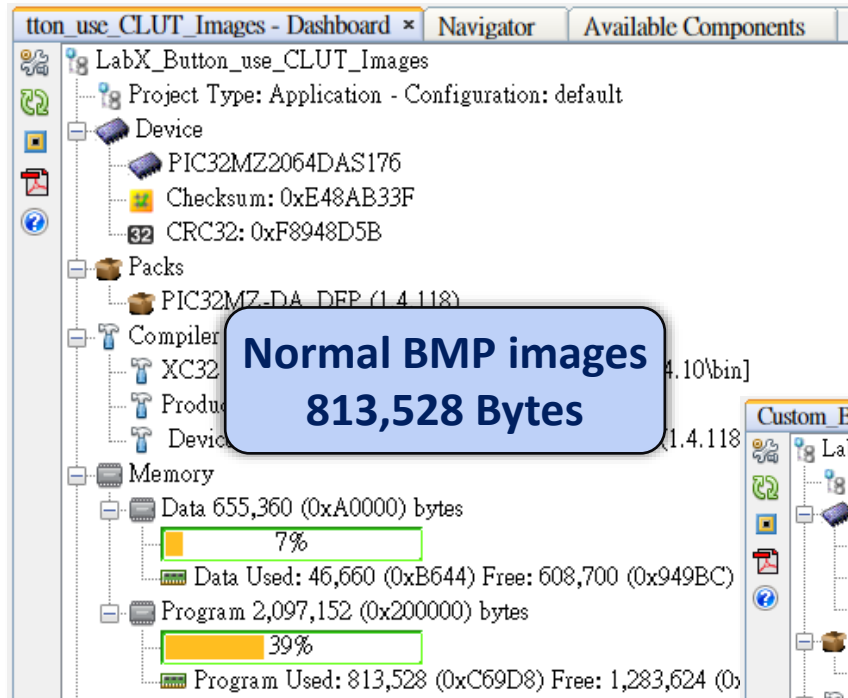
Palette Image	StartStop_Start_CL
Bits Per Pixel	8
Color Count	254
Output Size	17956
Enable Compr...	☐
Index Mask M	Index Value
Mask Index	

Click

MICROCHIP

Deducing Image Storage Space with Optimum Color Palette

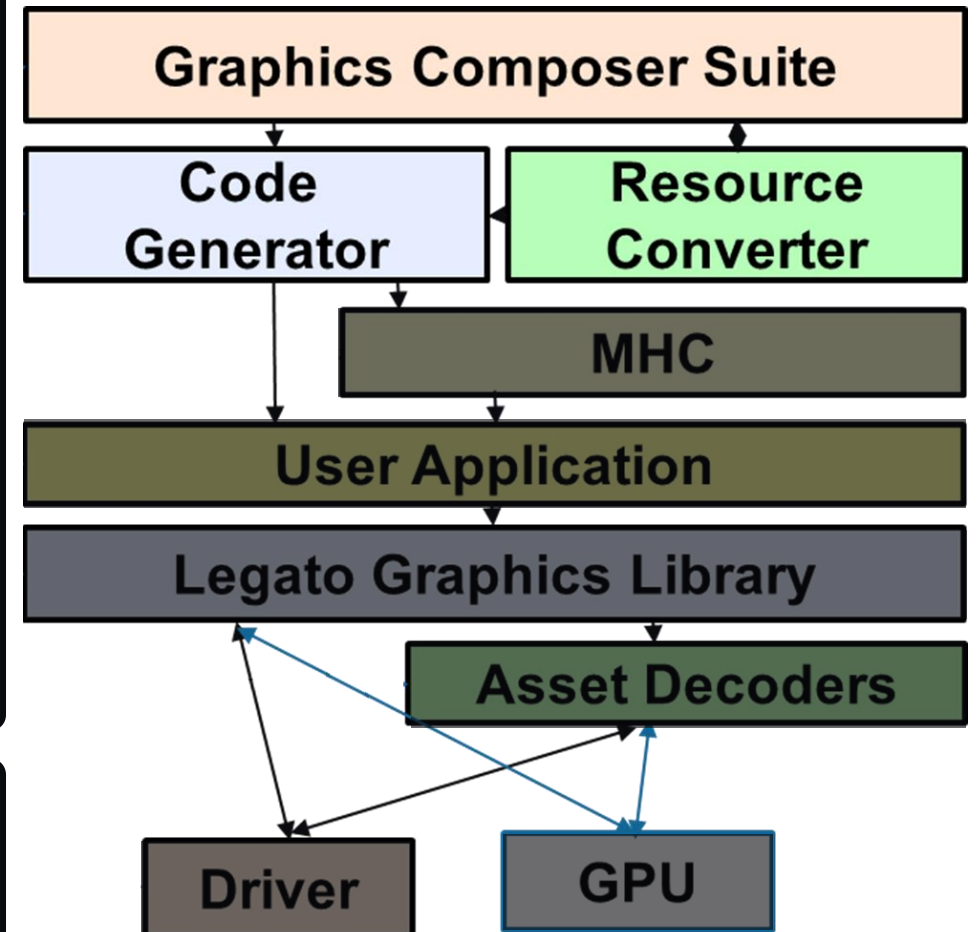
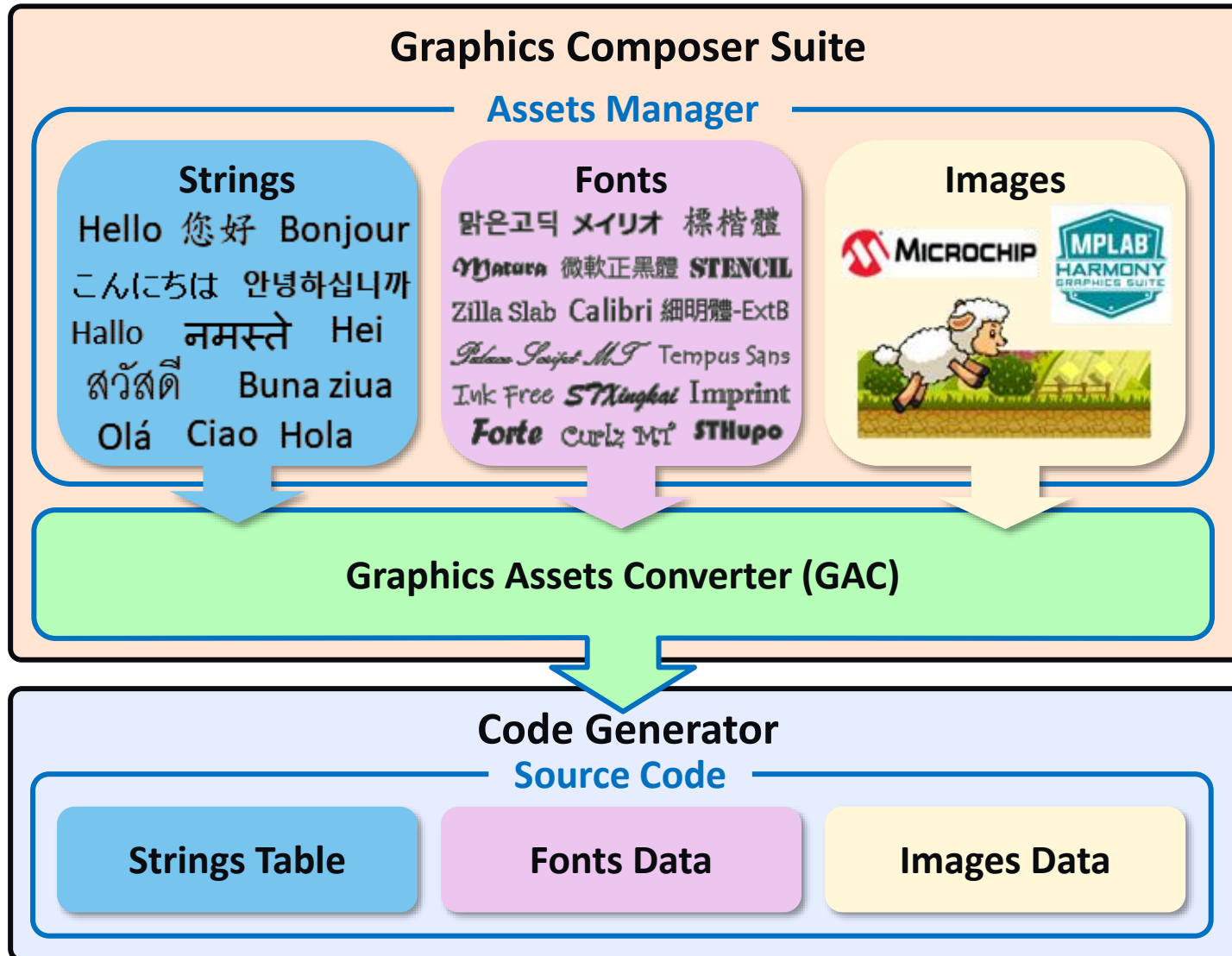
- Compare the Flash size between normal BMP, CLUT BMP and PNG images.



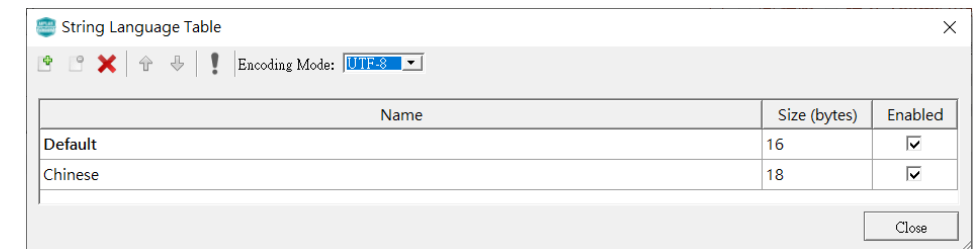
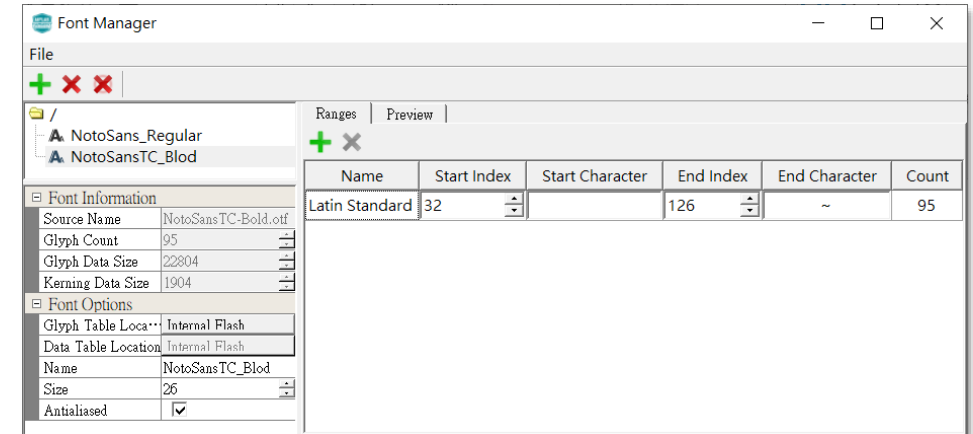
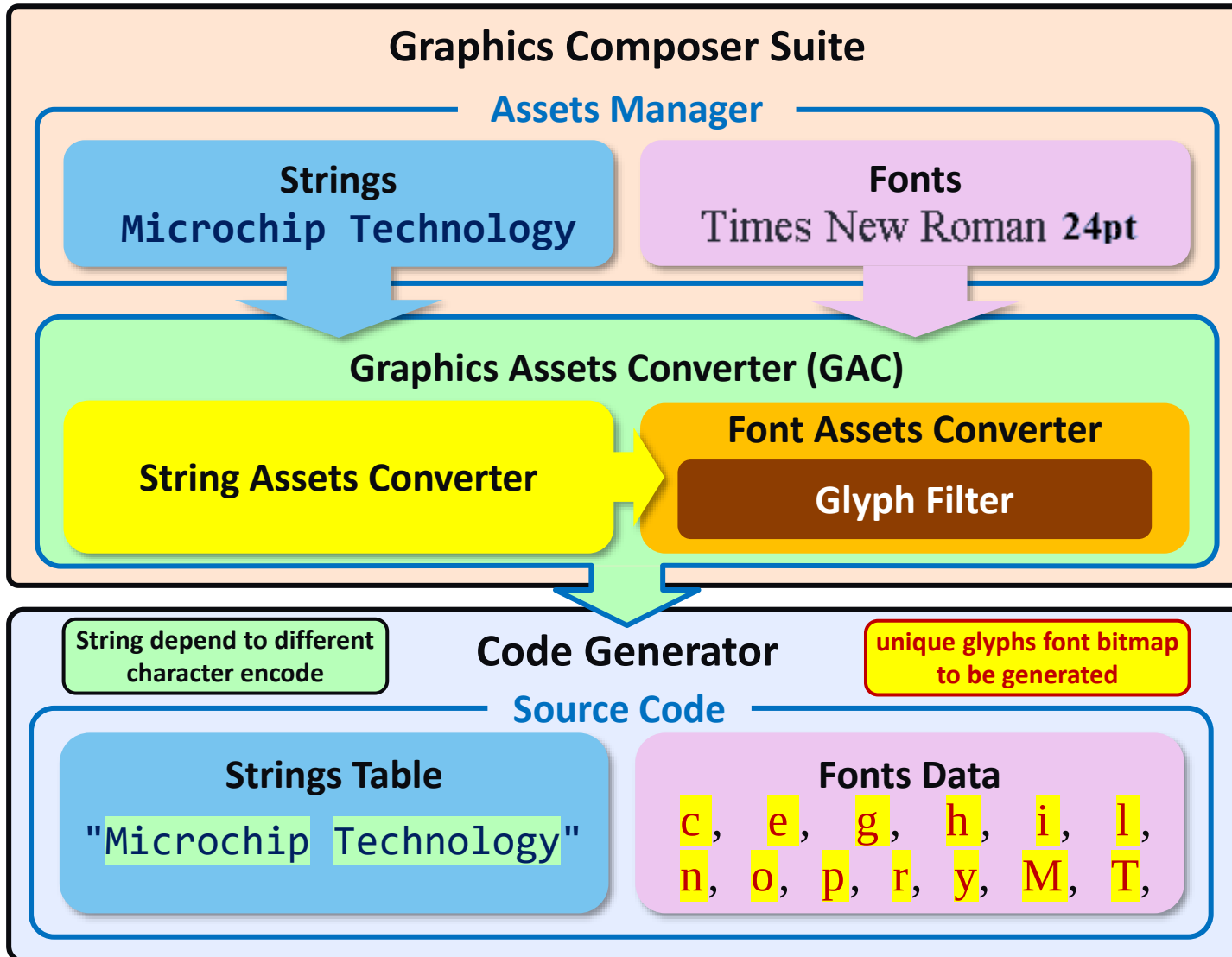
Font and String Assets Manager

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Graphics Assets Converter (GAC)



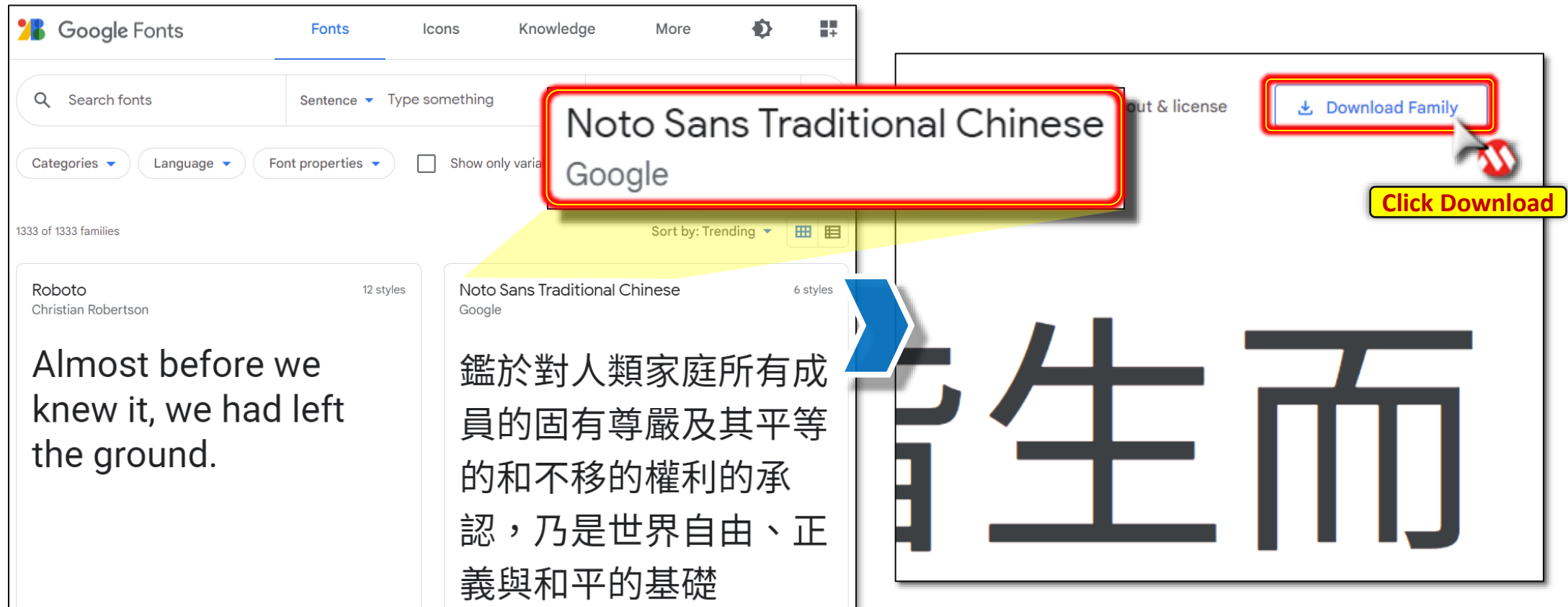
Font and String Assets Manager



Font Assets Manager

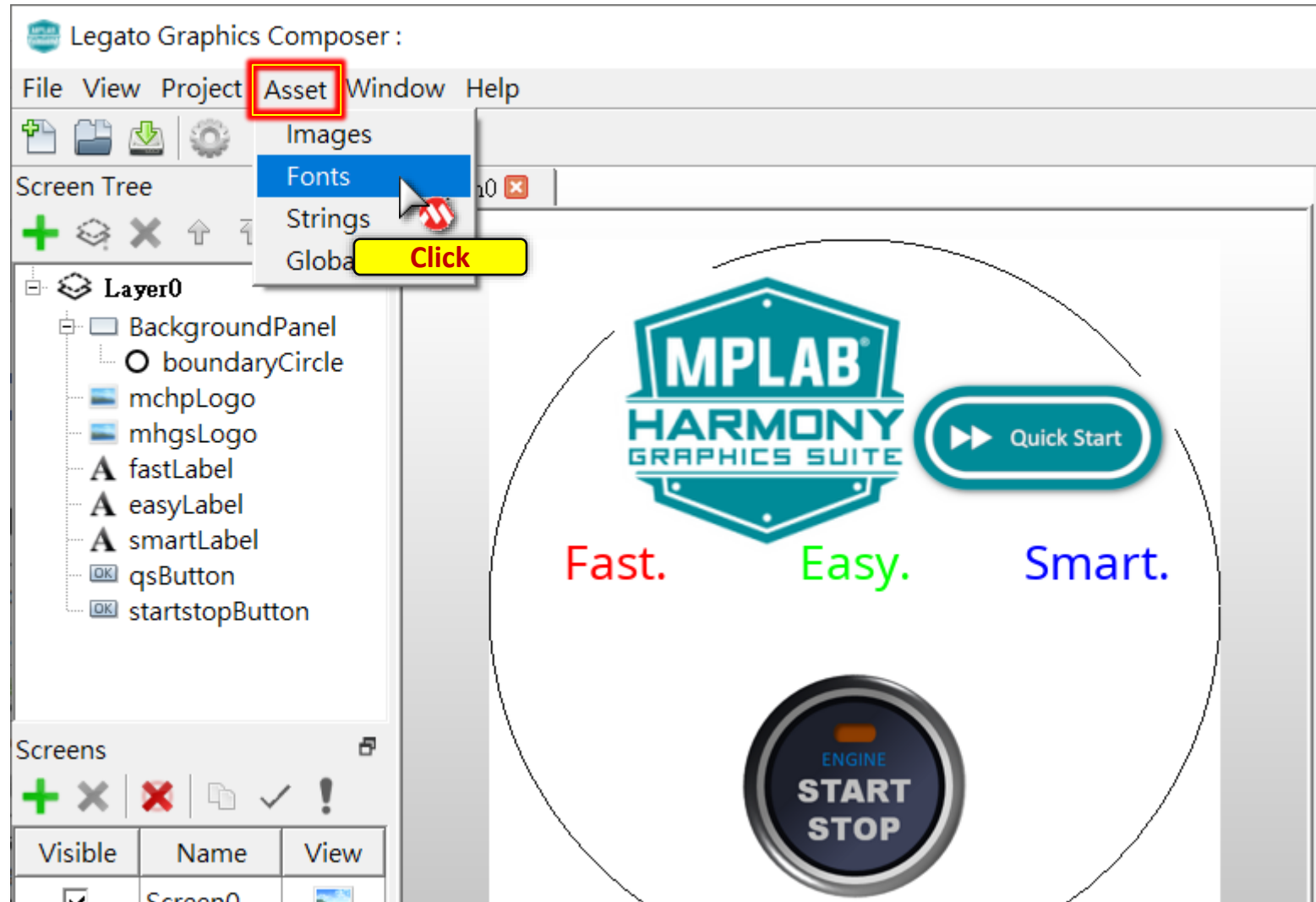
- Download free Chinese fonts from Google Fonts.

<https://fonts.google.com>



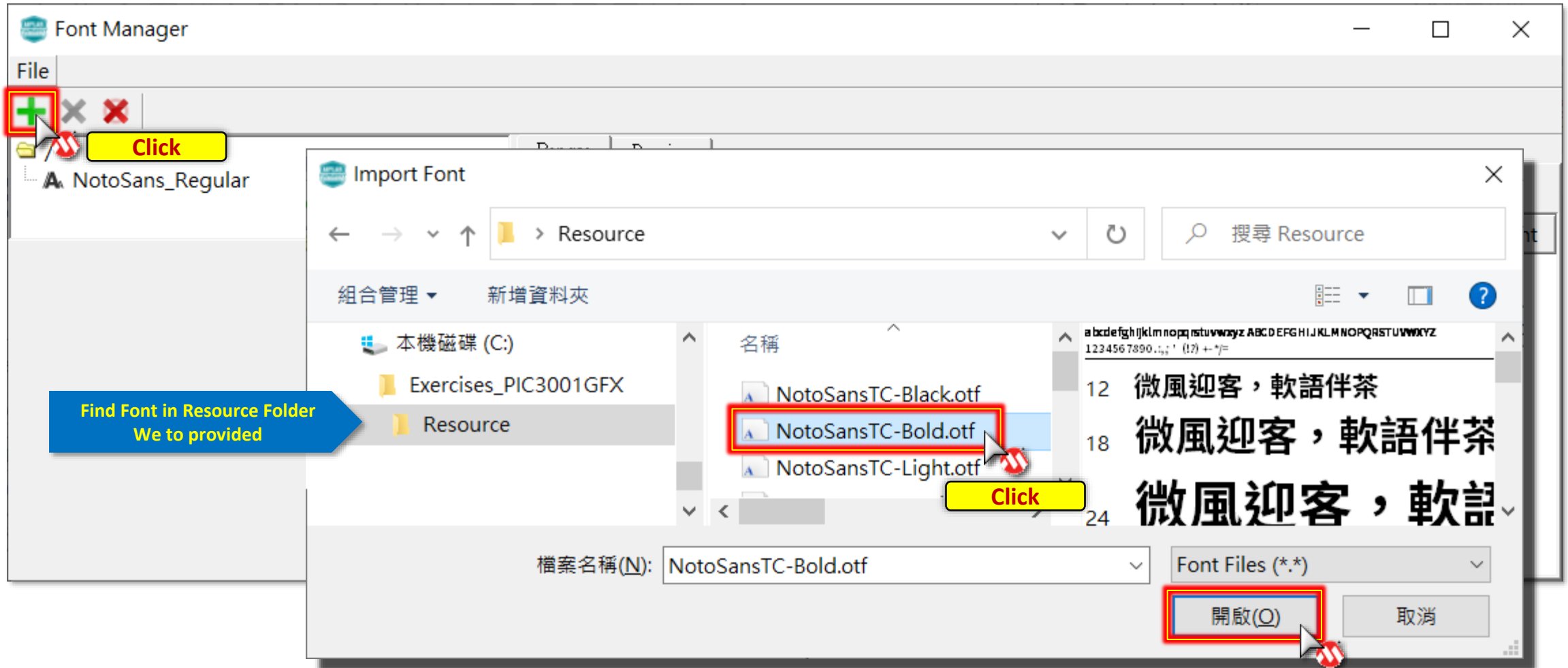
Font Assets Manager

- Add Fonts through the Asset “Font Manager”.



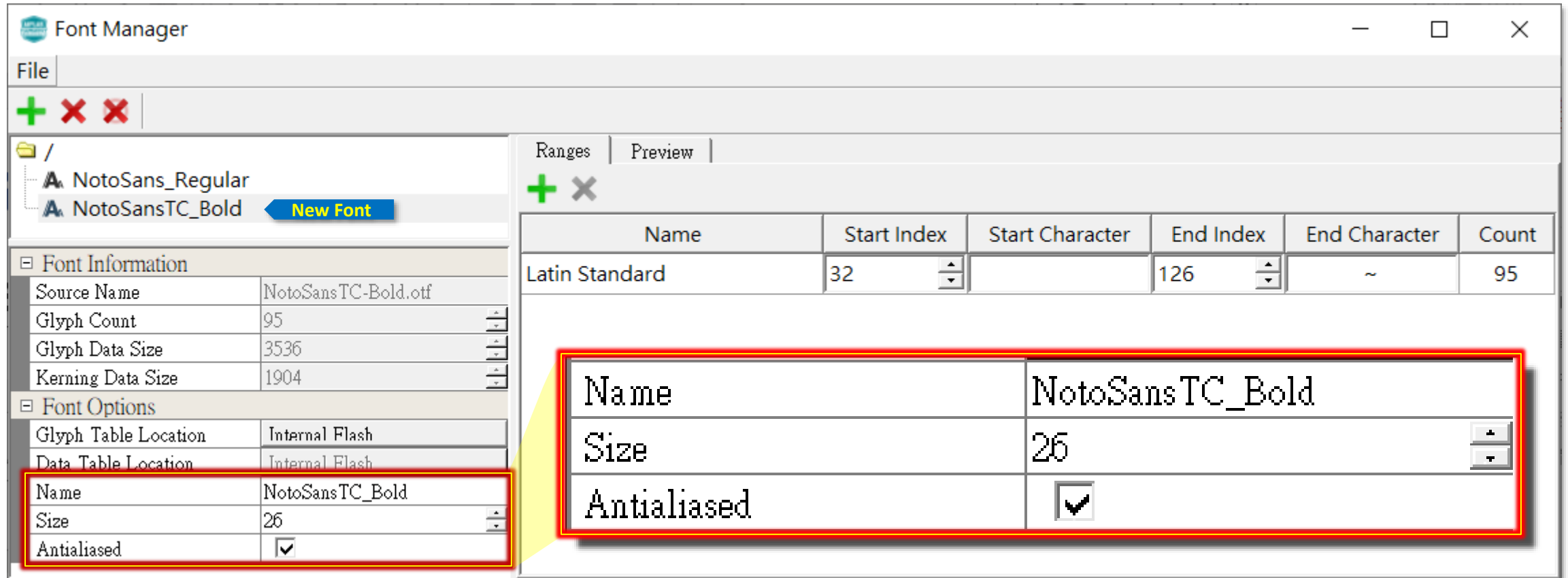
Font Assets Manager

- Add Chinese font [**NotoSansTC-Bold.otf**] from folder “**Resource**”



Font Assets Manager

- Rename to **NotoSansTC_Blod**, Size to **26** and Enable **Antialiased**.



智慧

Anti-aliased Disable

智慧

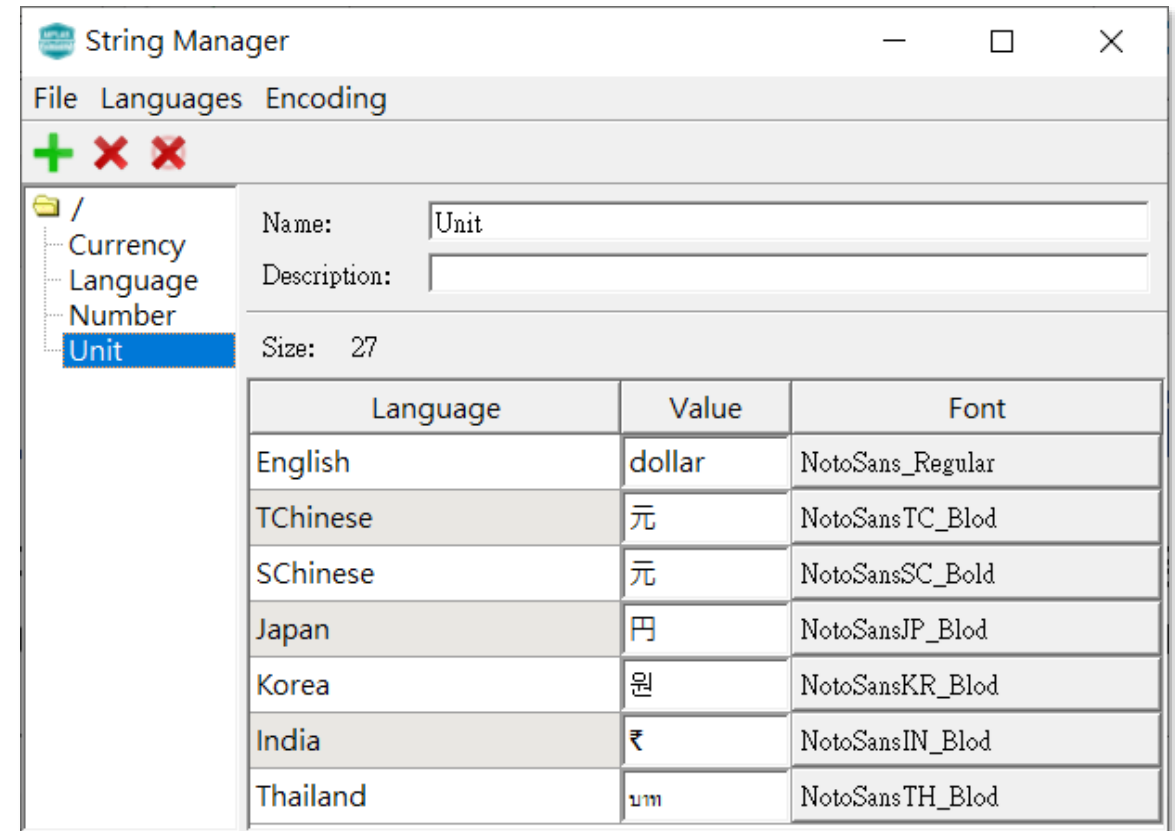
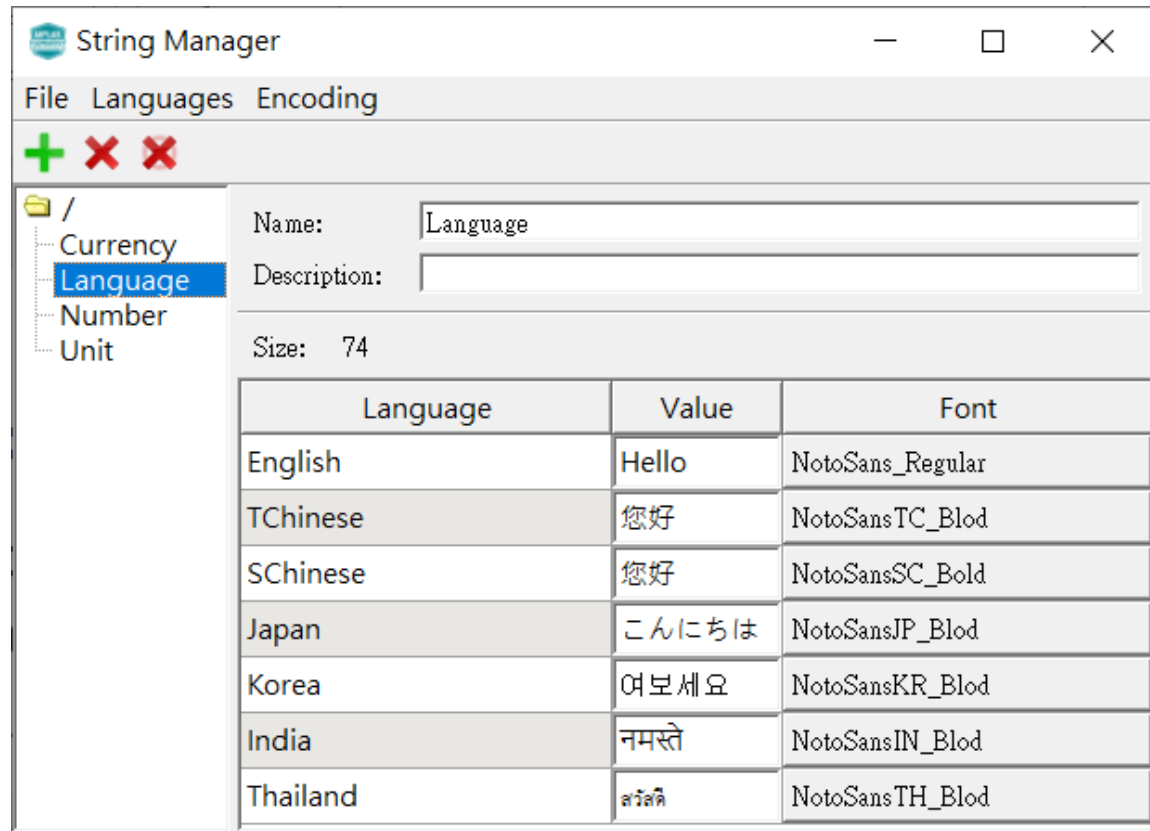
Anti-aliased Enable

String Assets Manager with Multi-Lingual Label

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

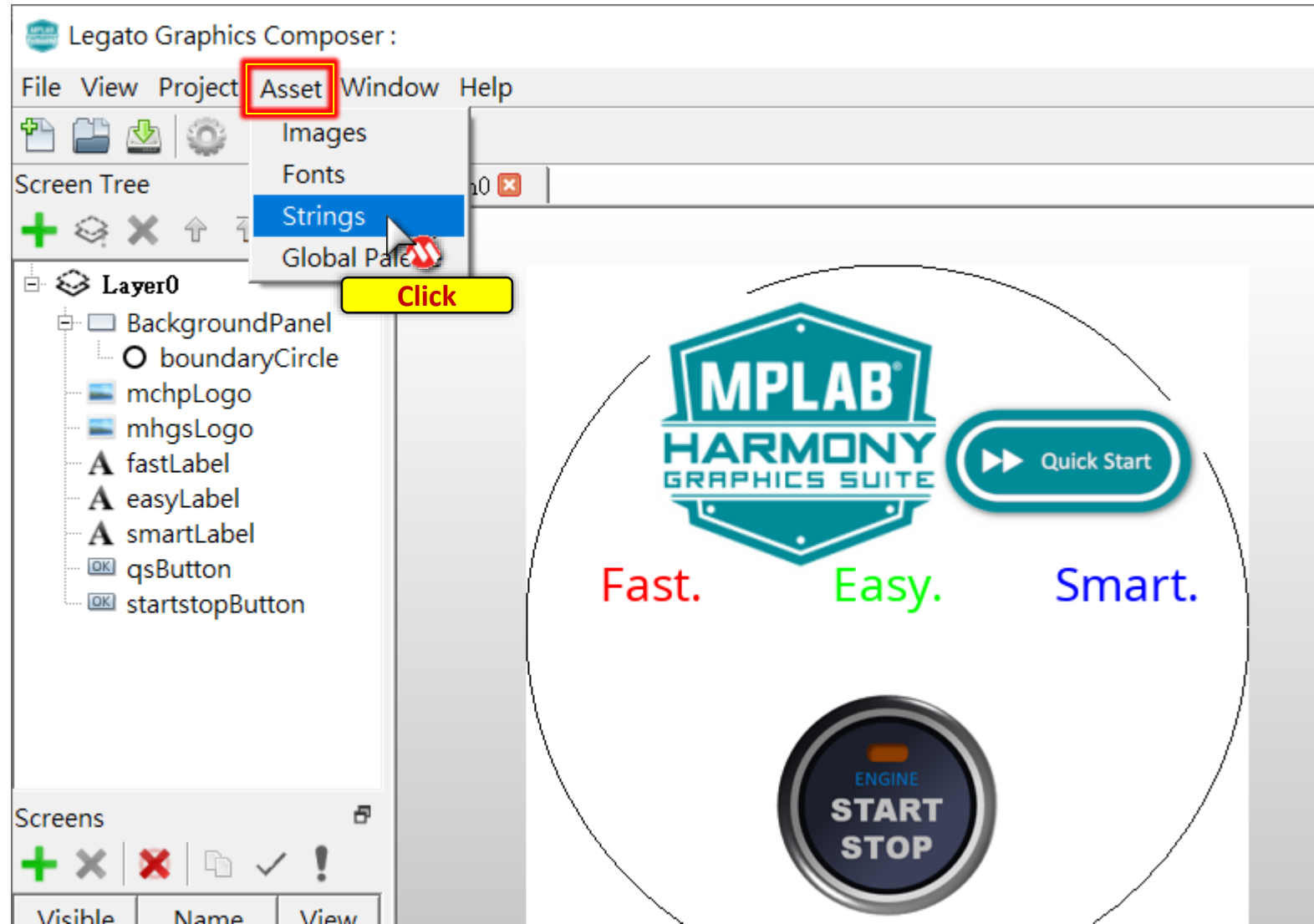
String Assets Manager with Multi-Lingual Label

- The string table stores associations of Strings, Language indices, and Font assets. Accepts characters in ASCII, UTF8 and UTF16 encoding.



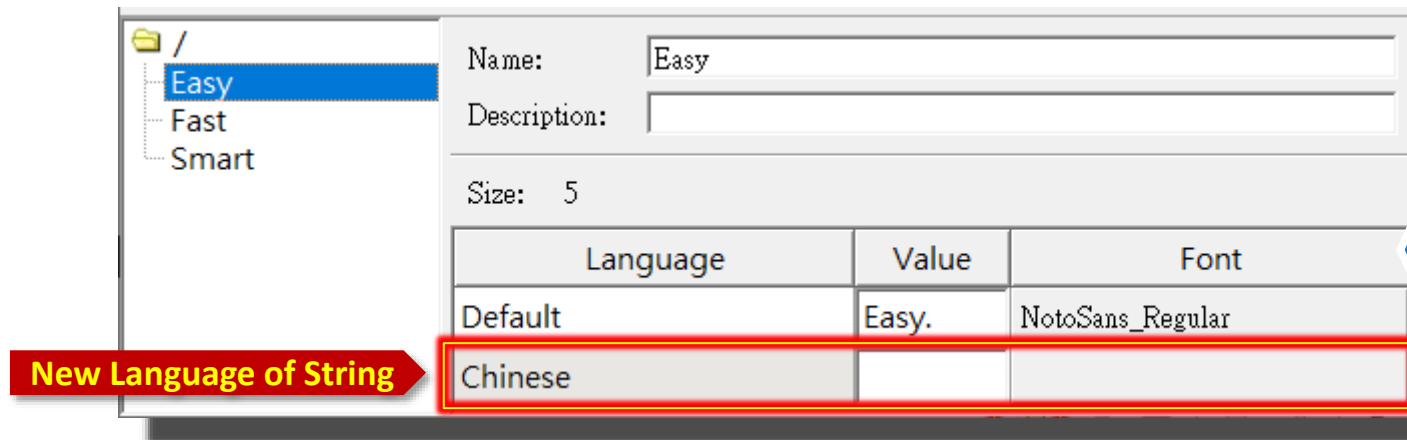
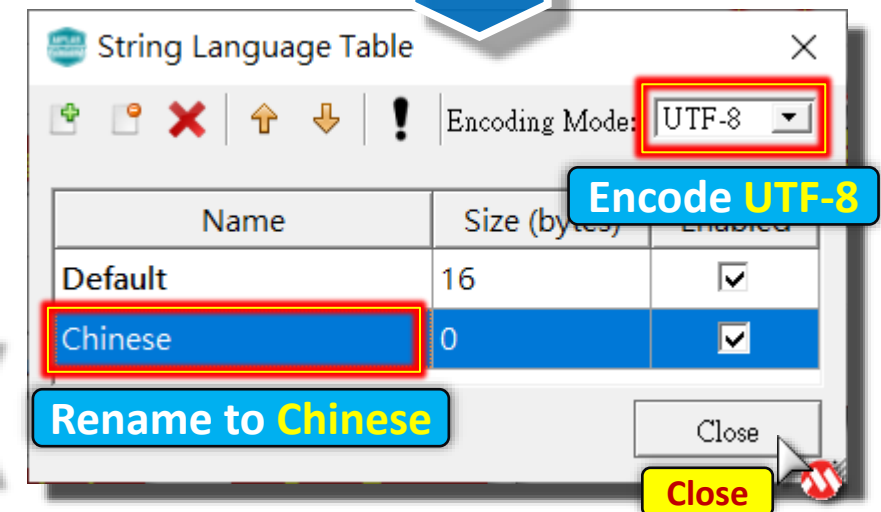
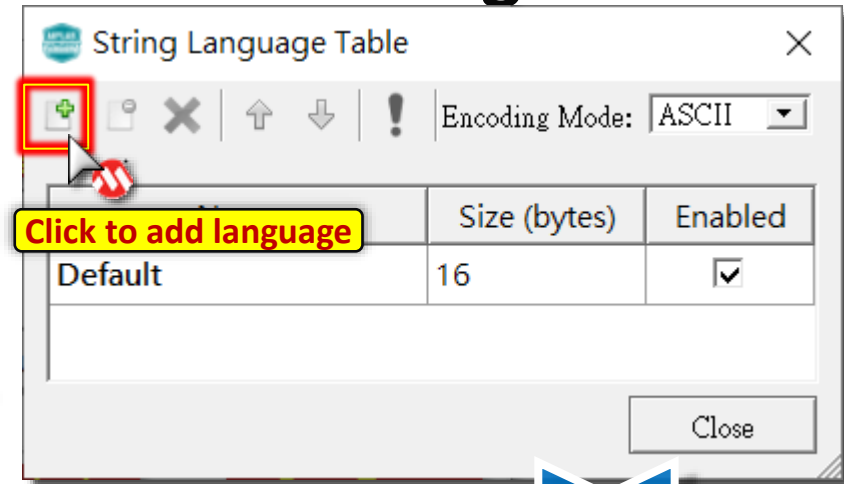
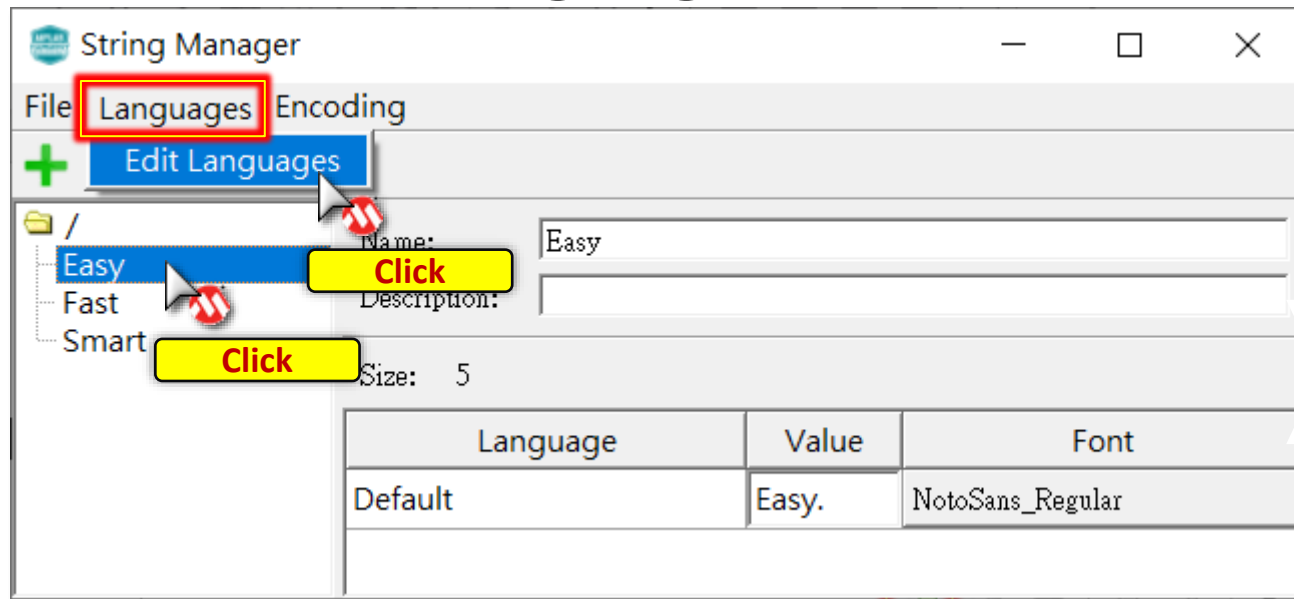
String Assets Manager with Multi-Lingual Label

- Modify Label Text through the Asset “String Manager”.



String Assets Manager with Multi-Lingual Label

- Add new Language “Chinese” with “UTF-8” Encoding Mode.



String Assets Manager with Multi-Lingual Label

- Edit Chinese String and given Chinese Font in String Manager.

The image displays three screenshots of the 'String Manager' application, illustrating the process of editing Chinese strings and fonts. The application has a menu bar with 'File', 'Languages', and 'Encoding'. Below the menu bar are three buttons: a green plus sign, a red 'x', and a red 'x'. The main area shows a tree view on the left with folders like '/' and 'Easy'. The right side has fields for 'Name', 'Description', and 'Size' (set to 11). At the bottom is a table with columns 'Language', 'Value', and 'Font'.

Top Left Screenshot: The 'Easy' folder is selected. The 'Name' field contains 'Easy'. The table shows 'Default' with 'Easy.' and 'NotoSans_Regular', and 'Chinese' with '簡易' and 'NotoSansTC_Blod'.

Top Right Screenshot: The 'Fast' folder is selected. The 'Name' field contains 'Fast'. The table shows 'Default' with 'Fast.' and 'NotoSans_Regular', and 'Chinese' with '迅捷' and 'NotoSansTC_Blod'.

Bottom Screenshot: The 'Smart' folder is selected. The 'Name' field contains 'Smart'. The table shows 'Default' with 'Smart.' and 'NotoSans_Regular', and 'Chinese' with '智慧' and 'NotoSansTC_Blod'.

A 'Select Font' dialog box is shown in the top left, listing 'NotoSans_Regular' and 'NotoSansTC_Blod'.

Value : 簡易 as Easy.
Value : 迅捷 as Fast.
Value : 智慧 as Smart.
Font : NotoSansTC_Bold

String Assets Manager with Multi-Lingual Label

- Preview multi-language Label string.



String Assets Manager with Multi-Lingual Label

- Implement Language Change in Quick Start Button Callback.

`\firmware\src\main.c`

```
...  
volatile bool StartStop = false;  
volatile uint32_t Language = language_Default;   
  
void event_Screen0_startstopButton_OnPressed(leButtonWidget* btn) { ... }  
void event_Screen0_startstopButton_OnReleased(leButtonWidget* btn) { ... }
```

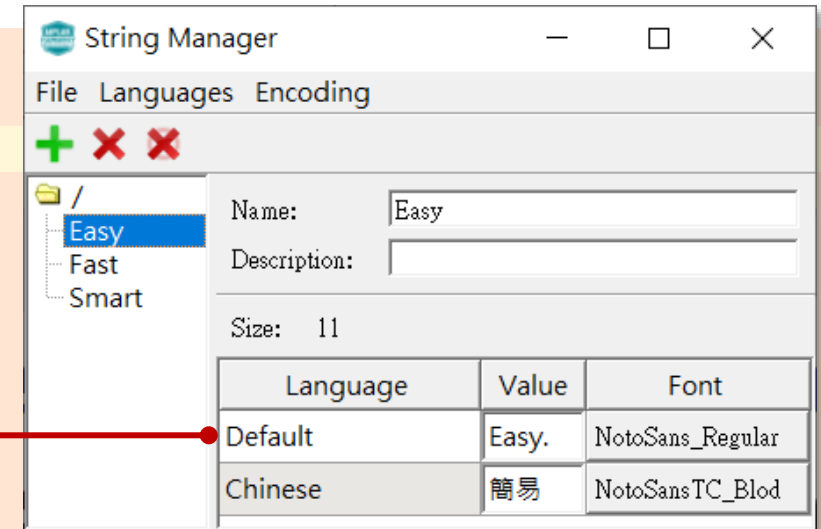
```
void event_Screen0_qsButton_OnPressed(leButtonWidget* btn)  
{  
    RGB_LED_B_On();  
}
```

```
void event_Screen0_qsButton_OnReleased(leButtonWidget* btn)   
{  
    RGB_LED_B_Off();
```

```
    Language = (Language+1) % leStringTable_GetLanguageCount( leGetStringTable() );  
    leSetStringLanguage( Language );
```

```
}
```

```
int main ( void ) { ... }
```



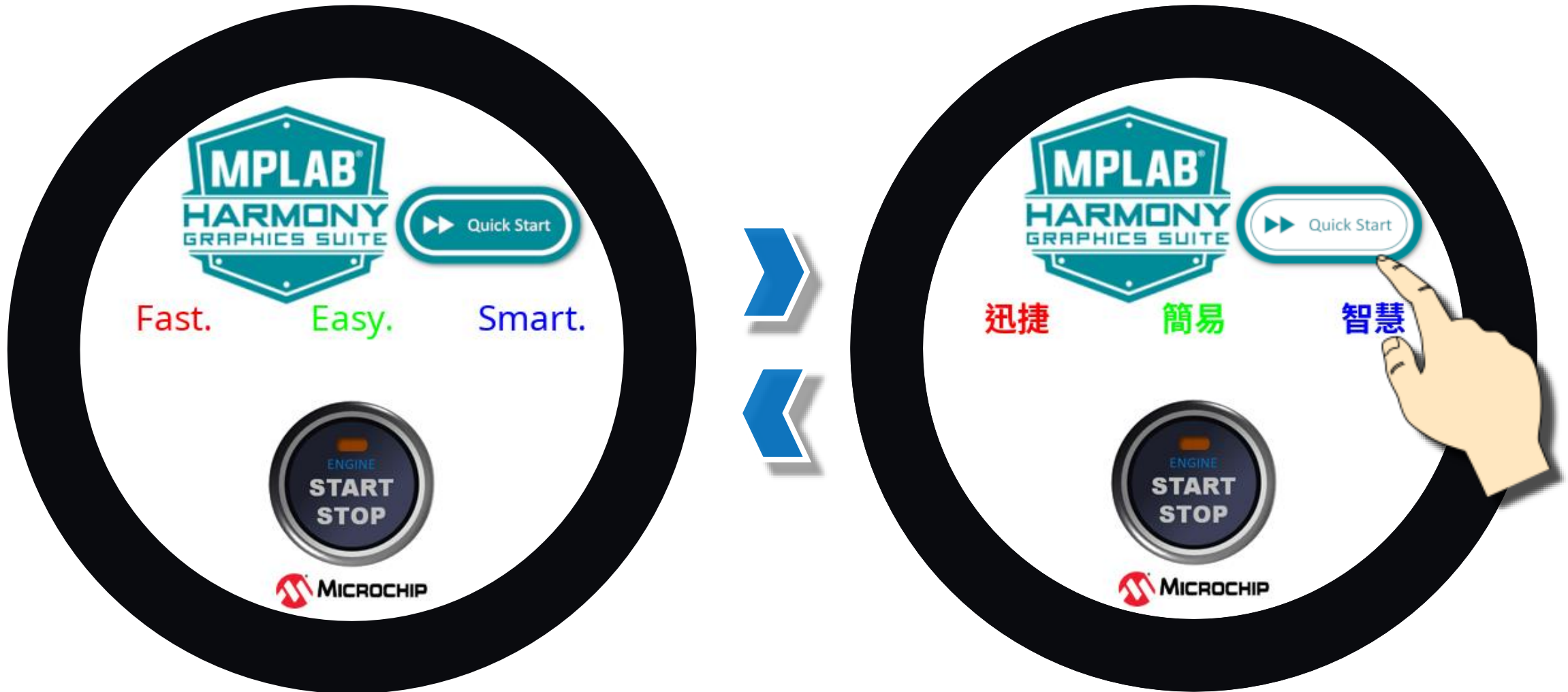
Initial Language ID to Language_Default

Callback function of Button released

Language Change looped

String Assets Manager with Multi-Lingual Label

- Press Quick Start button to toggle Language.

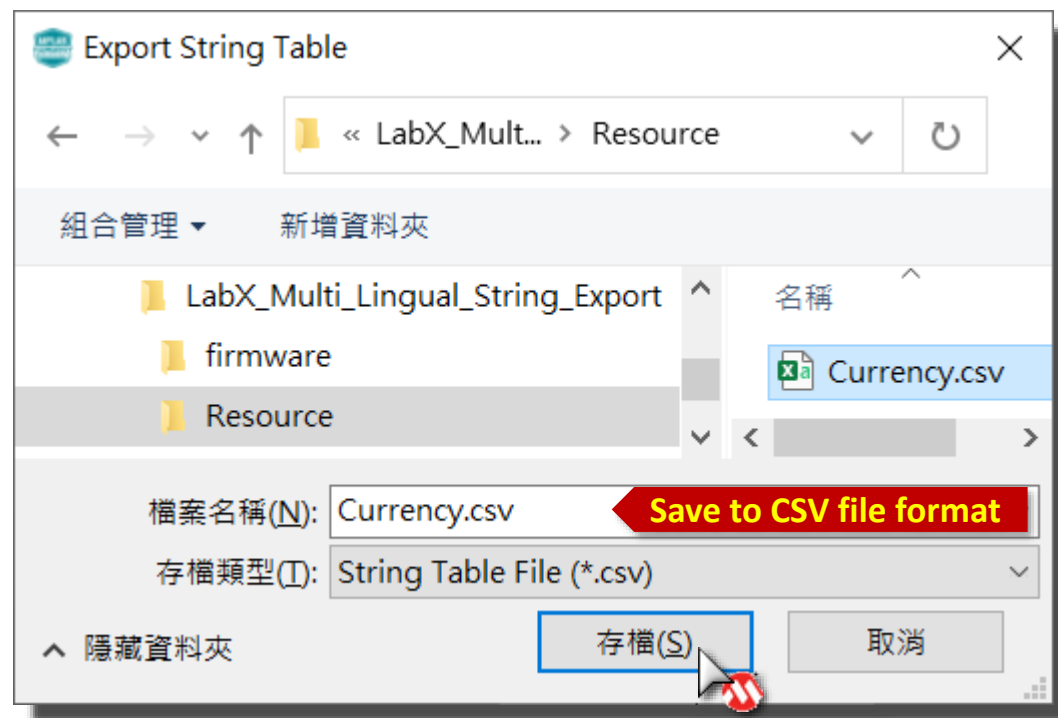
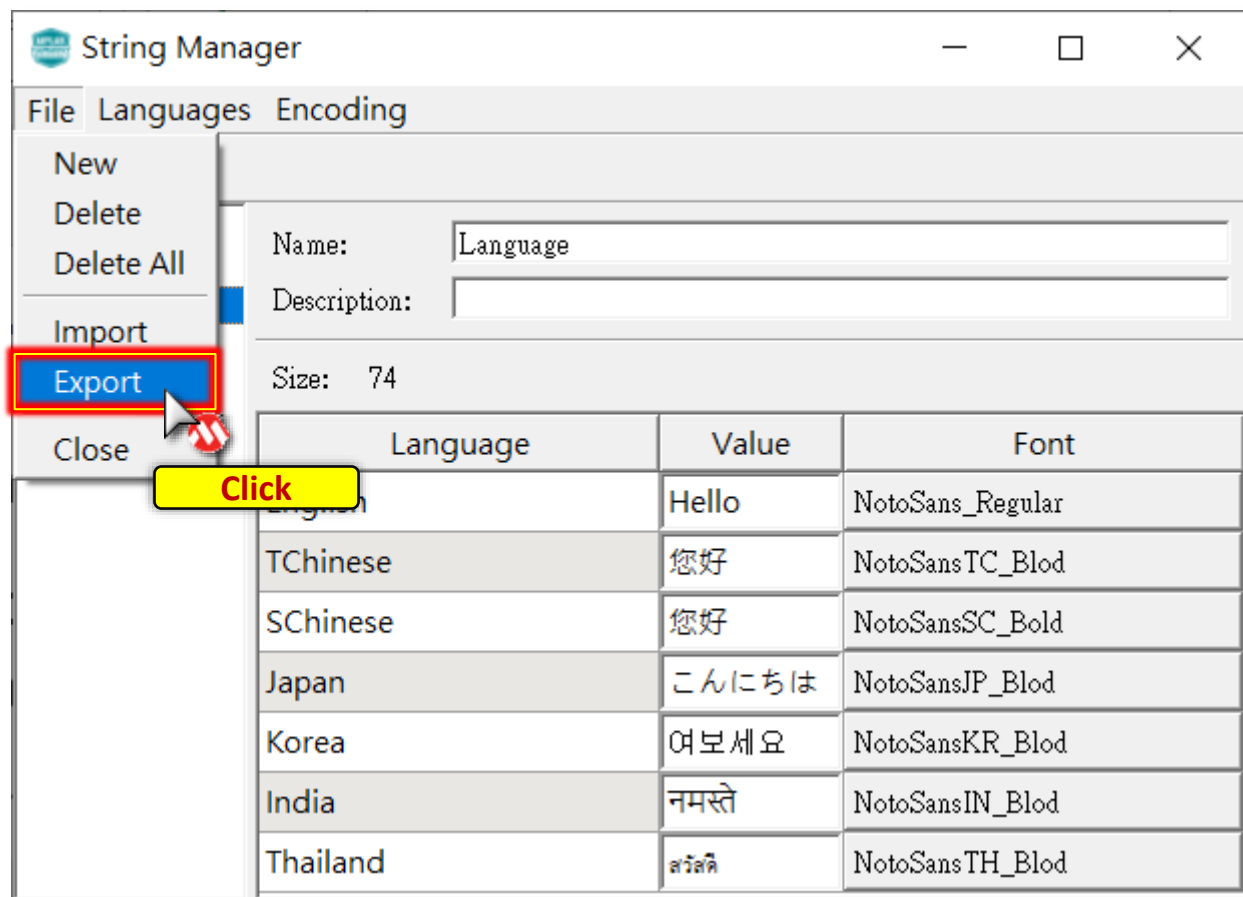


String Table Maintenance

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

String Table Maintenance

- Export String Table as “CSV” file format to easy edit.



String Table Maintenance

- Check “CSV” file in Notepad, the data fields separated in comma(,).

```
Currency.csv - 記事本
檔案(E) 編輯(E) 格式(O) 檢視(V) 說明
Languages (name),(enabled),,,
English,true,,,
TChinese,true,,,
SChinese,true,,,
Japan,true,,,
Korea,true,,,
India,true,,,
Thailand,true,,,
,,,,
Default Language,English,,,
,,,,
Strings (name),(description),(folder path),,
Unit,,/,
Number,,/,
Language,,/,
Currency,,/,
,,,,
String Entries (name),(language),(value),(font name),
Unit,English,dollar,NotoSans-Regular,
Unit,TChinese,元,NotoSansTC-Bold,
Unit,SChinese,元,NotoSansSC-Bold,
Unit,Japan,円,NotoSansJP-Bold,
Unit,Korea,원,NotoSansKR-Bold,
Unit,India,₹,NotoSansIN-Bold,
Unit,Thailand,บาท,NotoSansTH-Bold,
```

```
Number,English,1,NotoSans-Regular,
Number,TChinese,30.76,NotoSans-Regular,
Number,SChinese,6.98,NotoSans-Regular,
Number,Japan,136.2,NotoSans-Regular,
Number,Korea,"1,301.7",NotoSans-Regular,
Number,India,82.8,NotoSans-Regular,
Number,Thailand,34.81,NotoSans-Regular,
Language,English,Hello,NotoSans-Regular,
Language,TChinese,您好,NotoSansTC-Bold,
Language,SChinese,您好,NotoSansSC-Bold,
Language,Japan,こんにちは,NotoSansJP-Bold,
Language,Korea,여보세요,NotoSansKR-Bold,
Language,India,नमस्ते,NotoSansIN-Bold,
Language,Thailand,สวัสดี,NotoSansTH-Bold,
Currency,English,USD,NotoSans-Regular,
Currency,TChinese,新台幣,NotoSansTC-Bold,
Currency,SChinese,人民币,NotoSansSC-Bold,
Currency,Japan,にほんえん,NotoSansJP-Bold,
Currency,Korea,한국 원,NotoSansKR-Bold,
Currency,India,भारतीय मुद्रा,NotoSansIN-Bold,
Currency,Thailand,เงินไทย,NotoSansTH-Bold,
```

< 第 1 列 · 第 1 行 100% Unix (LF) UTF-8 >

String Table Maintenance

- Import CSV file using MS-EXCEL.

The screenshot shows the Microsoft Excel interface with the 'Data' tab selected. The 'Import Data' dialog box is open, displaying the file 'Currency.csv' in the 'Resource' folder. The file is highlighted with a red box, and a yellow 'Click' label points to it. The 'Import' button at the bottom right of the dialog is also highlighted with a red box and a yellow 'Import' label.

Excel Ribbon: 自動儲存 (關閉), 檔案, 常用, 插入, 頁面配置, 公式, 資料, 校閱, 檢視, Automate

Import Data Dialog Box:

- Path: << LabX_Multi_Lingual_String_Export > Resource
- File Name: Currency.csv
- File Type: 文字檔案 (*.prn;*.txt;*.csv)
- Buttons: 匯入(M), 取消

File Content (from Currency.csv):

名稱	修改日期	內容
LabX_Multi_Lingual_String_Export		Languages (name),(enabled),,,
firmware		English,true,,,
Resource		TChinese,true,,,
		Chinese,true,,,
		Japan,true,,,
		Korea,true,,,
		Tadiao,true,,,

String Table Maintenance

- Encoding is Unicode (UTF-8) and separated by comma(,).

Currency.csv

檔案原點 **UTF8 encoding** 分隔符號 **Separated by comma** 資料類型偵測

65001: Unicode (UTF-8) 逗號 依據前 200 個列

Column1	Column2	Column3	Column4	Column5
Languages (name)	(enabled)			
English	true			
TChinese	true			
SChinese	true			
Japan	true			
Korea	true			
India	true			
Thailand	true			

載入 轉換資料 取消

Import

String Table Maintenance

- You could save it as Excel format and deliver your translation agency.

自動儲存 ●關閉 活頁簿1 - Excel

檔案 常用 插入 頁面配置 公式 資料 校閱 檢視 Automate

表格名稱: Currency

調整表格大小

內容

工具

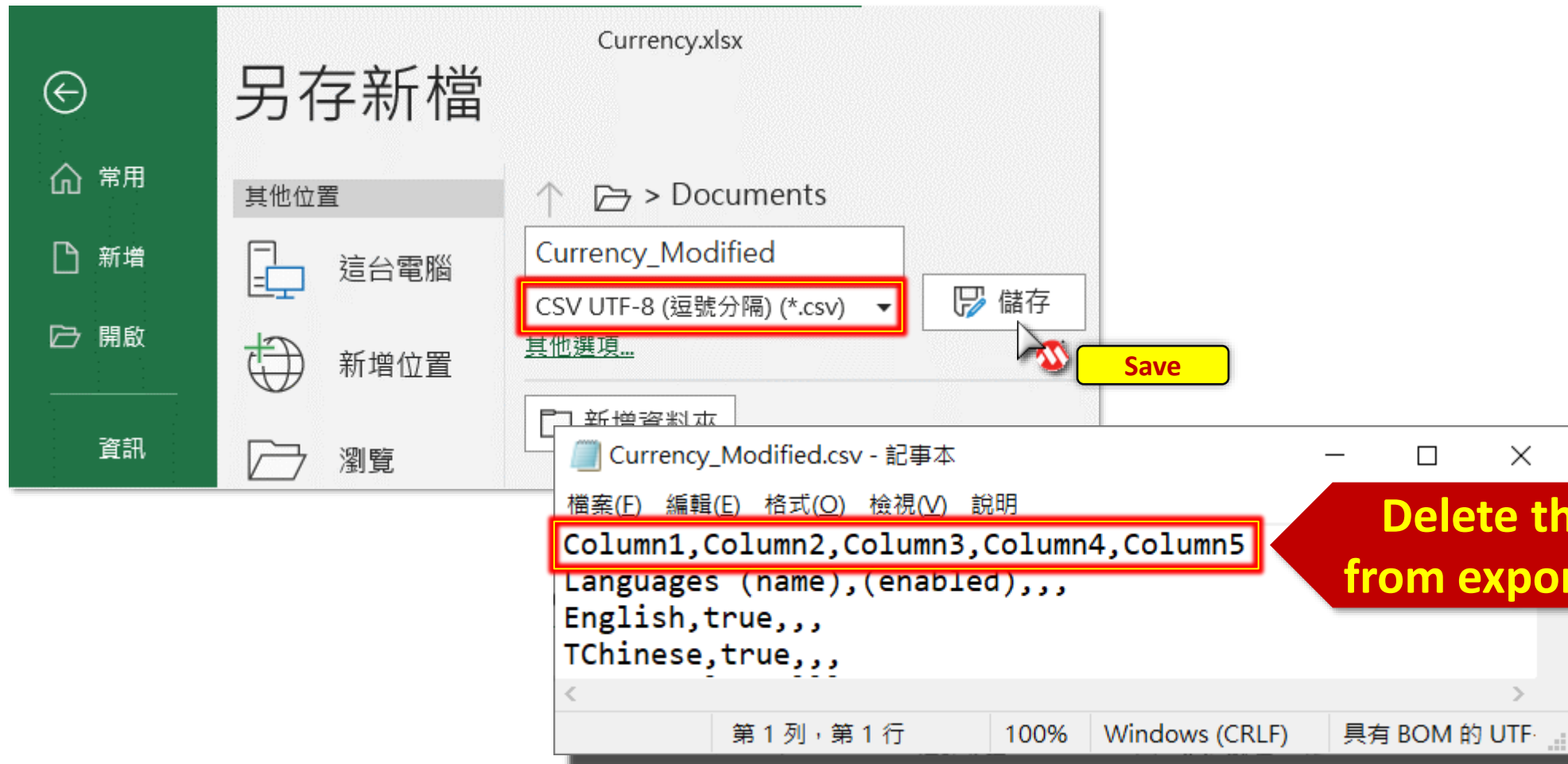
外部表格資料

	Column1	Column2	Column3	Column4	Column5
1	Column1	Column2	Column3	Column4	Column5
2	Languages (name)	(enabled)			
3	English	true			
4	TChinese	true			
5	SChinese	true			
6	Japan	true			
7	Korea	true			
8	India	true			
9	Thailand	true			
10					
11	Default Language	English			
12					
13	Strings (name)	(description)	(folder path)		
14	Unit		/		
15	Number		/		
16	Language		/		
17	Currency		/		

	Column1	Column2	Column3	Column4	Column5
19	String Entries (name)	(language)	(value)	(font name)	
20	Unit	English	dollar	NotoSans Regular	
21	Unit	TChinese	元	NotoSansTC Blod	
22	Unit	SChinese	元	NotoSansSC Bold	
23	Unit	Japan	円	NotoSansJP Blod	
24	Unit	Korea	원	NotoSansKR Blod	
25	Unit	India	₹	NotoSansIN Blod	
26	Unit	Thailand	บาท	NotoSansTH Blod	
27	Number	English	1	NotoSans Regular	
28	Number	TChinese	30.76	NotoSans Regular	
29	Number	SChinese	6.98	NotoSans Regular	
30	Number	Japan	136.2	NotoSans Regular	
31	Number	Korea	1,301.7	NotoSans Regular	
32	Number	India	82.8	NotoSans Regular	
33	Number	Thailand	34.81	NotoSans Regular	
34	Language	English	Hello	NotoSans Regular	
35	Language	TChinese	您好	NotoSansTC Blod	
36	Language	SChinese	您好	NotoSansSC Bold	
37	Language	Japan	こんにちは	NotoSansJP Blod	
38	Language	Korea	여보세요	NotoSansKR Blod	
39	Language	India	नमस्ते	NotoSansIN Blod	
40	Language	Thailand	สวัสดี	NotoSansTH Blod	
41	Currency	English	USD	NotoSans Regular	
42	Currency	TChinese	新台幣	NotoSansTC Blod	
43	Currency	SChinese	人民幣	NotoSansSC Bold	
44	Currency	Japan	にほんえん	NotoSansJP Blod	
45	Currency	Korea	한국 원	NotoSansKR Blod	
46	Currency	India	भारतीय मुद्रा	NotoSansIN Blod	
47	Currency	Thailand	เงินไทย	NotoSansTH Blod	

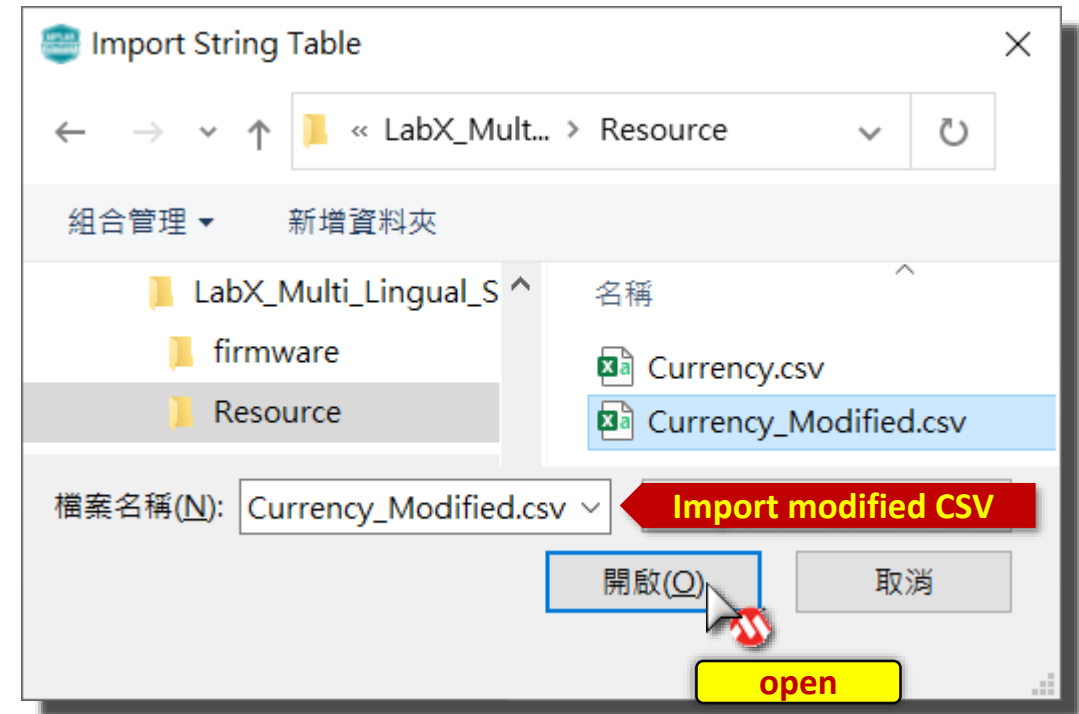
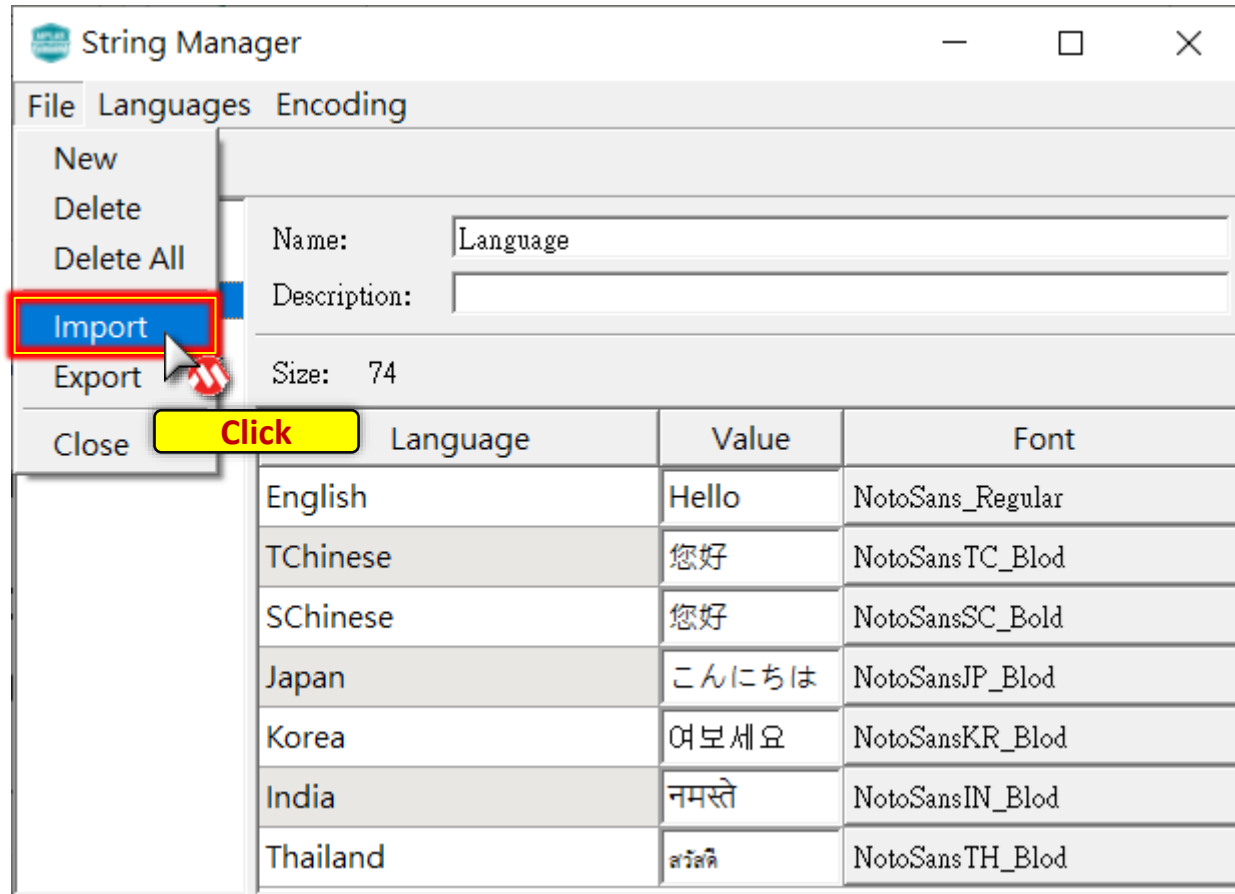
String Table Maintenance

- You could export modified Excel file to CSV format file by “Save As”.
- **Please remember to delete first row of CSV file.**



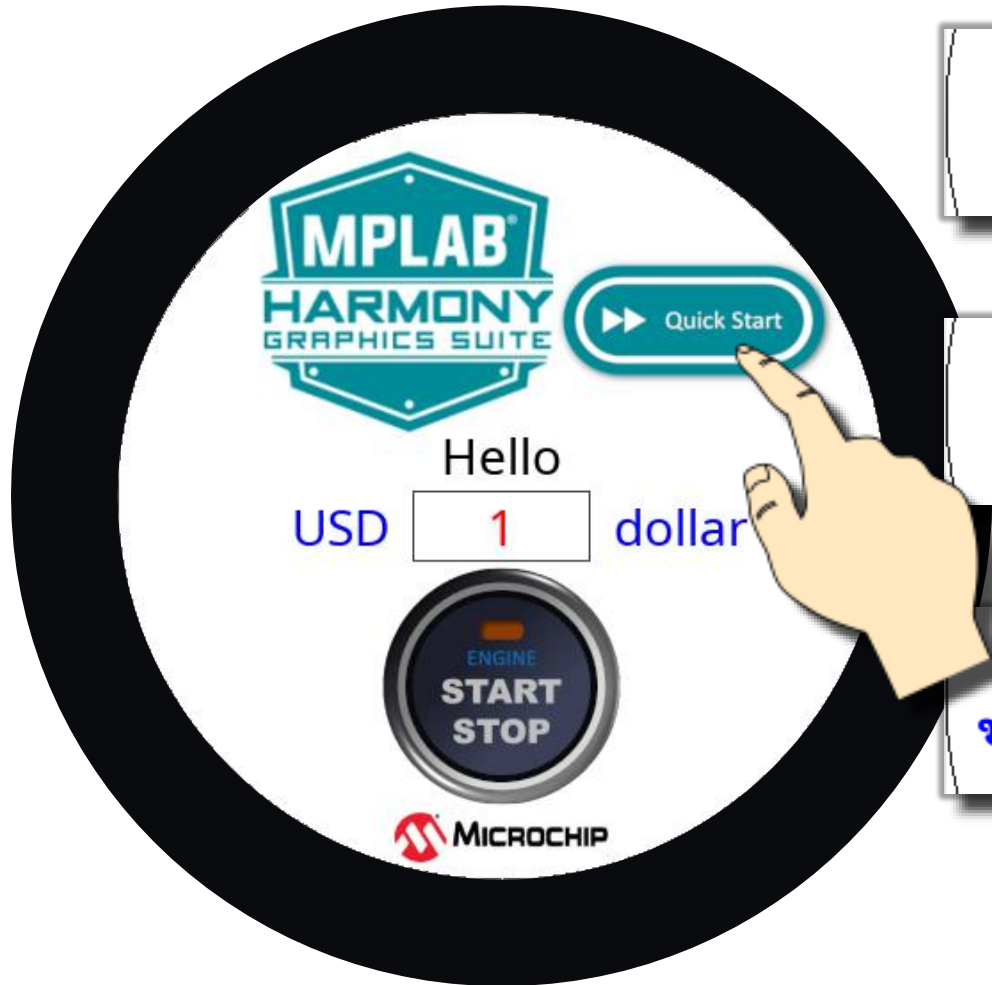
String Table Maintenance

- Import the String Table (Modified CSV file) in MHGS.



String Table Maintenance

- Currency convert example with multi-lingual strings.



您好
新台幣 30.76 元

您好
人民币 6.98 元

こんにちは
にほんえん 136.2 円

여보세요
한국 원 1,301.7 원

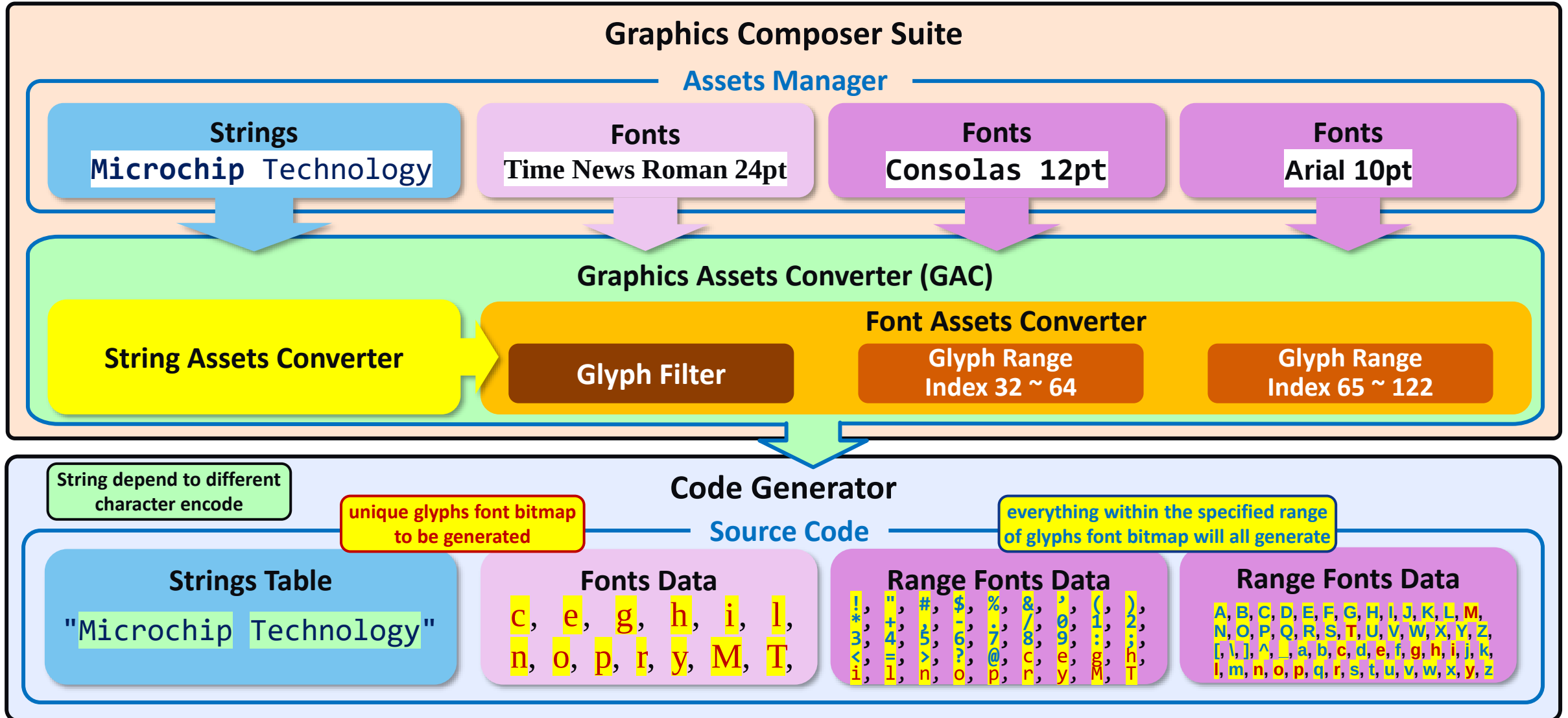
नमस्ते
भारतीय मुद्रा 82.8 ₹

สวัสดี
เงินไทย 34.81 บาท

Label Widget with Dynamic Text

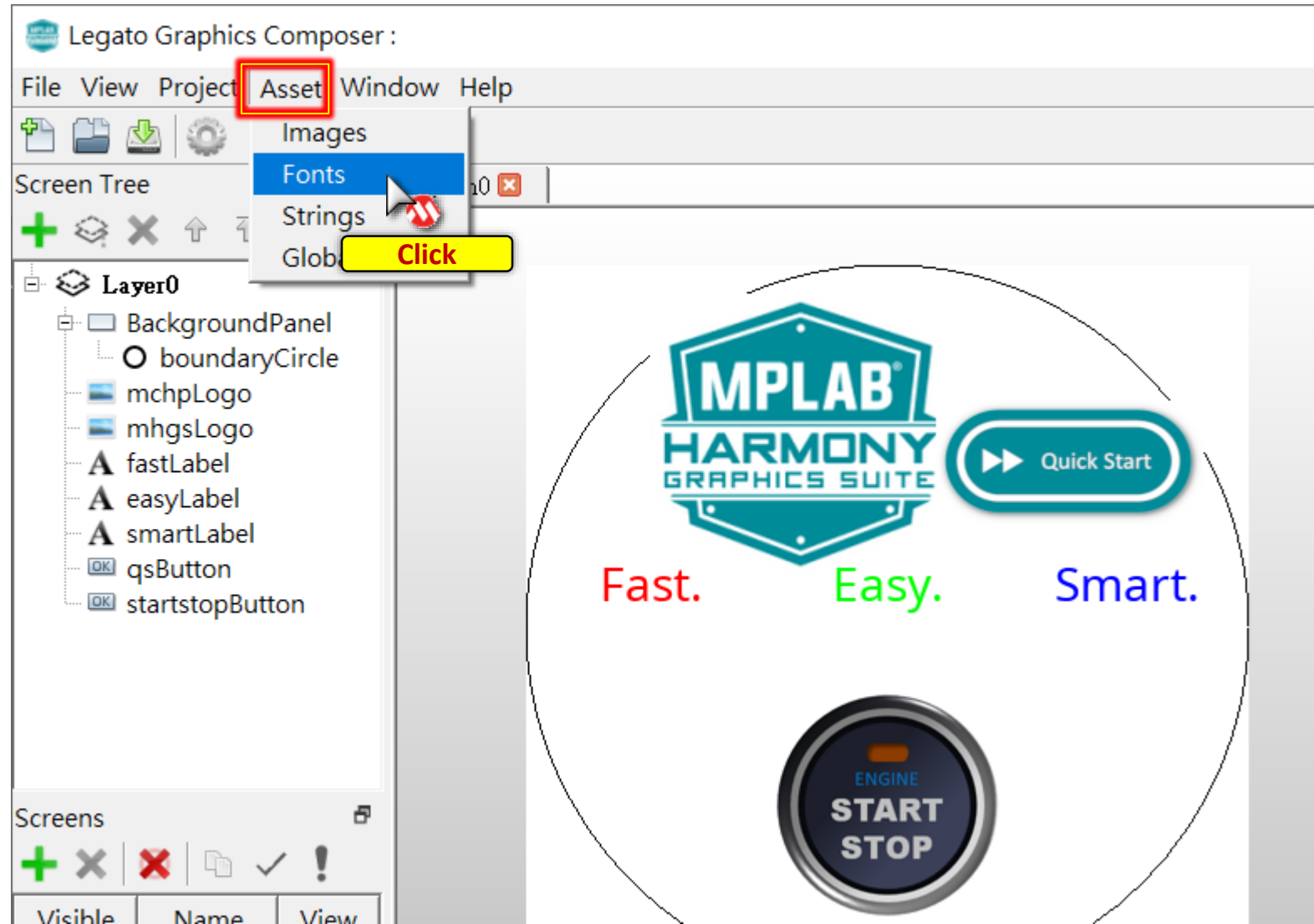
PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Label Widget with Dynamic Text



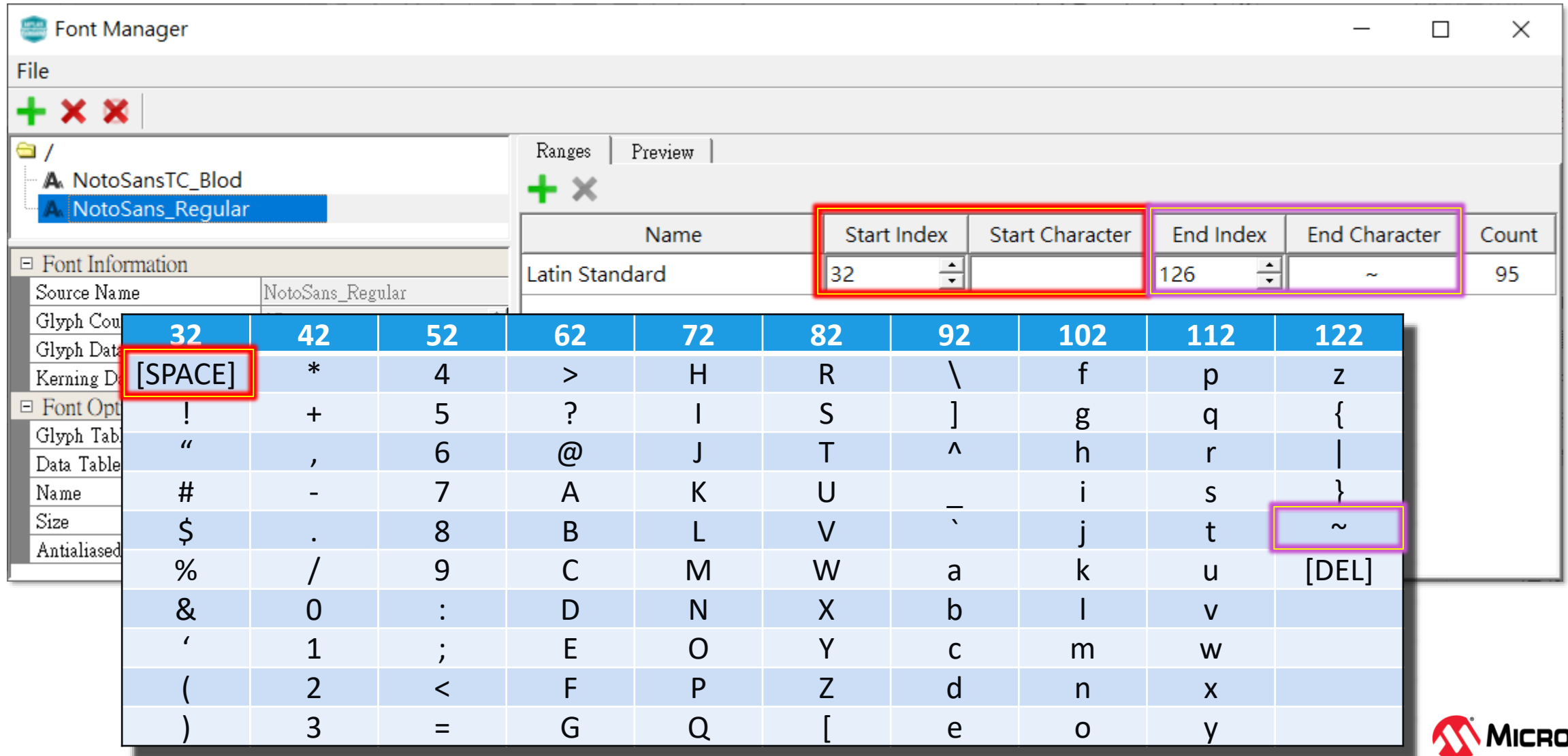
Label Widget with Dynamic Text

- Open Font Asset Manage.



Label Widget with Dynamic Text

- Check the Glyph range index from ASCII code index **32** to **126**.



Font Manager

File

+ × ×

/

A NotoSansTC_Blod

A NotoSans-Regular

Font Information

Source Name: NotoSans-Regular

Glyph Count: 95

Glyph Data Table

Kerning Data Table

Font Options

Glyph Table

Data Table

Name

Size

Antialiased

Ranges

Preview

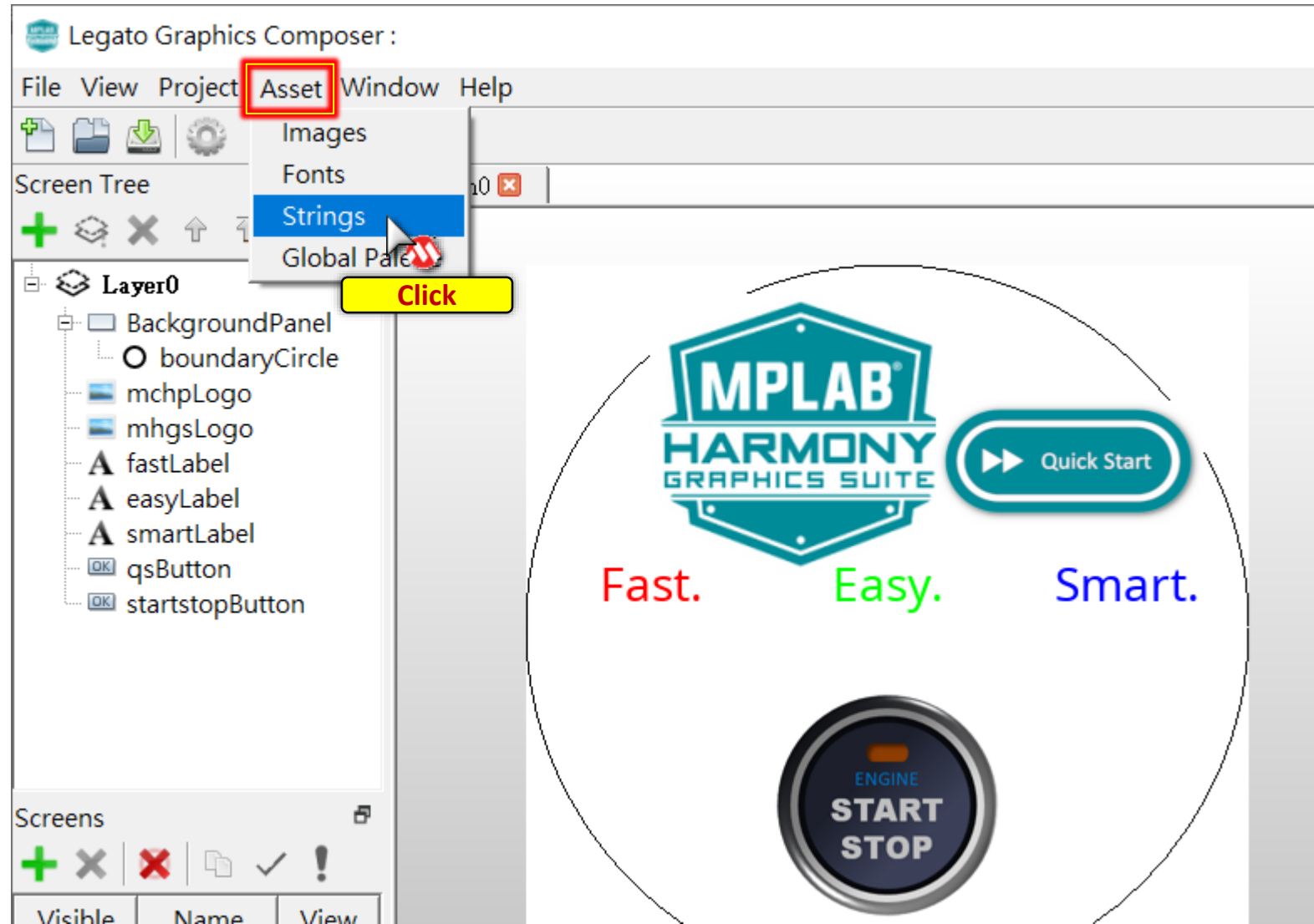
+ ×

Name	Start Index	Start Character	End Index	End Character	Count
Latin Standard	32		126	~	95

32	42	52	62	72	82	92	102	112	122
[SPACE]	*	4	>	H	R	\	f	p	z
!	+	5	?	I	S	]	g	q	{
"	,	6	@	J	T	^	h	r	
#	-	7	A	K	U	_	i	s	}
\$	.	8	B	L	V	`	j	t	~
%	/	9	C	M	W	a	k	u	[DEL]
&	0	:	D	N	X	b	l	v	
'	1	;	E	O	Y	c	m	w	
(	2	<	F	P	Z	d	n	x	
)	3	=	G	Q	[	e	o	y	

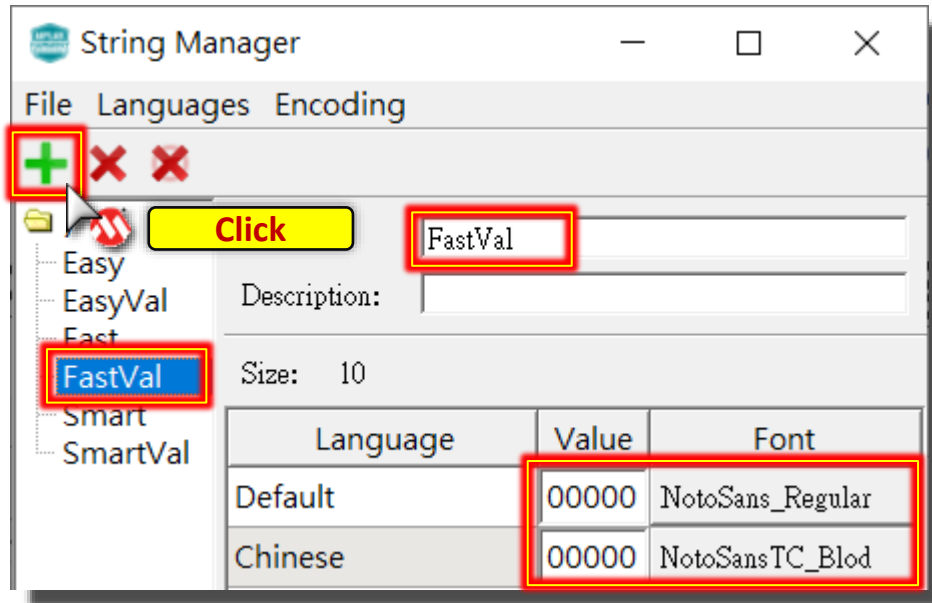
Label Widget with Dynamic Text

- Add new Label Text through the Asset “String Manager”.

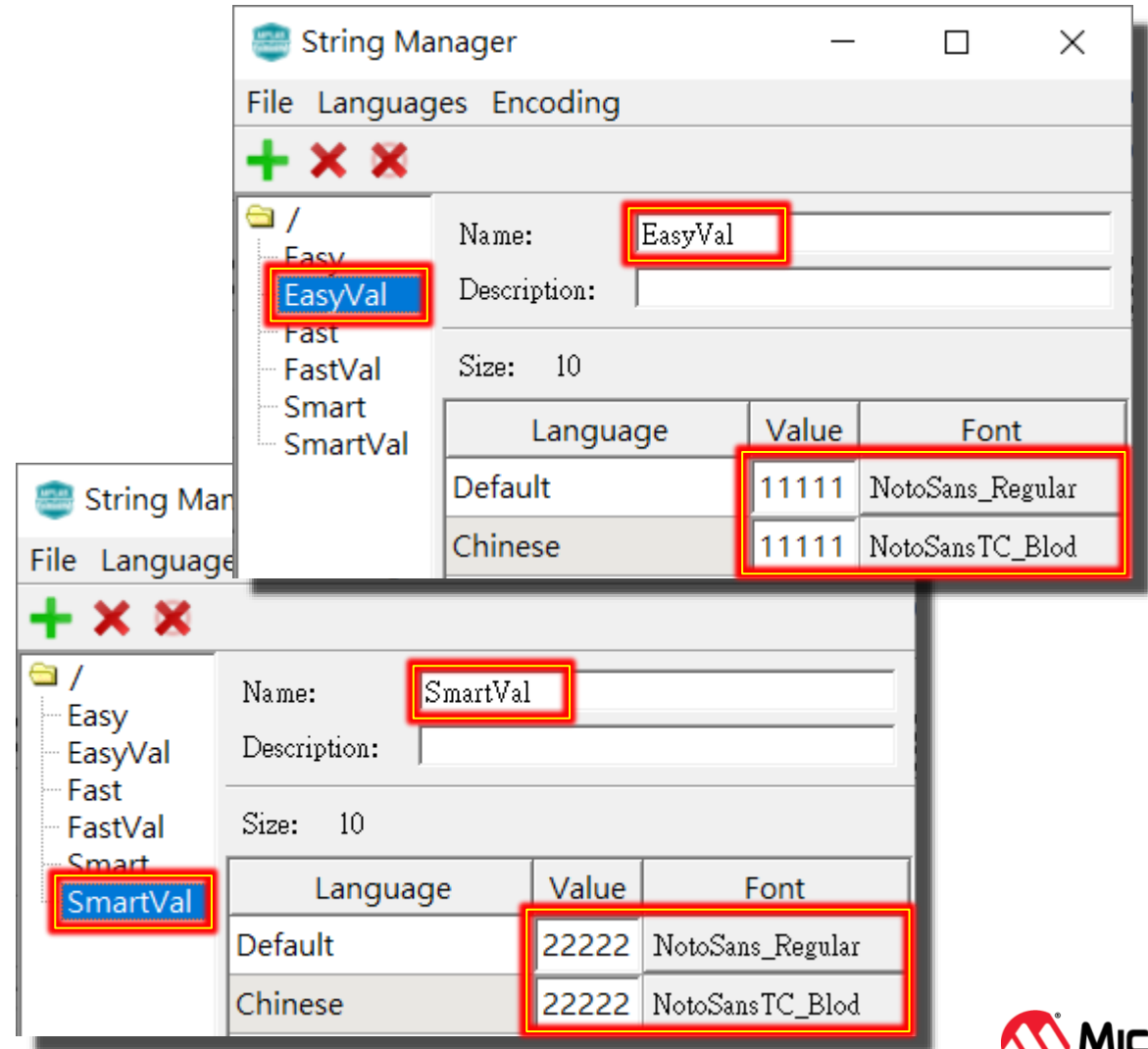


Label Widget with Dynamic Text

- Click icon  to add three new strings and configure as below.

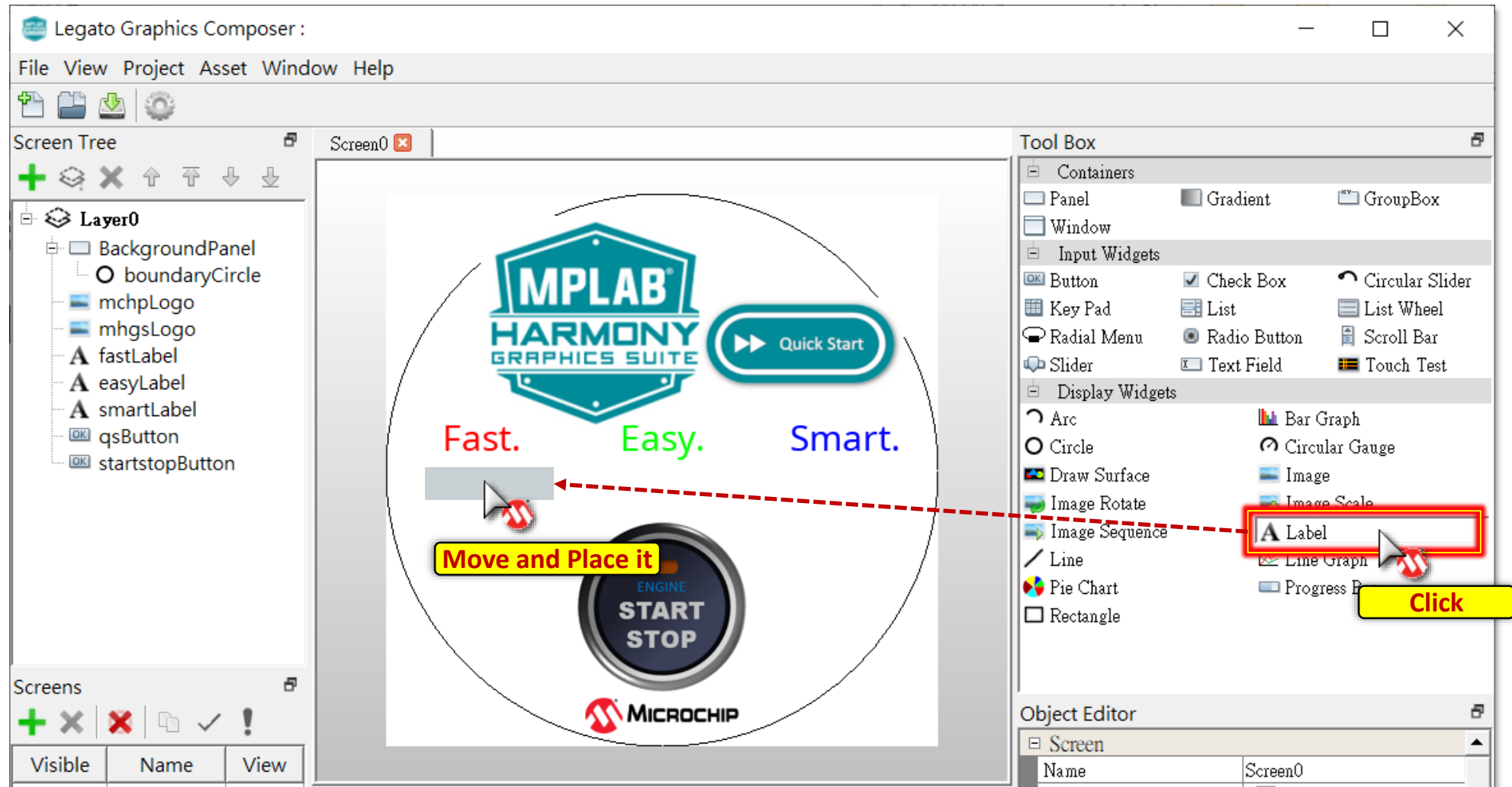


String Name	Value	Font
FastVal	00000	NotoSans_Regular
	00000	NotoSansTC_Blod
EasyVal	11111	NotoSans_Regular
	11111	NotoSansTC_Blod
SmartVal	22222	NotoSans_Regular
	22222	NotoSansTC_Blod



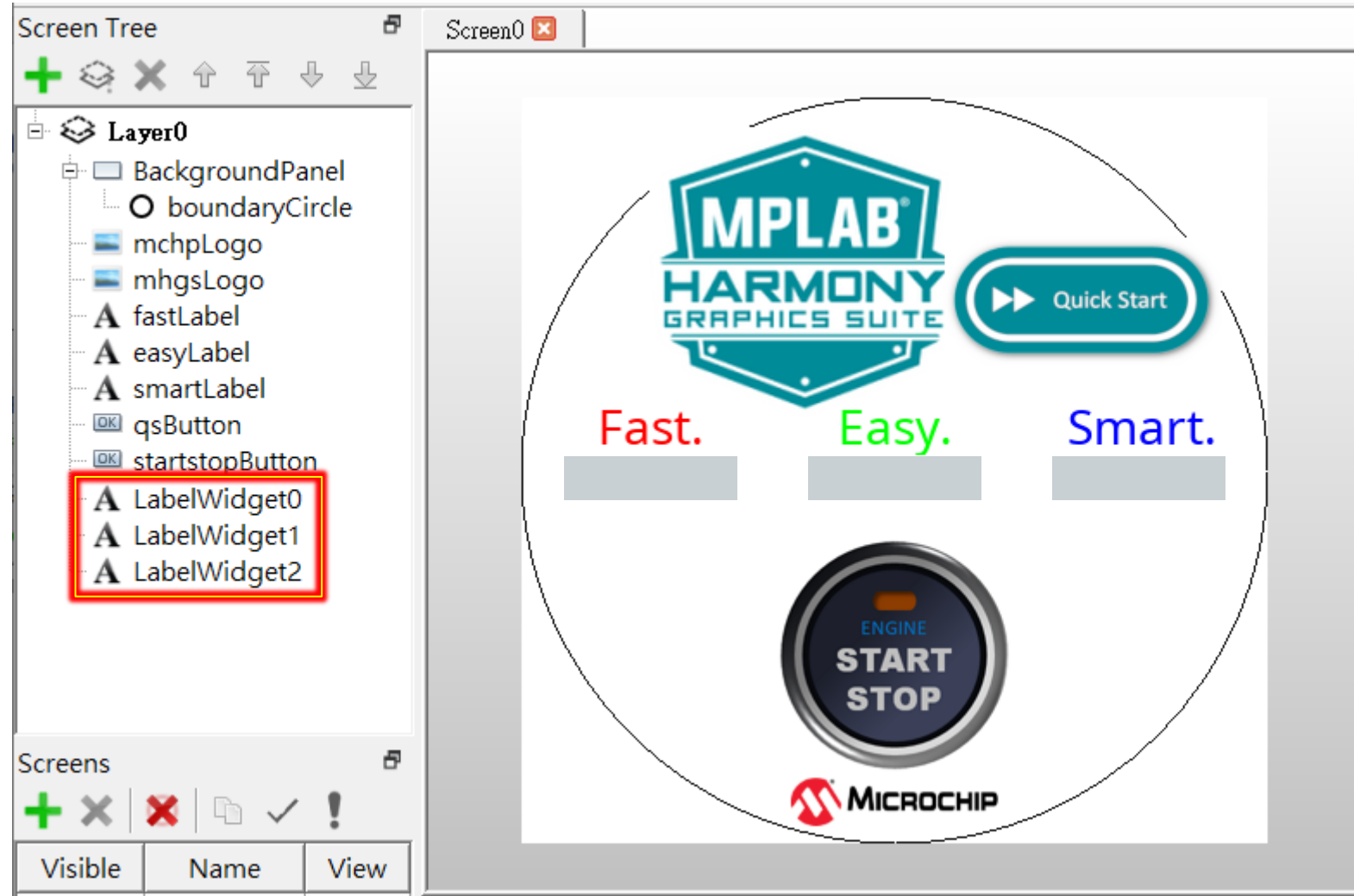
Label Widget with Dynamic Text

- Click [Label] of Display Widgets window to add 3 Label widgets.



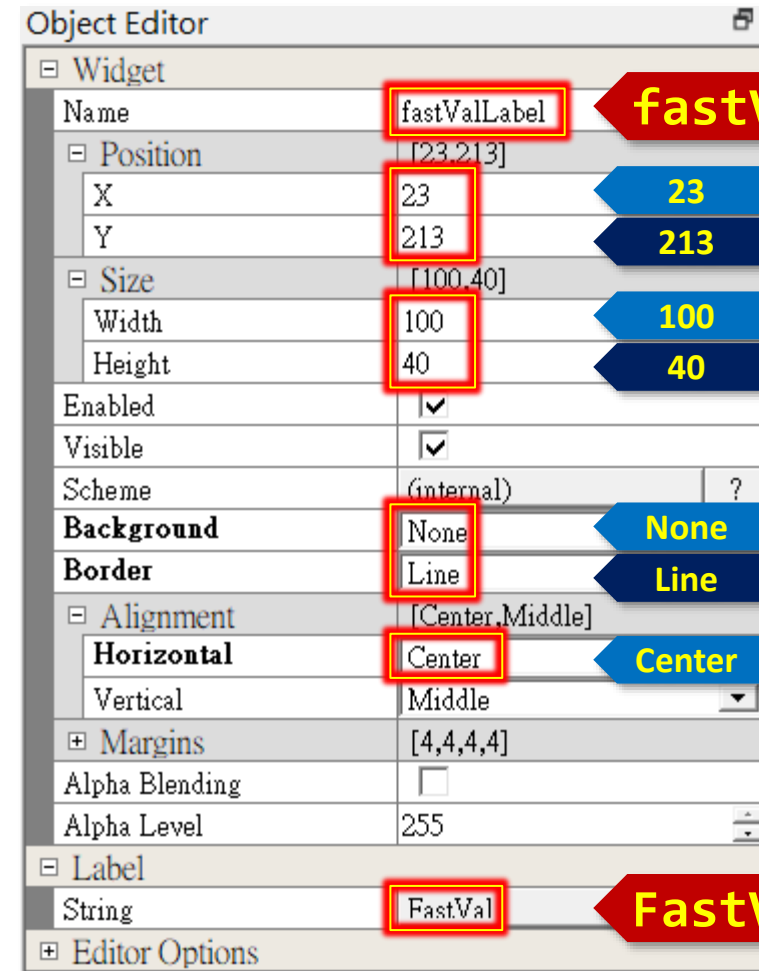
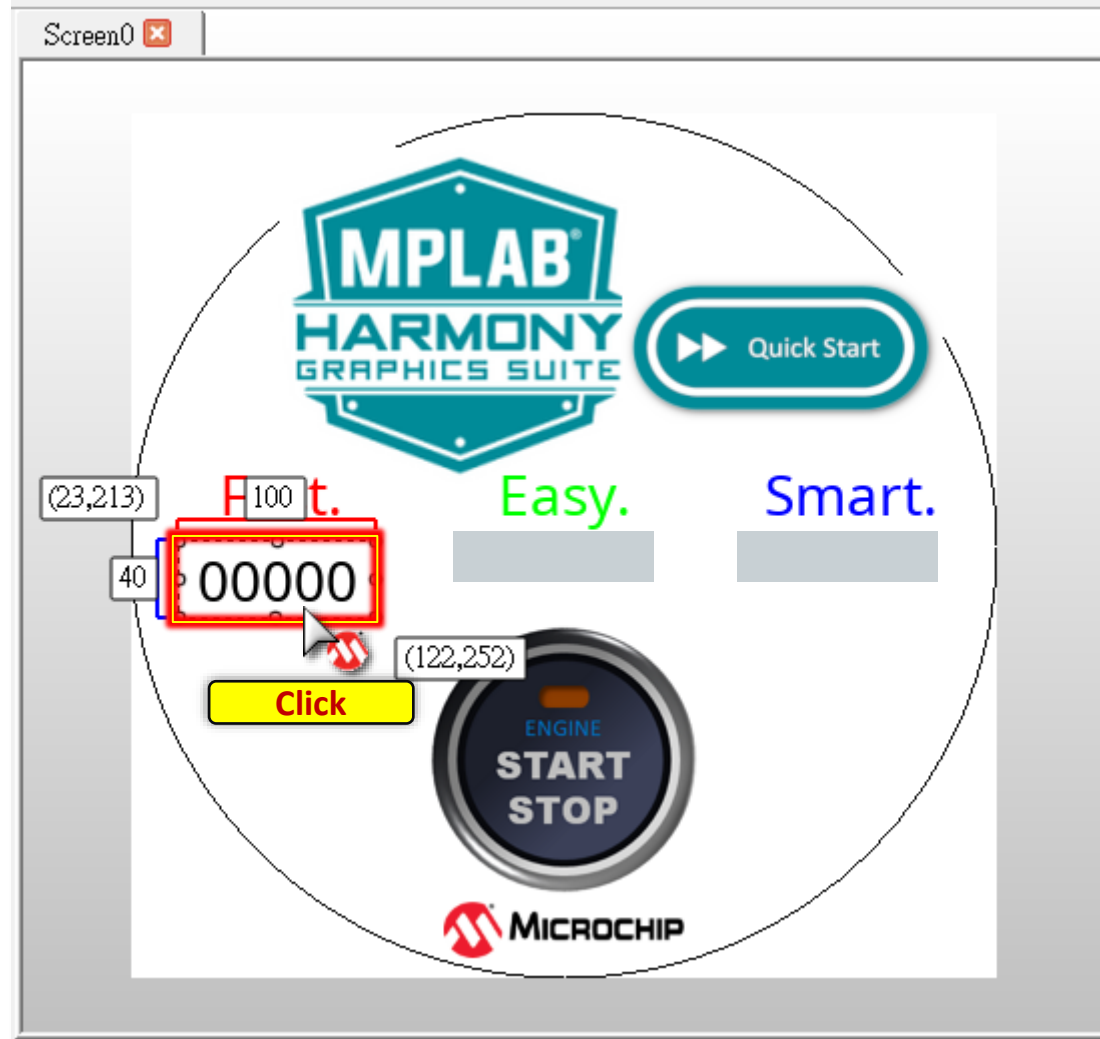
Label Widget with Dynamic Text

- After added three new Label widgets.



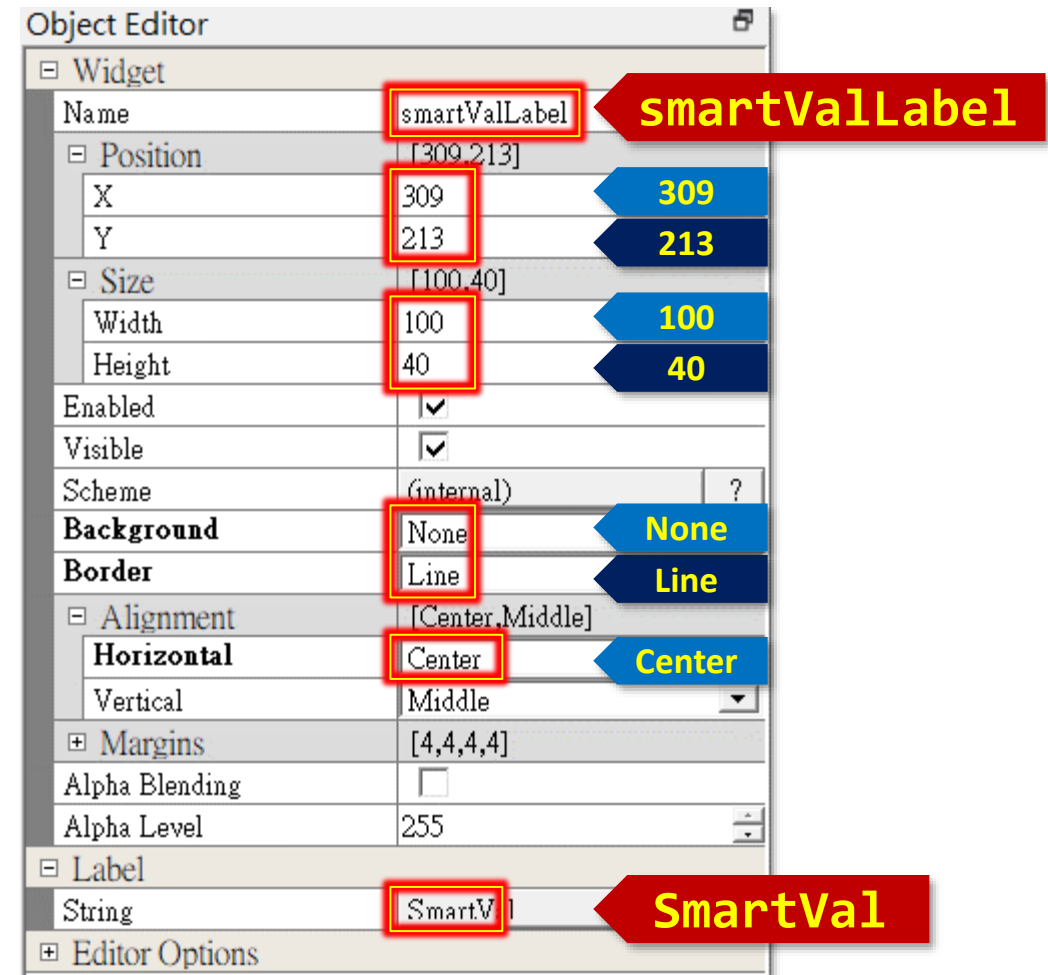
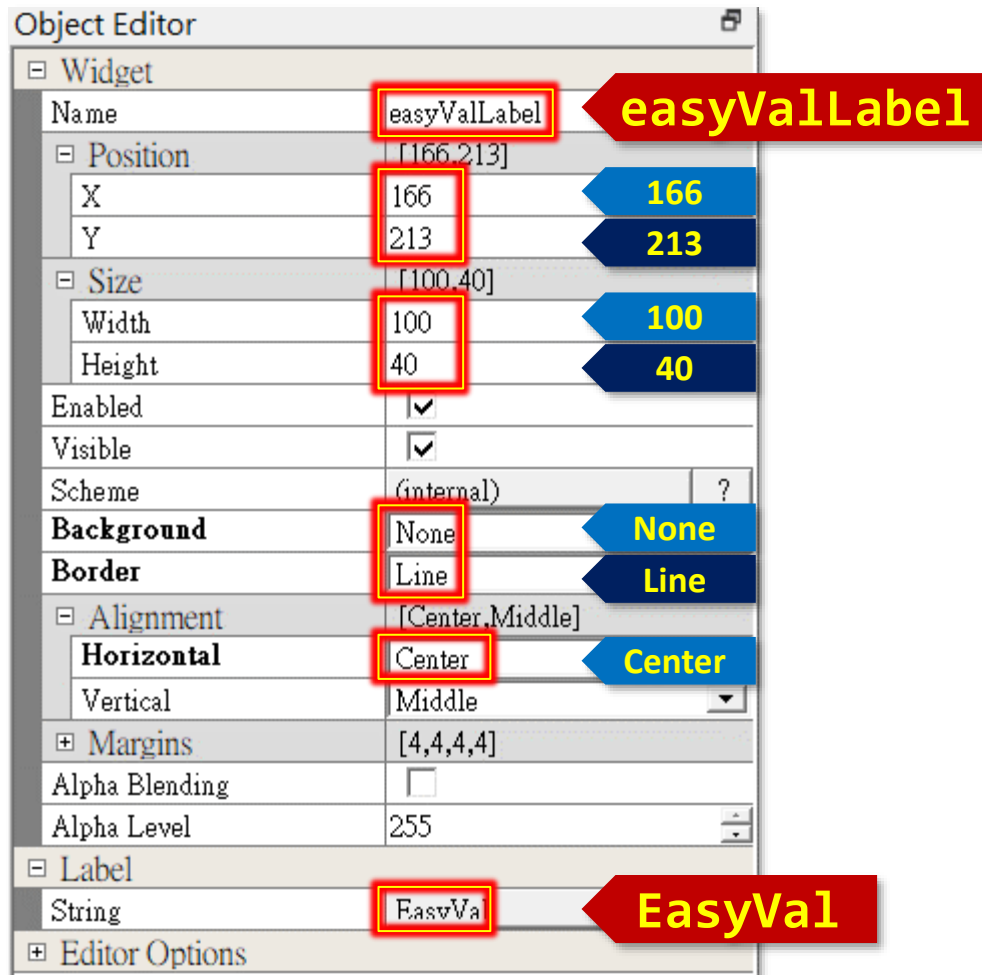
Label Widget with Dynamic Text

- Config Label Widget (for **FastVal** string) as below.



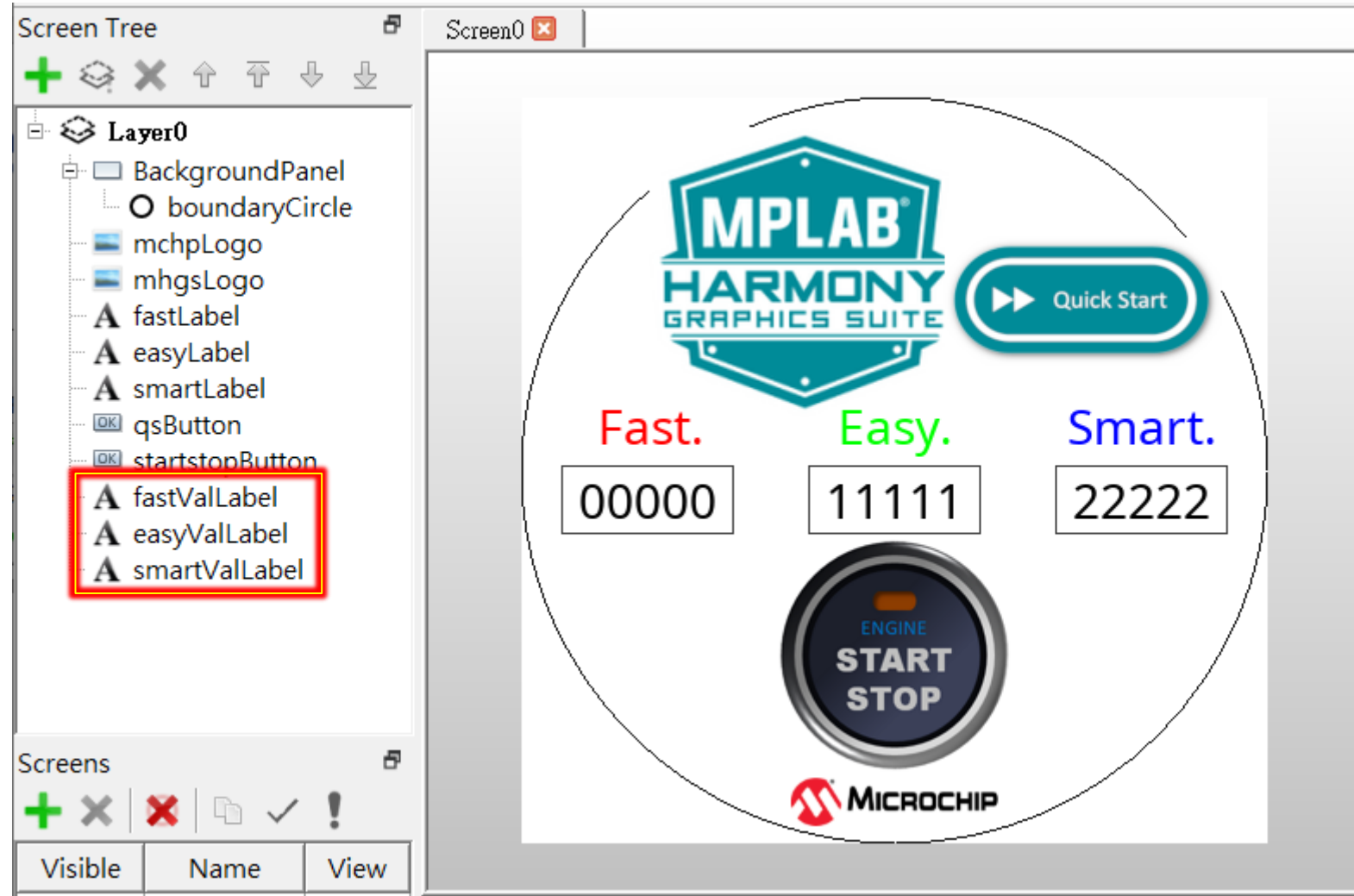
Label Widget with Dynamic Text

- Config Label Widgets (for **EasyVal** and **SmartVal** string) as below.



Label Widget with Dynamic Text

- After configure three new Label widgets.



Label Widget with Dynamic Text

- Generate Code and Implement Dynamic Text on Label widget.

\firmware\src\main.c

```
#include <stddef.h>
#include <stdbool.h>
#include <stdlib.h>
#include "definitions.h"
```

```
volatile bool StartStop = false;
volatile uint32_t Language = language_Default;
```

```
char cCharBuffer[8];
```

C character buffer

```
leChar fastLegatoCharBuffer[8] = {0};
```

```
leChar easyLegatoCharBuffer[8] = {0};
```

```
leChar smartLegatoCharBuffer[8] = {0};
```

```
leFixedString fastValString;
```

```
leFixedString easyValString;
```

```
leFixedString smartValString;
```

Legato Char buffer

Legato string object

```
void event_Screen0_startstopButton_OnPressed(leButtonWidget* btn)
{
    if( StartStop )
        Screen0_startstopButton->fn->setPressedImage(Screen0_startstopButton, (leImage*)&StartStop_StartPressed);
    else
        Screen0_startstopButton->fn->setPressedImage(Screen0_startstopButton, (leImage*)&StartStop_StopPressed);
}
...
```

```
typedef struct leFixedString
{
    leString base;
    const leFixedStringVTable* fn;
    leChar* data;
    uint16_t capacity;
    uint16_t length;
    const leFont* font;
} leFixedString;
```

/**< base data */
/**< function table */
/**< data storage */
/**< string capacity */
/**< string length */
/**< string font */

Label Widget with Dynamic Text

- Dynamic Strings and Font initialization depend to **Language**.

`\firmware\src\main.c`

```
...  
void DynamicStringInitFont( leFixedString* String, uint32_t StringFontID )  
{  
    String->fn->setFont( String, leStringTable_GetStringFont(leGetStringTable(), StringFontID, Language) );  
}
```

Dynamic String Font Initial

Dynamic Language ID (Toggle by qsButton)

```
void DynamicStringInit( leChar* leBuffer, uint32_t size, leFixedString* String, uint32_t StringFontID )  
{  
    leFixedString_Constructor( String, leBuffer, size );  
    DynamicStringInitFont( String, StringFontID );  
}
```

Dynamic String Initial

Create and initialize a Legato String object
Assign a String Font to the Legato String

Call Font Initial

```
void event_Screen0_startstopButton_OnPressed(leButtonWidget* btn) { ... }  
...
```

```
int main ( void )  
{  
    SYS_Initialize ( NULL );
```

```
    DynamicStringInit( fastLegatoCharBuffer, sizeof(fastLegatoCharBuffer), &fastValString, stringID_FastVal );  
    DynamicStringInit( easyLegatoCharBuffer, sizeof(easyLegatoCharBuffer), &easyValString, stringID_EasyVal );  
    DynamicStringInit( smartLegatoCharBuffer, sizeof(smartLegatoCharBuffer), &smartValString, stringID_SmartVal );
```

Dynamic String Initial

Label Widget with Dynamic Text

- Dynamic Strings update by given Text String buffer.

`\firmware\src\main.c`

```
void DymaticStringInit( leChar* leBuffer, uint32_t size, leFixedString* String, uint32_t StringFontID ) { ... }
```

```
void DymaticStringUpdate( char* cBuffer, leFixedString* String, leLabelWidget* Label )  
{  
    String->fn->setFromCStr( String, cBuffer );  
    Label->fn->setString( Label, (leString*)String );  
}
```

Dynamic String Update

Convert the C character string to Legato String
Set Legato String to target Label widget

```
void event_Screen0_startstopButton_OnPressed(leButtonWidget* btn) { ... }
```

```
void event_Screen0_startstopButton_OnReleased(leButtonWidget* btn)  
{
```

```
    ...  
    if( StartStop )  
    {  
        ...  
        sprintf( cCharBuffer, "Name" );  
        DymaticStringUpdate( cCharBuffer, &smartValString, Screen0_smartValLabel );  
    }  
    else  
    {  
        ...  
        sprintf( cCharBuffer, "Pause" );  
        DymaticStringUpdate( cCharBuffer, &smartValString, Screen0_smartValLabel );  
    }  
}
```

Modify to your Name

Show "Name" on SmartVal Label
while Start/Stop button enable

Dynamic String Update

Show "Pause" on SmartVal Label
while Start/Stop button disable

Dynamic String Update

Label Widget with Dynamic Text

- Reconstructor the Font of string while press **qsButton** to change **Language**.

`\firmware\src\main.c`

```
...
void DynamicStringInitFont( leFixedString* String, uint32_t StringFontID )
void DymaticStringInit( leChar* leBuffer, uint32_t size, leFixedString* String, uint32_t StringFontID ) { ... }
void DymaticStringUpdate( char* cBuffer, leFixedString* String, leLabelWidget* Label )
void event_Screen0_startstopButton_OnPressed(leButtonWidget* btn) { ... }
void event_Screen0_startstopButton_OnReleased(leButtonWidget* btn) { ... }
void event_Screen0_qsButton_OnPressed(leButtonWidget* btn) { ... }

void event_Screen0_qsButton_OnReleased(leButtonWidget* btn)
{
    RGB_LED_B_Off();

    Language = (Language+1) % leStringTable_GetLanguageCount( leGetStringTable() );
    leSetStringLanguage( Language );

    DynamicStringInitFont( &fastValString, stringID_FastVal );
    DynamicStringInitFont( &easyValString, stringID_EasyVal );
    DynamicStringInitFont( &smartValString, stringID_SmartVal );
}

int main ( void )
{
    ...
    while ( true )
    {
        ...
    }
}
```

Reconstructor the Font of string while
press qsButton to change **Language**.

Label Widget with Dynamic Text

- Create a Timer to increase a Counter and update String in given period.

`\firmware\src\main.c`

```
volatile bool UpdateFlag = false;  
volatile unsigned int Counter = 0;
```

GUI Update Flag

Counter value variable

```
char cCharBuffer[8];  
...  
leFixedString smartValString;
```

```
void TimerCallback ( uintptr_t context )  
{  
    if( StartStop ) {  
        Counter = (Counter + 1)%100000;  
        UpdateFlag = true;  
    }  
}
```

Timer Callback function

Counter start or stopped depend to Start/Stop button
Limit Counter value between 0 ~ 99999 for max 5 chars
Set the UI Update Flag for Dynamic Text Update in main loop

```
void DynamicStringInitFont( leFixedString* String, uint32_t StringFontID ) { ... }  
...
```

```
int main ( void )  
{  
    SYS_Initialize ( NULL );
```

```
    SYS_TIME_CallbackRegisterMS( TimerCallback, 0, 200, SYS_TIME_PERIODIC );  
    ...
```

Timer Callback Function Register in 200ms period

Label Widget with Dynamic Text

- Update the scroll animated Text string in Timer period.

`\firmware\src\main.c`

```
void UpdateScreen( void ) Task for Update UI
{
    if( UpdateFlag )
    {
        UpdateFlag = false;

        sprintf(cCharBuffer, "%u", Counter);
        DymaticStringUpdate( cCharBuffer, &fastValString, Screen0_fastValLabel );

        sprintf(cCharBuffer, "%c%c%c%c%c", 32 + ((Counter+0)%(126-32+1)), 32 + ((Counter+1)%(126-32+1)), 32 + ((Counter+2)%(126-32+1)),
            32 + ((Counter+3)%(126-32+1)), 32 + ((Counter+4)%(126-32+1)));
        DymaticStringUpdate( cCharBuffer, &easyValString, Screen0_easyValLabel );
    }
}

int main ( void )
{
    ...
    while ( true )
    {
        SYS_Tasks ( );

        UpdateScreen(); Task for Update UI
    }
    ...
}
```

Show **Counter** on **FastVal** Label

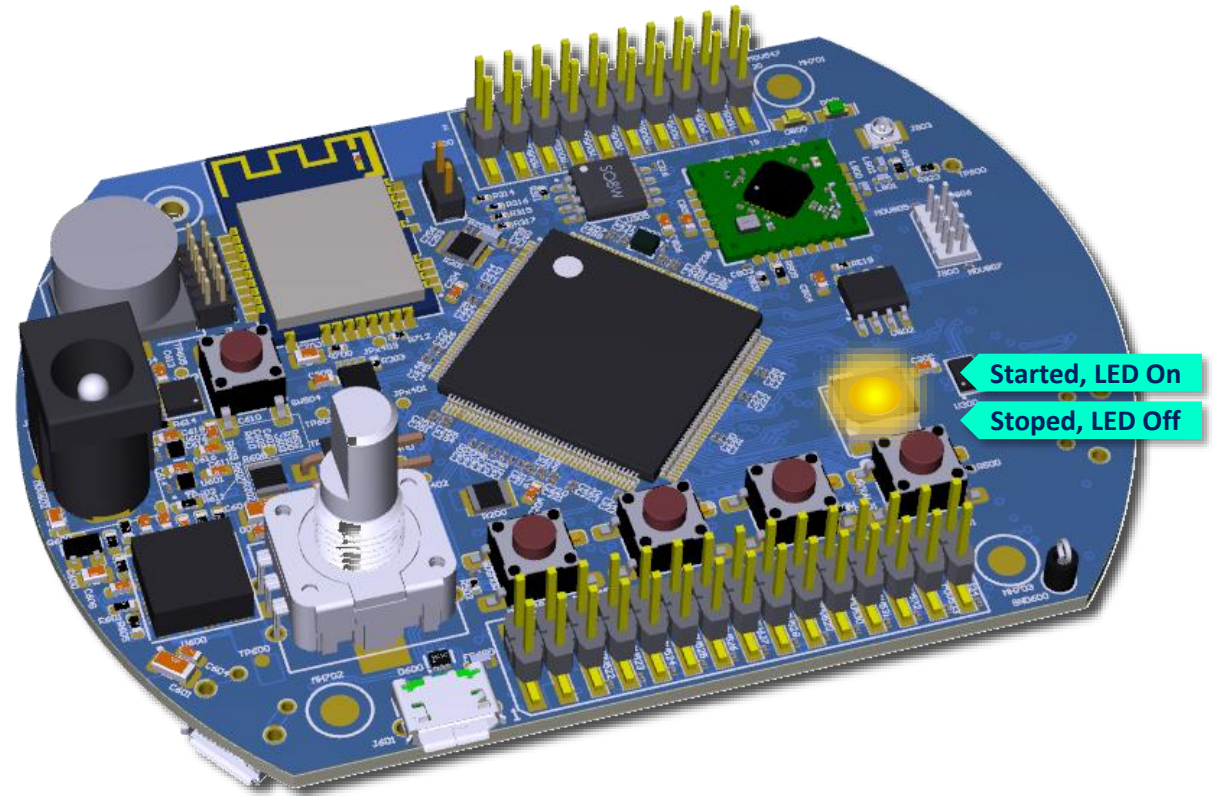
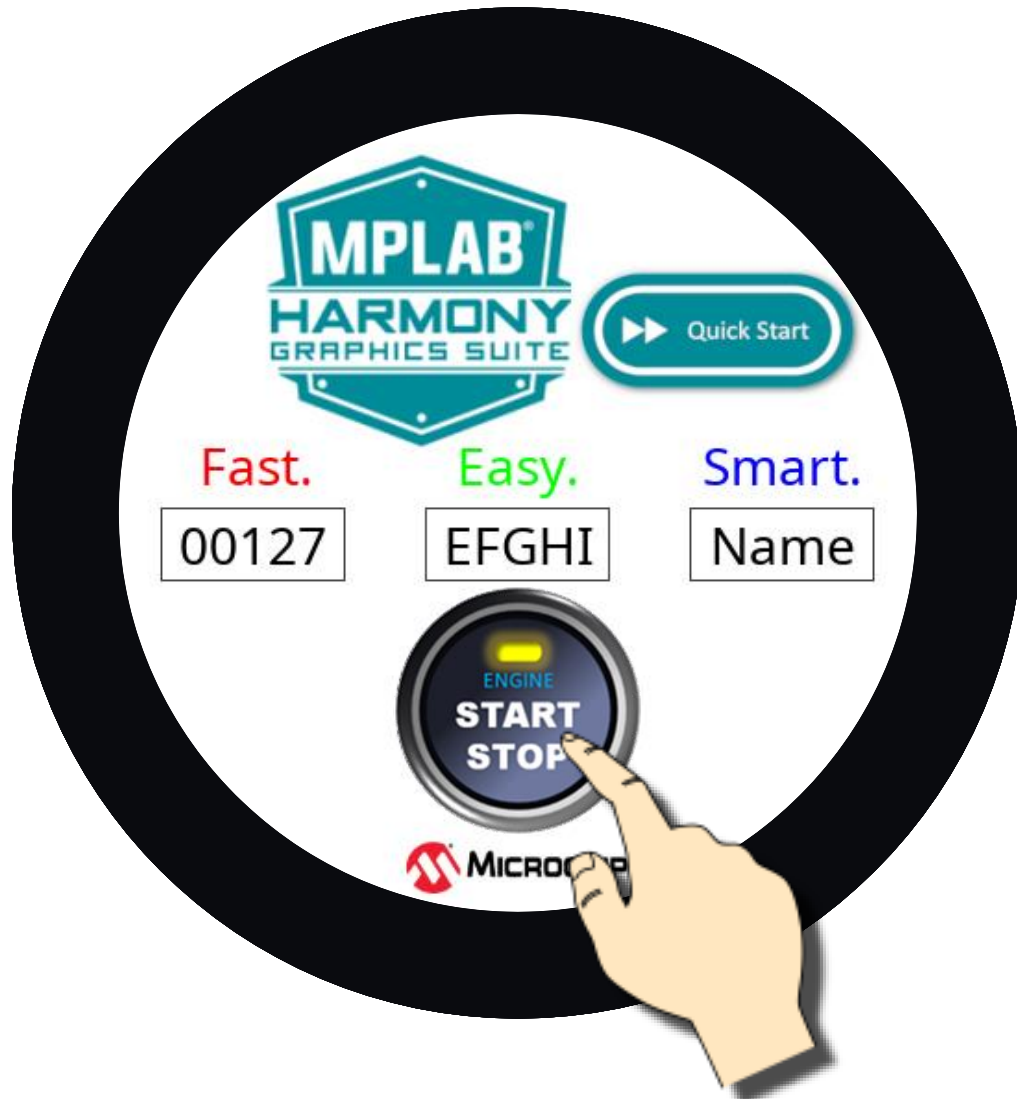
Show **ASCII** on **EasyVal** Label
It will show 5 chars as moving left

32	42	52	62	72	82	92	102	112	122
[SPACE]	*	4	+0	H	R	\	f	p	z
!	+	5	+1	I	S	]	g	q	{
"	,	6	+2	J	T	^	h	r	
#	-	7	+3	K	U	_	i	s	}
\$	.	8	+4	L	V	`	j	t	~
%	/	9		C	M	W	a	k	u [DEL]
&	0	:		D	N	X	b	l	v
'	1	;		E	O	Y	c	m	w
(	2	<		F	P	Z	d	n	x
)	3	=		G	Q	[	e	o	y

Font ASCII index
start from 32
Stop on 126
Total (126-32+1)

Label Widget with Dynamic Text

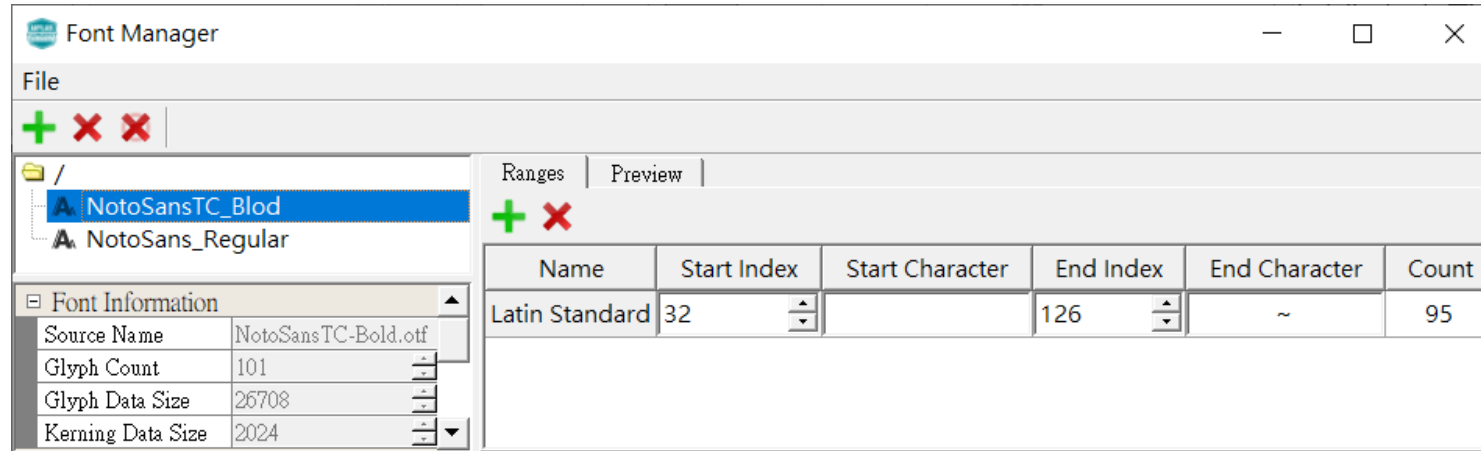
- The Start/Stop Button could Enable/Disable Dynamic Text.



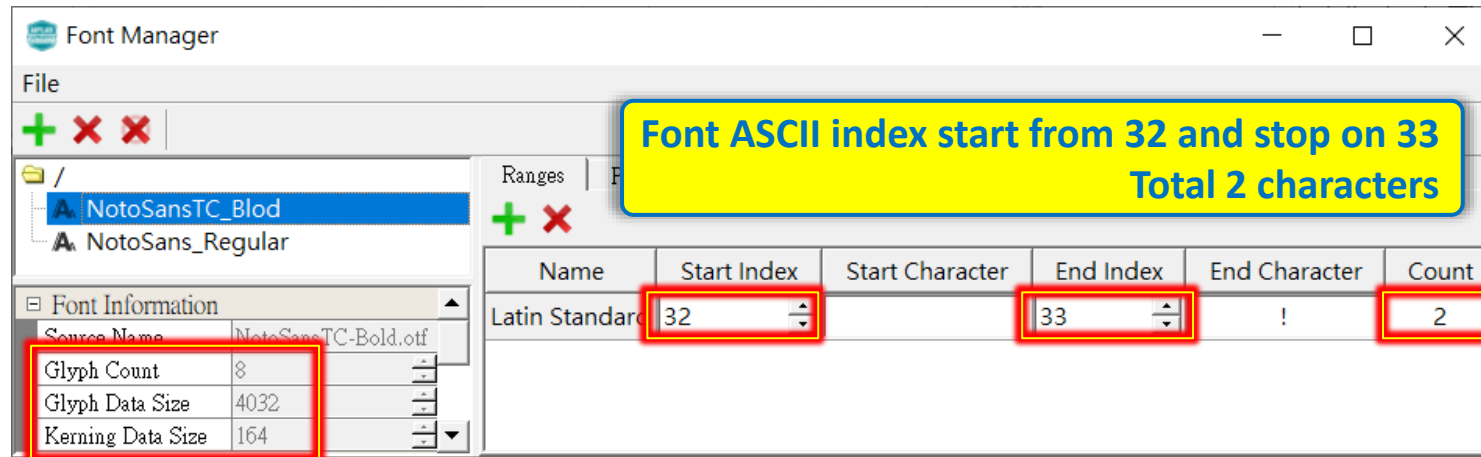
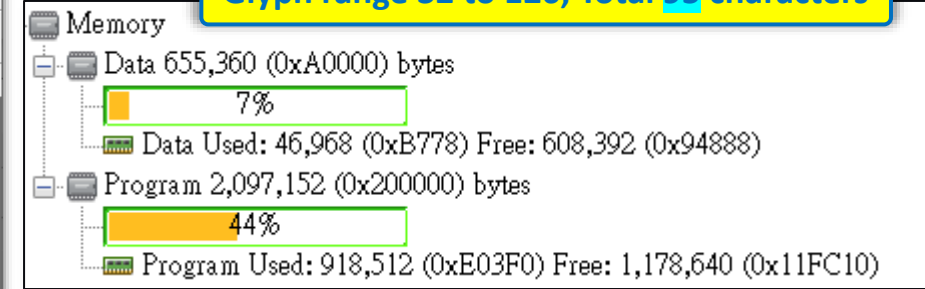
Label Widget with Dynamic Text

Discuss

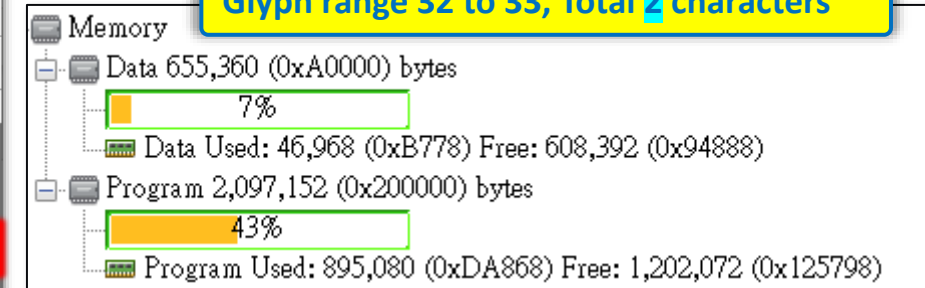
- How to reduce Flash size if not use alphabet characters of a Chinese Font ?
 - Please configure the Glyph Range to minimize the count of index range.



Glyph range 32 to 126, Total 95 characters



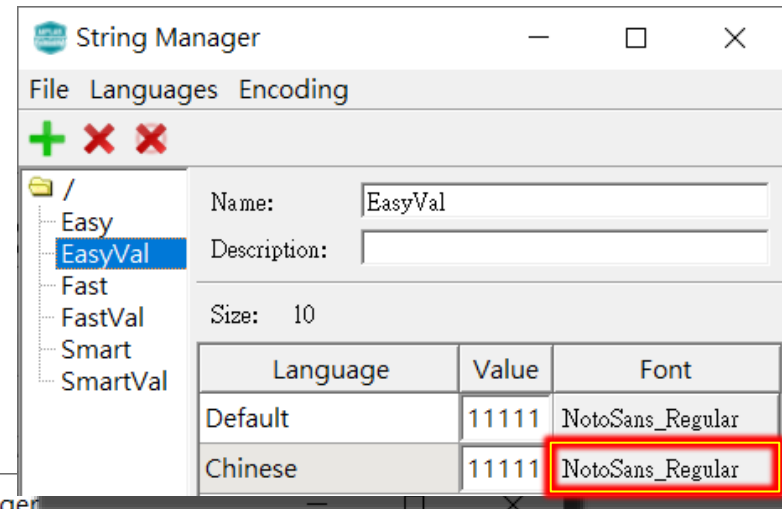
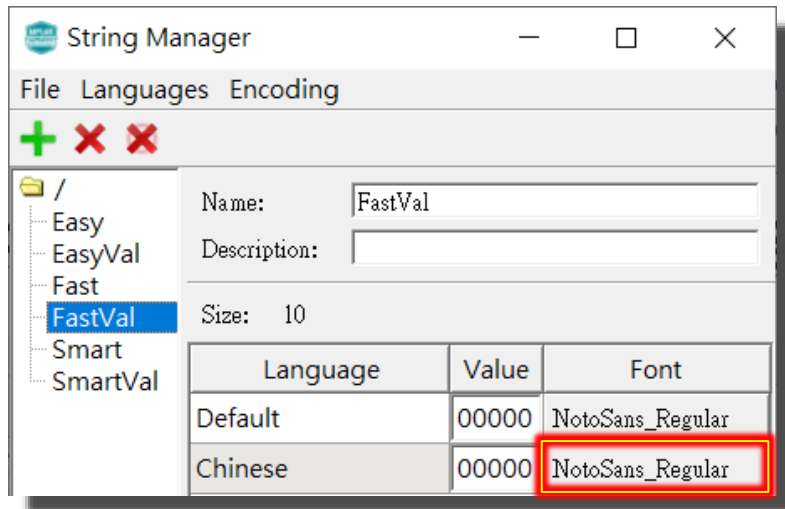
Glyph range 32 to 33, Total 2 characters



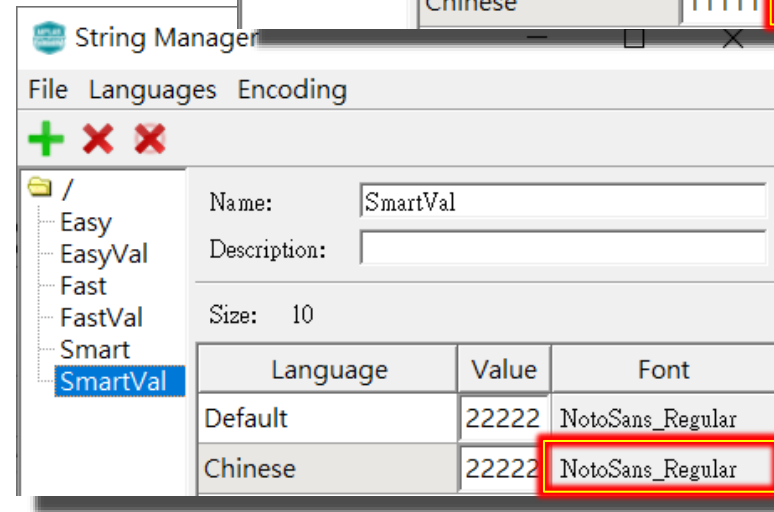
Label Widget with Dynamic Text

Discuss

- Label value to use the same font both of **Default** and **Chinese** language.



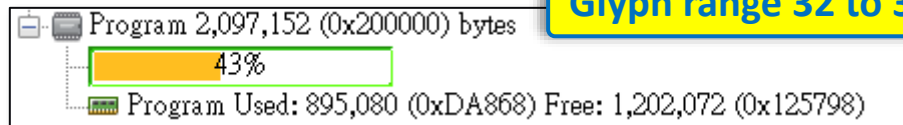
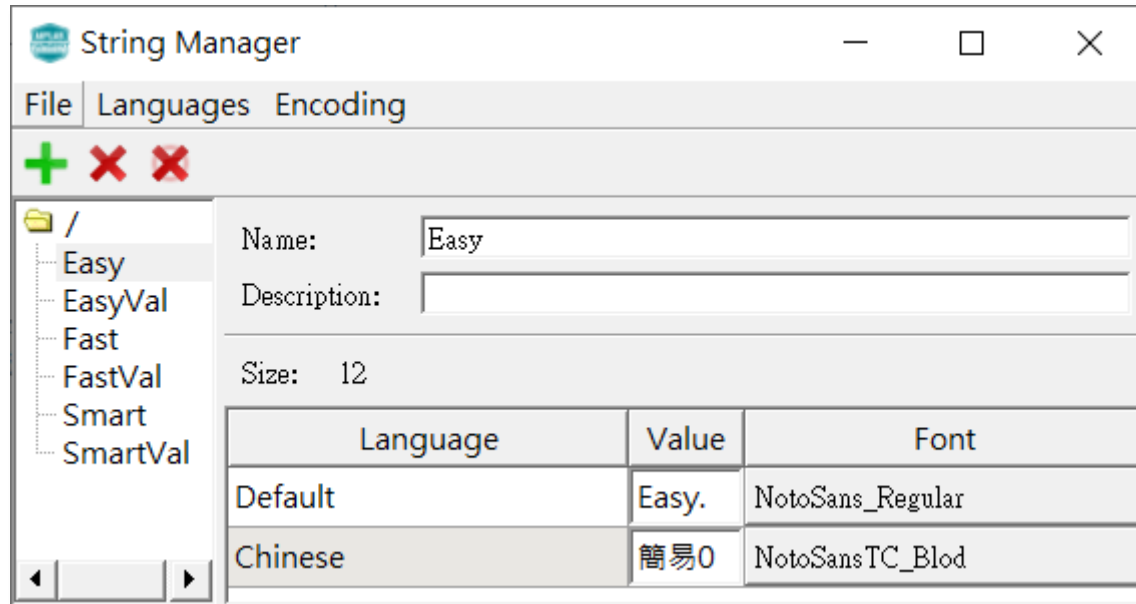
String Name	Value	Font
FastVal	00000	NotoSans_Regular
	00000	NotoSans_Regular
EasyVal	11111	NotoSans_Regular
	11111	NotoSans_Regular
SmartVal	22222	NotoSans_Regular
	22222	NotoSans_Regular



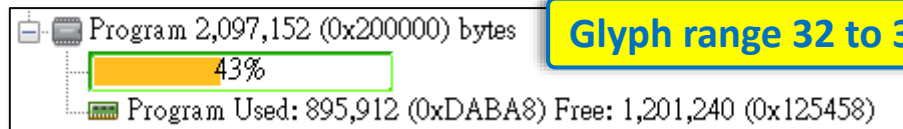
Label Widget with Dynamic Text

Discuss

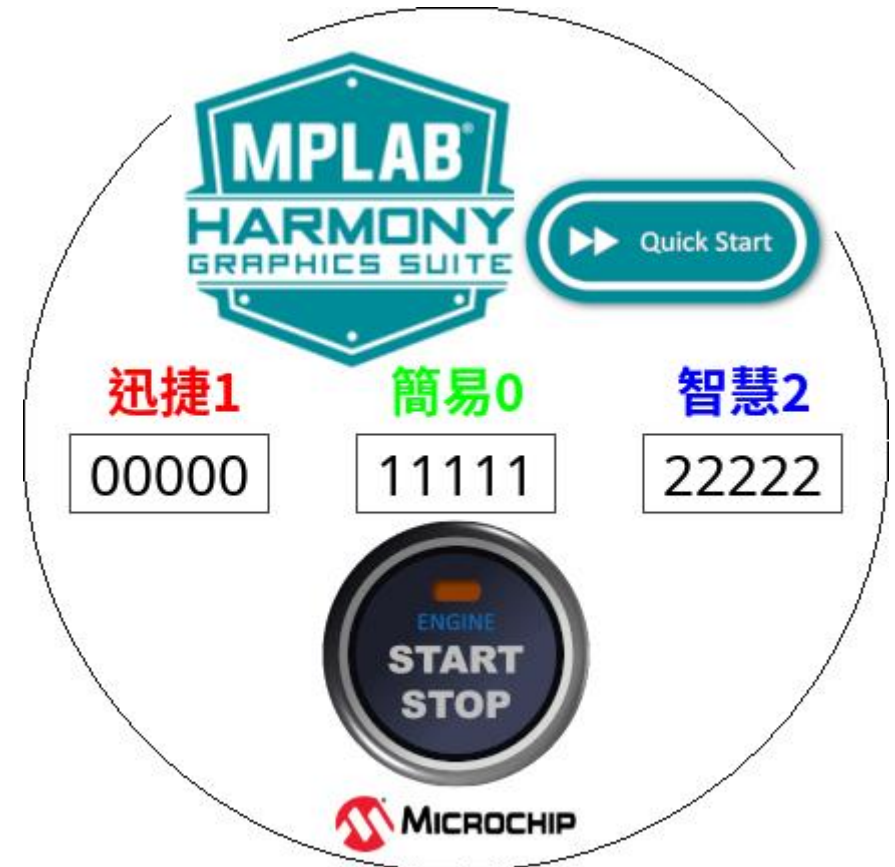
- What if to minimize Glyph range but have used partial characters ?



Glyph range 32 to 33 without '0', '1', '2'



Glyph range 32 to 33 with '0', '1', '2'

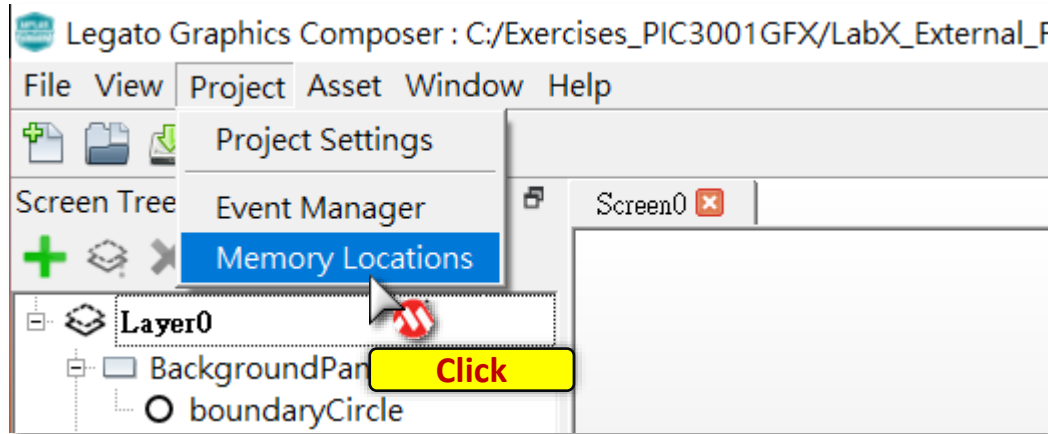


External Graphics Resources - Font

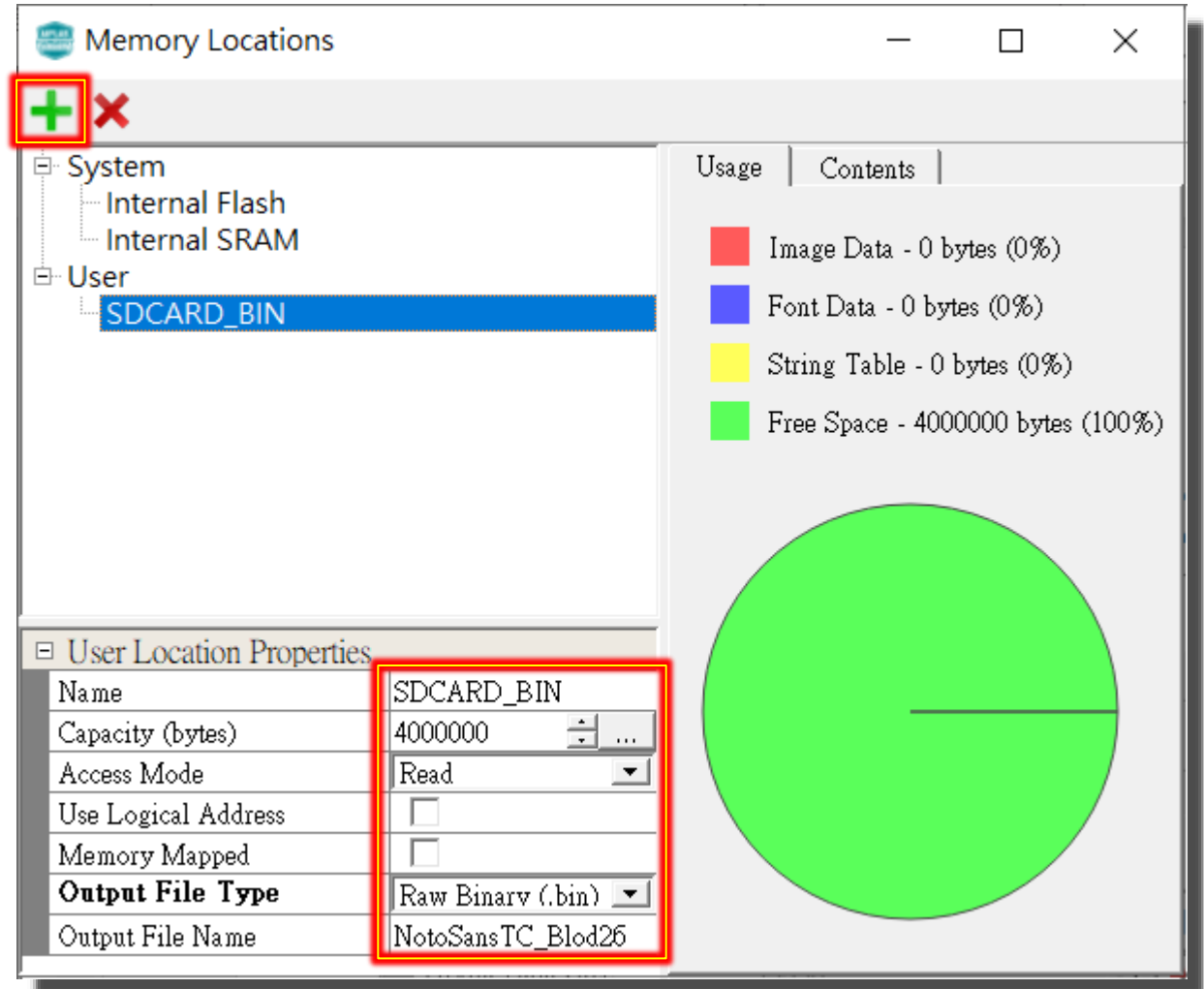
PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

External Graphics Resources - Font

- Add a customized Memory Location and configure as below.



Name	SDCARD_BIN
Capacity (bytes)	4000000
Access Mode	Read
Output File Type	Raw Binary (.bin)
Output File Name	NotoSansTC_Blod26



External Graphics Resources - Font

- Add a new Glyph range for more Chinese words in Font Asset Manager.

The screenshot shows the Legato Graphics Composer interface. The 'Asset' menu is open, and the 'Fonts' option is selected. A yellow callout box with the text 'Click' points to the 'Fonts' menu item. The 'Font Manager' window is open, showing the 'Ranges' tab. The 'NotoSansTC_Blod' font is selected. A yellow callout box with the text 'Click' points to the green plus icon in the 'Ranges' tab. The 'Font Information' section shows the source name 'NotoSansTC-Bold.otf' and the glyph count '4094'. The 'Font Options' section shows the 'Glyph Table Location' as 'SDCARD_BIN'. A yellow callout box with the text 'Click' points to the 'SDCARD_BIN' option. The 'Ranges' table shows the 'Chinese Word' range with a start index of 19968 and an end index of 23960. A yellow callout box with the text 'Click' points to the 'Chinese Word' range. The 'Select Memory Location' dialog is open, showing the 'SDCARD_BIN' location selected. A yellow callout box with the text 'Click' points to the 'SDCARD_BIN' option.

Font Manager

File

Ranges

Preview

Name	Start Index	Start Character	End Index	End Character	Count
Latin Standard	32		126	~	95
Chinese Word	19968	—	23960	機	3993

Font Information

Source Name: NotoSansTC-Bold.otf

Glyph Count: 4094

Glyph Data Size: 2570792

Kerning Data Size: 81884

Font Options

Glyph Table Location: SDCARD_BIN

Data Table Location: SDCARD_BIN

Name: NotoSansTC-Bold

Size: 26

Antialiased: ☒

Select Memory Location

System

Internal Flash

Internal SRAM

User

SDCARD_BIN

OK

Cancel

Click

Click

Click

Click

Name

End Index

Start Index

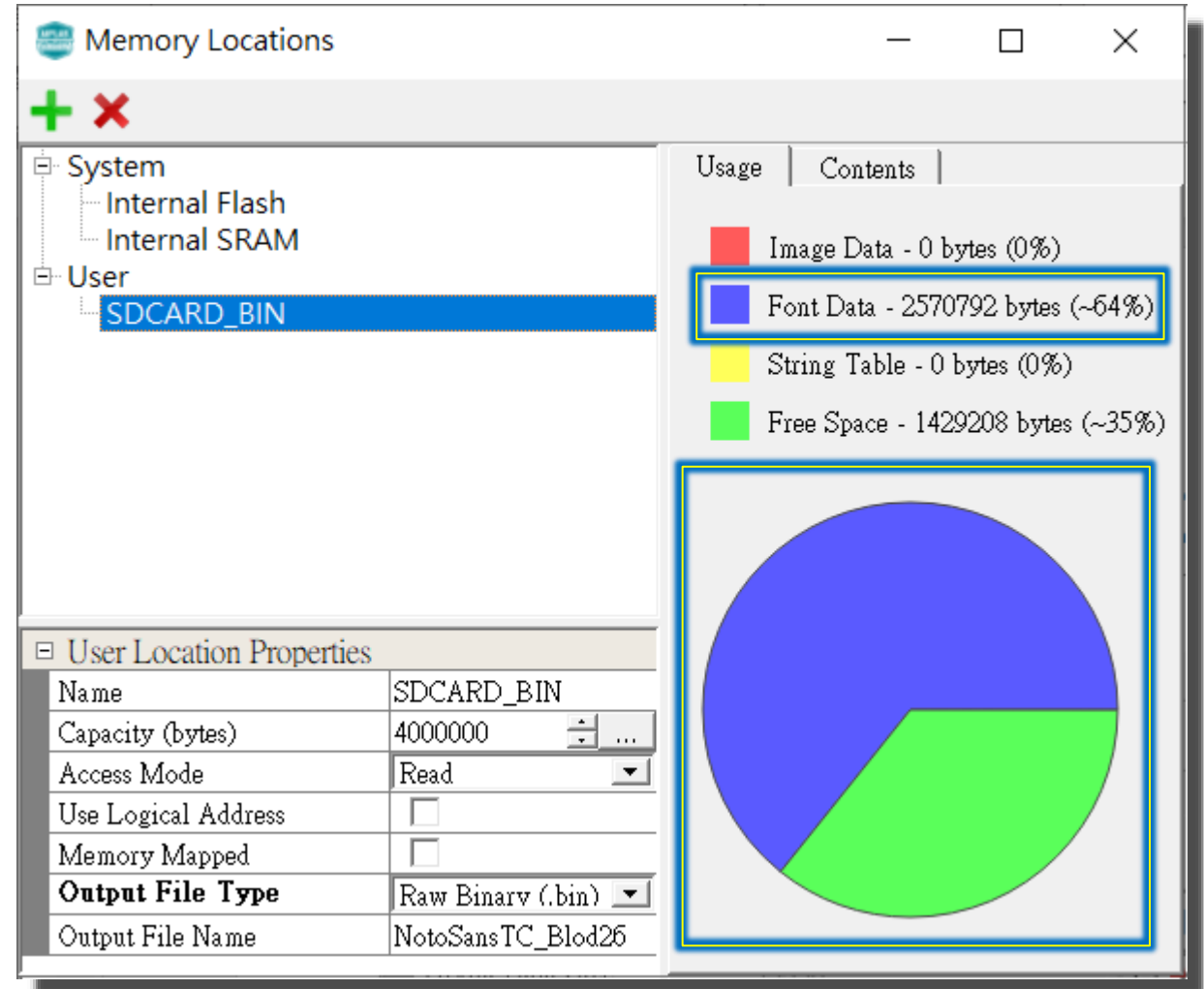
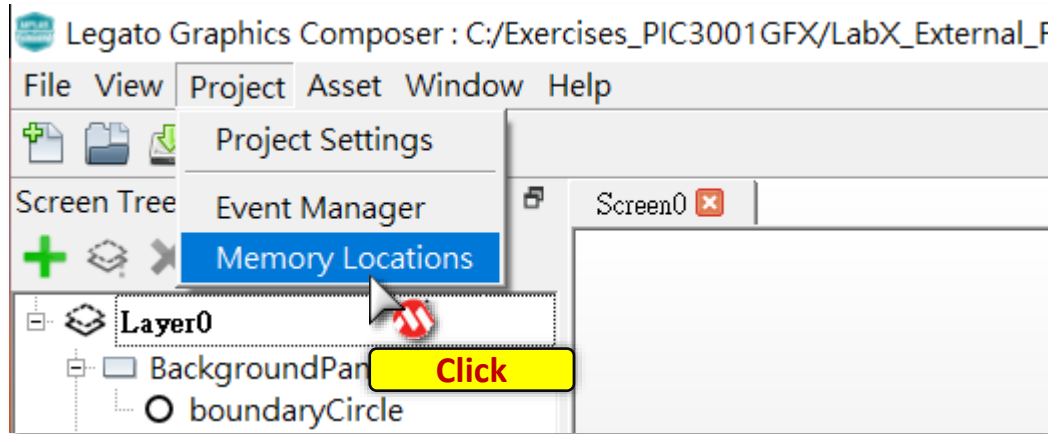
Chinese Word

23960

19968

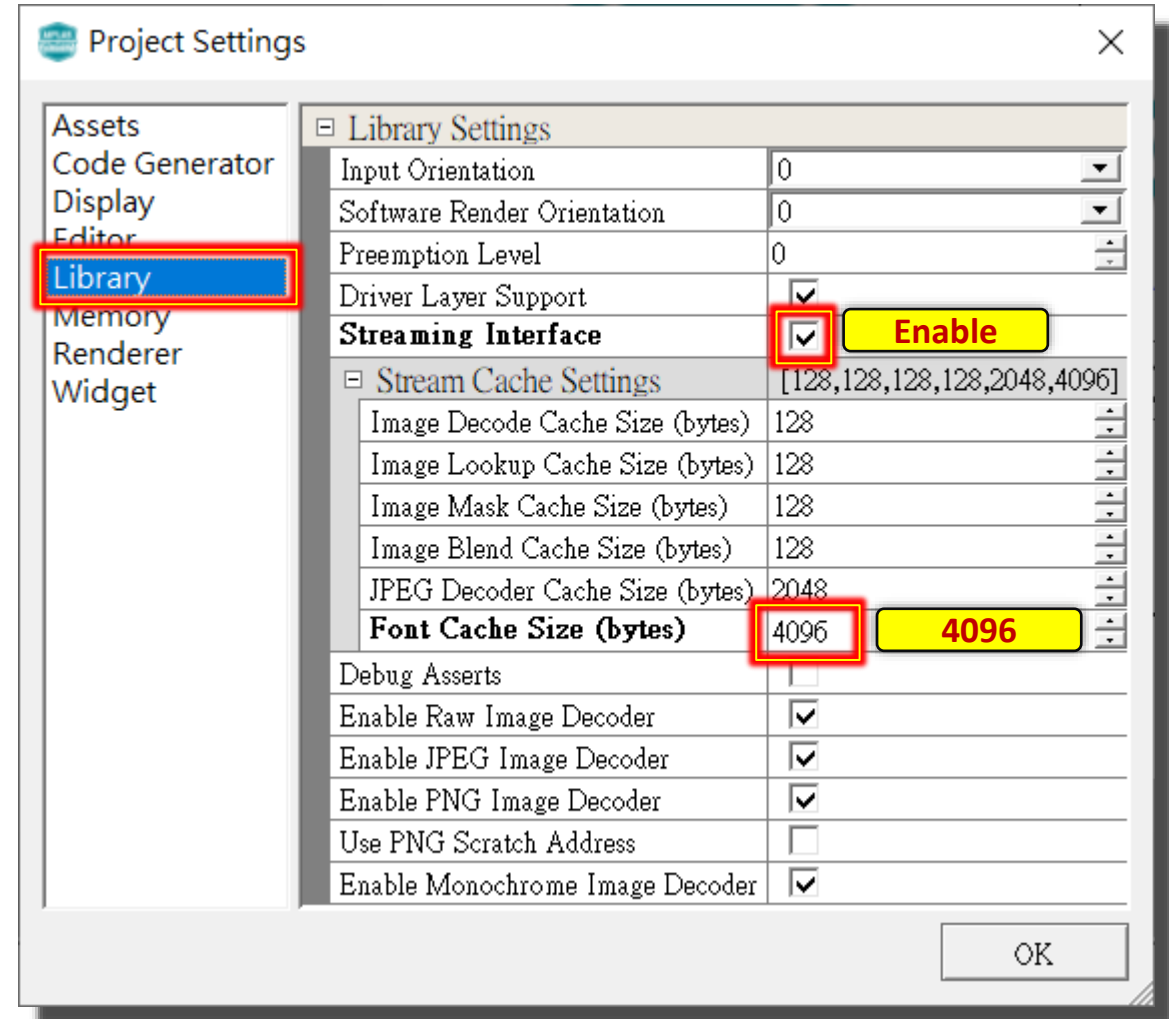
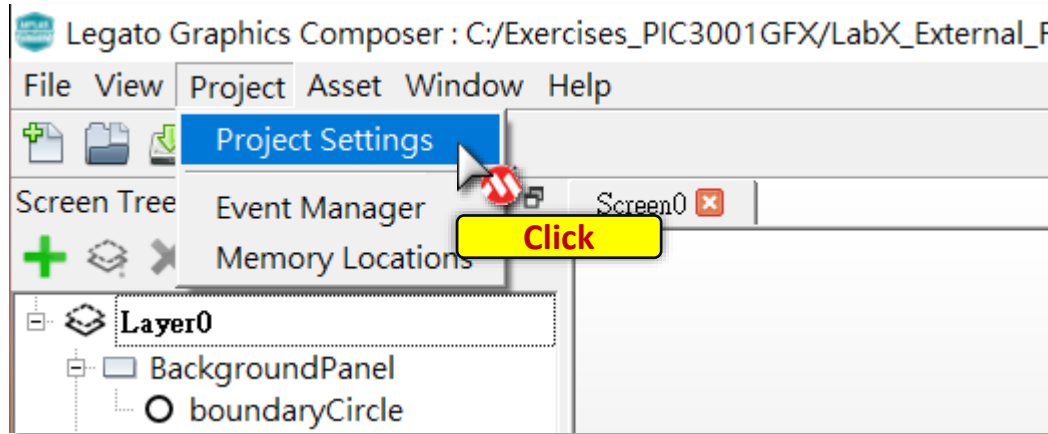
External Graphics Resources - Font

- Check Memory consumption in Memory Location window.



External Graphics Resources - Font

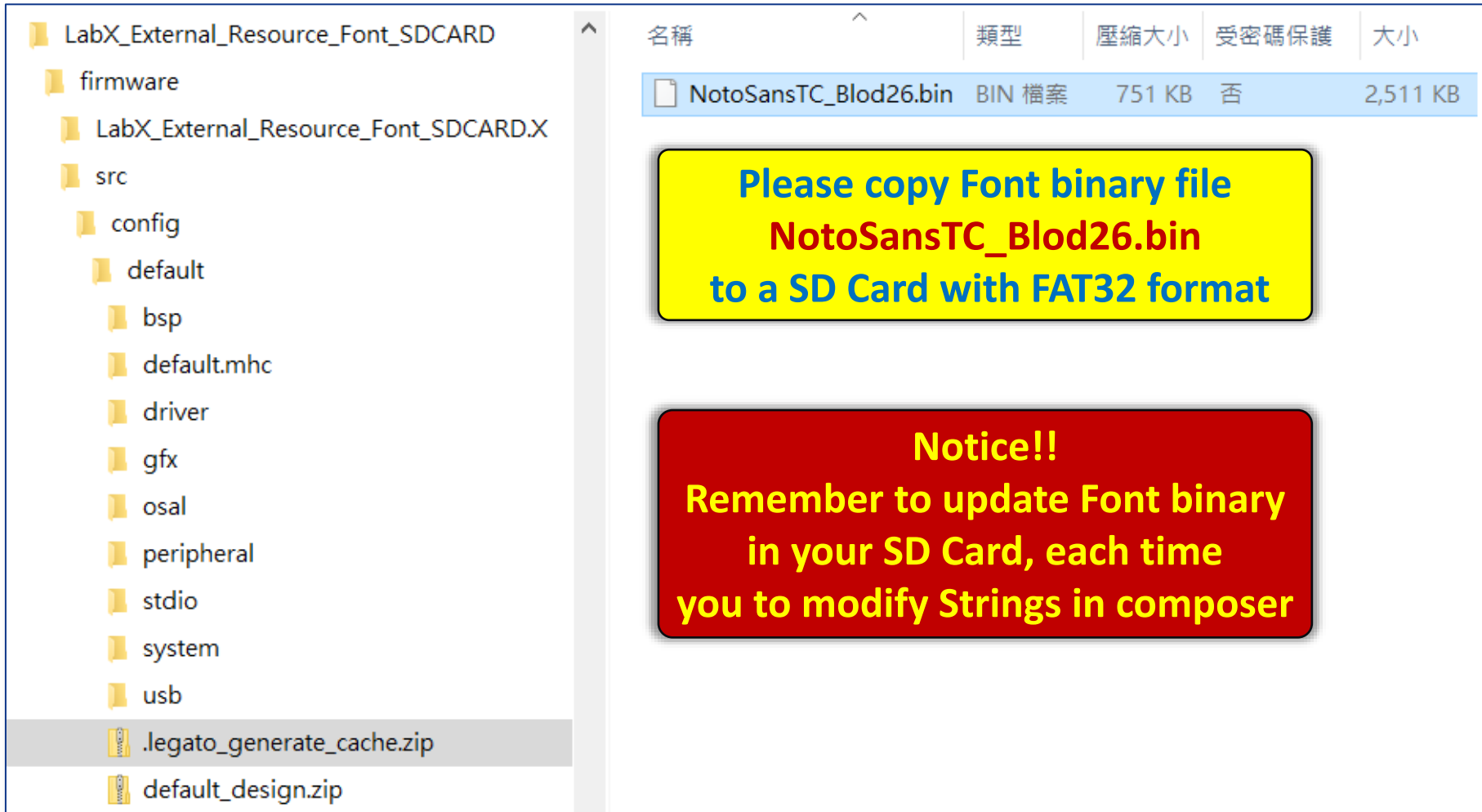
- Enable Stream feature in Project settings.



External Graphics Resources - Font

- Find the Font binary file **NotoSansTC_Blod26.bin** in below ZIP file.

\[Project]\firmware\src\config\default\.legato_generate_cache.zip



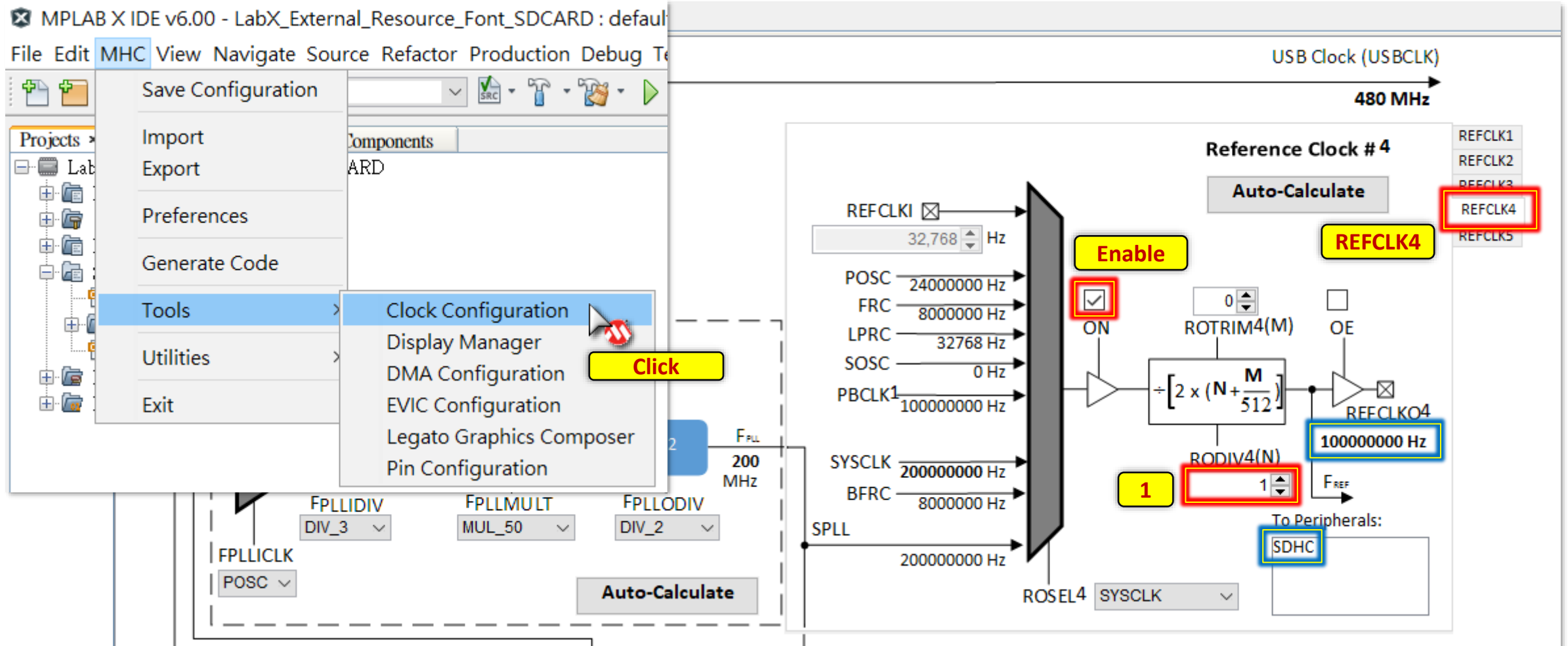
名稱	類型	壓縮大小	受密碼保護	大小
NotoSansTC_Blod26.bin	BIN 檔案	751 KB	否	2,511 KB

**Please copy Font binary file
NotoSansTC_Blod26.bin
to a SD Card with FAT32 format**

**Notice!!
Remember to update Font binary
in your SD Card, each time
you to modify Strings in composer**

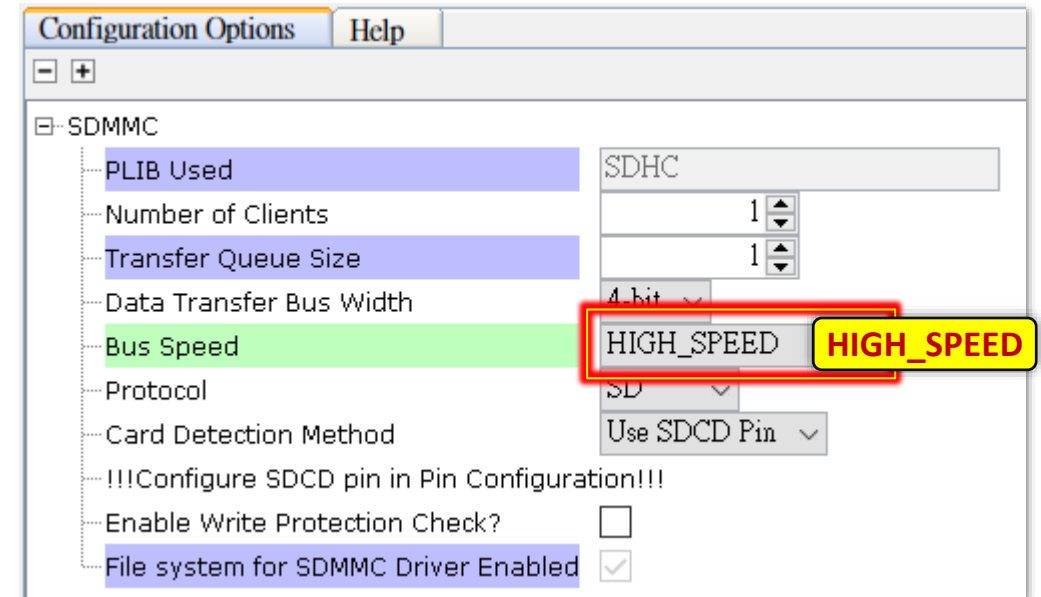
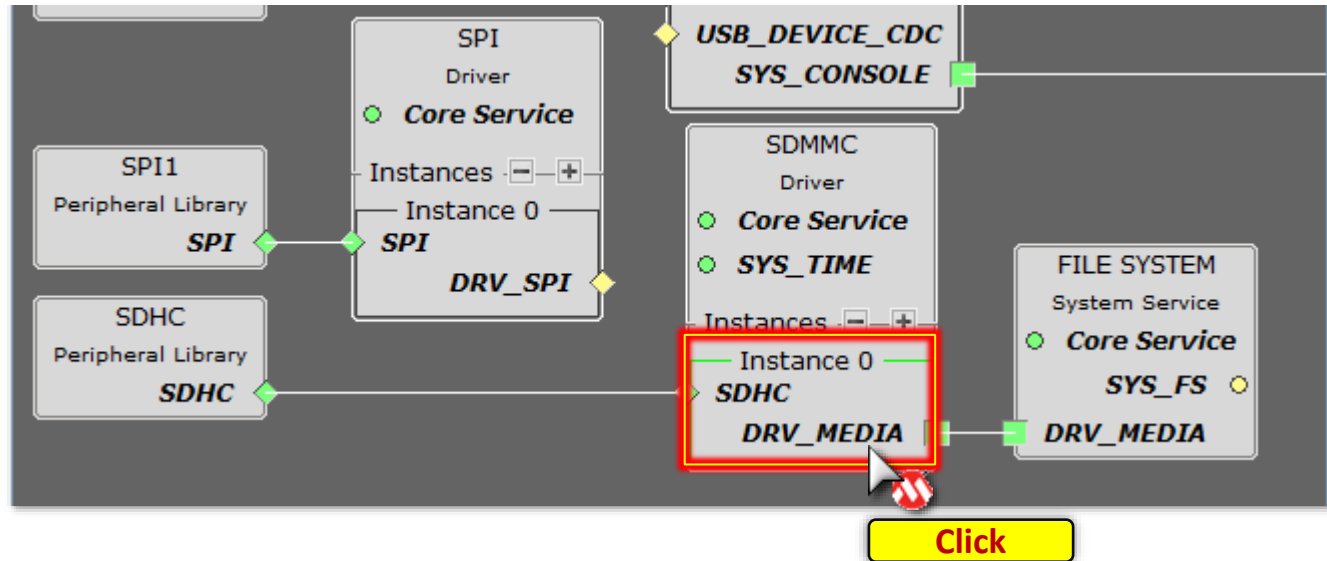
External Graphics Resources - Font

- Enable SDHC reference clock in Clock Configurator of Harmony.



External Graphics Resources - Font

- Configure SDMMC Bus Speed to **HIGH_SPEED** in Harmony.



External Graphics Resources - Font

- Implement SD Card auto mount feature.

```
bool SDCardMounted = 0;

static void AutoMountEventHandler( SYS_FS_EVENT event, void* eventData, uintptr_t context )
{
    switch(event)
    {
        case SYS_FS_EVENT_MOUNT:
        case SYS_FS_EVENT_MOUNT_WITH_NO_FILESYSTEM:
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0) == 0)
            {
                SDCardMounted = 1;
                SYS_CONSOLE_MESSAGE("\r\nMount SD card success!\r\n");
            }
            break;

        case SYS_FS_EVENT_UNMOUNT:
            if(strcmp((const char *)eventData, SYS_FS_MEDIA_IDX0_MOUNT_NAME_VOLUME_IDX0) == 0)
            {
                SDCardMounted = 0;
                SYS_CONSOLE_MESSAGE("\r\nUnmount SD card!\r\n");
            }
            break;

        case SYS_FS_EVENT_ERROR:
            break;
    }
}

int main ( void )
{
    SYS_Initialize ( NULL );

    SYS_FS_EventHandlerSet((void const*)AutoMountEventHandler, (uintptr_t)NULL);

    DynamicStringInit( fastLegatoCharBuffer, sizeof(fastLegatoCharBuffer), &fastValString, stringID_FastVal );
    DynamicStringInit( easyLegatoCharBuffer, sizeof(easyLegatoCharBuffer), &easyValString, stringID_EasyVal );
    DynamicStringInit( smartLegatoCharBuffer, sizeof(smartLegatoCharBuffer), &smartValString, stringID_SmartVal );
}
```


External Graphics Resources - Font

- Implement Font file open and Media file handle Get function .

```
#define FONT_BIN_FILENAME "/NotoSansTC_Blod26.bin"
SYS_FS_HANDLE fHandle_SDCARD = SYS_FS_HANDLE_INVALID;
bool AssetFileReady = 0;

static void APP_SDCardFileOpen( void )
{
    if( SDCardMounted==1 && AssetFileReady==0 )
    {
        fHandle_SDCARD = SYS_FS_FileOpen( FONT_BIN_FILENAME, SYS_FS_FILE_OPEN_READ );
        if( fHandle_SDCARD == SYS_FS_HANDLE_INVALID )
        {
            SYS_FS_FileClose( fHandle_SDCARD );
        }
        else if( SYS_FS_FileSize( fHandle_SDCARD ) > 0 )
        {
            AssetFileReady = 1;
            SYS_CONSOLE_PRINT("\r\nFont file %s Open success!\r\n", FONT_BIN_FILENAME );
        }
    }
}

SYS_FS_HANDLE APP_GetFileHandle( uint32_t dataLocation )
{
    SYS_FS_HANDLE handle = SYS_FS_HANDLE_INVALID;

    switch(dataLocation)
    {
        case LE_STREAM_LOCATION_ID_SDCARD_BIN:
        {
            handle = fHandle_SDCARD;
            break;
        }
        default:
            break;
    }
    return handle;
}
```

External Graphics Resources - Font

- Implement Legato Media Stream Callback functions.

```
#include <stddef.h>
#include <stdbool.h>
#include <stdlib.h>
#include "definitions.h"

#define FONT_BIN_FILENAME "/NotoSansTC_Blod26.bin"
bool SDCardMounted = 0;
SYS_FS_HANDLE fHandle_SDCARD = SYS_FS_HANDLE_INVALID;
bool AssetFileReady = 0;
leStream* MediaStream;
SYS_FS_HANDLE fHandle_MediaStream = SYS_FS_HANDLE_INVALID;

volatile bool StartStop = false;
...

static void AutoMountEventHandler( SYS_FS_EVENT event, void* eventData, uintptr_t context ) { ... }
static void APP_SDCardFileOpen( void ) { ... }
SYS_FS_HANDLE APP_GetFileHandle( uint32_t dataLocation ) {...}

leResult leApplication_MediaOpenRequest(leStream* stream)
{
    if ( AssetFileReady==0 )
        return LE_FAILURE;

    MediaStream = stream;
    fHandle_MediaStream = APP_GetFileHandle( MediaStream->desc->location );

    if( SYS_FS_FileEOF( fHandle_MediaStream ) == true )
    {
        if( SYS_FS_FileSeek( fHandle_MediaStream, (int32_t)0, SYS_FS_SEEK_SET ) == -1 )
        {
            return LE_FAILURE;
        }
    }

    return LE_SUCCESS;
}
```

External Graphics Resources - Font

- Implement Legato Media Stream Callback functions.

```
#define FONT_BIN_FILENAME "/NotoSansTC_Blod26.bin"
bool SDCardMounted = 0;
SYS_FS_HANDLE fHandle_SDCARD = SYS_FS_HANDLE_INVALID;
bool AssetFileReady = 0;
leStream* MediaStream;
SYS_FS_HANDLE fHandle_MediaStream = SYS_FS_HANDLE_INVALID;
...
static void AutoMountEventHandler( SYS_FS_EVENT event, void* eventData, uintptr_t context ) { ... }
static void APP_SDCardFileOpen( void ) { ... }
SYS_FS_HANDLE APP_GetFileHandle( uint32_t dataLocation ) {...}

leResult leApplication_MediaOpenRequest(leStream* stream) { ... }

leResult leApplication_MediaReadRequest(leStream* stream, // stream reader
                                       uint32_t address, // address
                                       uint32_t readSize, // dest size
                                       uint8_t* destBuffer)
{
    if( SYS_FS_FileSeek( fHandle_MediaStream, (int32_t)address, SYS_FS_SEEK_SET ) == -1 )
    {
        return LE_FAILURE;
    }

    if( SYS_FS_FileRead( fHandle_MediaStream, (void*)destBuffer, readSize) <= 0 )
    {
        return LE_FAILURE;
    }

    leStream_DataReady( MediaStream );

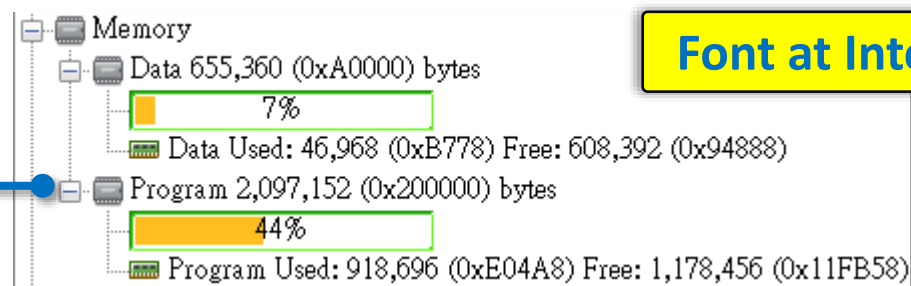
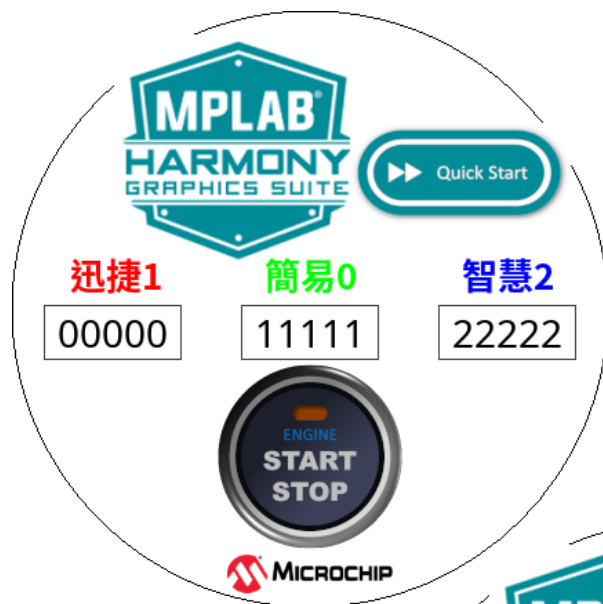
    return LE_SUCCESS; // success tells the decoder to keep going
}

void leApplication_MediaCloseRequest(leStream* stream)
{
}

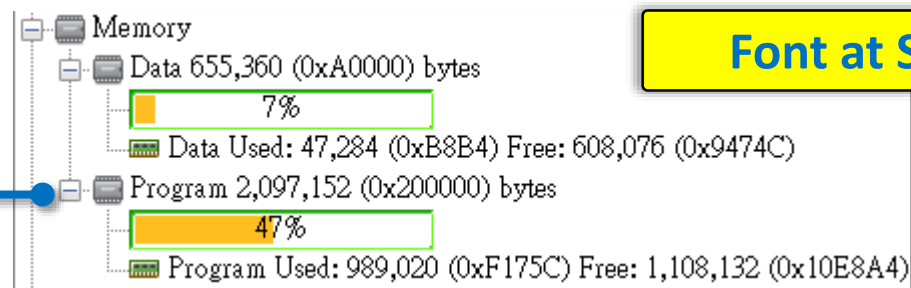
}
```

External Graphics Resources - Font

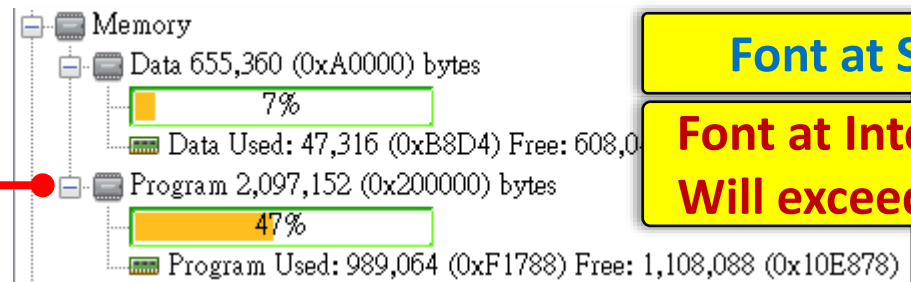
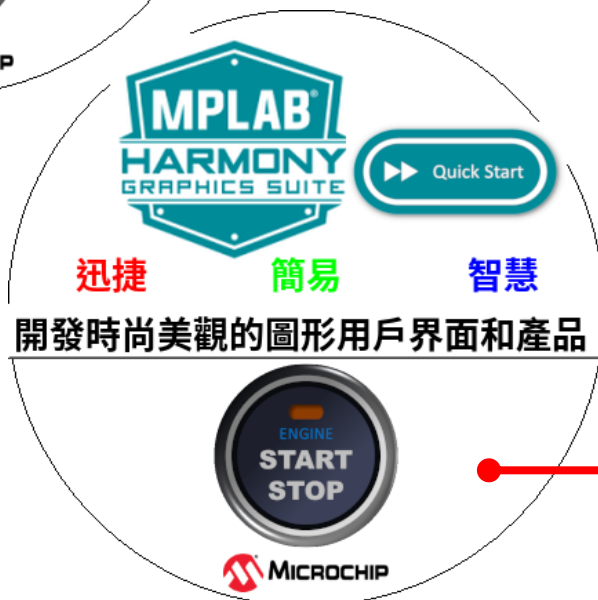
- Flash consume compare.



Font at Internal Flash



Font at SD CARD



Font at SD CARD

Font at Internal Flash
Will exceed flash size

Multi-Screen and Splash Screen

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Multi-Screen and Splash Screen

- Check “Screen and Scheme Pane” and select to “Screens”.

The screenshot displays the Legato Graphics Composer application window. The interface includes a menu bar (File, View, Project, Asset, Window, Help), a toolbar, and a central design canvas. The design canvas shows a circular interface with the MPLAB Harmony Graphics Suite logo, a 'Quick Start' button, and three digital displays labeled 'Fast.', 'Easy.', and 'Smart.' with values 00000, 11111, and 22222 respectively. A 'START STOP' button is at the bottom. The 'Screens' pane on the left lists various UI components like BackgroundPanel, boundaryCircle, mchpLogo, mhgsLogo, fastLabel, easyLabel, smartLabel, qsButton, startstopButton, fastValLabel, easyValLabel, and smartValLabel. The 'Object Editor' pane on the right shows properties for 'Screen0', including Name, Enabled, Persistent, and Events (On Show, On Hide, On Update). The 'Scheme Editor' pane is also visible. A console window at the bottom shows initialization logs. Red and yellow callouts with 'Click' text highlight the 'Screens' tab, the 'Screen0' entry in the Screens pane, and the 'Object Editor' tab.

Legato Graphics Composer :
File View Project Asset Window Help

Screen Tree

Screen0

Click

Screens

Visible Name View

Screen0

Object Editor

Screen

Name Screen0

Enabled ☒

Persistent ☐

Events

On Show ☐

On Hide ☐

On Update ☐

Object Editor Scheme Editor

Click

Console

(09:29:15) : Initializing Legato Graphics Composer v1.25.2b...
(09:29:15) : Initialization complete.
(09:29:15) : Welcome to Legato Graphics Composer!
(09:29:19) : Opening project: C:/Exercises_PIC3001GFX/
Lab1_BSP_Template_Legato/firmware/src/config/default/MyProject.zip

Screens Schemes

Click

Multi-Screen and Splash Screen

- Manage Screens and check “On Show” event of Screen0.

The screenshot displays the Legato Graphics Composer interface with several key components and annotations:

- Screens Panel:** Located on the left, it contains a table for managing screens. Annotations include:
 - New Screen:** Points to the green plus icon.
 - Delete All:** Points to the red X icon.
 - Set Screen as Primary:** Points to the checkmark icon.
 - Delete Screen:** Points to the red X icon in the table.
 - Duplicate Screen:** Points to the copy icon.
 - Set Screen Active:** Points to the exclamation mark icon.
- Screens Table:**

Visible	Name	View
☒	Screen0	[View Icon]
- Main Canvas:** Displays a circular splash screen for the MPLAB Harmony Graphics Suite. It includes a "Quick Start" button, a "FAST STOP" button, and three digital displays showing "Fast. 00000", "Easy. 11111", and "Smart. 22222".
- Object Editor:** A panel on the right showing properties for the selected "Screen" object. It includes:
 - Name:** Screen0
 - Enabled:** ☒
 - Persistent:** ☐
 - Events:**
 - On Show:** ☒ (Annotated with "Check On Show Event" and "Check")
 - On Hide:** ☐
 - On Update:** ☐

- Console:** A panel at the bottom showing system logs, including initialization and project opening messages.

Multi-Screen and Splash Screen

- Add New Screen1 and check “On Show” and “On Update” event.

The screenshot displays the Legato Graphics Composer interface with several key components highlighted to demonstrate the steps for adding and configuring a new screen:

- Screens Panel:** Located on the left, it contains a table with columns 'Visible', 'Name', and 'View'. A red box highlights the '+' icon in the top-left corner, with a yellow 'Click' label and an arrow pointing to it. Below this, the table lists 'Screen0' and 'Screen1'. 'Screen1' is selected, indicated by a red box around its row.
- Screen Tree:** Located in the top center, it shows a hierarchy of screens. 'Screen1' is highlighted with a red box and a yellow 'Click' label with an arrow pointing to it.
- Object Editor:** Located on the right, it shows the configuration for the selected screen. A red box highlights the 'Screen' section, which includes the name 'Screen1' (highlighted with a red box), 'Enabled' (checked), and 'Persistent' (unchecked). Below this, the 'Events' section is highlighted with a red box, showing 'On Show' (checked) and 'On Update' (checked). Yellow callouts with arrows point to these checked boxes, labeled 'Check On Show Event' and 'Check On Update Event'. At the bottom of the Object Editor, a yellow 'Check' label with an arrow points to the 'Scheme Editor' tab.
- Console:** Located at the bottom, it shows the output of the application, including initialization messages and project opening status.

Multi-Screen and Splash Screen

- Set Screen1 as Primary for the Start Up Screen.

The screenshot displays the Legato Graphics Composer interface with several panels and annotations. The 'Screens' panel on the left lists 'Screen0' and 'Screen1', both with the 'Visible' checkbox checked. A red box highlights the 'Screen1' row, with a yellow callout 'Click' pointing to the 'Visible' checkbox and another yellow callout 'Click' pointing to the 'Screen1' text. A blue callout 'Underline on Screen1' points to the 'Screen1' text. A red box highlights the 'Set Screen As Primary' button at the top of the 'Screens' panel. The 'Object Editor' panel on the right shows the configuration for 'Screen1', with the 'On Show' event checked. The 'Console' panel at the bottom shows the initialization process.

Screens Panel:

Visible	Name	View
☒	Screen0	
☒	Screen1	

Object Editor Panel (Screen1):

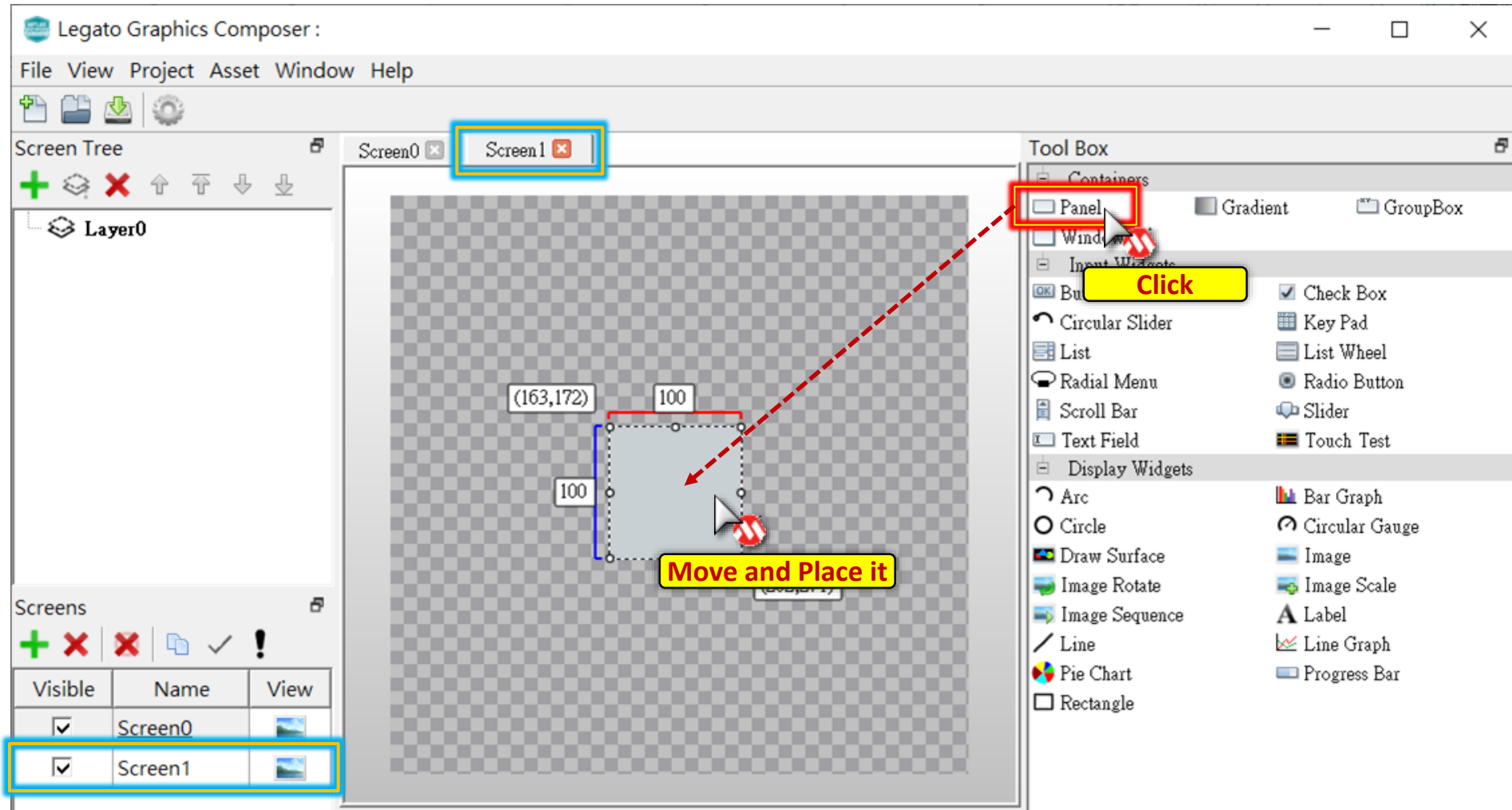
Screen	
Name	Screen1
Enabled	☒
Persistent	☐
Events	
On Show	☒
On Hide	☐
On Update	☒

Console Panel:

```
(09:29:15) : Initializing Legato Graphics Composer v1.25.2b...
(09:29:15) : Initialization complete.
(09:29:15) : Welcome to Legato Graphics Composer!
(09:29:19) : Opening project: C:/Exercises_PIC3001GFX/
Lab1_BSP_Template_Legato/firmware/src/config/default/MyProject.zip
```

Multi-Screen and Splash Screen

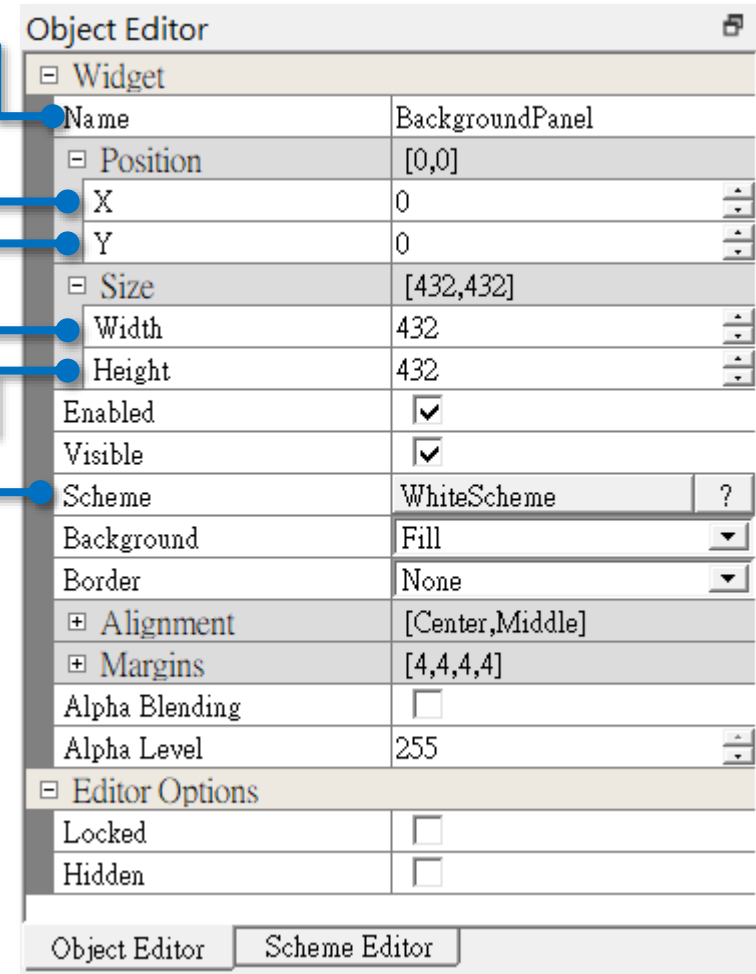
- Add background of Screen1 by using “Panel” of Containers.



Multi-Screen and Splash Screen

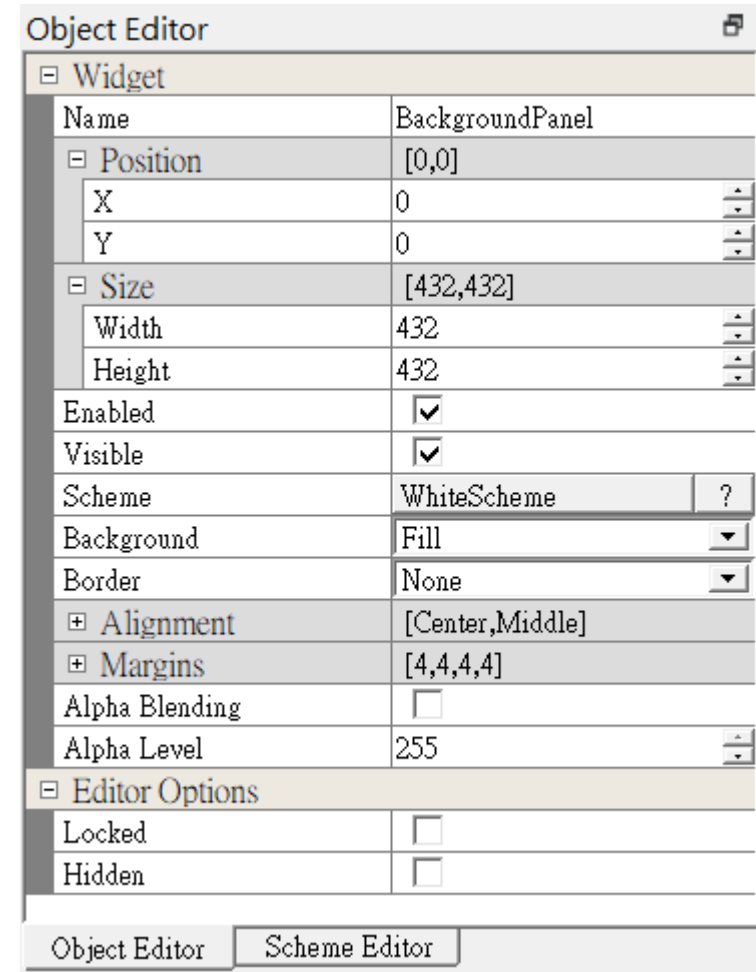
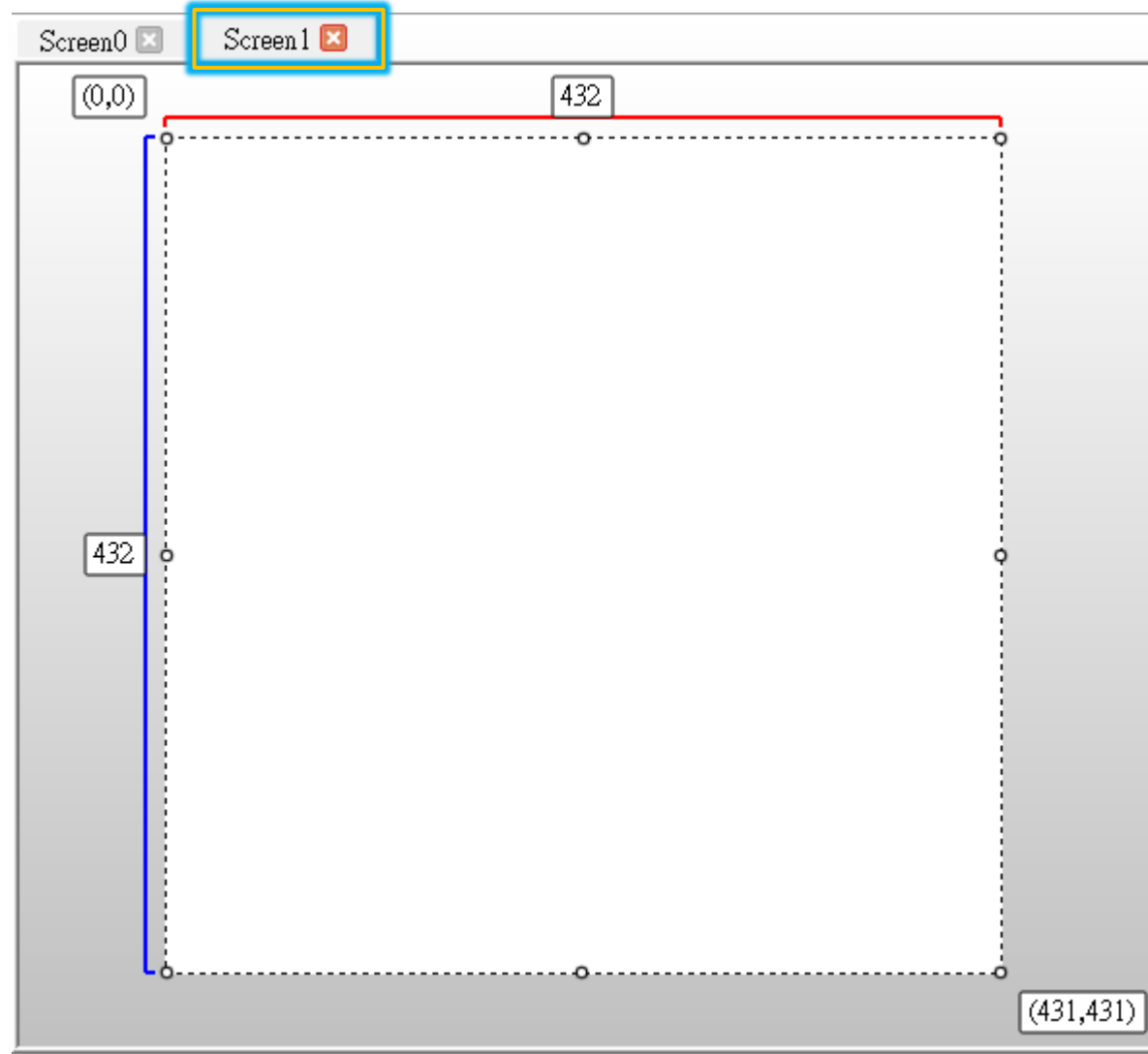
- **Configure New Container Panel as below.**

- Name : **BackgroundPanel**
- Position X : **0**
- Position Y : **0**
- Size Width : **432**
- Size Height : **432**
- Scheme : *WhiteScheme*



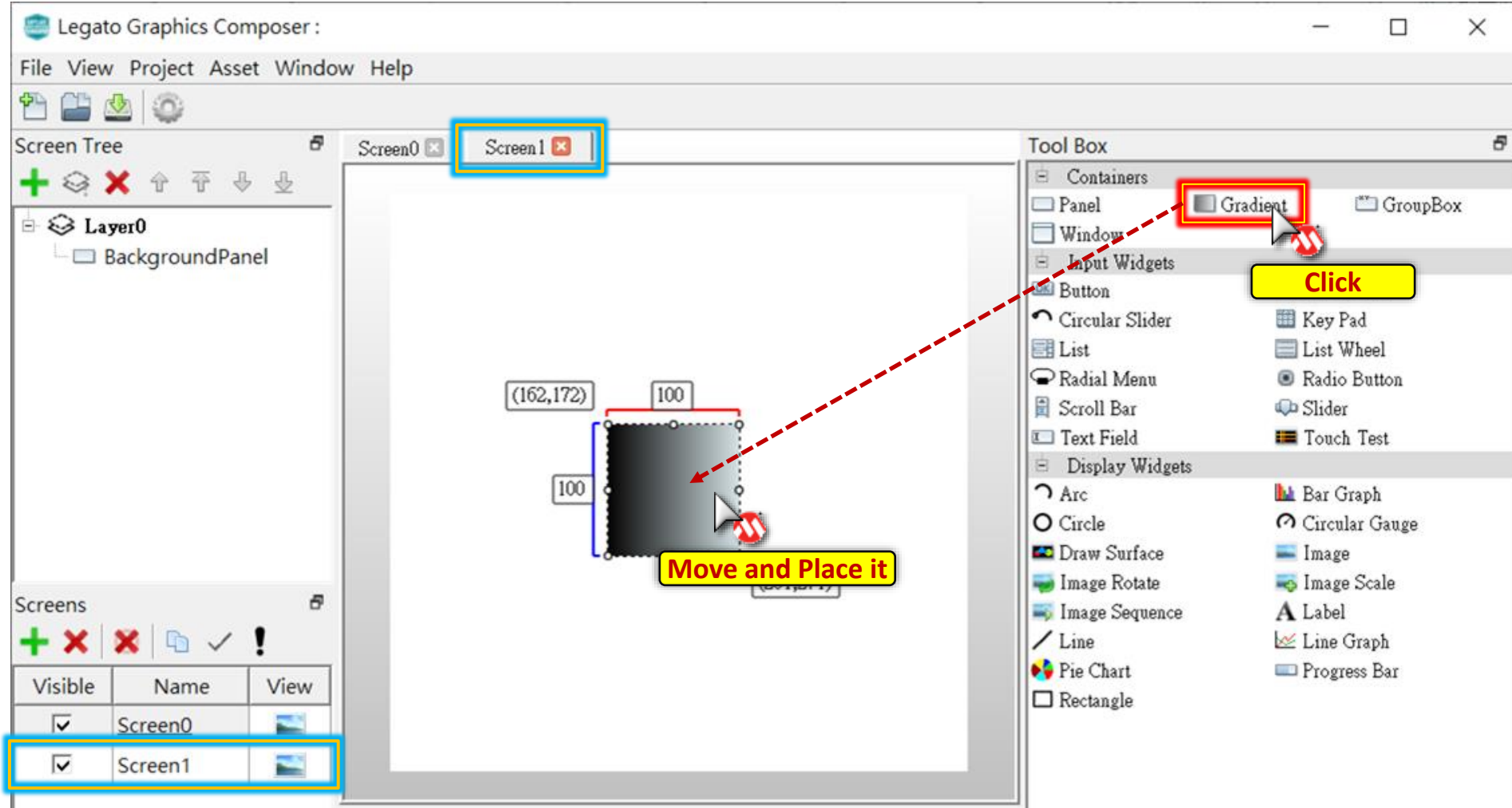
Multi-Screen and Splash Screen

- Container Panel after configured.



Multi-Screen and Splash Screen

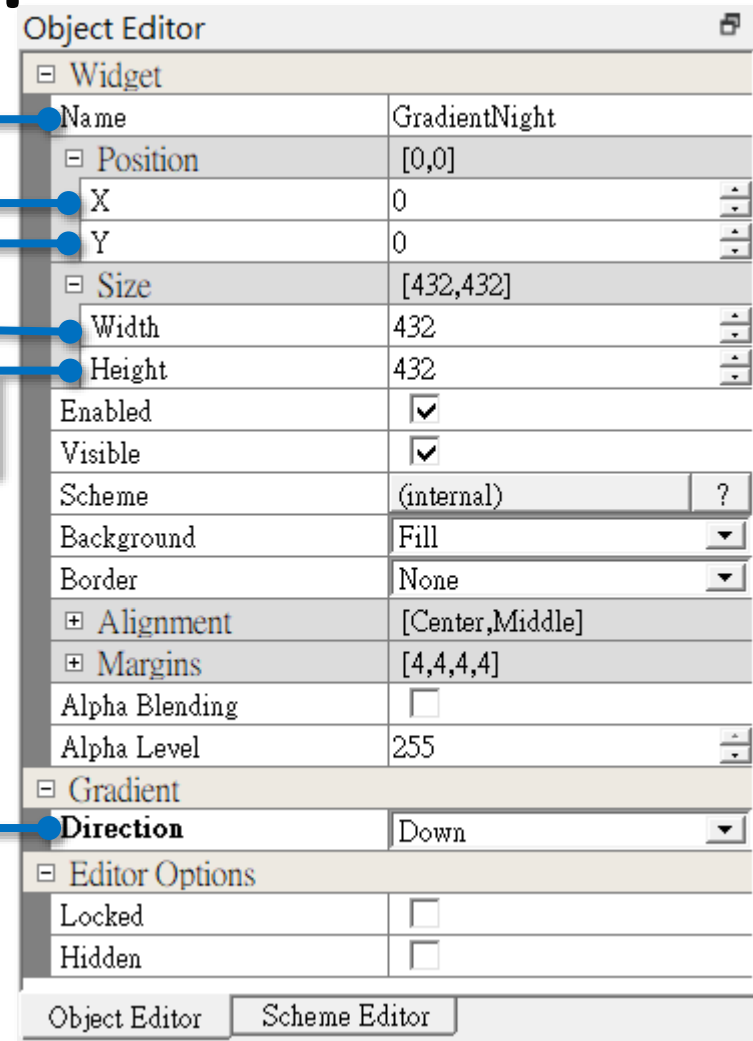
- Add a Gradient Background of Containers.



Multi-Screen and Splash Screen

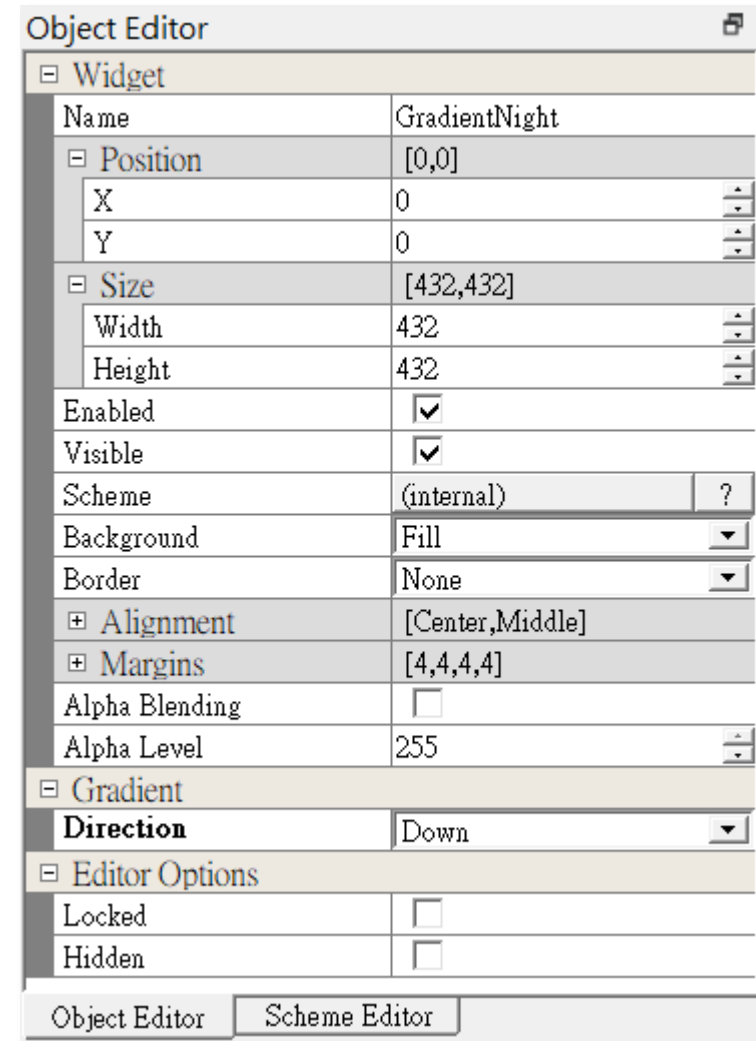
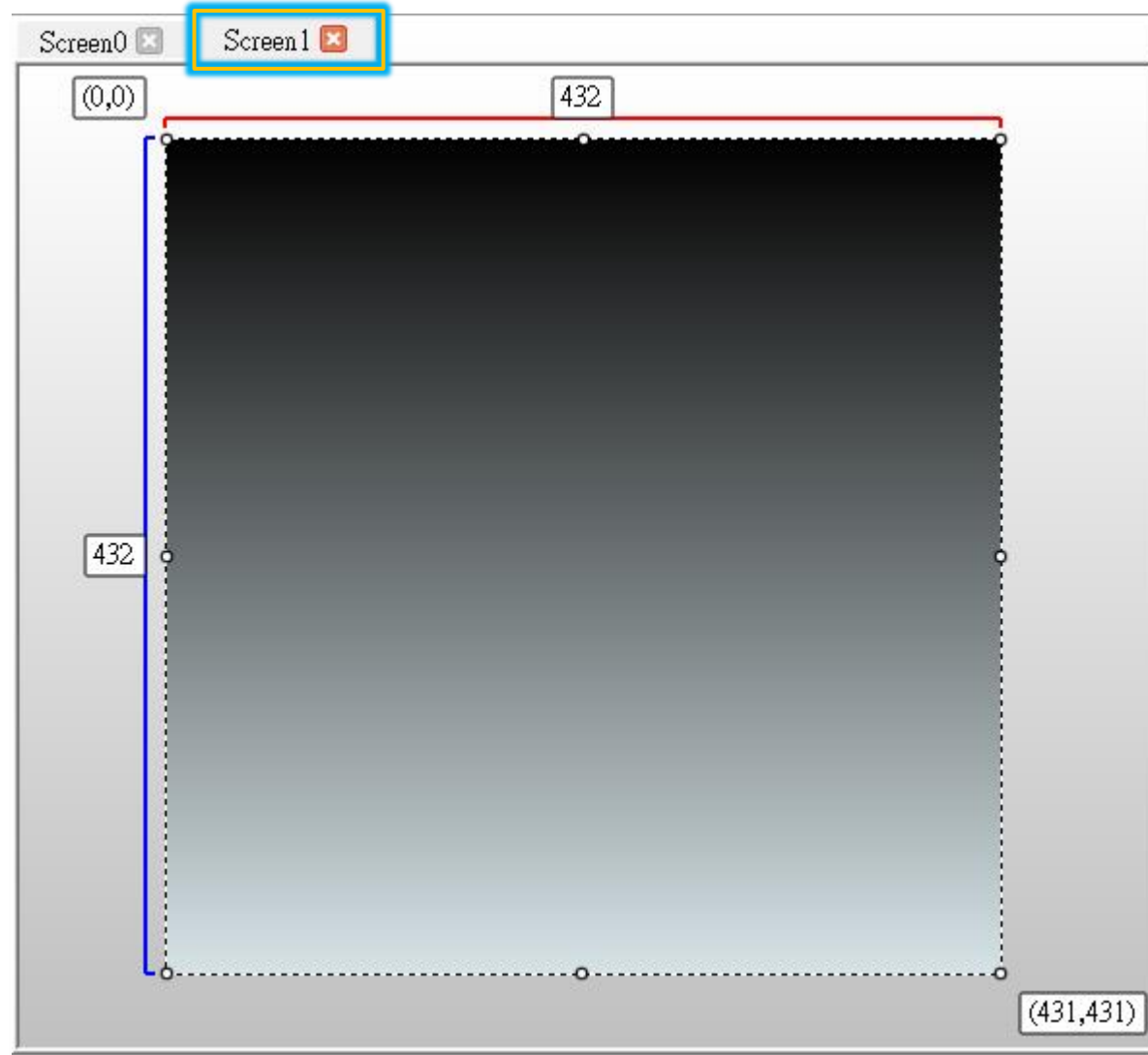
- **Configure Gradient Container as below.**

- Name : **GradientNight**
- Position X : **0**
- Position Y : **0**
- Size Width : **432**
- Size Height : **432**
- Direction : **Down**



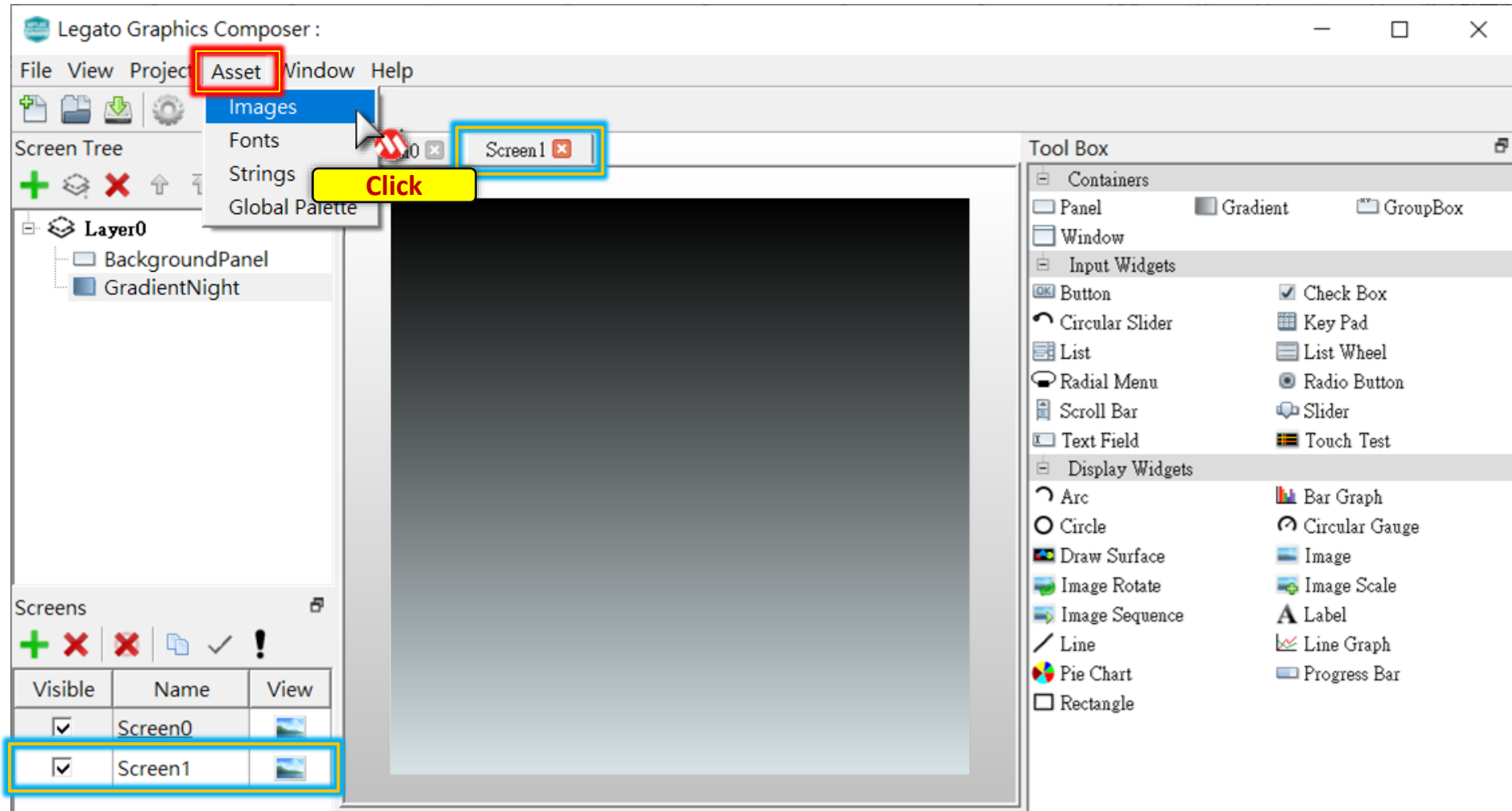
Multi-Screen and Splash Screen

- Gradient Container after configured.



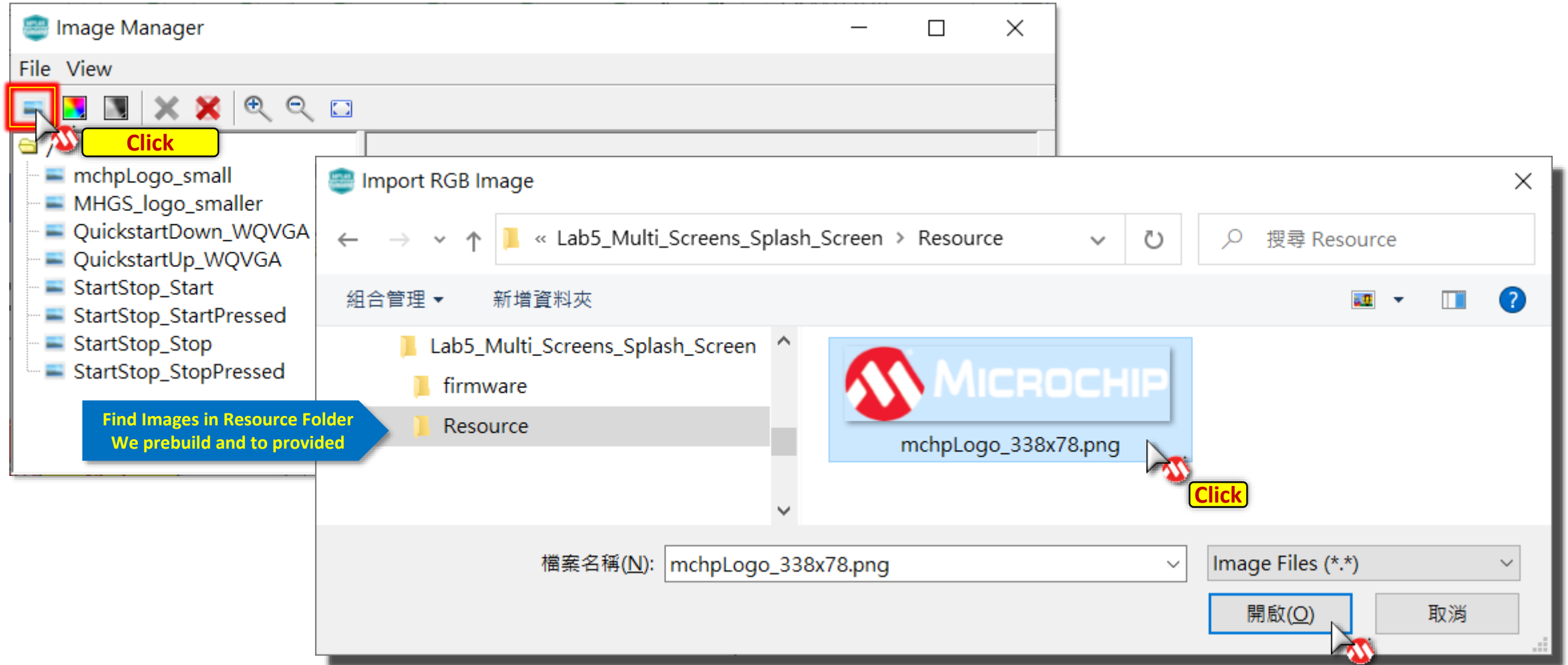
Multi-Screen and Splash Screen

- Add new Logo Image through the Asset “Image Manager”.



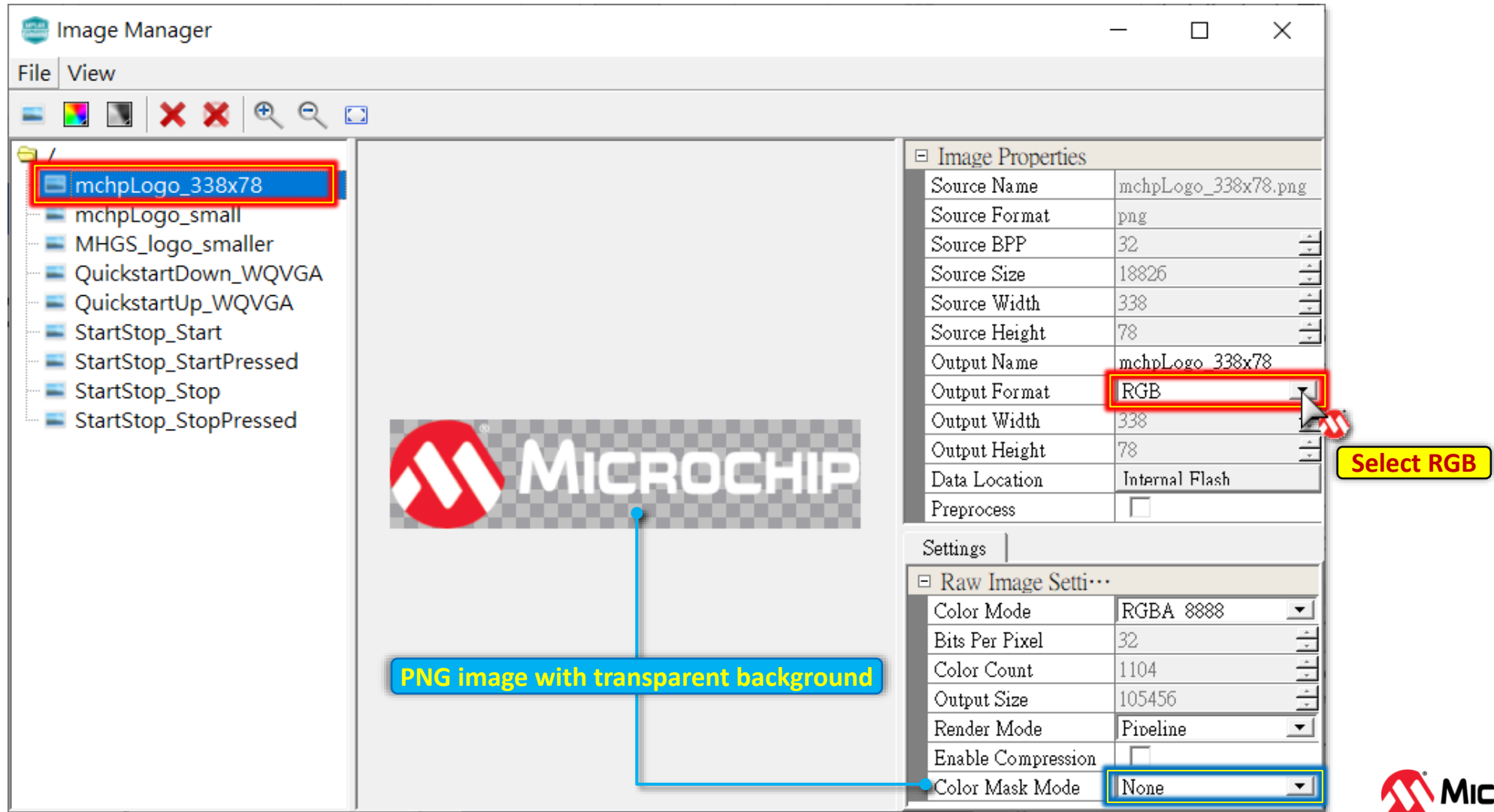
Multi-Screen and Splash Screen

- Add **mchpLogo_338x78.png** in “Image Manager”.



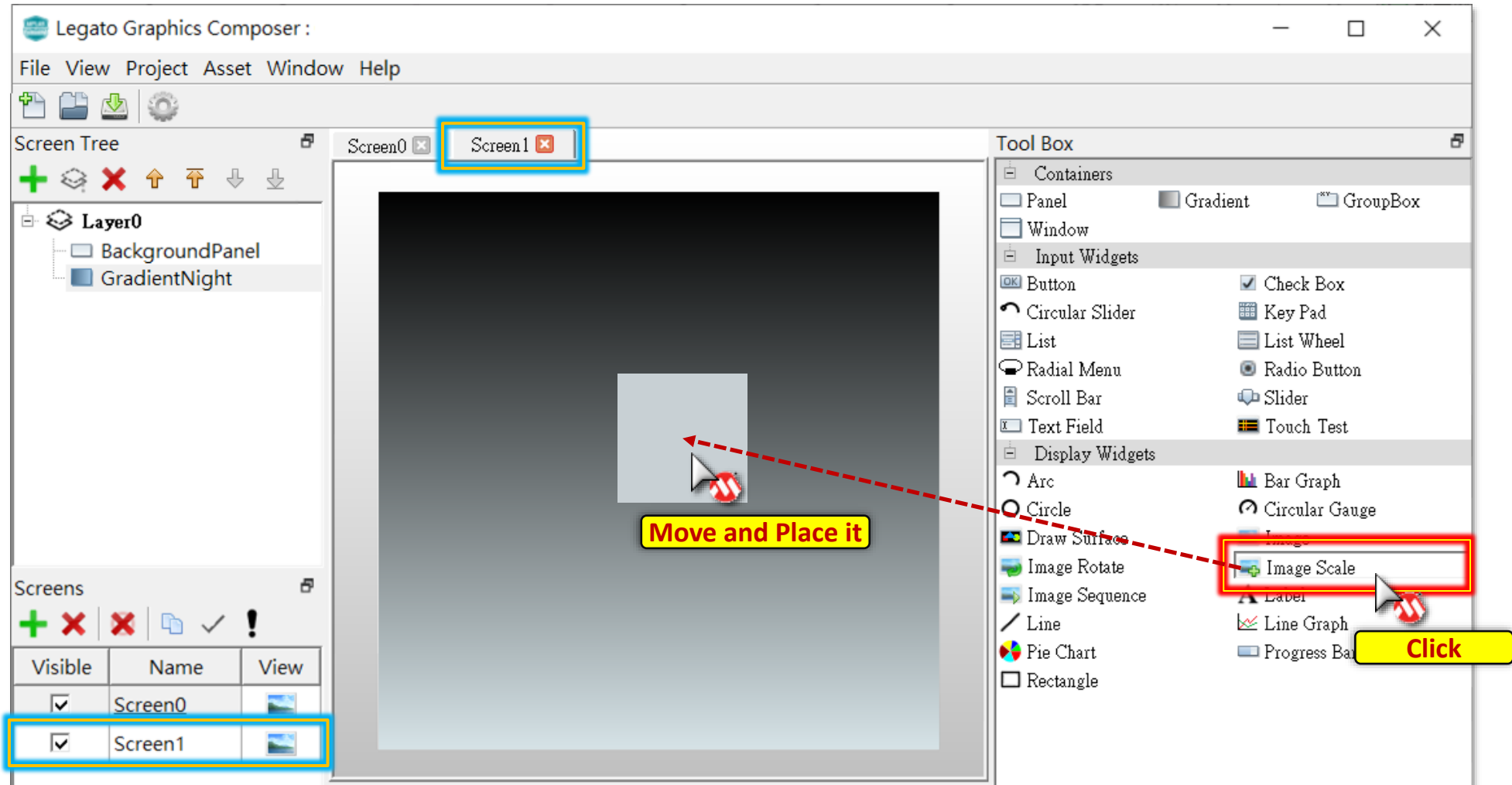
Multi-Screen and Splash Screen

- Configure Output Format to **RGB** of **mchpLogo_338x78** image.



Multi-Screen and Splash Screen

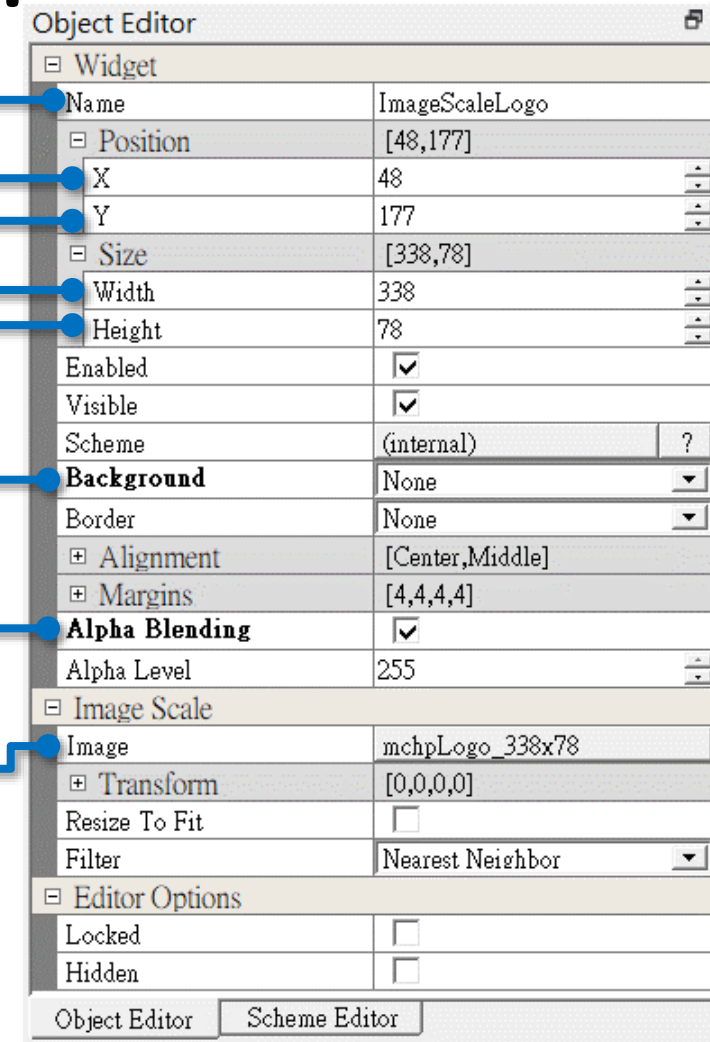
- Click [Image Scale] of Display Widgets to add an Image widget.



Multi-Screen and Splash Screen

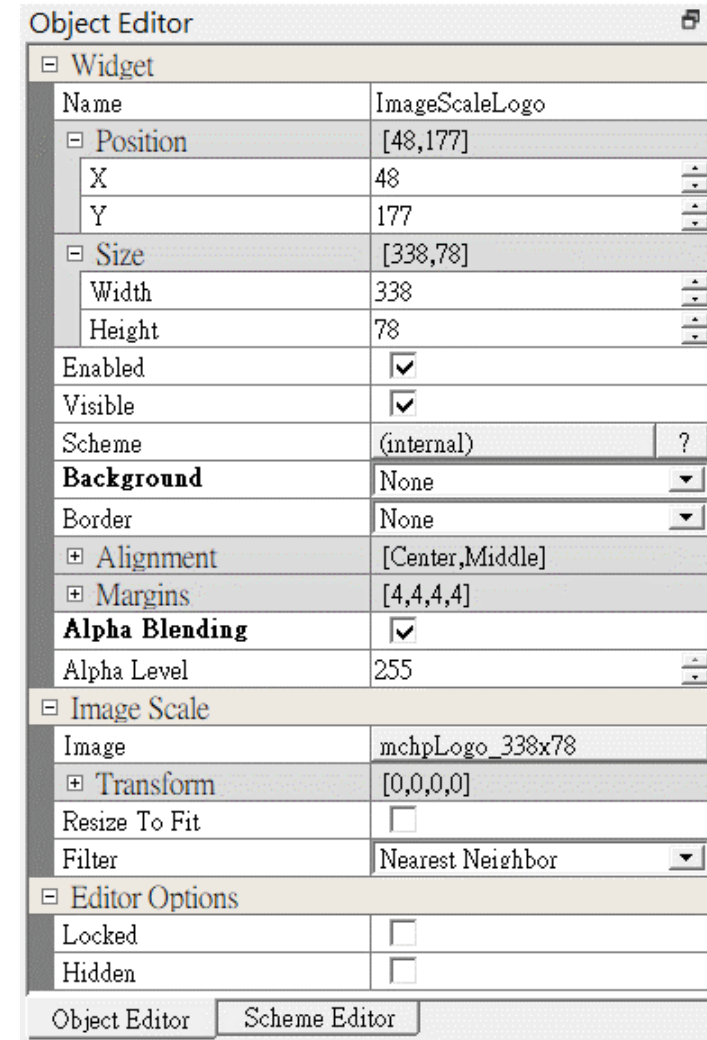
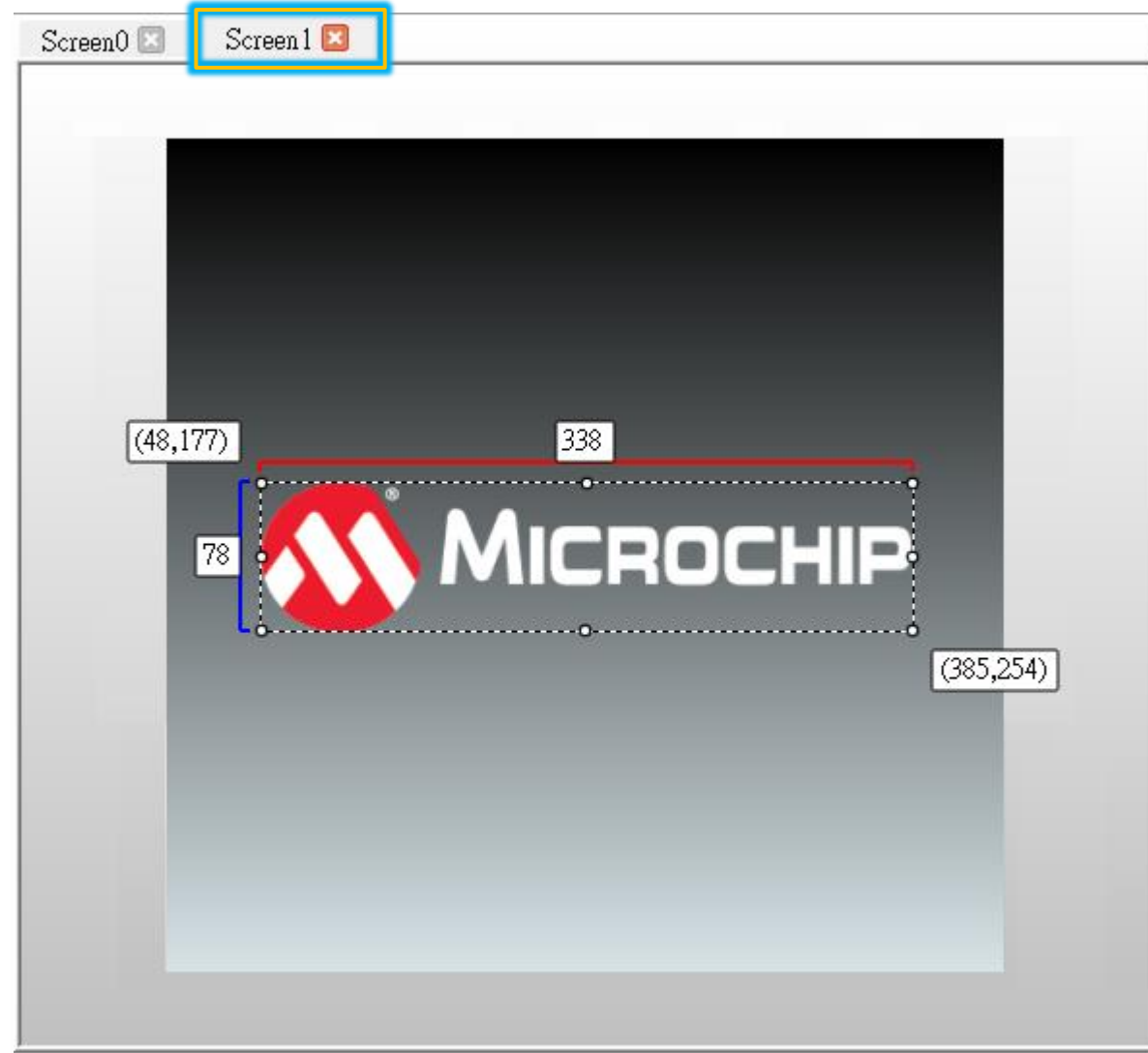
- Configure **Image Scale** Widget as below.

- Name : **ImageScaleLogo**
- Position X : **48**
- Position Y : **177**
- Size Width : **338**
- Size Height : **78**
- Background : **None**
- Alpha Blending : ☒
- Image : **mchpLogo_338x78**



Multi-Screen and Splash Screen

- **Image Scale** Widget after configurated.



Multi-Screen and Splash Screen

- The **lelsDrawing()** API

- Whether your application is running in single-loop Harmony or in multiple threaded RTOS configuration, initiating changes to the GUI while the previous changes are still being rendered can lead to undesirable to catastrophic outcome.
- A catastrophic one would be initiating a screen-change mid-render. During a screen change, memory allocated to widgets are freed to setup a new screen. If the rendering pipeline is still trying to access these widgets, it will often crash the application.
- The best way to prevent this is to safeguard application code from initiating any change to the GUI by checking if all rendering activities have completed.
- In the case of using an RTOS, the best practice is to treat the GUI state like it is running in another thread and lelsDrawing() is the mutex to use. The state should not be altered while lelsDrawing() is returning LE_LTRUE.

Multi-Screen and Splash Screen

- Generate Code and Implement **Splash animation Interval Timer**.

```
...
leFixedString smartValString;
```

```
#define SCREEN_W 432
```

Screen Width and Height

```
#define SCREEN_H 432
```

```
#define LOGO_W 338
```

Logo Width and Height

```
#define LOGO_H 78
```

```
#define SCALE_STEP 8
```

Scale Up Step

```
#define MOVE_STEP 15
```

Move Left Step

```
#define ALPHA_STEP 10
```

Alpha Diming Step

```
volatile bool SplashUpdateFlag = false;
```

Splash Update Flag

```
void SplashUpdateIntervalCallback( uintptr_t context )
```

Splash Update Interval Timer Callback function

```
{
```

```
    SplashUpdateFlag = true; // Set the SplashUpdateFlag for scale up animation in Screen1 OnUpdate() event
```

```
}
```

```
void TimerCallback ( uintptr_t context ) { ... }
```

```
int main ( void )
```

```
{
```

```
    SYS_Initialize ( NULL );
```

```
    SYS_TIME_CallbackRegisterMS( TimerCallback, 0, 200, SYS_TIME_PERIODIC );
```

```
    SYS_TIME_CallbackRegisterMS( SplashUpdateIntervalCallback, 0, 20, SYS_TIME_PERIODIC );
```

Splash Update Interval Timer Callback in 20ms period

```
    while ( true ) { ... }
```

Multi-Screen and Splash Screen

- Implement **Image Scale Up** Function.

```
...  
void ScaleUpIntervalCallback( uintptr_t context ) { ... }
```

```
bool ScaleUpLogo( uint32_t Init_W )
```

Image Scale Up Function

```
{  
    uint32_t W, H;
```

```
    if( Init_W )
```

```
    {  
        W = Init_W;
```

Set initial Image Width

```
    }
```

```
    else
```

```
    {  
        W = Screen1_ImageScaleLogo->fn->getWidth( Screen1_ImageScaleLogo );  
        W = W + SCALE_STEP;
```

Get current Image Width

Increase Image Width by given SCALE_STEP

```
    }
```

```
    if( W >= LOGO_W ) { return true; }
```

Scale Up finish when Image Size has exceeded Screen Width

```
    H = W * LOGO_H/LOGO_W;
```

```
    Screen1_ImageScaleLogo->fn->setPosition( Screen1_ImageScaleLogo, ( SCREEN_W - W )>>1, ( SCREEN_H - H )>>1 );
```

Keep ratio of image

Centering Scaled Image Position

```
    Screen1_ImageScaleLogo->fn->setSize( Screen1_ImageScaleLogo, W, H );
```

Update Scaled Image Size

```
    RGB_LED_G_Toggle();
```

Toggle Green LED

```
    return false;
```

```
}
```

Multi-Screen and Splash Screen

- Implement **Image Fade Out Moving Function**.

```
...  
void ScaleUpIntervalCallback( uintptr_t context ) { ... }  
bool ScaleUpLogo( uint32_t Init_W ) { ... }
```

```
bool FadeOutMoveLogo( void ) Image Fade Out Move Function
```

```
{
```

```
    int32_t X, Y;  
    uint32_t Alpha;
```

```
    X = Screen1_ImageScaleLogo->fn->getX( Screen1_ImageScaleLogo );
```

```
    X = X - MOVE_STEP;
```

```
    Alpha = Screen1_ImageScaleLogo->fn->getAlphaAmount( Screen1_ImageScaleLogo );
```

```
    Alpha = Alpha - ALPHA_STEP;
```

```
    if( X <= -LOGO_W )
```

```
    {
```

```
        return true; Moving finish while Image has move out of SCREEN
```

```
    }
```

```
    Y = Screen1_ImageScaleLogo->fn->getY( Screen1_ImageScaleLogo );
```

```
    Screen1_ImageScaleLogo->fn->setPosition( Screen1_ImageScaleLogo, X, Y );
```

```
    Screen1_ImageScaleLogo->fn->setAlphaAmount( Screen1_ImageScaleLogo, Alpha );
```

```
    RGB_LED_G_Toggle(); Toggle Green LED
```

```
    return false;
```

```
}
```

Get current Image X Position

Decrease X by given MOVE_STEP to move out to left

Get current Image Alpha amount

Decrease Alpha amount by given ALPHA_STEP

Get current Image Y Position

Update New Position of Image

Update New Alpha amount of Image

Multi-Screen and Splash Screen

- Implement **Screen0** and **Screen1** Event Functions.

```
...  
void ScaleUpIntervalCallback( uintptr_t context ) { ... }  
bool ScaleUpLogo( uint32_t Init_W ) { ... }  
bool FadeOutMoveLogo( void ) { ... }
```

```
void Screen1_OnShow(void)
```

```
{  
    ScaleUpLogo( 8 );  
    RGB_LED_G_On(); RGB_LED_B_Off();  
}
```

Image Scale Up process (Initialize), Set initial Width to 8

Turn on Green LED while Screen1 OnShow()

```
void Screen1_OnUpdate(void)
```

```
{  
    if( !IsDrawing() == LE_FALSE && SplashUpdateFlag )  
    {  
        SplashUpdateFlag = false;  
        if( ScaleUpLogo( 0 ) ) {  
            if( FadeOutMoveLogo() ) {  
                legato_showScreen( screenID_Screen0 );  
            }  
        }  
    }  
}
```

Update Splash Image in Screen1 OnUpdate()

Checking Drawing completion and Update animation by Interval Timer

Image Scale Up process (Update)

Image Fade Out Moving process

Animation finished, Switch screen to Screen0

```
void Screen0_OnShow(void)
```

```
{  
    RGB_LED_G_Off(); RGB_LED_B_On();  
}
```

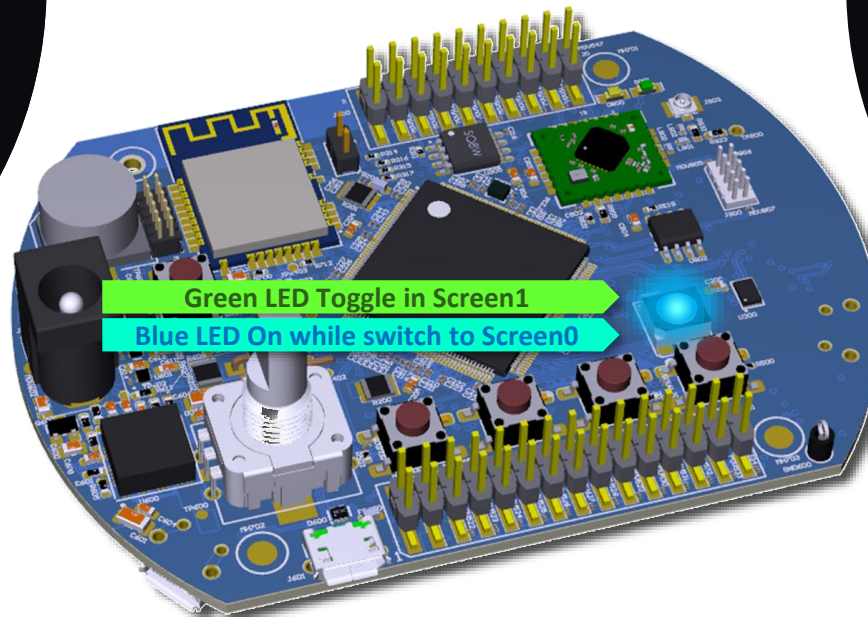
Turn on Blue LED while Screen0 OnShow()

Multi-Screen and Splash Screen

- Logo Scale Up and Fade Out Moving left in Screen1 then switch to Screen0.



Screen1



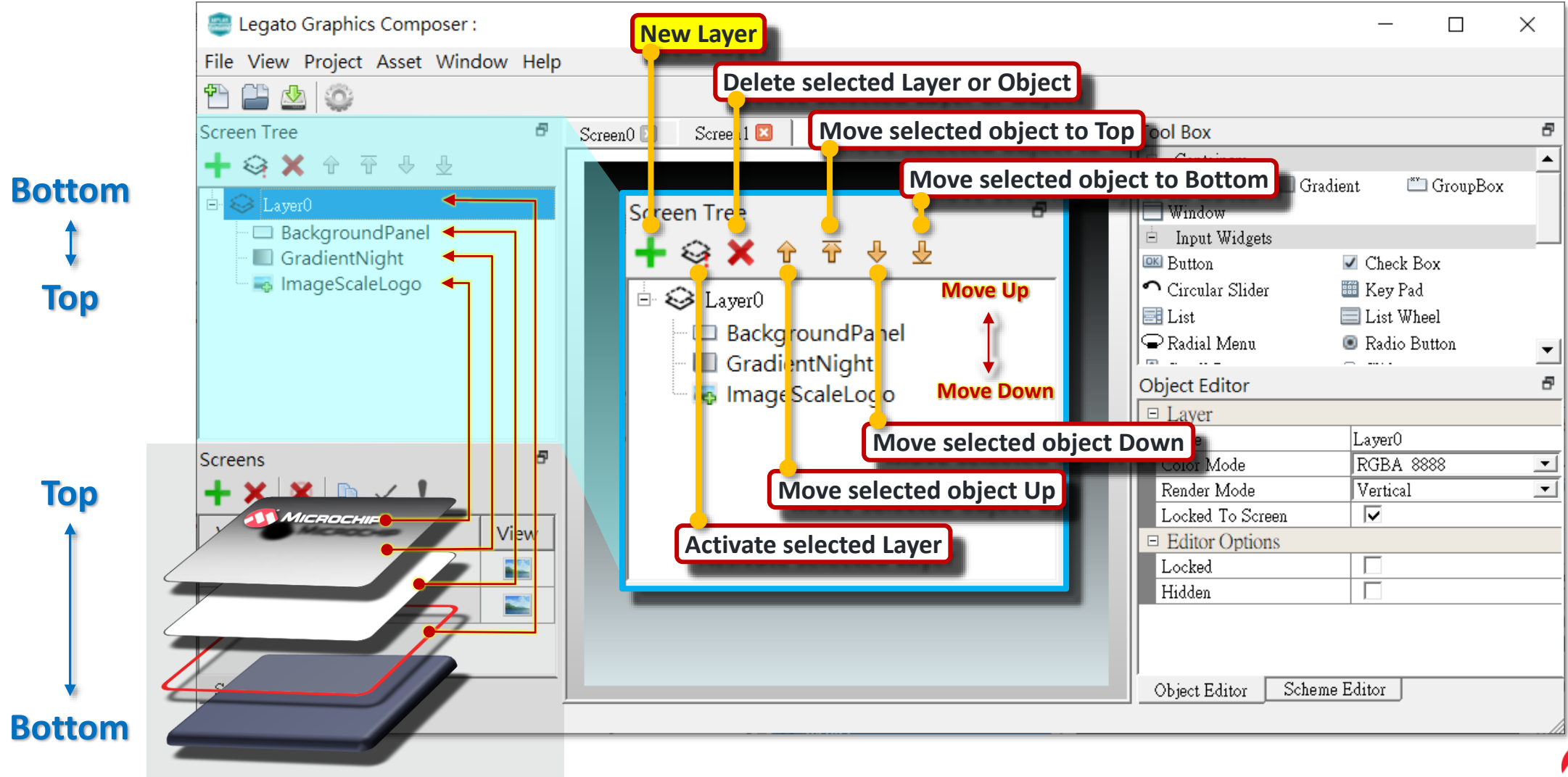
Screen0

Multi-Layering

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

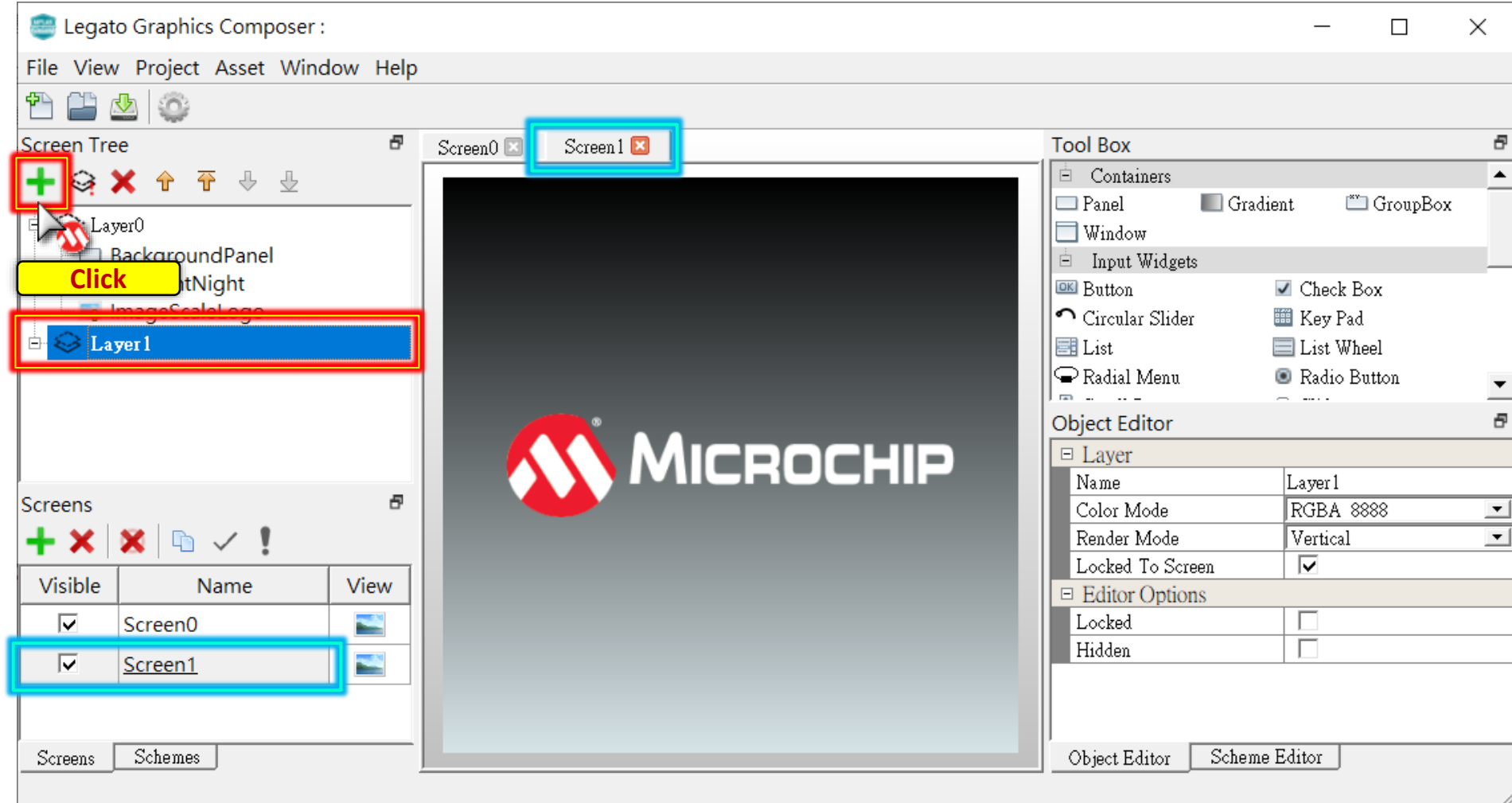
Multi-Layering

- Layer operation and Objects sequence arrange.



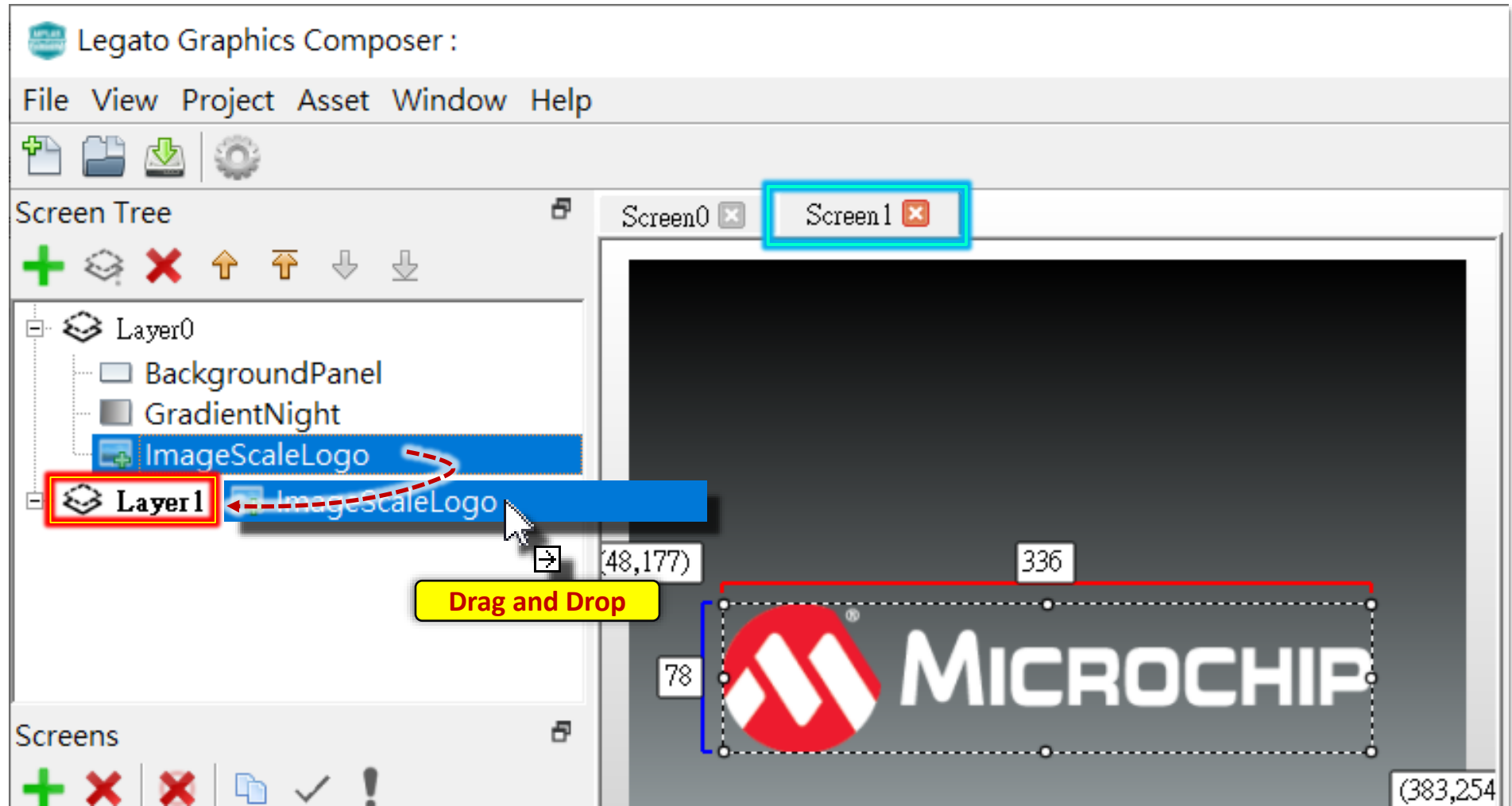
Multi-Layering

- Add **Layer1** of **Screen1**.



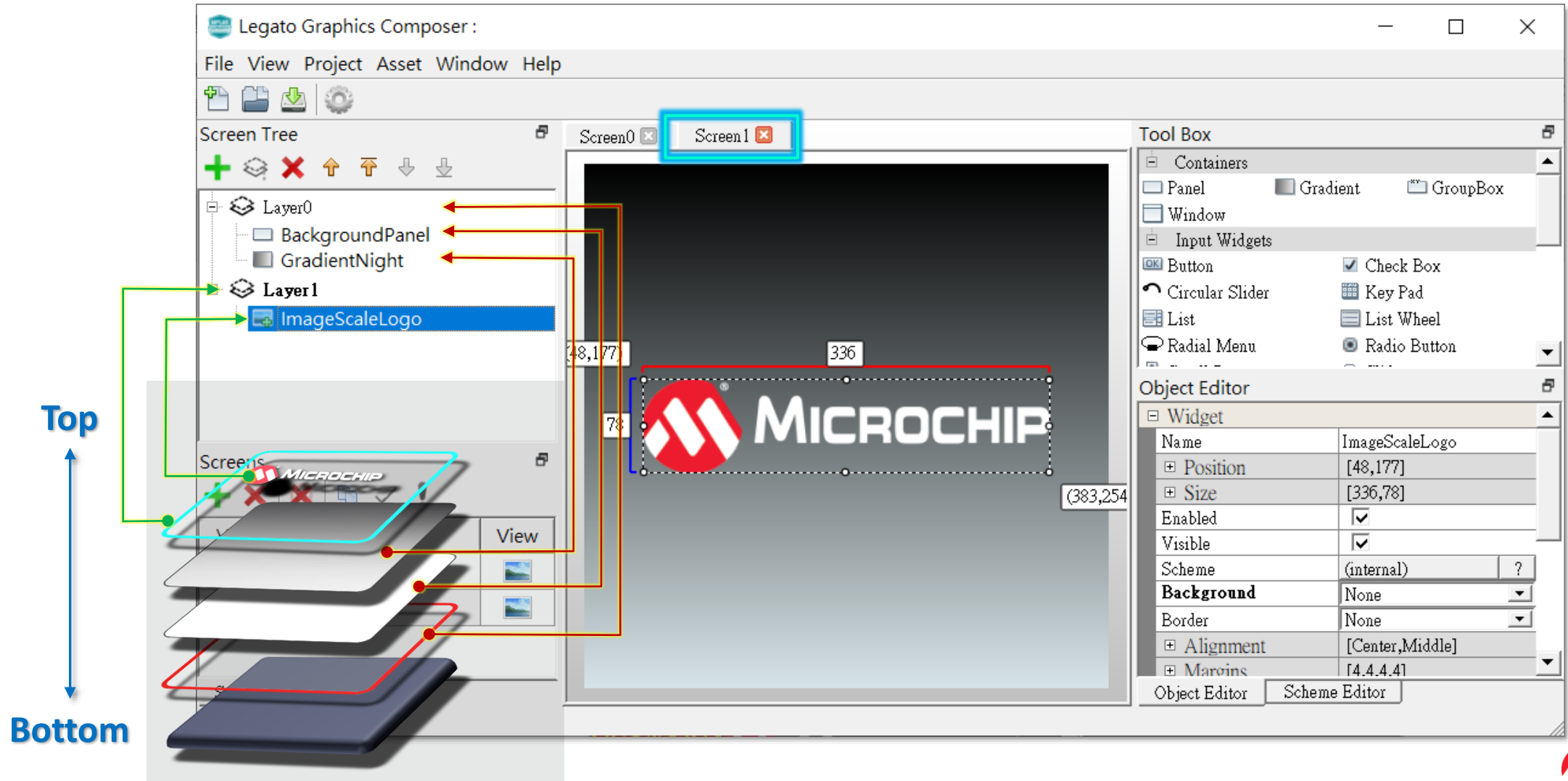
Multi-Layering

- Move **ImageScaleLogo** widget to **Layer1** of **Screen1**.



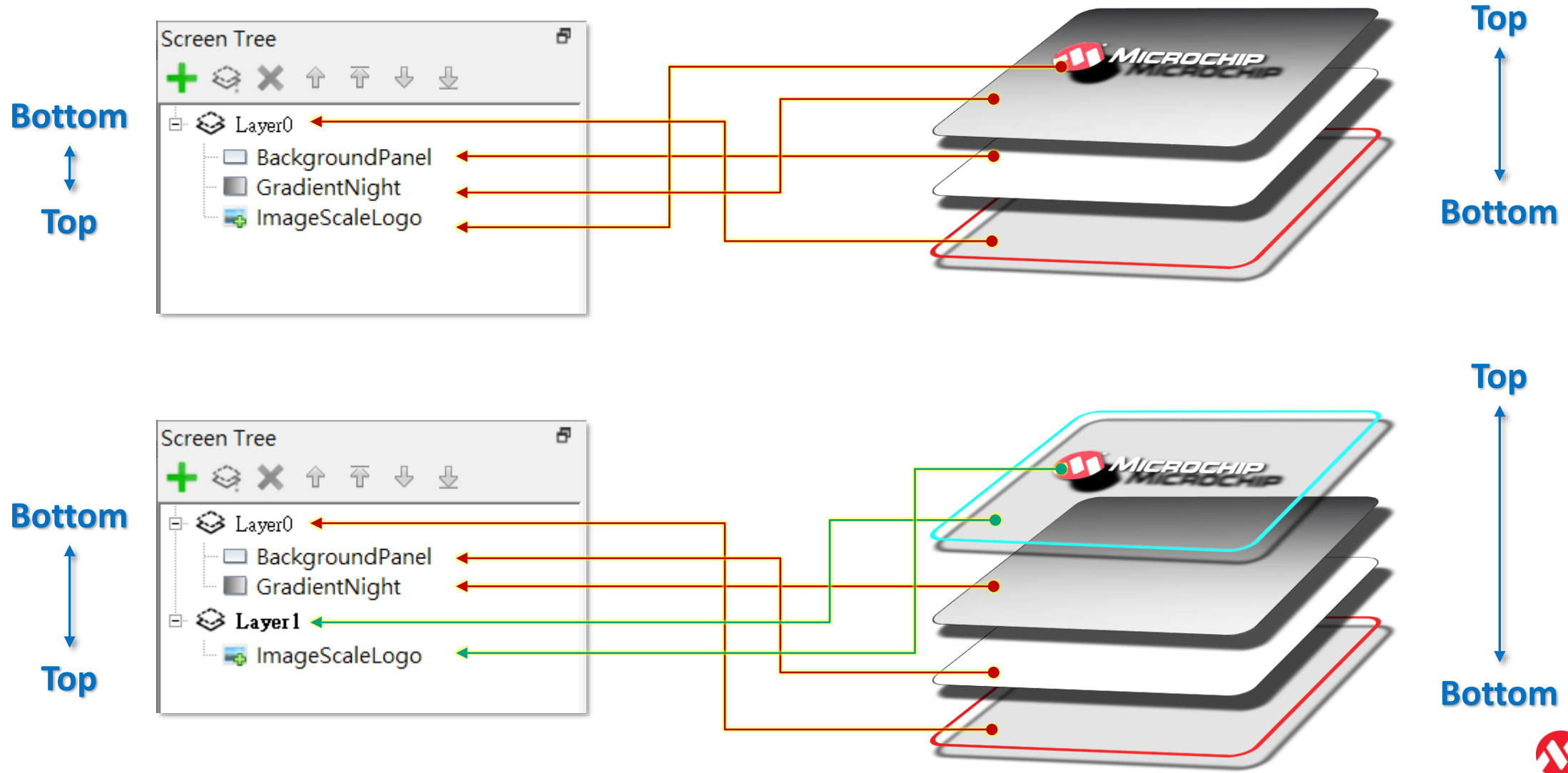
Multi-Layering

- After move **ImageScaleLogo** widget to **Layer1** of **Screen1**.



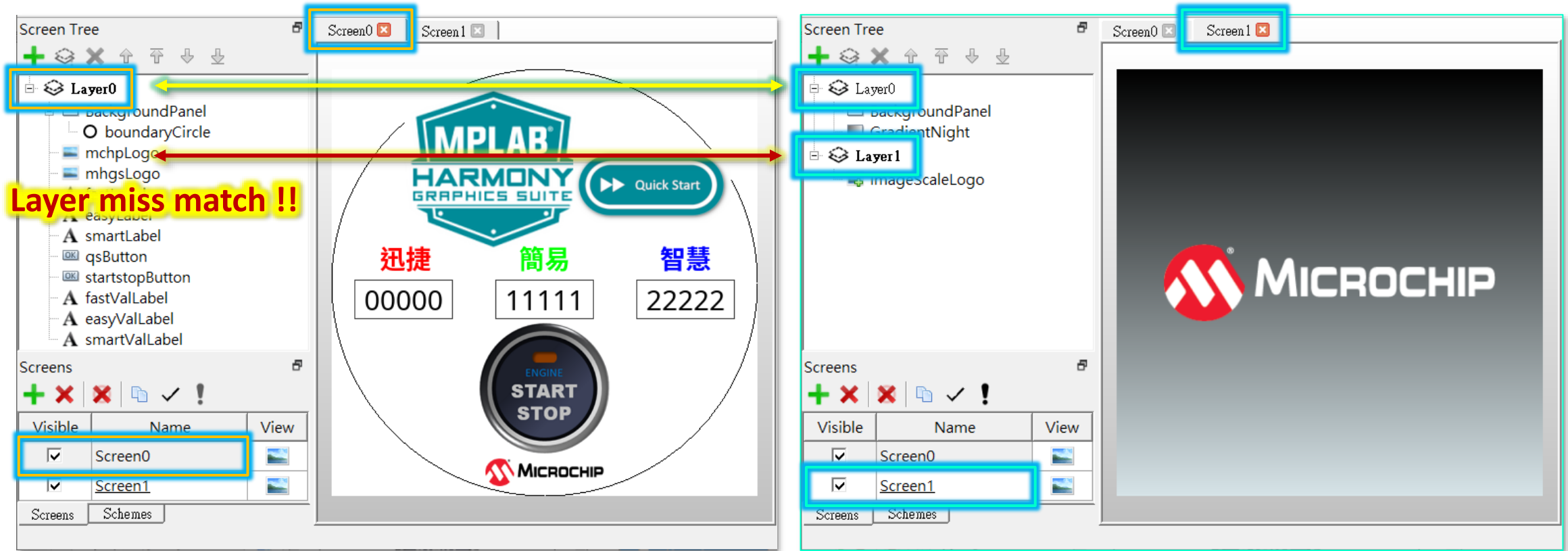
Multi-Layering

- Single Layer and Multi-Layers.



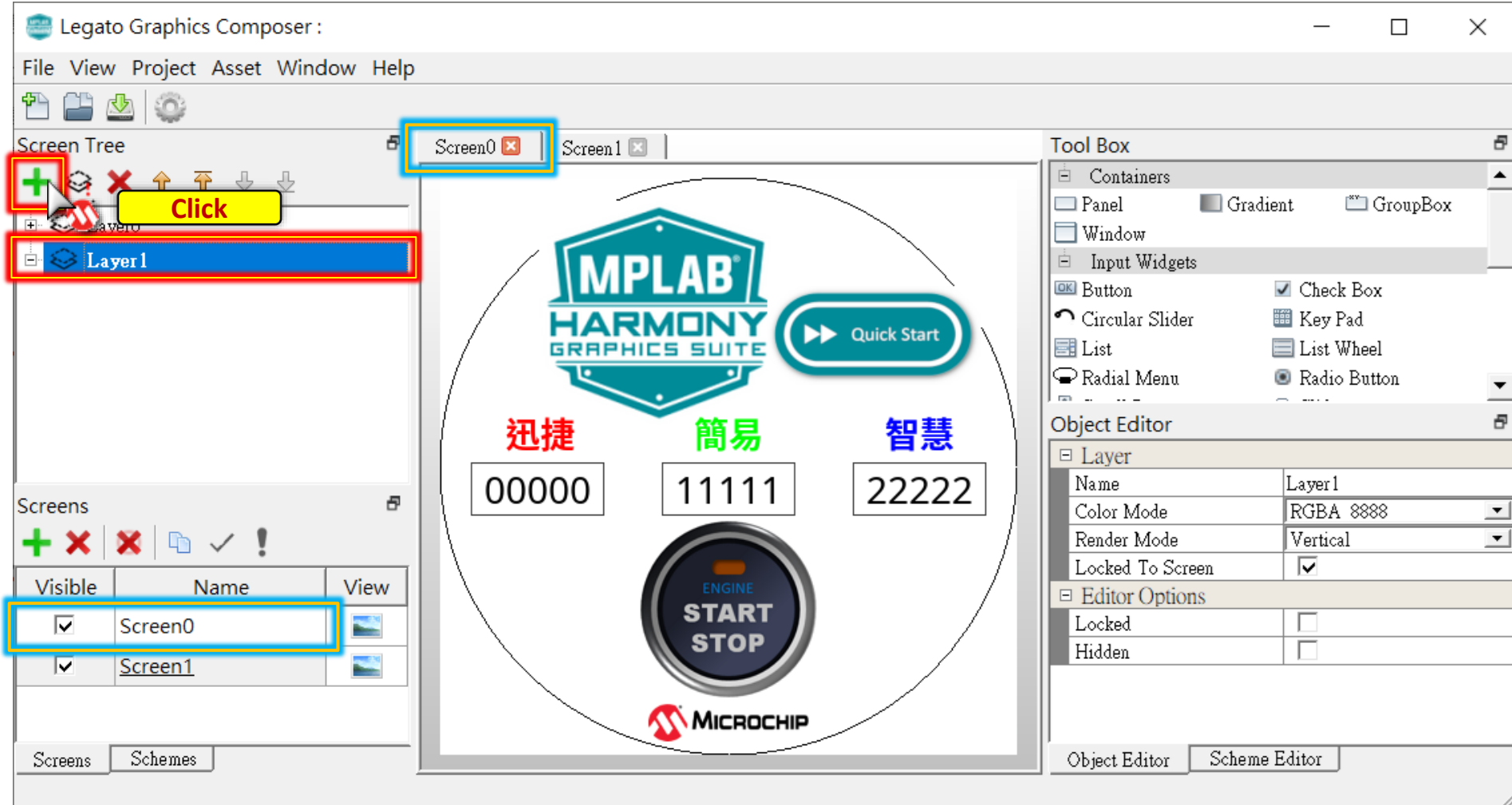
Multi-Layering

- Important !! Let all Screens to have the same number of Layers.



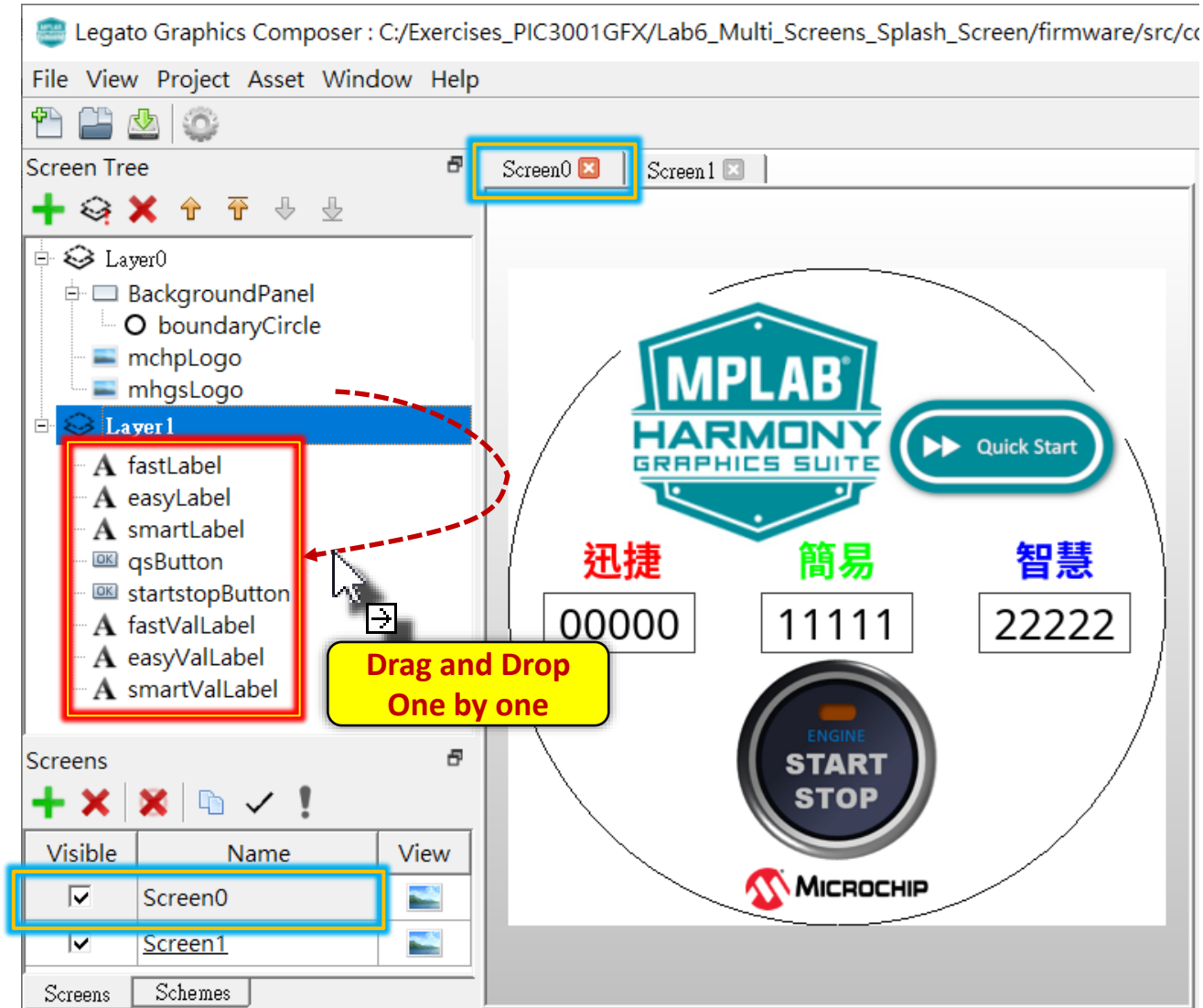
Multi-Layering

- Add **Layer1** in **Screen0**.



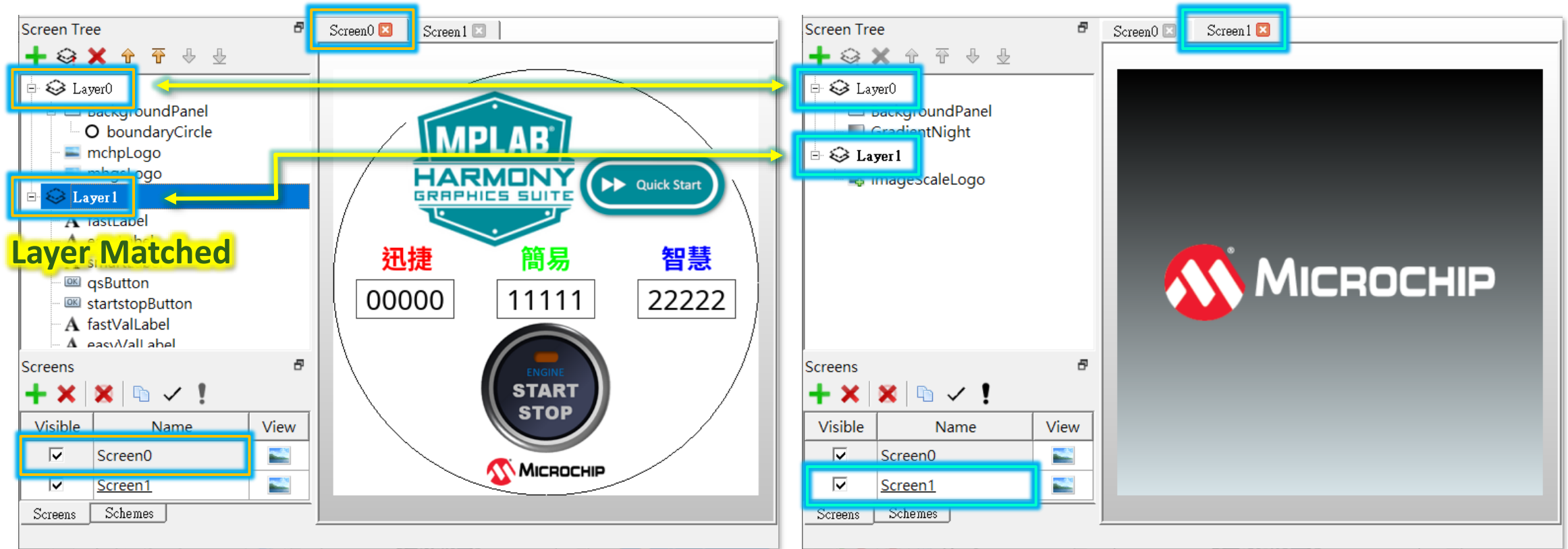
Multi-Layering

- Move **Widgets** to **Layer1** except the BackgroundPanel and Logo.



Multi-Layering

- All Screens now to have the same number of Layers.



Multi-Layering

- Generate code and programming to board to check the changes.

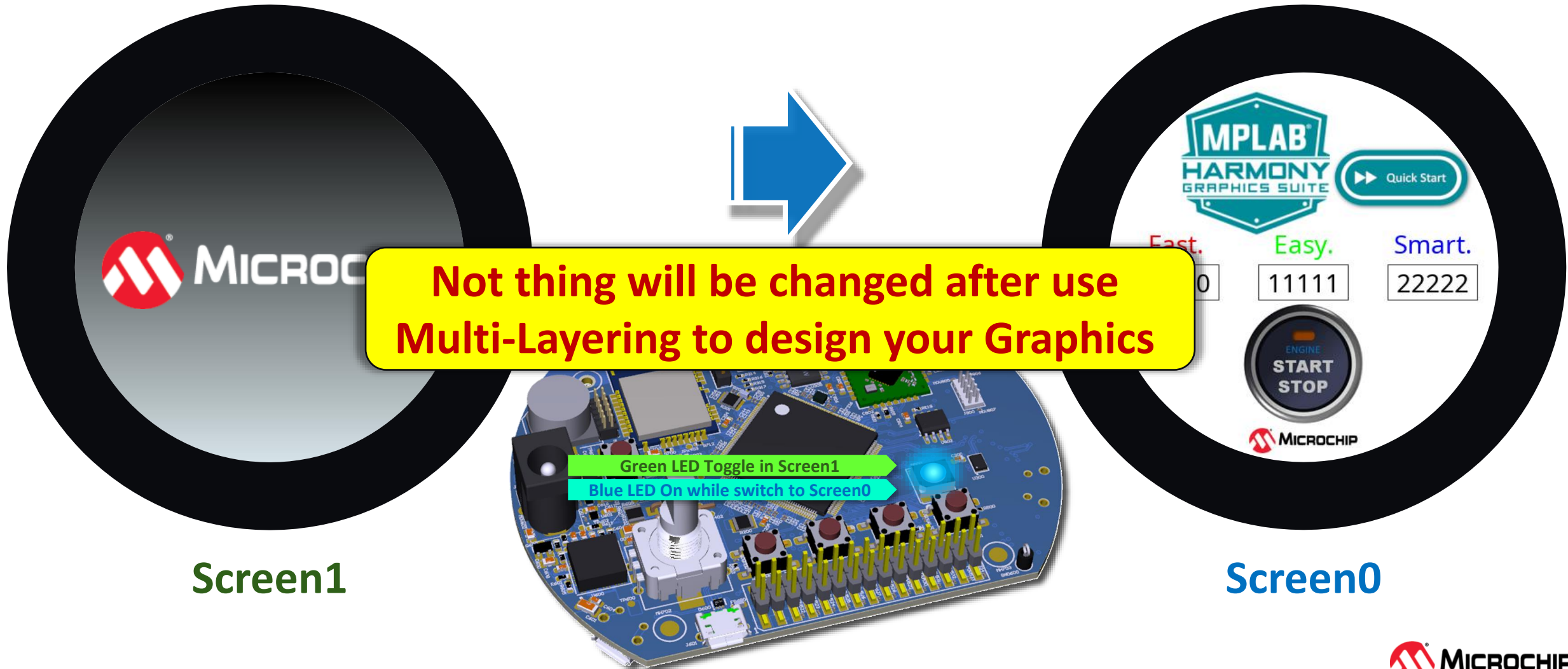


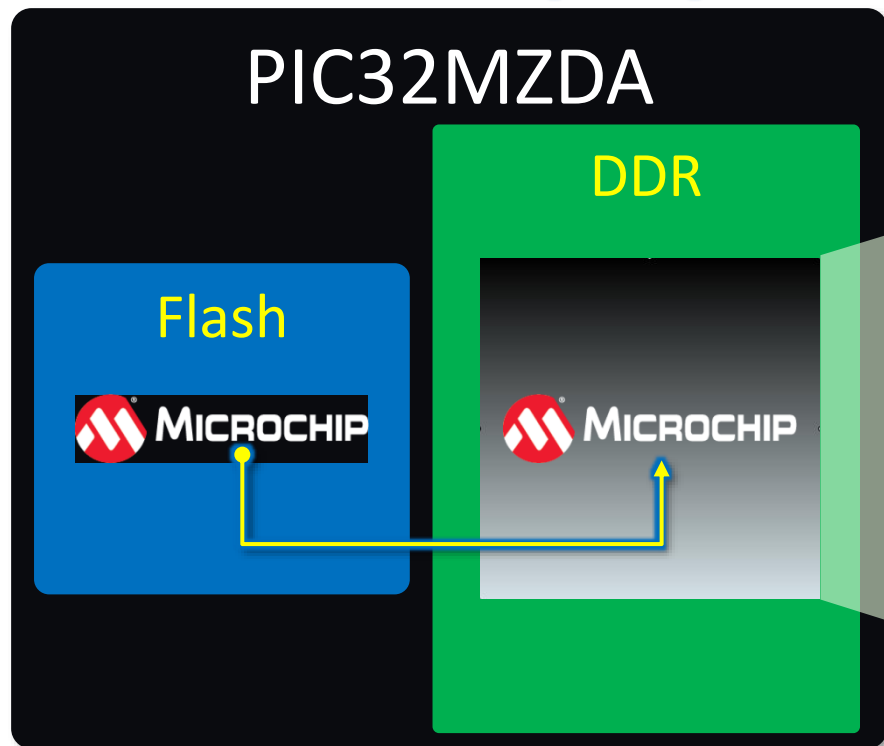
Image Preprocess

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Image Preprocess

- For fasten image processing, you could copy the Images from **internal Flash** to embedded **DDR2 SDRAM** before use it, we called the “**Preprocessing**”.

Flash to DDR (Slow)



DDR to DDR Bit BLIT (Quick)

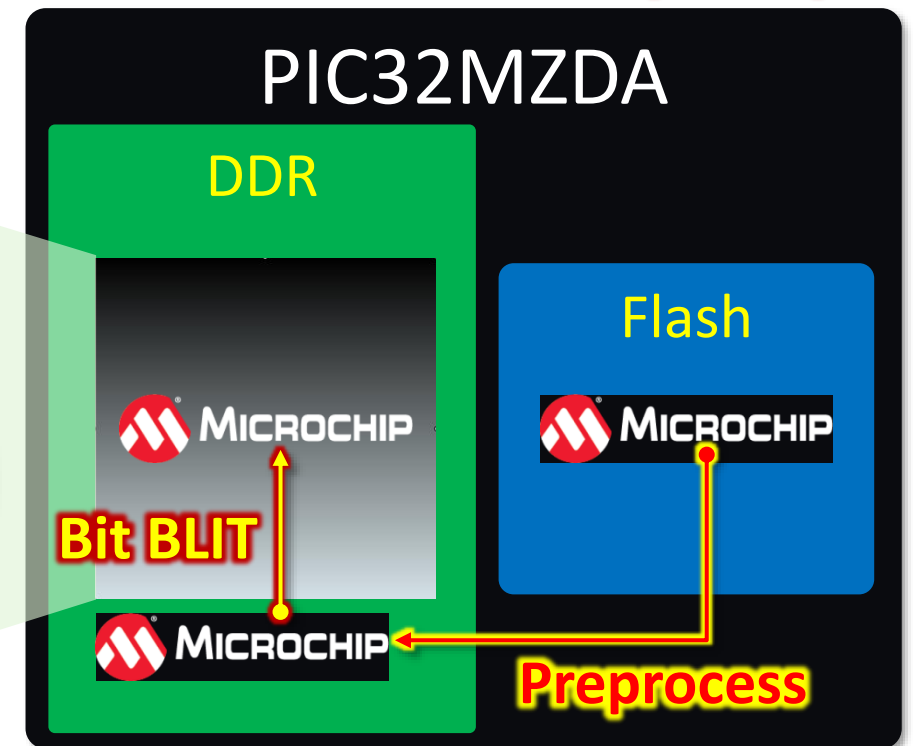


Image Preprocess

- Generally, PIC32MZDA embedded DDR2 SDRAM has not been using by application but **dedicate for Graphic frame buffer** or for **Graphics Objects preprocessing**.
- PIC32MZDA has **three physical hardware frame layers**, all layers need a dedicate DDR space for frame rendering.
- Double DDR space consumed if **Double Buffering** feature has enabled.



A8000000

AA000000

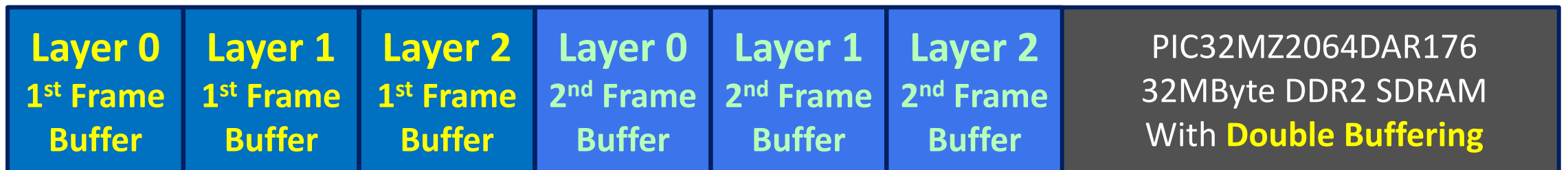
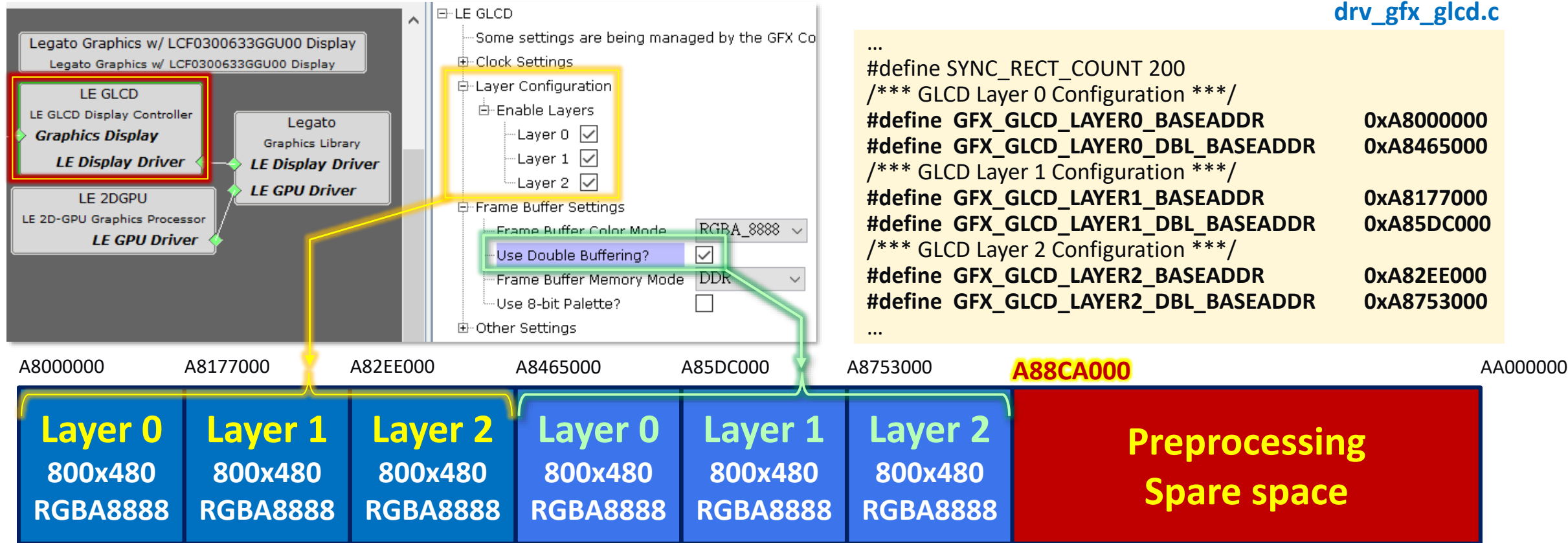


Image Preprocess

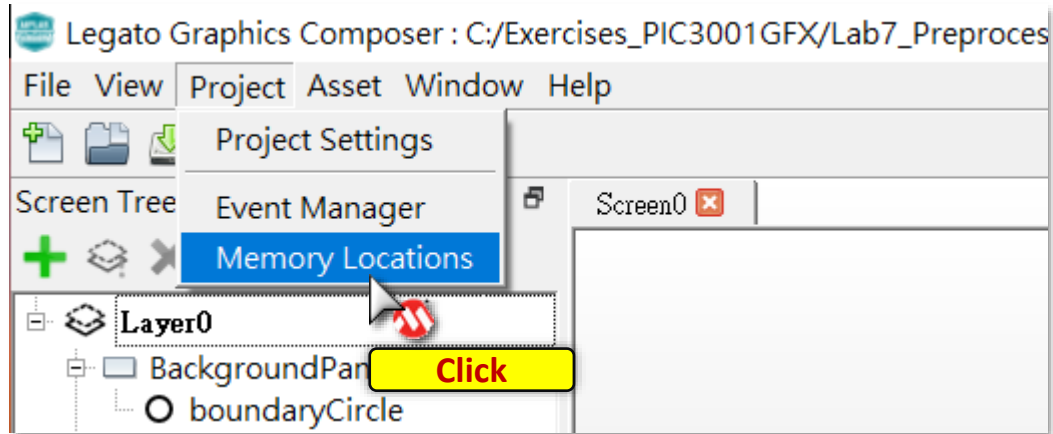
- Harmony could decide how many hardware layers to use in project and given a pre-definition **800x480** with **RGBA8888** frame buffer addressing.
- The **Preprocessing Images** should put after dedicate Layers Frame Buffer.



Frame Buffer Size is $800 \times 480 \times 4 = 0x177000$

Image Preprocess

- Reserve DDR Image Preprocessing Space in Memory Locations.



Name	DDR_PreProcess
Capacity (bytes)	131072 (128 KB)
Access Mode	Write
Address	A88CA000
Alignment	4

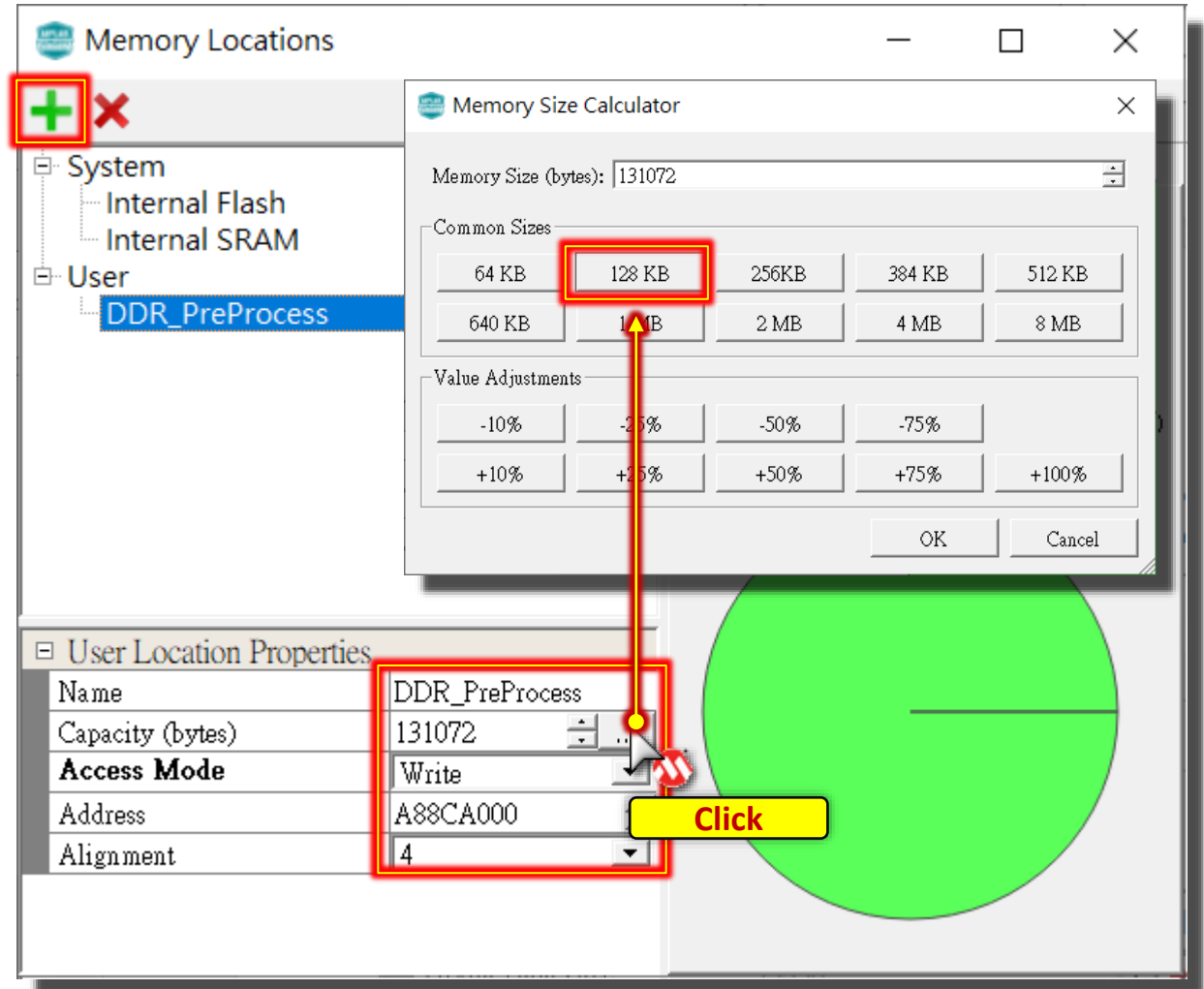


Image Preprocess

- Modify Logo Image through the Asset “Image Manager”.

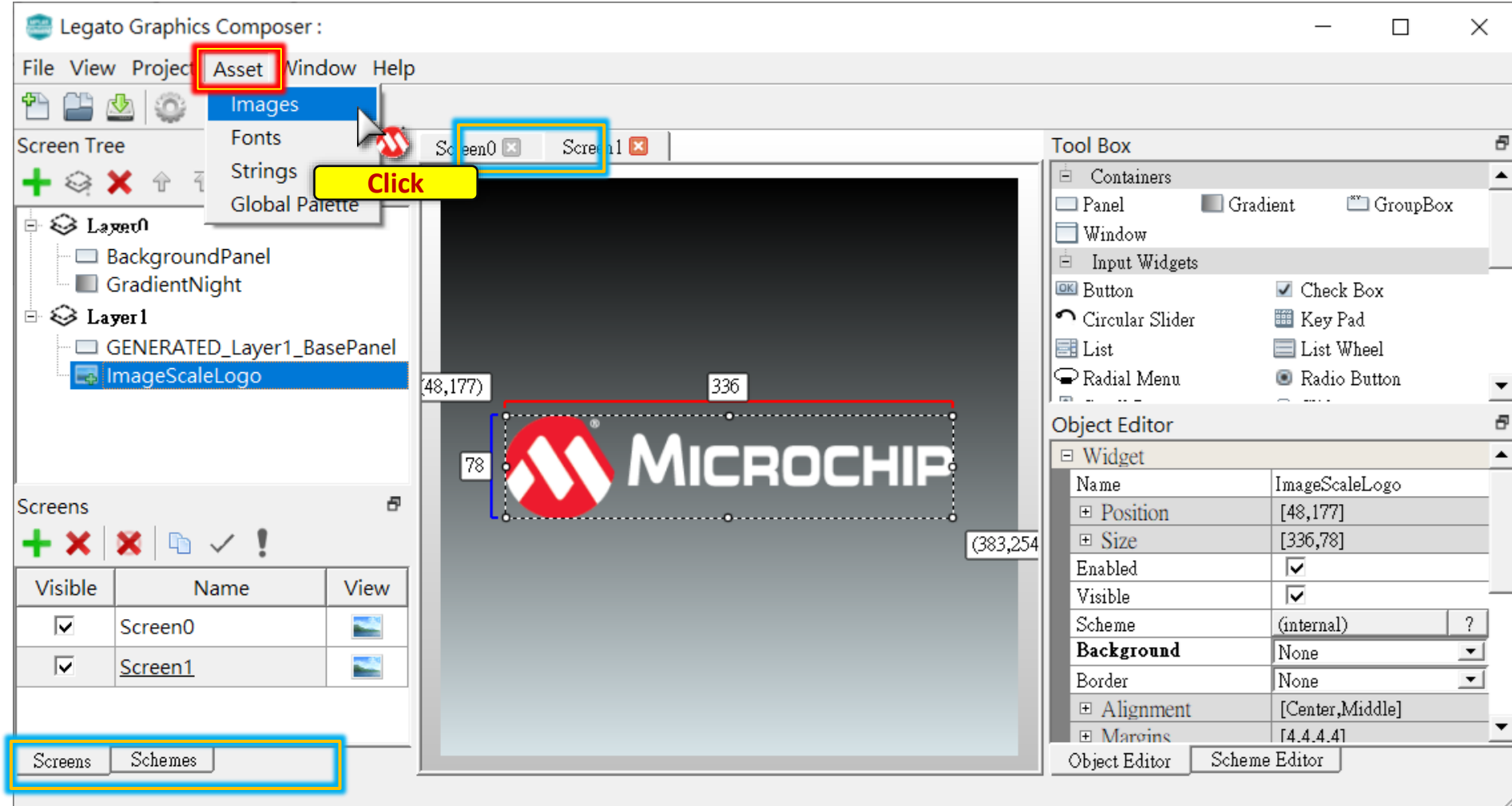


Image Preprocess

- Enable “**Preprocess**” of **mchpLogo_338x78** image as below.

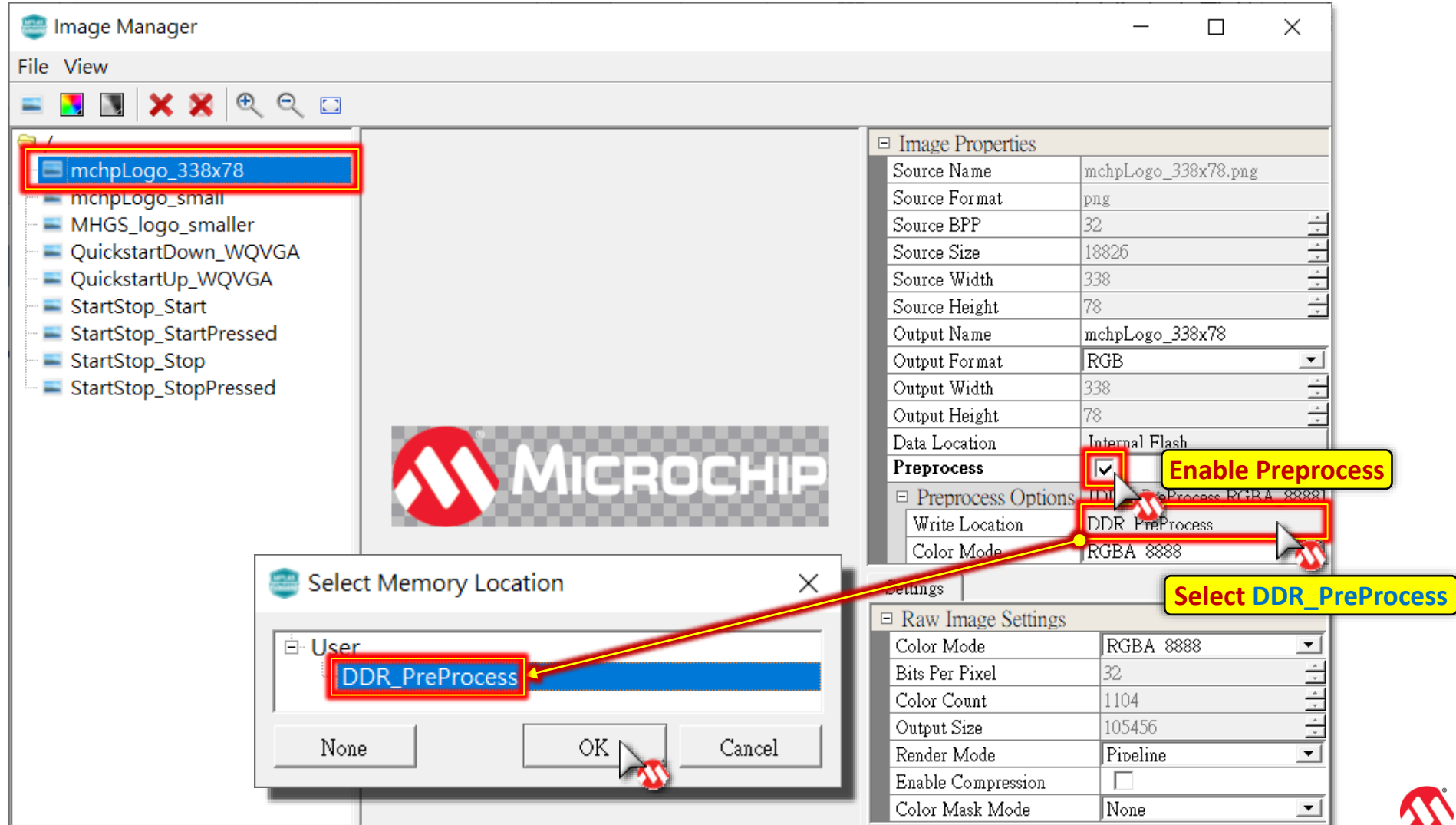
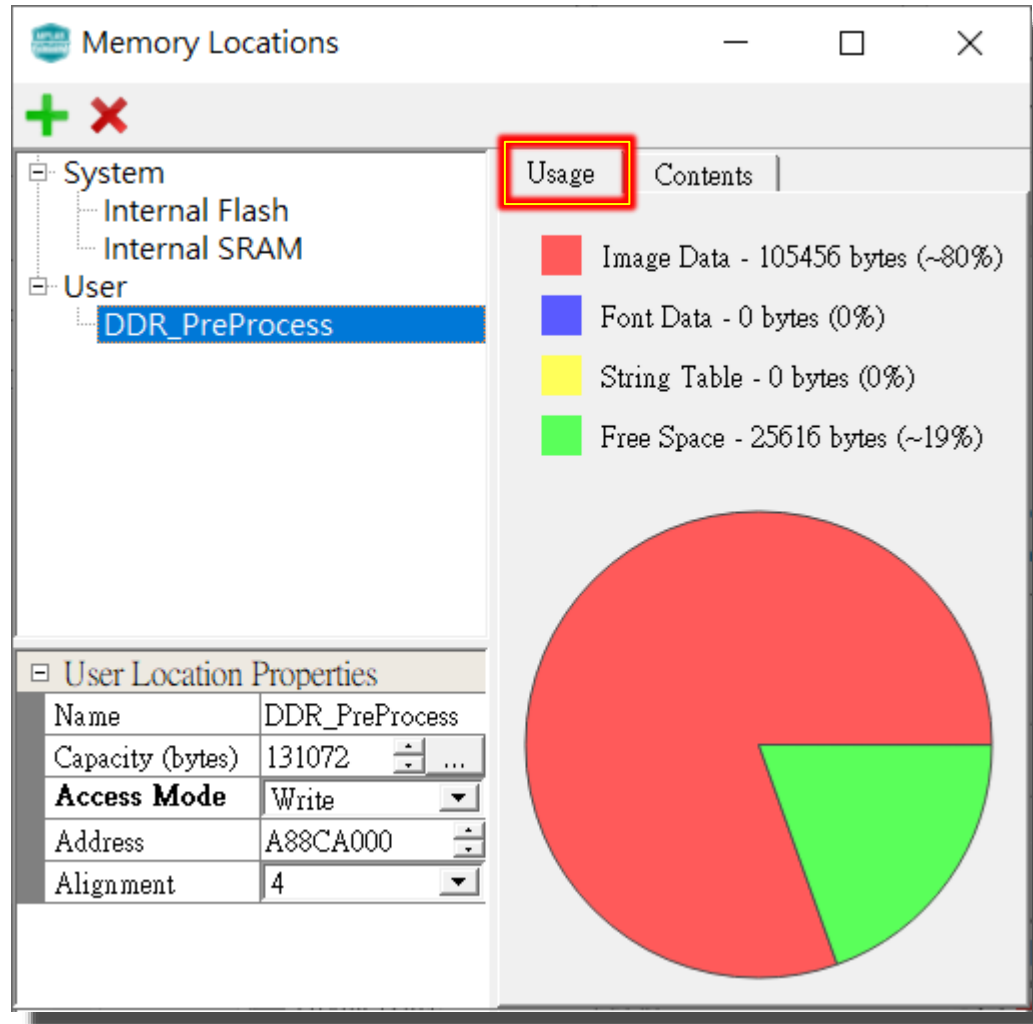


Image Preprocess

- Memory consume after enable Image Preprocess.



The screenshot shows the 'Memory Locations' window with the 'Contents' tab selected. The left sidebar shows a tree view with 'System' (Internal Flash, Internal SRAM) and 'User' (DDR_PreProcess). The main area displays a table of memory contents:

Name	Type	Size (bytes)
mchpLogo_338x78	Preprocessed Image	105456

Below the table, the 'User Location' properties are shown:

User Location ...	
Name	DDR_PreProcess
Capacity (bytes)	131072
Access Mode	Write
Address	A88CA000
Alignment	4

Image Preprocess

- What's new to generate after enable “**Preprocess**” of image ?

le_gen_init.c

```
void legato_initializeScreenState(void)
{
    leSetStringTable(&stringTable);

    initializeStrings();

    legato_preprocessImages();

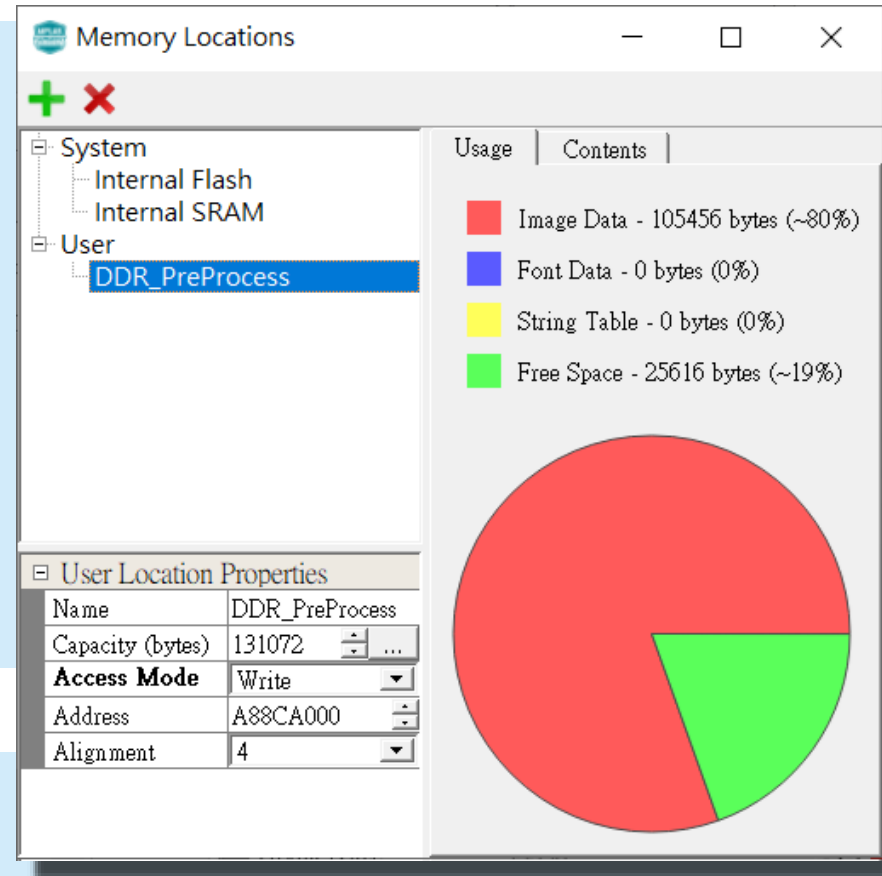
    screenInit_Screen0();
    screenInit_Screen1();

    currentScreen = -1;
    changingToScreen = -1;

    legato_showScreen(screenID_Screen1);
}
```

```
#include "gfx/legato/generated/le_gen_assets.h"
```

```
void legato_preprocessImages(void)
{
    leProcessImage(&mchpLogo_338x78, 0xA88CA000, LE_COLOR_MODE_RGBA_8888);
}
```



le_gen_preprocess.c

Image Preprocess

- How to work of `leProcessImage` function ?

`leProcessImage(&mchpLogo_338x78, 0xA88CA000, LE_COLOR_MODE_RGBA_8888)`

`legato_image.c`

```
void leProcessImage(leImage* img, size_t addr, leColorMode mode)
```

```
{
```

```
    leImage newImage;
```

```
    leRect srcRect;
```

```
    if(img == NULL) return;
```

Allocate `newImage` buffer to use target `DDR Address`

```
    leImage_Create(&newImage, img->buffer.size.width, img->buffer.size.height, mode, (void*)addr, LE_STREAM_LOCATION_ID_INTERNAL);
```

```
    srcRect.x = 0;
```

```
    srcRect.y = 0;
```

```
    srcRect.width = img->buffer.size.width;
```

```
    srcRect.height = img->buffer.size.height;
```

Get `Rectangle` of `Source Image`

```
    leImage_Render(img, &srcRect, 0, 0, LE_TRUE, LE_TRUE, &newImage);
```

Render `Source Image` to `DDR buffer` of `newImage`

```
    img->header = newImage.header;
```

```
    img->format = LE_IMAGE_FORMAT_RAW;
```

```
    img->buffer = newImage.buffer;
```

Point `Source Image` header to `newImage` header

```
}
```

Point `Source Image` buffer to `newImage` buffer

Image Preprocess

- **Important !!** Widget Image Scale or Image Rotate cannot support Preprocessed Image feature.
- Please delete old ImageScaleLogo and change to use a new basic Widget Image to instead of it.

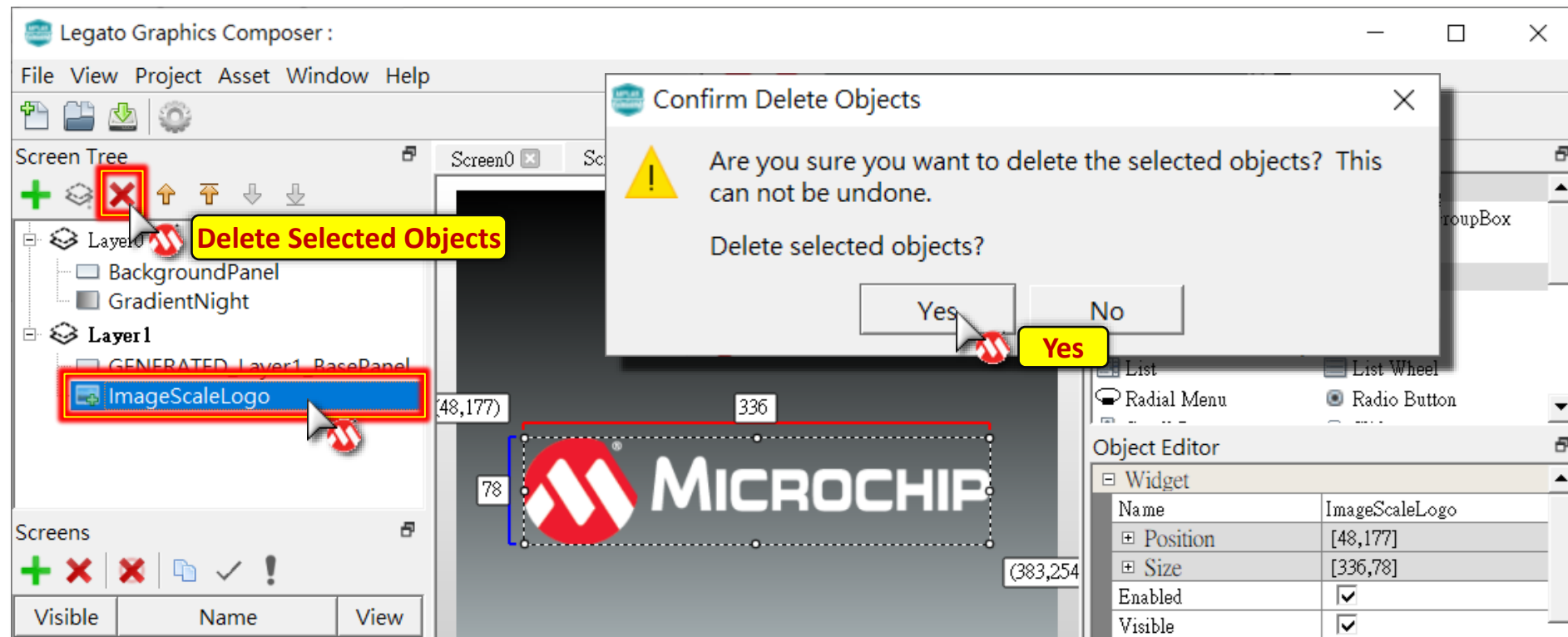


Image Preprocess

- Click [Image] of Display Widgets to add a New Image widget.

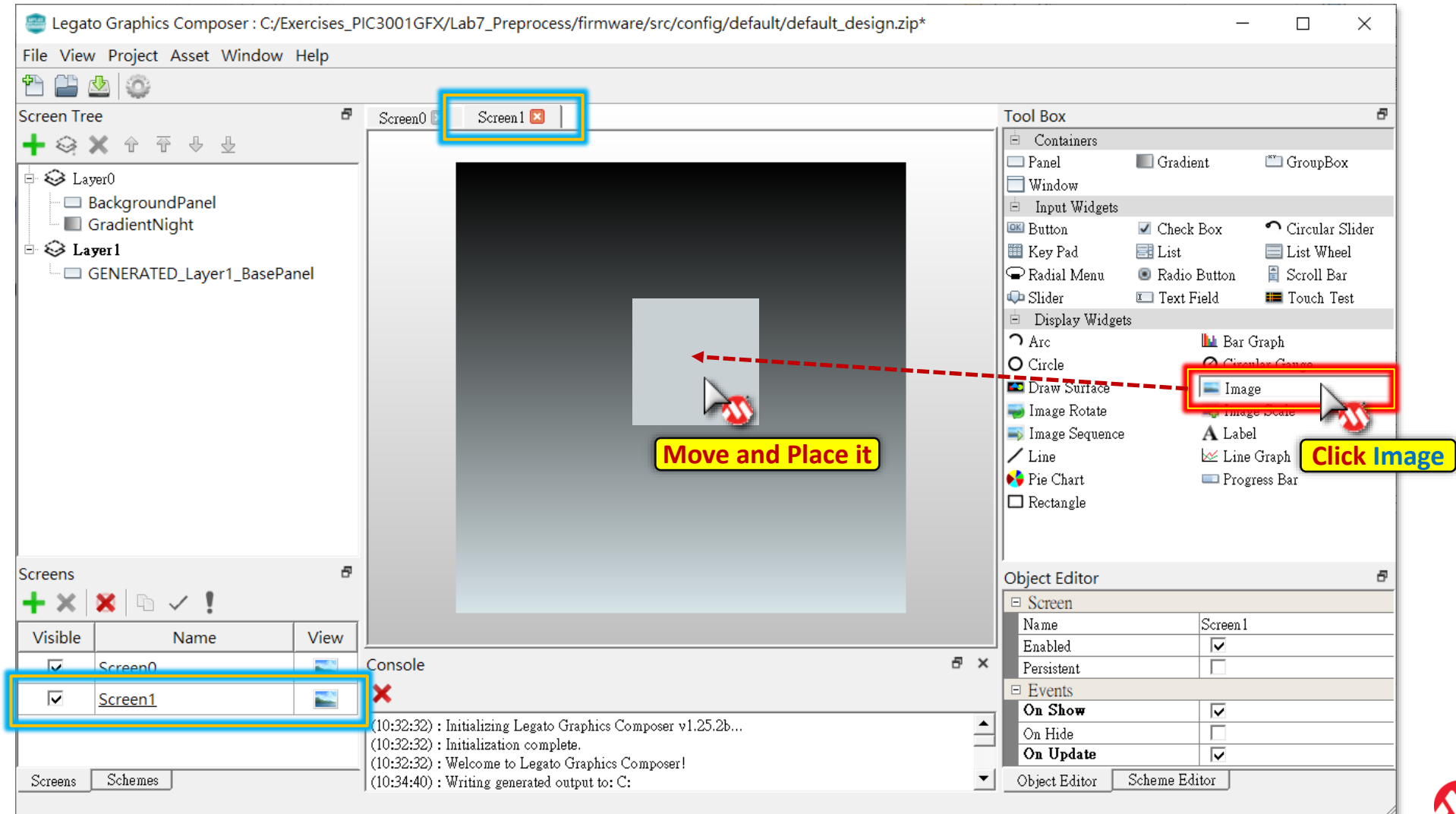


Image Preprocess

- **Configure new Image Widget as below.**

- Name : **ImageScaleLogo**
- Position X : **48**
- Position Y : **177**
- Size Width : **338**
- Size Height : **78**
- Background : **None**
- Alpha Blending : ☒
- Image : **mchpLogo_338x78**

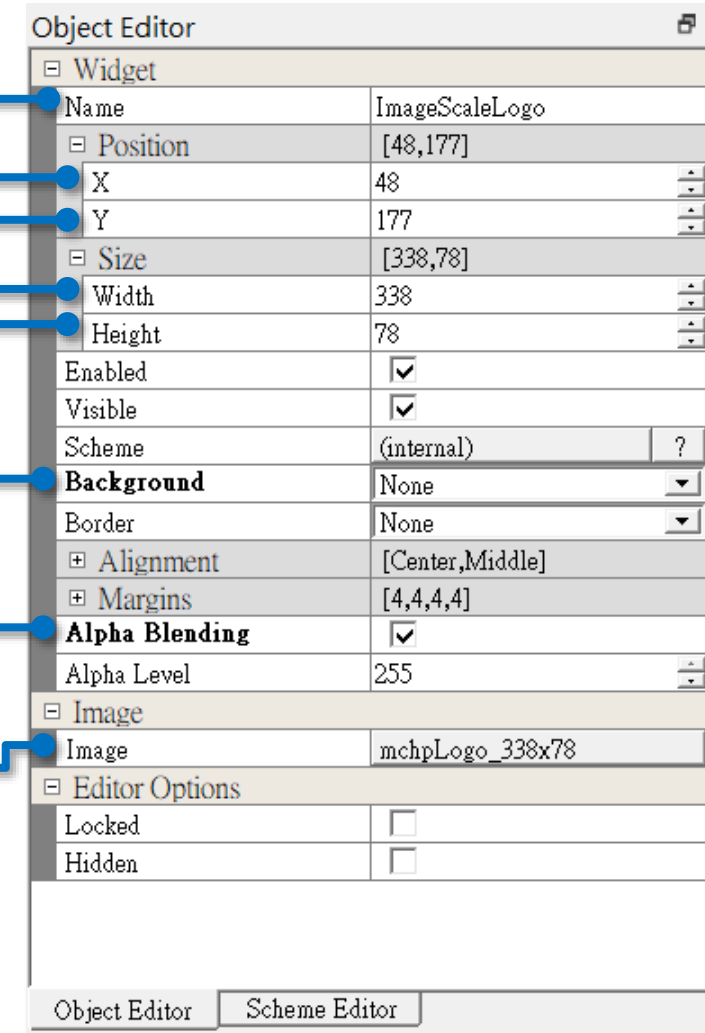


Image Preprocess

- Image Widget after configured.

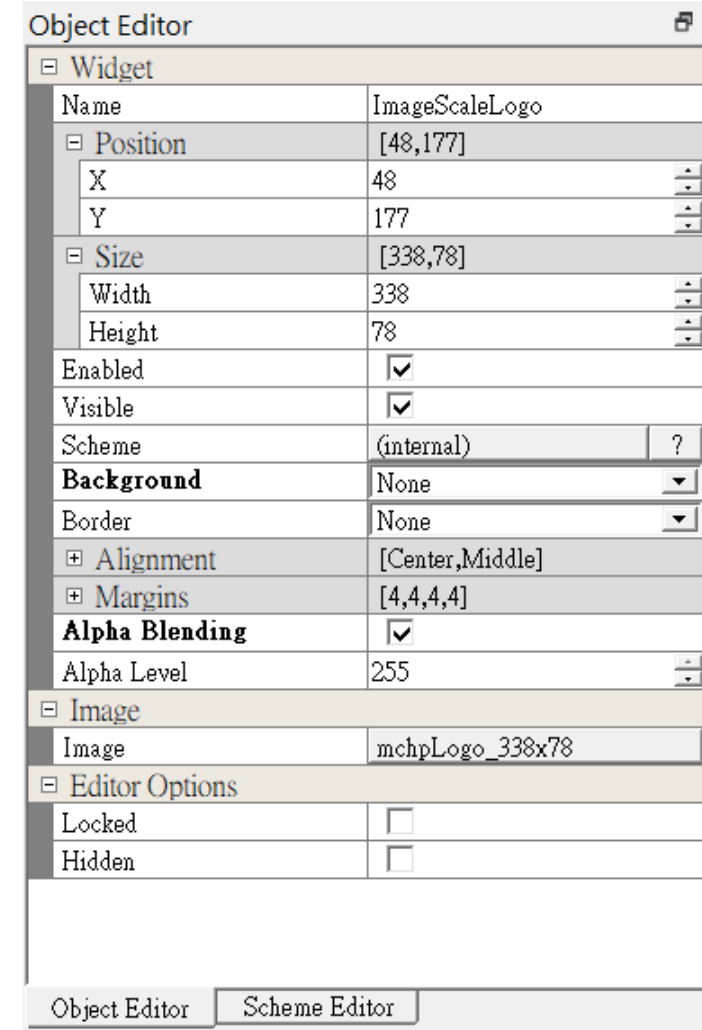
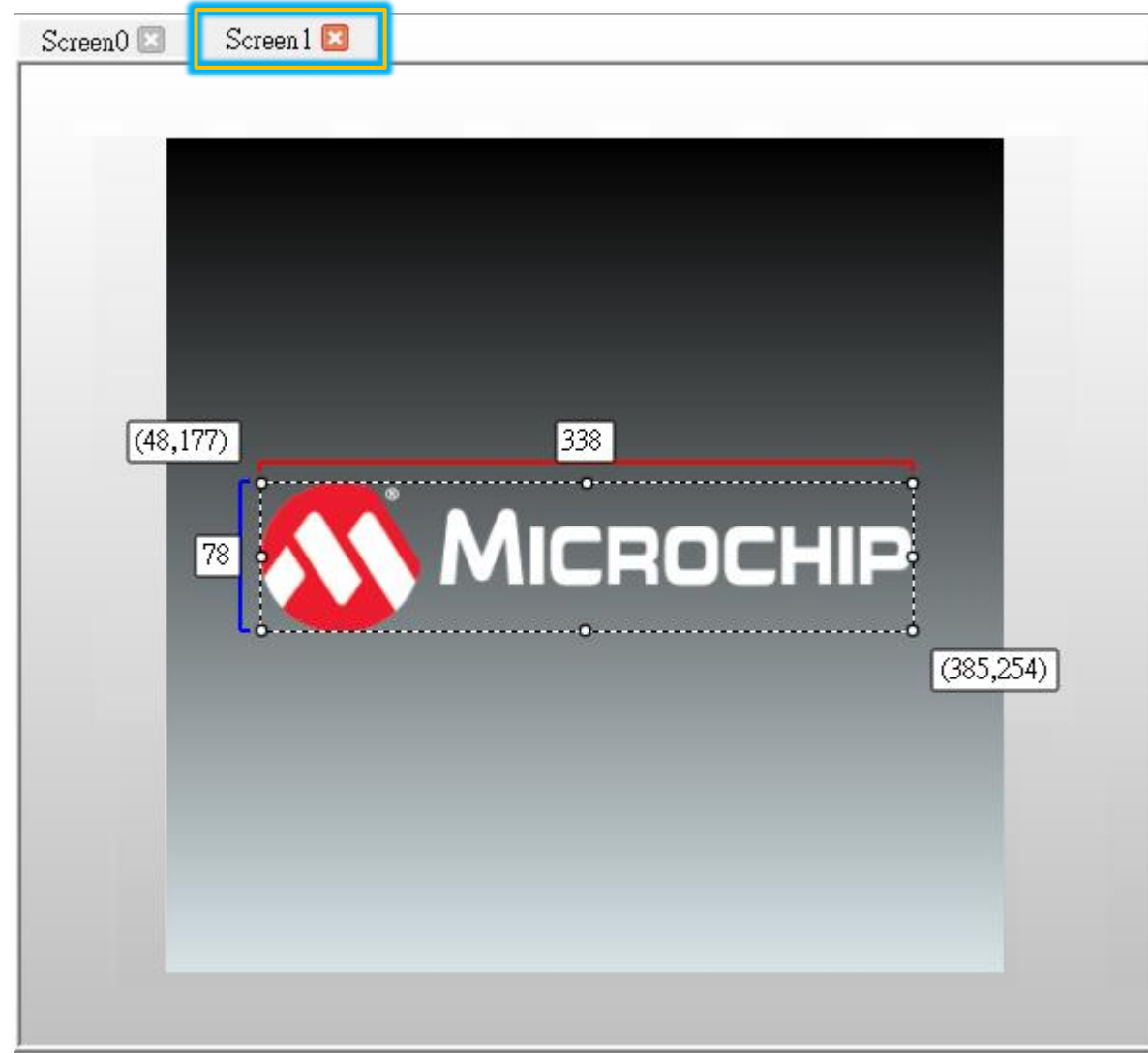
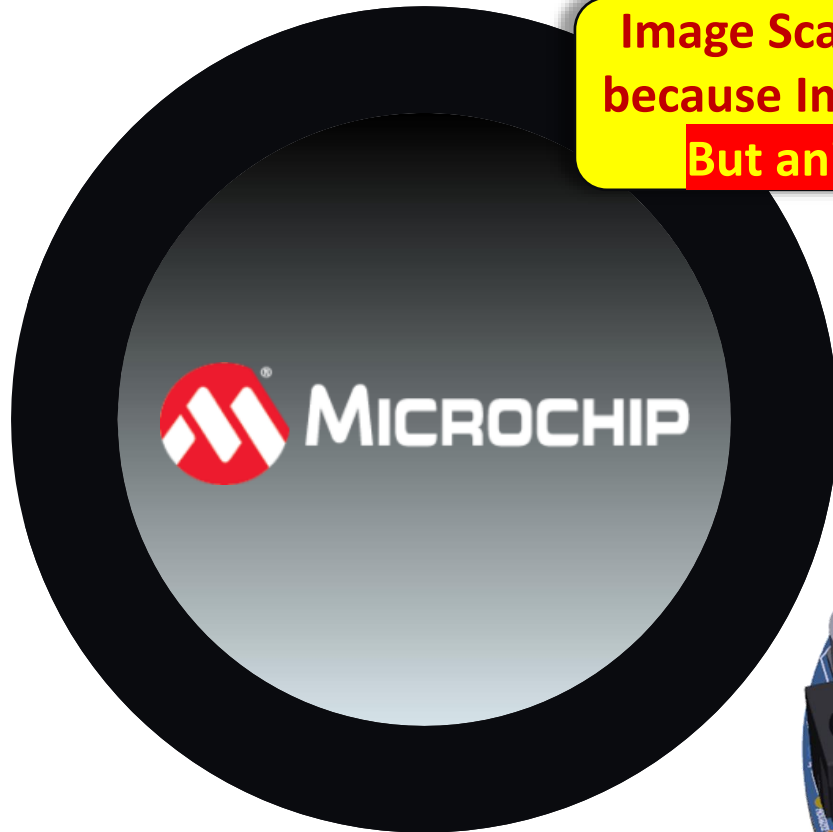


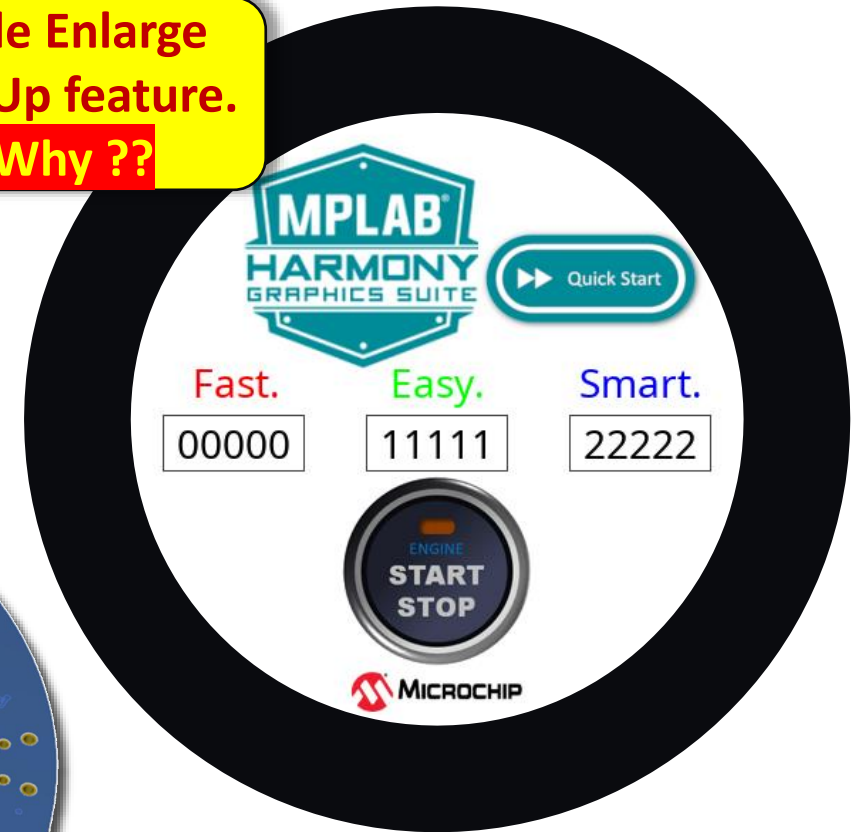
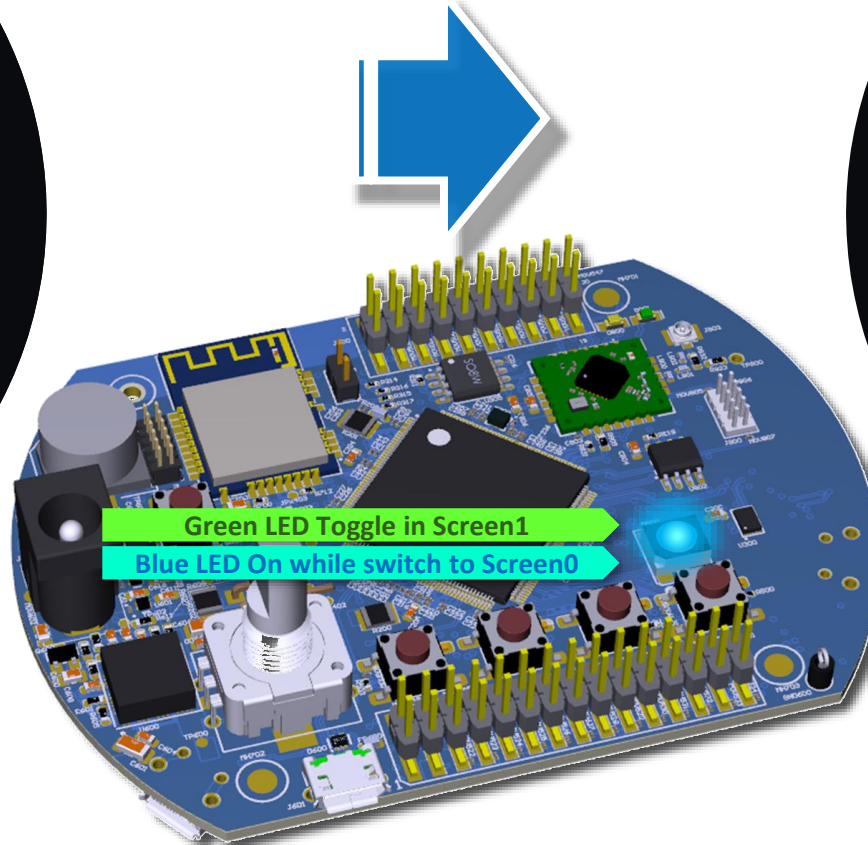
Image Preprocess

- What's different after using Image Preprocess and Image Widget?

Image Scale up animation change to Rectangle Enlarge because Image Widget has not support Scale Up feature.
But animation speed has not improved !! Why ??



Screen1



Screen0

Image Preprocess

- **Important !!** The Image **width** need be **multiple of 8** for proper GPU Memory Bit BLIT function.
- Please delete old **mchpLogo_338x78** and add a new PNG image **mchpLogo** from **\Resource** folder that has **336** pixels width.

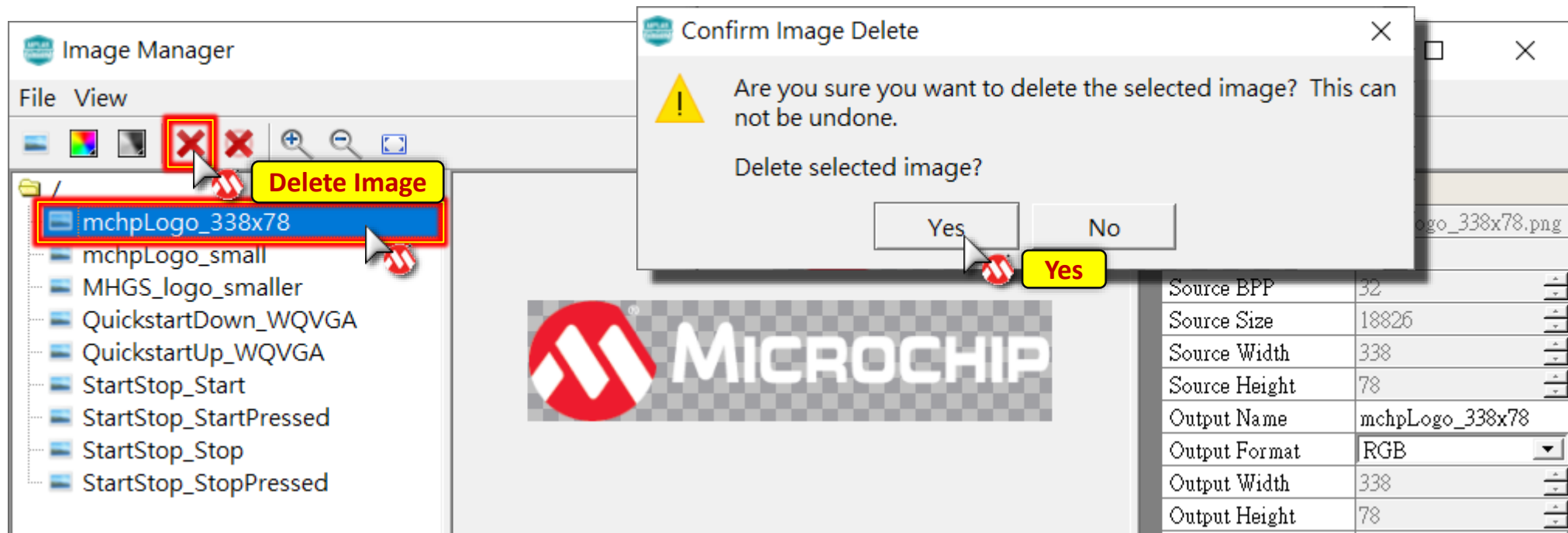


Image Preprocess

- Add **mchpLogo.png** in “Image Asset Manager”.

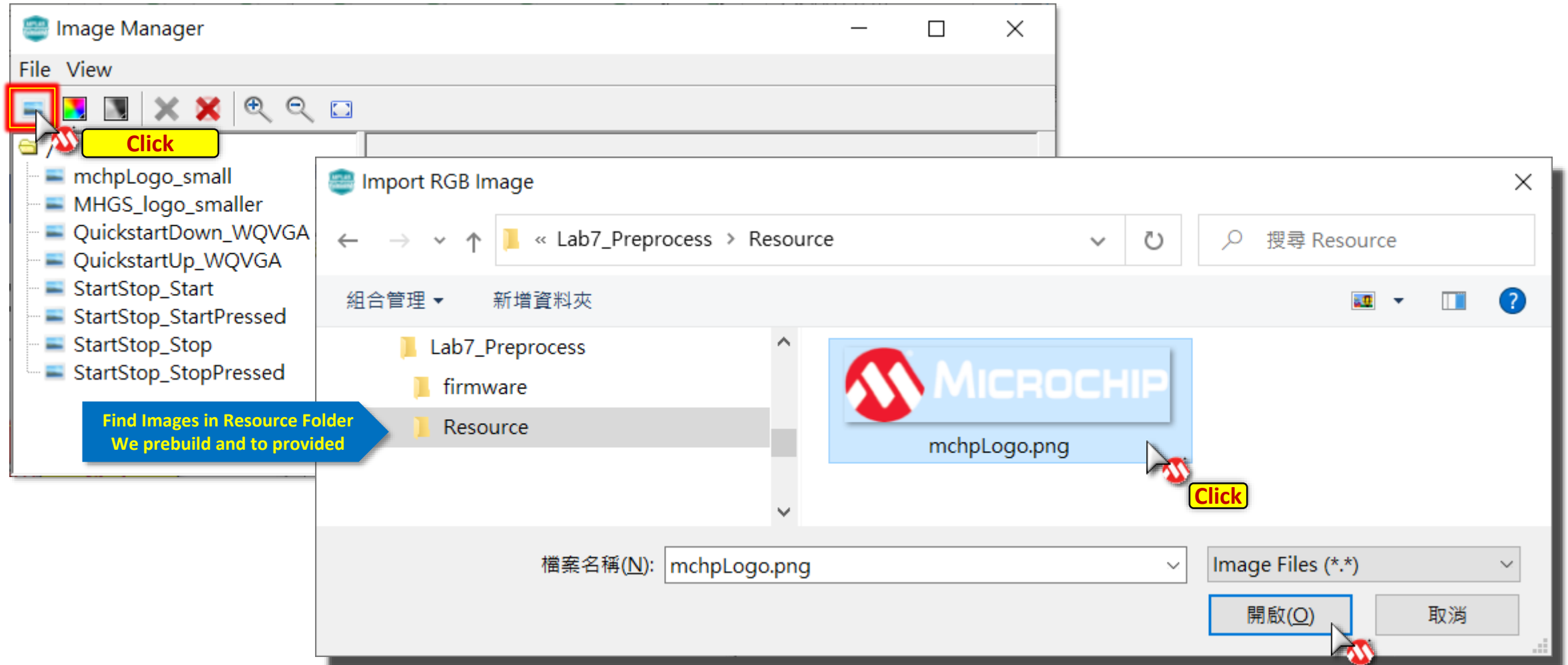


Image Preprocess

- Configure Output Format to **RGB** of **mchpLogo** image.

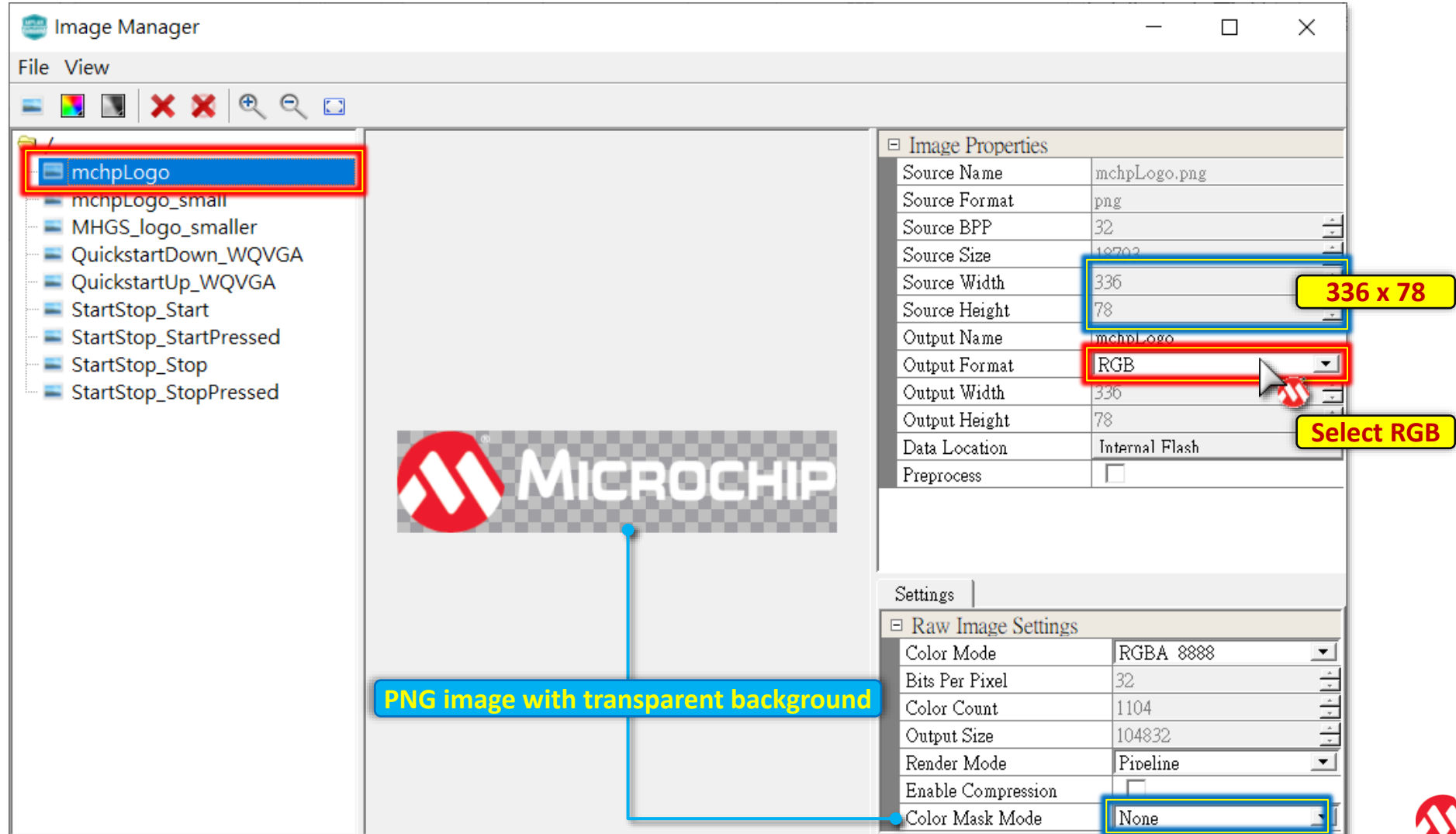


Image Preprocess

- Enable “**Preprocess**” of **mchpLogo** image as below.

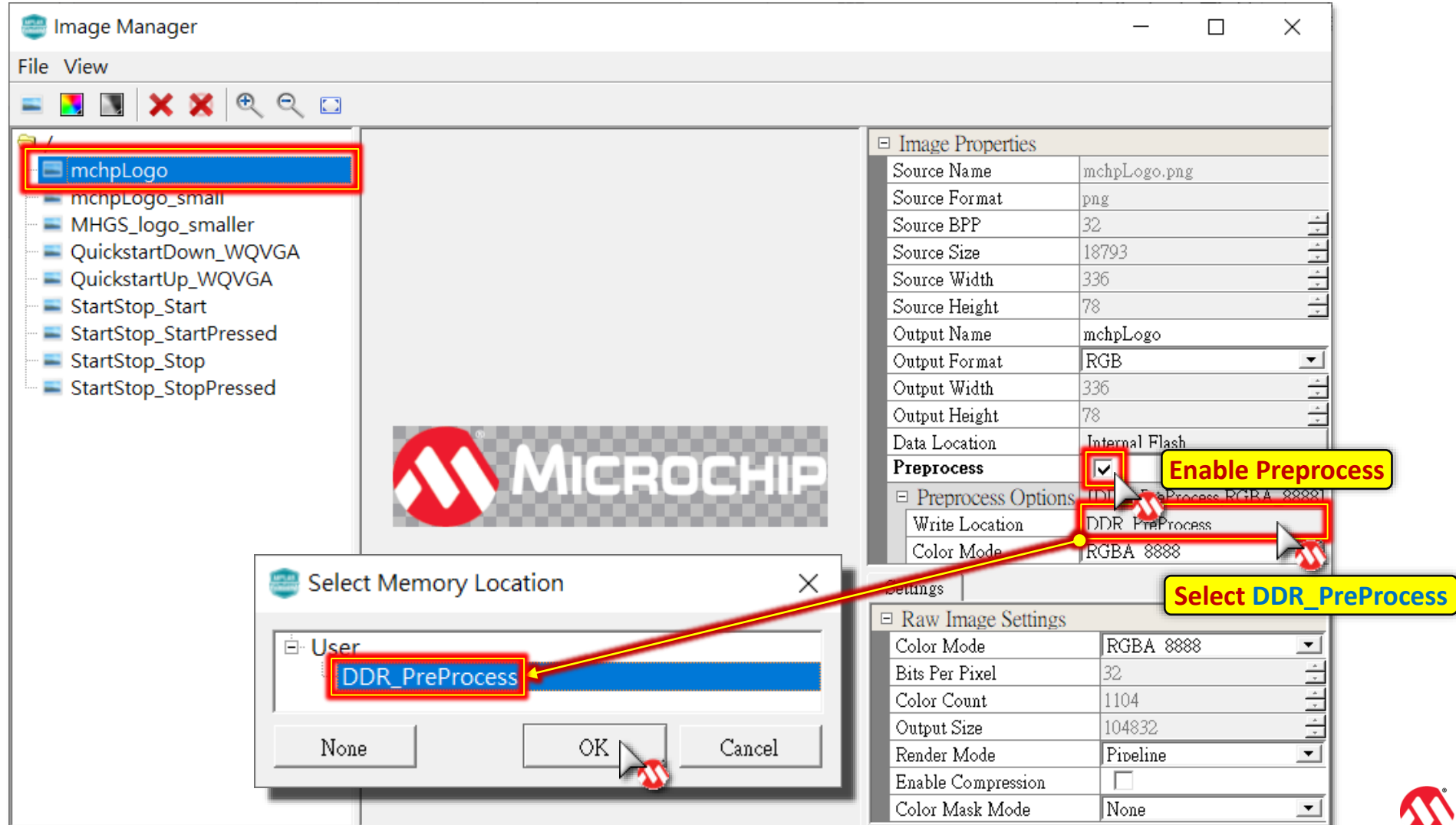


Image Preprocess

- Configure **ImageScaleLogo** Widget as below.

- Name :
- Position X :
- Position Y :
- Size Width :
- Size Height :
- Background :
- Alpha Blending :
- Image :

ImageScaleLogo

48

177

336

78

None

☒

mchpLogo

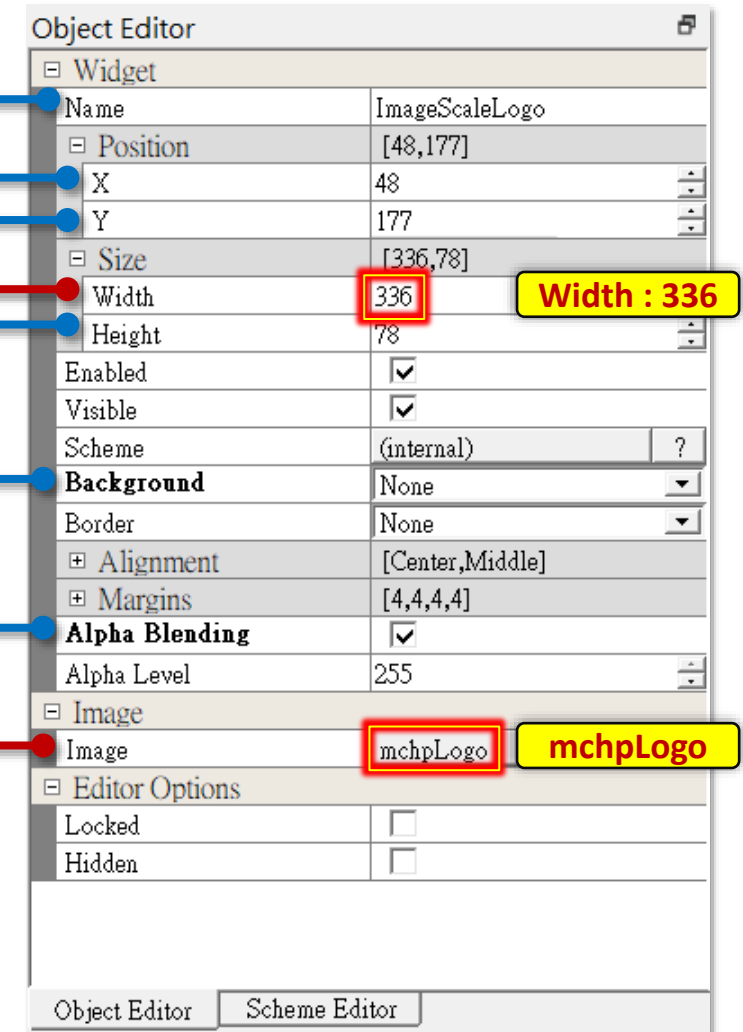
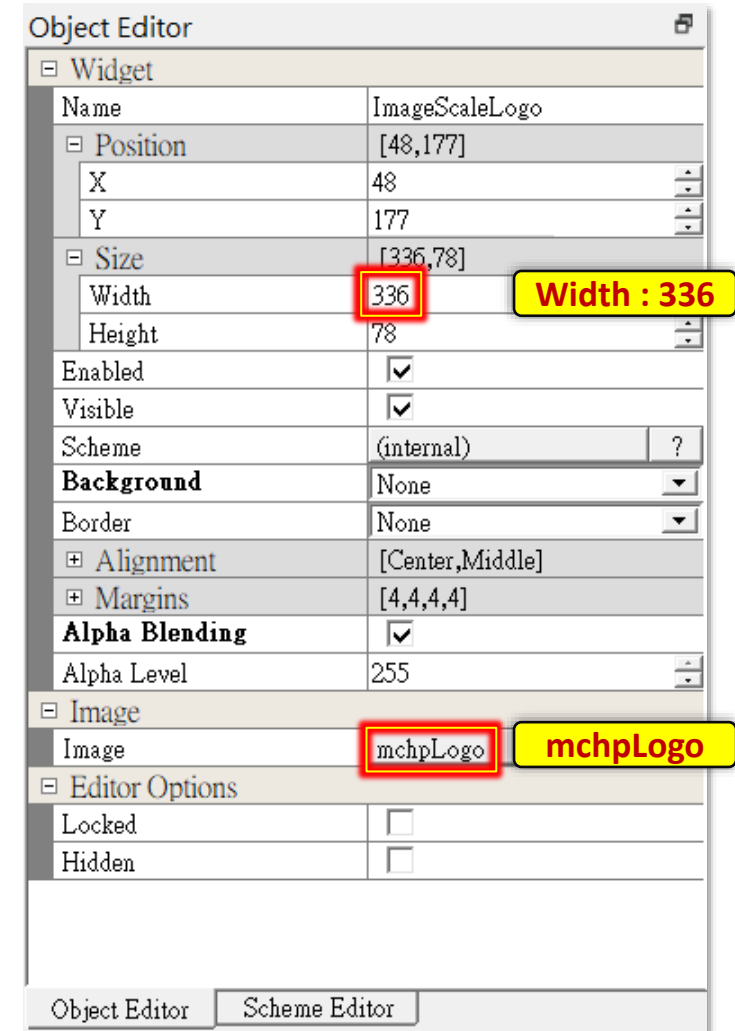
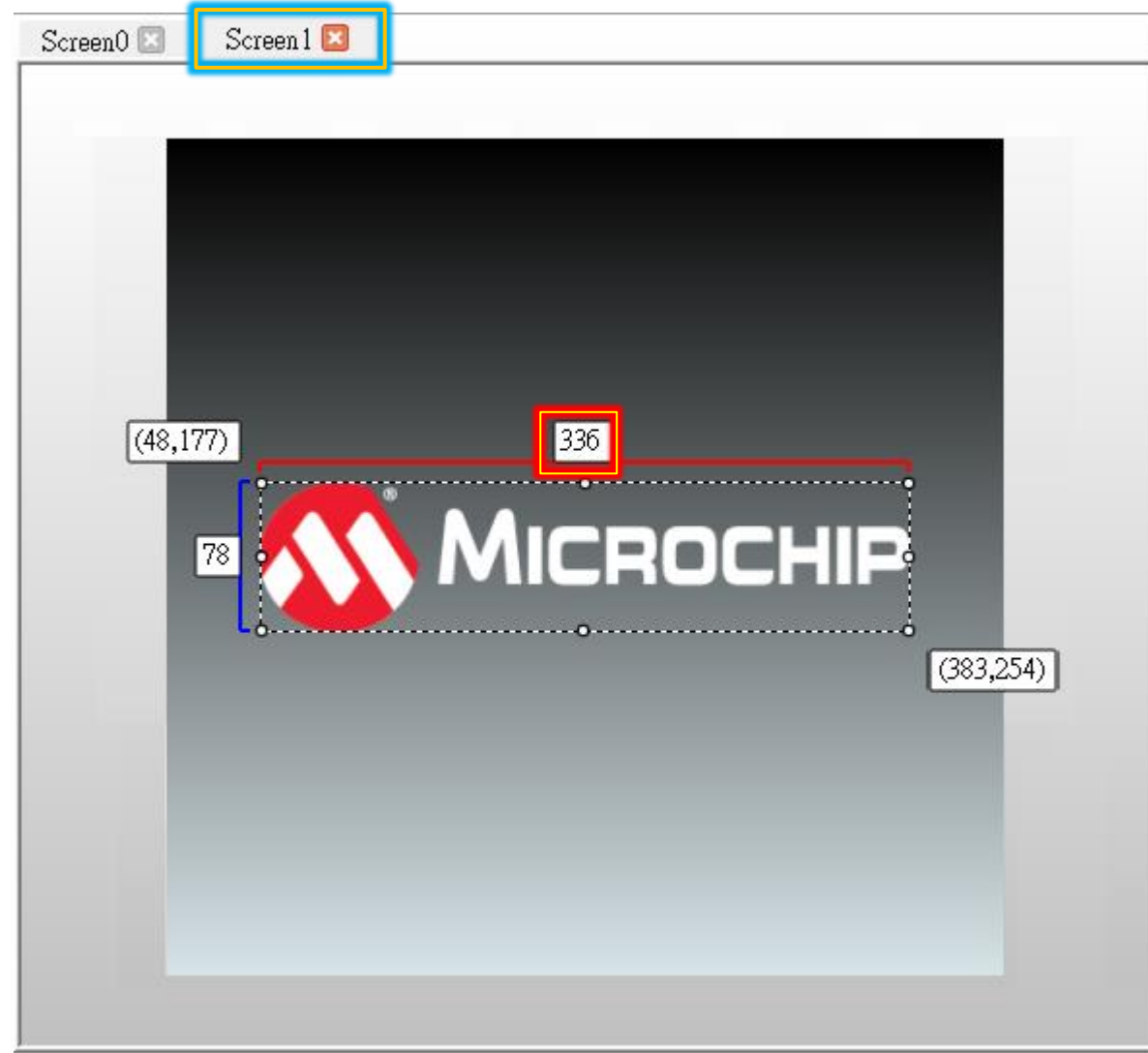


Image Preprocess

- **ImageScaleLogo** Widget after configurated.



Graphic Composer Issues (v3.12.x fixed)

- If you to met **Legato Graphic Composer** “**Crash** and **Close** Issue”, please do below steps could avoid it.
 - (1) Delete old mchpLogo_338x78 image in Image Assets Manager.
 - (2) Close Image Assets Manager.
 - (3) Save Current Design.
 - (4) Exit Legato Graphics Composer.
 - (5) Enter Legato Graphics Composer.
 - (6) Add new mchpLogo image in Image Assets Manager.
 - (7) Configure Output Format to “RGB”
 - (8) Enable “Preprocess” feature and select “DDR_PreProcess” section.
 - (9) Set Width to 336 and assign mchpLogo to ImageScaleLogo widget and.
 - (10) Save Current Design.

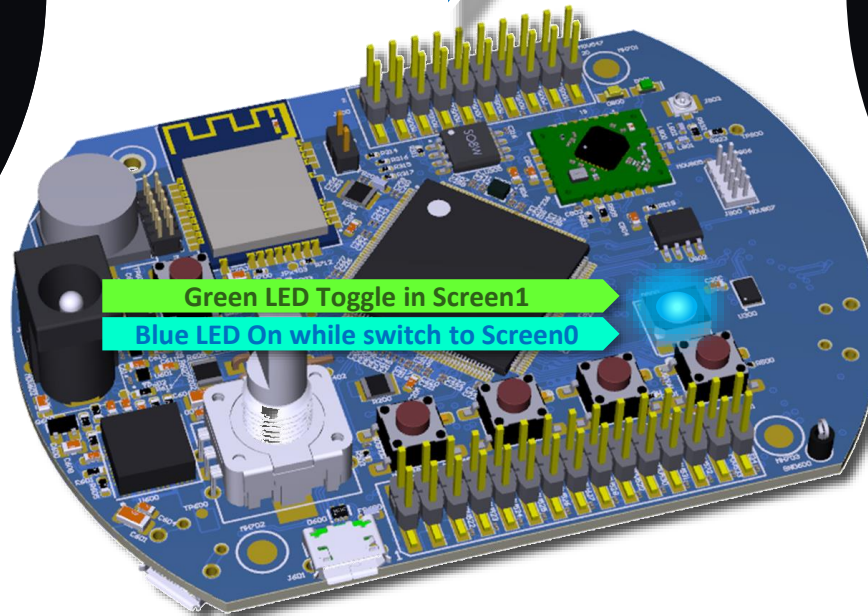
Image Preprocess

- What's different after using Image Preprocess and Image Widget?

Move Left and Fad Out animation speed has improved !!



Screen1



Green LED Toggle in Screen1

Blue LED On while switch to Screen0

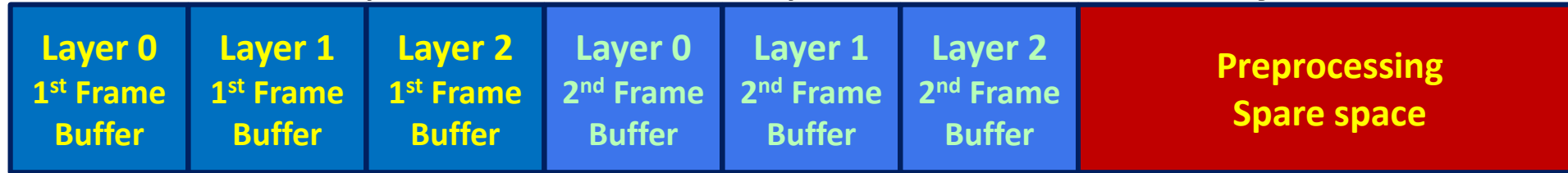


Screen0

Image Preprocess

- **Summary Image Preprocess important notes :**

- Create DDR Preprocessed Memory Section **after all Layers Frame Buffer.**



- **Width** of Preprocessed Image should be **multiple of 8**.



- **Image Scale** and **Image Rotate** Widget **can't** support Preprocessed Image.
- **Image** and **Image Sequence** Widget **could** support Preprocessed Image.



- **[Image Widget]**->fn->**setPosition()** support GPU DDR Bit BLIT.
- **[Image Widget]**->fn->**setAlphaAmount ()** support GPU Alpha Blending.

Image Preprocess without using Graphics Composer

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

Image Preprocess without using Graphics Composer

- If you have many Images would like to configure as Preprocess Image in Legato Graphics Composer, then it has many mouse click works to do and bring bad user experience !

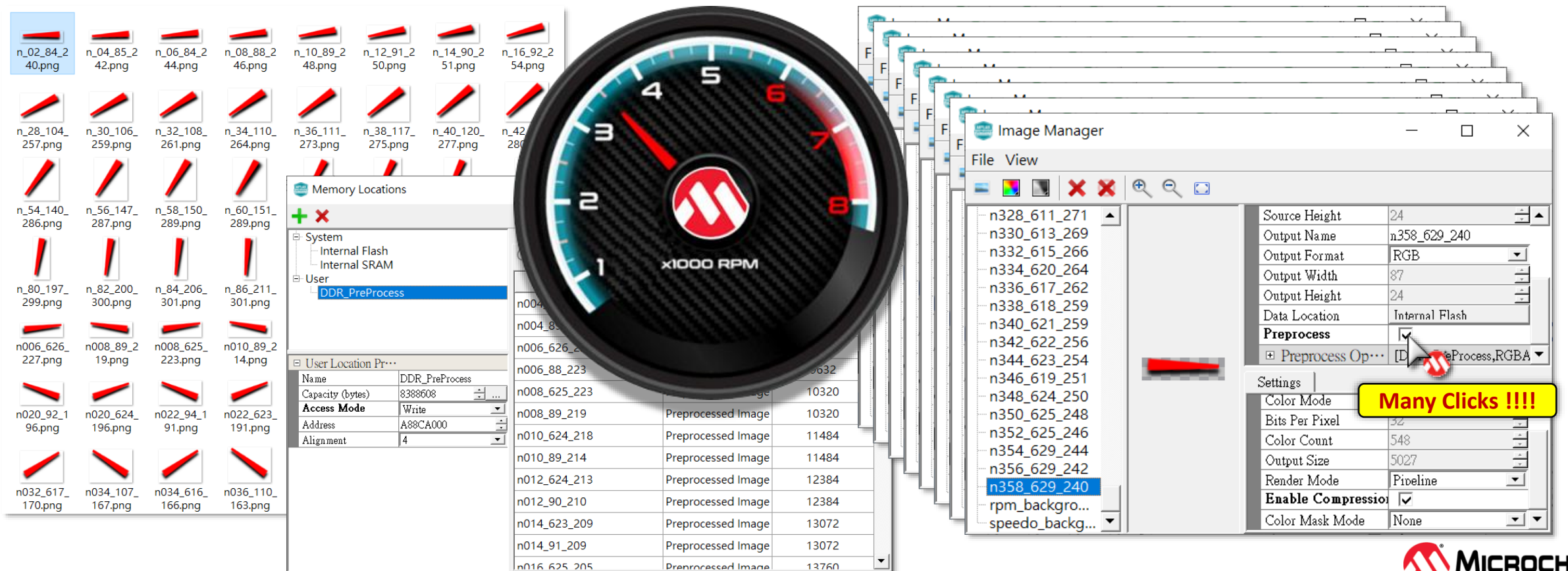


Image Preprocess without using Graphics Composer

- The Image Preprocess feature could direct implement in your project, and it will be much easy to insert/modify or remove.

[le_gen_preprocess.c](#)

```
#include "gfx/legato/generated/le_gen_assets.h"
```

```
void legato_preprocessImages(void)  
{
```

```
    leProcessImage(&n_02_84_240, 0xA88CA000, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_14_90_251, 0xA88D8C00, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_04_85_242, 0xA88CC0A0, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_06_84_244, 0xA88CE238, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_08_88_246, 0xA88D07D8, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_10_89_248, 0xA88D30A0, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_12_91_250, 0xA88D5CF8, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_16_92_254, 0xA88DC64C, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_18_93_256, 0xA88DFCC0, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_22_98_259, 0xA88E70EC, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_20_95_258, 0xA88E35DC, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_26_97_255, 0xA88EECDC, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_24_99_262, 0xA88EAE9C, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_28_104_257, 0xA88F2F7C, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_32_108_261, 0xA88FB7BC, LE_COLOR_MODE_RGBA_8888);  
    leProcessImage(&n_30_106_259, 0xA88F71BC, LE_COLOR_MODE_RGBA_8888);
```

```
    ...  
}
```

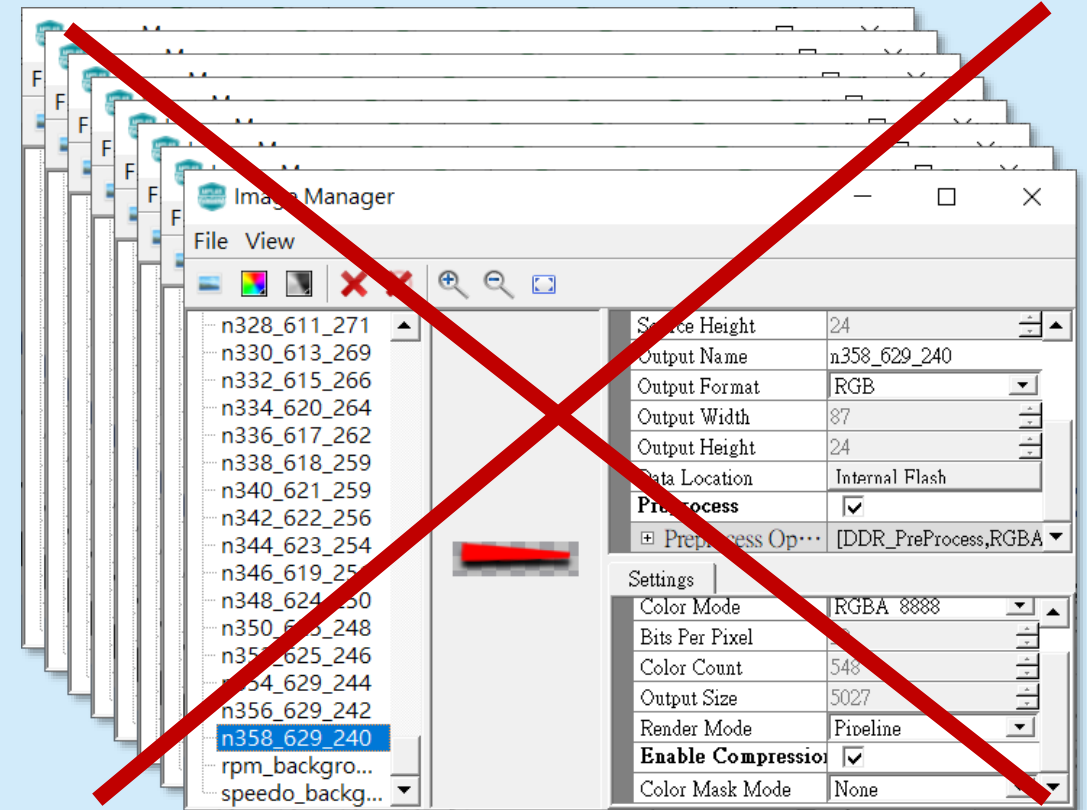


Image Preprocess without using Graphics Composer

- How auto calculate DDR address and alignment of each Image ?
- **Important !! DDR Address should in 16-byte alignment !!**

main.c

```
...  
  
bool FadeOutMoveLogo( void ) { ... }  
  
#define DDR_PREPROCESS_ADDR 0xA88CA000  
void legato_preprocessImages_runtime(void)  
{  
    size_t AddressDDR = DDR_PREPROCESS_ADDR;  
  
    leImage *pPreProcessImages[] =  
        { &mchpLogo, &mchpLogo_small, &MHGS_logo_smaller,  
          &QuickstartDown_WQVGA, &QuickstartUp_WQVGA,  
          &StartStop_Start, &StartStop_StartPressed,  
          &StartStop_Stop, &StartStop_StopPressed };  
  
    for( int i=0 ; i<sizeof(pPreProcessImages)/sizeof(leImage*) ; i++ )  
    {  
        leProcessImage(pPreProcessImages[i], AddressDDR, pPreProcessImages[i]->buffer.mode);  
        AddressDDR = AddressDDR + (pPreProcessImages[i]->buffer.buffer_length);  
        AddressDDR = AddressDDR + (AddressDDR%16?16-(AddressDDR%16):0);  
    }  
}
```

Images Pointer Array

16-bytes alignment

Image Preprocess without using Graphics Composer

- We could **Turn OFF** the LCM backlight while doing **Image Preprocess** because it's needed much time to finish.

main.c

```
...  
  
bool FadeOutMoveLogo( void ) { ... }  
  
#define DDR_PREPROCESS_ADDR  0xA88CA000  
void legato_preprocessImages_runtime(void)  
{  
    ...  
}  
  
void Screen1_OnShow(void)  
{  
    BACKLIGHT_TYPE_1_Clear();  
    legato_preprocessImages_runtime();  
    BACKLIGHT_TYPE_1_Set();  
  
    ScaleUpLogo( 8 );  
    RGB_LED_G_On();  
    RGB_LED_B_Off();  
}  
  
void Screen1_OnUpdate(void) { ... }
```

Execute Runtime Image Preprocess in **OnShow()** event of **Screen1** and Turn OFF the LCM backlight while doing Image Preprocess

Image Preprocess without using Graphics Composer

- Image placement simulation in DDR Memory.

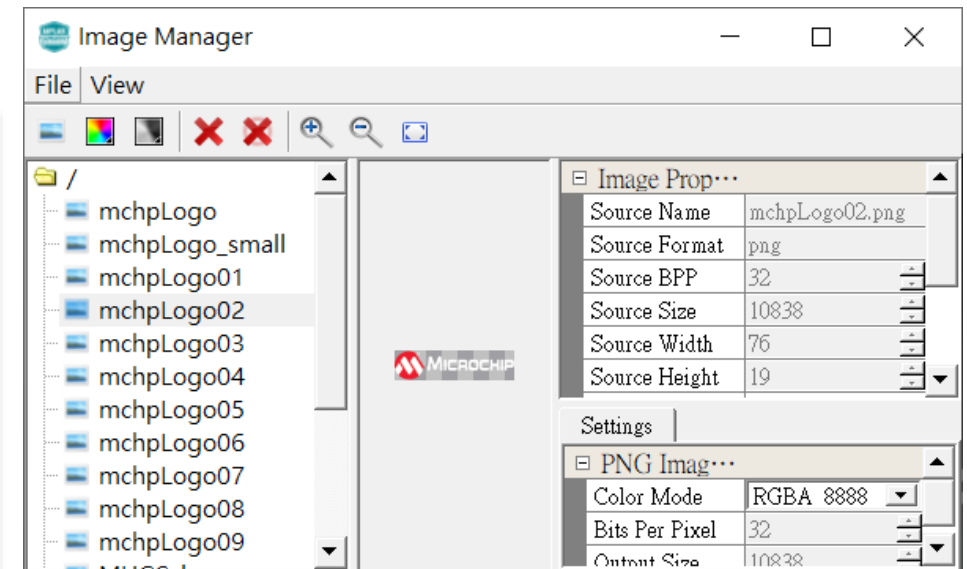
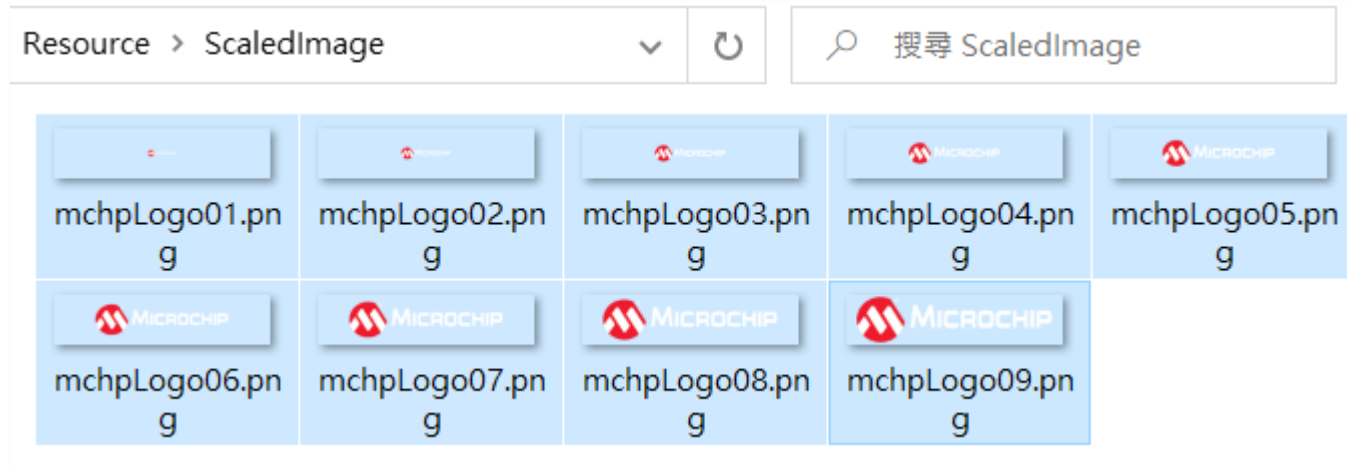


GPU Bit BLIT Stretch Image

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

GPU Bit BLIT Stretch Image

- In previous sections, we know the **Image Scale Widget** is implement in software and **cannot** support **Image Preprocess**.
- If we want to implement a **scaling up image animation**, then we can use **Image Preprocess** by sequence Images we pre-built.
- But it's difficulty to prepare many different images and put them in Legato Graphics Composer !!



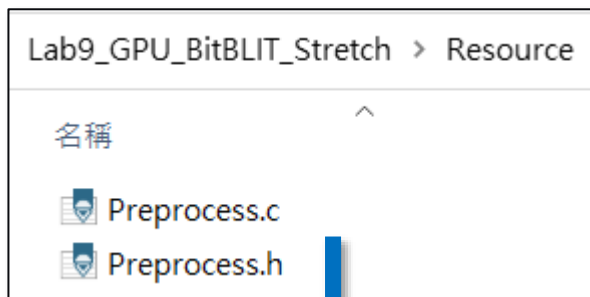
GPU Bit BLIT Stretch Image

- PIC32MZ DA support **GPU (Nano2D) Graphic Bit BLIT Stretch** could help to Pre-built sequences Images in efficiently.

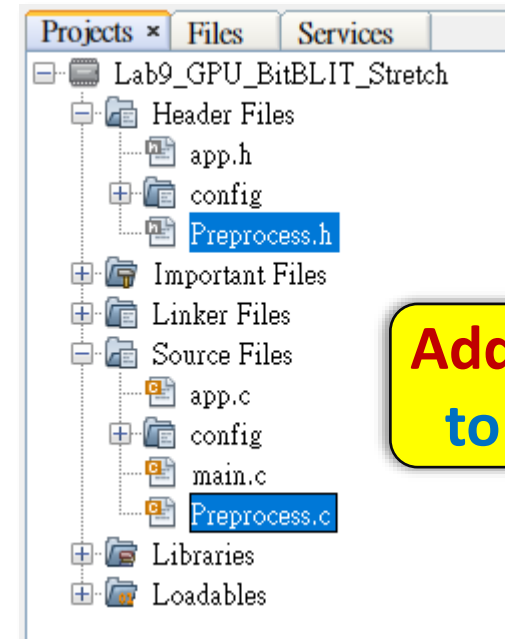
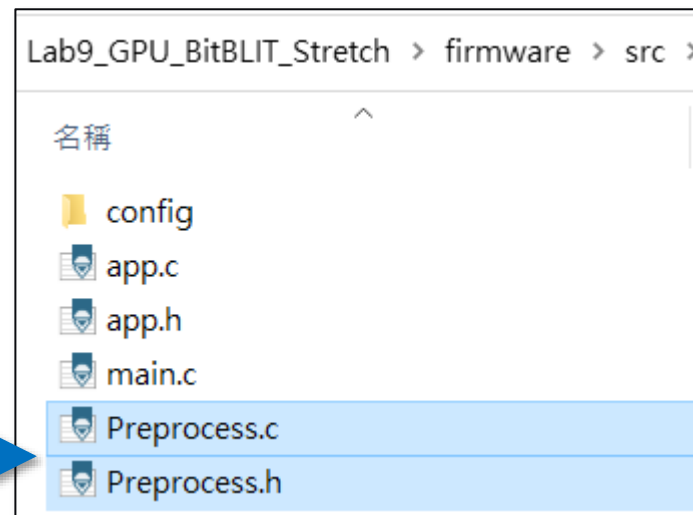


GPU Bit BLIT Stretch Image

- The default Legato Graphic Library has not ready function could help for easy implement **Scaled Up/Down** animation sequence of Images, then here has an example could help you to finish it.
- Please copy the implementation files **Preprocess.c** and **.h** in folder **\Resource** to **[Your Project]\firmware\src** folder and include them in your project.



Copy



**Add Existing Item
to your Project**

GPU Bit BLIT Stretch Image

- Let's look at Preprocess.c and go through in briefly. [Preprocess.c](#)

```
#include "Preprocess.h"
```

```
#define DDR_PREPROCESS_ADDR 0xA88CA000
```

```
#define MAX_SCALED_IMAGE 50
```

```
static leImage ScaledImage[MAX_SCALED_IMAGE];
```

```
static uint32_t ScaledImageCount;
```

```
// Legato Color Mode to GFX Color Mode
```

```
static gfxColorMode _convertColorMode(leColorMode mode)
```

```
{
```

```
...
```

Color Mode Convert

```
}
```

```
// Hardware GPU Bit Blit Stretch Image to Destination Buffer in DDR
```

```
extern leRenderState _renderState;
```

```
leResult leGPU_BlitStretchImageBuffer( const lePixelBuffer* sourceBuffer, const lePixelBuffer* destBuffer,  
                                       const leRect* sourceRect,      const leRect* destRect,  
                                       uint32_t a)
```

```
{
```

```
...
```

GPU Bit BLIT Stretch Image from Source Buffer to Destination Buffer

```
}
```

```
// Bit BLIT Stretch Source leImage to new Destination leImage in DDR
```

```
leResult leStretchImage( leImage* imgSrc, leImage* imgDst, leRect* dstRect, size_t Addr )
```

```
{
```

```
...
```

Stretch Source leImage to New Destination leImage and Allocate to DDR

```
}
```

GPU Bit BLIT Stretch Image

- Preprocessed Image sequence generate.

Preprocess.c

```
void leStretchImagePreprocess( leImage* imgSrc, int32_t StartWidth, int32_t EndWidth, int32_t StepWidth, size_t dstAddr )
```

```
{  
    float srcRaito;           Source Images      Start Width      End Width      Step Width      Dest Address  
    leRect dstRect;
```

```
    srcRaito = (float)(imgSrc->buffer.size.height)/(float)(imgSrc->buffer.size.width);
```

```
    dstRect.x = 0;
```

```
    dstRect.y = 0;
```

```
    dstRect.width = StartWidth; // Must be multiple of 8
```

```
    ScaledImageCount = 0;
```

```
    while( ScaledImageCount < MAX_SCALED_IMAGE )
```

```
    {
```

```
        if( StartWidth <= EndWidth )
```

```
        { if( dstRect.width > EndWidth ) break;
```

```
        }
```

```
        else
```

Stretch or Shrink finish check

```
        { if( dstRect.width < EndWidth ) break;
```

```
        }
```

```
    dstRect.height = (int16_t)((float)(dstRect.width) * srcRaito);
```

```
    while( leStretchImage( imgSrc, &ScaledImage[ScaledImageCount], &dstRect, dstAddr ) == LE_FAILURE ) {
```

```
        ScaledImageCount++;
```

```
        dstAddr = dstAddr + (dstRect.width * dstRect.height * 4);
```

```
        dstAddr = dstAddr + (dstAddr%16?16-(dstAddr%16):0);
```

```
        dstRect.width = dstRect.width + StepWidth; // Must be multiple of 8
```

```
    }
```

```
}
```

Keep Ratio as Source Image

Stretch Source to Destination

Increase Count of Stretched Image

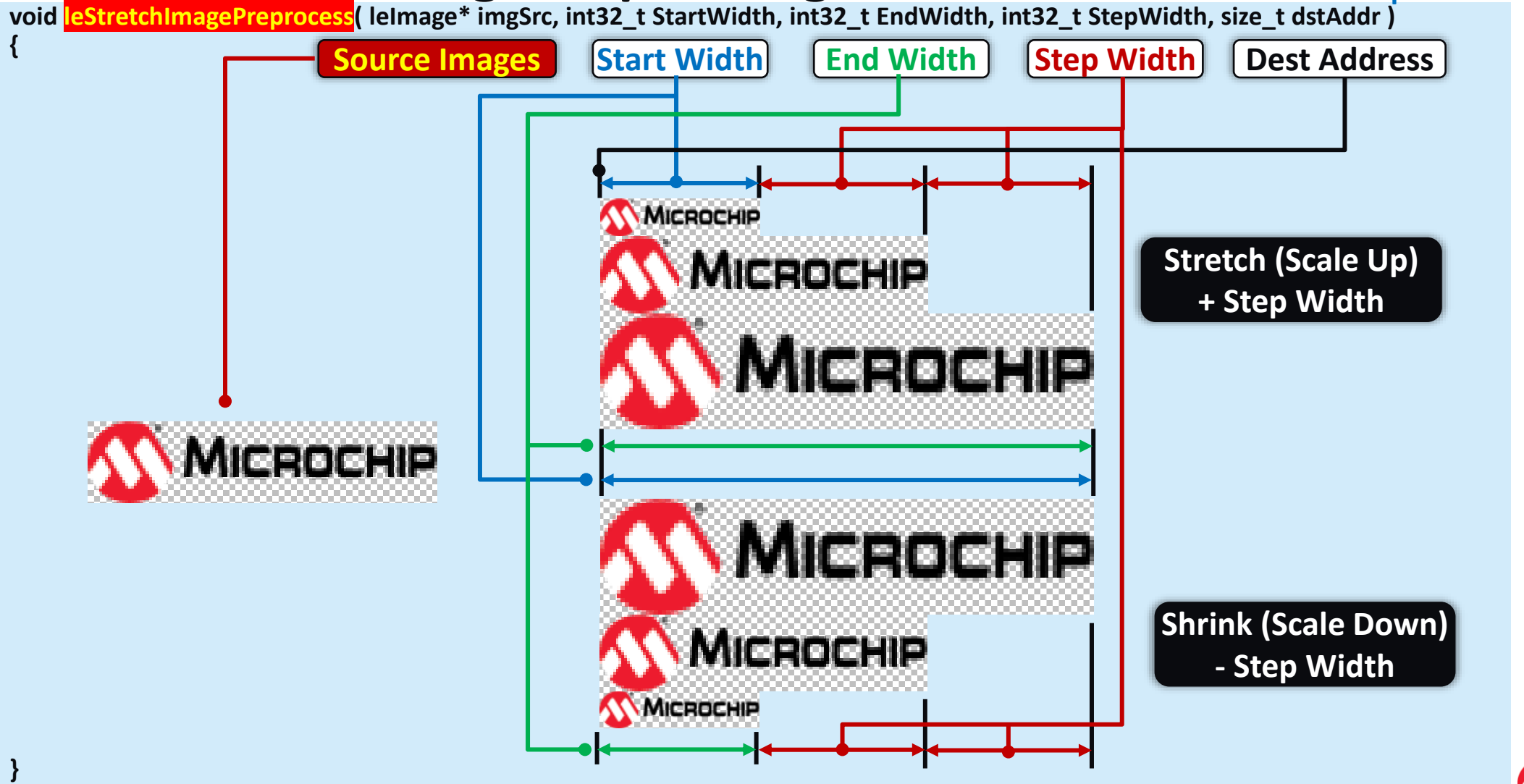
Offset Image Size in 4 bytes

Alignment 16 for GPU Bit BLIT

Image Width must be Multiple of 8

GPU Bit BLIT Stretch Image

- Preprocessed Image sequence generate.



GPU Bit BLIT Stretch Image

- Let's look at Preprocess.c

Preprocess.c

```
void legato_preprocessImages_runtime(void)
{
    size_t AddressDDR = DDR_PREPROCESS_ADDR;

    leImage *pPreProcessImages[] = { &mchpLogo, &mchpLogo_small, &MHGS_logo_smaller, ..., &StartStop_StopPressed };

    for( int i=0 ; i<sizeof(pPreProcessImages)/sizeof(leImage*) ; i++ )
    {
        leProcessImage(pPreProcessImages[i], AddressDDR, pPreProcessImages[i]->buffer.mode);
        AddressDDR = AddressDDR + (pPreProcessImages[i]->buffer.buffer_length);
        AddressDDR = AddressDDR + (AddressDDR%16?16-(AddressDDR%16):0);
    }

    // Must after leProcessImage
    leStretchImagePreprocess( &mchpLogo, 8, mchpLogo.buffer.size.width, 8, AddressDDR );
}

bool legato_preprocessImages_ScaleUp( leImageWidget* imgWidgetDst, uint32_t ScaledImageIdx )
{
    if( ScaledImageIdx < ScaledImageCount )
    {
        imgWidgetDst->fn->setImage( imgWidgetDst, &ScaledImage[ScaledImageIdx] );
        if( ScaledImageIdx == ScaledImageCount-1 )
            return false;
    }
    return true;
}
```

Images Pointer Array

16-bytes alignment

GPU Bit BLIT Stretch Image

- New **ScaleUpLogo** function to use Preprocessed Images.

main.c

```
...
#include "Preprocess.h"

bool ScaleUpLogo( uint32_t Init_W )
{
    uint32_t W, H;
    static uint32_t ScaledImageIdx = 0;

    if( Init_W )
    {
        ScaledImageIdx = 0;
    }

    if( legato_preprocessImages_ScaleUp( Screen1_ImageScaleLogo, ScaledImageIdx ) )
    {
        return true;
    }

    W = Screen1_ImageScaleLogo->fn->getWidth( Screen1_ImageScaleLogo );
    H = Screen1_ImageScaleLogo->fn->getHeight( Screen1_ImageScaleLogo );
    Screen1_ImageScaleLogo->fn->setPosition( Screen1_ImageScaleLogo, ( SCREEN_W - W )>>1, ( SCREEN_H - H )>>1 );
    ScaledImageIdx++;

    RGB_LED_G_Toggle();

    return false;
}
```

Include Preprocess.h

New Image Scale Up Function

Static Sequence Index

Reset Sequence Index

Assign given Indexed Image to ImageScaleLogo Widget

Scale Up finish when Index has exceeded Preprocess Count

Get current Image Width and Height

Centering Image Position

Increase Index

Toggle Green LED

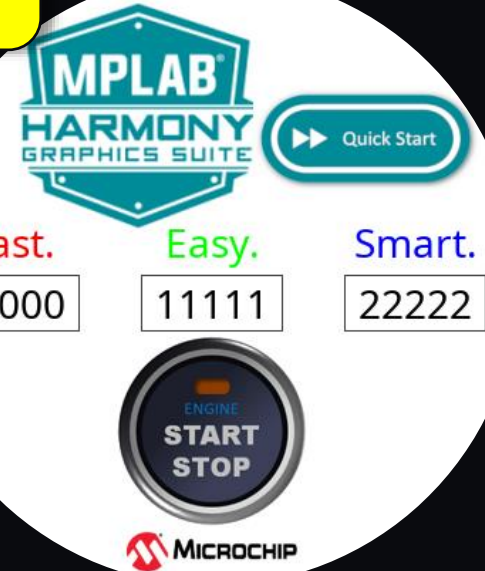
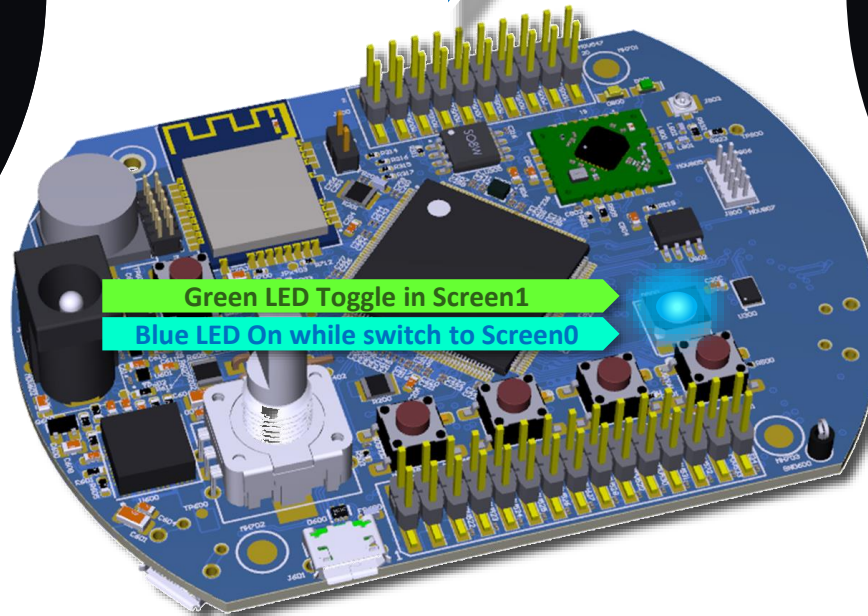
GPU Bit BLIT Stretch Image

- Congratulation!! You finally have all improved animation effect.

Image Scale Up and Move Left, Fad Out
animation speed are all improved !!



Screen1



Screen0

GPU Bit BLIT Stretch Image

Discuss

- How about if disable the Double Buffering in Harmony ?

The screenshot displays the Harmony IDE interface for configuring the LE GLCD component. The left pane shows a block diagram with the following components and connections:

- Legato Graphics w/ LCF0300633GGU00 Display (top)
- LE GLCD (LE GLCD Display Controller) - **Graphics Display** - **LE Display Driver** (highlighted with a red box)
- LE 2DGPU (LE 2D-GPU Graphics Processor) - **LE GPU Driver**
- Legato Graphics Library - **LE Display Driver** - **LE GPU Driver**

The right pane shows the configuration settings for the LE GLCD component:

- LE GLCD**
 - Some settings are being managed by the GFX Core
 - Clock Settings**
 - Master Clock Source: REFCLK5
 - Master Clock (Hz): 200,000,000
 - Pixel Clock Divider (1-32): 11
 - Pixel Clock (Hz): 18,181,818
 - Layer Configuration**
 - Enable Layers**
 - Layer 0: ☒
 - Layer 1: ☒
 - Layer 2: ☒
 - Frame Buffer Settings**
 - Frame Buffer Color Mode: RGBA_8888
 - Use Double Buffering?** ☐ (highlighted with a red box)
 - Frame Buffer Memory Mode: DDR
 - Use 8-bit Palette? ☐
 - Other Settings**
 - Update on Vblank: ☒
 - Canvas Mode: ☐
 - Use GPU for Blits? ☐

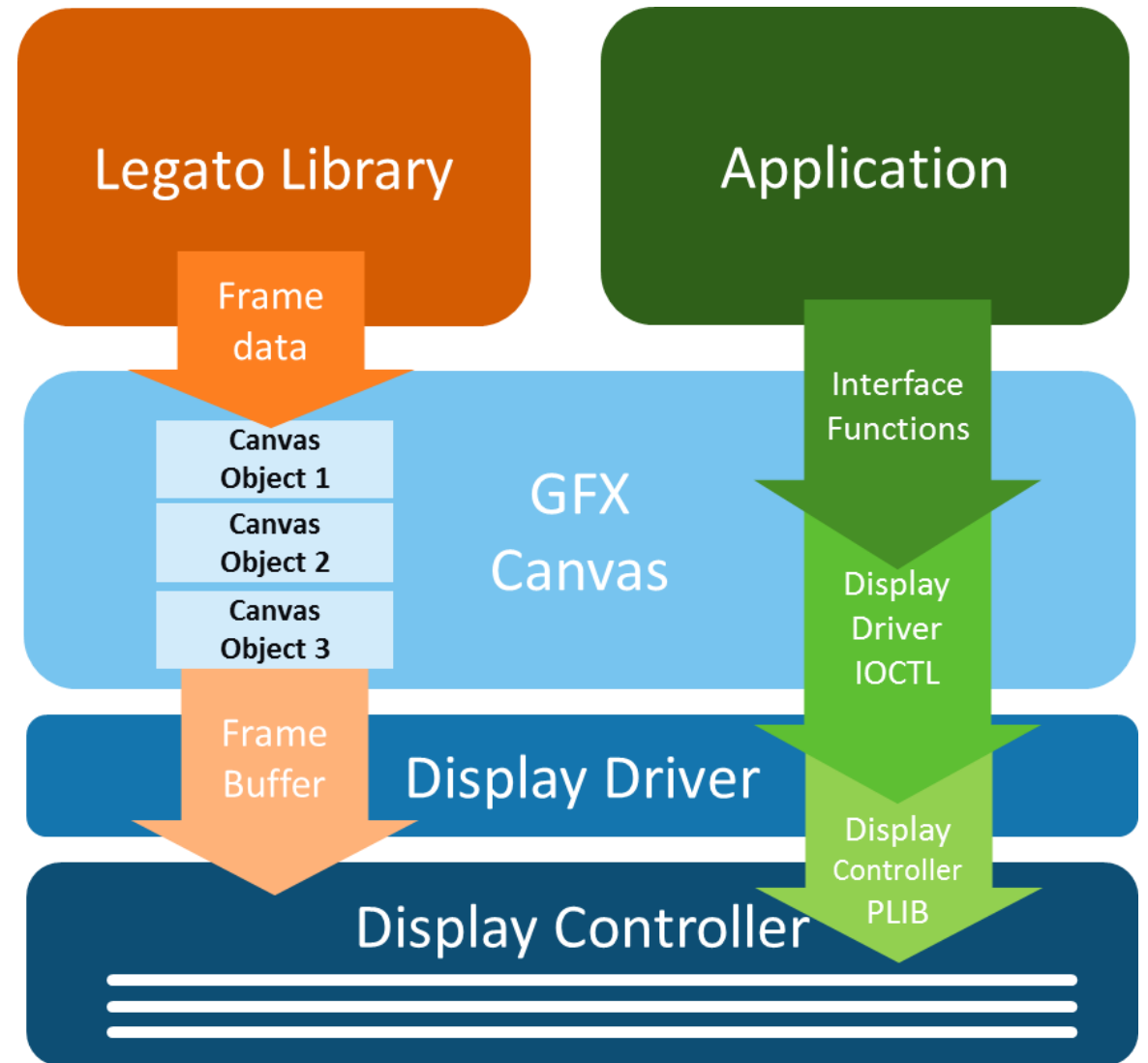
CANVAS Virtual Display

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

CANVAS Virtual Display

- **Canvas is a virtual display for:**
 - Storing or pre-rendering frame data using the Legato Graphics Library.
 - Configuring the Display Controller Layers to display the Frames.
 - Applying effects like Motion, Sprite Animation, Cropping and Fade In/Out using Display Controller Layers.

The Graphics Canvas feature is only supported for devices with internal graphics controller with layer support such as the **PIC32MZ DA** or the **SAM9x60**.



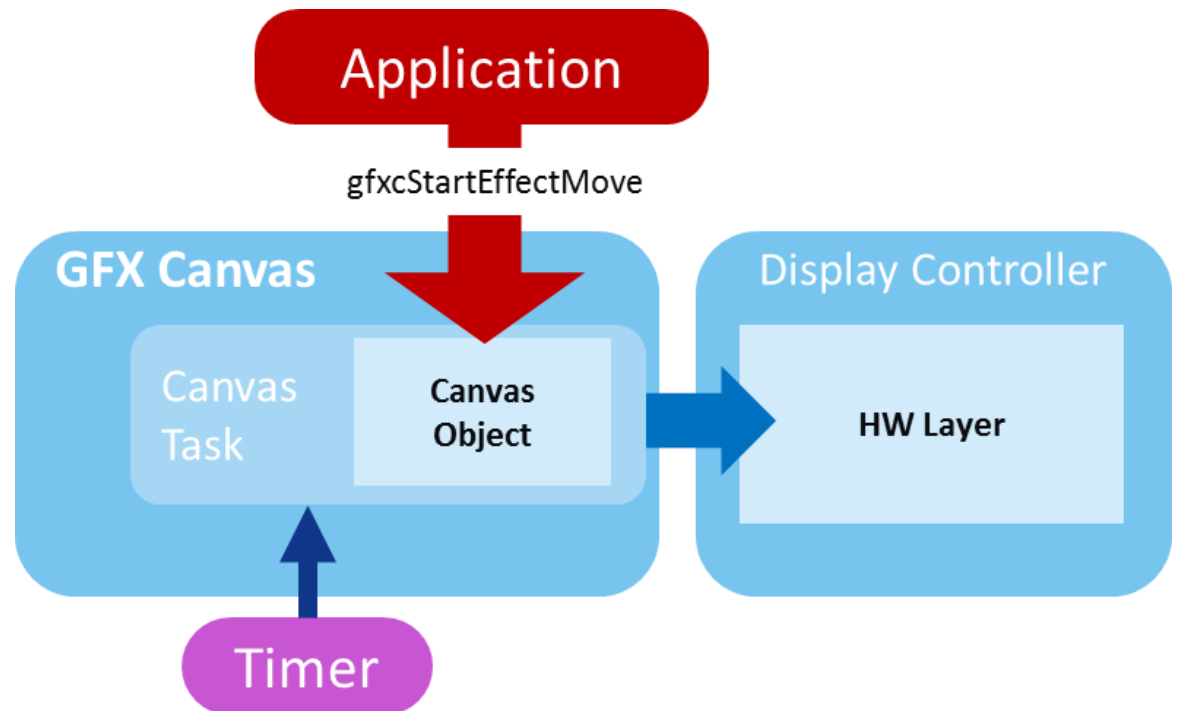
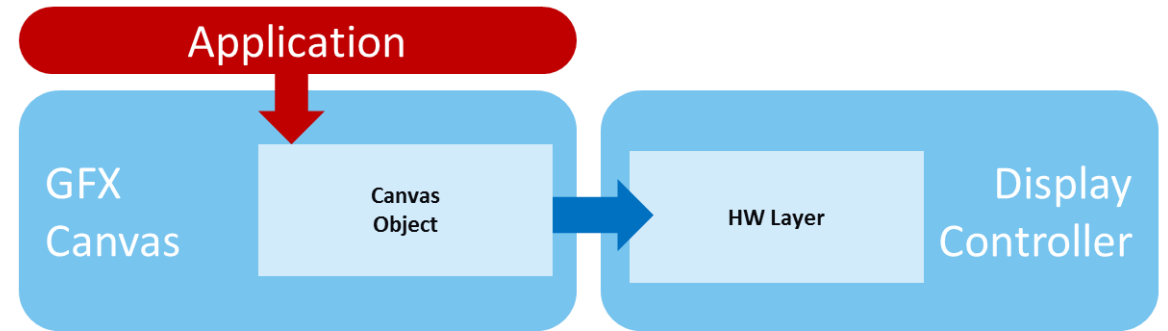
CANVAS Virtual Display

- **Canvas Objects**

- Contains frame information assigned a HW Layer in the Display Controller.
- Buffer address, resolution, color mode.
- Layer ID, size, position, alpha.
- Layer effects status and settings.

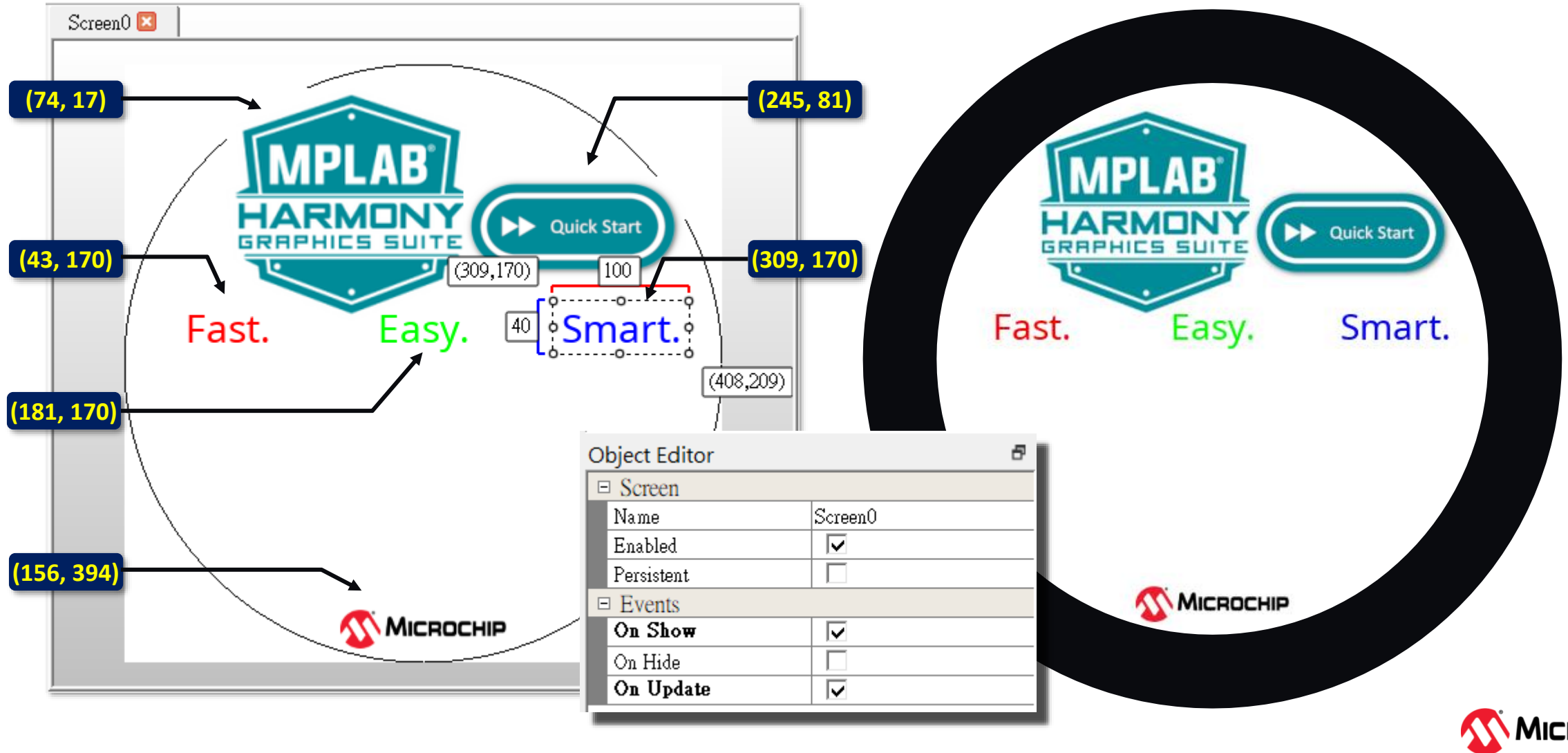
- **Canvas Effects**

- Graphics Canvas can be used to display a frame in various ways – cropped, translated, alpha blended, etc.
- Graphics Canvas has ready-to-use APIs to apply 'Effects' like **Motion** or **Fade In/Out**.



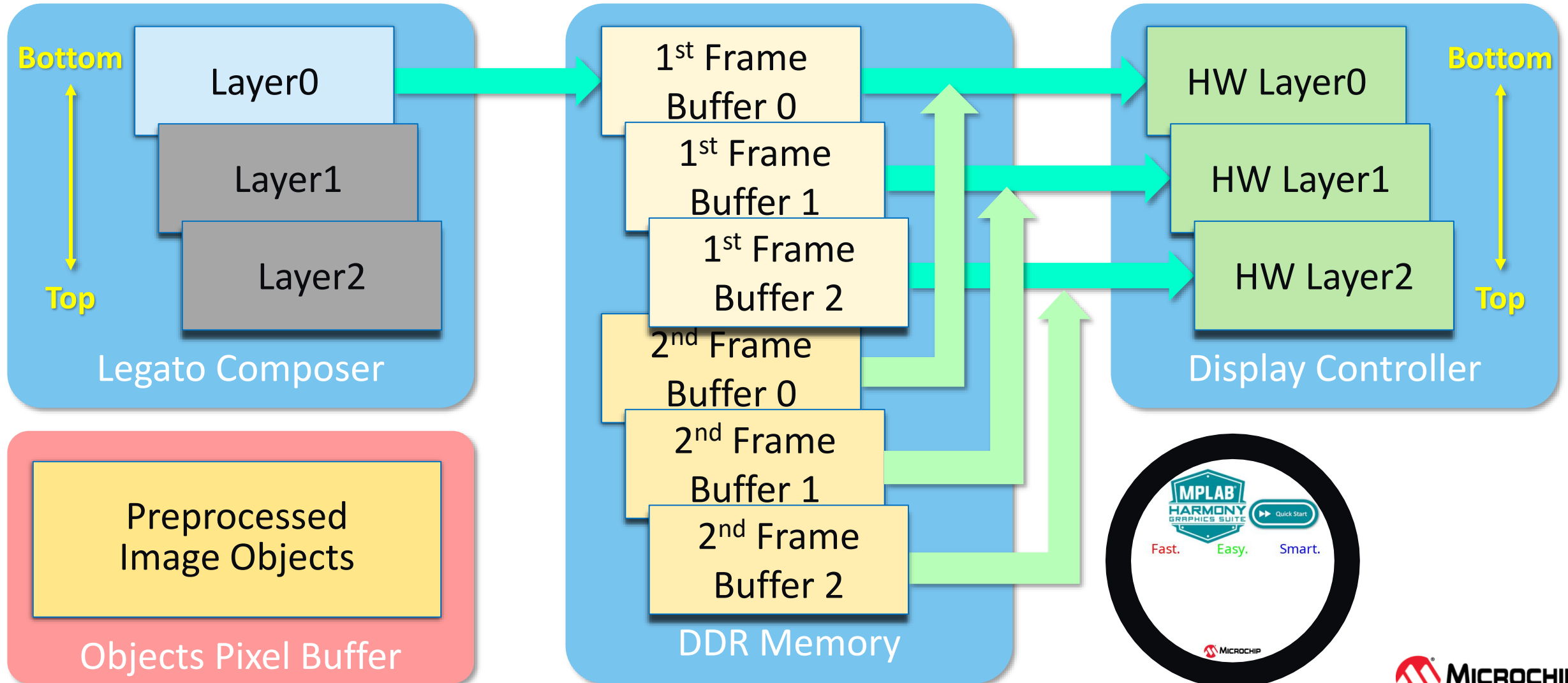
CANVAS Virtual Display

- Create new project and finish to Boundary Circle, add Screen0 OnShow/OnUpdate.



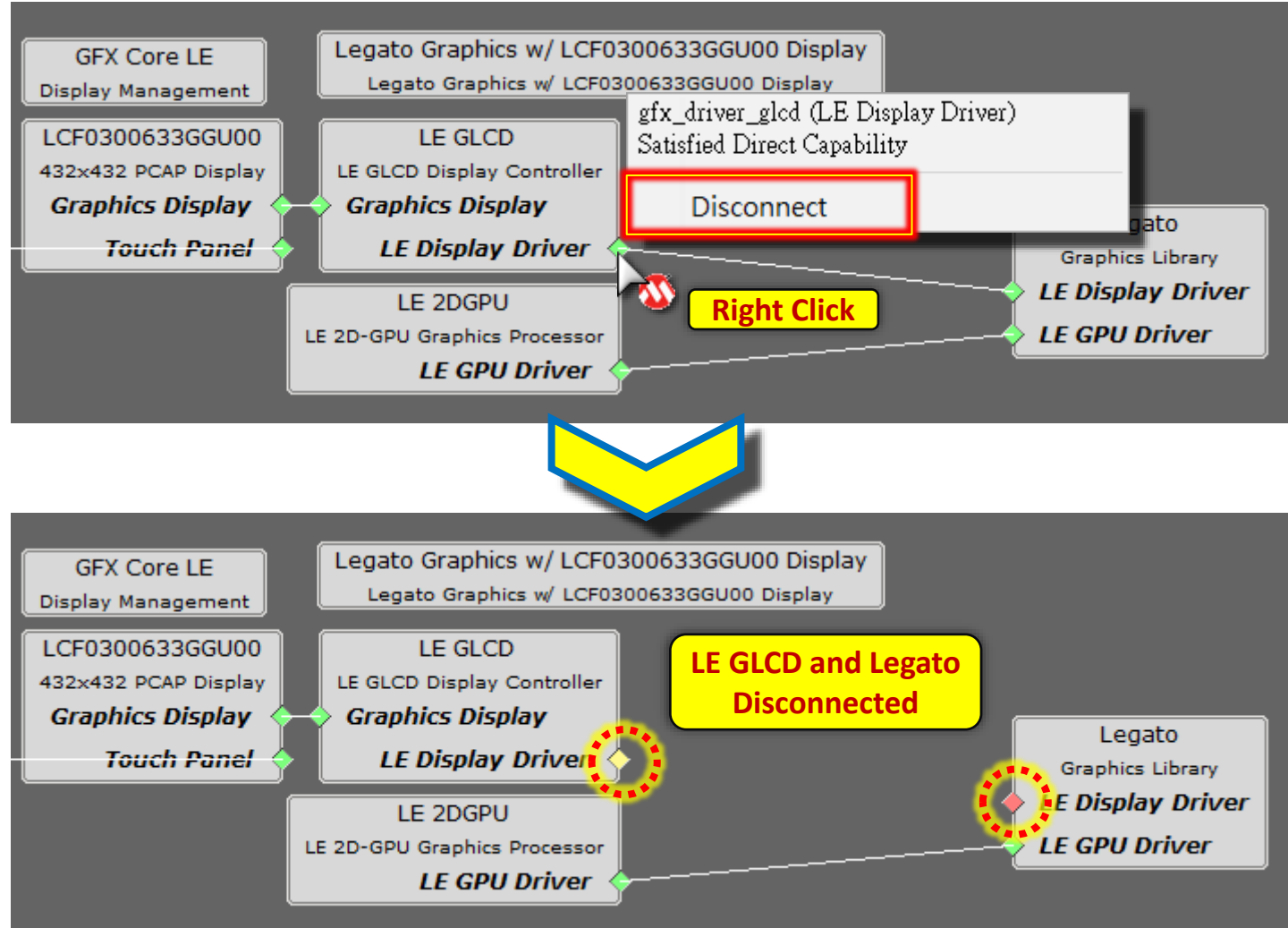
CANVAS Virtual Display

- Legato Graphics Composer Layers and Display HW Layer.



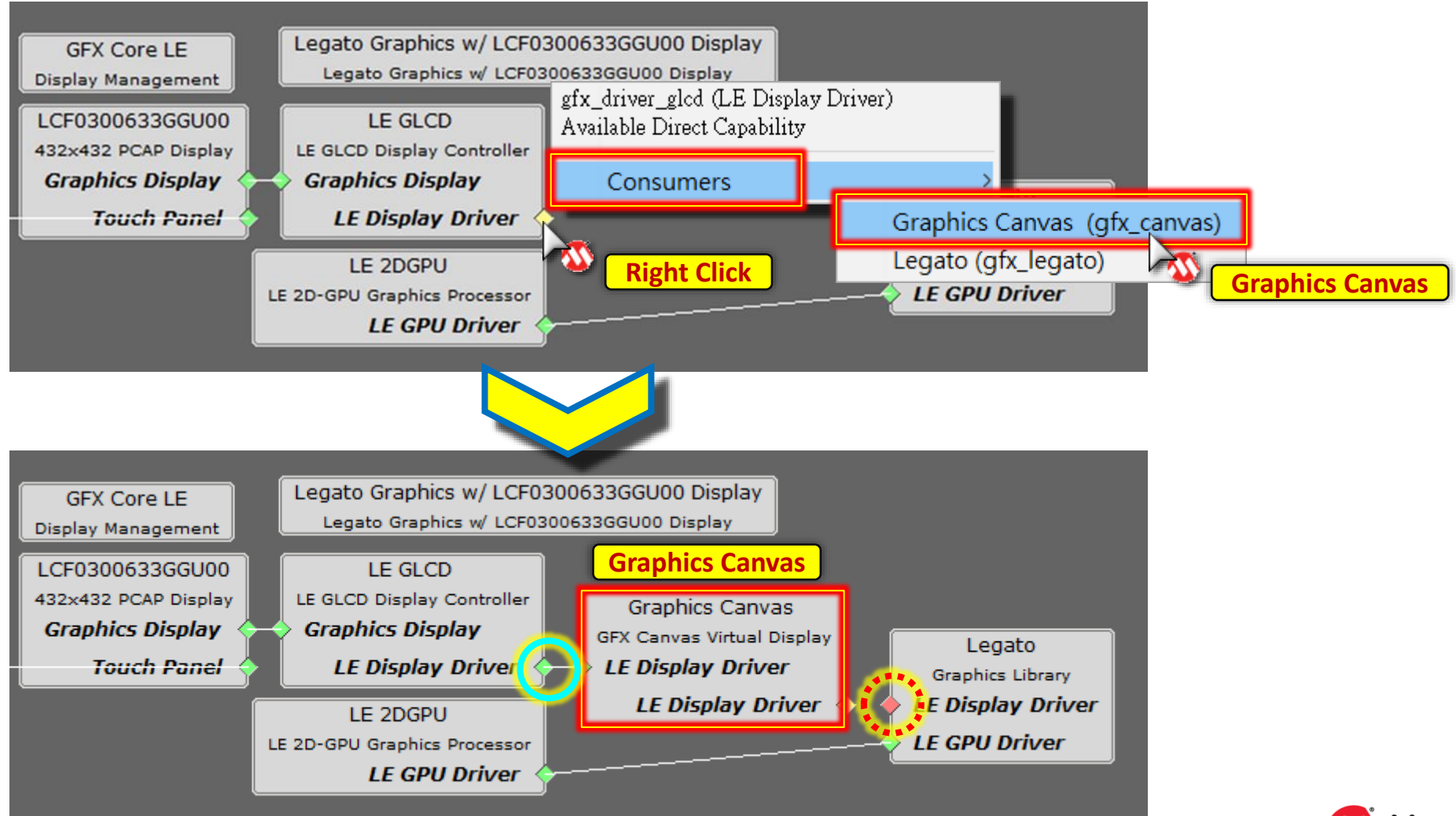
CANVAS Virtual Display

- Disconnected LE GLCD and Legato Module in Harmony



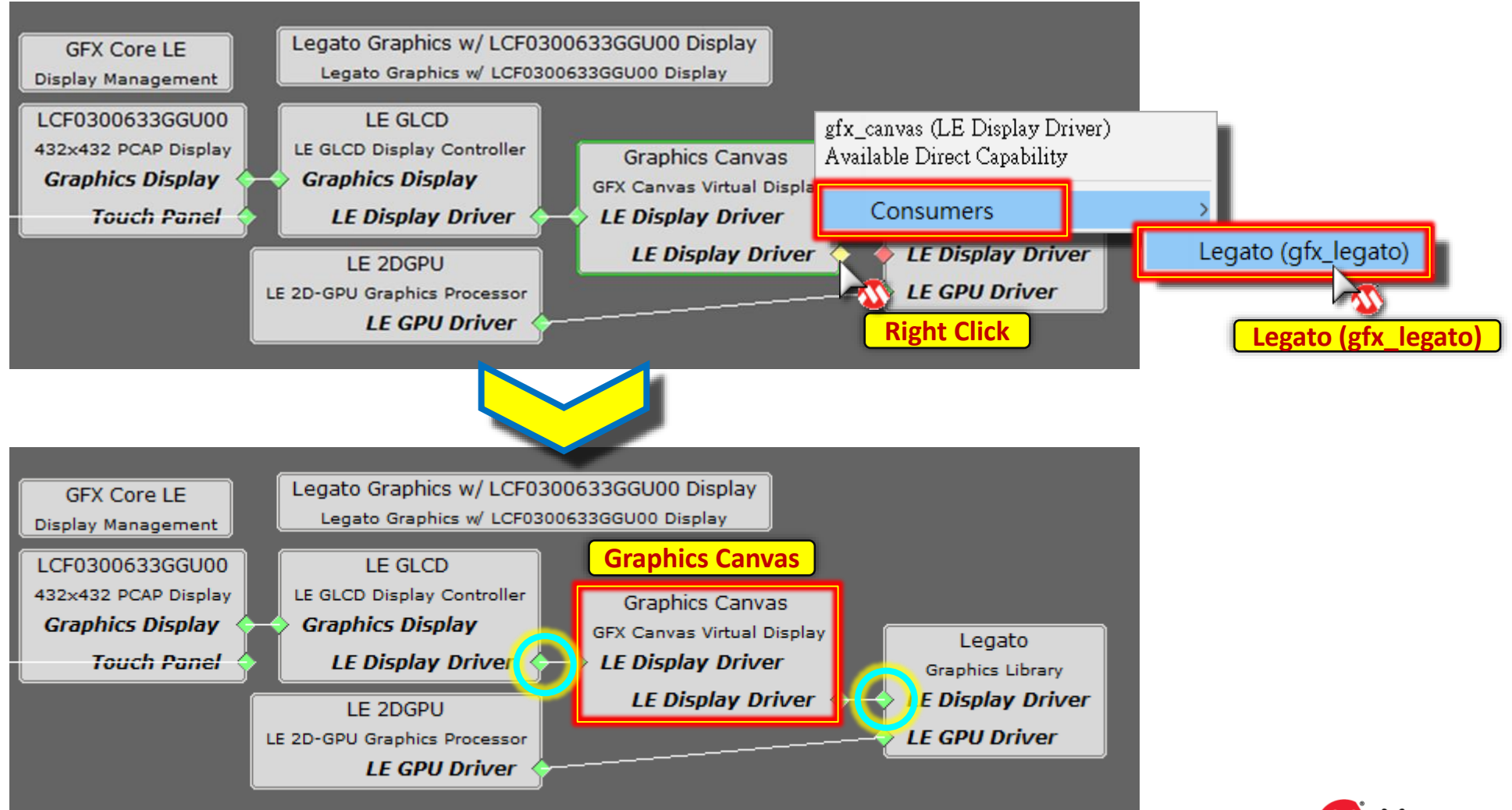
CANVAS Virtual Display

- Insert Graphics Canvas Module in Harmony



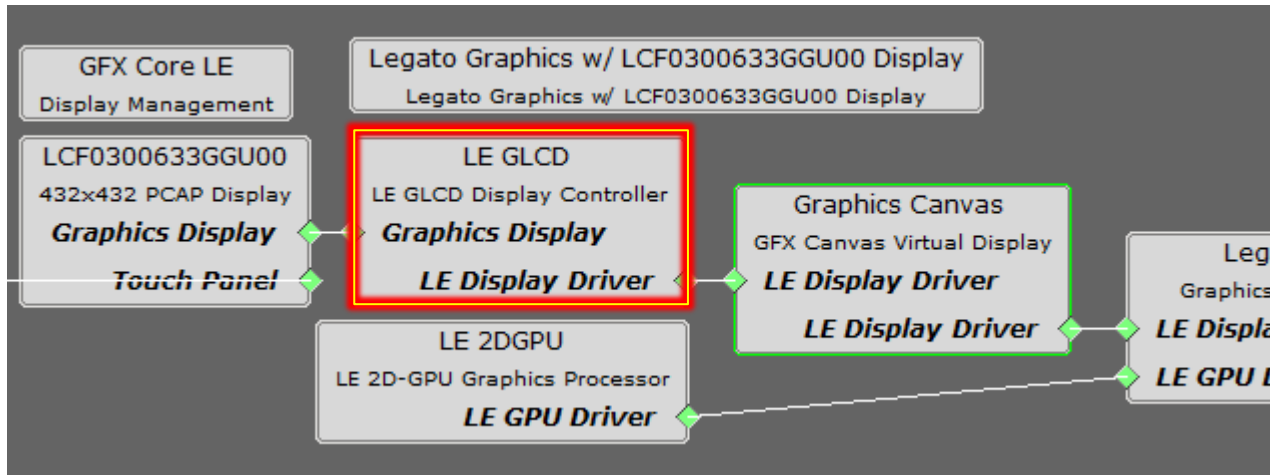
CANVAS Virtual Display

- Complete insert Graphics Canvas between LE GLCD and Legato Module.



CANVAS Virtual Display

- Enable **Canvas Mode** in LE GLCD.



LE GLCD

Some settings are being managed by the GFX Core LE and have been hidden.

Clock Settings

- Master Clock Source: REFCLK5
- Master Clock (Hz): 200,000,000
- Pixel Clock Divider (1-32): 11
- Pixel Clock (Hz): 18,181,818

Layer Configuration

Enable Layers

- Layer 0: ☒
- Layer 1: ☒
- Layer 2: ☒

Frame Buffer Settings

- Frame Buffer Color Mode: RGBA_8888
- Use Double Buffering?: ☒
- Frame Buffer Memory Mode: DDR
- Use 8-bit Palette?: ☐

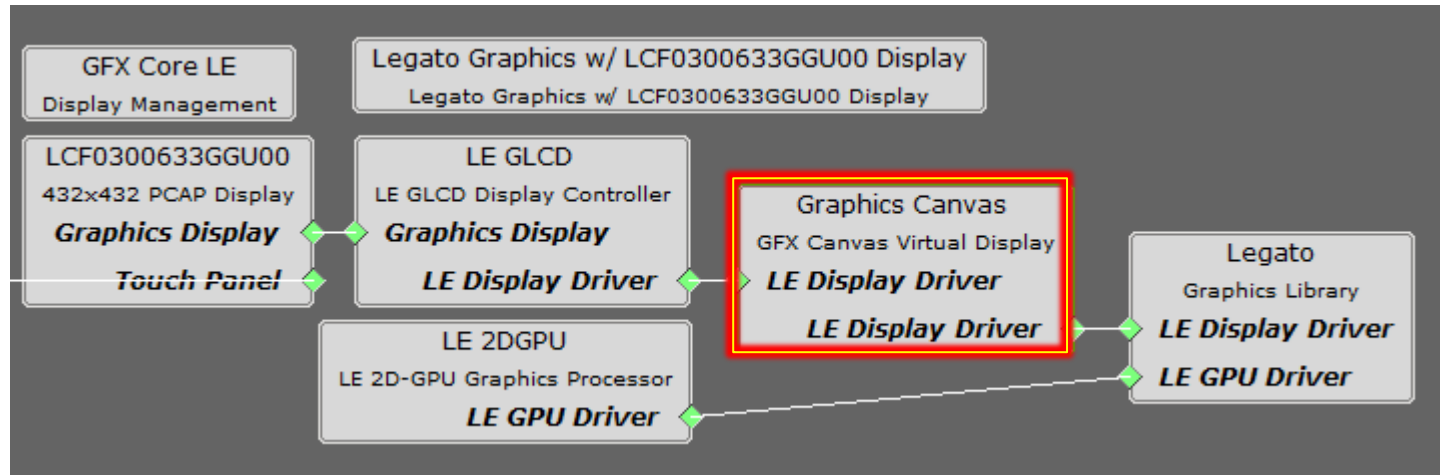
Other Settings

- Update on Vblank: ☒
- Canvas Mode: ☒**
- Use GPU for Blits?: ☐

Enable Canvas Mode

CANVAS Virtual Display

- Configure **Graphics Canvas** as below.



Objects	Width	Height	Allocation	Address
Canvas[0]	432	432	Manual	A8000000
Canvas[1]	432	432	Manual	A8177000
Canvas[2]	432	432	Manual	A82EE000

Graphics Canvas

- Canvas Settings**
 - Enable Window Clipping ☒
 - Default Color Mode **RGBA_8888**
 - Default Number of Layers **3**
 - Max Number of Parameters **20**
- Canvas Objects**
 - Canvas[0]**
 - Width **432**
 - Height **432**
 - Color Mode **RGBA_8888**
 - Frame Buffer Allocation **Manual**
 - Address **0x A8000000**
 - Canvas[1]**
 - Width **432**
 - Height **432**
 - Color Mode **RGBA_8888**
 - Frame Buffer Allocation **Manual**
 - Address **0x A8177000**
 - Canvas[2]**
 - Width **432**
 - Height **432**
 - Color Mode **RGBA_8888**
 - Frame Buffer Allocation **Manual**
 - Address **0x A82EE000**
- Effects Settings** ☒
 - Effects Interval (ms) **40**
 - Enable Fade Effects ☒
 - Enable Move Effects ☒

CANVAS Virtual Display

- Canvas API functions

gfxcSetPixelBuffer(unsigned int canvasID, unsigned int width, unsigned int height, const GFXC_COLOR_FORMAT format, void * buf)

gfxcCopyBuffer(unsigned int srcID, unsigned int destID, const gfxRect * srcRect, const gfxRect * destRect)

gfxcSetBaseCanvasID(unsigned int base)

gfxcSetLayer(unsigned int canvasID, unsigned int layerID)

gfxcGetlayer(unsigned int canvasID)

gfxcClearLayer(unsigned int canvasID)

gfxcSetWindowAlpha(unsigned int canvasID, uint8_t alpha)

gfxcSetWindowPosition(unsigned int canvasID, int xpos, int ypos)

gfxcGetWindowPosition(unsigned int canvasID, int* xpos, int* ypos)

gfxcSetWindowSize(unsigned int canvasID, unsigned int width, unsigned int height)

gfxcGetWindowSize(unsigned int canvasID, unsigned int* width, unsigned int* height)

gfxcShowCanvas(unsigned int canvasID)

gfxcHideCanvas(unsigned int canvasID)

gfxcCanvasUpdate(unsigned int canvasID)

gfxcStartEffectMove(unsigned int canvasID, GFXC_FX_MOVE_TYPE type, int startX, int startY, int endX, int endY, unsigned int delta)

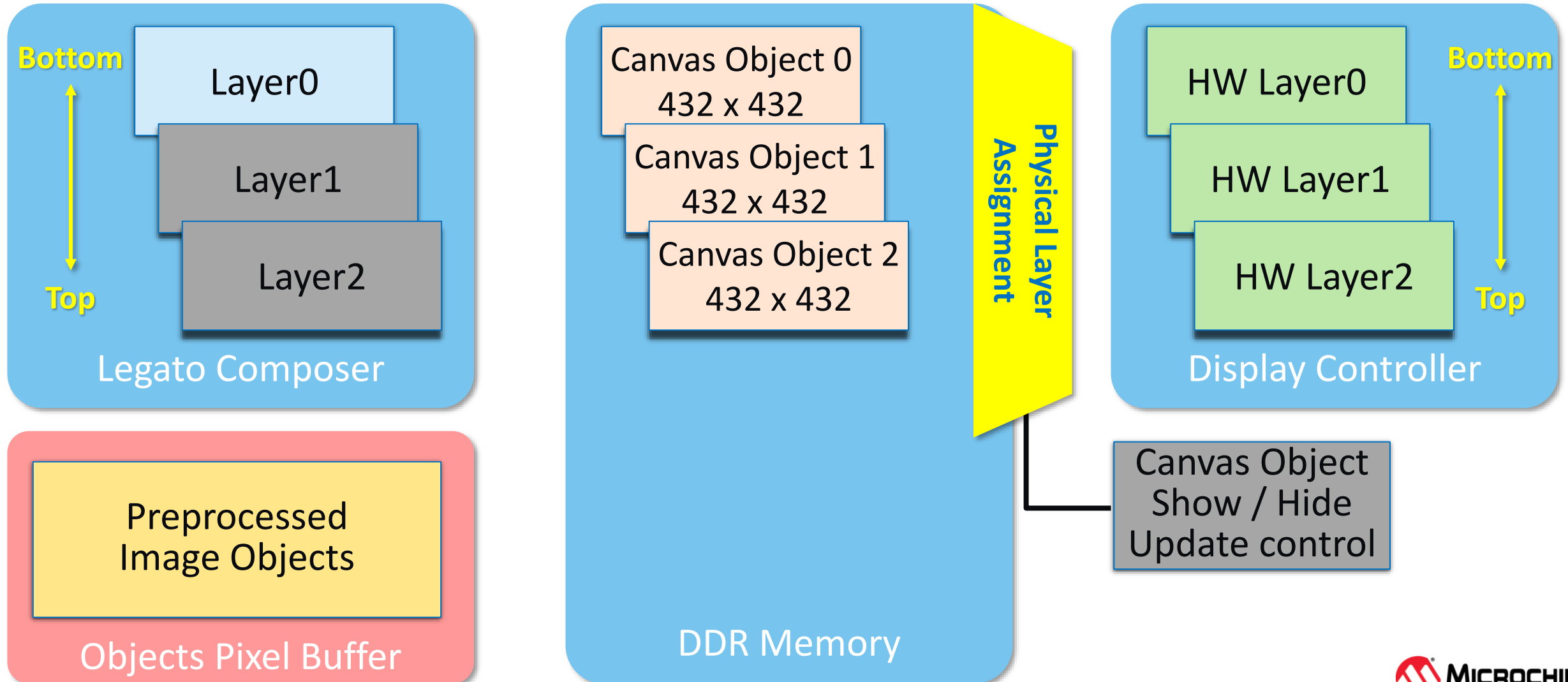
gfxcStartEffectFade(unsigned int canvasID, uint8_t startAlpha, uint8_t endAlpha, uint8_t delta)

gfxcSetEffectsCallback(unsigned int canvasID, gfxcCallback cb, void * parm)

gfxcStopEffect(unsigned int canvasID, GFXC_FX_TYPE effect)

CANVAS Virtual Display

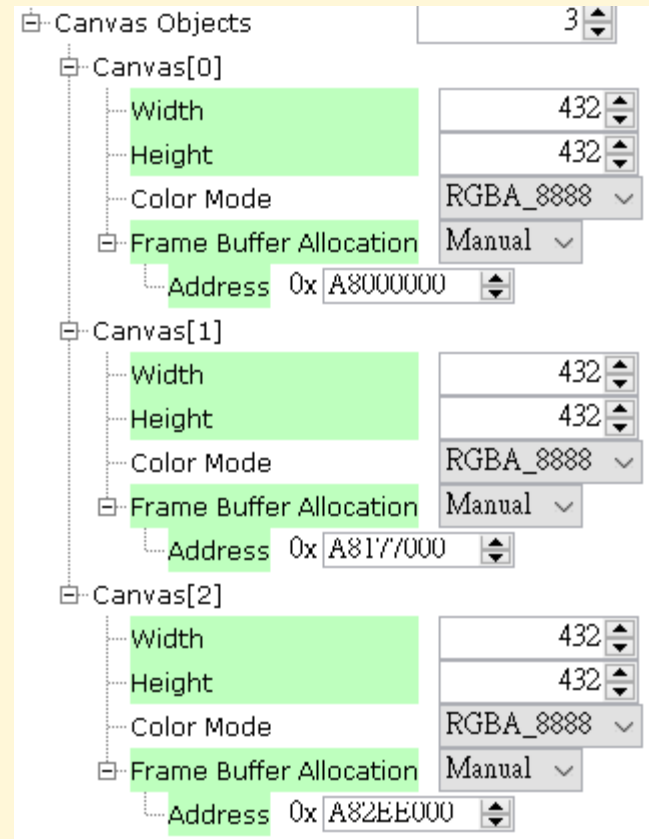
- Canvas Virtual Display feature enabled.



CANVAS Virtual Display

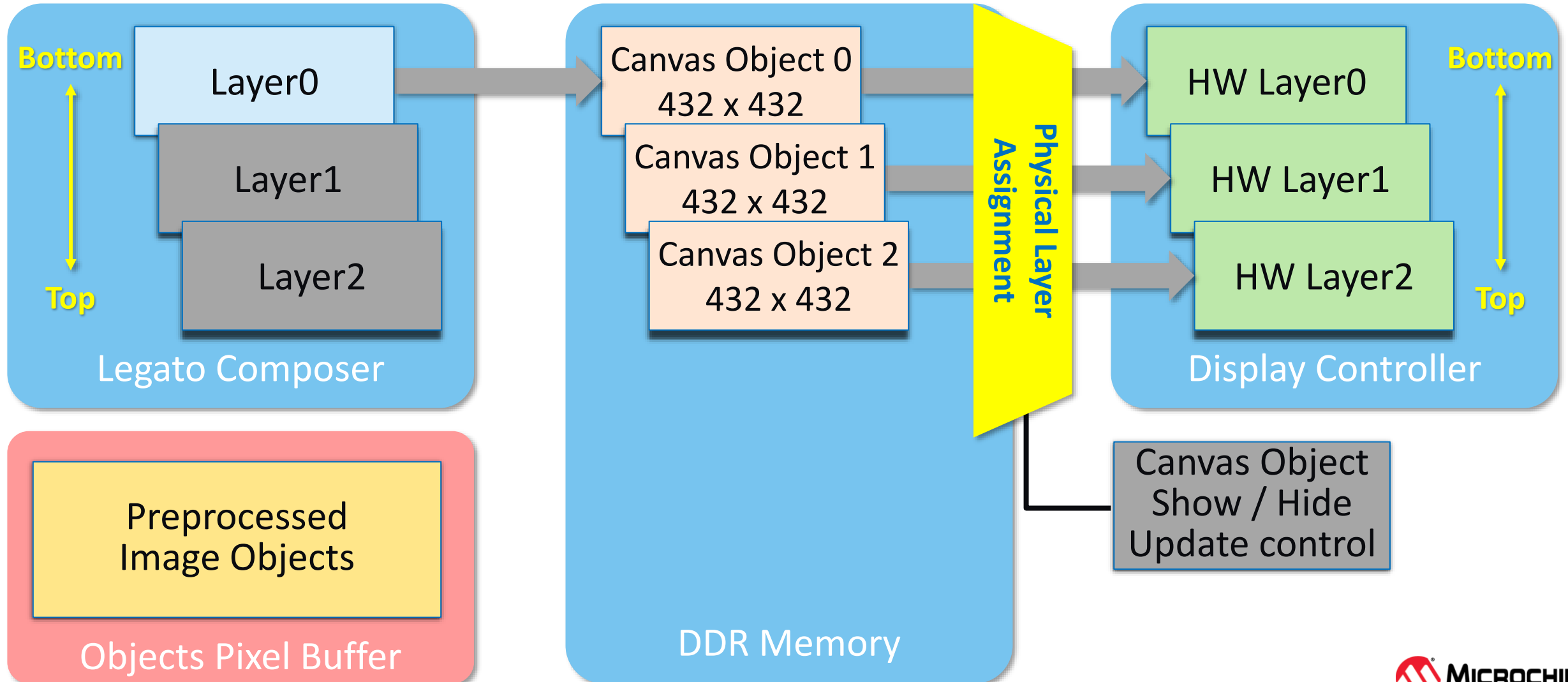
- Canvas Virtual Display Objects connect to Display HW Layer.

```
enum { CANVAS_ID_0, CANVAS_ID_1, CANVAS_ID_2 };  
enum { DSP_LAYER_0, DSP_LAYER_1, DSP_LAYER_2 };  
  
void Screen0_OnShow( void )  
{  
    memset( (char*)0xA8000000, 0, 432*432*4 );  
    memset( (char*)0xA8177000, 0, 432*432*4 );  
    memset( (char*)0xA82EE000, 0, 432*432*4 );  
    gfxcSetLayer( CANVAS_ID_0, DSP_LAYER_0 );  
    gfxcSetLayer( CANVAS_ID_1, DSP_LAYER_1 );  
    gfxcSetLayer( CANVAS_ID_2, DSP_LAYER_2 );  
}  
  
void Screen0_OnUpdate( void )  
{  
  
}  
  
void event_Screen0_qsButton_OnPressed(leButtonWidget* btn) { ... }  
void event_Screen0_qsButton_OnReleased(leButtonWidget* btn) { ... }  
  
int main ( void )  
{ ... }
```



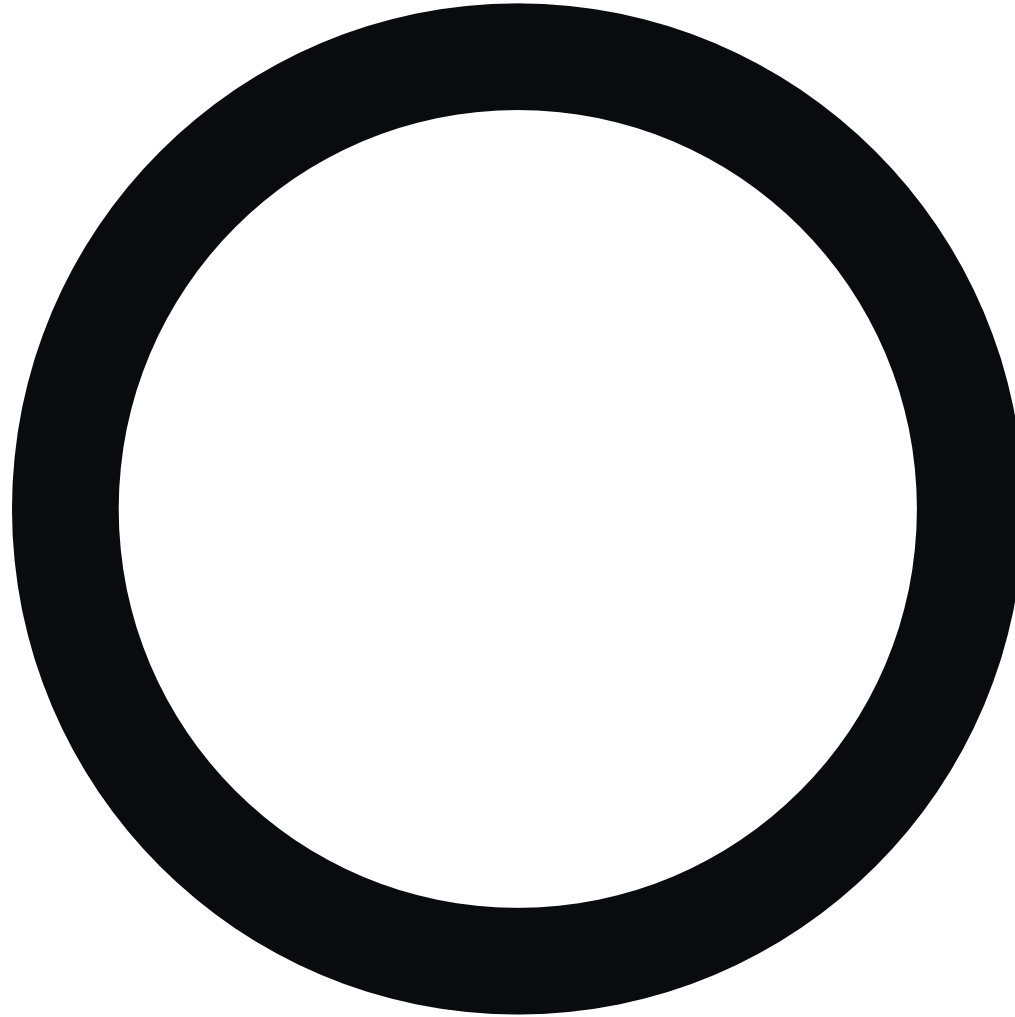
CANVAS Virtual Display

- After Canvas Virtual Display Objects connect to Display HW Layer.



CANVAS Virtual Display

- Legato Graphics Composer Layers cannot output.



CANVAS Virtual Display

- Show all Canvas Objects and Update all.

```
enum { CANVAS_ID_0, CANVAS_ID_1, CANVAS_ID_2 };  
enum { DSP_LAYER_0, DSP_LAYER_1, DSP_LAYER_2 };
```

CANVAS ID

Display HW Layer ID

```
void Screen0_OnShow( void )
```

```
{  
    memset( (char*)0xA8000000, 0, 432*432*4 );  
    memset( (char*)0xA8177000, 0, 432*432*4 );  
    memset( (char*)0xA82EE000, 0, 432*432*4 );  
    gfxcSetLayer( CANVAS_ID_0, DSP_LAYER_0 );  
    gfxcSetLayer( CANVAS_ID_1, DSP_LAYER_1 );  
    gfxcSetLayer( CANVAS_ID_2, DSP_LAYER_2 );  
    gfxcShowCanvas( CANVAS_ID_0 );  
    gfxcCanvasUpdate( CANVAS_ID_0 );  
    gfxcShowCanvas( CANVAS_ID_1 );  
    gfxcCanvasUpdate( CANVAS_ID_1 );  
    gfxcShowCanvas( CANVAS_ID_2 );  
    gfxcCanvasUpdate( CANVAS_ID_2 );  
}
```

Show Canvas 0 and Update

Show Canvas 1 and Update

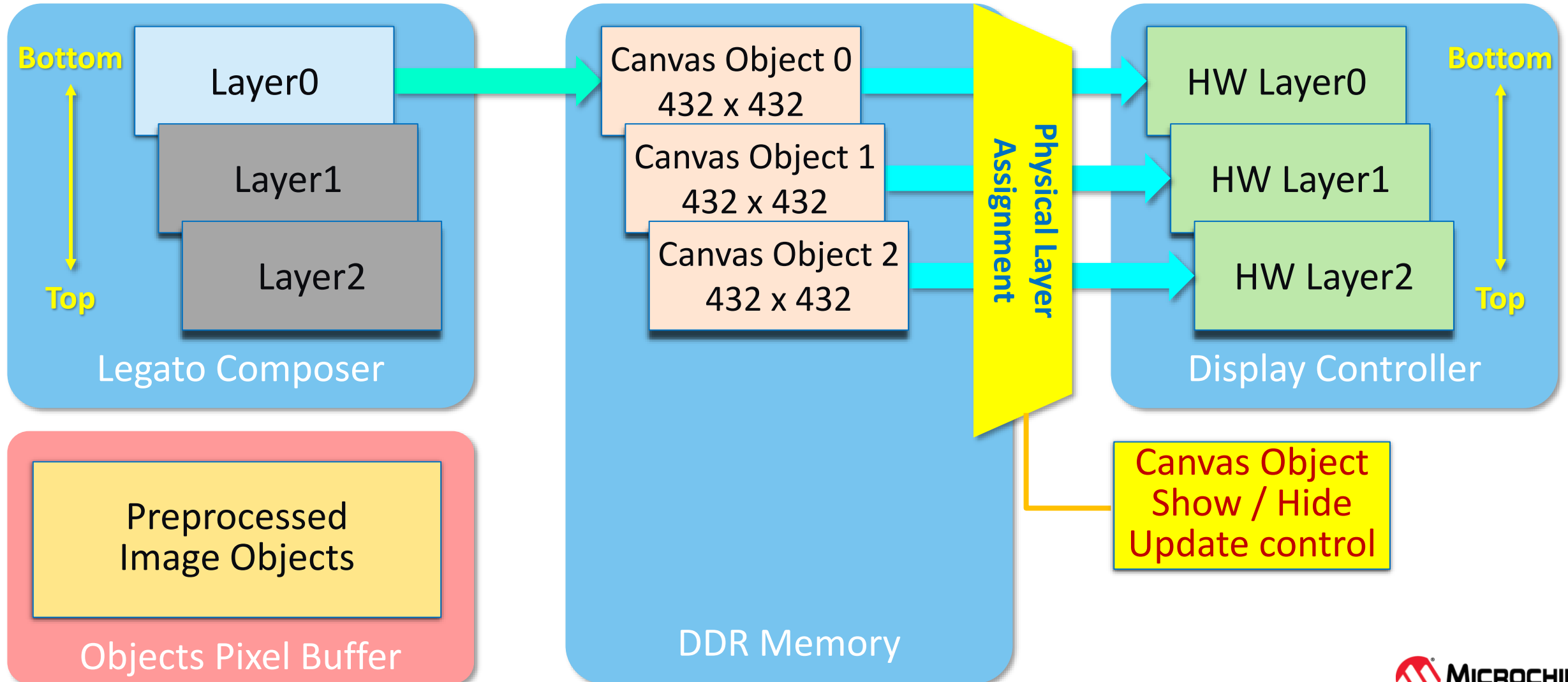
Show Canvas 2 and Update

```
void Screen0_OnUpdate( void )
```

```
{  
  
}
```

CANVAS Virtual Display

- Legato Composer could output Layers after show all Canvas Objects.



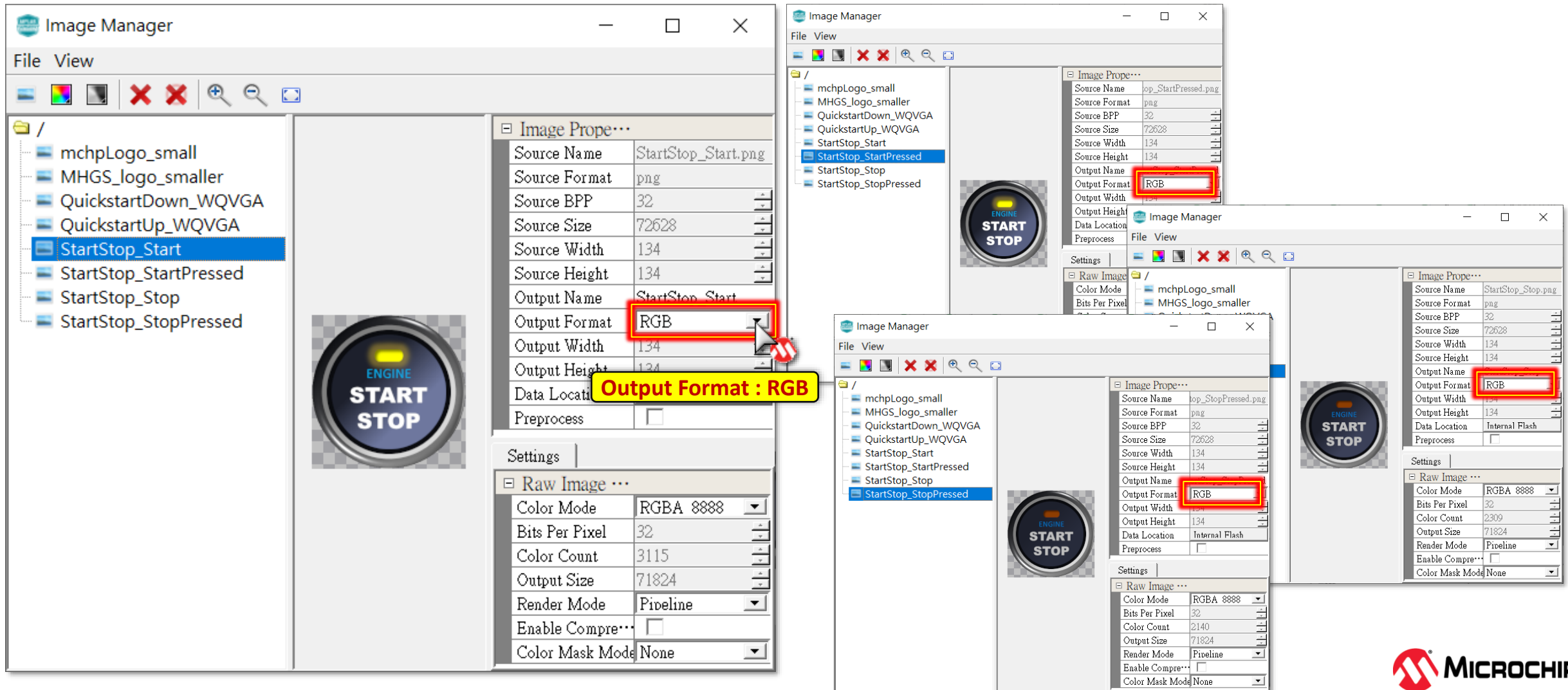
CANVAS Virtual Display

- Legato Graphics Composer Layers can output now.



CANVAS Virtual Display

- Add four Start Stop Button Images in Legato Graphics Composer.



CANVAS Virtual Display

- Implement Image **Preprocess** function to put all Image in **DDR**.

```
#define DDR_PREPROCESS_ADDR 0xA88CA000
void legato_preprocessImages_runtime(void)
{
    size_t AddressDDR = DDR_PREPROCESS_ADDR;

    leImage *pPreProcessImages[] = { &mchpLogo_small, &MHGS_logo_smaller,
                                      &QuickstartDown_WQVGA, &QuickstartUp_WQVGA,
                                      &StartStop_Start, &StartStop_StartPressed,
                                      &StartStop_Stop, &StartStop_StopPressed };

    for( int i=0 ; i<sizeof(pPreProcessImages)/sizeof(leImage*) ; i++ )
    {
        leProcessImage(pPreProcessImages[i], AddressDDR, pPreProcessImages[i]->buffer.mode);
        AddressDDR = AddressDDR + (pPreProcessImages[i]->buffer.buffer_length);
        AddressDDR = AddressDDR + (AddressDDR%16?16-(AddressDDR%16):0);
    }
}

enum { CANVAS_ID_0, CANVAS_ID_1, CANVAS_ID_2 };
enum { DSP_LAYER_0, DSP_LAYER_1, DSP_LAYER_2 };

void Screen0_OnShow( void )
{
    legato_preprocessImages_runtime();

    memset( (char*)0xA8000000, 0, 432*432*4 );
```

CANVAS Virtual Display

- Implement **leImageToCanvas** function to assign **leImage** object to **Canvas**.

```
#define DDR_PREPROCESS_ADDR 0xA88CA000
void legato_preprocessImages_runtime(void)
{
    ...
}

void leImageToCanvas( uint32_t canvasID, leImage *Image, int32_t X, int32_t Y, uint8_t Alpha )
{
    int32_t W, H;
    leBuffer *PixelBuffer;

    W = Image->buffer.size.width;
    H = Image->buffer.size.height;
    PixelBuffer = Image->buffer.pixels;
    gfxcSetWindowSize( canvasID, W, H );
    gfxcSetPixelBuffer( canvasID, W, H, GFX_COLOR_MODE_RGBA_8888, PixelBuffer );
    gfxcSetWindowPosition( canvasID, X, Y );
    gfxcSetWindowAlpha( canvasID, Alpha );
    gfxcShowCanvas( canvasID );
    gfxcCanvasUpdate( canvasID );
}

enum { CANVAS_ID_0, CANVAS_ID_1, CANVAS_ID_2 };
enum { DSP_LAYER_0, DSP_LAYER_1, DSP_LAYER_2 };
```

CANVAS Virtual Display

- Output **StartStop_Stop** Image to **Canvas 0** in end of **Screen0_OnShow()**.

```
#define DDR_PREPROCESS_ADDR 0xA88CA000
void legato_preprocessImages_runtime(void)
{
    ...
}

void lImageToCanvas( uint32_t canvasID, lImage *Image, int32_t X, int32_t Y, uint8_t Alpha )
{
    ...
}

enum { CANVAS_ID_0, CANVAS_ID_1, CANVAS_ID_2 };
enum { DSP_LAYER_0, DSP_LAYER_1, DSP_LAYER_2 };

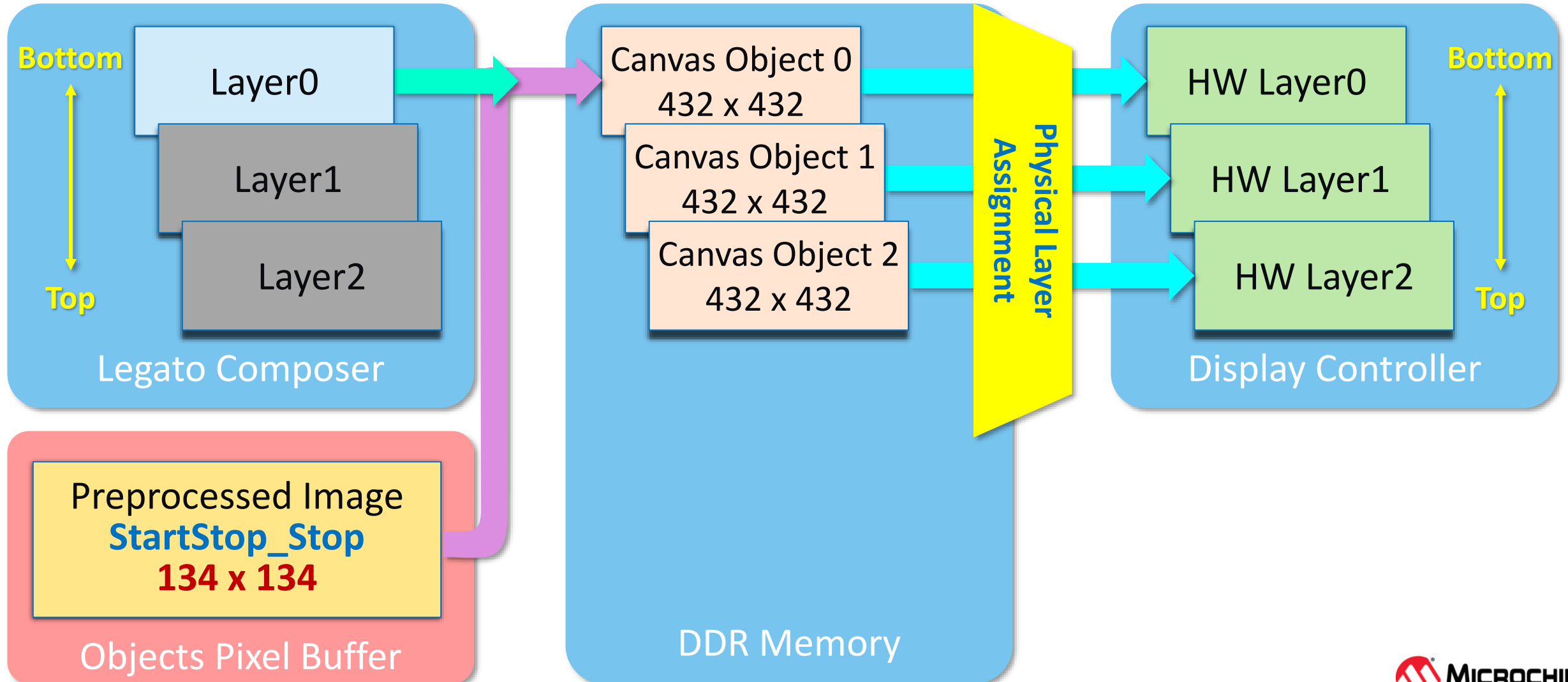
void Screen0_OnShow( void )
{
    legato_preprocessImages_runtime();

    //memset( (char*)0xA8000000, 0, 432*432*4 );
    memset( (char*)0xA8177000, 0, 432*432*4 );
    memset( (char*)0xA82EE000, 0, 432*432*4 );
    ...

    lImageToCanvas( CANVAS_ID_0, &StartStop_Stop, 149, 256, 255 );
}
```

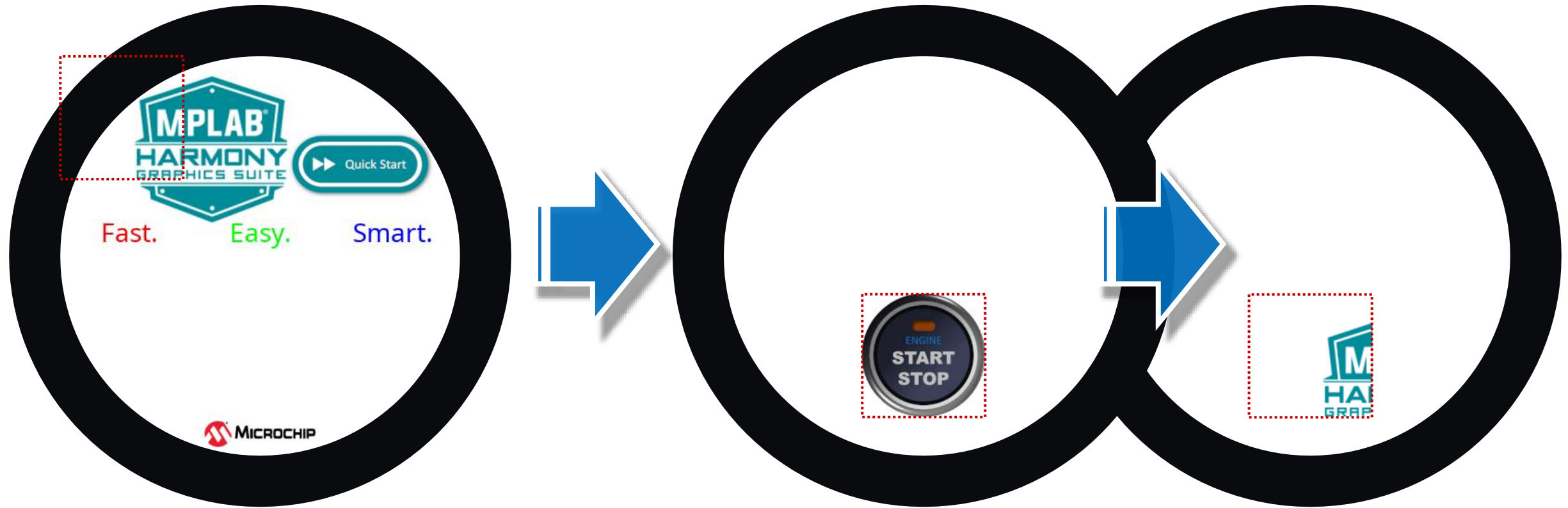
CANVAS Virtual Display

- The Pixel Buffer of **StartStop_Stop** Image will update to Canvas **0** object.



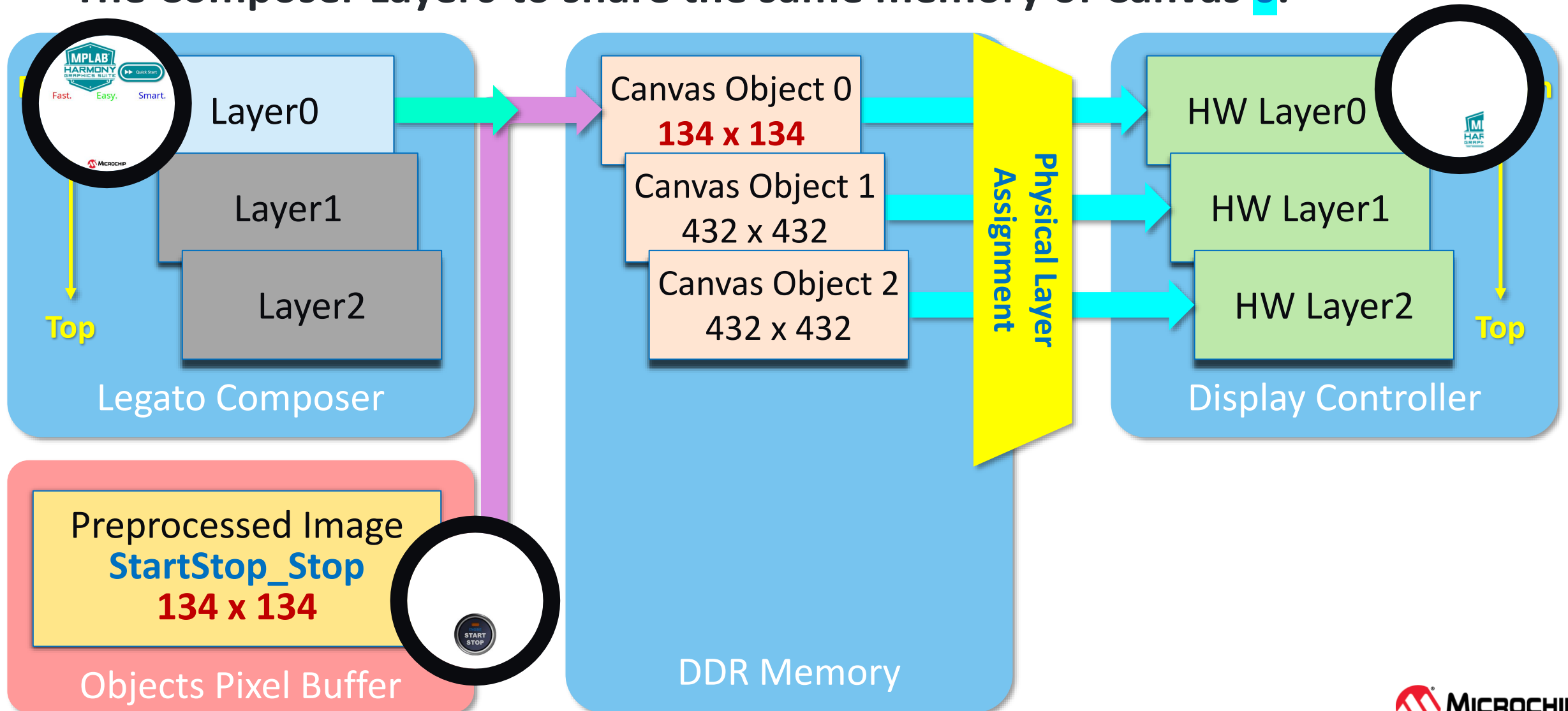
CANVAS Virtual Display

- Result seems bad, why ?



CANVAS Virtual Display

- The Composer Layer0 to share the same memory of Canvas 0.



CANVAS Virtual Display

- Please put **StartStop_Stop** Image to **Canvas 1** to separate from **Layer0**.

```
#define DDR_PREPROCESS_ADDR 0xA88CA000
void legato_preprocessImages_runtime(void)
{
    ...
}

void lImageToCanvas( uint32_t canvasID, lImage *Image, int32_t X, int32_t Y, uint8_t Alpha )
{
    ...
}

enum { CANVAS_ID_0, CANVAS_ID_1, CANVAS_ID_2 };
enum { DSP_LAYER_0, DSP_LAYER_1, DSP_LAYER_2 };

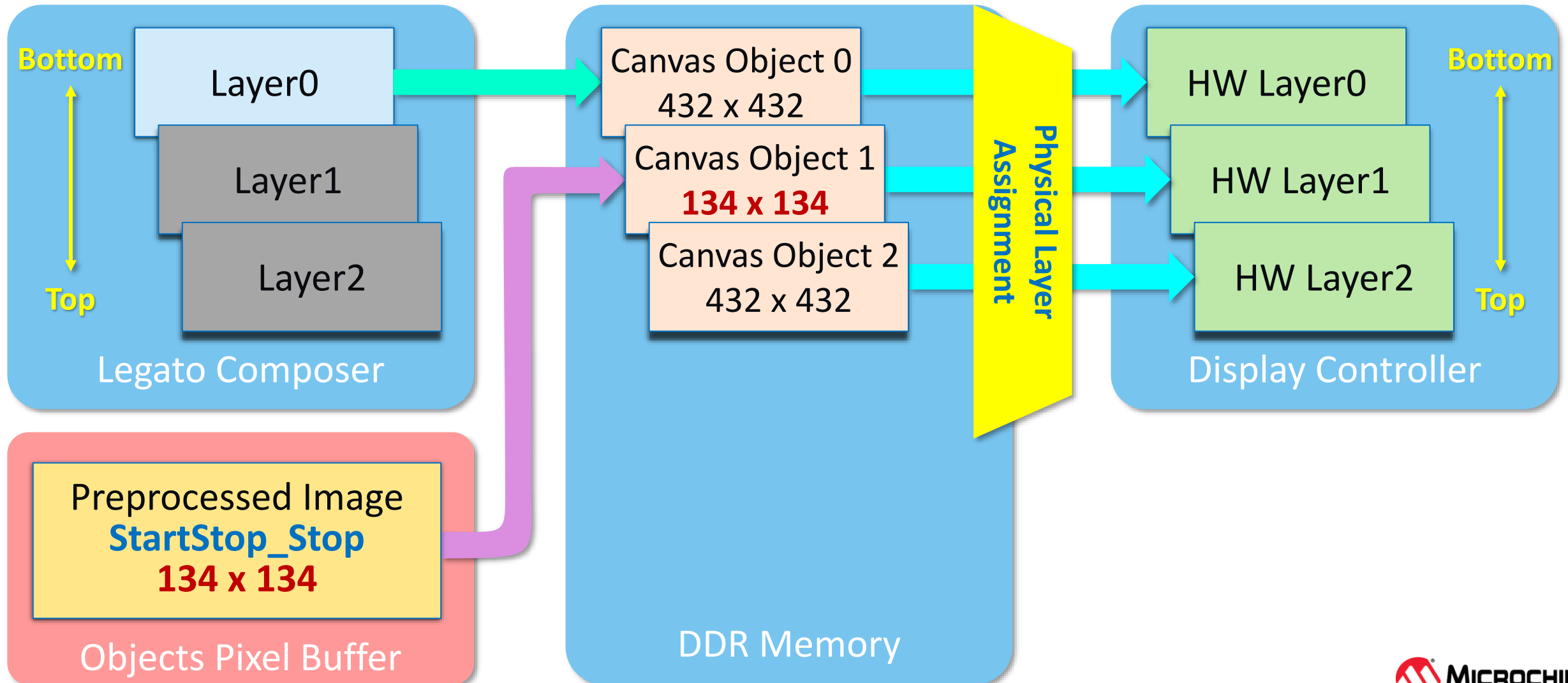
void Screen0_OnShow( void )
{
    legato_preprocessImages_runtime();

    // memset( (char*)0xA8000000, 0, 432*432*4 );
    memset( (char*)0xA8177000, 0, 432*432*4 );
    memset( (char*)0xA82EE000, 0, 432*432*4 );
    ...

    lImageToCanvas( CANVAS_ID_1, &StartStop_Stop, 149, 256, 255 );
}
```

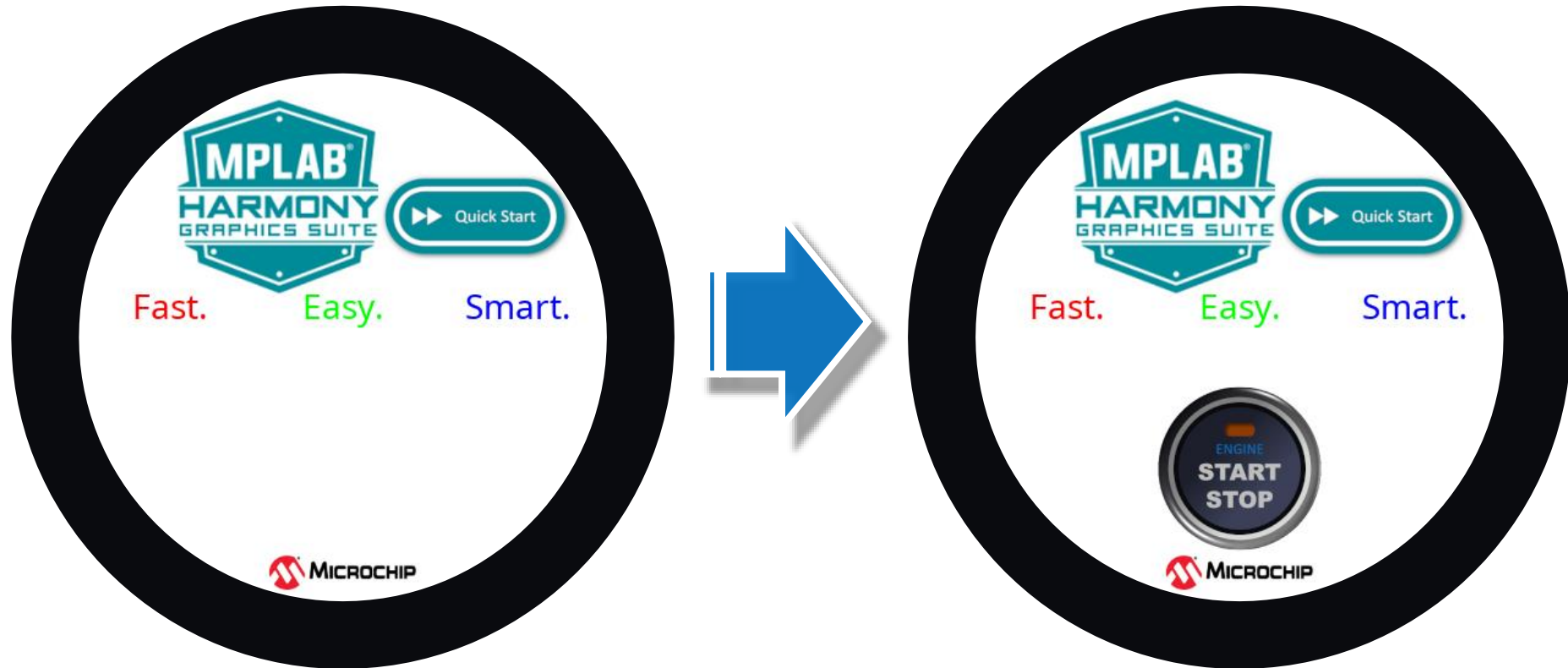
CANVAS Virtual Display

- The Pixel Buffer of **StartStop_Stop** Image now update to Canvas **1** object.



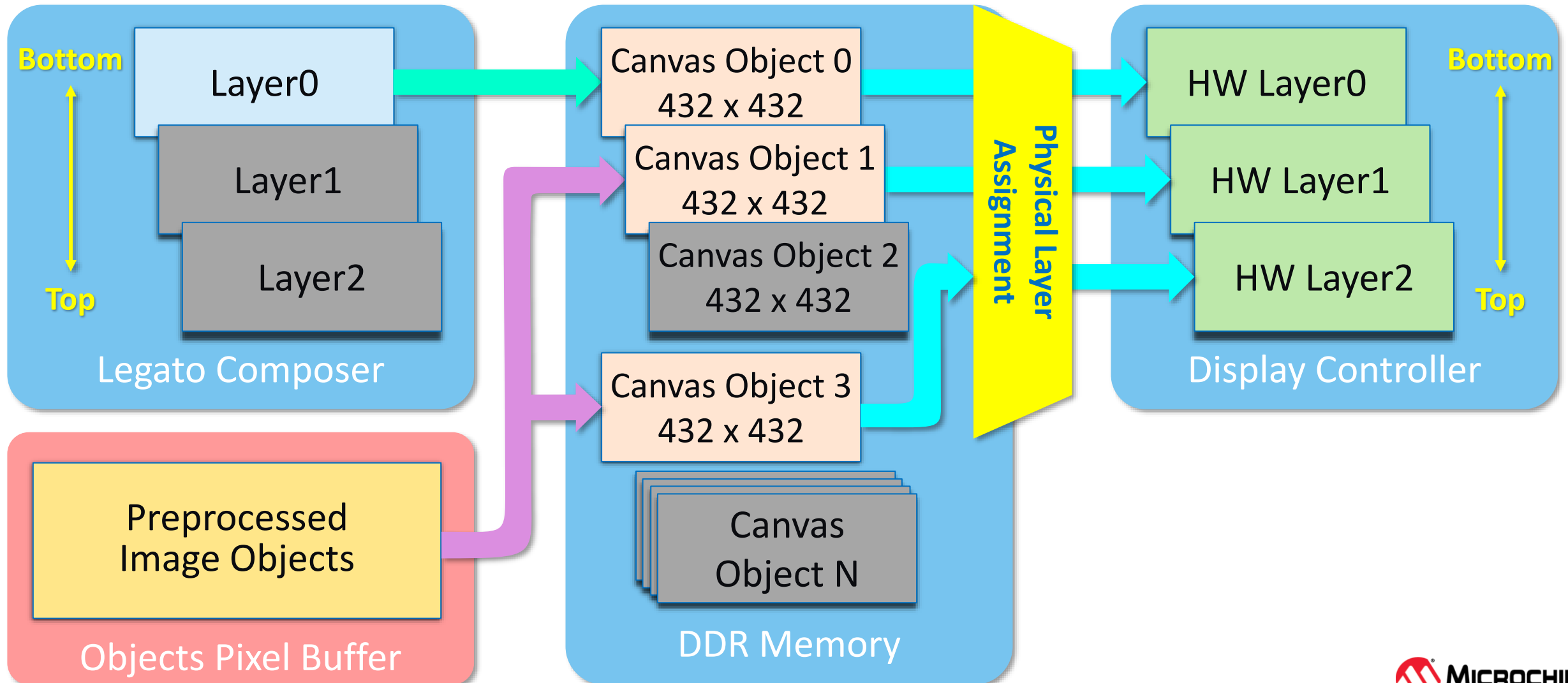
CANVAS Virtual Display

- Looks good now.



CANVAS Virtual Display

- Canvas objects are individual memory, you could reassign it dynamically.



CANVAS Button

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

CANVAS Button

- **Add a StartStop button as below.**

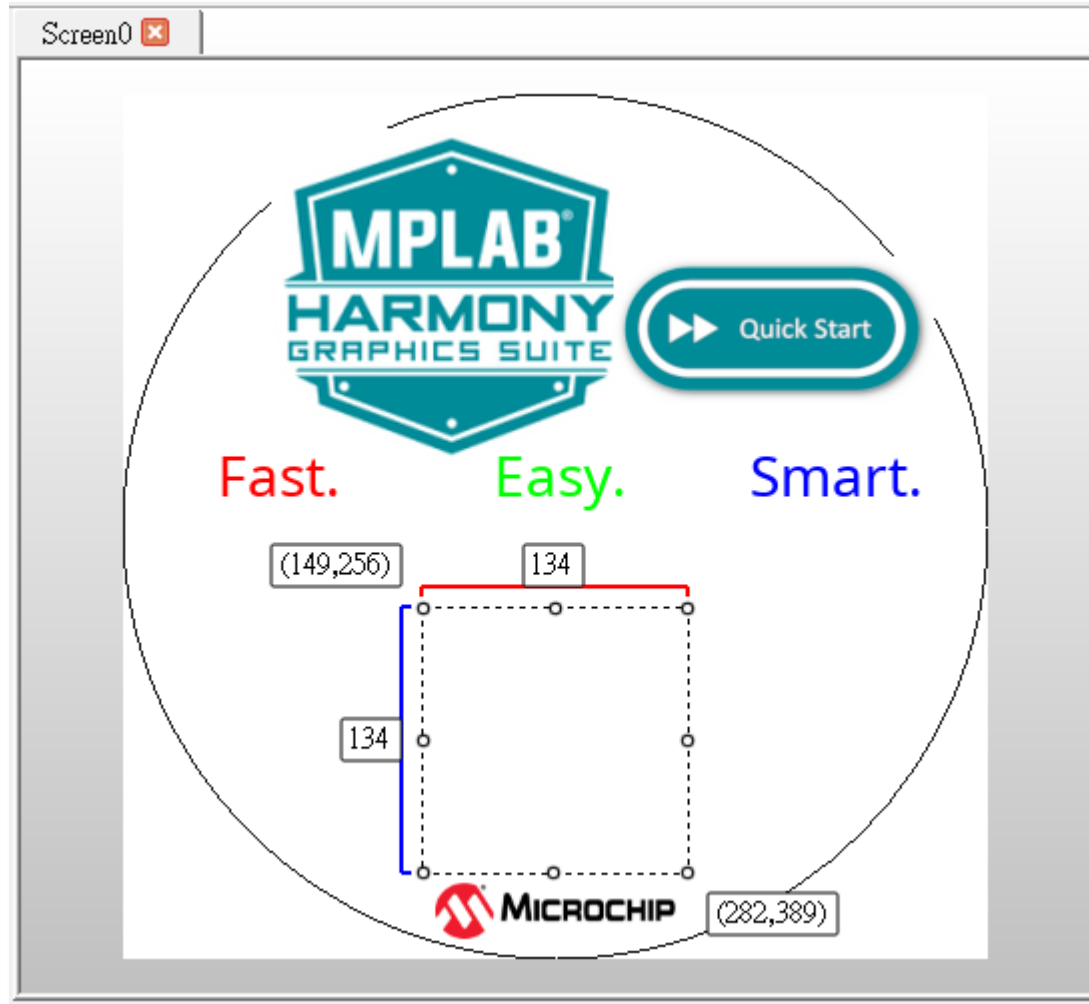
- Name : **startstopButton**
- Position X, Y : **149, 256**
- Size Width, Height : **134, 134**
- Background : **None**
- Border : **None**
- Pressed Image : **Keep Empty**
- Released Image : **Keep Empty**
- Press Offset : **0**
- Event Pressed : **Enable**
- Event Released : **Enable**

Object Editor	
Widget	
Name	startstopButton
Position	
X	149
Y	256
Size	
Width	134
Height	134
Enabled	☒
Visible	☒
Scheme	(internal) ?
Background	None
Border	None
Alignment	[Center,Middle]
Margins	[4,4,4,4]
Alpha Blending	☐
Alpha Level	255
Button	
Toggleable	☐
String	
Pressed Image	
Released Image	
Image Position	Left Of
Image Margin	10
Press Offset	0
Events	
Pressed	☒
Released	☒
Editor Options	
Locked	☐
Hidden	☐

Don't use the Images

CANVAS Button

- Button Widget after configured.



Object Editor	
Widget	
Name	startstopButton
Position	
X	149
Y	256
Size	
Width	134
Height	134
Enabled	☒
Visible	☒
Scheme	(internal) ?
Background	None
Border	None
Alignment	[Center,Middle]
Margins	[4,4,4,4]
Alpha Blending	☐
Alpha Level	255
Button	
Toggleable	☐
String	
Pressed Image	
Released Image	
Image Position	Left Of
Image Margin	10
Press Offset	0
Events	
Pressed	☒
Released	☒
Editor Options	
Locked	☐
Hidden	☐

CANVAS Button

- Implement **startstopButton** Pressed and Released event.

```
void event_Screen0_qsButton_OnPressed(leButtonWidget* btn)
{
    RGB_LED_B_On();
}
```

```
void event_Screen0_qsButton_OnReleased(leButtonWidget* btn)
{
    RGB_LED_B_Off();
}
```

```
void event_Screen0_startstopButton_OnPressed(leButtonWidget* btn)
{
    leImageToCanvas( CANVAS_ID_1, &StartStop_Start, 149, 256, 255 );
    RGB_LED_R_On();
    RGB_LED_G_On();
}
```

Callback function of Button pressed

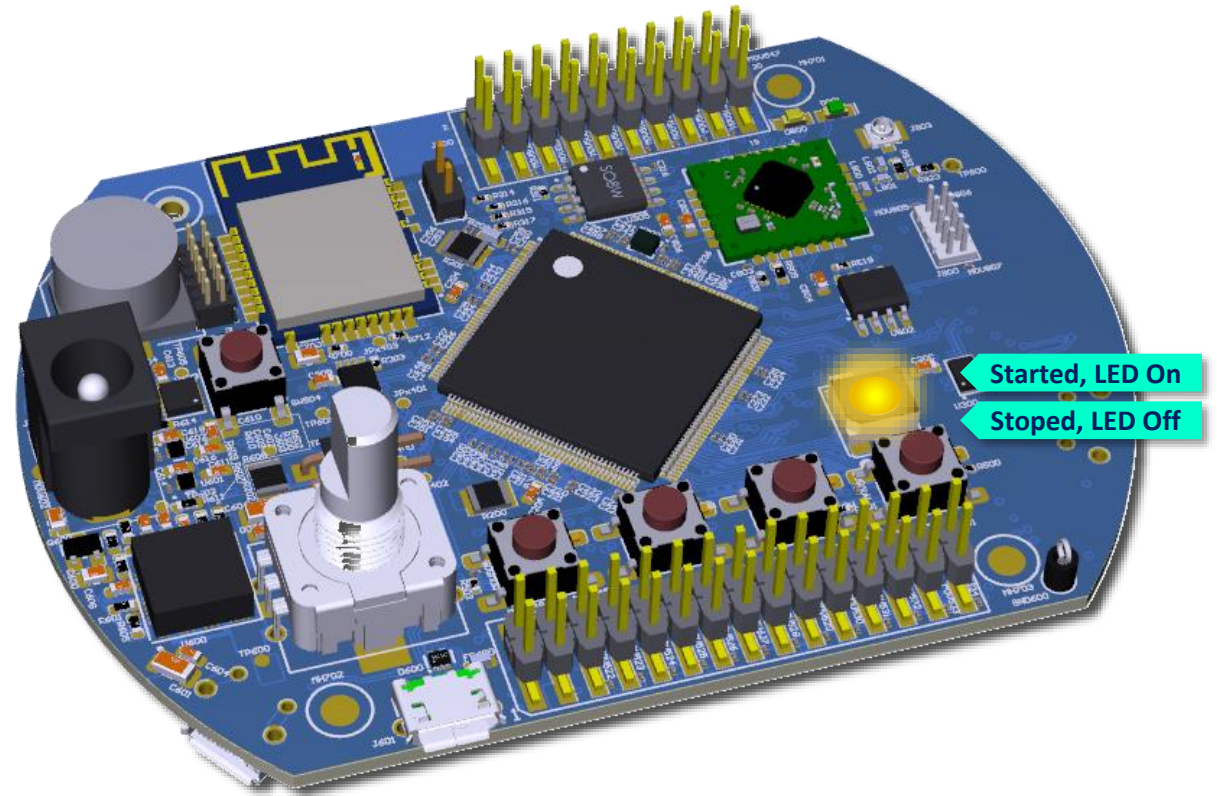
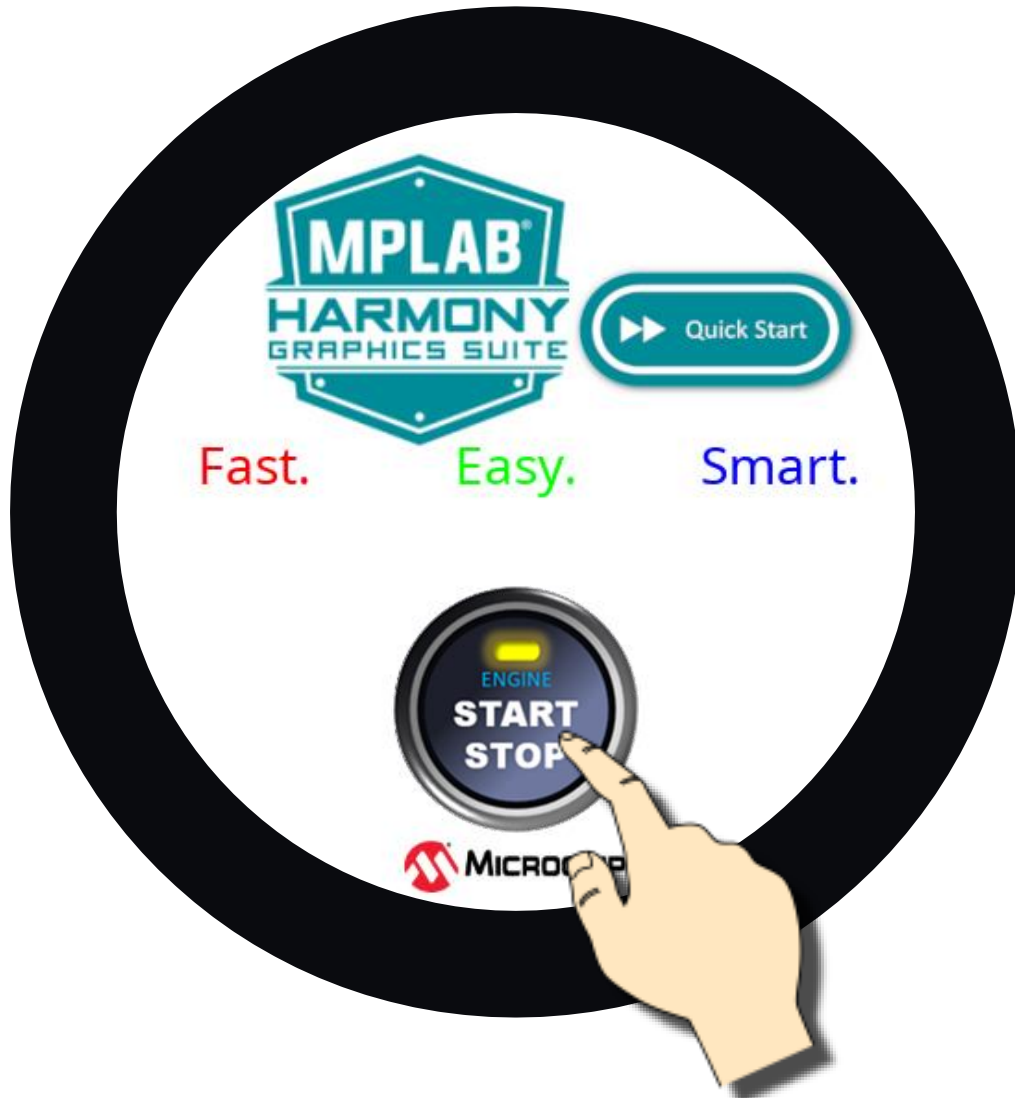
```
void event_Screen0_startstopButton_OnReleased(leButtonWidget* btn)
{
    leImageToCanvas( CANVAS_ID_1, &StartStop_Stop, 149, 256, 255 );
    RGB_LED_R_Off();
    RGB_LED_G_Off();
}
```

Callback function of Button released

```
int main ( void )
{
    ...
}
```


CANVAS Button

- The Start/Stop Button Widget Result after program.



CANVAS Button

Discuss

- What's different if to use Widget Image in Composer ?

Button	
Toggleable	☐
String	
Pressed Image	
Released Image	
Image Position	Left Of
Image Margin	10
Press Offset	0
Events	
Pressed	☒
Released	☒



Button	
Toggleable	☐
String	
Pressed Image	StartStop_StopPressed
Released Image	StartStop_Stop
Image Position	Left Of
Image Margin	10
Press Offset	0
Events	
Pressed	☒
Released	☒

```
void Screen0_OnShow( void ) { ...  
    //leImageToCanvas( CANVAS_ID_1, &StartStop_Stop, 149, 256, 255 );  
}  
  
void event_Screen0_startstopButton_OnPressed(leButtonWidget* btn) {  
    //leImageToCanvas( CANVAS_ID_1, &StartStop_Start, 149, 256, 255 );  
    ... }  
  
void event_Screen0_startstopButton_OnReleased(leButtonWidget* btn) {  
    //leImageToCanvas( CANVAS_ID_1, &StartStop_Stop, 149, 256, 255 );  
    ... }
```

CANVAS Animation Effect

PIC32MZ DA Radial Graphics Development Board and
MPLAB® Harmony Graphics Suite (MHGS)

CANVAS Animation Effect

- CANVAS support two animation application function could be easy to use.

gfxcStartEffectMove

(

unsigned int **canvasID**,

GFXC_FX_MOVE_TYPE **type**,

int **startX**,

int **startY**,

int **endX**,

int **endY**,

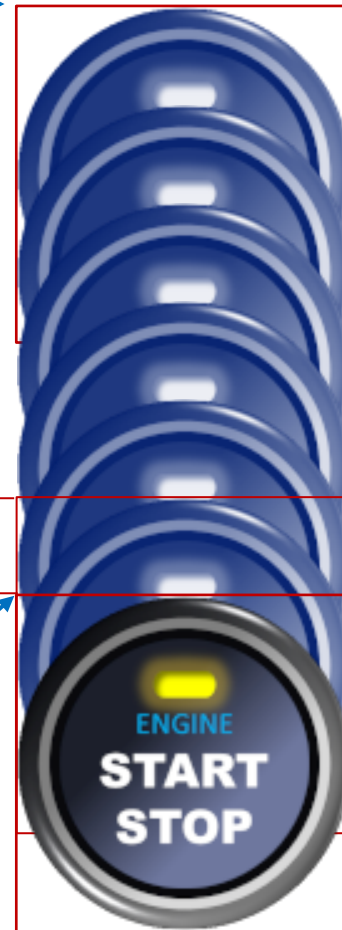
unsigned int **delta**

)

GFXC_FX_MOVE_FIXED

GFXC_FX_MOVE_ACC

GFXC_FX_MOVE_DEC

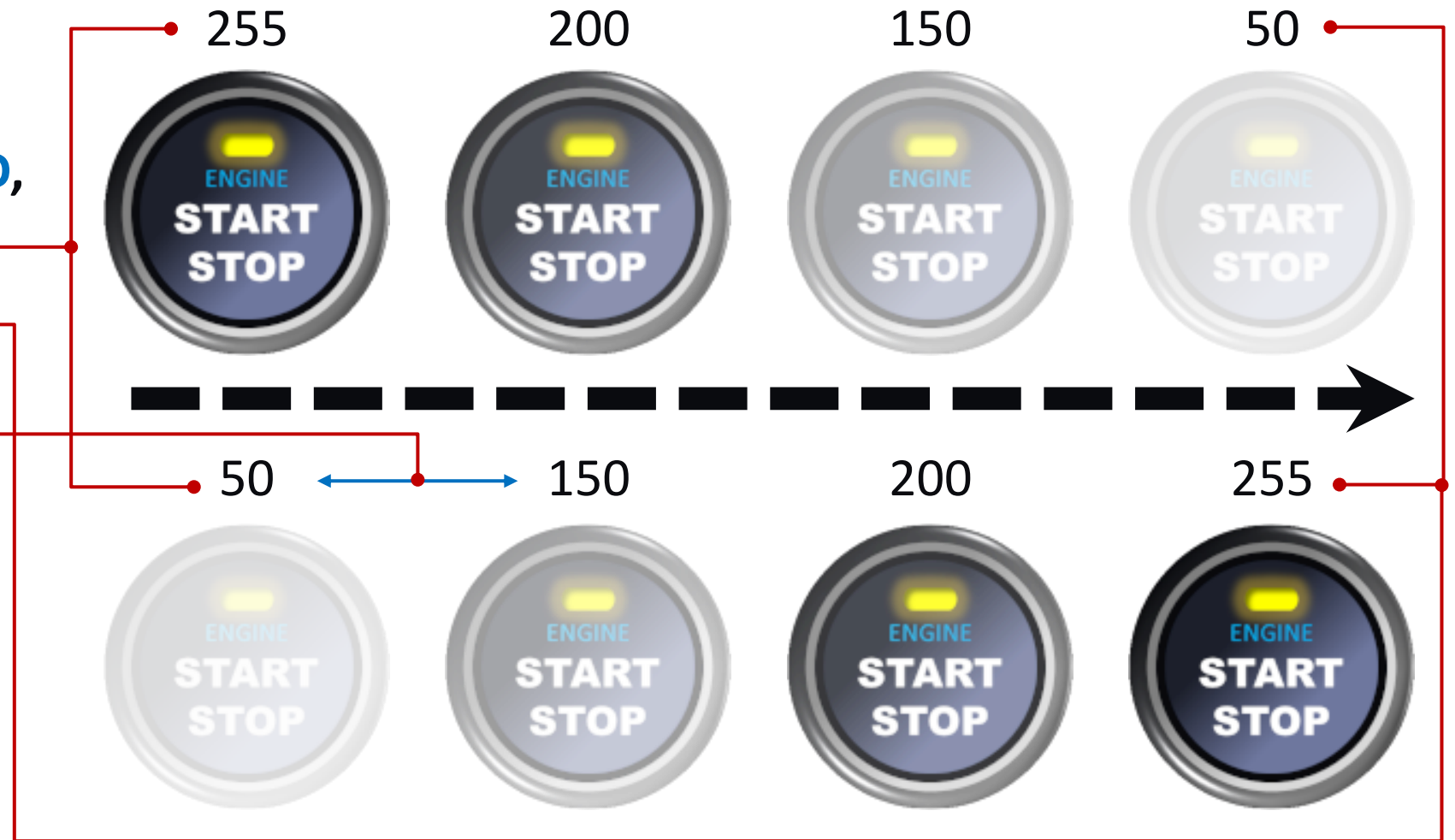


CANVAS Animation Effect

- CANVAS support two animation application function could be easy to use.

gfxStartEffectFade

```
(  
    unsigned int canvasID,  
    uint8_t startAlpha,  
    uint8_t endAlpha,  
    uint8_t delta  
)
```



CANVAS Animation Effect

- Register a CANVAS Effect Callback function could capture the animation Start, Run, Done /Idle status.

[gfx_canvas_api.h](#)

```
typedef enum
{
    GFXC_FX_IDLE,           //Done or idle, no active effects
    GFXC_FX_DONE = GFXC_FX_IDLE,
    GFXC_FX_START,         //Starting effect
    GFXC_FX_RUN,           //effect in progress
} GFXC_FX_STATUS;
```

[main.c](#)

```
GFXC_FX_STATUS MoveStatus = GFXC_FX_IDLE;
GFXC_FX_STATUS FadeStatus = GFXC_FX_IDLE;
void CanvasEffectCallback(unsigned int canvasID, GFXC_FX_TYPE effect, GFXC_FX_STATUS status, void * parm)
{
    if( canvasID == CANVAS_ID_1 )
    {
        if( effect == GFXC_FX_MOVE ) { MoveStatus = status; }
        else if( effect == GFXC_FX_FADE ) { FadeStatus = status; }
    }
}

void Screen0_OnShow( void )
{ ...
    leImageToCanvas( CANVAS_ID_1, &StartStop_Stop, 149, 256, 255 );

    gfxcSetEffectsCallback( CANVAS_ID_1, CanvasEffectCallback, 0 );
}
```

GFX v3.12.2 : CANVAS Effect Callback is necessary

CANVAS Animation Effect

- Press and hold **qsButton** to Move Out and Fade out StartStop button.
- Release **qsButton** to Move In and Fade In StartStop button.

```
void Screen0_OnShow( void ) { ... }

void Screen0_OnUpdate( void ) { ... }

void event_Screen0_qsButton_OnPressed(leButtonWidget* btn)
{
    RGB_LED_B_On();

    gfxcStartEffectMove( CANVAS_ID_1, GFXC_FX_MOVE_ACC, 149, 256, 149, -134, 10 );
    gfxcStartEffectFade( CANVAS_ID_1, 255, 0, 5 );
}

void event_Screen0_qsButton_OnReleased(leButtonWidget* btn)
{
    RGB_LED_B_Off();

    gfxcStartEffectMove( CANVAS_ID_1, GFXC_FX_MOVE_DEC, 149, -134, 149, 256, 10 );
    gfxcStartEffectFade( CANVAS_ID_1, 0, 255, 5 );
}

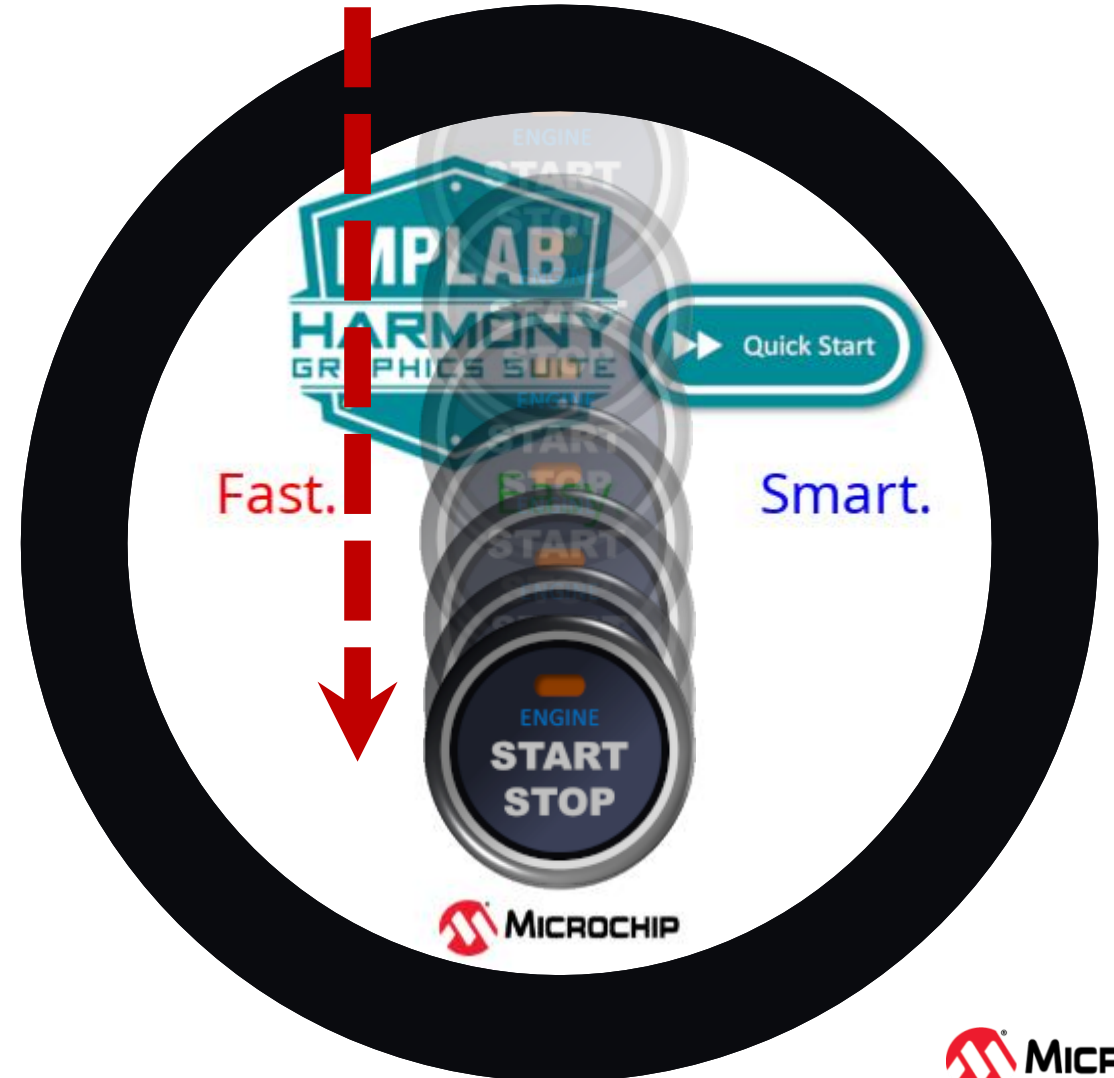
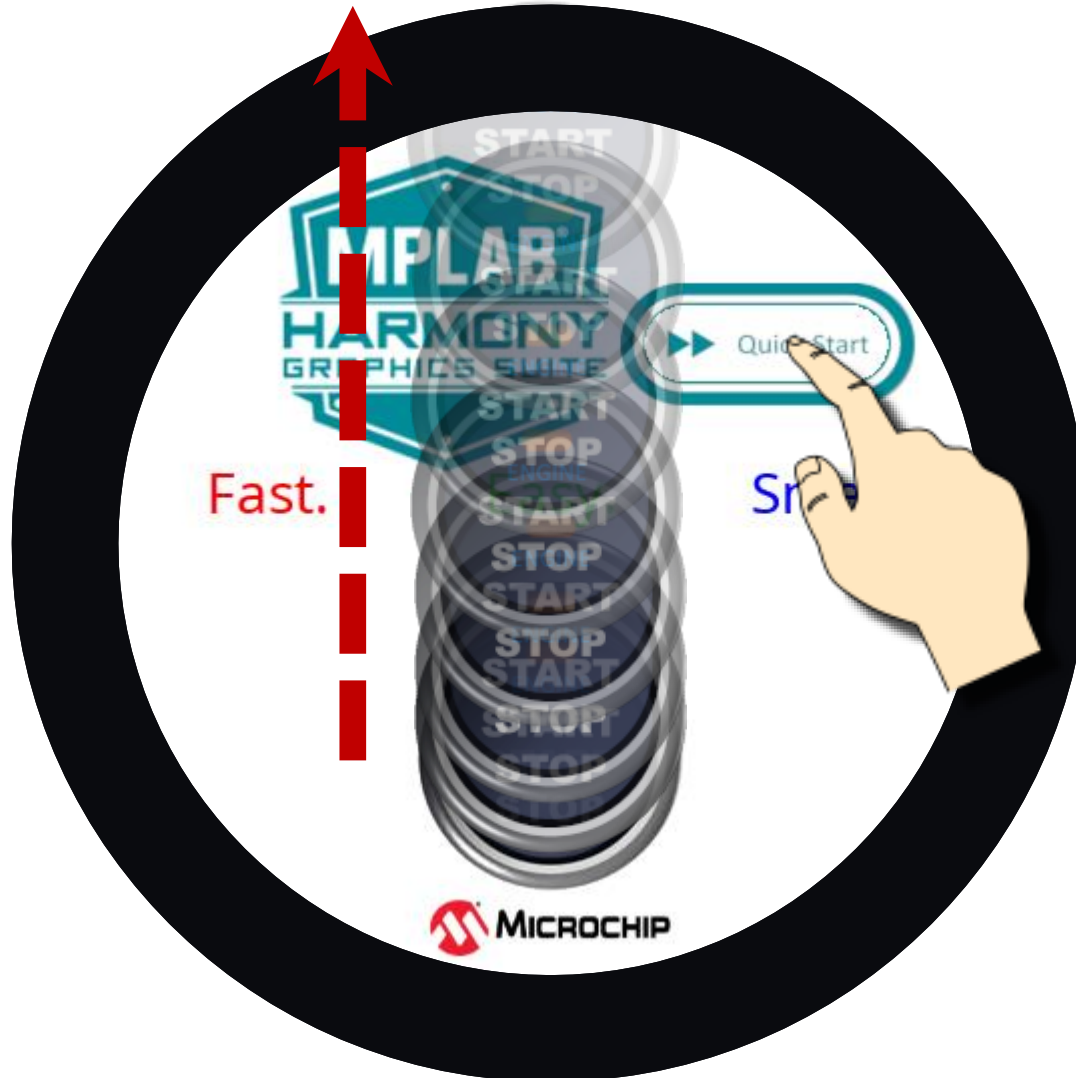
void event_Screen0_startstopButton_OnPressed(leButtonWidget* btn) { ... }

void event_Screen0_startstopButton_OnReleased(leButtonWidget* btn) { ... }

int main ( void )
```


CANVAS Button

- The Start/Stop Button Widget Result after program.



Microchip Trademarks

Overview

Microchip Technology Incorporated's products are marketed and sold under one or more of the following trademarks belonging to Microchip or one of Microchip's subsidiaries. Questions regarding use of these trademarks should be addressed to the Microchip's Legal Department via email: legal.department@microchip.com.

The following are registered trademarks of Microchip in the U.S.A. and other countries:

AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA

The following are registered trademarks of Microchip in the U.S.A:

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, WinPath, and ZL

The following are registered trademarks of Microchip in other countries: BeaconThings, GestIC, Silicon Storage Technology

The following are trademarks of Microchip in the U.S.A. and other countries:

AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICKit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA

The following is a service mark of Microchip in the U.S.A.: SQTP

The following are logos of Microchip in the U.S.A. and other countries: The Adaptec logo, Frequency on Demand, Silicon Storage Technology, Symmcom, and Trusted Time

The following is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries:

GestIC

<https://www.microchip.com/en-us/about/legal-information/microchip-trademarks>

