Develop high resolution HMI and Camera Applications with FreeRTOS Framework

Renesas RZ/A1LU

rev 1.03

Description: This embedded software lab will describe the usage the of the Renesas RZA1 Software package with FreeRTOS and GUI development package with TES Guiliani. This lab will show how hardware drivers on the RZA1 can be controlled and monitored with the GUI Framework**.**

Lab Objectives

1. Setup and Go
2. Understand the RZA1 Driver API
3. Understand GSE GUI IDE
4. Understand GUI Communication

Lab Materials

Please verify you have the following materials at your lab station.

- E2studio v6.3 (with RZ/A GCC toolchain installed)
- Segger J-Link Lite Debugger
- GCC compiler
- Tera-Term (or your favorite terminal emulator)
- Segger JLink Software Tools
- Renesas Stream-It! kit

Skill Level

1. This is so easy anyone can do it
2. Assuming of course they can write in C

Time to Complete Lab

120 Minutes

Lab Sections

(NOTE: There is No “LAB 6” in this set of labs for ACT)

1 Lab Setup

Overview:
These labs were designed based on the RZA1 FreeRTOS Software Framework that can be found in this Renesas URL.

For the purpose of this Lab, each laboratory procedure is already setup so you can start each lab without finishing the previous lab. These can be used as references for any future projects.

Ensure you have the Renesas e2Studio v6.3 environment installed. If not, you can get it here: e2Studio version 7.1. When you run the installer, ensure you select RZ/A support and the GCC toolchain.

After e2Studio is installed, you must have the GCC toolchain version 6.3. If you don't have it (you can see the versions installed by going to "Help|Add Renesas Toolchains" and see which ARM compilers you have. If you don't have it, you can download it here: Updated RZ/A Toolchain (6.3.1). Chose this version: "6-2017-q2-update" and either use the unsigned, or the version signed for your OS (either will work). Make sure e2Studio is closed when you install it, and e2 will find it the next time you start it.

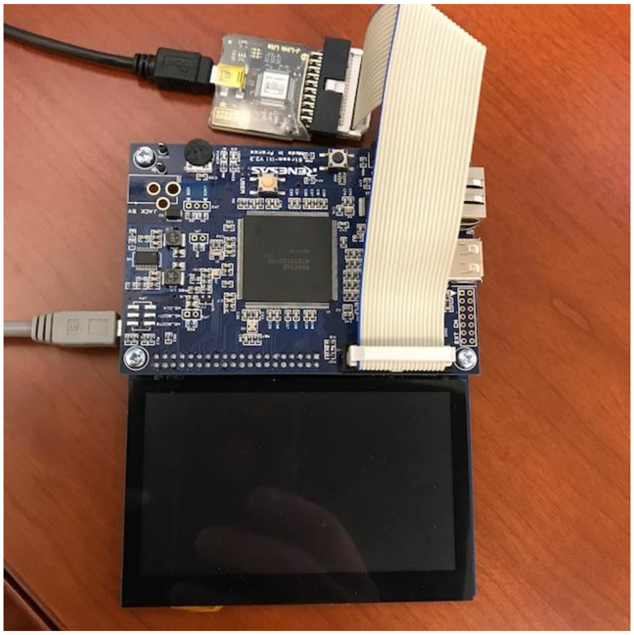
And for debugging, you will need to install the Segger JLINK drivers/support. You can get the installer from here: Segger JLINK Tools.Choose the Software and Documentation pack.

You will need a terminal emulator to look at runtime output. If you do not have a terminal emulator, you can download TeraTerm from this link:.TeraTerm

----

The following two activities (load bootloader, import projects into e2Studio) need only be done once for all labs:

Step 1.1 Set up and connect the Stream-It! Target Board as shown in the kit documentation.



Step 1.2 Find the “Workspace” folder on your PC
(usually in C:\users\<username>\e2studio\ ).

Create a folder to be used as a workspace for all lab projects (Suggested: “Labs_ACT”) and copy all the provided lab file folders into that folder (folders from “LearningLabs01” folder).

[Note: you do NOT have to copy the instruction document, or the “CodeCopy” text file (that text file makes it easy to cut/paste code into the projects) – suggestion: place it on your desktop.]

[NOTE: In future path definitions, this folder will be referred to as “<Workspace Folder>” or simply “.\”]

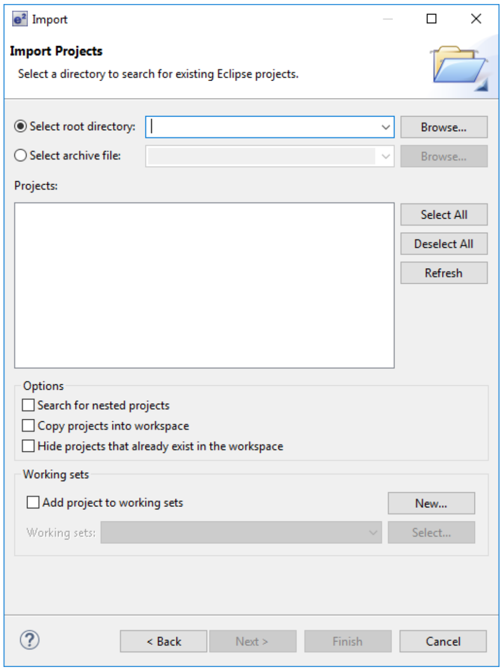
Step 1.3 Open e2Studio and set its WORKSPACE as follows: browse to the workspace folder you created as the workspace. E2Studio will open, Dismiss the start screen and news windows (if they appear).

Step 1.4 Right click in project explorer window and select “Import…”
then select “General” and “Existing project into workspace” and hit “Next”.”

Step 1.5 In the import dialog **[below]**, select the browse button then “OK” to select the current folder. You will see 6 projects show up in the list. Ensure all 6 are selected and click “Finished”,

Step 1.6 In Windows Explorer, open the windows batch file, FlashGuilianiDemo.bat, located in <workspace folder>\ RZA2LU_Lab1\src\tes\FlashTools\. By double-clicking it

Step 1.7 Execute number 1 – Flash QSPI bootloader, then select 2 for the Stream it board. When done, exit the batch file.
NOTE: Since the bootloader does not change, it will not be loaded again for any labs.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory:

○ Select archive file:

Projects:

Options

□ Search for nested projects

□ Copy projects into workspace

□ Hide projects that already exist in the workspace

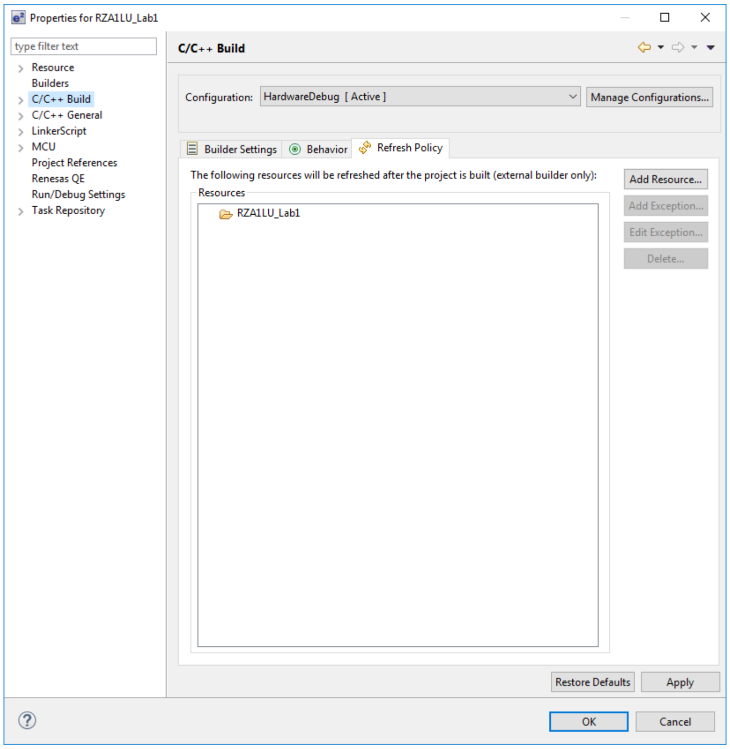
Working sets

□ Add project to working sets

Working sets:

To make the labs go faster, we can tell e2Studio to only compile/link files from the active project, and only the ones that have changed. When you import multiple projects at a time, e2studio assumes you will work on all of them at the same time.

To do this, select each project, and right click on it. Select “Properties” from the bottom of the context menu that pops up, and then select “C/C++ Build” on the left tree list. Select the “Refresh Policy” tab:



Configuration: HardwareDebug [Active]

The following resources will be refreshed after the project is built (external builder only):

Resources

RZA1LU_Lab1

Highlight whatever is listed in the “Resources” box, and click “Delete…” to delete the resources.

Click “Add…” and from the list of projects, select ONLY the active project. Then click “Apply” and then “OK” to close the dialog.

TIP: As an added tip, the first compile is always the longest. At the beginning of each lab, select the project you will work on, right click and select “Clean Project” and then “Build Project” which will do a full build. You can work on the GUI part first, and the compile will complete in the background.

As you make the changes in the lab, the subsequent builds will only be a few 10’s of seconds instead of 3-4 minutes.

2 Lab1 GSE Screens

Overview:

In this section we will explore the basics of the TES Guiliani IDE called GSE. This lab will illustrate the following:

- Creating two dialog windows.
- Buttons to control the transition between windows
- Apply a background to each of the dialogs
- Display Text on the Main Dialog
- Run this on the Simulator
- Run on Target.

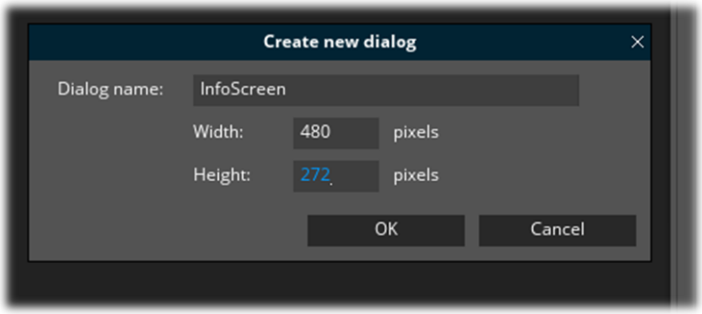
Procedural Steps

Step 2.1 Create a new GSE project called **Lab1** and store it in directory .\RZA1LU_Lab01\src\tes\GUI_Sample\GuilianiDemo.

[NOTE: To launch GSE, navigate in Windows Explorer to <Workspace Folder>\GSE and double-click the GSE icon]

Step 2.2 Create two dialogs screens one named **InfoScreen** and one named **ButtonScreen**

- Select File from the menu then New Dialog
- Set the resolution to the same resolution as the Stream it LCD (480 x 272).



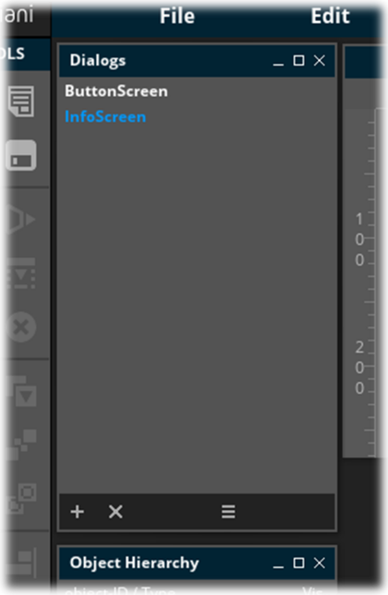
Create new dialog

Dialog name: InfoScreen

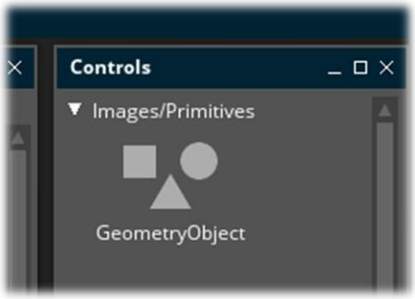
Width: 480 pixels

Height: 272 pixels

Step 2.3 You should see the two Dialogs in the Dialogs Window. Selecting the InfoScreen, the Attributes and Object Hierarchy will update.



Step 2.4 To add background color we need to add a Geometric Object. Select the GeometryObject in the Controls window.

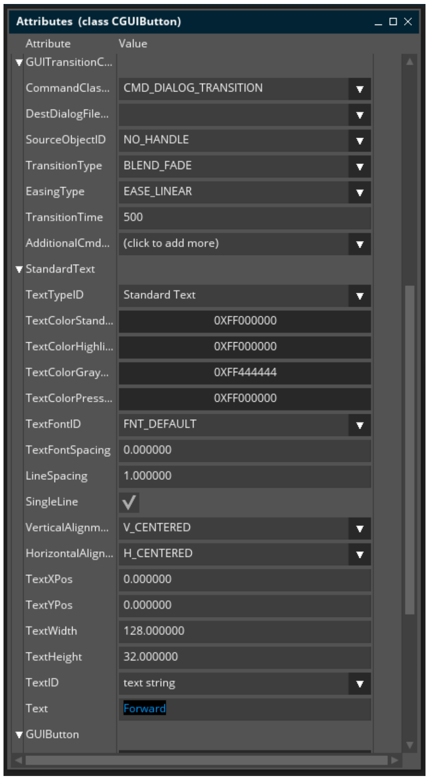


- Select this object in the View window. Note the GeometryObject attributes are modified in the Attribute window.
- Set the Width and Height to the same resolution as the Dialog Screen (480 x 272).
- Scroll down the Attributes View to the Color Attribute. Click on this and the color pallet dialog pops up. Pick a color for this dialog background.
- Do these steps again for the ButtonScreen

Step 2.5 Add buttons

- Select the InfoScreen Dialog and select Button from the Controls Window
- Select the Button in the View or Object Hierarchy Window to view the attributes.

- Use the Xpos and Ypos attributes to place the button at (330,225). Press “enter” for each value.
- In the StandardText group inside the Attributes window, set Text attribute to “Forward” and press enter.
- In the Attributes Window scroll to the GUICommand group. Set the sub attribute CommandClassID to CMD_DIALOG_TRANSITION. This displays additional sub-attributes.
- Set Sub-attributes
  - ○ DestDialogFileName = ButtonScreen
  - ○ SourceObjectID = this Button’s ID (AID_BUTTON_1)
  - ○ TransitionType = BLEND_FADE
  - ○ EasingType = EASE_LINEAR
  - ○ TransitionTime = 500


Attributes (class CGUIButton)

Attribute	Value
▼ GUITransitionC...	
CommandClas...	CMD_DIALOG_TRANSITION ▼
DestDialogFile...	▼
SourceObjectID	NO_HANDLE ▼
TransitionType	BLEND_FADE ▼
EasingType	EASE_LINEAR ▼
TransitionTime	500
AdditionalCmd...	(click to add more) ▼
▼ StandardText	
TextTypeID	Standard Text ▼
TextColorStand...	0XFF000000
TextColorHighli...	0XFF000000
TextColorGray...	0XFF444444
TextColorPress...	0XFF000000
TextFontID	FNT_DEFAULT ▼
TextFontSpacing	0.000000
LineSpacing	1.000000
SingleLine	✓
VerticalAlignm...	V_CENTERED ▼
HorizontalAlign...	H_CENTERED ▼
TextXPos	0.000000
TextYPos	0.000000
TextWidth	128.000000
TextHeight	32.000000
TextID	text string ▼
Text	Forward
▼ GUIButton	

- Now do the same steps again for the ButtonScreen Dialog.
- Place the button at (22,225).
- Set the Text attribute to “Back”
- Set CommandClassID to CMD_DIALOG_TRANSITION as above. Then set GUICommand Sub-attributes
    - DestDialogFileName = InfoScreen
    - SourceObjectID = this Button ID (AID_BUTTON_2)
    - TransitionType = BLEND_FADE
    - EasingType = EASE_LINEAR
    - TransitionTime = 500

Step 2.6 Run Simulator

- Save Project (Cntl-S) to .\ RZA1LU_Lab01\src\tes\GUI_Sample\GuilianiDemo\Lab1 directory
- Run Simulator
    - GSE IDE Menu select File -> Run Simulation
    - In the Run Simulation Popup Dialog set the Start dialog to InfoScreen. This will set the default dialog that appears first.
    - Select Run
    - Test the “Forward” and “Back” buttons for correct functionality.

Step 2.7 Now let’s run this on the Target board. To do this, start by exporting the resources from the IDE.

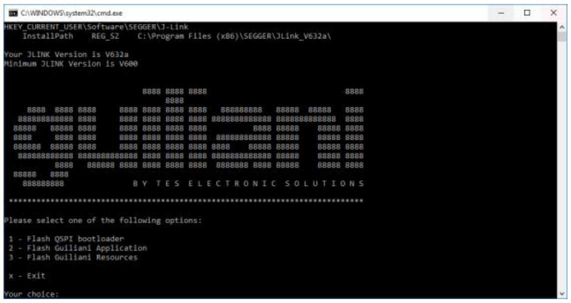
Select “Export” from the IDE “Resources” menu.

Step 2.8 Set the export settings.

- Export Directory :
  .\ RZA1LU_Lab01\src\tes\GUI_Sample\Include\GUIConfig
- Create resource file : Enabled
- Width : 480
- Height : 272
- Start Dialog : InfoScreen

[NOTE: By exporting you created some “resources” for the GUI runtime to use to create the screens. You must now download these resources into the flash memory of the board]

Step 2.9 Execute the windows batch file .\ RZA1LU_Lab01\src\tes\FlashTools\FlashGuilianiDemo.bat



```
HKEY_CURRENT_USER\Software\SEGGER\J-Link
    InstallPath    REG_SZ    C:\Program Files (x86)\SEGGER\JLink_V632a\

Your JLINK Version is V632a
Minimum JLINK Version is V600

                              8888 8888 8888                              8888
                                   8888
     8888  8888 8888     8888 8888 8888 8888   888888888   88888  88888   8888
   888888888888 8888     8888 8888 8888 8888 8888888888888 8888888888888  8888
  88888   88888 8888     8888 8888 8888 8888          8888 88888    88888 8888
  8888     8888 8888     8888 8888 8888 8888  888888888888 88888    88888 8888
  888888  88888 8888     8888 8888 8888 8888 8888    88888 88888    88888 8888
   888888888888 8888888888888 8888 8888 8888 8888888888888 88888    88888 8888
           8888   888888 8888 8888 8888 8888  8888888 8888 88888    88888 8888
  88888   8888
    888888888               B Y  T E S  E L E C T R O N I C  S O L U T I O N S

 *****************************************************************************

Please select one of the following options:

 1 - Flash QSPI bootloader
 2 - Flash Guiliani Application
 3 - Flash Guiliani Resources

 x - Exit

Your choice:
```

Step 2.10 Execute number 3 - Flash Guiliani Resources then select 2 for the Stream it board.

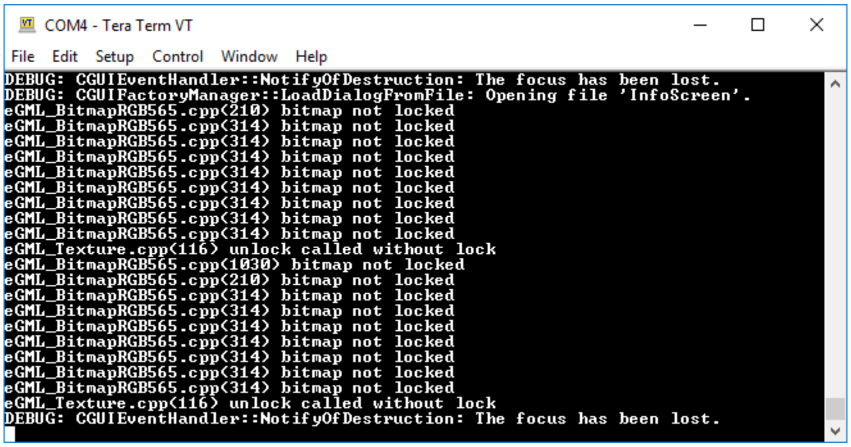
Step 2.11 After the code is loaded, follow the menu to exit the batch file.

Step 2.12 Open a TeraTerm Terminal on windows PC. We will use the serial port to observe debug information.

- Baudrate 115200
- Port (Set to the COM port of your device)
- Data bits 8
- Stop bits 1
- Parity None
- Flow control XON/XOFF
- Set the newline receive to Auto or "LF-CR"

Step 2.13 Open e2studio to the created workspace (if not already open).

- Ensure the project named "RZA1LU_Lab01" is the active project in the project explorer window. **Build** the project (by pressing the hammer icon in the middle of the toolbar – not the one on the left).
- **Download** and Debug the RZA1LU_Lab01 [HardwareDebug]
  - Select the arrow next to the debug icon located in the toolbar.
  - Select **Debug Configuration**.
  - Select **RZA1LU_Lab01 HardwareDebug**. (under Renesas GDB Hardware Debugging).
  - Press Debug
- Click the "**Resume**" icon (middle of the toolbar) until the code is running (code will run to "main()" and stop, run again).
- Use the buttons on the screen to switch screens.
- On the terminal emulator, you will see messages from the serial port on each transition:



```
DEBUG: CGUIEventHandler::NotifyOfDestruction: The focus has been lost.
DEBUG: CGUIFactoryManager::LoadDialogFromFile: Opening file 'InfoScreen'.
eGML_BitmapRGB565.cpp(210) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_Texture.cpp(116) unlock called without lock
eGML_BitmapRGB565.cpp(1030) bitmap not locked
eGML_BitmapRGB565.cpp(210) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_BitmapRGB565.cpp(314) bitmap not locked
eGML_Texture.cpp(116) unlock called without lock
DEBUG: CGUIEventHandler::NotifyOfDestruction: The focus has been lost.
```

3 Lab2 GUI Toggle Button Control Hardware LED

Overview:

In this section you will add a GUI Toggle Button to the Control Screen we created in section 2.2. The button will send a command event to StreamRuntime to turn the LED on and off when pressed.

Procedural Steps

Step 3.1 Create a new GSE project called **Lab2** and store it in directory .\RZA1LU_Lab02\src\tes\GUI_Sample\GuilianiDemo.

Step 3.2 Create a new Dialog named **MainButtons**. Set the resolution to 480x272.

Step 3.3 Add background color like in lab 1

Step 3.4 Add a Button and move it to roughly the center of the dialog using the mouse to drag it, or the Xpos and Ypos attributes. Set the Button Text attribute to “LED”.

Step 3.5 Set GUICommand CMD_CALLAPPLICATIONAPI.

Step 3.6 Set the attribute ApplicationAPI to “Led0”. This could be anything but you need to remember it for later.

Step 3.7 Set the attribute Parameter set to 0. NOTE This is not used.

Step 3.8 Set the Resources -> Export settings.

- Export Directory :
  .\ RZA2LU_Lab2\src\tes\GUI_Sample\Include\GUIConfig
- Create resource file : Enabled
- Width : 480
- Height : 272
- Start Dialog : MainButtons
- Press OK

Step 3.9 Execute the windows batch file .\ RZA2LU_Lab2\src\tes\FlashTools\FlashGuilianiDemo.bat

Step 3.10 Execute number 3 - Flash Guiliani Resources then select 2 for the Stream it board. When done exit the batch file.

Step 3.11 Open RZA1LU_Lab02 in e2studio.

Step 3.12 Open .\RZA1LU_Lab02\src\tes\GUI_Sample\Source\MyGUI_SR.cpp file.

Step 3.13 Modify the existing GSE method “CallApplicationAPI” in the C file: MyGUI_SR by adding the following code above the “return” line of code.

```
        // Debug print captured Command
        printf( "%s\n", (char*)kAPI.ToASCII_Alloc());

        if (kAPI == "Led0")
        {
                /*Toggle Led*/
                printf( "LED Button Pressed\n");
        }
```

CallApplicationAPI

This method handles GSE object command event CMD_CALLAPPLICATIONAPI. This method parses the KAPI string and the kParm. These are the ApplicationAPI and Parameter defined in the GSE IDE.

Step 3.14 Inside the if statement in the CallApplicationAPI method add the following code. This should be located inside the if statement for “Led0” after the LED toggling printf() function.

```
/* open LED driver */
led_handle = open( DEVICE_INDENTIFIER "led", O_RDWR);

/* check the value the static value variable for last position */
if ( value  == 0)
{
        /* LED OFF */
        control(led_handle, CTL_SET_LED_OFF, &led);
        value = 1;
}
else
{
        /* LED ON */
        control(led_handle, CTL_SET_LED_ON, &led);
        value = 0;
}
close(led_handle);
```

Step 3.15 Declare these variables in CallApplicationAPI.

```
int_t led_handle = (-1);
static uint8_t value = 0;
uint16_t led = LED0;
```

Step 3.16 Add include header declarations.
This is a “C” style header, so it must be included in the “extern C {“ section” at the top of the file:

```
#include "r_led_drv_api.h"
```

Step 3.17 Ensure **RZA1LU_Lab02** is the selected project. You can now compile, download the project, and run it. Once the GUI Lab is running, you will see the GUI demo with the button labeled LED. By pressing the button, you will see the following messages. You should also see the LED on the target board turn off and on.

```
DEBUG: CGUIFactoryManager::LoadDialogFromFile: Opening file 'MainButtons.xml'.
WARNING: CGUIResourceFileHandler::Open: Could not open file MainButtons.xml.xml.
DEBUG: CGfxWrapeGML::LoadImg: Opening file 'images/Standard/ButtonPressed.png'...
DEBUG: CGfxWrapeGML::LoadImg: Opening file 'images/Standard/ButtonHighlighted.png'...
DEBUG: CGfxWrapeGML::LoadImg: Opening file 'images/Standard/ButtonStandard.png'...
DEBUG: CGfxWrapeGML::LoadImg: Opening file 'images/Standard/ButtonGrayedout.png'...
DEBUG: CGfxWrapeGML::LoadImg: Opening file 'images/Standard/ButtonFocussed.png'...
Led0
LED Button Pressed
```

4 Lab3 GUI Switch Control LED

Overview:

In this lab we will create a complex GUI control. Similar to the previous lab, you will send a command to turn the LED on and off, depending on the switch position. In addition, we will be accessing the GUI Switch Position by using the TES Guiliani *Data Pool* and *Observer*.

Procedural Steps

> ℹ TES Guiliani does not have a switch control. We will build upon the GSE Control Check box to create a switch and import images to change how the switch functions and looks.
>
> **DataPool**
>
> The Guiliani DataPool Component is used for generic binding of UI Elements to arbitrary data sources. The DataPool stores application specific data and allows objects within the user interface to get notified when data changes, for which they have registered themselves as observers.
>
> **CGUIObserver**
>
> Use this base class if you wish to follow the observer-design-pattern within your application. Simply derive the class that you want to act as an observer from CGUIObserver, and implement the desired functionality within OnNotification(). This method will be called whenever an observed subject changes.

Step 4.1 Create a new GSE project named **Lab3** and save at .\RZA1LU_Lab03\src\tes\GUI_Sample\GuilianiDemo.

Step 4.2 Create a new Dialog named **MainDialog** for our screen (480x272) with a black background as before (add a geometry object and size/color it).

Step 4.3 Open Image Manager:
From the GSE IDE menu select Resources -> Manager -> images

Step 4.4 In the Image manager select the Insert new image button. Browse to images folder in the root workspace folder. Select the blue switch images (switch_blue_on, & switch_blue_off). We will use these to change the look of the GSE control.

Step 4.5 Add Check Box Control from Control window and move it near the center of the dialog (either drag or use X & Y attributes).

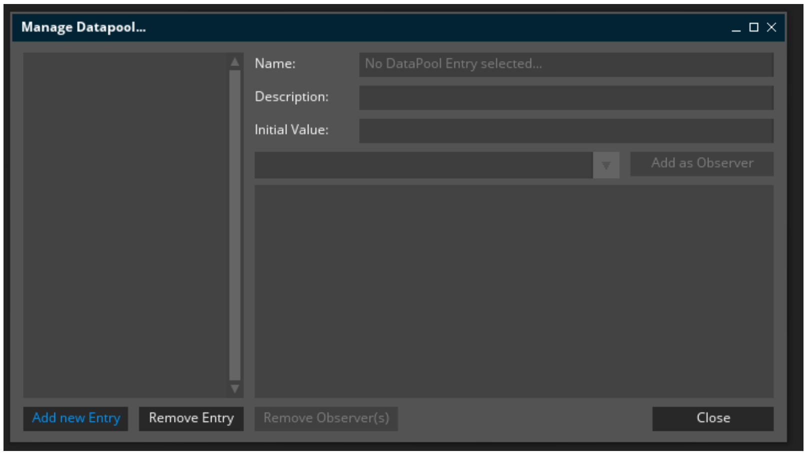
Step 4.6 In the Check Box Attributes scroll down to last 10 attributes. These are used to change how the control looks in the on, off, & faded states. Set the first five of these attributes to IMG_SWITCH_BLUE_ON. Set the last 5 of these to IMG_SWITCH_BLUE_OFF.

Step 4.7 Set the attribute CheckBoxLayout to ICON_FILL_OBJECT. This will let the image fill the size of the check box.

Step 4.8 There are two sets of “**width**” and “**height**” attributes (both the “Object” and “Text” sections). Set all the size attributes as follows.

- Width : 40.0
- Height : 20.0

Step 4.9 Now we will connect the control to the Data Pool. Open the **DataPool Manager**. From the IDE menu select Resources -> Manage -> Data Pool



Step 4.10 Select **Add New Entry**. Change the following:

- Name: DATAPOOL_LED
- Drop Down menu: Set ID to the CheckBox (i.e AID_CHECKBOX_1)
- Press Add as Observer. The CheckBox ID will show in the below.

Step 4.11 Close the Datapool Manager

Step 4.12 Run the project on the simulator and test that the switch changes to off and on. Refer to step 2.6 for reference.

Step 4.13 Export the GSE project to .\RZA1LU_Lab03\src\tes\GUI_Sample\Include\GUIConfig.

Step 4.14 Execute the windows batch file .\ RZA2LU_Lab3\src\tes\FlashTools\FlashGuilianiDemo.bat

Step 4.15 Execute number 3 - Flash Guiliani Resources, then select 2 for the Stream It board. Exit the batch file when done.

Step 4.16 Open RZA1LU_Lab03 project in e2studio.

Step 4.17 Open .\RZA1LU_Lab03\src\tes\GUI_Sample\Source\MyGUI_SR.cpp file.

Step 4.18 Create a CMyGUI method pvLedButtonCallback below. This function will handle changes to the DataPool

```
void CMyGUI::pvLedButtonCallback(CDataPoolEntry& data)
{
    CGUIValue value;
    uint16_t led = LED0;
    int_t led_handle = (-1);

    /* get the value of datapool for LED checkbox */
    CGUIDataPool::Get(DATAPOOL_LED,value);

    /* open LED driver */
    led_handle = open( DEVICE_INDENTIFIER "led", O_RDWR);
    /* check the value of datapool for LED checkbox */
    if (value.ToInt() == 0)
    {
        /* LED OFF */
        control(led_handle, CTL_SET_LED_OFF, &led);
    }
    else
    {
        /* LED ON */
        control(led_handle, CTL_SET_LED_ON, &led);
    }
    close(led_handle);
}
```

Step 4.19 Register the method pvLedButtonCallback with the DataPool. Do this in the CMyGUI constructor. (This is the method starting with “CMyGUI::CMyGUI”

```
/* register callback function */
CGUIDataPool::Register(DATAPOOL_LED, &pvLedButtonCallback);
```

Step 4.20 Add this include header. Put this after the include of the “GUIButton.h” header file.

```
#include "GUIDatapool.h"
```

Step 4.21 Open the MyGUI_SR.h and add the pvLedButtonCallback declaration with public scope.

(This file is located in the project under: src\tes\GUI_Sample\Include).

```
static void pvLedButtonCallback(CDataPoolEntry& data);
```

Step 4.22 Ensure the **RZA1LU_Lab03** is selected and build the project and load the project onto the RZA1 Target Board. Press “Resume”. The GUI Switch turns the LED off and on according to the position of the switch.

5 Lab4 Hardware Button Control GUI Toggle Indicator

Overview:

In this section we will be using the hardware button to control a GUI Toggle Indicator. This will show how the Renesas HAL can send an event command to the Guiliani StreamRuntime.

Procedural Steps

Step 5.1 Create a new GSE project named **Lab4** and save at .\RZA1LU_Lab04\src\tes\GUI_Sample\GuilianiDemo.

Step 5.2 Create a new 480x272 Dialog named **MainDialog** with a colored background.

Step 5.3 Open Image Manager. From the GSE IDE menu select Resources -> Manager -> images

Step 5.4 In the Image Manager select the Insert new image button. Browse to images folder in the workspace folder. Select the round button images. We will use all these to change the look of the GSE control.

Step 5.5 Add a Button from Control window. We will modify this into an Indicator.

Step 5.6 Scroll down to the set the button images. Set the button images as follows:

• ImageIDNormal	IMG_ROUND_BUTTON_BLUE
• ImageIDHighlighted	DUMMY_IMAGE
• ImageIDPressed	DUMMY_IMAGE
• ImageIDGrayed	IMG_ROUND_BUTTON_GRAY
• ImageIDFocused	DUMMY_IMAGE

Step 5.7 Check (turn on) these Attributes:

- Disabled
- GrayedOut

Step 5.8 Set the Width/Height and TextWidth/TextHeight dimensions.

- Width : 64
- Height : 64

Step 5.9 Uncheck Focusable.

Step 5.10 Remove the Text in the Text Field Attribute and drag the indicator to the middle of the dialog.

Step 5.11 Set the ObjectID to AID_BUTTON_1.

Step 5.12 Save Project.

Step 5.13 Export to .\RZA1LU_Lab04\src\tes\GUI_Sample\Include\GUIConfig. There is no need to run this on the simulator as it will not do anything at this point.

Step 5.14 Execute the windows batch file .\ RZA2LU_Lab4\src\tes\FlashTools\FlashGuilianiDemo.bat

Step 5.15 Execute number 3 - Flash Guiliani Resources then select 2 for the Stream It board.

Step 5.16 Open RZA1LU_Lab04 project in e2studio.

Step 5.17 Open .\RZA1LU_Lab02\src\tes\GUI_Sample\Source\MyGUI_SR.cpp file. We will add additional methods to initialize and handle switch interrupts.

Step 5.18 Create the following methods for CMyGUI. These can be placed at the bottom of CMyGUI_SR.cpp file.

```
static event_t gs_switch_event;

/* Handles switch press interrupts */
static void my_switch_pressed (void) {
}

/* handles asynchronous messages from the switch driver interrupts */
static void my_switch_task (void *parameters) {
}

/* Initialize the Renesas Switch driver */
static void initialise_switch_monitor_task ( void ) {
}
```

Step 5.19 Declare the function prototype initialise_switch_monitor_task, my_switch_task, and switch_pressed at the top of the CMyGUI_SR.cpp file after the all the header includes.

```
static void initialise_switch_monitor_task ( void );
static void my_switch_task (void *parameters);
static void switch_pressed (void);
```

Add this header inside the extern “C” code block.

```
#include “r_switch_driver.h”
```

Step 5.20 Add the following code to the initialise_switch_monitor_task method. This will initialize a queue for messaging, register the callback methods with the switch driver, and create a task.

```
os_task_t *p_os_task;

R_SWITCH_Init(true);
R_SWITCH_SetPressCallback(my_switch_pressed);

/* Create a task to monitor the switch */
p_os_task = R_OS_CreateTask("Switch", my_switch_task, NULL,
R_OS_ABSTRACTION_PRV_DEFAULT_STACK_SIZE, TASK_SWITCH_TASK_PRI);

/* NULL signifies that no task was created by R_OS_CreateTask */
if (NULL == p_os_task)
{
        /* Debug message */
}
```

Step 5.21 Add the code to the switch callback function *my_switch_pressed*. This will pass messages to the task. NOTE: Because these are interrupts FreeRTOS requires that we use the ISR API.

```
/* notify the switch task that the switch has been pressed */
R_OS_SetEvent(&gs_switch_event);
```

Step 5.22 In the function my_switch_task, add the following code to handle switch messages. When the switch has occurred the CGUIButton attribute GrayedOut is modified.

```
UNUSED_PARAM(parameters);
CGUIButton* m_pkButtonParent;

R_OS_CreateEvent(&gs_switch_event);

/* endless loop */
while (1)
{

        R_OS_WaitForEvent(&gs_switch_event, R_OS_ABSTRACTION_PRV_EV_WAIT_INFINITE);

        m_pkButtonParent = static_cast<CGUIButton*>(GETGUI.GetObjectByID(AID_BUTTON_1));
        if ( m_pkButtonParent->IsGrayedOut() ) {
                m_pkButtonParent->SetGrayedOut(false);
        } else {
                m_pkButtonParent->SetGrayedOut(true);
        }

}
```

Step 5.23 In the CMyGUI_SR.cpp file, add the following code to the constructor:

Constructors have the same name as the class, and have a “deconstructor” of the same name but starting with the “~” character. In this case, it looks like this:

```
CMyGUI::CMyGUI(
    eC_Value x, eC_Value y,
    eC_Value width, eC_Value height,
    ObjectHandle_t eID) :
    CStreamRuntimeGUI(x, y, width, height, eID)

{

}
```

```
        initialise_switch_monitor_task();
```

Step 5.24 Ensure you have the correct project selected, build the project, and load the project onto the RZA1 Target Board. Press Run. Press the USER button on the Stream It Board. The GUI indicator will change color.

6 Lab5 Set LED Duration with GUI Slider Control

Overview:

In this section we will be using a combination of GUI Button, Hardware LED, and GUI Slide Control. The Slide Control will set the duration that the LED will remain on after a button press.

Procedural Steps

Step 6.1 Open existing GSE project named Lab5 located at .\RZA1LU_Lab05\src\tes\GUI_Sample\GuilianiDemo.

Step 6.2 This demo already has a GUI Control. This control turns on the LED. Now we will add a slider to indicate how long the LED stays on.

Step 6.3 Add the Control Horizontal Slider. Set the ID to AID-SLIDER_1.

Step 6.4 Set the Control Attributes:

MinValue :	10
MaxValue :	250
StepSize :	1
Value :	30

- Place the slider wherever you want on the screen

Step 6.5 Add 2 TextFields using the Controls Pane (just like other objects).

- Set TextWidth and GUIObject Width for both to 50.
- Set the TextField Attribute of one TextField to “Min” and place it on the left side of the horizontal slider.
- Set the TextField Attribute of the second TextField to “Max” and place it on the right of the horizontal slider

Step 6.6 Export the GSE project to .\RZA1LU_Lab05\src\tes\GUI_Sample\Include\GUIConfig.

Step 6.7 Execute the windows batch file .\ RZA2LU_Lab5\src\tes\FlashTools\FlashGuilianiDemo.bat

Step 6.8 Execute number 3 - Flash Guiliani Resources then select 2 for the Stream It board.

Step 6.9 Open RZA1LU_Lab05 project in e2studio.

Step 6.10 Open .\RZA1LU_Lab05\src\tes\GUI_Sample\Source\MyGUI_SR.cpp file.

Step 6.11 Modifiy the *CallApplicationAPI* method code to set the CGUISlider value by adding the code below to the “if (kAPI == "Led0")” section.

The first two lines get the CGUISlider Object with the ID AID_SLIDER_1.

Using this object we can now get the value of the slider. Then use this to set the R_OS_TaskSleep time.

```
pkSlider = static_cast<CGUIBaseSlider*>(GETGUI.GetObjectByID(AID_SLIDER_1));
eC_UByte ubSlider = static_cast<eC_UByte>(eC_ToInt(pkSlider->GetRange().GetValue()));

/*Toggle Led*/
printf( "LED Button Pressed\n");

/* open LED driver */
led_handle = open( DEVICE_INDENTIFIER "led", O_RDWR);

/* LED OFF */
control(led_handle, CTL_SET_LED_OFF, &led);

/* Set time to sleep */
R_OS_TaskSleep ( ubSlider *10);

/* LED ON */
control(led_handle, CTL_SET_LED_ON, &led);


close(led_handle);
```

Step 6.12 Declare this CGUI object at the beginning of CallApplicationAPI method.

```
CGUIBaseSlider *pkSlider = NULL;
```

Step 6.13 Add the header file for the CGUISlider. This should be located under the GUI.h header.

```
#include "GUISlider.h"
```

Step 6.14 Build the project and load the project onto the RZA1 Target Board. Press Run.

Step 6.15 Once the GUI is running, press the **LED** button. The LED will turn off and on after a brief delay. Now increase the slider and press the button again and notice the LED delay is longer.

7 Lab7 Hardware POT Control to control the LED Duration

Overview:

This section you will use the POT connected to the RZA1 ADC to set the LED Duration. A GUI progress bar and text field will be added to display the ADC value

Procedural Steps

Step 7.1 Open GSE project named Lab7 located at .\RZA1LU_Lab07\src\tes\GUI_Sample\GuilianiDemo.

Step 7.2 This demo already has a GUI Button Control.

Step 7.3 Add the **Progress Bar** and **Text Field** Controls.

Step 7.4 Set the Progressbar Attributes

ID	AID_PROGRESSBAR_1
MinValue:	0
MaxValue:	100
StepSize:	1
Width	200
Height	20
BarWidth	200
BarHeight	20

- Place the progress bar anywhere on the screen you'd like.

Step 7.5 Open the Image manager. Browse to the image folder in the root directory of the workspace. Select the JPEG file named **PB_FILL**.

Step 7.6 Set the Progress Bar Image Attributes

ImageBackground :	IMG_STDCTRL_SLD_BG
ImageForeground :	IMG_PB_FILL

Step 7.7 Set the TextField Attributes

ID	AID_TEXTFIELD_1
Text	empty

- Width & TextWidth to 100

Step 7.8 Export the GSE project to .\RZA1LU_Lab07\src\tes\GUI_Sample\Include\GUIConfig.

Step 7.9 Execute the windows batch file .\ RZA2LU_Lab7\src\tes\FlashTools\FlashGuilianiDemo.bat

Step 7.10 Execute number 3 - Flash Guiliani Resources then select 2 for the Stream It board.

Step 7.11 Open **RZA1LU_Lab07** project in e2studio.

Step 7.12 Open .\RZA1LU_Lab07\src\tes\GUI_Sample\Include\MyGUI_SR.h file.

Step 7.13 Declare the following attributes and give them a private scope [near the bottom of the file].

```
uint16_t adc_val   = 0;
CGUITextField* pkTextField;
CGUIProgressBar* pkProgessBar;
```

Step 7.14 While you are in that file, also add the include for the progress bar near the top of the file:

```
#include "GUIProgressbar.h"
```

Step 7.15 Open .\RZA1LU_Lab07\src\tes\GUI_Sample\Source\MyGUI_SR.cpp file.

Step 7.16 In the MyGUI constructor add a CGUITimer Animation callback. This will set a periodic animation every 100ms for Object CMyGUI. When the timer ends the CMyGUI Object method DoAnimate is called. Later we will add the DoAnimate.

```
// add callback for polling RTC
GETTIMER.AddAnimationCallback(100, this);
```

Step 7.17 Also in the constructor, set the CGUITestField and CGUIProgressbar Objects to the control object IDs.

```
pkTextField = static_cast<CGUITextField*>(GETGUI.GetObjectByID(AID_TEXTFIELD_1));
pkProgessBar = static_cast<CGUIProgressBar*>(GETGUI.GetObjectByID(AID_PROGRESSBAR_1));
```

Step 7.18 Edit the CMyGUI method *DoAnimate*. Insert the following code.

```
char data_str[32];
uint32_t data = 0;

R_ADC_Open();
adc_val = (uint32_t)R_ADC_Read();

R_ADC_Close();

sprintf(data_str, "%d", (uint16_t)adc_val);
pkTextField->SetLabel(data_str);

data = (uint32_t)((((float32_t)adc_val)/ADC_MAX_VALUE)*100);

pkProgressBar->SetValue(data);
pkProgressBar->InvalidateArea();
```

Step 7.19 Edit the *CallApplicationAPI* method to read the ADC driver APIs. Then use the value read to set the LED duration. This code should be inside the “if (kAPI == "Led0")” section.

```
/* open LED driver */
led_handle = open( DEVICE_INDENTIFIER "led", O_RDWR);

/* LED OFF */
control(led_handle, CTL_SET_LED_OFF, &led);

R_OS_TaskSleep( adc_val / 2 +100);

/* LED ON */
control(led_handle, CTL_SET_LED_ON, &led);

close(led_handle);
```

Step 7.20 Add #include for the adc driver header. This is a “C” header so it needs to be declared inside the “extern”C” { “block.

```
#include "r_adc.h"
```

Step 7.21 Build the project and load the project onto the RZA1 Target Board. Run the project.

Step 7.22 Press the Button and watch the LED delay duration. **Turn the POT** CW/CCW and press the button again. The LED duration will increase/decrease. As the POT changes the TextField and Progress Bar will also change.

8 Lab8 GUI USB HID Trackpad

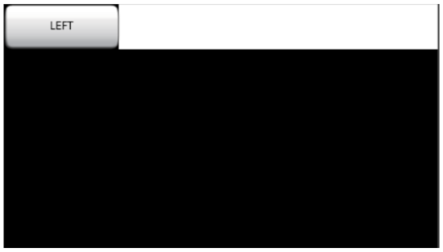
Overview:

This lab will create a PC Trackpad. This lab will use the USBF HID driver to implement the mouse left button press and mouse position to the PC. The GUI will create a trackpad interface and capture finger drag events.

Hardware Requirements

This demo will require a male to male USB cable. This is used to connect the Streamit USB connector CN4 to a laptop PC.

Procedural Steps



Step 8.1 Create this GUI trackpad. Use the setting below to create this demo on the right.

Step 8.2 Button

- Width : 128
- Height : 50
- ObjectID : AID_BUTTON_1
- Text : LEFT
- TextXPos : 95
- CallApplicationAPICmd ->
  - CommandClassID : CMD_CALLAPPLICATIONAPI
  - ApplicationAPI : leftClick
  - Parameter : Pressed

Step 8.3 White Geometric Object

- Xpos : 127
- YPos : 0
- Width : 353
- Height : 50
- Color : White

Step 8.4 Black Geometric Object

- Xpos : 0.0
- YPos : 0.0
- Width : 480
- Height : 272
- Color : Black

Step 8.5 Export the GSE project to .\RZA1LU_Lab08\src\tes\GUI_Sample\Include\GUIConfig.

Step 8.6 Execute the windows batch file .\ RZA2LU_Lab08\src\tes\FlashTools\FlashGuilianiDemo.bat

Step 8.7 Execute number 3 - Flash Guiliani Resources then select 2 for the Stream It board.

Step 8.8 Coding

Step 8.9 Open **RZA1LU_Lab08** project in e2studio.

Step 8.10 Enable the USB Driver and USB HID. Open the header file application_cfg.h located in ./src/renesas/configuration directory. Set the `R_SELF_INSERT_APP_HID_MOUSE to R_OPTION_ENABLE.` This will enable the USB HID Renesas Devlink API for USB HID.

Step 8.11 Open the MyGUI.cpp header file.

Step 8.12 On the defineistion of the CMyGUI class we need to inherit the CGUIBehavior and CGUIObserver. This will allow us to capture touch event and button events respectively through the Tes Guiliani platform.

```
class CMyGUI : public NStreamRuntime::CStreamRuntimeGUI, public CGUIObserver, public
CGUIBehaviour
```

Step 8.13 Declare the CGUIBehavior virtual method DoDrag to the CMyGUI class and give it a public scope. This function will handle the drag touch events.

```
virtual eC_Bool DoDrag(const eC_Value& vDeltaX, const eC_Value& vDeltaY, const eC_Value&
vAbsX, const eC_Value& vAbsY );
```

Step 8.14 Open the MyGUI_SR .cpp source file. Define the DoDrag method.

```
eC_Bool CMyGUI::DoDrag(const eC_Value& vDeltaX, const eC_Value& vDeltaY, const eC_Value&
vAbsX, const eC_Value& vAbsY )
{
return true;
}
```

Step 8.15 In the CMyGUI_SR.hpp header create this structure type. We will use this to define the message that will be sent to the USB HID Driver.

```
typedef struct usbM_t {
            bool_t left_button;
            int xPos;
            int yPos;
} usbM_t;
```

Step 8.16 Next Declare the following variables with **private** scope.

```
private:
   usbM_t usbMouseMsg;
   os_msg_queue_handle_t m_USBMouseQ;
   bool_t m_touchPressed = false;
```

Step 8.17 Declare the FreeRTOS header files in the MyGUI_SR.h header file.

```
#include "FreeRTOS.h"
#include "task.h"
```

```
#include "queue.h"
```

Step 8.18 Now we need to initialize the USB HID Driver. To do this we are going to use a modified version of the Console Application r_usb_hid_mouse_app.c/h. This source file is located in the supplemental directory ./LabSupportFile/Lab8. Add these to this RZA1LU_Lab8 project.

- Source file ./RZA1LU_Lab8/src/Renesas/application/src
- Header file /RZA1LU_Lab8/src/Renesas/application/inc

Step 8.19 In the MyGUI_SR.cpp file add the following function call in the CMyGUI constructor Create a RTOS message queue to handle mouse data sent to the USB driver task. We will set the queue size to 100. Set the data size to the size to of the usbM_t structure declared previously.

```
        // Create FreeRTOS message Queue
        m_USBMouseQ = xQueueCreate( 100, sizeof(usbM_t) );
```

Step 8.20 Next, Call this function to initialize the USB as a HID mouse. This queue will pass the message to the USB HID task.

```
// Initialize USB Mouse
gui_hid_mouse_init( m_USBMouseQ );
```

Step 8.21 In the DoDrag method we defined previously add the following code. The if (vAbsY > 50) filters any touch events that are located where the buttons are located at the top of the screen. The vDeltaX and VDeltaY are the relative movement in the x and y direction respectfully since the last call of DoDrag/ButtonDown.

```
if ( vAbsY > 50) {
        usbMouseMsg.left_button = false;
        usbMouseMsg.xPos = vDeltaX;
        usbMouseMsg.yPos = vDeltaY;
        xQueueSend( m_USBMouseQ, (const void *)&usbMouseMsg, 0 );
}
```

Step 8.22 Next we will handle button click events. In the CallApplicationAPI function add the following code.

```
if (kAPI == "leftClick") {

        usbMouseMsg.left_button = true;

usbMouseMsg.xPos = 0;
        usbMouseMsg.yPos = 0;
        xQueueSend( m_USBMouseQ, (const void *)&usbMouseMsg, 0 );

}
```

Step 8.23 Build the project and load the project onto the RZA1 Target Board. Run the project.

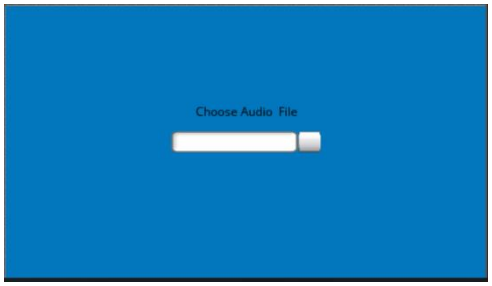
Step 8.24 Connect the USB cable form the PC to the target board. Press reset. You can now move the mouse pointer by dragging your finger in the black area. Press left button to click on the screen.

9 Lab11 Create a Complex

Overview:
This section you will implement a complex GUI control called Combo Box. This control will be used in later labs for file selection.

Procedural Steps

Step 9.1 Create the GUI displayed to the right using the Setting described below.



Step 9.2 Dialog

- Name : main
- Width : 480
- Height : 272

Step 9.3 Background (Geometry Object)

- XPos : 0
- YPos : 0
- Width : 480
- Height : 272
- Color : Blue

Step 9.4 Combo Box

- XPos : 165
- YPos : 125
- Width : 150
- Height : 22
- ObjectID : AID_COMBOBOX_1

Step 9.5 Text Field

- XPos : 158.0
- YPos : 96.0
- Width : 165.0
- Height : 22.0
- ObjectID : AID_TEXTFIELD_1
- Test : Choose File

Step 9.6 Export the GSE project to .\RZA1LU_Lab09\src\tes\GUI_Sample\Include\GUIConfig.

Step 9.7 Execute the windows batch file .\ RZA2LU_Lab09\src\tes\FlashTools\FlashGuilianiDemo.bat

Step 9.8 Execute number 3 - Flash Guiliani Resources then select 2 for the Stream It board.

Step 9.9 Coding

Step 9.10 Open **RZA1LU_Lab09** project in e2studio.

Step 9.11 Open File .\RZA1LU_Lab09\src\tes\GUI_Sample\Source\MyGUI_SR.cpp.

Step 9.12 First inside the CMyGUI constructor, obtain the Combo Box Object that was inserted in the GSE project. We will do this by calling the GetObjectByID using the Object ID we set for the Combo Box **AID_COMBOBOX_1.**

```
// Get Object pointer to ComboBox
m_pComboBox =  dynamic_cast<CGUIComboBox*>(GETGUI.GetObjectByID(AID_COMBOBOX_1));
```

Step 9.13 In order to populate the Combo box with a list of items to select we need to Create a ListItem for each selection in the combo box, set the format, color, and content. First make the content that will be past to the Combo box.

```
const eC_Char* kContent[] = { "Earth", "Mars", "Venus", "Jupiter", "Saturn" };
```

Step 9.14 Next a CGUIListItem needs to be created for each Combo Box Item and add it to the Combo Box. We will do this by iterating over each content item defined in the previous step.

```
for(int i=0;i<5;i++)
{
    CGUIListItem * pListItem =
       new CGUIListItem(NULL, 0, 0, eC_FromInt(80), eC_FromInt(20), kContent[i] );

    // Format the Item Center and set color
    pListItem->GetLabel()->SetAligned(CGUIText::V_CENTERED);
    pListItem->GetLabel()->SetTextColor(0xff000000, 0xffffffff, 0xff000000,
0xffffffff);

    // Add it to the ComboBox
    m_pComboBox->AddItem(pListItem);
}
```

Step 9.15 After the for loop add the following lines of code. These will configure the the Combo Box visualization.

```
// Set some visualization parameters. (e.g. Colors and Images)
m_pComboBox->SetItemSelectedColor(0xff0000cc);
m_pComboBox->SetHeaderButtonImages(
    IMG_STDCTRL_IMGBTN_STANDARD,
    IMG_STDCTRL_IMGBTN_PRESSED,IMG_STDCTRL_IMGBTN_HIGHLIGHTED,
    IMG_STDCTRL_IMGBTN_GRAYED_OUT,IMG_STDCTRL_IMGBTN_FOCUSED);
m_pComboBox->GetHeader()->SetInputFieldImages(
    IMG_STDCTRL_INPUTFIELD_STANDARD,
    IMG_STDCTRL_INPUTFIELD_HIGHLIGHTED,
    IMG_STDCTRL_INPUTFIELD_FOCUSSED,
    IMG_STDCTRL_INPUTFIELD_GRAYEDOUT);
```

Step 9.16 Set the Default Selection in the Combo box. This is the first item that is seen at startup.

```
// Select an item by index
m_pComboBox->SetSelection(2);
// Make the comboxbox-header editable, so that users can enter a search-string
m_pComboBox->SetHeaderEditable(true);
```

Step 9.17 Build the project and load the project onto the RZA1 Target Board. Run the project.

Step 9.18 You should now be able to select the planet from the Combo box.

10 Lab10 List Files from USB MSC Drive

Overview:
The section we explore populate the Combo Box that was created in the previous lab and with text files found in an attached USB stick.

Procedural Steps

Step 10.1 In this section we will not need to create a new GUI but use the same one from the previous lab. If the target board is does not have this GUI do the next steps otherwise skip to step

Step 10.2 Execute the windows batch file .\ RZA2LU_Lab09\src\tes\FlashTools\FlashGuilianiDemo.bat

Step 10.3 Execute number 3 - Flash Guiliani Resources then select 2 for the Stream It board.

Step 10.4 Setup the USB MSC.

Step 10.5 The RZA1LU SDK POSIX software layer called r_devlink includes API's for setting up the USB MSC. To setup the define the usb posix handle and call open. This is already implemented in main.c. To enable it Open application_cfg.h header file and set R_SELF_LOAD_MIDDLEWARE_USB_HOST_CONTROLLER (R_OPTION_DISABLE) to R_OPTION_ENABLE. This code is called before the GUI Task is created so the USB Host task is run.

```
int_t usb0_handle = ( -1);
usb0_handle = open( DEVICE_INDENTIFIER "usb0", O_RDWR);
```

Step 10.6 Coding GUI Backend

Step 10.7 Create a new member function called CreateFileList. It's return type is void and it will take a contant eC_Char pointer. declare it in the MyGUI_SR.hpp file and define it in the MyGUI_SR.cpp file.

```
// Header file
void CreateFileList ( const eC_Char* pattern );
```

```

//Source
void CMyGUI::CreateFileList ( const eC_Char* pattern ) {

}
```

Step 10.8 Copy the Code in the Constructor that we used to make the Combo Box into the CreateFileList. The Function should look like this.

```
void CMyGUI::CreateFileList ( const eC_Char* pattern ) {

        const eC_Char* kContent[] = { "Earth", "Mars", "Venus", "Jupiter", "Saturn" };
        for(int i=0;i<5;i++)
        {
          CGUIListItem * pListItem =
                new CGUIListItem(NULL, 0, 0, eC_FromInt(80), eC_FromInt(20), kContent[i] );

          // Modify the newly created item's label
          pListItem->GetLabel()->SetAligned(CGUIText::V_CENTERED);
          pListItem->GetLabel()->SetTextColor(0xff000000, 0xffffffff, 0xff000000, 0xffffffff);
          // Add it to the ComboBox
          m_pComboBox->AddItem(pListItem);
        }

        // Set some visualization parameters. (e.g. Colors and Images)
        m_pComboBox->SetItemSelectedColor(0xff0000cc);
        m_pComboBox->SetHeaderButtonImages(
          IMG_STDCTRL_IMGBTN_STANDARD,
          IMG_STDCTRL_IMGBTN_PRESSED,IMG_STDCTRL_IMGBTN_HIGHLIGHTED,
          IMG_STDCTRL_IMGBTN_GRAYED_OUT,IMG_STDCTRL_IMGBTN_FOCUSED);
        m_pComboBox->GetHeader()->SetInputFieldImages(
          IMG_STDCTRL_INPUTFIELD_STANDARD,
          IMG_STDCTRL_INPUTFIELD_HIGHLIGHTED,
          IMG_STDCTRL_INPUTFIELD_FOCUSSED,
          IMG_STDCTRL_INPUTFIELD_GRAYEDOUT);
        // Select an item by index
        m_pComboBox->SetSelection(2);
        // Make the comboxbox-header editable, so that users can enter a search-string
        m_pComboBox->SetHeaderEditable(true);
}
```

Step 10.9 Add the following C headers to the MyGUI_SR.cpp file. These are C headers so they need to be encapsulated in by the extern "C" { }. This will avoid C++ naming collisions. These header provide the API for the USB driver and FatFS middleware.

```
#include "r_devlink_wrapper_cfg.h"
#include "dskManager.h"
#include "r_fatfs_abstraction.h"
#include "ff.h"
```

Step 10.10 In the CMyGUI constructor we need to mount the USB MSC device. In this Lab we will assume that the device is already connected.

```
// Mount all Devices
if ( 0 < dskMountAllDevices() ) {

        f_chdrive("A:\\");
        // Query for Text files
        CreateFileList ("*.txt");
}
```

Step 10.11 At the begging of the CreateFileList function we need to open FAT Filesystem at the root directory of the USB by calling dskGetDrive, then find the first file with R_FAT_FindFirst.

```
void CMyGUI::CreateFileList ( const eC_Char* pattern ) {

       char full_path[] = “A:\\”;
       int iCount = 0;

       // Get pointer to FSFAT drive
       void *pDrive = dskGetDrive(‘A’);

       if ( pDrive == NULL)
              return;

       FATENTRY fatEntry;
       FRESULT fatResult;

       DIR dir;

      fatResult = R_FAT_FindFirst( &dir, &fatEntry, full_path, pattern);
```

Step 10.12 Now we want to change the for loop to a while loop that loops till the return value ‘fatResult’ is zero.

```
Change this :
for(int i=0;i<5;i++) {
to this :
while ( FR_OK == fatResult ) {
```

Step 10.13 Remove the structure kContent and change the code for creating a CGUIListItem so it takes the filename.

```
CGUIListItem * pListItem =
new CGUIListItem(NULL, 0, 0, eC_FromInt(80), eC_FromInt(20), fatEntry.FileName );
```

Step 10.14 Build the project and load the project onto the RZA1 Target Board. Run the project.

Step 10.15 From the drop-down list of the Combo Box you can see that the list of all files with the extension *.txt. NOTE : if no file listed eject your device and create a few text files in the root directory.

11 Lab11 Play Sound from Serial Flash

Overview:

This section explains how to play sound located in Serial Flash by using the I^2S with DMA using the R_SOUND driver API. A GUI design will control the play and stop.

Audio Software

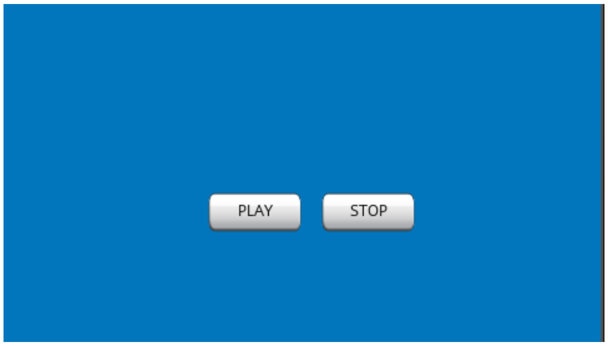
Included in this lab is the R_SOUND application demo for GUI. This code runs an audio player application task that will handle play the audio file in the background and the GUI task will run in the foreground. The application is similar to the R_SOUND application for the console demos that is part of the RZA1LU FreeRTOS SDK on the Renesas Website. The files are located in the Lab project at ./RZA1LU_Lab11/src/renesas/application/app_sound.

For this Lab we include the audio file snippet. This file is named LR_44_1K16B_S.dat and is located at ./RZA1LU_Lab11/src/renesas/application/app_sound/inc. This audio snippet is sampled at 44.1K hz with 16bit samples.

Procedural Steps

Step 11.1 GUI Design

Create the GUI to the right using the setting listed below.



Step 11.2 Dialog

- Name : main
- Width : 480
- Height : 272

Step 11.3 Background

- Create Geometry Object
- XPos : 0.0
- YPos : 0.0
- Width : 480.0
- Height : 272.0
- Color : Blue

Step 11.4 Button (Play)

- XPos : 165.0
- YPos : 152.0
- Width : 75.0
- Height : 32.0
- ObjectID : AID_BUTTON_1
- Text : PLAY
- DummyClassID : CMD_APPLICAITONAPI (Default : DUMMY_COMMAND)
    - ApplicationAPI : media_play
    - Parameter : 0

Step 11.5 Button (Stop)

- XPos : 256.0
- YPos : 152.0
- Width : 75.0
- Height : 32.0
- ObjectID : AID_BUTTON_2
- Text : Stop
- DummyClassID : CMD_APPLICAITONAPI (Default : DUMMY_COMMAND)
    - ApplicationAPI : media_stop
    - Parameter : 0
-

Step 11.6 Export the GSE project to .\RZA1LU_Lab11\src\tes\GUI_Sample\Include\GUIConfig.

Step 11.7 Execute the windows batch file .\ RZA2LU_Lab11 \src\tes\FlashTools\FlashGuilianiDemo.bat

Step 11.8 Execute number 3 - Flash Guiliani Resources then select 2 for the Stream It board.

Step 11.9 Coding GUI Backend

Step 11.10 Open .\RZA1LU_Lab11\src\tes\GUI_Sample\Source\MyGUI_SR.cpp

Step 11.11 Add the header r_sound.h. To avoid C++ naming collision, this C header file needs to be encapsulated in an extern "C" block.

```
#include "r_sound.h"
```

Step 11.12 In the CMyGUI constructor add this function. This will initialize the SSIF driver and task as well as the application sound play task.

```
R_SOUND_PlaySample_init();
```

Step 11.13 Now let's modify the CallApplicationAPI. Using the kAPI that were defined in the GUI Design portion of this Lab. In the next steps we will define R_SOUND_PlaySample and R_SOUND_StopSample.

```
if ( "media_play" == kAPI ) {
        char buf[80];
        R_SOUND_PlaySample();


} else if ( "media_stop" == kAPI ) {
        R_SOUND_StopSample();
}
```

Step 11.14 Coding Sound GUI Interface

Step 11.15 Open ./RZA1LU_Lab11/src/renesas/application/app_sound/src/r_sound.c and ./RZA1LU_Lab11/src/renesas/application/app_sound/inc/r_sound.h.

Step 11.16 In the r_sound.h file add the following functions. These provide hooks into the application sound task.

```
void R_SOUND_PlaySample(void);
void R_SOUND_StopSample(void);
```

Step 11.17 For this demo we will use the RTOS event flags to signal a play and stop event. In the st_sound_config_t typedef structure add these RTOS event flags.

```
event_t  task_play;
event_t  task_stop;
```

Step 11.18 Modify the application sound interface initialization function initalise_control_if to create the events task_play and task_stop.

```
…
R_OS_CreateEvent( &gsp_sound_control_t->task_running);
R_OS_CreateEvent( &gsp_sound_control_t->task_play);
R_OS_CreateEvent( &gsp_sound_control_t->task_stop);
```

Step 11.19 Create the functions R_SOUND_PlaySample and R_SOUND_StopSample. The functions will set the events we created above.

```
void R_SOUND_PlaySample (void)
{
        if ( gsp_sound_control_t->task_play != NULL )
                R_OS_SetEvent( &gsp_sound_control_t->task_play);
}

void R_SOUND_StopSample (void)
{
        if ( gsp_sound_control_t->task_stop != NULL )
                R_OS_SetEvent( &gsp_sound_control_t->task_stop);
}
```

Step 11.20 Now we will use these events to control the task_play_sound_demo. At the beginng of the task add these lines of code. This resets the events before the task loop is entered.

```
R_OS_ResetEvent( &gsp_sound_control_t->task_play);
R_OS_ResetEvent( &gsp_sound_control_t->task_stop);
```

Step 11.21 Next at the beginning of the infinite while loop (while (1)) we want to wait on a ‘play’ event before playing the song. We agian reset the ‘stop’ event for subsequent audio replay.

```
R_OS_WaitForEvent( &gsp_sound_control_t->task_play, R_OS_ABSTRACTION_PRV_EV_WAIT_INFINITE);
// Reset any pending stop requests
R_OS_ResetEvent( &gsp_sound_control_t->task_stop);
```

Step 11.22 To control when to stop the song, insert the following code. This will check to see if the 'stop' event is toggled. This needs to be a non-pending check of the event so as not to add glitches to the current play due to the DMA buffer becoming empty. We also reset the 'play and 'stop' events.

```
if ( R_OS_EventState( &gsp_sound_control_t->task_stop ) == EV_SET ) {
            R_OS_ResetEvent( &gsp_sound_control_t->task_play);
            R_OS_ResetEvent( &gsp_sound_control_t->task_stop);
            break;
}
```

Step 11.23 As part of the clean up. When the task loop is exited we need to delete all events. This should be located after the infinite while loop.

```
R_OS_DeleteEvent(&gsp_sound_control_t->task_stop);
R_OS_DeleteEvent(&gsp_sound_control_t->task_play);
```

Step 11.24 Build the project and load the project onto the RZA1 Target Board. Run the project.

Step 11.25 Connect the Speaker to the Audio connector CN4.

Step 11.26 Press the GUI Play button. You should here the audio file playing. Press stop to discontinue. Press play again the song will restart from the beginning.

Challenge

Add a pause button to the GUI and pause event to the sound application. The button will stop playing then restart from where it left off.

12 Lab12 Add Progress Bar

Overview:

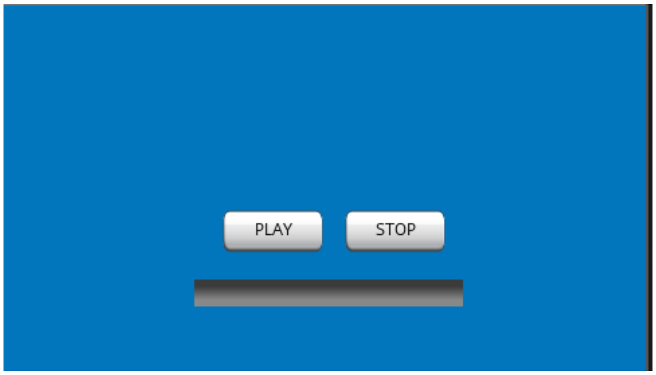
In this lab we will add a progress bar to show music current position. This Lab will build off Lab 11 Play Audio file from Serial Flash.

Procedural Steps

Step 12.1 GUI Design

Step 12.2 Progress Bar

- XPos : 144.0
- YPos : 204.0
- Width : 200.0
- Height : 20.0
- ObjectID : AID_PROGRESSBAR_1
- Focusable : Disable
- MinValue : 0
- MaxValue : 100



Step 12.3 Coding

Step 12.4 Open .\RZA1LU_Lab12\src\tes\GUI_Sample\Include\MyGUI_SR.h

Step 12.5 Add the C++ header for CGUI Progress Bar support.

```
#include "GUIProgressbar.h"
```

Step 12.6 Add the C header for the RTOS queue. Insert this inside the extern “C” block code.

```
#include "FreeRTOS.h"
#include "r_os_abstraction_api.h"
#include "queue.h"
#include "r_task_priority.h"
```

Step 12.7 The CMyGUI needs to inherit the CGUIObserver class.

```
class CMyGUI : public NStreamRuntime::CStreamRuntimeGUI, public CGUIObserver
```

Step 12.8 Declare a progress bar variable of CGUIProgressBar pointer type and declare the FreeRTOS Queue. The later will be used to handle messages from the audio sound application.

```
CGUIProgressBar* m_qAudioTrackInfo;
QueueHandle_t* m_qAudioTrackInfo;
```

Step 12.9 Override the DoAnimate method by declaring it in the CMyGUI class.

```
virtual void DoAnimate(const eC_Value &vTimes);
```

Step 12.10 Open .\RZA1LU_Lab12\src\tes\GUI_Sample\Source\MyGUI_SR.cpp

Step 12.11 In the CMyGUI constructor add the GETTIMER.AddAnimationCallback function. This sets the CGUIAnimation updates to 25 ms.

```
GETTIMER.AddAnimationCallback(25, this);
```

Step 12.12 In the CMyGUI constructor initialize the m_qAudioTrackInfo queue. This queue is used to receive track information form the R_SOUND task.

```
m_qAudioTrackInfo = xQueueCreate(10, sizeof(uint32_t));
```

Step 12.13 The CGUIProgressBar requires updates are done using CGUIAnimaiton. Define the DoAnimate function. This will periodically ping the RTOS queue for updates to the Progress Bar.

```
void CMyGUI::DoAnimate( const eC_Value &vTimers) {
}
```

Step 12.14 In the CMyGUI constructor initialize the pkProgressBar with the GETGUI.GetObjectByID method.

```
pkProgessBar = static_cast<CGUIProgressBar*>(GETGUI.GetObjectByID(AID_PROGRESSBAR_1));
```

Step 12.15 Create the following DoAnimate method. This method is periodically called. If a message is received the progress bar is updated.

```
void CMyGUI::DoAnimate( const eC_Value &vTimers) {
        uint32_t track;

        if ( pdPASS == xQueueReceive ( m_qAudioTrackInfo, &track, 0 )) {

                /* Set the Progress range from 0 -100 */
                if ( NULL != m_pkProgressBar ) {

                        m_pkProgressBar->SetValue(track);
                        m_pkProgressBar->InvalidateArea();
                }
        }
}
```

Step 12.16 Open the .\RZA1LU_Lab12\src\renesas\application\app_sound\inc\r_sound.h.

Step 12.17 Add RTOS headers.

```
#include "FreeRTOS.h"
#include "r_os_abstraction_api.h"
#include "queue.h"
#include "r_task_priority.h"
```

Step 12.18 Change R_Sound_PlaySample_init declaration.

```
void R_SOUND_PlaySample_init( QueueHandle_t q_AudioTrack );
```

Step 12.19 Open the .\RZA1LU_Lab12\src\renesas\application\app_sound\src\r_sound.c.

Step 12.20 Modify the structure type st_sound_config_t with the following variable element.

```
  ….
   os_msg_queue_handle_t track_queue;

} st_sound_config_t;
```

Step 12.21 Modify the initialize_control_if function to pass a parameter

```
static void initalize_control_if (QueueHandle_t q )  {
```

Step 12.22 Add initialization of the q in the initialize_control_if function.

```
gsp_sound_control_t->track_queue = q;
```

Step 12.23 Change the definition of the R_SOUND_PlaySample_init function to take QueueHandle_t type parameter.

```
void R_SOUND_PlaySample_init( QueueHandle_t q_AudioTrack ) {
```

Step 12.24 Change call to the initalize_control_if to pass the q_AudioTrack variable. NOTE : make this change to all the calls to this function. You can use NULL as a parameter for other calls.

```
initalize_control_if( q_AudioTrack );
```

Step 12.25 Modifiy the task task_play_sound_demo declare the variable task type uin32_t and wave_length with type float32_t.

```
uint32_t track = 0;
float32_t wave_length = 0.0;
```

Step 12.26 Set the wave_length to the number of iteration. This is the size of the audio sample divided by the DMA sample size.

```
wave_length = (float32_t)(SIZEOF_WAVEDATA_PRV_ / WAVE_DMA_SIZE_PRV_);
```

Step 12.27 Modify the task task_play_sound_demo to send audio track information. The following code needs to be located after the write to the SSIF driver. The track information will need to normalize from 1 to 100.

```
/* Send the Wave File tracking info */
track = (uint32_t)((loop / wave_length) * 100);
xQueueSend ( gsp_sound_control_t->track_queue, (void*)&track, 0);
```

Step 12.28 After the audio playback has finished reset the track information to zero. This code is locate at outside and after the for loop.

```
track = 0;
xQueueSend ( gsp_sound_control_t->track_queue, (void*)&track, 0);
```

13 Lab13 Play Sound From USB MSC

Overview:

The lab is a combination of the last three labs it. In this lab we will use the Combo Box we created in Lab 10 to view all WAV audio files, the R_SOUND interface developed in Lab 11 and the progress bar created in Lab 12.

Procedural Steps

Step 13.1