AN13110

LPC54608 Crank Storyboard Engine Integration with IAR

Rev. 0 — January 15, 2021 Application Note

by: NXP Semiconductors

Contents

1 Introduction

The LPC54608 is a family of Arm® Cortex®-M4 based microcontrollers used in embedded applications. As more and more embedded applications require Graphical User Interfaces (GUI), more GUI tools are introduced to developers. Crank Storyboard is a powerful GUI development suite. Storyboard Engine is the runtime component that delivers the content developed in Storyboard Designer to embedded devices.

This application note demonstrates how to integrate Storyboard engine to LPC54608 base on freeRTOS.

2 Creating an application by Storyboard designer

2.1 Storyboard designer description

Storyboard Designer enables UI designers to easily prototype the look and feel of a product and then deploy the prototype to an embedded target. Designers maintain full control over the UI without having to perform a hand off to an engineer for implementation.

2.2 Creating an application

2.2.1 Creating a new Storyboard Designer project

Create a new project and configure the parameters from the hardware.

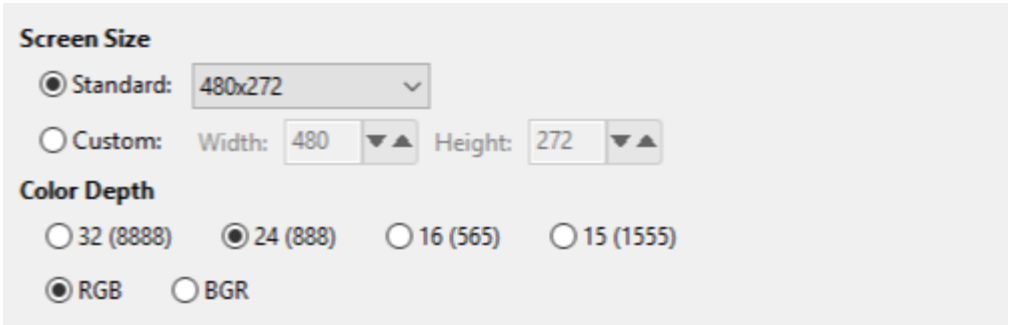

Figure 1. LCD parameter configurations

2.2.2 Adding assets to the application and create screens

To display different contents, we can create different screens with different assets, such as images, controls, and so on.

In this application, there are two screens.

- `screen_meter_start` is shown after the system starts up, as shown in Figure 2.

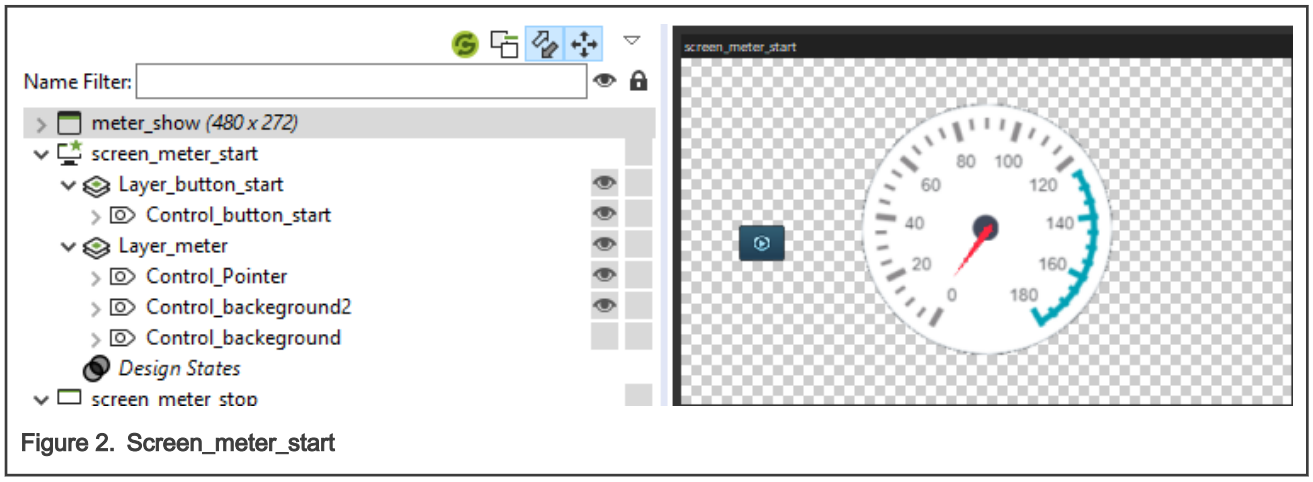

Figure 2. Screen_meter_start

- `screen_meter_stop` is shown after running from `screen_meter_start`, as shown in Figure 3.

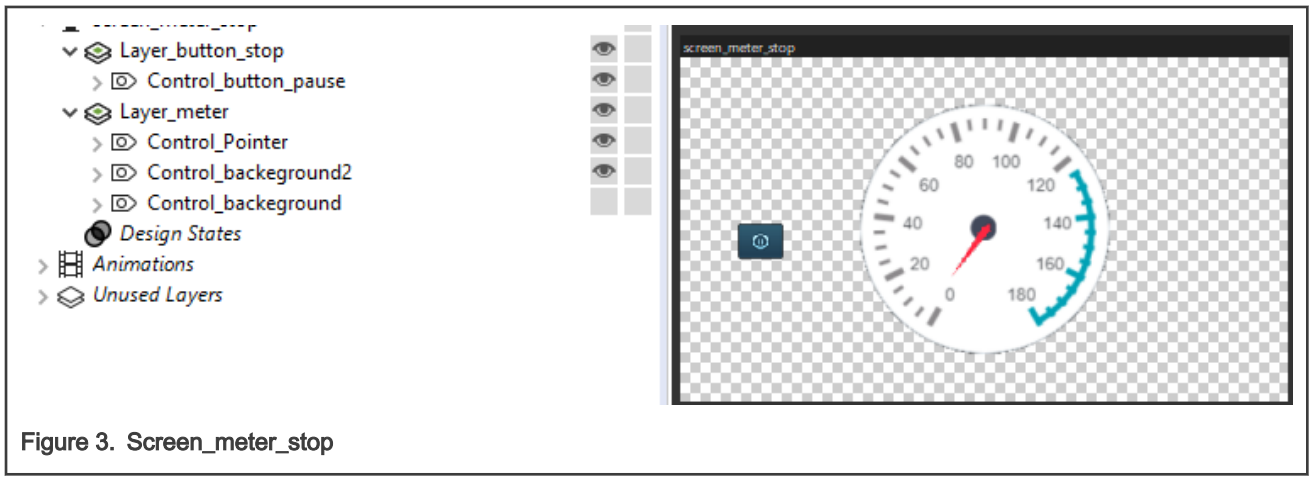

Figure 3. Screen_meter_stop

2.2.3 Creating pointer rotation animation

In the demonstration, when users click start button, the pointer will point from 0 to 180, and the **Start** button will change to the **Pause** button. If users click the **Pause** button, the pointer will point from 180 to 0. So we shall create two animations to display the pointer rotation.

It's very easy to create animation by Storyboard Designer.

1. Click **Animation** -> **Start Recording New Animation** to start and modify the rotation angle from -150 to 150.
2. Click **Animation** -> **Stop Recording Animation** to stop and save the new animation.
3. Click **Preview Animation** in **Animation Timeline** to preview.
4. Follow the same steps to create the second animation, and the only difference is the pointer rotation angle changes from 150 to -150.

2.2.4 Adding action to buttons

Add animation to press button action. If the button is pressed, it will call the animation created above.

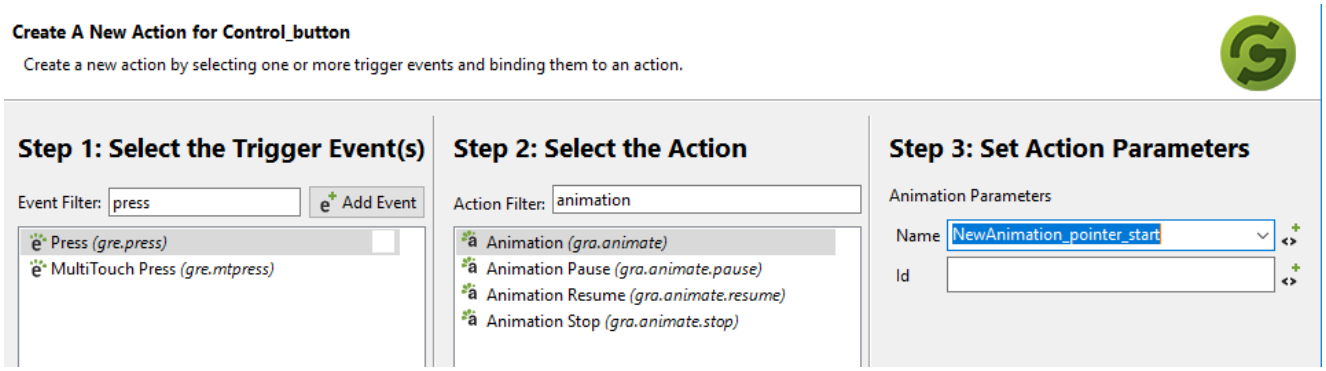

Figure 4. Add animation to Control button

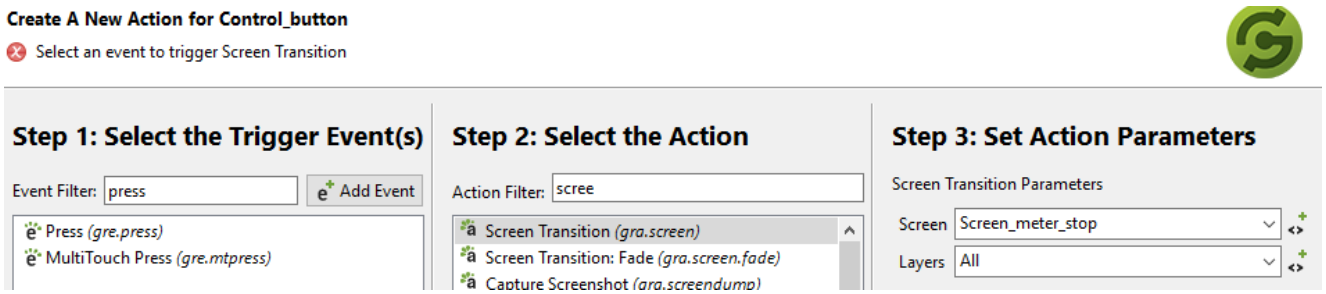

Figure 5. Add screen transition to Control button

2.2.5 Adding actions to animation events

In this application, click the **Play** button, and the pointer starts to rotate and the **Play** button canges to **Pause** button. Then, click the **Pause** button, and the pointer rotates to **0** and the Pause button changes to **Play** button.

To implement this feature, add actions to animation complete events.

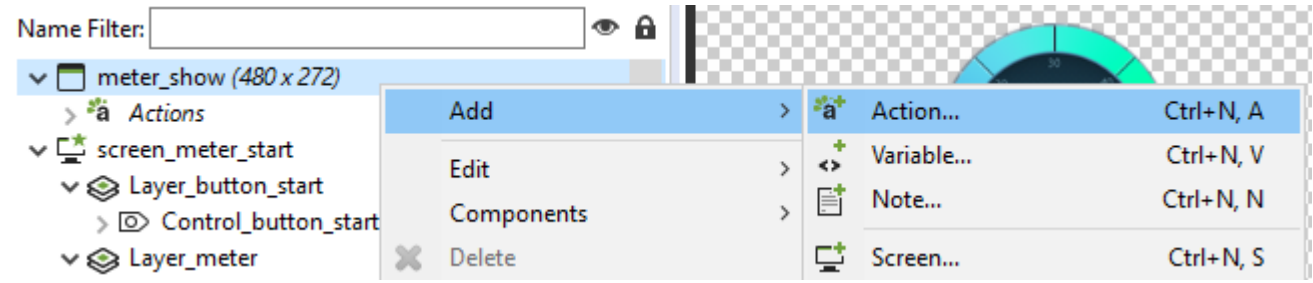

Figure 6. Add actions to animation events

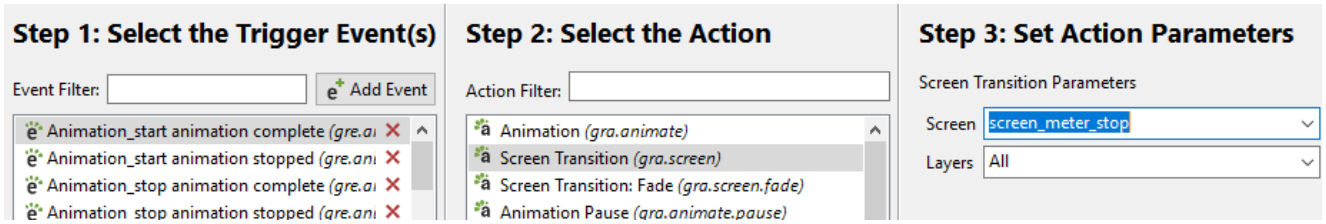

Figure 7. Add Screen Transition to Animation_start animation complete

2.2.6 Application export

So far, the demonstration Storyboard application is done. Click **Run** -> **Storyboard Application Export** and select **Packager** -> **Storyboard Embedded Resource Header (c/c++)** to generate a header file named **sbengine_model.h**, which will be used in LPC54608 project.

3 Porting Storyboard Engine to LPC54608

3.1 Hardware description

The LPC54608-EVK development board is designed to enable evaluation of and prototyping with the LPC54608 MCU devices leveraging the high-performance Arm Cotex-M4 core and the advanced security capabilities featured in the LPC54608 MCU family.

- 128 Mb MX25L12835FM2I-10G Quad-SPI flash
- Winbond 128 Mb W9812G6JB-6I SDRAM
- 480×272 capacitive touch TFT LCD screen

3.2 Crank software library

User who want to re-produce this application can access CRANK to get the library for Cortex-M4.

3.3 Installing SDK for LPC54608 from NXP

Create an NXP online account before downloading NXP SDK. You can create your account through the Dashboard tool and it is customizable for a particular IDE.

1. Configure and download the latest SDK (current 2.8.2 for LPC54628) from the Dashboard tool. Select the LPC54628 from the list of boards and then build the SDK for that board.
2. Request the IAR IDE for the SDK.
3. Be sure to include Amazon Freertos and emWIN for the SDK.
4. Download SDK.

3.4 Integrating Crank Storyboard engine to freertos_hello project

3.4.1 Porting LCD and Touch drivers

To display GUI content, initialize the LCD and its touch feature.

- Add *fsl_lcdc.c* and *fsl_lcdc.h* to driver group to initialize LCD controller.
- Add *fsl_i2c.c* and *fsl_i2c.h* to driver group to access touch chip of LCD.
- Add *fsl_sctimer.c* and *fsl_sctimer.h* to driver group to create PWM for LCD.
- Add a new group named **fsl_ft5406** which includes *fsl_ft5406.c* and *fsl_ft5406.h*.
- Add *fsl_ft5406* directory to **Project** -> **Options** -> **C/C++ Compiler** -> **Preprocessor** -> **Additional include directories**.
- Create *peripheral.c* and *peripheral.h* to implement initialization functions for LCD and touch. And add these two files to project.

3.4.2 Crank Storyboard engine integration

- Copy `freertos-iar-cortexm4-swrender-obj` released by Crank to the project root.
- Create new group named **sbengine** for Crank Storyboard engine.

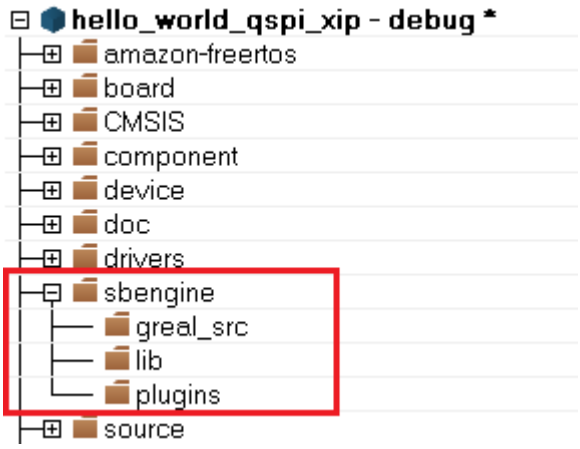

Figure 8. New groups for Crank Storyboard engine

- Add all files located in *freertos-iar-cortexm4-swrender-obj\src\lib\greal\freertos* to **greal_src** group.
- Add all files except *libgreal.a* located in *freertos-iar-cortexm4-swrender-obj\lib* to **lib** group.
- Add all files located in *freertos-iar-cortexm4-swrender-obj\plugins* to plugins group.
- Drag and drop the Storyboard engine template files from:

  freertos-iar-cortexm4-swrender-obj\src\sbengine_freertos\sbengine_task.c

  freertos-iar-cortexm4-swrender-obj\src\sbengine_freertos\sbengine_plugins.h

  to the source folders in the project:

  boards\lpcxpresso54628\rtos_examples\freertos_hello \sbengine_task.c

  boards\lpcxpresso54628\rtos_examples\freertos_hello \sbengine_plugins.h
- Add all the including path/directories to the project for Storyboard engine.

Figure 9. Including paths/directories

- Add defined symbols to **Object** -> **Options** -> **C/C++ Compiler** -> **Defined symbols**.

```
FSL_RTOS_FREE_RTOS
GRE_TARGET_CPU_cortexm4
GRE_TARGET_TOOLCHAIN_iar
GRE_TARGET_OS_freertos
GRE_ENABLE_STATIC_PLUGINS
GRE_FEATURE_VFS_RESOURCES
```

- Copy sbengine_model.h created by Storyboard Designer at to the source folder in the project.
- Configure for `freertos`.
  - Set *FreeRTOSConfig.h* options:

```
#define configFRTOS_MEMORY_SCHEME                 3
#define configUSE_TIME_SLICING                    1
#define configENABLE_BACKWARD_COMPATIBILITY       1
#define configTICK_RATE_HZ ((TickType_t)1000)
```

  - Replace *heap_4.c* by *heap_3.c*.
- Add `sbengine_main_task()` definition and task creation in *freertos_hello.c*.
  1. Place the definition below just after the define for the *hello_task_PRIORITY*.

```
void sbengine_main_task(void *argument);
```

  2. Replace the *Hello_task* code:

```
if (xTaskCreate(hello_task, "Hello_task", configMINIMAL_STACK_SIZE + 10,
NULL, hello_task_PRIORITY, NULL) != pdPASS)
```

  3. Replace with `sbengine_main_task` creation code.

```
if (xTaskCreate(sbengine_main_task, "sbengine", 4096, NULL,
configMAX_PRIORITIES / 2, NULL) != pdPASS)
```

- Add initialization functions call to *freertos_hello.c*.
  1. Ensure that the following headers are included in *freertos_hello.c*.

```
#include "peripheral.h"
```

  2. Add cock and peripheral initialization functions as below:

```
int main(void)
{
    /* Init board hardware. */
    /* attach 12 MHz clock to FLEXCOMM0 (debug console) */
    CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);

    /* Route Main clock to LCD. */
    CLOCK_AttachClk(kMAIN_CLK_to_LCD_CLK);
    CLOCK_SetClkDiv(kCLOCK_DivLcdClk, 1, true);
    /* attach 12 MHz clock to FLEXCOMM2 (I2C master for touch controller) */
    CLOCK_AttachClk(kFRO12M_to_FLEXCOMM2);
    CLOCK_EnableClock(kCLOCK_Gpio2);

    BOARD_InitPins();
    BOARD_BootClockPLL180M();
    BOARD_InitDebugConsole();
    BOARD_InitSDRAM();
    BOARD_InitPeripheral();
    if(xTaskCreate(sbengine_main_task, "sbengine", 4096, NULL, hello_task_PRIORITY, NULL) !=
pdPASS)
    {
         PRINTF("Task creation failed!.\r\n");
         while (1)
           ;
```

```
    }
    vTaskStartScheduler();
    for (;;)
      ;
}
```

- Update *sbengine_task.c*.
  1. Add including header files.

```
#include "gre.h"
#include "iodefs.h"
#include "generic_display.h"
#include "greal.h"
#include "peripheral.h"
#include "fsl_lcdc.h"
```

  2. Add global variables definition.

```
gr_generic_display_layer_info_t main_layer;
gr_application_t *app;
```

  3. Delete the `gr_application_t *app;` definition located in *run_storyboard_app()*.
  4. Implement the `int gr_generic_display_init()` function as below:

```
int
gr_generic_display_init(gr_generic_display_info_t *info) {
    info->num_layers = 1;
    main_layer.num_buffers = 2;
    info->layer_info = &main_layer;

    main_layer.buffer[0] = (void *)(VRAM_ADDR);
#if(LCD_BITS_PER_PIXEL == 16)
    main_layer.render_format = GR_RENDER_FMT_RGB565;
#elif(LCD_BITS_PER_PIXEL == 32)
    main_layer.render_format = GR_RENDER_FMT_ARGB8888;
#endif
    main_layer.width = 480;
    main_layer.height = 272;
    main_layer.stride = (uint16_t)(main_layer.width *
GR_RENDER_FMT_BYTESPP(main_layer.render_format));
    main_layer.buffer[1] = (void *)(VRAM_ADDR + VRAM_SIZE);
    return 0;
}
```

  5. Implement the `int gr_generic_display_update()` function as below:

```
int
gr_generic_display_update(const gr_generic_display_info_t *info) {
    s_frame_done = false;
    LCDC_SetPanelAddr(BOARD_LCD, kLCDC_UpperPanel, (uint32_t)info->layer_info[0].buffer[info-
>layer_info[0].buffer_draw_index]);
    while(s_frame_done == false);
    return 0;
}
```

- Update sbengine_plugins.h.

```
// gre plugins
extern int gre_plugin_animate(gr_plugin_state_t *);
//extern int gre_plugin_captureplayback(gr_plugin_state_t *);
extern int gre_plugin_logger(gr_plugin_state_t *);
extern int gre_plugin_timer(gr_plugin_state_t *);
extern int gre_plugin_greio(gr_plugin_state_t *);

// Render extension plugins
extern int gre_plugin_circle(gr_plugin_state_t *);
extern int gre_plugin_poly(gr_plugin_state_t *);
extern int gre_plugin_script_lua(gr_plugin_state_t *);
extern int gre_plugin_c_callback(gr_plugin_state_t *);
extern int gre_plugin_screen_path(gr_plugin_state_t *);


const gr_plugin_create_func_t sb_plugins[] = {
    gre_plugin_animate,
    gre_plugin_greio,
    gre_plugin_c_callback,
    //gre_plugin_circle,
    //gre_plugin_screen_path,
    gre_plugin_logger,
    //gre_plugin_poly,
    //gre_plugin_script_lua,
    gre_plugin_timer,
    NULL,
};
```

- Update *__heap_size__* defined in the linker file.
  1. Find the linker file as shown in Figure 10.

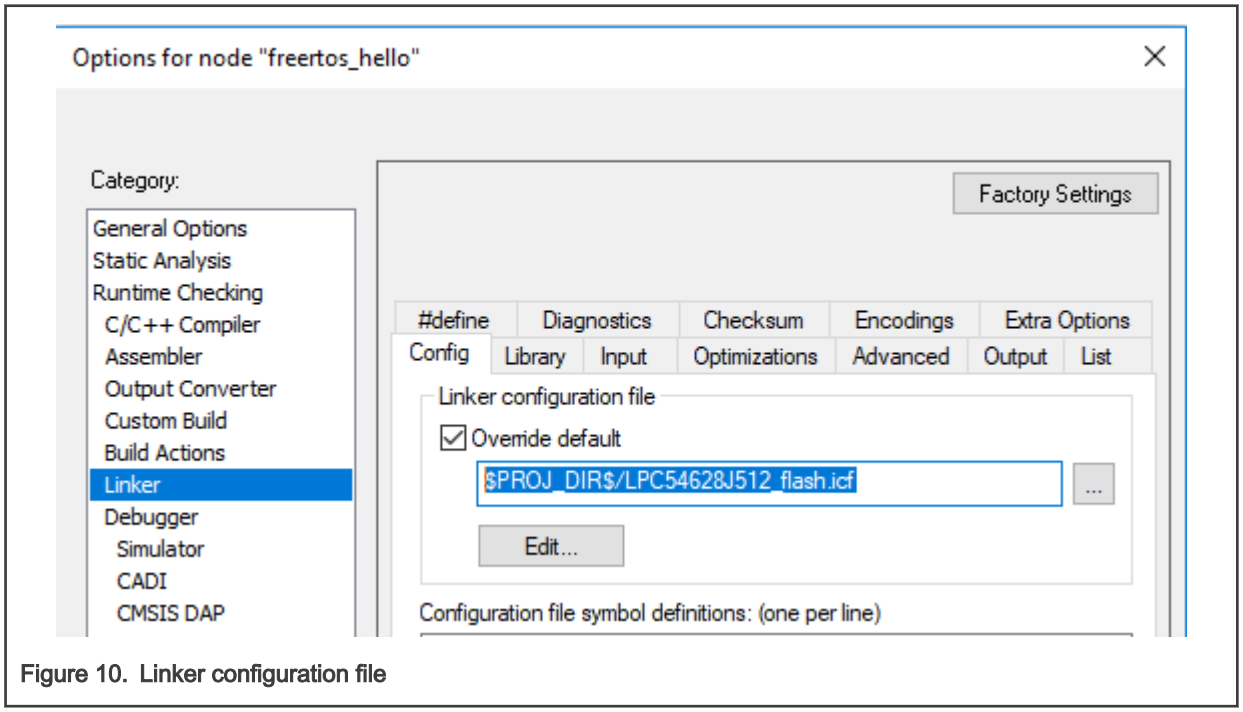

Figure 10. Linker configuration file

2. Click the ... button to open the folder and edit the linker configuration file, *LPC54628J512_flash.icf*.

3. Change the *__heap_size__* to **13000**.

```
if (isdefinedsymbol(__heap_size__)) {
      define symbol __size_heap__          = __heap_size__;
      } else {
      define symbol __size_heap__          = 1024*1024;
}
```

4. Place **HEAP** to SDRAM.

```
place at address mem: m_interrupts_start    { readonly section .intvec };
place in TEXT_region                        { block RO };
place in DATA_region                        { block RW };
place in DATA region                        { block ZI };
place in DATA_region3                       { block HEAP };
place in CSTACK_region                      { block CSTACK };
```

Figure 11. Place heap to SDRAM region

- Modify the linker file to allocate some **lib** file to SPIFI flash.

```
define symbol m_text_start2             = 0x10000000;
define symbol m_text_end2               = 0x11000000;
define region TEXT_region2 = mem:[from m_text_start2 to m_text_end2];
place in TEXT_region2                   { readonly object autofit.o, readonly object cff.o, readonly object ftbase.o, readonly objec
                                          readonly object ftglyph.o, readonly object ftgzip.o, readonly object ftraster.o,
                                          readonly object psaux.o, readonly object pshinter.o, readonly object truetype.o,
                                          readonly object type1.o, readonly object sfnt.o, readonly object type1cid.o,
                                          readonly object type42.o};
```

Figure 12. Code linker file configuration

- Compile the project and test.
  - If there is no error after the compile, users can download the image to LPC54608 EVK to test it. The LCD can display the meter and the button, as shown in Figure 13.

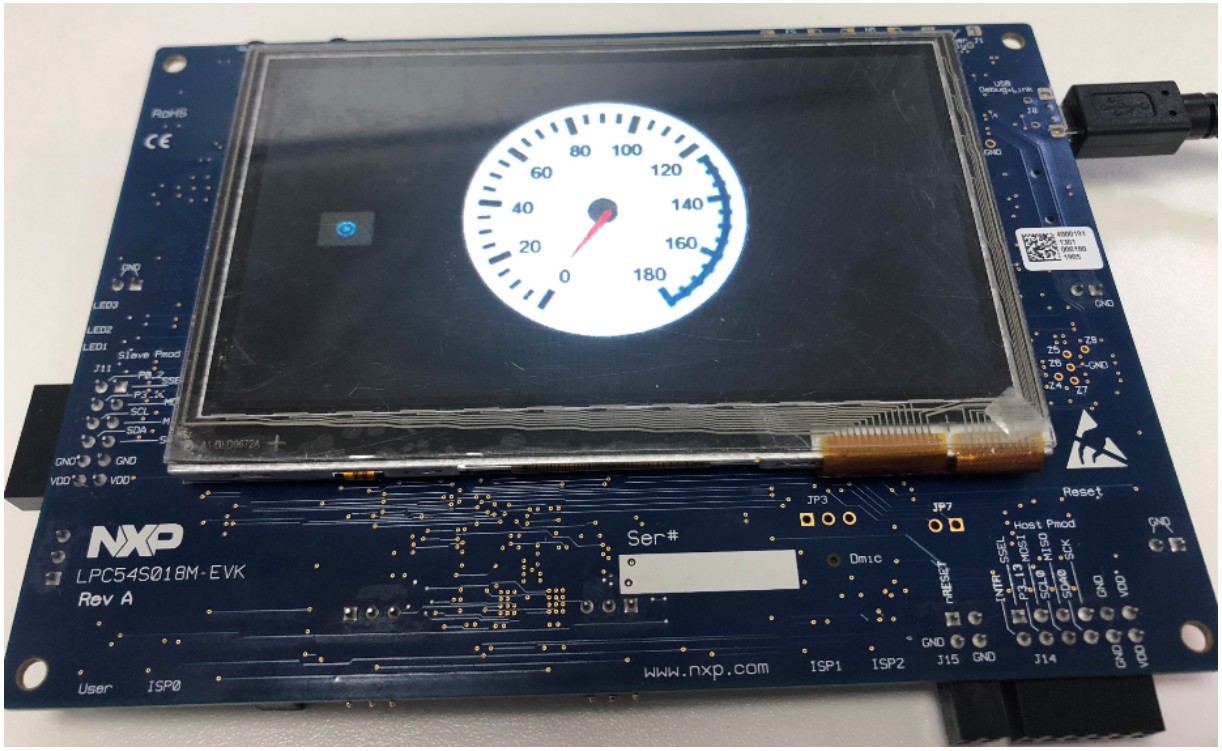

Figure 13. Meter and button display on LCD

— Without the support of the touch feature, the button on the LCD can not work.

3.4.3 Touch task implementation

The touch chip, **ft5406**, is initialized in `BOARD_InitPeripheral()` located in *peripheral.c*. Implement the touch task to support the touch feature.

Place the following function to poll for input from the touch screen and push the resulting event in to the Storyboard event queue.

```
void sbengine_input_task(void *arg)
{
     const int sleep_msec = 20;
     greal_timespec_t sleep_time = {
          .tv_sec = 0,
          .tv_nsec = sleep_msec * 1000000
     };

#if defined( APP_USE_STORYBOARD_IO )
    gr_key_event_t key_event = { 0, GR_KEY_ENTER, 0};

    /* initialise GPIO for USER LED and Button */
    BOARD_InitUserGPIO();
#endif

    touch_poll_state_t previous_touch_state = {0};
    touch_poll_state_t touch_state;
    bool pressed = false;

    while (1) {
        if (kStatus_Success != BOARD_Touch_Poll(&touch_state)) {
        greal_nanosleep(&sleep_time, NULL);
```

```
        continue;
        }

        // De-bounce inputs
        if (previous_touch_state.x == touch_state.x &&
        previous_touch_state.y == touch_state.y &&
        previous_touch_state.pressed == touch_state.pressed) {
               greal_nanosleep(&sleep_time, NULL);
               continue;
                 }

        if (touch_state.pressed) {
                 gr_ptr_event_t event = {
                     // No, this isn't a typo. The axes are literally inverted at the driver level.
                 .x = touch_state.y,
             .y = touch_state.x,
             .z = 1,
             .timestamp = gr_snapshot_app_time(app),
                  };

        if (pressed) {
              gr_application_send_event(app, NULL, GR_EVENT_MOTION, GR_EVENT_PTR_FMT, &event,
sizeof(event));
        } else {
              pressed = true;
              gr_application_send_event(app, NULL, GR_EVENT_PRESS, GR_EVENT_PTR_FMT, &event,
sizeof(event));
        }

        previous_touch_state = touch_state;
      } else if (pressed) {
           gr_ptr_event_t event = {
          // No, this isn't a typo. The axes are literally inverted at the driver level.
          .x = previous_touch_state.y,
          .y = previous_touch_state.x,
          .z = 1,
          .timestamp = gr_snapshot_app_time(app),
        };

        pressed = false;
        gr_application_send_event(app, NULL, GR_EVENT_RELEASE, GR_EVENT_PTR_FMT, &event,
sizeof(event));
        previous_touch_state = touch_state;
      }

      greal_nanosleep(&sleep_time, NULL);
    }
}
```

Add the touch task creation in freertos_hello.c.

```
if(xTaskCreate(sbengine_input_task, "touch task", 2048, NULL, hello_task_PRIORITY-1, NULL) != pdPASS)
{
    PRINTF("Task creation failed!.\r\n");
    while (1)
           ;
}
```

3.4.4 Compile and download

Compile and download the project to LPC54608 EVB and run. The LCD displays the meter screen. Press the **Play** button and the pointer will rotate.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: January 15, 2021

Document identifier: AN13110

arm