Name-Surname of the Student : Ilker KURTULAN

Student Number :19070005043

Lecture : EEE5528-1

Lecturer : Prof Dr. Mustafa GÜNDÜZALP

GUI DESIGN :

As it seen on the figure 1 , the simple design is built using GUIX Studio's properties. After right clicking the "splash screen" it can be seen that , it offers us different options like including text , buttons , sliders etc. On my design I used text button property and combined these buttons with the " event_base_handler.c " to give each button a functionality.

Figure 1.

The inserted buttons have to be edited. On the red indicator we can change it's API name in order to use it on " event_base_handler.c ". If we continue sliding down , we can see the name changing property for our button.

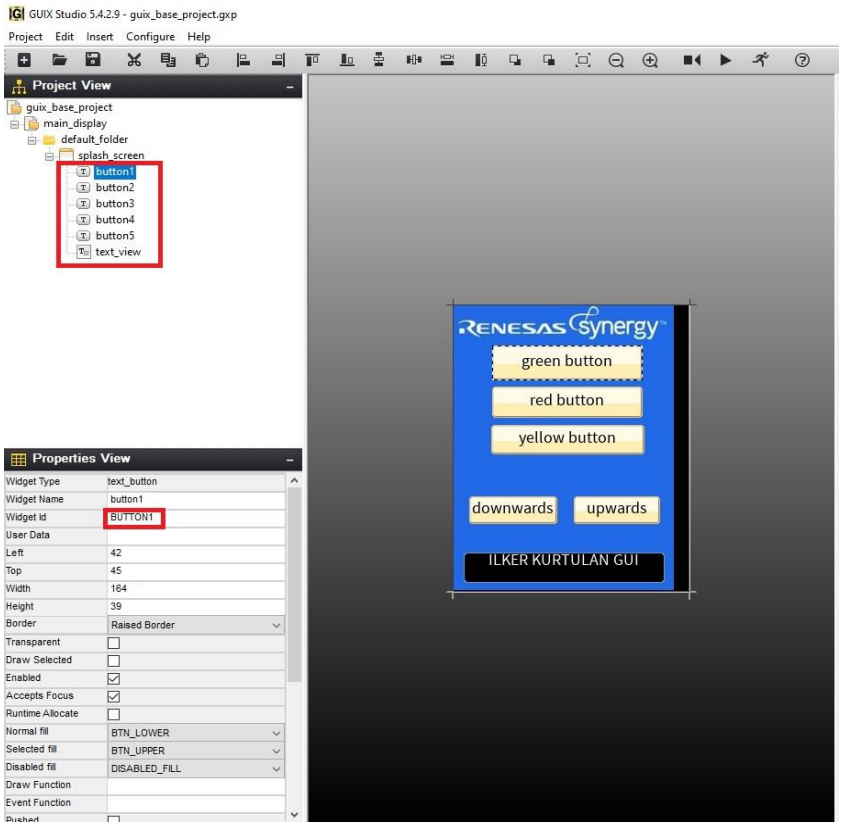

Figure 2.

MAIN CODE / FUNCTIONALITY OF BUTTONS :

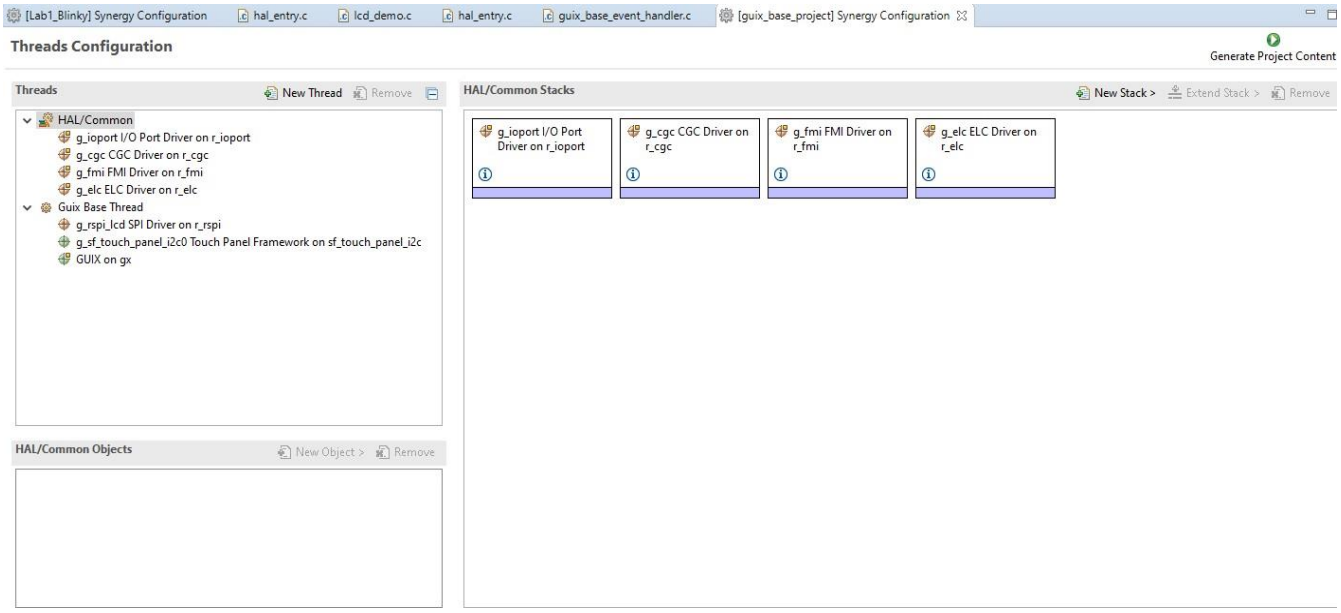

Figure 3.

In order to give functionality to buttons , editing has to be done on " event_base_handler.c " file indicated with red square. The blue square c / header files are coming from the configuration threads segment. When we include lcd and gui threads , it automatically includes these c / header files in our " src " folder. We can also see our gui design as "guix_base_project.gxp " file down below highlighted with red.

Inside the " event_base_handler.c " , for each gui property that user add, renesas bsp creates different cases for each property added. On the first three cases I programmed the led to toggle according to user's press on the lcd screen. In these cases I used bsp_leds api in order to reach every led state individually.

For button 4 & 5, I defined local variables like ; leds , pin_state , freq_in_hz , bsp_delay_units and delay. I wrote simple sequence that follows the led patterns upwards and downwards by opening and closing the leds with small delay.

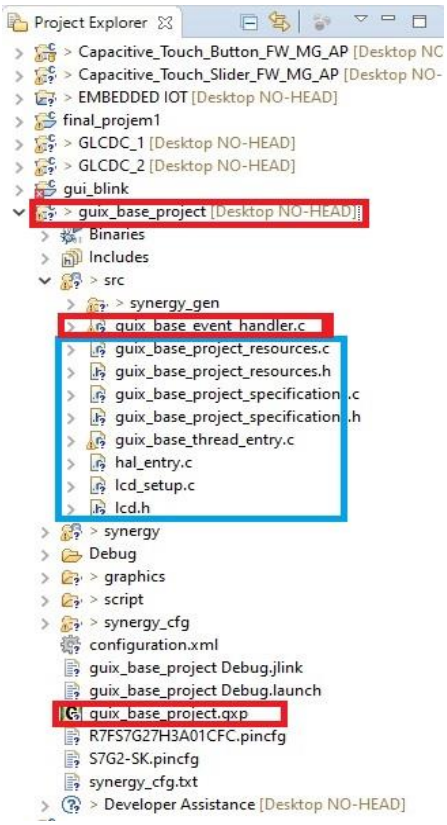

Figure 4.

APPENDIX

```
#include "guix_base_project_resources.h"
#include "guix_base_project_specifications.h"
#include "hal_data.h"


#define TIME_EVENT_TIMER     (100)
#define SPLASH_EVENT_TIMER 1
#define BUTTON_EVENT 1

/* Splash Screen Event Handler */
UINT SplashScreenEventHandler (GX_WINDOW * widget, GX_EVENT * event_ptr)
{
#if (BUTTON_EVENT)
    bsp_leds_t  pbsp_leds;
    ioport_port_pin_t led1_pin;
    ioport_port_pin_t led2_pin;
    ioport_port_pin_t led3_pin;

    R_BSP_LedsGet(&pbsp_leds);
    led1_pin = pbsp_leds.p_leds[BSP_LED_LED1];
    led2_pin = pbsp_leds.p_leds[BSP_LED_LED2];
    led3_pin = pbsp_leds.p_leds[BSP_LED_LED3];

#endif
#if (SPLASH_EVENT_TIMER)
    UINT status;
    GX_MULTI_LINE_TEXT_VIEW * my_text_view = &splash_screen.splash_screen_text_view;
#endif
    switch (event_ptr->gx_event_type)
    {
        case GX_EVENT_SHOW:
            gx_system_timer_start(widget, TIME_EVENT_TIMER, 20 * 5, 0);
            return gx_window_event_process(widget, event_ptr);
        break;
#if (SPLASH_EVENT_TIMER)
        case GX_EVENT_TIMER:
            status = gx_multi_line_text_view_text_set(my_text_view, "ILKER KURTULAN
GUI !!!");
            if (GX_SUCCESS != status)
            {
                while(1);
            }
        break;
#endif

#if (BUTTON_EVENT)
        case GX_SIGNAL(BUTTON1, GX_EVENT_CLICKED):
        {
            ioport_level_t pin_state;
```

```
    g_ioport.p_api->pinRead(led1_pin, &pin_state);
    if (IOPORT_LEVEL_LOW == pin_state)

        g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
    else
        g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
}
break;

case GX_SIGNAL(BUTTON2, GX_EVENT_CLICKED):
    {

    ioport_level_t pin_state;

    g_ioport.p_api->pinRead(led2_pin, &pin_state);
    if (IOPORT_LEVEL_LOW == pin_state)

        g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
    else
        g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);

    }
break;

case GX_SIGNAL(BUTTON3, GX_EVENT_CLICKED):
    {

    ioport_level_t pin_state;

    g_ioport.p_api->pinRead(led3_pin, &pin_state);
    if (IOPORT_LEVEL_LOW == pin_state)

        g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
    else
        g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);

    }
break;

case GX_SIGNAL(BUTTON4, GX_EVENT_CLICKED):
    {

    bsp_leds_t leds;
    /* LED state variable */


    ioport_level_t pin_state;

    g_ioport.p_api->pinRead(led1_pin, &pin_state);
    g_ioport.p_api->pinRead(led2_pin, &pin_state);
    g_ioport.p_api->pinRead(led3_pin, &pin_state);

    int freq_in_hz = 3;
```

```
bsp_delay_units_t bsp_delay_units = BSP_DELAY_UNITS_MILLISECONDS;

int delay = bsp_delay_units/freq_in_hz;



g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);
R_BSP_SoftwareDelay(delay, bsp_delay_units);
g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);
R_BSP_SoftwareDelay(delay, bsp_delay_units);
g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);
R_BSP_SoftwareDelay(delay, bsp_delay_units);
g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
R_BSP_SoftwareDelay(delay, bsp_delay_units);
g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
R_BSP_SoftwareDelay(delay, bsp_delay_units);
g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
R_BSP_SoftwareDelay(delay, bsp_delay_units);
g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);
R_BSP_SoftwareDelay(delay, bsp_delay_units);
g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);
R_BSP_SoftwareDelay(delay, bsp_delay_units);
g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);
R_BSP_SoftwareDelay(delay, bsp_delay_units);
g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
R_BSP_SoftwareDelay(delay, bsp_delay_units);
g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
R_BSP_SoftwareDelay(delay, bsp_delay_units);
g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
```

```
        g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
        R_BSP_SoftwareDelay(delay, bsp_delay_units);
        g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
        g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
        g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);


        }
    break;

    case GX_SIGNAL(BUTTON5, GX_EVENT_CLICKED):
        {

        bsp_leds_t leds;
        /* LED state variable */

        ioport_level_t pin_state;

        g_ioport.p_api->pinRead(led1_pin, &pin_state);
        g_ioport.p_api->pinRead(led2_pin, &pin_state);
        g_ioport.p_api->pinRead(led3_pin, &pin_state);

        int freq_in_hz = 3;

        bsp_delay_units_t bsp_delay_units = BSP_DELAY_UNITS_MILLISECONDS;

        int delay = bsp_delay_units/freq_in_hz;



        g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
        g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
        g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);
        R_BSP_SoftwareDelay(delay, bsp_delay_units);
        g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
        g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
        g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
        R_BSP_SoftwareDelay(delay, bsp_delay_units);
        g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
        g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
        g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
        R_BSP_SoftwareDelay(delay, bsp_delay_units);
        g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
        g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
        g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
        R_BSP_SoftwareDelay(delay, bsp_delay_units);
        g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
        g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
        g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);
        R_BSP_SoftwareDelay(delay, bsp_delay_units);
        g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
        g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
        g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);
        R_BSP_SoftwareDelay(delay, bsp_delay_units);
```

```
            g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
            g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
            g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);
            R_BSP_SoftwareDelay(delay, bsp_delay_units);
            g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
            g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
            g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
            R_BSP_SoftwareDelay(delay, bsp_delay_units);
            g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
            g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
            g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
            R_BSP_SoftwareDelay(delay, bsp_delay_units);
            g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
            g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
            g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_HIGH);
            R_BSP_SoftwareDelay(delay, bsp_delay_units);
            g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
            g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_HIGH);
            g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);
            R_BSP_SoftwareDelay(delay, bsp_delay_units);
            g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_HIGH);
            g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
            g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);
            R_BSP_SoftwareDelay(delay, bsp_delay_units);
            g_ioport.p_api->pinWrite(led1_pin, IOPORT_LEVEL_LOW);
            g_ioport.p_api->pinWrite(led2_pin, IOPORT_LEVEL_LOW);
            g_ioport.p_api->pinWrite(led3_pin, IOPORT_LEVEL_LOW);



        }
        break;
```

```
#endif
        default:
        return gx_window_event_process(widget, event_ptr);
    }
    return 0;
}
```

REFERANCES :

[1] https://en-support.renesas.com/knowledgeBase

[2] https://www.renesas.com/us/en/doc/products/renesas-synergy/apn/r12an0021eu0119-synergy-sk-s7g2-pk-s5d9-guix-hello-world.pdf

[3] https://www.renesas.com/in/en/doc/products/renesas-synergy/doc/r12um0004eu0100_synergy_sk_s7g2.pdf