ใบงานที่ 5

GPIO with Interrupt/Polling

อุปกรณ์ที่ต้องใช้ในการทดลอง

1. USB cable for program
2. ESP32 Board
3. Computer

ใบงานที่เกี่ยวข้อง: ใบงานที่ 1 เพื่อใช้ในการกำหนดขาที่ต้องการ

<u>วงจร</u>

ให้นิสิตต่อวงจรโดยเลือกขาที่ต้องใช้งานเป็น Input จำนวน 3 ขา (GPIO16, GPIO17, GPIO5) โดยปุ่มกดทั้ง 3 จะเป็นการต่อเมื่อมีการกดปุ่มจะเป็นการเชื่อมขาของบอร์ดที่เป็น Input เข้ากับ Ground

ในส่วนของ Output นิสิตจะต้องเลือก GPIO ที่ต้องการต่อจำนวน 2 ขา (GPIO18, GPIO19) โดยต่อ LED ในลักษณะที่เป็น Sink ทั้ง 2 ดวง ค่าความต้านทานที่ใช้สามารถเลือกใช้ค่าใดก็ได้ตามนี้ 330 Ohms, 470 Ohms, 1k Ohms

<u>Digital Output</u>

1. ทำการ include header file ที่เกี่ยวข้อง

```c
#include <stdio.h>
#include "driver/gpio.h"

//include FreeRTOS header for vTaskDelay and portTICK_PERIOD_MS
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

#define GPIO_OUTPUT_IO_0  CONFIG_GPIO_OUTPUT_0
#define GPIO_OUTPUT_IO_1  CONFIG_GPIO_OUTPUT_1
#define GPIO_OUTPUT_PIN_SET ((1ULL<<GPIO_OUTPUT_IO_0) | (1ULL<<GPIO_OUTPUT_IO_1))

/*
 * 1ULL<<GPIO_OUTPUT_IO_0 is set bit GPIO Pin 18
 * 1ULL<<GPIO_OUTPUT_IO_1 is set bit GPIO Pin 19
 *
 */
```

2. ทำการกำหนดค่าเพื่อจะทำการ configure ให้กับ GPIO ให้เหมาะสม โดยคำอธิบายแสดงใน comment

```
29  void app_main(void)
28  {
27      gpio_config_t io_conf = {};
26
25      //disable interrupt
24      io_conf.intr_type = GPIO_INTR_DISABLE;
23
22      //set as a output mode
21      io_conf.mode = GPIO_MODE_OUTPUT;
20
19      //bit mask of the pins that you want to set, e.g. GPIO18/19
18      io_conf.pin_bit_mask = GPIO_OUTPUT_PIN_SET;
17
16      //disable pull-down mode
15      io_conf.pull_down_en = 0;
14
13      //disable pull-up mode
12      io_conf.pull_up_en = 0;
11
10      //configuration to GPIO
9       gpio_config(&io_conf);
8
7       int cnt = 0;
6       for(;;){
5           printf("cnt: %d\n", cnt++);
4           vTaskDelay(1000 / portTICK_PERIOD_MS);
3           gpio_set_level(GPIO_OUTPUT_IO_0, cnt & 0x01);
2           gpio_set_level(GPIO_OUTPUT_IO_1, cnt & 0x02);
1       }
47  }
```

1) บรรทัดหมายเลข 27 เป็นการสร้างตัวแปรเพื่อจะใช้เป็นที่เก็บค่า configure ต่างๆ
2) บรรทัดหมายเลข 24 เป็นการกำหนดให้ GPIO ไม่ใช้การ interrupt
3) บรรทัดหมายเลข 21 เป็นการกำหนดให้ GPIO ในขาที่ต้องการเป็นโหมด Output
4) บรรทัดหมายเลข 18 เป็นการกำหนดขาที่ต้องการใช้ในที่นี้คือ GPIO18/GPIO19
5) บรรทัดหมายเลข 15 เป็นการกำหนดให้ GPIO ไม่ต้องใช้ pull-down
6) บรรทัดหมายเลข 12 เป็นการกำหนดให้ GPIO ไม่ต้องใช้ pull-up
7) บรรทัดหมายเลข 9 เป็นการเรียกใช้ function “gpio_config” เพื่อกำหนดให้กับรีจิสเตอร์ที่เกี่ยวข้อง

Function ที่ใช้ในการกำหนดค่าให้ออกเป็น Logic High/Low จะใช้ชื่อว่า “gpio_set_level” โดยพารามิเตอร์แรกจะเป็นการกำหนดขาของ GPIO ส่วนพารามิเตอร์ตัวถัดมาจะเป็นระดับของโลจิก (1 – High, 0 – Low)

3. ทำการสร้างไฟล์ Kconfig.projbuild ใน folder main ของโปรเจค เพื่อใช้เป็นตัวเอาไว้ใช้ปรับแต่งหมายเลขของ GPIO Pin ตามความต้องการ โดยในไฟล์ที่สร้างขึ้นมานี้ จะกำหนด GPIO18 และ GPIO19 เป็นค่า default หากใช้คำสั่ง idf.py menuconfig จะมี “Example Configuration” ขึ้นมาเมื่อเข้าไปในเมนูดังกล่าวจะมีหัวข้อให้กำหนดเลข GPIO ได้

```
Lab02.c ×   Kconfig.projbuild ×
menu "Example Configuration"

config GPIO_OUTPUT_0
    int "GPIO output pin #0"
    default 18
    help
        Set the GPIO pin for Pin #0

config GPIO_OUTPUT_1
    int "GPIO output pin #1"
    default 19
    help
        Set the GPIO pin for Pin #1

endmenu
```

```
(Top) → Example Configuration

(18) GPIO output pin #0
(19) GPIO output pin #1
```

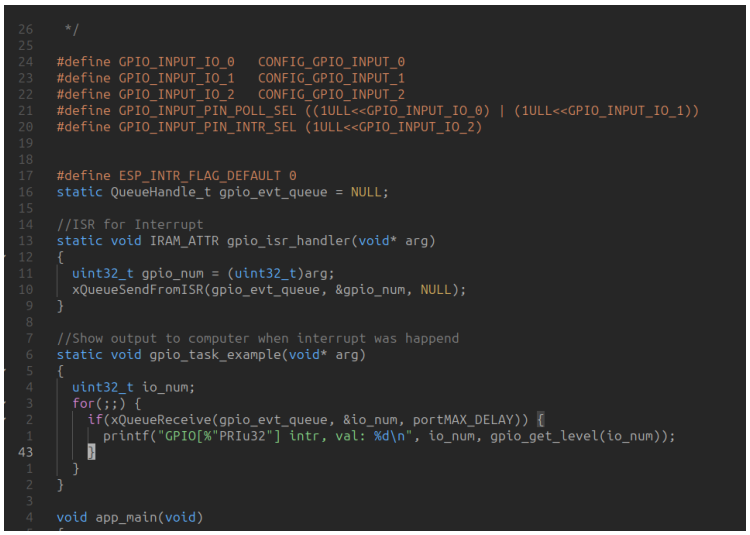
4. ทำการ build และ burn ลงบอร์ด จากนั้นดูหลอดไฟ LED ที่ต่อกับขา GPIO18/19

Digital Input ในโหมดการทำงานแบบ Polling & Interrupt

1. ทำการเพิ่มการ include เนื่องจากจะมีการใช้งาน queue เพื่อทำหน้าที่รับ-ส่ง ข้อมูลระหว่าง task

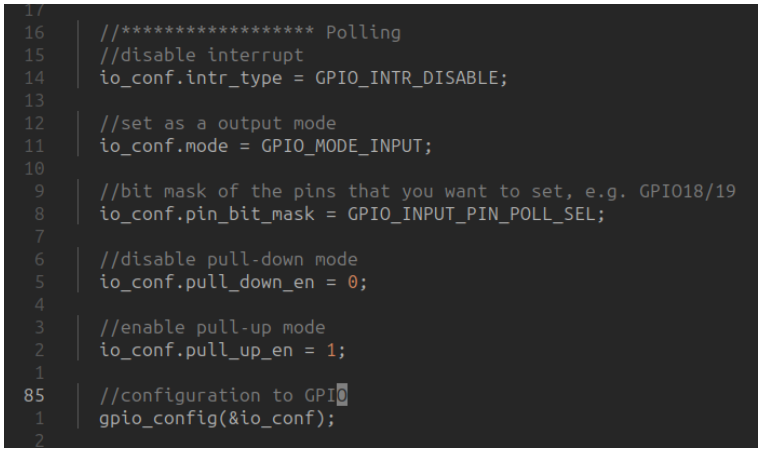
```
#include <stdio.h>
#include <inttypes.h>
#include "driver/gpio.h"

//include FreeRTOS header for vTaskDelay and portTICK_PERIOD_MS
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
```

2. เพิ่มโค้ดในส่วนของ input เพื่อรับค่าจากปุ่มกด ทั้งแบบ Polling และ Interrupt รวมทั้ง task ที่ใช้จัดการเอาค่าที่ได้จาก interrupt ไปแสดงที่คอมพิวเตอร์ “gpio_task_example”



```
 */

#define GPIO_INPUT_IO_0   CONFIG_GPIO_INPUT_0
#define GPIO_INPUT_IO_1   CONFIG_GPIO_INPUT_1
#define GPIO_INPUT_IO_2   CONFIG_GPIO_INPUT_2
#define GPIO_INPUT_PIN_POLL_SEL ((1ULL<<GPIO_INPUT_IO_0) | (1ULL<<GPIO_INPUT_IO_1))
#define GPIO_INPUT_PIN_INTR_SEL (1ULL<<GPIO_INPUT_IO_2)


#define ESP_INTR_FLAG_DEFAULT 0
static QueueHandle_t gpio_evt_queue = NULL;

//ISR for Interrupt
static void IRAM_ATTR gpio_isr_handler(void* arg)
{
  uint32_t gpio_num = (uint32_t)arg;
  xQueueSendFromISR(gpio_evt_queue, &gpio_num, NULL);
}

//Show output to computer when interrupt was happend
static void gpio_task_example(void* arg)
{
  uint32_t io_num;
  for(;;) {
    if(xQueueReceive(gpio_evt_queue, &io_num, portMAX_DELAY)) {
      printf("GPIO[%"PRIu32"] intr, val: %d\n", io_num, gpio_get_level(io_num));
    }
  }
}

void app_main(void)
```

3. เพิ่มโค้ดในส่วนของการ initial ตัว GPIO โมดูล โดยกำหนดให้ขาที่ต้องการ (ในที่นี้จะเป็นขา 16 และ 17) เป็นขาในโหมด Input และไม่มีการ interrupt ใดๆ ทั้งสิ้น



```
  //****************** Polling
  //disable interrupt
  io_conf.intr_type = GPIO_INTR_DISABLE;

  //set as a output mode
  io_conf.mode = GPIO_MODE_INPUT;

  //bit mask of the pins that you want to set, e.g. GPIO18/19
  io_conf.pin_bit_mask = GPIO_INPUT_PIN_POLL_SEL;

  //disable pull-down mode
  io_conf.pull_down_en = 0;

  //enable pull-up mode
  io_conf.pull_up_en = 1;

  //configuration to GPIO
  gpio_config(&io_conf);
```

4. โค้ดในส่วนของการกำหนดให้ขาของ GPIO ที่ต้องการสามารถกระทำการ interrupt ได้ (GPIO5) รวมทั้งการสร้าง queue เพื่อรับค่าจากเหตุการณ์ตอนเกิด interrupt และสร้าง task ที่ใช้ในการส่งค่าไปแสดงที่คอมพิวเตอร์

```
    //***************** Interrupt
    //enable interrupt
    io_conf.intr_type = GPIO_INTR_NEGEDGE; //NEGEDGE, POSEDGE, ANYEDGE

    //set as a output mode
    io_conf.mode = GPIO_MODE_INPUT;

    //bit mask of the pins that you want to set, e.g. GPIO
    io_conf.pin_bit_mask = GPIO_INPUT_PIN_INTR_SEL;

    //disable pull-down mode
    io_conf.pull_down_en = 0;

    //enable pull-up mode
    io_conf.pull_up_en = 1;

    //configuration to GPIO
    gpio_config(&io_conf);

    //set ISR for interrupt service
    gpio_install_isr_service(ESP_INTR_FLAG_DEFAULT);

    //Hook isr handler for specific GPIO Pin
    gpio_isr_handler_add(GPIO_INPUT_IO_2, gpio_isr_handler, (void*)GPIO_INPUT_IO_2);

    //*************** Create a Task
    //create a queue to handle gpio event from isr
    gpio_evt_queue = xQueueCreate(10, sizeof(uint32_t));
    //create a display task for ISR
    xTaskCreate(gpio_task_example, "gpio_task_example", 2048, NULL, 10, NULL);
```

5. ใน Idle task (loop-forever) เพิ่มเติมโค้ดในส่วนของการอ่านค่าจากการกดปุ่มที่ไม่มีการ hook ISR (เป็นการวนอ่านค่าไปทีละปุ่ม; Polling)

```
    int cnt = 0;
    for(;;){
      printf("cnt: %d\n", cnt++);
      vTaskDelay(1000 / portTICK_PERIOD_MS);
      gpio_set_level(GPIO_OUTPUT_IO_0, cnt & 0x01);
      gpio_set_level(GPIO_OUTPUT_IO_1, cnt & 0x02);
      printf("GPIO[%d] poll, val: %d\n", GPIO_INPUT_IO_0, gpio_get_level(GPIO_INPUT_IO_0));
      printf("GPIO[%d] poll, val: %d\n", GPIO_INPUT_IO_1, gpio_get_level(GPIO_INPUT_IO_1));
    }
}
c
```

ใบงานท้ายการทดลอง

ให้นิสิตเขียนโปรแกรมตามข้อกำหนดด้านล่าง

1. มี LED จำนวนทั้งสิ้น 2 ดวง โดยจะต้องมี LED 1 ดวงที่จะกระพริบในทุกๆ 1 วินาที (ติด 1 วินาที ดับ 1 วินาที)
2. เมื่อมีการกดปุ่ม LED อีกดวงที่ต่างจากข้อ 1 จะต้องทำการ Toggle สถานะ (จากติดเป็นดับ หรือจากดับเป็นติด) หากไม่มีการกดปุ่ม LED ดวงนี้จะต้องคงสถานะเดิมไว้แบบนั้น