ใบงานที่ 6

Serial Communication (UART/SPI/I2C)

อุปกรณ์ที่ต้องใช้ในการทดลอง

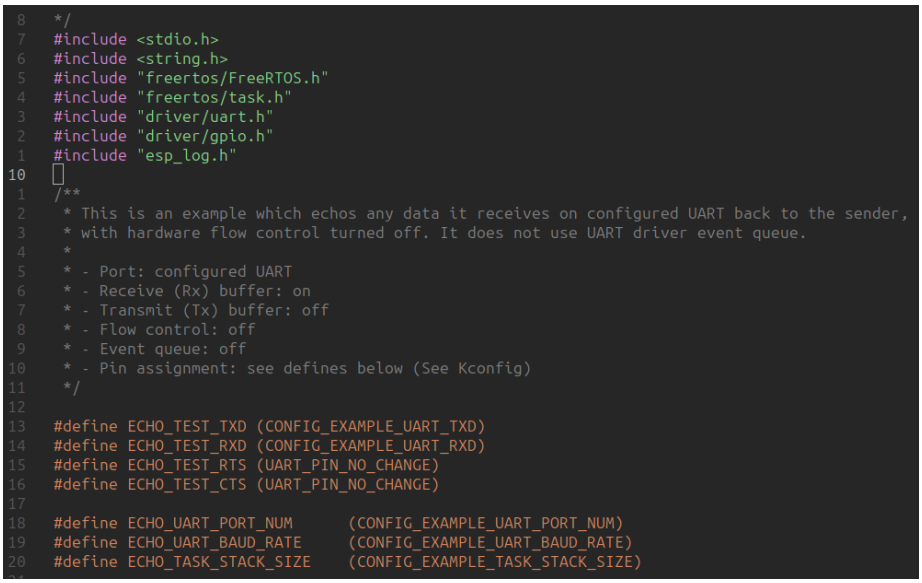
1. สาย USB
2. ESP32 Board
3. Computer
4. สายไฟเพื่อทำ Loopback

<u>วงจร</u>

ให้นิสิตทำการเชื่อมสาย Tx (GPIO17) และ Rx (GPIO16) ของ UART#2 เข้าหากันเพื่อทำ Loopback สัญญาณข้อมูลเข้าหาตัวเอง

UART communication (Echo or Loopback Application)

1. โค้ดในส่วนของ header file กับ parameter ต่างๆ ที่เกี่ยวข้อง



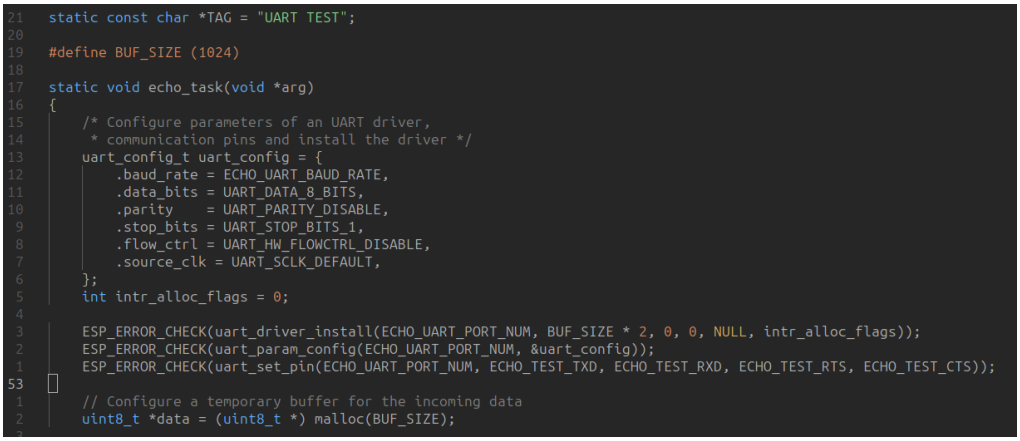
```
 */
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/uart.h"
#include "driver/gpio.h"
#include "esp_log.h"

/**
 * This is an example which echos any data it receives on configured UART back to the sender,
 * with hardware flow control turned off. It does not use UART driver event queue.
 *
 * - Port: configured UART
 * - Receive (Rx) buffer: on
 * - Transmit (Tx) buffer: off
 * - Flow control: off
 * - Event queue: off
 * - Pin assignment: see defines below (See Kconfig)
 */

#define ECHO_TEST_TXD (CONFIG_EXAMPLE_UART_TXD)
#define ECHO_TEST_RXD (CONFIG_EXAMPLE_UART_RXD)
#define ECHO_TEST_RTS (UART_PIN_NO_CHANGE)
#define ECHO_TEST_CTS (UART_PIN_NO_CHANGE)

#define ECHO_UART_PORT_NUM      (CONFIG_EXAMPLE_UART_PORT_NUM)
#define ECHO_UART_BAUD_RATE     (CONFIG_EXAMPLE_UART_BAUD_RATE)
#define ECHO_TASK_STACK_SIZE    (CONFIG_EXAMPLE_TASK_STACK_SIZE)
```

2. โค้ดในส่วนต้นของ echo task



```
21  static const char *TAG = "UART TEST";
20
19  #define BUF_SIZE (1024)
18
17  static void echo_task(void *arg)
16  {
15      /* Configure parameters of an UART driver,
14       * communication pins and install the driver */
13      uart_config_t uart_config = {
12          .baud_rate = ECHO_UART_BAUD_RATE,
11          .data_bits = UART_DATA_8_BITS,
10          .parity    = UART_PARITY_DISABLE,
9           .stop_bits = UART_STOP_BITS_1,
8           .flow_ctrl = UART_HW_FLOWCTRL_DISABLE,
7           .source_clk = UART_SCLK_DEFAULT,
6       };
5       int intr_alloc_flags = 0;
4
3       ESP_ERROR_CHECK(uart_driver_install(ECHO_UART_PORT_NUM, BUF_SIZE * 2, 0, 0, NULL, intr_alloc_flags));
2       ESP_ERROR_CHECK(uart_param_config(ECHO_UART_PORT_NUM, &uart_config));
1       ESP_ERROR_CHECK(uart_set_pin(ECHO_UART_PORT_NUM, ECHO_TEST_TXD, ECHO_TEST_RXD, ECHO_TEST_RTS, ECHO_TEST_CTS));
53
1       // Configure a temporary buffer for the incoming data
2       uint8_t *data = (uint8_t *) malloc(BUF_SIZE);
```

บรรทัดที่ 13 – 6 เป็นการกำหนดค่าพารามิเตอร์ที่เกี่ยวข้อง

a. กำหนด baud rate โดยค่าเริ่มต้นจะเป็น 115200
b. กำหนดขนาดของข้อมูลที่จะส่ง ในที่นี้จะเป็น 8 บิต
c. กำหนดพาริตี้เป็น ไม่ใช่พาริตี้
d. กำหนดจำนวนบิตหยุด (Stop Bit) เป็น 1 บิต
e. กำหนดการใช้ flow control เป็น ไม่ใช่
f. กำหนดสัญญาณนาฬิกาเป็นใช้สัญญาณนาฬิกาปกติ

บรรทัดที่ 3 เป็นการเรียกฟังก์ชันเพื่อทำการติดตั้งไดร์เวอร์

บรรทัดที่ 2 เป็นการ configure ค่าพารามิเตอร์ที่กำหนดให้กับโมดูล

บรรทัดที่ 1 เป็นการกำหนดขา GPIO ที่ต้องการให้เป็นขาส่งข้อมูล (Tx) และขาสำหรับรับข้อมูล (Rx)

จากนั้นจึงทำการจองหน่วยความจำเพื่อใช้เก็บข้อมูลเมื่อ UART ทำการรับข้อมูลเข้ามาเรียบร้อย

3. โค้ดส่วนท้ายเพื่อทำการสร้าง task และ ลูปไม่รู้จบเพื่อวนรับข้อมูลจากผู้ใช้งานและส่งข้อมูลออกไปหาผู้ใช้งาน

```c
    ESP_ERROR_CHECK(uart_param_config(ECHO_UART_PORT_NUM, &uart_config));
    ESP_ERROR_CHECK(uart_set_pin(ECHO_UART_PORT_NUM, ECHO_TEST_TXD, ECHO_TEST_RXD, ECHO_TEST_RTS, ECHO_TEST_CTS));

    // Configure a temporary buffer for the incoming data
    uint8_t *data = (uint8_t *) malloc(BUF_SIZE);

    while (1) {

        //Receive data from user (UART0; USB-to-Serial)
        printf("Enter data for test loopback: ");
        scanf("%s",data);

        int len = strlen((const char *)data);

        // Write data to UART ECHO_UART_PORT_NUM (No. 2)
        uart_write_bytes(ECHO_UART_PORT_NUM, (const char *) data, len);
        if (len) {
            data[len] = '\0';
            ESP_LOGI(TAG, "Recv str: %s", (char *) data);
        }

        // Read data from the UART
        uart_read_bytes(ECHO_UART_PORT_NUM, data, (BUF_SIZE - 1), 20 / portTICK_PERIOD_MS);
    }
}

void app_main(void)
{
    xTaskCreate(echo_task, "uart_echo_task", ECHO_TASK_STACK_SIZE, NULL, 10, NULL);
}
```

4. โหลดโปรแกรมลงบอร์ดพร้อมทั้งใช้ โปรแกรมในการรับค่าจาก Serial port ทดลองส่งตัวอักษร
5. ใช้ oscilloscope ในการจับสัญญาณจากขา Tx ของ UART#2 จากนั้น save ภาพหน้าจอ oscilloscope ลงเป็นรูปภาพและนำมาแปะในพื้นที่ว่างด้านล่าง พร้อมทั้งหาความกว้างของข้อมูลในแต่ละบิตของข้อมูลที่ส่งออกมาด้วย

ใบงานท้ายบท

ให้นิสิตทำการรับข้อมูลจาก GPS โมดูล จากนั้นนำค่าที่ได้แสดงผลในหน้า monitor ให้แก่ผู้ใช้งาน ทำการ capture หน้าจอบนคอมพิวเตอร์และใส่รูปที่ capture ได้แนบเป็นผลการทดลอง