Doan Viet Tu – 1952521

EMBEDDED SYSTEMS – CC01
LAB 2
ESP

ESP-L04

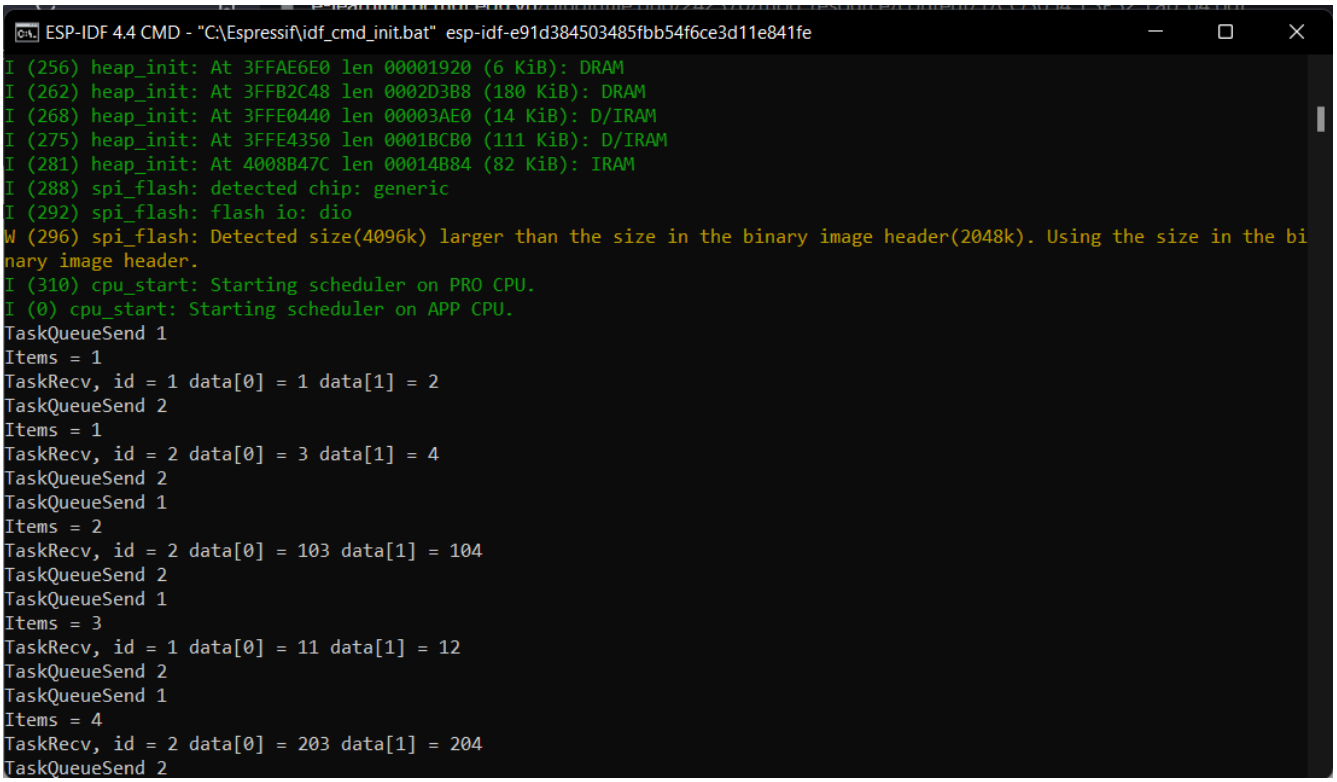
Source code:

```
//
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "esp_system.h"

TaskHandle_t taskHandle[3];
QueueHandle_t xQueue;

typedef struct
{
    int id;
    int data[10];
} QMsg_t;

bool DataSent = false;
bool DataReceived = false;

void TaskQueueSend1(void *args)
{
    QueueHandle_t queue = (QueueHandle_t)args;
    QMsg_t msg1;
    msg1.id = 1;
    msg1.data[0] = 1;
    msg1.data[1] = 2;
    while (true)
    {
        xQueueSend(queue, (void *)&msg1, 0);
        printf("TaskQueueSend 1\r\n");
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        msg1.data[0] += 10;
        msg1.data[1] += 10;
    }
    DataSent = true;
    vTaskDelete(taskHandle[1]);
}

void TaskQueueSend2(void *args)
{
    QueueHandle_t queue = (QueueHandle_t)args;
    QMsg_t msg2;
    msg2.id = 2;
    msg2.data[0] = 3;
    msg2.data[1] = 4;
    while (true)
    {
        xQueueSend(queue, (void *)&msg2, 0);
        printf("TaskQueueSend 2\r\n");
```

```
    msg2.data[0] = 3;
    msg2.data[1] = 4;
    while (true)
    {
        xQueueSend(queue, (void *)&msg2, 0);
        printf("TaskQueueSend 2\r\n");
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        msg2.data[0] += 100;
        msg2.data[1] += 100;
    }
    DataSent = true;
    vTaskDelete(taskHandle[1]);
}

void TaskQueueRecv(void *args)
{
    QueueHandle_t queue = (QueueHandle_t)args;
    QMsg_t msg;
    UBaseType_t itemNums;
    BaseType_t ret;
    while (true)
    {
        itemNums = uxQueueMessagesWaiting(queue);
        printf("Items = %d\r\n", itemNums);
        if (itemNums)
        {
            ret = xQueueReceive(queue, &msg, 0);
            if (ret == pdTRUE)
            {
                printf("TaskRecv, id = %d data[0] = %d data[1] = %d \r\n", msg.id, msg.data[0], msg.data[1]);
            }
            else
            {
                printf("TaskRecv, not received.\r\n");
            }
        }
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
    DataReceived = true;
    vTaskDelete(taskHandle[0]);
}

void app_main()
{
    xQueue = xQueueCreate(10, sizeof(QMsg_t));
    xTaskCreatePinnedToCore(TaskQueueSend1, "TaskQueueSend1", 4096, xQueue, 10, &taskHandle[1], APP_CPU_NUM);
    xTaskCreatePinnedToCore(TaskQueueSend2, "TaskQueueSend2", 4096, xQueue, 10, &taskHandle[1], APP_CPU_NUM);
    xTaskCreatePinnedToCore(TaskQueueRecv, "TaskQueueRecv", 4096, xQueue, 11, &taskHandle[0], APP_CPU_NUM);
    while (true)
```

```
                printf("TaskRecv, not received.\r\n");
            }
        }
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
    DataReceived = true;
    vTaskDelete(taskHandle[0]);
}

void app_main()
{
    xQueue = xQueueCreate(10, sizeof(QMsg_t));
    xTaskCreatePinnedToCore(TaskQueueSend1, "TaskQueueSend1", 4096, xQueue, 10, &taskHandle[1], APP_CPU_NUM);
    xTaskCreatePinnedToCore(TaskQueueSend2, "TaskQueueSend2", 4096, xQueue, 10, &taskHandle[1], APP_CPU_NUM);
    xTaskCreatePinnedToCore(TaskQueueRecv, "TaskQueueRecv", 4096, xQueue, 11, &taskHandle[0], APP_CPU_NUM);
    while (true)
    {
        if (DataSent && DataReceived)
        {
            printf("Restarting now.\n");
            vTaskDelay(1000 / portTICK_PERIOD_MS);
            fflush(stdout);
            esp_restart();
        }
    }
}
```

Results:



ESP-L03

Source code:

```c
#include <stdio.h>
#include <stdbool.h>
#include <unistd.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/FreeRTOSConfig.h"
#include "esp_system.h"
#include "esp_spi_flash.h"
#include "rtc_wdt.h"

static const char *pcTextForTask1 = "Task 1 is running\r\n";
static const char *pcTextForTask2 = "Task 2 is running\r\n";

void vTask1Function(void *pvParameters)
{
    char *pcTaskName;
    const TickType_t xDelay250ms = pdMS_TO_TICKS(250);
    pcTaskName = (char * ) pvParameters;
    for(;;)
    {
        printf ( pcTaskName );
        vTaskDelay(xDelay250ms);
    }
}
void vTask2Function(void *pvParameters)
{
    char *pcTaskName;
    const TickType_t xDelay250ms = pdMS_TO_TICKS(1250);
    pcTaskName = (char * ) pvParameters;
    for(;;)
    {
        printf ( pcTaskName );
        vTaskDelay(xDelay250ms);
    }
}

void app_main(void)
{
    //create task 1 at priority 1
    xTaskCreate(vTask1Function, "Task 1", 4096, (void*)pcTextForTask1, 1, NULL);
    //create task 2 at priority 2
    xTaskCreate(vTask2Function, "Task 2", 4096, (void*)pcTextForTask2, 2, NULL);
    //start task scheduler
    //rtc_wdt_protect_off();
    //rtc_wdt_disable();
    //rtc_wdt_feed();
    //vTaskStartScheduler();
}
```