

## EMBEDDED SYSTEMS – CC01

### LAB 2

### ESP

#### ESP-L04

*Source code:*

```
79 //
80 #include <stdio.h>
81 #include "freertos/FreeRTOS.h"
82 #include "freertos/task.h"
83 #include "freertos/queue.h"
84 #include "esp_system.h"
85
86 TaskHandle_t taskHandle[3];
87 QueueHandle_t xQueue;
88
89 typedef struct
90 {
91     int id;
92     int data[10];
93 } QMsg_t;
94
95 bool DataSent = false;
96 bool DataReceived = false;
97
98 void TaskQueueSend1(void *args)
99 {
100     QueueHandle_t queue = (QueueHandle_t)args;
101     QMsg_t msg1;
102     msg1.id = 1;
103     msg1.data[0] = 1;
104     msg1.data[1] = 2;
105     while (true)
106     {
107         xQueueSend(queue, (void *)&msg1, 0);
108         printf("TaskQueueSend 1\r\n");
109         vTaskDelay(1000 / portTICK_PERIOD_MS);
110         msg1.data[0] += 10;
111         msg1.data[1] += 10;
112     }
113     DataSent = true;
114     vTaskDelete(taskHandle[1]);
115 }
116
117 void TaskQueueSend2(void *args)
118 {
119     QueueHandle_t queue = (QueueHandle_t)args;
120     QMsg_t msg2;
121     msg2.id = 2;
122     msg2.data[0] = 3;
123     msg2.data[1] = 4;
124     while (true)
125     {
126         xQueueSend(queue, (void *)&msg2, 0);
127         printf("TaskQueueSend 2\r\n");
```

```

122     msg2.data[0] = 3;
123     msg2.data[1] = 4;
124     while (true)
125     {
126         xQueueSend(queue, (void *)&msg2, 0);
127         printf("TaskQueueSend 2\r\n");
128         vTaskDelay(1000 / portTICK_PERIOD_MS);
129         msg2.data[0] += 100;
130         msg2.data[1] += 100;
131     }
132     DataSent = true;
133     vTaskDelete(taskHandle[1]);
134 }
135
136 void TaskQueueRecv(void *args)
137 {
138     QueueHandle_t queue = (QueueHandle_t)args;
139     QMsg_t msg;
140     BaseType_t itemNums;
141     BaseType_t ret;
142     while (true)
143     {
144         itemNums = uxQueueMessagesWaiting(queue);
145         printf("Items = %d\r\n", itemNums);
146         if (itemNums)
147         {
148             ret = xQueueReceive(queue, &msg, 0);
149             if (ret == pdTRUE)
150             {
151                 printf("TaskRecv, id = %d data[0] = %d data[1] = %d \r\n", msg.id, msg.data[0], msg.data[1]);
152             }
153             else
154             {
155                 printf("TaskRecv, not received.\r\n");
156             }
157         }
158         vTaskDelay(1000 / portTICK_PERIOD_MS);
159     }
160     DataReceived = true;
161     vTaskDelete(taskHandle[0]);
162 }
163
164 void app_main()
165 {
166     xQueue = xQueueCreate(10, sizeof(QMsg_t));
167     xTaskCreatePinnedToCore(TaskQueueSend1, "TaskQueueSend1", 4096, xQueue, 10, &taskHandle[1], APP_CPU_NUM);
168     xTaskCreatePinnedToCore(TaskQueueSend2, "TaskQueueSend2", 4096, xQueue, 10, &taskHandle[1], APP_CPU_NUM);
169     xTaskCreatePinnedToCore(TaskQueueRecv, "TaskQueueRecv", 4096, xQueue, 11, &taskHandle[0], APP_CPU_NUM);
170     while (true)

```

```

155         printf("TaskRecv, not received.\r\n");
156     }
157 }
158     vTaskDelay(1000 / portTICK_PERIOD_MS);
159 }
160     DataReceived = true;
161     vTaskDelete(taskHandle[0]);
162 }
163
164 void app_main()
165 {
166     xQueue = xQueueCreate(10, sizeof(QMsg_t));
167     xTaskCreatePinnedToCore(TaskQueueSend1, "TaskQueueSend1", 4096, xQueue, 10, &taskHandle[1], APP_CPU_NUM);
168     xTaskCreatePinnedToCore(TaskQueueSend2, "TaskQueueSend2", 4096, xQueue, 10, &taskHandle[1], APP_CPU_NUM);
169     xTaskCreatePinnedToCore(TaskQueueRecv, "TaskQueueRecv", 4096, xQueue, 11, &taskHandle[0], APP_CPU_NUM);
170     while (true)
171     {
172         if (DataSent && DataReceived)
173         {
174             printf("Restarting now.\n");
175             vTaskDelay(1000 / portTICK_PERIOD_MS);
176             fflush(stdout);
177             esp_restart();
178         }
179     }
180 }

```

*Results:*

```
ESP-IDF 4.4 CMD - "C:\Espressif\idf_cmd_init.bat" esp-idf-e91d384503485fbb54f6ce3d11e841fe
I (256) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (262) heap_init: At 3FFB2C48 len 0002D3B8 (180 KiB): DRAM
I (268) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (275) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (281) heap_init: At 4008B47C len 00014B84 (82 KiB): IRAM
I (288) spi_flash: detected chip: generic
I (292) spi_flash: flash io: dio
W (296) spi_flash: Detected size(4096k) larger than the size in the binary image header(2048k). Using the size in the binary image header.
I (310) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
TaskQueueSend 1
Items = 1
TaskRcv, id = 1 data[0] = 1 data[1] = 2
TaskQueueSend 2
Items = 1
TaskRcv, id = 2 data[0] = 3 data[1] = 4
TaskQueueSend 2
TaskQueueSend 1
Items = 2
TaskRcv, id = 2 data[0] = 103 data[1] = 104
TaskQueueSend 2
TaskQueueSend 1
Items = 3
TaskRcv, id = 1 data[0] = 11 data[1] = 12
TaskQueueSend 2
TaskQueueSend 1
Items = 4
TaskRcv, id = 2 data[0] = 203 data[1] = 204
TaskQueueSend 2
```

## ESP-L03

*Source code:*

```

1  #include <stdio.h>
2  #include <stdbool.h>
3  #include <unistd.h>
4  #include "sdkconfig.h"
5  #include "freertos/FreeRTOS.h"
6  #include "freertos/task.h"
7  #include "freertos/FreeRTOSConfig.h"
8  #include "esp_system.h"
9  #include "esp_spi_flash.h"
10 #include "rtc_wdt.h"
11
12 static const char *pcTextForTask1 = "Task 1 is running\r\n";
13 static const char *pcTextForTask2 = "Task 2 is running\r\n";
14
15 void vTask1Function(void *pvParameters)
16 {
17     char *pcTaskName;
18     const TickType_t xDelay250ms = pdMS_TO_TICKS(250);
19     pcTaskName = (char * ) pvParameters;
20     for(;;)
21     {
22         printf ( pcTaskName );
23         vTaskDelay(xDelay250ms);
24     }
25 }
26 void vTask2Function(void *pvParameters)
27 {
28     char *pcTaskName;
29     const TickType_t xDelay250ms = pdMS_TO_TICKS(1250);
30     pcTaskName = (char * ) pvParameters;
31     for(;;)
32     {
33         printf ( pcTaskName );
34         vTaskDelay(xDelay250ms);
35     }
36 }
37
38 void app_main(void)
39 {
40     //create task 1 at priority 1
41     xTaskCreate(vTask1Function, "Task 1", 4096, (void*)pcTextForTask1, 1, NULL);
42     //create task 2 at priority 2
43     xTaskCreate(vTask2Function, "Task 2", 4096, (void*)pcTextForTask2, 2, NULL);
44     //start task scheduler
45     //rtc_wdt_protect_off();
46     //rtc_wdt_disable();
47     //rtc_wdt_feed();
48     //vTaskStartScheduler();
49 }

```