**REPORT LAB**

**EMBEDDED SYSTEM - CO3054**

*Group:* **CC02**

*Student:* **Dương Gia An – 1952163**

1. **LAB 3 - Preemptive with Slicing**
2. Prioritized Pre-emptive Scheduling with Time Slicing.  
   Graphical user interface, text, application

   Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, website

Description automatically generated

Code:

|  |
| --- |
| #include <stdio.h>  #include "sdkconfig.h"  #include "freertos/FreeRTOS.h"  #include "freertos/task.h"  #include "esp\_system.h"  #include "esp\_spi\_flash.h"  #include "freertos/FreeRTOSConfig.h"  **void** **vTaskFunction**(**void** \*pvParameters)  {  **char** \*pcTaskName;  **const** TickType\_t xDelay250ms = pdMS\_TO\_TICKS(**250**);  /\*The string to print out is passed in via the parameter .  Cast this to a character pointer . \*/  pcTaskName = (**char**\*) pvParameters;  /\*As per most tasks, this task is implemented in  an infinite loop . \*/  **for** (;;)  {  /\*Print out the name of this task . \*/  printf(pcTaskName);  printf("Hello**\n**");  /\*Delay for a period . This time a call to vTaskDelay ()  is used which places the task into the Blocked state  until the delay period has expired . The parameter takes  a time specified in " ticks ", and the pdMS\_TO\_TICKS () macro  is used (where the xDelay250ms constant is declared) to  convert 250 milliseconds into an equivalent time in ticks .\*/  vTaskDelay(xDelay250ms);  }  vTaskDelete(NULL);  }  **void** **vTaskFunction1**(**void** \*pvParameters)  {  **char** \*pcTaskName;  **const** TickType\_t xDelay500ms = pdMS\_TO\_TICKS(**500**);  /\*The string to print out is passed in via the parameter .  Cast this to a character pointer . \*/  pcTaskName = (**char**\*) pvParameters;  /\*As per most tasks, this task is implemented in  an infinite loop . \*/  **for** (;;)  {  /\*Print out the name of this task . \*/  printf(pcTaskName);  printf("Hello**\t**Yo yo**\n**");  /\*Delay for a period . This time a call to vTaskDelay ()  is used which places the task into the Blocked state  until the delay period has expired . The parameter takes  a time specified in " ticks ", and the pdMS\_TO\_TICKS () macro  is used (where the xDelay250ms constant is declared) to  convert 250 milliseconds into an equivalent time in ticks .\*/  vTaskDelay(xDelay500ms);  }  vTaskDelete(NULL);  }  **static**  **const** **char** \*pcTextForTask1 = "Task 1 is running **\r\n**";  **static**  **const** **char** \*pcTextForTask2 = "Task 2 is running **\r\n**";  **static**  **const** **char** \*pcTextForTask0 = "Task 0 is running **\r\n**";  **void** **app\_main**(**void**)  {  printf("===============**\n**");  xTaskCreate(vTaskFunction, " Task 0 Ilde", **2048**,  (**void**\*) pcTextForTask0, **0**, NULL);  /\*Create the first task at priority 1.  The priority is the second to last parameter . \*/  xTaskCreate(vTaskFunction1, " Task 1", **2048**,  (**void**\*) pcTextForTask1, **1**, NULL);  /\*Create the second task at priority 2, which is higher than a priority of 1.  The priority is the second to last parameter . \*/  xTaskCreate(vTaskFunction, " Task 2", **2048**,  (**void**\*) pcTextForTask2, **0**, NULL);    /\*Start the scheduler so the tasks start executing . \*/  //vTaskStartScheduler();  /\*Will not reach here . \*/  } |

Text

Description automatically generated with medium confidence

**Explain result:** Task Idle , task 1, task 2 add to schedule same time.

And task 1 has Largest Priotirity is 1 so it run first ( print Hello Yo yo).

Then task 0 ( Idle task ) and task 2 run.

1. **Prioritized Pre-emptive Scheduling (without Time Slicing)**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, text, website

Description automatically generated**

**Code:**

|  |
| --- |
| **void** **app\_main**(**void**)  {  printf("===============**\n**");  xTaskCreate(vTaskFunction, " Task 0 Ilde", **2048**,  (**void** \*)pcTextForTask0, **0**, NULL);  /\* Create the first task at priority 1.  The priority is the second to last parameter . \*/    /\* Create the second task at priority 2 ,  which is higher than a priority of 1.  The priority is the second to last parameter . \*/  xTaskCreate(vTaskFunction, " Task 2", **2048**,  (**void** \*)pcTextForTask2, **0**, NULL);  //Delay add Task 1 after 1000ms  **vTaskDelay(1000 / portTICK\_PERIOD\_MS);**  xTaskCreate(vTaskFunction1, " Task 1", **2048**,  (**void** \*)pcTextForTask1, **1**, NULL);    /\* Start the scheduler so the tasks start executing . \*/  //vTaskStartScheduler();  /\* Will not reach here . \*/  } |

**Result:**

**Text

Description automatically generated with medium confidence**

**Explain Result:**

I add Task Idle and Task 2 first and they run. (delay 250ms each)

After 1000ms ( 4 times for Idle and Task 2 runed) , I add task 1 to schedule. Task 1 has Highest Priority, so it run.

1. **Graphical user interface, text, application

   Description automatically generated  
   Text

   Description automatically generated**

**Code:**

|  |
| --- |
| #include <stdio.h>  #include "sdkconfig.h"  #include "freertos/FreeRTOS.h"  #include "freertos/task.h"  #include "esp\_system.h"  #include "esp\_spi\_flash.h"  #include "freertos/FreeRTOSConfig.h"  **void** **vTaskFunction**(**void** \*pvParameters)  {  **char** \*pcTaskName;  **const** TickType\_t xDelay250ms = pdMS\_TO\_TICKS(**250**);  TickType\_t count1000ms = pdMS\_TO\_TICKS(**0**);  /\* The string to print out is passed in via the parameter .  Cast this to a character pointer . \*/  pcTaskName = (**char** \*)pvParameters;  /\* As per most tasks , this task is implemented in  an infinite loop . \*/  **for** (;;)  {  /\* Print out the name of this task . \*/  printf(pcTaskName);  printf("Hello**\n**");  /\* Delay for a period . This time a call to vTaskDelay ()  is used which places the task into the Blocked state  until the delay period has expired . The parameter takes  a time specified in " ticks " , and the pdMS\_TO\_TICKS () macro  is used ( where the xDelay250ms constant is declared ) to  convert 250 milliseconds into an equivalent time in ticks .\*/  vTaskDelay(xDelay250ms);  **if** (count1000ms == pdMS\_TO\_TICKS(**1000**)){  taskYIELD();  }  **else** count1000ms += pdMS\_TO\_TICKS(**250**);  }  vTaskDelete(NULL);  }  **void** **vTaskFunction1**(**void** \*pvParameters)  {  **char** \*pcTaskName;  **const** TickType\_t xDelay500ms = pdMS\_TO\_TICKS(**500**);  /\* The string to print out is passed in via the parameter .  Cast this to a character pointer . \*/  pcTaskName = (**char** \*)pvParameters;  /\* As per most tasks , this task is implemented in  an infinite loop . \*/  **for** (;;)  {  /\* Print out the name of this task . \*/  printf(pcTaskName);  printf("Hello**\t**Yo yo**\n**");  /\* Delay for a period . This time a call to vTaskDelay ()  is used which places the task into the Blocked state  until the delay period has expired . The parameter takes  a time specified in " ticks " , and the pdMS\_TO\_TICKS () macro  is used ( where the xDelay250ms constant is declared ) to  convert 250 milliseconds into an equivalent time in ticks .\*/  vTaskDelay(xDelay500ms);  }  vTaskDelete(NULL);  }  **static** **const** **char** \*pcTextForTask1 = "Task 1 is running **\r\n**";  **static** **const** **char** \*pcTextForTask2 = "Task 2 is running **\r\n**";  **static** **const** **char** \*pcTextForTask0 = "Task 0 is running **\r\n**";  **void** **app\_main**(**void**)  {  printf("===============**\n**");  xTaskCreate(vTaskFunction, " Task 0 Ilde", **2048**,  (**void** \*)pcTextForTask0, **0**, NULL);  /\* Create the first task at priority 1.  The priority is the second to last parameter . \*/    /\* Create the second task at priority 2 ,  which is higher than a priority of 1.  The priority is the second to last parameter . \*/    //Delay add Task 1 and Task 2 after 1000ms  vTaskDelay(**1000** / portTICK\_PERIOD\_MS);  xTaskCreate(vTaskFunction1, " Task 1", **2048**,  (**void** \*)pcTextForTask1, **1**, NULL);  xTaskCreate(vTaskFunction, " Task 2", **2048**,  (**void** \*)pcTextForTask2, **0**, NULL);  /\* Start the scheduler so the tasks start executing . \*/  //vTaskStartScheduler();  /\* Will not reach here . \*/  } |

**Result:**

**Text

Description automatically generated**

**Explain Result:**

I add task 0 ( Idle first).  
After 1000ms, I add task 1 and 2. As the same time, TaskYIELD() in Idle run so Task 1 has Largest Priority run then.

1. **Extra Exercise**

|  |
| --- |
| 1. #include <stdio.h> 2. #include "sdkconfig.h" 3. #include "freertos/FreeRTOS.h" 4. #include "freertos/task.h" 5. #include "esp\_system.h" 6. #include "esp\_spi\_flash.h" 7. #include "freertos/FreeRTOSConfig.h" 8. **void** **vTaskFunction\_Idle**(**void** \*pvParameters) 9. { 10. **char** \*pcTaskName; 11. **const** TickType\_t xDelay250ms = pdMS\_TO\_TICKS(**250**); 12. TickType\_t count1000ms = pdMS\_TO\_TICKS(**0**); 13. /\* The string to print out is passed in via the parameter . 14. Cast this to a character pointer . \*/ 15. pcTaskName = (**char** \*)pvParameters; 16. /\* As per most tasks , this task is implemented in 17. an infinite loop . \*/ 18. **for** (;;) 19. { 20. /\* Print out the name of this task . \*/ 21. printf(pcTaskName);   // Hook IDLE TASK   1. /\* Print chip information \*/ 2. **esp\_chip\_info\_t** chip\_info; 3. esp\_chip\_info(&chip\_info); 4. printf("This is %s chip with %d CPU core(s), WiFi%s%s, ", 5. CONFIG\_IDF\_TARGET, 6. chip\_info.cores, 7. (chip\_info.features & CHIP\_FEATURE\_BT) ? "/BT" : "", 8. (chip\_info.features & CHIP\_FEATURE\_BLE) ? "/BLE" : ""); 9. printf("silicon revision %d, ", chip\_info.revision); 10. printf("%dMB %s flash**\n**", spi\_flash\_get\_chip\_size() / (**1024** \* **1024**), 11. (chip\_info.features & CHIP\_FEATURE\_EMB\_FLASH) ? "embedded" : "external"); 12. printf("Minimum free heap size: %d bytes**\n**", esp\_get\_minimum\_free\_heap\_size()); 13. /\* Delay for a period . This time a call to vTaskDelay () 14. is used which places the task into the Blocked state 15. until the delay period has expired . The parameter takes 16. a time specified in " ticks " , and the pdMS\_TO\_TICKS () macro 17. is used ( where the xDelay250ms constant is declared ) to 18. convert 250 milliseconds into an equivalent time in ticks .\*/ 19. vTaskDelay(xDelay250ms); 20. **if** (count1000ms == pdMS\_TO\_TICKS(**1000**)){ 21. taskYIELD(); 22. } 23. **else** count1000ms += pdMS\_TO\_TICKS(**250**); 24. } 25. vTaskDelete(NULL); 26. } 27. **void** **vTaskFunction2**(**void** \*pvParameters) 28. { 29. **char** \*pcTaskName; 30. **const** TickType\_t xDelay250ms = pdMS\_TO\_TICKS(**250**); 31. /\* The string to print out is passed in via the parameter . 32. Cast this to a character pointer . \*/ 33. pcTaskName = (**char** \*)pvParameters; 34. /\* As per most tasks , this task is implemented in 35. an infinite loop . \*/ 36. **for** (;;) 37. { 38. /\* Print out the name of this task . \*/ 39. printf(pcTaskName); 40. printf("Hello **\n**"); 41. /\* Delay for a period . This time a call to vTaskDelay () 42. is used which places the task into the Blocked state 43. until the delay period has expired . The parameter takes 44. a time specified in " ticks " , and the pdMS\_TO\_TICKS () macro 45. is used ( where the xDelay250ms constant is declared ) to 46. convert 250 milliseconds into an equivalent time in ticks .\*/ 47. vTaskDelay(xDelay250ms); 49. } 50. vTaskDelete(NULL); 51. } 52. **void** **vTaskFunction1**(**void** \*pvParameters) 53. { 54. **char** \*pcTaskName; 55. **const** TickType\_t xDelay500ms = pdMS\_TO\_TICKS(**500**); 56. /\* The string to print out is passed in via the parameter . 57. Cast this to a character pointer . \*/ 58. pcTaskName = (**char** \*)pvParameters; 59. /\* As per most tasks , this task is implemented in 60. an infinite loop . \*/ 61. **for** (;;) 62. { 63. /\* Print out the name of this task . \*/ 64. printf(pcTaskName); 65. printf("Hello**\t**Yo yo**\n**"); 66. /\* Delay for a period . This time a call to vTaskDelay () 67. is used which places the task into the Blocked state 68. until the delay period has expired . The parameter takes 69. a time specified in " ticks " , and the pdMS\_TO\_TICKS () macro 70. is used ( where the xDelay250ms constant is declared ) to 71. convert 500 milliseconds into an equivalent time in ticks .\*/ 72. vTaskDelay(xDelay500ms); 73. } 74. vTaskDelete(NULL); 75. } 76. **static** **const** **char** \*pcTextForTask1 = "Task 1 is running **\r\n**"; 77. **static** **const** **char** \*pcTextForTask2 = "Task 2 is running **\r\n**"; 78. **static** **const** **char** \*pcTextForTask0 = "Task 0 is running **\r\n**"; 79. **void** **app\_main**(**void**) 80. { 82. printf("===============**\n**"); 83. xTaskCreate(vTaskFunction\_Idle, " Task 0 Ilde", **2048**, 84. (**void** \*)pcTextForTask0, **0**, NULL); 85. /\* Create the first task at priority 1. 86. The priority is the second to last parameter . \*/ 88. /\* Create the second task at priority 2 , 89. which is higher than a priority of 1. 90. The priority is the second to last parameter . \*/ 92. //Delay add Task 1 after 1000ms 93. vTaskDelay(**1000** / portTICK\_PERIOD\_MS); 94. xTaskCreate(vTaskFunction1, " Task 1", **2048**, 95. (**void** \*)pcTextForTask1, **1**, NULL); 96. xTaskCreate(vTaskFunction2, " Task 2", **2048**, 97. (**void** \*)pcTextForTask2, **0**, NULL); 98. /\* Start the scheduler so the tasks start executing . \*/ 99. //vTaskStartScheduler(); 100. /\* Will not reach here . \*/ 101. } |

**Result:**

**Text

Description automatically generated**

1. **FreeRTOS Queue Management**

Not finish yet.