RYERSON UNIVERSITY

Faculty of Engineering and Architectural Science

Department of Electrical and Computer Engineering

Course Number	COE 718
Course Title	Embedded Systems Design
Semester/Year	F2020
Lab No	4
Instructor Name	Saber Amini
Section No	03

Submission Date	11/18/2020
Due Date	11/18/2020

Name	Student ID	Signature*
Vatsal Shreekant	500771363	

**By signing above, you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at:*

www.ryerson.ca/senate/current/pol60.pdf

Introduction

The purpose of this lab is to understand real-time scheduling with uVision and the ARM Cortex-M3. Specifically, you will learn how to schedule and implement a Rate Monotonic Scheduling (RMS) algorithm (a type of Fixed Priority Scheduling (FPS)), and a solution to priority inversion. To implement these methods, you will be introduced to concepts such as virtual timers, inter-thread communication methods (signals and waits for threads), along with static priority and dynamic priority inversions. You are expected to have a thorough understanding of the previous lab dealing with uVision and RTX for the completion of this lab. Refer to lecture notes on background information pertaining to RMS and priority inversion.

Procedure

1) Load virtual_demo.c and priority_inv.c example project and complete the instructions in the lab manual.
2) Select the following packages under 'Manage Run-Time Environment' window and select OK button:
    - CMSIS>CORE.
    - CMSIS>RTOS(API)>Keil RTX.
    - Device > Startup.
3) Modify the 'virtual_demo.c' file to implement a RMS algorithm example using 3 different process as listed in the part 1 of the lab manual. Modify the 'priority_inv.c' file to schedule and implement the Mars Pathfinder (MP) problem and solution within a single .c file. The code implemented in parts 1 and 2 are listed as follows:

PART 1:

C:\Users\Owner\OneDrive - Ryerson University\4th Year\COE 718\Labs_new\Lab4\example1\Thread.c

```
1   /*----------------------------------------------------------------------------
2    * COE 718 - Lab4 - Part 1
3    * CMSIS-RTOS 'main' function template
4    *---------------------------------------------------------------------------*/
5
6   #define osObjectsPublic                     // define objects in main module
7   #include "osObjects.h"                      // RTOS object definitions
8   #include "cmsis_os.h"                                          // CMSIS RTOS header file
9   #include <stdio.h>
10  #include <math.h>
11  #include "Board_LED.h"                  // ::Board Support:LED
12  #include "LPC17xx.h"
13
14  void task_A (void const *argument);
15  void task_B (void const *argument);
16  void task_C (void const *argument);
17  osTimerId timer_0;
18  osTimerId timer_1;
19  osTimerId timer_2;
20  osThreadDef(task_A, osPriorityBelowNormal, 1, 0);
21  osThreadDef(task_B, osPriorityNormal, 1, 0);
22  osThreadDef(task_C, osPriorityAboveNormal, 1, 0);
23
24  //Virtual Timer declaration and call back method
25  osThreadId T_led_ID1;
26  osThreadId T_led_ID2;
27  osThreadId T_led_ID3;
28  void delay(int z)
29  {int x;
30  int y = z * 20000;
31  for ( x=0; x<y; x++){}}
32
33  // Toggle the LED associated with the timer
34  void callback (void const *param){
35    switch ( (uint32_t) param){
36      case 0:
37        osSignalSet (T_led_ID1,0x0A);
38        break;
39      case 1:
40        osSignalSet (T_led_ID2,0x0B);
41        break;
42      case 2:
43        osSignalSet (T_led_ID3,0x0C);
44        break;
45    }
46  }
47  osTimerDef(timer0_handle, callback);
48  osTimerDef(timer1_handle, callback);
49  osTimerDef(timer2_handle, callback);
50  // Flash LED 0, signal to thread 2, wait for 3 to finish
51  void task_A (void const *argument) {
52    for (;;) {
53      LED_On(0);
54    //  LED_On(1);
55      delay(20);
56      LED_Off(0);
57      //LED_Off(1);
58      osSignalWait (0x0A,osWaitForever);
59    }
60  }
61  // Flash LED 2, signal to thread 3, wait for thread 1 to finish
62  void task_B (void const *argument) {
63    for (;;)  {
64      LED_On(2);
65      //LED_On(3);
66      delay(10);
67      LED_Off(2);
68      //LED_Off(3);
69      osSignalWait (0x0B,osWaitForever);
70    }
71  }
```


Figure 1: Page 1 of Thread.c

```
72  // Flash LED 4, signal to thread 1, wait for thread 2 to finish
73  void task_C (void const *argument){
74    for (;;)  {
75      LED_On(4);
76      //LED_On(5);
77      delay(5);
78      LED_Off(4);
79      //LED_Off(5);
80      osSignalWait (0x0C,osWaitForever);
81    }
82  }
83
84  // Create and start threads
85  int main (void) {
86      LED_Init ();
87    //Virtual timer create and start
88    osThreadSetPriority(osThreadGetId() ,osPriorityHigh);
89    timer_0 = osTimerCreate(osTimer(timer0_handle), osTimerPeriodic, (void *)0);
90    timer_1 = osTimerCreate(osTimer(timer1_handle), osTimerPeriodic, (void *)1);
91    timer_2 = osTimerCreate(osTimer(timer2_handle), osTimerPeriodic, (void *)2);
92    //Signal and wait threads
93    T_led_ID1 = osThreadCreate(osThread(task_A), NULL);
94    T_led_ID2 = osThreadCreate(osThread(task_B), NULL);
95    T_led_ID3 = osThreadCreate(osThread(task_C), NULL);
96    osTimerStart(timer_0, 400);
97    osTimerStart(timer_1, 400);
98    osTimerStart(timer_2, 200);
99
100   osDelay(osWaitForever);
101
102   for (; ;);
103 }
104
105
106
107
```


Figure 2: Page 2 of Thread.c

PART 2:

C:\Users\Owner\OneDrive - Ryerson University\4th Year\COE 718\Labs_new\Lab4\example1\Semaphore.c

```
1   /*----------------------------------------------------------------------------
2    * COE 718 - Lab4 - Part 2
3    * CMSIS-RTOS 'main' function template
4    *---------------------------------------------------------------------------*/
5
6   #define osObjectsPublic                     // define objects in main module
7   #include "osObjects.h"                      // RTOS object definitions
8   #include "cmsis_os.h"                                          // CMSIS RTOS header file
9   #include <stdio.h>
10  #include <math.h>
11  #include "Board_LED.h"                  // ::Board Support:LED
12  #include "RTE_Components.h"             // Component selection
13
14
15  /*----------------------------------------------------------------------------
16    CMSIS RTX Priority Inversion Example
17    Priority Inversion = leave commented lines commented
18    Priority Elevation = uncomment the 2 commented lines
19    Anita Tino
20  *---------------------------------------------------------------------------*/
21
22  int main(void)
23  {
24    osKernelInitialize ();
25    LED_Initialize();
26    t_main = osThreadGetId ();
27    osThreadSetPriority(t_main,osPriorityHigh);
28    semaphore = osSemaphoreCreate(osSemaphore(semaphore), 10);
29
30    t_P3 = osThreadCreate(osThread(P3), NULL);
31
32     osDelay(500);
33
34    t_P2 = osThreadCreate(osThread(P2), NULL);
35
36    osDelay(100);
37    t_P1 = osThreadCreate(osThread(P1), NULL);
38
39    osThreadTerminate(t_main);
40     osKernelStart ();
41    for (;;) {}
42  }
43
44
45  int flag = 0;
46
47  osSemaphoreId semaphore; //Semaphore ID
48  osSemaphoreDef(semaphore); //Semaphore definition
49
50  void P1 (void const *argument);
51  void P2 (void const *argument);
52  void P3 (void const *argument);
53
54  osThreadDef(P1, osPriorityHigh, 1, 0);
55  osThreadDef(P2, osPriorityNormal, 1, 0);
56  osThreadDef(P3, osPriorityBelowNormal, 1, 0);
57
58  osThreadId t_main,t_P1,t_P2,t_P3;
59
60  void delay(){
61    long k, count = 0;
62    for(k = 0; k < 100000; k++){
63            count++;
64        }
65    }
66
67  void P1 (void const *argument) {
68
69    int32_t val;
70
71    for (;;)
```


Figure 3: Page 1 of Semaphore.c

```
72      {
73        osSemaphoreWait (semaphore, osWaitForever); //Wait for the semaphore
74        LED_On(0);
75        delay();
76        delay();
77        delay();
78        delay();
79        delay();
80        delay();
81        delay();
82        delay();
83        delay();
84        osSignalSet(t_P3,0x01); //Call P3 to finish the task
85        osSignalWait(0x02, osWaitForever); //On recieving the Error for priority inversion, P2 will run
    instead, but is blocked by semaphore
86        LED_On(6);
87        LED_Off(6);
88        osSemaphoreRelease(semaphore); //Return the token back to the semaphore
89      }
90    }
91
92    void P2 (void const *argument) {
93
94      int32_t val;
95
96      for (;;)  {
97        osDelay(700); //Pass control to other tasks for 700 ms
98        val = osSemaphoreWait(semaphore,1); //Wait 1ms for free semaphore
99
100       if(val>9){ //if no time out, semaphore was acquired
101         LED_On(1); // free to run
102         LED_Off(1);
103         osSemaphoreRelease(semaphore); //Return the token back to the semaphore
104
105       }
106     }
107   }
108
109   void P3 (void const *argument) {
110     for (;;)
111     {
112       osSemaphoreWait (semaphore, osWaitForever); //Wait for the semaphore
113       LED_On(0);
114       delay();
115       delay();
116       delay();
117       delay();
118       delay();
119       delay();
120       delay();
121       delay();
122       delay();
123
124       osSignalWait(0x01, osWaitForever); //On recieving the Error for priority inversion, P2 will run
    instead, but is blocked by semaphore
125       LED_Off(6);
126       osSignalSet(t_P1,0x02); //Call P3 to finish the task
127
128       osSemaphoreRelease(semaphore); //Return the token back to the semaphore
129     }
130   }
131
132
```


Figure 4: Page 2 of Semaphore.c

4) Compile project using the build button and start the simulation by selecting the debug button.
5) Select debug mode to analyze performance of the threads using Performance Analyzer and the Event Viewer.

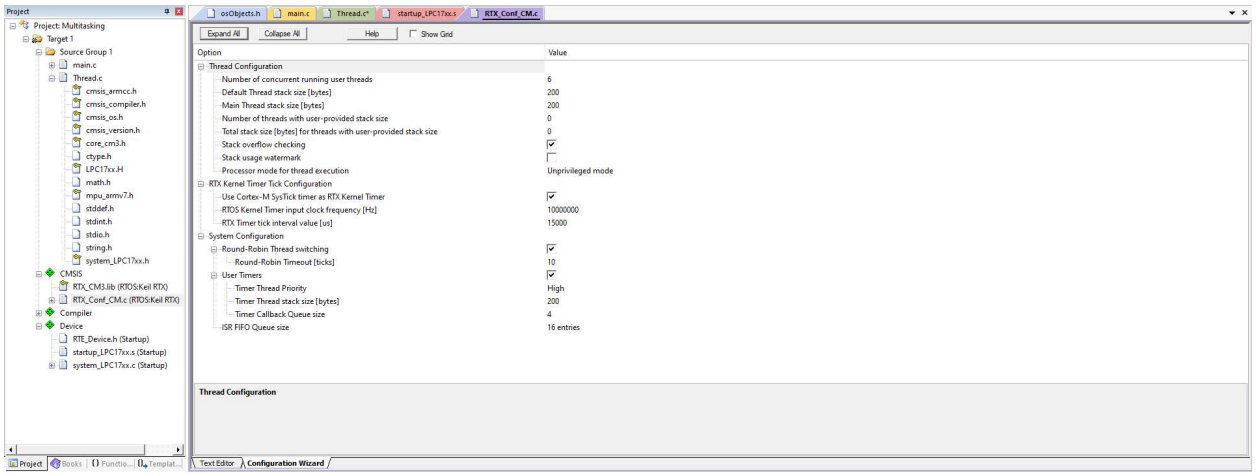

Figure 5: RTX_Conf_CM.c Configuration Wizard

Conclusion

When comparing and analyzing the results under debug mode, it is evident that the tasks are being prioritized and executed as per the priority thread and this was the initial assumption before implementing the code. Refer to figures 6, 7 and 8 for the results. As per the instructions in the lab manual, the code for the LED was not required. It is evident by looking at figure 6 that the lower the number, the higher the priority and hence the priorities are followed accordingly. Moreover, figures 7 and 8 highlight how the, the lower priority task C is executing its thread's workload on resource R1 and is pre-empted by the medium priority task B at 50ms. It can also be observed that task C will not execute again since it is pre-empted by both Tasks B and A.

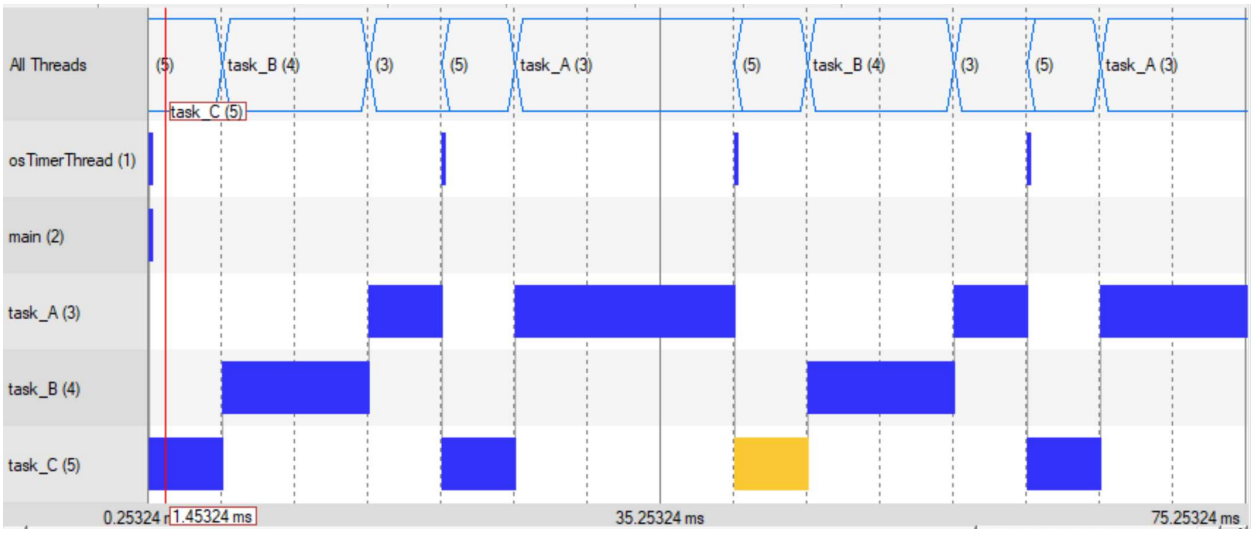

Figure 6: Event Viewer Window for Rate Monotonic Scheduling output

Figure 7: Problem representation of Mars Pathfinder

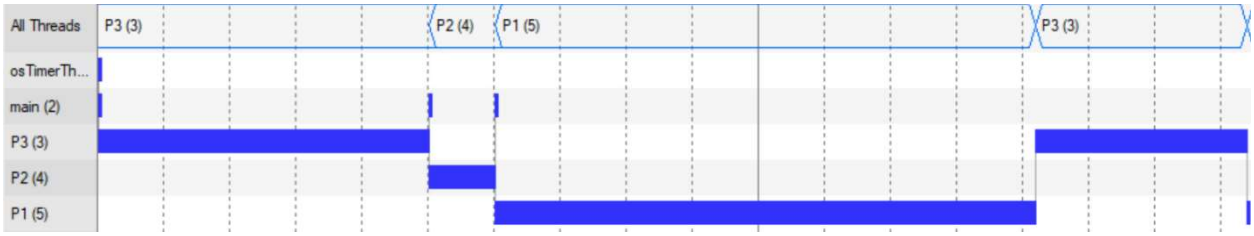

Figure 8: Solution to Mars Pathfinder

References

1) NXP User Manual, https://www.nxp.com/docs/en/user-guide/UM10360.pdf, 2020
2) ARM Keil User Guide, https://www.keil.com/support/man/docs/mcb1700/mcb1700_intro.htm, 2020