

| **Course Title:** | Embedded System Design |
| --- | --- |
| **Course Number:** | COE718 |
| **Semester/Year (e.g.F2016)** | F2022 |

| **Instructor:** | Dr. Gul Khan |
| --- | --- |

| **Assignment/Lab Number:** | 4 |
| --- | --- |
| **Assignment/Lab Title:** | Real-time Scheduling and Priority Inversion Understanding |

| **Submission Date:** | 11/05/2022 |
| --- | --- |
| **Due Date:** | 10/31/2022 |

| **Student**  **LAST Name** | **Student**  **FIRST Name** | **Student**  **Number** | **Section** | **Signature\*** |
| --- | --- | --- | --- | --- |
| Sharma | Nabhanya | 500826185 | 8 | NS |
|  |  |  |  |  |
|  |  |  |  |  |

\*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report 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: <http://www.ryerson.ca/senate/current/pol60.pdf>

Introduction

The objective of this lab is to provide a better understanding of real-time scheduling using uVision for ARM Cortex-M3 by practicing how to schedule and implement a Rate Monotonic Scheduling technique and Fixed Priority Scheduling. Furthermore, to implement these techniques, this lab introduces concepts of virtual timers, inter-thread communication methods (signals and waits for threads), along with static and dynamic priority inversions.

Assignment

**-Part 1:**

1. Virtual\_demo.c:

/\*----------------------------------------------------------------------------

\* CMSIS-RTOS 'main' function template

\*---------------------------------------------------------------------------\*/

#define osObjectsPublic // define objects in main module

#include "osObjects.h" // RTOS object definitions

#include "cmsis\_os.h" // CMSIS RTOS header file

#include <stdio.h>

#include <math.h>

#include "LED.h" // ::Board Support:LED

void led\_Thread1 (void const \*argument); // thread function

void led\_Thread2 (void const \*argument); // thread function

void led\_Thread3 (void const \*argument);

osThreadDef (led\_Thread1, osPriorityBelowNormal, 1, 0); // thread object

osThreadDef (led\_Thread2, osPriorityNormal, 1, 0); // thread object

osThreadDef (led\_Thread3, osPriorityAboveNormal, 1, 0);

/\*###########################################################

Virtual Timer declaration and call back method

############################################################\*/

osThreadId T\_led\_ID1;

osThreadId T\_led\_ID2;

osThreadId T\_led\_ID3;

uint32\_t delayA=20000;

uint32\_t delayB=10000;

uint32\_t delayC=5000;

void delay(int a){

int i;

int j;

int k;

j=(a\*1000);

for(i=0;i<=j;i++){

k++;

}

}

// Toggle the LED associated with the timer

void callback(void const \*param){

switch( (uint32\_t) param){

case 0:

osSignalSet(T\_led\_ID1,0x01);

break;

case 1:

osSignalSet(T\_led\_ID2,0x02);

break;

case 2:

osSignalSet(T\_led\_ID3,0x03);

break;

}

}

osTimerDef(timer0\_handle, callback);

osTimerDef(timer1\_handle, callback);

osTimerDef(timer2\_handle, callback);

//#############################################################

/\*#############################################################

Flash LED 0, signal to thread 2, wait for 3 to finish

\*#############################################################\*/

void led\_Thread1 (void const \*argument) {

for (;;) {

LED\_On(0);

delay(20000);

osSignalWait (0x01,osWaitForever);

LED\_Off(0);

}

}

/\*################################################################

Flash LED 2, signal to thread 3, wait for thread 1 to finish

\*################################################################\*/

void led\_Thread2 (void const \*argument) {

for (;;) {

LED\_On(2);

delay(10000);

osSignalWait (0x02,osWaitForever);

LED\_Off(2);

}

}

/\*################################################################

Flash LED 4, signal to thread 1, wait for thread 2 to finish

\*################################################################\*/

void led\_Thread3 (void const \*argument){

for (;;) {

LED\_On(4);

delay(5000);

osSignalWait (0x03,osWaitForever);

LED\_Off(4);

}

}

/\*###################################################################

Create and start threads

\*###################################################################\*/

int main (void) {

osKernelInitialize ();

//Virtual timer create and start

osTimerId timer\_0 = osTimerCreate(osTimer(timer0\_handle), osTimerPeriodic, (void \*)0);

osTimerId timer\_1 = osTimerCreate(osTimer(timer1\_handle), osTimerPeriodic, (void \*)1);

osTimerId timer\_2 = osTimerCreate(osTimer(timer2\_handle), osTimerPeriodic, (void \*)2);

LED\_Init();

osTimerStart(timer\_0, 40000);

osTimerStart(timer\_1, 40000);

osTimerStart(timer\_2, 20000);

//Signal and wait threads

T\_led\_ID1 = osThreadCreate(osThread(led\_Thread1), NULL);

T\_led\_ID2 = osThreadCreate(osThread(led\_Thread2), NULL);

T\_led\_ID3 = osThreadCreate(osThread(led\_Thread3), NULL);

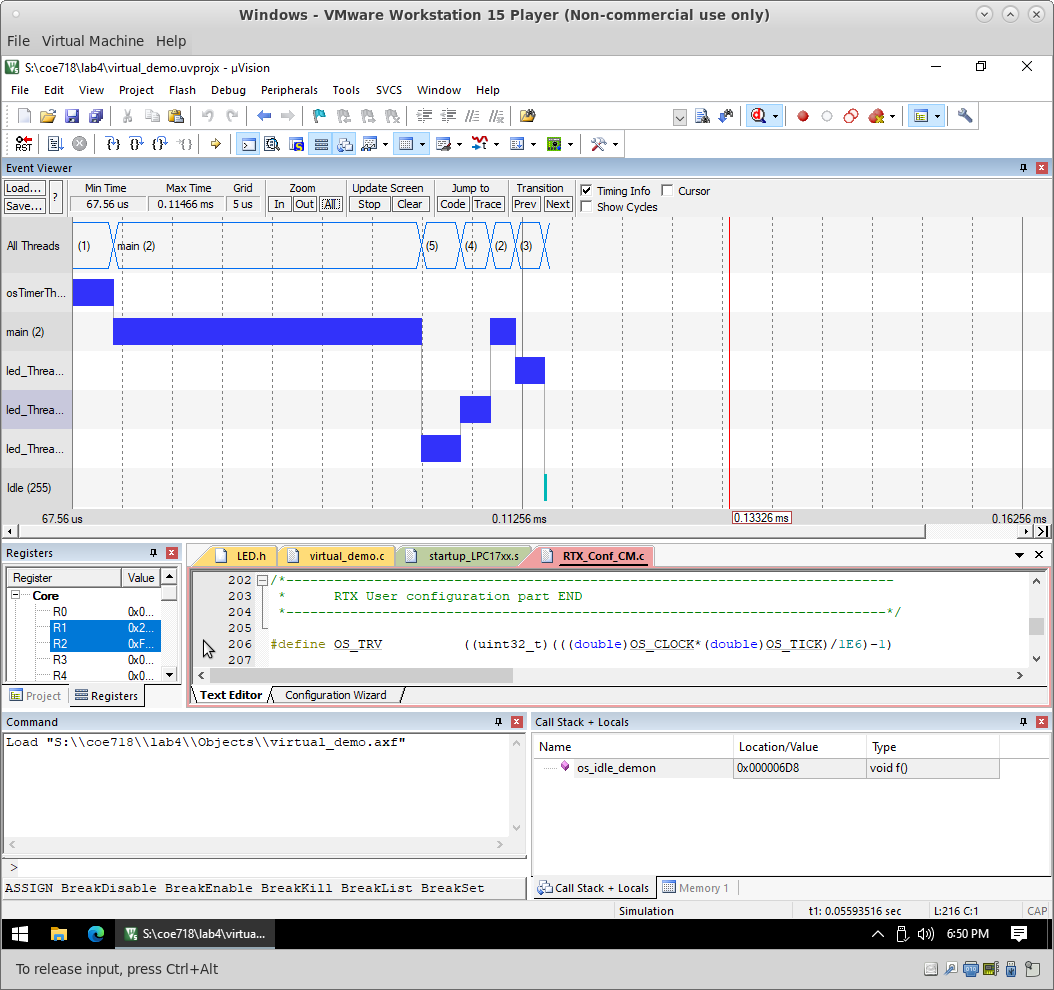
osKernelStart ();

osDelay(osWaitForever);

for (;;);

}

1. Event Viewer:



**-Part 2:**

1. priority\_inv.c:

/\*----------------------------------------------------------------------------

\* CMSIS-RTOS 'main' function template

\*---------------------------------------------------------------------------\*/

#define osObjectsPublic // define objects in main module

#include "osObjects.h" // RTOS object definitions

#include "cmsis\_os.h" // CMSIS RTOS header file

#include <stdio.h>

#include <math.h>

#include "LED.h" // ::Board Support:LED

#include "RTE\_Components.h" // Component selection

/\*----------------------------------------------------------------------------

CMSIS RTX Priority Inversion Example

Priority Inversion = leave commented lines commented

Priority Elevation = uncomment the 2 commented lines

Anita Tino

\*----------------------------------------------------------------------------\*/

void P1 (void const \*argument);

void P2 (void const \*argument);

void P3 (void const \*argument);

osThreadDef(P1, osPriorityHigh, 1, 0);

osThreadDef(P2, osPriorityNormal, 1, 0);

osThreadDef(P3, osPriorityBelowNormal, 1, 0);

osThreadId t\_main,t\_P1,t\_P2,t\_P3;

void delay(){

long k, count = 0;

for(k = 0; k < 100000000; k++){

count++;

}

}

void P1 (void const \*argument) {

for (;;)

{

//LED\_On(0);

delay(); //execute something, and after requires service from P3

//osThreadSetPriority(t\_P3, osPriorityHigh); //\*\*solution uncomment

osSignalSet(t\_P3,0x01); //call P3 to finish the task

//osSignalWait(0x02,osWaitForever); //Error => priority inversion, P2 will run instead

osThreadSetPriority(t\_P3,osPriorityBelowNormal); //\*\*solution uncomment

//LED\_On(6);

//LED\_Off(6);

}

}

void P2 (void const \*argument) {

for (;;) {

//LED\_On(1);

//LED\_Off(1);

}

}

void P3 (void const \*argument) {

for (;;) {

delay(); //do something

osSignalWait(0x01,osWaitForever);

//LED\_Off(0); //critical function to be requested by P1

osSignalSet(t\_P1,0x02);

}

}

int main(void)

{

osKernelInitialize ();

//LED\_Init();

t\_main = osThreadGetId ();

osThreadSetPriority(t\_main,osPriorityHigh);

t\_P3 = osThreadCreate(osThread(P3), NULL);

osDelay(5);

t\_P2 = osThreadCreate(osThread(P2), NULL);

osDelay(5);

t\_P1 = osThreadCreate(osThread(P1), NULL);

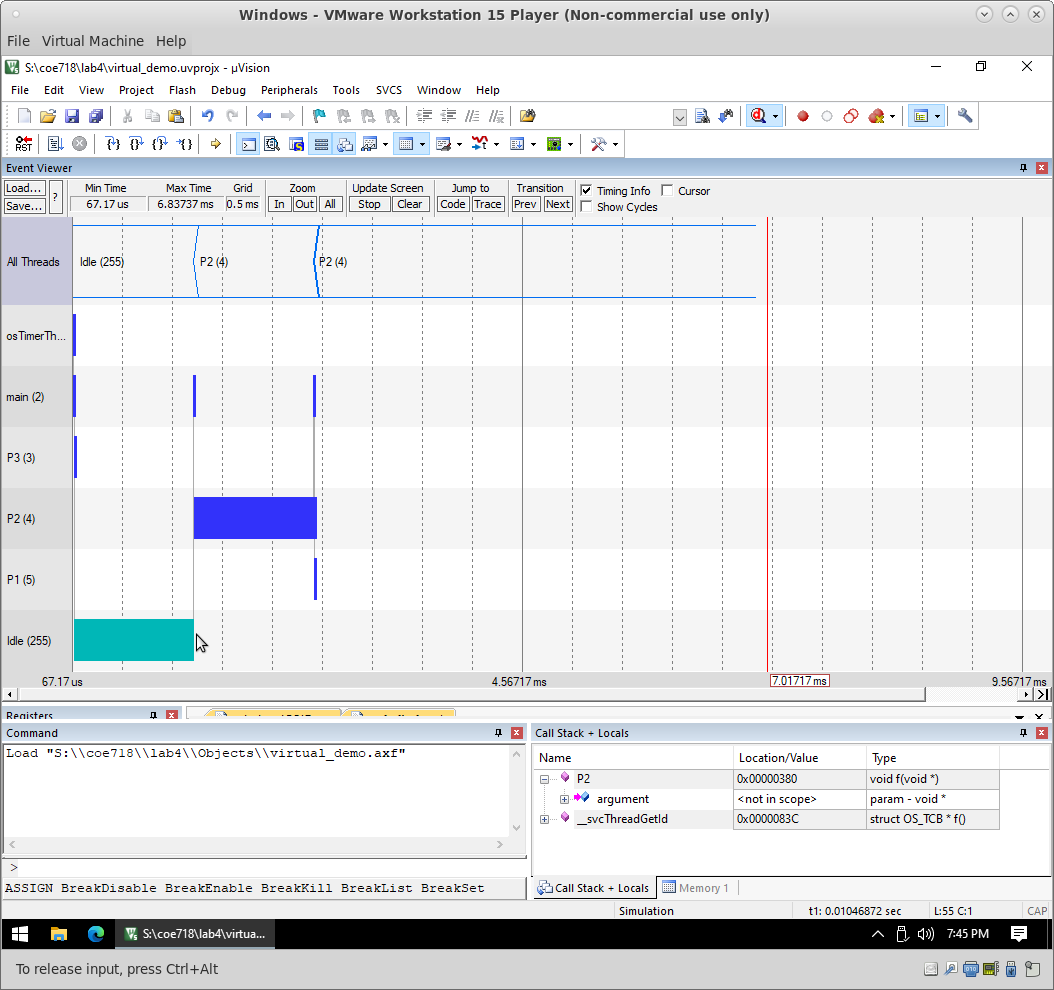
osThreadTerminate(t\_main);

osKernelStart ();

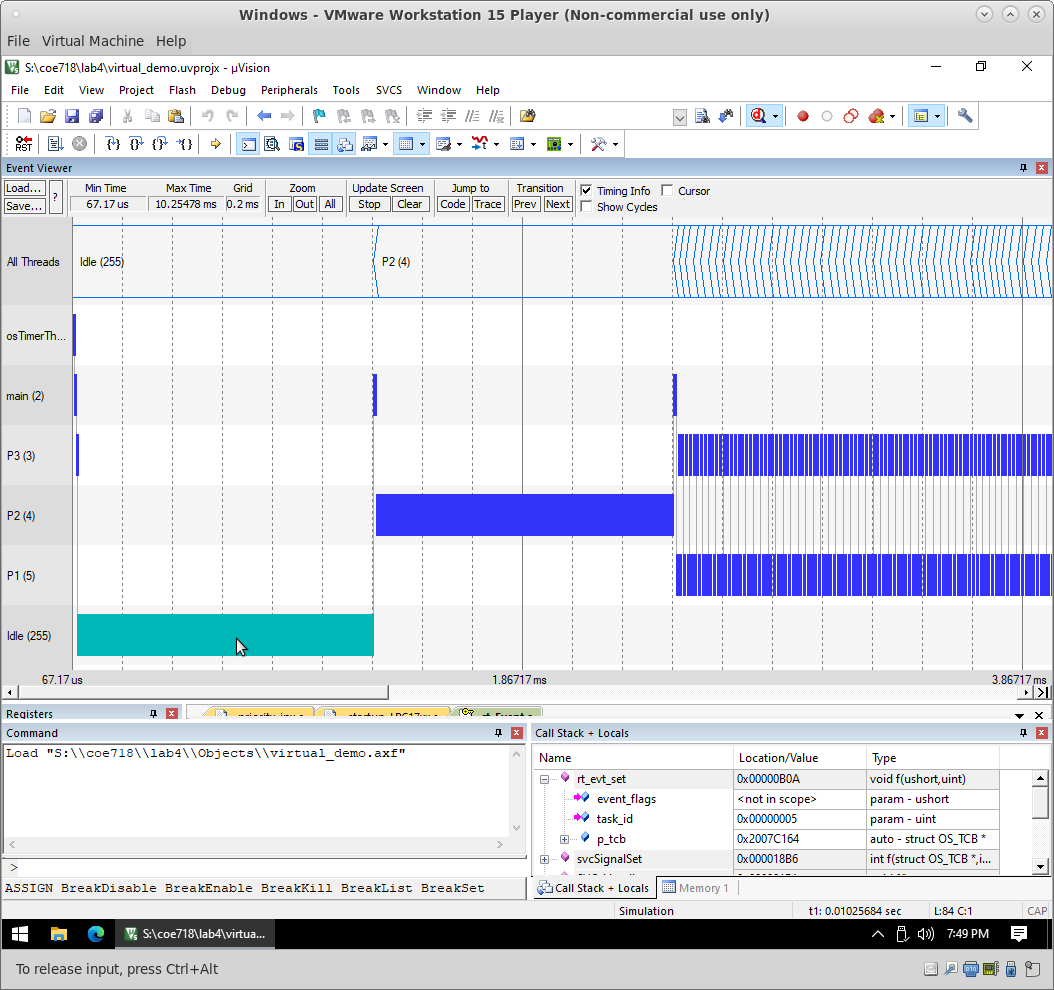
for (;;) {}

}

1. Event Viewer(Problem):



1. Event Viewer(Solution):

Conclusion

To summarize the assignment by performing the tasks of the lab it provided a better understanding of real-time scheduling using uVision for ARM Cortex-M3 by practicing how to schedule and implement a Rate Monotonic Scheduling technique and Fixed Priority Scheduling.