Phillip Sortomme
CPE 403
TI-RTOS Lab1

Date Submitted:

Youtube link: https://youtu.be/MEoSHbbxs40

Lab 01:

The task was to simply copy and run the code provided with no modifications

---

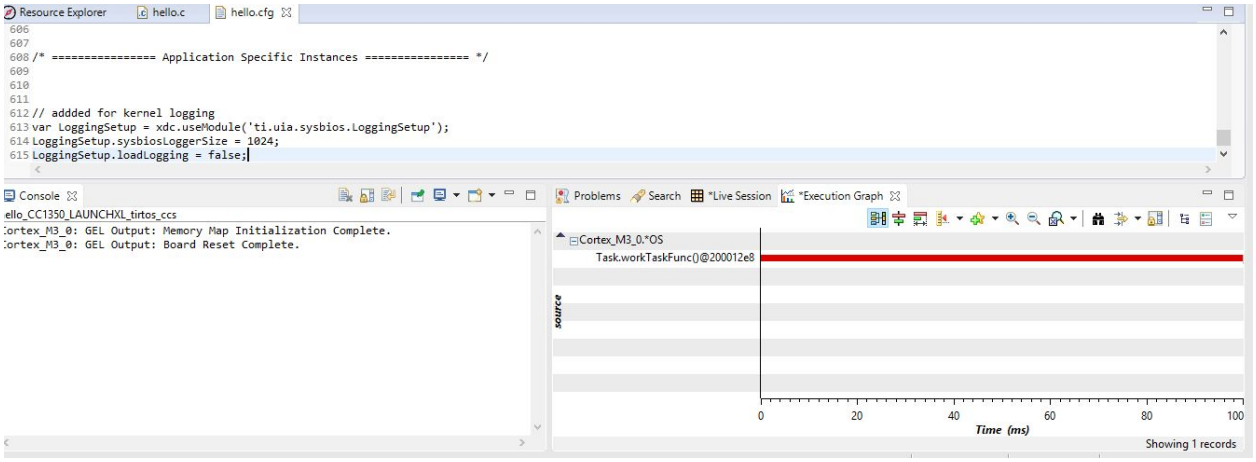
Lab 02:

Youtube Link:

Modified Code: modified hello.cfg to allow logging.
lines modified

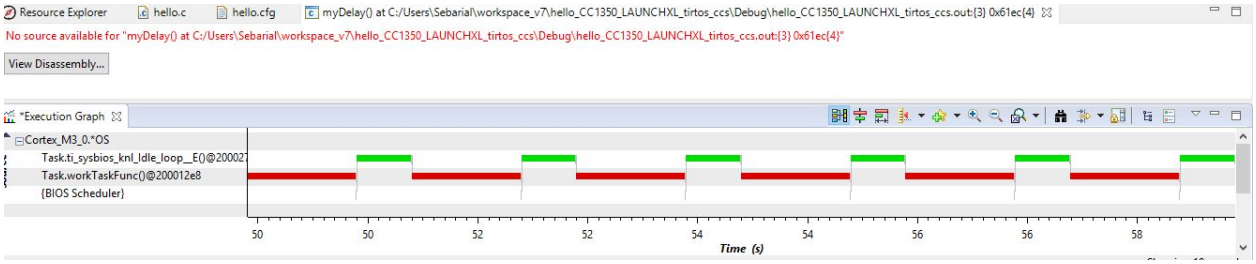
```
BIOS.logsEnabled = true          //uncommented
//BIOS.logsEnabled = false;      //commented

/* commented out
var ROM = xdc.useModule('ti.sysbios.rom.ROM');
if (Program.cpu.deviceName.match(/CC2640R2F/)) {
    ROM.romName = ROM.CC2640R2F;
}
else if (Program.cpu.deviceName.match(/CC26.2/)) {
    ROM.romName = ROM.CC26X2;
}
else if (Program.cpu.deviceName.match(/CC13.2/)) {
    ROM.romName = ROM.CC13X2;
}
else if (Program.cpu.deviceName.match(/CC26/)) {
    ROM.romName = ROM.CC2650;
}
else if (Program.cpu.deviceName.match(/CC13/)) {
    ROM.romName = ROM.CC1350;
}
*/
var LoggingSetup = xdc.useModule('ti.uia.sysbios.LoggingSetup');
LoggingSetup.sysbiosLoggerSize = 1024;
LoggingSetup.loadLogging = false;
```



Task 03:

```
Void workTaskFunc(UArg arg0, UArg arg1)
{
    while (1) {
        /* Do work */
        doWork();
        /* Wait a while, because doWork should be a periodic thing, not continuous.*/
        //myDelay(24000000);  // commented out
        Task_sleep((1000/Clock_tickPeriod)*500); // added in
    }
}
```



Task 04:

```
#include <xdc/std.h>
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Task.h>
#include <ti/drivers/GPIO.h>
#include <ti/sysbios/knl/Clock.h>

/* Example/Board Header files */
#include "Board.h"

void myDelay(int count);
```

```
/* Could be anything, like computing primes */
#define FakeBlockingSlowWork() myDelay(12000000)
#define FakeBlockingFastWork() myDelay(2000000)
Task_Struct workTask;
Task_Struct urgentWorkTask;

/* Make sure we have nice 8-byte alignment on the stack to avoid wasting memory */
#pragma DATA_ALIGN(workTaskStack, 8)
#pragma DATA_ALIGN(urgentWorkTaskStack, 8)
#define STACKSIZE 1024
static uint8_t workTaskStack[STACKSIZE];
static uint8_t urgentWorkTaskStack[STACKSIZE];


void doUrgentWork(void)
{
   GPIO_write(Board_GPIO_LED1, Board_GPIO_LED_OFF);
   FakeBlockingFastWork(); /* Pretend to do something useful but time-consuming */
   GPIO_write(Board_GPIO_LED1, Board_GPIO_LED_ON);
}

void doWork(void)
{
   GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_OFF);
   FakeBlockingSlowWork(); /* Pretend to do something useful but time-consuming */
   GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_ON);
}

Void workTaskFunc(UArg arg0, UArg arg1)
{
   while (1) {
      /* Do work */
      doWork();
      /* Wait a while, because doWork should be a periodic thing, not continuous.*/
      //myDelay(24000000);
      Task_sleep((1000/Clock_tickPeriod)*500);
   }
}
//  function added for urgent work
Void urgentWorkTaskFunc(UArg arg0, UArg arg1)
{
   while (1) {
      /* Do work */
      doUrgentWork();
      /* Wait a while, because doWork should be a periodic thing, not continuous.*/
      //myDelay(24000000);
      Task_sleep((1000/Clock_tickPeriod)*50);
   }
}
/*
* ======== main ========
*
*/
int main(void)
{
   Board_initGeneral();
   GPIO_init();
/* Set up the led task */
   Task_Params workTaskParams;
   Task_Params_init(&workTaskParams);
   workTaskParams.stackSize = STACKSIZE;
   workTaskParams.priority = 2;
   workTaskParams.stack = &workTaskStack;
   Task_construct(&workTask, workTaskFunc, &workTaskParams, NULL);

   //set the urgentWorkTask priority(highest for greatest priority)
```

```
    workTaskParams.priority = 3;
    workTaskParams.stack = &urgentWorkTaskStack;
    Task_construct(&urgentWorkTask, urgentWorkTaskFunc, &workTaskParams, NULL);
    /* Start kernel. */
    BIOS_start();

    return (0);
   }
/*
* ======== myDelay ========
* Assembly function to delay. Decrements the count until it is zero
* The exact duration depends on the processor speed.
*/
__asm(" .sect \".text:myDelay\"\n"
" .clink\n"
" .thumbfunc myDelay\n"
" .thumb\n"
" .global myDelay\n"
"myDelay:\n"
" subs r0, #1\n"
" bne.n myDelay\n"
" bx lr\n");
```