Abenezer Namaga
CPE 403
LAB 01

Date Submitted: 11/13/2018
Github root directory: https://github.com/Ayertena/CC1350-LABs

Task 01: LED Flash

```
/* TI-RTOS Header files */
#include <xdc/std.h>
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Task.h>
#include <ti/drivers/GPIO.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;
/* Make sure we have nice 8-byte alignment on the stack to avoid wasting memory */
#pragma DATA_ALIGN(workTaskStack, 8)
#define STACKSIZE 1024
static uint8_t workTaskStack[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);
    }
}
/*
* ======== 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);
    /* 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");
```

Task02: Debugging tools

```
*
 * Enable logs in the BIOS library.
 *
 * Pick one:
 *  - true (default)
 *      Enables logs for debugging purposes.
 *  - false
 *      Disables logging for reduced code footprint and improved runtime
 *      performance.
 *
 *  When using BIOS in ROM:
 *      This option must be set to false.
 */
BIOS.logsEnabled = true;
//BIOS.logsEnabled = false;

.

.

.

var LoggingSetup = xdc.useModule('ti.uia.sysbios.LoggingSetup');
LoggingSetup.sysbiosLoggerSize = 1024;
LoggingSetup.loadLogging = false;
```

Task 03: Sleep Well

```
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(500*(1000/Clock_tickPeriod));
    }
}
```

Task 04: Execute 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(50*(1000/Clock_tickPeriod));
    }
}
```

Task 05: Change Priority

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

.

.

int main(void)
{
       .
       .
    workTaskParams.priority = 3;
    workTaskParams.stack = &urgentWorkTaskStack;
    Task_construct(&urgentWorkTask, urgentWorkTaskFunc, &workTaskParams, NULL);}
```