Nathan Hanuscin

Github root directory: https://github.com/nhanuscin/HappyFunStuff

Date Submitted: 11/18/2019

Task 01:

Youtube Link: https://www.youtube.com/watch?v=d0STdqgm4dk

Modified Schematic (if applicable): N/A

Modified Code:
```c
/* 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);
    }
}
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```c
/*
 * ======== 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");
```

Task 02:

Youtube Link: N/A same as task 1

Modified Schematic (if applicable): N/A

Modified Code:

```c
/* 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 ========
*
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```c
*/
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");
```

Task 03:

Youtube Link: N/A same as task 1
Modified Schematic (if applicable): N/A

Modified Code:
```c
/* TI-RTOS Header files */
#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;
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```c
/* 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)
{
    int numTicks = Clock_tickPeriod;
    while (1)
    {
        /* Do work */
        doWork();
        /* Wait a while, because doWork should be a periodic thing, not continuous.*/
        //myDelay(24000000);
        Task_sleep(numTicks*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);
    /* 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.
 */
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
__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");
```

Task 04:

Youtube Link: https://www.youtube.com/watch?v=cpQ5woVzQTU

Modified Schematic (if applicable): N/A

Modified Code:
```
/* TI-RTOS Header files */
#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)
#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)
{
    //int numTicks = Clock_tickPeriod;
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```c
    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));
    }
}

Void urgentWorkTaskFunc(UArg arg0, UArg arg1)
{
    //int numTicks = Clock_tickPeriod;
    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));
    }
}
/*
 * ======== 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);

    workTaskParams.priority = 3;
    workTaskParams.stack = &urgentWorkTaskStack;
    Task_construct(&urgentWorkTask, urgentWorkTaskFunc, &workTaskParams, NULL);
    /* Start kernel. */
    BIOS_start();
    return (0);
}
__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");
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.