**Date Submitted: 11-15-2018**

**Task 00: Execute provided code (No submission required)**

**------------------------------------------------------------------------------------**

**Task 01: Task that toggles LED**

Youtube Link: <https://youtu.be/4LguQUOtmRc>

**Modified Code:**

//task01: task that toggles LED

/\* 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"

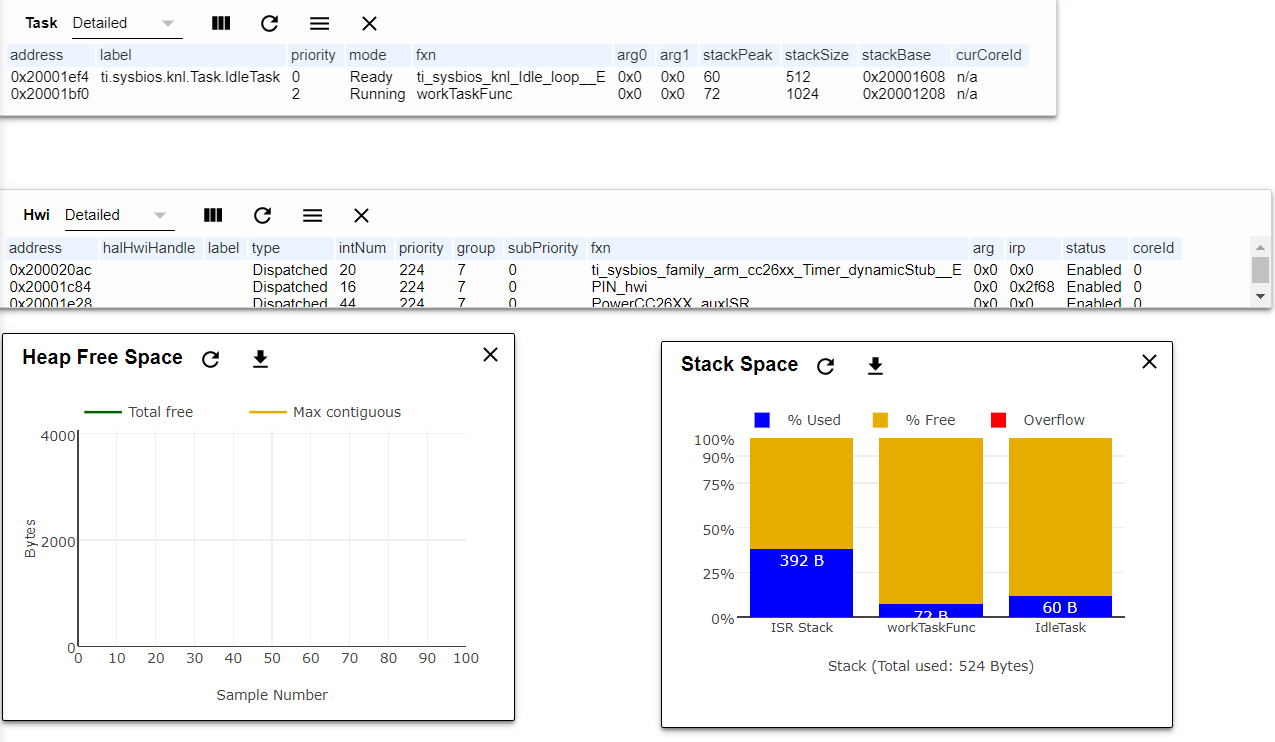
" bne.n myDelay\n"

" bx lr\n");

**------------------------------------------------------------------------------------**

**Task 02: Use Runtime Object View to show stack peak, heap space, and stack information that the program uses to run its tasks. Use execution graph to get a graphical representation of tasks running with respect to system time.**

Youtube Link: <https://youtu.be/4zCnpAgvwaM>



Runtime Object view stats.



Execution Analysis graph.

**Modified Code:**

hello.cfg

.

.

.

\*

\* When using BIOS in ROM:

\* This option must be set to false.

\*/

BIOS.logsEnabled = true;

//BIOS.logsEnabled = false;

/\* ================ ROM configuration ================ \*/

/\*

\* To use BIOS in flash, comment out the code block below.

\*/

/\*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;

}

\*/

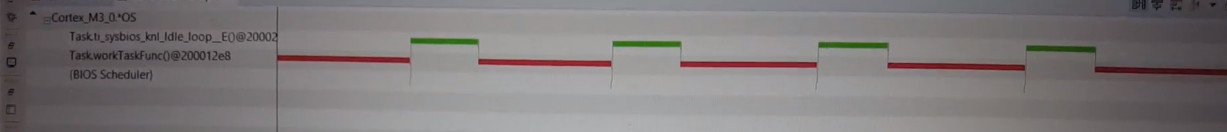
.

.

.

**------------------------------------------------------------------------------------**

**Task 03: Replace the use of myDelay to sleep with Task\_sleep and use it to sleep for 500ms. Show Execution Analysis Graph.**



Youtube Link: <https://youtu.be/3VF_l1MzWz0>

**Modified Code:**

Task02.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**;**

/\* 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.\*/

Task\_sleep**(**500 **\*** **(**1000 **/** Clock\_tickPeriod**));** //delay 500ms

**}**

**}**

/\*

\* ======== 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"

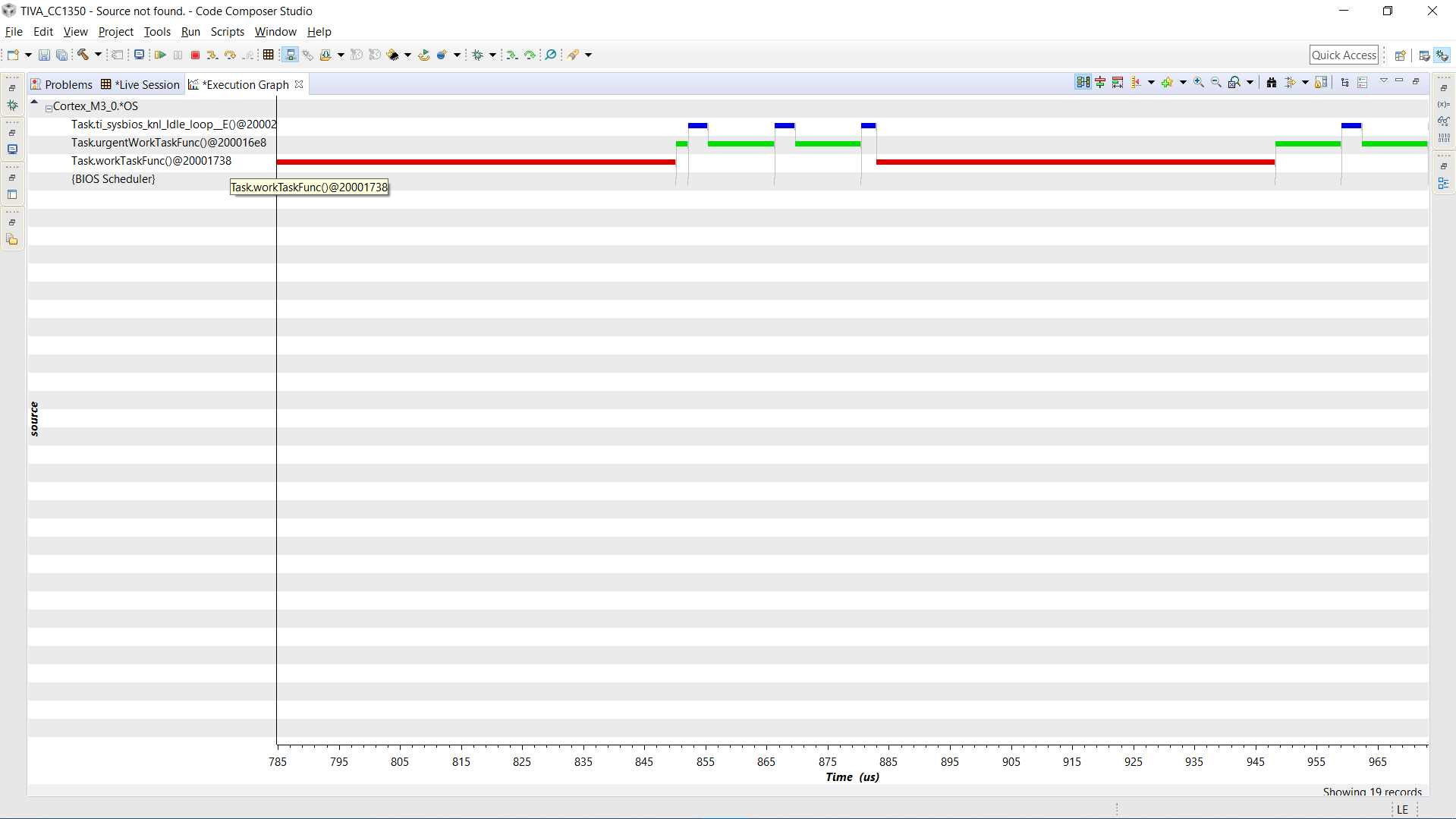
" subs r0, #1\n"

" bne.n myDelay\n"

" bx lr\n");

**------------------------------------------------------------------------------------**

**Task 04: Set up a new task called urgent work which has higher priority than previous task called work task. Show the Execution Analysis Graph**



Execution Analysis Graph showing that UrgentTask interrupts lower priority workTask.

Youtube Link: <https://youtu.be/6dLklaQxVqo>

**Modified Code:**

Task04.c

/\* TI-RTOS Header files \*/

#include <xdc/std.h>

#include <ti/sysbios/BIOS.h>

#include <ti/sysbios/knl/Task.h>

#include <ti/sysbios/knl/Clock.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**;**

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**)**

**{**

**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)

{

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 = 1;

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");