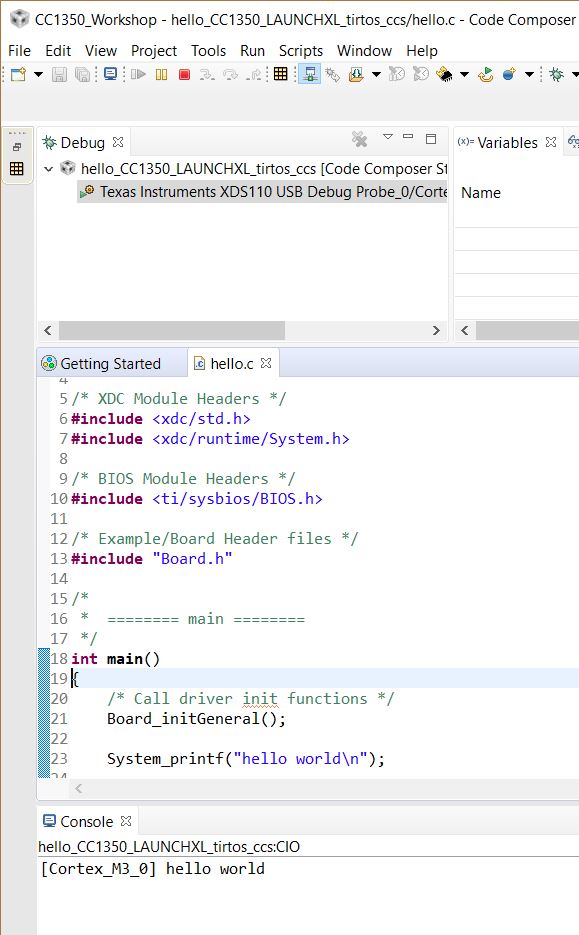
**Date Submitted: 11/13/18**

**Task 00: Execute provided code**

**Youtube Link: No Video Required**

The image below shows the output of “hello world” in the console by running the hello.c file in the hello project.

****

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

**Task 01:**

Youtube Link: <https://youtu.be/VaGRHzYelVA>

**Modified Schematic (if applicable): N/A**

**Modified Code:**

**// Insert code here**

/\*

\* Copyright (c) 2015-2017, Texas Instruments Incorporated

\* All rights reserved.

\*

\* Redistribution and use in source and binary forms, with or without

\* modification, are permitted provided that the following conditions

\* are met:

\*

\* \* Redistributions of source code must retain the above copyright

\* notice, this list of conditions and the following disclaimer.

\*

\* \* Redistributions in binary form must reproduce the above copyright

\* notice, this list of conditions and the following disclaimer in the

\* documentation and/or other materials provided with the distribution.

\*

\* \* Neither the name of Texas Instruments Incorporated nor the names of

\* its contributors may be used to endorse or promote products derived

\* from this software without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

\* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,

\* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR

\* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR

\* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,

\* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

\* PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;

\* OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,

\* WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR

\* OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,

\* EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*/

/\*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){

doWork(); //do work

myDelay(24000000); //wait a while, because doWork should be a periodic thing, not continuous

}

}

**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

//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"

" .global myDelay\n"

"myDelay:\n"

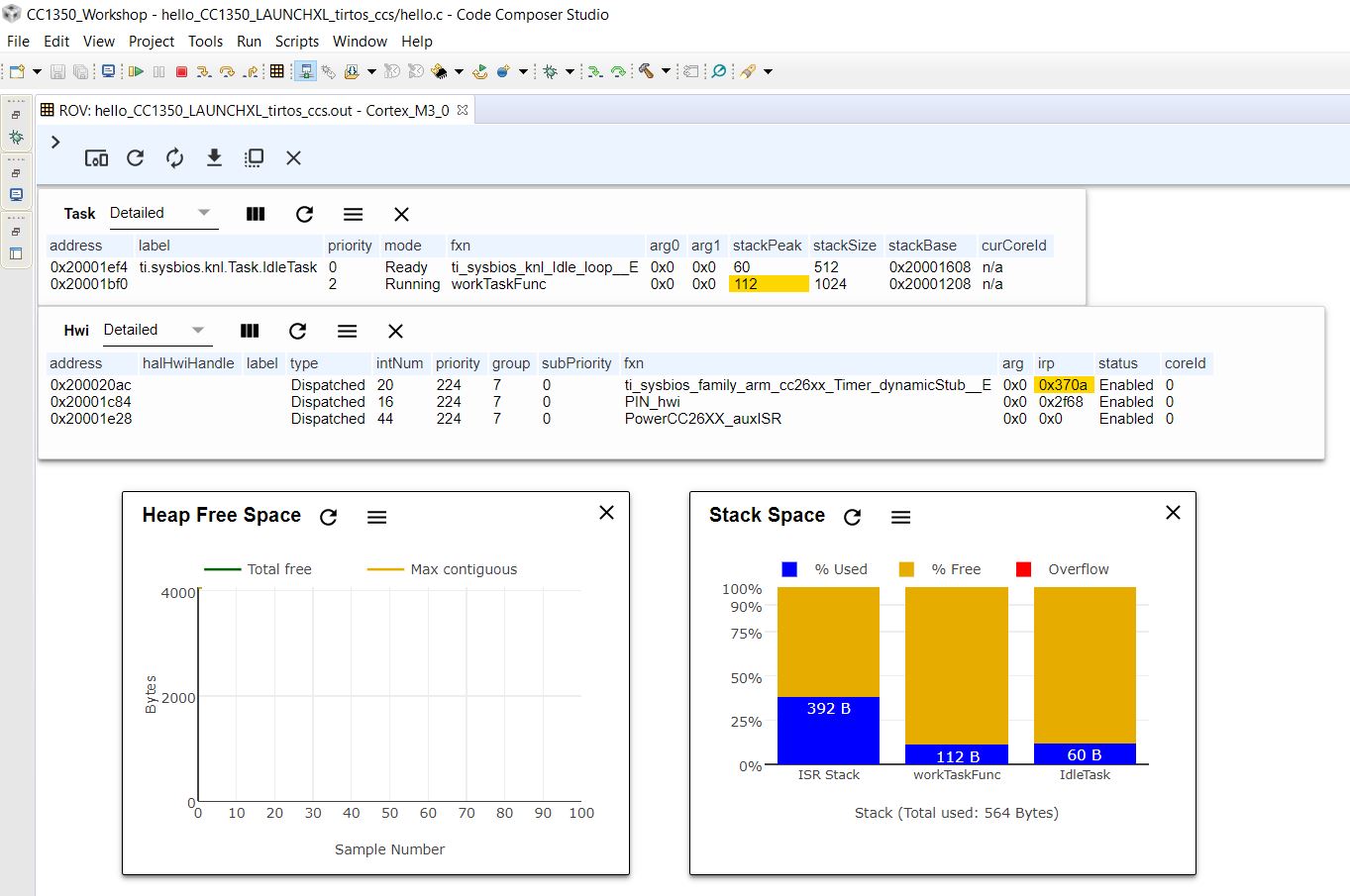
" subs r0, #1\n"

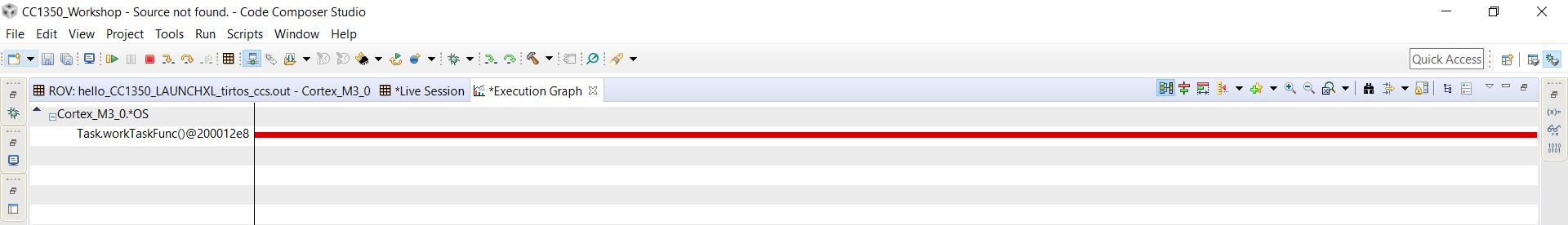
" bne.n myDelay\n"

" bx lr\n");

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

**Task 02:**

****

****

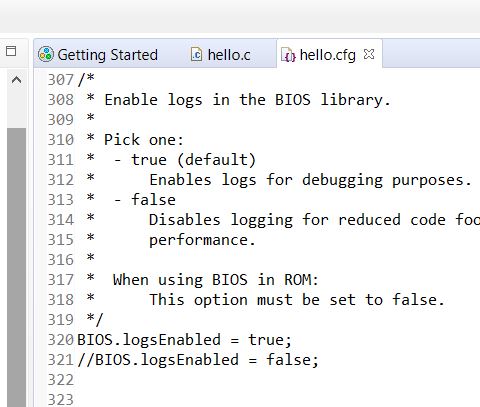
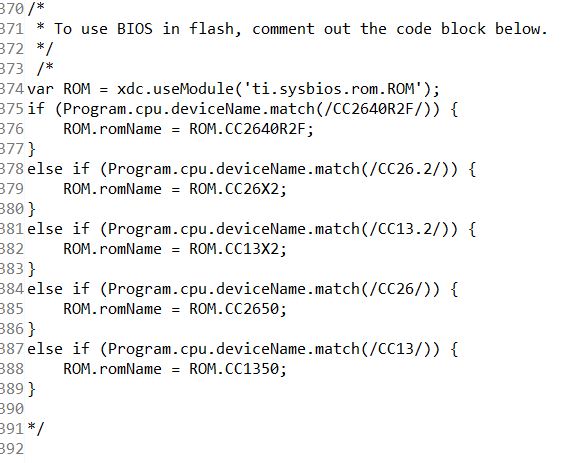
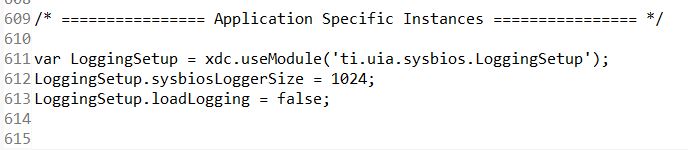
Youtube Link: <https://youtu.be/qWqQjzkqxIo>

**Modified Schematic (if applicable): N/A**

**Modified Code:**

The hello.c file is kept the same as in Task 1, only changes were asked to be made in the cfg file in task 2.

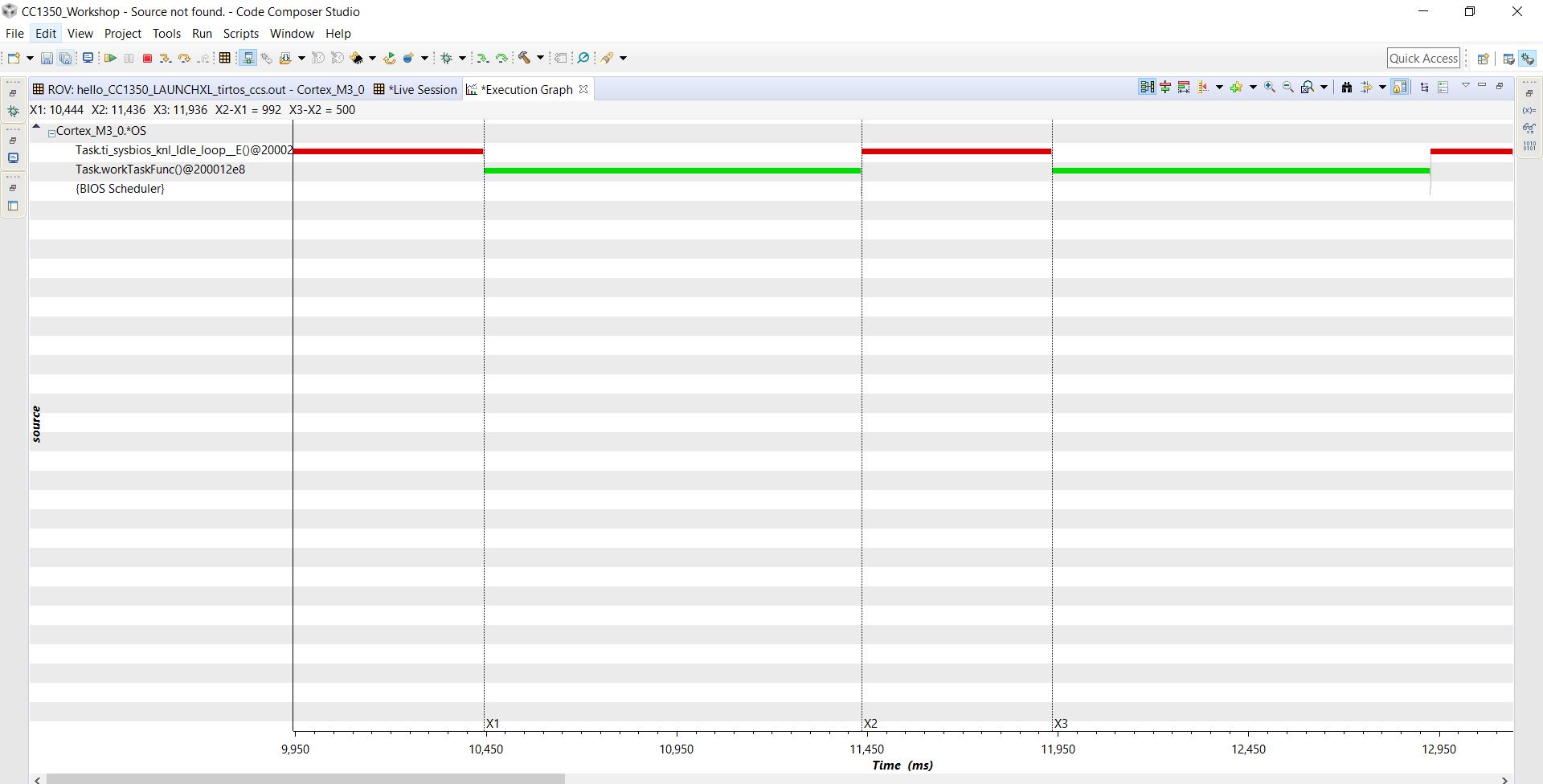
The three images below show the modifications made in the cfg file.

1. 
2. 
3. 

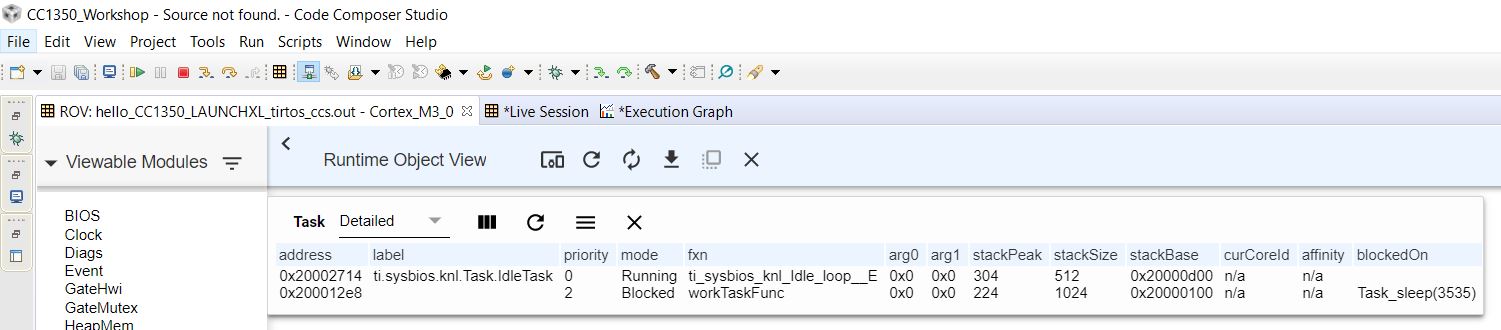
**------------------------------------------------------------------------------------**

**Task 03:**

The image below shows the execution graph after completing task 3.



The image below shows the task details where we can see that idle was finally given a chance to run. (also shown on the execution graph above)



Youtube Link: <https://youtu.be/tH_R-QCKtpQ>

**Modified Schematic (if applicable): N/A**

**Modified Code:**

**// Insert code here**

/\*

\* Copyright (c) 2015-2017, Texas Instruments Incorporated

\* All rights reserved.

\*

\* Redistribution and use in source and binary forms, with or without

\* modification, are permitted provided that the following conditions

\* are met:

\*

\* \* Redistributions of source code must retain the above copyright

\* notice, this list of conditions and the following disclaimer.

\*

\* \* Redistributions in binary form must reproduce the above copyright

\* notice, this list of conditions and the following disclaimer in the

\* documentation and/or other materials provided with the distribution.

\*

\* \* Neither the name of Texas Instruments Incorporated nor the names of

\* its contributors may be used to endorse or promote products derived

\* from this software without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

\* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,

\* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR

\* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR

\* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,

\* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

\* PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;

\* OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,

\* WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR

\* OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,

\* EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*/

/\*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"

**#include** <ti/sysbios/knl/Clock.h>

**void** **myDelay**(**int** count);

//could be anything, like computing primes

**#define** FakeBlockingSlowWork() myDelay(12000000)

**#define** FakeBlockingFastWork() myDelay(2000000)

//Make sure we have nice 8-byte alignment on the stack to avoid wasting memory

Task\_Struct workTask;

**#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){

doWork(); // do work

//myDelay(24000000);

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

}

}

**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

//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"

" .global myDelay\n"

"myDelay:\n"

" subs r0, #1\n"

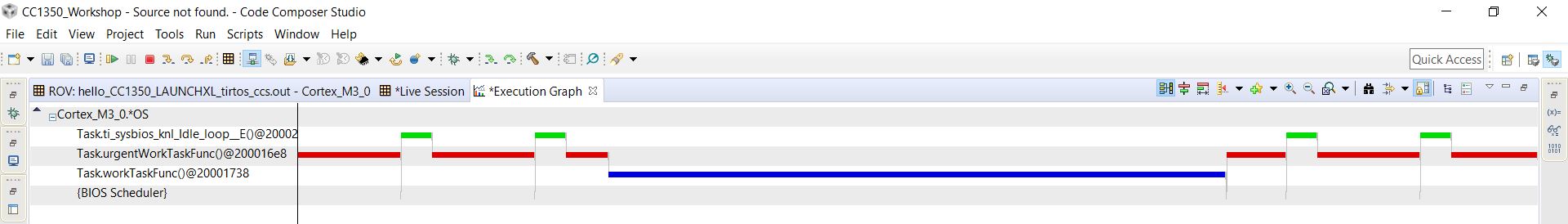
" bne.n myDelay\n"

" bx lr\n");

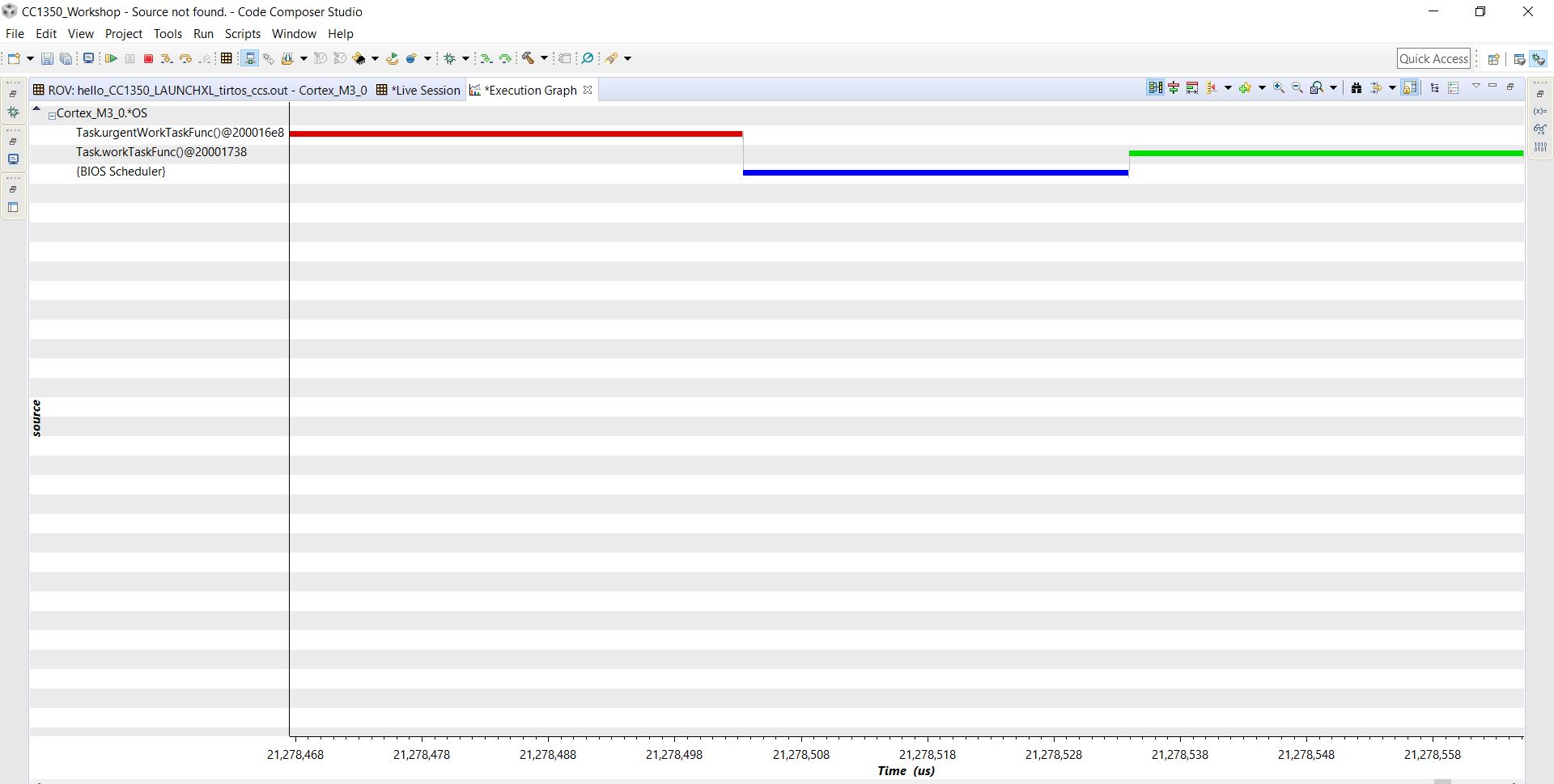
**------------------------------------------------------------------------------------**

**Task 04:**

The image below is the execution graph when the second workTaskParams.priority = 1. The video for this shows LED1 flashing at the undesired rate.

****

The image below is the execution graph when the second workTaskParams.priority = 3. The video for priority 3 also shows LED1 flashing at the desired rate.



Youtube Links:

Priority 1: <https://youtu.be/XCzYWjZbhJ0>

Priority 3: <https://youtu.be/YbgIWrmIqcA>

**Modified Schematic (if applicable): N/A**

**Modified Code:**

**// Insert code here**

/\*

\* Copyright (c) 2015-2017, Texas Instruments Incorporated

\* All rights reserved.

\*

\* Redistribution and use in source and binary forms, with or without

\* modification, are permitted provided that the following conditions

\* are met:

\*

\* \* Redistributions of source code must retain the above copyright

\* notice, this list of conditions and the following disclaimer.

\*

\* \* Redistributions in binary form must reproduce the above copyright

\* notice, this list of conditions and the following disclaimer in the

\* documentation and/or other materials provided with the distribution.

\*

\* \* Neither the name of Texas Instruments Incorporated nor the names of

\* its contributors may be used to endorse or promote products derived

\* from this software without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

\* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,

\* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR

\* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR

\* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,

\* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

\* PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;

\* OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,

\* WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR

\* OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,

\* EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*/

//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"

**#include** <ti/sysbios/knl/Clock.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 a 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){

doWork(); //do work

//myDelay(24000000);

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

}

}

**void** **urgentWorkTaskFunc**(UArg arg0, UArg arg1)

{

**while**(1)

{

doUrgentWork(); //do urgent work

Task\_sleep(50 \* (1000 / Clock\_tickPeriod));

}

}

**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"

" .global myDelay\n"

"myDelay:\n"

" subs r0, #1\n"

" bne.n myDelay\n"

" bx lr\n");

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