**Date Submitted: 11/14/2019**

**Task 00: Execute provided code**

**Youtube Link:** https://www.youtube.com/watch?v=X\_6Bf5Zf\_uI

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

**All Tasks (Overall Code):**

Youtube Links:

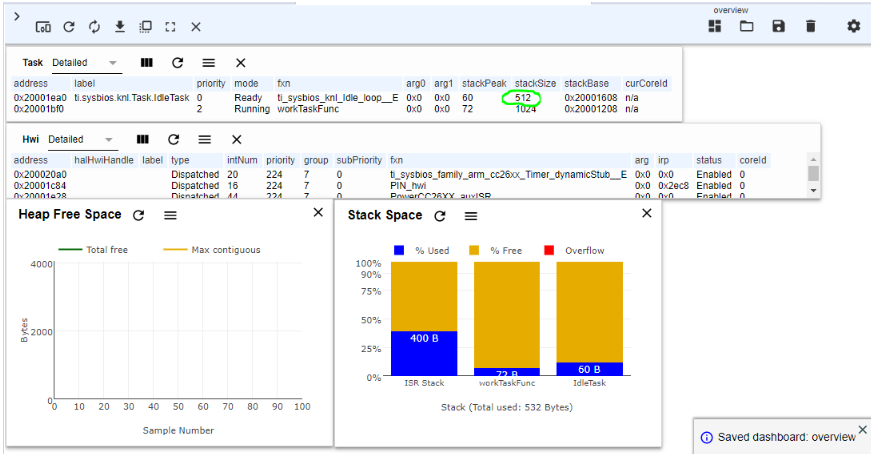
Lab1-Task1: https://www.youtube.com/watch?v=X\_6Bf5Zf\_uI

Lab1-Task2: https://www.youtube.com/watch?v=2uyWGUvzSQI

Lab1-Task3: https://www.youtube.com/watch?v=oFAiOXIq0NU

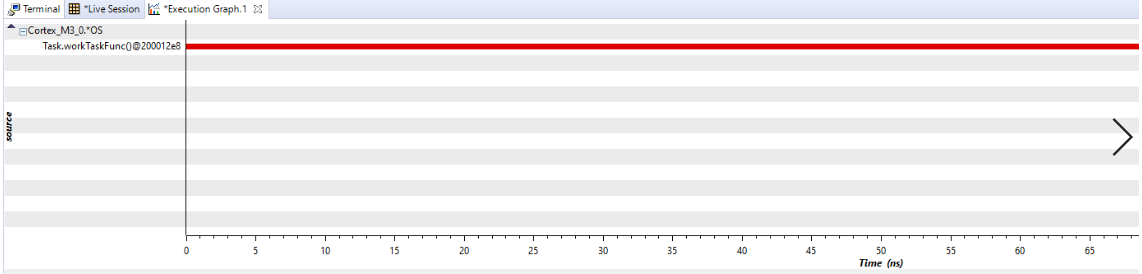
**Modified Schematic (if applicable):**

**ROV**

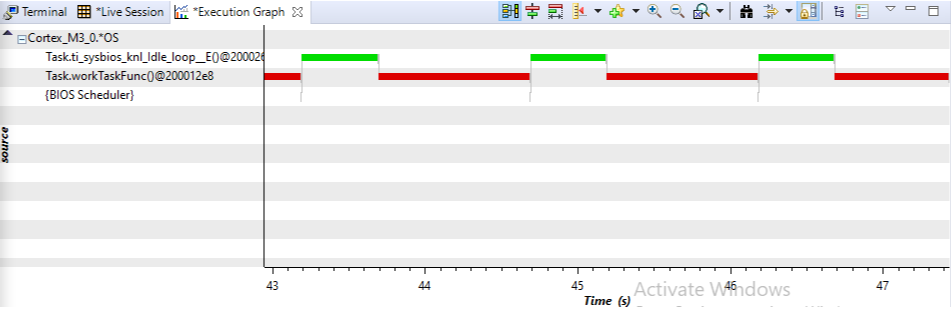


**Execution graphs:**

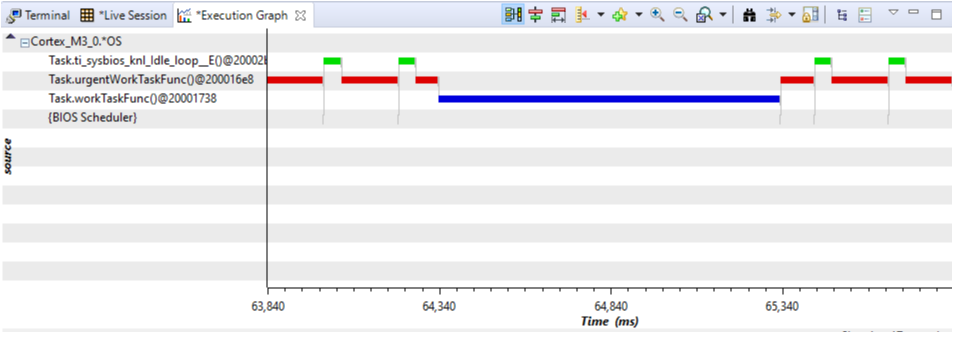
**Only 1 Task**



**Task function + idle task**



**workTaskFunc + urgentWorkTaskFunc**



**Modified Code:**

**// Insert code here**

/\* 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> //use of 'Clock\_tickPeriod'

/\* 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)

//information regarding each task - how the kernel will act for each task

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];

//doUrgentWork will utilize LED 1 (not LED 0)

**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 afrg1)

{

**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)); //sleep for 500ms, does not waste CPU cycles

}

}

**void** **urgentWorkTaskFunc**(UArg arg0, UArg afrg1)

{

**while** (1) {

/\* Do 'urgent' work \*/

doUrgentWork();

/\* Wait a while, because doWork should be a periodic thing, not continuous.\*/

//myDelay(24000000);

Task\_sleep(50 \* (1000/Clock\_tickPeriod)); //sleep for 50ms, does not waste CPU cycles

}

}

/\*

\* ======== 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;

//create the task..

Task\_construct(&workTask, workTaskFunc, &workTaskParams, NULL);

//set up new task (urgent task)

workTaskParams.priority = 3; //priority previously was 1

workTaskParams.stack = &urgentWorkTaskStack;

//create the new task..

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");