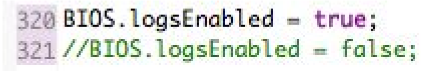
Joseph Sharp Halpin
CpE 403 Section 1001
Date Submitted: 11/12/2018

Youtube Link: https://youtu.be/BkgxqN_qY8E

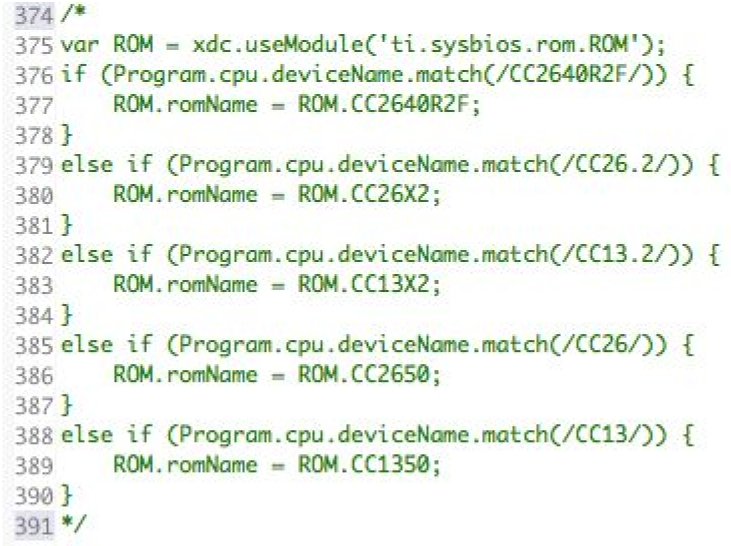
Edited Code:

cfg file 1:



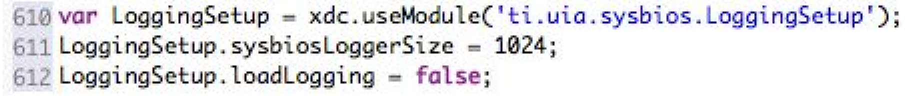
```
BIOS.logsEnabled = true;
//BIOS.logsEnabled = false;
```

cfg file 2:



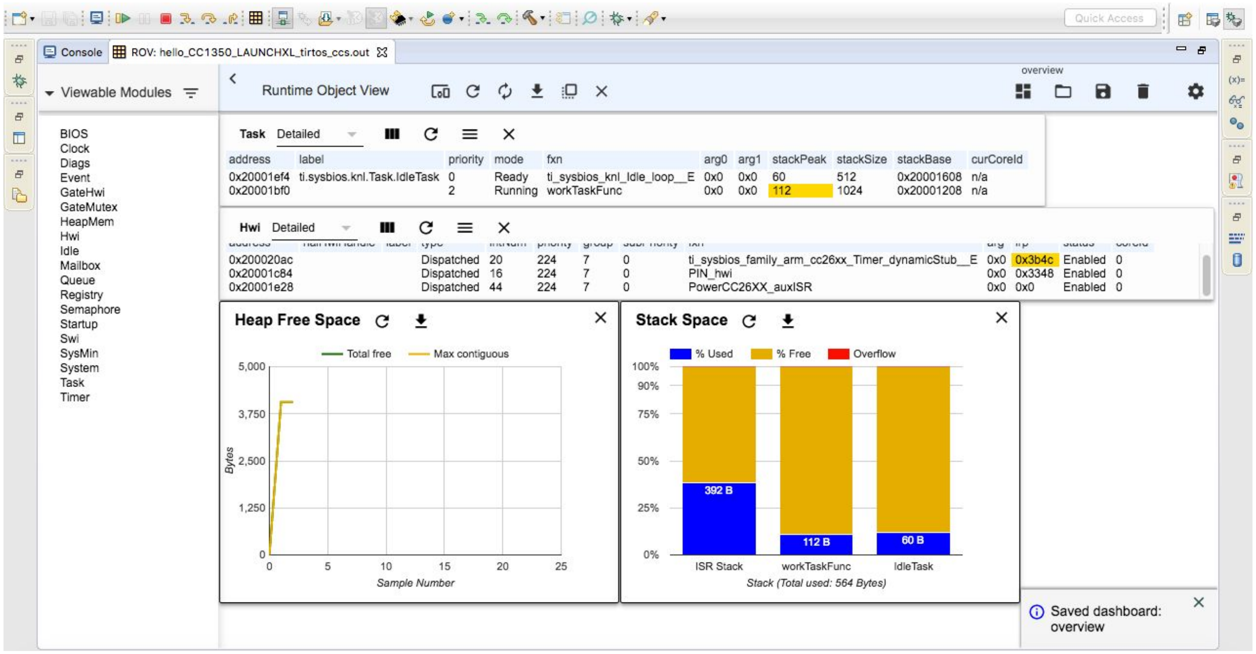
```
/*
var ROM = xdc.useModule('ti.sysbios.rom.ROM');
if (Program.cpu.deviceName.match(/CC2640R2F/)) {
    ROM.romName = ROM.CC2640R2F;
}
else if (Program.cpu.deviceName.match(/CC26.2/)) {
    ROM.romName = ROM.CC26X2;
}
else if (Program.cpu.deviceName.match(/CC13.2/)) {
    ROM.romName = ROM.CC13X2;
}
else if (Program.cpu.deviceName.match(/CC26/)) {
    ROM.romName = ROM.CC2650;
}
else if (Program.cpu.deviceName.match(/CC13/)) {
    ROM.romName = ROM.CC1350;
}
*/
```

cfg file 3:



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

Stack Usage:

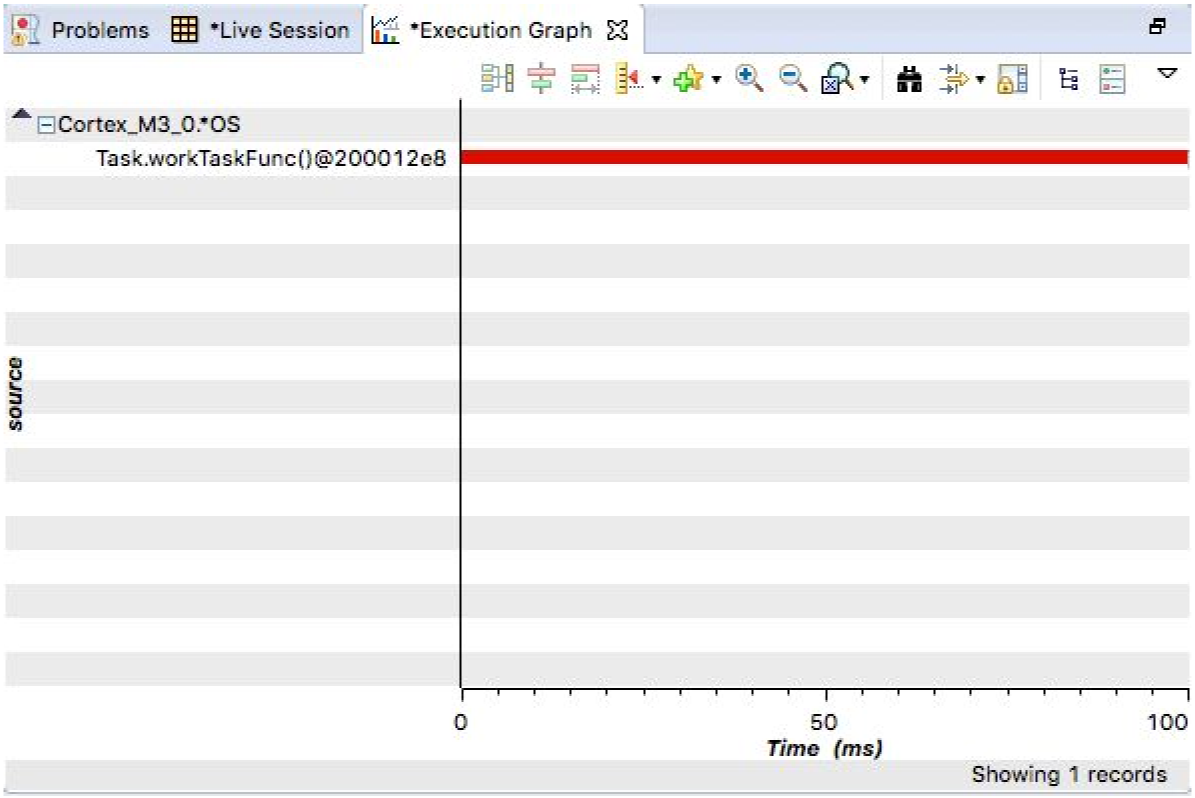


address	label	priority	mode	fxn	arg0	arg1	stackPeak	stackSize	stackBase	curCoreId
0x20001ef4	ti.sysbios.knl.Task.IdleTask	0	Ready	ti_sysbios_knl_Idle_loop__E	0x0	0x0	60	512	0x20001608	n/a
0x20001bf0		2	Running	workTaskFunc	0x0	0x0	112	1024	0x20001208	n/a

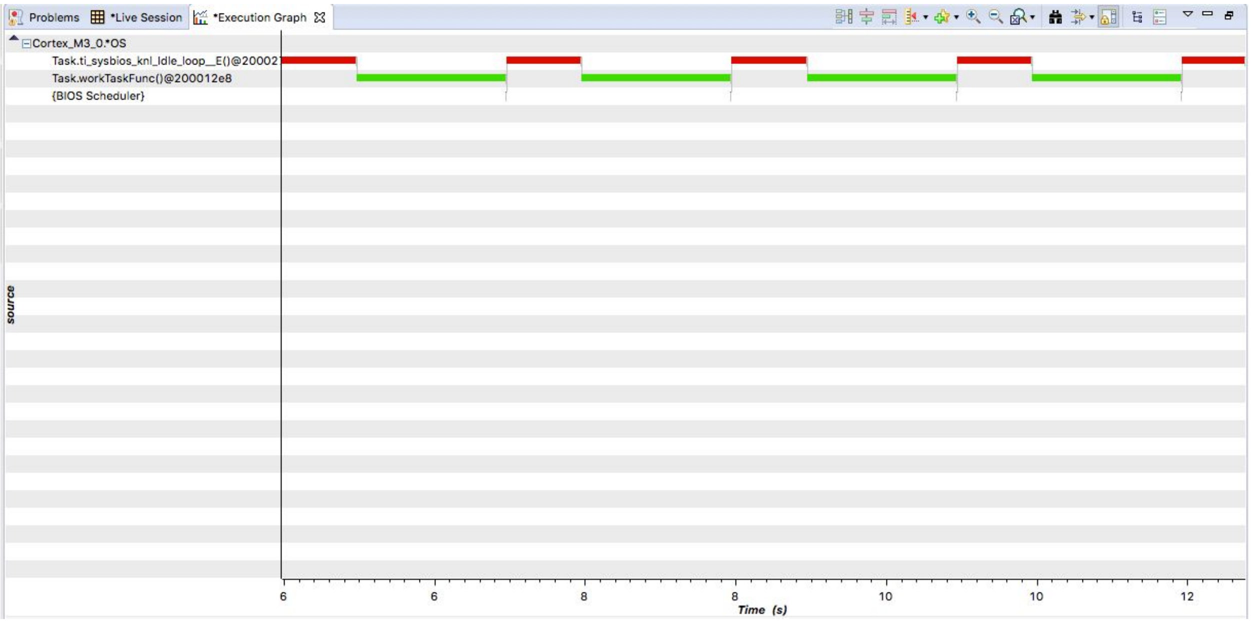
address	halHwiHandle	label	type	intNum	priority	group	subPriority	fxn	arg	irp	status	coreId
0x200020ac			Dispatched	20	224	7	0	ti_sysbios_family_arm_cc26xx_Timer_dynamicStub__E	0x0	0x3b4c	Enabled	0
0x20001c84			Dispatched	16	224	7	0	PIN_hwi	0x0	0x3348	Enabled	0
0x20001e28			Dispatched	44	224	7	0	PowerCC26XX_auxISR	0x0	0x0	Enabled	0

Execution Graphs:

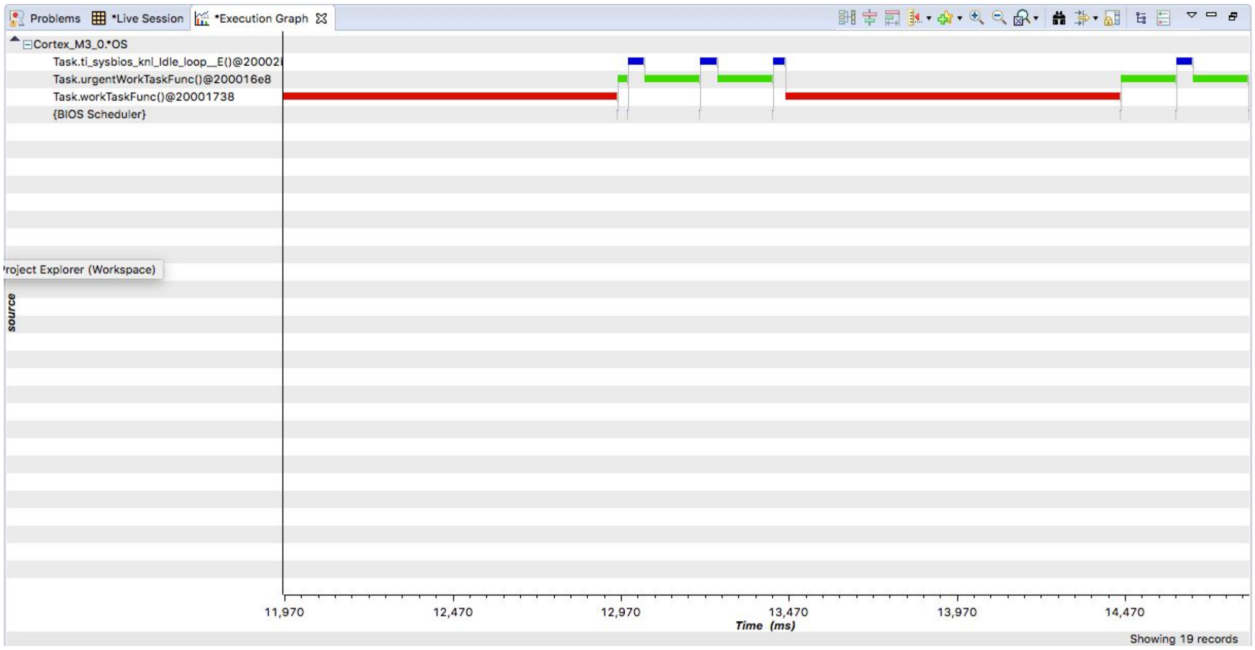
Graph 1:



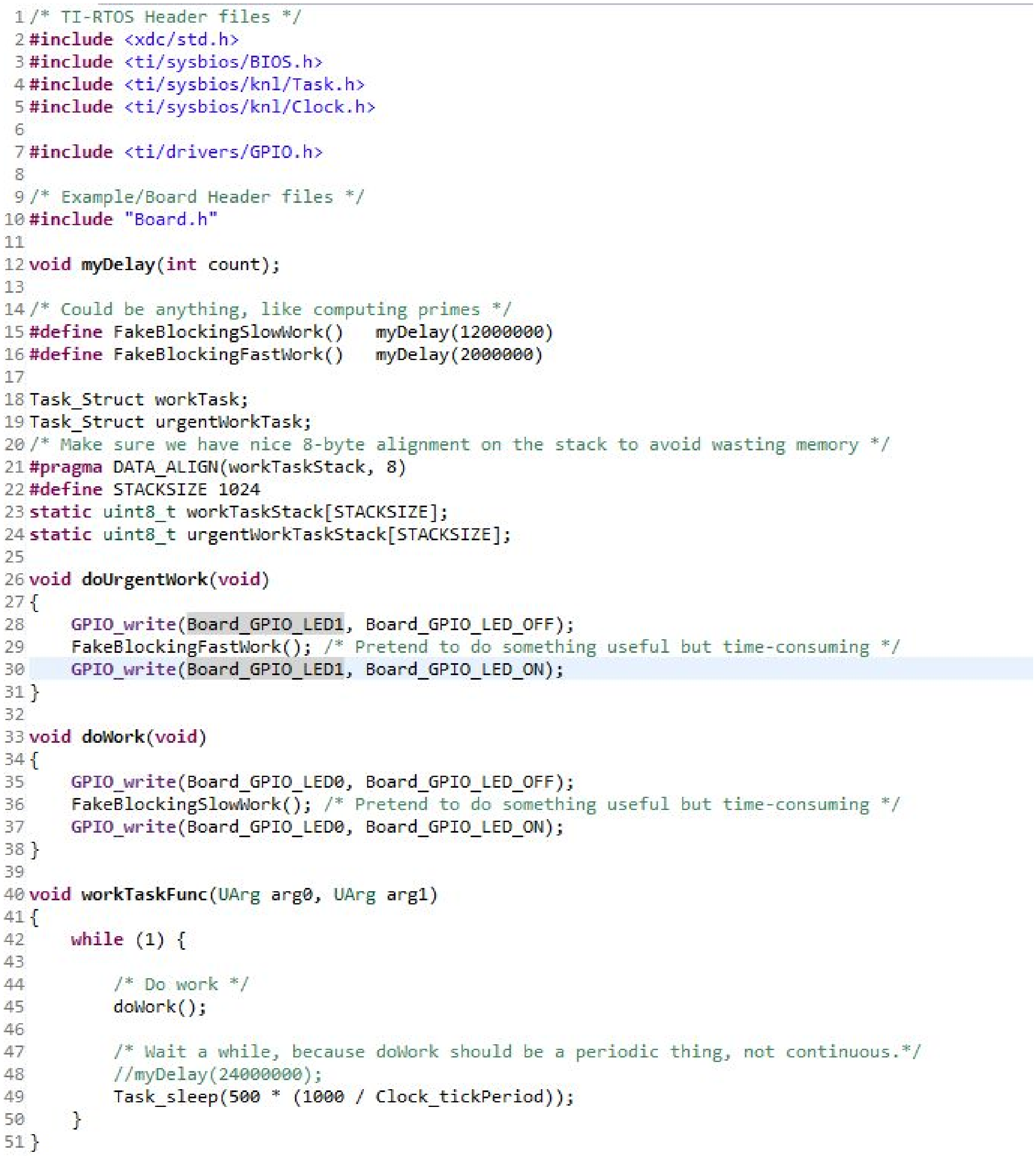
Graph 2:



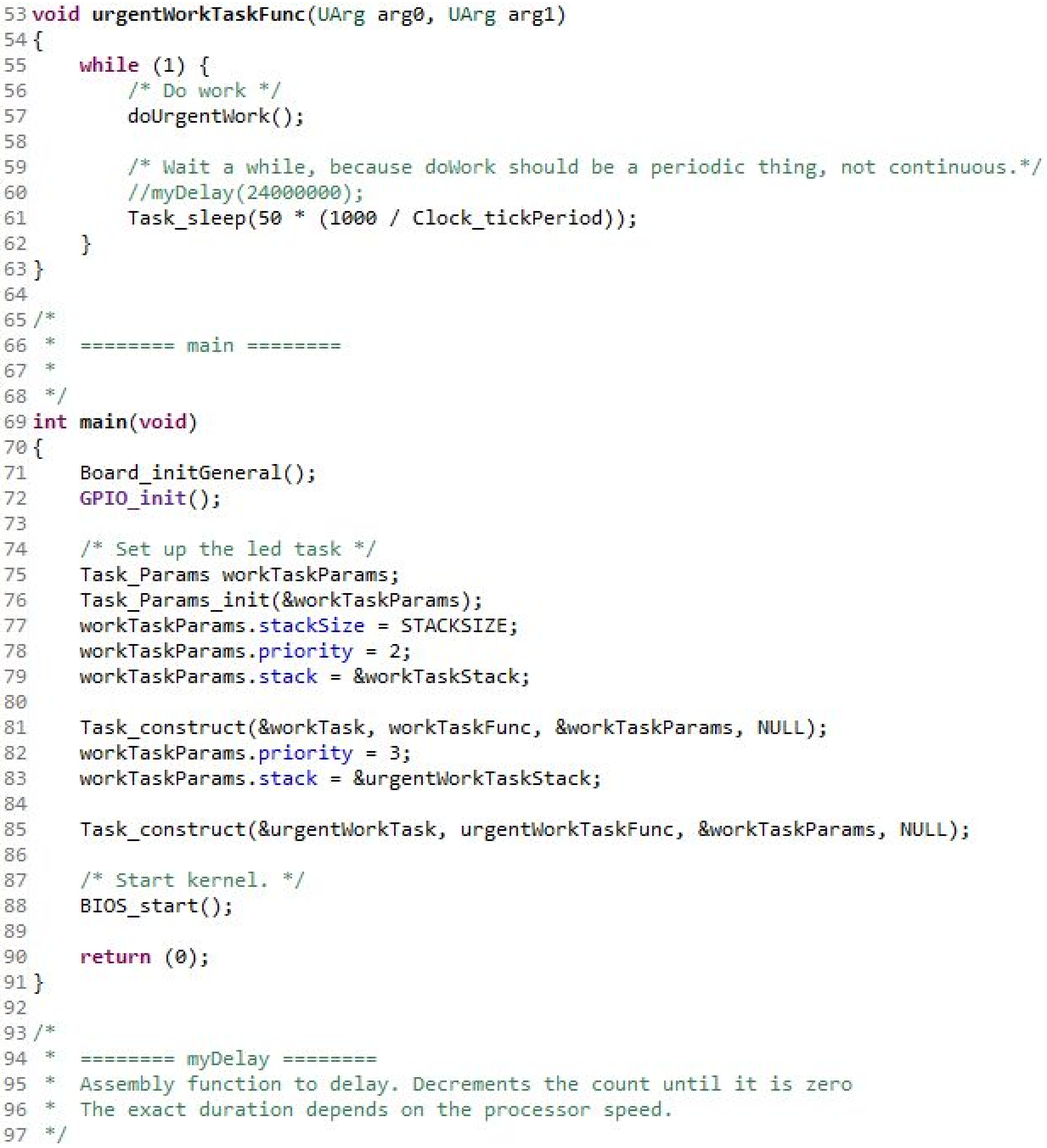
Graph 3:



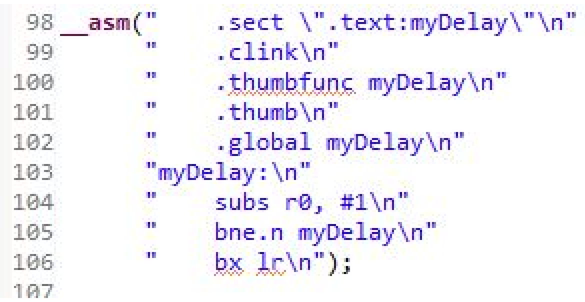
Final Code:



```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));
    }
}
```



```c
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 = 3;
    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.
 */
```



```c
__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");

```