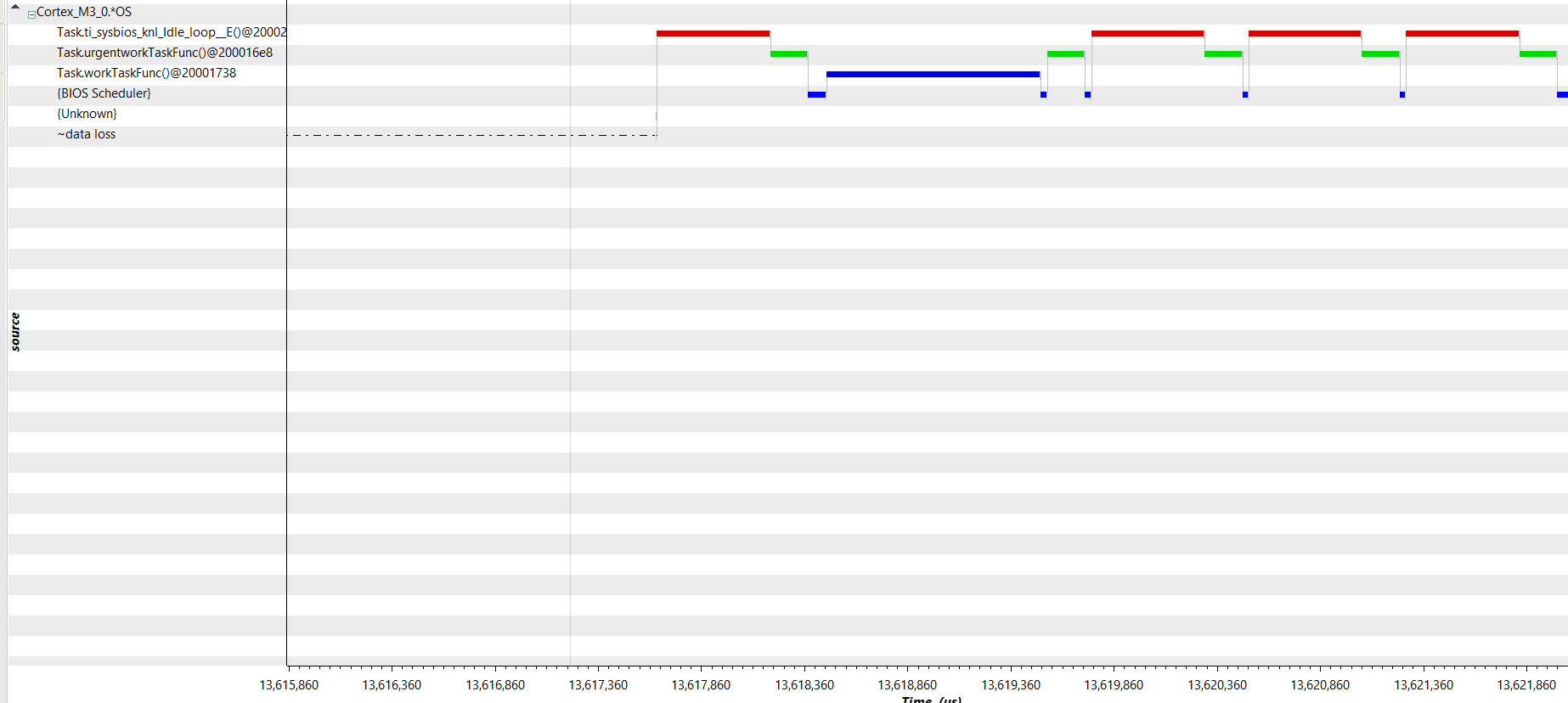
Adrian Ruiz

CPE 403

CC1350 Lab 1

Image of Execution Graph:



Objective: Get a better understanding of the resources and see how RTOS is handles tasks. Urgent work task grabs the system more often because it has higher priority. The normal work task function gets the system longer because its longer delay.

Code:

/\* 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(12000)

#define FakeBlockingFastWork() myDelay(2000)

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**];**

Task\_Struct urgentworkTask**;**

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

#pragma DATA\_ALIGN(urgentworkTaskStack, 8)

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**();**

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

**}**

**}**

Void urgentworkTaskFunc**(**UArg arg0**,** UArg arg1**)**

**{**

**while** **(**1**)** **{**

/\* Do work \*/

doUrgentWork**();**

Task\_sleep**(**50 **\*** **(**10 **/** 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 = 3;

workTaskParams.stack = &workTaskStack;

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

/\* Set up the led task \*/

Task\_Params urgentworkTaskParams;

Task\_Params\_init(&urgentworkTaskParams);

urgentworkTaskParams.stackSize = STACKSIZE;

urgentworkTaskParams.priority = 2;

urgentworkTaskParams.stack = &urgentworkTaskStack;

Task\_construct(&urgentworkTask, urgentworkTaskFunc, &urgentworkTaskParams, 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");