**Date Submitted: 12/4/19**

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

**Task 01:**

Youtube Link: <https://youtu.be/5vN4CUxyeCA>

Modified Code:

/\* 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:**

**Youtube link: https://youtu.be/5vN4CUxyeCA**

**A screenshot of a cell phone

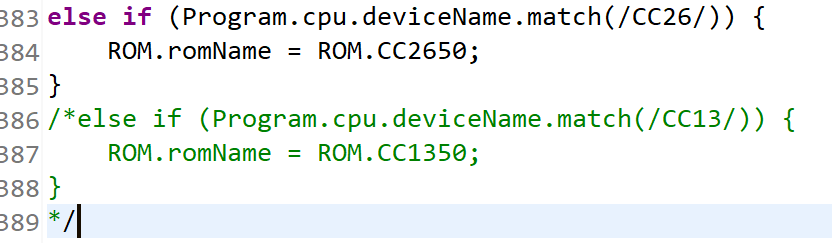
Description automatically generated**

**Enable logs:**

**A picture containing sky, indoor, orange

Description automatically generated**

**Enable store to flash:**

****

**Added at bottom of config file:**

**A screenshot of a cell phone

Description automatically generated**

**A picture containing screenshot

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

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

**Task 03:**

Youtube Link: https://youtu.be/lv6XkVPCMO4

Modified Code:

/\* 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.\*/

//myDelay(24000000);

Task\_sleep(500 \* (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);

/\* 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");

A screenshot of a cell phone

Description automatically generated

A picture containing screenshot

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

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

**Task 04:**

Youtube Link: https://youtu.be/60jNbsg7I1A

Modified 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(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");

A screenshot of a cell phone

Description automatically generated

A picture containing screenshot

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

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

**Task 05: CHANGING PRIORITY [FROM TASK 4]**

Youtube Link: https://youtu.be/pLeWJ3xMKJ8

**Modified 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(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 = 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.

\*/

**\_\_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");

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**