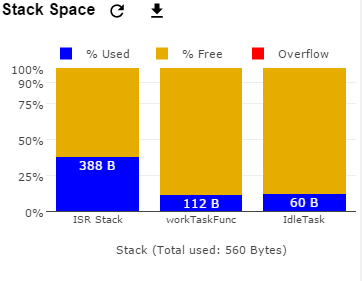
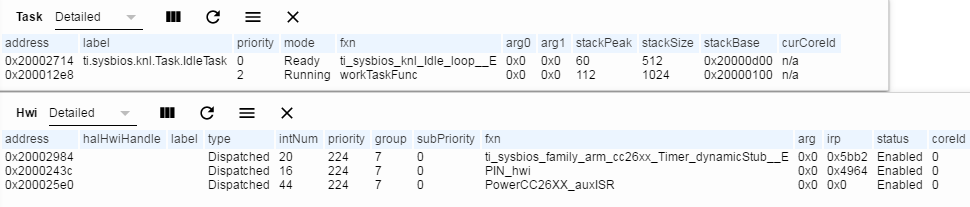
TASK 2:





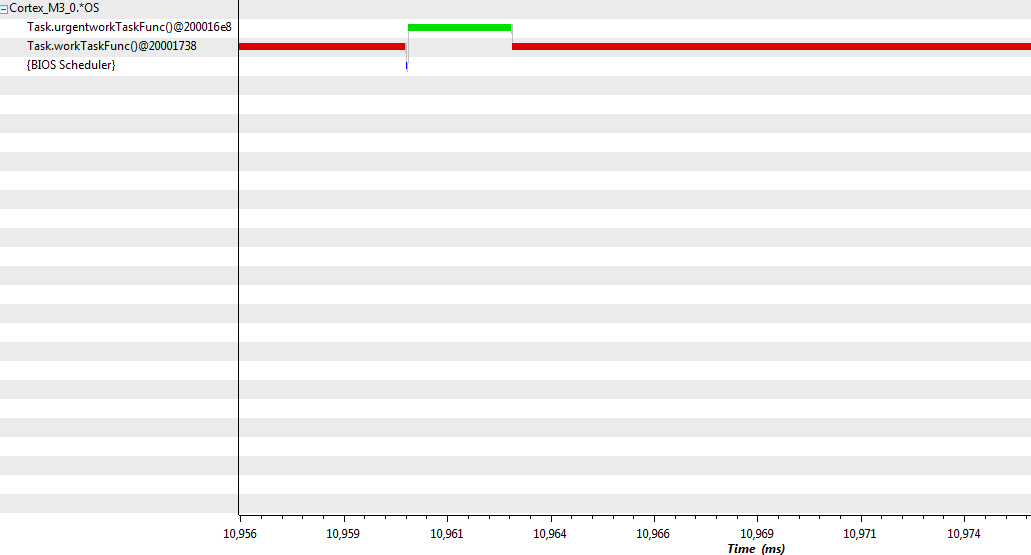
TASK 3:



TASK 3.1:



TASK 4:



TASK 5:

/\* TI-RTOS Header files \*/

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

#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];

Task\_Struct urgentworkTask;

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

#pragma DATA\_ALIGN(urgentworkTaskStack, 8)

#define STACKSIZE 1024

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(50 \* (50 / 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.\*/

//Task\_sleep(24000000);

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

}

}

/\*

\* ======== main ========

\*

\*/

int main(void)

{

Board\_initGeneral();

GPIO\_init();

/\* Set up the led task \*/

Task\_Params urgentworkTaskParams;

Task\_Params\_init(&urgentworkTaskParams);

urgentworkTaskParams.stackSize = STACKSIZE;

urgentworkTaskParams.priority = 3;

urgentworkTaskParams.stack = &urgentworkTaskStack;

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

/\* 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");