Youtube Link: https://youtu.be/f7YE24AT9O4

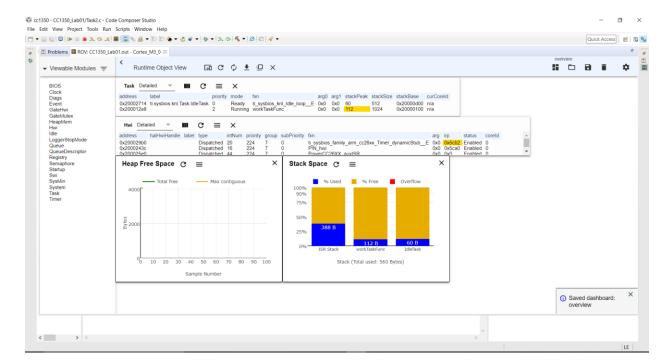
```
#include <xdc/std.h>
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Task.h>
#include <ti/drivers/GPIO.h>
/* Example/Board Header files */
#include "Board.h"
void myDelay(int count);
/* Could be anything, like computing primes */
#define FakeBlockingSlowWork() myDelay(12000000)
#define FakeBlockingFastWork() 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"
" .\underline{\text{thumbfunc}} myDelay\n"
" .thumb\n"
" .global myDelay\n"
"myDelay:\n"
" subs r0, #1\n"
" bne.n myDelay\n"
" bx lr\n");
```

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

```
/* 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
```

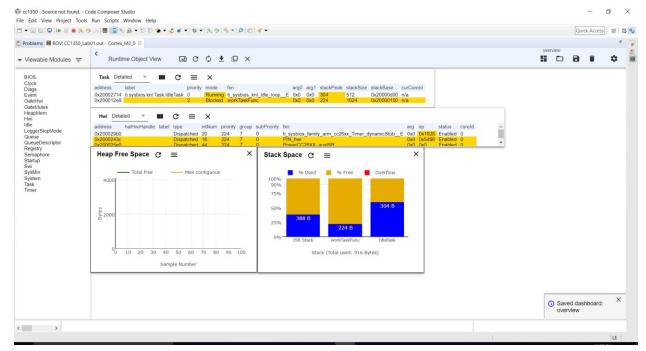
```
static uint8 t workTaskStack[STACKSIZE];
void doUrgentWork(void)
{
 GPIO_write(Board_GPIO_LED1, Board_GPIO_LED_OFF);
FakeBlockingFastWork(); /* Pretend to do something useful but time-consuming */
GPIO write(Board GPIO LED1, Board GPIO LED ON);
}
void doWork(void)
{
GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_OFF);
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");
```

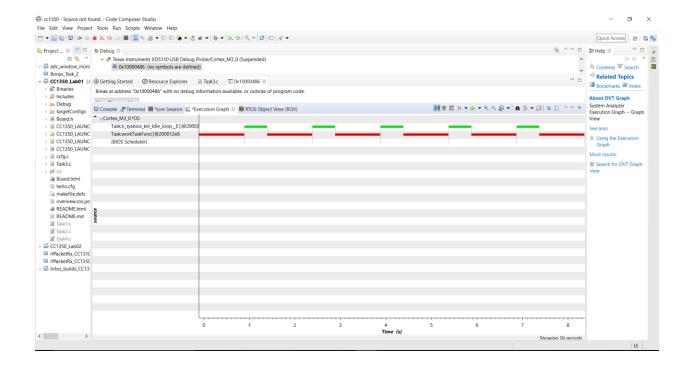


Youtube Link: https://youtu.be/lyOhdatvnAk

```
/* 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.*/
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"
 " .<u>thumbfunc</u> myDelay\n"
 " .thumb\n"
 " .global myDelay\n"
 "myDelay:\n"
 " subs r0, #1\n"
 " bne.n myDelay\n"
 " <u>bx</u> <u>lr</u>\n");
```



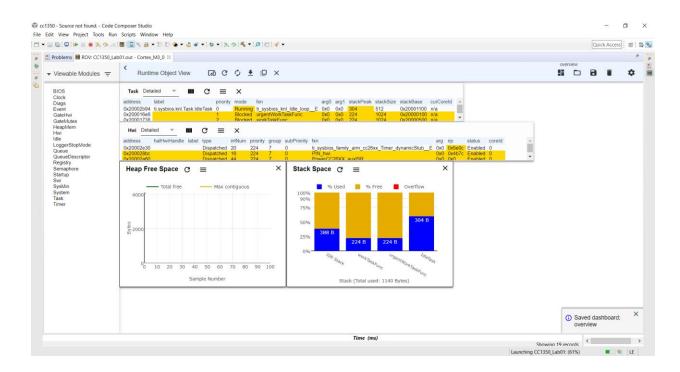


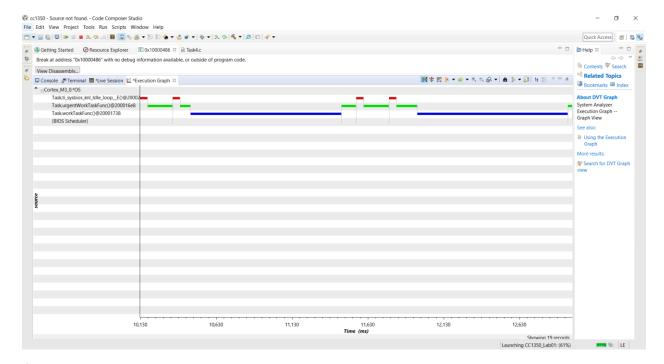
Youtube Link: https://youtu.be/3MWHZATSe0M

```
/* 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;
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 urgentWorkTaskFunc(UArg arg0, UArg arg1)
while (1) {
/* Do work */
doUrgentWork();
/* Wait a while, because doWork should be a periodic thing, not continuous.*/
Task_sleep(50 * (1000 / Clock_tickPeriod));
void workTaskFunc(UArg arg0, UArg arg1)
while (1) {
/* Do work */
doWork();
/* Wait a while, because doWork should be a periodic thing, not continuous.*/
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);
 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");
```





cc1350 - CC1350_Lab01/hello.cfg - Code Composer Studio File Edit View Navigate Project Run Scripts Window Help 🔓 Project Explorer 🛭 🕒 😘 🔻 🗖 🗖 Getting Started 🕜 Resource Exp... 🙆 scif.c 📵 scif_framewo... 📵 Task2.c 📧 myDelay() a... 📦 hello.cfg 🗵 ု 😘 - -587 * 588 * Pick one: 589 * - true (default) 590 * This option 1 > 🛎 adc_window_monitor_launchpad_tirtos Bonus_Task_2 ∨ GC1350_Lab01 [Active - Debug] This option loads test string into the .const for easier debugging. > 🗱 Binaries 591 * - false > 🛍 Includes This option reduces the .const footprint. 593 */ > 📂 Debug 594//Text.isLoaded = true; 595 Text.isLoaded = false; > b targetConfigs > 🖹 Board.h 596 597 > @ CC1350_LAUNCHXL_fxns.c > D CC1350_LAUNCHXL_TIRTOS.cmd 598 599/* ======== Types configuration === > @ CC1350 LAUNCHXL.c. 599/* =========== // 600 var Types = xdc.useModule('xdc.runtime.Types'); 601/* 602 * This module defines basic constants and types used throughout the 603 * xdc.runtime package. 604 */ > h CC1350_LAUNCHXL.h > 🖻 ccfg.c > 🖻 Task2.c > 📂 src Roard.html 605 hello.cfg 606 makefile.defs overview.rov.json 608 README.html ------ Application Specific Instances ------README.md 610 var LoggingSetup = xdc.useModule('ti.uia.sysbios.LoggingSetup'); ☑ Task1.c 612 ☑ Task3.c ☑ Task4.c 614 615 rfPacketRx_CC1310_LAUNCHXL_tirtos_ccs rfPacketRx_CC1350_LAUNCHXL_RX 🗎 🔝 🕑 🖻 ▼ 😁 ▼ 🗆 🖺 Problems 🖓 Advice 🛭 🚜 Terminal ■ Console ⊠ ## tirtos_builds_CC1350_LAUNCHXL_release_ccs CC1350_Lab01 3 items Cortex_M3_0: GEL Output: Memory Map Initialization Compl Description Resource Cortex_M3_0: GEL Output: Board Reset Complete. > i Optimization Advice (3 items) <

610:1

Writable

Insert