main.c

```

#include <stdint.h>
#include <stdbool.h>
#include "main.h"
#include "drivers/pinout.h"
#include "utils/uartstdio.h"


// TivaWare includes
#include "driverlib/sysctl.h"
#include "driverlib/debug.h"
#include "driverlib/rom_map.h"
#include "driverlib/rom.h"
#include "driverlib/timer.h"
#include "driverlib/inc/hw_memmap.h"
#include "driverlib/inc/hw_ints.h"

// FreeRTOS includes
#include "FreeRTOSConfig.h"
#include "FreeRTOS.h"
#include <timers.h>
#include <semphr.h>
#include "task.h"
#include "queue.h"
#include "limits.h"


#define FIB_LIMIT_FOR_32_BIT 47
#define TIME_TO_RUN 240 //ms

SemaphoreHandle_t task1SyncSemaphore;
TaskHandle_t Task1_handle;
double Hz = 100;
uint32_t ulPeriod;



void Timer0Isr(void)
{
    TickType_t xCurrentTick = xTaskGetTickCount();
    BaseType_t xHigherPriorityTaskWoken = pdFALSE;
    ROM_TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);  // Clear the timer interrupt

        xTaskNotifyFromISR(Task1_handle, xCurrentTick, eSetValueWithOverwrite, &
xHigherPriorityTaskWoken);
        portYIELD_FROM_ISR( xHigherPriorityTaskWoken );

}



// Process 1
void xTask1(void * pvParameters)
```

```
62  {
63
64
65      const TickType_t xMaxBlockTime = pdMS_TO_TICKS( 5000 );
66      BaseType_t xResult;
67      uint32_t ulNotifiedValue;
68
69      while(1){
70
71          xResult = xTaskNotifyWait( pdFALSE,
72                          /* Don't clear bits on entry. */
73                          ULONG_MAX,
74                          /* Clear all bits on exit. */
75                          &ulNotifiedValue, /* Stores the notified value. */
76                          xMaxBlockTime );
77
78          if( xResult == pdPASS )
79          {
80
81              TickType_t xCurrentTick = xTaskGetTickCount();
82              UARTprintf("Task 1 completed at %d ms and Timer interrupt data: %d\n",
    xCurrentTick, ulNotifiedValue);
83
84          }
85      }
86  }
87
88
89
90  // Main function
91  int main(void)
92  {
93      // Initialize system clock to 120 MHz
94      uint32_t output_clock_rate_hz;
95      output_clock_rate_hz = ROM_SysCtlClockFreqSet(
96                                  (SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN |
97                                   SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480),
98                                  SYSTEM_CLOCK);
99      ASSERT(output_clock_rate_hz == SYSTEM_CLOCK);
100
101
102     // Initialize the GPIO pins for the Launchpad
103     PinoutSet(false, false);
104     UARTStdioConfig(0, 230400, SYSTEM_CLOCK);
105
106     ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
107     ROM_TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);   // 32 bits Timer
108     TimerIntRegister(TIMER0_BASE, TIMER_A, Timer0Isr);    // Registering  isr
109
110
111     ulPeriod = (SYSTEM_CLOCK / Hz);
112     ROM_TimerLoadSet(TIMER0_BASE, TIMER_A, ulPeriod -1);
113
114     ROM_TimerEnable(TIMER0_BASE, TIMER_A);
115     ROM_IntEnable(INT_TIMER0A);
116     ROM_TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
```

```
117
118     task1SyncSemaphore = xSemaphoreCreateBinary();
119
120
121     xTaskCreate(xTask1, "Task1", configMINIMAL_STACK_SIZE, NULL, 2, &Task1_handle);
122
123     vTaskStartScheduler();
124
125     return (0);
126 }
127
128
129 /*  ASSERT() Error function
130  *
131  *  failed ASSERTS() from driverlib/debug.h are executed in this function
132  */
133 void __error__(char *pcFilename, uint32_t ui32Line)
134 {
135     // Place a breakpoint here to capture errors until logging routine is finished
136     while (1)
137     {
138     }
139 }
```