ASSIGNMENT 4(Programming)

Name: K.Vishwanath
Course Title: Development of Real-Time System
Date: 23-06-2022

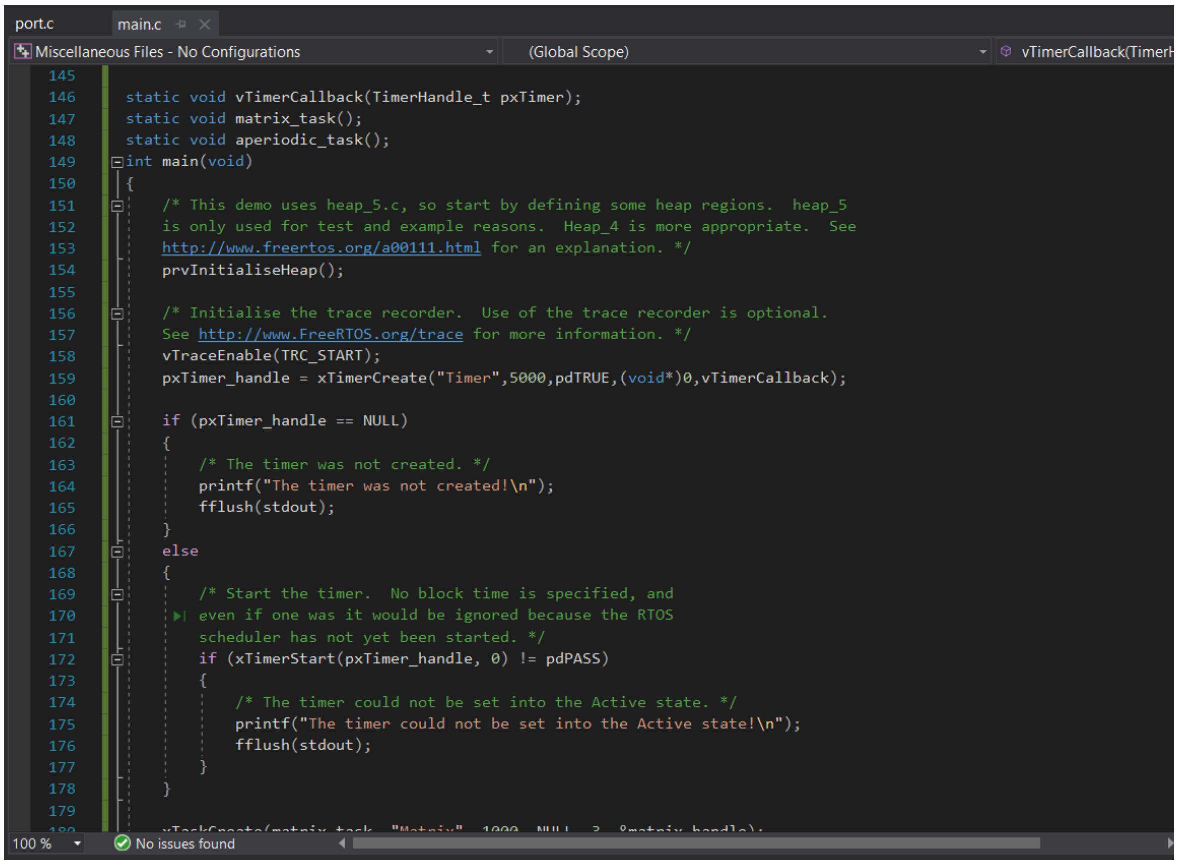
—---

Assignment:
Create a task "matrixtask" containing the functionality given in Assignment 2.
Add a software timer in main() to trigger a software interrupt every 5 seconds.
Define a Timer callback function outside main() with the following functionality:

```
long lExpireCounters = 0;
void vTimerCallback(TimerHandle_t pxTimer)
{
    printf("Timer callback!\n");
    xTaskCreate((pdTASK_CODE)aperiodic_task, (signed char
*)"Aperiodic", configMINIMAL_STACK_SIZE, NULL, 2, &aperiodic_handle);
    long lArrayIndex;
    const long xMaxExpiryCountBeforeStopping = 10;
    configASSERT(pxTimer);
    lExpireCounters += 1;
    /* If the timer has expired 10 times then stop it from running. */
if (lExpireCounters == xMaxExpiryCountBeforeStopping) {
/* Do not use a block time if calling a timer API function from a
timer callback function, as doing so could cause a deadlock! */
     xTimerStop(pxTimer, 0);
    }
}
```

Create an aperiodic task using the following functionality:

```
static void aperiodic_task()
{
    printf("Aperiodic task started!\n");
    fflush(stdout);
    long i;
    for (i = 0; i<1000000000; i++); //Dummy workload
    printf("Aperiodic task done!\n");
    fflush(stdout);
    vTaskDelete(aperiodic_handle);
}
```



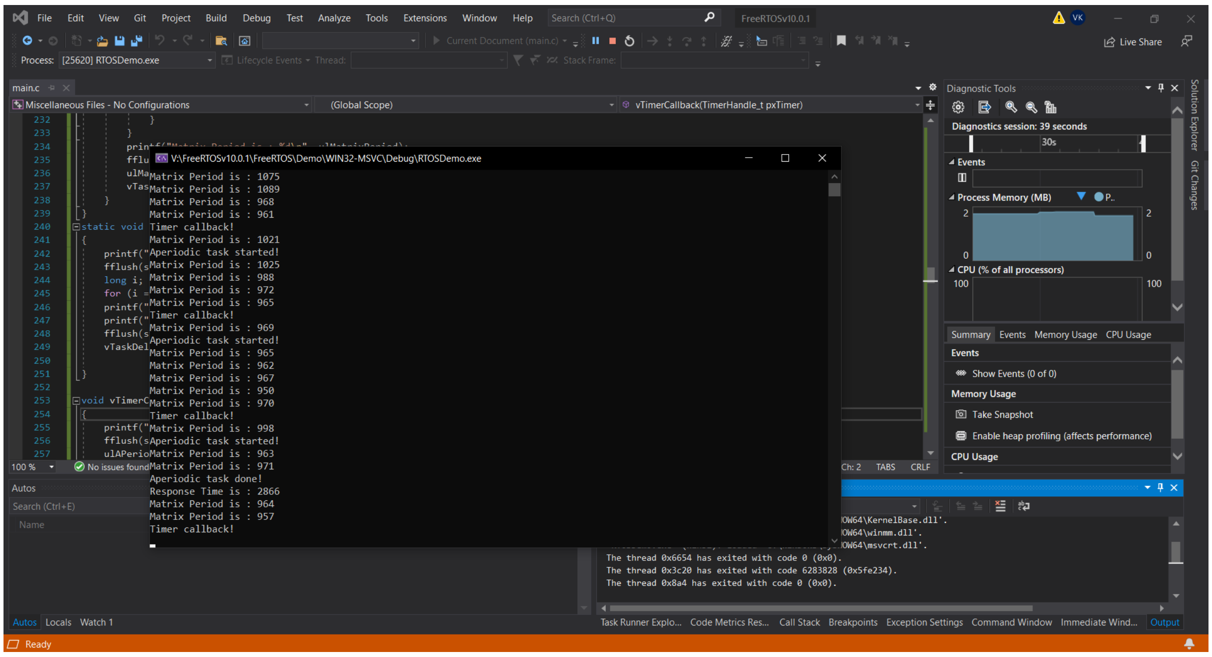
```c

static void vTimerCallback(TimerHandle_t pxTimer);
static void matrix_task();
static void aperiodic_task();
int main(void)
{
    /* This demo uses heap_5.c, so start by defining some heap regions.  heap_5
    is only used for test and example reasons.  Heap_4 is more appropriate.  See
    http://www.freertos.org/a00111.html for an explanation. */
    prvInitialiseHeap();

    /* Initialise the trace recorder.  Use of the trace recorder is optional.
    See http://www.FreeRTOS.org/trace for more information. */
    vTraceEnable(TRC_START);
    pxTimer_handle = xTimerCreate("Timer",5000,pdTRUE,(void*)0,vTimerCallback);

    if (pxTimer_handle == NULL)
    {
        /* The timer was not created. */
        printf("The timer was not created!\n");
        fflush(stdout);
    }
    else
    {
        /* Start the timer.  No block time is specified, and
        even if one was it would be ignored because the RTOS
        scheduler has not yet been started. */
        if (xTimerStart(pxTimer_handle, 0) != pdPASS)
        {
            /* The timer could not be set into the Active state. */
            printf("The timer could not be set into the Active state!\n");
            fflush(stdout);
        }
    }

```



V:\FreeRTOSv10.0.1\FreeRTOS\Demo\WIN32-MSVC\Debug\RTOSDemo.exe

```
Matrix Period is : 1075
Matrix Period is : 1089
Matrix Period is : 968
Matrix Period is : 961
Timer callback!
Matrix Period is : 1021
Aperiodic task started!
Matrix Period is : 1025
Matrix Period is : 988
Matrix Period is : 972
Matrix Period is : 965
Timer callback!
Matrix Period is : 969
Aperiodic task started!
Matrix Period is : 965
Matrix Period is : 962
Matrix Period is : 967
Matrix Period is : 950
Matrix Period is : 970
Timer callback!
Matrix Period is : 998
Aperiodic task started!
Matrix Period is : 963
Matrix Period is : 971
Aperiodic task done!
Response Time is : 2866
Matrix Period is : 964
Matrix Period is : 957
Timer callback!
```

```
OW64\KernelBase.dll'.
OW64\winmm.dll'.
OW64\msvcrt.dll'.
The thread 0x6654 has exited with code 0 (0x0).
The thread 0x3c20 has exited with code 6283828 (0x5fe234).
The thread 0x8a4 has exited with code 0 (0x0).
```

1. **Is the system fast enough to handle all aperiodic tasks? Why?**
   Yes, the system is fast enough to handle the aperiodic task.
   Because, aperiodic task is generated only when there is a timer interrupt every 5 seconds. But the matrix multiplication only takes 600 to 700ms.

2. **If not, solve this problem without alter the functionality of any task**
   I do not see a problem on my machine. In case, if I cannot handle all aperiodic task within next period (5 seconds in this case), I could have created a set priority task and increased the priority of the aperiodic task.

3. **What is the response time of the aperiodic task?**
   Response time of the aperiodic task depends on the execution state of Matrix Multiplication task at the time of 5 seconds timer interrupt. For me, I was able to see response time of 400ms on an average.

4. **Provide a screenshot of the running system**
   Please find the attached output.png image.