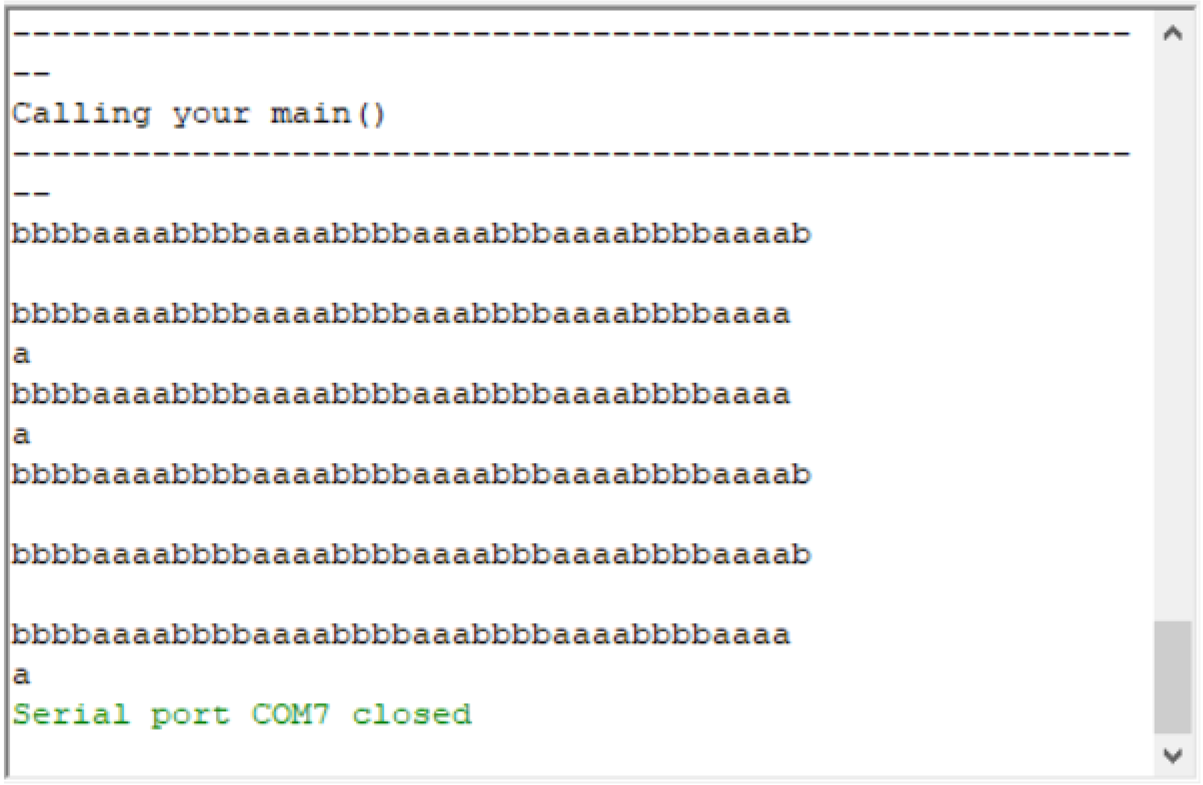
Schiffer, Nickolas

ID # 012279319

CMPE-146: Preet

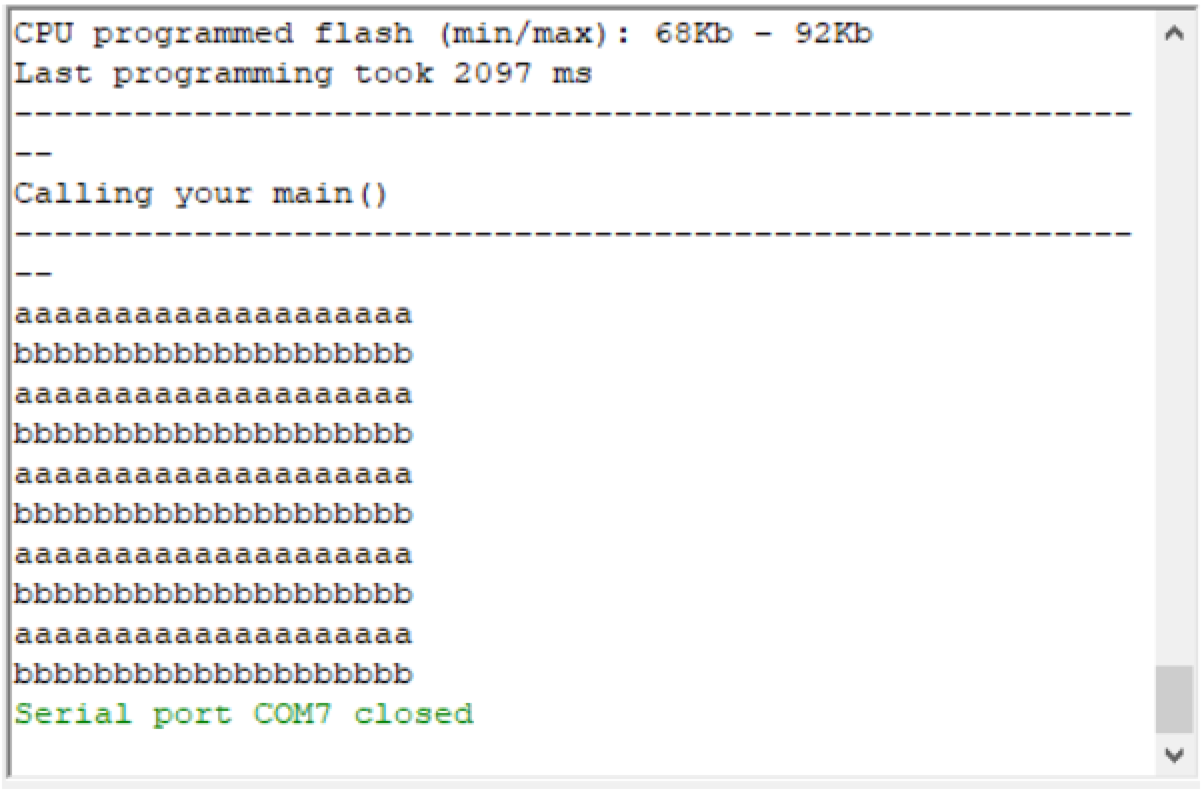
Assignment: FreeRTOS Tasks

1. With the tasks having the same priority, the scheduler services them in round-robin fashion. According to the assignment reference, the uart0_puts function transfers bits at 38400bps and it takes 10 bits to send an 8-bit character. This means that (38400 / 10) = 3840 characters per second can be transferred. The round robin scheduler will switch between the tasks at the system tick rate of 1ms. This means that the scheduler will switch between the tasks 1000 times per second. This averages to 3.84 characters per turn. This is why 3 characters will generally be sent, with an extra character sometimes being sent on the tasks next turn as shown below.



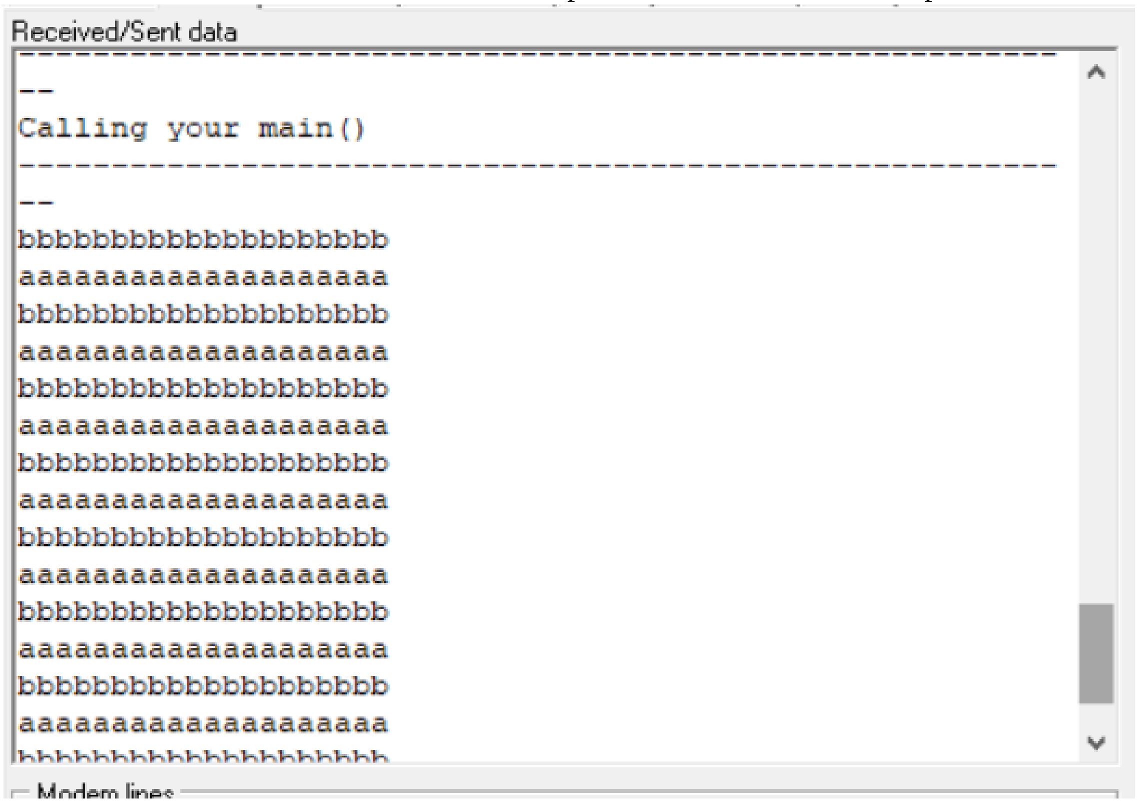
```
---------------------------------------------------------
--
Calling your main()
---------------------------------------------------------
--
bbbbaaaabbbbaaaabbbbaaaabbbaaaabbbbaaaab

bbbbaaaabbbbaaaabbbbaaabbbbaaaabbbbaaaa
a
bbbbaaaabbbbaaaabbbbaaabbbbaaaabbbbaaaa
a
bbbbaaaabbbbaaaabbbbaaaabbbaaaabbbbaaaab

bbbbaaaabbbbaaaabbbbaaaabbbaaaabbbbaaaab

bbbbaaaabbbbaaaabbbbaaabbbbaaaabbbbaaaa
a
Serial port COM7 closed
```

2. In this scenario, Task A has a higher priority than priority B. Due to the main rule of FreeRTOS, the higher priority tasks are serviced first before any lower priority tasks. In this case, Task A will run without interruption until it sleeps. During this turn, Task A has all of the time that it needs to complete its print statement and then sleep. It is not until this point that Task B is allowed to run. While Task A is sleeping, Task B has plenty of time to complete its print statement as well. During this sleep state, there is time to print up to 384 characters. When Task A's sleep time expires, the process repeats.



```
CPU programmed flash (min/max): 68Kb - 92Kb
Last programming took 2097 ms
---------------------------------------------------------
--
Calling your main()
---------------------------------------------------------
--
aaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
Serial port COM7 closed
```

3. This scenario is identical to the second, except with Task B having the higher priority. Task B will run uninterrupted until it completes its print statement and sleeps. At this time Task A has time to run and finishes its print statement while B sleeps.



```
---------------------------------------------------------
--
Calling your main()
---------------------------------------------------------
--
bbbbbbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbb
```

```c
#include "FreeRTOS.h"
#include "task.h"
#include "uart0_min.h"

#define SCENARIO 3

void vTaskOneCode(void *p)
{
    while(1)
    {
     uart0_puts("aaaaaaaaaaaaaaaaaaaa");
     vTaskDelay(100); // This sleeps the task for 100ms (because 1
RTOS tick = 1 millisecond)
    }
}

// Create another task and run this code in a while(1) loop
void vTaskTwoCode(void *p)
{
    while(1)
    {
     uart0_puts("bbbbbbbbbbbbbbbbbbbb");
     vTaskDelay(100);
    }
}

// You can comment out the sample code of lpc1758_freertos project and
run this code instead
int main(int argc, char const *argv[])
{
    /// This "stack" memory is enough for each task to run properly
(512 * 32-bit) = 2Kbytes stack
    const uint32_t STACK_SIZE_WORDS = 512;

    /**
     * Observe and explain the following scenarios:
     *
     * 1) Same Priority: A = 1, B = 1
     * 2) Different Priority: A = 2, B = 1
     * 3) Different Priority: A = 1, B = 2
     *
     * Turn in screen shots of what you observed
     * as well as an explanation of what you observed
     */
```

```
#ifndef SCENARIO
# error "SCENARIO is not defined."
#endif
#if SCENARIO == 1
    //Scenario 1
    xTaskCreate(vTaskOneCode,"Task A", STACK_SIZE_WORDS, NULL, 
PRIORITY_LOW, NULL);
    xTaskCreate(vTaskTwoCode,"Task B", STACK_SIZE_WORDS, NULL, 
PRIORITY_LOW, NULL);
#endif
#if SCENARIO == 2
    //Scenario 2
    xTaskCreate(vTaskOneCode,"Task A", STACK_SIZE_WORDS, NULL, 
PRIORITY_MEDIUM, NULL);
    xTaskCreate(vTaskTwoCode,"Task B", STACK_SIZE_WORDS, NULL, 
PRIORITY_LOW, NULL);
#endif

#if SCENARIO == 3
    //Scenario 3
    xTaskCreate(vTaskOneCode,"Task A", STACK_SIZE_WORDS, NULL, 
PRIORITY_LOW, NULL);
    xTaskCreate(vTaskTwoCode,"Task B", STACK_SIZE_WORDS, NULL, 
PRIORITY_MEDIUM, NULL);
#endif
    /* Start Scheduler - This will not return, and your tasks will 
start to run their while(1) loop */
    vTaskStartScheduler();

    return 0;
}
```