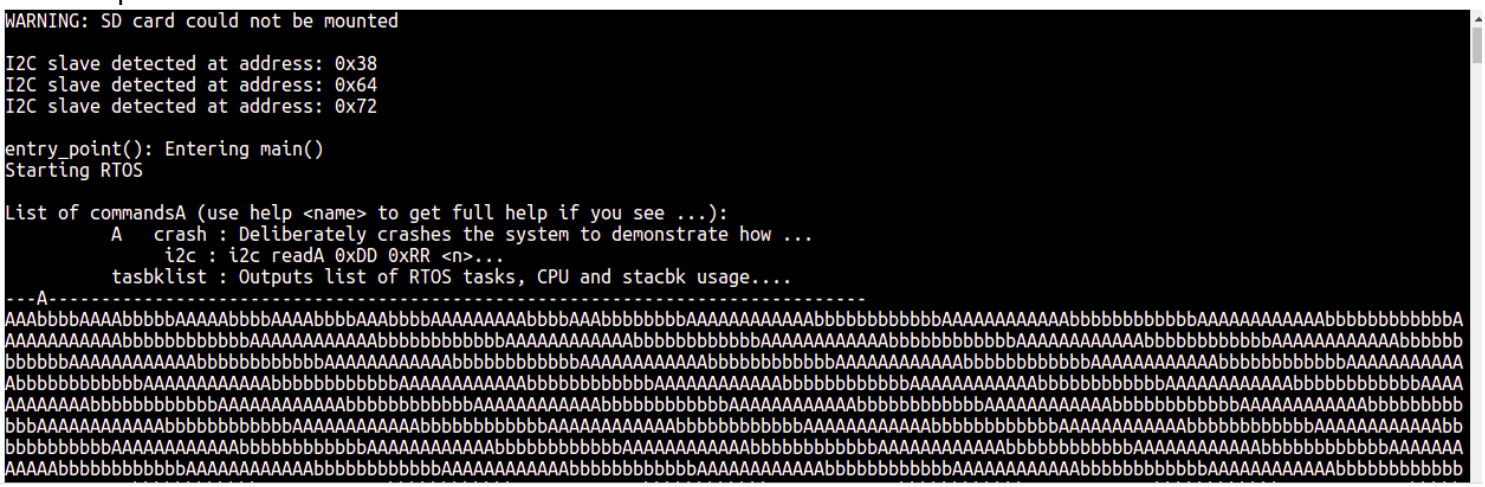
Multiple Tasks Assignment

Name: Tejas Madhukar Pidkalwar
Student Id: 014635491

1) When Task_one and Task_two has equal priority = PRIORITY_LOW

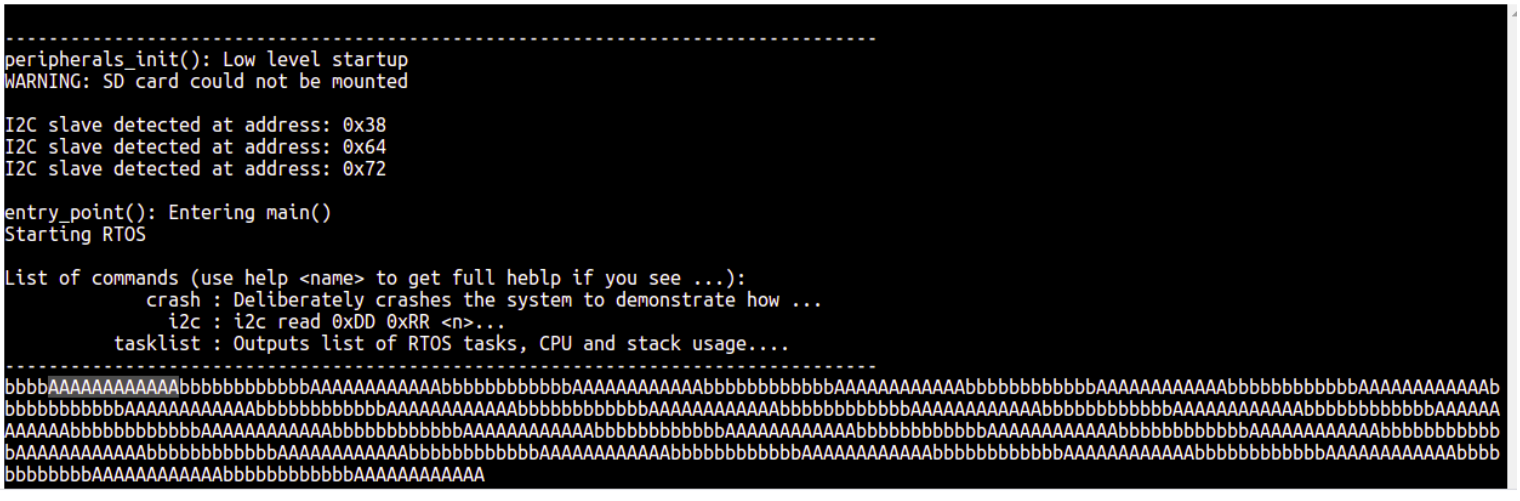
output:



Solution:
From the above output observation we could see that, characters A is printed 3-4 times and character b is printed same way. This is because:
→ The baudrate is 38400, i.e. the serial communication can transfer 38400 bits per second.
→ One character needs 10 bit transfer, thus transfer rate is 3840 characters per second
→ As, Task_one and Task_two has same priority == LOW, RTOS will jump between these two tasks for every tick time = (1/1kHz = 1mSec). Thus, each task would run to transfer 3.8 characters i.e. 3-4 characters before switching to other task.

2) When priority of Task_one = PRIORITY_LOW, and Task_two = PRIORITY_MEDIUM

Output:



Solution:

The above output observation we can see that,

→ character b is printed 4 times and then switched to Task_one

- Task_two with priority 2 ran for 1 mSec, then the higher priority task (UART_TASK) preempted it (Task_two went to sleep for 1 sec) to print data. UART_TASK went to sleep after printing data.

→ character A is printed 12 times, .i.e. Task_one ran fully without preemption, as other tasks were sleeping.

→ character b is printed 12 times .i.e. Task_two ran fully without preemption as other tasks were sleeping.

→ pattern repeated same way.

3) When priority of Task_one = PRIORITY_MEDIUM, and Task_two = PRIORITY_LOW

Output:

```
--------------------------------------------------------------------------------
peripherals_init(): Low level startup
WARNING: SD card could not be mounted

I2C slave detected at address: 0x38
I2C slave detected at address: 0x64
I2C slave detected at address: 0x72

entry_point(): Entering main()
Starting RTOS

List of commands (use help <name> to get full heAlp if you see ...):
             crash : Deliberately crashes the system to demonstrate how ...
               i2c : i2c read 0xDD 0xRR <n>...
          tasklist : Outputs list of RTOS tasks, CPU and stack usage....
--------------------------------------------------------------------------------
AAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbA
AAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbb
bbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAA
AbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAA
AAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbb
bbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbbbAAAAAAAAAAAAbb
bbbbbbbbbbAAAAAAAAAAAAbbbbbbbbbbb
```

Analysis:

From the above output, I found the following observations:

→ Initial 4 characters printed from Task_one, Because:

- It has higher priority than the task printing character b
- It only printed 4 characters, as after 1 tick time RTOS preempted this task to rum higher priority task of UART_TASK

→ next characters printed are b, 12 times. i.e. Task_two ran fully without preemption

- As, other tasks were sleeping, no task was available to preempt

→ next characters printed are A, 12 times, i.e. Task_one ran fully without preemption

- As other tasks were sleeping, no one was available to preempt this one

→ This pattern repeated throughout processing