

NAME: ABDULLAH NAZISH

PRN:210550354121

RTOS ASSIGNMENT

1. Find what is the task priority numbering for the RTOS you are using. eg. Higher the number higher the priority or vice-versa. Find the range of priority that can be assigned to a task for your RTOS.

Each task is assigned a priority from 0 to (configMAX_PRIORITIES - 1), (configMAX_PRIORITIES Is defined in FreeRTOSConfig.h). Low priority numbers denote low priority tasks. The idle task has priority zero (tskIDLE_PRIORITY).

Defines:

(tskIDLE_PRIORITY + 1)

(configMAX_PRIORITIES -1)

// -1 is the maximum priority available

2. What is the mechanism used to make a task periodic for the RTOS you are using? Write a program to make a task periodic with periodicity of 500ms.

```
#include <stdio.h>
```

```
#include "freertos/FreeRTOS.h"
```

```
#include "freertos/task.h"
```

```
void task(void *data)
```

```
{
    while(1)
    {
        printf("task started\n");
        printf("task ended\n");
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}
```

```
void app_main(void)
```

```
{
    xTaskCreate(task, "task1", 1024, NULL, 3, NULL);
}
```

```
abdullah@nazish: ~/Desktop/RTOS/ques2
I (268) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (274) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (281) heap_init: At 4008ADA0 len 00015260 (84 KiB): IRAM
I (288) spi_flash: detected chip: generic
I (292) spi_flash: flash io: dio
I (297) sleep: Configure to isolate all GPIO pins in sleep state
I (303) sleep: Enable automatic switching of GPIO sleep configuration
I (310) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
task started
task ended
task started
task ended
task started
task ended
task started
task ended
task started
task ended
task started
task ended
task started
task ended
task started
task ended
```

3. Find the APIs in your RTOS that provides timestamp and use it to print the periodic task. Observe the jitter in the timestamp vs the periodicity. Enhance the code to 10 periodic tasks with different periodicity. Further observe the jitter in each of the task.

```
#include <stdio.h>
```

```
#include "freertos/FreeRTOS.h"
```

```
#include "freertos/task.h"
```

```
void task(void *data)
```

```
{
    while(1)
    {
        printf("task started\n");
        printf("task ended\n");
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}
```

```
void app_main(void)
```

```
{
    BaseType_t x;
    x = xTaskCreate(task, "task1", 1024, NULL, 3, NULL);
    printf("TASK %d", x);
}
```

```
abdullah@nazish: ~/Desktop/RTOS/ques3
I (217) cpu_start: cpu freq: 160000000
I (217) cpu_start: Application information:
I (221) cpu_start: Project name:      q2
I (226) cpu_start: App version:      1
I (230) cpu_start: Compile time:      Sep 20 2021 07:52:58
I (236) cpu_start: ELF file SHA256:  145f4d6a22381b11...
I (242) cpu_start: ESP-IDF:           v4.4-dev-2944-g316988bd2d
I (249) heap_init: Initializing. RAM available for dynamic allocation:
I (256) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (262) heap_init: At 3FFB2CC8 len 0002D338 (180 KiB): DRAM
I (268) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (274) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (281) heap_init: At 4008ADA0 len 00015260 (84 KiB): IRAM
I (288) spi_flash: detected chip: generic
I (292) spi_flash: flash io: dio
I (297) sleep: Configure to isolate all GPIO pins in sleep state
I (303) sleep: Enable automatic switching of GPIO sleep configuration
I (310) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
TASK 1task started
task ended
task started
task ended
```

4. Create two task with priority 10 and 20. Each task prints its own custom message.

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

void task(void *data)
{
    while(1)
    {
        printf("task1 started\n");
        vTaskDelay(5000 / portTICK_PERIOD_MS);
    }
}

void task2(void *data)
{
    while(1)
    {
        printf("task2 started\n");
        vTaskDelay(5000 / portTICK_PERIOD_MS);
    }
}
```

```

void app_main(void)
{

    xTaskCreate(task, "task1", 1024, NULL, 10, NULL);
    xTaskCreate(task2, "task2", 1024, NULL, 20, NULL );

}

```

```

I (221) cpu_start: Project name:      q2
I (226) cpu_start: App version:      1
I (230) cpu_start: Compile time:      Sep 20 2021 07:52:58
I (236) cpu_start: ELF file SHA256:  e9a360e79ed7ad88...
I (242) cpu_start: ESP-IDF:          v4.4-dev-2944-g316988bd2d
I (249) heap_init: Initializing. RAM available for dynamic allocation:
I (256) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (262) heap_init: At 3FFB2CC8 len 0002D338 (180 KiB): DRAM
I (268) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (274) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (281) heap_init: At 4008ADA0 len 00015260 (84 KiB): IRAM
I (288) spi_flash: detected chip: generic
I (292) spi_flash: flash io: dio
I (297) sleep: Configure to isolate all GPIO pins in sleep state
I (303) sleep: Enable automatic switching of GPIO sleep configuration
I (310) cpu_start: Starting scheduler on PRO CPU.
I (0)  cpu_start: Starting scheduler on APP CPU.
task1 started
task2 started
task2 started
task1 started
task2 started
task1 started

```

5. Swap the priority and observe the changes in the output. What is your conclusion on the sequence of printing the messages.

```

#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

void task(void *data)
{
    while(1)
    {
        printf("task1 started\n");
        vTaskDelay(5000 / portTICK_PERIOD_MS);
    }
}

```

```

void task2(void *data)
{
    while(1)
    {
        printf("task2 started\n");
        vTaskDelay(5000 / portTICK_PERIOD_MS);
    }
}

void app_main(void)
{
    xTaskCreate(task, "task1", 1024, NULL, 20, NULL);
    xTaskCreate(task2, "task2", 1024, NULL, 10, NULL );
}

```

```

I (230) cpu_start: Compile time:      Sep 20 2021 07:52:58
I (236) cpu_start: ELF file SHA256:  9ec9e7af36566f8e...
I (242) cpu_start: ESP-IDF:          v4.4-dev-2944-g316988bd2d
I (249) heap_init: Initializing. RAM available for dynamic allocation:
I (256) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (262) heap_init: At 3FFB2CC8 len 0002D338 (180 KiB): DRAM
I (268) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (274) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (281) heap_init: At 4008ADA0 len 00015260 (84 KiB): IRAM
I (288) spi_flash: detected chip: generic
I (292) spi_flash: flash io: dio
I (297) sleep: Configure to isolate all GPIO pins in sleep state
I (303) sleep: Enable automatic switching of GPIO sleep configuration
I (310) cpu_start: Starting scheduler on PRO CPU.
I (0)  cpu_start: Starting scheduler on APP CPU.
task1 started
task2 started
task1 started
task2 started
task1 started
task2 started
task1 started
task2 started

```

6. What are the maximum number of tasks that can be created on the RTOS you are using? Is it limited by the RTOS design or underlying hardware resources or both.
Yes , we can create multiple task but it is depend upon memory, by default ram size is 4 MB.
7. What is the scheduling algorithm used by your RTOS?
Generally , used priority pre-emption scheduling in RTOS

8. List the customization options for creating a task for the RTOS you are using. eg. Priority, etc.

```
BaseType_t xTaskCreate( TaskFunction_t pvTaskCode,
                        const char * const pcName,
                        configSTACK_DEPTH_TYPE usStackDepth,
                        void *pvParameters,
                        UBaseType_t uxPriority,
                        TaskHandle_t *pxCreatedTask
                        );
```

9. Find the fields that are maintained in the Task Control Block / Process Control Block of the RTOS you are using.

- Process State. This specifies the process state i.e. new, ready, running, waiting or terminated.
- Process Number.
- Program Counter.
- Registers.
- CPU Scheduling Information.
- Memory Management Information.
- I/O Status Information.

10. Draw a process or task state diagram for the RTOS you are using.

- DORMANT
- READY
- RUNNING
- WAITING
- ISR

11. What is the API for deleting a task? Write a program demonstrate this capability.

```
void vTaskDelete( TaskHandle_t xTask );
```

```
-----
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
void task(void *data)
{

    printf("task1 created\n");
    vTaskDelete(NULL);
```

```

}
void app_main(void)
{

    xTaskCreate(task, "task1", 1024, NULL, 20, NULL);

}

```

```

I (0) cpu_start: App cpu up.
I (216) cpu_start: Pro cpu start user code
I (217) cpu_start: cpu freq: 160000000
I (217) cpu_start: Application information:
I (221) cpu_start: Project name:      q2
I (226) cpu_start: App version:      1
I (230) cpu_start: Compile time:      Sep 20 2021 07:52:58
I (236) cpu_start: ELF file SHA256:  612b1bbe5c24b128...
I (242) cpu_start: ESP-IDF:          v4.4-dev-2944-g316988bd2d
I (249) heap_init: Initializing. RAM available for dynamic allocation:
I (256) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (262) heap_init: At 3FFB2CC8 len 0002D338 (180 KiB): DRAM
I (268) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (274) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (281) heap_init: At 4008ADA0 len 00015260 (84 KiB): IRAM
I (288) spi_flash: detected chip: generic
I (292) spi_flash: flash io: dio
I (297) sleep: Configure to isolate all GPIO pins in sleep state
I (303) sleep: Enable automatic switching of GPIO sleep configuration
I (310) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
task1 created

```

12. What are the APIs provided by your RTOS for enabling and disabling the interrupts?
Write a program to demonstrate this capability?

```

void taskENABLE_INTERRUPTS( void );
void taskDISABLE_INTERRUPTS( void );

```

13. Does your RTOS provide APIs to collect task statistics. If yes, list the statistics parameters that are collected and write a program to display the runtime task statistics?

```
TaskHandle_t xTaskCreateStatic( TaskFunction_t pxTaskCode,  
                               const char * const pcName,  
                               const uint32_t ulStackDepth,  
                               void * const pvParameters,  
                               UBaseType_t uxPriority,  
                               StackType_t * const puxStackBuffer,  
                               StaticTask_t * const pxTaskBuffer );
```

```
#include <stdio.h>
```

```
#include "freertos/FreeRTOS.h"
```

```
#include "freertos/task.h"
```

```
#define STACK_SIZE 2048
```

```
StaticTask_t xTaskBuffer;
```

```
StackType_t xStack[ STACK_SIZE ];
```

```
void task(void *data)
```

```
{  
    while(1)  
    {  
        printf("task started\n");  
        printf("task ended\n");  
        vTaskDelay(500 / portTICK_PERIOD_MS);  
    }  
}
```

```
void app_main(void)
```

```
{  
    TaskHandle_t xHandle = NULL;  
    xHandle = xTaskCreateStatic(task, "task1", STACK_SIZE, ( void * ) 1, 1, xStack  
, &xTaskBuffer);  
}
```



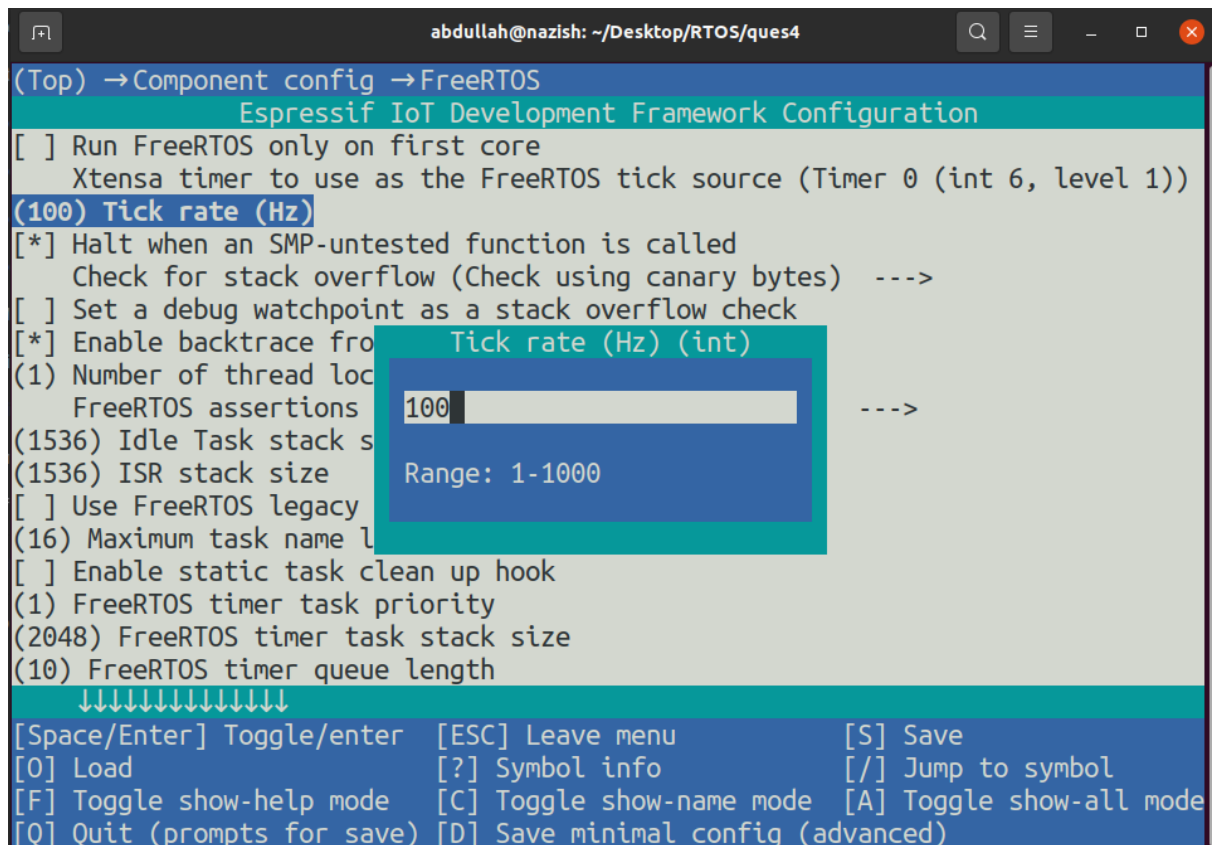
```

I (281) heap_init: At 4008ADA0 len 00015260 (84 KiB): IRAM
I (288) spi_flash: detected chip: generic
I (292) spi_flash: flash io: dio
I (297) sleep: Configure to isolate all GPIO pins in sleep state
I (303) sleep: Enable automatic switching of GPIO sleep configuration
I (310) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
task started
task ended
task started
task ended
task started
task ended
task started
task ended
task started
task ended
task started
task ended
task started
task ended
task started
task ended
task started
task ended

```

14. Find the tick frequency configuration for your RTOS.

By default, it is 100 Hz and range is 1-1000 Hz



15. Create a task to suspend itself after 1200 ms and resume it from another task

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

TaskHandle_t TaskHandle_1;
TaskHandle_t TaskHandle_2;

void task1(void *data);
void task2(void *data);

static void MyTask1(void* data)
{
    printf("task1 started\n");
    printf("Task1 Resuming Task2\n");
    vTaskResume(TaskHandle_2);
    vTaskDelete(NULL);
}

static void MyTask2(void* data)
{
    printf("Task2 started\n");
    vTaskSuspend(NULL);
    printf("task2 deleting itself\n");
    vTaskDelete(NULL);
}

void app_main(void)
{
    xTaskCreate(MyTask2, "Task2", 1024, NULL, 2, &TaskHandle_2);
    xTaskCreate(MyTask1, "Task1", 1024, NULL, 1, &TaskHandle_1);
}
```

```

I (216) cpu_start: cpu freq: 160000000
I (217) cpu_start: Application information:
I (221) cpu_start: Project name:      q2
I (226) cpu_start: App version:      1
I (230) cpu_start: Compile time:      Sep 20 2021 07:52:58
I (236) cpu_start: ELF file SHA256:   d9f0f5117a8d4c3d...
I (242) cpu_start: ESP-IDF:           v4.4-dev-2944-g316988bd2d
I (249) heap_init: Initializing. RAM available for dynamic allocation:
I (256) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (262) heap_init: At 3FFB2CD0 len 0002D330 (180 KiB): DRAM
I (268) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (274) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (281) heap_init: At 4008B024 len 00014FDC (83 KiB): IRAM
I (288) spi_flash: detected chip: generic
I (292) spi_flash: flash io: dio
I (297) sleep: Configure to isolate all GPIO pins in sleep state
I (303) sleep: Enable automatic switching of GPIO sleep configuration
I (310) cpu_start: Starting scheduler on PRO CPU.
I (0)  cpu_start: Starting scheduler on APP CPU.
Task2 started
task1 started
Task1 Resuming Task2
task2 deleting itself

```

16. Write a RTOS application to demonstrate the use of changing priority

```

#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"

TaskHandle_t TaskHandle_1; // handler for Task1
TaskHandle_t TaskHandle_2; // handler for Task2

void Task1(void *data)
{
    UBaseType_t uxPriority = uxTaskPriorityGet( NULL );
    while(1)
    {
        printf("task1 started\n");
        vTaskPrioritySet( TaskHandle_2, ( uxPriority + 1 ) );
        printf("task1 ended\n");
        vTaskDelay(3000 / portTICK_PERIOD_MS);
    }
}

void Task2(void *data)
{
    UBaseType_t uxPriority = uxTaskPriorityGet( NULL );
    while(1)

```

```

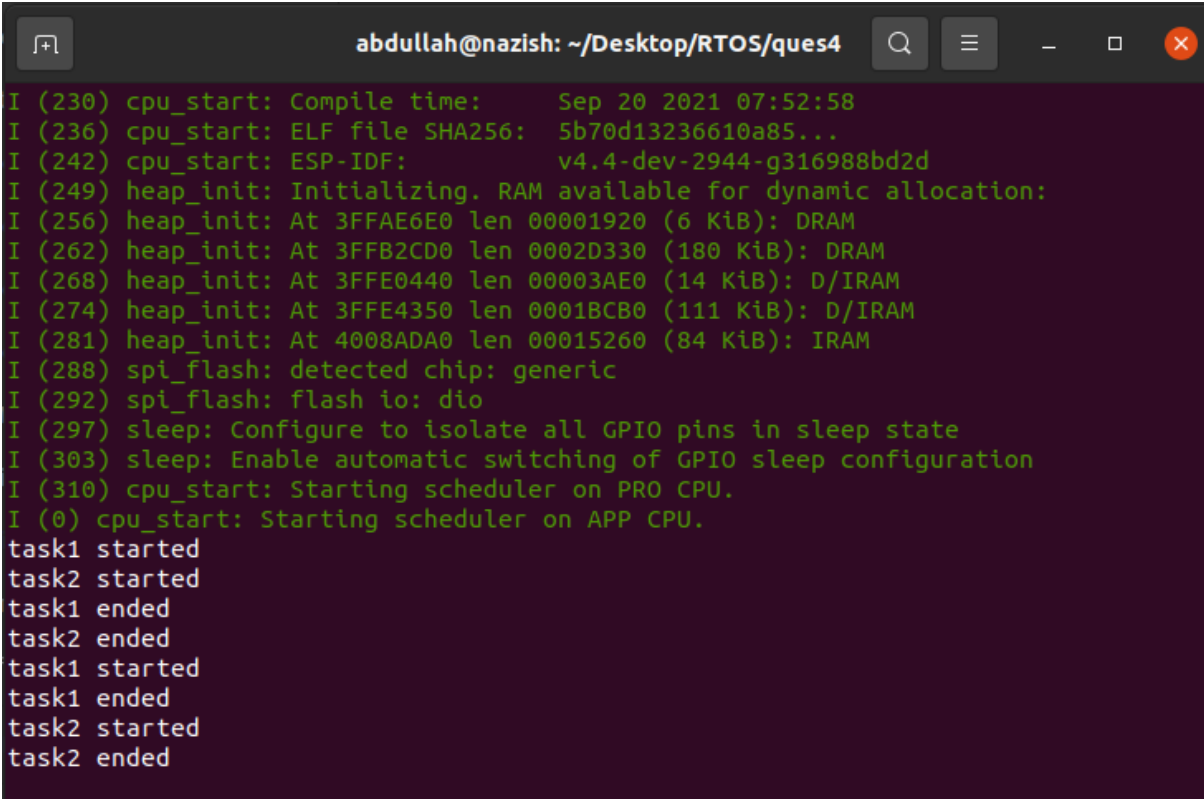
    {
        printf("task2 started\n");
        vTaskPrioritySet( TaskHandle_2, ( uxPriority - 2 ) );
        printf("task2 ended\n");
        vTaskDelay(3000 / portTICK_PERIOD_MS);
    }
}

void app_main(void)
{

    xTaskCreate(Task1, "task1", 2048, NULL, 3, &TaskHandle_1);
    xTaskCreate(Task2, "task2", 2048, NULL, 2, &TaskHandle_2);

}

```



```

abdullah@nazish: ~/Desktop/RTOS/ques4
I (230) cpu_start: Compile time:      Sep 20 2021 07:52:58
I (236) cpu_start: ELF file SHA256:  5b70d13236610a85...
I (242) cpu_start: ESP-IDF:          v4.4-dev-2944-g316988bd2d
I (249) heap_init: Initializing. RAM available for dynamic allocation:
I (256) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (262) heap_init: At 3FFB2CD0 len 0002D330 (180 KiB): DRAM
I (268) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (274) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (281) heap_init: At 4008ADA0 len 00015260 (84 KiB): IRAM
I (288) spi_flash: detected chip: generic
I (292) spi_flash: flash io: dio
I (297) sleep: Configure to isolate all GPIO pins in sleep state
I (303) sleep: Enable automatic switching of GPIO sleep configuration
I (310) cpu_start: Starting scheduler on PRO CPU.
I (0)  cpu_start: Starting scheduler on APP CPU.
task1 started
task2 started
task1 ended
task2 ended
task1 started
task1 ended
task2 started
task2 ended

```