Assignment RTOS

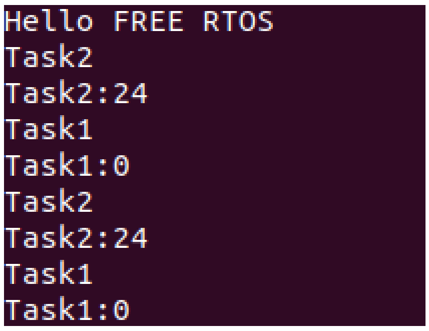
Question 1: Find what is the task priority numbering for the RTOS you are using. eg. Higher the number higher the priority or vice-versa. Find the range of priority that can be assigned to a task for your RTOS.

Answer: To find task priority numbering for RTOS we used :

```
uxTaskPriorityGet( TaskHandle_t xTask );
```

#define configMAX_PRIORITIES (25) . Each task is assigned a priority from 0 to (configMAX_PRIORITIES - 1) .so MAX priority for RTOS is 24.

output: //see program



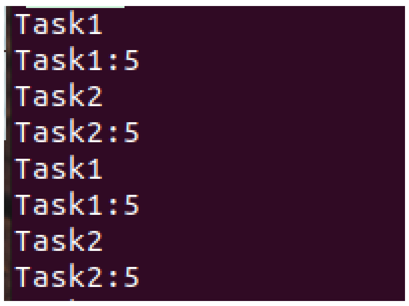
```
Hello FREE RTOS
Task2
Task2:24
Task1
Task1:0
Task2
Task2:24
Task1
Task1:0
```

As we can see the higher task priority is getting printed 1st.

Question 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.

Answer: Mechanism used are: vTaskDelay(1000 /portTICK_PERIOD_MS);

output: //see program

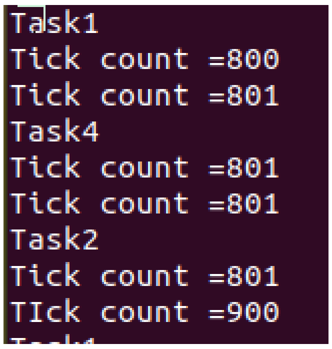


```
Task1
Task1:5
Task2
Task2:5
Task1
Task1:5
Task2
Task2:5
```

both the task have some priority and 500ms periodic so they are printing simultaneously.

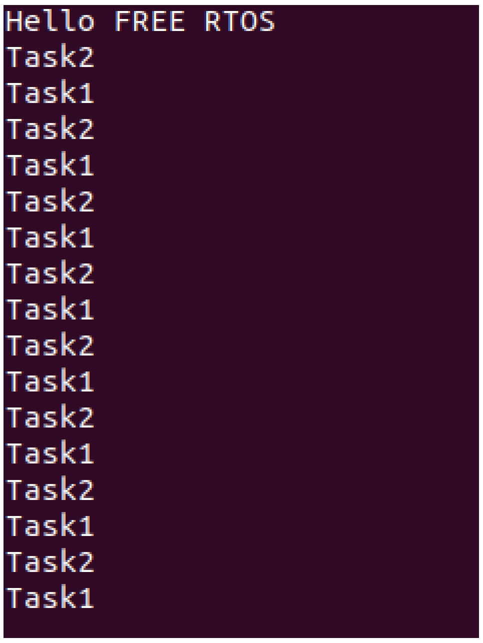
Question 3: Find the APIs in your RTOS that provides timestamp and use it to print the periodic task.
Observe the jitter in the timestamp vs the periodicity. Enhance the code to 10 periodic tasks with different periodicity. Futher observe the jitter in each of the task.

Answer: We use xTaskGetTickCount(void) to find tick of each task . Output:


```
Task1
Tick count =800
Tick count =801
Task4
Tick count =801
Tick count =801
Task2
Tick count =801
TIck count =900
```

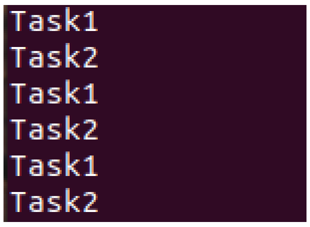
Question 4: Create two task with priority 10 and 20. Each task prints its own custom message.

Answer: output: Task 2 as 20 priority and 1 as 10 priority. //see program


```
Hello FREE RTOS
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
Task2
Task1
```

Question5: Swap the priority and observe the changes in the output. What is your conclusion on the
sequence of printing the messages. //see program

output: Task1 priority =20 ; Task2 priority =10;


```
Task1
Task2
Task1
Task2
Task1
Task2
```

Conclusion: As the priority change we can see Task sequence to change task1 get printed 1st.

Question6: 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.

Answer: The Maximum number of task can be created is depend upon the Heap size of the RTOS if the heap size is full we cannot created the new Task . Yes its limited by the Heap size design by the Hardware.

Question 7: What is the scheduling algorithm used by your RTOS?

Answer: A Scheduler algorithm used by the RTOS are:
1 - Premitive Scheduling algorithm : Allow the interrupted of current running task so urgent task can executed first.
2- Non - premitive Scheduling algorithm(Co-operative Scheduling): Its Does not allow the interrupted of task even if the task is urgent its first finish the task which it has started then take the urgent task. Scheduling Algorithm are: Round Robin Scheduling , Short Job first , FIFO (FIRST IN FIRST OUT),Priority Scheduling.

Question 8: List the customization options for creating a task for the RTOS you are using. eg. priority
etc?

Answer: The Customised option for creating task are in **FreeRTOSConfig.h** :

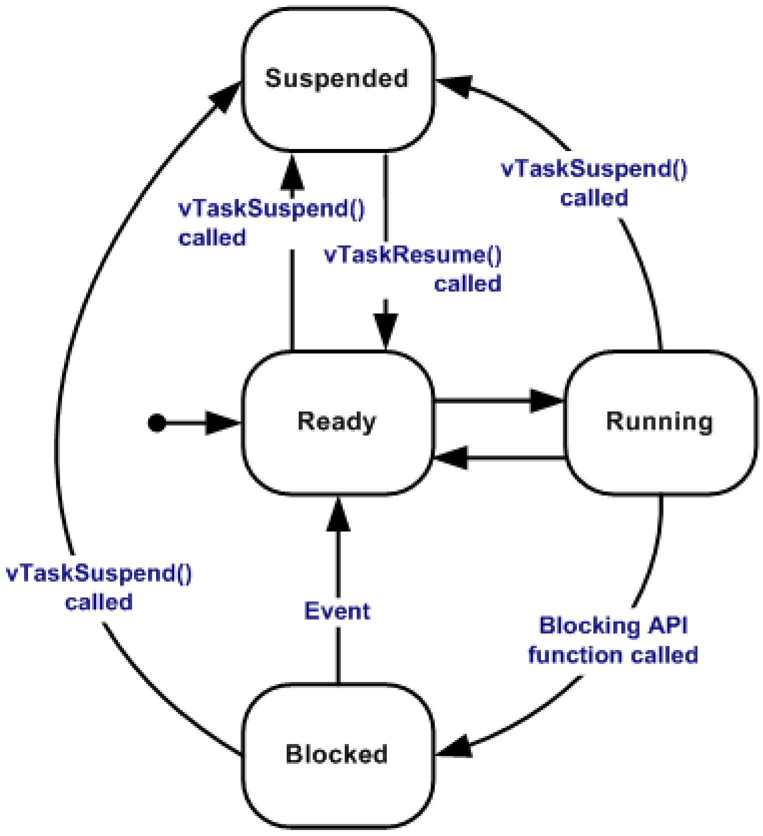
```
#define INCLUDE_vTaskPrioritySet                1
#define INCLUDE_uxTaskPriorityGet               1
#define INCLUDE_vTaskDelete                    1
#define INCLUDE_vTaskSuspend                    1
#define INCLUDE_vTaskDelayUntil                 1
#define INCLUDE_vTaskDelay                      1
#define INCLUDE_xTaskGetSchedulerState          1
#define INCLUDE_xTaskGetCurrentTaskHandle       1
#define INCLUDE_uxTaskGetStackHighWaterMark   0
#define INCLUDE_eTaskGetState                   0
#define INCLUDE_xTaskAbortDelay                 0
#define INCLUDE_xTaskGetHandle                  0
```

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

Answer:A process control block is a data structure that contain information of process related to it.Its also know as Task control block, Entry block etc.
Field of Process control block are: Process state , Process Number , Program Counter ,Register ,Memort=y Limit , List of open Files.

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

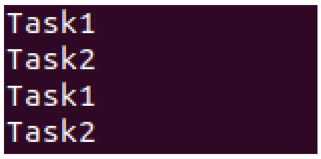
Answer :



Question 11:what is the API for deleting a task? Write a program demonstrate this capability?

Answer : VTaskDelete :This API is use to delete the Task .We need to pass the Task Handle of the task to delete the Task . To Delete the Taask we can pass null eg vTaskDelete(NULL); .

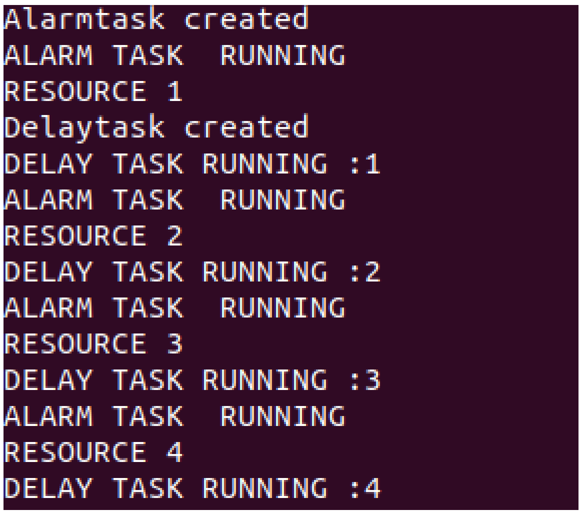
output program: //see program



```
Task1
Task2
Task1
Task2
```

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

Answer: The API for enabling interrupt is taskENTER_CRITICAL() ; and for disabling the interrupt we use taskEXIT_CRITICAL();. //see program

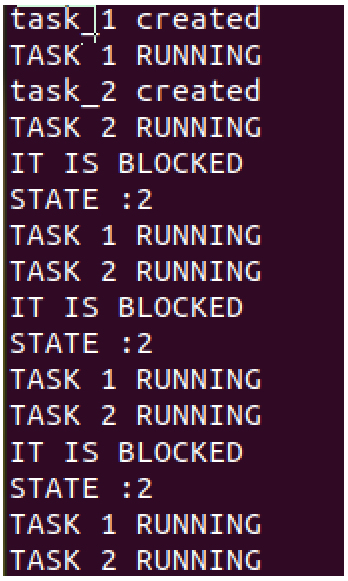

```
Alarmtask created
ALARM TASK  RUNNING
RESOURCE 1
Delaytask created
DELAY TASK RUNNING :1
ALARM TASK  RUNNING
RESOURCE 2
DELAY TASK RUNNING :2
ALARM TASK  RUNNING
RESOURCE 3
DELAY TASK RUNNING :3
ALARM TASK  RUNNING
RESOURCE 4
DELAY TASK RUNNING :4
```

**

Question 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?

Answer : Task statistics are - Create, Ready, Block, Suspend , Running

output://see programming


```
task_1 created
TASK 1 RUNNING
task_2 created
TASK 2 RUNNING
IT IS BLOCKED
STATE :2
TASK 1 RUNNING
TASK 2 RUNNING
IT IS BLOCKED
STATE :2
TASK 1 RUNNING
TASK 2 RUNNING
IT IS BLOCKED
STATE :2
TASK 1 RUNNING
TASK 2 RUNNING
```

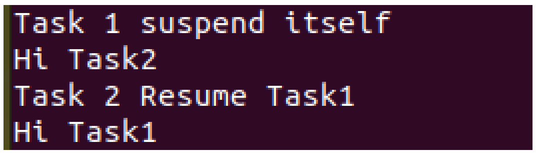
Question 14: Find the tick frequency configuration for your RTOS.

Answer: The Tick Frequency configuration for RTOS is 100 in menu config we can change it eg if its 100Hz then VTaskDelay(100)= 1 sec.

#define configTICK_RATE_HZ   (CONFIG_FREERTOS_HZ)

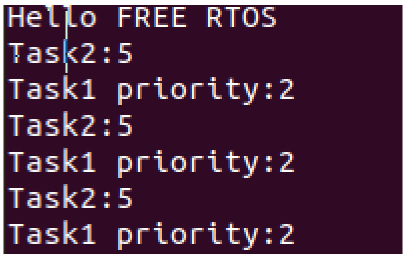
Question 15: Create a task to suspend itself after 1200 ms and resume it from another task?

Answer:**Output:**As task 1 is suspend by 1200ms and it resume in task2


```
Task 1 suspend itself
Hi Task2
Task 2 Resume Task1
Hi Task1
```

Question 16: Write a RTOS application to demonstrate the use of changing priority?

Answer :Output: creating both the task have same priority but in task1 i change priority to 2 .so task2 is printing 1st with higher priority. //see program

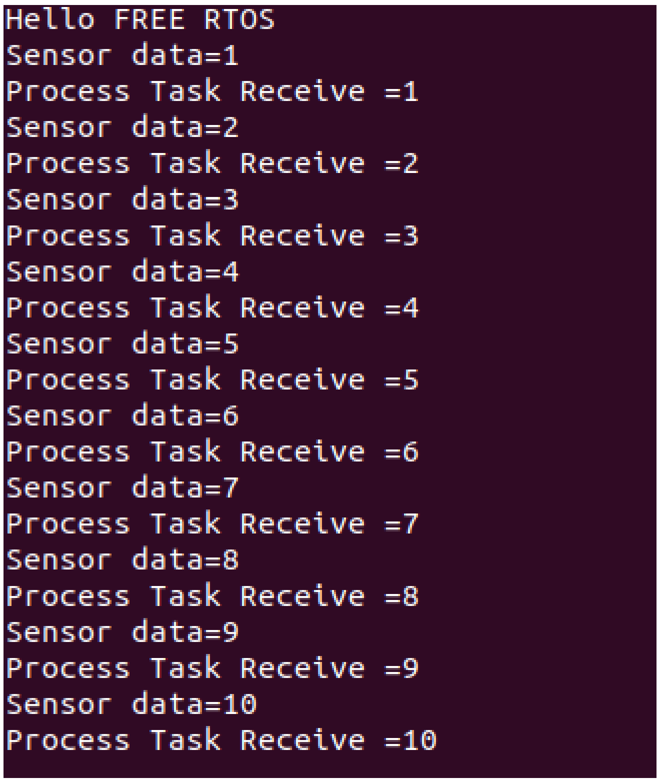

```
Hello FREE RTOS
Task2:5
Task1 priority:2
Task2:5
Task1 priority:2
Task2:5
Task1 priority:2
```

Question 17:f you RTOS supports idle task hooking, write a program to demonstrate it

Answer: Yes RTOS is supports idle task hooking // see program.

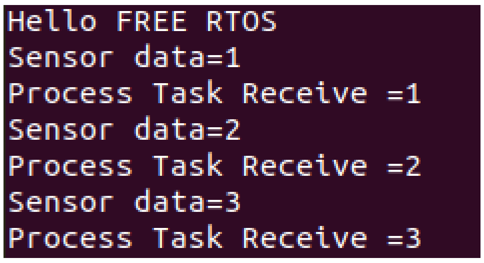
Question 18: Write a RTOS application to demonstrate the use of task to task communication using
Queue management APIs. Also demonstrate blocking on a queue.

Answer: //see program


```
Hello FREE RTOS
Sensor data=1
Process Task Receive =1
Sensor data=2
Process Task Receive =2
Sensor data=3
Process Task Receive =3
Sensor data=4
Process Task Receive =4
Sensor data=5
Process Task Receive =5
Sensor data=6
Process Task Receive =6
Sensor data=7
Process Task Receive =7
Sensor data=8
Process Task Receive =8
Sensor data=9
Process Task Receive =9
Sensor data=10
Process Task Receive =10
```

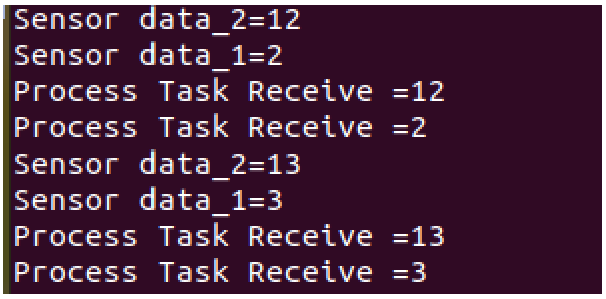
Question 19: Write a RTOS application to demonstrate the effects of task priorities when sending to and
receiving from a queue.

Answer: Output 1: //see program :only one sending task and received task with sending priority 5, and receiving priority 3:



```
Hello FREE RTOS
Sensor data=1
Process Task Receive =1
Sensor data=2
Process Task Receive =2
Sensor data=3
Process Task Receive =3
```

Output 2: Two sending task with differrent priority SendingTask_2 =7 and SendingTask_1=5 Receiv =3



```
Sensor data_2=12
Sensor data_1=2
Process Task Receive =12
Process Task Receive =2
Sensor data_2=13
Sensor data_1=3
Process Task Receive =13
Process Task Receive =3
```

as higher priority task is send and recieved 1st.

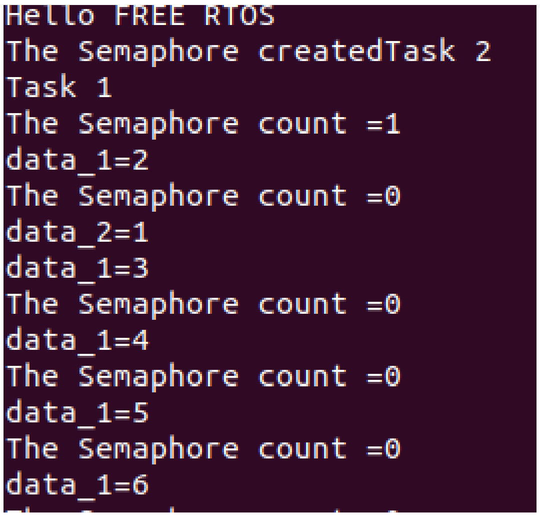
Question 20: Write a RTOS application to demonstrate deferred interrupt processing using binary semaphores

Answer: //see program

Question 21: Write a RTOS application to demonstrate counting semaphores to synchronize a task

Answer: //see program : As i have used share resources in two task but is synchronise using counting semaphore but semaphore count is 1 so it will execute only one task1 and put task2 into blocking stage

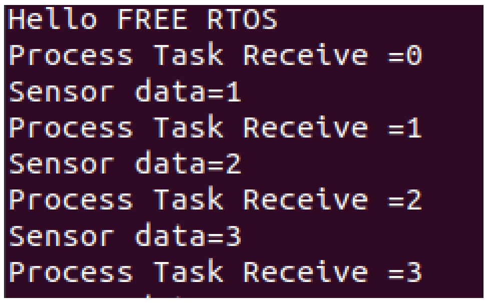
Output:



```
Hello FREE RTOS
The Semaphore createdTask 2
Task 1
The Semaphore count =1
data_1=2
The Semaphore count =0
data_2=1
data_1=3
The Semaphore count =0
data_1=4
The Semaphore count =0
data_1=5
The Semaphore count =0
data_1=6
```

Question22: Write a RTOS application to demonstrate the usage of queues within an interrupt service
routine

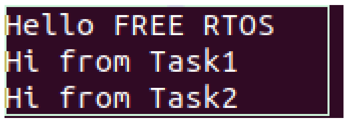
Answer:output //see program



```
Hello FREE RTOS
Process Task Receive =0
Sensor data=1
Process Task Receive =1
Sensor data=2
Process Task Receive =2
Sensor data=3
Process Task Receive =3
```

Question 23: Write a RTOS application to manage resources using mutual exclusion

Answer://see program :Output: mutex is used

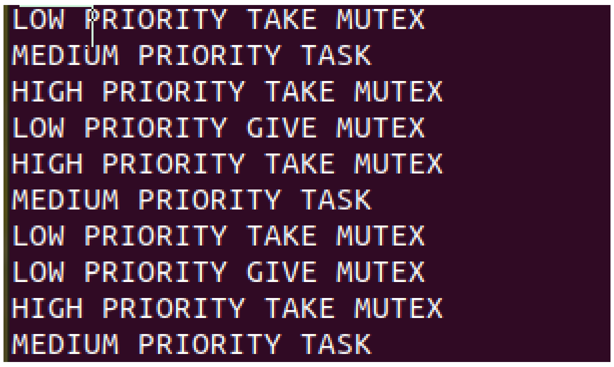


```
Hello FREE RTOS
Hi from Task1
Hi from Task2
```

Question 24: Write a RTOS application to demonstrate a priority inversion problem. If your RTOS sup-
ports priority inheritance or priority ceiling, use it to solve the priority inversion problem.

Answer :

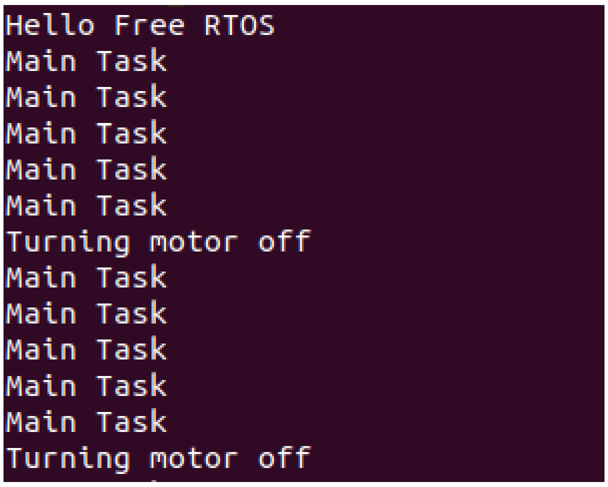
output:// see program



```
LOW PRIORITY TAKE MUTEX
MEDIUM PRIORITY TASK
HIGH PRIORITY TAKE MUTEX
LOW PRIORITY GIVE MUTEX
HIGH PRIORITY TAKE MUTEX
MEDIUM PRIORITY TASK
LOW PRIORITY TAKE MUTEX
LOW PRIORITY GIVE MUTEX
HIGH PRIORITY TAKE MUTEX
MEDIUM PRIORITY TASK
```

Question 25: Write a RTOS application to create a software timer that invokes a callback function every
5 seconds.

Answer: //see program : Output: Every 5 second the motor turned off.



```
Hello Free RTOS
Main Task
Main Task
Main Task
Main Task
Main Task
Turning motor off
Main Task
Main Task
Main Task
Main Task
Main Task
Turning motor off
```