

**GROUP-13**

**Real Time Operating Systems**

**Review – 04**

| NAME | USN |
| --- | --- |
| Aleena Mulla | 02FE21BEC006 |
| Basavesh Patil | 02FE21BEC018 |
| Harsh Patil | 02FE21BEC035 |
| Pratik Patil | 02FE21BEC063 |

**Problem statement:**

**Write a C program with two tasks and resource stepper motor.**

**Under the Guidance of**

**Prof. Swati Mavinkattimath Guide Signature** :

**Code:**

#include <rtl.h>

#include <lpc213x.h>

#include <string.h>

#include "stepper.h"

////////////////////////

#define SLAVE\_ADDR 78

#define MAX 12

#define AA 2

#define SI 3

#define STO 4

#define STA 5

#define I2EN 6

unsigned int i=0;

OS\_TID tsk1, tsk2;

OS\_SEM semaphore1;

/\*----------------------------------------------------------------------------

\* Task 1 - High Priority - Active every 3 ticks

\*---------------------------------------------------------------------------\*/

\_\_task void task1 (void) {

OS\_RESULT ret;

while (1) {

/\* Pass control to other tasks for 3 OS ticks \*/

os\_dly\_wait(3);

/\* Wait 1 ticks for the free semaphore \*/

ret = os\_sem\_wait (semaphore1, 1);

if (ret != OS\_R\_TMO) {

/\* If there was no time-out the semaphore was aquired \*/

//printf ("Task 1\n");

anti\_clock\_wise();

anti\_clock\_wise();

anti\_clock\_wise();

anti\_clock\_wise();

anti\_clock\_wise();

anti\_clock\_wise();

anti\_clock\_wise();

anti\_clock\_wise();

anti\_clock\_wise();

anti\_clock\_wise();

/\* Return a token back to a semaphore \*/

os\_sem\_send (semaphore1);

}

}

}

/\*----------------------------------------------------------------------------

\* Task 2 - Low Priority - looks for a free semaphore and uses the resource

\* whenever it is available

\*---------------------------------------------------------------------------\*/

\_\_task void task2 (void) {

while (1)

{

/\* Wait indefinetly for a free semaphore \*/

os\_sem\_wait (semaphore1, 0xFFFF);

/\* OK, the serial interface is free now, use it. \*/

clock\_wise();

clock\_wise();

clock\_wise();

clock\_wise();

clock\_wise();

clock\_wise();

clock\_wise();

clock\_wise();

clock\_wise();

clock\_wise();

/\* Return a token back to a semaphore. \*/

os\_sem\_send (semaphore1);

}

}

\_\_task void init (void) {

/\* Initialize the Semaphore before the first use \*/

os\_sem\_init (semaphore1, 1);

/\* Create an instance of task1 with priority 10 \*/

tsk1 = os\_tsk\_create (task1, 10);

/\* Create an instance of task2 with default priority 1 \*/

tsk2 = os\_tsk\_create (task2, 0);

/\* Delete the init task \*/

os\_tsk\_delete\_self ();

}

int main (void)

{

PINSEL1 = 0X00000000;

IODIR0 = 0X0000F000;

os\_sys\_init (init);

}

**Output:**

