

Practical Laboratory Work On Real Time Operating System: RTX51TINY[MCS 8051]

Developed by Dr. Alka Dubey

Sr. No.	Programs
1.	Write a program for creation of tasks for toggle the bit 0,1 and 2 of port 1 of 89C52 microcontroller using TINY OS
2.	Write a program for creation of tasks for toggle the port 0, port 1 and port 2 of 89C52 microcontroller using TINY OS
3.	Write a program for creation of tasks and sending signal to another task using TINY OS
4.	Write a program for creation of tasks then made another task as pended task and delete the main task TINY OS
5.	Write a program for suspended task for TINY OS
6.	Write a program for pended suspended delayed task for TINY OS
7.	Write a program to understand DEADLOCK in TINY OS
8.	Write a program for traffic light control using TINY OS

Program 1: Write a program for creation of tasks for toggle the bit 0,1 and 2 of port 1 of 89C52 microcontroller using TINY OS

Code:

```

/*****
/* task creation using tiny os */
*****/
#include <rtx51tiny.h> // header file for tiny os
#include <reg52.h> // header file for 8051

sbit PIN1=P1^0; // bit declaration
sbit PIN2=P1^1;
sbit PIN3=P1^2;
unsigned long count=0; // counter declaration

/*****
/*      task 0
*****/
*****/
job0 () _task_0
{
    os_create_task (1); // creation of task 1
    os_create_task (2); // creation of task 2
    while(1)
    {
        PIN1=0;
        os_wait (K_TMO, 5, 0); // wait for 50ms
        PIN1=1;
        os_wait (K_TMO, 5, 0); // wait for 50ms
    }
}
/*****
/*      task 1
*****/
*****/
job1 () _task_1
{
    while(1)
    {
        PIN2=0;
        os_wait (K_TMO, 30, 0); // wait for 300ms
        PIN2=1;
        os_wait (K_TMO, 30, 0); // wait for 300ms
    }
}
/*****
/*      task 2
*****/
*****/
job2 () _task_2
{
    while(1)
    {
        count=0;

```

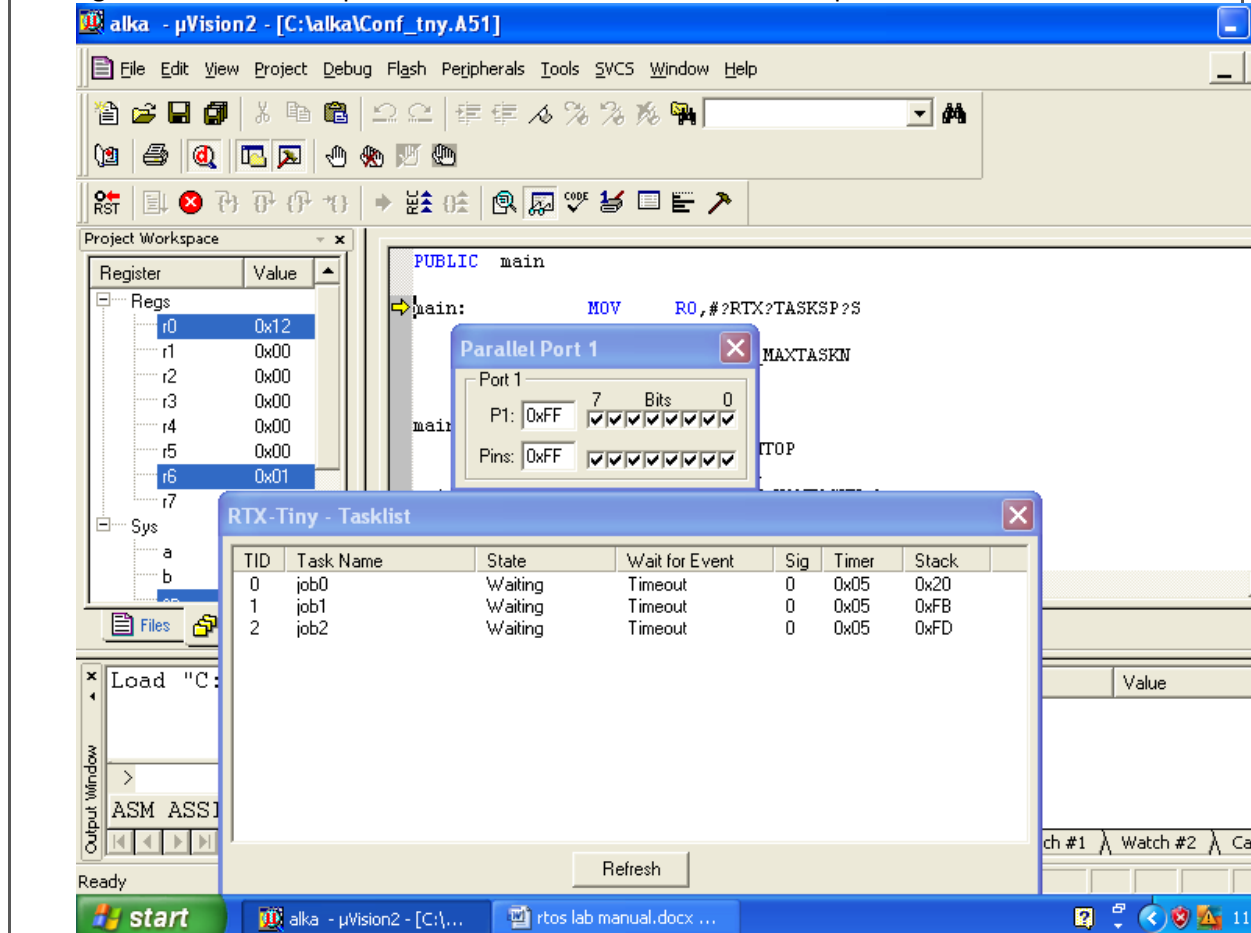
```

    PIN3=0;
    os_wait (K_TMO, 15, 0); // wait for 150 ms
    PIN3=1;
    os_wait (K_TMO, 15, 0); // wait for 150 ms
}
}

```

Result:

Figure 1 shows the snapshot of the simulated result on C51 compiler



Program 2: Write a program for creation of tasks for toggle the port 0, port 1 and port 2 of 89C52 microcontroller using TINY OS

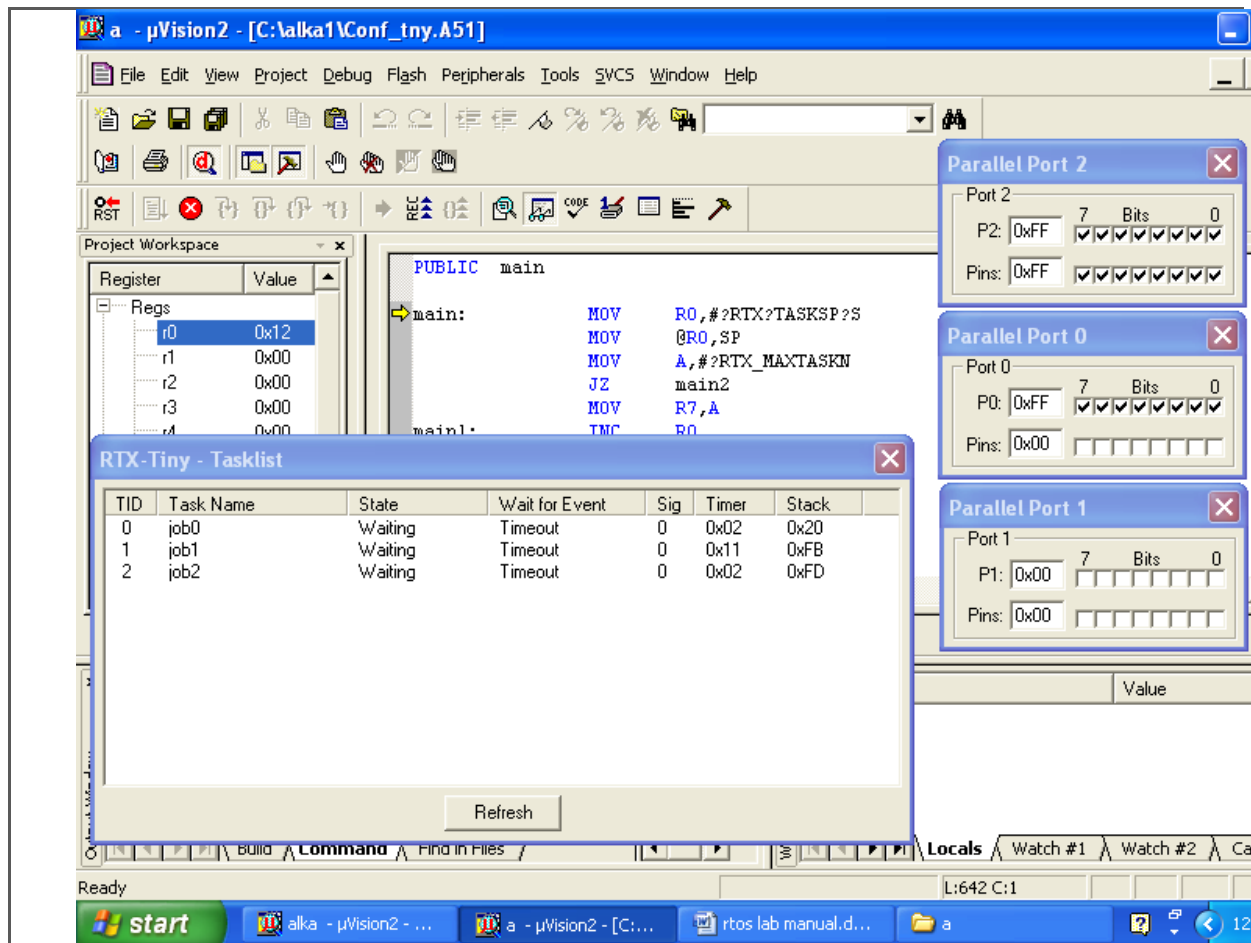
Code:

```

/*****
/* task creation using tiny OS*/
*****/
#include <rtx51tiny.h> //header file for tiny os
#include <reg52.h> // header file for 8051
/*****
/*      task 0      */
*****/
job0 () _task_ 0
{
    os_create_task (1); // creation of task 1
    os_create_task (2); // creation of task 2
    while(1)
    {
        P0=0X00;
        os_wait (K_TMO, 5, 0); // wait for 50ms
        P0= 0Xff;
        os_wait (K_TMO, 5, 0); // wait for 50ms
    }
}
/*****
/*      task 1      */
*****/
job1 () _task_ 1
{
    while(1)
    {
        P1=0X00;
        os_wait (K_TMO, 30, 0); //wait for 300ms
        P1 = 0Xff;
        os_wait (K_TMO, 30, 0); // wait for 300ms
    }
}
/*****
/*      task 2      */
*****/
job2 () _task_ 2
{
    while(1)
    {
        P2=0X00;
        os_wait (K_TMO, 15, 0); // wait for 150 ms
        P2=0Xff;
        os_wait (K_TMO, 15, 0); // wait for 150 ms
    }
}
}
```

Result:

Figure 2 shows the snapshot of the simulated result on C51 compiler



Program 3: Write a program for creation of tasks and sending signal to another task using TINY OS**Code:**

```
/*
*****
/* task creation & sending signal to another task using tiny os&*/
*****
#include <rtx51tiny.h>// header file for tiny rtos
#include <reg52.h>// header file for 89c52

sbit PIN1=P1^0;//bit declaration
sbit PIN2=P1^1;
sbit PIN3=P1^2;
sbit PIN4=P1^3;
unsigned long count=0;//counter declaration

/*
*****
/*
          task 0
          */
*****
job0 () _task_ 0
{
    os_create_task (1);//create task 1
    os_create_task (2);// create task 2
    while(1)
    {
        PIN1=0;
        os_wait (K_TMO, 5, 0);//wait for 5 system ticks
        PIN1=1;
        os_wait (K_TMO, 5, 0);// wait for 5 system ticks
    }
}

/*
*****
/*
          task 1
          */
*****
job1 () _task_ 1
{
    while(1)
    {
        PIN2=0;
        os_wait (K_TMO, 30, 0);// wait for 30 system ticks
        PIN2=1;
        os_wait (K_TMO, 30, 0);// wait for 30 system ticks
        os_send_signal (2);// sens signal to task 2
    }
}

/*
*****
/*
          task 2
          */
*****
job2 () _task_ 2
```

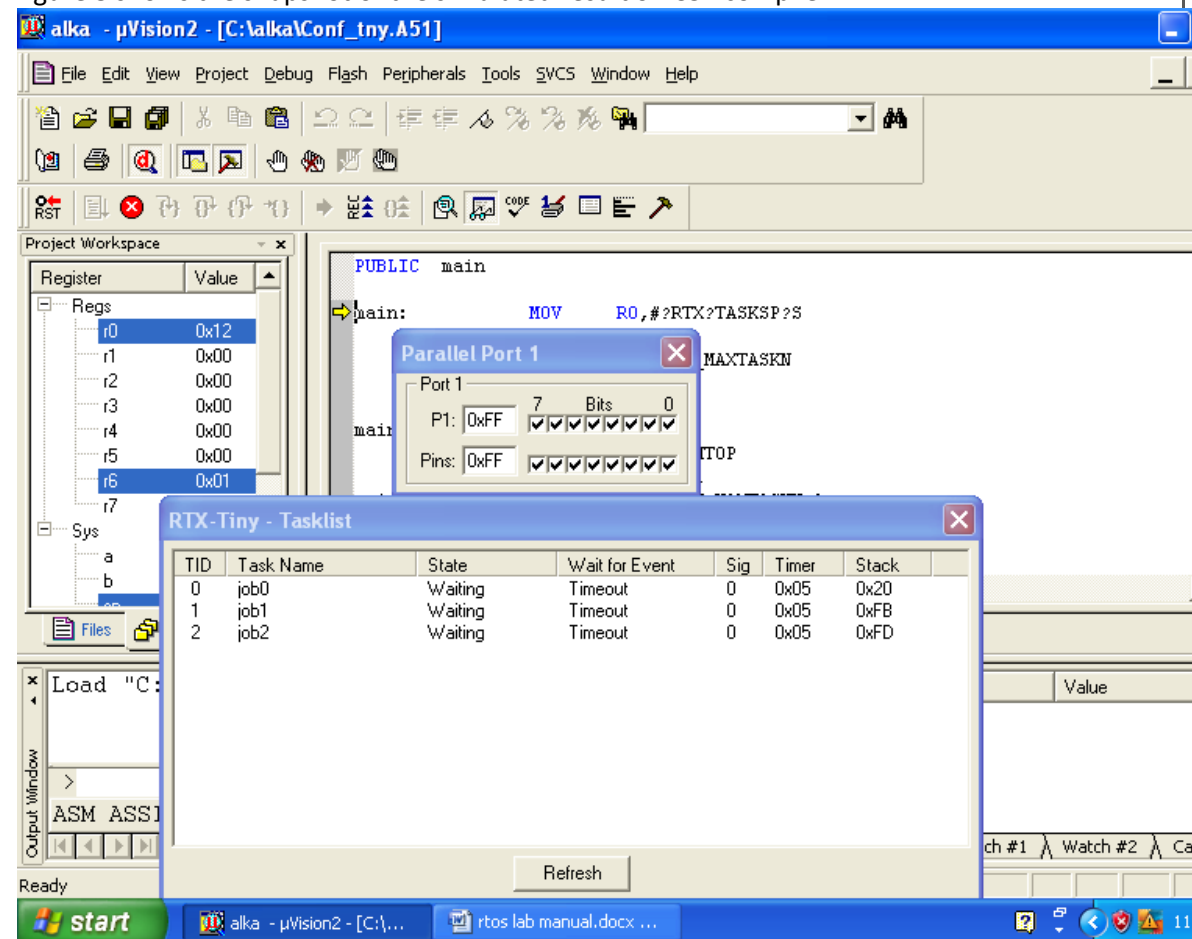
```

{
    while(1)
    {
        os_wait (K_SIG, 0, 0);
        count++;
        if(count==200)
        {
            count=0;
            PIN3=~PIN3;
        }
    }
}

```

Result:

Figure 3 shows the snapshot of the simulated result on C51 compiler



Program 4: Write a program for creation of tasks then made another task as pended task and delete the main task TINY OS

Code:

```

/*****
/*          task creation and deletion          */
*****/
#include <rtx51tny.h> //header file for tiny os
#include <reg52.h> // header filr for 89C52
sbit PIN1=P1^0; // bit declaration
sbit PIN2=P1^1;
sbit PIN3=P1^2;
sbit PIN4=P1^3;

unsigned long count=0; // counter declaration

/*****
/*          task 0          */
*****/
job0 () _task_ 0
{
    os_create_task (1); // create task 1
    os_create_task (2); // createtask 2
    while(1)
    {
        PIN1=0;
        os_wait (K_TMO, 5, 0); // wait for 5 system ticks
        PIN1=1;
        os_wait (K_TMO, 5, 0); // wait for 5 system ticks
    }
}

/*****
/*          task 1          */
*****/
job1 () _task_ 1
{
    while(1)
    {
        PIN2=0;
        os_wait (K_TMO, 10, 0);
        PIN2=1;
        os_wait (K_TMO, 10, 0);
        os_delete_task(0); // deletion of task 0
        os_send_signal (2); // sending signal to task 2
    }
}

/*****
/*          task 2          */
*****/

```

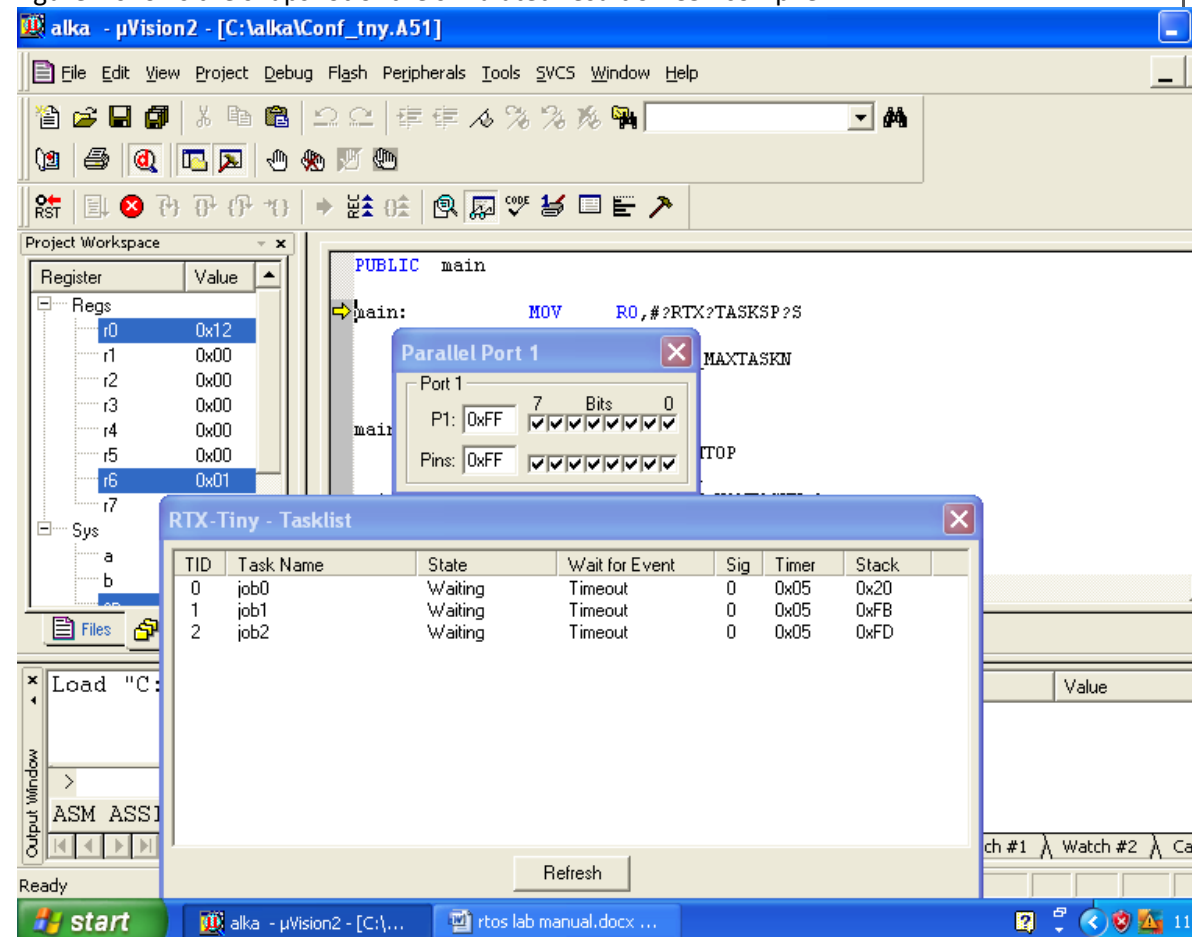
```

/*****
job2 () _task_ 2
{
    while(1)
    {
        os_wait (K_SIG, 0, 0); // wait for signal
        count++;
        if(count==200)
        {
            count=0;
            PIN3=~PIN3;
        }
    }
}
*****/

```

Result:

Figure 4 shows the snapshot of the simulated result on C51 compiler



Program 5: Write a program for suspended task for TINY OS

Code:

```

/*****
/*          suspended task          */
*****/
#include <rtx51tiny.h> //header file for tiny os
#include <reg52.h> // header filr for 89C52
sbit PIN1=P1^0; // bit declaration
sbit PIN2=P1^1;
sbit PIN3=P1^2;
sbit PIN4=P1^3;

unsigned long count=0; // counter declaration

/*****
/*          task 0          */
*****/
job0 () _task_ 0
{
    os_create_task (1); // create task 1
    os_create_task (2); // createtask 2
    while(1)
    {
        PIN1=0;
        os_wait (K_TMO, 5, 0); // wait for 5 system ticks
        PIN1=1;
        os_wait (K_TMO, 5, 0); // wait for 5 system ticks
    }
}
/*****
/*          task 1          */
*****/
job1 () _task_ 1
{
    while(1)
    {
        PIN2=0;
        os_wait (K_TMO, 10, 0);
        PIN2=1;
        os_wait (K_TMO, 10, 0);
        os_delete_task(0); // deletion of task 0
        os_send_signal (2); // sending signal to task 2
    }
}
/*****
/*          task 2          */
*****/
job2 () _task_ 2
{
    while(1)

```

```

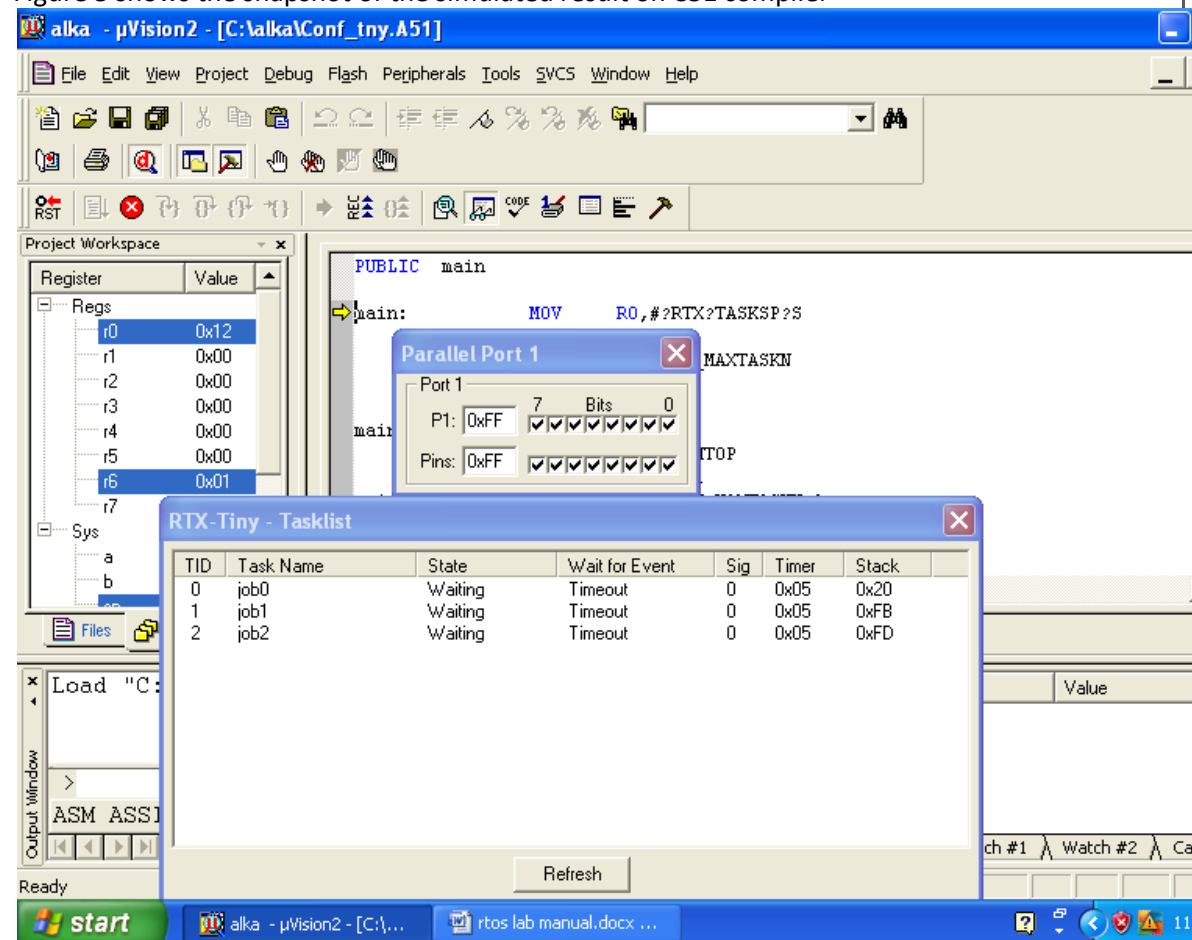
{
    os_wait (K_SIG, 0, 0); // wait for signal

    count++;
    if(count==200)
    {
        count=0;
        PIN3=~PIN3;
    }
}
}
}

```

Result:

Figure 5 shows the snapshot of the simulated result on C51 compiler



Program 6: Write a program for pended suspended delayed task for TINY OS

Code:

```

/*****
/*          pended suspended delayed task          */
*****/
#include <rtx51tiny.h>//header file for tiny os
#include <reg52.h>// header file for 89C52
sbit PIN1=P1^0;// bit declaration
sbit PIN2=P1^1;
sbit PIN3=P1^2;

unsigned long count=0;// counter declaration

/*****
/*          task 0          */
*****/
job0 () _task_ 0
{
os_create_task (1);// create task 1
os_create_task (2);// create task 2
while(1)
{

PIN1=0;
os_wait (K_TMO, 5, 0);// wait for 5 system ticks
PIN1=1;
os_wait (K_TMO, 5, 0);// wait for 5 system ticks
}
}
/*****
/*          task 1          */
*****/

job1 () _task_ 1
{
while(1)
{
PIN2=0;
os_wait (K_TMO, 10, 0);
PIN2=1;
os_wait (K_TMO, 10, 0);

os_send_signal (2);// sending signal to task 2
}
}
/*****
/*          task 2          */
*****/

job2 () _task_ 2
{
while(1)
{

```

```

os_wait (K_SIG, 0, 0); // wait for signal

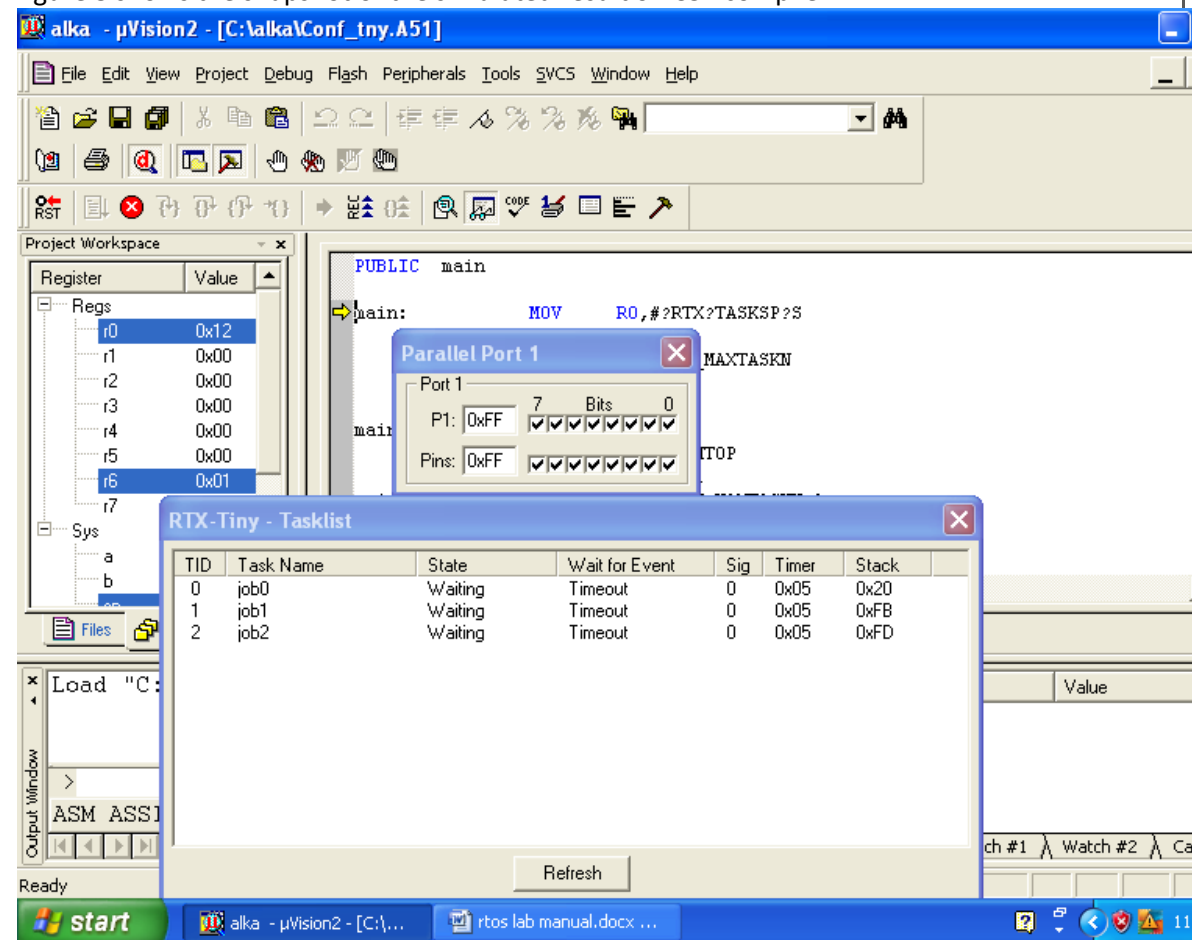
count++;
if(count==200)
{
    count=0;
    PIN3=~PIN3;
    os_delete_task(2); // suspend task 2

    os_wait(K_TMO,10,0); //delayed by 10 system ticks
    os_create_task(2); //again start the task 0
}
}
}

```

Result:

Figure 3 shows the snapshot of the simulated result on C51 compiler

**Program 7: Write a program to understand DEADLOCK in TINY OS****Code:**

```

/*****
/*                                Deadlock                                */
*/

```

```

/*****
#include <rtx51tiny.h>//header file for tiny os
#include <reg52.h>// header filr for 89C52
sbit PIN1=P1^0;// bit declaration
sbit PIN2=P1^1;
sbit PIN3=P1^2;
sbit PIN4=P1^3;

unsigned long count=0;// counter declaration

/*****
/*                                     task 0                                     */
/*****
job0 () _task_ 0
{
os_create_task (1);// create task 1
os_create_task (2);// createtask 2
os_create_task (3);// createtask 3

while(1)
{
PIN1=0;
os_wait (K_TMO, 5, 0);// wait for 5 system ticks
PIN1=1;
os_wait (K_TMO, 5, 0);// wait for 5 system ticks
os_send_signal (1);// sending signal to task 1

}
}
/*****
/*                                     task 1                                     */
/*****
job1 () _task_ 1
{
os_wait (K_SIG, 0, 0);

while(1)
{
PIN2=0;
os_wait (K_TMO, 10, 0);
PIN2=1;
os_wait (K_TMO, 10, 0);

os_send_signal (2);// sending signal to task 2
}
}
/*****
/*                                     task 2                                     */
/*****
job2 () _task_ 2
{

```

```

    os_wait (K_SIG, 10, 0);

while(1)
{
    os_wait (K_SIG, 0, 0); // wait for signal

    count++;
    if(count==200)
    {
        count=0;
        PIN3=~PIN3;
    }
    os_send_signal (3); // sending signal to task 3

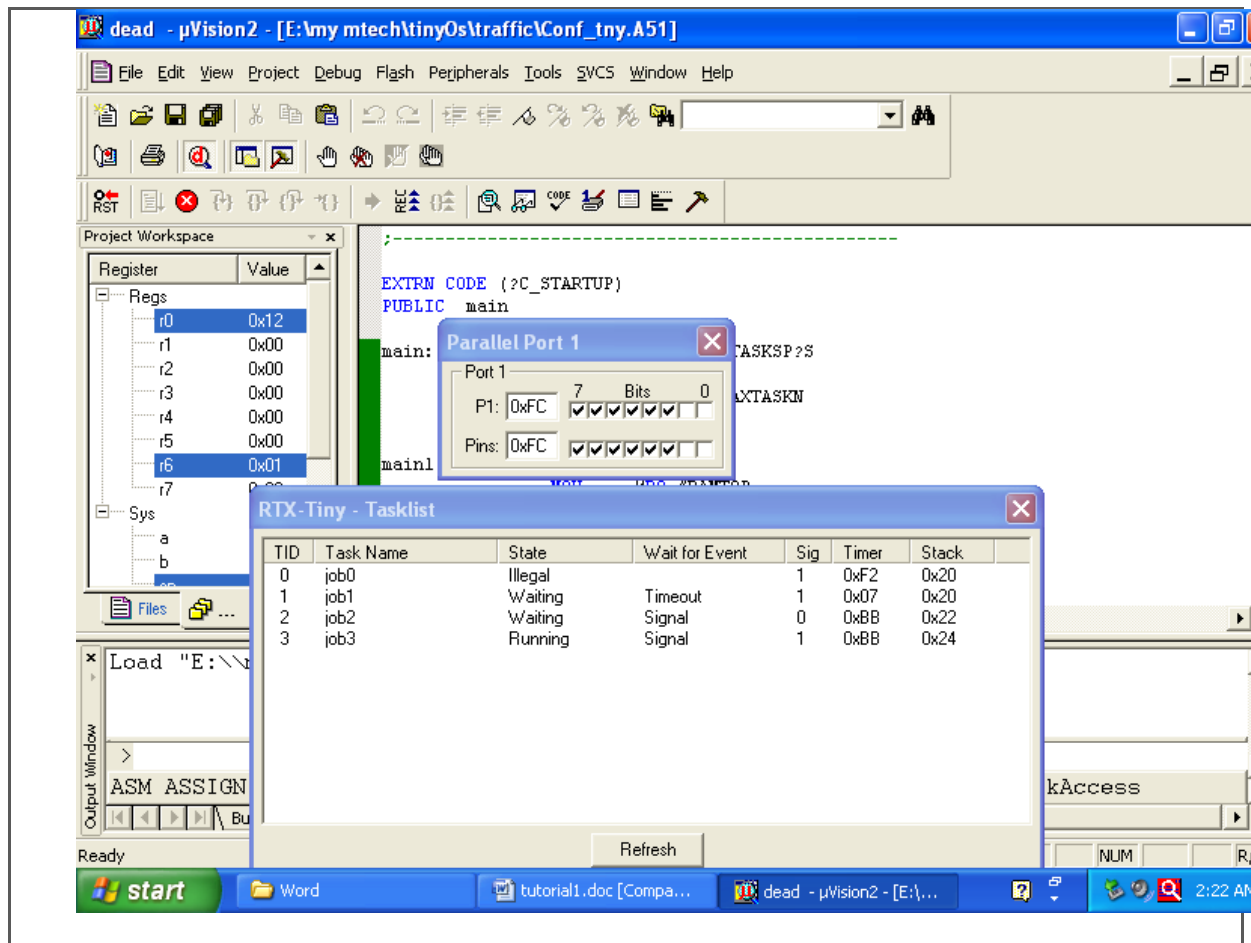
}
}
/*****
/*                                task 3                                */
*****/
job2 () _task_ 3
{
    os_wait (K_SIG, 0, 0); // wait for signal

while(1)
{
    count++;
    if(count==100)
    {
        count=0;
        PIN4=~PIN4;
    }
    os_send_signal (0); // sending signal to task 0
    os_delete_task(0); // delete task 0 , formation of deadlock
}
}

```

Result:

Figure 7 shows the snapshot of the simulated result on C51 compiler



Program 8: Write a program for traffic light control using TINY OS**Code:**

```

/*****
*   The Program for Controlling Traffic at a Square                               *
*   It controls the traffic through 4 directions at 4 poles namely the North, South, East & West and *
*   also controls the walk light for crossing the road at zebra crossing           *
*                                                                                   *
*****/

#include<rtx51tny.h> //Declaring TINY Header File
#include<reg52.h>

sbit g_north=P1^0; //Define the pins for connections
sbit y_north=P1^1;
sbit r_north=P1^2;
sbit w_north=P1^6;
sbit g_south=P3^3;
sbit y_south=P3^4;
sbit r_south=P3^5;
sbit w_south=P3^7;
sbit g_east=P1^5;
sbit y_east=P1^4;
sbit r_east=P1^3;
sbit w_east=P1^7;
sbit g_west=P3^2;
sbit y_west=P3^1;
sbit r_west=P3^0;
sbit w_west=P3^6;

/* This task initializes all the lights all the lights except red are switched OFF*/
/* Only one thing is to be kept in mind that do not give transition on the right as it increases the chances of accidents. Pass
the transition either on Left or OppositeHere the transition is from NORTH => SOUTH => EAST => WEST*/
/* The walk Function is based on the signals on the RIGHT ie if the Signal on the RIGHT is green then you can use the zebra
crossing to cross the road*/

traffic () _task_ 0
{
    while(1)
    {
        g_north= 0;
        y_north= 0;
        r_north= 0;
        g_south= 0;
        y_south= 0;
        r_south= 0;
        g_east= 0;
        y_east= 0;
        r_east= 0;
        g_west= 0;
        y_west= 0;
        r_west= 0;
        os_create_task(1);
        os_send_signal(1);
        r_north= 1;
        r_south= 1;
        r_east= 1;
        r_west= 1;
    }
}

```

```

        w_north= 0;
        w_south= 0;
        w_east= 0;
        w_west= 0;
    }
}
/* This task controls the Northern side signals */

north() _task_1
{
    while(1)
    {
        os_wait (K_SIG, 0, 0);
        os_delete_task(4);
/*in all the tasks the first step is to delete the task from which it has been called*/
        os_delete_task(0);
        r_north= 0;
        g_north=1;
        w_west=1;
        y_north=0;
        os_wait(K_TMO, 250000, 0);
        g_north=0;
        y_north= 1;
        os_wait(K_TMO, 50000, 0);
        w_west=0;
        y_north= 0;
        r_north= 1;
        os_create_task (2);
        os_send_signal (2);
    }
}

/* This task controls the Southern side signals */
south() _task_2
{
    while(1)
    {
        os_wait (K_SIG,0, 0);
        os_delete_task(1);

        r_south= 0;
        g_south= 1;
        w_east= 1;
        os_wait(K_TMO, 250000, 0);
        g_south= 0;
        y_south= 1;
        os_wait(K_TMO, 50000, 0);
        w_east= 0;
        y_south= 0;
        r_south= 1;
        os_create_task (3);
        os_send_signal(3);
    }
}

/* This task controls the East side signals */

east() _task_3
{
    while(1)
    {
        os_wait (K_SIG,0, 0);
        os_delete_task(2);
        r_east= 0;

```

```

        g_east = 1;
        w_north = 1;
        os_wait(K_TMO, 250000, 0);
        g_east = 0;
        y_east = 1;
        os_wait(K_TMO, 50000, 0);
        w_north = 0;
        y_east = 0;
        r_east = 1;
        os_create_task(4);
        os_send_signal(4);
    }
}

/* This task controls the West side signals */

west() _task_4
{
    while(1)
    {
        os_wait(K_SIG, 0, 0);
        os_delete_task(3);
        r_west = 0;
        g_west = 1;
        w_south = 1;
        os_wait(K_TMO, 250000, 0);
        g_west = 0;
        y_west = 1;
        os_wait(K_TMO, 50000, 0);
        w_south = 0;
        y_west = 0;
        r_west = 1;
        os_create_task(1);
        os_send_signal(1);
    }
}

```

Result: