

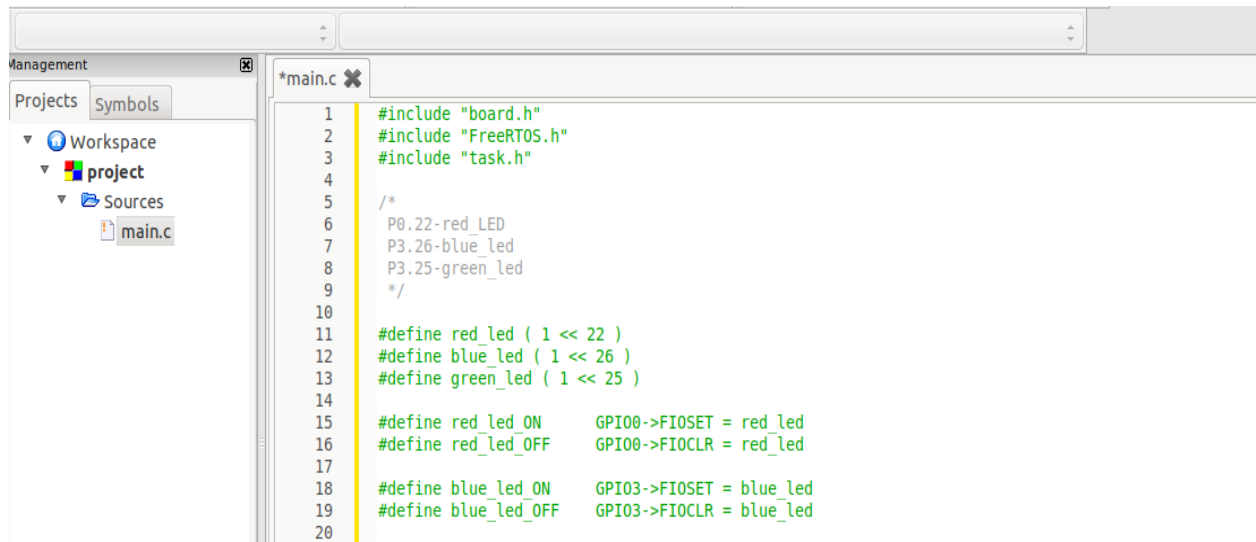
TEST-2
ESE – 3025
EMBEDDED REAL TIME OPERATING SYSTEMS

Submitted to :
Takis Zourntos

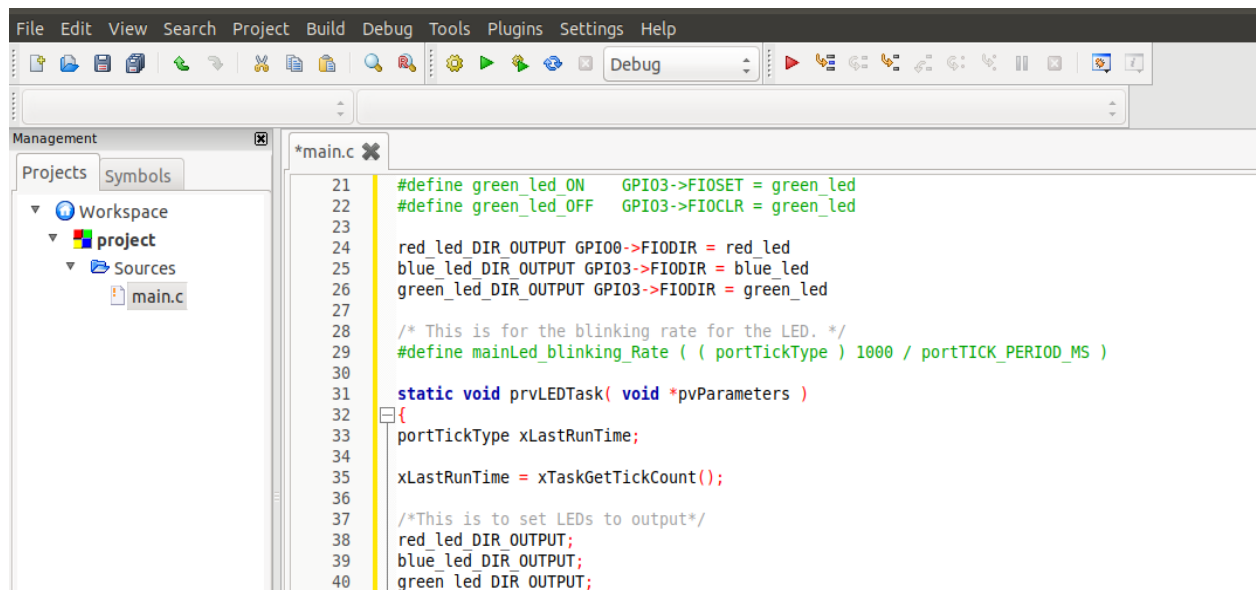
Submitted by:
Shweta(747925)

A. Design a code for task function for blinking red, green and blue LED in sequence with no overlapping colors with output as red LED on and then pause, Green led on and then pause, Blue led on and then pause. Consider only one task function while writing the code.

Solution:



```
1  #include "board.h"
2  #include "FreeRTOS.h"
3  #include "task.h"
4
5  /*
6   P0.22-red_LED
7   P3.26-blue_led
8   P3.25-green_led
9  */
10
11 #define red_led ( 1 << 22 )
12 #define blue_led ( 1 << 26 )
13 #define green_led ( 1 << 25 )
14
15 #define red_led_ON      GPIO0->FIOSET = red_led
16 #define red_led_OFF    GPIO0->FIOCLR = red_led
17
18 #define blue_led_ON     GPIO3->FIOSET = blue_led
19 #define blue_led_OFF    GPIO3->FIOCLR = blue_led
20
```



```
21 #define green_led_ON    GPIO3->FIOSET = green_led
22 #define green_led_OFF   GPIO3->FIOCLR = green_led
23
24 red_led_DIR OUTPUT GPIO0->FIODIR = red_led
25 blue_led_DIR OUTPUT GPIO3->FIODIR = blue_led
26 green_led_DIR OUTPUT GPIO3->FIODIR = green_led
27
28 /* This is for the blinking rate for the LED. */
29 #define mainLed_blinking_Rate ( ( portTickType ) 1000 / portTICK_PERIOD_MS )
30
31 static void prvLEDTask( void *pvParameters )
32 {
33     portTickType xLastRunTime;
34
35     xLastRunTime = xTaskGetTickCount();
36
37     /*This is to set LEDs to output*/
38     red_led_DIR OUTPUT;
39     blue_led_DIR OUTPUT;
40     green_led_DIR OUTPUT;

```

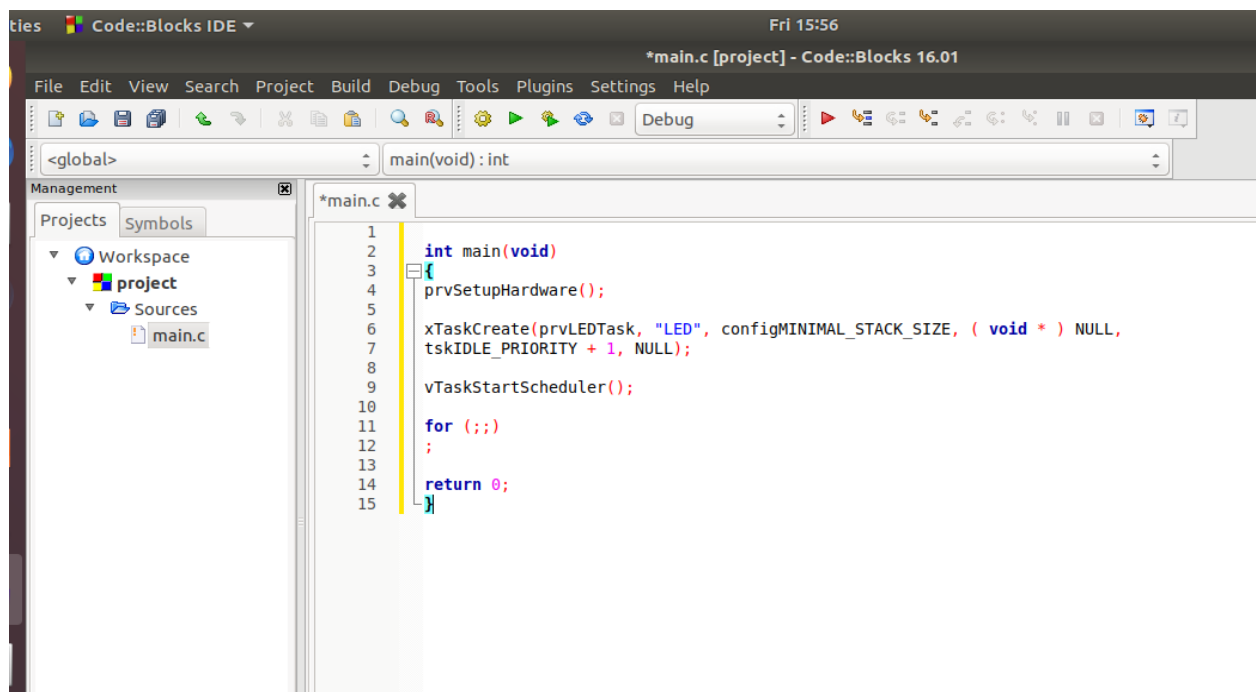
```

41
42  /*This is to off all leds in beginning*/
43  red_led_OFF;
44  blue_led_OFF;
45  green_led_OFF;
46
47  for(;;)
48  {
49      red_led_ON;
50      vTaskDelayUntil( &xLastRunTime, mainLed_blinking_Rate );
51      red_led_OFF;
52      vTaskDelayUntil( &xLastRunTime, mainLed_blinking_Rate );
53
54      blue_led_ON;
55      vTaskDelayUntil( &xLastRunTime, mainLed_blinking_Rate );
56      blue_led_OFF;
57      vTaskDelayUntil( &xLastRunTime, mainLed_blinking_Rate );
58
59      green_led_ON;
60      vTaskDelayUntil( &xLastRunTime, mainLed_blinking_Rate );
61      green_led_OFF;
62      vTaskDelayUntil( &xLastRunTime, mainLed_blinking_Rate );
63  }
64  }

```

B. Write a code showing main function of your code.

Solution-



The screenshot shows the Code::Blocks IDE interface. The top menu bar includes File, Edit, View, Search, Project, Build, Debug, Tools, Plugins, Settings, and Help. The toolbar contains various icons for file operations, compilation, and debugging. The status bar at the top right indicates 'Fri 15:56' and '*main.c [project] - Code::Blocks 16.01'. The left sidebar shows the 'Management' panel with 'Projects' and 'Symbols' tabs. Under 'Projects', a tree view shows 'Workspace' containing 'project', which contains 'Sources' with the file 'main.c'. The main editor window displays the code for 'main.c' with the following content:

```

1
2  int main(void)
3  {
4      prvSetupHardware();
5
6      xTaskCreate(prvLEDTask, "LED", configMINIMAL_STACK_SIZE, ( void * ) NULL,
7      tskIDLE_PRIORITY + 1, NULL);
8
9      vTaskStartScheduler();
10
11     for (;;)
12     ;
13
14     return 0;
15 }

```

C. Describe the operation of your code in terms of the scheduler. What sort of policy is used by the scheduler?

Solution- Here, the scheduler is using preemptive policy which is based on priority. This is the policy which is used by freeRTOS by default. So, at first higher priority task will complete its execution .After that lower priority task will be executed. A task can be blocked if a task having priority more than that comes in ready state. In this code, LED task's priority is more than that of idle task ,so it gets executed whenever it comes to ready state.