HW#6

Name: Mrinmoy Sarkar
Dept.:ECE
Course: ECEN-621

C code for LEDShow :

```c
#include "msp.h"


#define LED_RED BIT0
#define LED_R BIT0
#define LED_G BIT1
#define LED_B BIT2

#define B0ON (P1->OUT |= LED_RED)
#define B0OFF (P1->OUT &= ~LED_RED)
#define B1ON (P2->OUT |= LED_R)
#define B1OFF (P2->OUT &= ~LED_R)
#define B2ON (P2->OUT |= LED_G)
#define B2OFF (P2->OUT &= ~LED_G)
#define B3ON (P2->OUT |= LED_B)
#define B3OFF (P2->OUT &= ~LED_B)

#define TIMER_PERIOD 24000000  // equivalent to 500ms

typedef struct task {
  int state; // Current state of the task
  unsigned long period; // Rate at which the task should tick
  unsigned long elapsedTime; // Time since task's previous tick
  int (*TickFct)(int); // Function to call for task's tick
} task;

task tasks[2];

const unsigned char tasksNum = 2;
const unsigned long tasksPeriodGCD = 500;
const unsigned long periodBlinkLED = 1500;
const unsigned long periodThreeLEDs = 500;

void SysTick_Init(void);
void initPorts(void);

enum BL_States { BL_SMStart, BL_s1 };
int TickFct_BlinkLED(int state);

int TickFct_ThreeLEDs(int state);
enum TL_States { TL_SMStart, TL_s1, TL_s2, TL_s3 };

void main(void)
{
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;
    unsigned char i=0;
    tasks[i].state = BL_SMStart;
    tasks[i].period = periodBlinkLED;
    tasks[i].elapsedTime = tasks[i].period;
    tasks[i].TickFct = &TickFct_BlinkLED;
    ++i;
    tasks[i].state = TL_SMStart;
    tasks[i].period = periodThreeLEDs;
    tasks[i].elapsedTime = tasks[i].period;
    tasks[i].TickFct = &TickFct_ThreeLEDs;
    initPorts();
    SysTick_Init();
    __enable_irq();

    while(1) {
     __sleep();
    }
}

void SysTick_Handler(void){
    unsigned char i;
      for (i = 0; i < tasksNum; ++i) { // Heart of the scheduler code
         if ( tasks[i].elapsedTime >= tasks[i].period ) { // Ready
            tasks[i].state = tasks[i].TickFct(tasks[i].state);
            tasks[i].elapsedTime = 0;
         }
         tasks[i].elapsedTime += tasksPeriodGCD;
      }
}

void SysTick_Init(void)
{
     while(PCM->CTL1 & PCM_CTL1_PMR_BUSY);
     PCM->CTL0 = PCM_CTL0_KEY_VAL | PCM_CTL0_AMR_1;
     while(PCM->CTL1 & PCM_CTL1_PMR_BUSY);
     FLCTL->BANK0_RDCTL = (FLCTL->BANK0_RDCTL & ~(FLCTL_BANK0_RDCTL_WAIT_MASK)) |
FLCTL_BANK0_RDCTL_WAIT_1;
     FLCTL->BANK1_RDCTL = (FLCTL->BANK1_RDCTL & ~(FLCTL_BANK1_RDCTL_WAIT_MASK)) |
FLCTL_BANK1_RDCTL_WAIT_1;
     CS->KEY = CS_KEY_VAL;
     CS->CTL0 |= CS_CTL0_DCORSEL_5;
     CS->KEY = 0;

     SysTick->CTRL |= SysTick_CTRL_CLKSOURCE_Msk | SysTick_CTRL_ENABLE_Msk;
     SysTick->LOAD = TIMER_PERIOD;
     SysTick->VAL = 0;
     SysTick->CTRL |= SysTick_CTRL_TICKINT_Msk;

}

void initPorts(void)
{
    P1->DIR = LED_RED;
    P1->OUT = 0x00;
    P2->DIR = LED_R|LED_G|LED_B;
    P2->OUT = 0x00;
}

int TickFct_BlinkLED(int state) {
    static int flag = 0;
  switch(state) { // Transitions
     case BL_SMStart: // Initial transition
        B0OFF; // Initialization behavior
        state = BL_s1;
        break;
     case BL_s1:
        state = BL_s1;
```

```
         break;
      default:
         state = BL_SMStart;
    } // Transitions

   switch(state) { // State actions
      case BL_s1:
         if(flag)B0ON;
         else B0OFF;
         flag = !flag;
         break;
      default:
         break;
   } // State actions
   return state;
}

int TickFct_ThreeLEDs(int state) {
   switch(state) { // Transitions
      case TL_SMStart: // Initial transition
         state = TL_s1;
         break;
      case TL_s1:
         state = TL_s2;
         break;
      case TL_s2:
         state = TL_s3;
         break;
      case TL_s3:
         state = TL_s1;
         break;
      default:
         state = TL_SMStart;
    } // Transitions

   switch(state) { // State actions
      case TL_s1:
         B1ON; B2OFF; B3OFF;
         break;
      case TL_s2:
         B1OFF; B2ON; B3OFF;
         break;
      case TL_s3:
         B1OFF; B2OFF; B3ON;
         break;
      default:
         break;
   } // State actions
   return state;
}
```

C code for LEDShow+FlashLED :

```
#include "msp.h"
```

```
#define LED_RED BIT0
#define LED_R BIT0
#define LED_G BIT1
#define LED_B BIT2
#define LED_BOOSTER_GREEN BIT4
#define S1 BIT1
#define A0 ((P5->IN & S1) == 0x00)

#define B0ON (P1->OUT |= LED_RED)
#define B0OFF (P1->OUT &= ~LED_RED)
#define B1ON (P2->OUT |= LED_R)
#define B1OFF (P2->OUT &= ~LED_R)
#define B2ON (P2->OUT |= LED_G)
#define B2OFF (P2->OUT &= ~LED_G)
#define B3ON (P2->OUT |= LED_B)
#define B3OFF (P2->OUT &= ~LED_B)
#define B4ON (P2->OUT |= LED_BOOSTER_GREEN)
#define B4OFF (P2->OUT &= ~LED_BOOSTER_GREEN)

#define TIMER_PERIOD 2400000  // equivalent to 50ms



typedef struct task {
  int state; // Current state of the task
  unsigned long period; // Rate at which the task should tick
  unsigned long elapsedTime; // Time since task's previous tick
  int (*TickFct)(int); // Function to call for task's tick
} task;

task tasks[3];

const unsigned char tasksNum = 3;
const unsigned long tasksPeriodGCD = 50;
const unsigned long periodBlinkLED = 1500;
const unsigned long periodThreeLEDs = 500;
const unsigned long periodFlashLED = 50;

void SysTick_Init(void);
void initPorts(void);

enum BL_States { BL_SMStart, BL_s1 };
int TickFct_BlinkLED(int state);

enum TL_States { TL_SMStart, TL_s1, TL_s2, TL_s3 };
int TickFct_ThreeLEDs(int state);

enum FL_States { FL_SMStart, FL_s1, FL_s2, FL_s3, FL_s4 };
int TickFct_FlashLED(int state);


void main(void)
{
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;
    unsigned char i=0;
    tasks[i].state = BL_SMStart;
    tasks[i].period = periodBlinkLED;
    tasks[i].elapsedTime = tasks[i].period;
    tasks[i].TickFct = &TickFct_BlinkLED;
    ++i;
    tasks[i].state = TL_SMStart;
    tasks[i].period = periodThreeLEDs;
    tasks[i].elapsedTime = tasks[i].period;
    tasks[i].TickFct = &TickFct_ThreeLEDs;
    ++i;
    tasks[i].state = FL_SMStart;
    tasks[i].period = periodFlashLED;
    tasks[i].elapsedTime = tasks[i].period;
    tasks[i].TickFct = &TickFct_FlashLED;

    initPorts();
    SysTick_Init();
    __enable_irq();

    while(1) {
     __sleep();
    }
}


void SysTick_Handler(void){
    unsigned char i;
      for (i = 0; i < tasksNum; ++i) { // Heart of the scheduler code
         if ( tasks[i].elapsedTime >= tasks[i].period ) { // Ready
            tasks[i].state = tasks[i].TickFct(tasks[i].state);
            tasks[i].elapsedTime = 0;
         }
         tasks[i].elapsedTime += tasksPeriodGCD;
      }
}

void SysTick_Init(void)
{
     while(PCM->CTL1 & PCM_CTL1_PMR_BUSY);
     PCM->CTL0 = PCM_CTL0_KEY_VAL | PCM_CTL0_AMR_1;
     while(PCM->CTL1 & PCM_CTL1_PMR_BUSY);
     FLCTL->BANK0_RDCTL = (FLCTL->BANK0_RDCTL & ~(FLCTL_BANK0_RDCTL_WAIT_MASK)) |
FLCTL_BANK0_RDCTL_WAIT_1;
     FLCTL->BANK1_RDCTL = (FLCTL->BANK1_RDCTL & ~(FLCTL_BANK1_RDCTL_WAIT_MASK)) |
FLCTL_BANK1_RDCTL_WAIT_1;
     CS->KEY = CS_KEY_VAL;
     CS->CTL0 |= CS_CTL0_DCORSEL_5;
     CS->KEY = 0;

     SysTick->CTRL |= SysTick_CTRL_CLKSOURCE_Msk | SysTick_CTRL_ENABLE_Msk;
     SysTick->LOAD = TIMER_PERIOD;
     SysTick->VAL = 0;
     SysTick->CTRL |= SysTick_CTRL_TICKINT_Msk;

}


void initPorts(void)
{
    P5->DIR &= ~S1;
    P5->REN = S1;
    P5->OUT = S1;
    P1->DIR = LED_RED;
    P1->OUT = 0x00;
    P2->DIR = LED_R|LED_G|LED_B|LED_BOOSTER_GREEN;
    P2->OUT = 0x00;
}



int TickFct_BlinkLED(int state) {
    static int flag = 0;
  switch(state) { // Transitions
     case BL_SMStart: // Initial transition
        BOOFF; // Initialization behavior
        state = BL_s1;
        break;
     case BL_s1:
        state = BL_s1;
        break;
     default:
        state = BL_SMStart;
   } // Transitions

  switch(state) { // State actions
     case BL_s1:
        if(flag)BOON;
        else BOOFF;
        flag = !flag;
        break;
     default:
        break;
  } // State actions
  return state;
}

int TickFct_ThreeLEDs(int state) {
  switch(state) { // Transitions
     case TL_SMStart: // Initial transition
        state = TL_s1;
        break;
     case TL_s1:
        state = TL_s2;
        break;
     case TL_s2:
        state = TL_s3;
        break;
     case TL_s3:
        state = TL_s1;
        break;
     default:
        state = TL_SMStart;
   } // Transitions

  switch(state) { // State actions
     case TL_s1:
        B1ON; B2OFF; B3OFF;
        break;
     case TL_s2:
        B1OFF; B2ON; B3OFF;
        break;
     case TL_s3:
        B1OFF; B2OFF; B3ON;
        break;
     default:
        break;
  } // State actions
  return state;
}

int TickFct_FlashLED(int state) {
  switch(state) { // Transitions
     case FL_SMStart: // Initial transition
        B4OFF; // Initialization behavior
        state = FL_s1;
        break;
     case FL_s1:
        if(A0)
        {
            state = FL_s2;
        }
        break;
     case FL_s2:
        if(!A0)
        {
          state = FL_s3;
        }
        break;
     case FL_s3:
        if(A0)
        {
          state = FL_s4;
        }
        break;
     case FL_s4:
        if(!A0)
        {
           state = FL_s1;
        }
        break;
     default:
        state = FL_SMStart;
   } // Transitions

  switch(state) { // State actions
     case FL_s1:
        B4OFF;
        break;
     case FL_s2:
        B4ON;
        break;
     case FL_s3:
        B4ON;
        break;
     case FL_s4:
        B4OFF;
        break;
     default:
        break;
  } // State actions
  return state;
}
```