```c
#include "stdint.h"
#include "stdio.h"
#include "stdlib.h"
#include "tm4c123gh6pm.h"
#include "UART.h"
#include "PLL.h"

//pwm def
#define PWM_1_GENB_ACTCMPBD_ONE 0x00000C00  // Set the output signal to 1
#define PWM_1_GENB_ACTLOAD_ZERO 0x00000008  // Set the output signal to 0

#define SYSCTL_RCC_USEPWMDIV    0x00100000  // Enable PWM Clock Divisor
#define SYSCTL_RCC_PWMDIV_M     0x000E0000  // PWM Unit Clock Divisor
#define SYSCTL_RCC_PWMDIV_2     0x00000000  // /2

#define UART_FR_TXFF            0x00000020  // UART Transmit FIFO Full
#define UART_FR_RXFE            0x00000010  // UART Receive FIFO Empty
#define UART_LCRH_WLEN_8        0x00000060  // 8 bit word length
#define UART_LCRH_FEN           0x00000010  // UART Enable FIFOs
#define UART_CTL_UARTEN         0x00000001  // UART Enable

#define blue 0x04
#define red 0x02
#define green 0x08
#define white 0x0E


unsigned char UART7_NonBlockingInChar(void);
unsigned char UART7_InChar(void);
void UART7_OutChar(unsigned char data);
unsigned long UART7_InUDec(void);
void UART7_Init(void);
void UART7_OutString(char *pt);
void UART7_OutUDec(unsigned long n);

void PortE_UART7_Init(void);
// basic functions defined at end of startup.s
void DisableInterrupts(void); // Disable interrupts
void EnableInterrupts(void);  // Enable interrupts
void WaitForInterrupt(void);  // low power mode
void OutCRLF(void);

char value[5];
char string[4];
char buffer[4];
char freq;
unsigned int a,b,c, bright;
//char string [20];
char temp;
unsigned short i;
unsigned long dutyCycle;
unsigned long press;
char msg;
char *test;
char in;

int val;
void slice_str( char * str, char * buffer, unsigned int start, unsigned int end);

void OutCRLF(void){
  UART_OutChar(CR);
  UART_OutChar(LF);
}

void PortF_Init(void){
    volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000020;       // 1) F clock
  delay = SYSCTL_RCGC2_R;             // delay

```

```
70    GPIO_PORTF_LOCK_R   =  0x4C4F434B;  // 2) unlock PortF PF0
71    GPIO_PORTF_CR_R    |=  0x1F;        // allow changes to PF4-0
72    GPIO_PORTF_DEN_R   |=  0x1F;
73    GPIO_PORTF_AMSEL_R &= ~0x1F;        // 3) disable analog function
74    GPIO_PORTF_PCTL_R  &= ~0x000F0F0F;  // 4) GPIO clear bit PCTL
75      //GPIO_PORTF_PCTL_R  |=  0x00055555;
76
77      GPIO_PORTF_DIR_R   &=  ~0x10;        // PF2 output
78
79    GPIO_PORTF_DIR_R   |=  0x0F;        // PF2 output
80    GPIO_PORTF_AFSEL_R &= ~0x1F;        // 6) no alternate function
81    GPIO_PORTF_PUR_R   |=  0x1F;        //     enable weak pull-up on PF4,0
82
83      GPIO_PORTF_IS_R &= ~0x10;
84      GPIO_PORTF_IBE_R &= ~0x10;
85      GPIO_PORTF_IEV_R &= ~0x10;
86
87      GPIO_PORTF_ICR_R = 0x10;
88      GPIO_PORTF_IM_R |=0x10;
89
90    NVIC_PRI7_R = (NVIC_PRI7_R&0xFF0FFFFF)|0x00400000; // (g) priority 2
91    NVIC_EN0_R = 0x40000000;      // (h) enable interrupt 30 in NVIC
92  }
93
94  void PWM1A_Init( uint32_t period, uint32_t duty){ unsigned long volatile delay;
95      //dutyCycle = 1255;                     // duty cycle for red LED
96    SYSCTL_RCGCPWM_R |= 0x02;             // 1) activate PWM1
97      delay = SYSCTL_RCGCGPIO_R;
98      SYSCTL_RCC_R = 0x00100000 |               // 3) use PWM divider
99                   (SYSCTL_RCC_R & (~0x000E0000)); // ~ might mess up
100     PWM1_3_CTL_R             = 0x00000000;      // 4) re-loading down-counting mode
101   PWM1_3_GENA_R            = 0x000000C8;            // Generator B set-up
102   PWM1_3_LOAD_R       = period - 1;           // 5) cycles needed to count down to 0
103   PWM1_3_CMPA_R          = duty - 1;             // 6) count value when output rises
104   PWM1_3_CTL_R       |= 1;           // 7) start PWM1
105   PWM1_ENABLE_R    |= 0x00000040;           // enable PF2 M1PWM6
106
107 }
108
109 void PWM1A_Duty( uint16_t duty){
110      PWM1_3_CMPA_R = duty -1;
111 }
112 void GPIOPortF_Handler(void){ // called on touch of either SW1 or SW2
113
114   if(GPIO_PORTF_RIS_R&0x10){  // SW2 touch
115     GPIO_PORTF_ICR_R = 0x10;  // acknowledge flag0
116         press++;
117     }
118     OutCRLF();
119 if(press == 1){
120         GPIO_PORTF_DATA_R &= ~0x0E;
121         GPIO_PORTF_DATA_R |= red;
122 }
123     if(press==2){
124         UART_OutString("First Press");
125         UART1_OutString("First Press");
126         GPIO_PORTF_DATA_R &= ~0x0E;
127         GPIO_PORTF_DATA_R |= green;
128
129     }
130     if(press==3){
131         UART_OutString("second Press");
132         UART1_OutString("second Press");
133         GPIO_PORTF_DATA_R &= ~0x0E;
134         GPIO_PORTF_DATA_R |= blue;
135     }
136     if(press==4){
137         UART_OutString("third Press");
138         UART1_OutString("third Press");
```

```
        GPIO_PORTF_DATA_R &= ~0x0E;
        GPIO_PORTF_DATA_R |= white;
        press = 0;
    }
OutCRLF();
} // end of handler




int main(void){
    PLL_Init();
  PortF_Init();
    PortE_UART7_Init();
    UART_Init();
    PortB_UART1_Init();
    PWM1A_Init(256, 80);
    EnableInterrupts();
    GPIO_PORTF_DATA_R = 0x02;
    press = 0;
  while(1){
        /*
        UART1_InString(value, 5);



        if(value[0] == 'b'){
            slice_str(value,buffer,1,3);
            val = atoi(buffer);
            if(val < 0 || val > 255){
                val = 010;
                UART_OutString("Invalid LED value... try again\r\n");
                UART1_OutString("\r\nInvalid value... try again\r\n");
            }
            PWM1A_Duty(val);

            UART_OutString(buffer);
            UART1_OutString(" brightness\r\n");

            UART_OutString(" brightness here\r\n");
        } //end of LED change
        if(value[0] == 'f'){

            slice_str(value,buffer,1,3);

            UART7_OutString(buffer);
            UART7_OutString("\r\n");
            UART1_OutString(" frequency\r\n");
            UART_OutString(buffer);
            UART_OutString(" frequency here\r\n");
        } // end of freq

    */

    } // end of while(1)
} //end of main

void slice_str(char *value, char *buffer, unsigned int start, unsigned int end)
{
    unsigned int j = 0;
        unsigned int i = 0;
    for (i = start; i <= end; i++) {
        buffer[j++] = value[i];
    }
    buffer[j] = 0;
}

void PortE_UART7_Init(void){

```

```
208     SYSCTL_RCGCUART_R |= 0x80; // activate UART7
209     SYSCTL_RCGCGPIO_R |= 0x10;
210     SYSCTL_RCGC2_R |= 0x10; // activate port E
211
212     UART7_CTL_R &= ~UART_CTL_UARTEN;      // disable UART
213     UART7_IBRD_R = 86;                    // IBRD = int(50,000,000 / (16 * 115,200)) =
        int(27.1267)  1152000
214     UART7_FBRD_R = 52;                     // FBRD = int(0.1267 * 64 + 0.5) = 8
215                                           // 8 bit word length (no parity bits, one stop
                                              bit, FIFOs)
216     UART7_LCRH_R = 0x0070;
217     UART7_CTL_R |= UART_CTL_UARTEN;       // enable UART
218     GPIO_PORTE_AFSEL_R |= 0x03;           // enable alt funct on PE1-0
219     GPIO_PORTE_DEN_R |= 0x03;             // enable digital I/O on PE1-0
220                                           // configure PE1-0 as UART
221     GPIO_PORTE_PCTL_R = (GPIO_PORTE_PCTL_R&0xFFFFFF00)+0x00000011;
222     GPIO_PORTE_AMSEL_R &= ~0x03;
223   }
224
225   unsigned char UART7_NonBlockingInChar(void){
226       if((UART7_FR_R&UART_FR_RXFE)==0)
227           return((unsigned char)(UART7_DR_R&0xFF));
228       else
229           return 0;
230   }
231
232   unsigned char UART7_InChar(void){
233     while((UART7_FR_R&UART_FR_RXFE) != 0);
234     return((unsigned char)(UART7_DR_R&0xFF));
235   }
236
237   void UART7_OutChar(unsigned char data){
238     while((UART7_FR_R&UART_FR_TXFF) != 0);
239     UART7_DR_R = data;
240   }
241
242   unsigned long UART7_InUDec(void){
243   unsigned long number=0, length=0;
244   char character;
245     character = UART7_InChar();
246     while(character != CR){
247       if((character>='0') && (character<='9')) {
248         number = 10*number+(character-'0');
249         length++;
250         UART7_OutChar(character);
251       }
252       else if((character==BS) && length){
253         number /= 10;
254         length--;
255         UART7_OutChar(character);
256       }
257       character = UART7_InChar();
258     }
259     return number;
260   }
261
262   void UART7_OutString(char *pt){
263     while(*pt){
264       UART7_OutChar(*pt);
265       pt++;
266     }
267   }
268
269   void UART7_OutUDec(unsigned long n){
270   // This function uses recursion to convert decimal number
271   //   of unspecified length as an ASCII string
272     if(n >= 10){
273       UART7_OutUDec(n/10);
274       n = n%10;
```

```
275     }
276     UART7_OutChar(n+'0'); /* n is between 0 and 9 */
277   }
278
279
280
```