```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 IR  (*((volatile unsigned long *)0x40006040)) //PC4
#define blue 0x04
#define red 0x02
#define green 0x08
#define white 0x0E



// 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 temp;
unsigned short i;
unsigned long dutyCycle;
unsigned long press;
char msg;
char *test;
char in;

unsigned int device;
unsigned char cmd;
unsigned long IR_current_time=0;    // counter tracking IR time
unsigned char IR_busy=0;            // IR busy flag
unsigned char Got_IR=0;             // IR transmission request



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

/*****************************************************************************************
********
    PORT F INIT
*****************************************************************************************
*******/
void PortF_Init(void){
    volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000020;       // 1) F clock
  delay = SYSCTL_RCGC2_R;             // delay
  GPIO_PORTF_LOCK_R   =  0x4C4F434B;  // 2) unlock PortF PF0
  GPIO_PORTF_CR_R    |=  0x1F;        // allow changes to PF4-0
  GPIO_PORTF_DEN_R   |=  0x1F;
  GPIO_PORTF_AMSEL_R &= ~0x1F;        // 3) disable analog function
  GPIO_PORTF_PCTL_R  &= ~0x000F0F0F;  // 4) GPIO clear bit PCTL
    //GPIO_PORTF_PCTL_R  |=  0x00055555;

    GPIO_PORTF_DIR_R   &=  ~0x10;        // PF2 output

  GPIO_PORTF_DIR_R   |=  0x0F;        // PF2 output
  GPIO_PORTF_AFSEL_R &= ~0x1F;        // 6) no alternate function
  GPIO_PORTF_PUR_R   |=  0x1F;        //     enable weak pull-up on PF4,0

    GPIO_PORTF_IS_R &= ~0x10;
    GPIO_PORTF_IBE_R &= ~0x10;
    GPIO_PORTF_IEV_R &= ~0x10;

    GPIO_PORTF_ICR_R = 0x10;
    GPIO_PORTF_IM_R |=0x10;

  NVIC_PRI7_R = (NVIC_PRI7_R&0xFF0FFFFF)|0x00400000; // (g) priority 2
  NVIC_EN0_R = 0x40000000;      // (h) enable interrupt 30 in NVIC
}

/*****************************************************************************************
    PORT B INIT
****************************************************************************************/
/*
void PortB_Init(void){
    unsigned volatile delay;
    SYSCTL_RCGCGPIO_R |= 0x00000002;  // activate clock for Port B
    delay = SYSCTL_RCGCGPIO_R;
    GPIO_PORTB_AMSEL_R &= 0x00;              // disable analog on PB
    GPIO_PORTB_PCTL_R &= ~0xF0000000;   // PCTL GPIO on PB7
    GPIO_PORTB_PCTL_R |= GPIO_PCTL_PB7_M0PWM1;
    GPIO_PORTB_DIR_R |= 0x80;                    // PB0-7 are outputs
    GPIO_PORTB_AFSEL_R |= 0x80;              // enable alt. function on PB7
    GPIO_PORTB_DEN_R |= 0x80;                    // enable digital I/O on PB7

    // PWM FOR PB7
    SYSCTL_RCGCPWM_R |= 1;                       // enable PWM module 0
    delay = SYSCTL_RCGCPWM_R;
    SYSCTL_RCC_R |= 0x00100000;              // start the clock divider
    SYSCTL_RCC_R &= ~0x000C0000;             // 0x1, 80Mhz/4, 20Mhz clock
    PWM0_0_CTL_R &= ~0x01;                       // disable PWM for setup
    PWM0_0_GENB_R = 0x0000080C;              // M0PWM1 output set when reload
    PWM0_0_LOAD_R = 526 - 1;                     // seeting the reload to be 40KHz
    PWM0_0_CMPB_R = 263 - 1;                     // compare the values for low signal
}
*/

/*****************************************************************************************
    PORTC Init for IR LED
****************************************************************************************/
void PortC_Init(void){ volatile unsigned long delay;
    SYSCTL_RCGCGPIO_R |= 0x04;             // 2) activate port C
    GPIO_PORTC_LOCK_R = GPIO_LOCK_KEY;
    delay = SYSCTL_RCGCGPIO_R;             // allow time to finish activating delay here
    GPIO_PORTC_AFSEL_R &= ~0x10;           // enable alt funct on PC4
    GPIO_PORTC_PCTL_R  &= ~0x000F0000;    // configure PC4 as GPIO
    GPIO_PORTC_AMSEL_R &= ~0x10;           // disable analog functionality on PC4
    GPIO_PORTC_DIR_R |= 0x10;              // Output PC4
    GPIO_PORTC_DEN_R |= 0x10;              // enable digital I/O on PC4
    IR = 0;                                // Turn off IR LED
```

```c
}


/*****************************************************************************
********
    IR LED PWM INIT
*****************************************************************************
*******/
void PWM1_A6_Init(uint32_t period, uint32_t duty){ unsigned long volatile delay;

    SYSCTL_RCGCPWM_R |= 0x02;              // 1) activate PWM1
    //SYSCTL_RCGCGPIO_R |=0x01; // PortA
    delay = SYSCTL_RCGCGPIO_R;
    SYSCTL_RCC_R = 0x00100000 |                 // 3) use PWM divider
                (SYSCTL_RCC_R & (~0x000E0000)); // ~ might mess up
    PWM1_1_CTL_R             = 0x00000000;      // 4) re-loading down-counting mode
  PWM1_1_GENA_R             = 0x000000C8;            // Generator B set-up
  PWM1_1_LOAD_R       = period - 1;           // 5) cycles needed to count down to 0
  PWM1_1_CMPA_R          = duty - 1;              // 6) count value when output rises
  PWM1_1_CTL_R       |= 0x00000002;           // 7) start PWM1
  PWM1_ENABLE_R    |= 0x00000002;           // enable PF2 M1PWM6
}
void PWM1_A6_Duty(uint16_t duty){
 PWM1_1_CMPA_R = duty -1;
}

/*****************************************************************************
********
    LED DIM PWM
*****************************************************************************
*******/

void PWM1A_Init( uint32_t period, uint32_t duty){ unsigned long volatile delay;
    //dutyCycle = 1255;                     // duty cycle for red LED

  SYSCTL_RCGCPWM_R |= 0x02;              // 1) activate PWM1
    delay = SYSCTL_RCGCGPIO_R;
      SYSCTL_RCGCGPIO_R     |= 0x01;           // 2) activate port A
  while((SYSCTL_PRGPIO_R&0x01) == 0){};
  GPIO_PORTA_AFSEL_R     |= 0x40;           // enable alt funct on PA6
  GPIO_PORTA_PCTL_R     &= ~0x0F000000;   // configure PA6 as PWM0
  GPIO_PORTA_PCTL_R     |= 0x04000000;
  GPIO_PORTA_AMSEL_R     &= ~0x40;          // disable analog functionality on PA6
  GPIO_PORTA_DEN_R         |= 0x40;           // enable digital I/O on PA6

    SYSCTL_RCC_R = 0x00100000 |                 // 3) use PWM divider
                (SYSCTL_RCC_R & (~0x000E0000)); // ~ might mess up
    PWM1_3_CTL_R             = 0x00000000;      // 4) re-loading down-counting mode
  PWM1_3_GENA_R             = 0x000000C8;            // Generator B set-up
  PWM1_3_LOAD_R       = period - 1;           // 5) cycles needed to count down to 0
  PWM1_3_CMPA_R          = duty - 1;              // 6) count value when output rises
  PWM1_3_CTL_R       |= 1;           // 7) start PWM1
  PWM1_ENABLE_R    |= 0x00000040;           // enable PF2 M1PWM6

}

void PWM1A_Duty( uint16_t duty){
     PWM1_3_CMPA_R = duty -1;
}
/*****************************************************************************
********
    SysTckInit *note* might need (unsigned long period) if nothing
                                    can be passed in the SysTick_Init()in main()
*****************************************************************************
*******/
void SysTick_Init(unsigned long period) {
    NVIC_ST_CTRL_R = 0;          // disable SysTick during setup
    NVIC_ST_RELOAD_R = NVIC_ST_RELOAD_M;// reload value
    NVIC_ST_CURRENT_R = 0;      // any write to current clears it
    //NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0xC0000000; // priority 6
    // enable SysTick with core clock and interrupts
    NVIC_ST_CTRL_R = NVIC_ST_CTRL_ENABLE+NVIC_ST_CTRL_CLK_SRC;
}

/*****************************************************************************************
********
    System Wait for different delays
*****************************************************************************************
*******/
void SysTick_Wait(unsigned long delay){
    volatile unsigned long elapsedTime;
    unsigned long startTime = NVIC_ST_CURRENT_R;
    do{
        elapsedTime = (startTime-NVIC_ST_CURRENT_R)&0x00FFFFFF;
    }
    while(elapsedTime <= delay);
} // end of systick_wait

void SysTick_Wait1ms(uint32_t delay){
    uint32_t i;
    for( i = 0; i<delay; i++)
    {
        SysTick_Wait(80000);
}
    }// systick_wait1ms

void SysTick_Wait800us( uint32_t delay){
    uint32_t i;
    for( i = 0; i<delay; i++)
    {
        SysTick_Wait(64000);
}
    }// systick_wait800us

void SysTick_Wait450us( uint32_t delay){
    uint32_t i;
    for( i = 0; i<delay; i++)
    {
        SysTick_Wait(36000);
}
    }// systick_wait450us

void SysTick_Wait900us( uint32_t delay){
    uint32_t i;
    for( i = 0; i<delay; i++)
    {
        SysTick_Wait(72000);
}
    }// systick_wait800us


/////////////////////////////////////////////////////////////////////////////////////////
////////
/*****************************************************************************************
********
    Data frame function
*****************************************************************************************
*******/
/*
//device is 2bits, cmd is 3bits
void data_frame(uint32_t device, uint32_t cmd){
    UART_OutString("switch here");
     UART_OutUDec(cmd);
            switch(cmd){
                case 0 : UART_OutString("found it");

                case 1 : GPIO_PORTA_DATA_R = device | 0x001;

```

```c
                case 2 : GPIO_PORTA_DATA_R = device | 0x010;

                case 3 : GPIO_PORTA_DATA_R = device | 0x011;

                case 4 : GPIO_PORTA_DATA_R = device | 0x100;

                case 5 : GPIO_PORTA_DATA_R = device | 0x101;

                case 6 : GPIO_PORTA_DATA_R = device | 0x110;

                case 7 : GPIO_PORTA_DATA_R = device | 0x111;

                default: GPIO_PORTA_DATA_R = 0x00000;
            }
            }// end
*/
/************************************************************************************************
    Start Function
************************************************************************************************/
void start_Pulse(void){
    PWM1_ENABLE_R |= 0x00000002;
    SysTick_Wait1ms(1);
    PWM1_ENABLE_R &= ~0x00000002;
    SysTick_Wait800us(1);


}
/************************************************************************************************
    Logic 0 function
************************************************************************************************/
void logic0(){
    //PWM1_1_CTL_R |= # ;
    PWM1_ENABLE_R |= 0x00000002;
    SysTick_Wait450us(1);
    //PWM1_1_CTL_R &= ~# ;
    PWM1_ENABLE_R &= ~0x00000002;
    SysTick_Wait450us(1);
}

/************************************************************************************************
    Logic 1 function
************************************************************************************************/
void logic1(){
    //PWM1_1_CTL_R |= #;
    PWM1_ENABLE_R |= 0x00000002;
    SysTick_Wait900us(1);
    //PWM1_1_CTL_R &= ~# ;
    PWM1_ENABLE_R &= ~0x00000002;
    SysTick_Wait450us(1);
}


void address_0(void){
    logic0();
    logic0();

}

void address_1(void){
    logic0();
    logic1();

}
```

```

void address_2(void){
    logic1();
    logic0();

}

void address_3(void){
    logic1();
    logic1();

}

void Command_0(void){
    logic0();
    logic0();
    logic0();

}

void Command_1(void){
    logic0();
    logic0();
    logic1();
}

void Command_2(void){
    logic0();
    logic1();
    logic0();

}
void Command_3(void){
    logic0();
    logic1();
    logic1();

}

void Command_4(void){
    logic1();
    logic0();
    logic0();

}

void Command_5(void){
    logic1();
    logic0();
    logic1();

}

void Command_6(void){
    logic1();
    logic1();
    logic0();

}

void Command_7(void){
    logic1();
    logic1();
    logic1();

}

void device_select(uint16_t device_sel){
    if( device_sel == 0 ){ address_0(); }
    else if( device_sel == 1 ){ address_1(); }
    else if( device_sel == 2 ){ address_2(); }
    else if( device_sel == 3 ){ address_3(); }
    else{address_0();}
}
/*****************************************************************************************
    Port F handler for button push
****************************************************************************************/
void GPIOPortF_Handler(void){ // called on touch of either SW1 or SW2

  if(GPIO_PORTF_RIS_R&0x10){  // SW2 touch
    GPIO_PORTF_ICR_R = 0x10;  // acknowledge flag0
        press++;
    }

if(press == 1){

        GPIO_PORTF_DATA_R &= ~0x0E;
        GPIO_PORTF_DATA_R |= red;
        device = 0x00;
        UART_OutString("device has been changed to 0 \r\n");
        UART1_OutString("device has been changed to 0 \r\n");
}
    else if(press==2){

        GPIO_PORTF_DATA_R &= ~0x0E;
        GPIO_PORTF_DATA_R |= green;
        device = 0x01;
        UART_OutString("device has been changed to 1 \r\n");
        UART1_OutString("device has been changed to 1 \r\n");
    }
    else if(press==3){

        GPIO_PORTF_DATA_R &= ~0x0E;
        GPIO_PORTF_DATA_R |= blue;
        device = 0x02;
        UART_OutString("device has been changed to 2 \r\n");
        UART1_OutString("device has been changed to 2 \r\n");
    }
    else if(press==4){

        GPIO_PORTF_DATA_R &= ~0x0E;
        GPIO_PORTF_DATA_R |= white;
        device = 0x03;
        UART_OutString("device has been changed to 3 \r\n");
        UART1_OutString("device has been changed to 3 \r\n");
        press = 0;
    }
    //else{
        //GPIO_PORTF_DATA_R &= ~0x0E;
        //GPIO_PORTF_DATA_R |= red;
        //device = 0x00;
    //}


} // end of handler




int main(void){
    PLL_Init();
  PortF_Init();
    PortC_Init();
    UART_Init();
    PortB_UART1_Init();
    PWM1A_Init(256, 80);
    PWM1_A6_Init(1053,526); // change value
    EnableInterrupts();
    GPIO_PORTF_DATA_R = 0x02;

    //SysTick_Init();
    //device = 0;
    SysTick_Init(625);  // 80KHz interrupt
  while(1){
        cmd = UART1_InChar();
        //UART_OutChar(cmd);
        OutCRLF();

        if(cmd != 0){

            switch(cmd){

                case '0' :  start_Pulse();

                                    UART1_OutString("Start\r\n");

                                    device_select(device);

                                    UART1_OutString("device# ");
                                    UART1_OutUDec(device);
                                    UART1_OutString("\r\n");

                                    Command_0();

                                    UART1_OutString("command# ");
                                    UART1_OutChar(cmd);
                                    UART1_OutString("\r\n");

                                    break;

                case '1' : start_Pulse();
                                    device_select(device);
                                    Command_1();

                                    break;

                case '2': start_Pulse();
                                    device_select(device);
                                    Command_2();
                                    break;

                case '3' : start_Pulse();
                                    device_select(device);
                                    Command_3();
                                    break;

                case '4' : start_Pulse();
                                    device_select(device);
                                    Command_4();
                                    break;

                case '5' : start_Pulse();
                                    device_select(device);
                                    Command_5();
                                    break;

                case '6' : start_Pulse();
                                    device_select(device);
                                    Command_6();
                                    break;

                case '7' : start_Pulse();
                                    device_select(device);
                                    Command_7();
                                    break;
```

```c

                default: break;

            } // end of switch case


        } // end of if cmd is valid




    } // end of while(1)

} //end of main

/*************************************************************************************************
    UART Clear Line Feed
*************************************************************************************************/
void OutCRLF(void){
  UART_OutChar(CR);
  UART_OutChar(LF);
}

/*************************************************************************************************
    String Slice for UART7
*************************************************************************************************/
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;
}

//////// END OF CODE

```