```
/*********************************************************************************
*******
Author: Group 2 (Hang Xu, Wen Wu, Wenjun Ma)
Date complete: 19/2/2018
Filename: EE2A Experiment4 USB, Serial Interfacing, Analogue-to-Digital Conversion and Closed-loop
Controllers
Target device: PIC18F27K40
Fuse settings:NOMCLR, NOWDT,NOPROTECT,NOCPD
Program function:
(i)PC Control of LEDs via USB interface
(ii)PC Control of LED Brightness using a PWM Module
(iii)Analogue-to-Digital Converter with USB Output
(iv)PC interface to aDC Motor Closed-Loop Armature CurrentController using a PWM Module
*********************************************************************************
******/
#include <18F27K40.h>
#device adc=10
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#fuses NOMCLR, NOWDT,NOPROTECT,NOCPD

/***************main frequency setting***************/
#use delay(internal=32Mhz,clock_out)

/***************rs232 setting***************/
#pin_select U1TX=PIN_C0 // transmit data
#pin_select U1RX=PIN_C1 // receive data
#use rs232(uart1, baud=9600, ERRORS)

/***************pwm setting***************/
#pin_select PWM4=PIN_C3

/***************structure***************/
struct IO_Port_Definition
    {
    int unusedA:7;//PIN_A0..6
    int1 ADC;//PIN_A7
    int unusedB:8;//PIN_B0..7
    int1 ts;//PIN_C0
    int1 rc;//PIN_C1
    int1 debug; //PIN_C2
    int1 PWM; //PIN_C3
    int1 LED4; //PIN_C4
    int unusedC:3; //PIN_C5..7
    };
struct IO_Port_Definition Port;
struct IO_Port_Definition PortDirection;
#byte Port = 0xF8D
#byte PortDirection = 0xF88

/***************variables***************/
//RDA//
char CommandString[32];
int    index=0;
//Sentence//
char        led4on[]="LED4 ON";
```

```
char        led4off[]="LED4 OFF";
char        led4flash[]="LED4 FLASH";
char        led3brightness[]="LED3 BRIGHTNESS";
char        led3brightnessjudge[16];
char        collectdata[]="COLLECT DATA";
char        armaturecurrent[]="ARMATURE CURRENT";
char        armaturecurrentjudge[17];
//ADC//
float      adcresult;
int16      Buffer_adc[1024];
long        count=0;
int         adcswitch=2;
//Main//
int         pause=1;//1 for continue;0 for stop
int         flag=0;

//ARMATURE CURRENT//
float Desire_Result;
float adc_average;
unsigned int16 duty1=0;

/***************RDA_interrupt***************/
#INT_RDA
void rda_isr(void)
{
    Port.debug=0b1;
    pause=1;
    adcswitch=2;
    CommandString[0]=0;//reset c
    index=0;
    do
    {
        CommandString[index]=getc();
        putc(CommandString[index]);
        if(CommandString[index]==127)//backspace check
        {
            index=index-2;
        }
        index++;
    }
    while((index<31)&&(CommandString[index-1]!=13));
    CommandString[index-1]=0;
    putc(13);//enter
    putc(10);//back to first column
    Port.debug=0b0;
    /***brightness***/
    strncpy(led3brightnessjudge,CommandString,15);//extract first 15 characters in CommandString
and put them into led3brightnessjudge for later comparison
    led3brightnessjudge[15]=0;// add string end
    /***current***/
    strncpy(armaturecurrentjudge,CommandString,16);
    armaturecurrentjudge[16]=0;
    //ERROR JUDGEMENT//

if(STRICMP(CommandString,led4on)!=0&&STRICMP(CommandString,led4off)!=0&&STRICMP(Comma
ndString,led4flash)!=0&&STRICMP(led3brightnessjudge,led3brightness)!=0&&STRICMP(CommandStri
ng,collectdata)!=0&&STRICMP(armaturecurrentjudge,armaturecurrent)!=0)
```

```
    {
        puts("ERROR");
    }
}

/***************ADC_interrupt***************/
#INT_AD
void adc_isr(void)
{
    if(adcswitch==0)// adc is on, read data
    {
        if(count<1024)
        {
            adcresult = read_adc(ADC_READ_ONLY);
            Buffer_adc[count++] = adcresult;
        }
    }
    if(adcswitch==1)// adc is off, process data
    {
        if(count<2048)
        {
            adcresult = (0.9*adcresult)+(0.1*read_adc(ADC_READ_ONLY)); // reduce noise
            count = count+1;
        }
        if(count=2048) // data procession completed
        {
            adc_average = adcresult;///1024;
            if(Desire_Result-adc_average>=1)//negative feedback
            {
                duty1=duty1+1;
            }
            if(Desire_Result-adc_average<=-1)
            {
                if(duty1<10)
                {
                    duty1=0;
                }
                else
                {
                    duty1=duty1-1;
                }
            }
            if(duty1>=1000)
            {
                duty1=1000;
            }
            set_pwm4_duty(duty1);
            count=0;
        }
    }
}

/***************main_function***************/
void main()
{
    //Port Setting//
    PortDirection.ADC=0b1;
```

```
    PortDirection.ts=0b0;
    PortDirection.rc=0b1;
    PortDirection.debug=0b0;
    PortDirection.PWM = 0b0;
    PortDirection.LED4 = 0b0;
    // Weak Pull Up//
    int WPUC;//weak pull up PinC
    #byte WPUC=0x0F18;
    WPUC=0b11111111;
    //RDA//
    enable_interrupts(INT_RDA);
    //PWM//
    setup_timer_2(T2_CLK_INTERNAL|T2_DIV_BY_1,249,1);
    setup_ccp2(CCP_PWM|CCP_USE_TIMER1_AND_TIMER2);
    setup_pwm4(PWM_ENABLED|PWM_ACTIVE_HIGH|PWM_TIMER2);
    //ADC//
    setup_adc_ports(sAN7,VSS_FVR);
    setup_adc(ADC_LEGACY_MODE|ADC_CLOCK_DIV_64);
    setup_vref(VREF_ON|VREF_ADC_4v096);
    set_adc_channel(7);
    set_adc_trigger(ADC_TRIGGER_TIMER2);
    enable_interrupts(INT_AD);
    //GLOBAL//
    enable_interrupts(GLOBAL);
    while(1)
    {
        flag=0;//ERROR JUDGEFLAG,0 for ERROR;1 for OK

/**************LED SWITCH CONTROL*****************/
        if (STRICMP(CommandString,led4on)==0)
        {
            puts("OK");
            flag=1;
            pause=0;
            Port.LED4=0b1;
            while(pause==0)//Loop for stop
            {
            }
        }
        if (STRICMP(CommandString,led4off)==0)
        {
            puts("OK");
            flag=1;
            pause=0;
            Port.LED4=0b0;
            while(pause==0)
            {
            }
        }
        if (STRICMP(CommandString,led4flash)==0)
        {
            puts("OK");
            flag=1;
            pause=0;
            while(pause==0)
            {
            Port.LED4=0b1;
```

```
            delay_ms(500);
            Port.LED4=0b0;
            delay_ms(500);
            }
        }

/**************LED BRIGHTNESS CONTROL*****************/
        if (STRICMP(led3brightnessjudge,led3brightness)==0)
        {
            char brt[4]={""};
            int32 num_brt;
            strncpy(brt,CommandString+16,3);
            num_brt=atoi(brt);// convert char to int
            if((num_brt<=100)&&(num_brt>=0))
            {
                puts("OK");
                flag=1;
                int32 pwmvalue=50;
                pwmvalue=(1000*num_brt)/100;
                set_pwm4_duty(pwmvalue);
            }
            else
            {
                puts("ERROR");
            }
            pause=0;
            while(pause==0)
            {
            }
        }

/**************ADC COLLECT DATA*****************/
        if (STRICMP(CommandString,collectdata)==0)
        {
            puts("OK");
            flag=1;
            pause=0;
            count=0;
            adcswitch=0;
            while(pause==0)
            {
                if(count==1024)
                {
                    long ii;
                    printf("[");
                    for(ii=0;ii<=1023;ii++)
                    {
                        float adcvalue = (Buffer_adc[ii]*4.096)/1023; //rescale according to reference
voltage
                        if(ii<=1022)
                        {
                            printf("%f ",adcvalue);
                        }
                        if(ii==1023)
                        {
                            printf("%f",adcvalue);
                        }
```

```
                    }
                    printf("]");
                    putc(13);
                    putc(10);
                    while(pause==0);
                }
            }
        }
/**************ARMATURE CURRENT CONTROL*****************/
        if (STRICMP(armaturecurrentjudge,armaturecurrent)==0)
          {
            char crt[5]={""};
            long num_crt;
            count=0;
            adcresult=0;
            strncpy(crt,CommandString+17,4);
            crt[4]=0;
            num_crt=atol(crt);//convert char to long
            if((num_crt<=2000)&&(num_crt>=0))
            {
                puts("OK");
                flag=1;
                pause=0;
                adcswitch=1;
                Desire_Result=((num_crt*0.5/1000)*1023)/4.096;// resistor==0.5 ohms
                duty1=1000;
                while(pause==0)
                {
                }
            }
            else
            {
                puts("ERROR");
            }
          }
          if (flag==0)
          {
            pause=0;
            while(pause==0);
          }
    }
}
```