#include "led.h"

#include "delay.h"

#include "key.h"

#include "sys.h"

#include "usart.h"

#include "timer.h"

#include "adc.h"

int main(void)

 {

u16 adcx1=0,adcx2=0,adcx3=0,adcx4=0;

u16 Bat1=0,Bat2=0,Bat3=0,Bat4=0;

u16 max=0,min=0;

delay\_init();

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

Adc\_Init();

while(1)

{

adcx1=Get\_Adc\_Average(ADC\_Channel\_0,10);

adcx2=Get\_Adc\_Average(ADC\_Channel\_1,10);

adcx3=Get\_Adc\_Average(ADC\_Channel\_2,10);

adcx4=Get\_Adc\_Average(ADC\_Channel\_3,10);

Bat1=adcx1\*(3300/4096); //

Bat2=adcx2\*(3300/4096);

Bat3=adcx3\*(3300/4096);

Bat4=adcx4\*(3300/4096);

if(Bat2>Bat1)

{

max=Bat2;

}

else

{

max=Bat1;

}

if(Bat3>max)

{

max=Bat3;

}

if(Bat4>max)

{

max=Bat4;

}

if(Bat2<Bat1)

{

min=Bat2;

}

else

{

min=Bat1;

}

if(Bat3<min)

{

min=Bat3;

}

if(Bat4<min)

{

min=Bat4;

}

if(max-min>100)

{

RCC\_Config() ;

TIM11\_Config() ;

TIM12\_Config() ;

}

else

{

TIM\_SetCompare1(TIM1,0);

}

}

}

#include "adc.h"

#include "delay.h"

void Adc\_Init(void)

{

ADC\_InitTypeDef ADC\_InitStructure;

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA |RCC\_APB2Periph\_ADC1 , ENABLE );

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_4|GPIO\_Pin\_5|GPIO\_Pin\_6|GPIO\_Pin\_7;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AIN;

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

RCC\_ADCCLKConfig(RCC\_PCLK2\_Div6);

ADC\_DeInit(ADC1);

ADC\_InitStructure.ADC\_Mode = ADC\_Mode\_Independent;

ADC\_InitStructure.ADC\_ScanConvMode = ENABLE;

ADC\_InitStructure.ADC\_ContinuousConvMode = ENABLE;

ADC\_InitStructure.ADC\_ExternalTrigConv = ADC\_ExternalTrigConv\_None;

ADC\_InitStructure.ADC\_DataAlign = ADC\_DataAlign\_Right;

ADC\_InitStructure.ADC\_NbrOfChannel = 4;

ADC\_Init(ADC1, &ADC\_InitStructure);

ADC\_Cmd(ADC1, ENABLE);

ADC\_ResetCalibration(ADC1);

while(ADC\_GetResetCalibrationStatus(ADC1));

ADC\_StartCalibration(ADC1);

while(ADC\_GetCalibrationStatus(ADC1));

}

u16 Get\_Adc(u8 ch)

{

ADC\_RegularChannelConfig(ADC1, ADC\_Channel\_0, 1, ADC\_SampleTime\_239Cycles5 );

ADC\_RegularChannelConfig(ADC1, ADC\_Channel\_1, 2, ADC\_SampleTime\_239Cycles5 );

ADC\_RegularChannelConfig(ADC1, ADC\_Channel\_2, 3, ADC\_SampleTime\_239Cycles5 );

ADC\_RegularChannelConfig(ADC1, ADC\_Channel\_3, 4, ADC\_SampleTime\_239Cycles5 );

ADC\_SoftwareStartConvCmd(ADC1, ENABLE);

while(!ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_EOC ));

return ADC\_GetConversionValue(ADC1);

}

u16 Get\_Adc\_Average(u8 ch,u8 times)

{

u32 temp\_val=0;

u32 temp\_va2=0;

u32 temp\_va3=0;

u32 temp\_va4=0;

u8 t;

for(t=0;t<times;t++)

{

temp\_val+=Get\_Adc(ADC\_Channel\_0);

temp\_va2+=Get\_Adc(ADC\_Channel\_1);

temp\_va3+=Get\_Adc(ADC\_Channel\_2);

temp\_va4+=Get\_Adc(ADC\_Channel\_3);

delay\_ms(5);

}

return temp\_val/times;

}

#include "timer.h"

#include "led.h"

#include "usart.h"

void RCC\_Config(void)

{

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOE|RCC\_APB2Periph\_AFIO|RCC\_APB2Periph\_GPIOA|RCC\_APB2Periph\_GPIOB, ENABLE);

GPIO\_PinRemapConfig(GPIO\_FullRemap\_TIM1, ENABLE); //È«Ó³Éä

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_TIM1, ENABLE);

TIM\_InternalClockConfig(TIM1);

TIM\_DeInit(TIM1);

}

void TIM11\_Config(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

TIM\_OCInitTypeDef TIM\_OCInitStructure;

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_11;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOE, &GPIO\_InitStructure);

TIM\_TimeBaseStructure.TIM\_Period = 100-1;

TIM\_TimeBaseStructure.TIM\_Prescaler = 72-1;

TIM\_TimeBaseStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1;

TIM\_TimeBaseStructure.TIM\_CounterMode=TIM\_CounterMode\_CenterAligned2;

TIM\_TimeBaseStructure.TIM\_RepetitionCounter=1;

TIM\_TimeBaseInit(TIM1, &TIM\_TimeBaseStructure);

TIM\_OCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM2;

TIM\_OCInitStructure.TIM\_OutputState = TIM\_OutputState\_Enable;

TIM\_OCInitStructure.TIM\_Pulse =60;

TIM\_OCInitStructure.TIM\_OCPolarity = TIM\_OCPolarity\_Low;

TIM\_OC2Init(TIM1, &TIM\_OCInitStructure);

TIM\_OC2PreloadConfig(TIM1, TIM\_OCPreload\_Enable);

TIM\_ARRPreloadConfig(TIM1, ENABLE);

TIM\_Cmd(TIM1, ENABLE);

TIM\_CtrlPWMOutputs(TIM1, ENABLE);

}

void TIM12\_Config(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

TIM\_OCInitTypeDef TIM\_OCInitStructure;

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOE, &GPIO\_InitStructure);

TIM\_TimeBaseStructure.TIM\_Period = 100-1;

TIM\_TimeBaseStructure.TIM\_Prescaler = 72-1;

TIM\_TimeBaseStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1;

TIM\_TimeBaseStructure.TIM\_CounterMode=TIM\_CounterMode\_CenterAligned3 ;

TIM\_TimeBaseInit(TIM1, &TIM\_TimeBaseStructure);

TIM\_OCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM2;

TIM\_OCInitStructure.TIM\_OutputState = TIM\_OutputState\_Enable;

TIM\_OCInitStructure.TIM\_OCPolarity = TIM\_OCPolarity\_High;

TIM\_OCInitStructure.TIM\_Pulse =80;

TIM\_OC1Init(TIM1,&TIM\_OCInitStructure);

TIM\_OC1PreloadConfig(TIM1, TIM\_OCPreload\_Enable);

TIM\_ARRPreloadConfig(TIM1, ENABLE);

TIM\_Cmd(TIM1, ENABLE);

TIM\_CtrlPWMOutputs(TIM1, ENABLE);

}