//按键识别

//宏定义

#define KEY\_ONE 1

#define KEY\_TWO 2

#define KEY\_THREE 3

#define KEY\_FOUR 4

#define KEY\_ERR 0

#define KEY\_ONE\_SCAN (!!(GPIOA->IDR&(0x1<<0)))

unit\_8 key\_scanf()

{

u8 key\_res=KEY\_ERR;

static u8 key\_state=0;//判断之前是否按下

if(KEY\_ONE\_SCAN&&key\_state==0)

{

key\_delay(5000);//防抖动

if(KEY\_ONE\_SCAN)

{

key\_state=1;

key\_res=KEY\_ONE;

}

}

else if(!KEY\_ONE)//按键松手

{

key\_state=0;

}

if(KEY\_TWO==0&&key\_state==0)

return key\_res;

}

//异或的作用 1反，0不变

a^=0x1<<n;//控制灯灭

//按键延时

static void key\_delay(u32 time)

{

time \*=42;//42为1微妙

while(time)

{

time--;

}

}

/\*串口接收字符串\*/

void uart\_reciceve\_string(u8 \*recieve\_buf)

{

while(1)

{

while(!USART->SR&(0x1<<5));

\*recieve\_buf=USART1->DR;

if(\*recieve\_buf=='\r'||\*recieve\_buf=='\n')

{

\*recieve\_buf='\0';

break;

}

else

{

recieve\_buf++;

}

}

}

/\*指针数组存放\*/

u32 switch\_led(char \*buf1)

{

u32 i;

char \*buf[7]={"led1","","","",""};

for(i=0;i<7;i++)

{

if(strcmp(buf1,buf[i])==0)

return i;

}

}

//usart中断接收数据

/\*中断函数

//优先级分组3~7

分组，抢占响应合成优先级值

值写入具体中断\*/

u32 i=0;

NVIC\_SetPriorityGrouping(7-2);//抢占位

i= NVIC\_EncodePriority(7 - 2, 2, 2);//抢占与响应优先级

NVIC\_SetPriority(USART1\_IRQn, i);//中断类型

//外设寄存器

NVIC\_EnableIRQ(USART1\_IRQn);

USART1->CR1=|(0X1<<5);//接收中断使能标志位

//服务函数,判断中断类型

void USART1\_IRQHandler(void)

{

if(USART1->SR&(0X1<<5))

{

USART1->SR&=~(0X1<<5);//清除标志位

VAL=USART1->DR;

while(!USART1->SR&(0x1<<7));

USART1->DR=VAL;

}

}

//空闲位中断使能标志位

if(USART1->SR&(0x1<<4))

//清除标志位

u8 i=USART1->DR;

/\*初始GPIO

时钟，模式，输入，浮空\*/

//外部中断

RCC->APB2|=(0x1<<14);//时钟

/\*中断服务函数

判断标志位

清除标志位

程序

\*/

//看门狗

/\*初始化

开启LSI

等待LSI

配置密钥寄存器0x5555

分频

重载

第一次喂狗写入0xaaa

配置密钥寄存器写入0xccc启动看门狗\*/

RCC->CR|=(0x1<<0); //打开HSI的时钟使能

while(!RCC->CSR&(0x1<<1)) //等待LSI的时钟使能打开

{

}

IWDG->KR|=(0x5555<<0); //清除写保护，配置密钥寄存器

IWDG->PR|=2;//16分频，2500HZ

IWDG->RLR=2500;//计数值为2500，复位时间1秒

IWDG->KR|=0XAAAA;//第一次喂狗

IWDG->KR|=0XCCCC;//启动看门狗

/\*不断喂狗0xaaaa

不要等待时间0\*/

IWDG->KR|=0XAAAA;

//权威指南，9.5，滴答定时器

//参考手册 20，基本定时器

//滴答

void SysTick\_init(u32 ms)

{

SysTick->CTRL|=(0X0<<2);//设置滴答定时器的时钟源21M

SysTick->VAL|=0;//清0计数值

SysTick->LOAD|=ms\*21000;//写入重载的计数值

if(ms>0xffffff/21000) //判断该计数的范围是否大于最大的而计数值

return ;

SysTick->CTRL|=(0x1<<0);//打开计数使能

while(!SysTick->CTRL&(0X1<<16));//计数开始，当16位状态位为1，退出

SysTick->CTRL|=(0x0<<0); //计数结束后关闭使能

}

void systick\_interrupt(u32 ms)

{

SysTick->CTRL|=(0X0<<2);//定时器的时钟源

SysTick->VAL|=0;

SysTick->LOAD|=ms\*21000-1;//载入值

if(ms>0xffffff/21000) //判断计数的次数

return ;

SysTick->CTRL|=(0x1<<0); //打开滴答定时器的使能

SysTick->CTRL|=(0x1<<1); //打开滴答定时器的中断使能位

}

void SysTick\_Handler()

{

if(SysTick->CTRL&(0X1<<16))

{

}

}

//超时接收

while(1)

{

void SysTick\_init(u32 ms)

{

SysTick->CTRL|=(0X0<<2);//设置滴答定时器的时钟源21M

SysTick->VAL|=0;//清0计数值

SysTick->LOAD|=80\*2100;//写入重载的计数值

SysTick->CTRL|=(0x1<<0);//打开计数使能

SysTick->CTRL|=(0x1<<1); //打开滴答定时器的使能位

while(!(USART1->SR&(0x1<<6)));//时间条件

USART1->DR=\*send\_buf;

send\_buf++;

SysTick->VAL|=0;//清0计数值

}

}

//基本定时器的延时

RCC->APB1ENR|=(0X4<<1);

TIM6->PSC=8399;

TIM6->ARR=nms\*10-1;

TIM6->CNT=0;

TIM6->EGR|=(0X1<<1);

TIM6->SR&=~(0X1<<0);

TIM6->CR1=0;

TIM6->CR1|=(0X1<<0);

TIM6->CR1&=~(0x1<<1);

TIM6->CR1|=(0x1<<2);

TIM6->CR1|=(0X1<<3);

TIM6->CR1&=~(0X1<<7);//关闭缓冲区

while(!TIM6->SR&(0X1<<0))

{

}

TIM6->CR1&=~(0X1<<0);

//中断，判断，清除标志；

RCC->APB2ENR |= 0X1 << 17; //开启TIM6的时钟

TIM10->PSC = 8400 - 1; //进行8400分频，计数器时钟为10KHz

TIM10->CNT = 0; //清空计数器的值

TIM10->ARR = nms \* 10 - 1; //延时为毫秒级

TIM10->EGR |= 0X1 << 0; //设置一次UG位

TIM10->SR &= ~(0X1 << 0); //清除一次更新标志位(防止出错)

TIM10->DIER|=(0X1<<0);

TIM10->CR1 = 0; //清空控制寄存器1的值

TIM10->CR1 |= 0X0 << 3; //定时完成以后自动停止计数

TIM10->CR1|=(0X1<<2);

TIM10->CR1 |= 0X1 << 0; //开启计数器

NVIC\_SetPriorityGrouping(7-2);//抢占位

inter6=NVIC\_EncodePriority(7-2,0, 0);//抢占与响应优先级

NVIC\_SetPriority(TIM10\_IRQn, inter6);//中断类型

NVIC\_EnableIRQ(TIM10\_IRQn);

/\*PWM比较输出 GPIO 是否具有复用功能定时器

PA6 TIM3\_CH1\*/

//初始化PWM

//开启GPIO,模式，输出类型，速度，上下拉，复用功能为TIM3

RCC->AHB1ENR|=(0x1<<0); //同上

GPIOA->MODER&=~(0x3<<2\*6);

GPIOA->MODER|=(0x2<<2\*6);

GPIOA->OTYPER&=~(0x1<<6);

GPIOA->PUPDR&=~(0X3<<2\*6);

GPIOA->OSPEEDR&=~(0X3<<2\*6);

GPIOA->AFR[0]&=~(0XF<<4\*6);

GPIOA->AFR[0]|=(0X2<<4\*6);//tim3

/\*基本定时器开启时钟使能，外设时钟使能，配置分频器，清空计数位，配置自动重载

一次UG，清除标志位，配置寄存器不停止\*/

RCC->APB1ENR|=(0X1<<1); //打开定时器使能

TIM3->SMCR&=~(0X7<<0);

TIM3->SMCR&=~(0X1<<14);//外设时钟源

TIM3->PSC=83;//频率1M

TIM3->ARR=1000-1;//1毫秒

TIM3->CNT=0;

TIM3->EGR|=(0X1<<0);//初始化计数器，并产生更新功能

TIM3->SR&=~(0X1<<0);

TIM3->CR1=0;

TIM3->CR1|=(0X1<<0); //打开定时器的使能

TIM3->CR1&=~(0x1<<1); // 开启事件更新的功能

TIM3->CR1|=(0x1<<2); //更新请求源的条件

TIM3->CR1|=(0X1<<3); //计数时间到不会停止计数

/\*比较寄存器1，配置CCMR1，CCER\*/

TIM3->CCR1=900-1;//比较输出下写入占空比

TIM3->CCMR1&=~(0XFF<<0);

TIM3->CCMR1|=(0X6<<4); //输出PWM模式1，小于比较值为有效状态

TIM3->CCER&=~(0XF<<0);

TIM3->CCER|=(0X1<<0);//使能PWM的输出

/\*GPIO 基本定时器 CCMR1 CCER EGR SMCR 捕获中断 NVIC\*/

RCC->AHB1ENR|=(0x1<<0); //同上

GPIOA->MODER&=~(0x3<<2\*1);

GPIOA->MODER|=(0x2<<2\*1);

GPIOA->PUPDR&=~(0X3<<2\*1);

GPIOA->AFR[0]&=~(0XF<<4\*1);

GPIOA->AFR[0]|=(0X1<<4\*1);//tim2输入

RCC->APB1ENR|=(0X1<<0);//打开时钟源

TIM2->SMCR&=~(0X7<<0); //从模式选择寄存器

TIM2->SMCR&=~(0X1<<14);//时钟源

TIM2->PSC=83;//分频为1M

TIM2->ARR=0xffffff-1;//写入的数据长度为

TIM2->CNT=0;//清空后计数器的值得

TIM2->EGR|=(0X1<<0);//打开一次UG使

TIM2->SR&=~(0X1<<0);

TIM2->CR1=0;

TIM2->CR1|=(0X1<<0);//打开时钟使能

TIM2->CCMR1=0；

TIM2->CCMR1|=(0X1<<8);//映射IC2,TI2

TIM2->CCMR1|=(0X2<<0);//映射IC1，TI2

TIM2->CCER&=~(0xff<<0);

TIM2->CCER|=(0X1<<4);//IC2捕获使能，上升沿触发

TIM2->CCER|=(0X1<<0);//

TIM2->CCER|=(0X1<<1);//IC1捕获使能，下降沿触发

TIM2->EGR&=~(0X1<<1);//硬件置1，消除软件置1

TIN2->SMCR|=(0X6<<4); //从模式为复位功能

TIN2->SMCR|=(0X4<<0);//复用与触发方式

TIM2->DIER|=(0X1<<2);//使能中断

NVIC\_SetPriorityGrouping(7-2);//抢占位

inter6=NVIC\_EncodePriority(7-2,0, 0);//抢占与响应优先级

NVIC\_SetPriority(TIM12\_IRQn, inter6);//中断类型

NVIC\_EnableIRQ(TIM2\_IRQn);

void TIM2\_IRQHandler()

{

if(TIM2->SR&(0X1<<10))

{

TIM2->SR&=~(0X1<<2);

TIM2->SR&=~(0X1<<10);//重复捕获与捕获标志位

if(TIM2->CCR2==999)

GPIOC->BSRRH|=0X1<<4;

}

}

RCC->AHB1ENR|=(0x1<<0);

GPIOA->MODER&=~(0x3<<2\*1);

GPIOA->MODER|=(0x2<<2\*1);//打开复用输入功能

GPIOA->PUPDR&=~(0X3<<2\*1);

GPIOA->AFR[0]&=~(0XF<<4\*1);

GPIOA->AFR[0]|=(0X1<<4\*1);//TIM2输入

RCC->APB1ENR|=(0X1<<0);

TIM2->SMCR&=~(0X7<<0);

TIM2->SMCR&=~(0X1<<14);//外设时钟源

TIM2->PSC=83;//分屏后频率为1M

TIM2->ARR=0xffffff-1;//最大计数长度

TIM2->CNT=0;

TIM2->EGR|=(0X1<<0); //一次UG更新

TIM2->SR&=~(0X1<<0); //清除更新中断标志位

TIM2->CR1=0;

TIM2->CR1|=(0X1<<0); //开启定时器使能

TIM2->CCMR1=0;

TIM2->CCMR1|=(0X1<<1);//映射IC1,TI2

TIM2->CCER&=~(0xff<<0);

TIM2->CCER|=(0X1<<0);//IC1捕获使能，上升沿触发

TIM2->EGR&=~(0X1<<1);//硬件置1，不让软件清0

TIM2->SMCR|=(0X6<<4); //从模式为复位功能

TIM2->SMCR|=(0X4<<0);//复用与触发方式

TIM2->DIER|=(0X1<<1);//IC1使能中断

NVIC\_SetPriorityGrouping(7-2);//抢占位

inter8=NVIC\_EncodePriority(7-2,0, 0);//抢占与响应优先级

NVIC\_SetPriority(TIM2\_IRQn, inter8);//中断类型

NVIC\_EnableIRQ(TIM2\_IRQn);

TIM2->CCER|=(0X1<<1);//IC1捕获下降沿触发

}

void TIM2\_IRQHandler()

{

if(TIM2->SR&(0X1<<10)) //判断重复捕获标志位

{

TIM2->SR&=~(0X1<<2);

TIM2->SR&=~(0X1<<10);//重复捕获与捕获标志位

if(TIM2->CCR1==999) //判断计数次数是否达到一个周期

GPIOC->BSRRH|=0X1<<4;

}

}

//DMA 串口数据搬运从存储器到外设 PA9\PA10

//初始化串口115200 PA10初始化，做复用输出

}

//开时钟 关闭数据流，确认关闭，配置寄存器，直接模式

//数据项数目，外设地址，存储器地址，打开使能，使能一次数据流

void dam2\_stream7\_init(u32 \*m\_addr,u32 \*p\_addr,u32 data\_num)

{

RCC->AHB1ENR|=(0x1<<22);//打开DMA2的时钟使能

DMA2\_Stream7->CR&=~(0x1<<0); //关闭数据流

while(DMA2\_Stream7->CR&(0X1<<0));//等待数据流关闭

DMA2\_Stream7->CR|=(0x4<<25);//选择通道7

/\*单次传输，不采用双缓冲，低优先级，外设增量与PSIZE相关

存储器大小与外设数据大小都为1个字节\*/

DMA2\_Stream7->CR|=(0x1<<10);//存储器的地址要递增

DMA2\_Stream7->CR|=(0x1<<6);//传输方向位存储器到外设

DMA2\_Stream7->CR&=~(0x1<<2);//使用直接模式

DMA2\_Stream7->NDTR&=~(0XFF<<0);

DMA2\_Stream7->NDTR=data\_num;//数据项的数目

DMA2\_Stream7->PAR=p\_addr;//外设地址

DMA2\_Stream7->M0AR= m\_addr; //存储器的地址

DMA2\_Stream7->CR|=(0X1<<0)；//开启外设DMA

USART1->CR3|=1<<7; //打开USART的DMA使能

//库函数实现呼吸灯

int main (void) {

int i, b;

LED\_GPIO\_Init();

b = 20000;

while (1) {

for (i = 0; i < b; i ++) {

GPIO\_ResetBits(GPIOF, GPIO\_Pin\_6);

delay(b - i);

GPIO\_SetBits(GPIOF, GPIO\_Pin\_6);

delay(i);

}

for (i = 0; i < b; i++) {

GPIO\_ResetBits(GPIOF, GPIO\_Pin\_6);

delay(i);

GPIO\_SetBits(GPIOF, GPIO\_Pin\_6);

delay(b - i);

}

delay(100);

/\*DMA数据从存储器到存储器默认外设的寄存器写源地址\*/

RCC->AHB1ENR|=0x1<<22;

DMA2\_Stream3->CR=0;

while(DMA2\_Stream3->CR&(0X1<<0));//等待使能位关闭

DMA2\_Stream3->CR|=(0x3<<25);//选择数据流3通道3

DMA2\_Stream3->CR|=(0x1<<23);//4个节拍

DMA2\_Stream3->CR|=(0x1<<16);//优先级为中

DMA2\_Stream3->CR|=(0X1<<13);//存储器数据大小为半字

DMA2\_Stream3->CR|=(0X1<<11);//外设数据大小为半字

DMA2\_Stream3->CR|=(0X1<<10);//每次传输后地址增量为MSIZE

DMA2\_Stream3->CR|=(0X1<<9);//每次传输后地址增量为PSIZE

DMA2\_Stream3->CR|=(0X2<<6);//传输方向为存储器到存储器

DMA2\_Stream3->CR|=(0X3<<3);//传输完成中断，传输半完成中断

DMA2\_Stream3->NDTR=data\_num;

DMA2\_Stream3->PAR=p\_addr;

DMA2\_Stream3->M0AR=m\_addr;

DMA2\_Stream3->FCR|=(0X1<<2);//打开FIFO模式

DMA2\_Stream3->FCR&=~(0x3<<0);

DMA2\_Stream3->FCR|=(0X1<<0);//将FIFO容量设为一半，8个字节

NVIC\_SetPriority(DMA2\_Stream3\_IRQn,NVIC\_SetPriorityGrouping(7-2,0,0));

NVIC\_EncodePriority(DMA2\_Stream3\_IRQn);//打开NVIC响应

DMA2\_Stream3->CR|=0X1<<0; //打开使能

void DMA2\_Stream3\_IRQHandler()

{

if(DMA2->LISR&(0X1<<26))//判断半传输状态

{

DMA2->LIFCR|=(0X1<<26);//清除标志位

}

if(DMA2->LISR&(0X1<<27))//判断传输完成

{

DAM2->LISCR|=(0x1<<27);//清除标志位

}

}

float USARTDIV\_1=0;

u16 DIV\_MAN=0;

u16 DIV\_FRA=0;

GPIOA->CRL|=(0X1<<8);

GPIOA->CRL|=(0X2<<10);

GPIOA->CRL&=~(0X3<<12);//PA2复用输出，PA3复用输入

GPIOA->CRL|=(0X1<<14);

RCC->

USART2->CR1=0;

USART2->CR1|=(0X3<<2);

USART2->CR1|=(0X1<<0);//打开使能

USARTDIV\_1=(float)84000000ul/(16\*B);//波特率

DIV\_MAN=(u16)USARTDIV\_1; //整数部分

DIV\_FRA=(USARTDIV\_1-DIV\_MAN)\*16+(float)0.5; //小数部分4舍5入

USART1->BRR=(DIV\_MAN<<4)|DIV\_FRA; //波特率写入寄存器

}

}

—————————