# 实验报告

PB21511897 李霄奕

实验1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 变量名 | 数据类型 | 变量值 | 寄存器 | 内容 |
| ui\_a | unsigned int | 1 | r5 | 0x01 |
| ui\_b | unsigned int | 2 | r6 | 0x02 |
| ui\_c | unsigned int | 0xFF | r7 | 0xFF |
| 寄存器标志位 | |  | x-PSR^N | 1 |
| i\_tmp | static int | -1 | r9 | 0xFFFFFFFF |
| s16\_tmp | static float | -32768 | r0 | 0xFFFF8000 |
| f\_tmp | static float | -0.5 | s0 | 0xBF000000 |
|  |  | 0 | FAULT | 0 |
|  |  | 0x00 | BASEPRI | 0x00 |
|  |  | 0 | PRIMASK | 0 |
|  |  | 0x04 | CONTORL | 0x04 |
|  |  |  | xPSR | 0x61000000 |
|  |  |  | MSP | 0x20000674 |

实验2

位带映射公式：

#define BITBAND(addr,bitnum)

((addr&0xF0000000)+0x2000000+((addr&0xFFFFF)<<5)+(bitnum<<2))

内存中关于GPIO的地址分配如下：

#define PERIPH\_BASE ((uint32\_t)0x40000000)

#define AHB1PERIPH\_BASE (PERIPH\_BASE + 0x00020000)

#define GPIOA\_BASE (AHB1PERIPH\_BASE + 0x0000)

#define GPIOA\_ODR\_Addr (GPIOA\_BASE+20)

#define GPIOA\_IDR\_Addr (GPIOA\_BASE+16)

#define PAout(n) BIT\_ADDR(GPIOA\_ODR\_Addr,n)

由此可知，GPIOA\_ODR\_Addr==0x40000000+0x00020000+0x0000+20==0x40020014

则有：BITBAND(GPIOA\_ODR\_Addr,0)==0x42400280;

BITBAND(GPIOA\_ODR\_Addr,1)==0x42400284;

BITBAND(GPIOA\_ODR\_Addr,7)==0x4240029C;

因此能够得出，位带操作扩充了32倍

综合实验

1. 功能描述

利用定时器生成 PWM 波，并在 LCD 显示占空比。

1. 功能模块划分

定时器模块、中断服务模块、PWM波模块、LCD显示模块

1. 系统结构图

稳定精确计时

占空比信号

中断服务唤醒

1. 实现功能的核心代码
2. main函数：实现主要逻辑

int main(){

uint32\_t pwm=0;

uint8\_t data[7]={0};

delay\_init(168);

LCD\_GPIO\_Init();

LCD\_Init();

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

TIM2\_PWM\_Init(3000,0);

LCD\_Clear();**//初始化，启动相关时钟、优先级中断**

while(1)

{**//循环实现PWM波的呼吸灯效果**

for(pwm=0;pwm<3000;pwm+=5){

delay\_ms(1);

TIM\_SetCompare3(TIM2, pwm);

update(data,pwm);**//根据pwm的值计算要显示的占空比内容**

LCD\_Display\_Words(0,0,data);

}

for(pwm=3000;pwm>0;pwm-=5){

delay\_ms(1);

TIM\_SetCompare3(TIM2, pwm);

update(data,pwm);

LCD\_Display\_Words(0,0,data);

}

}

}

1. update函数：输入pwm值，计算占空比，将结果以百分比的形式输出到字符串上

void update(uint8\_t\* data,uint32\_t pwmm){**//利用整型变量乘除的性质，逐位计算显示值**

data[0]=(pwmm\*10/3000)%10+48;

data[1]=(pwmm\*100/3000)%10+48;

data[2]='.';

data[3]=(pwmm\*1000/3000)%10+48;

data[4]=(pwmm\*10000/3000)%10+48;

data[5]='%';

data[6]=0;

}

1. 初始化LCD的GPIO接口

void LCD\_GPIO\_Init(){

GPIO\_InitTypeDef GPIO\_InitStructure; //GPIO初始化结构体

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOG|RCC\_AHB1Periph\_GPIOF, ENABLE);//使能GPIOG、GPIOF时钟

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_1;//初始化G引脚1

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_OUT;

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_100MHz;

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;

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

GPIO\_ResetBits(GPIOG,GPIO\_Pin\_1);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_14 | GPIO\_Pin\_15;//初始化F引脚14、15

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_OUT;

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_100MHz;

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;

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

GPIO\_ResetBits(GPIOF,GPIO\_Pin\_14 | GPIO\_Pin\_15);

CS=1;

SID=1;

SCLK=1;

}

1. 初始化中断函数TIM2

void TIM2\_PWM\_Init(u32 arr, u32 psc)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

TIM\_OCInitTypeDef TIM\_OCInitStructure;

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM2,ENABLE);

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOB, ENABLE);

GPIO\_PinAFConfig(GPIOB,GPIO\_PinSource10,GPIO\_AF\_TIM2);

GPIO\_PinAFConfig(GPIOB,GPIO\_PinSource10,GPIO\_AF\_TIM2);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_10;

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

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_100MHz;

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;

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

TIM\_TimeBaseStructure.TIM\_Prescaler=psc;

TIM\_TimeBaseStructure.TIM\_CounterMode=TIM\_CounterMode\_Up;

TIM\_TimeBaseStructure.TIM\_Period=arr;

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

TIM\_TimeBaseInit(TIM2,&TIM\_TimeBaseStructure);

TIM\_OCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM1;

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

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

TIM\_OC3Init(TIM2, &TIM\_OCInitStructure);

TIM\_Cmd(TIM2, ENABLE);

}