四线风扇驱动以及测速

第一步 了解四线风扇相关信息

本实验所用的四线风扇采用5V供电，需要将VCC接5V，GND接地，PWM调速信号线输入3.3V高电平即可，其转速反馈信号为开漏输出需要上拉电阻。

测速线的信号：该信号线输出占空比50%的测速信号，转速越快其频率越高，二者呈线性关系，其具体的计算如下。风扇转动一圈信号线发出两个周期的信号，故设信号线周期为T，每分钟转速为N，其频率F=1/T=2N/60，综上所述N=30F。

第二步 实验设计思路

风扇驱动：风扇的驱动信号由定时器2通道1产生。每次按下调速按键（即用户按键“USER”）触发13号外部中断，则改变定时器工作模式“mode”（mode从1到5轮转）。紧接着关闭定时器2通道1，并根据“mode”值重新初始化定时器2（计数周期不变，改变高电平计数值即可改变占空比），最后打开定时器2通道1让其产生新的PWM波形。占空比不同，风扇转速也就不同。

风扇测速：①将风扇的测速信号线接上拉电阻之后连接到PC9端口，用以触发9号外部中断，每当中断被触发则测速信号下降沿个数加一。②用定时器3通道1产生频率为1Hz的信号，并送到PC8端口用以触发8号外部中断实现计时。每当中断来临则出当前计数值就是测速信号线的频率F（注：1S时间内下降沿个数，即1S之间内周期个数也就是频率F），由第一步公式可知，用此时计数值乘以30即风扇的每分钟转速，将此转速输出到LCD12864屏幕即可。最后将计数值清零进行新的测速。

第三步 创建工程

时钟配置：在时钟配置页面下的“HCLK”设置其频率为8MHz，此时系统时钟将变为8MHz。

定时器配置：①将定时器2通道1设置在“PWM Generation CH1”模式，预分频系数填入7（实现8分频），计数周期1000，脉冲500（即高电平持续的计数值）。此时定时器2通道1产生初始占空比50%，频率为8MHz/8/1000=1KHz的风扇驱动信号。再将定时器2通道1输出TIM2\_CH1映射在PA15引脚。②将定时器3通道1同样设置在“PWM Generation CH1”模式，预分频系数填入127（实现128分频），计数周期62500，脉冲31250。此时定时器2通道1产生初始占空比50%，频率为8MHz/128/62500=1Hz的定时信号。最后将定时器3通道1输出TIM3\_CH1映射在PC6引脚。

外部中断设置：将PC8、PC9以及PC13均设置为下降沿触发的外部中断，并使能4-15号外部中断。

LCD1602相关信号线端口设置：将PA0到PA7均设置为通用输出模式，用以向LCD1602数据总线发送数据。将PB0到PB2也设置为通用输出模式，用以向LCD1602控制总线发送命令。

第四步 硬件电路连接说明

风扇驱动引脚----定时器2通道1输出PA15[CN7\_17]

风扇测速引脚----9号外部中断PC9[CN10\_1]

定时器3通道1输出PC6[CN10\_4]----8号外部\_中断PC8[CN10\_2]

LCD1602的D0引脚----通用输出端口PA0[CN8\_1]

LCD1602的D1引脚----通用输出端口PA1[CN8\_2]

LCD1602的D2引脚----通用输出端口PA2[CN10\_34]

LCD1602的D3引脚----通用输出端口PA3[CN10\_6]

LCD1602的D4引脚----通用输出端口PA4[CN8\_3]

LCD1602的D5引脚----通用输出端口PA5[CN5\_6]

LCD1602的D6引脚----通用输出端口PA6[CN5\_5]

LCD1602的D7引脚----通用输出端口PA7[CN5\_4]

LCD1602的RS引脚----通用输出端口PB0[CN5\_3]

LCD1602的RW引脚----通用输出端口PB1[CN8\_44]

LCD1602的E引脚----通用输出端口PB2[CN10\_22]

第五步 编写代码

1. 变量定义及函数声明

#define u8 unsigned char //定义数据类型

#define u16 unsigned int

int mode=1; //调速模式

int number=0; //下降沿计数

int number\_f=0; //每分钟转速

int number\_old=0; //下降沿计数寄存值

unsigned char tstr[5];//显示器字符串

/\*LCD1602相关函数\*/

void delay(void);

void lcdwritecmd(unsigned char cmd);

void lcdwritedata(unsigned char dat);

void lcdinit(void);

void lcdsetcursor(unsigned char x, unsigned char y);

void lcdshowstr(unsigned char x, unsigned char y, unsigned char \*str);

void MX\_TIM2\_Init\_NEW(int i); //自定义新的定时器初始化函数

1. 主程序进入循环工作状态前的部分初始化程序

HAL\_TIM\_PWM\_Start(&htim2,TIM\_CHANNEL\_1); //打开定时器2通道1

HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_1); //打开定时器3通道1

lcdinit();//LCD1602初始化

delay();

lcdshowstr(0,0,"Speed"); //在LCD1602输出起始提示字符

lcdshowstr(6,0,"0 ")；

1. 进入循环工作状态后，主程序不进行任何操作，HAL\_Delay(100);延时即可

所有程序均在中断服务程序进行

1. 编写LCD1602显示相关函数

void delay ()

{

for(int i=0;i<99;i++)

for(int j=0;j<99;j++)

{}

}

void lcdwritecmd(unsigned char cmd)

{

delay();

GPIOB->ODR=0x00; //E=0\RW=0\RS=0

GPIOA->ODR=cmd;

GPIOB->ODR=0x04; //E=1\RW=0\RS=0

delay();

GPIOB->ODR=0x00; //E=0\RW=0\RS=0

}

void lcdwritedata(unsigned char dat)

{

delay();

GPIOB->ODR=0x01; //E=0\RW=0\RS=1

GPIOA->ODR=dat;

GPIOB->ODR=0x05; //E=1\RW=0\RS=1

delay();

GPIOB->ODR=0x01; //E=0\RW=0\RS=1

}

void lcdinit()

{

lcdwritecmd(0x38);

lcdwritecmd(0x0c);

lcdwritecmd(0x06);

lcdwritecmd(0x01);

}

void lcdsetcursor(unsigned char x, unsigned char y)

{

unsigned char address;

if(y==0)

address=0x00+x;

else

address=0x40+x;

lcdwritecmd(address|0x80);

}

void lcdshowstr(unsigned char x, unsigned char y, unsigned char \*str)

{

lcdsetcursor(x,y);

while((\*str)!='\0')

{

lcdwritedata(\*str);

str++;

}

}

1. 编写中断服务程序

void HAL\_GPIO\_EXTI\_Falling\_Callback(uint16\_t GPIO\_Pin)

{

if(GPIO\_Pin==0x2000) //PC13

{

if(mode<5) //mode1到5轮转

mode++;

else

mode=1;

HAL\_TIM\_PWM\_Stop(&htim2,TIM\_CHANNEL\_1); //关闭定时器2通道1

MX\_TIM2\_Init\_NEW(mode); //重新配置定时器2

HAL\_TIM\_PWM\_Start(&htim2,TIM\_CHANNEL\_1); //关闭定时器2通道1

}

else if(GPIO\_Pin==0x0100) //PC8 定时器3-----输出+清零

{

HAL\_TIM\_PWM\_Stop(&htim3,TIM\_CHANNEL\_1); //打开定时器3通道1

if(number!=number\_old) //降低LCD1602刷新频率

{

number\_f=number\*30;

lcdshowstr(6,0," "); //清除原来的数据（显存）

sprintf(tstr,"%d",number\_f); //转换数据类型

lcdshowstr(6,0,tstr); //输出number\_f每分钟转速

number\_old=number; //刷新number\_old值

}

HAL\_TIM\_PWM\_Start(&htim3,TIM\_CHANNEL\_1); //打开定时器3通道1

number=0;

}

else if(GPIO\_Pin==0x0200) //PC9 风扇计数

{

number++;

}

else ;

}

1. 自定义新的定时器初始化函数

void MX\_TIM2\_Init\_NEW(int i) //自定义新的定时器初始化函数

{

TIM\_MasterConfigTypeDef sMasterConfig = {0};

TIM\_OC\_InitTypeDef sConfigOC = {0};

htim2.Instance = TIM2;

htim2.Init.Prescaler = 7;

htim2.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim2.Init.Period = 1000;

htim2.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim2.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_PWM\_Init(&htim2) != HAL\_OK)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL\_OK)

{

Error\_Handler();

}

sConfigOC.OCMode = TIM\_OCMODE\_PWM1;

sConfigOC.Pulse = 200\*i; //500

sConfigOC.OCPolarity = TIM\_OCPOLARITY\_HIGH;

sConfigOC.OCFastMode = TIM\_OCFAST\_DISABLE;

if (HAL\_TIM\_PWM\_ConfigChannel(&htim2, &sConfigOC, TIM\_CHANNEL\_1) != HAL\_OK)

{

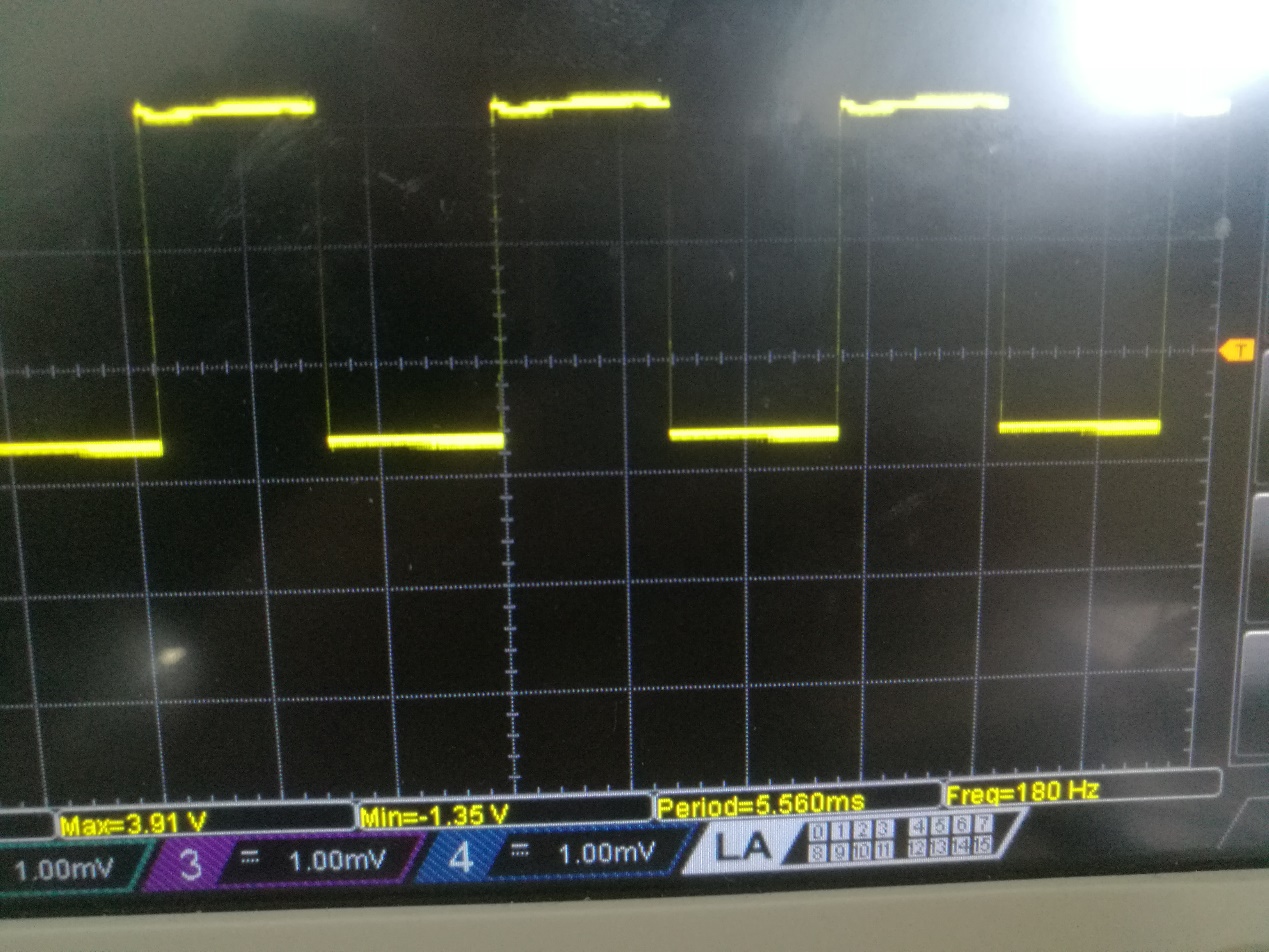
Error\_Handler();

}

HAL\_TIM\_MspPostInit(&htim2);

}

第六步 结果展示



由上图可知，测速信号线频率为180，转速为每分钟5400转，满足上述关系N=30F



由上图可知，测速信号线频率为284，转速为每分钟8460转，满足上述关系N=30F

第七步 设计亮点

PWM调速时采用以下流程：更改mode值，关闭定时器2通道1，调用自定义的定时器初始化函数根据mode值重新配置定时器2改变定时器驱动信号占空比。

定时器3下降沿触发清零时number\_old的设置，如果更新后的计数值number还等于原来的计数值，则不用刷新LCD1602显示屏，显示屏保持上一显示状态即可。在不影响结果的前提下，减少了LCD1602显示器刷新次数，提高MCU效率。

定时器3下降沿出发时，先关闭定时器3通道1，待LCD1602显示屏刷新成功后再打开，紧接着风扇测速信号计数值清零开始新的计数。如此设置，不论LCD1602刷新时间是多少，都不会影响对测速信号的计数周期，消除了显示屏刷新造成的延时影响，确保了计数周期为1S，确保了计数的准确性。