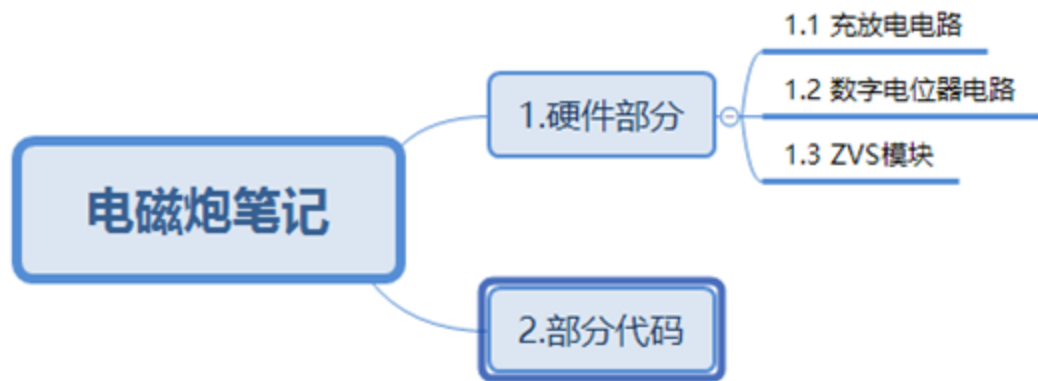




https://blog.csdn.net/weixin_42121843

1.硬件笔记

1.1充放电回路

电容充电电路采用继电器对充电电路部分的通断进行控制，放电电路部分通过70TPS12单向可控硅对电路开关进行控制。由此实现单片机对各部分开关的控制，D2作用为释放线圈多余能量。

三极管Q1放大输入信号，使其能驱动继电器

1.2 数字电位器电路

数字电位器用于控制ZVS的电压输出大小

芯片资料：（控制程序在后面）

BLOCK DIAGRAM

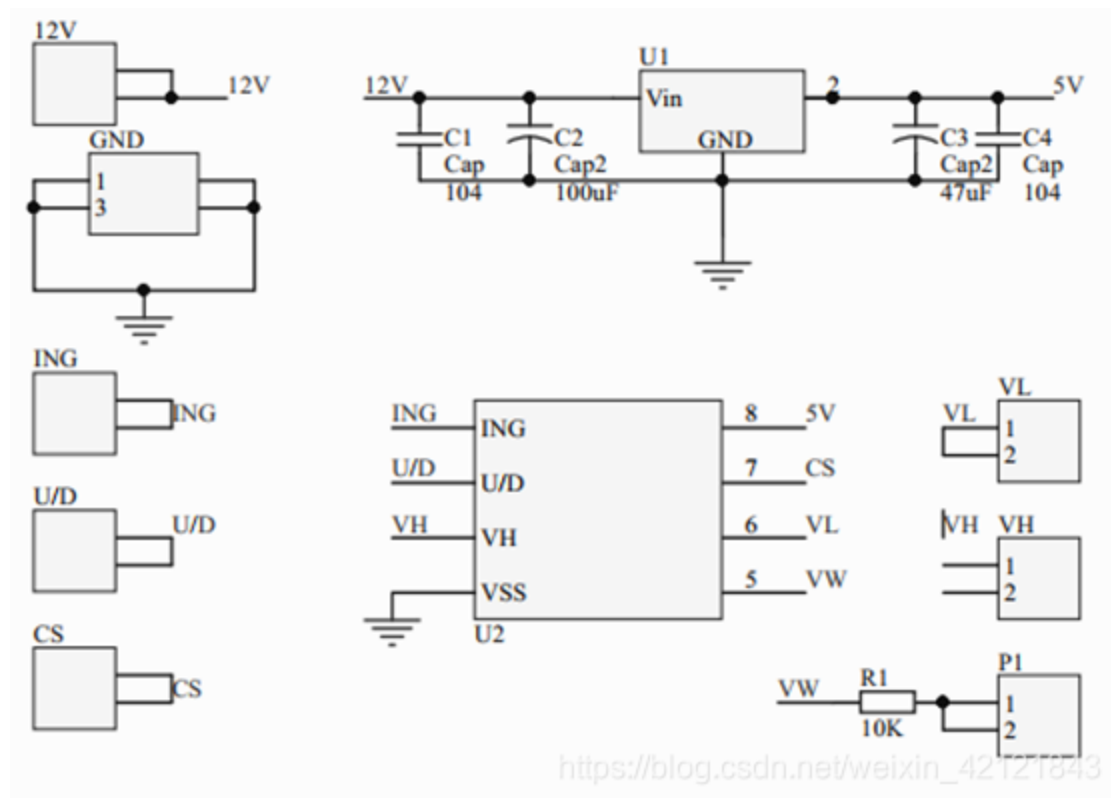


PIN DESCRIPTIONS

Pin	Symbol	Brief Description
1	INC	Increment . The INC input is negative-edge triggered. Toggling INC will move the wiper and either increment or decrement the counter in the direction indicated by the logic level on the U/D input.
2	U/D	Up/Down . The U/D input controls the direction of the wiper movement and whether the counter is incremented or decremented.
3	R_H/V_H	R_H/V_H . The high (V_H/R_H) terminals of the X9C102/103/104/503 are equivalent to the fixed terminals of a mechanical potentiometer. The minimum voltage is -5V and the maximum is +5V. The terminology of V_H/R_H and V_L/R_L references the relative position of the terminal in relation to wiper movement direction selected by the U/D input and not the voltage potential on the terminal.
4	V_{SS}	V_{SS}
5	V_W/R_W	V_W/R_W . V_W/R_W is the wiper terminal, and is equivalent to the movable terminal of a mechanical potentiometer. The position of the wiper within the array is determined by the control inputs. The wiper terminal series resistance is typically 40Ω.
6	R_L/V_L	R_L/V_L . The low (V_L/R_L) terminals of the X9C102/103/104/503 are equivalent to the fixed terminals of a mechanical potentiometer. The minimum voltage is -5V and the maximum is +5V. The terminology of V_H/R_H and V_L/R_L references the relative position of the terminal in relation to wiper movement direction selected by the U/D input and not the voltage potential on the terminal.
7	$\overline{CS}$	$\overline{CS}$. The device is selected when the $\overline{CS}$ input is LOW. The current counter value is stored in nonvolatile memory when $\overline{CS}$ is returned HIGH while the INC input is also HIGH. After the store operation is complete the X9C102/103/104/503 device will be placed in the low power standby mode until the device is selected once again.
8	V_{CC}	V_{CC}

https://blog.csdn.net/weixin_42121843

数字电位器选用具有100抽头的X9C503芯片，可调电阻范围为0-50k欧，可以通过单片机调节数字电位器进而对ZVS升压模块输出电压进行控制，电路图如图 所示。



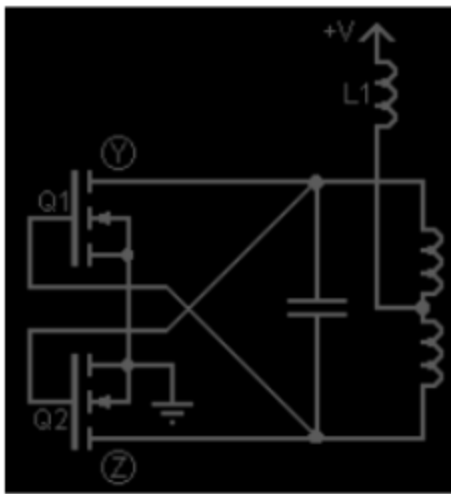
https://blog.csdn.net/weixin_42121843

1.3 ZVS模块

原理：

它通常被用于产生高频正弦波的场合，电磁炮中这个电路可用于逆变升压，整流后的高压电可以给电容充电。

这里有一个简化版的ZVS，以说明其原理。

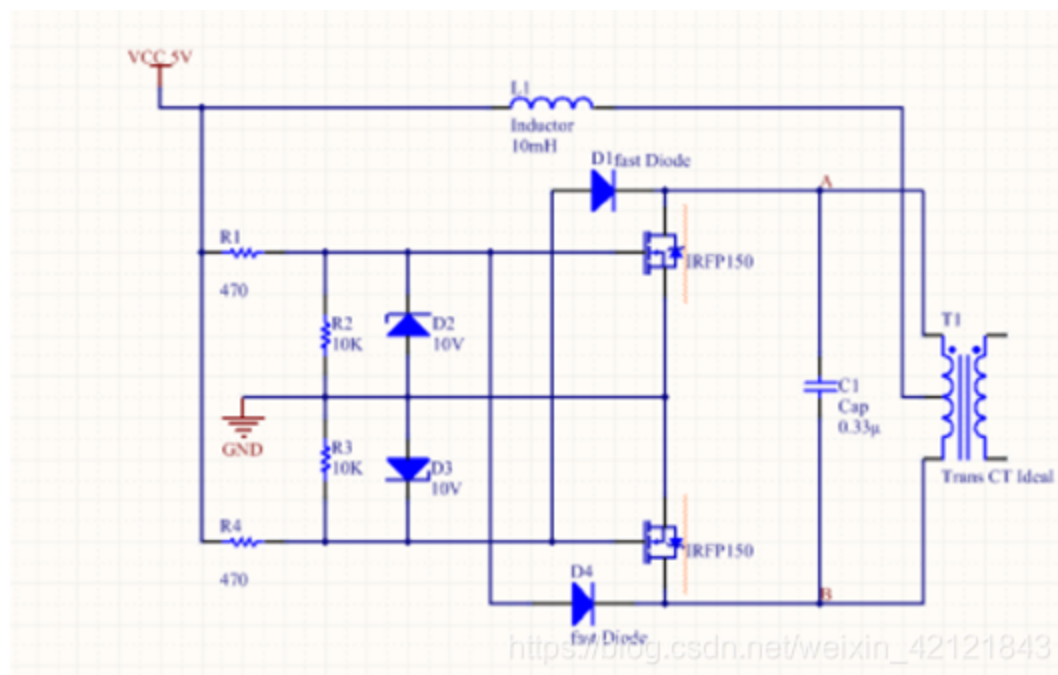


当电源电压作用于 $V+$ ，电流开始同时通过两侧的初级并施加到 MOS 的漏极（D）上。电压会同时出现在 MOS 的门极（G）上并开始将 MOS 开启。因为没有任何两个元件是完全一样的，一个 MOS 比另一个开的快一些，更多的电流将流过这个 MOS。通过导通侧初级绕组的电流将另一侧 MOS 的门极电压拉低并开始关断它。图中电容和初级的电感发生 LC 谐振并使电压按正弦规律变化。如果没有这个电容，通过 MOS 的电流会一直增大，直到器件损坏。

假设 Q1 首先开启。当 Z 点电压跟着 LC 谐振的半个周期上升到峰值再回掉时，Y 点电压会接近 0。随着 Z 点电压下降到 0，Q1 的门极（G）电压消失，Q1 关闭。同时 Q2 开启，此时 Y 点电压开始上升。Q2 的导通把 Z 点电压拉低到接近地，这可以确保 Q1 完全关断。Q2 完成 LC 振荡的半周后会重复同样的过程，此振荡器继续循环工作。为了防止本电路从电源拉取巨大的峰值电流而损坏，增加了 L1 在变压器抽头处和 $V+$ 之间作为缓冲。LC 阻抗限制着实际的电流（L1 只是减少峰值电流，因为电感有续流作用）。

此振荡器是一个零电压开关电路（zero-voltage switching ZVS），这意味着 MOS 将在其两端电压为零时关断。这对 MOS 有好处，因为它允许 MOS 在承受应力比较低的时候进行开关动作，这意味着不再需要像硬开关变换器那样的巨大散热器。

原理图及器件选型



原理分析：

上电瞬间，假设 Q1 通得多那么一点，T1 的感应电流使 B 点之电动势略高于 A 点，而且 Q2 处于有一点导

通的状态，这些条件都是有利于彻底关断 Q2 的，那么 Q2 就关断，Q1 导通，T1 的感应电流彻底把 C1 充成 B+、A-，
然后，C1 放电完（或者 A 点电位有那么一点点的正），而且 Q1 处于导通状态，马上 Q1 之 G 极通过 D4，C1，Q1 被拉到相当低的电位而截止，Q2 之 G 极自然被 R4 拉到 5V 而导通，完成了在零电压状态时的切换，然后 T1 的感应电流彻底把 C1 充成 A+、B-，之后 C1 放电完毕又致 Q1 导通 Q2 截止。如此循环下去。

原理图中器件作用：

- 1.R1为驱动电阻，阻值较小，用于防止震荡；R2为GS电阻，阻值较大，用于结电容放电，加快开关速度。
2. 稳压二极管将MOS的门极（G）的电压限制在稳压二极管（12V、15V、18V）的击穿电压之内
3. 当一侧MOS导通时，UF4007将另一侧MOS的门极（G）电压拉低
- 4.谐振部分电容耐压要高，最好选电磁炉电容。
- 5.mos管耐压要四倍的输入电压以上

2.部分代码（STM32）

2.1 数字电位器控制代码

```
//片选信号PB6
#define CS_H GPIO_SetBits(GPIOE,GPIO_Pin_4)
#define CS_L GPIO_ResetBits(GPIOE,GPIO_Pin_4)
//方向选择引脚PB7
#define UD_H GPIO_SetBits(GPIOE,GPIO_Pin_5)
#define UD_L GPIO_ResetBits(GPIOE,GPIO_Pin_5)
//脉冲信号引脚PB8
#define INC_H GPIO_SetBits(GPIOE,GPIO_Pin_6)
#define INC_L GPIO_ResetBits(GPIOE,GPIO_Pin_6)

/*X9C103初始化*/
void x9c_init(void)
{
    IO_Init();
    x9c_dec_step(99);
    x9c_inc_step(1);
}
/*递增*/
void x9c_inc_step(unsigned char num)
{
    unsigned char i=0;
    UD_H;
    CS_L;
```

```

    for(i=num;i>0;i--)
    {
        INC_H;
        delay_us(20);
        INC_L;
        delay_us(100);
    }
    INC_H;
    CS_H;
}
/*递减*/
void x9c_dec_step(unsigned char num)
{
    unsigned char i=0;
    UD_L;
    CS_L;
    for(i=num;i>0;i--)
    {
        INC_H;
        delay_us(20);
        INC_L;
        delay_us(100);
    }
    INC_H;
    CS_H;
}

```

2.2舵机控制代码

```

void setHorizontalAngle(int angle)
{
    uint32_t compare=0;
    //if(angle>30) angle=30;
    //if(angle<-30) angle=-30;
    if(angle>=0)
        compare = 1.515*40000/20+(2.48-1.515)*40000*angle/20/135;
    else
        compare = 1.515*40000/20-(1.515-0.49)*40000*(-angle)/20/135;
    TIM_SetCompare3(TIM4,compare);
}
void setVerticalAngle(int angle)
{

```

```

uint32_t compare=0;
if(angle>90) angle=90;
if(angle<0) angle=0;
if(angle>=0)
    compare = 2.14*40000/20-(2.14-0.57)*40000*angle/20/225;
else
    compare = 2.14*40000/20+(2.5-2.14)*40000*(-angle)/20/45;
TIM_SetCompare4(TIM4,compare);
}

void Servo_Reset(void)
{
    setHorizontalAngle(0);
    setVerticalAngle(42);
}

void Servo_cruise(){
    uint16_t delta = 2574;
    uint8_t dir = 0;
    setHorizontalAngle(-30);

    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
    while(1){
        TIM_SetCompare3(TIM4, delta);
        if(dir==0){
            if(x_pos<-2&&x_pos>-5) break;
            delta += 4;
        }
        else if(dir == 1){
            if(x_pos>2&&x_pos<5) break;
            delta -= 4;
        }
        if(dir==0&&delta>=3445){
            dir = 1;
            delay_ms(50);
        }
        else if(dir==1&&delta<=2574){
            dir = 0;
            delay_ms(50);
        }
        delay_ms(5);
    }
    USART_ITConfig(USART1, USART_IT_RXNE, DISABLE);
}

```

2.3 PID控制代码

```
PID pid_A;
PID pid_B;

Float speed1;
Float speed2;
float kp = 0.17/2;// 0.05
float ki = 0.15/2;//0.05
float kd = 0.2/2;
/*float kp = 0.35;
float kd = 0.5;*/
//float kp = 0.000175f;
//float kd = 0.00025f;

void PID_init(void){
    pid_A.exception = 0;
    pid_A.cur_error = 0;
    pid_A.last_error = 0;
    pid_A.last_last_error = 0;

    pid_A.Kp = kp;
    pid_A.Ki = ki;
    pid_A.Kd = kd;

    pid_A.deltaU = 0;
    pid_A.U = 0;
}

void PID_setException_A(float exception){
    pid_A.exception = exception;
}
void PID_setException_B(float exception){
}
void PID_cal_A(void){
    pid_A.cur_error = x_pos - pid_A.exception;
    pid_A.deltaU = pid_A.Kp*(pid_A.cur_error-pid_A.last_error) + pid_A.Ki*pid_A.cur_err
    pid_A.U += pid_A.deltaU;
    pid_A.last_last_error = pid_A.last_error;
    pid_A.last_error = pid_A.cur_error;
    PWM_limitAmp_A();
}
void PID_cal_B(void){
}
```

```
void PWM_limitAmp_A(void){
    if(pid_A.U >= 1000)
        pid_A.U = 1000;
    if(pid_A.U <= -1000)
        pid_A.U = -1000;
}
void PWM_limitAmp_B(void){
}
void PID_setMotor_A(void){
    TIM_SetCompare3(TIM4, 3000+pid_A.U);
}
void PID_setMotor_B(void){
}
```

实物图

