电子烟方案基本的功能分为:充电管理、按键及咪头检测、显示部分（LED 灯、LED 屏、LCD 屏功能）、输出(恒压输出、恒有效输出、恒功率输出)及保护、电池电量管理.

1.1. 选型及开发环境

官网地址：www.holychip.cn

1.1.1. OTP

ROM_2K-SQL5820（MSOP10）
ROM_4K-HC18P133L（QFN20）

开发环境：
编译软件：HCIDE_V3.0.18.9S
上位机软件：HC-PM18 V5.0.23.0/HC-ProgrammerV1.0.5.4
烧录器：HC-PM18/HC-Programmer
仿真工具：黑板/HC-LINK_V5
Tips:官网获取最新版本

删除[张 FEI]: ...

1.1.2. MTP

ROM_2K-HC18M582X（HC18M5821-QFN16、HC18M5820-MSOP10）
ROM_4K-HC18M5830（QFN20）
ROM_8K-HC18M584xA（HC18M5841A-QFN24（3*3）、HC18M5840A-QFN20）--支持带电升级
ROM_8K-HC18M003A（QFN24（4*4）、QFN20、QFN16）--支持带电升级

集成电子烟芯片：
ROM_4K-HC18M5831B（QFN24（3*3））--支持带电升级

删除[张 FEI]: ...

开发环境：
编译软件：HCIDE_V3.0.18.9S
上位机软件：HC-PM18 PRO V1.0.2.3/HC-ProgrammerV1.0.5.4
烧录器：HC-PM18Pro/HC-Programmer
仿真工具：HC-ICD PRO/HC-LINK_V5
Tips:官网获取最新版本

1.1.3. FLASH--支持带电升级

ROM_16K-HC89S5840（QFN20）
ROM_32K-HC89S5860B（QFN32）
ROM_64K-HC89S5860（QFN32）

开发环境：
编译软件： Keil C51 + 支持包 HC-LINK V4.0.18.14--LINK V4 调试器
Keil C51 + 支持包 HC-LINK V5.0.0.6--LINK V5 调试器

删除[张 FEI]: ...
...

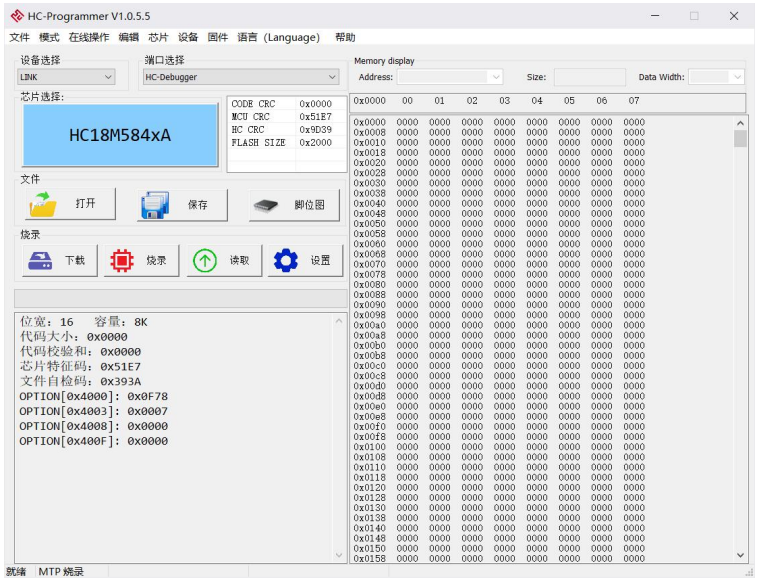
上位机软件：HC-PM51 V6.0.21.14/HC-ProgrammerV1.0.5.4
烧录器：HC-PM51/HC-Programmer
仿真工具：HC-LINK_V4/HC-LINK_V5
其他：ISP 固化、HC-ISP 工具上位机

Tips:官网获取最新版本

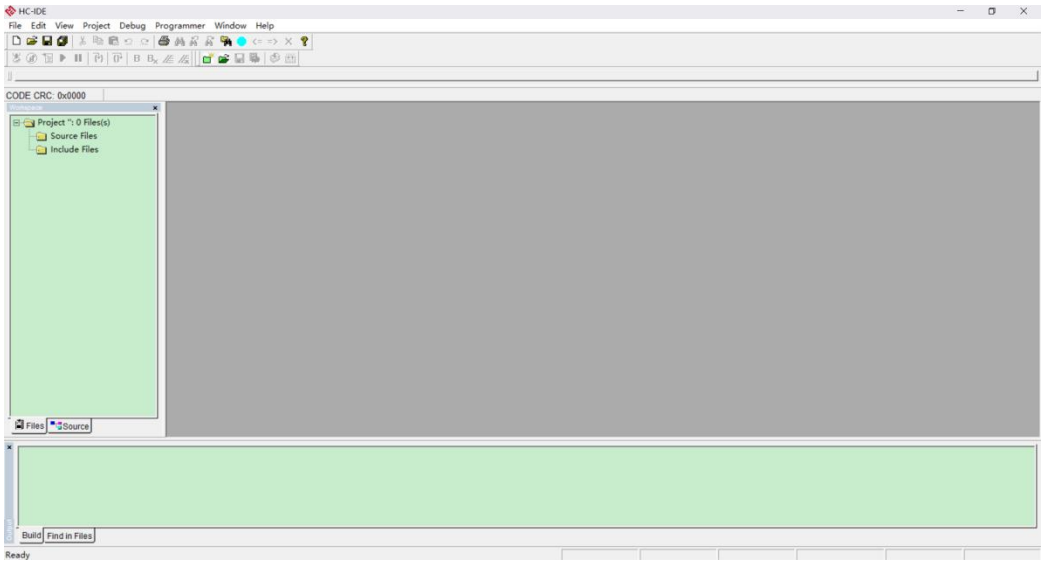
1.1.4. 工具图片

LINK V5：

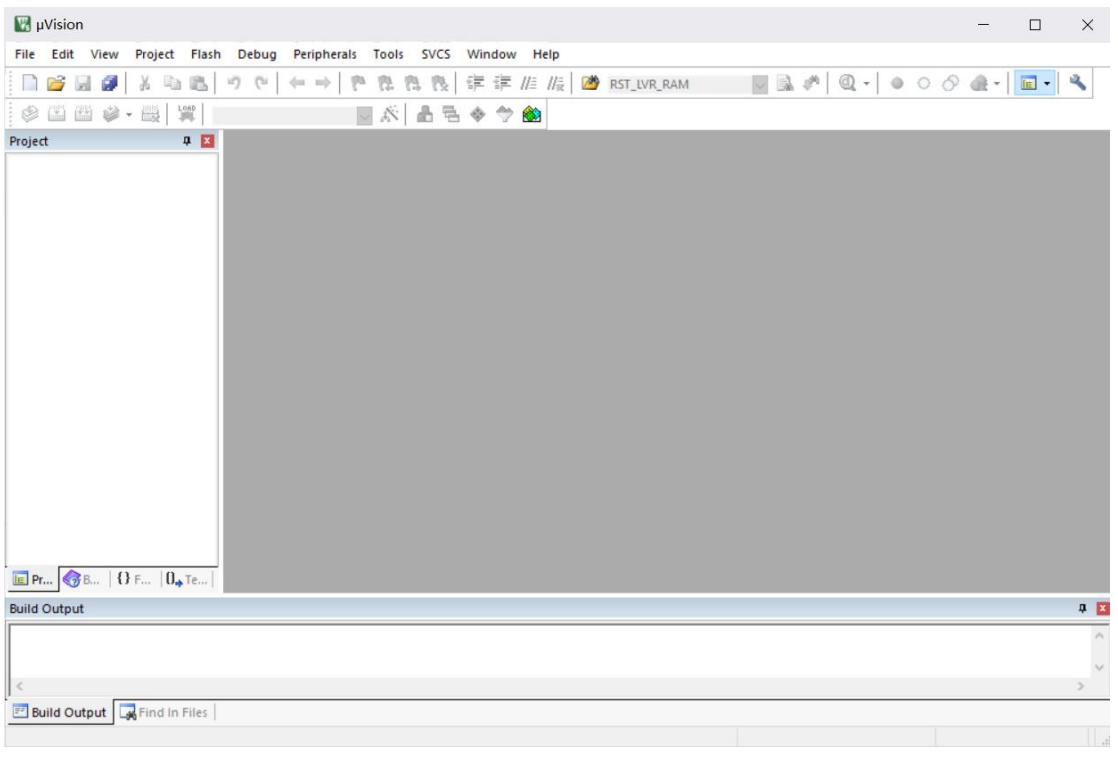
上位机软件：HC-ProgrammerV1.0.5.4



开发环境 IDE：HCIDE_V3.0.18.9S--OTP/MTP 系列芯片



开发环境 IDE：KEIL C51

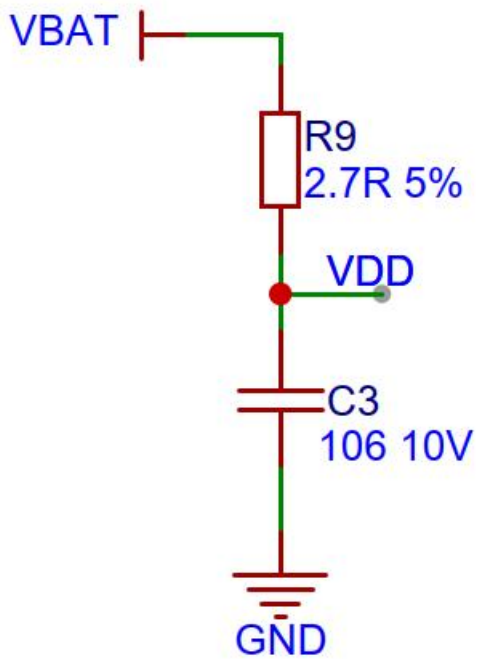


1.2. 应用注意

1.2.1. VCC 防浪涌保护

焊接电池瞬间，电源端存在较大浪涌电流，可能引起芯片上电瞬态过冲或地电位漂移导致失效。此处建议增加 RC。接地点靠近芯片的 GND。

另并联 TVS 管，吸收 VBAT 端浪涌尖峰



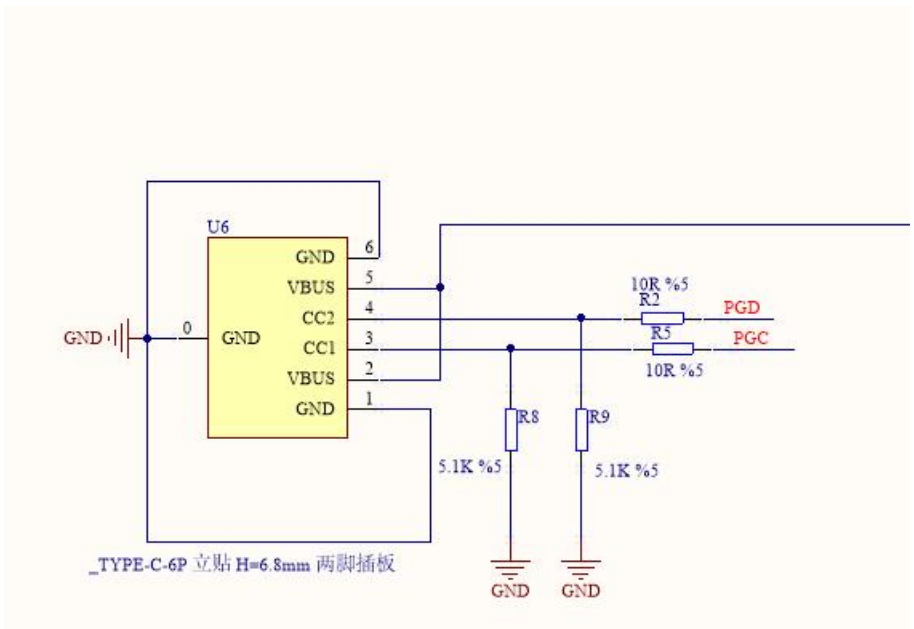
1.2.2. 烧录及带电升级注意

OTP\MTP 系列芯片烧录需 VDD\GND\PGC\PGD\PCK\VPP 引脚，不建议在板烧录。

HC18M584xA 芯片的烧录 VDD\GND\PA0\ PA1,支持在板烧录，注意编程引脚的外围电路需进行针对性设计， 以保证外围电路不会影响在板编程时端口上的电压、电流、时序等。

带电升级下，需将芯片的内部 DBG0EN 进行使能，且此 PA0 PA1 连接到 USB 接口的 CC1/CC2 管脚，用做方案的带电升级功能，便于后续产品升级。

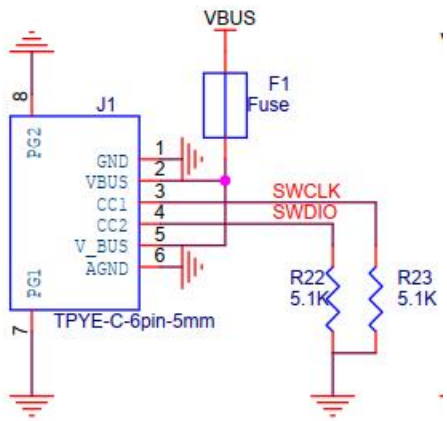
参考原理图：



FLASH 芯片系列：芯片的烧录口是 SDA、 SCK，在板烧录需要注意外围电路不能影响此端口上时序及电压。

带电升级下，芯片中需要保证已经固化 ISP 程序，且 SDA\SCK 连接到 USB 接口的 CC1/CC2

管脚。



1.2.3.休眠时处理

所有的 IO 口，在睡眠时，可能会由于浮空状态出现漏电情况 故所有 IO 在进入睡眠时，都应配置为一个固定的状态，如：输入下拉、输入上拉等。

ADC，关闭 ADC 电源， 切换到内部参考电压（内部 2V、1.3V）,避免 ADC 漏电功耗。

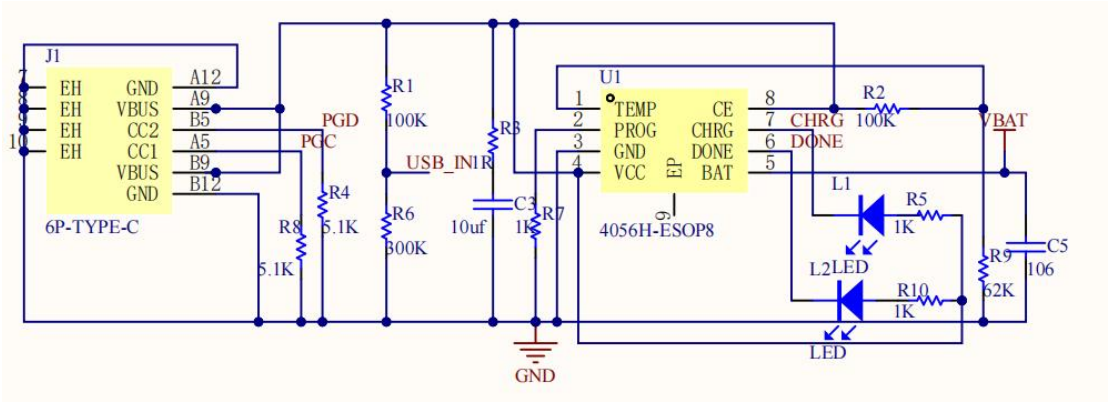
BOR ，关闭 BOR ，唤醒后重新使能。

带电升级：M584XA 休眠前需要关闭 DBG0EN，配置 PA0、PA1 唤醒，唤醒 SLEEP 后，重新使能 DBG0EN。

1.3. 充电管理

主要检测 USB 插入，充电状态，满电状态，以及部分方案带有 OVP 保护

1.3.1. 典型原理图



1.3.2. 充电相关说明

详细参数见 4056H 规格，编写方案时，需要拿到充电管理芯片的资料，便于编写程序，判断各种状态

PGC/PGD 接口可用于 MTP：HC18M003A/HC18M584xA/HC18M5831B...

FLASH：HC89S5840/HC89S5860/HC89S5860B...

等芯片的带电升级功能

USB_IN 接口：用于检测 USB 是否插入，需要根据芯片的翻转电平进行配置，一般建议 2/3 或者 3/4 的比例；避免 1/2 的比例分压，电平判断会处于临界状态，不容易判断到。

CHRG 接口：充电指示灯，电池充电时为低电平，不充/满电为高电平

DONE 接口：充满指示灯，电池充满时为低电平，不充/未满为高电平

一般情况下，仅用 CHRG 一个 IO 判断充电和满电状态，配合 1/4VDD 通道采集电池电压进行判断，避免暂用 IO 资源

若充电管理芯片不带 OVP，则需要通过 USB_IN 的电压进行检测并保护，具体看方案需求

1.3.3. 充电部分软件说明

1.3.3.1. USB 插入检测判断：

原理：通过检测 USB_IN 的高低电平来判断是否接入 USB

软件需要配置 IO 为数字输入，不需要配置内部上下拉 注意滤波，避免误判断 置相关标志位 Flag

注意：部分应用中，USB 接入后不可吸烟，需要做相关处理，确认好需求；若有边充边放的功能，一般需要控制充电管理芯片的 EN 脚位，将其断开充电

1.3.3.2. 充电状态判断：

原理：在判断 USB 插入后，判断充电管理芯片的 CHRG 脚位的状态 高电平-->满电/未充电 低电平-->充电中

软件配置相关 IO 为数字输入 注意滤波，避免误判断 置相关标志位 Flag

若 CHRG 为低电平，则为充电状态 做相关的状态显示

注意：插入 USB 时，默认充电状态（理解为 USB 插入后相关充电状态需要显示，比如充电时的呼吸灯或者闪烁等），再判断 CHRG 的脚位的状态，调整相关 Flag 标志位，修改显示状态

1.3.3.3. 满电状态判断

原理：在判断 USB 插入后，判断充电管理芯片的 CHRG 脚位的状态 高电平-->满电/未充电 低电平-->充电中

软件配置相关 IO 为数字输入 注意滤波，避免误判断 置相关标志位 Flag

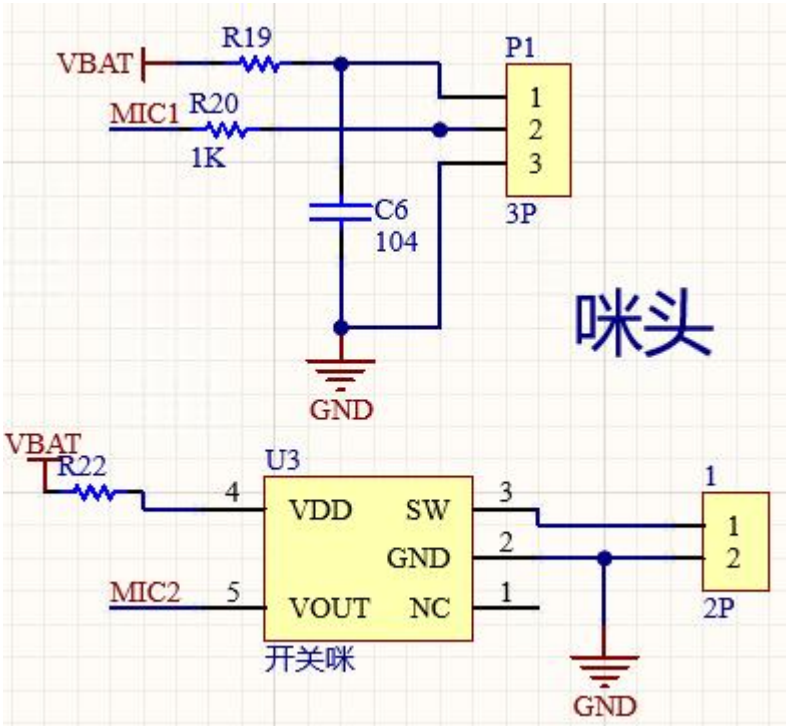
若 CHRG 为高电平，且采集 1/4VDD 电压也为满电状态，则根据需求进行相关显示

一般可能会继续充几分钟的时间再显示满电（根据需求文档考虑）

1.4. 按键/开关

主要包含咪头开关、按键等

1.4.1. 咪头



常见的咪头有三线咪、两线咪、硅麦等，原理都是采集咪头输出信号进行判断是否在吸烟

三线咪头：VDD、OUT、GND，三根线，一般未触发的时候 OUT 为低电平，触发时为高电平

两线咪头：OUT、GND 空咪+开关咪 一般未触发的时候 OUT 为低电平，触发时为高电平

硅麦：一般未触发的时候 OUT 为低电平，触发时为高电平

具体看方案相关资料进行调整，注意判断电平时的滤波算法

1.4.2. 按键

一般按键用于切换模式、预热、开关机等

注意外部的上下拉，评估原理图时，需要确认客户选择的芯片 IO 是否具备唤醒功能

1.4.3. 软件部分

1.4.3.1. 按键部分

原理：通过判断按键的高低电平，给出对应的 Flag--->对应不同的功能

软件配置：对应 IO 需要配置数字输入，原理图中若无外部上下拉，则需要配置内部上下拉 一般需要做唤醒，注意在 SLEEP 前做好唤醒初始化 注意滤波处理

1.4.3.2. 咪头部分

原理：通过判断咪头输出的信号，给出对应的 Flag

软件配置：对应 IO 需要配置数字输入，原理图中若无外部上下拉，则需要配置内部上下拉 一般需要做唤醒，注意在 SLEEP 前做好唤醒初始化 注意滤波处理

1.5. 显示部分

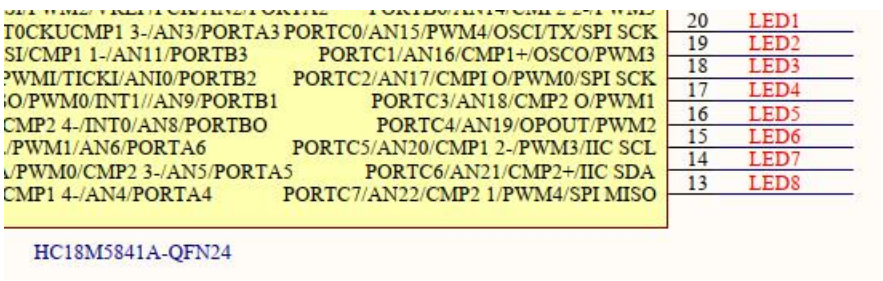
1.5.1. 硬件说明

包含 LED 显示（单色、三色、幻彩、188 等）、LCD 显示

评估原理图时，部分 IC 的拉灌电流档位可选，或部分 IC 里面的不同组 IO 的拉灌不一致，需要参考芯片数据手册；另外带屏显的方案，需要评估芯片的 ROM 空间大小，避免 ROM 不够。

HC18M584XA:

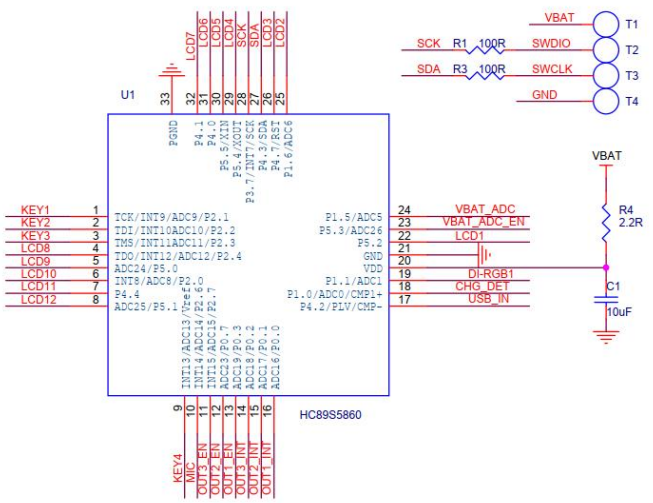
188 显示： 推荐使用 PORTC 组 IO，拉灌电流：14mA/42.5mA。



幻彩:支持 2 路幻彩输出，IO 选择，第一路从 PA4\PC4\PB5\PA2 中选择 1 个。第二路可从 PA5/PC2/PB1/PA0 中选择 1 个。

HC89S5840/5860

LCD：根据实际来选择引脚。



1.5.2. 软件说明

LED 显示：主要为闪烁和呼吸，若原理图选择的 IO 没有 PWM，则需要利用 TIMER 模拟 PWM

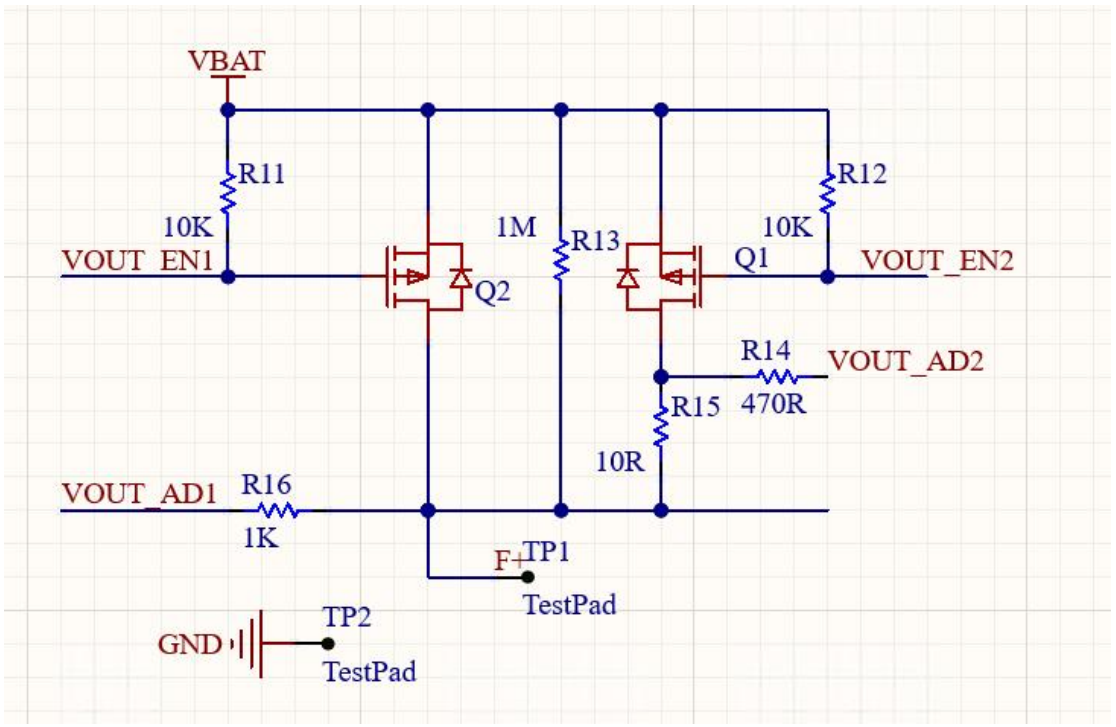
幻彩：幻彩 LED 需要提醒客户原理图上选择 PWM0/PWM2 的 IO 需要使用 LED 级联功能

LCD 显示：根据客户给的真值表进行编写 注意显示刷新率，避免显示抖动；注意刷屏时的 IO 状态，避免鬼影

1.6. 输出部分

1.6.1. 硬件部分

输出一般为恒压、恒功率、恒有效三种输出方式
通过控制 PMOS 输出，外接负载电路进行加热
典型电路如下：以下为恒功率电路 恒功率电路需要检测负载阻值



说明：

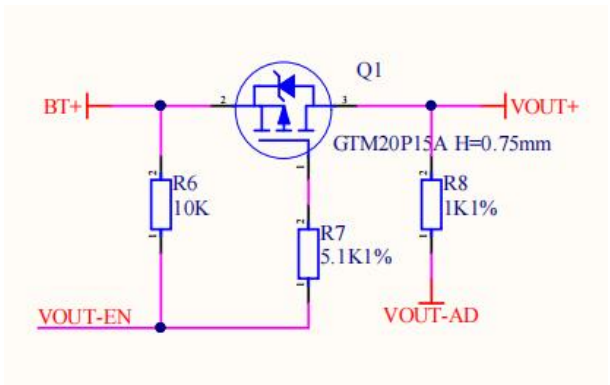
VOUT_EN1：PMOS 控制端 G

VOUT_AD1：VOUT_EN2 开启下检测负载阻值/短路保护判断

R15：通过与负载分压检测负载阻值 R13：上拉，负载插拔用途。

VOUT_EN2：PMOS 控制 G 用于检测负载阻值

典型电路如下：以下为恒压电路 不需要负载阻值检测



删除[杨磊]: 输出端电压采集

删除[杨磊]: R13：负载检测 作用：上拉

删除[杨磊]: VOUT_AD2：输出电压采集 用于检测负载阻值

设置格式[holychip-665]: 两端对齐，缩进：左侧: 22.2 毫米，首行缩进: 7.4 毫米

1.6.2. 软件部分

1.6.2.1. 输出控制

原理：控制 PMOS 的 G 端来导通 PMOS

软件：一般控制的 PWM 频率为 100Hz（10ms）

恒压：通过采集 1/4VDD，得到电池电压，Vout = Vbat * D 通过此公式，得到 PWM 占空比，启动输出 注意：PMOS 的内阻需要考虑，需要微调 Vout 的理论值

恒功率：通过采集 1/4VDD，得到电池电压，通过公式 $P = U^2/R$ 得到 Vout = P * R / Vbat / Vbat * D

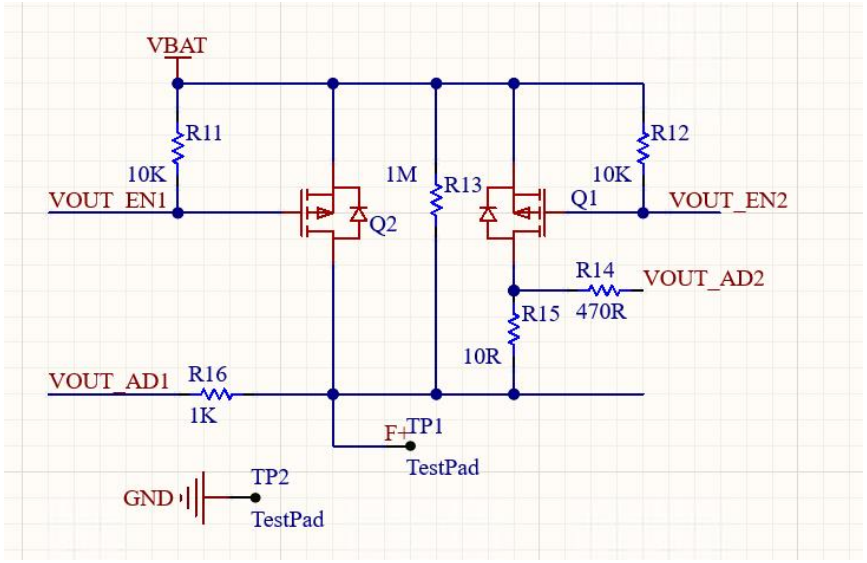
恒有效：通过采集 1/4VDD，得到电池电压，通过公式 $Vout^2 = Vbat^2 * D$

以上三种公式均通过电池电压计算，得到占空比 需要注意 IDE 的编译效率

```
    /*注意：
    1、输出控制时 不允许全功率输出 必须以PWM形式控制MOS
    2、SmokeVoltage恒压控制需要考虑硬件因素 线损以及MOS参数等 需软件补偿
    */
    /////恒压输出
    SmokingClk = SmokeVoltage / BeBatteryData;
    /////恒功率输出
//  SmokingClk = SmokePower / BeBatteryData / BeBatteryData;
    /////恒有效输出
//  SmokingClk = SmokeVo / BeBatteryData / BeBatteryData;
    if(SmokingClk >= SmokeClkMax)
    {
    SmokingClk = SmokeClkMax;
    }
    if(SmokingClk <= SmokeClkMin)
    {
    SmokingClk = SmokeClkMin;
    }

    uOutDuty = SmokingClk;
    isSmokeErr = 0;
    isSmoke = 1;
```

恒功率计算：



VOUT_EN1：PMOS 控制端 G

VOUT_AD1：VOUT_EN2 开启下检测负载阻值/短路保护判断

R15：通过与负载分压检测负载阻值 R13：上拉，负载插拔用途。

VOUT_EN2：PMOS 控制 G 用于检测负载阻值

原理：VBAT —— 10Ω(R15) —— RF(发热丝) —— GND（忽略 MOS 内阻）

|

Vout_AD1

- VBAT：电源电压（通过 1/4VDD 通道,内部参考电压 2V）
- VF：RF 上的电压（通过 Vout_AD1 通道，VDD 做参考）

Vrf = R15* (Vbat - VF)/VF

```
VOUT_EN2 = 0;      //开启 MOS
Delay_Us(200);
isLcdWait = 1;      //关闭 LED\LCD 扫描

uTempResVol2 = Get_Adc(10);//分压检测 Vrf---内参 VDD
uTemp3 = Get_Adc(3);
g_tmpbuff = MUL_U32(uTemp3,8000); // 4*2V*1000 (mV)
uTempResBattery2 = DIV_U32(g_tmpbuff,4096);//反推得到 BAT 电压
VOUT_EN2 = 1;   //关闭 MOS
isLcdWait = 0;    //恢复刷新

uTemp_1 = MUL_U32(uTempResVol1,1000); //先把 at 转换成电压
uTemp_2 = DIV_U32(uTemp_1,4096);
uTemp_1 = MUL_U32(uTemp_2,uTempResBattery1);     //等到 Urf 负载的电压
```

删除[杨磊]: 输出端电压采集

删除[杨磊]: R13：负载检测

删除[杨磊]: 作用

删除[杨磊]: ：

删除[杨磊]: VOUT_AD2：输出电压采集 用于检测负载阻值

删除[杨磊]: 2

```
    uTemp_2 = DIV_U32(uTemp_1,1000);            //
    uTemp_3 = uTempResBattery1 - uTemp_2;
    uTemp_1 = MUL_U32(uTemp_2,1000);
    uTempRes1 = DIV_U32(uTemp_1,uTemp_3);//得到阻值

  uTempPower = 13000;//12500     //功率

  uTemp_1 = MUL_U32(uTempPower,uTempRes1);
  uTemp_2 = DIV_U32(uTemp_1,(uTemp11));
  uTemp_3 = MUL_U32(uTemp_2,SMOKE_CONVERT_OUTPUT_FREQUENCY);
  uTemp_1 = MUL_U32(uTemp_3,10);          //10R
  uPowerTemp = DIV_U32(uTemp_1,(uTemp11));
  //占空比输出
  if(uPowerTemp > SMOKE_CONVERT_OUTPUT_FREQUENCY)
    uPowerTemp = (SMOKE_CONVERT_OUTPUT_FREQUENCY);//防溢出
  uOutDuty = uPowerTemp;
```

```
u32 MUL_U32(u32 multiplier, u32 multiplicand)
{
        u32             product = 0;

        do {
                if(multiplier & 1)
                        product += multiplicand;
                multiplicand <<= 1;
                multiplier >>= 1;
        } while(multiplier != 0);
        return product;
}


u32 DIV_U32(u32 dividend,u32 divisor)
{
    u32             DIV_HighBit=0x80000000;
    u32                     quotient = 0;
    u8                      counter;

    if(divisor != 0) {
            counter = 1;
            while((divisor & DIV_HighBit) == 0) {
                    divisor <<= 1;
                    counter++;
            }
```

```
                do {
                        quotient <<= 1;
                        if(divisor <= dividend) {
                                dividend -= divisor;
                                quotient |= 1;
                        }
                        divisor >>= 1;
                } while(--counter != 0);
        }
        remainder = dividend;
        return quotient;
}
```

1.6.2.2. 短路保护

原理：负载瞬间短路，PMOS 还处于导通状态，VBAT 和 GND 短路，需要及时关闭 PMOS，否则会烧坏
软件：在中断中判断保护响应更快，一般在 TIMER 中断里面处理
在 PMOS 导通的时候，判断输出端的电平，瞬间短路 VOUT_AD 的电平会拉低，此时需要立即关闭 PMOS .

注意复位产生，若产生 BOR 复位，则需要调整复位后的参数，避免数据丢失。大多数情况下，短路测试不会复位，这里为了覆盖 BOR 复位情况，通过非完整掉电保存参数方式进行记忆处理，区分正常上电还是复位导致的上电。

设置格式[holychip-665]: 缩进: 左侧: 0 毫米, 首行缩进: 0 毫米

简单说明：

1.定义掉电保存的参数，persistent volatile U16 Password;
2.在触发吸烟后，赋值该参数 Password = 0x55aa。
3.在吸烟结束后清除该参数 Password = 0 。
4.在上电阶段判断该参数

设置格式[holychip-665]: 缩进: 左侧: 7.4 毫米, 首行缩进: 7.4 毫米

```c
void Sys_Init(void)
{
    Gpio_Int(); //GPIO初始化
    //系统上电状态判断
    if( (POR == 0 ) && (Password == 0X55aa ))   //系统产生了复位上电
    {
        LED_Show_RLSHORT();
    }
    else
    {
        Ram_Int();      ///RAM清零
    }
```

软件 PWM 短路检测：

设置格式[holychip-665]: 4 级

吸烟前短路：开启 PMOS 判断 VOUT_AD 电平或 ADC 数据。

```
//////注意：该方式必须添加滤波 避免IO抖动误触发
//////注意：使用该方式时 IO模式必须设置为数字输入状态
ANSELH = 0B00000000;          ///切换数字模式
TRISB2 = 1;    ///配置输入状态
CTRL_MOS = PMOS_ON;
Delay_Ms(5);  ////延时5-10ms保持稳定

if(!DET_Short)    ////短路
{
    ShortCnt += 1;
    if(ShortCnt >= 20)
    {
    ShortCnt = 0;
    PORTB0 = 1;   /////PB0为高   用于查看现象
    }
}
if(DET_Short)    ////正常
{
    NoShortCnt += 1;
    if(NoShortCnt >= 20)
    {
    NoShortCnt = 0;
    PORTB0 = 0;
    }
}
```

吸烟中短路：需要在 PMOS 开启的时候，判断 VOUT_AD

说明：if 判断中 (uOutClk >= 1) && (uOutClk < uOutDuty) 保证在 PMOS 开启的时候判断 VOUT_AD

```
if((uOutClk >= 1) && (uOutClk < uOutDuty) && (!isSmokeAd))
{
    if((VOUT_EN == 0) && (VOUT_AD == 0))
    {
        VOUT_EN = 1;
        isSmokeLvd = 1;
    }
}
```

硬件 PWM 输出短路检测：

HC18M584xA/9229 系列

PWM 配置：

```
//20khz---50us
void PWM1_Int(void)
{
    //bit 4-6    PWM01 时钟分频选择
    //PWM01CON0 |= 7 << 4;    //128 分频   32MHZ /128 = 0.25MHZ
    //PWM01CON0 |= 0x50;   //32 分频
    PWM01CON0 |= 0x40;   //16 分频
    ///输出类型选择    0：边沿对齐   1：中心沿对齐
    TY01          = 0;
    ///bit0：  LED0 级联控制      bit1：LED0 级联发送数据控制
    ///bit2：  PWM0 输出取反控制  bit3：PWM1 输出取反控制
    PWM01CON1   = 0x08; //低有效，反向开启
    ///PWM 周期寄存器
    PWM01TH      = 0x00;  //20khz
```

设置格式[holychip-665]: 缩进: 左侧: 14.8 毫米, 首行缩进: 7.4 毫米

设置格式[holychip-665]: 缩进: 首行缩进: 0 毫米

设置格式[holychip-665]: 4 级

设置格式[holychip-665]: 字体颜色: 红色

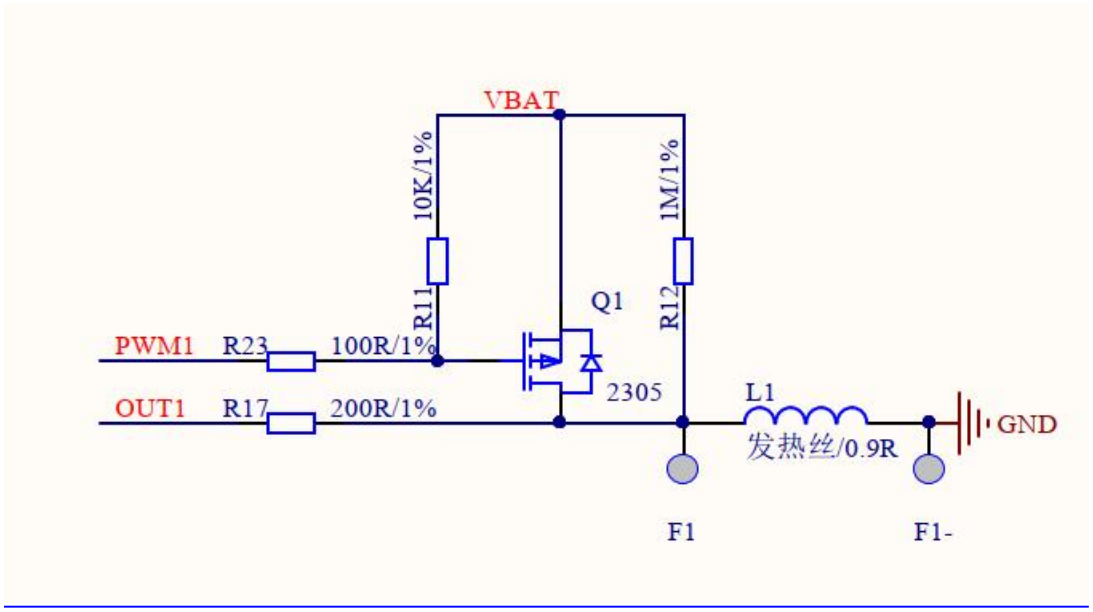
设置格式[holychip-665]: 正文文本

```
    PWM01TL     = 0x64;

    PWM1DH      = 0x00;  //PWM1 周期高 4 位
    PWM1DL      = 0x00;  //PWM1 周期低 8 位

    PWM_MAP  | = 0x0C;  //映射  PWM1  PA6
    //
    PORTA6 = 1;
    ANSELA6 = 1;
    TRISA6 = 1;
    //
//  PWM1OEN = 1;        //禁止时，做 IO
//  PWM01EN = 1;
}
```

硬件：



1. LVD

LVD 做法： CMP1 可检测 VDD 电压，负端选择 1.3V，正端选择 VDD 经过分压电阻后的电压进行比较。

做法说明：触发吸烟动作后，根据当前的 1/4VDD 通道采集的电压进行划分设置检测等级，检测电压表格可以根据给规格书来进行设置。

设置格式[holychip-665]: 制表位: 1.49 字符, 左对齐, 编号 + 级别: 1 + 编号样式: 1, 2, 3, ... + 起始编号: 1 + 对齐方式: 左侧 + 对齐位置: 0 毫米 + 缩进位置: 0 毫米

设置格式[holychip-665]: 项目符号和编号

10.4.3 CMP 监测电源电压

根据图 10-1 的 CMP1 功能框图和 10.4.2 里的公式可知，当 CMP1 的负端选择 1.3V，正端选择内部电阻分压输出 VR1 时，可以通过 CMP1 来检测电源电压，当电源电压低于设定值时 CMP1 输出 0,电源电压高于设定值时 CMP1 输出 1，通过配置 RBIAS1_H、RBIAS1_L、LVDS1<3:0>的值可设定不同的电压监测点，具体如下表：

RBIAS1_H	RBIAS1_L	LVDS1[3:0]	检测值(V)	RBIAS1_H	RBIAS1_L	LVDS1[3:0]	检测值(V)
1	0	0010	4.73	0	1	1010	2.84
0	0	0000	4.62	0	0	0110	2.77
0	1	0110	4.46	1	0	1010	2.74
1	0	0011	4.33	1	0	1011	2.60
0	0	0001	4.16	1	0	1100	2.48
1	0	0100	4.00	0	0	1000	2.45
0	1	0111	3.90	0	1	1100	2.40
0	0	0010	3.78	1	0	1101	2.36
1	0	0101	3.71	0	0	1001	2.31
1	0	0110	3.47	1	0	1110	2.26
1	0	0111	3.25	0	1	1101	2.23
0	0	0100	3.20	0	0	1010	2.19
0	1	1001	3.12	1	0	1111	2.17
1	0	1000	3.06	0	0	1011	2.08
0	0	0101	2.97	0	0	1100	1.98
1	0	1001	2.89				

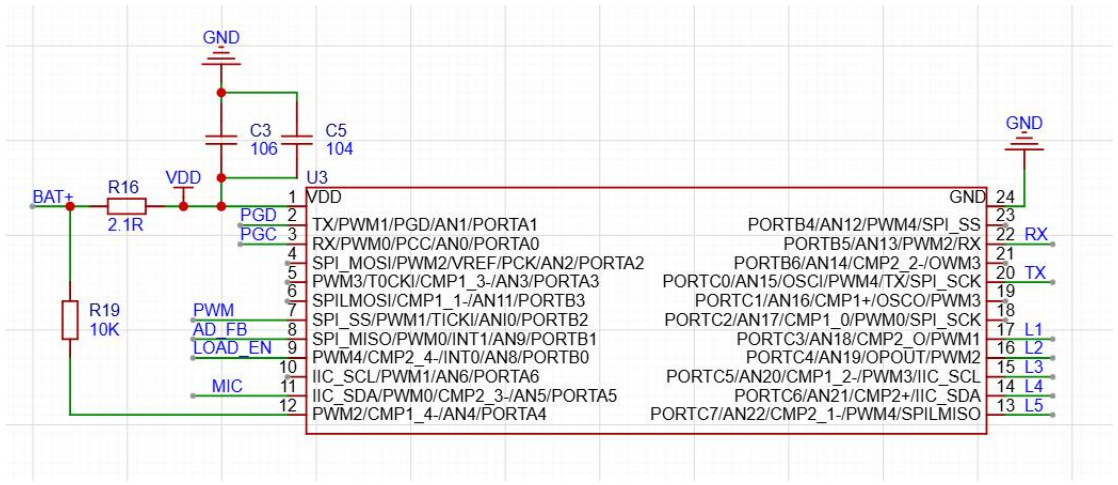
```
//短路使用比较器处理
void CMP_init(void)
{
    ///bit4-5   && bit0-3   需要参考数据手册表格选择不同电压
    CMP1CON1 = 0;
    CMP1CON1 |= LVD_BAT_2_48|(1<<7);//下降沿触发
    CMP1DBC   = 0X85;          // 滤波使能为bit7    使能 CMP1DBC 滤波时   bit0-bit3 需要
填非 0 的值
    //CMP+ 选择 VDD 经过电阻分压的电压
    //CMP- 选择 内部 1.3V
    CMP1CON0 = 0x28;//0x29;          //0010 1000
    CMP1EN = 0;  //使能 CMP1
    //Delay_Us(20);
    CMP1IE = 0;   //中断使能
    CMP1IF = 0;
}
void CMP_Gear(unsigned char cmp_g)
{
    CMP1IE = 0;   //中断使能
    CMP1IF = 0;
    CMP1CON1 = 0;
    if(0 == cmp_g)
    {
      CMP1EN = 0; //关闭 CMP1
    }
    else
    {
        if(1 == cmp_g)
```

```
        {
        CMP1CON1 &= 0xC0;
        CMP1CON1 |= LVD_BAT_2_48|(1<<7);//下降沿触发
        }
        else if(2 == cmp_g)
        {
            CMP1CON1 &= 0xC0;
            CMP1CON1 |= LVD_BAT_2_84|(1<<7);//下降沿触发
        }
        else
        {
            CMP1CON1 &= 0xC0;
            CMP1CON1 |= LVD_BAT_3_71|(1<<7);//下降沿触发      默认
        }
        CMP1EN = 1;    //使能 CMP1
        Delay_Us(20);
        CMP1IE = 1;     //中断使能
        CMP1IF = 0;
    }
}
```

2. CMP

CMP 方式需要占用一个 GPIO 资源。

硬件如图：



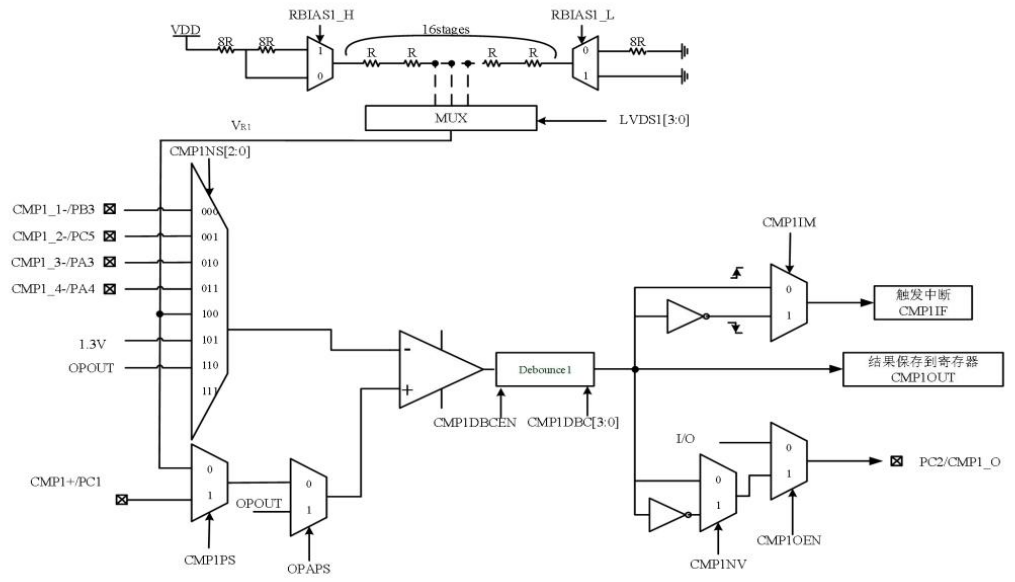
CMP1_4- 负端通过电阻接入 BAT+，正端使用 VDD 经过分压电阻后的电压。

RBIASm_H = 0, RBIASm_L = 0。

1/4VDD + (N+1)/32 VDD

n=15 ----那就是所有都是 0.75VDD

设置格式[holychip-665]: 制表位: 不在 1.49 字符, 无项目符号或编号



软件配置：

```
void Cmp1_Int(void)
{
    ANSELA4 = 0;
    TRISA4 = 0;          //PA4

    CMP1CON1 = 0;
    ///bit4-5   && bit0-3
    CMP1CON1 |=0x0F;
    CMP1DBC   = 0X81;     // 滤波使能为bit7    使能CMP1DBC滤波时   bit0-bit3需要填非0的值

    //CMP+ 选择 VDD经过电阻分压的电压
    //CMP- 选择 PA4
    CMP1CON0 = 0x18;
    delay_xus(20);
    CMP1EN = 1;
    CMP1IE = 1;   //中断使能
    CMP1IF = 0;
}
```

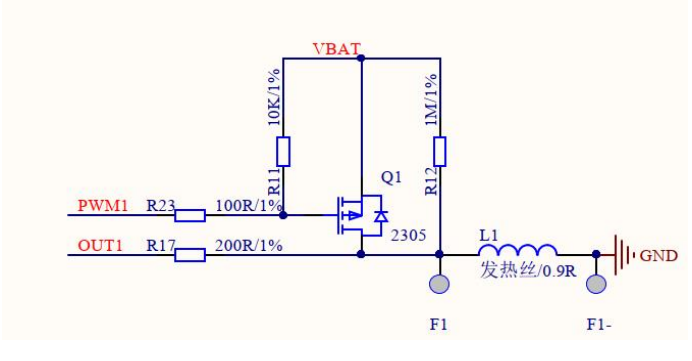
中断：

```
//中断服务函数
void interrupt All_Isr(void)
{
    if(CMP1IE && CMP1IF)
    {
        CMP1IF = 0;
```

```
        CMP1IE = 0;
        MOS_EN= 1; //关闭输出
        BitOuten = 0;  //清除标志
    }
}
```

1.6.2.3. 负载检测

检测是否存在负载，插拔检测功能，部分应用需求中，无负载不支持抽烟
参考上述恒功率原理图中的上拉电阻，1M 电阻上拉到 VBAT



原理：无负载时，VOUT_AD 为高电平　负载接上时，VOUT_AD 为低电平　注意滤波

1.6.2.4. 低阻保护

参考恒功率原理图

```
CTRL_MOS = 0;
Delay_Us(20);
DEVICE_RES =  Get_Adc(8);       //假设分压电阻阻值为3欧,被测阻值为R,  R/(3 + R) = D/4096;
CTRL_MOS = 1;
if(DEVICE_RES < 481)            //阻值0.4欧
{
    //可添加相关显示
}
else if(DEVICE_RES < 683)       //阻值0.6欧
{
    //可添加相关显示
}
```

利用电阻分压原理，软件上利用 VDD 作参考电压，采集 VOUT_AD 的电压，用采集值与 VDD 就是 4096 之间的差值进行判断阻值大小，注意需要开启 PMOS 的情况采集 实际应用中需要测试不同阻值的差值数据进行判断

1.6.2.5. 低压保护

低压保护主要为电池电压

吸烟前低压检测：吸烟前采集 1/4VDD 进行判断，需要关屏下进行采集。
吸烟中低压检测：在 PMOS 开启的时候采集 1/4VDD 进行判断，需要关屏下进行采集。

1.6.2.6. 超时保护

超时保护为了保护 PMOS，不能长时间工作，一般为 8-10s
软件：从开始吸烟开始计时

1.6.2.7. 掉电参数保存

设置格式[holychip-665]: 标题 2, 缩进: 左侧: 0 毫米, 悬挂缩进: 15 毫米, 编号 + 级别: 4 + 编号样式: 1, 2, 3, ... + 起始编号: 1 + 对齐方式: 左侧 + 对齐位置: 0 毫米 + 缩进位置: 15 毫米, 3 级, 从左向右

非完全掉电：
51 系列，若方案上有保存配置参数的需求（如吸烟油电量、输出档位等），可利用 SRAM（xdata） 的上电随机性和非完全掉电（>1.0V）数据不丢失的特性，即使用 SRAM（xdata）作参数保存。
实现方法如下：
（1）通过“xdata”和“_at_”关键字定义保存数据的变量，并指定保存地址。

```
/* USER DATA saved in the 32 bytes of XDATA（0xE0~0xFF） */
    xdata USER_DATA_T g_userdata_t _at_ 0xE0;
```

（2）在系统启动文件对保存数据的 SRAM（xdata）地址不清 0。

```
; <o> XDATALEN: XDATA memory size <0x0-0xFFFF>
;       <i> The length of XDATA memory in bytes.
XDATALEN            EQU       0E0H
;
```

这里就是只清零 0x00--0xDF,0XE0--0XFF 不清零，保存掉电数据。

MTP 系列：RAM 电压 0.7V 还能保持

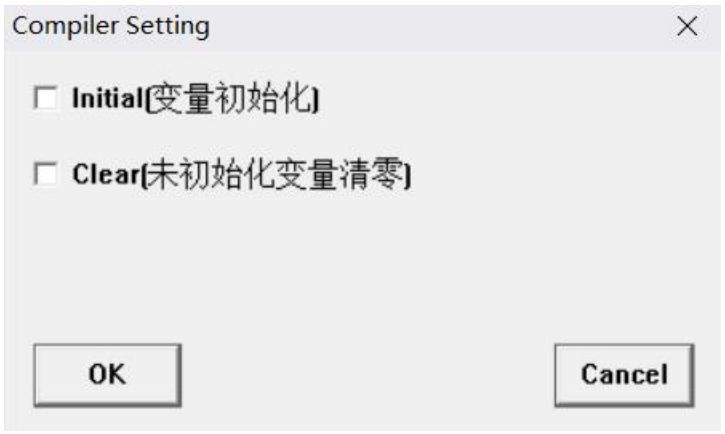
方式 1：persistent 关键字

```
2
3persistent unsigned int ice_cnt;
4
```

方式 2: 使用 at 指定 RAM 地址

```
5volatile unsigned char          ice_cnt              _at_ 0x1e3;
6volatile unsigned int           ice_time             _at_ 0x1e4;
```

方式 3：使用编译器中配置，不勾选下图配置--(HC18M584XA/HC18M003A/HC18M9229 系列)。



1.7. 电量处理部分

1.7.1. 1/4VDD 通道采集

实现电池电压的采样，电池电量等级分配，分配为 100 个等级（0~99）。

```
//这里修改位 ADC 数据
const u16 TABLE_BATVOLT_LEVEL[BAT_VOLT_LEVEL_MAX] =
{
      1690,   // 0%        3.3V
      1862,   // 10%           3.63
      1894,   // 20%           3.69
      1910,   // 30%           3.73
      1919,   // 40%           3.74
      1935,   // 50%           3.77
      1957,   // 60%           3.82
      2025,   // 70%           3.95
      2042,   // 80%           3.99
      2068,   // 90%           4.05
      2120    // 100%          4.15
};
 u8 battery_get_init_level()
{
      u8 Temp_Level;
    //获取电池电压
      battery_get_volt();
  //
  Temp_Level = BAT_VOLT_LEVEL_MAX - 1;
    if(g_bat_s.idle_volt <= TABLE_BATVOLT_LEVEL[0])       //电压小于低压门限
      {
            return 0;
      }
```

设置格式[holychip-665]: 缩进: 首行缩进: 2 字符

```
    else if(g_bat_s.idle_volt >= TABLE_BATVOLT_LEVEL[Temp_Level])      //最大电压门限
    {
        return BAT_LEVEL_MAX;
    }
    else
    {
        for(i = 0; i < BAT_VOLT_LEVEL_MAX-1; i++)
        {
            i_temp =   i + 1;
            if(g_bat_s.idle_volt <= TABLE_BATVOLT_LEVEL[i_temp]) //判断当前电压在哪个区
间
            {
                //计算每 1%对应多少 mV 电压
                step = (TABLE_BATVOLT_LEVEL[i_temp] - TABLE_BATVOLT_LEVEL[i]) / 10;
                //计算理论电压对应的百分比电量
                return (i * 10 + (g_bat_s.idle_volt - TABLE_BATVOLT_LEVEL[i]) / step);
            }
        }
    }
    return BAT_LEVEL_MAX;
}
```

删除[holychip-665]: BAT_VOLT_LEVEL_MAX - 1

删除[杨磊]:

版本修正记录

版本	日期	描述
V1.0	2025-10-27	初版
V1.1	2025-10-30	修改MOS输出电路、恒功率电路负载检测解释
V1.2	2026-01-28	增加短路保护处理