基于单片机辅助的音频功率放大器

# 摘要：

本文以常见的甲乙类OCL互补推挽音频功率放大器作为实验骨架，增加PIC16F818单片机实现各种辅助功能。实现一个低成本的数字增益控制和智能保护放大器。也可以通过改变骨架放大器的种类，用在如大功率或者高保真等更有价值的放大器上。重点介绍了数字音量控制，智能衰减过高输入，输出中点电压自动调整，根据工作点状态开机延时，串行数码显示等。并且已经在万能板上成功将电路实现。实测电路能完成本文所规划的功能。

**关键字：单片机 音频放大器 PIC16F818 数字音量控制**

# 引言：

多媒体时代，晶体管音频放大器的功率越来越大，种类也越来越多。由于元器件的制造工艺不断提高，噪音和不稳定性也日益改善。于是人们不再停留在以前音响界盛誉的“简洁至上（Simple is the Best）”的阶段。纷纷使用大规模高复杂电路，以求实现良好的可控制性以补偿元器件本身缺陷带来的影响。特别是D类功放和数字功放的快速发展，更是对音响技术带来了一场科技革命。但是复杂电路和放大器种类多样性带来了很多不可避免的问题，如放大器的稳定性，各级间相互匹配，出现问题时是否会影响其他设备，开关机时器件能否协调等。必须采取措施去控制各种不利情况的发生。并且新式放大器加入了更多人性化的东西，如实时显示状态信息，支持遥控或者触摸控制等。

# 实验骨架：

一个标准的全对称甲乙类OCL互补推挽音频功率放大器，使用两对2SC1815,2SA1015作为对称差分输入放大级；名管2SB649,2SD669作为推动级；TIP41C,TIP42C作为输出级；使用大环路电压负反馈。相比于标准电路，此电路做了一些改动以适应单片机控制。电路在Multisim10.1中仿真表明：电源电压±20V，输入信号1kHz 100mV时，电路总谐波失真THD<0.2%。

# 骨架中电路可能存在的问题：

上述音频放大器能正常工作，可是仍然存在一些问题：

## 由于避免低频损失，输出没有使用耦合电容，而是直接接到了喇叭负载上。在开机非线性元件未进入稳定状态或输入信号存在直流分量时，导致输出电压直流分量不为0。这种不稳定会导致喇叭通过直流大电流瞬时烧毁。

## ….

## 电路处于稳定增益的状态，即UO=KUI。但是由于电源电压的限制，这种理想的状态是不可能实现的，此电路中，输出电压峰值到达接近2Vcc时，将会出现削峰失真。

## ….

## 负载改变时，中点电压可能会改变，导致对称管中一方电流剧增，而且由于半导体负温度系数的特性，导致电路失衡后一般都不能自主回复到正常状态。而是向破坏性状态发展，直至电流剧增的晶体管烧毁为止。

## ….

## 音量调整由电位器完成。由于电位器存在机械磨损，特别是廉价音响普遍使用的碳膜音量电位器。碳膜长期与滑片摩擦导致磨损，使旋转电位器时电阻值出现跳动，经过放大器放大后导致巨大噪音。并且电位器不能直观显示当前音量值。

## 由于全模拟信号传输，不适合远距离控制，特别是遥控改变音量等。

## 对此，我们使用一块微控制器芯片和M62429电子电位器去解决上述问题。并且利用单片机独特的优势设计了一些交互功能，方便用户控制电路工作。

# 此电路的解决方案：

## 不断使用单片机A/D采样中点电压，当中点电压偏移值超过预设范围时，通过M62429电子电位器校正中点电压。不断重复校正，直到中点电压为零。

## 采样音量峰值，如果某次音量峰值超过一定上限n dB。则将此时音量值衰减n dB，并且保持t秒。t由编程决定。典型值为0.5s。

## 对输出点的电压与负载电流进行监控。任意一个值超过上限，即静音输入并且断开负载t秒，t由编程决定。t秒后先检测输出点的电压是否为0，如果为0则取消静音并且重新连接负载。

## 使用M62429电子电位器调整音量。由于电位器主体非机械部件，无机械磨损。不会产生巨大噪音，使用寿命比机械式电位器长很多。并且能通过单片机控制显示当前电位器衰减量大小。

## 大部分调整信号已经数字化，传输损失少，方便如红外遥控等远距离控制。

# 硬件电路设计实现：

## 音频放大器引出的电路检测和控制工作点：

### 电源控制点：控制电源的开关操作。

### 音量控制点：开机未准备好时静音；开机就绪后自动调整为上次存储的音量；数字控制调整音量；检测到关机动作时立即静音。

### 交流电压检测点：直接检测来自交流电源整流输出的电压，此电压在关闭交流电源时立即为0，但此时由于大容量滤波电容仍然存有电荷，功放仍然处于工作状态直至大容量滤波电容放电完毕。此时应该断开所有输出负载以免欠压导致中点漂移。

### 负载电流检测点：检测负载电流情况，判断负载是否处于过载状态。如果持续过载，则应断开负载。

### 输出中点电位检测点：开机时先检测中点电位是否为0，若不为0则需要校正才能接入负载。工作时定时检测0电位并校正。

### 喇叭继电器控制点：控制负载的接通和断开。

## M64629连接音频放大器的接点：

### 通道1输入端：至参考电压点。接参考0电位，以便校正使用。

### 通道1输出端：至中点调整端。M64629通过此端口控制中点电压。

### 通道2输入端：至音频端子输入。接来自外部的音频输入到M64629中。

### 通道2输出端：至功放音频输入端。外部的音频经过M64629进行音量处理后由此端口输出到音频放大器。

### PIC16F818的引脚分配：

### A/D转换模拟信号输入通道0，输入端：键盘线（重置Timer0继续使用）。

### A/D转换模拟信号输入通道1，输入端：音频放大器输出电流采样 （电容平滑输入）

### A/D转换模拟信号输入通道2，输入端：零点电压采样 （电容平滑输入）

### 基本输入输出端口0，输出端：音量调整/中点调整数据线（串行信号输出到M62429）

### 基本输入输出端口1，输出端：音量调整/中点调整时钟线（输出方波到M62429作为时钟）

### 基本输入输出端口2，输出端：开机信号（电压线，Timer0第一次使用）

### 时钟输入，输入端: PIC16F818 Timer低速时钟输入。

### 基本输入输出端口3，输出端：显示屏数据线。

### 基本输入输出端口4，输出端：显示屏时钟线。

### 基本输入输出端口5，输出端：显示屏片选线。

# 元器件选用：

## PIC16F818简介：

基于上述需求与大量筛选，本例中使用PIC16F818单片机作为核心的控制元件。PIC16F818是一片带A/D转换及微功耗纳瓦技术的单片机芯片，使用取指令和执行指令并行的哈佛总线结构以及只有35条单字指令的精简指令集。

硬件资源：具有1.75k字节程序存储单元(Program Memory)；128字节的数据存储器(RAM)；128字节非易失性数据存储器(EEPROM)；16个基本输入输出引脚；2个8位、1个16位定时/计数器；1个增强型捕捉/比较/PWM模块；1个10位5通道A/D转换器；并且支持SPI/I2C串行通讯技术以及欠压复位、看门狗、休眠模式等其他资源。通过大量的引脚复用技术将所有功能集合于一个18脚的芯片中。

在本电路中，单片机工作在20MHz时钟频率上。指令周期0.2us(5MHz)。典型工作电压5V，不带负载电流2.5~4.0mA(20MHz,5V)，休眠状态最小电流200nA(2V)。

## 电子音量控制控制模块：

本电路使用M64629进行电子音量调整，其工作状态由单片机控制。M64629是串行控制双通道电子音量控制芯片。可控音量范围【0~-83dB(步长1dB)，-∞】。控制数据流格式如下图：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 |
| 0/1 | 0/1 | 音量值控制位 | | | | | | | 1 | 1 |

其中：

D0=0时使用第一通道，D0=1时使用第二通道

D1=0时一次调整两个通道音量，D1=1时一次调整所选通道音量

音量值控制位由实际作用的高至低位排列为：D6D5D4D3D2D8D7=0000000~1111111时，音量衰减相对应[-∞,-83dB~0dB]。

## 当前音量数码显示模块：

本文音量值需要显示在一个两位数码管中。一般的解决方案有三种：

### 第一种是全静态显示，单片机给每个数码管引出8只脚，根据段码设置各引脚高低即可。编程最简单，但这种方法驱动4个数码管单片机引出8个段驱动x4=32只脚，非常浪费单片机资源，极少使用；

### 第二种是数码管片选动态显示，将所有数码管的a-g段连起来接到单片机的8个引脚，数码管公共极接到二-十进制译码器的输出端。通过公共端控制每时刻只开启一个数码管，并以超过人眼可分辨的频率轮流开启不同位的数码管。接数码管段的8个引脚输出相应时刻的段码。由于视觉暂留特性，人眼将看到四个数码管同时显示不同数字。这是最常见的方法。编程稍复杂。驱动4个数码管单片机需要引出8个段驱动+2个片选=10只脚；

### 第三种是数码管段动态显示，通过74164串-并行转换芯片将段驱动的八只脚串行化。使用8个周期将段码输出到74164后锁定，同时通过片选信号将数字显示到相应位上。这种方法最为复杂，器件也要求最高速度。但是引脚占用最少。驱动4个数码管单片机只需要引出1个串行段码输出+1个时钟输出+2个片选=4只脚。如果将片选信号也串行化(使用移位寄存器)而且时钟足够快的话，理论上几只引脚就能驱动十几位数码管。

本文由于PIC16F818属于高速少引脚器件。只能使用第三种方法，74164时钟和串行输入端分别接单片机的引脚RB6,RB7，数码管公共端接RB4,RB5静态片选。具体电路可见总原理图。

## 按钮检测模块：

本电路由于单片机引脚紧张，按钮检测通过接通不同的电阻组合产生相应的电压信号。输入到单片机中进行A/D转换得到不同按钮值。这种方法可以实现一条线上复用多个按钮，不过考虑到按钮用久了可能会产生电阻变化，所以本电路使用了近似值检测，即允许按钮电阻值在一定范围内变化。在实际的软件实现上非常简单，只需要将A/D转换的结果部分低位置0即可。即如某个键期望A/D值是10110010B，检测时我们先将低4位置0，那么只要结果为10110000B~10111111B之间都会得到10110000B。当结果是这个值就判定这个按键按下了。这样就允许按钮有一定程度的电阻误差了。

## 音频放大器连接模块：

由于本文的主题是音频放大器的辅助电路，所以单片机部分要与音频放大器主体建立桥梁，使单片机发出的指令能作用音频放大器上。由于音频放大器的工作电压一般比单片机的要高很多。音频放大器一般工作在双电源±15~50V，单片机只有单电源5V。所以必须区分两者之间的电源。并且由音频放大器原理可知，输入输出信号的参考点都是双电源的中点地。所以单片机与音频放大器必须共地，控制才能做到最简便。输出方面：单片机的大电流开关操作要通过晶体管扩流或者控制继电器间接作用到放大器上；而数字量与模拟量的转换用M64629完成。输入方面：输出中点电压、负载电流等检测均使用A/D转换或者使用部分外围电路先进行信号类型和电压转换再输入单片机。总而言之，要将所有检测控制信号通过接口电路转换成与单片机可识别的0~5v电压，可承受的25mA电流内。

# 硬件电路原理图：

综合上述想法，得到最终的电路原理总图如下。

---

# PIC16F818软件设计：

## 开机延时3秒，等待系统准备完毕才接上喇叭。使用Timer0实现较长时间延时，并且控制开机等待指示灯闪动4次。计时完成之后接通喇叭继电器。

具体代码：

;----------------------------------------------------------

;函数名称：Delay\_3s

;功能描述：延时3秒启动系统，期间呼吸灯闪4次，计时完成之后接通喇叭继电器。

;----------------------------------------------------------

Delay\_3s

BSF OPTION\_REG,0 ;分频比256

BSF OPTION\_REG,1

BSF OPTION\_REG,2

BCF INTCON,T0IF

MOVLW 0xE5 ;外循环76次,用时1s

CLRF TMR0 ;重置TMR0

MOVWF Delay\_3s\_Cnt

Delay\_3s\_1 BTFSS INTCON,T0IF

GOTO Delay\_3s\_1

;呼吸灯开始

BTFSC Delay\_3s\_Cnt,6 ;3秒内,呼吸灯闪4次

BCF PORTB,1

BTFSS Delay\_3s\_Cnt,6

BSF PORTB,1

;呼吸灯结束

DECFSZ Delay\_3s\_Cnt,1

GOTO Delay\_3s\_1

BSF PORTB,2 ;开启喇叭继电器

END

1. 键盘检测部分(按键中断时响应执行)：

* 音量升/降按钮：按钮100ms～3s属于短按，音量变化1dB，触发一次；按钮3～5s属于中按，音量变化3dB/s连发；按钮>5s属于长按，音量变化5dB/s连发
* 静音按钮：按下静音按钮立即静音，再按一次20dB/s升回原音量。单片机上电时不启用静音。
* 电源按钮：单片机上电时关机，按一次电源按钮启动，再按关机。开关操作都以20dB/s速度渐入渐出改变音量。

1. 数码管驱动部分(每10次程序循环执行一次)：

* 以n(n在可接受范围内尽量大)次循环。读取一次当前音量电位器位置值，并且显示到屏幕数码管。刷新频率超过视觉暂留时间即可(>25Hz)。
* 读得电位器位置值后，先转换为BCD码，再通过查表方法得到数码管段码，以串行信号形式输出到74164中。时钟信号控制74164移位及输出。

;----------------------------------------------------------

;函数名称：LED\_Display

;输入参数：音量Volume\_Data;Main\_Status,5:音量升降标志;Main\_Status,6:音量直调标志

;输出参数：

;功能描述：音量调整

;----------------------------------------------------------

LED\_Display

;输入数字分离BCD码结果到w

MOVLW 0x1E

MOVWF LED\_Data ;输入LED\_Data到W

MOVF LED\_Data,0

MOVWF LED\_HalfDat

SWAPF LED\_Data,0

ANDLW 0X0F

CALL BIN\_HIGHHALF\_BCD\_TABLE

MOVWF LED\_Data

MOVF LED\_HalfDat,0

ANDLW 0X0F

CALL BIN\_LOWHALF\_BCD\_TABLE

ADDWF LED\_Data,0

MOVWF LED\_HalfDat

ANDLW 0XF0

MOVWF LED\_Data

MOVF LED\_HalfDat,0

ANDLW 0X0F

CALL BIN\_LOWHALF\_BCD\_TABLE

ADDWF LED\_Data,1 ;转换结果到LED\_Data

MOVLW 0xF0 ;将0F送LED\_DataH，等AND做筛选

MOVWF LED\_DataH

MOVLW 0x0F ;将0F送LED\_DataL，等AND做筛选

MOVWF LED\_DataL

MOVF LED\_Data,0

ANDWF LED\_DataH,1

SWAPF LED\_DataH,1

ANDWF LED\_DataL,1 ;将AND结果放w，准备查表转换

LOOP\_LED BTFSS LED\_CS,0 ;根据LED\_CS将LED\_Data转到w

MOVF LED\_DataH,0

BTFSC LED\_CS,0

MOVF LED\_DataL,0

CALL TABLE1 ;按照w查表

MOVWF LED\_HalfDat ;输出值

BCF PORTA,2 ;片选脚PORTA,2输出0,选个位数码管

MOVLW 0x08 ;8次循环

MOVWF LED\_OutCnt ;次数存LED\_OutCnt

LOOP\_BYTE

BTFSS LED\_HalfDat,7 ;测试LED\_Data.7是否1,是则跳下句

BCF PORTA,1 ;PORTA,1置0

BTFSC LED\_HalfDat,7 ;测试LED\_Data.7是否0,是则跳下句

BSF PORTA,1 ;PORTA,1置1

RLF LED\_HalfDat,1 ;左移一次LED\_Data

BSF PORTA,0 ;给一个时钟脉冲PORTA,0

BCF PORTA,0

DECFSZ LED\_OutCnt,1 ;减一次LED\_OutCnt,到0跳下句

GOTO LOOP\_BYTE ;继续LOOP\_BYTE

BCF TRISA,2 ;输出数字已准备好,设置TRISA,2输出

BTFSC LED\_CS,0

BSF PORTA,2 ;片选脚PORTA,2输出1,选十位数码管

BTFSS LED\_CS,0

BCF PORTA,2 ;片选脚PORTA,2输出0,选个位数码管

CALL delay20ms ;延时20ms

BSF TRISA,2 ;时间到，收回输出，设置片选高阻

COMF LED\_CS,1 ;如果LED\_CS取反

GOTO LOOP\_LED

SLEEP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*段码表\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE1 ADDWF PCL,1

RETLW 0X40;

RETLW 0X79;

RETLW 0X24;

RETLW 0X30;

RETLW 0X19;

RETLW 0X12;

RETLW 0X02;

RETLW 0X58;

RETLW 0X00;

RETLW 0X10;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*用于将高半字节的BIN码换成整字节的压缩BCD码\*\*\*\*\*\*\*\*\*\*

BIN\_HIGHHALF\_BCD\_TABLE

ADDWF PCL,1

RETLW B'00000000' ;0

RETLW B'00010110' ;16

RETLW B'00110010' ;32

RETLW B'01001000' ;48

RETLW B'01100100' ;64

RETLW B'10000000' ;80

RETLW B'10010110' ;96

RETLW B'00101000' ;128,失掉了百位

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*用于将低半字节的BIN码换成整字节的压缩BCD码\*\*\*\*\*\*\*\*\*\*

BIN\_LOWHALF\_BCD\_TABLE

ADDWF PCL,1

RETLW B'00000000' ;0

RETLW B'00000001' ;1

RETLW B'00000010' ;2

RETLW B'00000011' ;3

RETLW B'00000100' ;4

RETLW B'00000101' ;5

RETLW B'00000110' ;6

RETLW B'00000111' ;7

RETLW B'00001000' ;8

RETLW B'00001001' ;9

RETLW B'00010000' ;A

RETLW B'00010001' ;B

RETLW B'00010010' ;C

RETLW B'00010011' ;D

RETLW B'00010100' ;E

RETLW B'00010101' ;F

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*延时20ms子程序\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

delay20ms

MOVLW 0x12

MOVWF Delay\_Cnt0

delayLoopA CLRF Delay\_Cnt1

delayLoopB DECFSZ Delay\_Cnt1

GOTO delayLoopB

DECFSZ Delay\_Cnt0

GOTO delayLoopA

RETURN

END

1. M64629音量控制(实时执行)：

* 通过算法将需要音量设定值转换成M64629能识别的方式，以串行信号形式输出到M64629中。并且以时钟信号同步控制。M64629控制信号格式如下图(来自M64629数据手册)：
* 将音量调整，音量渐入，渐出分别作为两个子程序，便于调用。尽可能提高控制速度。

;----------------------------------------------------------

;函数名称：SET\_Volume

;输入参数：音量值Volume\_Data

;功能描述：M64629驱动程序

;----------------------------------------------------------

MOVLW 0x78 ;输入音量值

SET\_Volume

MOVWF Volume\_Data

BSF PORTB,4 ;DATA:D0\_RISE

BSF PORTB,5 ;CLOCK:0\_RISE

BCF PORTB,4 ;DATA:D0\_FALL

BCF PORTB,5 ;CLOCK:0\_FALL

BSF PORTB,5 ;CLOCK:1\_RISE

BCF PORTB,5 ;CLOCK:1\_FALL

BTFSS Volume\_Data,2

BSF PORTB,4 ;DATA:D2\_RISE

BSF PORTB,5 ;CLOCK:2\_RISE

BCF PORTB,4 ;DATA:D2\_FALL

BCF PORTB,5 ;CLOCK:2\_FALL

BTFSS Volume\_Data,3

BSF PORTB,4 ;DATA:D3\_RISE

BSF PORTB,5 ;CLOCK:3\_RISE

BCF PORTB,4 ;DATA:D3\_FALL

BCF PORTB,5 ;CLOCK:3\_FALL

BTFSS Volume\_Data,4

BSF PORTB,4 ;DATA:D4\_RISE

BSF PORTB,5 ;CLOCK:4\_RISE

BCF PORTB,4 ;DATA:D4\_FALL

BCF PORTB,5 ;CLOCK:4\_FALL

BTFSS Volume\_Data,5

BSF PORTB,4 ;DATA:D5\_RISE

BSF PORTB,5 ;CLOCK:5\_RISE

BCF PORTB,4 ;DATA:D5\_FALL

BCF PORTB,5 ;CLOCK:5\_FALL

BTFSS Volume\_Data,6

BSF PORTB,4 ;DATA:D6\_RISE

BSF PORTB,5 ;CLOCK:6\_RISE

BCF PORTB,4 ;DATA:D6\_FALL

BCF PORTB,5 ;CLOCK:6\_FALL

BTFSS Volume\_Data,0

BSF PORTB,4 ;DATA:D7\_RISE

BSF PORTB,5 ;CLOCK:7\_RISE

BCF PORTB,4 ;DATA:D7\_FALL

BCF PORTB,5 ;CLOCK:7\_FALL

BTFSS Volume\_Data,1

BSF PORTB,4 ;DATA:D8\_RISE

BSF PORTB,5 ;CLOCK:8\_RISE

BCF PORTB,4 ;DATA:D8\_FALL

BCF PORTB,5 ;CLOCK:8\_FALL

BSF PORTB,4 ;DATA:D9.10\_RISE

BSF PORTB,5 ;CLOCK:9\_RISE

BCF PORTB,5 ;CLOCK:9\_FALL

BSF PORTB,5 ;CLOCK:10\_RISE

BCF PORTB,5 ;CLOCK:10\_FALL

END