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：

## 课题概述

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。本文正是借用一个实例，探讨功率放大器电路的智能控制方法。并且在研究过程中注重计算机辅助设计技术EDA的应用，借助EDA技术使电路开发更快速准确。

## 系统总体设计方案

系统要求：

### 输出有效功率在8Ω负载上≥5w，失真率≤1%，以满足日常演示要求。

### 有音量自动调整，负载保护，用户交互，状态显示等功能。

### 工作电压：单片机部分使用稳压DC 5v，功率放大器部分使用DC ±15v以上。

### 尽量减少单片机模块与功率放大器模块之间的耦合，实现功率放大器可以随意替换为其他形式的放大器。

### 尽可能采用常见元件，便于日常替换。并且要易于扩展。

概要设计：

鉴于上述系统功能要求以及音频放大器低成本，较精密，需要较多数模转换接口的要求，本系统在硬件设计方面具有如下特点：

### 主控单片机采用的芯片必须带有AD转换器，以便接受音频放大器的信号；由于使用了大量串并行转换。所以需要使用支持较高速度的单片机芯片，以实现高速串行传输；还有必须带有EEPROM，以便断电保存音量值或者中点校正值。选择了PIC16F818单片机。

### 而音量控制以及中点电位调整等由于不能通过单片机输出的数字信号直接控制，只能通过模数转换或者电子音量控制芯片完成。

### 显示部分由于单片机输出引脚较少，不能如日常的方法花费大量引脚于显示方面，只有采用串并行转换，利用单片机的高速度分时显示相应数码管段。

### 键盘输入也需要考虑到引脚紧张这个问题，可以通过CD4017十进制计数器分时复用按键线。也可以简单的通过电阻分压网络产生不同电压点，然后在不同电压点通过按钮输入单片机。使用AD转换识别按下了哪个按钮。分时复用方法比较稳定，能同时复用很多个开关。但是需要多加一片IC，电路比较复杂；使用AD识别方法能充分利用单片机自有资源，电路简单，但是会因为按钮老化电阻不稳定导致电路不稳定，适用于少量按钮组合。由于本例只有几个开关，精度不高。所以使用AD识别方法更方便快捷。

## 软件设计

### 音频功率放大器来自于经典的全对称式OCL电路，电路有性能较好，制作调试方便的优点。制作实物之前先使用EDA软件对电路进行修改与仿真，以便将其修改为适合于单片机控制的形式。电路的搭建使用National Instruments公司的Multisim 10.1（原名EWB）。该软件的优点是仪表功能十分强大，能实时观察到一个模拟电路的输入输出波形，幅频响应，瞬态分析等。非常适合仿真OCL音频放大器类电路。

### 单片机控制部分Microchip公司的PIC16系列中档单片机支持35条单字节汇编指令，开发可以使用官方提供的MPLAB IDE。代码编写可以使用一些第三方的编辑器，在此使用Notepad2作为主要的代码编辑器。MPLAB IDE主要用于代码编译调试。单片机仿真可以用Proteus或者Tina实现。Proteus与MPLAB IDE有很好的融合，能在MPLAB里面仿真调试电路；Tina的特点是便于操作、结果直观。并且对PIC系列单片机支持比Proteus更好。所以本例使用Tina作为主要仿真调试软件。

### 并且在此论文编写过程中使用了SVN版本管理技术管理整个工程的文件。SVN是一个开源的版本控制系统Subversion的简称。Subversion管理着随时间改变的数据。这些数据放置在一个中央资料档案库(repository)中。这个档案库很像一个普通的文件服务器，不过它会记住每一次文件的变动。这样你就可以把档案恢复到旧的版本，或是浏览文件的变动历史。许多人会把版本控制系統想象成某种“时光机器”。而我使用的是Google提供的免费SVN服务。SVN有很多优点：第一个优点是由于SVN服务器能记住我每一次修改，所以一旦文件出现问题，就能使用SVN工具立即回溯到最近一次修改的版本；第二就是无论身在何处，只要有电脑的地方就能将整个工程下载下来继续工作。修改之后提交到服务器即可。回到家里只需要更新最新修改过那部分文件即可，不需要因为忘记了修改了哪些文件而需要整个工程上传下载；第三就是支持多人协作修改，当多人在不同的计算机修改同一个工程时，SVN能以合并修改、解决冲突、产生分支等几种方法协调多用户产生不同版本的问题。

# 实验骨架：

### 本例采用一个标准的全对称甲乙类OCL互补推挽音频功率放大器，电路的第一级采用双互补对称差分电路，每管的静态工作电流约1mA，选用低噪声互补管2SC1815、2SA1015作差分对管，有较低的噪声和较高的动态范围。第二级电压放大采用互补推挽电路，仍然采用2SC1815、2SA1015，工作电流约5mA。两管集电极串接的发光二极管为缓冲级提供约1.6V~2.0V的偏置电压，避免末级产生交越失真。射随器缓冲驱动级由两只互补对管2SB649、2SD669构成，增设射随器缓冲驱动级是现代OCL电路的主要特点之一，它主电压放大级具有较高的负载阻抗，有稳定而较高的增益。同时它又为输出级提供较低的输出内阻，可加快对输出管结电容Cbe的充电速度改善电路的瞬态特性和频率特性。该级的工作电流也取得较大，一般为10~20mA，个别机型甚至高达100mA,与输出级的静态电流差不多，可使输出级得到充分驱动。其发射极电阻采用了悬浮接法（不接中点），可迫使该级处于完全的甲类工作状态，同时又为输出级提供了偏置电压。输出级为传统的互补OCL电路，采用了韩国KEC生产的大功率互补对管TIP41C、TIP42C对管，极限输出功率可达65w。电路使用大环路电压负反馈，电路总增益由反馈网络决定，本例设计增益为33倍。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%。

-------插音频放大器电路图-------

# 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：

## 由于避免耦合电容对超低频会因为容抗作用产生损失，电路输出没有使用耦合电容，而是直接接到了喇叭负载上。由于开机时，容量非常大的电源滤波电容还没充满电。在开机的几秒电源内未达到标准电源电压。此时电路中的如三极管等非线性元件未能进入稳定状态。或者如果某时刻由于输入设备故障等原因导致输入信号存在直流分量，没有电容分隔开直流分量的话，就会导致电路输出电压直流分量不为0。这种不稳定会导致喇叭持续通过直流大电流而过热烧毁。

-------插输出电路图-------

## 在正常工作情况下，电路处于稳定增益的状态，即UO=K\*UI。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。UO最大不可能超越总电源电压2Vcc，并且由于末级输出三极管存在正向压降0.7v左右，理论最大输出电压约为2Vcc-2\*0.7V。此电路中，输出电压峰值到达接近2Vcc时，将会出现削峰失真。削峰失真容易使三极管长期处于满负载状态而发热烧毁，并且会严重影响音质。

-------插削峰失真图-------

## 电路输出中点电压可能会在负载阻抗改变，或者非线性晶体管工作温度变化时产生零点漂移。此时如果直接接入了能通过直流的负载（如喇叭），大量电流会通过负载流向接地点。漂移会导致互补输出管中一方电流剧增。而且由于硅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或者负载烧毁为止。

-------插球在不同凹凸平面上图-------

## 一般功率放大器音量调整由机械电位器完成。由于电位器长期使用会产生机械磨损，特别是廉价音响普遍使用的碳膜音量电位器。很薄的碳膜长期与滑片摩擦会导致磨损甚至磨穿，使旋转电位器时电阻值出现跳动，经过放大器放大k倍之后将会导致巨大噪音，严重影响电路工作。并且电位器不能准确显示当前音量值。

## 由于全模拟信号传输，不适合远距离控制，如遥控开关机，遥控改变音量等。

# 此电路的解决方案：

## 动态中点电压校正：不断使用单片机A/D采样中点电压，并且不断与参考0电位相比，得出当前中点电压值，然后通过计算得出修正值。当中点电压偏移超过预设范围时，由单片机控制M62429电子电位器校正中点电压。不断执行检测与校正，使中点电压保持为零。

## 输入信号动态限幅：使用动态压缩模式(DRC)限幅器限制输出幅度。具体实现为不断采样音量值，如果某次采样音量峰值超过一定上限xdB则将此时音量值衰减ydB，并且保持t秒。如果保持期间输入音量恢复到正常范围，则达到t秒后释放刚施加的衰减，回到原始状态。t由单片机程序决定。典型值为200~500ms。

## 负载过压过载保护：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。

## 电子音量控制：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。

## 数字化人机交互：大部分调整信号已经数字化或者提供了数字化接口，如中点电位、当前音量，限幅值，按键事件等信号均以数字信号存在单片机内，可以通过串行输出等方法可以扩展到其他设备上，传输损失少，方便如红外遥控，状态监测等远距离控制。并且音量值与衰减状态以数码管显示出来，更直观，可以扩展的功能也更多。

# 硬件电路设计实现：

由于单片机与功率放大器需要建立连接，所以要先设计它们的连接点。

## 音频放大器引出的电路检测和控制工作点：

### 电源控制点：连接继电器，控制功率放大器电源的开关操作。实现待机功能。

### 音量控制点：单片机通过M64629控制放大器音量。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；输入过高时衰减音量均由此点控制。

### 交流电压检测点：直接检测来自交流电源的电压，必须取自电源滤波电容前面并且以二极管隔离。做到交流电源一旦断电，此电压检测点立即电压为0，由于上文所述滤波电容充放电时产生的欠压工作状态会令电路失衡。此时功放应立即静音，并且断开所有输出负载中点漂移造成损坏。

### 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，保护喇叭不受损坏。

## M64629连接音频放大器的接点：

### 通道1输入端：至电路接地点。视作参考0电位，校正中点电压参考此电压执行。

### 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中作音量大小处理。

### 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：键盘输入线（按钮需要区分长按和短按，需要使用定时器，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（输入前通过电容平滑输入，滤出其中的直流成分）。

### A/D转换模拟信号输入通道2，输入端：中点电压采样（输入前通过电容平滑输入，滤出其中的直流成分）。

### 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数据接口）。

### 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：至显示模块74164数据线。

### 基本输入输出端口4，输出端：显示模块74164时钟线。

### 基本输入输出端口5，输出端：显示模块的数码管片选线。

# 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---注明引自数据手册----：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)。单片机基本输入输出引脚数，AD转换器数目，程序存储单元，EEPROM，指令执行速度等均合乎要求，不缺少且不过于浪费。是较佳的选择。

## 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)，-∞】。其中通过将对数据输入引脚注入控制串行数据流来控制M64629的工作状态。

## 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。

----总电路图----

# PIC16F818软件设计：

## 工作流程：

由于电路需要长时间工作，所以总体设计思想必须将单片机执行于无限循环中。在一次循环中，必须完成所有的检测调整工作。并且由于按键事件，过载事件等事件是不定时发生的，所以代码执行中途遇到不定事件则转去中断处理程序。处理完毕之后回到原来的无限循环中。遇到代码执行错误导致崩溃时使用单片机看门狗进行复位操作。

下面列出软件设计总流程：

----总流程图----

## 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。

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

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渐变升回原音量。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。

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

;函数名称：Key\_Scan

;输入参数：原有按键时间值Key\_AD\_Count

;输出参数：按键值 Key\_Record，按键时间值Key\_AD\_Count

;功能描述：按键扫描，扫描一次，总共次数累计于Key\_AD\_Count。如果结果不为0，结果放Key\_Record

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

Key\_Scan

BCF Key\_Record,0 ;置转换成功标志0

BCF STATUS,RP0 ;体0

BSF ADCON0,2 ;开启A/D

Wait\_AD

BTFSS PIR1,6 ;等待A/D完成

GOTO Wait\_AD

BTFSC ADRES,7 ;如果A/D结果高4位全为0则舍弃，本次扫描没有检测到键

GOTO Key\_Scan\_Success

BTFSC ADRES,6

GOTO Key\_Scan\_Success

BTFSC ADRES,5

GOTO Key\_Scan\_Success

BTFSC ADRES,4

GOTO Key\_Scan\_Success

GOTO Key\_Scan\_0

Key\_Scan\_Success

MOVF ADRES,W ;A/D值到W

ANDLW B'11111000' ;低位 置0 舍弃，防止按键不稳干扰

XORWF Key\_Record,W ;新旧按键值对比

BTFSC STATUS,Z ;测试异或结果，新旧按键不同则重置计数

GOTO Key\_Scan\_0

INCF Key\_AD\_Count ;如果相同计数+1

NO\_Recount

MOVF ADRES,W ;更新按键记录

ANDLW B'11111000' ;低位 置0 舍弃，防止按键不稳干扰

MOVWF Key\_Record ;本次AD扫描结果放Key\_Record，本次扫描成功检测键值结束

BSF Key\_Record,0 ;置Key\_Record,0位为1，做转换成功标志

GOTO Key\_Scan\_End

Key\_Scan\_0

MOVLW 0 ;重置计数

MOVWF Key\_AD\_Count

BCF Key\_Record,0 ;置转换成功标志0

Key\_Scan\_End

RETURN ;函数返回

END

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(所有图表以及数据来自M64629生产厂家提供的数据手册)：

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。

M64629数据与时钟信号关系图如下：

M64629数据时钟关系图.png

由图中可以看出，当时钟上升沿时，读取数据信号。下降沿锁存信号暂时没用到，暂时忽略。按照时序将D0~D10输出数据信号，时钟信号以使用一次BSF,一次BCF的方法产生一个脉冲穿插在中间。每次改变音量值只需要执行一次音量调整即可。

### 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]。

由于需要按照数据手册给出的控制数据流来控制M64629，并且PIC单片机不支持指针操作。所以只能使用最原始的顺序执行方式。通过BSF和BCF控制数据引脚和时钟引脚的电平高低。

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

;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

# 总结

# 致谢

# 参考文献