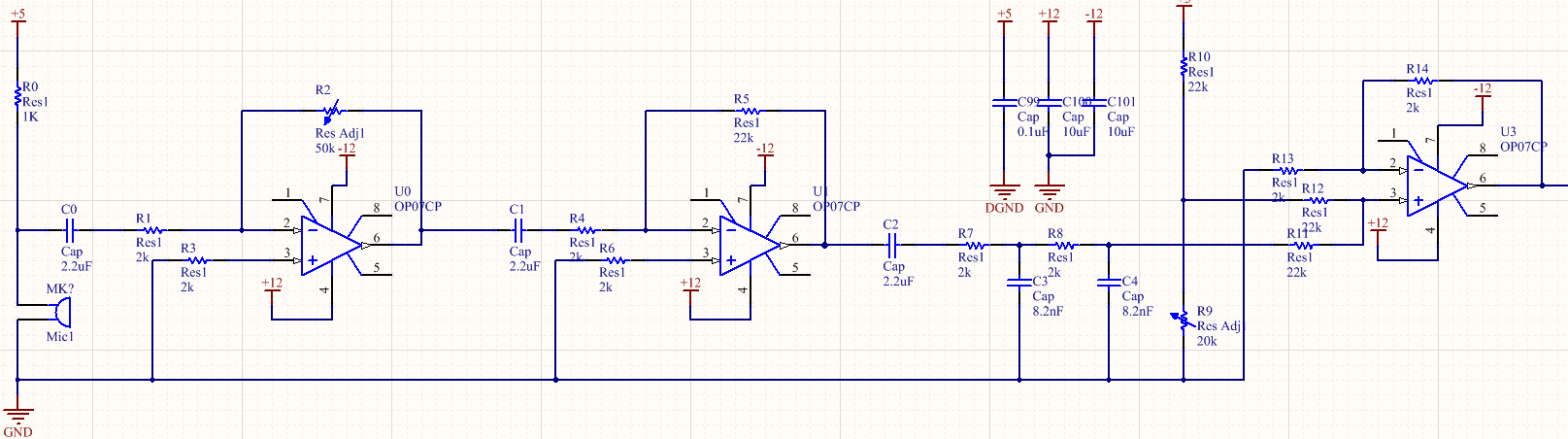
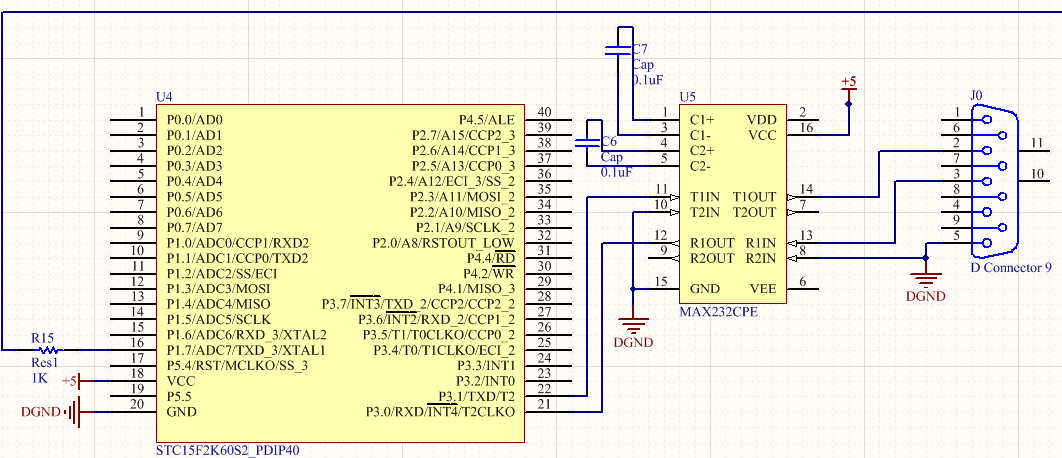
录音电路 实验报告

16307130006 陈幸豪

1. 电路整体原理图



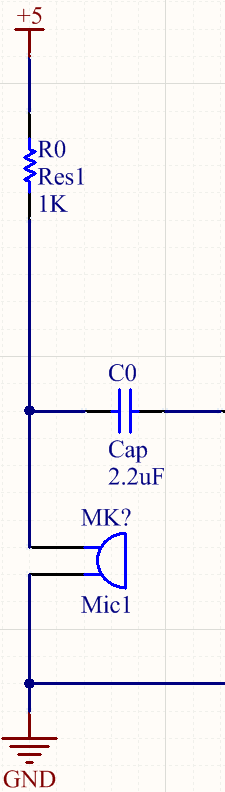


需要注意：**运放OP07CP的电源输入±12V在原理图中画反了**（应为7脚接+12V，4脚接-12V）。实际电路板上按正确方法接入电源即可。+5V电源的退耦电容C99不可只用0.1uF，应使用10uF或更大。实际制作中即使不接退耦电容，效果也很好。各IC芯片最好插在芯片座上，而非直接焊接。

这个原理图仅使用了3种定值电阻，1种运放，2种电容（不使用退耦），给实际制作带来很大方便。可变电阻的封装使用VR5。

* 1. 驻极体话筒（咪头）驱动电路

从网上查到，咪头在直流上可以相当于一个电容。而受声波影响时，咪头两端会产生交流电压，幅度约几十毫伏。咪头需要直流偏置和限流电阻才能工作。于是咪头驱动电路如下：

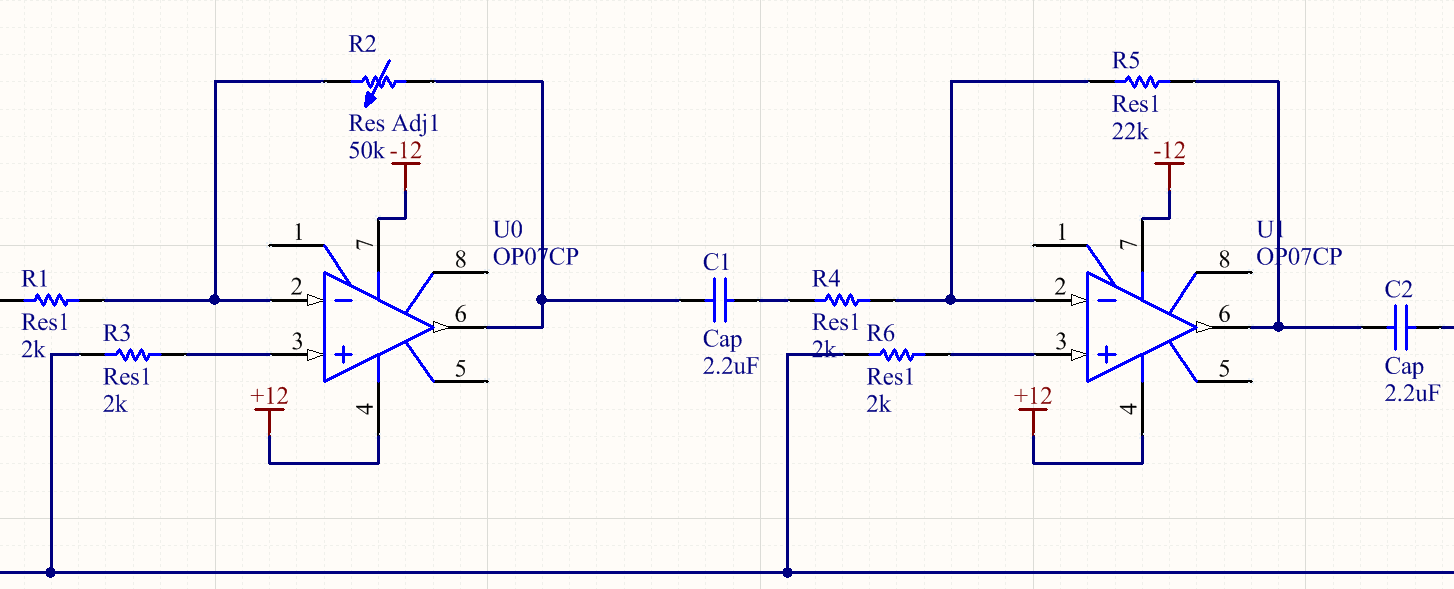


为了保证低频声音不会被电容严重抑制，这里把隔直电容C0选得很大。需要注意C0的左端直流电压比右端高。C0正负极接反时，咪头输出信号无法到达放大器，甚至可能炸毁电容。

实际制作中，从淘宝买了灵敏度仅为-52dB的便宜咪头。灵敏度过高的咪头可能录入各种杂音，因此难以控制输入音频的内容，也难以控制噪音。经过测试发现，在适当的音量下，可以认为咪头的输出电压通常为20mV量级。

* 1. 放大器

使用两级极其朴素的反相放大器，其中第一级增益可调：



以第一级为例来计算每一级的参数：输入阻抗约等于R1，输出阻抗极小，放大倍数约为R2/R1。

* + 1. 运放选择

好的运放型号是确保声音质量的关键因素之一。对于咪头放大电路，运放应具有很低的输入失调电压（远低于咪头输出信号的幅度），较低的噪声，以及足够的压摆率和增益带宽积。

最初我选择运放型号的方法是：去Ti等生产厂家的官网，输入运放的参数要求，直接筛选。然而购买时遇到了这样的尴尬：满足要求的运放有的价格太高，有的只有贴片封装而没有直插封装，有的在国内难以买到。如果产品购买困难，则在官网上筛选产品变得毫无意义。于是改变了运放选择方式：先去淘宝看看能买到什么运放。意外地发现，淘宝上的运放有很多是生产厂家不再推荐使用的旧产品。

最终选择OP07CP的几个重要原因是：

1. 极低的失调电压(85uV)和噪声(10.2nV/√Hz)，以及较多用于音频放大的成功先例。
2. 属于仍未被淘汰的古董产品，因此容易买到且价格便宜(少量购买时也只需0.3元一个)。
3. 可以调零(虽然并不希望真的用到这个功能)。
4. 压摆率适当(0.3V/us,在输出幅值2.5V时可完整保留19.1kHz频率)，增益带宽积足够(最小值为400kHz)

OP07CP的缺点也很明显：

1. 每个DIP-8封装内只有1个而不是2个运放。
2. 反接电源时，极易烧毁运放，且导致两个电源脚之间的直流电阻只有几百欧姆，使电源短路。

要克服第一个缺点，只能减少运放使用数量。

* + 1. 放大器结构选择

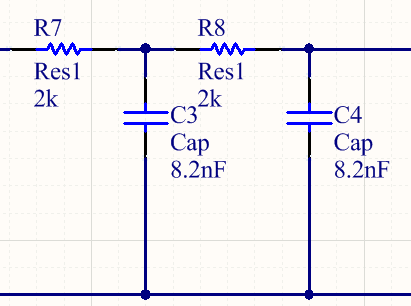
最初设计时考虑过仪表放大器和同相放大器。仪表放大器具有极高的共模抑制比，可有效减少直流输出，然而需要用3个运放。双运放芯片由于零售价格太贵，只有贴片封装等原因，最终没有采用，于是也放弃了仪表放大器的结构。

同向放大器虽然具有极高的输入阻抗，但共模抑制比和输入动态范围不佳。

最终选择反相放大器的原因是：反向放大器输入阻抗不足的缺点相对更容易克服。我们随时可以换用更大的输入电阻，或者用更高的放大倍数来弥补。此外，咪头内的输出阻抗变换电路已经使得输出阻抗足够小。而同向放大器的共模抑制比和输入动态范围的缺点则难以克服。

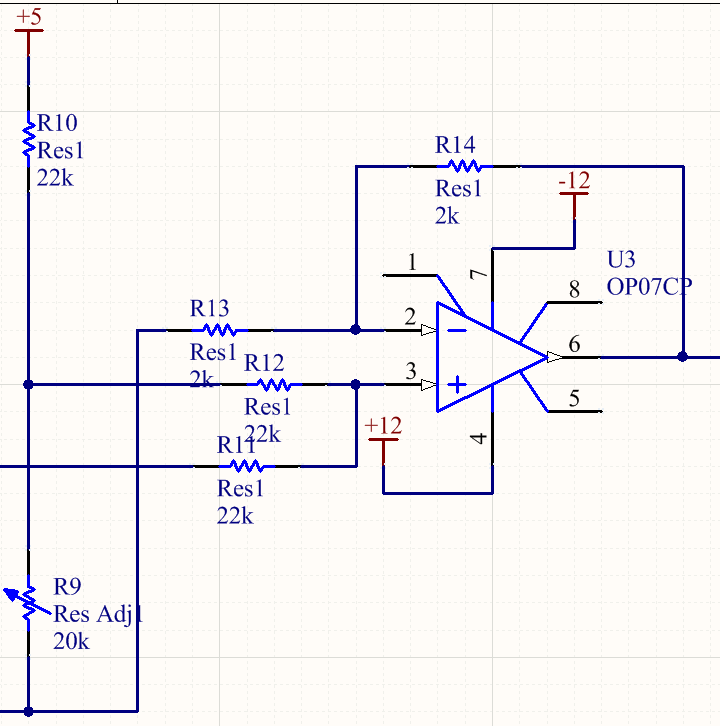
采用一级放大时，不一定能获得足够大的信号幅度，且调节放大幅度时需要可变电阻的改变量很大。因此选择更为保守的两级放大方案。

* 1. 滤波器



这里采用了极其朴素的两阶RC无源滤波来去除高频噪声。实际制作中采用2.7nF电容，目标是滤除10kHz以上的频率。理论上把这个电路移到第二级运放的输入端也可以，但这样布线将更困难，且无法消除第二级运放本身的输出噪声。另一种方法是专门做一级有源滤波，但这样又需要一个运放，占用电路板面积实在太大。

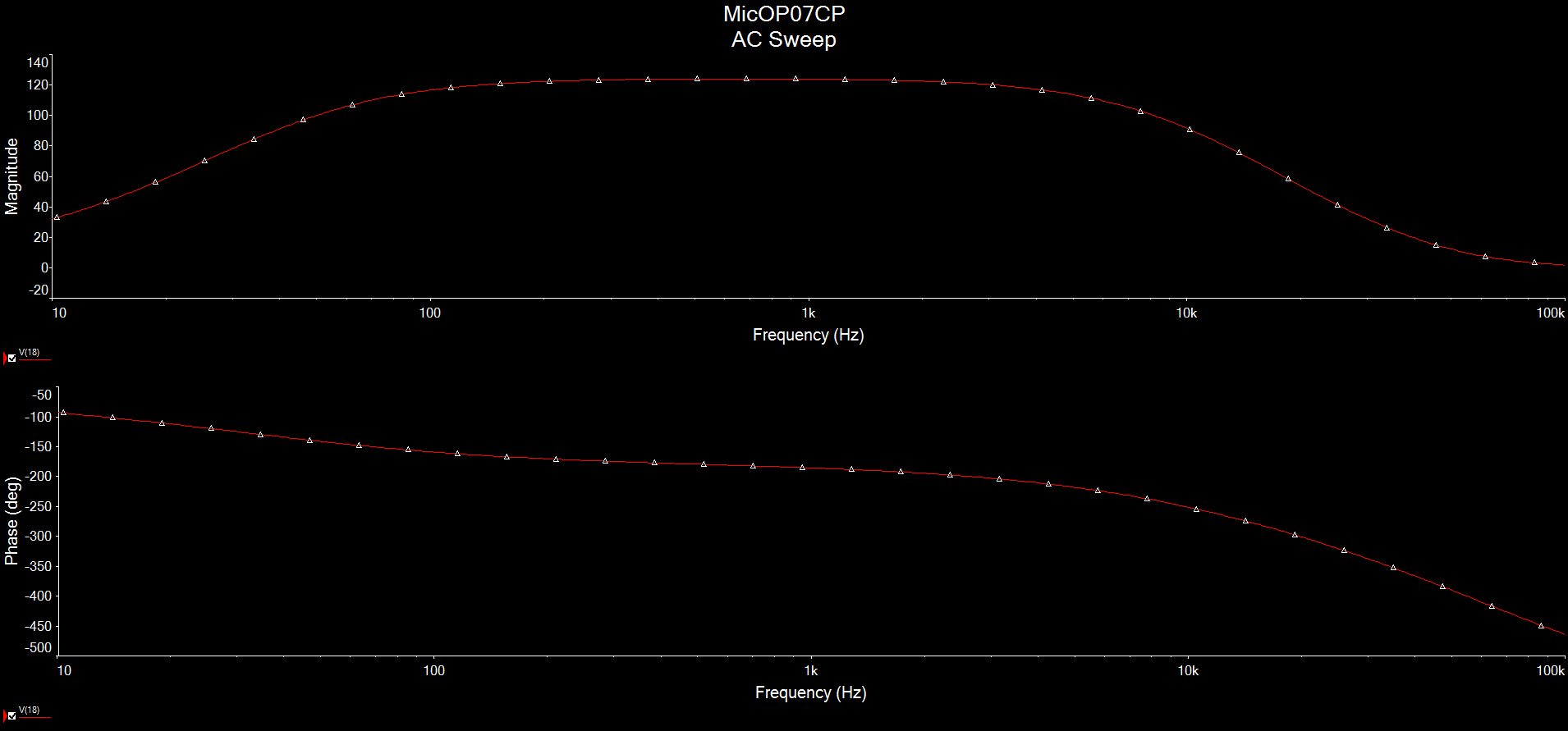
* 1. 加法器



极其朴素的同相加法器。加法器的目的是，将放大器的输出电压抬高2.5V左右，使单片机的ADC能正常工作。前级滤波器在转折频率的输出阻抗显得较高(2k左右)，于是使用22k作为加法器输入电阻，且使用同相加法器。网上大佬们都喜欢抨击同相加法器精度不高（不能认为两个输入电压之和就精确等于输出电压），但这在直流电压抬升的任务中并不会产生问题。我们只要适当调节R9使输出直流电压为2.5V即可。

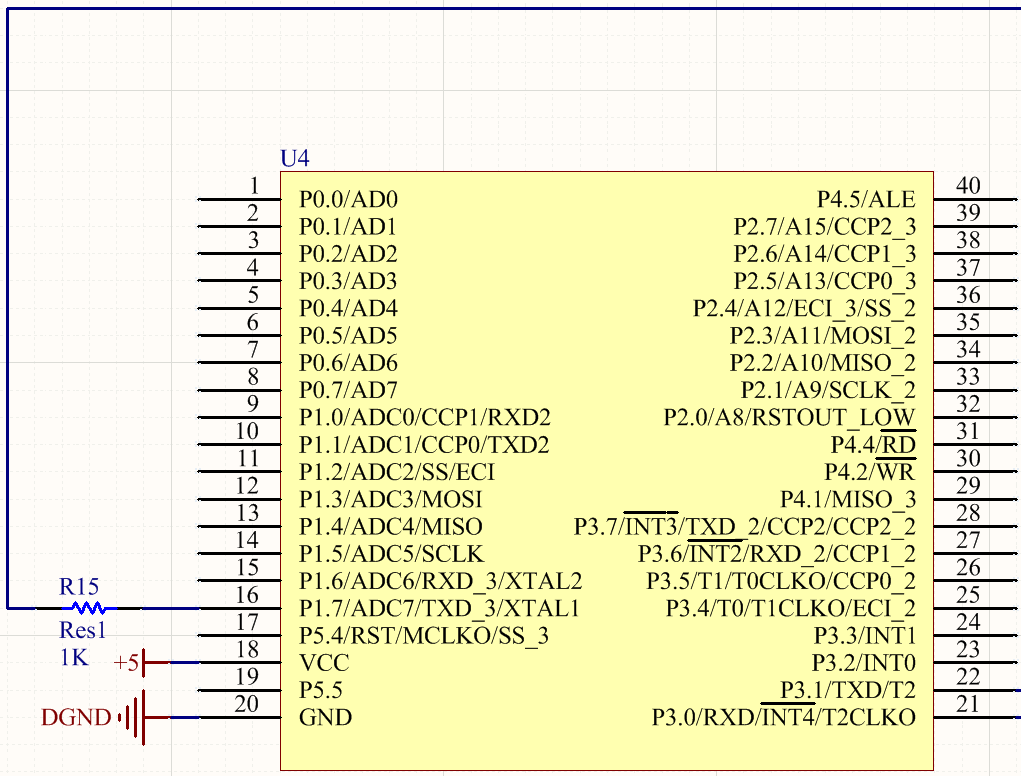
* 1. 模拟电路频率响应

使用Multisim 14.0仿真得到：



图中纵坐标为线性标度，横坐标为对数标度。幅频曲线比较理想，但相位曲线在通带内不甚平坦。然而考虑到精调相频曲线需要极其复杂的电路，因此不再改进相频曲线。

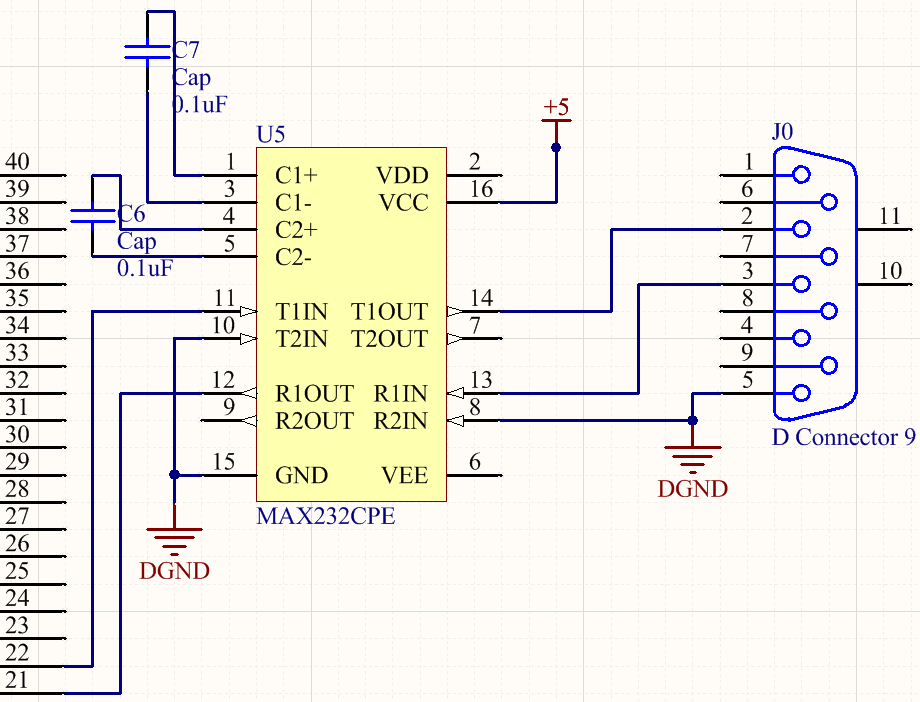
* 1. 单片机



我开发单片机的经验不多，因此听从老师的建议，采用STC公司的IAP15F2K61S2。该单片机内置时钟、高速ADC和硬件串口通信，足够使用。ADC输入端应有1k限流电阻。

STC公司的资料对开发者非常友好。不仅有详细的手册，还有大量工具包和示例代码。具体单片机程序在后文给出。

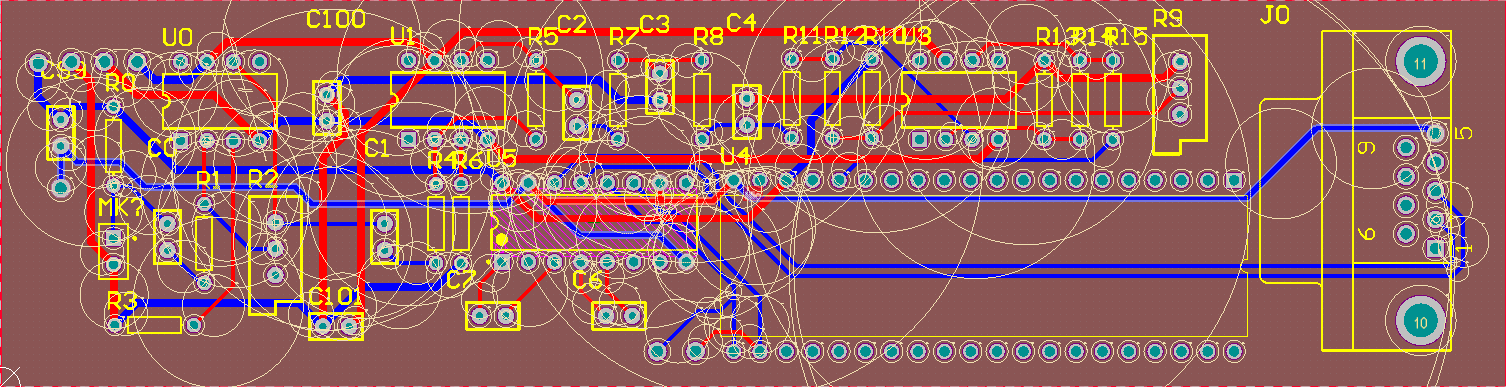
* 1. 串口通信



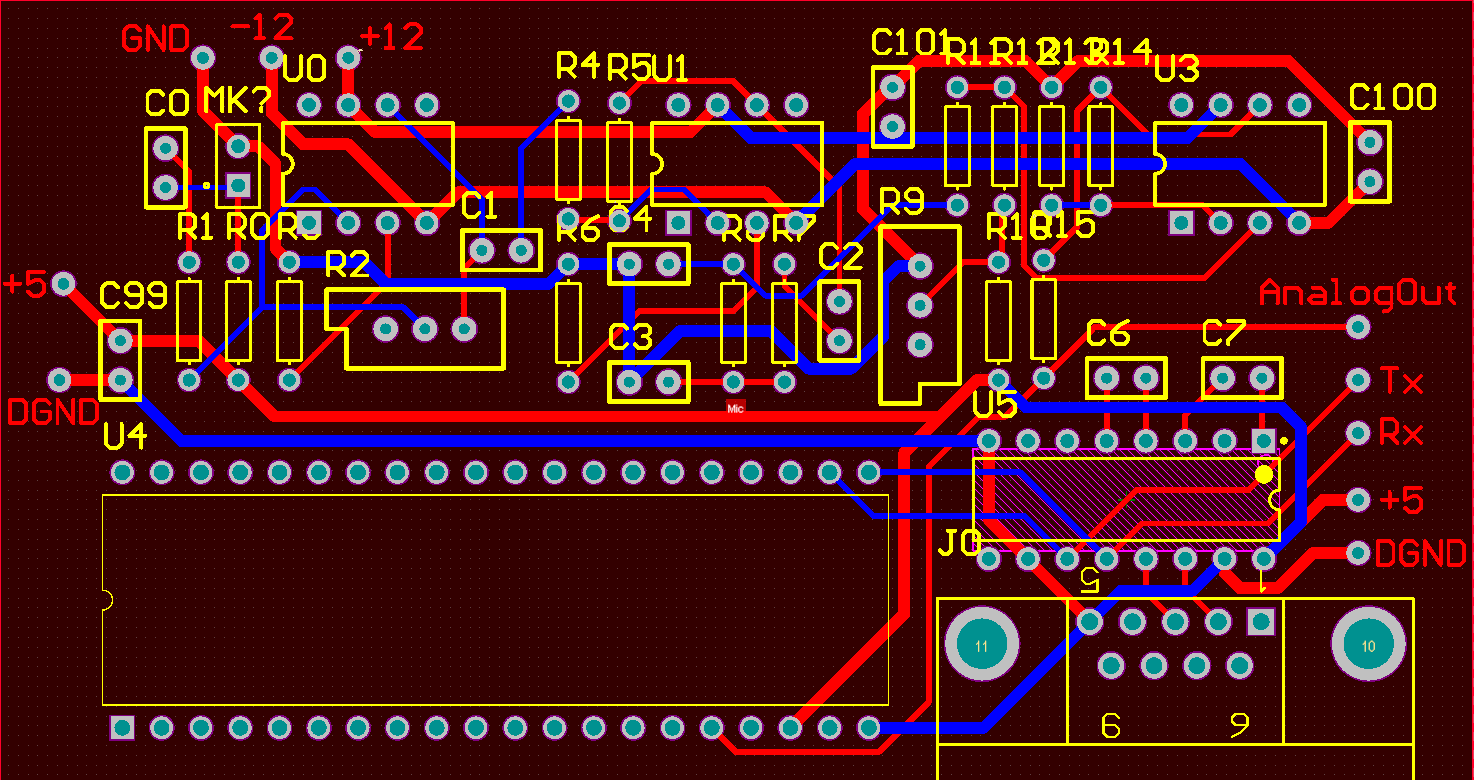
使用MAX232CPE将TTL电平转为串口通信使用的电平。实际制作中完全没有用到串口和MAX232芯片，而直接使用淘宝上极易买到的STC单片机编程器代替。单片机编程器通常内置CH340，直接使用USB接入电脑，且对于电脑和单片机而言都相当于串口。

1. 布线

布线可能是本实验中最有技术含量的工作。最初设计了一块横向长度极长，纵向高度极短的电路板。虽然该板面积很小，但被告知制作价格很高。于是意识到面积并不是决定价格的唯一因素。当长宽均小于10cm时，才能得到足够低的报价。



于是修改板子如下（±12V按正确接法标出）：



这个板子的纵向高度限制为刚好能塞进我的塑料盒子。布线的经验是，首先安排需要直连电源和GND的元件。为便于焊接，焊接点之间不能太过靠近，也不能太接近bottom面的其他网络的导线（上图中用蓝色表示）。板子对外的接口所在的网络全部布置于top面，以保证外接导线（焊接于bottom面）不易损坏板上的走线。AnalogOut端口是输入单片机ADC的信号（已经通过限流电阻）。打孔时应保证杜邦线可直插，且不易脱落。

事实上这块电路板的尺寸还能继续缩减。

1. 单片机程序

单片机程序是本实验需要的核心技术之一。本实验涉及单片机内三种硬件：CPU、ADC和串口。其中串口的工作速度远远慢于其他两者（最高波特率115200），因此音频的采样将受限于串口。

本实验中串口的传输速度无法实时发送高质量音频，所以应选取合适的采样率和采样位数来保住音质。ADC可采样的深度为10 bit，但我最终只取前8 bit。串口达到最大发送速度时，每秒可发送14400 Byte。

一般而言，两个不同工作速度的硬件相互配合时，有两种模式。以ADC和CPU为例，我们可以让CPU不断地询问ADC是否已完成转换，也可以在ADC完成转换时主动通知CPU去取出结果。在STC的示例代码中，这两者分别被称为查询方式和中断方式。查询方式的具体实现为：

while(1)

{

if(ADC采样完成)

{

ADC开始下一次采样;

串口发送(ADC上次采样结果);

}

}

而中断方式需要依赖中断函数。当ADC转换完成后，不论CPU原本在干什么，中断函数将立即被CPU执行。具体实现为：

void adc\_isr() interrupt 5

{//此时ADC刚刚完成一次采样

ADC开始下一次采样;

串口发送(ADC上次采样结果);

}

中断方式免去了CPU不断查询的操作，因此效率显然远高于查询方式。然而**上述中断方式的代码在实际运行时无法输出正确结果**！这是由于串口发送的速度远低于ADC采样速度。当ADC采样结束时，串口发送被立即打断，导致输出的结果错误。

考虑到运行速度上CPU>>ADC>>串口，因此我们应当让串口掌握主动权，在串口完成传输时通知CPU发出下一个采样结果。于是我在串口上使用中断方式，而ADC使用查询方式。理论上ADC也可以使用中断方式，但同时使用2个中断时，一旦ADC出现某种异常（例如尚未采样完成）导致时序混乱，很可能使CPU在串口传输的过程中执行ADC的代码，导致后续ADC与串口相对时序全部出错。于是我的代码结构为：

void UART1\_int (void) interrupt UART1\_VECTOR

{//串口中断函数。当串口发送数据完成后，立即执行此函数

停止发送;//清除发送标志位，表示单片机现在并不打算发数据

}

void main()

{

while(1)

{

启动ADC;

while(ADC采样尚未完成)

{

//什么也不做

}

//现在ADC已经采样完成

ADC开始下一次采样;

串口发送(ADC上次采样结果);

}

}

这样的代码工作流程为：

1. 第一次启动ADC。CPU不断询问ADC是否完成
2. 完成第一次采样。CPU取出结果并立即重启ADC。之后CPU开始忙于串口发送。同时ADC也在工作。
3. ADC采样完成。此时串口尚未完成发送。
4. 串口完成发送。CPU先通过中断函数停止串口发送，然后回到while(1)中启动ADC。

正常情况下，ADC本来就是启动状态，且已经采样完成，因此这一步不会改变任何东西。while(ADC采样尚未完成)也直接被跳过（因为事实上已经采样完成了）。串口立即开始发送新的数据。

非正常情况下，ADC可能没有采样完成，或者根本没有启动。此时会等到ADC采样完成再开始新下一次发送。每出现一次非正常情况的代价仅为一个采样点的采样间隔出错。

此代码可以有效控制ADC出错的后果，保证ADC与串口的相对时序，且CPU并没有花费大量时间查询ADC是否转换完成。大多数情况下，CPU第一次查询时ADC就已转换完成。

编程中参考了<https://blog.csdn.net/liyu3519/article/details/73436653>的代码，以及STC官方烧录软件中给出的示例代码。

csdn参考代码的原始功能为：电脑向单片机发送数据，单片机原样发回前10ms收到的所有数据。因此我的最终代码还带有一些发送字符串的函数，且串口中断函数会清除接收标志位（即停止从电脑接收一切内容）。

1. 电脑端播放程序

STC单片机烧录软件带有串口助手，可以记录电脑从串口收到的数据。因此不需要自己编写串口实时接收程序，只需保存串口助手的结果，再解码播放即可。

1. 经验总结
   1. 面向淘宝选择元件。先看能买到什么，再从能买到的元件里选最好的。
   2. 确保元件性能足够。例如本实验中的运放选择（我仿佛能听见LM358放大得到的宇宙背景噪声）。
   3. 做原理图和PCB板需要战略思想。要便于实际采购和制作；要考虑到各种可能出现的问题，兼容各种实现方式。如果出现的错误必须要通过修改电路板来弥补，会产生巨大的成本。

例如本实验的原理图经过非常认真的仿真，确保实际元件参数变化较大时，仍可调节可变电阻使电路组成工作。PCB中，编程器和MAX232都能用，放大倍数和加法器直流抬升幅度都可调节，关键节点电压值容易测量，不用花时间区分花花绿绿的电阻。

* 1. 编程时，不能带着以CPU为中心的思想。通常IO是最慢的。不同硬件配合时，在最慢的硬件上运行的代码应拥有最大的主动权（例如由最慢的串口来主动要求CPU清除发送标志位）。