基于 DSP5402 的频谱分析仪设计

硬件课程设计报告

课　　题：　频谱分析仪设计

组　　员：黄翘磊　提高 0901　　U200914500

任世伟　提高 0901　　U200914766

陈雷　　提高 0901　　U200913674

指导老师：王莹

课设评价：

课设成绩：

课程名称：简易频谱分析仪

【摘要】

信号在频域中表示的波形称为信号的频谱，对信号进行频域分析，这对理解信号含义、正确处理信号具有重要的辅助作用。但现有的频谱分析仪一般价格昂贵。本课程设计实现一种功能相对简单但价格相对低廉，使用更加方便的简易频谱分析仪，以满足需要。我们以 TMS320C5402 为主要的 DSP 芯片，对音频编解码芯片 TLV320AIC23 采样音频信号得来的数据计算 FFT，得到输入信号的频谱数据，并对频谱数据用 UDP 协议通过网线传送到电脑上加以显示。项目分为三大块。

（1） 利用 TLV320AIC23 音频编解码芯片，采样频率为 88.2KHz，对输入的音频信号进行 16bit 的采样， 将采样结果存在缓存中。

（2） 读取缓存中的采样数据，FFT 算法用 TI 公司提供的 TMS320C5402 芯片来实现，本项目可以实现 1024 点的 FFT 运算。

（3） 用网线将电脑与 TMS320C5402 开发板相连，利用 UDP 协议，在电脑上捕获 DSP 芯片 FFT 后得到频谱数据，并加以实时显示。

【关键词】 音频解码，FFT，频谱，UDP

0 引言

频谱分析仪是研究电信号频谱结构的仪器，用于信号失真度、调制度、谱纯度、频率稳定度和交调失真等信号参数的测量，可用以测量放大器和滤波器等电路系统的某些参数，是一种多用途的电子测量仪器。它又可称为频域示波器、跟踪示波器、分析示波器、谐波分析器、频率特性分析仪或傅里叶分析仪等。现代频谱分析仪能以模拟方式或数字方式显示分析结果，能分析 1 赫以下的甚低频到亚毫米波段的全部无线电频段的电信号。仪器内部若采用数字电路和微处理器，具有存储和运算功能；配置标准接口，就容易构成自动测试系统。

频谱分析仪系统主要的功能是在频域里显示输入信号的频谱特性。频谱分析仪依信号处理方式的不同，一般有两种类型；即时频谱分析仪与扫描调谐频谱分析仪。即时频率分析仪的功能为在同一瞬间显示频域的信号振幅，其工作原理是针对不同的频率信号而有相对应的滤波器与检知器，再经由同步的多工扫描器将信号传送到 CRT 或液晶等显示仪器上进行显示，其优点是能显示周期性杂散波的瞬间反应，其缺点是价昂且性能受限于频宽范围，滤波器的数目与最大的多工交换时间。最常用的频谱分析仪是扫描调谐频谱分析仪，其基本结构类似超外差式接收器，工作原理是输入信号经衰减器直接外加到混波器，可调变的本地振荡器经与 CRT 同步的扫描产生器产生随时间作线性变化的振荡频率，经混波器与输入信号混波降频后的中频信号（IF）再放大，滤波与检波传送到 CRT 的垂直方向板，因此在 CRT 的纵轴显示信号振幅与频率的对应关系。影响信号反应的重要部份为滤波器频宽，滤波器之特性为高斯滤波器，影响的功能就是量测时常见到的解析频宽（RBW,ResolutionBandwidth).在频谱测量中，适当的 RBW 宽度是正确使用频谱分析仪重要的概念。

现代频谱分析仪基于快速傅里叶变换(FFT)，通过傅里叶运算将被测信号分解成分立的频率分量，达到与传统频谱分析仪同样的结果，。这种新型的频谱分析仪采用数字方法直接由模拟/数字转换器（ADC）对输入信号取样,再经 FFT 处理后获得频谱分布图。在这种频谱分析仪中，为获得良好的仪器线性度和高分辨率，对信号进行数据采集时 ADC 的取样率最少等于输入信号最高频率的两倍。FFT 的性能用取样点数和取样率来表征，例如用 100KHz 的取样率对输入信号取样 1024 点，则最高输入频率是 50KHz 和分辨率是 100Hz。如果取样点数为 2048 点，则分辨率提高到 50Hz。由此可知，最高输人频率取决于取样率，分辨率取决于取样点数。FFT 运算时间与取样，点数成对数关系，频谱分析仪需要高频率、高分辨率和高速运算时，要选用高速的 FFT 硬件，或者相应的数字信号处理器(DSP)芯片。例如，10MHz 输入频率的 1024 点的运算时间 80μs，而 10KHz 的 1024 点的运算时间变为 64ms,1KHz 的 1024 点的运算时间增加至 640ms。当运算时间超过 200ms 时，屏幕的反应变慢，不适于眼睛的观察，补救办法是减少取样点数，使运算时间降低至 200ms 以下。

总之，高分辨率，快速的频谱分析仪需要更昂贵的价格，这样的价格是学生个体很难承受的，所以，根据我们已学的硬件软件知识设计并制作一个简易的频谱分析仪是很有意义的。

1 设计目标

1.1 总体任务

设计并制作一个简易频谱分析仪，该频谱分析仪对输入信号进行简单频谱分析，并能够显示信号的总功率、各频率分量的功率信息。

1.2 任务要求

基本部分：
1 对不高于 20kHz 的输入信号能够在 LCD 上显示信号的总功率
2 各频率分量的功率等信息；
发挥部分：
1 扩展能够处理的输入信号频率范围；
2 增加显示失真度信息；
3 其它。

1.3 实现程度

可以对麦克风输入的音频信号，或信号发生器输入的低于 20KHz 信号，进行频谱分析，得到各频率分量的幅度和信号总功率，并在电脑上加以实时显示。

2 成员分工

黄翘磊：该同学负责对音频的 AD 采样部分
陈雷：　该同学负责对时域信号进行 FFT 运算
任世伟：该同学负责将得到的频域数据传输到电脑上，并实时显示

3 系统方案

3.1 系统方案的比较和选择

由于本次课程设计题目较新，基本没有往届学长的经验，所以在本次课程设计中，我们在前期讨论过多种方案，其中有的很新颖但完成难度较大，有的看

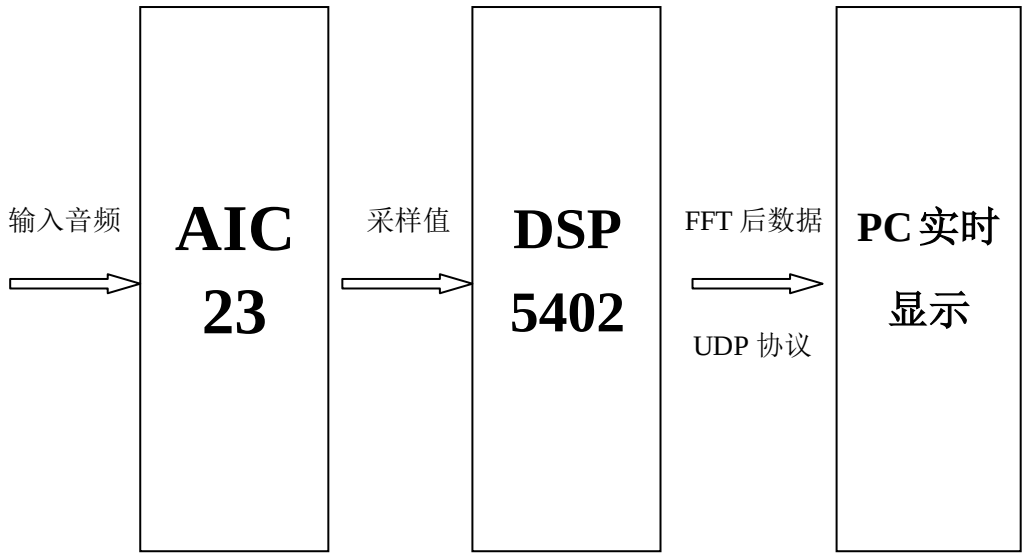
起来很简单但实施过程却很麻烦，这里我们主要分析以下两种方案。

1 利用 DSP 实验课程上用到的 TRDSSP5409 实验箱，该实验箱配备 TLV15711 作为 AD 采样芯片，用 TLV5619 芯片作为 DA 模块输出，主控 DSP 芯片为 TMS320VC5409。本实验箱有较好的集成，并且课程上用过此试验箱，也很熟悉。

2 利用买的 TMS320VC5402 学习板 ，本开发板配备 TLV320AIC23 音频编解码芯片，此芯片内置耳机输出放大器，对输入信号进行 AD 采样，主控芯片 TMS320VC5402，对采样数据进行 FFT 运算得到频谱信息，然后通过网线输入电脑实时显示。

第一种方案因为选用了我们上 DSP 实验课程的实验箱，我们感觉很熟悉，也很有信心，但后来深入时就发现问题：扩展 IO 太少，FFT 后输出显示难以实现，实验箱上的液晶显示屏几乎没有可编程性。此方案最终放弃。第二种方案，我们通过上网查找资料，选择了 5402 这一开发板，它可以完全满足我们的三个需求：AD 采样，FFT 运算，显示输出。

3.2 整体框图



框图描述：

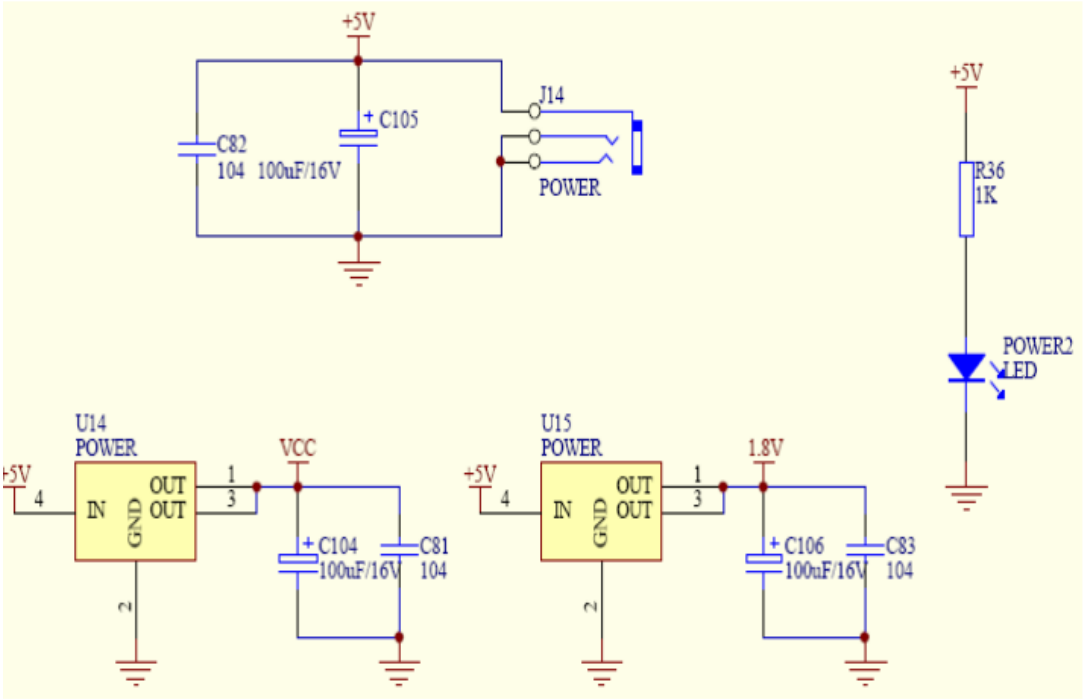
最开始通过麦克风输入音频信号（也可以用信号发生器外接频率在 5KHz 以下的电平信号），由 DSP 开发板上的 TLV320AIC23 来进行 AD 采样，TLV320AIC23 是 TI 公司推出的一款高性能立体声音频编解码器，内置耳机输出放大器，可很好地完成输入信号的放大和 AD 采样，将采样值保存于缓存中，然后 DSP5402 读取，并进行 FFT 运算，算得结果通过网线，利用 UDP 协议发送到 PC 上，实时接收，并对频谱分量和总功率加以显示。

4 系统硬件设计与实现

硬件部分虽然不是完全我们设计制造的，但选择硬件和分析硬件以适应我们的功能要求也是我们工作的很大一部分。硬件部分主要分为下面几块。

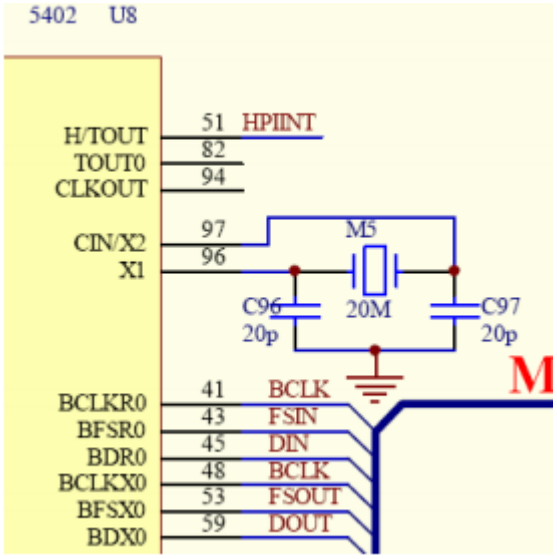
4.1 供电电路

电源可由外部电源引入，电源插孔 J14 标识为内正外负，+5V 稳压直流电源输入。 电源也可由 USB 接口引入，通过跳线 JP11 进行选择（接上用 USB 供电）。 AMS1117-3.3 电源转换芯片作为 5V 转 3.3V 的高性能稳压芯片，为这个开发板提供稳定可靠的主电源 3.3V 。CX1117-1.8 电源转换芯片提供 1.8V 给 DSP 内核使用。 1117 输出后的 47uF（或 100uF）的电容不能省略，这样能更好的保证电源质量。具体电路如下



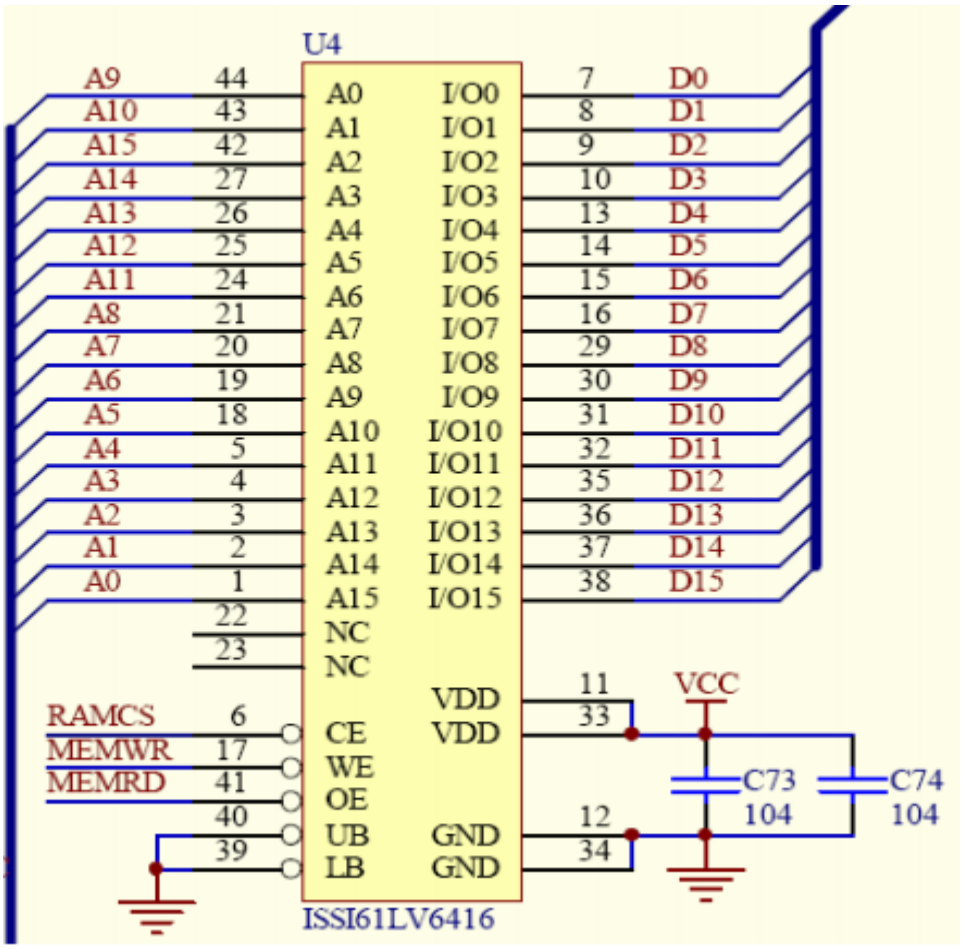
4.2 时钟振荡电路

锁向环（PLL）模块主要用来控制 DSP 内核的工作频率，外部提供一个参考时钟输入，经过 PLL 倍频或分频后提供给 DSP 内核。 下图为本开发板上采用的电路，采用的内部振荡器方式。选用的外部晶振为 20M 的电路。



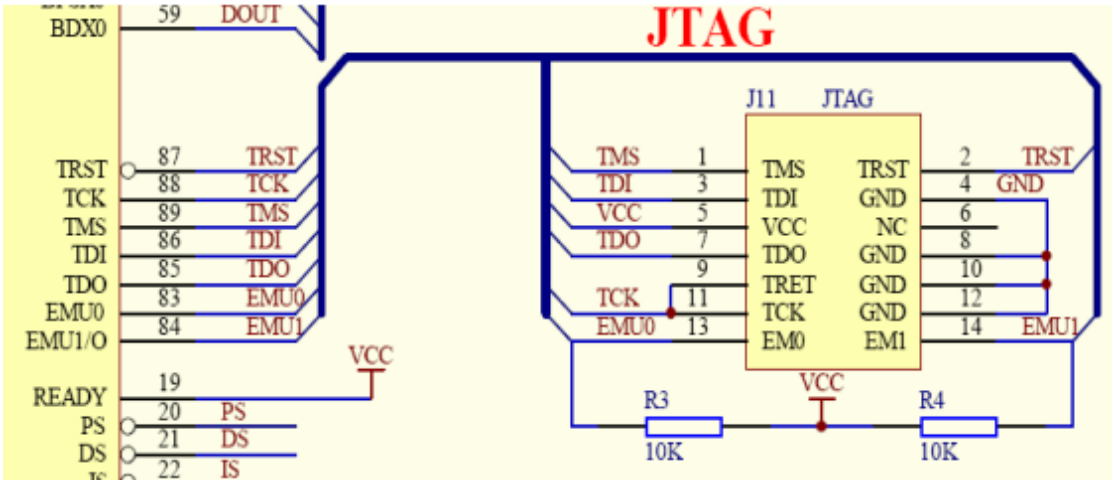
4.3 外扩 RAM 电路

选用的 RAM 型号为 IS61LV6416， 64K×16bit 大小。其数据总线和地址总线直接和 VC5402 进行无缝连接。电路图如下



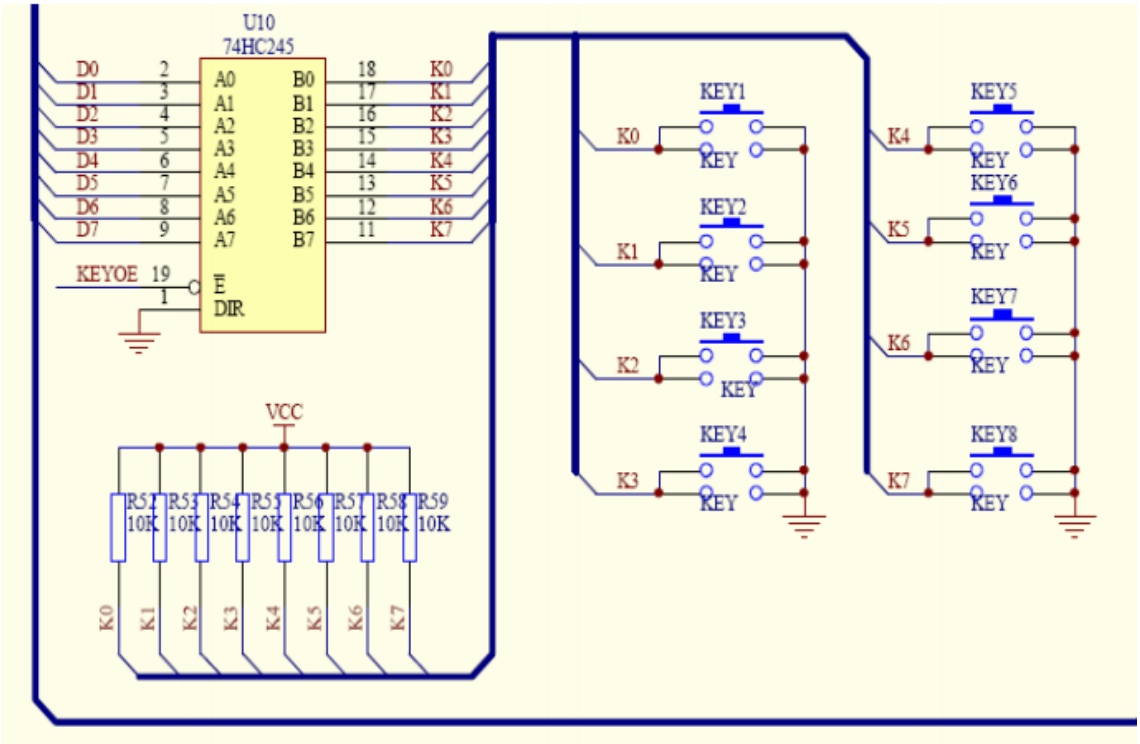
4.4 JTAG 接口电路 接口电路 接口电路 接口电路

JTAG 接口提供对 DSP 的内部 FLASH 的烧写和仿真通讯。该部分的引脚定义和连接如下



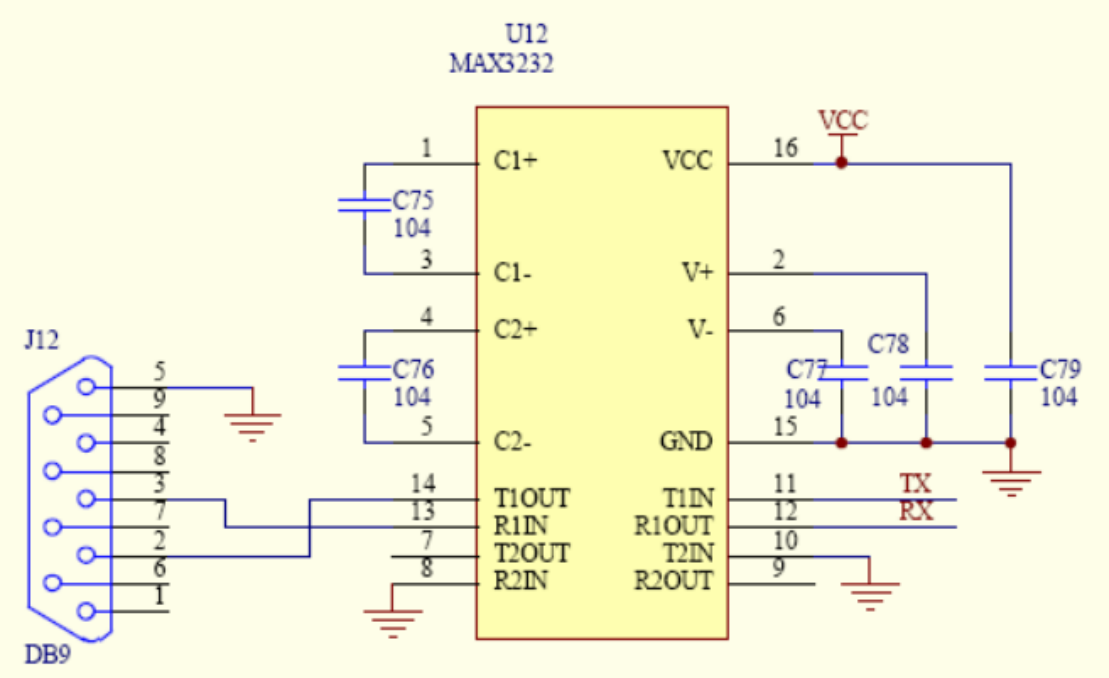
4.5 按键电路

开发板上配置有 8 个按键（是 K1-K8），对 8 个按键的控制是通过 74HC245 进行缓冲输出到 VC5402 数据接口上的。



4.6 串口通讯电路

RS-232 接口芯片是 MAX3232，是 3.3V 供电的芯片。DB9 接头上的 X_232，RX_232，标识是指 PC 端的发送和接收，串行通讯垢双方的 TX，RX 必须反接。所以在原理图电路中 PC 的 TX_232 输出的目标板的 RXD，目标板的 TXD 经过 MAX232 上输出的是 PC 的 RX_232。实验时需要用一要直连线（一头针一头孔），MAX3232（或 MAX232）更详细的资料请参考其对应的数据手册。



4.7 音频控制电路

本部分涉及放大和 AD 采样，是本课程设计最重要的三个部分之一。本开发板芯片为 TLV320AIC23，是 3.3V 供电的芯片。外接有耳机插口（PHONE）和话筒插口（MIC），可以进行录音试验和边录边放的试验。本项目中，我们使用 MIC 接麦克风，输入待分析的音频信号。

TLV320AIC23 是 TI 公司推出的一款高性能立体声音频编解码器，内置耳机输出放大器，支持 mic 和 line in 二选一的输入方式，本项目使用 mic 输入音频信号，其输入和输出都具有可编程的增益调节功能。TLV320AIC23 的模／数转换器(ADC)和数，模转换器(DAC)集成在芯片内部．采用先进的 Σ 一Δ过采样技术．可以在 8kHz 至 96kHz 的采样率下提供 16bit、20bit、24bit 和 32bit 的采样数据。ADC 和 DAC 的输出信噪比分别可达 90dB 和 100dB。同时。TLV320AIC23 还具有很低的功耗(回放模式为 23mW。节电模式为 15μw)。上述优点使得 TLV320AIC23 成为一款非常理想的音频编解码器，与 TI 的 DSP 系列相配合更是相得益彰，这也是我们选择本方案的很大的原因。

AIC23 的管脚和内部结构框图如下。从下图可以看出，AIC23 主要的外围接口分为以下几个部分：

1　数字音频接口：主要管脚为
BCLK－数字音频接口时钟信号（bit 时钟），当 AIC23 为从模式时（通常情况），该时钟由 DSP 产生；AIC23 为主模式时，该时钟由 AIC23 产生；
LRCIN－数字音频接口 DAC 方向的帧信号（I2S 模式下 word 时钟）
LRCOUT－数字音频接口 ADC 方向的帧信号
DIN－数字音频接口 DAC 方向的数据输入
DOUT－数字音频接口 ADC 方向的数据输出
这部分可以和 DSP 的 McBSP（Multi-channel buffered serial port，多通道缓存串口）无缝连接，唯一要注意的地方是 McBSP 的接收时钟和 AIC23 的 BCLK 都由 McBSP 的发送时钟提供。

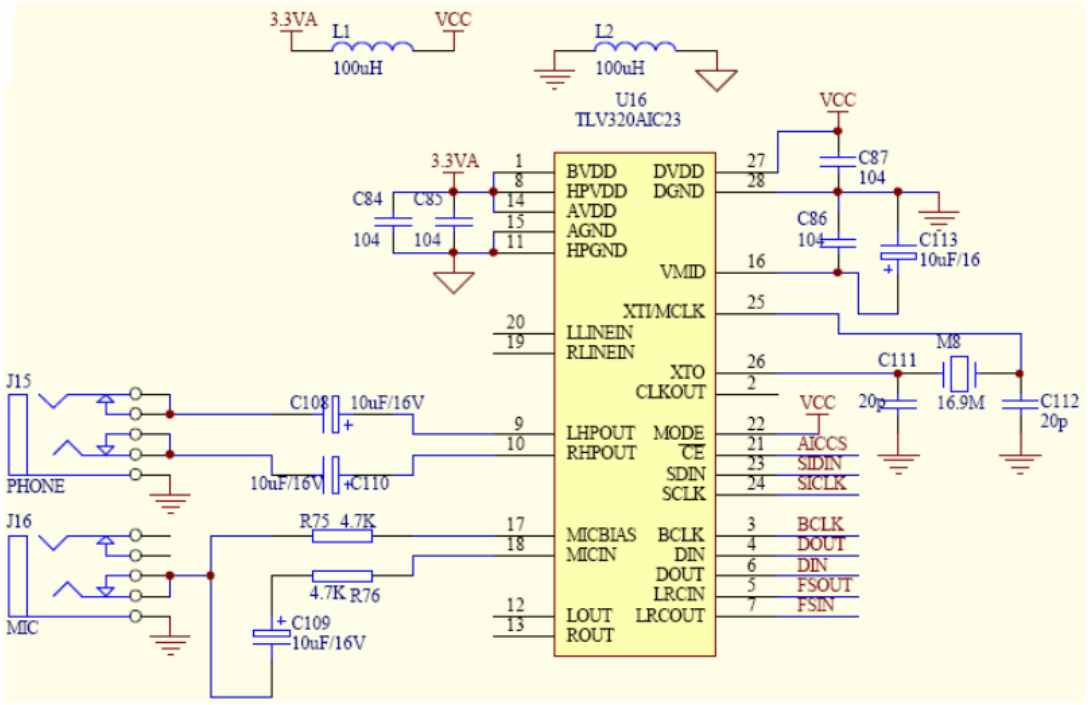
2　麦克风输入接口：主要管脚为
MICBIAS－提供麦克风偏压，通常是 3/4 AVDD
MICIN－麦克风输入，由 AIC 结构框图可以看出放大器默认是 5 倍增益

3　LINE IN 输入接口：主要管脚为
LLINEIN－左声道 LINE IN 输入
RLINEIN－右声道 LINE IN 输入

4　耳机输出接口：主要管脚为
LHPOUT－左声道耳机放大输出
RHPOUT－右声道耳机放大输出
LOUT－左声道输出
ROUT－右声道输出

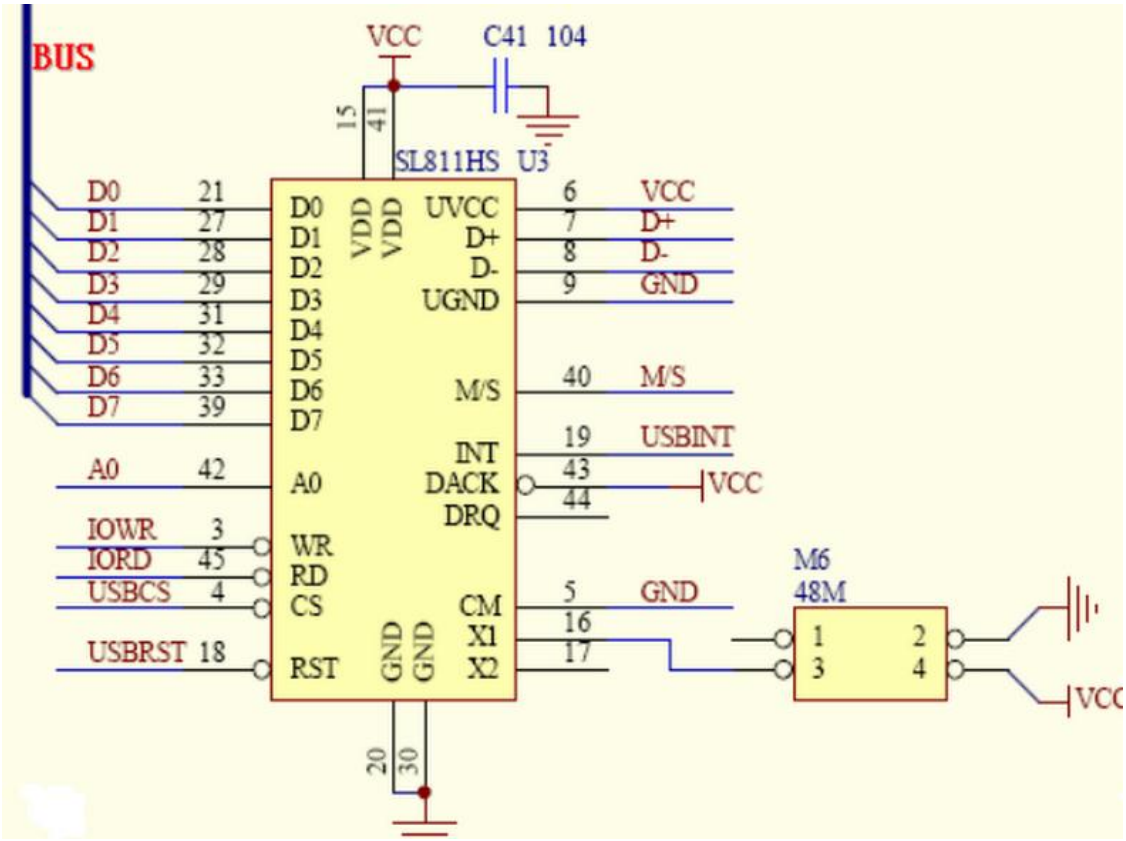
5　配置接口：主要管脚为
SDIN－配置数据输入
SCLK－配置时钟
DSP 通过该部分配置 AIC23 的内部寄存器，每个 word 的前 7bit 为寄存器地址，后 9bit 为寄存器内容。

6　其他：主要管脚为
MCLK－芯片时钟输入(12.288M、11.2896M、18.432M、16.9344M)
VMID－半压输入，通常由一个 10U 和一个 0.1U 电容并联接地
MODE－芯片工作模式选择，Master 或者 Slave
CS－片选信号（配置时有效）
CLKOUT－时钟输出，可以为 MCLK 或者 MCLK/2
总结的来说，在我们实际使用中，采样频率为 88.2KHz，每次采样提供 16bit 的采样数据



4.9 USB 控制电路

本开发板 USB 控制电路采用了兼容性较好的 SL811HS 芯片，可以 U 盘读写等实验，但此功能部分本项目没有用到。



5 系统软件设计与实现

本节简述了本课设软件设计中最重要的三个部分：AD 采样，FFT，显示输出。

5.1 采样部分原理与实现

本项目中放大与 AD 采样部分我们使用的是 TI 推出的一款专门针对 DSP 芯片的音频编解码器 TLV320AIC23。AIC23 是 TI 公司推出的一款高性能立体声音频编解码器，内置耳机输出放大器，输入和输出都具有可编程的增益调节功能，本项目没有使用 AIC23 的 DA 输出部分。

除了复位寄存器外，TLV320AIC23 芯片共有 9 个控制寄存器．每个寄存器控制字长为 9bit．地址位为 7bit，共有 16bit。地址位为高 7 位而控制字在低 9 位。具体如下：

Register0：左声道输入音量控制，缺省值为 0x0017
Register1：右声道输入音量控制，缺省值为 0x0017
Register 2：左声道输出音量控制，缺省值为 0x01F9
Register 3：右声道输出音量控制，缺省值为 0x01F9
Register 4：模拟音频通道设置，缺省值为 0x0011
Register 5：数字音频通道设置，缺省值为 0x0000
Register 6：节电模式控制，缺省值为 0x0000
Register 7：数字音频接口格式控制，缺省值为 0x0043
Register 8：采样率控制，缺省为 48kHz
Register 9：数字音频接口激活开关．缺省值为 0x0001

在一般应用中，上述寄存器中大多数参数无需更改，本项目中修改的主要是 Register 8，需要通过它来改变 TLV320AIC23 的采样率。本项目中，我们用的是 88.2KHz 的采样率。

AD 模块软件编写如下：

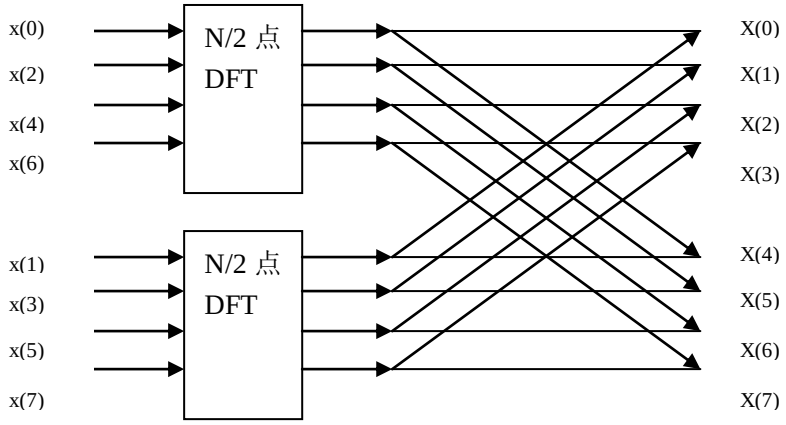
```
int data_ptr;
DATA ADbuf[1024]; //缓存区，存储采样数据，然后送入之后的FFT部分
int flag=0;
int     DA_wptr,DA_rptr,y;
int     Flag = 0;
int     play_mode;
//播放模式，0为播放给定警报音，不采样，仅为测试用，1为读取采样数据，本项目中为1
interrupt void  timer0()
{
    ms++;
}
 interrupt void  codec_ch0_in()
//接收0中断，前面部分有开接收0中断，篇幅所限，可参考完整程序中注释部分
```

```
{
	int temp;
	temp=*DRR10;   //数据接收寄存器值赋给temp，暂时保存采样点值
	DA_rptr++;
	if(DA_rptr>=1024)
		DA_rptr=0;  //缓存区有1024个存储空间
	ADbuf[DA_rptr]=temp;   //保存录音数据在缓存区，供FFT模块读取
	x[DA_rptr]=temp;
		if(DA_rptr==1023) //若缓存区读取慢，存够1024个，开始FFT读取运算
	{
	//DA_rptr=511;
	cbrev(x,x,NX/2);
	rfft(x,NX,1);
	UDPActiveTxData(x);
	}
}

\
```

5.2 频谱运算 FFT 运算模块

若按照定义来直接计算离散傅里叶变换，则时间复杂度有 $O(n^2)$,其中 n 为进行傅里叶变换的点数。这个时间复杂度对于实时计算来说要求较高，为此我们使用快速傅里叶变换算法来进行计算。快速傅里叶变换有许多种，我们使用基 2 时间抽取的 fft 算法。该算法的基本思想是，将长度为 n 的序列分为两个长度为 n/2 的序列来计算离散傅里叶变换，对每个 n/2 长的序列，再将其分为两个 n/4 长的序列，如此下去，就可以将 n 点傅里叶变换的计算时间复杂度降低至 O(nlog(n))。



如上图所示，在进行快速傅里叶变换时，若需要变换后的序列的顺序是正确的，则输入序列需要按特殊的顺序排列，这就是所谓的码位倒置。在得到全部

的 n 点输入信号后，我们必须先将其进行码位倒置。

蝶形运算的优点是可以进行所谓同址或原位计算。设将输入 x(0),x(4),x(2),x(6),x(1),…,x(7)分别存入存储单元的 M(1),M(2),M(3),…,M(8)中。首先，存储单元 M(1)和 M(2)中的数据 x(0)和 x(4)进入运算其并进行蝶形运算，由于以后不会再用到这两个数据，因此不需要保留。这样，蝶形运算的结果便可以送到 M(1)和 M(2)中保存起来。类似的，M(3)和 M(4)中的 x(2)和 x(6)进行蝶形运算后的结果也被送回到 M(3)和 M(4)中保存等等。这样直到最后一级的最后一个蝶形运算完成。可以看出，每一级的蝶形运算的输入与输出在运算后可以储存在同一个地址的存储单元中，这种运算的优点是可以节省存储单元，从而降低硬件成本。

5.3 实时显示输出模块

由于没有找到合适的与开发板接口相符的 LCD 显示器，且使用转接板 12864 与 1602LCD 显示屏分辨率过低，这样会导致显示结果不能够给出直观且较准确的信号频谱分析结果，故本系统通过 UDP 协议将数据传输，在 PC 端接收后进行显示。

1 发送端

UDP 是不面向连接的网络协议，使用 UDP 协议进行发送可以简化控制流程，不需要进行连接管理，减少主芯片的压力。并且通过广播的方式将数据发向指定端口，这样将接收 PC 与开发板用网线直接相连便可接收数据。通过按指定方式进行组帧，并用软件控制，将数据从以太网适配器发送出去。

2 PC 接收端

PC 端在每次图形要刷新前将阻塞的就收 UDP 数据包，在接收到之后将 FFT 结果通过直方图的方式在 PC 的显示器上显示出来。

6 模块测试与调试

本项目进行过程中，出现过很多问题，有硬件的问题和软件的问题，但我们静下心来，一步一步解决了碰到了大部分问题，得到了最终的结果

6.1 采样失败的问题

在 AD 采样时，会存在采样值读取不进去的问题，这在我测试完成声音回放程序时碰到的，声音回放程序不进行 FFT 运算，直接把读入的数据送到 DXR10 音频输出。这个问题的解决方式是把麦克风输入和耳机输出分开，单独用麦克风

和耳机接入到 5402 开发板上。我开始使用的是带耳麦的耳机，就出现了这样的问题，可能与这种耳麦里面的接线有关系，因为单独地看，耳机分三根线，左声道，右声道和地线，麦克风分为两根线；但是在耳麦中，只有四根线，可能这种接线造成了输入信号的混乱，AD 采集不成功。我换了单独的麦克风，单独的耳机，接入开发板后，此问题得到解决。

6.2 测试是否正确的方法

提供一个验证课设是否正确的方法，我们最后在调试过程中主要用到了此方法。因为本项目主要针对音频输入信号进行频谱分析，而人的输入音频没有确定的值的，不适合验证。我们将麦克风的两根输入信号线剪开，接上信号发生器，输入特定频率的正弦信号即可验证本课设是否正确。为满足 AIC23 芯片的采样特性和麦克风的信道特性，此时输入信号峰峰值约为 0.05V。

7 实验总结

7.1 方案的优缺点分析

方案优点：TLV320AIC23 音频编解码器与 DSP5402 衔接很好，是 TI 针对数据采集变换特定的芯片，使用过程几乎没有硬件 bug，编写程序时很方便，特别的，TLV320AIC23 是自带放大和 AD 采样的，且功能稳定，这对我们的工作有很大的简化。

方案缺点：本开发板自带的 TLV320AIC23 音频编解码芯片虽然功能强大且稳定，但针对音频解码的特性也使得我们的采样信号频率不能太大，最大的采样频率选项为 96KHz。在本次课设的调试过程中，信号最大频率不能高于 40K，这对于基本信号是可以满足的，但我们知道，一般需要进行频谱分析的信号许多都是高频信号，受限于 TLV320AIC23 音频编解码芯片，我们本次课设不能针对这样的信号进行处理。

7.2 项目的核心及实用价值

1 项目的核心在于三个部分联调合作来完成频谱分析功能：

对于 AD 采样部分，我用声音回放程序来测试正确性，确定无误后参与和后面 FFT 模块的联调。对于 FFT 部分，每次针对特定的 1024 个实数数据进行 FFT 运算，得到结果。对于显示部分，通过网线把 DSP5402 算得的数据利用 UDP 协议传输到电脑端，再利用我们编写的 java 接收显示程序，得到界面友好并且结果正确的频谱结果。项目进行中，我们先单独测试每个模块功能正常，再连接起整体，一步一步得到理想的结果。

2 项目的使用价值：

高分辨率，快速的频谱分析仪需要很昂贵的价格，这样的价格是学生个体很难承受的，但有一台可以实时分析信号频谱信息的仪器对电类学生学习信号相关课程是很有帮助的。所以，我们根据已学的硬件软件知识设计并制作了这个简易频谱分析仪是很有意义的。

7.3 改进意见

1 可以考虑换一块可以提供更高采样频率的 AD 信号采集芯片，当然这样的芯片可能就不一定集成了放大功能了，所以更换后就需要加入放大模块，这样就可以对更加高频的芯片进行频谱分析了。

2 可以换一块计算性能更强大的 DSP 芯片，提高一次 FFT 运算的点数，当然这样的代价是牺牲时间的。本项目中 FFT 模块一次用 1024 点做 FFT，采样频率为 88.2KHz，频率分辨率约为 85Hz。如果一次 FFT 点数可以增加到 2048 或者更高，则频率分辨率也会相应地增加。

7.4 课程设计心得

在本次课程设计中，我们都有较大收获。下面从以下几个方面讲述心得体会：

1 项目规划：

在本课程设计的进行过程中，我们有两个方案，之前的那个方案看起来很熟悉看起来也好像很简单很好上手，但真正实施起来却有很多问题，最终选择了第二种方案，并按时很成功地完成了任务。虽然饶了一段很大的弯路，但这也正是硬件课设这门课程要求我们的，多尝试方案，找到最适合自己团队的那个。

2 团队分工合作：

这次课程设计是一个大课题，一个人无法在这么短的时期内完成所有的任务，所以团队合作是首选的。每个人负责自己的部分，并完成和下一位同学的衔接是很有必要的。在合作中要多讨论，这样更容易解决问题。

8 参考文献

【1】 谢自美，电子线路综合设计，华中科技大学出版社，2006-6
【2】 康华光，《电子技术基础：模拟部分（第五版）》，高等教育出版社，2006
【3】 康华光，《电子技术基础：数字部分（第五版）》，高等教育出版社，2006
【4】 罗杰，电子线路设计.实验.测试，电子工业出版社，2008
【5】 TMS320C54x C 语言编译器用户指导

【6】 TLV320AIC23 datasheet 等相关 TI 官方手册
【7】 http://www.ti.com.cn
【8】 http://www.google.com

9 附录（源码清单）

源码清单：

main.c 文件

```
#include    "cpu_reg.h"
#include    "string.h"
#include    "ip.h"



//============================MCBSP PART=====================================
#include <math.h>
#include <tms320.h>
#include <dsplib.h>
#include "dataspace.h"

int data_ptr;
DATA ADbuf[1024];
//DATA ADbuf2[1024];
int flag=0;

int     DA_wptr,DA_rptr,y;
int     Flag = 0;
int     play_mode;
int     f=1;


unsigned int read_subreg0(unsigned int addr)
{
       *(SPSA0)=addr;
       return(*SPSD0);   // READ PROCESS
}

void write_subreg0(unsigned int addr,unsigned int val)
{
      *(SPSA0)=addr;   //WRITE PROCESS
      *(SPSD0)=val;
}
```

```
void McBsp0_init()
{

     write_subreg0(SPCR1,0);
     write_subreg0(SPCR2,0);

     write_subreg0(SPCR1,SPCR10_VAL);
     write_subreg0(SPCR2,SPCR20_VAL);
     write_subreg0(PCR,PCR0_VAL);

     write_subreg0(RCR1,RCR10_VAL);
     write_subreg0(RCR2,RCR20_VAL);
     write_subreg0(XCR1,XCR10_VAL);
     write_subreg0(XCR2,XCR20_VAL);
     delay(10);

     *(DXR10)=0;

     /*now enable McBSP transmit and receive*/
     write_subreg0(SPCR1,SPCR10_VAL|1);
     write_subreg0(SPCR2,SPCR20_VAL|1);
     delay(10);

     *(IMR)|=0x0010;    //开接收0中断

}
//=========================MCBSP PART END=====================================



Uint16  FrameLenth;
Uint16  RxSuccessFlag;
int     ms;
//-------------------
Uint16  TxEthnetFrameBuffer[1518/2];
Uint16  RxEthnetFrameBuffer[1518/2];
//-------------------
extern  struct  ipaddr  server_ipaddr;
Uint16  Temp;
extern  Uint16  TCPTimeout;
void    main()
        {
            unsigned int temp,i;
//          Uint16  Temp;
```

```
            asm("  STM #0000h,CLKMD ");
            while(*CLKMD & 0x01 );
            asm("  STM #40c7h,CLKMD ");  //设置CPU运行频率＝100M
/*  40C7h:5*clkin  =100M
    30c7h:4*clkin  =80M
    20c7h:3*clkin  =60M
    10C7h:2*clkin  =40M
*/
            *SWCR = 0x0001;
            asm("  stm   #4240h, SWWSR ");
     //2 wait except for on-chip program 1
            asm("  stm   #00A0h, PMST  "); //MP/MC = 0, IPTR = 001,ovly=1
            asm("  stm   #0802h, BSCR  ");
            asm("  STM   #0h,    IMR   ");

            asm("  STM #0010h,TCR  "); //关定时器
            asm("  STM #0186ah,PRD  ");//1ms
            asm("  STM #0C2fh,TCR  "); //TCR=最后四位
            asm("  STM #0008h,IFR  ");
            asm("  ORM #0008h,*(IMR) ");/*开时间中断*/

            DA_wptr = 0;
            DA_rptr = 0;

            McBsp0_init();            //串口初始化
            play_mode =1;

            asm("  RSBX  INTM ");      /*开中断*/
            Init8019();
            TCP_Init();

            ms = 0;
            TCPTimeout = 0;
            while(ms<2000);

//          ArpRequest();


            while(1)
                {
                   for(temp = 30000;temp > 0;temp -= 100)
                       for(i = 0;i < 2;i++)
                           {
                               y = 5000;
```

```
                                delay(temp);
                                y = -5000;
                                delay(temp);
                            }
                    }
        }

interrupt void  timer0()
{
     ms++;

}

/*interrupt void int0()
{
       Temp = RecFrame();
       if(Temp)     DoNetworkStuff();
       page(0);
       delay(10);
       Reg07 = 0xFF;
}*/

interrupt void  codec_ch0_in() //接收0中断
{
      int temp;

      temp=*DRR10;

      DA_rptr++;
      if(DA_rptr>=1024)
            DA_rptr=0;

      ADbuf[DA_rptr]=temp;   //保存录音数据
      x[DA_rptr]=temp;
      //ADbuf2[DA_rptr]=x[DA_rptr];   //
      if(DA_rptr==1023)
      {
      //DA_rptr=511;
      cbrev(x,x,NX/2);
      rfft(x,NX,1);
      UDPActiveTxData(x);

      }
      /*if(play_mode==0)          //0发警报声，1代表声音播放
```

```
      else
                        //  temp=2*temp;
      *DXR10=temp;       //放音

   */

}
```

RTL8019.h

```
#define     RTL8019BASE     0x2000

#define     Reg00       port2000
#define     Reg01       port2001
#define     Reg02       port2002
#define     Reg03       port2003
#define     Reg04       port2004
#define     Reg05       port2005
#define     Reg06       port2006
#define     Reg07       port2007
#define     Reg08       port2008
#define     Reg09       port2009
#define     Reg0a       port200a
#define     Reg0b       port200b
#define     Reg0c       port200c
#define     Reg0d       port200d
#define     Reg0e       port200e
#define     Reg0f       port200f
#define     Reg10       port2010

ioport      unsigned    short   Reg00;
ioport      unsigned    short   Reg01;
ioport      unsigned    short   Reg02;
ioport      unsigned    short   Reg03;
ioport      unsigned    short   Reg04;
ioport      unsigned    short   Reg05;
ioport      unsigned    short   Reg06;
ioport      unsigned    short   Reg07;
ioport      unsigned    short   Reg08;
ioport      unsigned    short   Reg09;
ioport      unsigned    short   Reg0a;
ioport      unsigned    short   Reg0b;
ioport      unsigned    short   Reg0c;
ioport      unsigned    short   Reg0d;
```

```
ioport      unsigned    short   Reg0e;
ioport      unsigned    short   Reg0f;
ioport      unsigned    short   Reg10;

#define     BroadCast     1
#define     RequestArp    2
#define     AnswerArp     3
#define     Nod           4
#define     ARP       1
#define     UDP       2
#define     IGMP      3
#define     LSS           4
#define     TCP           5
#define     ICMP          6

#define     IP_FRAME    0X0800
#define     ARP_FRAME   0X0806

extern  void    SendFrame(Uint16    *buf,Uint16 len);
extern  Uint16  SwapByte(Uint16value);

struct  ipaddr
        {
            Uint16  addr2_1;
            Uint16  addr4_3;
        };
struct  mac
        {
            Uint16  addr2_1;
            Uint16  addr4_3;
            Uint16  addr6_5;
        };

struct  iphdr
        {
            Uint16  tos_version;
            Uint16  tol_len;
            Uint16  id;
            Uint16  frag_off;
            Uint16  protocal_ttl;
            Uint16  chksum;
            struct  ipaddr saddr;
            struct  ipaddr daddr;
        };
```

```
struct  udphdr
        {
            Uint16  sport;
            Uint16  dport;
            Uint16  length;
            Uint16  chksum;
        };

struct  igmphdr
        {
            Uint16  type_mrt;
            Uint16   chksum;
            struct   ipaddr groupaddr;
        };
struct  pre_udphdr
        {
            struct  ipaddr saddr;
            struct  ipaddr daddr;
            Uint16  protocal_value;
            Uint16   length;
};
struct  arp
        {
            Uint16  hard_type;              //硬件类型
            Uint16  proto_type;             //协议类型
            Uint16  proto_hard_length;      //硬件及协议地址长度
            Uint16  op_code;                //操作字段
            struct  mac    send_macaddr;    //发送端以太网地址
            struct  ipaddr send_ipaddr;     //发送断IP地址
            struct  mac    rec_macaddr;     //接收端以太网地址
            struct  ipaddr rec_ipaddr;      //接收端IP地址
};

struct Socket_Type{
                    Uint16      My_Port;                //本机端口
                    Uint16      Dest_Port;              //对方端口
                    Uint16      Dest_Ip[2];             //对方ip
                    Uint16      Dest_Mac_Id[3];         //对方的以太网地址
                    Uint32      IRS;                    //初始化顺序号
                    Uint32      ISS;                    //我的初始化序列号
                    Uint32      Rcv_Next;               //对方的顺序号
                    Uint32      Send_Next;              //我的已经发送顺序号
                    Uint32      Sent_UnAck;             //我的还没有确认顺序号
                                                      //unsigned long
```

```
dest_ack_number;
                    Uint16      Rcv_Window;             //对方的window大小
                    Uint16      Snd_Window;             //我的window大小
                Uint16      Dest_Max_Seg_Size;      //对方接受的最大的数据包大小
MTU
                    Uint16      My_Max_Seg_Size;        //我能接受的最大的数据包
大小
                    Uint32      My_Wl1;     //seq
                    Uint32      My_Wl2;     //ack
                    Uint16      State;      //连接状态
                    Uint16      Open;
                 };

//========================== end ==========================
```

Dataspace.h

```
#define NX 1024
#pragma DATA_SECTION (x,".input")

DATA x[NX];
```

cpu_reg.h

```
#define int0_VAL       1<<0
#define tint_VAL       1<<3
#define rint_VAL       1<<4
#define xint_VAL       1<<5

#define SPCR10_VAL     0x0000
#define SPCR20_VAL     0x0200
//2 words per frame
#define RCR10_VAL      0x0140
#define RCR20_VAL      0x04
//2 words per frame
#define XCR10_VAL      0x0140
#define XCR20_VAL      0x04
#define PCR0_VAL       0x01


#define SRGR1_VAL2      (23<<8)+47
#define SRGR1_VAL1      (23<<8)+47

//帧长度=48、CLK由CPU的时钟驱动、由FSG驱动
```

```
#define SRGR2_VAL1      0x3000+47

//发送帧同步、时钟由内部产生（11 9位=1)
//接收帧同步、时钟由外部产生（10 8位=0)
//帧同步高有效，（2 3位=0），   下降沿有效（2 3位=1）
//时钟上升沿有效（最后2位=0），下降沿有效（最后2位=1）
//引脚配置成串口，
#define PCR_VAL1       0x0a03

//DX is on(7=True)
//RINT by RRDY
#define SPCR1_VAL1     0x0080

// 一帧2个字，每字24bit
#define XCR1_VAL1      0x0180
// 单相帧、无压缩、0-bit延迟
#define XCR2_VAL1      0x0004

// 一帧2个字，每字24bit
#define RCR1_VAL1      0x0180
// 单相帧、无压缩、0-bit延迟
#define RCR2_VAL1      0x0004


//McBSP Memory Mapped Registers
#define CLKMD         (unsigned int *)0x58
#define SWCR          (unsigned int *)0x2b
#define IMR           (unsigned int *)0x00

#define SPSA0         (unsigned int *)0x38
#define SPSD0         (unsigned int *)0x39
#define DRR20         (unsigned int *)0x20
#define DRR10         (unsigned int *)0x21
#define DXR20         (unsigned int *)0x22
#define DXR10         (unsigned int *)0x23

#define SPSA1         (unsigned int *)0x48
#define SPSD1         (unsigned int *)0x49
#define DRR21         (unsigned int *)0x40
#define DRR11         (unsigned int *)0x41
#define DXR21         (unsigned int *)0x42
#define DXR11         (unsigned int *)0x43

// McBSP Subaddresed Registers
```

```
#define SPCR1          0x00
#define SPCR2          0x01
#define RCR1           0x02
#define RCR2           0x03
#define XCR1           0x04
#define XCR2           0x05
#define SRGR1          0x06
#define SRGR2          0x07
#define PCR            0x0E

typedef unsigned    int     Uint16;
typedef unsigned    long    Uint32;
```

IP.h

```
#define     Reg00       port2000
#define     Reg01       port2001
#define     Reg02       port2002
#define     Reg03       port2003
#define     Reg04       port2004
#define     Reg05       port2005
#define     Reg06       port2006
#define     Reg07       port2007
#define     Reg08       port2008
#define     Reg09       port2009
#define     Reg0a       port200a
#define     Reg0b       port200b
#define     Reg0c       port200c
#define     Reg0d       port200d
#define     Reg0e       port200e
#define     Reg0f       port200f
#define     Reg10       port2010

ioport      Uint16      Reg00;
ioport      Uint16      Reg01;
ioport      Uint16      Reg02;
ioport      Uint16      Reg03;
ioport      Uint16      Reg04;
ioport      Uint16      Reg05;
ioport      Uint16      Reg06;
ioport      Uint16      Reg07;
ioport      Uint16      Reg08;
ioport      Uint16      Reg09;
ioport      Uint16      Reg0a;
ioport      Uint16      Reg0b;
```

```
ioport      Uint16      Reg0c;
ioport      Uint16      Reg0d;
ioport      Uint16      Reg0e;
ioport      Uint16      Reg0f;
ioport      Uint16      Reg10;

#define MY_TCP_PORT         1024
#define MY_UDP_PORT         1025

#define ETH_HEADER_START     0
#define IP_HEADER_START      7
#define ARP_HEADER_START     7
#define TCP_HEADER_START     17
#define UDP_HEADER_START     17
#define ICMP_HEADER_START    17
#define USER_DATA_START      27

#define RTL8019_HEADER_SIZE  2
#define ETH_HEADER_SIZE      7
#define IP_HEADER_SIZE       10
#define TCP_HEADER_SIZE      10
#define UDP_HEADER_SIZE      4
#define ARP_FRAME_SIZE       14
#define ICMP_HEADER_SIZE     2
#define DUMMY_HEADER_SIZE    6
#define MY_MAX_SEG_SIZE      1460

#define Frame_ARP           0x0806
#define Frame_IP            0x0800
#define Ip_Edition          0x4500              //Ip 版本和IP首部长度
#define DEFUALT_TTL         128
#define ICMP_ECHO           8
#define ICMP_ECHO_REPLY      0
//ARP

#define HARDW_ETH           1
#define IP_HLEN_PLEN        0x0604
#define OP_ARP_REQUEST       1
#define OP_ARP_ANSWER        2

#define PROTOCOL_ICMP        1
#define PROTOCOL_TCP         6
#define PROTOCOL_UDP         17
```

```
/////TCP define

#define TCP_MAX_RE_TXDNUM    8

#define TCP_CODE_FIN        0x0001
#define TCP_CODE_SYN        0x0002
#define TCP_CODE_RST        0x0004
#define TCP_CODE_PSH        0x0008
#define TCP_CODE_ACK        0x0010
#define TCP_CODE_URG        0x0020

#define     TCP_STATE_LISTEN        0
#define     TCP_STATE_SYN_RCVD      1
#define     TCP_STATE_SYN_SENT      2
#define     TCP_STATE_ESTABLISHED   3
#define     TCP_STATE_FIN_WAIT1     4
#define     TCP_STATE_FIN_WAIT2     5
#define     TCP_STATE_CLOSING       6
#define     TCP_STATE_CLOSE_WAIT    7
#define     TCP_STATE_LAST_ACK      8
#define     TCP_STATE_CLOSED        9
#define     TCP_STATE_TIME_WAIT     10

extern  void    SendFrame(Uint16   *buf,Uint16 len);
extern  Uint16  SwapWord(Uint16data);

//prototypes

void TCP_Listen(void);
void TCP_Syn_Rec(void);
void TCP_Syn_Sent(void);
void TCP_Established(void);
void TCP_Close_Wait(void);
void TCP_Last_Ack(void);
void DoNetworkStuff(void);
void TCPActiveTxData(void);
void RTL8019ActiveOpen(void);
void Prepare_ICMP_Answer(void);
void Prepare_TCP_Frame(Uint16 TCPCode);
void CopyFrameFromBE(Uint16 Offset,Uint16 Size);
void ProcessEthBroadcastFrame(void);

//========================== end ==========================
```

TCP.h

```
#define MY_TCP_PORT         1024
#define MY_UDP_PORT         1025

#define ETH_HEADER_START     0
#define IP_HEADER_START      7
#define ARP_HEADER_START     7
#define TCP_HEADER_START     17
#define UDP_HEADER_START     17
#define ICMP_HEADER_START    17
#define USER_DATA_START      27

#define RTL8019_HEADER_SIZE  2
#define ETH_HEADER_SIZE      7
#define IP_HEADER_SIZE       10
#define TCP_HEADER_SIZE      10
#define UDP_HEADER_SIZE      4
#define ARP_FRAME_SIZE       14
#define ICMP_HEADER_SIZE     2
#define DUMMY_HEADER_SIZE    6
#define MY_MAX_SEG_SIZE      1460

#define Frame_ARP           0x0806
#define Frame_IP            0x0800
#define Ip_Edition          0x4500              //Ip 版本和IP首部长度
#define DEFUALT_TTL         128
#define ICMP_ECHO           8
#define ICMP_ECHO_REPLY      0
//ARP

#define HARDW_ETH           1
#define IP_HLEN_PLEN         0x0604
#define OP_ARP_REQUEST       1
#define OP_ARP_ANSWER        2

#define PROTOCOL_ICMP        1
#define PROTOCOL_TCP         6
#define PROTOCOL_UDP         17

/////TCP define

#define TCP_MAX_RE_TXDNUM    8

#define TCP_CODE_FIN         0x0001
```

```
#define TCP_CODE_SYN        0x0002
#define TCP_CODE_RST        0x0004
#define TCP_CODE_PSH        0x0008
#define TCP_CODE_ACK        0x0010
#define TCP_CODE_URG        0x0020

#define     TCP_STATE_LISTEN       0
#define     TCP_STATE_SYN_RCVD     1
#define     TCP_STATE_SYN_SENT     2
#define     TCP_STATE_ESTABLISHED  3
#define     TCP_STATE_FIN_WAIT1    4
#define     TCP_STATE_FIN_WAIT2    5
#define     TCP_STATE_CLOSING      6
#define     TCP_STATE_CLOSE_WAIT   7
#define     TCP_STATE_LAST_ACK     8
#define     TCP_STATE_CLOSED       9
#define     TCP_STATE_TIME_WAIT    10

extern  void    SendFrame(Uint16   *buf,Uint16 len);
extern  Uint16  SwapWord(Uint16data);

//prototypes

void TCP_Listen(void);
void TCP_Syn_Rec(void);
void TCP_Syn_Sent(void);
void TCP_Established(void);
void TCP_Close_Wait(void);
void TCP_Last_Ack(void);
void DoNetworkStuff(void);
void TCPActiveTxData(void);
void UDPActiveTxData(DATA* ADbuf);
void RTL8019ActiveOpen(void);
void Prepare_ICMP_Answer(void);
void Prepare_TCP_Frame(Uint16 TCPCode);
void CopyFrameFromBE(Uint16 Offset,Uint16 Size);
void ProcessEthBroadcastFrame(void);

//========================== end ==========================
```

INRAM.cmd

```
MEMORY  {
    PAGE 0:
```

```
        RESEVE:   org   00h    len = 0x80
    PAGE 0:
        PROG1:    org = 0x0100    len = 0x4000
    PAGE 0:
        VECT:     org = 0x0080,  len = 0x80


    PAGE 1:
        RESEVE1:   org   00h   len = 0x1300
    PAGE 1:
        DARAM2:   org = 0x1300   len = 0x6000
    PAGE 1:
        DARAM1:   org = 0x7300   len = 0x8000
}

SECTIONS{
        .text :  >       PROG1   PAGE 0
        .cinit : >       PROG1   PAGE 0
        .switch: >       PROG1   PAGE 0
        .vectors:>       VECT    PAGE 0

        .sintab: >       DARAM1   PAGE 1
        .input   >       DARAM1   PAGE 1,align(2048)
        .const:  >       DARAM1   PAGE 1
        .bss  :  >       DARAM1   PAGE 1
        //.text :  >       DARAM1   PAGE 1

        .stack : >       DARAM2   PAGE 1
        .system: >       DARAM2   PAGE 1
        .data :  >       DARAM2   PAGE 1
        }
```

CVECTORS.asm

```
DRR11A  .set 0x41
DXR11A  .set 0x43

        .ref  _timer0,_codec_ch0_in
        .ref  _c_int00

        .sect ".vectors"

rs: BD _c_int00
    nop
```

```
    nop
nmi: rete       ;NMI, SINT16
    nop
    NOP
    NOP
sint17: BD _c_int00    ;SINT17
    NOP
    NOP
sint18: BD _c_int00    ;SINT18
    NOP
    NOP
sint19: BD _c_int00    ;SINT19
    NOP
    NOP
sint20: BD _c_int00    ;SINT20
    NOP
    NOP
sint21: BD _c_int00    ;SINT21
    NOP
    NOP
sint22: BD _c_int00    ;SINT22
    NOP
    NOP
sint23: BD sint23  ;SINT23
    NOP
    NOP
sint24: BD sint24  ;SINT24
    NOP
    NOP
sint25: BD sint25  ;SINT25
    NOP
    NOP
sint26: BD sint26  ;SINT26
    NOP
    NOP
sint27: BD sint27  ;SINT27
    NOP
    NOP
sint28: BD sint28  ;SINT28
    NOP
    NOP
sint29: BD sint29  ;SINT29
    NOP
    NOP
```

```
sint30: BD sint30  ;SINT30
    NOP
    NOP
int0: BD int0      ;INT0, SINT0
    NOP
    NOP
int1: BD int1      ;INT1, SINT1
    NOP
    NOP
int2: BD int2      ;INT2, SINT2
    NOP
    NOP
tint0: BD _timer0  ;TINT0, SINT3
    NOP
    NOP
brint0: BD _codec_ch0_in   ;BRINT0, SINT4
    NOP
    NOP
bxint0: BD bxint0              ;BXINT0, SINT5
    NOP
    NOP
dmac0: BD dmac0                ;DMAC0, brint2, SINT6
    NOP
    NOP
dmac1: BD _c_int00             ;DMAC1, bxint2, SINT7 ??
    NOP
    NOP
int3: BD int3                  ;INT3, SINT8
    NOP
    NOP
hpint: BD _c_int00             ;HPINT, SINT9
    NOP
    NOP
brint1: BD brint1              ;BRINT1 or DMAC2, SINT10
    NOP
    NOP
;bxint1: BD mcbsp1             ;BXINT1 or DMAC3, SINT11
bxint1: BD _c_int00                ;BXINT1 or DMAC3, SINT11
    NOP
    NOP
dmac4: BD dmac4                ;DMAC4, SINT12
    NOP
    NOP
dmac5: BD dmac5                ;DMAC5, SINT13
```

```
    NOP
    NOP
rsvd1: BD rsvd1                ;reserved
    NOP
    NOP
rsvd2: BD rsvd2                ;reserved
    NOP
    NOP


    .end
```