

第1章 DSP 应用系统开发实例

1.1 DSP 实现音频滤波系统

音频信号是带有语音、音乐和音效的有规律的声波的频率、幅度变化信息载体，它存在于生活中的各种角落，如人们的对话、日常听的音乐等。

音频信号处理是信号处理的重要组成部分，主要用来实现音量调节、延时、回声、回响、均衡、去噪等处理过程，对音频进行数字信号处理的流程同常规的数字信号处理相同，包括音频采样、音频处理、和音频输出三个过程。

音频信号处理系统是常见的信号处理系统，一般由节目源、信号处理系统、扬声器系统等部分组成，节目源即音频信号的输入源，目前较多使用计算机和手机，信号处理系统包括解码器、放大器、压缩器、限制器等，扬声器是音频输出的最后一个环节，常见的扬声器有音响、喇叭等。

随着科技的进步，模拟的音频信号处理已经不能满足需求，数字化的音频处理方式成了音频信号的发展趋势，数字化的音频处理是利用数字滤波算法对采集到的信号进行变换处理。

DSP 实现的音频滤波系统主要由语音解码芯片 TLV320AIC23 和 TMS320C5509 DSP 组成，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。其 ADC 和 DAC 集成在芯片内部，可以在 8kHz 至 96kHz 的采样率下提供 16bit、20bit、24bit 和 32bit 的采样数据，输入和输出都具有可编程的增益调节功能。DSP 使用 I²C 对编解码芯片进行配置，通过 McBSP 与 TLV320AIC23 进行通信。

McBSP 是 TI 公司生产的数字信号处理芯片的多通道缓冲串行口。McBSP 是在标准串行接口的基础之上对功能进行扩展,因此,具有与标准串行接口相同的基本功能。此外 McBSP 还支持多通道发送和接收，每个串行口最多支持 128 通道，其串行字长度可选，包括 8、12、16、20、24 和 32 位，另外它还支持 μ -Law 和 A-Law 数据压缩扩展；

TMS320C5509 可以方便的通过其自带的多通道缓冲串行口 McBSP 与语音编解码芯片进行通信，其接口电路如图 6.1 所示。

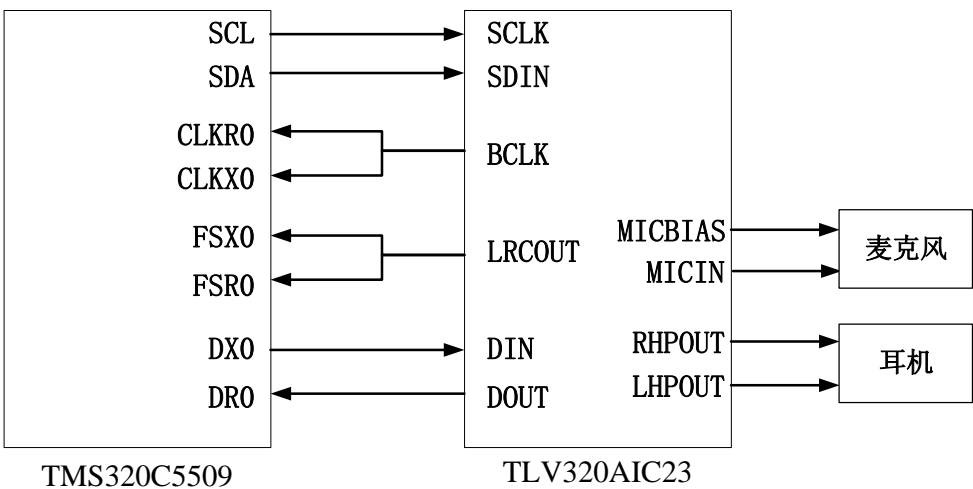


图 6.1 TMS320C5509 与 TLV320AIC23 接口图

编解码芯片 TLV320AIC23 管脚说明如下：

SCLK：串口配置时钟。

SDIN：串口数据输入。

BLK: 数字音频接口时钟信号, 当编解码芯片处于从模式时, 此时钟由 DSP 产生, 本实验中令编解码芯片处于主模式, 时钟由 AIC23 提供。

LRCOUT: 数据口 ADC 输出的帧同步信号

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

TMS320C5509 引脚功能如下: 数据发送引脚 DX 负责数据的发送, 数据接收引脚 DR 负责数据的接收, 发送时钟引脚 CLKX、接收时钟引脚 CLKR、发送帧同步引脚 FSX 和接收帧同步引脚 FSR 提供串行时钟和控制信号。

DSP 通过 I²C 总线完成对编解码芯片 AIC23 的配置, AIC23 在完成语音信号的采样后通过 McBSP 接口发送到 DSP 上。输出同理, DSP 将语音数据通过 McBSP 发送到 AIC23, AIC23 将其转换成模拟信号通过输出设备输出。

1.1.1 采集语音

语音的采集通过编解码芯片 TLV320AIC23 来实现, AIC23 芯片内部集成了数模转换器 ADC 以及模数转换器 DAC, 采样率可以达到 96kHz。DSP 通过 I²C 总线完成对 AIC23 芯片采样以及数据传输的设置。

首先对 DSP 的 McBSP 进行配置, 配置过程如第四章相同, 在此不再赘述, AIC23 通过 McBSP 与 DSP 进行通信, 一次传输位数为 16 位, 将数据存进数组中。

之后对 AIC23 进行控制, AIC23 的控制字长度为 16 位, 其中低 9 位为写入寄存器的数值, 高 7 位为该寄存器的地址。I²C 模块主要由双向的串行数据信号 SDA 以及串行时钟信号 SCL 组成。在配置好 I²C 接口后, 分别配置 AIC23 的数字接口、模拟通路、数字通路、采样率、耳机音量、节点模式等参数。

表 6-1 详细说明 TLV320AIC23 寄存器含义及地址。

表 6-1 TLV320AIC23 寄存器映射地址及含义

地址	寄存器
0000000	左声道输入音量控制
0000001	右声道输入音量控制
0000010	左声道输出音量控制
0000011	右声道输出音量控制
0000100	模拟音频通道控制
0000101	数字音频通道控制
0000110	节电模式控制
0000111	数字音频接口控制
0001000	采样率控制
0001001	数字音频激活开关
0001111	初始化寄存器

寄存器的详细说明如下:

表 6-2 左声道输入音量控制 (地址: 0x0000000)

位数	8	7	6	5	4	3	2	1	0
功能	LRS	LIM	X	X	LIV4	LIV3	LIV2	LIV1	LIV0
缺省	0	1	0	0	1	0	1	1	1

LRS: 左右声道同时更新 0=禁止 1=使能
LIM: 左声道输入衰减 0=正常 1=衰减
LIV[4: 0]: 左声道输入控制衰减 (10111 = 0dB 缺省)

ADCHP: ADC 滤波器 0=禁止 1=开启
X: 保留

表 6-8 节电模式控制（地址：0x0000110）

位数	8	7	6	5	4	3	2	1	0
功能	X	OFF	CLK	OSC	OUT	DAC	ADC	MIC	LINE
缺省	0	0	0	0	0	0	1	1	1

OFF 设备电源 0=开 1=关
CLK 时钟 0=开 1=关
OSC 振荡器 0=开 1=关
OUT 输出 0=开 1=关
DAC DAC 0=开 1=关
ADC ADC 0=开 1=关
MIC 麦克风输入 0=开 1=关
LINE line 输入 0=开 1=关
X: 保留

表 6-9 数字音频接口格式控制（地址：0x0000111）

位数	8	7	6	5	4	3	2	1	0
功能	X	X	MS	LRSWA	LRP	IWL1	IWL0	FOR1	FOR0
缺省	0	0	0	0	0	0	0	0	1

MS 主从模式 0=从模式 1=主模式
LRSWAP: DAC 左右通道交换 0=禁止 1=开启
LRP: DAC 左右通道设定 0=右通道在 LRCIN 高电平
 1=左通道在 LRCIN 低电平
IWL[1:0]: 输入长度 00=16bit 01=20bit 10=24bit 11=32bit
FOR[1:0]: 数据初始化 11=DSP 初始化，帧同步来自两个字
 10=I²S 初始化
 01=MSP 优先，左对齐
 00=MSP 优先，右对齐

X: 保留

表 6-10 采样率控制（地址：0x0001000）

位数	8	7	6	5	4	3	2	1	0
功能	X	CLKOUT	CLKIN	SR3	SR2	SR1	SR0	BOSR	USB/normal
缺省	0	0	0	1	0	0	0	0	0

CLKOUT: 输出时钟分频 0=MCLK1=MCLK/2
CLKIN: 输入时钟分频 0=MCLK1=MCLK/2
SR[3: 0] 样本速度控制
BOSR 基本速度比
 USB 模式: 0=250fs 1=272fs
 普通模式: 0=256fs 1=384fs
USB/normal 时钟模式选择 0=normal 1=USB

X: 保留

此外向数据接口激活寄存器（0x0001001）最低位写 1 即可激活接口，向 RESET 寄存器（0x0001111）写入全 0 即可重置芯片。

对寄存器设置程序如下所示：

```

// 数字接口格式设置为主模式，DSP 模式，数据位数为 16 位
Uint16 Digital_Audio_Interface_Format[2]={
    Codec_DAIF_REV,
    DAIF_MS(1)+DAIF_LRSWAP(0)+DAIF_LRP(1)+DAIF_IWL(0)+DAIF_FOR(3));
// AIC23 的波特率设置,采样率为 48k
Uint16 Sample_Rate_Control[2] = {
    Codec_SRC_REV,
    SRC_CLKIN(0)+SRC_CLKOUT(0)+SRC_SR(6)+SRC_BOSR(0)+SRC_USB(0));
// AIC23 寄存器复位
Uint16 Reset[2]={
    Codec_RST_REV,
    RST_RES};
// AIC23 节电方式设置为默认
Uint16 Power_Down_Control[2]={
    Codec_PDC_REV,
    PDC_DEFAULT};
// AIC23 模拟音频的控制:关掉侧音
// DAC 使能,ADC 输入选择为音频输入
Uint16 Analog_Audio_Path_Control[2] = {
    Codec_AAPC_STA2(0),
    AAPC_STA10(0)+AAPC_STE(0)+AAPC_DAC(1)+AAPC_BYP(0)+AAPC_INSEL(1)
+AAPC_MICM(0)+AAPC_MICB(0));
// AIC23 数字音频通路的控制
// 使能 ADC 高通滤波
Uint16 Digital_Audio_Path_Control[2] = {
    Codec_DAPC_REV,
    DAPC_DACM(0)+DAPC_DEEMP(0)+DAPC_ADCHP(1));
// AIC23 数字接口的使能
Uint16 Digital_Interface_Activation[2] = {
    Codec_DIA_REV,
    DIA_ACT(1)};
// AIC23 左通路音频调节
Uint16 Left_Line_Input_Volume_Control[2] = {
    Codec_LLIVC_LPS(1),
    LLIVC_LIM(0)+LLIVC_LIV(23));
// AIC23 右通路音频调节
Uint16 Right_Line_Input_Volume_Control[2] = {
    Codec_RLIVC_RLS(1),
    RLIVC_RIM(0)+RLIVC_RIV(23));
// AIC23 耳机左通路音频调节
Uint16 Left_Headphone_Volume_Control[2] = {
    Codec_LHPVC_LRS(1),
    LHPVC_LZC(1)+LHPVC_LHV(127));
// AIC23 耳机右通路音频调节

```

```
Uint16 Right_Headphone_Volume_Control[2] = {  
    Codec_RHPVC_RLS(1),  
    LHPVC_RZC(1)+LHPVC_RHV(127)};
```

1.1.2 时域滤波

滤波即有选择的提取或者去掉信号频谱中的一段频率或几段频率，数字滤波器是利用数字器件对信号实行滤波的数字系统，常用 DSP、FPGA 等可编程芯片进行编程实现，数字滤波器按照单位冲激响应的时域特性可以分为无限冲激响应(IIR)滤波器和有限冲激响应(FIR)滤波器。本实验利用 FIR 滤波器对在时域上输入音频信号进行低通滤波。

FIR 数字滤波器是一种非递归系统，其冲击响应 $h(n)$ 是有限长序列，因此可以在时域上直接对信号进行卷积操作达到滤波的目的，其差分方程表达式为：

$$y(n) = \sum_{i=0}^{N-1} h(i)x(n-i)$$

N 为 FIR 滤波器的阶数。

在数字信号处理应用中往往需要设计线性相位的滤波器，FIR 滤波器在保证幅度特性满足技术要求的同时，很容易做到严格的线性相位特性。为了使滤波器满足线性相位条件，要求其单位脉冲响应 $h(n)$ 为实序列，且满足偶对称或奇对称条件，即 $h(n) = h(N-1-n)$ 或 $h(n) = -h(N-1-n)$ 。这样，当 N 为偶数时，偶对称线性相位 FIR 滤波器的差分方程表达式为

$$h(n) = h(N-1-n) \text{ 或 } h(n) = -h(N-1-n)$$

由上可见，FIR 滤波器不断地对输入样本 $x(n)$ 延时后，再做乘法累加运算，将滤波器结果 $y(n)$ 输出。因此，FIR 实际上是一种乘法累加运算。而对于线性相位 FIR 而言，利用线性相位 FIR 滤波器系数的对称特性，可以采用结构精简的 FIR 结构将乘法器数目减少一半。

一般我们采用 MATLAB 的 FDATOOL 工具设计 FIR 滤波器。FDATool 是 Matlab 数字信号处理工具箱中一种图形化的滤波器设计与分析工具，使用该工具可以快速设计各种类型的滤波器，并计算出滤波器系数，还可以画出滤波器的幅度、相位响应、群/相位延迟和零极点分布图。

在 MATLAB 中设计 FIR 滤波器步骤如下：

- (1) 在 MATLAB 命令行中输入 `fdatool`，弹出如图 6.2 所示窗口：

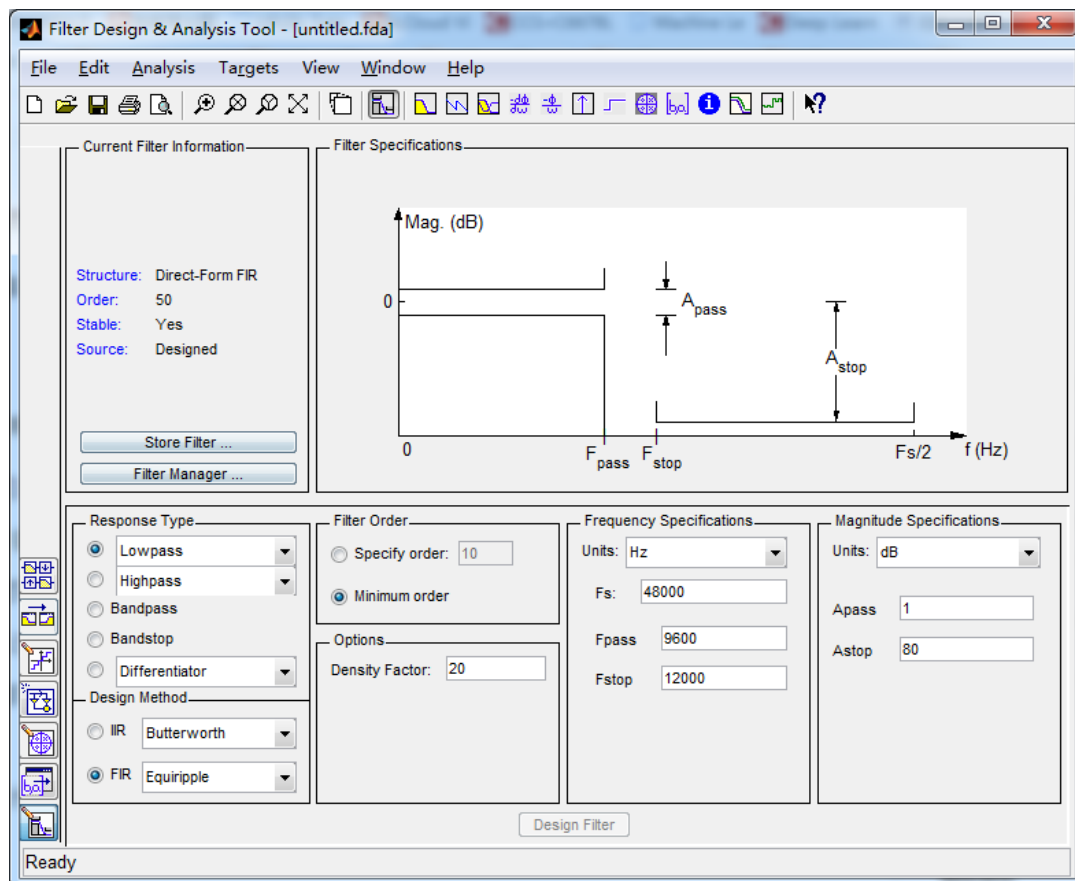


图 6.2 Fdatool 界面

- (2) 设置滤波器为低通滤波器，FIR 类型，最小阶数为 100 阶，采样频率为 48000Hz，考虑到声音信号的带宽，取通带为 600Hz，阻带为 1000Hz。生成的频率响应图如图 6.3 所示：

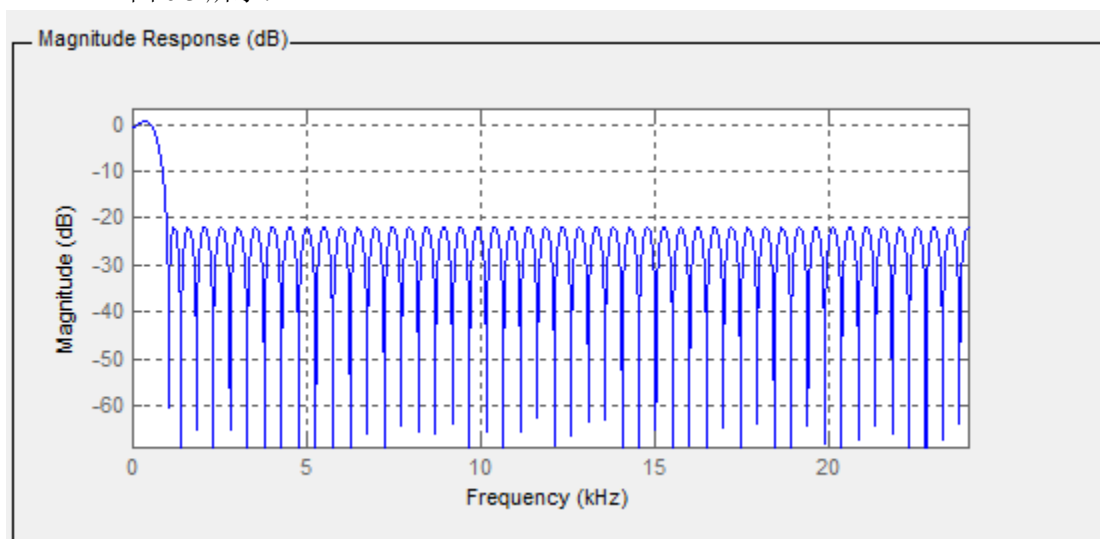


图 6.3 FIR 滤波器频率响应

- (3) 将滤波器参数导出。单击 Target 选项，选择 Generate C Header，弹出窗口如下图所示，选择输出格式为 32bit 浮点类型，单击 Generate，更改.h 文件名称和路径，生成.h 文件。

在 MATLAB 完成 FIR 滤波器的设计后，下面对硬件程序进行编写。首先在 CCS 新建

并配置工程步骤如下：

- (1) 新建工程，选项设置如图 6.3 所示：

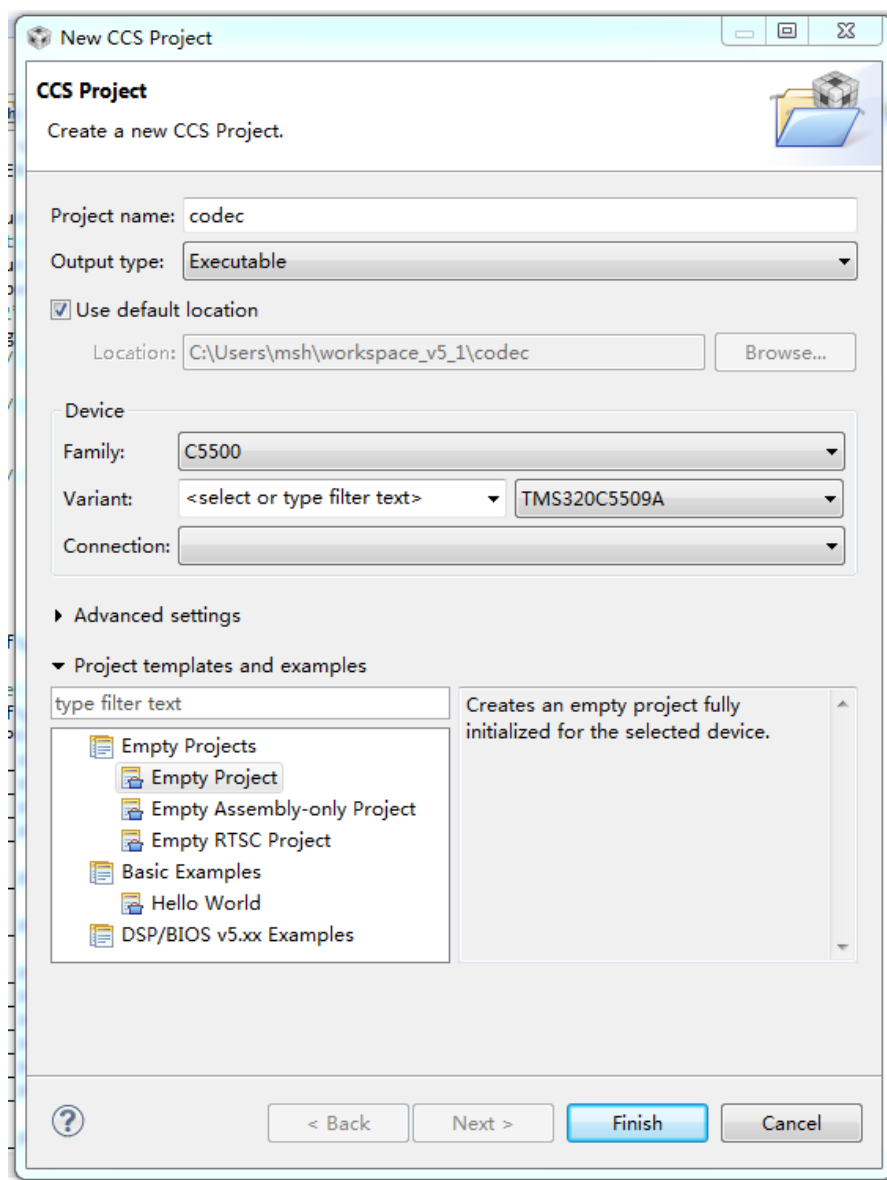


图 6.4 新建工程界面设置

- (2) 由于该工程中使用到了 CSL 库，因此需要添加工程头文件路径和库函数路径，首先右击当前工程，选择 properties 选项，在 bulid 选项卡中 C5500 Compiler 的 Include Options 添加 CSL 库的头文件路径，具体设置如图 6.5 所示

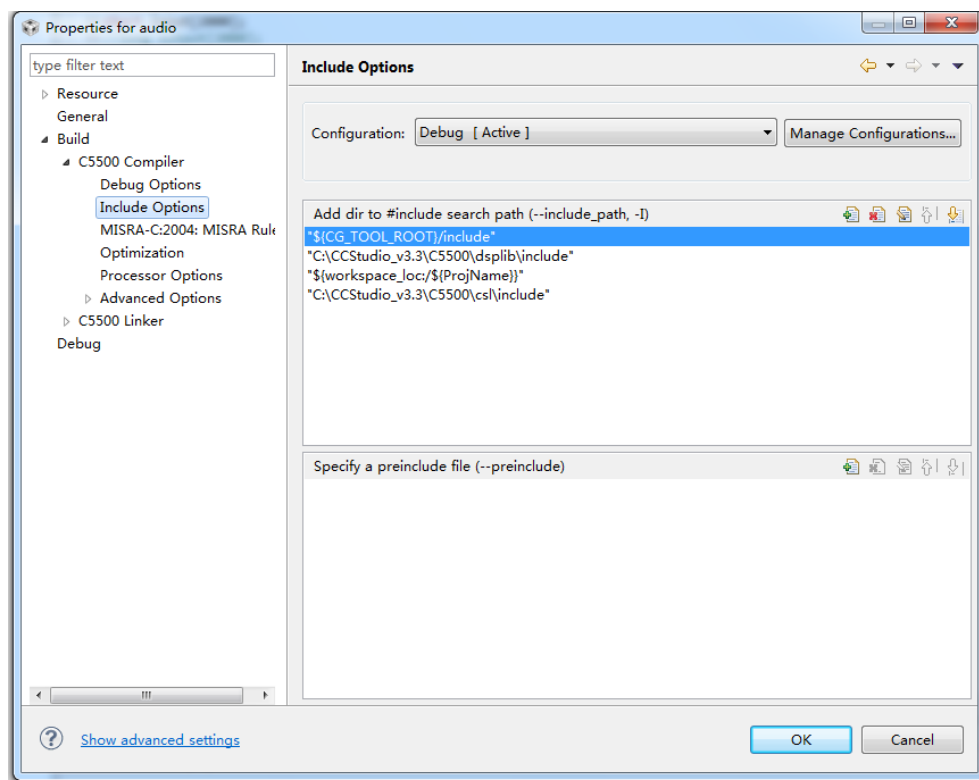


图 6.5 设置头文件包含路径

在 C5500 Linker 中的 File search path 中加入 csl5509ax.lib。路径如图 6.6 所示：

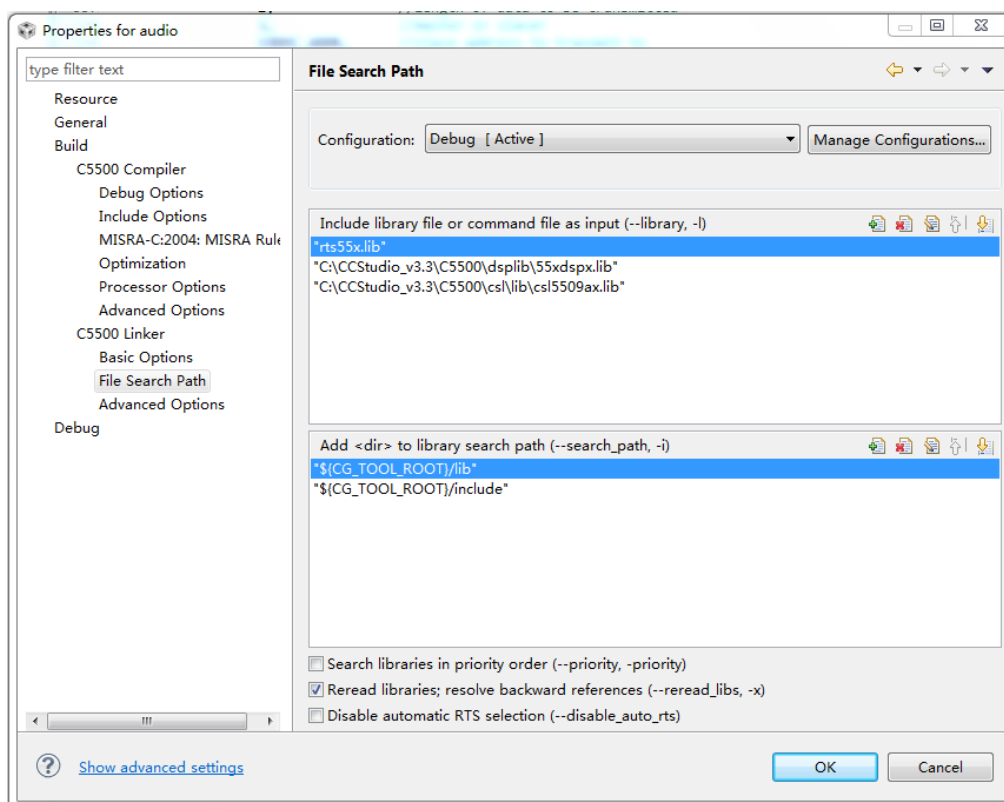


图 6.6 库函数引用路径设置

(4) 在完成头文件路径和库的添加操作之后，为工程添加配置文件。右键工程，选择 NEW 选项卡里的 Target Configuration File，配置如图 6.7 所示：

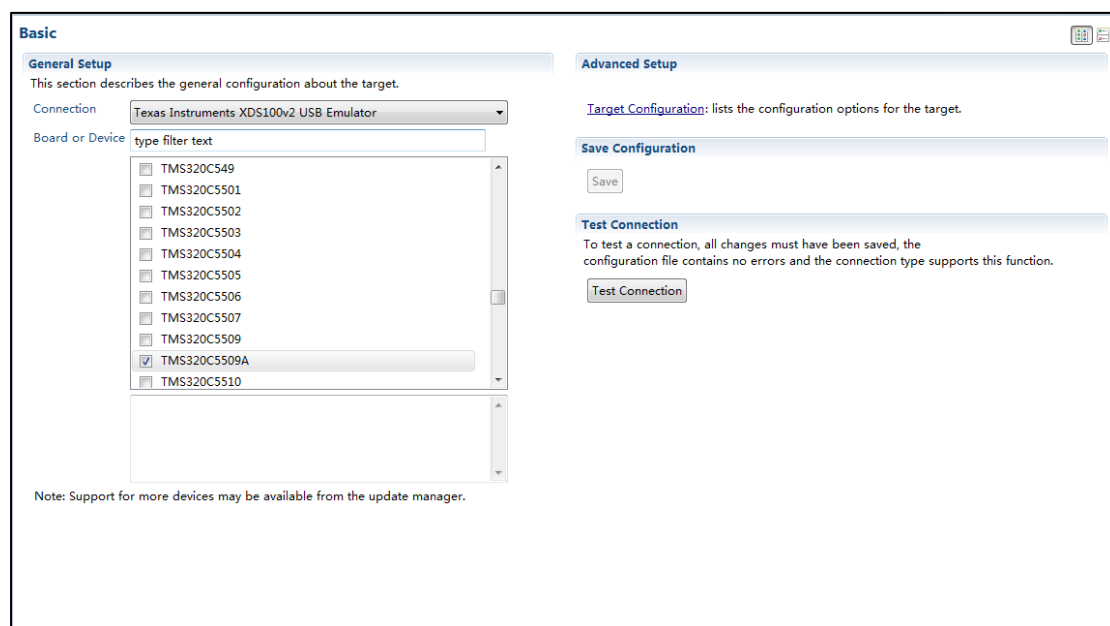


图 6.7 配置文件设置

(5) 添加 CMD 文件，将写好的 CMD 文件通过 Add File 加入工程
在配置完工程之后，进行时域滤波程序的编写，时域滤波程序流程如图 6.8:

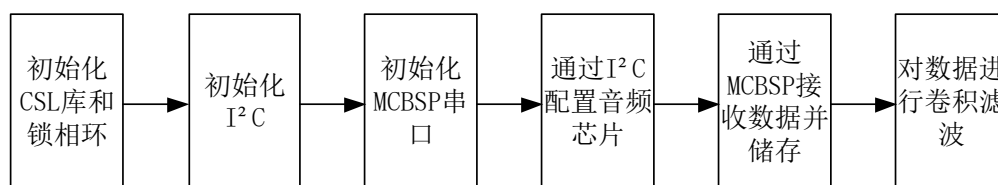




图 6.8 时域滤波程序流程

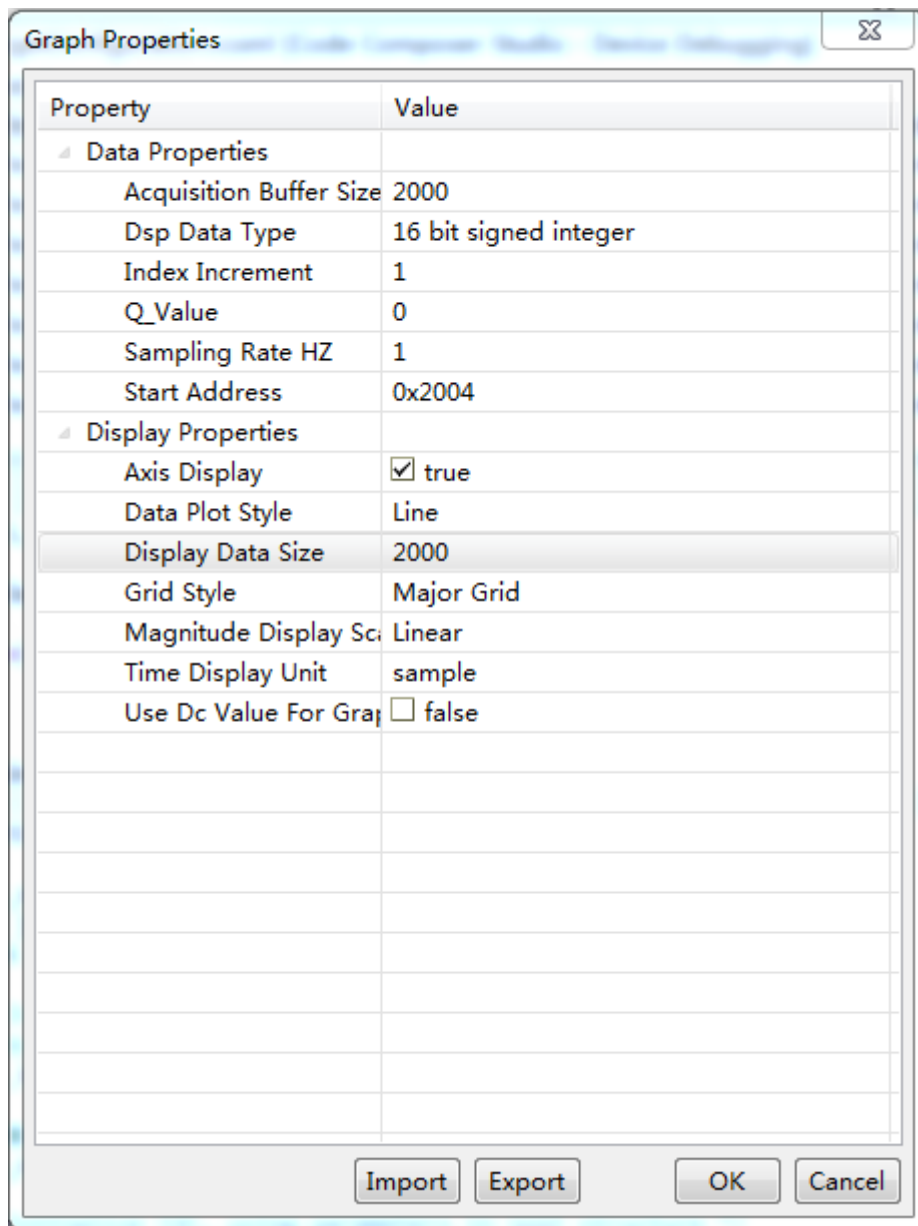
I²C 总线和 McBsp 串口的介绍以及配置方法见第四章内容，在此不过多赘述，FIR 滤波过程为卷积过程，通过两个 FOR 循环控制进行乘加操作即可完成。主要核心程序代码如下所示：

```
for (j = 0; j < 2000; j++)
{
    yn = 0;
    for (i = 0; i < N; i++)
    {
        yn += B[i] * (float)input[j+i]; // 强制类型转换，否则会产生溢出错误
    }
    output[j] = yn;
}
```

最后对 TMS320C5509 板进行硬件仿真并观察信号波形，步骤如下：

- (1) 连接 TMS320C5509 开发板电源线、下载线以及语音接口。
- (2) 编译工程并单击  进入调试界面。
- (3) 在主函数末尾括号前双击加入断点，单击 DUBUG 栏中运行按钮  运行程序，使程序执行之至函数结束。

- (4) 观察输入输出波形。单击 TOOL—>Graph—>Single Time。弹出窗口如下，输入信号（即采集到的语音信号）选择 16 位有符号数，起始地址通过 Expression 观察填入，数据长度为 2000 个数，单击 OK,如图 6.9 所示：



The image shows a 'Graph Properties' dialog box with a table of settings. The 'Data Properties' section includes 'Acquisition Buffer Size' (2000), 'Dsp Data Type' (16 bit signed integer), 'Index Increment' (1), 'Q_Value' (0), 'Sampling Rate HZ' (1), and 'Start Address' (0x2004). The 'Display Properties' section includes 'Axis Display' (checked), 'Data Plot Style' (Line), 'Display Data Size' (2000), 'Grid Style' (Major Grid), 'Magnitude Display Scale' (Linear), 'Time Display Unit' (sample), and 'Use Dc Value For Graph' (unchecked). At the bottom are 'Import', 'Export', 'OK', and 'Cancel' buttons.

Property	Value
Data Properties	
Acquisition Buffer Size	2000
Dsp Data Type	16 bit signed integer
Index Increment	1
Q_Value	0
Sampling Rate HZ	1
Start Address	0x2004
Display Properties	
Axis Display	<input checked="" type="checkbox"/> true
Data Plot Style	Line
Display Data Size	2000
Grid Style	Major Grid
Magnitude Display Scale	Linear
Time Display Unit	sample
Use Dc Value For Graph	<input type="checkbox"/> false

图 6.9 波形绘制设置

得到输入的波形如下图所示：

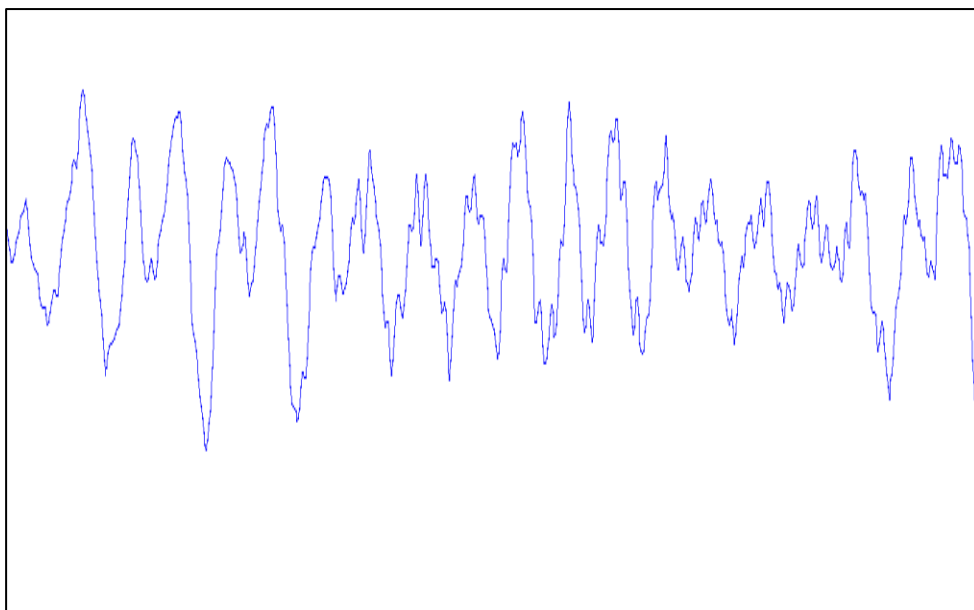


图 6.10 采集到的音频波形

- (5) 使用同样的方法对输出波形进行显示，得到 FIR 滤波后波形如图 6.11 所示：

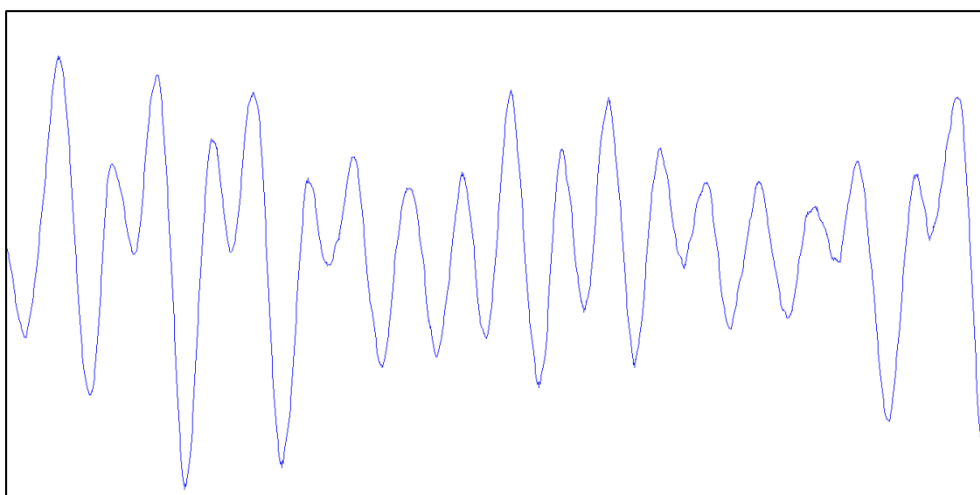
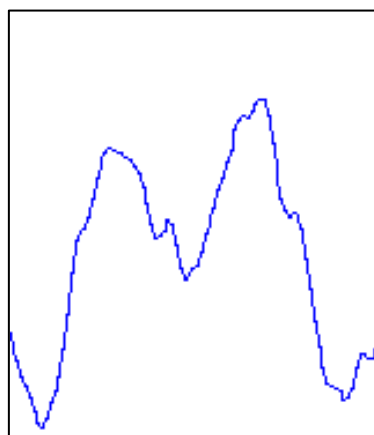
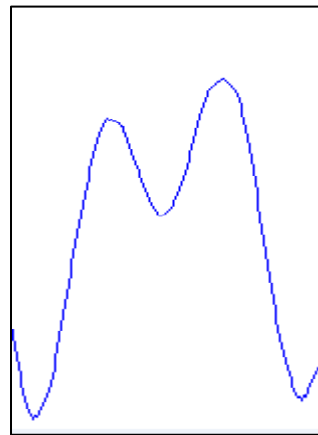


图 6.11 滤波后的音频波形

- (6) 放大对比滤波前后的细节，从图 6.12 可以看到，滤波后的信号整体形状没有变化，但是高频的部分被滤除，信号变得更加平滑，从而达到低通滤波的目的。



(a) 滤波前信号细节图



(b) 滤波后信号细节图

图 6.12 滤波前后音频波形对比

1.1.3 频域滤波

同时域滤波直接对信号进行卷积操作不同，频域滤波是在频域对信号的频谱进行加窗达到滤波的功能。

傅里叶变换是数字信号处理的常用工具，傅里叶变换将信号的时域和频域联系在一起，通过对信号频域进行有目的的选取可以达到滤波的目的。本例中通过给频域加矩形窗即去除所需频率之外频率分量来起到低通滤波的效果，频域滤波流程如图 6.13 所示：

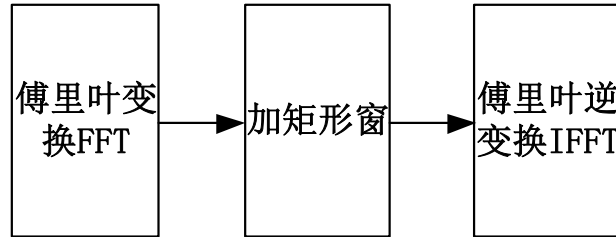


图 6.13 频域滤波流程图

在频域滤波中，傅里叶变换起到非常重要的作用，其算法如下：

FFT 是 DFT 的一种快速算法，将 DFT 的 N^2 次运算量减少为 $(N/2) \log_2 N$ 次，极大的提高了运算速度，FFT 就是利用了旋转因子的对称性和周期性来减少运算量的。FFT 算法将长序列的 DFT 分解为短序列的 DFT。N 点的 DFT 先分解为两个 N/2 点的 DFT，每个 N/2 点的 DFT 又分解为两个 N/4 点的 DFT 等等。最小变换的点数即基数，基数为 2 的 FFT 算法的最小变换时 2 点 DFT。

一般而言，FFT 算法分为时间抽取（DIT）FFT 和频率抽取（DIF）FFT 两大类。时间抽取 FFT 算法的特点是每一级处理都是在时域里把输入序列依次按奇/偶一分为二分解成较短的序列；频率抽取 FFT 算法的特点是在频域里把序列依次按奇/偶一分为二分解成较短的序列来计算。DIT 和 DIF 两种 FFT 算法的区别是旋转因子 W_N^k 出现的位置不同。DIT FFT 中旋转因子 W_N^k 在输入端，DIF FFT 中旋转因子 W_N^k 在输出端，除此之外，两种算法是一样的。在本设计中实现的是基 2 的时间抽取 FFT 算法。

时间抽取 FFT 是将 N 点输入序列按照偶数和奇数分解为偶序列和奇序列两个序列：偶序列：x(0),x(2),x(4),……,x(N-2)，奇序列：x(1),x(3),x(5),……,x(N-1)。

因此，x(n)的 N 点 FFT 可表示为：

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n)W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)W_N^{(2n+1)k} = Y(k) + W_N^k Z(k)$$

上式中，Y(k),Z(k)分别是一个 N/2 点的 DFT。以同样的方式进一步抽取，就可以得到 N/4 点的 DFT，重复这个抽取过程就可以使 N 点的 DFT 用一组 2 点的 DFT。

因为 FFT 程序较为复杂，可读性较差，因此采用直接调用 DSPLIB 中的库函数实现傅里叶变换以及逆变换，DSPLIB 是德州仪器公司为方便用户使用 DSP 芯片推出一系列的库函数中的数字信号处理库，使用这个库函数，可以更方便的实现 FIR、FFT、IFFT 等诸多功能，且库函数本身运算速度快，可移植性好。

使用库函数完成频域滤波的程序如下：

```
//进行 512 点的复数 FFT
cfft_SCALE(input,512);
cbrev(input,input,512);
```

```
//加矩形窗滤波
for(i=20;i<980;i++)
{
    input[i]=0;
}
//进行 512 点的复数 IFFT
cifft_NOSCALE(input,512);
cbrev(input,input,512);
for(i=0;i<1000;i++)
{
    output_s[i]=input[2*i];
}
```

利用 CCS 编写频域滤波的过程和时域滤波基本相同，不同的是只需在 Include Path 和 Link File 里加入 DSPlib 的相关路径和文件即可。滤波结果如下：

滤波前采样语音信号波形：

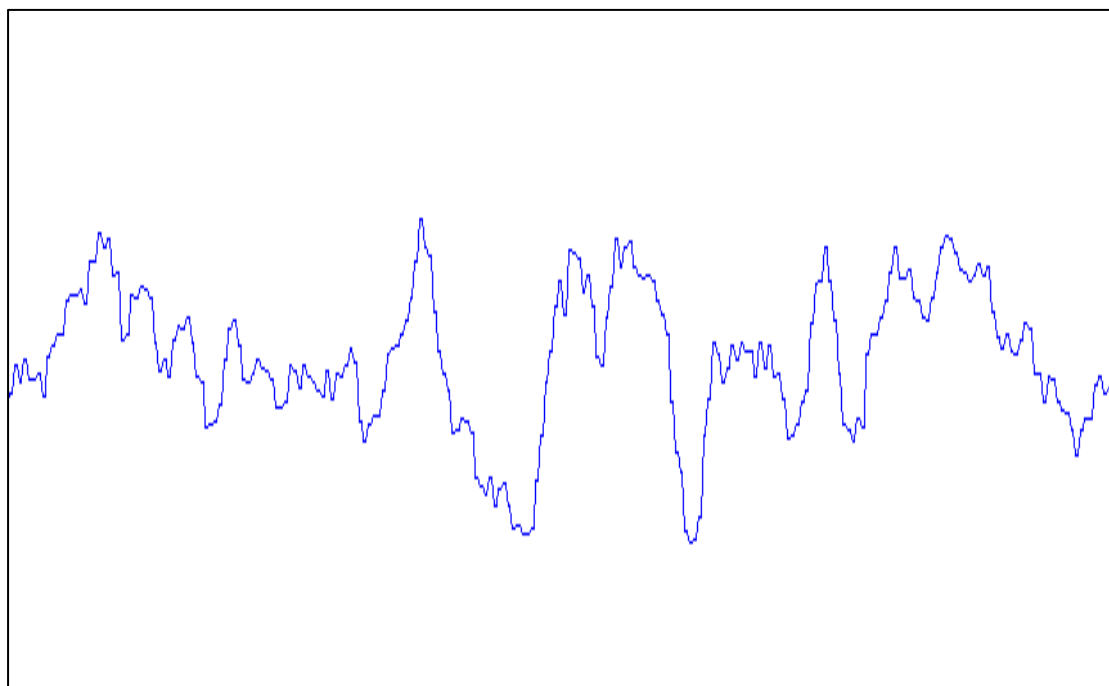


图 6.14 采集到的音频波形

滤波后采样信号波形：

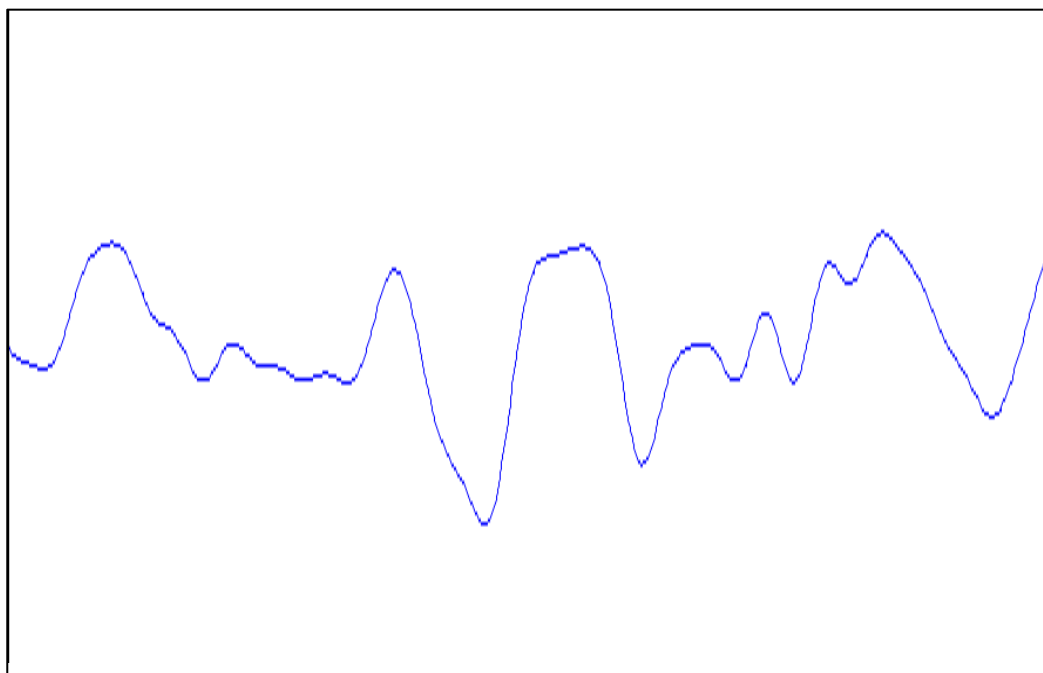


图 6.15 滤波后的音频波形

根据时域波形可以看出，信号形状相似，但是滤波后的信号更加圆滑，损失了很多高频细节，实现了低通滤波的功能。

除了对语音信号的采集和滤波处理，TMS320C5509DSP 同时也可以控制 TLV320AIC23 芯片对处理后的语音数据利用耳机进行播放。形成语音采集播放系统。

1.1.4 播放语音

语音采集过程在上文已经详细介绍过，主要过程为配置 McBsp，配置 I²C 总线，通过 I²C 总线对 TLV320AIC23 芯片进行配置，最后利用 McBsp 串口读取音频数据。语音播放的配置同语音采集基本相同。

首先对 McBsp 进行配置，将 McBsp 设置工作在 SPI 模式下，同时作为从设备连接 TLV320AIC23 芯片，即由音频芯片 AIC23 提供时钟，将发送数据的长度定为 32bit，即左右声道各 16bit。

其次是 I²C 的设置，将 I²C 设置为 7 位地址模式，并设置时钟，最后通过 I²C_write 函数对 AIC23 进行配置，需要配置的寄存器有：左声道输出音量控制、右声道输出音量控制、模拟音频通道控制、数字音频通道控制、节电模式控制、数字音频接口格式控制、采样率控制、数字音频激活开关、初始化寄存器。配置参数见上一节内容。

在进行了 Msbsp 配置和 AIC23 芯片的配置之后，语音数据的播放变得比较简单。通过函数 McBsp_write16(hMcbsp)将存入数组的 16 位数据通过 McBsp 串口通过 DX 线发送给 AIC23 芯片，AIC 芯片通过其内置的 DAC 将其转化为模拟信号通过 LHPOUT 输出到耳机输出。

输出主要程序如下：

```
while(i<length)
{
    /* 左声道耳机输出 */
    while(!MCBSP_xrdy(hMcbsp)) {};
    MCBSP_write16(hMcbsp,output[i]);
}
```

```
/* 右声道耳机输出 */
while(!MCBSP_xrdy(hMcbbsp)) {};
MCBSP_write16(hMcbbsp,output[i+1]);
i=i+2;
};
```

当 MCBSP 的准备发送状态寄存器 XRDY 置一时,证明 MCBSP 串口可以发送下一个数据,在其为 0 时使程序一直工作在等待状态。分别将左右声道的两个 16bit 音频数据通过 MCBSP 发送给 AIC23 芯片。

通过实验,我们可以从耳机中听到存储在数组或者是存储器中的音频信号转化为的声音,可以对比处理前以及处理后的音频信号,可以明显的感觉到经过低通滤波的信号更加的低沉,对于交响乐等带宽较宽的声音信号,会损失一部分乐器的高频分量。

上述为音频信号的采集、滤波及播放的过程,主要介绍了 TMS320C5509 与芯片 TLV320AIC23 的接口、MsBsp 以及利用 I²C 对 AIC23 芯片的配置。语音信号处理部分讲了在时域滤波以及频域滤波的原理及过程,音频信号的处理方式多种多样,读者可以根据需求自行学习。