实验一、IIS 音频接口实验

1 实验目的

（1） 学习 S3C44B0X 内置 IIS 总线接口的相关知识和应用；

（2） 学习 UDA1341TS 音频解码芯片的功能特性与应用。

本实验是 IIS 接口应用基础实验，为下一个实验 DMA 模式下 IIS 总线接口数据传输与音乐播放作准备工作。

2 实验要求

（1） 连接核心模块与 UDA1341TS 解码模块（IIS-BUS Interface AudioCODEC）之间的电路；

（2） 编写程序实现对 UDA1341TS 的初始化、对 S3C44B0XIIS 接口初始化，对 UDA1341TS 进行设置读写功能。

3 实验分析

3．1 关于 IIS 总线接口

很多的数字音频系统进入了音频消费市场，包括音频压缩唱片，数字音频磁带，数字声音处理器，和数字声音 TV。S3C44B0X 的 IIS（内部声音集成电路）总线接口可以用来实现对外部 8/16 位立体声音频数字信号编解码器电路的接口功能，从而实现迷你型放音机和其它便携式的应用。它支持 IIS 总线数据格式和 MSB-justified 数据格式。IIS 总线接口为 FIFO 操作提供 DMA 传输模式，代替中断模式，它可以同时传送或接收数据。

特性：

——兼容 IIS，MSB-justified 格式数据

——每通道 8/16 位数据

——每通道 16，32，48fs（采样频率）串行位时钟

——256，384fs 主设备采样时钟频率

——可编程的分频器提供给主设备时钟和编解码时钟

——供给发送和接收用的 32 字节（2×16）的 FIFO

——普通传输模式和 DMA 传输模式

3．1．1 传输模式

这里我们主要讨论普通传输模式，在下一章 DMA 传输应用实验中将重点讨论 DMA 传输模式。

IIS 控制寄存器中有一个 FIFO 准备好标志位用于 FIFO 发送和接收。当 FIFO 准备好发送数据，如果发送 FIFO 中不为空，FIFO 准备好标志将被设置为 1。

如果发送 FIFO 为空，FIFO 准备好标志将被置 0，当接收 FIFO 装满，接收 FIFO 准备好标志位被设置为 0，这些标志可以决定 CPU 读写 FIFO 的时机。串行数据就通过这种方式被发送或者接收的。

3．1．2 音频串行接口格式

IIS 总线格式

IIS 总线具有 4 根信号线，包括串行数据输入（IISDI），串行数据输出（IISDO），左/右声道选择（IISLRCK），和串行数据时钟（IISCLK）；产生 IISLRCK 和 IISCLK 的是主设备。

串行数据总是以偶数个数据（为奇数时填充）且高位在先（MSB）发送。高位在先是因为发送器和接收器可能具有不同的字长。发送器没有必要了解接收器能够处理多少位数据，接收器也不需要了解多少位的数据正在被发送。

被发送器发出的串行数据可以依据时钟信号的下降沿或者上升沿来同步。但是，串行数据必须在上升沿处锁入接收器。左右声道选择线决定被传输的通道。IISLRCK 可以在下降沿或者上升沿处改变，它并不要求是均匀的。在从设备端，这个信号在上升沿处被锁定。IISLRCK 信号线改变到 MSB 发送之间有一个时钟的周期的时间。

MSB-Justified 格式

MSB-Justified 格式与 IIS 格式有相同的信号线，唯一的不同是，IISLRCK 信号线改变后，MSB 立即发送，期间没有一个时钟的周期的时间。

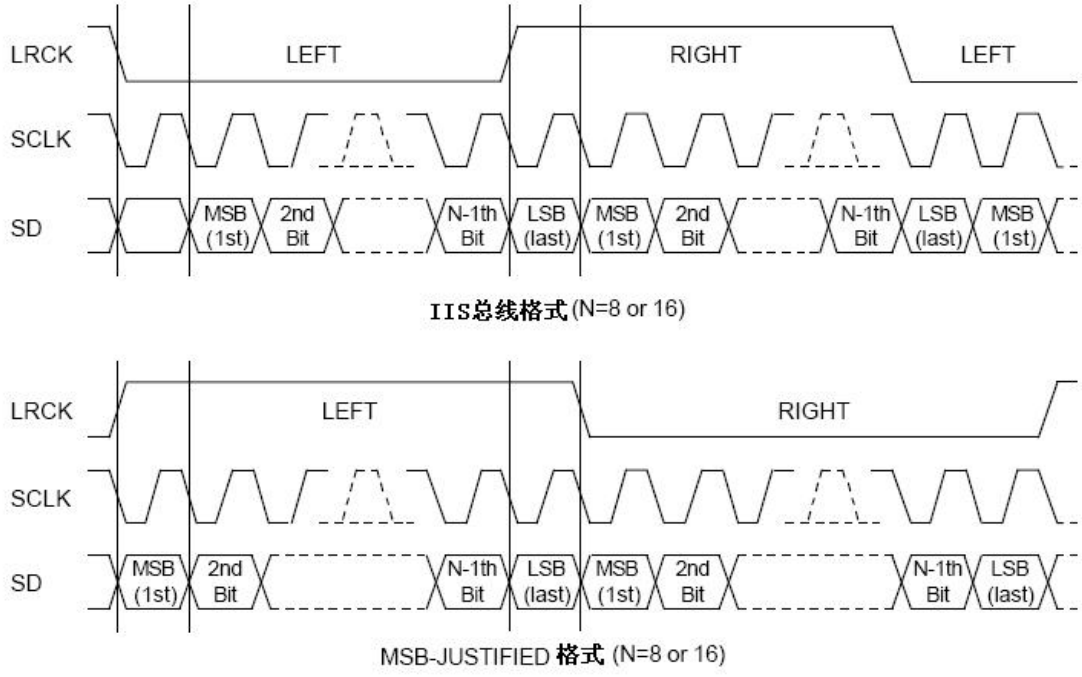

图 28－1 IIS 总线接口 MSB-Justified 格式接口

3．1．3 采集频率和主设备时钟

IIS 主设备时钟频率可以通过采样频率来选择，如下表 28－1 所示。因为 IIS 主设备时钟频率是由 IIS 预分频器产生的（主设备时钟频率＝MCLK / 预分频器值），因此必须选择合适的预分频器的值和 CODECLK 的采样频率类型（256 或者 384fs），才能获得合适的 IISLRCK 频率（IISLRCK 频率＝主设备时钟频率/CODECLK）。

串行位采样频率类型（16/32/48fs）可以通过配置每通道的串行位数和 CODECLK 采样频率类型来完

成（串行位时钟频率类型＝CODECLK 的采样频率类型 / 串行数据位数），如表 28－2 所示：

IISLRCK (fs)	8.000 KHz	11.025 KHz	16.000 KHz	22.050 KHz	32.000 KHz	44.100 KHz	48.000 KHz	64.000 KHz	88.200 KHz	96.000 KHz
CODECLK (MHz)	256fs									
	2.0480	2.8224	4.0960	5.6448	8.1920	11.2896	12.2880	16.3840	22.5792	24.5760
	384fs									
	3.0720	4.2336	6.1440	8.4672	12.2880	16.9344	18.4320	24.5760	33.8688	36.8640

表 28－1　CODEC 时钟（CODECLK=256fs 或者 384fs）

每通道的串行位	8-bit	16-bit
串行时钟频率 (IISCLK)		
@CODECLK=256fs	16fs, 32fs	32fs
@CODECLK=384fs	16fs, 32fs, 48fs	32fs, 48fs

表 28－2 可用的串行位时钟频率（IISCLK＝16 或者 32 或者 48fs）

3．1．4 IIS 串行接口专用寄存器

控制寄存器

IISCON	位	描述	初始化状态
左右通道索引（只读）	[8]	0=左通道 1=右通道	1
发送 FIFO 准备好标志（只读）	[7]	0=没有准备好（空） 1=准备好（非空）	0
接收 FIFO 准备好标志（只读）	[6]	0=没有准备好（未满） 1=准备好(满)	0
发送 DMA 服务请求使能	[5]	0=请求未使能 1=请求使能	0
接收 DMA 服务请求使能	[4]	0=请求未使能 1＝请求使能	0
发送通道空闲命令	[3]	在空闲状态下 IISLRCK 是未激活的（暂停 Rx）。这个位仅在作为 IIS 主机时有效 0=IISLRCK 产生； 1=IISLRCK 不产生。	0
发送通道空闲命令	[2]	在空闲状态下 IISLRCK 是未激活的（暂停 Rx）。这个位仅在作为 IIS 主机时有效 0=IISLRCK 产生； 1=IISLRCK 不产生。	0
IIS 预分频使能	[1]	0=预分频禁止 1=预分频使能	0
IIS 接口使能（启动）	[0]	0=IIS 禁止（停止） 1=IIS 使能（启动）	0

模式寄存器

IISMOD	位	描述	初始状态
主/从模式选择	[8]	0=主模式（IISLRCK 和 IISCLK 是输出模式） 1=从模式（IISLRCK 和 IISCLK 是输入模式）	0
发送/接收模式选择	[7:6]	00=无传输；01=接收模式 10=发送模式；11=发送和接收模式	00
激活左右声道选择电平	[5]	0=左声道低电平（右声道高电平） 1=左声道高电平（右声道低电平）	0
串行接口格式	[4]	0=IIS 兼容格式； 1=MSB-Justified 格式	0
串行数据位每通道	[3]	0=8 位　　1=16 位	0
主时钟（CODECLK）频率选择	[2]	0=256fs　　1= 384fs	0
串行时钟频率选择	[1:0]	00=16fs　　01=32fs 10=48fs　　11=N/A	00

IIS 预分频寄存器

IISPSR	位	描述	初始化状态
预分频器值 A	[7:4]	预分频器 A 的除数 clock_prescaler_A = MCLK/<除数>	0x0
预分频器值 B	[3:0]	预分频器 B 的除数 clock_prescaler_B = MCLK/<除数>	0x0

取值表如下：

IISPSR[3:0]/[7:4]	除数值	IISPSR[3:0]/[7:4]	除数值
0000b	2	1000b	1
0001b	4	1001b	—
0010b	6	1010b	3
0011b	8	1011b	—
0100b	10	1100b	5
0101b	12	1101b	—
0110b	14	1110b	7
0111b	16	1111b	—

IIS 的 FIFO 控制寄存器（IISFCON）

为了启动 IIS 操作，需要按照下列的步骤进行：

1） 通过 IISFCON 寄存器使能 FIFO 操作；

2） 通过 IISCON 寄存器使能 DMA 请求；

3） 通过 IISCON 寄存器启动 IIS 接口。

结束 IIS 操作，需要按照下述的步骤进行：

1） 禁止 FIFO，如果你想要发送保存在 FIFO 中的数据，你必须不要禁止 FIFO 并略过这一步；

2） 通过 IISCON 寄存器来禁止 DMA 请求；

3） 通过 IISCON 寄存器来停止 IIS 接口。

IISFCON	位	描述	初始化状态
发送 FIFO 操作模式选择	[11]	0=普通操作模式 1=DMA 操作模式	0
接收 FIFO 操作模式选择	[10]	0=普通操作模式 1=DMA 操作模式	0
发送 FIFO 使能	[9]	0=FIFO 禁止　1=FIFO 使能	0
接收 FIFO 使能	[8]	0=FIFO 禁止　1=FIFO 使能	0
发送 FIFO 数据计数（只读）	[7:4]	数据计数值= 0-8	000
接收 FIFO 数据计数（只读）	[7:4]	数据计数值= 0-8	000

IIS 的 FIFO 寄存器

IIS 总线接口包含两个 16 字节的 FIFO 分别用于发送和接收模式。每个 FIFO 具有 16 级宽度和 8 级深度模式，允许 FIFO 以半字节为单元操作数据，忽略数据的尺寸。发送和接收 FIFO 操作是通过 FIFO 入口来进行的，入口地址是 0x01D19010。

IISFIF	位	描述	初始化状态
FENTRY	[15:0]	IIS 发送/接收的数据	0

3．2 数字音频编解码器芯片 UDA1341TS 简介

UDA1341TS 是一款集成了 A/D 转换器和 D/A 转换器，采用位流转换技术，具有单一操作特性的芯片。数字音频操作特性，使该芯片是家庭立体声 mini 唱片播放器的尚佳选择，同时由于它的低功耗特性，又特别适合于手持应用，例如 MD/CD，笔记本电脑和数字视频摄像机等。

UDA1341TS 支持 IIS 总线数据格式和 MSB-justified 数据格式，字长最高达到 20 位，LSB-justified 数据格式，字长为 16、18 和 20 位。UDA1341 具备的数字音频操作特性，可以通过 L3 接口与处理器接口，并对回放、重音、音量、低音增强等模式进行控制。

更详细的内容，请查看光盘\芯片资料\UDA1341TS.pdf。关于处理器与该芯片的接口与控制，下面将详细说明。

3．3 与处理器的接口

3．4．1 接口电路

UDA1341TS 与 S3C44B0X 之间的接口电路如下图所示：

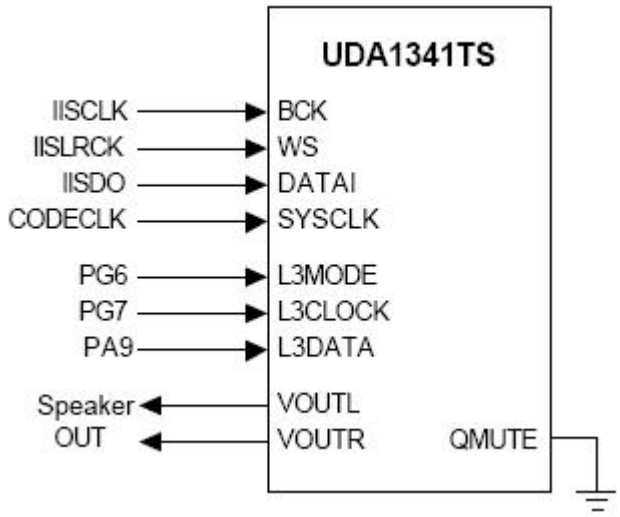

图 28－2 SST39VF160 与 S3C44B0X 的接口电路

UDA1341TS 的控制接口（L3MOD, L3CLOCK, L3DATA）都是由通用的 I/O 端口来控制的。L3 接口方式实际上是一种串行接口，它由 3 根信号线组成，完成处理器和 UDA1341TS 之间的数据和控制信号交换：

L3DATA：处理器接口数据线；

L3MODE：处理器接口模式信号线；

L3CLOCK：处理器接口时钟信号线。

控制接口操作分为两种模式：地址模式和数据传输模式，地址模式下请求选择设备通讯，并定义数据传输模式下的目标寄存器；数据传输可以是双向的，而输入到 UDA1341TS 中的对它的声音操作和系统控制特性进行操作。

3．4．2 接口传输模式

地址模式

地址模式通过 L3MODE 信号置低来进入，伴随 8 个 L3CLOCK 上的时钟信号，向 UDA1341TS 输入 8 个数据位。位 7 到位 2 是 6 位的设备地址，对于 UDA1341TS 来说，设备地址是 000101。位 0 到 1 表达了后来数据的类型，如下表所示：

位 1	位 0	模式	传输
0	0	DATA0	直接寻址寄存器设置；
			扩展寻址寄存器设置；
0	1	DATA1	峰值值读出；
1	0	STATUS	复位，系统时钟频率，数据输入格式，DC 滤波器，输入增益切换、输出增益切换、奇偶控制、双速度和功耗控制；
1	1	未用	

数据传输模式

当 L3MODE 信号置高就进入了数据传输模式，数据传输都是按字节来进行的，最大的输入时钟和数

据传输率是 64fs。

L3 接口的多字节传输时序如下图所示：

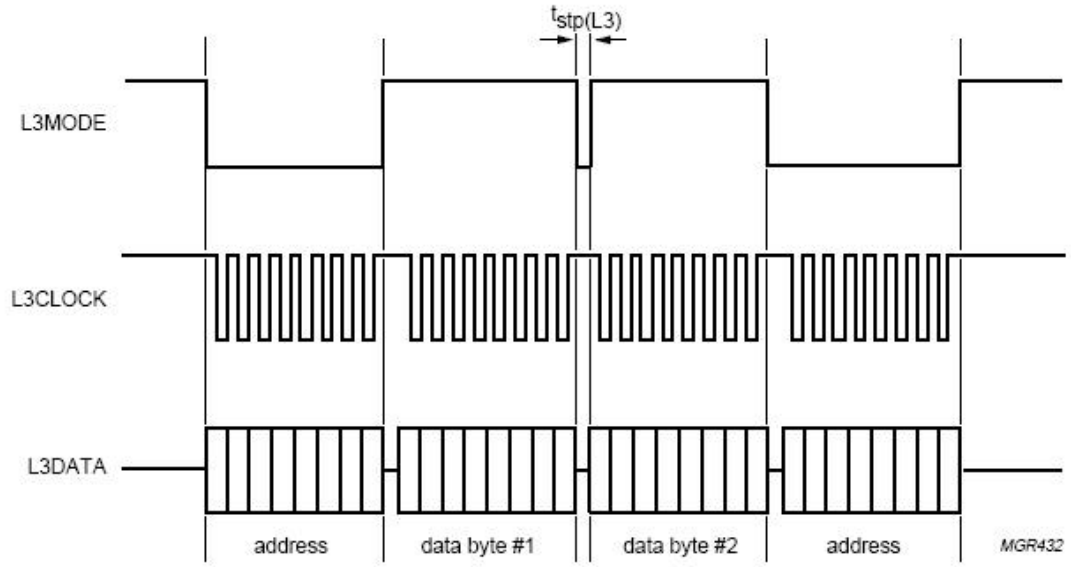

图 28－3　L3 接口多字节传输时序

3．4．3 UDA1341TS 的初始化

UDA1341 的初始化是由对状态寄存器的设置操作来完成。从图 28－3 可以了解到，对寄存器的选择首先由地址模式下的位 0 和位 1 来设定；下面的设定通过第 2 或者第 3 数据模式字节中的 7 和 6 位或者 7、6 和 5 位来设定）。数据字节中 5 到 0 或者 4 到 0 位则是选定寄存器的值。接口程序的编写参考

4 实验内容与步骤

本实验需要连接核心模块与 **IIS 总线接口音频解码器**（IIS-BUS Interface AudioCODEC）模块之间的电路。参考图 28－2 完成电路的连接。

编写完成符合实验要求的源程序。建立工程、编译除错、下载仿真，最终调试出符合要求的源程序。参考光盘中的 source\IIStest 目录下的内容完成实验。

4．1 对 IIS 接口的编程

以下代码完成对 IIS 接口的初始化，实现将 S3C44B0X 的 IIS 接口初始化为主设备，DMA 模式发送，并启动发送：

```
//------------系统时钟修改--------------------------
    //修改 PLL 值，修改 MCLK
    ChangePllValue(0x69,0x17,0x0);  //MCLK=45.1584MHz <-- 5.6448MHz*8
    //对应修改 UART 的波特率
     Uart_Init(45200000,115200);
//------------I/O 口设置-----------------------------
    //备份利用到的 I/O 口的设置
```

```
    save_A=rPCONA;//L3DATA
    save_C=rPCONC;//IIS port
    save_E=rPCONE;//CODEC clk
    save_G=rPCONG;//L3CLK,L3MOD
    save_PC=rPUPC;
    save_PE=rPUPE;
    save_PG=rPUPG;
//工作在总线宽度为 16 位的情况下
#if (BUSWIDTH==32)
    Uart_Printf("IIS test should be configured 16bit data bus\n");
    return;
#else //BUSWIDTH=16
    rPCONC |=0xff;
    rPUPC |= 0xf;
#endif
//设定 CODECLK 引脚功能
rPCONE=(rPCONE&0xffff)+(2<<16); //PE:CODECLK

//----------IIS 接口初始化 ------------------------
    rIISCON=0x22;    //Tx DMA 使能，Rx 空闲，预分频使能
    rIISMOD=0x89;    //主设备状态，发送，左通道低电平有效，IIS 总线格式，16 位数据位，主时钟频率
codeclk=256fs，串行时钟频率=32fs
    rIISPSR=0x33;    //预分频器 Prescaler_A/B 使能，除数=8
    rIISFCON=0xa00;  //Tx/Rx DMA,Tx/Rx FIFO

/****** 启动 IIS 发送 ******/
    rIISCON |=0x1;
    ……
    /****** IIS 发送停止 ******/
    rIISCON=0x0;     //IIS stop
    ……
```

4．2 对 UDA1341TS 控制的编程

4．2．1 L3 控制接口 I/O 的初始化

以下代码实现对 L3 控制接口 I/O 功能的设定：

```
#define L3D (0x200)
#define L3M (0x40)
#define L3C (0x80)
/****** 端口初始化 ******/
    rPCONA = 0x1ff;        //PA9(out)：L3Data
    rPCONG = 0x5000;       //PG6：L3Mode，PG7：L3Clock
    rPUPG   |= 0xc0;       //禁止上拉
```

```
    rPDATG = L3M|L3C;    //L3M=H(起始状态)
                         //L3C=H(起始状态)
```

4．2．2 地址模式写入

以下代码实现地址模式下的 1 字节写入：

```
void _WrL3Addr(U8 data)
{
    U32 vPdata = 0x0;  //L3D=L
    U32 vPdatg = 0x0; //L3M=L(在地址模式下)/L3C=L
    S32 i,j;

    rPDATG = vPdatg; //L3M=L
    rPDATG |= L3C;    //L3C=H

    for(j=0;j<4;j++);          //tsu(L3) > 190ns

    //PA9:L3D PG6:L3M PG7:L3C
    for(i=0;i<8;i++)
    {
    if(data&0x1)//如果数据位为高(‘1’)
    {
        rPDATG = vPdatg; //L3C=L
        rPDATA = L3D;     //L3D=H
        for(j=0;j<4;j++);    //tcy(L3) > 500ns
        rPDATG = L3C;     //L3C=H
        rPDATA = L3D;     //L3D=H
        for(j=0;j<4;j++);    //tcy(L3) > 500ns
    }
    else        //如果数据位为低（‘0’）
    {
        rPDATG=vPdatg;   //L3C=L
        rPDATA=vPdata;   //L3D=L
        for(j=0;j<4;j++);    //tcy(L3) > 500ns
        rPDATG=L3C;       //L3C=H
        rPDATA=vPdata;   //L3D=L
        for(j=0;j<4;j++);    //tcy(L3) > 500ns
    }
    data >>=1;
    }
    rPDATG=L3C|L3M;      //L3M=H,L3C=H
}
```

4．2．3 数据模式写入

以下代码实现数据模式下的 1 字节写入：

```
void _WrL3Data(U8 data,int halt)
{
    U32 vPdata = 0x0;     //L3D=L
    U32 vPdatg = 0x0;     //L3M/L3C=L
    S32 i,j;
    if(halt)
    {
        rPDATG=L3C;        //L3C=H(while tstp, L3 interface halt condition)
        for(j=0;j<4;j++);     //tstp(L3) > 190ns
    }
    rPDATG=L3C|L3M;     //L3M=H(在数据传输模式下)
    for(j=0;j<4;j++);           //tsu(L3)D > 190ns

    //PA9:L3MODE PG6:L3DATA PG7:L3CLOCK
    for(i=0;i<8;i++)
    {
        if(data&0x1)  //如果数据位为高（‘1’）
        {
        rPDATG=L3M;            //L3C=L
            rPDATA=L3D;      //L3D=H
            for(j=0;j<4;j++);    //tcy(L3) > 500ns
            rPDATG=L3C|L3M;     //L3C=H,L3D=H
        rPDATA=L3D;
            for(j=0;j<4;j++);    //tcy(L3) > 500ns
        }
        else       //如果数据位为低（‘0’）
        {
            rPDATG=L3M;            //L3C=L
        rPDATA=vPdatg;   //L3D=L
            for(j=0;j<4;j++);    //tcy(L3) > 500ns
            rPDATG=L3C|L3M;     //L3C=H
        rPDATA=vPdatg;   //L3D=L
            for(j=0;j<4;j++);    //tcy(L3) > 500ns
        }
        data>>=1;
    }
    rPDATG=L3C|L3M;     //L3M=H,L3C=H
}
```

4．2．4 UDA1341 的初始化

整个对 UDA1341 的初始化程序如下：

```
#define FS441KHZ       //定义采样频率 44.1KHz
void Init1341(void)
{
    /******  端口初始化  ******/
    rPCONA = 0x1ff;
```

```
    rPCONG = 0x5000;
    rPUPG    |= 0xc0;

    rPDATG = L3M|L3C;

    /******** L3 接口 *******/
    _WrL3Addr(0x14+2);   //目标地址：状态（status）寄存器 (000101xx+10)
#ifdef FS441KHZ
    _WrL3Data(0x60,0); //0,1,10,000,0 复位,256fs,无 DC 滤波器,iis 格式
#else
    _WrL3Data(0x40,0); //0,1,00,000,0 复位,512fs,无 DC 滤波器,iis 格式
#endif

    _WrL3Addr(0x14+2); //目标地址：状态寄存器(000101xx+10)
#ifdef FS441KHZ
    _WrL3Data(0x20,0); //0,0,10,000,0 不复位,256fs, 无 DC 滤波器,iis 格式
#else
    _WrL3Data(0x00,0); //0,0,00,000,0 不复位,512fs, 无 DC 滤波器,iis 格式
#endif
    _WrL3Addr(0x14+2); //目标地址：状态寄存器(000101xx+10) ， Bit7=1
    _WrL3Data(0x83,0);
    //1,0,0,0,0,0,11 OGS=0,IGS=0,ADC_NI,DAC_NI,sngl speed,AonDon
}
```

5 实验报告

（1） S3C44B0X 的 IIS 总线接口支持哪两种音频串行接口格式？说明这两种格式的区别。

（2） 如何设置 IISLRCK 采样频率，给出相关的计算公式。假设要播放采样频率为 44.1KHz 的 PCM 音乐段，如何对 IIS 接口进行初始化？给出实现的代码段和注释。

（3） S3C44B0X 如何与 UDA1341TS 进行接口？给出接口电路。并说明每一根接口信号线的作用。给出对 S3C44B0X 接口 I/O 进行初始化的代码段，并加以注释。

（4） UDA1341TS 的地址模式写入和数据模式写入在信号时序上有什么区别？给出分别实现 1 字节地址模式写入和 1 字节数据模式写入的函数定义，并加以注释。

（5） 如何对 UDA1341TS 进行初始化？给出实现的代码段，并加以注释。

实验二、DMA 应用实验

1 实验目的

（1）学习 S3C44B0X 的 DMA 通道的功能特性和工作原理；
（2）学习 DMA 在 IIS 总线接口音乐播放中的应用。

2 实验要求

（1） 连接核心模块和 UDA1341TS 解码模块（IIS-BUS Interface AudioCODEC）的电路；为了检验播放效果，需要连接解码器模块和音频功放模块（Audio Power AmplifierInterface）之间的电路，或者插入自备耳机，来收听放音效果。

（2） 理解 DMA 工作原理，编写程序，实现 DMA 模式下的 IIS 总线接口的音乐播放。音乐文件格式为 PCM，采用 44.1KHz 的采样频率。可以首先利用 AXD 中的下载工具将测试播放的音乐数据（testMusic.wav）下载到 0xC200000 地址处，然后运行程序进行播放，直到按下任意键时停止播放。

3 实验分析

3．1 关于 PCM 音乐文件格式

数码音乐的文件格式主要有 WAVE、MP3 及 RA 等，CD 唱片是以音轨的格式储存在光盘内，扩展名为 CDA。格式虽然多，但大致上可分为两类——非压缩性与压缩性；一般专业的数码音乐制作人都会使用非压缩性的 WAVE 格式，而普通用户则比较喜欢压缩率高、文件体积相对较小的 MP3 或 RA 格式。

非压缩性格式 Windows PCM：专业的数码音乐人都喜欢使用非压缩性的数码音乐档案，由于没有经过任何压缩，所以无论进行多少次修改，都不会产生失真的情况，而且处理速度也相对较快。这类文件最典型的代表就是 Windows PCM 格式文件，它是 Windows 的专用数码音乐文件格式，扩展名为 WAV。这种格式文件容许储存单声道或立体声的讯息。要制作 WAV 文件，可使用 Windows 内的录音程序。在本实验中播放的正是这种 PCM 格式的音乐。

WAV 文件作为最经典的 Windows 多媒体音频格式，应用非常广泛，它使用三个参数来表示声音：采样位数、采样频率和声道数。声道有单声道和立体声之分，采样频率一般有 11025Hz（11kHz）、22050Hz（22kHz）和 44100Hz（44kHz）三种。WAV 文件所占存储空间 = (采样频率×采样位数×声道) × 时间/8 (1 字节=8bit)。

关于压缩格式(MP3)音乐，请参考下一个实验——MP3 解码器应用实验。

3．2 S3C44B0X 中的 DMA 控制器

S3C44B0X 具有 4 通道的 DMA 控制器。其中两个 DMA 通道称做 ZDMA（通用 DMA）连接在 SSB（三星系统总线）上，另外两个 DMA 通道称做 BDMA（桥 DMA）连接于 SSB 和 SPB（三星外设总线）之间的接口层。

连接于 SSB 上的 ZDMA 控制器可以用于从存储器到存储器，从存储器到固定目标的 I/O 存储器，和从 I/O 设备到存储器之间的数据传输。另外的两个 BDMA 控制器用于传输数据从存储器到 I/O 设备或者 I/O 设备到存储器；这里的 I/O 设备意味着外设，连接于 SPB，例如 SIO，IIS 和 UART。

DMA 的主要特点是它可以在两地自动传输数据而不需要 CPU 的干涉。ZDMA 和 BDMA 操作都可以通过软件，内部外设的请求或者外部请求引脚（nXDREQ0，1）来启动。

ZDMA 中最重要的特性是 on-the-fly 模式，减少了在外部存储器和固定外设（固定的源或者目标地址设备）之间的 DMA 操作中的周期数。通常，DMA 传输包括两个分开的周期，一个是从源存储器或者 I/O 设备的“读”，另一个是向存储器或者目标 I/O 设备“写”入。为了进行这些操作，存储器控制器必须首先从数据总线读数据，然后再将这个数据写入数据总线。on-the-fly 模式则具有合并的读/写周期。换句话来说，就是存储器控制器对源或者目标设备在总线上的读或写都会产生响应信号，与此同时，存储器控制器也为存储器的操作提供读或者写对应的控制信号。这样一种 on-the-fly 模式可以减小 DMA 请求周期的数目，与普通的 DMA 周期不同，后者具有分开的读和写周期。为了完成 on-the-fly 模式，源的总线尺宽度应该与目标总线宽度相同。

3．2．1 ZDMA（通用 DMA）

连接于 SSB 的 ZDMA 通道可以完成从外部存储器到外部存储器的数据传输。不同于 BDMA（桥 DMA），ZDMA 可以用来在映射于存储空间中的设备或者存储器之间进行数据的传输。换句话说，在固定源和外部存储器之间的数据传输，外部存储器和外部存储器之间的，外部存储器和固定目标之间的数据传输都可以采用这种 DMA 完成。DMA 操作可以通过软件或者外部 DMA 请求信号启动。

在 ZDMA 中，具有一个允许多重传输的暂存缓冲器来提高总线利用率和传输速度。换句话说，S3C44B0X 具有一个 4 字的 FIFO 类型的缓冲器来支持 DMA 操作中的 4 字的猝发传输。例如，在存储器之间的 DMA 操作，紧接着一个 4 字的猝发读之后，是一个 4 字的猝发写操作。

3．2．2 BDMA（桥 DMA）

BDMA 连接于 SSB 和 SPB 的接口层——桥中。BDMA 的主要角色是在外部存储器和内部外设之间传输数据，这些内部外设包括 UART, IIS 和 SIO，它们都是连接在 SPB 上的。定时器也可以在任何时候请求 BDMA 操作。这对于进行自动 ADC 应用很有用。

因为 BDMA 是在桥内的，是 SSB 和 SPB 之间的一个接口层，它可以在连接于 SSB 或者 SPB 上的两种设备之间传输数据，因此它可以实现在存储器（连接于 SSB）到连接于 SPB 上的外设（或存储器）之间传输数据。

BDMA 不支持 4 字的猝发传输（块传输模式）因为 BDMA 不具有暂存缓冲器，也因为附属于 SPB 的外设是很慢的。特别的，BDMA 也可以支持从外部存储器到外部存储器之间数据传输，但它不是一个非常有效率的数据传输方法。推荐采用 ZDMA 来代替，以获得更快的速度和总线利用率。但如果需要更多的 DMA 数据传输通道，则可以采用 BDMA。

3．2．3 外部 DMA 请求/应答协议

有 4 种类型的外部 DMA 请求/应答协议。每种类型定义了与这些协议相关的 DMA 请求和应答。尽管 ZDMA 和 BDMA 可以支持外部触发，这些协议仅对应于 ZMDA，BDMA 传输中不使用到。由于在本实验中我们仅运用到 BDMA，这里就不详加介绍了。

3．2．4 DMA 请求源选择

在 ZDMA 模式下，由软件或者硬件产生 nXDREQ（外部 DMA 请求信号），它们就是 DMA 请求源。软件启动可以通过在 ZDCON0/1 寄存器写 CMD 区域为 01 来实现。例如，在 DMA 传输的开始，即在启动 DMA 前，对 DMA 相关的参数，例如源地址，目标地址，传输计数等等，必须全部配置好。然后，DMA 操作就通过 CMD 区域写入 01 来启动。

在 BDMA 模式下，有 6 种硬件请求源，UART0, UART1, SIO, 定时器和 IIS。BDMA 与 ZDMA 一样可以通过软件启动。这些请求源可以通过写入 BDICNT 寄存器的 QSC 区域来选择。

3．2．5 自动重载模式

在自动重载模式下，每当 DMA 计数器的值减少到 0，Z(B)DCSRCn, Z(B)DCDSTn, 和 Z(B)DCCNTn 寄存器的值将从 Z(B)DISRCn，Z(B)DIDESn, Z(B)DICNTn 寄存器载入。在寄存器 Z(B)DISRCn，Z(B)DIDESn, Z(B)DICNTn 中，保存了与 DMA 操作相关的配置参数，例如，源/目标地址和源/目标传输计数。这种自动重载可以恢复 DMA 操作的设定。换句话说，要修改配置，可以将 Z(B)DISRCn，Z(B)DIDESn, Z(B)DICNTn 寄存器中的值进行修改，但不影响基于原配置工作的当前 DMA 操作。但是这种参数的自动重载不能够保证在当前的 DMA 操作之后自动重新运行 DMA，如果 Z(B)DCONn 的 CMD 区域重新被写入或者外部 DMA 请求引发则 DMA 将会重新运行。

为了支持自动重载模式，DMA 应该具有两套寄存器的设置，寄存器 Z(B)DISRCn，Z(B)DIDESn, Z(B)DICNTn 有初始的 DMA 配置，前面提到的寄存器 Z(B)DCSRCn, Z(B)DCDSTn, 和 Z(B)DCCNTn 则具有反应当前 DMA 操作的配置。在 DMA 操作期间，这些寄存器对于源地址，目标地址，和剩余的传输计数值都具有动态的值。

3．3 BDMA 相关寄存器

3．3．1 BDMA 控制寄存器（BDCONn）

BDCONn	位	描述	初始化状态
INT	[7:6]	保留	00
STE	[5:4]	DMA 通道的状态（只读） 00＝准备好　　01＝还没有终止计数 10＝终止计数　　11＝N/A 在 DMA 计数器从初始的计数值开始减少之前，STE 将始终是准备好状态。	00
QDS	[3:2]	禁止/使能外部/内部 DMA 请求 （来自 UARTn, SIO, IIS, Timer） 00＝使能　　其他＝禁止	00
CMD	[1:0]	软件命令 00：无命令，在写入其他数值后 CMD 位将被自动清 0。 01：保留 10：保留 11：取消 DMA 操作	00

3．3．2 BDMA0 的初始化源/目标地址和计数寄存器（BDISRC0, BDIDES0, BDICNT0）

寄存器	地址	读/写	描述	复位值
BDISRC0	0x01f80004	R/W	BDMA0 源地址寄存器的初始值	0x00000000
BDIDEC0	0x01f80008	R/W	BDMA0 目标地址寄存器的初始值	0x00000000
BDICNT0	0x01f8000C	R/W	BDMA0 计数寄存器的初始值	0x00000000

3．3．3 BDMA0 的当前源/目标地址和计数寄存器（BDCSRC0, BDCDES0, BDCCNT0）

寄存器	地址	读/写	描述	复位值
BDCSRC0	0x01f80010	R	BDMA0 源地址寄存器的当前值	0x00000000
BDCDEC0	0x01f80014	R	BDMA0 目标地址寄存器的当前值	0x00000000
BDCCNT0	0x01f80018	R	BDMA0 计数寄存器的当前值	0x00000000

3．3．4 BDMA1 的初始化源/目标地址和计数寄存器（BDISRC1, BDIDES1, BDICNT1）

寄存器	地址	读/写	描述	复位值
BDISRC1	0x01f80024	R/W	BDMA1 源地址寄存器的初始值	0x00000000
BDIDEC1	0x01f80028	R/W	BDMA1 目标地址寄存器的初始值	0x00000000
BDICNT1	0x01f8002C	R/W	BDMA1 计数寄存器的初始值	0x00000000

3．3．5 BDMA1 的当前源/目标地址和计数寄存器（BDCSRC1, BDCDES1, BDCCNT1）

寄存器	地址	读/写	描述	复位值
BDCSRC1	0x01f80030	R	BDMA1 源地址寄存器的当前值	0x00000000
BDCDEC1	0x01f80034	R	BDMA1 目标地址寄存器的当前值	0x00000000
BDCCNT1	0x01f80038	R	BDMA1 计数寄存器的当前值	0x00000000

3．3．6 BDMAn 初始/当前源地址寄存器（BDISRC, BDCSRC）

BDISRCn/BDCSRCn	位	描述	初始化状态
DST	[31:30]	传输数据的尺寸 00＝字节　　01＝半字 10＝字　　11＝未用	00
DAL	[29:28]	载入地址的方向	00

		00＝N/A　　01＝增加 10＝减少 11＝内部外设（固定地址）	
ISADDR/CSADDR	[27:0]	BDMAn 的初始化/当前源地址 如果源地址是内部的外设，特殊寄存器的地址将被使用。 例如，如果源是 UART0 输入缓冲区，那么这里可以填入它的地址。	0x0000000

3．3．7 BDMAn 初始/当前的目标地址寄存器（BDIDES，BDCDES）

BDISRCn/BDCSRCn	位	描述	初始化状态
TDM	[31:30]	传输方向模式设定 00＝保留 01＝M2IO（从外部存储器到内部设备） 10＝IO2M（从内部设备到外部存储器） 11＝IO2IO（从内部设备到内部设备）	00
DAS	[29:28]	保存地址的方向 00＝N/A　　01＝增加 10＝减少 11＝内部外设（固定地址）	00
ISADDR/CSADDR	[27:0]	BDMAn 的初始化/当前目标地址 如果目标地址是内部的外设，特殊寄存器的地址将被使用。 例如，如果源是 UART0 输出缓冲区，那么这里可以填入它的地址。	0x0000000

3．3．8 BDMAn 初始/当前计数寄存器（BDICNTn BDCCNTn）

BDICNT0/BDCCNT0	位	描述	初始化状态
QSC	[31:30]	DMA 请求源选择 00＝N/A　　01＝IIS 10＝UARTn　11＝SIO	00
保留	[29:28]	00：握手模式	00
保留	[27:26]	01：unit transfer mode	01
保留	[25:24]	00：on-the-fly 模式在 BDMAn 下并不支持	00
INTS	[23:22]	中断模式设置 00＝轮流检测模式　01＝N/A 10＝在传输时发生中断； 11＝终止计数时发生中断。	00
AR	[21]	自动重载和自动启动在 DMA 计数器达到 0。 0＝禁止； 1＝使能。甚至在 DMA 计数到达 0，DMA	0

		使能位（EN 位）仍然是 1。但是 DMA 将会启动来操作仅在启动命令或者 DMA 请求被激活	
EN	[20]	DMA 的硬件使能/禁止 0＝禁止 DMA； 1＝使能 DMA 如果 QDS 位为 00b，DMA 请求会得到服务。同样如果软件命令开始，DMA 操作将会发生。如果 EN 位为 0，DMA 将不会操作尽管软件命令启动。 如果软件命令取消，DMA 操作将会被取消，EN 位将会被清 0。 在计数的最后，EN 位将会被清 0。 注意：不要同 BDICNT 寄存器的其他位一起设置。用户应该在设置好其他位后，再设置 EN： 1．设置 BDICNT 寄存器同时对 EN 位清 0； 2．设置 EN 位使能。	0
ICNT/CCNT	[19:0]	BDMA0 的传输计数器。传输计数器必须是正确的值，例如，如果 DST 是字，ICNT 必须是 4 的倍数。 如果 1 字节被发送，ICNT 将会减 1。 如果 1 个半字被发送，ICNT 必须减 2。 如果 1 个字被发送，ICNT 必须减 4。	0x00000

3．4 IIS 接口的 DMA 传输模式

在这个模式中，IIS 的发送和接收 FIFO 操作都由 DMA 控制器来完成，在发送和接收模式中由 FIFO 准备好标志来自动产生 DMA 服务请求。

4 实验内容与步骤

本实验需要连接核心模块与 **IIS 总线接口音频解码器**（IIS-BUS Interface AudioCODEC）模块之间的电路。参考图 28－2 完成电路的连接。为了检验播放效果，需要连接解码器模块和**音频功放模块**（Audio Power AmplifierInterface）之间的电路，或者插入自备耳机，来收听放音效果。

编写完成符合实验要求的源程序。建立工程、编译除错、下载仿真，最终调试出符合要求的源程序。参考光盘中的 source\IIStest 目录下的内容完成实验。在进行音乐播放程序调试时，可以利用 AXD 中的文件下载工具来将音乐文件下载到特定地址处。方法如下：

点击菜单项“File”下的“Load Memory From File……”，弹出对话框：

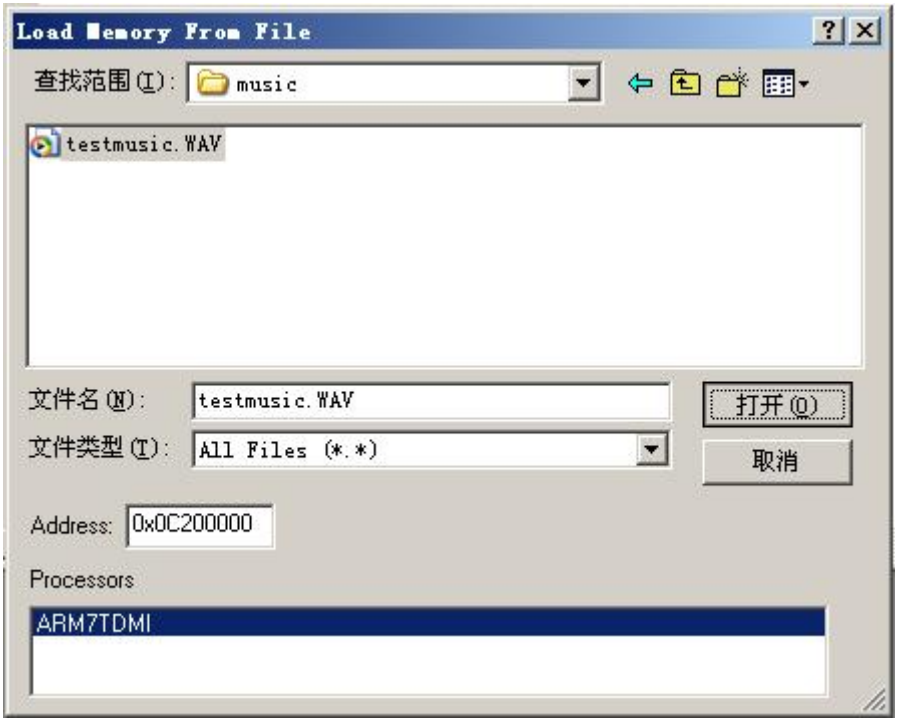

图 29－1 音乐格式文件下载

注意在 Address 栏中填入 0xc200000，选取音乐文件，点击“打开”。调试器即把音乐文件下载到指定的地址处。

在上一个实验中，我们已经对如何初始化 IIS 接口进行了学习，现在我们使 IIS 传输工作在 DMA 模式下来进行音乐的播放。工作在 DMA 模式下就是由 DMA 控制器来管理 IIS 传输，FIFO 准备好标志的置位将自动产生 DMA 服务请求，CPU 因此可以得到释放而去做其他的事情。在本实验中，音乐的播放，实际上是将音频数据从外部存储器（SDRAM）读出，然后写入到 IIS 接口设备，是从外部存储器到内部外设的数据传输，由此可知可以采用 BDMA 通道来实现。

4．1 对 DMA 的初始化和启动编程

实现 BDMA 模式下的 IIS 传输非常简单，我们仅需要对 BDMA 相关的几个寄存器进行正确的设置，并启动 DMA 服务即可。这里采用了 BDMA0 通道。

代码如下：

```
#define   RAM_ADDRESS 0xc200000   //定义音乐存放的起始空间
Buf=(unsigned char *)RAM_ADDRESS; //指针指向起始地址
/************** BDMA0 初始化**************/
rBDISRC0=(1<<30)+(1<<28)+(int)(Buf+0x30);  //Half-word,增加,源地址为 Buf+30
//注意：音乐文件的前 30 个字节用来放置非音乐数据的其他信息；

rBDIDES0=(1<<30)+(3<<28)+((int)IISFIF);       //M2IO,fix,IISFIF
//这里 IISFIF 是 IIS 接口 FIFO 的入口寄存器，参考实验一的 3.1.4 节

rBDICNT0=(1<<30)+(1<<26)+(3<<22)+(1<<21)+(0<<20)+size;
//DMA 请求源：IIS，当计数结束产生中断，自动重载使能, DMA 未使能
//这里的 size 是发送文件的长度
```

```
rBDICNT0 |= (1<<20);//使能 DMA 操作

rBDCON0 = 0x0<<2;
```

完成 DMA 初始化并使能 DMA 操作后，再对 IIS 进行初始和启动，IIS 传输就开始了。

4．2 DMA 中断处理编程

在完成指定计数值数据的传输后，DMA 的传输工作完成，由于初始化时 rBDICNT0 寄存器的 INTS（中断模式设置）的设定，就会产生 DMA 中断，这时就可以在中断处理程序中做一些必要的处理。

指定中断处理函数和开启中断：

```
pISR_BDMA0=(unsigned)TR_Done;
rINTMSK=~(BIT_GLOBAL|BIT_BDMA0);
```

中断处理程序：

```
void __irq BDMA0_Done(void)
{
    rI_ISPC=BIT_BDMA0; //清除 pending（中断请求）位
    WrUTXH0('.');
}
```

5．实验报告

（1） S3C44B0X 的 DMA 控制器具有哪两种类型 DMA 通道？说明这两种 DMA 通道的特性和区别。

（2） BDMA 具有哪几种 DMA 请求源？如何设定请求源的类型？

（3） IIS 总线接口的 DMA 传输模式是如何工作的？

（4） 针对实验要求，如何进行 BDMA 相关寄存器进行设置，给出初始化 DMA 传输的初始化和启动代码段，并就以注释。

（5） 给出调试成功能够正确播放音乐文件的完整代码清单，注意加以必要的注释。