第8章 串行通信接口

8.1 串行通信基础知识

8.2 S3C2410串行接口

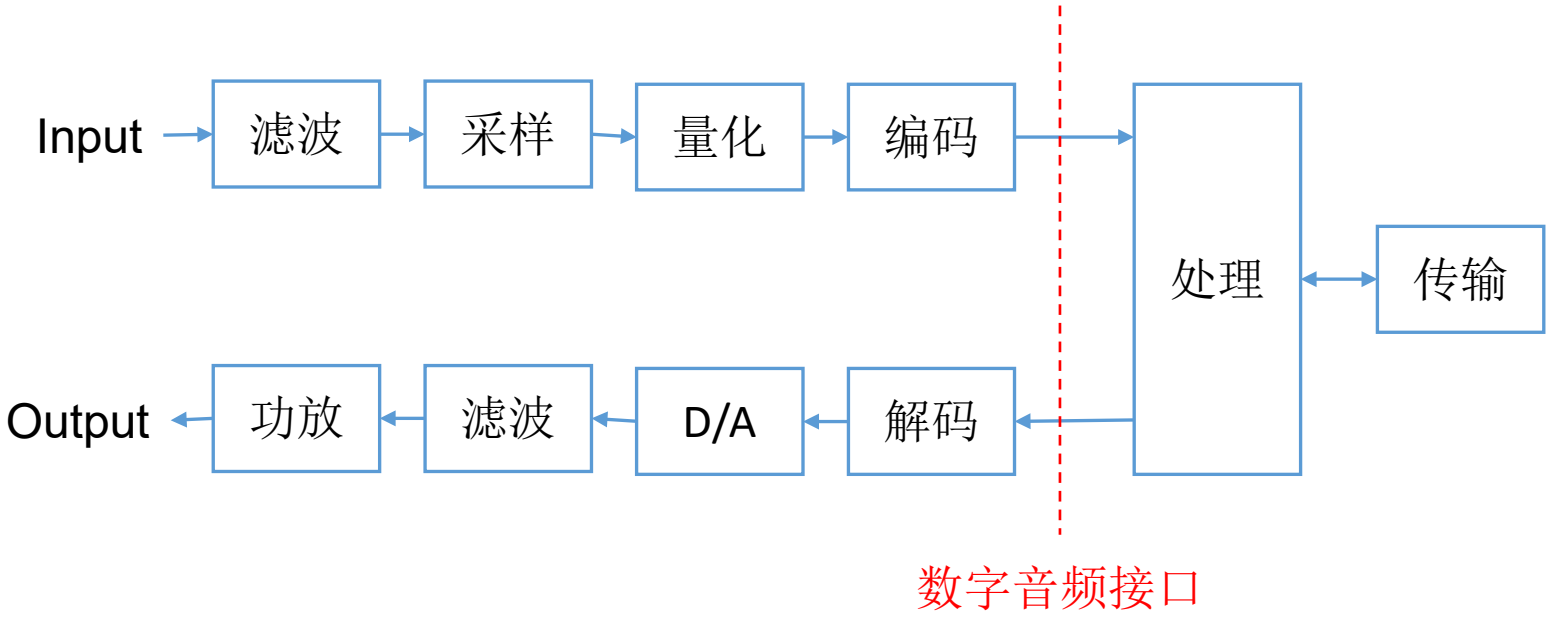
8.3 串行通信举例

8.4 IIS串行数字音频接口

8.4 串行数字音频接口

一、数字音频信号处理基础知识

- 音频信号的处理大都采用数字化的处理方法，包括音频信号的采集、滤波、压缩、分析、存储、传播等
- 数字音频处理流程：



- 二维的模拟音频信号数字化的两个过程：时间离散化（采样）和幅度离散化（量化）
  - 时间离散化涉及的主要参数：采样率—单位时间内采集样本的次数，表征时间分辨率；
  - 幅度离散化涉及的主要参数：位数—信号幅度的动态范围和分辨率；
- 模拟音频信号经过采样、量化后按一定的格式进行数字化表达（编码），构成数字化的序列信号，就是数字音频信号；
- 数字音频信号还原成模拟音频信号的过程则相反，将连续的数字音频序列按采样时的频率和顺序转换成对应的模拟信号

- 采样是时间上的数字化，采样频率的选择应该遵循奈奎斯特（Nyquist）采样理论：信号频带在(0，f_h)的时间连续信号m(t)，如果以T<1 / 2 fh的间隔对它进行等间隔抽样，则m(t)将被所得到的抽样值完全确定。
- 简单的说，就是：$f_s > 2B = 2f_H$
- 人耳所能感知的音频范围大约在20Hz~20kHz，对于音乐信号，采样频率应该在40kHz左右。常用的音频采样频率有8kHz、11.025kHz、22.05kHz、16kHz、37.8kHz、44.1kHz、48kHz、96kHz等。

- 量化则是信号幅度上的数字化。量化位数决定了模拟信号数字化以后的动态范围。常用的有8位、12位和16位。
- 量化位越高，信号的动态范围越大，数字化后的音频信号就越接近原始信号，但所需要的存储空间也越大。

- 声道数是反映音频数字化质量的另一个重要参数。有单声道、双声道和多声道之分。
- 双声道又称为立体声，在硬件中有两条线路，音质和音色都要优于单声道，但数字化后占据的存储空间的大小要比单声道多一倍。
- 多声道能提供更好的听觉感受，不过占用的存储空间也更大。

常见的音频文件格式

数字音频数据格式有PCM、MP3、WMA、WAV、AAC、Ogg Vorbis、RA、ATRAC-3等多种不同的文件格式。

1．PCM (Pulse Code Modulation，脉冲编码调制)

PCM音频格式是音频信号数字化后的原始数字序列，是计算机应用中能够达到的最高保真水平编码格式。

PCM格式音频文件采样率一般为44.1kHz，精度为16位或32位。16位双声道PCM音频数据速率为1.41Mbps，32位精度时为2.42Mbps。

2．MP3(MPEG1 Layer-3音频文件)

压缩比达1：10~1：12，有损压缩。

MP3（全称为MPEG1 Layer-3音频文件）是对PCM音频数据进行有损压缩编码得到的音频文件。立体声MP3数据速率为112kbps至128kbps、256kbps等。

3．WMA(Windows Media Audio)

针对网络市场，质量高，加入版权保护和流媒体。

WMA是由微软开发的Windows Media Audio编码的文件格式。在64kbps码率下即可达到接近CD的音质。WMA支持流技术，可以实现在线广播。

4．WAV

音质好，能轻松转换成其它格式。

WAV由微软开发的符合RIFF（Resource Interchange File Format，资源互换文件格式）规范的音频文件。文件头保存了音频流的编码参数，文件数据块是PCM格式的样本。

- WAV文件中，声道0代表左声道，声道1代表右声道。在多声道WAV文件中，样本是交替出现的。
- 16位的WAV文件首先存储PCM样本的低有效字节，再存储高有效字节。多声道WAV文件，样本是交替的。

表 WAV文件头格式说明

偏移地址	字节数	数据类型	内　　容
00H	4	char	“RIFF”标志
04H	4	long int	文件长度
08H	4	char	“WAVE”标志
0CH	4	char	“fmt”标志
10H	4	过渡字节、类型内容不定	
14H	2	int	格式类别，取值“10H”为 PCM 形式的声音数据
16H	2	int	通道数，单声道为 1，双声道为 2
18H	2	int	采样率，每秒样本数，表示每个通道的播放速度
1CH	4	long in t	波形音频数据传送速率，其值为通道数×每秒数据位数×每样本的数据位数/8。播放软件利用此值可以估计缓冲区的大小
20H	2	int	数据块的调整数（按字节算），其值为通道数×每样本的数据位值/8。播放软件需要一次处理多个该值大小的字节数据，以便将其值用于缓冲区的调整
22H	2		每样本的数据位数，表示每个声道中各个样本的数据位数。如果有多个声道，对每个声道而言，样本大小都一样
24H	4	char	数据标记符“data”
28H	4	long in	语音数据的长度

5．Ogg Vorbis

开源，免费，灵活，开放。

Ogg Vorbis（Vorbis是OGG项目中音频编码的正式命名）是一个高质量的音频编码方案，Ogg Vorbis可以在相对较低的数据速率下实现比MP3更好的音质，而且它可以支持多声道。Ogg Vorbis是一种灵活开放的音频编码，能够在编码方案已经固定下来后，继续对音质进行明显的调节和新算法的改良。Ogg Vorbis像一个音频编码框架，可以不断导入新技术逐步完善。

6．RA (Real Audio)

针对网络市场，功能丰富 。

RA（Real Audio）格式完全针对网络上的媒体市场，具有非常丰富的功能。RA格式最大的亮点是这种格式可以根据听众的带宽来控制码率，在保证流畅的前提下尽可能提高音质。RA可以支持多种音频编码，其中包括ATRAC3。和WMA一样，RA不但支持边读边放，也同样支持使用特殊协议来隐匿文件的真实网络地址，从而实现只在线播放而不提供下载的欣赏方式。

7．APE

无损压缩，压缩比高。

APE是Monkey’s Audio提供的一种无损压缩格式，压缩后的文件是与MP3一样可以播放的音频文件格式。APE的压缩比远低于其他格式，但由于能够做到真正无损，因此获得了不少发烧用户的青睐。在现在有不少的无损压缩方案中，APE具有令人满意的压缩比，以及飞快的压缩速度，成为发烧友的唯一选择。

8．AAC (Advanced Audio Coding)

专为音乐社区设计。

AAC（Advanced Audio Coding，高级音频编码技术）是杜比实验室为音乐社区提供的技术，声称最大能容纳48通道的音轨，采样率达96kHz。AAC在320kbps的数据速率下能为5.1声道音乐节目提供相当于ITU-R广播的品质。AAC是遵循MPEG-2的规格所开发的技术，与MP3比起来，它的音质比较好，也能够节省大约30%的存储空间与带宽。

9．ATRAC 3 (Adaptive Transform Acoustic Coding3)

有版权保护，音质与MP3相当(压缩比达1:10)。

ATRAC3（Adaptive Transform Acoustic Coding3:自适应声学转换编码技术），是日本SONY公司开发的一种有损压缩格式。它是一项基于听觉心理学领域的研究和不损伤可闻声质量的数码音频译码压缩技术，对音乐资讯有效的进行压缩，其压缩率（约为ATRAC的2倍）和音质均与MP3相当。

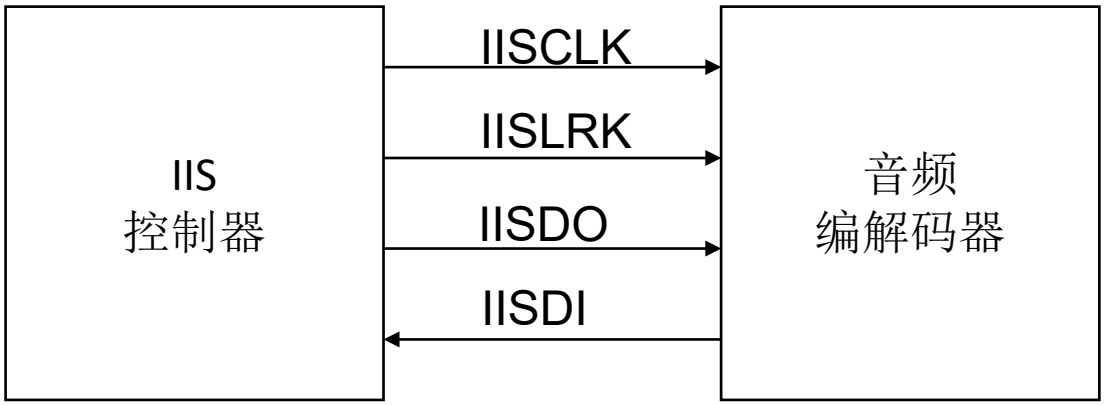
二. 基于IIS接口的音频系统

1. I2S总线结构

I2S总线(I2S/IIS/I^2S，Inter-IC Sound Bus，集成电路内置音频总线)是由Philip公司提出的一种串行数字音频总线协议，是音频编解码器常用的串行音频数字接口。

I2S总线定义了四根信号线：

- IISDI，串行数据输入；
- IISDO，串行数据输出；
- IISLRCK，左右声道选择线；
- IISCLK，串行数据位时钟线；



IISLRCK在MSB位发送的前一个时钟周期内发生改变，可以使从收发器建立数据序列的传输同步。

2. S3C2410 I2S总线控制器结构

S3C2410 I2S总线接口的内部结构见下图所示:

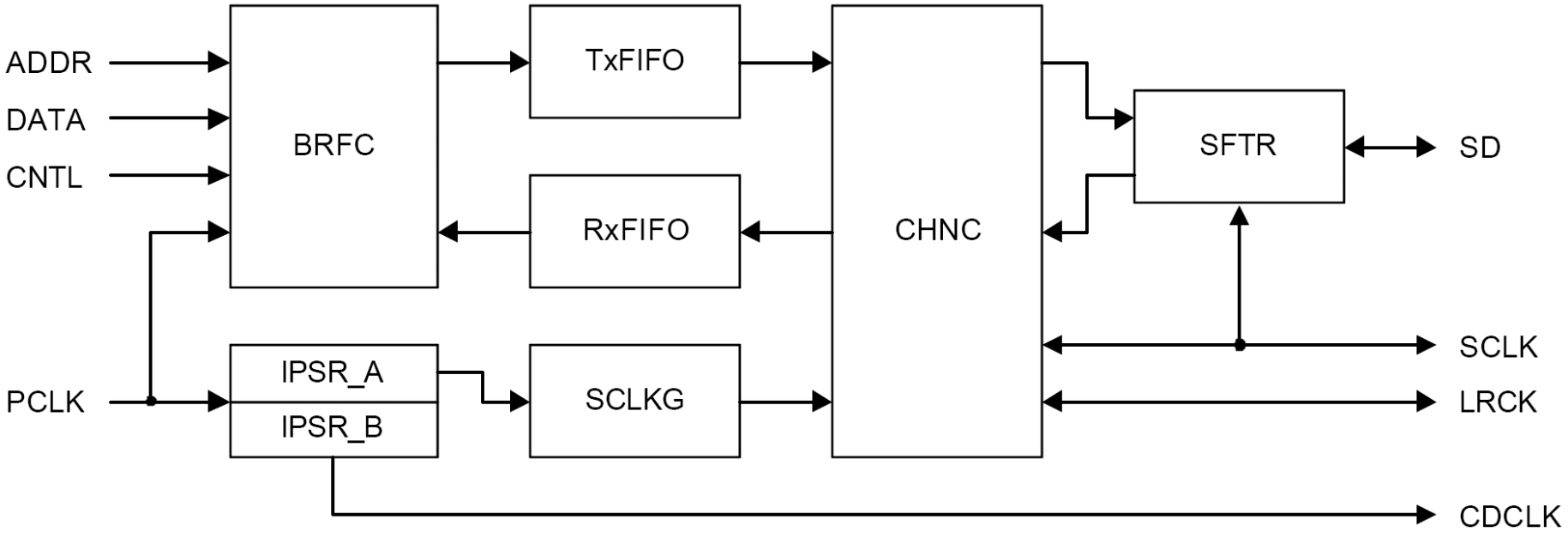


图 S3C2410 I2S总线控制器结构图

S3C2410 I2S总线接口各模块的功能描述如下：

BRFC：总线接口、寄存器区和状态机。总线接口逻辑和FIFO访问控制。

IPSR：两个5bit的预分频器IPSR_A和IPSR_B，分别用于I2S总线接口主时钟发生器和外部CODEC时钟发生器

TxFIFO和RxFIFO：收发各32byte的FIFO。发送数据时，数据写到TxFIFO；接收数据时，数据从RxFIFO读取。

SCLKG：主IISCLK发生器。在主设备模式时，由主时钟产生串行位时钟。

CHNC：通道发生器和状态机，产生和控制IISSCLK和IISLRCK

SFTR：16位移位寄存器。发送时，数据移入SFTR并转换成串行数据输出；接收时，串行数据移入SFTR并转换成并行数据输出。

2. S3C2410 I2S总线控制器结构

- S3C2410A提供一个I2S（Inter-IC Sound）总线接口，可以连接一个外部8/16位立体声音频CODEC，支持I2S总线数据格式和MSB-justified数据格式。
- 该接口对FIFO的访问采用DMA传输模式。
- IIS接口可以同时发送数据和接收数据，也可以只发送或只接收数据。

3. S3C2410的I2S总线控制器的3种工作方式：

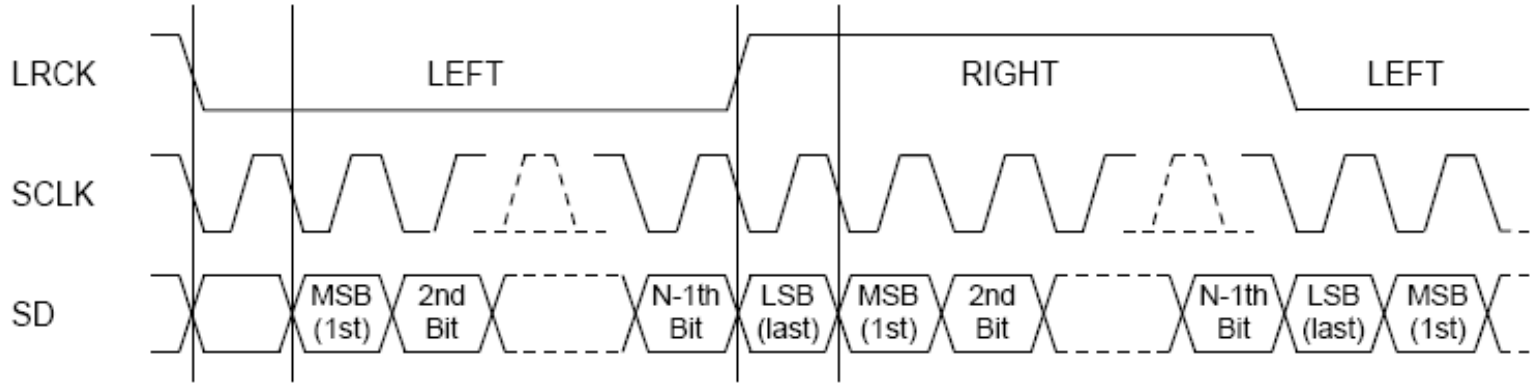
①正常传输方式：使用IISCON寄存器对FIFO进行控制。I2S控制寄存器有一个FIFO就绪标志位，发送数据时，如果发送FIFO不空，则FIFO就绪标志位为“1”；如果发送FIFO为空，该标志为“0”。在接收数据时，当接收FIFO是不满时，FIFO就绪标志位为“1”，指示可以接收数据；若接收FIFO满，则该标志为“0”。通过FIFO就绪标志位，可以确定CPU读/写FIFO的时间。

②DMA传输方式：利用DMA控制器来控制发送和接收FIFO的数据存取，由FIFO就绪标志来自动请求DMA的服务。

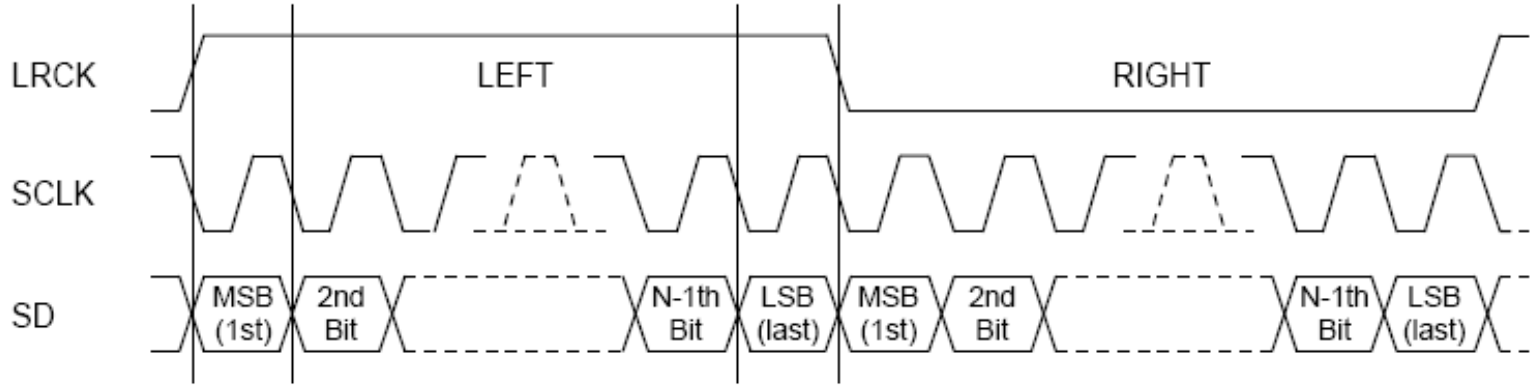
③发送和接收方式：在发送和接收方式，I2S总线接口可以同时发送和接收数据。

4. S3C2410 I2S总线格式和MSB-justified格式

S3C2410的I2S总线接口支持I2S总线数据格式和MSB-justified(MSB-调整)总线数据格式。



IIS-bus Format (N=8 or 16)



MSB-justified Format (N=8 or 16)

IISLRCK与CODECLK的关系如下表所示，表中*f*s为采样频率。

表 IISLRCK与CODECLK的关系

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

(4) S3C2410A I2S总线控制器的寄存器

S3C2410 I2S总线控制器有5个寄存器：控制寄存器（IISCON）、模式寄存器（IISMOD）、预分频器（IISPSR）、FIFO控制寄存器（IISFCON）、FIFO寄存器（IISFIFO）

表 IIS相关寄存器

寄存器	地址		说明	复位后的值
IISCON	0x55000000	读/写	IIS控制寄存器	0x100
IISMOD	0x55000004		IIS模式寄存器	0x000
IISPSR	0X55000008		IIS分频寄存器	0x000
IISFCON	0x4800000C		FIFO控制寄存器	0x0000
IISFIFO	0x48000010		FIFO寄存器	0x0000

表 IISCON的位功能

IISCON位名	位	功能
左/右通道索引(只读)	[8]	0：左通道；1：右通道
发送FIFO就绪标志(只读)	[7]	0：发送FIFO空；1：发送FIFO不空
接收FIFO就绪标志(只读)	[6]	0：接收FIFO满；1：接收FIFO未满
发送DMA服务请求	[5]	0：禁止；1：使能
接收DMA服务请求	[4]	0：禁止；1：使能
发送通道空闲命令	[3]	在空闲状态，IISLRCK无效(暂停发送)。 0：不空闲；1：空闲
接收通道空闲命令	[2]	在空闲状态，IISLRCK无效(暂停接收)。 0：不空闲；1：空闲
I2S预分频器	[1]	0：禁止；1：使能
I2S接口	[0]	0：禁止(停止)；1：使能(启动)

表　IISMOD的位功能

IISMOD位名	位	功能
主/从模式选择	[8]	0：主模式(IISLRCK和IISCLK为输出) 1：从模式(IISLRCK和IISCLK为输入)
发送/接收模式选择	[7:6]	00：不传输；01：接收模式 10：发送模式；11：发送和接收模式
左/右通道的有效电平	[5]	0：左通道为低电平(右通道为高电平); 1：左通道为高电平(右通道为低电平)
串行接口格式	[4]	0：I2S格式；1：MSB (left)-justified格式
每个通道的串行数据位	[3]	0：8位；1：16位
主时钟频率选择	[2]	0：256 fs；1：384 fs (fs为采样频率)
串行位时钟频率选择	[1:0]	00：16 fs；01：32 fs；10：48 fs；11：N/A

表 **IISPSR**的位功能

IISPSR位名	位	描述
预分频器A控制	[9:5]	数据值(N)：0～31 注：预分频器A使主时钟用于内部模块，分频系数为N＋1
预分频器B控制	[4:0]	数据值(N)：0～31 注：预分频器B使主时钟用于外部模块，分频系数为N＋1

表 IISFCON的位功能

IISFCON位名	位	描述
发送FIFO访问模式选择	[15]	0：正常模式；1：DMA模式
接收FIFO访问模式选择	[14]	0：正常模式；1：DMA模式
发送FIFO使能控制	[13]	0：禁止；1：使能
接收FIFO使能控制	[12]	0：禁止；1：使能
发送FIFO数据计数(只读)	[11:6]	数据计数值：0～32
接收FIFO数据计数(只读)	[5:0]	数据计数值：0～32

⑤ IISFIFO (IIS FIFO寄存器) 是一个可读/写的寄存器，该寄存器有2个地址：0x5500 0010 (小端/半字、小端/字、大端/字)和0x5500 0012 (大端/半字)。复位后的初始值为0x0。该寄存器为I2S总线接口发送和接收数据。

(5) I2S总线接口的启动与停止

●启动I2S操作，需要执行如下过程：

① 允许IISFCON寄存器的FIFO；

② 允许IISFCON寄存器的DMA请求；

③ 允许IISFCON寄存器的启动。

●结束I2S操作，需要执行如下过程：

① 禁止IISFCON寄存器的FIFO，如果还想发送FIFO的剩余数据，跳过这一步；

② 禁止IISFCON寄存器的DMA请求；

③ 禁止IISFCON寄存器的启动。

8.4.2 IIS接口应用举例

1. 音频接口电路设计

UDA1341TS可把通过麦克风音频输入通道输入的立体声模拟信号转化为数字信号，同样也能把数字信号转换成模拟信号，通过SPEAKER音频输出通道输出。

通过L3总线可以利用UDA1341TS内部的PGA(可编程增益放大器)、AGC(自动增益控制)功能对模拟信号进行处理。对于数字信号，UDA1341TS提供DSP（数字音频处理）功能。

对于UDA1341TS芯片的详细使用说明，请查阅该芯片的数据手册（datasheet）。

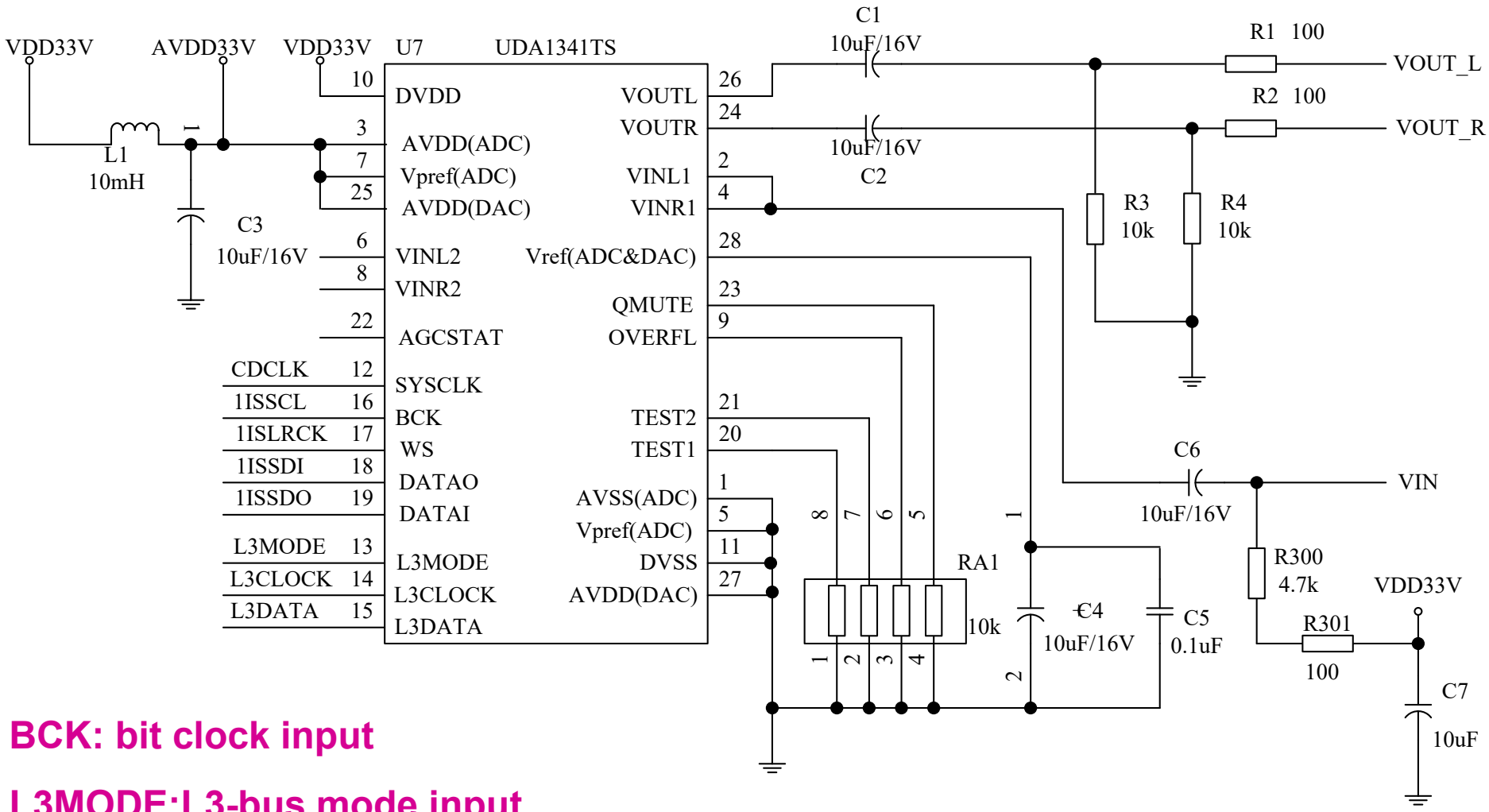


图 IIS总线与UDA1341TS的接口电路

BCK: bit clock input

L3MODE:L3-bus mode input

TEST1/2: test control 1/2 (pull-down)

QMUTE: quick mute input

OVERFL: decimation filter overflow output

- S3C2410的I2S接口线分别与UDA1341TS的I2S总线管脚BCK、WS、DATAI、DATAO、SYSCLK相连。
- 当UDA1341TS芯片工作在微处理器操作模式时，UDA1341TS的L3总线（L3DATA、L3MODE和L3CLOCK），分别与微处理器接口的数据线（L3DATA）、模式控制线（L3MODE）和时钟线（L3CLOCK）相连。微处理器通过UDA1341TS的L3总线对数字音频处理参数和系统控制参数进行配置。
- S3C2410没有与L3总线配套的专用接口，可以利用通用I/O口进行控制。

2. 音频录放的编程实例

在使用IIS总线接口发送和接收音频数据时，首先要启动IIS操作，传输数据结束后应结束IIS操作。

启动IIS操作，需要执行如下过程：

允许IISFCON寄存器的FIFO；

允许IISFCON寄存器的DMA请求；

允许IISFCON寄存器的启动。

结束IIS操作，需要执行如下过程：

禁止IISFCON寄存器的FIFO，如果还想发送FIFO的剩余数据，跳过这一步；

禁止IISFCON寄存器的DMA请求；

禁止IISFCON寄存器的启动。

【例8.3】本实例实现了对语音的实时录制和实时播放功能。该功能主要是通过函数IIS_RxTx（void）来实现，具体代码如下：

```
void IIS_RxTx ( void ) {
        unsigned int I;
        unsigned short * rxdata;
        Rx_Done=0;
        Tx_Done=0;
 //由于使用DMA方式进行语音录放，因此这里需要注册DMA中断
        PISR_DMA2= ( unsigned ) TX_Done;
        PISR_DMA1= ( unsigned ) TX_Done;
        rINTMSK & = ～( BIT_DMA1 );
        rINTMSK & = ～( BIT_DMA2 );
        rxdata= ( unsigned short * ) malloc ( 0x80000 ) ; //384 KB
        for ( i= 0 ; i< 0xffff0 ; i++ )
          * ( rxdata+ i )= 0 ;
```

```
while ( 1 ) {
        //录音过程，DMA1用于音频输入
        rDMASKTRIG1 = ( 1<<2 ) | ( 0<<1 ) ;
        //初始化DMA通道1
        rDISRC1    = ( U32 ) IISFIFO ;  //接收FIFO地址
        rDISRCC1   = ( 1<<1 ) | ( 1<<0 ) ;          //源=APB，地址固定
        rDIDSTC1   = ( 0<<1 ) | ( 0<<0 ) ;          //目录=AHB，地址增加
        rDCON1 = ( 0<<31) | (0<<30) | ( 1<<29 ) | (0<<28 ) | ( 0<<27 ) | ( 2<<24 ) |
( 1<<23 ) |          ( 1<<22 ) | ( 1<<20 ) | ( 0xffff0 ) ;
    //握手模式，与APB同步，中断使能，单元发送，单个服务模式，目标=IISSDI
    //硬件请求模式，不自动重加载，半字
        rDMASKTRIG1 = ( 1<<1 ) ;     //DMA1通道打开
        //初始化IIS，用于接收
        rIISCON  = ( 0<<5 ) | ( 1<<4 ) | ( 1<<1 ) ;
     //发送DMA请求禁止，接收DMA请求使能，IIS预分频器使能
        rIISMOD = ( 0<<8 ) | ( 1<<6 ) | ( 0<<5 ) | ( 4<<4 ) | ( 1<<3 ) | ( 0<<2 ) | (1<<0) ;
     //主模式，接收模式，IIS格式，16位，主时钟频率256fs，串行数据位时钟频
率32fs
```

```
    rIISPSR  = ( 1<<8 ) | ( 1<<3 ) ;        //预分频值A=45MHZ/8
                                            //预分频值B=45MHZ/8
    rIISFCON = ( 0<<15) | (1<<14 ) | ( 0<<13 ) | (1<<12 ) ;
//发送FIFO=正常，接收FIFO=DMA，发送FIFO禁止，接收FIFO使能
    rIISCON | = ( 1<<0 ) ;                  //IIS使能
    while (!Rx_Done) ;
    Rx_Done = 0 ;
    //IIS停止
    Delay ( 10 ) ;        //用于结束半字/字接收
    rIISCON   = 0x0 ;   //IIS停止
    rDMASKTRIG1 = ( 1<<2 ) ;  //DMA1停止
    rIISFCON    = 0x0 ;             //发送/接收FIFO禁止
    //放音过程，DMA2用于音频输出
    rDMASKTRIG2 = ( 1<<2) | ( 0<<1 ) ;
```

```
        //初始化DMA通道2
        rDISRC2  = ( U32 ) ( rxdate ) ;
        rDISRCC2 = ( 0<<1 ) | ( 0<<0 ) ; //源=AHB，地址增加
        rDIDST2  = ( U32 ) IISFIFO ;                //发送=FIFO地址
        rDIDSTC2 = ( 1<<1 ) | ( 1<< 0 ) ;           //目标=APB，地址固定
        rDCON2  = ( 0<<31 ) (0<<30 ) | ( 1<<29 ) | ( 0<<28 ) | ( 0<<27 ) | ( 0<<24 ) |
( 1<<23 ) | ( 1<<22 ) | ( 1<<20 ) | ( 0xffff0 ) ;
    //握手模式，与APB同步，中断使能，单元发送，单个服务模式，目标=IISSDO
    //硬件请求模式，不自动重加载，半字
        rDMASKTRIG1 = ( 1<<1 ) ;     //DMA2通道打开
        //初始化IIS，用于发送
        rIISCON  = ( 1<<5 ) | ( 0<<4 ) | ( 1<<1 ) ;
     //发送DMA请求使能，接收DMA请求禁止，IIS预分频器使能
        rIISMOD = ( 0<<8 ) | ( 1<<6 ) | ( 0<<4 ) | ( 1<<3 ) | ( 0<<2 ) | ( 1<<0 )
         //主模式，发送模式，IIS格式，16位，主时钟频率256fs
        //串行数据位时钟频率32fs
```

```
    rIISPSR  = ( 1<<8 ) | ( 1<<3 ) ; //预分频值A=45MHZ/8
                                          //预分频值B=45MHZ/8
    rIISFCON= ( 1<<15) | ( 0<<14 ) | ( 1<<13 ) | ( 0<<12 );
//发送FIFO=DMA，接收FIFO=正常，发送FIFO使能，接收FIFO禁止
    rIISCON |= ( 1<<0 ) ;              //IIS使能
    while (!Tx_Done) ;
    Tx_Done= 0 ;
    rIISCON  = 0x0 ;                        //IIS停止
    rDMASKTRIG2 = ( 1<<2 ) ;                //DMA2停止
    rIISFCON    = 0x0 ;              //发送/接收FIFO禁止
    }
    free ( rxdata ) ;
    rINTMSK | = ( BIT_DMA2 ) ;
    rINTMSK | = ( BIT_DMA1 ) ;
    ChangeClockDivider ( 1, 1 ) ;  //1:2:4
    ChangeMPllValue ( 0xa1, 0x3 ,0x1 );              //FCLK=202.8MHZ
    }
```

Linux环境下音频设备程序的实现(*: 了解)

对于Linux应用程序员来讲，音频编程接口实际上就是一组音频设备文件。

声卡是一个特殊的设备。声卡主要提供3个重要的特征(经常用到的设备文件):

- 数字采样输入/输出(/dev/dsp)
- 频率调制输出(/dev/sequencer)
- MIDI接口(/dev/mixer)

音频编程接口(*: 了解)

首先使用open打开设备文件；接着使用read系统调用从设备接收数据，或者使用write向设备写入数据，其他不属于IO的操作，由ioctl来完成；最后，使用close关闭设备。

1．open系统调用：open可以获得对声卡的访问权，同时还能为随后的操作做好准备

2．read系统调用：read用来从声卡读取数据

3．write系统调用：write用来向声卡写入数据

4．ioctl系统调用：设备文件的非IO操作(即I/O之外的操作)，都是通过ioctl来完成的，可用来向设备传递控制信息或从设备获取状态信息

5．close系统调用：使用完之后，用close将其关闭

音频设备文件(*: 了解)

- /dev/sndstat

  该设备文件用于报告声卡的状态。

- /dev/dsp

  该设备文件用于采样或数字录音。

- /dev/audio

  类似于/dev/dsp，它兼容SUN工作站上的音频设备文件。同一时刻只能使用/dev/dsp或 /dev/audio之一，因为它们是相同硬件的不同软件接口。

- /dev/mixer

  该设备文件用于声卡内建的波表合成器进行操作或对MIDI上的乐器进行控制。

- /dev/sequencer

  该设备文件是应用程序对混音器进行操作的软件接口。

音频设备编程设计(*: 了解)

在Linux下进行音频编程时，重点在于如何正确地操作声卡驱动程序所提供的各种设备文件。下面是两种简单的基于通用框架的音频编程方案(参考于明编：ARM9嵌入式系统设计与开发教程，电子工业出版社，2006)。

1．DSP编程

打开设备

设置缓冲区

设置声道数

读写设备

2．Mixer编程

在编写实用的音频程序时，混音器是在涉及兼容性时需要重点考虑的一个对象，这是因为不同的声卡所提供的混音器资源是有所区别的。

声卡驱动程序提供了多个ioctl系统调用来获得混音器的信息，它们通常返回一个整型的位掩码（bitmask），其中每一位分别代表一个特定的混音通道，如果相应的位为1，则说明与之对应的混音通道是可用的。

例如，通过SOUND_ MIXER_READ_DEVMASK返回的位掩码，可以查询出能够被声卡支持的每一个混音通道，而通过SOUND_MIXER_READ_RECMAS返回的位掩码，则可以查询出能够被当做录音源的每一个通道。

下面的代码可以用来检查CD输入是否是一个有效的混音通道。

```
ioctl(fd, SOUND_MIXER_READ_DEVMASK, &devmask);
if (devmask & SOUND_MIXER_CD) printf(“The CD input is supported”);
```

如果进一步还想知道其是不是一个有效的录音源，则可以使用如下函数：

```
ioctl(fd, SOUND_MIXER_READ_RECMASK, &recmask);
if (recmask & SOUND_MIXER_CD) printf("The CD input can be a recording source");
```

目前，大多数声卡提供多个录音源，通过SOUND_MIXER_READ_RECSRC可以查询出当前正在使用的录音源，同一时刻能够使用几个录音源是由声卡硬件决定的。类似地，使用SOUND_MIXER_WRITE_RECSRC可以设置声卡当前使用的录音源，例如，下面的代码可以将CD输入作为声卡的录音源使用。

```
devmask = SOUND_MIXER_CD;
ioctl(fd, SOUND_MIXER_WRITE_RECSRC, &devmask);
```

此外，所有的混音通道都有单声道和双声道的区别，如果需要知道哪些混音通道提供了对立体声的支持，可以通过SOUND_MIXER_READ_STEREODEVS来获取。

综合训练之媒体播放器移植(*:结合实验学习)

- MPlayer是Linux下强大的媒体播放器，对媒体格式广泛支持，最新的版本可以支持Divx、H.264、MPEG4等最新的媒体格式，可以实时在线播放视频流，是目前嵌入式媒体播放器的首选。

- MPlayer支持相当多的媒体格式，无论是在音频播放方面还是在视频播放方面，可以说它支持的格式是相当全面的。它可以支持MPEG、AVI、ASF与WMV、QuickTime与OGG/OGM、SDP、PVA、GIF等视频格式，以及MP3、WAV、OGG/OGM文件(Vorbis)、WMA与ASF、MP4、CD音频、XMMS等音频格式。

- MPlayer支持广泛的输出设备，可以在X11、Xv、DGA、OpenGL、SVGAlib、fbdev、AAlib、DirectFB下工作，而且也能支持GGI、SDL(由此可以使用它们支持的各种驱动模式)和一些低级的硬件相关的驱动模式(比如，Matrox、3Dfx和RADEON、Mach64、Permedia3)。同时，MPlayer还支持通过硬件MPEG解码卡显示，例如，DVB和DXR3与Hollywood+。

1．安装和编译

MPlayer的源代码可以从其主页http://www.mplayerhq.hu下载。最新的文件名是MPlayer-current.tar.bz2。对下载的文件解压缩：

$tar zxvf MPlayer-current.tar.bz2

在解压缩得到的MPlayer-0.93目录下有一个脚本文件mkall，这个文件是一个编译脚本，在该目录下直接执行：

$ ./ mkall

该脚本将配置并编译mplayer，下面是该脚本所进行的配置和编译命令：

./configure --cc=usr/local/arm/2.95.3/bin/arm-linux-gcc

--target=arm-linux –with-extralibdir=/usr/local/arm/2.95.3/arm-linux/lib

–with-extraincdir=/usr/ local/arm/2.95.3/arm-linux/include/--disable-sdl

–enable-static –disable- dvdnav –disable-tv –disable-gui –disable-mpdvdkit

–enable-linux-devfs

make

编译成功后，将在MPlayer-0.93目录下生成mplayer文件，该文件为mplayer媒体播放程序。

2．下载运行

启动目标板后，以网络移动的方式下载应用程序。首先把mplayer复制到ftp共享目录，在PC端执行：

#cp mplayer /home/ftp

在SBC-2410端进入bin目录，并登录ftp服务器：

#cd /bin

#ftp 192.168.0.1

然后下载mplayer和测试视频文件，并修改mplayer可执行权限：

>get mplayer

>get test.mpeg

>bye

#chmod a+x mplayer

最后在命令行下输入如下命令：

#mplayer –vo test.mpeg

课外作业

3、简述IIS启动操作和结束操作的过程。

End of Chapter 8