```
    INTC_Enable(NUM_AC97);
    AC97_EnableInt(CODEC_READY_INT);//AC97 启动
    while(!uCodecReadyIrq)
    {
        Disp(".");
            Delay(3000);
            i++;
            if(i==20)
        break;
    }
    Disp("\n");
    if(i>=20)
    {
        Disp("\nAC97 codec is not ready.");
        Disp("\nCheck on connection between AP and AC97 CODEC.\n");
        Disp("\nBye. ");
        return false;
    }
    else
    {
        return true;
    }
}
```

AC97 控制器冷复位的代码如下：

```
//AC97 冷复位的一段程序
void AC97_ColdReset(void)
{
    u32 uGlobalCon;
    Disp("\n=>Cold Reset\n");
```

```
    //uGlobalCon = Inp32(AC97_BASE+rACGLBCTRL);
    uGlobalCon = (1<<0);
    AC97Outp32(rACGLBCTRL, uGlobalCon);
    Delay(1000);
    uGlobalCon &= ~(1<<0);
    AC97Outp32(rACGLBCTRL, uGlobalCon);
    Delay(1000);
    uGlobalCon = (1<<0);
    AC97Outp32(rACGLBCTRL, uGlobalCon);
    Delay(1000);
    uGlobalCon &= ~(1<<0);
    AC97Outp32(rACGLBCTRL, uGlobalCon);
    Delay(1000);
    AC97_ControllerState();
    #if (AC97_CODEC_NAME== WM9713)
        AC97_WarmReset();
    #endif
    uGlobalCon |= (1<<2);
    AC97Outp32(rACGLBCTRL, uGlobalCon);
    AC97_ControllerState();
    uGlobalCon |= (1<<3);
    AC97Outp32(rACGLBCTRL, uGlobalCon);
    AC97_ControllerState();
  //Disp("AC97 Codec Powerdown Ctrl/Stat Reg. Value (at 0x26): 0x%x\n",
  AC97_CodecCmd(READ,0x26,0x0000));
}
```

确保 GPIO 已经准备就绪。然后允许多媒体数字信号编解码器有准备中断信号。通过轮询和中断检查多媒体数字信号编解码器准备中断信号。当发现中断信号，必须声明多媒体数字信号编解码器准备中断信号。利用DMA或PIO能够立即从内存到寄存器或从寄存器到内存传输数据。如果内部FIFO(TX FIFO或RX FIFO)

不为空，则传输数据。此外，还可以在 AC-link 上转换以前的 FIFO。如图 35-3 所示，显示 AC97 操作流程图。

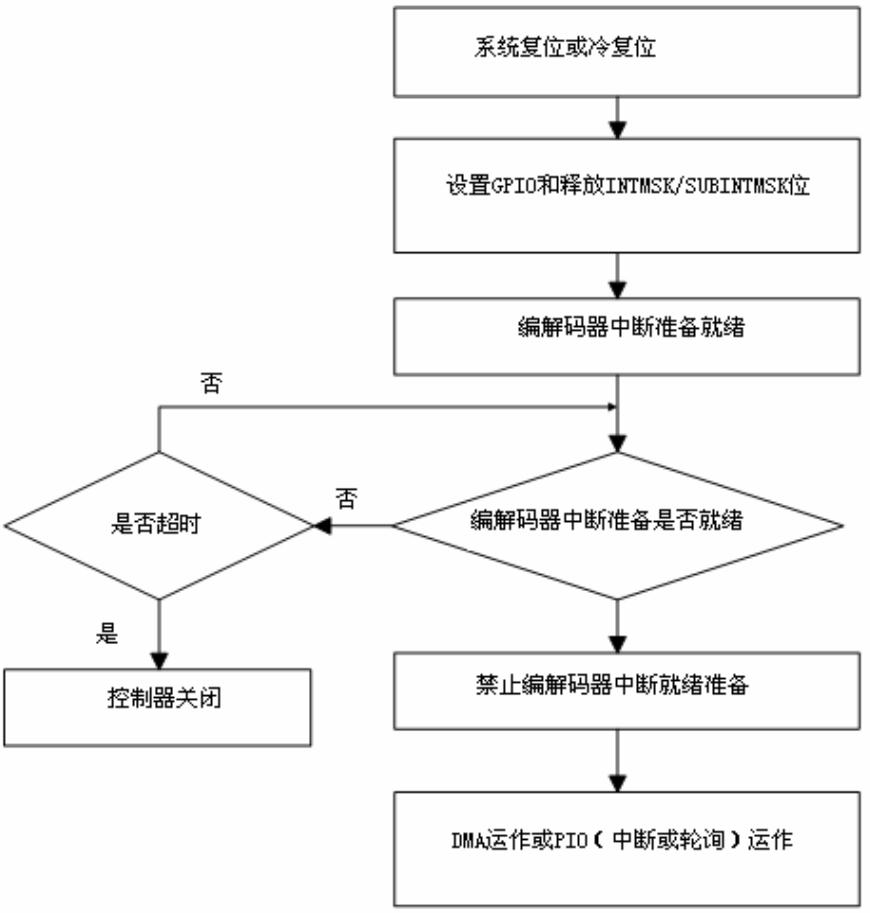

图 35-3　AC97 操作流程图

下面是 ARM11 中 AC97 控制器状态控制代码部分，AC97 控制器状态分为启动状态和禁止状态两种，具体的代码实现如下：

```
//AC97 控制器状态定义
void AC97_ControllerState(void)
{
    u32 uState;
    uState= Inp32(AC97_BASE+rACGLBSTAT);
    if((uState & 0x7) == 0)
```

```
        Disp("AC97 Controller State: Idle\n");
    else if ((uState & 0x7) == 1)
        Disp("AC97 Controller State: Init\n");
    else if ((uState & 0x7) == 2)
        Disp("AC97 Controller State: Ready\n");
    else if ((uState & 0x7) == 3)
        Disp("AC97 Controller State: Active\n");
    else if ((uState & 0x7) == 4)
        Disp("AC97 Controller State: LP\n");
    else if ((uState & 0x7) == 5)
        Disp("AC97 Controller State: Warm\n");
}
//AC97 控制器状态-启动状态
void AC97_EnableInt(AC97_INT eInt)
{
    u32 uInt;
    uInt= Inp32(AC97_BASE+rACGLBCTRL);
    if(eInt == CODEC_READY_INT)
        uInt |= 1<<22;
    else if (eInt == PCMOUT_UNDERRUN_INT)
        uInt |= 1<<21;
    else if (eInt == PCMIN_OVERRUN_INT)
        uInt |= 1<<20;
    else if (eInt == MICIN_OVERRUN_INT)
        uInt |= 1<<19;
    else if (eInt == PCMOUT_THRESHOLD_INT)
        uInt |= 1<<18;
    else if (eInt == PCMIN_THRESHOLD_INT)
        uInt |= 1<<17;
```

```
    else if (eInt == MICIN_THRESHOLD_INT)
        uInt |= 1<<16;
    //Disp("rACGLBCTRL: 0x%x\n", uInt);
    AC97Outp32(rACGLBCTRL, uInt);
}
//AC97 控制器状态-禁止状态
void AC97_DisableInt(AC97_INT eInt)
{
    u32 uInt;

    uInt= Inp32(AC97_BASE+rACGLBCTRL);
    if(eInt == CODEC_READY_INT)
        AC97Outp32(rACGLBCTRL, uInt & ~(1<<22));
    else if (eInt == PCMOUT_UNDERRUN_INT)
        AC97Outp32(rACGLBCTRL, uInt & ~(1<<21));
    else if (eInt == PCMIN_OVERRUN_INT)
        AC97Outp32(rACGLBCTRL, uInt & ~(1<<20));
    else if (eInt == MICIN_OVERRUN_INT)
        AC97Outp32(rACGLBCTRL, uInt & ~(1<<19));
    else if (eInt == PCMOUT_THRESHOLD_INT)
        AC97Outp32(rACGLBCTRL, uInt & ~(1<<18));
    else if (eInt == PCMIN_THRESHOLD_INT)
        AC97Outp32(rACGLBCTRL, uInt & ~(1<<17));
    else if (eInt == MICIN_THRESHOLD_INT)
        AC97Outp32(rACGLBCTRL, uInt & ~(1<<16));
}
```

35.3 AC-LINK 数字接口协议

每个 AC97 多媒体数字信号编解码器合并了一个 5 引脚数字串行接口，它连接到 S3C6410 AC97 控制器。

AC-link 是一个全双工，固定时钟和 PCM 数字流。它采用一个时间分割多元来配置处理控制寄存器的通道和多重输入输出音频流。AC-link 结构划分每个音频结构为 12 个输出和 12 引入数据流。每个流有 29 位样本分辨率，需要一个 DAC 和一个有最低 16 位分辨率的模数转换器（ADC）。

支持 S3C6410 AC97 控制器的插槽定义。该 S3C6410 AC97 控制器在 AC-link 上提供同步的所有数据处理。如图所示，如图 35-4 所示，显示带有插槽转让的双向 AC-link。

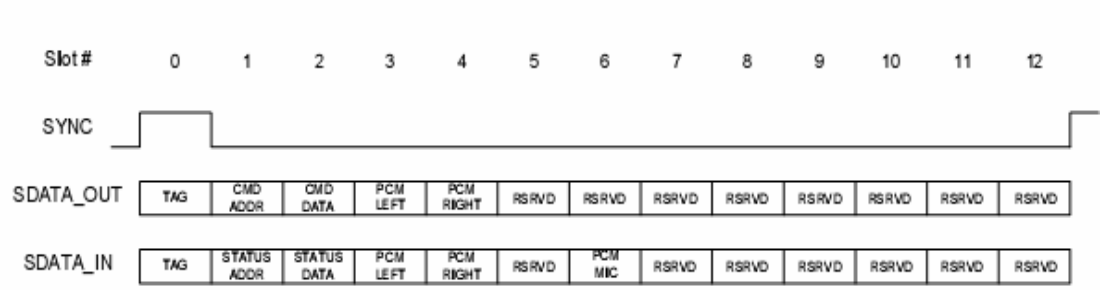

图 35-4 带有插槽转让的双向 AC-link

数据处理由 256 个信息位和 13 时段组成，即称为帧。时段 0 被称为标记相位，它为 16 位长。其它 12 个时段被称为数据相位。标记相位包含一个位，用来确定有效帧，和在数据相位确认时段的 12 位，包含有效数据。在数据相位每个时段长为 20 位。当 SYNC（同步）变高时，一个帧开始。同步是高位的时段的数量，符合标记相位。AC97 帧发生在固定的 48kHz 间隔，并同步到 12.288MHz 位时钟，BITCLK。当发送传输数据和接受数据时，控制器和媒体数字信号编解码器用于 SYNC 和 BITCLK 的测定。 在每个 BITCLK 的上升边缘，发送器转换串行数据流，在 BIRCLK 的下降边缘，接收器取样串行数据流。

在其串行数据流中，变送器必须标记有效时段。有效插槽标记在插槽 0 中 。在 AC-link 上的串行数据被排序从最高有效位（MSB）至最低有效位（LSB）。标记相位的首位是位 15，在数据状态里的各个插槽首位是位 19 。任何插槽的最后位是位 0.

1.AC-LINK 的输出帧（SDATA_OUT）

（1）插槽 0 ：标记相位

在插槽 0 中，第一位是一个位（SDAT_OUT，位 15），其代表全部帧的有效。如果位 15 是 1，当前帧至少包含一个有效时段。下一个 12 位的位置符合每 12 时段，包含有效数据。为了编解码器寄存器的 I/O 读和写的描述，在下一项中插槽 0 的位 0 和位 1 被用于编解码器 IO 位。在这种方式中，不同采样速率的数据流可以越过 AC-link 在其固定的 48kHz 音频帧速率上被传输。

（2）插槽 1 ：指令地址端口

在插槽 1 中，它传达控制寄存器地址和写/读指令信息到 AC97 控制器。

当软件访问主要编解码器时，硬件配置帧如下：

- 在插槽 0 中，设置用于插槽 1，2 的有效位。
- 在插槽 1 中，位 19 被设置（读）或清除（写）。配置位 18-12（插槽 1 的）来指定编解码寄存器。其它的均为 0（保留）。
- 在插槽 2 中，由于输出 FRAME，用写入的数据来配置它。

（3）插槽 2 ：指令数据端口

在插槽 2 中，写入以 16 位分辨率的数据。([19:4]是有效数据)

（4）插槽 3 ：PCM 重放左声道

插槽 3 是音频输出帧，是综合数字音频左流。如果采样的分辨率少于 16 位，在插槽为 0 中，AC97 控制器供应所有培训的非有效位的位置。

（5）插槽 4 ：PCM 重放右声道

插槽 3 是音频输出帧，是综合数字音频右流。如果采样的分辨率少于 16 位，在插槽为 0 中，AC97 控制器供应所有训练的非有效位的位置。如图 35-5（a）所示，显示 AC97 控制器输出结构。

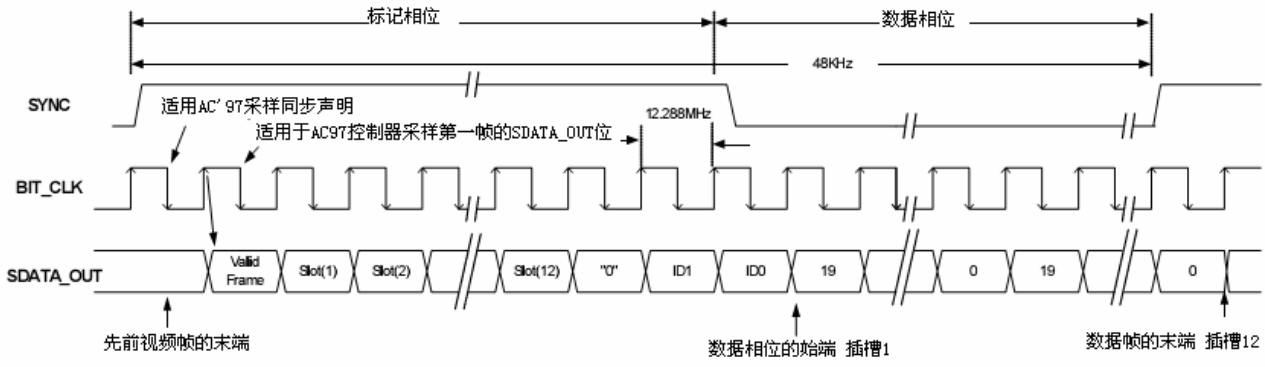

图 35-5（a） AC97 控制器输出结构

2.AC-LINK 的输入帧(SDATA_IN)

（1）插槽 0 ：标记相位

在插槽 0 中，第一位（SDATA_OUT, 位 15）表示在多媒体数字信号编解码器中 AC97 控制器是否准备就绪状态。如果多媒体数字信号编解码器准备位置为 0，这意味着该 AC97 控制器没有准备好正常工作。有效复位后的电源和 AC97 控制器的电压是稳定的，这种情况是正常的。

（2）插槽 1 ：状态地址 Port/SLOTREQ 位（端口/插槽）

状态端口监视用于 AC97 控制器功能的状态。它不是限制混音器设置和状态管理。如果控制器标记符插槽 1 和 2 有效期间为插槽 0，音频输入帧插 1 秒流回音控制寄存器指数是为数据返回在插槽 2 。如果伴随状态的地址匹配与最后有效的地址发出的命令和最近读取命令之间时，该控制器只接受状态数据。对于多种取样速率输出，编解码器检查它的取样速率控制寄存器、FIFO 的状态，并且在每一个音频输出帧引入 SDATA_OUT 标志位来确定哪个 SLOTREQ 位设置为有效。在当前音频输入帧指定输出通道请求数据期间，SLOTREQ 位有效。对于固定的 48kHz 的操作，SLOTREQ 位设置位有效，并且样品被传送。对于多种取样速率，每个输入通道的“标签”位指示数据是否有效。如表 35-2 所示显示标记位。

表 35-2　显示标记位

位	描述
19	保留（填充 0）
18-12	控制寄存器指数（填充 0 ，如果 AC97 标记是无效）
11	插槽 3 请求：左声道的 PCM
10	插槽 4 请求：右声道的 PCM
9	插槽 5 请求：不可用
8	插槽 6 请求：MIC 通道
7	插槽 7 请求：不可用
6	插槽 8 请求：不可用
5	插槽 9 请求：不可用
4	插槽 10 请求：不可用
3	插槽 11 请求：不可用
2	插槽 12 请求：不可用
1、0	保留（填充 0）

（3）插槽 2 ：状态数据端口

插槽 2 带有 16 位分辨率的状态数据。([19:4]有效数据)。

（4）插槽 3：左声道的 PCM

插槽 3 是音频输入结构左声道音频输出的 AC97 编解码器。如果样品的分辨率少于 16 位，AC97 编解码器填充所有的培训（training）的非有效位安置插槽 0 内。

（5）插槽 4：右声道的 PCM

插槽 4 是音频输入结构右声道音频输出的 AC97 编解码器。如果样品的分辨率少于 16 位，在插槽 0 中 AC97 编解码器填补了所有的训练非有效位位置。

（6）插槽 6 ：麦克风记录数据

该 AC97 控制器只支持 16 位分辨率，用于麦克风接口通道。如图 35-5（b）所示，显示 AC97 控制器输入结构。

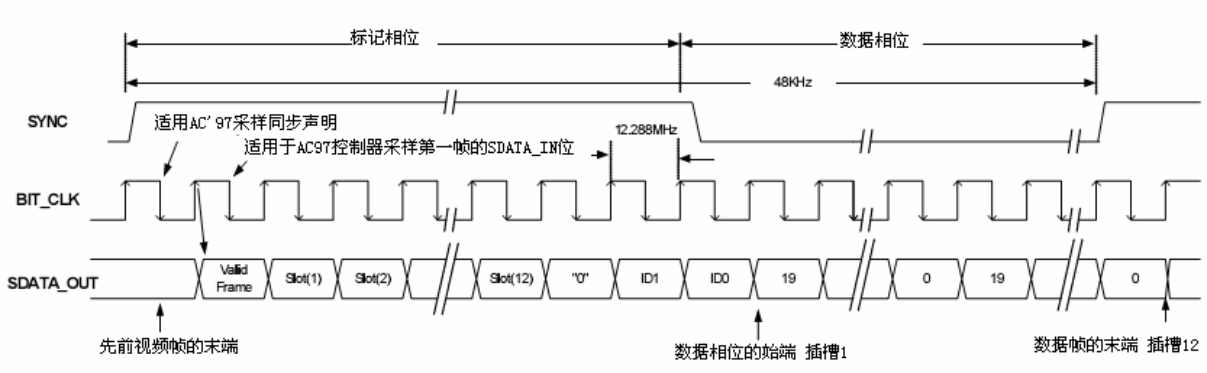

图 35-5（b）AC97 控制器输入结构

35.4 AC97 掉电模式

1.AC-link 电源关闭

当 AC97 编解码器的电源关闭寄存器（0x26）PR4 设置为 1（写入 0x1000），AC-link 信号进入一个低功耗模式。主要的编解码器驱动 BITCLK 和 SDATA_IN 入逻辑低电平的水平。序列时序图，如图 35-6 所示。

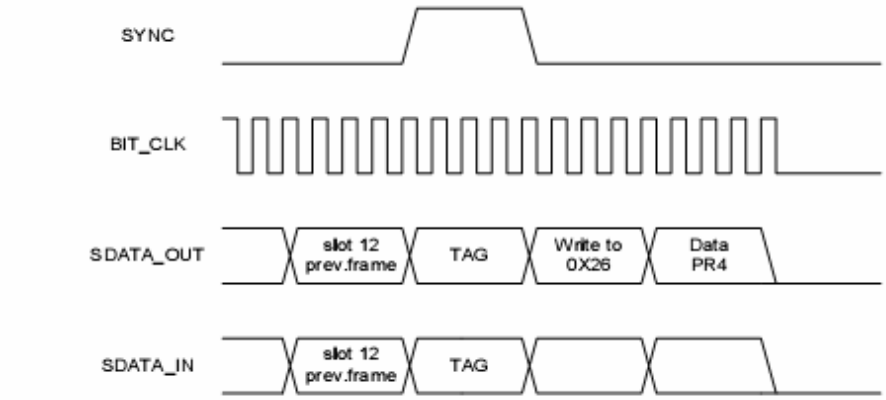

图 35-6　AC97 电源关闭时间

该 AC97 控制器的传输越过 AC-link 写入电源关闭寄存器（0x26）。建立 AC97 控制器是为了当它写入关闭电源寄存器位 PR4 (data 0x1000)时，它不传送数据到插槽 3-12 的位置，并当收到关闭电源请求时，它不需要编解码器处理其它数据。当编解码器处理请求时，它立即转换 BITCLK 和 SDATA_IN 为逻辑低水平。设计 AC_GLBCTRL 寄存器后，改变 AC97 控制器驱动 SYNC 和 SDATA_OUT 为逻辑低水平。

2.唤醒 AC-link——通过 AC97 控制器触发唤醒

AC-link 协议提供了一个 AC97 冷复位和一个 AC97 热复位。目前省电状态最终发送 AC97 复位指令。在所有掉电模式下，寄存器必须停在同一个状态，除非执行一个 AC97 冷复位。在一个 AC97 冷复位后，AC97 被初始化为它的默认值。关闭电源后，AC—Link 必须等待该帧后的四个音频帧中的一个最小值。该帧被 SYNC 信号恢复前掉电。AC—Link 上电后，它通过编解码准备位（输入插槽 0，位 15）指示读准备就绪。如图 35-7 所示，显示 AC97 关闭/启动电源流。

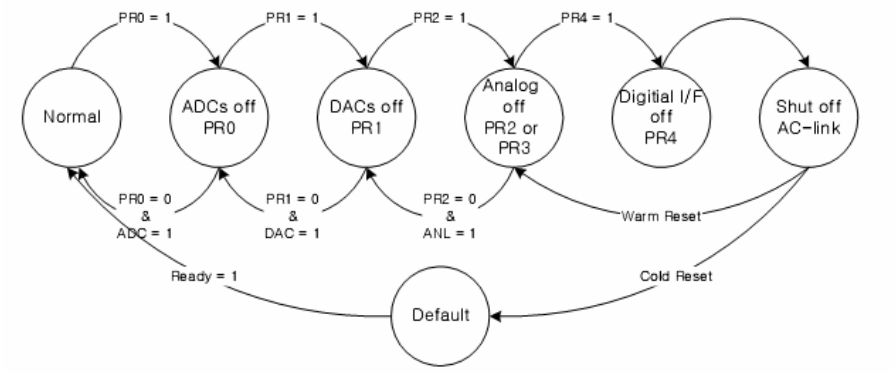

图 35-7　AC97 关闭/启动电源流

3. AC97 冷复位

当通过 AC_CLBCTRL，nRESET 引脚是有效时，产生一个冷复位。有效和无效 nRESET 启动 BITCLK 和 SDATA_OUT。所有 AC97 控制寄存器都被初始化为它们默认启动复位值。nRESET 是一个异步 AC97 输入。

4.AC97 热复位

AC97 热复位重新启动 AC-link，不需要改变当前 AC97 寄存器的值。当缺少 BITCLK 和高驱动 SYNC 时会生成一个热复位。在正规的音频结构中，SYNC 是一个同步的 AC97 输入。当 BITCLK 不存在，SYNC 被视为一个异步输入，用来产生 AC97 热复位。该 AC97 控制器不能启动 BITCLK，直到采样 SYNC 再次为低电平。