

3. 14 扬声器播放

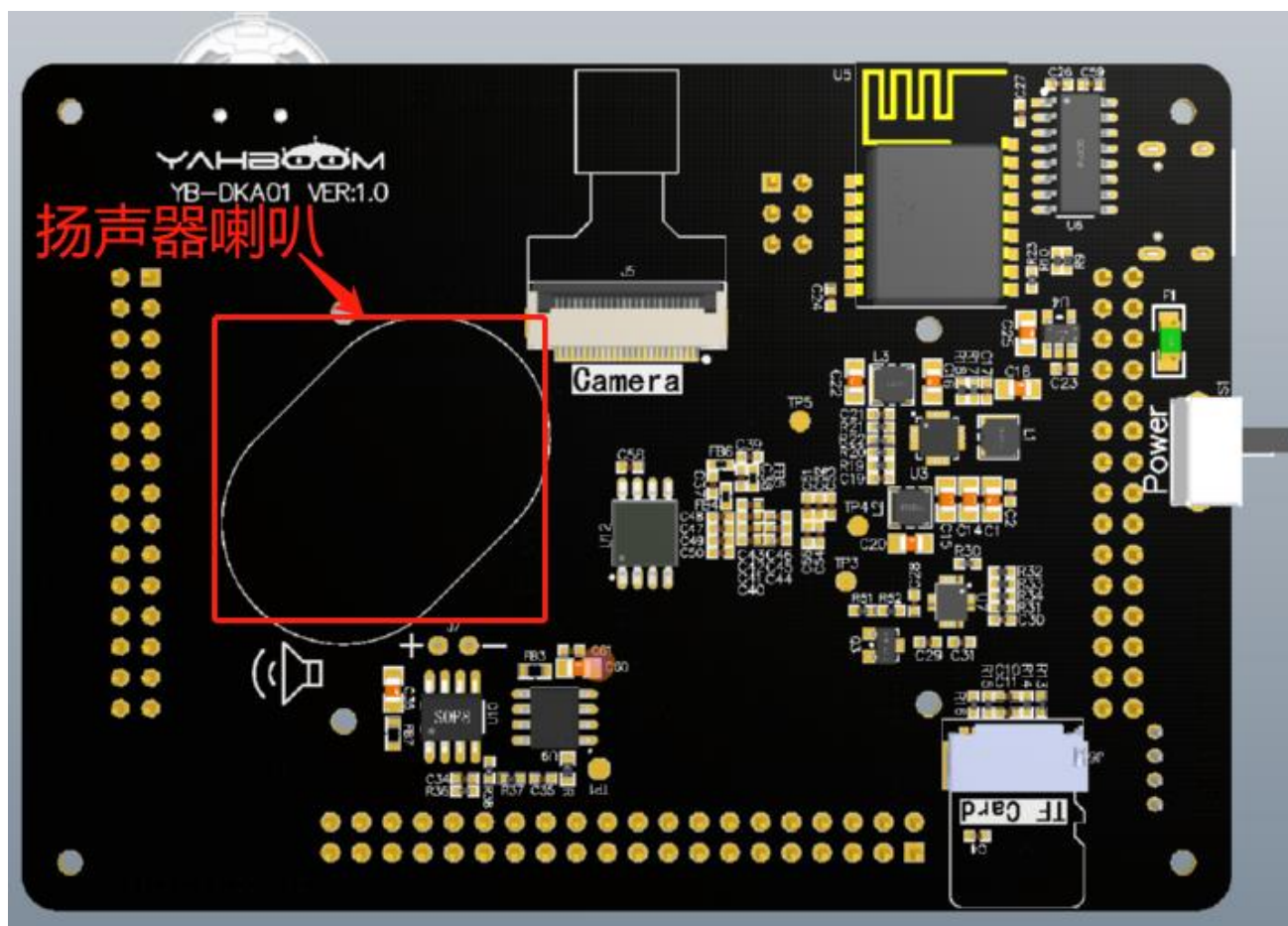
一、实验目的

本节课主要学习 K210 通过 I2S 播放 PCM 音频数据的功能。

二、实验准备

1. 实验元件

扬声器



2. 元件特性

K210 开发板默认已经焊接好功放和扬声器，功放的作用是把音源的弱信号放大，然后推动扬声器发声。而功放是通过 I2S 协议与 K210 芯片通讯的。

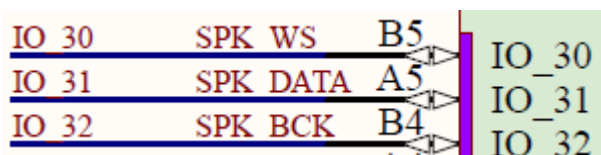
I2S 总线英文简称 Inter-IC Sound，是飞利浦公司为数字音频设备之间的音频数据传输而制定的一种总线标准，该总线专门用于音频设备之间的数据传输，广泛应用于各种多媒体系统。I2S 也是同步串行通信接口。

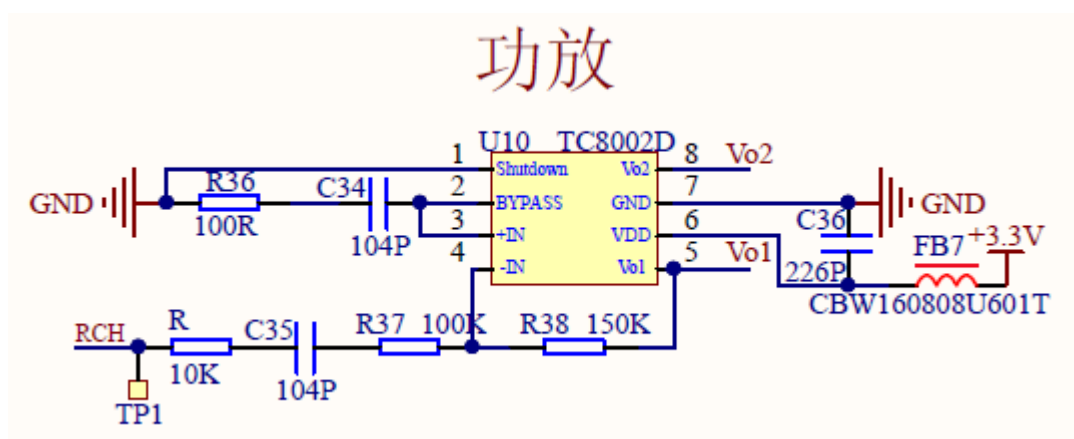
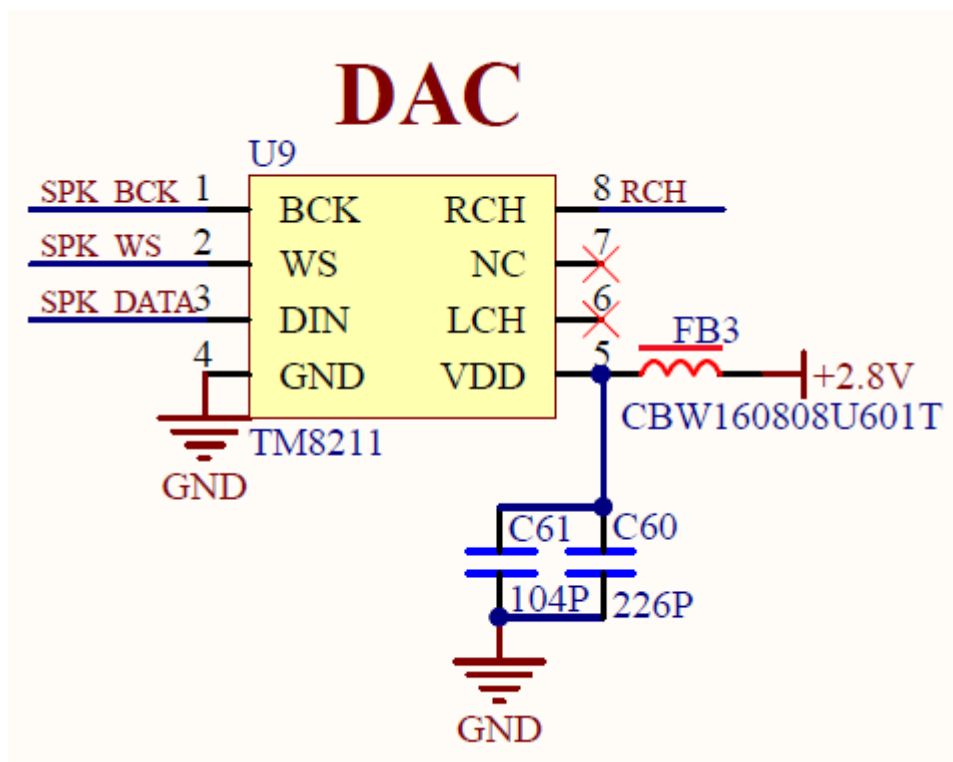
集成电路内置音频总线共有 3 个(I²S0、I²S1、I²S2)，都是 MASTER 模式。其中 I²S0 支持可配置连接语音处理模块，实现语音增强和声源定向的功能。下面是一些共有的特性：

- 总线宽度可配置为 8, 16, 和 32 位
- 每个接口最多支持 4 个立体声通道
- 由于发送器和接收器的独立性，所以支持全双工通讯
- APB 总线和 I²S SCLK 的异步时钟
- 音频数据分辨率为 12, 16, 20, 24 和 32 位
- I²S0 发送 FIFO 深度为 64 字节，接收为 8 字节，I²S1 和 I²S2 的发送和接收 FIFO 深度都为 8 字节
- 支持 DMA 传输
- 可编程 FIFO 阈值

3. 硬件连接

K210 开发板出厂默认已经焊接扬声器及相关配件，其中 SPK_WS 连接到 IO30，SPK_DATA 连接到 IO31，SPK_BCK 连接到 IO32。





4. SDK 中对应 API 功能

对应的头文件 i2s.h

I2S 标准总线定义了三种信号：时钟信号 BCK、声道选择信号 WS 和串行数据信号 SD。一个基本的 I2S 数据总线有一个主机和一个从机。主机和从机的角色在通信过程中保持不变。I2S 模块包含独立的发送和接收声道，能够保证优良的通信性能。

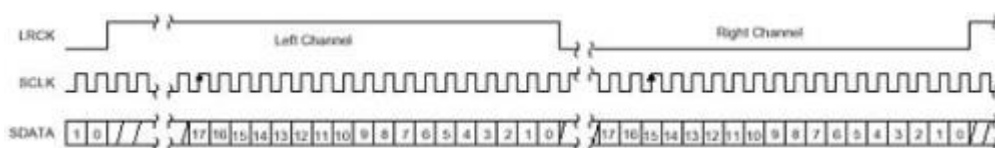
为用户提供以下接口：

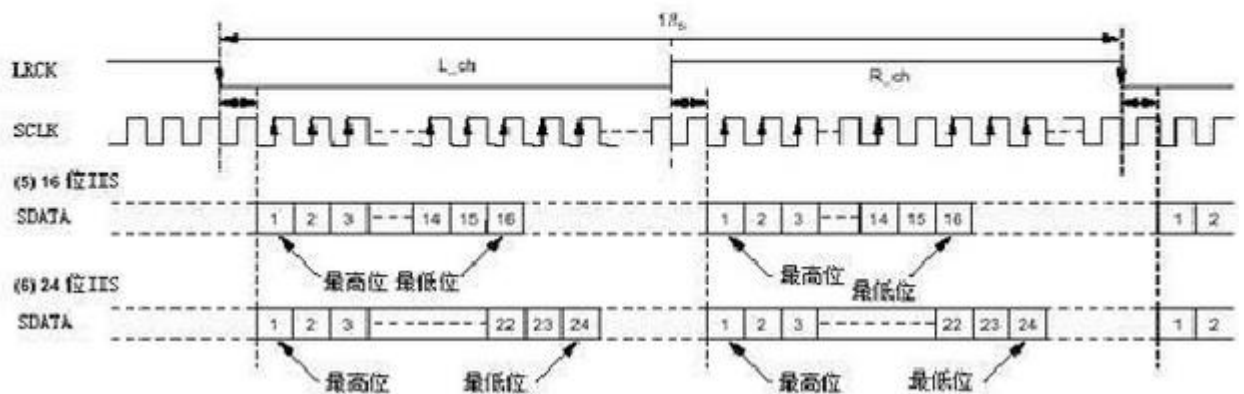
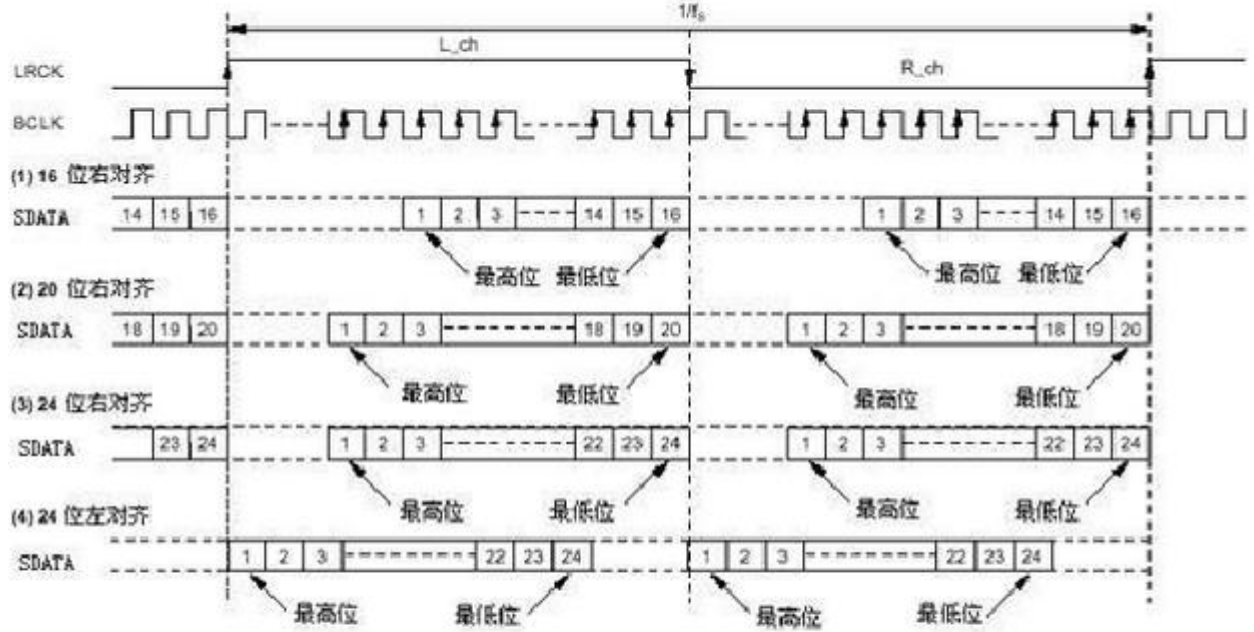
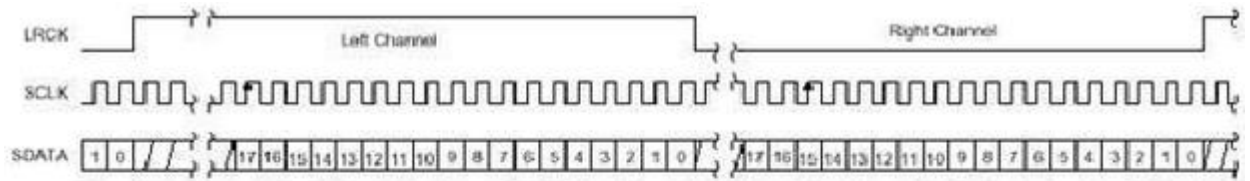
- i2s_init: 初始化 I2S 设备，设置接收或发送模式，通道掩码。
- i2s_send_data_dma: I2S 发送数据。
- i2s_recv_data_dma: I2S 接收数据。
- i2s_rx_channel_config: 设置接收通道参数。
- i2s_tx_channel_config: 设置发送通道参数。
- i2s_play: 发送 PCM 数据，比如播放音乐
- i2s_set_sample_rate: 设置采样率。
- i2s_set_dma_divide_16: 设置 dma_divide_16, 16 位数据时设置 dma_divide_16, DMA 传输时自动将 32 比特 INT32 数据分成两个 16 比特的左右声道数据。
- i2s_get_dma_divide_16: 获取 dma_divide_16 值。用于判断是否需要设置 dma_divide_16。
- i2s_handle_data_dma: I2S 通过 DMA 传输数据。

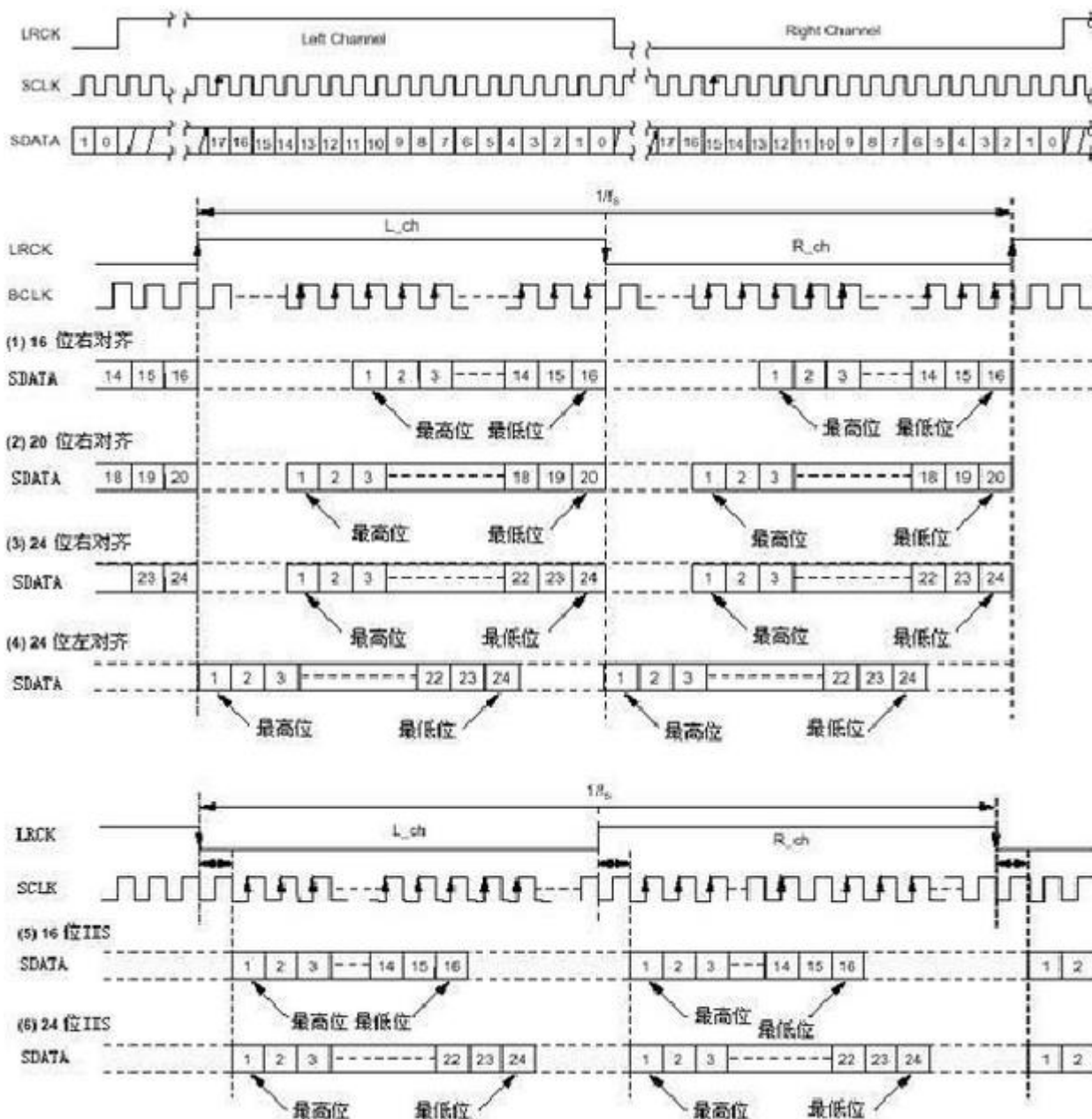
三、实验原理

I2S 有 3 个主要信号：

1. 串行时钟 SCLK，也叫位时钟 (BCLK)，即对应数字音频的每一位数据，SCLK 都有 1 个脉冲。SCLK 的频率=2×采样频率×采样位数
2. 帧时钟 LRCK，用于切换左右声道的数据。LRCK 为“1”表示正在传输的是左声道的数据，为“0”则表示正在传输的是右声道的数据。LRCK 的频率等于采样频率。
3. 串行数据 SDATA，就是用二进制补码表示的音频数据。







I2S 格式的信号无论有多少位有效数据，数据的最高位总是出现在 LRCK 变化（也就是一帧开始）后的第 2 个 SCLK 脉冲处（如上）。这就使得接收端与发送端的有效位数可以不同。如果接收端能处理的有效位数少于发送端，可以放弃数据帧中多余的低位数据；如果接收端能处理的有效位数多于发送端，可以自行补足剩余的位。这种同步机制使得数字音频设备的互连更加方便，而且不会造成数据错位。对于 MCLK 是主时钟或系统时钟，能够保证两个芯片之间同步。

DAC 芯片将 I2S 信号转化成 RCH 信号输送给功放，功放将 RCH 信号发大，然后扬声器播放出对应的声音。

四、实验过程

1. 首先初始化 K210 的硬件引脚和软件功能使用的是 FPIOA 映射关系。

```
/******HARDWARE-PIN*****  
// 硬件IO口，与原理图对应  
#define PIN_SPK_WS          (30)  
#define PIN_SPK_DATA        (31)  
#define PIN_SPK_BCK         (32)  
  
/******SOFTWARE-GPIO*****  
// 软件GPIO口，与程序对应  
  
/******FUNC-GPIO*****  
// GPIO口的功能，绑定到硬件IO口  
#define FUNC_SPK_WS          FUNC_I2S2_WS  
#define FUNC_SPK_DATA        FUNC_I2S2_OUT_D0  
#define FUNC_SPK_BCK         FUNC_I2S2_SCLK
```

```
void hardware_init(void)  
{  
    /*  
    ** SPK_WS---IO30  
    ** SPK_DATA-IO31  
    ** SPK_BCK--IO32  
    */  
    fpioa_set_function(PIN_SPK_WS, FUNC_SPK_WS);  
    fpioa_set_function(PIN_SPK_DATA, FUNC_SPK_DATA);  
    fpioa_set_function(PIN_SPK_BCK, FUNC_SPK_BCK);  
}
```

2. 设置系统时钟频率，由于 uarths 的时钟来自 PLL0，所以设置 PLL0 之后需要重新初始化以下 uarths，否则 printf 可能会打印乱码。

```
/* 设置系统时钟 */
sysctl_pll_set_freq(SYSCTL_PLL0, 32000000UL);
sysctl_pll_set_freq(SYSCTL_PLL1, 16000000UL);
sysctl_pll_set_freq(SYSCTL_PLL2, 45158400UL);
uarths_init();
```

3. 初始化 I2S 设备 2 为发送模式，注意通道掩码对应通道 0:0x03, 通道 1: 0x0C, 通道 2: 0x30, 通道 3:0xC0,。

```
/* 初始化I2S，第三个参数为设置通道掩码，通道0:0x03,通道1: 0x0C,通道2: 0x30,通道3:0xC0 */
i2s_init(I2S_DEVICE_2, I2S_TRANSMITTER, 0x03);
```

4. 设置 I2S 发送数据通道 0 的参数，接收数据位 16bit, 单个时钟数据位 32, DMA 触发 FIFO 深度为 4, 工作模式为右声道。

```
/* 设置I2S发送数据的通道参数 */
i2s_tx_channel_config(
    I2S_DEVICE_2, /* I2S设备号*/
    I2S_CHANNEL_0, /* I2S通道 */
    RESOLUTION_16_BIT, /* 接收数据位数 */
    SCLK_CYCLES_32, /* 单个数据时钟数 */
    TRIGGER_LEVEL_4, /* DMA触发时FIFO深度 */
    RIGHT_JUSTIFYING_MODE); /* 工作模式 */
```

5. I2S 播放音乐，播放音乐需要使用 DMA 通道，只要选择一个空闲的 DMA 通道即可，PCM 数据可以通过 wav 音乐文件转化。

```
/* 播放一次音乐，如果需要重新播放请按复位键 */
i2s_play(
    I2S_DEVICE_2, /* I2S设备号 */
    DMAC_CHANNEL0, /* DMA通道号 */
    (uint8_t *)test_pcm, /* 播放的PCM数据 */
    sizeof(test_pcm), /* PCM数据的长度 */
    1024, /* 单次发送数量 */
    16, /* 单次采样位宽 */
    2); /* 声道数 */
```

6. 最后是一个 while(1) 循环，不让系统退出。如果需要重新播放音乐，请按一下开发板的复位键。


```
while (1);  
return 0;
```

7. 编译调试，烧录运行

把本课程资料中的 play_pcm 复制到 SDK 中的 src 目录下，然后进入 build 目录，运行以下命令编译。

```
cmake .. -DPROJ=play_pcm -G "MinGW Makefiles"
```

```
make
```

```
[ 97%] Building C object CMakeFiles/play_pcm.dir/src/play_pcm/main.c.obj  
[100%] Linking C executable play_pcm  
Generating .bin file ...  
[100%] Built target play_pcm  
PS C:\K210\SDK\kendryte-standalone-sdk-develop\build> []
```

编译完成后，在 build 文件夹下会生成 play_pcm.bin 文件。

使用 type-C 数据线连接电脑与 K210 开发板，打开 kflash，选择对应的设备，再将程序固件烧录到 K210 开发板上。

五、实验现象

K210 开发板播放一次音乐，最后停止，当需要再次播放音乐时，请按一下 K210 开发板的复位键。

六、实验总结

1. 扬声器不是直接接收 K210 芯片数据的，而是经过 DAC 转化后经过功放放大信号，最后才是扬声器接收的信号。
2. 扬声器播放音乐需要使用 DMA 通道播放。
3. I2S 是专用于声音传输的协议，只需要三根线就可以传输音乐数据。

附：API

i2s_init

描述

初始化 I2S。

函数原型

```
void i2s_init(i2s_device_number_t device_num, i2s_transmit_t rxtx_mode,
uint32_t channel_mask)
```

参数

成员名称	描述	输入输出
device_num	I2S 号	输入
rxtx_mode	接收或发送模式	输入
channel_mask	通道掩码	输入

返回值

无。

i2s_send_data_dma

描述

I2S 发送数据。

函数原型

```
void i2s_send_data_dma(i2s_device_number_t device_num, const void *buf, size_t
buf_len, dmac_channel_number_t channel_num)
```

参数

成员名称	描述	输入输出
device_num	I2S 号	输入
buf	发送数据地址	输入
buf_len	数据长度	输入
channel_num	DMA 通道号	输入

返回值

无。

i2s_recv_data_dma

描述

I2S 接收数据。

函数原型

```
void i2s_recv_data_dma(i2s_device_number_t device_num, uint32_t *buf, size_t buf_len, dma_channel_number_t channel_num)
```

参数

成员名称	描述	输入输出
device_num	I2S 号	输入
buf	接收数据地址	输出
buf_len	数据长度	输入
channel_num	DMA 通道号	输入

返回值

无

i2s_rx_channel_config

描述

设置接收通道参数。

函数原型

```
void i2s_rx_channel_config(i2s_device_number_t device_num, i2s_channel_num_t channel_num, i2s_word_length_t word_length, i2s_word_select_cycles_t word_select_size, i2s_fifo_threshold_t trigger_level, i2s_work_mode_t word_mode)
```

参数

成员名称	描述	输入输出
device_num	I2S 号	输入

成员名称	描述	输入输出
channel_num	通道号	输入
word_length	接收数据位数	输出
word_select_size	单个数据时钟数	输入
trigger_level	DMA 触发时 FIFO 深度	输入
word_mode	工作模式	输入

返回值

无。

i2s_tx_channel_config

描述

设置发送通道参数。

函数原型

```
void i2s_tx_channel_config(i2s_device_number_t device_num, i2s_channel_num_t channel_num, i2s_word_length_t word_length, i2s_word_select_cycles_t word_select_size, i2s_fifo_threshold_t trigger_level, i2s_work_mode_t word_mode)
```

参数

成员名称	描述	输入输出
device_num	I2S 号	输入
channel_num	通道号	输入
word_length	接收数据位数	输出
word_select_size	单个数据时钟数	输入
trigger_level	DMA 触发时 FIFO 深度	输入
word_mode	工作模式	输入

返回值

无。

i2s_play

描述

发送 PCM 数据，比如播放音乐

函数原型

```
void i2s_play(i2s_device_number_t device_num, dmac_channel_number_t
channel_num,
             const uint8_t *buf, size_t buf_len, size_t frame, size_t
bits_per_sample, uint8_t track_num)
```

参数

成员名称	描述	输入输出
device_num	I2S 号	输入
channel_num	通道号	输入
buf	PCM 数据	输入
buf_len	PCM 数据长度	输入
frame	单次发送数量	输入
bits_per_sample	单次采样位宽	输入
track_num	声道数	输入

返回值

无。

i2s_set_sample_rate

描述

设置采样率。

函数原型

```
uint32_t i2s_set_sample_rate(i2s_device_number_t device_num, uint32_t
sample_rate)
```

参数

成员名称	描述	输入输出
device_num	I2S 号	输入
sample_rate	采样率	输入

返回值

实际的采样率。

i2s_set_dma_divide_16

描述

设置 dma_divide_16，16 位数据时设置 dma_divide_16，DMA 传输时自动将 32 比特 INT32 数据分成两个 16 比特的左右声道数据。

函数原型

```
int i2s_set_dma_divide_16(i2s_device_number_t device_num, uint32_t enable)
```

参数

成员名称	描述	输入输出
device_num	I2S 号	输入
enable	0:禁用 1: 使能	输入

返回值

返回值 描述

0 成功
非 0 失败

i2s_get_dma_divide_16

描述

获取 dma_divide_16 值。用于判断是否需要设置 dma_divide_16。

函数原型

```
int i2s_get_dma_divide_16(i2s_device_number_t device_num)
```

参数

成员名称	描述	输入输出
device_num	I2S 号	输入

返回值

返回值	描述
1	使能
0	禁用
<0	失败

i2s_handle_data_dma

描述

I2S 通过 DMA 传输数据。

函数原型

```
void i2s_handle_data_dma(i2s_device_number_t device_num, i2s_data_t data,
plic_interrupt_t *cb);
```

参数

参数名称	描述	输入输出
device_num	I2S 号	输入
data	I2S 数据相关的参数，详见 i2s_data_t 说明	输入
cb	dma 中断回调函数，如果设置为 NULL 则为阻塞模式，直至传输完毕后退出函数	输入

返回值

无

举例

```
/* I2S0 通道 0 设置为接收通道，接收 16 位数据，单次传输 32 个时钟，FIFO 深度为 4，标准模式。
接收 8 组数据*/
/* I2S2 通道 1 设置为发送通道，发送 16 位数据，单次传输 32 个时钟，FIFO 深度为 4，右对齐模
式。发送 8 组数据*/
uint32_t buf[8];
i2s_init(I2S_DEVICE_0, I2S_RECEIVER, 0x3);
i2s_init(I2S_DEVICE_2, I2S_TRANSMITTER, 0xC);
i2s_rx_channel_config(I2S_DEVICE_0, I2S_CHANNEL_0, RESOLUTION_16_BIT,
SCLK_CYCLES_32, TRIGGER_LEVEL_4, STANDARD_MODE);
i2s_tx_channel_config(I2S_DEVICE_2, I2S_CHANNEL_1, RESOLUTION_16_BIT,
SCLK_CYCLES_32, TRIGGER_LEVEL_4, RIGHT_JUSTIFYING_MODE);
i2s_recv_data_dma(I2S_DEVICE_0, rx_buf, 8, DMAC_CHANNEL1);
i2s_send_data_dma(I2S_DEVICE_2, buf, 8, DMAC_CHANNEL0);
```

数据类型

相关数据类型、数据结构定义如下：

- i2s_device_number_t: I2S 编号。
- i2s_channel_num_t: I2S 通道号。
- i2s_transmit_t: I2S 传输模式。
- i2s_work_mode_t: I2S 工作模式。
- i2s_word_select_cycles_t: I2S 单次传输时钟数。
- i2s_word_length_t: I2S 传输数据位数。
- i2s_fifo_threshold_t: I2S FIFO 深度。
- i2s_data_t: 通过 DMA 传输时数据相关的参数。
- i2s_transfer_mode_t: I2S 传输方式。

i2s_device_number_t

描述

I2S 编号。

定义

```
typedef enum _i2s_device_number
{
    I2S_DEVICE_0 = 0,
    I2S_DEVICE_1 = 1,
    I2S_DEVICE_2 = 2,
    I2S_DEVICE_MAX
} i2s_device_number_t;
```

成员

成员名称	描述
I2S_DEVICE_0	I2S 0
I2S_DEVICE_1	I2S 1
I2S_DEVICE_2	I2S 2

i2s_channel_num_t

描述

I2S 通道号。

定义

```
typedef enum _i2s_channel_num
{
    I2S_CHANNEL_0 = 0,
```

```
I2S_CHANNEL_1 = 1,  
I2S_CHANNEL_2 = 2,  
I2S_CHANNEL_3 = 3  
} i2s_channel_num_t;
```

成员

成员名称	描述
I2S_CHANNEL_0	I2S 通道 0
I2S_CHANNEL_1	I2S 通道 1
I2S_CHANNEL_2	I2S 通道 2
I2S_CHANNEL_3	I2S 通道 3

i2s_transmit_t

描述

I2S 传输模式。

定义

```
typedef enum _i2s_transmit  
{  
    I2S_TRANSMITTER = 0,  
    I2S_RECEIVER = 1  
} i2s_transmit_t;
```

成员

成员名称	描述
I2S_TRANSMITTER	发送模式
I2S_RECEIVER	接收模式

i2s_work_mode_t

描述

I2S 工作模式。

定义

```
typedef enum _i2s_work_mode  
{  
    STANDARD_MODE = 1,
```

```
    RIGHT_JUSTIFYING_MODE = 2,  
    LEFT_JUSTIFYING_MODE = 4  
} i2s_work_mode_t;
```

成员

成员名称	描述
STANDARD_MODE	标准模式
RIGHT_JUSTIFYING_MODE	右对齐模式
LEFT_JUSTIFYING_MODE	左对齐模式

i2s_word_select_cycles_t

描述

I2S 单次传输时钟数。

定义

```
typedef enum _word_select_cycles  
{  
    SCLK_CYCLES_16 = 0x0,  
    SCLK_CYCLES_24 = 0x1,  
    SCLK_CYCLES_32 = 0x2  
} i2s_word_select_cycles_t;
```

成员

成员名称	描述
SCLK_CYCLES_16	16 个时钟
SCLK_CYCLES_24	24 个时钟
SCLK_CYCLES_32	32 个时钟

i2s_word_length_t

描述

I2S 传输数据位数。

定义

```
typedef enum _word_length  
{  
    IGNORE_WORD_LENGTH = 0x0,
```

```
RESOLUTION_12_BIT = 0x1,  
RESOLUTION_16_BIT = 0x2,  
RESOLUTION_20_BIT = 0x3,  
RESOLUTION_24_BIT = 0x4,  
RESOLUTION_32_BIT = 0x5  
} i2s_word_length_t;
```

成员

成员名称	描述
IGNORE_WORD_LENGTH	忽略长度
RESOLUTION_12_BIT	12 位数据长度
RESOLUTION_16_BIT	16 位数据长度
RESOLUTION_20_BIT	20 位数据长度
RESOLUTION_24_BIT	24 位数据长度
RESOLUTION_32_BIT	32 位数据长度

i2s_fifo_threshold_t

描述

I2S FIFO 深度。

定义

```
typedef enum _fifo_threshold  
{  
    /* Interrupt trigger when FIFO level is 1 */  
    TRIGGER_LEVEL_1 = 0x0,  
    /* Interrupt trigger when FIFO level is 2 */  
    TRIGGER_LEVEL_2 = 0x1,  
    /* Interrupt trigger when FIFO level is 3 */  
    TRIGGER_LEVEL_3 = 0x2,  
    /* Interrupt trigger when FIFO level is 4 */  
    TRIGGER_LEVEL_4 = 0x3,  
    /* Interrupt trigger when FIFO level is 5 */  
    TRIGGER_LEVEL_5 = 0x4,  
    /* Interrupt trigger when FIFO level is 6 */  
    TRIGGER_LEVEL_6 = 0x5,  
    /* Interrupt trigger when FIFO level is 7 */  
    TRIGGER_LEVEL_7 = 0x6,  
    /* Interrupt trigger when FIFO level is 8 */  
    TRIGGER_LEVEL_8 = 0x7,  
}
```

```

/* Interrupt trigger when FIFO level is 9 */
TRIGGER_LEVEL_9 = 0x8,
/* Interrupt trigger when FIFO level is 10 */
TRIGGER_LEVEL_10 = 0x9,
/* Interrupt trigger when FIFO level is 11 */
TRIGGER_LEVEL_11 = 0xa,
/* Interrupt trigger when FIFO level is 12 */
TRIGGER_LEVEL_12 = 0xb,
/* Interrupt trigger when FIFO level is 13 */
TRIGGER_LEVEL_13 = 0xc,
/* Interrupt trigger when FIFO level is 14 */
TRIGGER_LEVEL_14 = 0xd,
/* Interrupt trigger when FIFO level is 15 */
TRIGGER_LEVEL_15 = 0xe,
/* Interrupt trigger when FIFO level is 16 */
TRIGGER_LEVEL_16 = 0xf
} i2s_fifo_threshold_t;

```

成员

成员名称	描述
TRIGGER_LEVEL_1	1 字节 FIFO 深度
TRIGGER_LEVEL_2	2 字节 FIFO 深度
TRIGGER_LEVEL_3	3 字节 FIFO 深度
TRIGGER_LEVEL_4	4 字节 FIFO 深度
TRIGGER_LEVEL_5	5 字节 FIFO 深度
TRIGGER_LEVEL_6	6 字节 FIFO 深度
TRIGGER_LEVEL_7	7 字节 FIFO 深度
TRIGGER_LEVEL_8	8 字节 FIFO 深度
TRIGGER_LEVEL_9	9 字节 FIFO 深度
TRIGGER_LEVEL_10	10 字节 FIFO 深度
TRIGGER_LEVEL_11	11 字节 FIFO 深度
TRIGGER_LEVEL_12	12 字节 FIFO 深度
TRIGGER_LEVEL_13	13 字节 FIFO 深度
TRIGGER_LEVEL_14	14 字节 FIFO 深度
TRIGGER_LEVEL_15	15 字节 FIFO 深度
TRIGGER_LEVEL_16	16 字节 FIFO 深度

i2s_data_t

描述

通过 DMA 传输时数据相关的参数。

定义

```
typedef struct _i2s_data_t
{
    dma_channel_number_t tx_channel;
    dma_channel_number_t rx_channel;
    uint32_t *tx_buf;
    size_t tx_len;
    uint32_t *rx_buf;
    size_t rx_len;
    i2s_transfer_mode_t transfer_mode;
    bool nowait_dma_idle;
    bool wait_dma_done;
} i2s_data_t;
```

成员

成员名称	描述
tx_channel	发送时使用的 DMA 通道号
rx_channel	接收时使用的 DMA 通道号
tx_buf	发送的数据
tx_len	发送数据的长度
rx_buf	接收的数据
rx_len	接收数据长度
transfer_mode	传输模式，发送或接收
nowait_dma_idle	DMA 传输前是否等待 DMA 通道空闲
wait_dma_done	DMA 传输后是否等待传输完成，如果 cb 不为空则这个函数无效

i2s_transfer_mode_t

描述

I2S 传输方式。

定义

```
typedef enum _i2s_transfer_mode
{
    I2S_SEND,
    I2S_RECEIVE,
} i2s_transfer_mode_t;
```

成员

成员名称	描述
I2S_SEND	发送
I2S_RECEIVE	软件