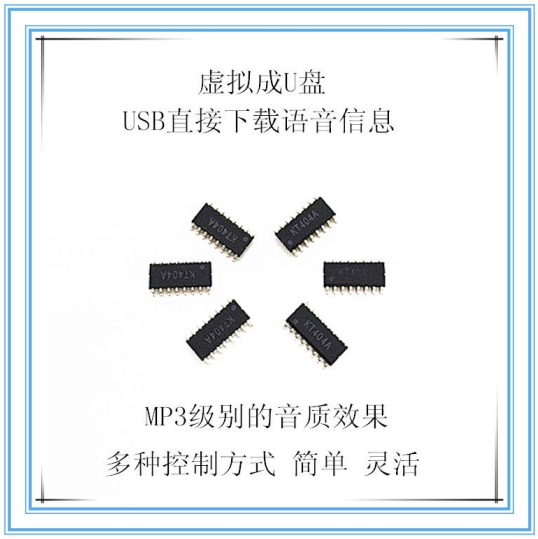
KT404C 芯片使用手册

文件状态：	文件标识：	
[] 草稿	当前版本：	V1.5
[√] 正式发布	作　　者：	
[　] 正在修改	完成日期：	2021-04-12



版 本 历 史

版本	日期	原因
V1.0	2014/08/18	内测版本和优化版本
V1.1	2014/12/01	1、在 PTUF1FS_V1.6版本的基础上进行剪裁 2、芯片方案精简为 SOP16封装的 KT604C 3、不可以同时支持 TF 卡和 SPIFLASH
V1.2	2015/02/08	1、完善芯片的说明
V1.3	2016/01/28	1、增加芯片的相关注意事项
V1.4	2016/08/28	1、插播指令使用0x25
V1.5	2021/04/15	1、应对 KT404C 芯片缺货和即将停产，新开发的 KT404C 2、注意 KT404C 和 KT404C 的芯片脚位排列不一样 3、软件上面基本做到无缝兼容，可能还是有略微的细小区别 4、新增了设定芯片的地址指令，详见3.4.19章节 5、新增设置芯片 uart 的波特率的指令，详见3.3.7章节

目 录

1.概述

1.1 简介

KT404C 是一个提供串口的语音芯片，完美的集成了 MP3、WAV 的硬解码。同时软件支持工业级别的串口通信协议，以 SPIFLASH 作为存储介质，用户可以灵活的选用其中的任何一种设备作为语音的存储介质。通过简单的串口指令即可完成播放指定的语音，以及如何播放语音等功能，无需繁琐的底层操作，使用方便，稳定可靠是此款产品的最大特点。

无需任何烧录器，无需任何软件，USB 直接烧写 FLASH。
同时可以以 SPIFLASH 为存储介质的 U 盘功能,后面章节详细说明

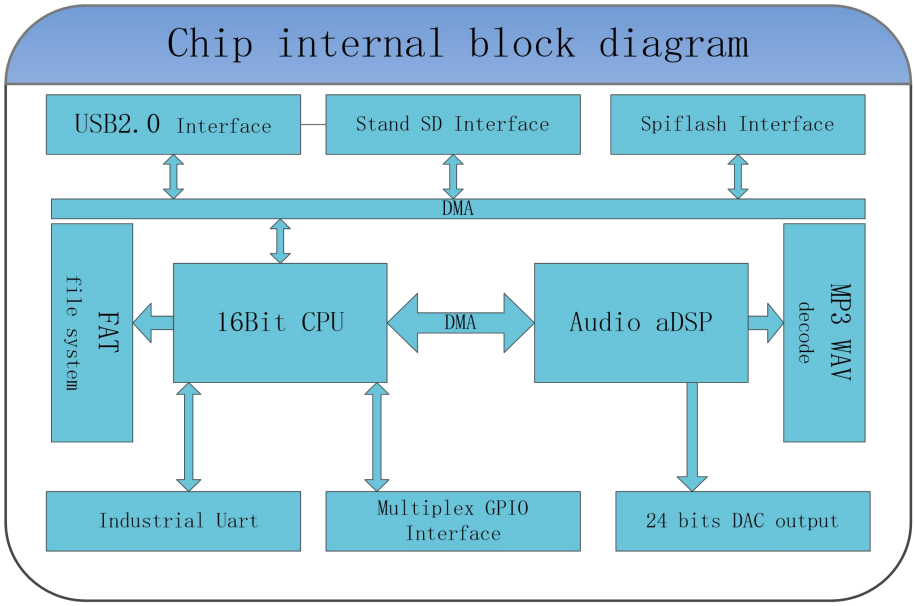
1.2 功能

1、支持采样率(KHz):8/11.025/12/16/22.05/24/32/44.1/48
2、24 位 DAC 输出，动态范围支持 90dB，信噪比支持 85dB
3、最大支持 16M 字节的 SPIFLASH。例如 W25Q16[2M 字节]、W25Q128[16M 字节]
4、多种控制模式，并口控制模式、串口模式、AD 按键控制模式
5、Miniusb 接口更新语音文件，无需安装任何软件。支持 XP 和 WIN7 系统。
6、支持组合播放功能，可以实现报时、报温度，在一定程度上可以替代一些昂贵的 TTS 方案
7、30 级音量可调，5 级 EQ 可调[此功能暂不开放]
8、支持外挂多种类型的功放，如 3W、5W、10W 等等
9、支持上电循环播放的功能
10、可以支持多种存储介质，如：U 盘、TF 卡、SPIFLASH[超小型封装]作为存储介质

1.3 应用

1、 车载导航语音播报
2、 公路运输稽查、收费站语音提示；
3、 火车站、汽车站安全检查语音提示；
4、 电力、通信、金融营业厅语音提示；
5、 车辆进、出通道验证语音提示；
6、 公安边防检查通道语音提示；
8、 电动观光车安全行驶语音告示；
9、 机电设备故障自动报警；
10、消防语音报警提示；

1. 方案说明

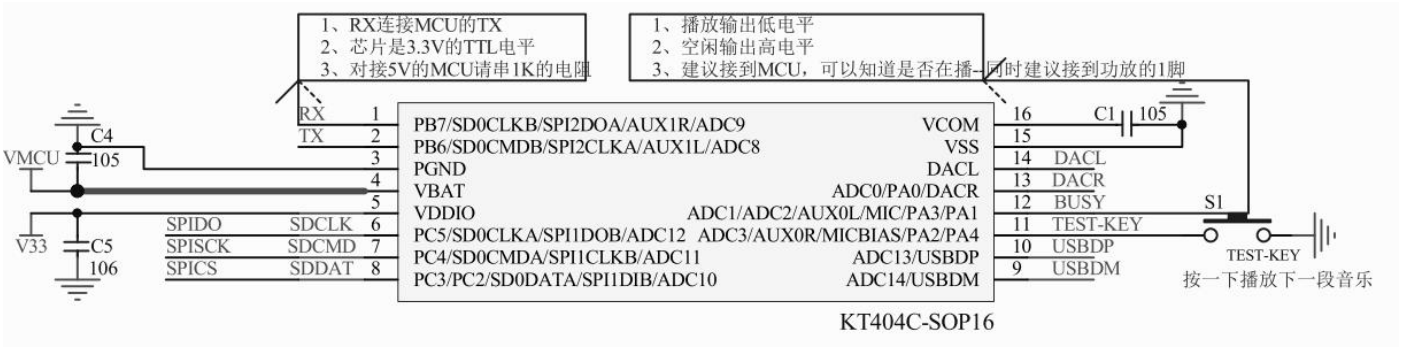


芯片选用的是 SOC 方案，集成了一个 16 位的 MCU，以及一个专门针对音频解码的 aDSP，采用硬解码的方式，更加保证了系统的稳定性和音质。小巧的封装尺寸更加满足嵌入其它产品的需求

2.1 参数说明

名称	参数
MP3文件格式	1、支持所有比特率11172-3和 ISO13813-3 layer3音频解码
	2、采样率支持(KHZ):8/11.025/12/16/22.05/24/32/44.1/48
	3、支持 Normal、Jazz、Classic、Pop、Rock 等音效
USB 接口	2.0标准
UART 接口	标准串口，TTL 电平,波特率可设[用户不可设]
输入电压	3.7V-5.2V[7805后级串一个二极管为最佳]
额定电流	10MA[静态]
低功耗电流	<200uA
功放功率	负载32欧姆0.25W[只支持驱动耳机][不支持直接驱动喇叭]
尺寸	参见封装章节
工作温度	[-40度] -- [80度]
湿度	5% ~ 95%
主芯片型号	KT404C[SOP16]

2.2 管脚说明



引脚序号	引脚名称	功能描述	备注
1	PB7	串口-RX	连接 MCU 的 TX 脚
2	PB6	串口-TX	连接 MCU 的 RX 脚
3	PGND	数字地	
4	VBAT	电源输入引脚	4.2V 为最佳
5	VDDIO	3.3V LDO 输出[**是 KT404C 的输出**]	最大 100MA
6	PC5	SPI 的数据口	
7	PC4	SPI 的时钟口	
8	PC3	SPI 的片选	
9	USBDM-D-	USB 接口/通用输入输出口	必须预留测试点
10	USBDM-D+	USB 接口/通用输入输出口	必须预留测试点
11	PA4	测试引脚	低有效
12	PA1	输出忙信号	有声音输出低，空闲输出高
13	DACR	右声道输出	
14	DACL	左声道输出	
15	VSS	模拟音频地	
16	VCOM	音频信号偏置	

只要需要外接喇叭的应用，即使是 0.5W 的，也是需要外加功放的.DACL 和 DACR 只能推动耳机

2.3 测试简述

1、用户拿到芯片之后，可以直接插上 USB 线缆，对芯片进行下载语音，正常，插入 USB 线之后，电脑会显示如 U 盘插入一样的窗口。并且电脑会自动安装 USB 驱动，无需用户关心

2、如果下载语音完成之后，有两种选择测试

(1)、用户可以直接拔掉 USB 线缆，再供电测试[注意不要插入电脑]，相当于对芯片进行一次重启。[建议这样测试]

(2)、下载完声音之后，可以直接使用外接电源或者 USB 充电头来供电测试 。然后串口发指令测试

3、如果用户需要简单的测试一下音质效果，搭建好电路之后，在芯片的第 11 脚，连接一个按键[微动开关]到地，按一下芯片就会播放语音，一直触发，会一直循环播放设备里面的语音。

3. 串口通讯协议

串口作为一种在控制领域常用的通信，我们进行了工业级别的优化，加入的帧的校验、重发、错误处理等措施，大大加强通信的稳定性和可靠性，同时可以在此基础上扩展更加强大的 RS485
进行组网功能，串口的通信波特率可自行设置，默认为 9600

3.1 通讯格式

支持异步串口通讯模式,通过串口接受上位机发送的命令 **通讯标准:9600 bps　　数据位　:8　　停止位　:1　　校验位　:none　　流控制　:none**		
格式：$S　VER　Len　CMD　Feedback　para1　para2　checksum　$0		
$S	起始位0x7E	每条命令反馈均以$开头,即0x7E
ADDR	地址	地址信息--默认为0xFF
Len	len 后字节个数	校验和不计算在内
CMD	命令字	表示具体的操作,比如播放/暂停等等
Feedback	命令反馈	是否需要反馈信息,1反馈,0不反馈
dat	参数	和前面的 len 相关联，不限制长度
checksum	校验和[占两个字节]	累加和校验[不计起始位$]
$0	结束位	结束位0xEF

举个例子，如果我们指定播放 SPIFLASH，就需要发送:7E FF 06 09 00 00 04 FF dd EF
数据长度为 6 ,这 6 个字节分别是[FF 06 09 00 00 04] 。不计算起始、结束、和校验。然后对得到的结果进行累加，再用 0 减，即“0-checksum=校验数据”。如果这里不明白，可以参考我们的“QYMxFS 辅助说明文档”。另外用户也可以直接忽视校验，参考我们的 5.3.4 章节说明。

3.2 通讯指令

我们的通讯分为以下两大块

- 控制指令
- 查询芯片的参数以及状态

3.2.1 控制指令

CMD 命令	对应的功能	参数(16位)
0x01	下一曲	
0x02	上一曲	
0x03	指定曲目(NUM)	详见3.4.1
0x04	音量+	
0x05	音量-	
0x06	指定音量	详见3.4.2
0x08	单曲循环指定曲目播放	详见3.4.3
0x09	指定播放设备	保留
0x0A	进入睡眠 -- 低功耗	详见3.4.5
0x0B	指定波特率	详见3.3.7
0x0C	芯片复位	任何状态有效
0x0D	播放	
0x0E	暂停	
0x0F	指定文件夹文件名播放	详见3.4.6
0x14	单个文件夹支持1000首曲目	保留
0x15	停止插播播放背景音乐	详见3.4.9
0x16	停止	
0x17	指定文件夹循环播放	详见3.4.10
0x18	指定根目录随机播放	详见3.4.11
0x19	对当前播放的曲目设置为循环播放	详见3.4.12
0x1A	开启和关闭芯片的 DAC 输出	详见3.4.13
0x21	组合播放	详见3.4.14
0x25	多文件夹插播	详见3.4.16
0x28	指定文件夹随机播放	详见3.4.18
0xC0	设定地址	详见3.4.19
0xC1	恢复出厂设置	详见3.4.20

3.2.2 查询指令

这里是查询芯片的状态和相关的参数

CMD 命令详解(查询)	对应的功能	参数(16位)
0x3C	保留	
0x3D	保留	
0x3E	保留	
0x3F	查询在线的设备	详见3.5.1
0x40	返回错误，请求重发	
0x41	应答	
0x42	查询当前状态	详见3.4.10
0x43	查询当前音量	
0x44	查询当前 EQ	保留
0x45	保留	该版本保留此功能
0x46	保留	该版本保留此功能
0x49	查询 FLASH 的总文件数	5个文件夹的总文件数
0x4D	查询 FLASH 的当前曲目	返回文件夹号和曲目指针
0x4E	查询指定文件夹的曲目总数	详见3.5.3
0x4F	查询当前设备的总文件夹数	详见3.5.4
0x61	查询当前文件夹指针	仅支持 FLASH

3.3 芯片返回的数据

芯片在关键地方均会有数据返回。供用户掌控芯片的工作状态

- 芯片上电初始化成功的数据
- 芯片播放完当前曲目的数据
- 芯片成功接收到指令返回的 ACK(应答)
- 芯片接收一帧数据出错[包括数据没收完整、校验出错两种情况]
- 芯片在繁忙时，有数据过来，芯片会返回忙的指令
- U 盘、TF 卡插入拔出，均有数据返回

3.3.1 芯片上电返回的数据

PC -- 在线	7E FF 06 3F 00 00 04 xx xx EF	即下载模式
FLASH -- 在线	7E FF 06 3F 00 00 08 xx xx EF	

1、芯片上电，需要一定的时间初始化，这个时间是需要根据 SPIFLASH 设备的文件多少决定的，一般情况在 1000ms。如果超过这个时间芯片的初始化数据还没有发送出来，说明芯片初始化出错，请检查硬件

2、芯片初始化返回的数据为当前的有效文件夹,譬如返回 7E FF 06 3F 00 00 03 xx xx EF

==>其中 0x03 代表的是 U 盘和 TF 这两个设备在线

3、MCU 必须等待芯片初始化指令发出之后才能发送相应的控制指令，否则发送的指令芯片将不予处理。

3.3.2 曲目播放完毕返回的数据[0x3C][0x3D][0x3E]

U 盘播放完第1曲	7E FF 06 3C 00 00 01 xx xx EF
U 盘播放完第2曲	7E FF 06 3C 00 00 02 xx xx EF
TF 卡播放完第1曲	7E FF 06 3D 00 00 01 xx xx EF
TF 卡播放完第2曲	7E FF 06 3D 00 00 02 xx xx EF
FLASH 播放完第1曲	7E FF 06 3E 00 00 01 xx xx EF
FLASH 播放完第2曲	7E FF 06 3E 00 00 02 xx xx EF

1、芯片默认播放完毕之后，自动停止，等待用户发送新的指令

2、另外我们专门开辟一个 IO 作为播放和未播放的状态指示。**请参见第 BUSY 脚**

4、另外用户在指定设备之后，需要等待 200ms 的时间，再发送指定的曲目，因为一旦指定设备之后，系统会对指定的设备进行文件系统的初始化，如果立刻发送指定的曲目命令，会导致模块接收不到。

5、设备播放完每一段语音，都会有数据返回，举例说明:7E FF 06 3C 00 **00 01** xx xx EF

(1)、其中 0x3C 代表的是 U 盘，详细的请参见上表

(2)、其中 0x00,0x01 代表的是第 1 段语音播放完毕 。0x01,0xF4=第 500 段声音。0x01F4 = 500.这里代表的是高低字节

6、由于设备中所有的文件均是以物理顺序进行识别的，包括指定文件夹文件名播放，播放完毕之后，返回的数据还是其存储的物理顺序编号，这点请用户朋友留意。

3.3.3 模块应答返回的数据[0x41]

模块返回 ACK	7E FF 06 41 00 00 00 xx xx EF	说明成功接收数据

(1)、为了加强数据通信之间的稳定性，我们增加应答处理，ACKB 字节就是设置是否需要回复应答。这样做的好处是保证每次通信都有握手信号，收到应答就表示 MCU 发送的数据，模块已经成功收到，马上处理。
(2)、对于一般的应用，客户可以自由选择，不加这个应答处理也是可以的。

3.3.4 模块错误返回的数据[0x40][0x50]

返回忙	7E FF 06 40 00 00 01 xx xx EF	芯片在文件系统初始化时
当前是睡眠模式	7E FF 06 40 00 00 02 xx xx EF	睡眠模式只支持指定设备
串口接收错误	7E FF 06 40 00 00 03 xx xx EF	串口一帧数据没接收完毕
校验出错	7E FF 06 40 00 00 04 xx xx EF	和校验出错
指定文件超范围	7E FF 06 40 00 00 05 xx xx EF	文件的指定超过设定的范围
未找到指定文件	7E FF 06 40 00 00 06 xx xx EF	指定为文件没有被找到
插播错误	7E FF 06 40 00 00 07 xx xx EF	插播只允许在播放的状态下进行
播放 TF 卡错误	7E FF 06 40 00 00 08 xx xx EF	TF 卡读取失败或者 TF 卡被拔出或者有坏区
FLASH 初始化出错	7E FF 06 40 00 00 09 xx xx EF	FLASH 里面的文件系统信息错误
进入睡眠	7E FF 06 40 00 00 0A xx xx EF	进入 SLEPP 模式

1、为了加强数据通信之间的稳定性，增加了数据错误处理机制。模块收到不符合格式的数据，或者异常均会有信息反馈
2、在环境比较恶劣的情况下，强烈建议客户处理此命令。如果应用环境一般，可以不用处理。
3、模块返回忙，基本上是模块上电初始化的时候才会返回，因为模块需要初始化文件系统
4、只要参考我们给出的测试 SDK 程序，移植里面的串口操作部分，就不会出现校验出错，在这里强烈建议用户使用我们给出的校验方式。因为谁都不能保证数据的传输不会出错。

3.3.5 设备插入拔出消息[0x3A][0x3B]

U 盘插入	7E FF 06 3A 00 00 01 xx xx EF
TF 插入	7E FF 06 3A 00 00 02 xx xx EF
PC 插入	7E FF 06 3A 00 00 04 xx xx EF
U 盘拔出	7E FF 06 3B 00 00 01 xx xx EF
TF 拔出	7E FF 06 3B 00 00 02 xx xx EF
PC 拔出	7E FF 06 3B 00 00 04 xx xx EF

1、为了加强模块灵活性，我们特别增加了，设备插入、拔出的指令反馈。方便知道模块的工作状态。
2、设备插入的时候，我们默认进入到设备等待状态，如果用户插入的是带灯的 U 盘，可以看到 U 盘灯闪烁。也可以接收到设备插入的串口消息。

3.3.6 设备文件系统初始化成功返回[0x90]

U 盘挂载文件系统 OK	7E FF 06 90 00 00 01 xx xx EF
TF 卡挂载文件系统 OK	7E FF 06 90 00 00 02 xx xx EF
spiflash 挂载文件系统 OK	7E FF 06 90 00 00 04 xx xx EF

1、主控芯片初始化设备，并且挂载文件系统成功之后，新增此数据的返回，方便用户掌握状态

2、三种设备，挂载成功之后均会返回对应的数据

3.3.7 指定波特率[0x0B]

7E FF 06 0B 00 00 01 EF--设置 9600
7E FF 06 0B 00 00 02 EF--设置 19200
7E FF 06 0B 00 00 03 EF--设置 38400
7E FF 06 0B 00 00 04 EF--设置 57600
7E FF 06 0B 00 00 05 EF--设置 115200
7E FF 06 0B 00 00 06 EF--设置 256000
7E FF 06 0B 00 00 07 EF--设置 31250
7E FF 06 0B 00 00 08 EF--设置 2400
7E FF 06 0B 00 00 09 EF--设置 4800
1、一旦设置了波特率之后，芯片会记忆。下一次开机，波特率就变成了您所设置的.不支持查询
2、设置完波特率之后，请等待 1 秒钟，再发送复位[0x0C 指令]，或者断电一下即可
3、如果要恢复默认的波特率，请发送恢复出厂设置的命令，此时芯片会自动擦除所有的配置
4、由于我们芯片的主频很高，**所以尽量把串口的波特率调高，越高越好**

3.4 串口控制指令详解

以下我们对关键的地方进行详细的说明--针对控制指令:

- 指定曲目播放
- 指定播放的音量
- 指定播放的设备
- 全部循环播放指令
- 组合播放功能[亮点]
- 带音量参数的指定曲目播放

3.4.1 指定歌曲播放指令[0x03]

我们给出的指令是支持指定曲目播放的，歌曲的选择范围为 0~3000.其实是可以支持更多的，因为涉及到文件管理的原因，支持过多的歌曲，会导致系统操作缓慢，一般的应用也不需要支持这么多的文件。如果客户有非常规的应用，请事前和我们沟通。此指令在 TF 卡和 U 盘状态是按照存储的物理顺序指定的。FLASH 同样也是按照物理顺序[拷贝的先后顺序]

(1)、例如选择第一首歌播放，串口的发送部分 7E FF 06 03 00 00 01 FE E7 EF

数据	详解
0x7E	起始字节
0xFF	芯片的地址信息
0x06	数据长度[不包含校验]
0x03	命令字节
0x00	是否需要应答[0x01=需要应答　0x00=不需要应答]
0x00	曲目的高字节[DH]
0x01	曲目的低字节[DL]，这里代表的是一首歌播放
0xFE	校验高字节
0xF7	校验低字节
0xEF	结束字节

(2)、对于选曲，如果选择第 100 首，首先将 100 转化为 16 进制,默认为双字节,就为 0x0064。

DH = 0x00　; DL = 0x64

(3)、其它的操作依次类推即可，因为在嵌入式领域采用 16 进制是最为方便的一种操作。

3.4.2 指定音量播放指令[0x06]

(1)、我们系统上电默认的音量为 30 级，如果要设置音量的话,直接发送相应的指令即可

(2)、芯片内部设置的音量细分级数为 0--30.请用户注意

(3)、例如指定音量为 15 级,串口发送的指令:7E FF 06 06 00 00 0F FF D5 EF

(4)、DH = 0x00 ; DL = 0x0F ，15 转化为 16 进制为 0x000F。可以参照播放曲目部分的说明

3.4.3 单曲循环播放指令[0x08]

循环播放第一曲	7E FF 06 08 00 00 01 xx xx EF
循环播放第二曲	7E FF 06 08 00 00 02 xx xx EF

(1)、争对一些需要单曲循环播放的要求，我们改进这一条控制指令 0x08。在操作 TF 卡或者 U 盘/以及 FLASH 时，按照的是文件存储的物理顺序指定，这点请用户注意。

(2)、在循环播放的过程中，可以正常的操作播放/暂停，上一曲、下一曲、音量调节，包括 EQ 等等并且状态仍然是循环播放.可以通过指定单曲触发播放或者停止来关闭循环播放状态

3.4.4 指定播放设备[0x09]

1、我们的模块默认是支持 4 种类型的播放设备，只有设备在线才能指定设备去播放。设备是否在线，我们软件会自动检测，无需用户关系。

2、看下表，选择合适的指令发送

(3)、指定设备之后。模块会自动进入停止解码状态，等待用户指定曲目播放。从接收到指定设备到模块内部完成初始化文件系统。大概需要 200ms。请等待 200ms 之后再发送指定曲目的指令。

指定播放设备-U 盘	7E FF 06 09 00 00 01 xx xx EF	xx xx：代表校验
指定播放设备-TF 卡	7E FF 06 09 00 00 02 xx xx EF	
指定播放设备-FLASH	7E FF 06 09 00 00 04 xx xx EF	
指定播放设备-PC	7E FF 06 09 00 00 05 xx xx EF	指[下载]模式
指定播放设备-SLEEP	7E FF 06 09 00 00 06 xx xx EF	

3.4.5 进入睡眠[0x0A]

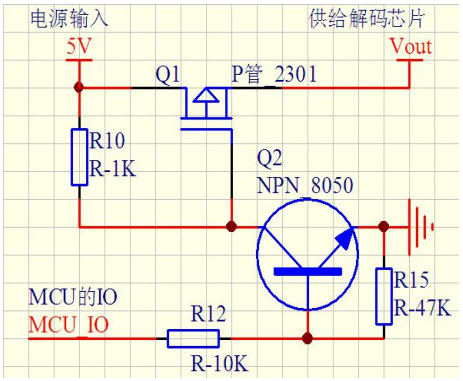
进入睡眠模式	7E FF 06 0A 00 00 00 FE F1 EF	xx xx：代表校验

指定芯片进入睡眠之后，有两种方式对芯片或者模块进行唤醒

(1)、指定一下播放的设备，指定播放 U 盘或者 TF 卡，或者 FLASH 都可以唤醒

(2)、对当前的设备进行一次拔出，再插入也是可以直接唤醒

芯片或者模块进入睡眠之后，待机功耗大概在 10MA。功耗依然较大，所以用户对功耗有要求的地方，请注意了。可以选择使用一个 pmos 管和一个三极管来控制芯片或者模块的供电，不需要的时候可以完全断电。如右图



3.4.6 指定文件夹文件名播放[0x0F]

文件夹01的001xxx.mp3	7E FF 06 0F 00 01 01 xx xx EF
文件夹11的100xxx.mp3	7E FF 06 0F 00 0B 64 xx xx EF
文件夹99的255xxx.mp3	7E FF 06 0F 00 63 FF xx xx EF

1、指定文件夹播放是我们制定的扩展功能，默认文件夹的命名方式为"01","11"这样的方式，为了系统的稳定性和歌曲切换的速度，每个文件夹下默认最大支持 255 首歌,最多支持 99 个文件夹

2、例如指定"01"文件夹的 100xxx.MP3 文件,串口发送的指令为:7E FF 06 0F 00 01 64 xx xx EF

DH:代表的是文件夹的名字,默认支持 99 个文件,即 01 -- 99 的命名

DL:代表的是曲目,默认最多 255 首歌，即 0x01 ~ 0xFF

3、为了模块的标准性，必须同时指定文件夹和文件名，来锁定一个文件。单独指定文件夹或者单独指定文件名也是可以的，但是这样文件的管理会变差。指定文件夹和指定曲目是支持 MP3、WAV

4、下面截两个图说明文件夹和文件名的指定[分左右两个图]

5、SPIFLASH 的操作和 TF 卡以及 U 盘一致，只是 SPIFLASH 的空间有限,能存放的语音数量有限这个还需要根据客户对音质的要求进行压缩语音，来存放更多的语音信息

3.4.8 全部循环播放指令[0x11]

循环播放开始	7E FF 06 11 00 00 01 xx xx EF	循环播放所有曲目
循环播放停止	7E FF 06 11 00 00 00 xx xx EF	停止循环播放曲目

1、争对一些需要循环播放根目录下面曲目的要求，我们加多这一条控制指令 0x11。

2、在循环播放的过程中，可以正常的操作播放/暂停，上一曲、下一曲、音量调节，包括 EQ 等等

3、循环播放开始之后，模块会不停的播放设备里面的曲目，按照存储的物理顺序。播完一遍之后会继续再播放一边，直到接收到播放完成，或者暂停等等指令

3.4.9 播放停止指令[0x15][0x16]

停止播放广告	7E FF 06 15 00 00 00 FE E6 EF	停止当广告，回到背景音乐继续播
停止播放	7E FF 06 16 00 00 00 FE E5 EF	停止软件解码

1、在模块的播放过程中，我们有两种停止方式，一种是停止当前的插播广告，回到当前断点处继续播放背景音乐。另一种是停止所有的播放，包括背景音乐

2、假如当前在播放插播广告，这时发送停止指令 0x16，芯片会停止所有播放任务.0x16 的停止指令的优先级别是最高的，请用户留意

3.4.10 指定文件夹开始循环顺序播放[0x17]

指定文件夹循环播放	7E FF 06 17 00 00 02 FE E2 EF	指定02文件夹循环播放
	7E FF 06 17 00 00 01 FE E3 EF	指定01文件夹循环播放

1、对于 TF 卡和 U 盘或者 SPIFLASH，文件夹的命名方式必须是”01” --- “99”。不可以超过 99

2、一旦指定文件夹循环之后，可以使用播放/暂停/上一曲/下一曲。这些操作命令都不会打断当前的文件夹循环播放状态。也就是说，发送下一曲指令之后，还是会循环当前的文件夹。

3、用户可以发送停止指令 0x16 来结束循环播放，返回至触发播放状态

3.4.11 随机播放设备文件[0x18]

随机播放	7E FF 06 18 00 00 00 xx xx EF	整个设备的随机播放

1、此指令是随机播放设备里面存储的所有语音文件，是按照物理顺序随机播放，不分设备里面是否带有文件夹。并且播放的第一个语音文件一定是设备里面的第一个语音文件

3.4.12 对当前的曲目设置为循环播放[0x19]

循环播放开启关闭	7E FF 06 19 00 00 00 FE E2 EF	单曲循环播放开启
	7E FF 06 19 00 00 01 FE E1 EF	单曲循环播放关闭

1、在播放的过程中发送此指令，会循环播放当前的曲目。如果当前是处理暂停或者停止状态，则芯片不会响应此指令

2、如果要关闭单曲循环播放，发送关闭的指令即可，这样会把当前的曲目播放完毕之后，就停止。

3.4.14 组合播放功能指令[0x21]

组合播放	7E FF 09 21 01 02 02 03 01 04 EF	播放[1,2][2,3][1,4]
	7E FF 15 21 01 02 02 03 01 04 01 03 01 04 01 05 02 08 03 04 03 01 FE 9A EF	带校验 [1,2][2,3][1,4][1,3][1,4][1,5][1,8][3,4][3,1]
	7E FF 15 21 01 02 02 03 01 04 01 03 01 04 01 05 02 08 03 04 03 01 EF	[1,2][2,3][1,4][1,3][1,4][1,5][1,8][3,4][3,1]

1、很多应用场合需要用到 TTS 的功能，譬如报时，报温度，报金额等等应用，如果用户拿我们的模块模仿简单的 TTS 功能的话，可能会在效果上面大打折扣，矛盾点就是在语音和语音之间的延时。直接按照一个一个文件的指定播放的话，会存在语音和语音之间的延时，这样是不能接受的。因此我们增加了组合播放的功能，同时最多支持指定播放 30 个语音，按照串口发送的顺序依次播放。

2、发送 7E FF **15** 21 **01 02 02 64 01 04 01 40 01 06 01 07 02 08 03 04 03 02** EF 这一帧数据，我们分析一下

CMD= 0x21 --- 查阅指令表

LEN = 0x15 = 21 个字节 ---**FF 15 21 01 02 02 64 01 04 01 40 01 06 01 07 02 08 03 04 03 02**[其中一段语音，由两个参数组成，即“文件夹编号”和“文件名编号”]

模块会依次播放 01 文件夹第 002 曲、02 文件夹第 100 曲、01 文件夹第 004 曲、01 文件夹第 064 曲,01 文件夹第 006 曲、01 文件夹第 007 曲、02 文件夹第 008 曲、03 文件夹第 004 曲、03 文件夹第 002 曲这 9 段语音。播放完毕就停止。其中每播放完一段语音都会有串口数据返回[按照物理编号]这样就实现了跨文件夹的语音组合播放功能。

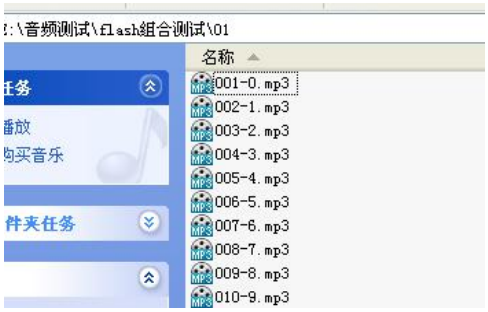
特别注意，这里面所有的参数是 16 进制，请用户注意，如 02 64 = 代表 02 文件夹里面的 100.mp3 这个文件

3、在组合播放的过程中，是允许用户进入播放暂停和设置音量，但是不允许指定上下曲。如果用户对组合播放的要求比较高的话,请用户对音源自行编辑一下,去掉音源头和尾的一些静音。这样就可以减少语音和语音之间的延时,可以采用“Adobe Audition CS5.5”或者“GoldWave.exe”等等专业音频软件制作。

4、有了这个功能，就可以很方便的实现“欢迎光临，现在时间是 13 年 12 月 24 日”这样的灵活播报方式，大大提高了产品的竞争力。

5、另外在组合播放的过程中，需要停止，可以直接发送停止指令。组合播放的过程中，不允许穿插其他的组合播放，要打断组合播放之前，必须先发送停止指令。

6、组合播放的文件，必须存放在“01”或者“02”或者其他“03-99”文件夹里面，文件必须重命名为“001xx.mp3 或者其他，如下截图。组合播放一定是要带文件夹的根目录里面是不支持的，因为根目录是不利于文件管理



7、如果组合播放发送的指定文件播放，而设备中没有对应的文件的话，组合播放会在当前停止。

请一定让发送的指令能找到相对应的文件。一旦出错，就会停止在出错的文件位置

8、此指令比较长，所以我们去掉了反馈字节，也就是“FEEDBACK”请用户留意

9、在组合播放的过程中，busy 引脚是持续拉低的，直到播放完成之后才会拉高，这个可以用来检测组合播放结束

3.4.16 多文件夹插播功能[0x25]

插播"ADVERT1"的文件夹，曲目为"001"	7E FF 06 25 00 01 01 FE D4 EF
插播"ADVERT1"的文件夹，曲目为"002"	7E FF 06 25 00 01 02 FE D3 EF
插播"ADVERT2"的文件夹，曲目为"001"	7E FF 06 25 00 02 01 FE D3 EF

1、在之前我们的插播基础上，我们增加了多文件夹的插播功能，命名的方式如上图

2、文件夹最多支持 9 个，也就是从 ADVERT1 --- ADVERT9 ，文件夹的命令请一定按照我们给出的规则，否则会导致出错。请注意上图的文件夹命名格式

3、单个插播文件夹下的文件最大不能超过 255，也就是“255xxx.MP3/WAV”，请注意上图的文件名的命名格式。

4、完善了插播的相关应用，如当前播放的文件处于单曲循环或者当前文件夹循环，即使有插播进来，也不会改变当前状态，还会是单曲循环或者文件夹循环播放，除非用户使用停止指令。

5、插播的原理请参见 3.4.6.

6、此插播的功能，支持 TF 卡、U 盘以及 SPIFLASH。插播的文件仅仅允许在同一个设备内进行。

7、在停止状态，可以直接播放 ADVERTn 文件夹里面的曲目，就像指定文件夹文件名播放一样。

只要是播放 ADVERTn 文件夹里面的曲目，就不存在插播的问题，也就是说，当前播放的“ADVERT1”里面的 001 文件，在没有播放完之前，还可以继续指定播放 002 文件[此时会打断 001 文件]。

3.4.17 复位和睡眠的说明[0x0A][0x0B][0x0C]

[进入睡眠模式]	7E FF 06 0A 00 00 00 FE F1 EF
[模块复位]	7E FF 06 0C 00 00 00 FE EF EF

1、因为芯片是属于硬解码，所以低功耗部分一直是得不到很好的解决，所以用户朋友请注意，需要低功耗的场合，请一定使用控制芯片供电的方法，详细请参考 3.4.5 章节的说明

2、芯片的复位指令为软复位，这点请用户朋友注意一下，因为芯片内部集成了 MAX831L 的看门狗芯片，采用的是内部的独立 RC 震荡时钟，虽然不准，但是确实独立的，这样可以保证芯片在超强干扰下会自动复位。复位的时间为 5S-8S。

3、在任何状态下，都可以发送复位指令让芯片复位。

3.4.18 指定文件夹循环随机播放[0x28]

指定01文件夹随机播放	7E FF 06 28 00 00 01 FE D2 EF
指定02文件夹随机播放	7E FF 06 28 00 00 02 FE D1 EF
指定03文件夹随机播放	7E FF 06 28 00 00 03 FE D0 EF

1、指定文件夹随机播放是我们制定的扩展功能，默认文件夹的命名方式为"01","11"这样的方式，为了系统的稳定性和歌曲切换的速度,每个文件夹下默认最大支持 255 首歌,最多支持 99 个文件夹	 截图说明文件夹的命名方式
2、SPIFLASH 的操作和 TF 卡以及 U 盘一致，只是 SPIFLASH 的空间有限,能存放的语音数量有限 这个还需要根据客户对音质的要求进行压缩语音,来存放更多的语音信息	

3.4.19 设定芯片的地址[0xC0]

设定芯片地址为1	7E FF 06 C0 00 00 01 EF
设定芯片地址为99	7E FF 06 C0 00 00 63 EF

这里配置文件增加至 5 位，举例说明上面的截图“12001”. (1)、“1”代表的是按键的功能 (2)、“20”代表的是音量 **(3)、“01”代表的是芯片的地址。这里地址分两种，一个自己这样设置的。可以设置为 99。这个代表的是 10 进制，芯片返回地址就是 0x63。设置地址的时候，一定要不要超过 99。** (4)、**还有一个是超级地址，即 0xFF**。就是你串口发送指令，地址设置为“FF”，就不论你配置文件设置成什么，芯片都是有效的。 (2)、设置地址，这样 485 总线上就可以挂多个设备，相当于给每一个设备起一个唯一名字一样，这样就可以单独对每一个芯片进行控制。详细的可以去搜索 485 的原理	 举例： 7E FF 06 3A 00 01 01 xx xx EF --- 这个就是用了超级地址，-- 这个地址可以不用告诉用户 7E 01 06 3A 00 01 01 xx xx EF --- 这个就是 01 地址，必须和配置文件对应上才会有反应
同时支持串口发指令去设置这个地址 默认 TXT 的配置文件优先。指令如上表格举例 如果芯片没设置过地址，那么默认就是 0xFF	注意，地址芯片，芯片内部会记忆 下次上电有效

3.4.20 恢复出厂设置[0xC1]

恢复出厂设置	7E FF 06 C1 00 00 00 EF

这个只争对一些记忆型的参数设置，然后恢复到初始化状态

1、串口的波特率，出厂设置为 9600

2、芯片的地址，出厂设置之后为 0xff 。

3、其他的，后续可能添加的

3.5 串口查询指令详解

一些基本的控制播放的功能，就无需查询，此章节可以跳过不看

以下我们对关键的地方进行详细的说明--针对查询指令:

- 查询当前在线的设备
- 播放状态查询指令
- 指定文件夹曲目总数查询
- 当前设备的总文件夹数查询

3.5.1 查询当前在线的设备

查询在线设备	7E FF 06 3F 00 00 00 FE BC EF	U 盘正在播放

(1)、芯片在工作过程中，会不断的检测设备的在线情况，用户也可以通过 0x3F 这条指令进行查询

(2)、举例说明，如果芯片返回的数据为 7E FF 06 3F 00 00 0A xx xx EF

DL=0x0A = 0000 1010 代表了 TF 卡和 FLASH 在线

如果 DL=0x1F= 0000 1111 代表了 U 盘、TF 卡、PC、FLASH 均在线

(3)、0x0F--低四位均代表一种设备。

3.5.2 播放状态查询指令

正在播放	7E FF 06 42 00 01 01 xx xx EF	U 盘正在播放
暂停播放	7E FF 06 42 00 02 02 xx xx EF	TF 卡播放过程中被暂停
停止播放	7E FF 06 42 00 04 00 xx xx EF	在 PC 连接下载模式
	7E FF 06 42 00 08 01 xx xx EF	FLASH 正在播放
	7E FF 06 42 00 10 00 xx xx EF	芯片处于睡眠

(1)、芯片在解码过程中会有 3 种状态对用户开放。用户可以通过指令查询获取芯片的当前状态

(2)、播放暂停是指，正在播放一首曲目，人为的发送指令暂停播放，

播放停止是指，一首曲目播放完毕，芯片就处于播放停止的状态

(3)、如果返回的数据为 7E FF 06 42 00 02 02 xx xx EF 代表的意思详解如下：

DH = 0x02 --- 代表的是当前是 TF 卡设备，

DL = 0x02 --- 代表的是当前“TF 卡播放过程中被暂停”

(4)、如果返回的数据为 7E FF 06 42 00 02 02 xx xx EF 代表的意思详解如下：

DH 的含义		DL 的含义	
0x01	当前设备是 U 盘	0x00	当前处于播放停止状态
0x02	当前设备是 TF 卡	0x01	当前处于正在播放状态
0x04	当前设备是 USB 盘	0x02	当前处于正在暂停状态
0x08	当前设备是 FLASH 盘		
0x10	当前是 SLEEP 模式		

3.5.3 指定文件夹曲目总数查询[0x4E]

查询文件夹曲目总数	7E FF 06 4E 00 00 01 FE AC EF	查询01文件夹的总曲目数
	7E FF 06 4E 00 00 0B FE A2 EF	查询11文件夹的总曲目数

1、如果用户按照我们设定的规则命名文件，“01”、“02”等等，这样就可以对这些文件夹里面的曲目总数进行查询。查询的有效文件包括 MP3、WAV。其它格式的文件忽视。

2、如果查询的文件夹为空[表示无有效文件]，那么串口会直接返回以下信息

显示查询文件夹出错

3.5.4 当前设备的总文件夹数目查询[0x4F]

查询文件夹总数	7E FF 06 4F 00 00 00 FE AC EF	查询当前设备的文件夹总数
返回文件夹总数	7E FF 06 4F 00 00 03 FE AC EF	有3个文件夹

(1)、用户可以对当前的设备进行文件夹总数的查询。我们只支持“根目录”下的文件夹的数目查询。不支持文件夹里面包含文件夹。另外请用户不要建立空的文件夹，这样会造成识别错误。

(2)、假如设备中有 5 个有效文件夹[文件夹里面有 MP3/WAV 文件]，一个空文件夹。那么查询文件夹的总数时，会返回有 6 个文件夹。所以建议用户不要建立空的文件夹。

(3)、TF 卡和 U 盘、SPIFLASH 是一样的。查询的是当前的设备，如果当前处于 U 盘播放状态，则查询到的是 U 盘内部根目录的文件夹总数

3.5.6 查询当前播放的音乐的总时间和已经播放的时间[0x80][0x81]

查询当前播放的曲目总时间	7E FF 06 80 00 00 00 FE 7B EF
查询当前播放的曲目已经播放的时间	7E FF 06 81 00 00 00 FE 7A EF

这里我们扩展了两条指令，用于给用户查询当前播放的曲目总时间和已经播放的时间

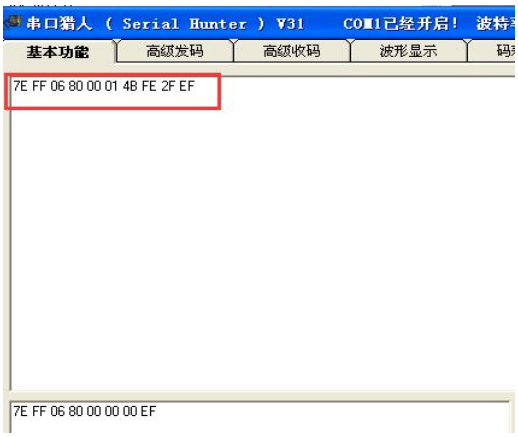
1、当前播放的曲目的总时间，这个是每个 MP3 文件已经固定了的，直接读取 MP3 文件就可以获取这个时间

2、当前播放的曲目已经播放的时间，我们内部是通过读取 MP3 文件获取的，并不是内部自己定时计数产生的，所以是很准的

3、如下截图，是我们实际测试中，芯片返回的数据 7E FF 06 80 00 01 4B FE 2F EF

其中 01 4B 就是芯片内部获取的总时间，合并起来就是 0x014B 转换为十进制就是 331

我们推荐的算法是 331/60 = 5 分钟，331%60 = 31 秒。这样就得出这首曲目的时间为 5 分 31 秒



4、用户在使用的过程中，如果要显示歌曲的播放时间，我们建议用户不要频繁的去查询，可以获取总时间之后，自己的 MCU 内部做一个计时处理，因为一般的播放时间不需要要求很准的。当然如果要去频繁的查询也是可以的。

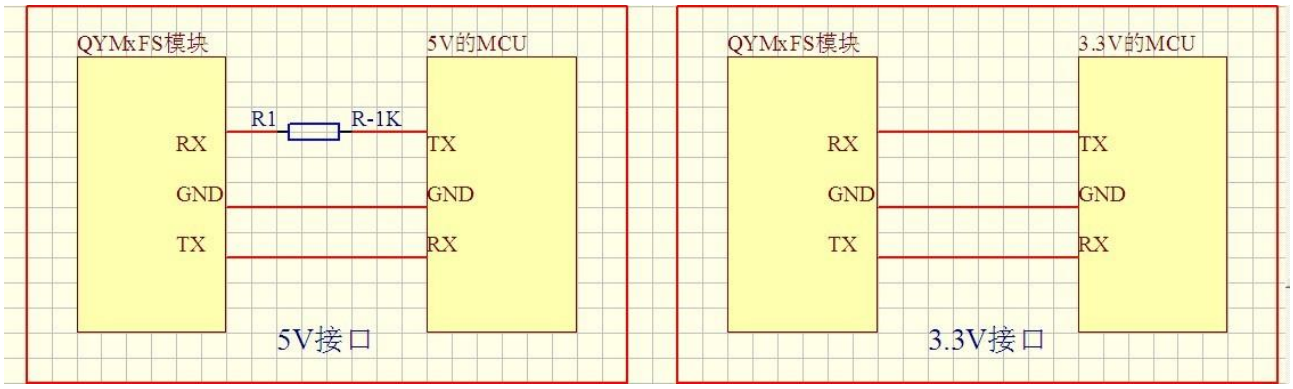
5、查询歌曲的总时间和当前已经播放的时间，请一定要在当前曲目解码成功之后，才能获取正确的时间。否则查询回来的就是 0x0000.

4. 参考电路

争对芯片的应用，我们提供了详细的设计参考，让您可以更快的上手体验到该芯片的强大功能

- 串行通信接口，波特率默认 9600，可以根据客户的要求修改
- 外部的 IO 按键的功能可以按照客户需求订制
- 外部单声道功放参考电路

4.1 串行接口



1、芯片的串口为 3.3V 的 TTL 电平，所以默认的接口的电平为 3.3V。

2、如果系统是 5V。那么建议在串口的对接接口串联一个 1K 的电阻。这样足以满足一般的要求，

3、如果应用于强电磁干扰的场合，请参考“注意事项”的说明。

4、芯片在 5V 和 3.3V 的系统中均正常的测试过，一切正常。均采用的是直连的方式，并没有串 1K 的电阻。一般的芯片都是能够兼容 3.3V 和 5V 的电平。

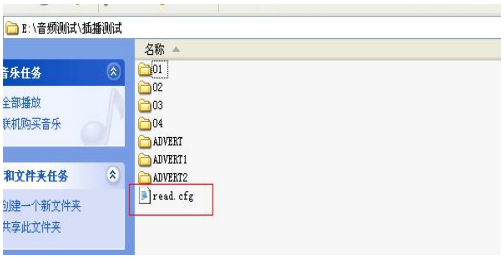
5、但是用户在实际的产品开发过程中，一定要严格的测试，留意电平的转换。强烈建议用户在能修改的条件下，使用 3.3V 的 MCU，响应环保、低功耗的号召。

4.2 按键接口

由于 KT404C 芯片可以被用来做按键的 GPIO 只有 GPIOA3，所以在按键这一块，我们没做什么强大的功能，仅仅只做了按键按下就整体播放 FLASH 里面的所有语音。但是我们预留了一下说明的功能，这个需要根据用户的需求去定制了，或者可以参考 KT404C 这一颗芯片，

4.2.1 通过 CFG 文件来配置

由于我们的方案支持 FAT 文件系统，所以设备中的文件的内容，可以很轻易的读出来，所以我们就扩展了此功能，通过设备内的 read.cfg 文件来读取按键的配置，后期我们还会扩展更多的功能，请拭目以待



(1)、配置文件必须命名为 read.cfg，这样才能被系统所识别，其它的命名系统则不识别，原理是这样的，芯片上电之后初始化文件系统，会首先搜索 read.cfg。这个文件，搜索到之后，对里面的数据进行读取和处理。请严格的按照我们给出的配置参数的方法,文件的属性必须是“xxx.txt”文件

(2)、配置文件在同一个设备中，最多只能有一个，可以没有，但是一定不能出现两个或者以上，否则会导致识别出错。

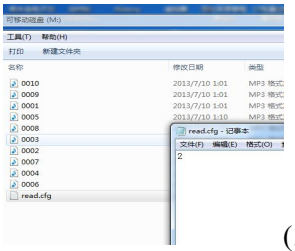
(3)、如果要配置 FLASH 模式下的按键功能，必须要要在 FLASH 里面建立一个 read.cfg 文件

如果要配置 TF 卡模式下的按键功能，也要在 TF 卡里面建立一个 read.cfg 文件。U 盘也是一样。也就是说任何一个设备需要配置按键功能，都需要建立一个 read.cfg 在相应的设备中。

(4)、配置文件的参数取值为 0-F

数值	对应的按键状态	数值	对应的按键状态
0	一对一触发播放[可打断]	8	reserve
1	按键抬起停止播放	9	reserve
2	一对一触发播放[不可打断]	A	reserve
3	标准 MP3功能	B	reserve
4	指定文件夹文件名播放	C	reserve
5	按下播放一段/抬起再播放一段	D	reserve
6	reserve	E	reserve
7	reserve	F	reserve

(5)、错误举例说明



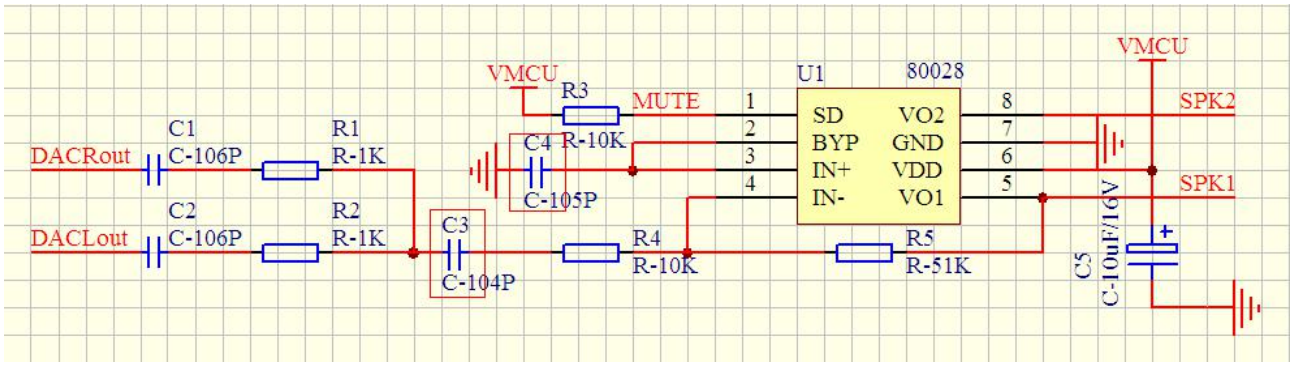
(左图)

(右图)

很明显左图是错误的，因为这样配置下来，实际的文件就是 read.cfg.txt，这样是不被系统识别的

右图是对的，用户可以参考

4.2 外接单声道功放



这里功放我们采用的是 CSC8002，具体参数请参考 IC 的 datasheet。应用于一般场合足以，如果追求更高的音质，请客户自行寻找合适的功放。上图中红色的框框的参数，请按照我们给出的推荐值，这样在调试的过程中，就会少很多杂音

4.5 USB 更新语音说明[业内首创功能]

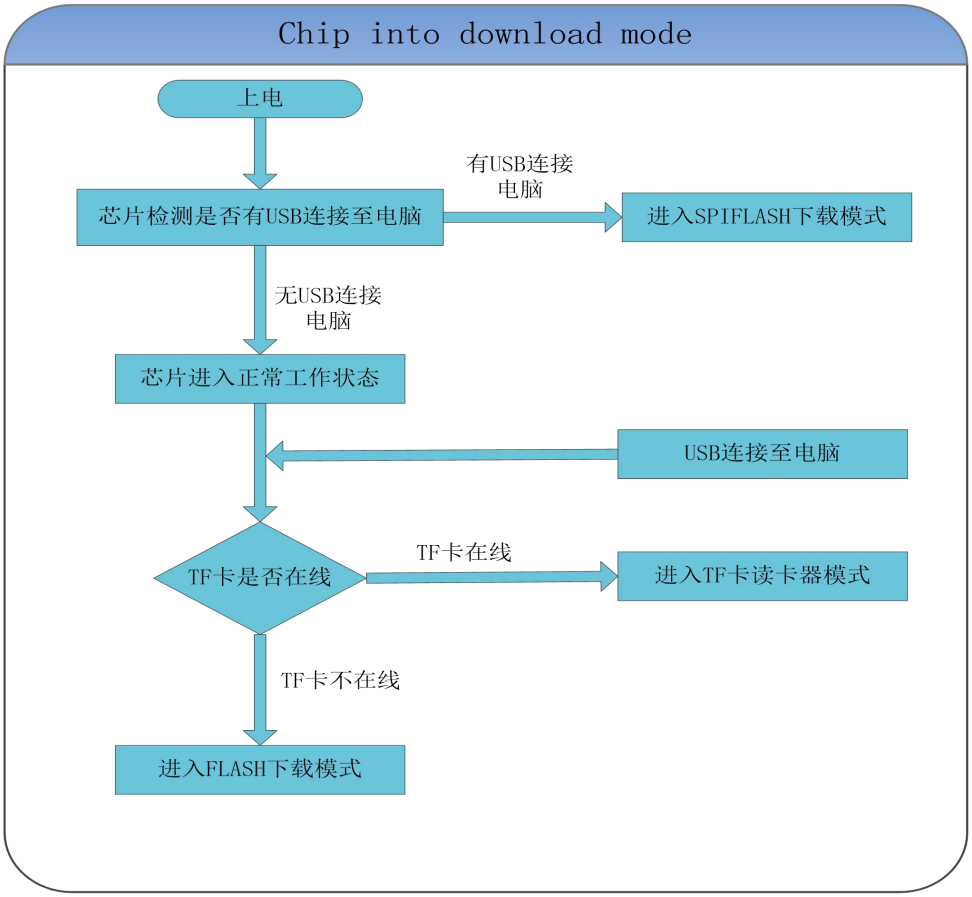
我们的芯片可以使用手机充电线直接更新语音，方便、灵活。这里分两种设备

- USB 更新 SPIFLASH 的语音模式
- USB 更新 TF 卡内的语音模式

其实 SPIFLASH 和 TF 卡，在插入 USB 连接电脑，原则上是一致的，使用的都是 MASSSTORAGE 协议。但是目前技术上面，暂时还没办法实现插上 USB 连接电脑，同时显示 TF 卡和 SPIFLASH 的盘符功能。这里就分为两种操作，针对芯片说明

1、芯片一上电检测到 USB 连接电脑，则进入 SPIFLASH 的读卡器功能

2、芯片上电没检测到 USB 连接电脑，进入正常工作模式。如果有 USB 连接至电脑，则进入 TF 卡读卡器模式。如果此时没有 TF 卡在线，则还是进入 SPIFLASH 的盘符模式。

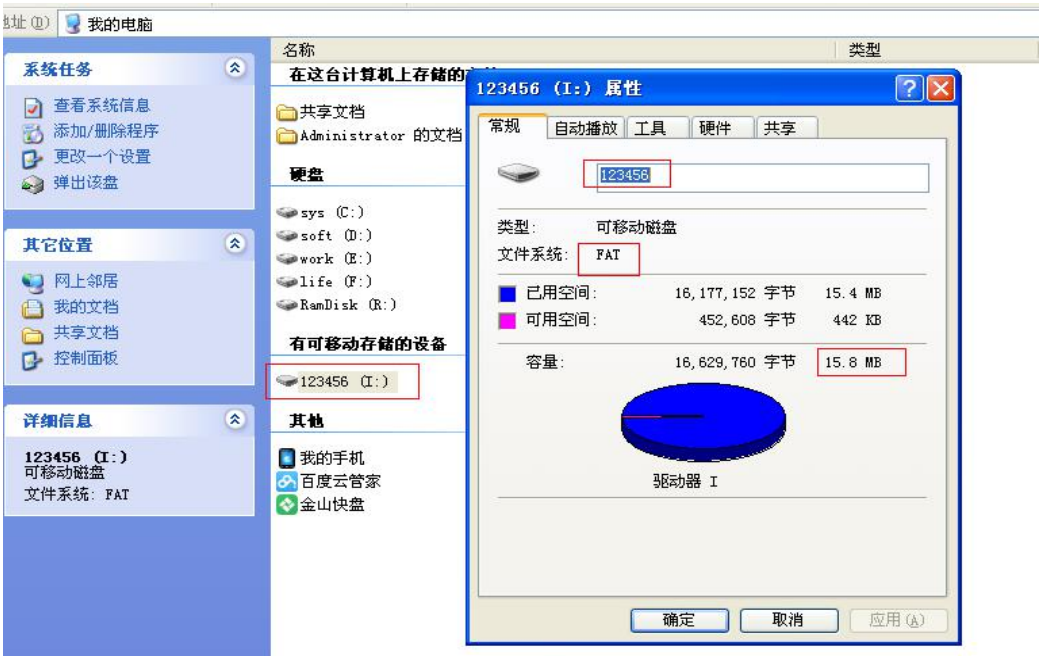


4.7.1 USB 更新 SPIFLASH 的语音详细说明

我们的模块可以使用手机充电线直接更新语音，方便、灵活。我们的优势如下

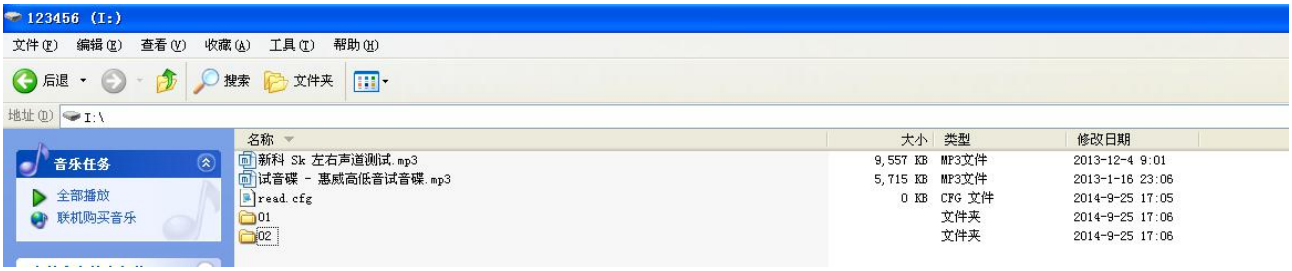
- 可以按照客户的要求，更正下载语音的窗口信息
- 无需安装任何软件，直接更新，也不需要专用下载器
- 对音质无任何压缩和损坏，保证更高的音质体验

1、插上我们模块的 USB 之后,可以以 SPIFLASH 作为存储介质的 U 盘，如下图



(1)、可以从上图看到 FLASH 的总容量为 15.8M 字节。已经使用的空间为 15.4M 字节。虚拟出来的设备的文件系统的为 FAT 格式。FAT 文件系统占的存储空间为 442K

(2)、进入设备之后，如下图



可以很清晰的看到设备里面的文件，以及文件名称。可以像操作 U 盘或者读卡器一样操作 FLASH.只是速度会比他们慢。至于为什么后面会详细解释。

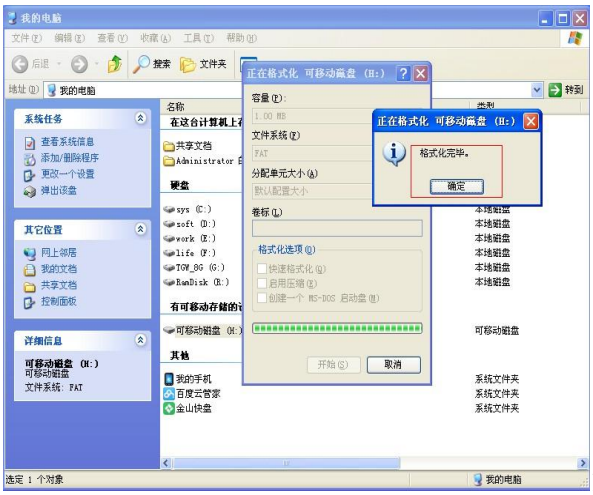
(3)、无论用户使用多大容量的 FLASH，我们模块都是支持的，并且内部已经做了自动识别，无需用户操心，用户也仅仅需要根据自己的需求来确定 FLASH 的容量和型号。

(4)、目前经过我们反复的测试和验证，SPIFLASH 支持最大的容量为 16M 字节，对应型号 W25Q128 因为再大容量的 FLASH。由于技术的原因暂时还没有突破，后续会直接更新，请用户知晓

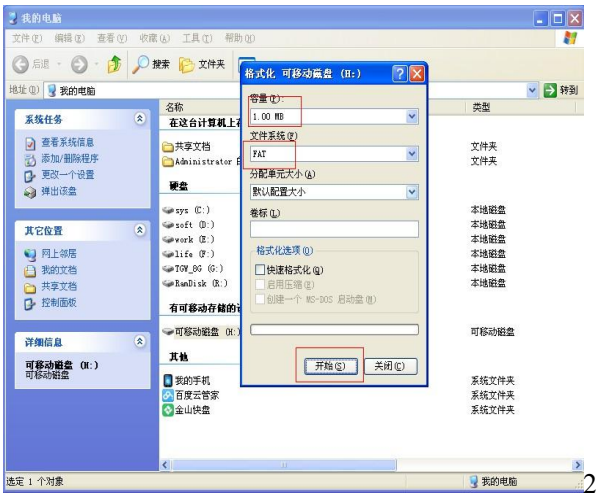
4.8 用户使用空白的 FLASH 说明

用户在调试的过程中，会按照自己的需求更换 FLASH 的大小来满足自己的需求，这样就需要以下三个步骤来完成 FLASH 的替换。

- 将新的空白的 FLASH 焊接在板子上面
- 通过 USB 接口对空白的 FLASH 进行枚举和格式化
- 格式化完毕，就可以像使用 U 盘一样使用

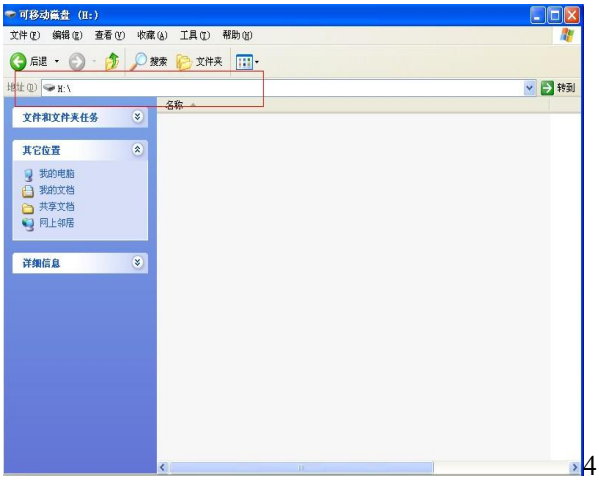


1



2

3



4

如上面的 4 个图片，就是使用空白 FLASH，FLASH 的型号为 W25Q80，容量为 1Mbyte。使用 USB 连接电脑第一次的处理过程。上面的截图可以很详细的看出步骤

1、我们的方案目前最大支持 16M 字节的 FLASH。换算为 FLASH 一般的型号，如：W25Q128.但是市面上请用户注意，并且封装还不是 SOP8 的

型号	容量	封装
W25Q80	1Mbyte	SOP8L[宽体]
W25Q16	2Mbyte	SOP8L[宽体]
W25Q32	4Mbyte	SOP8L[宽体]
W25Q64	8Mbyte	SOP8L[宽体]
W25Q128	16Mbyte	SOP8L[宽体]

2、我们的芯片支持自动识别 FLASH 的容量大小。所以用户无需关心，只需要按照自己的需求使用合适大小的 FLASH 即可

3、因为空白的 FLASH 里面什么都没有，所以拿到空白的 FLASH 第一件事情，就需要对 FLASH 进行格式化。将文件系统的链表写入 FLASH 中。

4、等到格式化成功之后，再拔掉 USB，再插上 USB 之后，就可以进行 SPIFLASH 的读写了。另外不同容量的 SPIFLASH，格式化的所需要的时间长度是不一致的。也就是说，FLASH 的容量越大，格式化所需要的时间越长。

5、经过我们大量的测试，基本市面上大部分的 FLASH 都是支持的，如：GD[兆易]、华邦、旺宏、飞索、港宏等等市场上最常见的。都是无缝支持的，这点请用户朋友放心。

5. 注意事项

芯片的使用,关键的地方做如下说明:

- 芯片的 GPIO 的特性
- 应用的中注意事项
- 串口编程部分的注意

5.1 GPIO 的特性

IO 输入特性						
符号	参数	最小	典型	最大	单位	测试条件
VIL	Low-Level Input Voltage	-0.3	-	0.3*VDD	V	VDD=3.3V
VIH	High-Level Input Voltage	0.7VDD	-	VDD+0.3	V	VDD=3.3V
IO 输出特性						
符号	参数	最小	典型	最大	单位	测试条件
VOL	Low-Level Output Voltage	-	-	0.33	V	VDD=3.3V
VOH	High-Level Output Voltage	2.7	-	-	V	VDD=3.3V

5.2 应用中的注意点

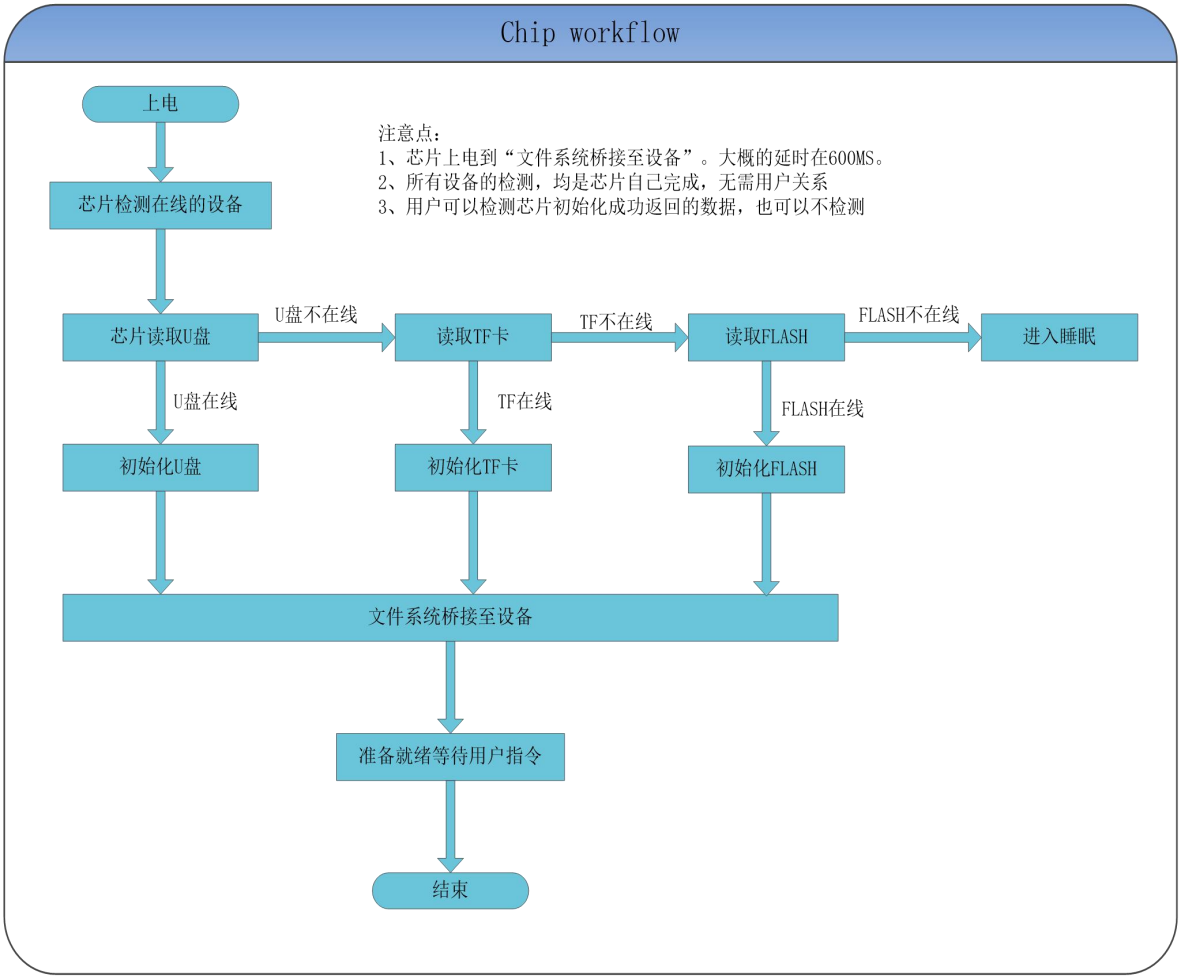
1、芯片对外的接口均是 3.3V 的 TTL 电平，所以在硬件电路的设计中，请注意电平的转换问题。另外在强干扰的环境中，请注意电磁兼容的一些保护措施，GPIO 采用光耦隔离，增加 TVS 等等
2、串口通信，在一般的使用环境下，注意好电平转换即可。如果强干扰环境，或者长距离的 RS485 应用，那么请注意信号的隔离，严格按照工业的标准设计通信电路。可以联系我们，我们提供设计参考
3、我们支持音频文件的采样率最低为 8KHZ。也就是说低于 8KHZ 的音频文件是不支持的，不能正常解码播放。用户可以使用音频处理软件，提高音频文件的采样率来解决这个问题。
4、芯片在睡眠状态的电流在 10MA 左右，播放中，依据音量的大小以及外接的喇叭负载，峰值电流可以达到 1A。功耗会比较大。如果使用在低功耗场合，请用户控制芯片的供电。这样可以减小芯片的功耗
5、用户如果直接使用我们芯片自带的功放，请选择合适的喇叭即可。推荐使用 4 欧姆/3W。这个是使用效果最好的配置。选用其它的喇叭，请注意负载大小，以及功率这两个参数
6、芯片支持 MP3、WAV 二种主流音频格式。推荐使用 MP3 格式，因为这样就更节省 FLASH 的容量。
7、我们的芯片支持 8/11.025/12/16/22.05/24/32/44.1/48KHZ 采样率的音频文件，这些也是网络上绝大多数的音频文件的参数。如果用户的音频文件的采样率不在此范围内，是不支持播放的，但是可以通过专用的软件转换一下即可。我们的优势就是无压缩播放和高音质，所以不太建议用户对音频进行压缩。

5.3 注意事项点

串口部分的操作，参见下面的流程，我们提供了完整的参照例程，供用户参考:

- 芯片上电的流程
- 串口编程参考的说明
- 串口操作需要延时的注意事项
- 串口协议的校验的说明
- 串口协议的校验的算法详解
- 外接 MCU 的晶振选择说明
- 芯片的播放说明[物理顺序等等]

5.3.1 芯片上电的工作流程图



1、我司提供的所有芯片的串口部分的操作，均是一样的协议，所以不用担心不同芯片的不兼容

2、如果对串口的操作，有任何不明白的，请一定联系我们，索取串口编程参考例程。

3、我们产品的更新，也一定会按照当前的协议版本，做到向下兼容。

5.3.2 串口编程参考的说明

目前我们提供的串口编程参考代码，有两部分
(1)、第一部分是我们测试版的测试代码，相关的串口操作比较全面
(2)、另一个是基本版，只是指定曲目的范例。请用户耐心消化

5.3.3 串口编程需要适当延时的注意点

1、芯片上电之后，需要大概 1S-1.5S 时间进行初花的相关操作，初始化完毕之后，会有初始化的相关数据发送出来。用户也可以直接不理会这些数据

2、当指定设备播放之后，需要延时 200ms 的时间，再发送指定曲目等等相关指令。

3、因为芯片自带文件系统，正常情况下，在曲目不大于 1000 首的话，响应速度是低于 50ms 的
曲目超过 3000 首之后，文件系统的切换速度会变慢一点，响应速度在 100ms --- 1S 之间不等

4、芯片内部对串口的处理是 10MS 处理一次，所以连续的指令发送时，必须要间隔 20MS 的延时。否则前面的指令将会被覆盖而得不到执行

5、如果指定文件夹文件名播放[0x0F、0x14]延时必须大于 40ms，因为芯片锁定文件是需要时间的。
只要涉及到文件夹文件名查找的相关指令，40MS 的延时是必不可少的。如果芯片当前正在查找文件，串口的数据过来太频繁，会导致芯片的工作不正常

5.3.4 校验的重要说明

1、针对很多用户不太习惯校验的通信方式，我们特别推出了带校验和不带校验的兼容方式。举例说明。如果我们发送组合播放指令如下:

组合播放[不带校验]	7E FF 09 21 00 05 01 02 03 04 EF	播放5、1、2、3、4
组合播放[带校验]	7E FF 09 21 00 05 01 02 03 04 FE C8 EF	播放5、1、2、3、4

比较两条指令的区别，就是省略掉的校验的 2 个字节。这两帧数据均可以被芯片所接收。
2、因为很多用户在使用的过程中，很多都是使用不带晶振的 MCU。这样的话，我们必须建议您加上校验这种方式，来保证通信的稳定性。
3、假如用户使用 STM32 或者 STC 等等 MCU，并且是外挂晶振的，就可以适当的省掉校验。因为不带晶振的 MCU，时钟是相对不那么准的，所以串口会存在误差，一旦误差过大，会导致通信出错。请用户朋友自行斟酌。

5.3.5 校验的计算说明

发送的指令，去掉起始和结束。将中间的 6 个字节进行累加，最后取反再+1 得到校验码。接收端就将接收到的一帧数据，去掉起始和结束。将中间的数据累加，再加上接收到的校验字节。刚好为 0.这样就代表接收到的数据完全正确。

举例说明：

举个例子，如果我们播放下一曲，就需要发送:7E FF 06 01 00 00 00 FE FA EF

数据长度为 6 ,这 6 个字节是[FF 06 01 00 00 00] 。不计算起始、结束、和校验。校验字节为 FE FA。

发送端得到校验码的过程：

相加的过程: FF+06+01+00+00+00 = 0x0106

取反加 1 的过程：0x0106 取反 = FEF9 / 再+1 = 0xFEFA，和我们的校验值对比一下

接收端进行校验的过程:

相减的过程: 0 - 0x0106 = 0xFEFA 。再和我们的结果比较一下。

最后可以参考一下我们给出的参考例程，或者直接移植我们给出的函数。就很容易理解了

5.3.6 MCU 的晶振选择

1、原则上我们建议用户，使用 11.0592MHZ 或者相倍数的晶振。这样可以让串口产生 9600 的波特率会更准确。我们的芯片串口误差是允许在 3%以内的

2、如果用户在 12M 的晶振时。首先要做如下判断，

(1)、看是什么 MCU，51 或者 PIC、STM32 等等，基本都自带波特率发生器，所以产生 9600 的波特率基本没压力。

(2)、看 MCU 是否为硬件串口，如果是 IO 模拟的串口的话，强烈建议用户使用 11.0592 的晶振

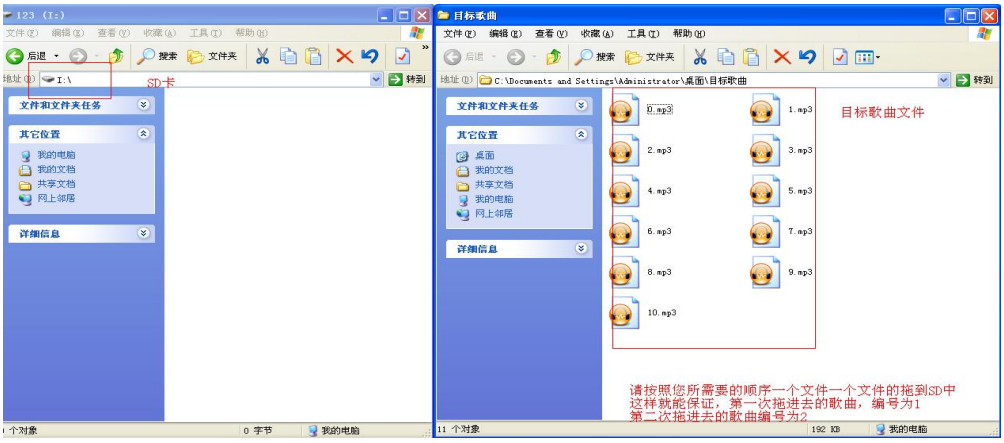
(3)、标准的 51，如：STC89C52 或者 AT89C52 等等都是采用定时器产生波特率的，经过简单的计算就可以算出，12M 晶体做 9600 波特率的误差是是 0.16%，正常运行是没有任何问题，但还是需要用户进行全面测试

5.3.7 指定播放的说明

- 指定曲目播放[按照物理顺序]
- 指定曲目播放[按照文件夹和文件名的名称]

1、按照物理顺序指定曲目播放:

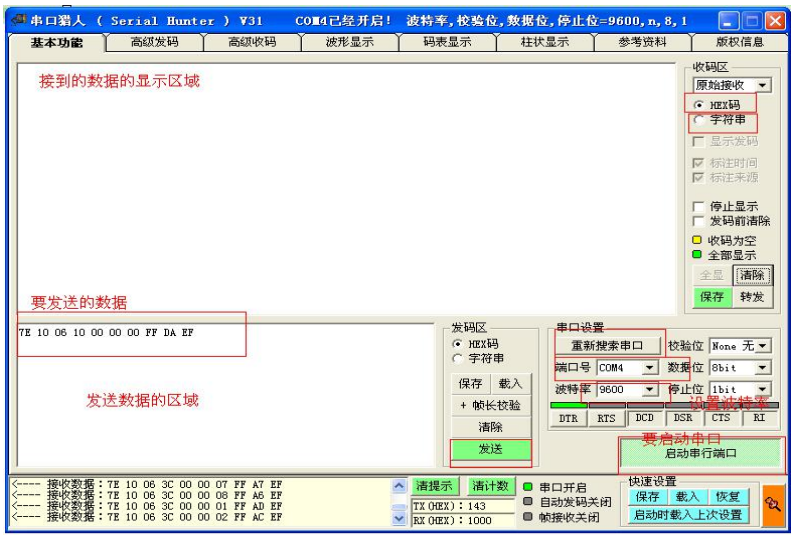
需要按照下图的方法往 SPIFLASH 中拷贝歌曲。先拷贝进设备的，编号为第一首，以此类推



2、指定文件夹和文件名播放

详情请参考芯片数据手册的“3.4.6 指定文件夹文件名的播放”章节

5.3.8 串口调试说明



1、首先现在网上下载一个软件”串口猎人”，为什么选择这个，因为它有自动搜索串口的功能，十分好用。打开之后的主界面如上图所示，可以看到红色部分，依次设置即可

2、打开软件，首先要搜索串口，找到指定的端口之后，指定“波特率”，我们的芯片默认的波特率为 9600，最后就是“启动串行端口”，这样软件就配置好了。这里有两个概念需要明确一下

第一是“HEX 码”，我们默认是这个，这个是用来显示数据的。所以必须设置这里

第二是“字符串”，这个是用来显示打印字符的，我们这里用不到。

(3)、软件配置 OK 之后，将需要的指令复制到发送区域即可。具体的指令请参照芯片的数据手册

(4)、如果芯片的数据手册没有的测试指令的话，请自行计算，尤其需要注意的是“校验和”这两个字节，如何计算不对的话，芯片是不接受指令的,同时会返回相应的错误指令

5.3.9 校验代码的移植

我们这里的说明，争对的是用户的 MCU 给我们的芯片发送控制指令

1、发送端 --- 用户的 MCU

```
/*******************************************************************************
 - 功能描述：  串口向外发送命令[包括控制和查询]
 - 参数说明：  CMD:表示控制指令，请查阅指令表，还包括查询的相关指令
               feedback:是否需要应答[0:不需要应答，1:需要应答]
               data:传送的参数
*******************************************************************************/
void Uart_SendCMD(INT8U CMD ,INT8U feedback , INT16U dat)
{
    Send_buf[0] = 0xff;      //保留字节
    Send_buf[1] = 0x06;      //长度
    Send_buf[2] = CMD;       //控制指令
    Send_buf[3] = feedback;//是否需要反馈
    Send_buf[4] = (INT8U)(dat >> 8);//datah
    Send_buf[5] = (INT8U)(dat);       //datal
    DoSum(&Send_buf[0],6);            //校验
    SendCmd(8);           //发送此帧数据
}
```

DoSum(&Send_buf[0],6); 这里是指对 Send_buf[0]---Send_buf[5]这 6 个字节进行校验的算法生成校验的两个字节，并且存储于 Send_buf[6]和 Send_buf[7]中

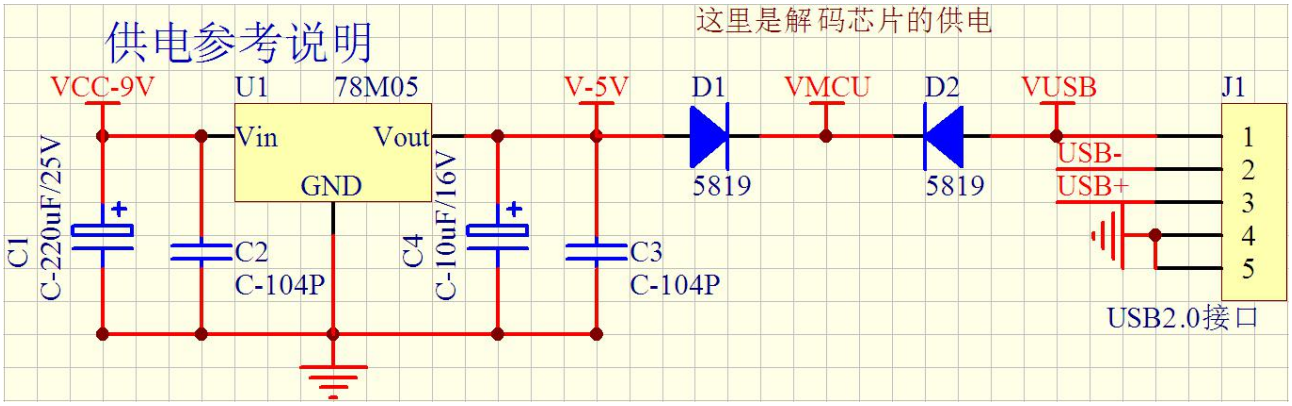
```
/*******************************************************************************
 - 功能描述：求和校验
 - 和校验的思路如下：
    发送的指令，去掉起始和结束。将中间的 6 个字节进行累加，最后取反码。接收端就将接收到的一帧数据，去掉起始和结束。将中间的数据累加，再加上接收到的校验字节。刚好为 0.这样就代表接收到的数据完全正确。
*******************************************************************************/
void DoSum( INT8U *Str, INT8U len)
{
    INT16U xorsum = 0;
    INT8U i;
    for(i=0; i<len; i++)
    {
        xorsum    = xorsum + Str[i];
    }
    xorsum        = 0 -xorsum;
    *(Str+i)     = (INT8U)(xorsum >>8);//得到高字节
    *(Str+i+1) = (INT8U)(xorsum & 0x00ff);//得到低字节
}
```

2、接收端，芯片发送数据给用户的 MCU

```
If(一帧数据接收完毕)
{
    for(i=0; i<(*(pi+1)); i++)//这里 pi 指向的是接收缓冲区，*(pi+1)是获取数据长度。
    {
        xorsum    = xorsum + pi[i] ;//将接受到的数据进行累加
    }
    xorsum1 = ((u16)((*(pi+i))<<8)) | (*(pi+i+1));//这里是接收到的校验字节，16 位
    xorsum    = xorsum + xorsum1;//将接收到的校验字节和自己算的校验数据相加
    if(!xorsum)
    {
        Uart_Task(pi);//串口处理--对接收到的指令进行处理
    }
    else
    {
         ErrorStatus = ERROR_CheckSError ;//接收校验出错
         //校验码出错之后的处理
    }
}
```

3、用户无论使用的是什么 MCU，这两个函数均可以平行的移植到自己的程序中。

5.3.10 芯片或者芯片的供电说明



这里为什么增加两个二极管，分两种情况说明

(1)、外部只需要外接 U 盘播放，那么 D2 是可以省略掉的

(2)、如果外接电脑，则需要 D2，这样是为了防止 7805 前端无电压输入时，直接插上电脑，会把电脑端的 USB 电压拉低，导致不正常。

1、我们的芯片或者芯片，供电的范围是 3.3V--5V。不可以超过 5V，否则会造成芯片的永久性损坏。

2、我们的芯片，是音频类的产品，对电源的纹波是有要求的，建议用户最好使用线性电源[带变压器的电源]，后级使用 7805 之类的线性稳压芯片供电。

3、7805 后一级最好增加一个二极管，这里的 4148 其实并不是最合适的，因为 4148 的正向电流只有 500MA。如果我们后级的功放功率过大，会导致 4148 永久性损坏。这里选用 IN4001 或者 IN4007 才是最合适的。

4、很多用户在应用过程中，往往很多供电是 12V 或者 9V，如果用户使用 7805 之类的线性稳压 IC 时，一定要注意芯片的发热，线性稳压 IC 的原理，基本上都是将多余的电压以热量的形式表现出去。举个例子，假如 12V 输入，经过 7805 之后，压差为 7V。假如后级的耗电流为 200MA，那么 7805 产生的热量就是 1.4W，这个热量就很烫手了，会导致 7805 过热自保护，所以这样的硬件设计是很不合理的。所以此时可以选用合适的 DCDC 芯片，我们推荐使用 LM2596 之类的纹波小的芯片

6. 免责声明

■ **开发预备知识**

KT 系列产品将提供尽可能全面的开发模版、驱动程序及其应用说明文档以方便用户使用但也需要用户熟悉自己设计产品所采用的硬件平台及相关 C 语言的知识

■ **EMI 和 EMC**

KT 系列芯片机械结构决定了其 EMI 性能必然与一体化电路设计有所差异。KT 系列芯片的 EMI 能满足绝大部分应用场合，用户如有特殊要求，必须事先与我们协商.

KT 系列芯片的 EMC 性能与用户底板的设计密切相关，尤其是电源电路、I/O 隔离、复位电路，用户在设计底板时必须充分考虑以上因素。我们将努力完善 KT 系列芯片的电磁兼容特性，但不对用户最终应用产品 EMC 性能提供任何保证.

■ **修改文档的权力**

KT 工作室能保留任何时候在不事先声明的情况下对 KT 系列产品相关文档的修改权力

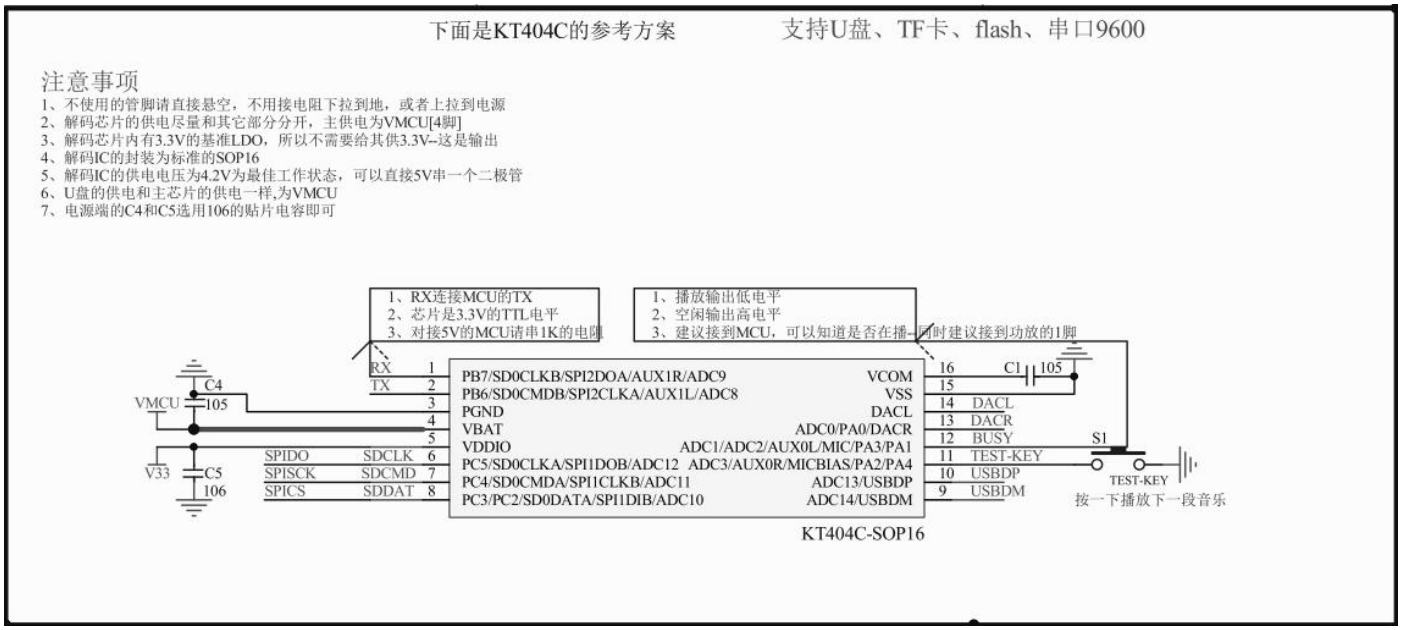
■ **ESD 静电放点保护**

KT 系列产品部分元器件内置 ESD 保护电路，但在使用环境恶劣的场合，依然建议用户在设计底板时提供 ESD 保护措施，特别是电源与 IO 设计，以保证产品的稳定运行，安装 KT 系列产品为确保安全请先将积累在身体上的静电释放，例如佩戴可靠接地的静电环，触摸接入大地的自来水管等

7. 订货信息

我们默认出货都是管装，每管 50pcs。一盒是 5K。一箱是 50K

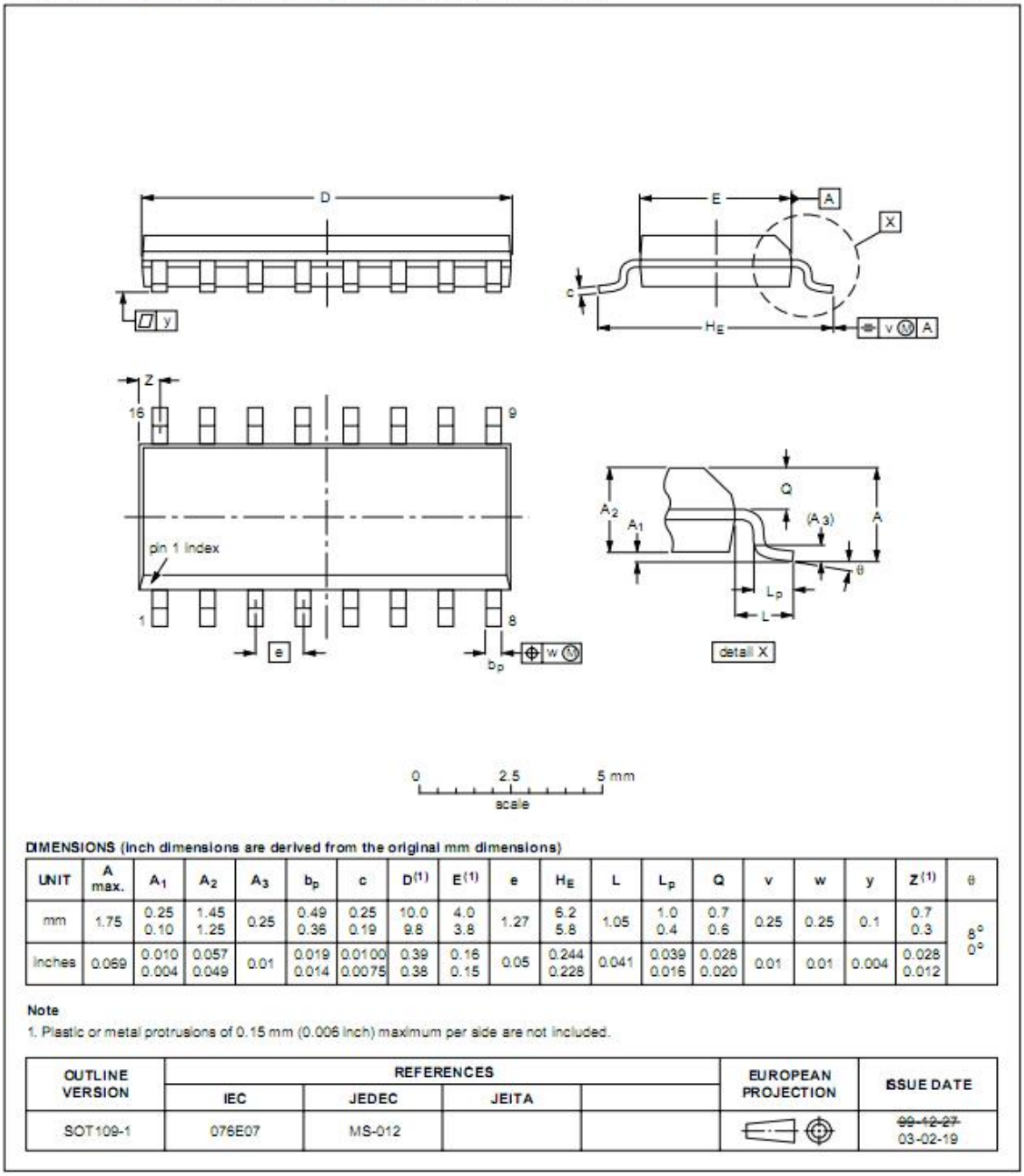
7.1 参考原理图



7.2 封装尺寸

我们的封装和常用的逻辑芯片是一致的，如：74HC595、74HC138 等等 SOP16 封装的芯片

SO16: plastic small outline package; 16 leads; body width 3.9 mm **SOT109-1**



DIMENSIONS (inch dimensions are derived from the original mm dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_E	L	L_p	Q	v	w	y	$Z^{(1)}$	θ
mm	1.75	0.25 0.10	1.45 1.25	0.25	0.49 0.36	0.25 0.19	10.0 9.8	4.0 3.8	1.27	6.2 5.8	1.05	1.0 0.4	0.7 0.6	0.25	0.25	0.1	0.7 0.3	8° 0°
inches	0.069	0.010 0.004	0.057 0.049	0.01	0.019 0.014	0.0100 0.0075	0.39 0.38	0.16 0.15	0.05	0.244 0.228	0.041	0.039 0.016	0.028 0.020	0.01	0.01	0.004	0.028 0.012	

Note

1. Plastic or metal protrusions of 0.15 mm (0.006 inch) maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT109-1	076E07	MS-012				~~99-12-27~~ 03-02-19

深圳市清月电子科技有限公司

技术支持：13510250437

淘宝网址：http://qyvhome.taobao.com/

百度分享：http://pan.baidu.com/s/1F5JVS [辅助资料可以在这里下载]

8. 参考例程

```
/*********************************************************************************
 - 实现功能：实现芯片上电分别指定播放第一曲和第二曲，基本的程序供用户测试
 - 日期      ：2013-05-06
 - 运行环境：STC     晶振：11.0592M        波特率:9600
 - 备注      ：在普中科技的 51 开发板上调试 OK --- STC89C516RD+
          该测试程序必须是芯片或者芯片方案中有设备在线，譬如 U 盘、TF 卡、FLASH
*********************************************************************************/
#include "REG52.h"

#define COMM_BAUD_RATE    9600      //串口波特率
#define OSC_FREQ            11059200      //运行晶振：11.05926MHZ
static INT8U Send_buf[10] = {0} ;

void Delay_Ms(INT32U z)
{
      INT32U x=0 , y=0;
      for(x=110 ; x>0 ;x--)
      for(y=z; y>0;y-- );
}

/*********************************************************************************
 - 功能描述：  串口 1 初始化
 - 注：            设置为 9600 波特率
*********************************************************************************/
void Serial_init(void)
{
      TMOD = 0x20;                        // 设置 T1 为波特率发生器
      SCON = 0x50;                        // 0101,0000 8 位数据位，无奇偶校验
      PCON = 0x00;                        //PCON=0;
      TH1=256-(OSC_FREQ/COMM_BAUD_RATE/32/12);//设置为 9600 波特率
      TL1=256-(OSC_FREQ/COMM_BAUD_RATE/32/12);
     TR1        = 1;                    //定时器 1 打开
     REN        = 1;                    //串口 1 接收使能
     ES         = 1;                    //串口 1 中断使能
}
void Uart_PutByte(INT8U ch)
{
     SBUF    = ch;
     while(!TI){;}
     TI = 0;
}

/*********************************************************************************
 - 功能描述：  串口向外发送命令[包括控制和查询]
 - 参数说明：  CMD:表示控制指令，请查阅指令表，还包括查询的相关指令
                feedback:是否需要应答[0:不需要应答，1:需要应答]
                data:传送的参数
*********************************************************************************/
void SendCmd(INT8U len)
{
     INT8U i = 0 ;
```

```
    Uart_PutByte(0x7E); //起始
    for(i=0; i<len; i++)//数据
    {
        Uart_PutByte(Send_buf[i]) ;
    }
    Uart_PutByte(0xEF) ;//结束
}

/*******************************************************************************
 - 功能描述：求和校验 --- 用户也可以省略此校验，参见 5.3.4 的说明
 - 和校验的思路如下：
    发送的指令，去掉起始和结束。将中间的 6 个字节进行累加，最后取反码。接收端就将接收到的一帧数据，去掉起始
和结束。将中间的数据累加，再加上接收到的校验字节。刚好为 0.这样就代表接收到的数据完全正确。
*******************************************************************************/
void DoSum( INT8U *Str, INT8U len)
{
    INT16U xorsum = 0;
    INT8U i;
    for(i=0; i<len; i++)
    {
        xorsum    = xorsum + Str[i];
    }
    xorsum        = 0 -xorsum;
    *(Str+i)     = (INT8U)(xorsum >>8);
    *(Str+i+1) = (INT8U)(xorsum & 0x00ff);
}

void Uart_SendCMD(INT8U CMD ,INT8U feedback , INT16U dat)
{
    Send_buf[0] = 0xff;      //保留字节
    Send_buf[1] = 0x06;      //长度
    Send_buf[2] = CMD;       //控制指令
    Send_buf[3] = feedback;//是否需要反馈
    Send_buf[4] = (INT8U)(dat >> 8);//datah
    Send_buf[5] = (INT8U)(dat);       //datal
    DoSum(&Send_buf[0],6);            //校验
    SendCmd(8);           //发送此帧数据
}

void main()
{
    Serial_init() ;//串口寄存器的初始化设置
    Uart_SendCMD(0x03 , 0 , 0x01) ;//播放第一首
    Delay_Ms(1000) ;//延时大概 6S
    Uart_SendCMD(0x03 , 0 , 0x02) ;//播放第二首
    Delay_Ms(1000) ;//延时大概 6S
    Uart_SendCMD(0x03 , 0 , 0x04) ;//播放第四首
    while(1);
}
```

9. PC 端串口调试指令举例

用户可以通过电脑端的串口软件，对芯片进行测试。我们的芯片串口为 TTL 电平，请注意电平转换

- 控制指令
- 查询参数指令

9.1 控制指令

串口调试助手进行测试	发送的命令[带校验]	发送的命令[不带校验]	备注
[下一首]	7E FF 06 01 00 00 00 FE FA EF	7E FF 06 01 00 00 00 EF	
[上一首]	7E FF 06 02 00 00 00 FE F9 EF	7E FF 06 02 00 00 00 EF	
	7E FF 06 03 00 00 01 FE F7 EF	7E FF 06 03 00 00 01 EF	指定第一首播放
[指定曲目]	7E FF 06 03 00 00 02 FE F6 EF	7E FF 06 03 00 00 02 EF	指定第二首
	7E FF 06 03 00 00 0A FE EE EF	7E FF 06 03 00 00 0A EF	指定第10首
音量加	7E FF 06 04 00 00 00 FE F7 EF	7E FF 06 04 00 00 00 EF	
音量减	7E FF 06 05 00 00 00 FE F6 EF	7E FF 06 05 00 00 00 EF	
[指定音量]	7E FF 06 06 00 00 1E FE D7 EF	7E FF 06 06 00 00 1E EF	指定音量为30级
[指定 EQ]	7E FF 06 07 00 00 01 FE F3 EF	7E FF 06 07 00 00 01 EF	保留
	7E FF 06 08 00 00 01 FE F2 EF	7E FF 06 08 00 00 01 EF	循环播放第一首
	7E FF 06 08 00 00 02 FE F1 EF	7E FF 06 08 00 00 02 EF	循环播放第二首
[循环播放曲目]	7E FF 06 08 00 00 0A FE E9 EF	7E FF 06 08 00 00 0A EF	循环播放第十首
	7E FF 06 08 00 01 01 FE F1 EF	7E FF 06 08 00 01 01 EF	循环播放 FLASH 的 FOLDER1的第一曲
	7E FF 06 08 00 02 01 FE F0 EF	7E FF 06 08 00 02 01 EF	循环播放 FLASH 的 FOLDER2的第一曲
	7E FF 06 09 00 00 01 FE F1 EF	7E FF 06 09 00 00 01 EF	指定播放 UDISK
[指定播放设备]	7E FF 06 09 00 00 02 FE F0 EF	7E FF 06 09 00 00 02 EF	指定播放 TF
	7E FF 06 09 00 00 04 FE ED EF	7E FF 06 09 00 00 04 EF	指定播放 FLASH
[进入睡眠模式]	7E FF 06 0A 00 00 00 FE F1 EF	7E FF 06 0A 00 00 00 EF	
[唤醒睡眠]	7E FF 06 0B 00 00 00 FE F0 EF	7E FF 06 0B 00 00 00 EF	
[芯片复位]	7E FF 06 0C 00 00 00 FE EF EF	7E FF 06 0C 00 00 00 EF	
[播放]	7E FF 06 0D 00 00 00 FE EE EF	7E FF 06 0D 00 00 00 EF	
[暂停]	7E FF 06 0E 00 00 00 FE ED EF	7E FF 06 0E 00 00 00 EF	
[指定文件夹文件名]	7E FF 06 0F 00 01 01 FE EA EF	7E FF 06 0F 00 01 01 EF	"01"的文件夹，曲目为"001"
	7E FF 06 0F 00 01 02 FE E9 EF	7E FF 06 0F 00 01 02 EF	"01"的文件夹，曲目为"002"
停止播放	7E FF 06 16 00 00 00 FE E5 EF	7E FF 06 16 00 00 00 EF	停止软件解码

指定文件夹循环播放	7E FF 06 17 00 02 00 FE E2 EF	7E FF 06 17 00 02 00 EF	指定 FOLDER2循环播放
	7E FF 06 17 00 01 00 FE E3 EF	7E FF 06 17 00 01 00 EF	指定 FOLDER1循环播放
单曲循环播放	7E FF 06 19 00 00 00 FE E2 EF	7E FF 06 19 00 00 00 EF	单曲循环播放开启
	7E FF 06 19 00 00 01 FE E1 EF	7E FF 06 19 00 00 01 EF	单曲循环播放关闭
带音量播放	7E FF 06 22 00 1E 01 FE BA EF	7E FF 06 22 00 1E 01 EF	30级音量播放第1曲
	7E FF 06 22 00 0F 01 FE C9 EF	7E FF 06 22 00 0F 01 EF	15级音量播放第1曲
	7E FF 06 22 00 0F 02 FE C8 EF	7E FF 06 22 00 0F 02 EF	15级音量播放第2曲

9.2 查询参数指令

串口调试助手进行测试	发送的命令[带校验]	发送的命令[不带校验]	备注
查询在线的设备	7E FF 06 3F 00 00 00 FE BC EF	7E FF 06 3F 00 00 00 EF	
查询当前状态	7E FF 06 42 00 00 00 FE B9 EF	7E FF 06 42 00 00 00 EF	
[查询音量]	7E FF 06 43 00 00 00 FE B8 EF	7E FF 06 43 00 00 00 EF	
[查询当前 EQ]	7E FF 06 44 00 00 00 FE B7 EF	7E FF 06 44 00 00 00 EF	
U 盘总文件数	7E FF 06 47 00 00 00 FE B4 EF	7E FF 06 47 00 00 00 EF	当前设备的总文件数
TF 总文件数	7E FF 06 48 00 00 00 FE B3 EF	7E FF 06 48 00 00 00 EF	
FLASH 总文件数	7E FF 06 49 00 00 00 FE B2 EF	7E FF 06 49 00 00 00 EF	
U 盘当前曲目	7E FF 06 4B 00 00 00 FE B0 EF	7E FF 06 4B 00 00 00 EF	当前播放的曲目
TF 当前曲目	7E FF 06 4C 00 00 00 FE AF EF	7E FF 06 4C 00 00 00 EF	
FLASH 当前文件夹曲目指针	7E FF 06 4D 00 00 00 FE AE EF	7E FF 06 4D 00 00 00 EF	
指定文件夹曲目总数查询	7E FF 06 4E 00 00 01 FE AC EF	7E FF 06 4E 00 01 00 EF	
查询当前设备总文件夹数	7E FF 06 4F 00 00 00 FE AC EF	7E FF 06 4F 00 00 00 EF	支持 TF 卡和 U 盘、FLASH