
KT603C 芯片使用手册

文件状态：	文件标识：	QYVHOME
<input type="checkbox"/> 草稿	当前版本：	V1.1
<input checked="" type="checkbox"/> 正式发布	作 者：	
<input type="checkbox"/> 正在修改	完成日期：	2014-09-01



版本历史

版本	日期	原因
V1.0	2014/02/18	内测版本和优化版本
		1、增加 FLASH 的文件系统讲解 2、增加调试的方法，串口调试，按键调试 3、增加进入下载模式的详细说明 4、增加随机播放指令 5、增加一些框图详解，说明更直观 6、增加指定音量播放 7、增加组合播放功能 8、完善睡眠模式下的说明
V1.1	2014/09/01	9、增加芯片或者芯片的供电说明

目 录

1. 概述.....	6
1.1 简介.....	6
KT603C 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、TF 卡或者 U 盘作为存储介质，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。.....	6
1.2 功能.....	6
1.3 应用.....	6
2. 方案说明.....	7
2.1 参数说明.....	7
2.2 管脚说明.....	8
3. 串口通讯协议.....	9
3.1 通讯格式.....	9
3.2 通讯指令.....	10
3.2.1 控制指令.....	10
3.2.2 查询指令.....	11
3.3 芯片返回的数据.....	12
3.3.1 芯片上电返回的数据.....	12
3.3.2 曲目播放完毕返回的数据.....	13
3.3.3 芯片应答返回的数据.....	13
3.3.4 芯片错误返回的数据.....	14
3.3.5 设备插入拔出消息.....	14
3.4 串口控制指令详解.....	15
3.4.1 指定歌曲播放指令[可以直接参考 3.4.7].....	15
3.4.2 指定音量播放指令.....	15
3.4.3 单曲循环播放指令.....	16
3.4.4 指定播放设备.....	16
3.4.5 进入睡眠.....	16
3.4.6 指定文件夹文件名播放.....	17
3.4.7 插播 ADVERT 文件夹下的广告.....	18
3.4.9 播放停止指令.....	18
3.4.10 指定文件夹开始循环播放.....	19
3.4.11 随机播放设备文件.....	19
3.4.12 对当前的曲目设置为循环播放.....	19
3.4.13 开启和关闭 DAC.....	19
3.4.14 组合播放功能指令.....	20

3.4.15 带音量参数的指令播放.....	21
3.4.16 多文件夹插播功能.....	21
3.5 串口查询指令详解.....	22
3.5.1 查询当前在线的设备.....	22
3.5.2 播放状态查询指令.....	22
3.5.3 指定文件夹曲目总数查询.....	23
3.5.4 当前设备的总文件夹数目查询.....	23
4. 参考电路.....	24
4.1 串行接口.....	24
4.2 按键接口.....	25
4.2.1 通过 2 个电阻来配置按键功能.....	25
4.2.2 通过 CFG 文件来配置.....	26
4.2.1 按键的功能.....	27
4.3 外接单声道功放.....	30
4.5 芯片的指示灯说明.....	30
4.7 USB 更新语音说明[业内首创功能].....	31
4.7.1 USB 更新 SPIFLASH 的语音详细说明.....	33
4.8 用户使用空白的 FLASH 说明.....	34
5. 注意事项.....	35
5.1 GPIO 的特性.....	35
5.2 应用中的注意点.....	36
5.3 注意事项点.....	37
5.3.1 芯片上电的工作流程图.....	37
5.3.2 串口编程参考的说明.....	38
5.3.3 串口编程需要适当延时的注意点.....	38
5.3.4 校验的重要说明.....	38
5.3.5 校验的计算说明.....	39
5.3.6 MCU 的晶振选择.....	39
5.3.7 指定播放的说明.....	40
5.3.8 串口调试说明.....	41
5.3.9 校验代码的移植.....	42
5.3.10 芯片或者芯片的供电说明.....	44
6. 免责声明.....	45
7. 订货信息.....	46
8. 参考例程.....	47
9. PC 端串口调试指令举例.....	49
9.1 控制指令.....	49

9.2 查询参数指令 50

1. 概述

1.1 简介

KT603C 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、TF 卡或者 U 盘作为存储介质，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

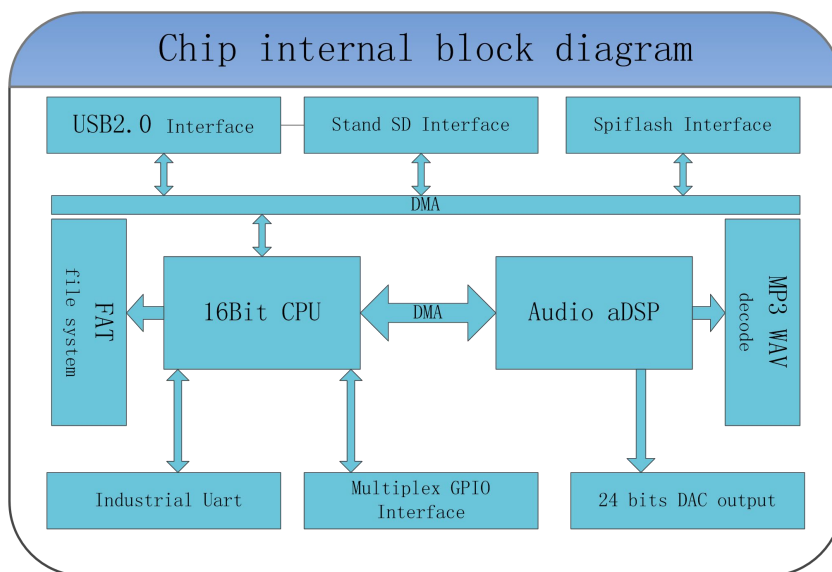
1.2 功能

- 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、自带 3W 的功放，直接外接喇叭即可完成收音
- 9、支持 6 段语音的触发播放，IO 检测的方式，所以适合碳膜按键等等场合
- 10、可以同时支持 U 盘、TF 卡以及 SPIFLASH 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；

2. 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

1、另外 KT603C 不仅可以完成音频解码功能，其内部的 MCU 功能，不会逊色于 STC 系列的芯片，所以充当控制的角色是毫无压力的。

2、默认 KT603C 烧录我们的标准程序。如果有其它的需求,请提前和我们沟通, 我们提供技术服务

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 IS013813-3 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	3.3V-5V[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
封装	SSOP24
工作温度	[-40度] -- [80度]
湿度	5% ~ 95%

2.2 管脚说明

引脚序号	引脚名称	功能描述	备注
1	DACL	左声道	
2	DACR	右声道	
3	3V3	3.3V 稳压输出	
4	VIN	电源输入	
5	GND	电源地	
6	TX	通用输入输出口	串口的发送脚
7	RX	通用输入输出口	串口的接收脚
8	X1	晶振输入	可以做 GPIO
9	X2	晶振输出	可以做 GPIO
10	INT/GPIOA0	通用输入输出口	外部中断[低触发]
11	GPIOA1	通用输入输出口	SPI 的输入
12	GPIOA2	通用输入输出口	SPI 的时钟
13	GPIOA3	通用输入输出口	SPI 的输出
14	GPIOA4	通用输入输出口	
15	GPIOA5	通用输入输出口	
16	GPIOA6	通用输入输出口	
17	GPIOB4	通用输入输出口	SDCLK
18	GPIOB3	通用输入输出口	SDCMD
19	GPIOB2	通用输入输出口	SDDAT
20	GPIOB1	通用输入输出口	USB-
21	GPIOB0	通用输入输出口	USB+
22	RST	复位脚	复位脚
23	VCOM	DAC 的参考电压	DAC 的参考电压
24	DACVSS	DAC 的输出地	DAC 的输出地

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 :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
VER	版本	版本信息
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	详见3.4.4
0x0A	进入睡眠 -- 低功耗	详见3.4.5
0x0B	唤醒睡眠	
0x0C	芯片复位	
0x0D	播放	
0x0E	暂停	
0x0F	指定文件夹文件名播放	详见3.4.6
0x13	插播广告	详见3.4.7
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
0x22	指定音量同时指定文件播放	详见3.4.15
0x25	多文件夹插播	详见3.4.16

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
0x47	查询 UDISK 文件总数	设备的总文件数
0x48	查询 TF 文件总数	设备的总文件数
0x49	查询 FLASH 的总文件数	5个文件夹的总文件数
0x4B	查询 UDISK 的当前曲目	物理顺序
0x4C	查询 TF 的当前曲目	物理顺序
0x4D	查询 FLASH 的当前曲目	返回文件夹号和曲目标针
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

(1)、芯片上电，需要一定的时间初始化，这个时间是需要根据 TF 卡、U 盘、SPIFLASH 设备的文件多少决定的，一般在小于 500ms 这个时间。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，请检查硬件的连接

(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

U 盘 -- 在线	7E FF 06 3F 00 00 01 xx xx EF	各设备之间是或的关系
TF -- 在线	7E FF 06 3F 00 00 02 xx xx EF	
PC -- 在线	7E FF 06 3F 00 00 04 xx xx EF	
FLASH -- 在线	7E FF 06 3F 00 00 08 xx xx EF	
U 盘、TF -- 在线	7E FF 06 3F 00 00 03 xx xx EF	

(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。同时还会影响芯片的正常初始化。

3.3.2 曲目播放完毕返回的数据

U 盘播放完第1曲	7E FF 06 3C 00 00 01 xx xx EF	U 盘播放第1曲完毕
U 盘播放完第2曲	7E FF 06 3C 00 00 02 xx xx EF	U 盘播放第2曲完毕
TF 卡播放完第1曲	7E FF 06 3D 00 00 01 xx xx EF	TF 卡播放第1曲完毕
TF 卡播放完第2曲	7E FF 06 3D 00 00 02 xx xx EF	TF 卡播放第2曲完毕
FLASH 播放完第1曲	7E FF 06 3E 00 01 01 xx xx EF	FOLDER1的第1曲播放完
FLASH 播放完第2曲	7E FF 06 3E 00 02 02 xx xx EF	FOLDER2的第2曲播放完

- 1、针对很多的触发型的播放需求，我们芯片更正为播放一曲之后自动进入停止状态。如果用户需要此类应用。只需要指定曲目播放即可。这样，曲目播放完毕会自动停止，等待指令
- 2、另外我们专门开辟一个 IO 作为解码和暂停的状态指示。请参见第 5 脚
 - (1)、播放状态输出低电平[很多功放有静音脚，可以通过此 IO 直接进行控制]
 - (2)、播放暂停状态，输出高电平。芯片睡眠状态。也是低电平
- 3、芯片通电之后，初始化正常，芯片会自动进入设备播放状态。并且停止解码，等待用户发送播放的相关指令
- 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，因为一旦指定曲目之后，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，会导致芯片接收不到。

3.3.3 芯片应答返回的数据

芯片返回 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，我们增加了应答处理，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，芯片已经成功收到，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 芯片错误返回的数据

返回忙	7E FF 06 40 00 00 01 xx xx EF	芯片在文件系统初始化时
当前处于睡眠模式	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
未找到指定的文件	7E FF 06 40 00 00 06 xx xx EF	指定为文件没有被找到
数据不符合规则	7E FF 06 40 00 00 08 xx xx EF	如最小为1的地方，发送为0

- (1)、为了加强数据通信之间的稳定性，我们增加了数据错误处理机制。芯片收到不符合格式的数据，均会有信息反馈出来
- (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)、芯片返回忙，基本上是芯片上电初始化的时候才会返回，因为芯片需要初始化文件系统
- (4)、芯片上电之后，进入的是设备状态，设备是 SPIFLASH。如果 SPIFLASH 不在线的话，会自动进入睡眠状态。
- (5)、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。
- (6)、文件指定部分出错，请参考下面的详解

3.3.5 设备插入拔出消息

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)、为了加强芯片的灵活性，我们特别增加了，设备插入、拔出的指令反馈。方便用户知道芯片的工作状态。
- (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。

3.4 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[可以直接参考 3.4.7]

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 则是默认为 FOLDER1 区域

(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 FF E6 EF

7E --- 起始命令

FF --- 版本信息

06 --- 数据长度(不包含校验)

03 --- 代表产品编号

00 --- 是否需要应答[0x01:需要应答，0x00:不需要返回应答]

00 --- 曲目的高字节[DH]

01 --- 曲目的低字节[DL],这里代表的是第一首歌播放

FE --- 校验的高字节

F7 --- 校验的低字节

EF --- 结束命令

(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

(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

循环播放指定曲目	7E FF 06 08 00 00 01 xx xx EF	循环播放第一曲
	7E FF 06 08 00 00 02 xx xx EF	循环播放第二曲
	7E FF 06 08 00 01 01 xx xx EF	循环播放 FOLDER1 的第1曲

(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 盘时，按照的是文件**存储的物理顺序**指定，这点请用户注意。但是在操作 FLASH 时，是按照文件夹分区指定的，请参考上面的测试指令。

(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

(1)、我们的芯片默认是支持 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。

(4)、看下表，选择合适的指令发送

(3)、指定设备之后。芯片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芯片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-PC	7E FF 06 09 00 00 03 xx xx EF	指[读卡、声卡]模式
指定播放设备-FLASH	7E FF 06 09 00 00 04 xx xx EF	
指定播放设备-SLEEP	7E FF 06 09 00 00 05 xx xx EF	

3.4.5 进入睡眠

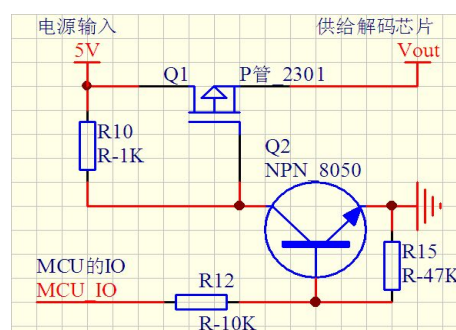
进入睡眠模式	7E FF 06 0A 00 00 00 FE F1 EF	xx xx：代表校验
--------	-------------------------------	------------

指定芯片进入睡眠之后，有两种方式对芯片或者芯片进行唤醒

(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

芯片或者芯片进入睡眠之后，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 mos 管和一个三极管来控制芯片或者芯片的供电，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

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

(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)、为了芯片的标准性，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

3.4.7 插播 ADVERT 文件夹下的广告

插播广告	7E FF 06 13 00 00 01 FE E7 EF	"ADVERT"文件夹，曲目为"0001"
	7E FF 06 13 00 00 02 FE E6 EF	"ADVERT"文件夹，曲目为"0002"
	7E FF 06 13 00 00 FF FD E9 EF	"ADVERT"文件夹，曲目为"0255"
	7E FF 06 13 00 07 CF FE 12 EF	"ADVERT"文件夹，曲目为"1999"
	7E FF 06 13 00 0B B8 FE 25 EF	"ADVERT"文件夹，曲目为"3000"

(1)、我们支持选曲播放过程中插播其它的曲目进行播放，这样就可以满足需要在背景音乐播放的过程中插播一下广告的需求。

(2)、发送 0x13 指令之后，系统会存储当前正在播放的曲目的 IDV3 信息，然后播放指定的插播曲目。等到插播的曲目播放完毕之后，系统会回到保存的那个播放断点处继续播放的。直至播放完毕。

(3)、设置的格式是，在设备中建立一个“ADVERT”名字的文件夹，在里面存储需要插播的曲目即可，曲目设置为“0xxx+曲目名.MP3/WAV”

(4)、另外系统如果当前处于暂停或者停止状态，发送插播指令，均得不到响应，并且会有错误信息返回。如果在插播的过程中，可以继续插播其它曲目，但是播放完毕之后，还是回到第一次存储的 IDV3 信息处。

(5)、指定插播广告设置如下：



3.4.9 播放停止指令

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)、在芯片的播放过程中，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

3.4.10 指定文件夹开始循环播放

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

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必须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

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。

3.4.13 开启和关闭 DAC

设置 DAC	7E FF 06 1A 00 00 00 FE E1 EF	开 DAC
	7E FF 06 1A 00 00 01 FE E0 EF	关 DAC[高阻]

- (1)、在一些用户需要叠加自己音源的场合，可以先暂停当前播放的语音，再将我们芯片的 DAC 输出设置为高阻，这样用户就可以共用一个功放来播放自己的音源了，但是 DAC 的开启和关闭，会有一声 po 音，请用户朋友们注意。
- (2)、芯片任何时候都可以关闭 DAC。如果当前正在播放语音，关闭了 DAC，芯片还会继续播放，只是没有声音了而已。芯片上电之后是默认开启 DAC 的，只有被设置为关闭之后，才会被关闭。如果再需要打开，就需要通过指令打开 DAC 了

3.4.14 组合播放功能指令

组合播放	7E FF 09 21 00 05 01 02 03 04 FE C8 EF	播放5、1、2、3、4
组合播放	7E FF 0C 21 00 05 01 02 03 04 06 07 08 FE B0 EF	播放5、1、2、3、4、6、7、8

(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，如果用户拿我们的芯片模仿简单的 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，同时支持指定播放 12 个语音，按照串口发送的顺序依次播放。

(2)、如果发送 7E FF 0C 21 00 05 01 02 03 04 06 07 08 FE B0 EF 这一帧数据，我们分析一下

CMD= 0x21 --- 查阅指令表

LEN = 0x0c = 12 个字节 --- FF 0C 21 00 05 01 02 03 04 06 07 08[去掉红色标记的字节，就是曲目的顺序] 芯片会依次播放第 5 曲、第 1 曲、2、3、4、6、7，8 这 8 段语音。播放完毕就停止

(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”文件夹里面，文件必须重命名为“001xx.mp3 或者其他”，如下截图



文件夹为“01”，文件名的序号对应我们的指令

如果发送 7E FF 0C 21 00 05 01 02 03 04 06 07 08 FE B0 EF 这一帧数据，我们分析一下

芯片会依次播放 005.mp3/001.mp3/002.mp3/003.mp3/004.mp3/006/007.mp3/008.mp3。播放完毕就停止

(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。

3.4.15 带音量参数的指令播放

带音量播放	7E FF 06 22 00 1E 01 xx xx EF	30级音量播放第1曲
	7E FF 06 22 00 0F 02 xx xx EF	15级音量播放第2曲

- (1)、针对一些用户希望，对不同的语音设置不同的音量进行播放，如果按照以前的老方法，就是先设置完音量，再指定曲目播放，这样操作繁琐，不方便。特此我们增加此条指令 0x22
- (2)、具体的操作可以参考上面给出的两条测试指令。
- (3)、对于 U 盘或者 TF 卡，我们按照是物理顺序指定播放的。对于 FLASH，则是默认是 FOLDER1 文件夹下面。

3.4.16 多文件夹插播功能

插播广告-多文件夹	7E FF 06 25 00 01 01 FE D4 EF	“ADVERT1”的文件夹，曲目为“001”
	7E FF 06 25 00 01 02 FE D3 EF	“ADVERT1”的文件夹，曲目为“002”
	7E FF 06 25 00 02 01 FE D3 EF	“ADVERT2”的文件夹，曲目为“001”



- (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，也不会改变当前的状态，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或者其他的。
- (5)、插播的原理请参见 3.4.6.

3.5 串口查询指令详解

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 代表的意义详解如下：

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 代表的意义详解如下：

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

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

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
返回的数据	7E FF 06 4F 00 00 03 FE AC EF	显示有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

4. 参考电路

针对芯片的应用，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

芯片我们采用的是 IO 按键的方式，取代了 AD 键盘的接法，这样做的好处是充分利用了 MCU 越来越多的 GPIO。设计繁琐但不简单，我们芯片默认配置 6 个按键的功能分配，可以在任何恶劣的场合随意的控制，甚至也可以作为与 MCU 的通信接口。我们的按键分配多种不同类型的功能。

并且按键的配置有两种方式：

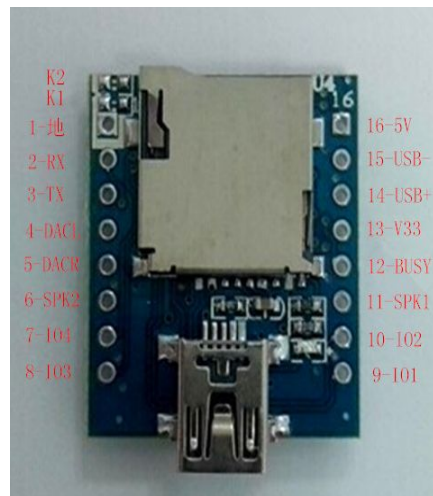
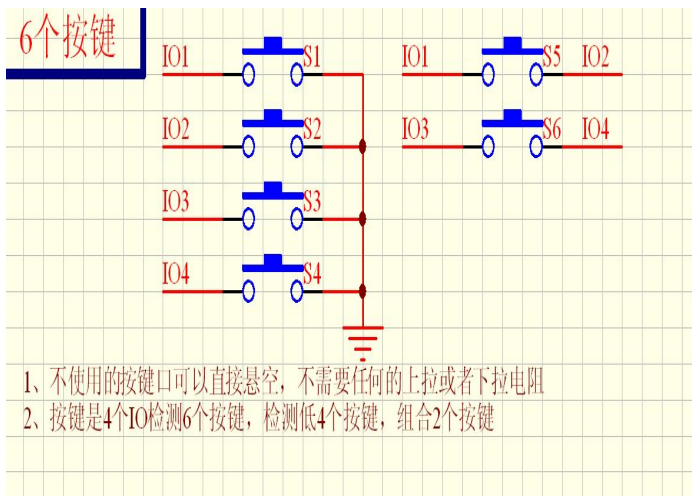
- 根据 2 个电阻的对地来选择，参考“4.2.1”章节
- 根据设备中 read.cfg 文件进行配置，参考“4.2.2”章节

4.2.1 通过 2 个电阻来配置按键功能

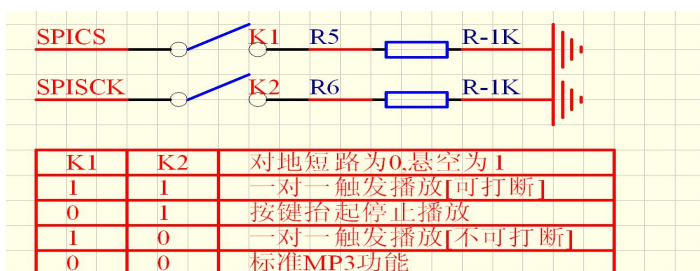
以下支持 4 种按键模式。

- 一对一触发播放，可打断[K1 和 K2 均是悬空状态]
- 按键抬起停止播放[K1 外接 1K 短接到地，K2 悬空]
- 一对一触发播放，不可打断[K1 悬空，K2 外接 1K 短接到地]
- 标准的播放功能，如上下曲、播放暂停等等[K1、K2 都短接 1K 电阻到地]

按键的接口图



按键的功能选择图



备注:标准MP3功能包含上下曲和音量加减以及暂停和停止

4.2.2 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,得到用户配置的参数。**请严格的按照我们给出的配置参数的方法**
- (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

4.2.1 按键的功能

(1)、一对一触发播放

一对一触发	短按	长按	按着不松	按键抬起
S1	第1段			
S2	第2段			
S3	第3段			
S4	第4段			
S5	第5段			
S6	第6段			

备注:此为一对一触发播放,可打断[对地短路触发80ms有效]。这里文件是按照物理地址顺序。如果外接按键,则是按下就会响应,不需要按键抬起[同时使用的是]

(2)、按键抬起停止播放功能

按键抬起停止	短按	长按	按着不松	按键抬起
S1		第1段循环		停止
S2		第2段循环		停止
S3		第3段循环		停止
S4		第4段循环		停止
S5		第5段循环		停止
S6		第6段循环		停止

备注:此为抬起停止,长按循环播放[对地短路触发100ms有效]。这里文件是按照物理地址顺序。

(3)、一对一触发不可打断

触发不中断	短按	长按	按着不松	按键抬起
S1	第1段不可打断			
S2	第2段不可打断			
S3	第3段不可打断			
S4	第4段不可打断			
S5	第5段不可打断			
S6	第6段不可打断			

备注:此为一对一触发播放,可打断[对地短路触发80ms有效]。这里文件是按照物理地址顺序。如果外接按键,则是按下就会响应,不需要按键抬起

(4)、标准播放功能

播放功能按键	短按	长按	按着不松	按键抬起
S1	上一曲			
S2	下一曲			
S3	播放暂停			
S4	停止			
S5	音量+			
S6	音量-			

备注:此为标准的MP3功能。如果同时短接 I01到地之后,芯片会上电就开始自动播放。同时其他的按键依然有效

(5)、指定文件夹文件名播放

播放功能按键	短按	长按	按着不松	按键抬起
S1	01文件夹001文件			
S2	01文件夹002文件			
S3	01文件夹003文件			
S4	01文件夹004文件			
S5	01文件夹005文件			
S6	01文件夹006文件			

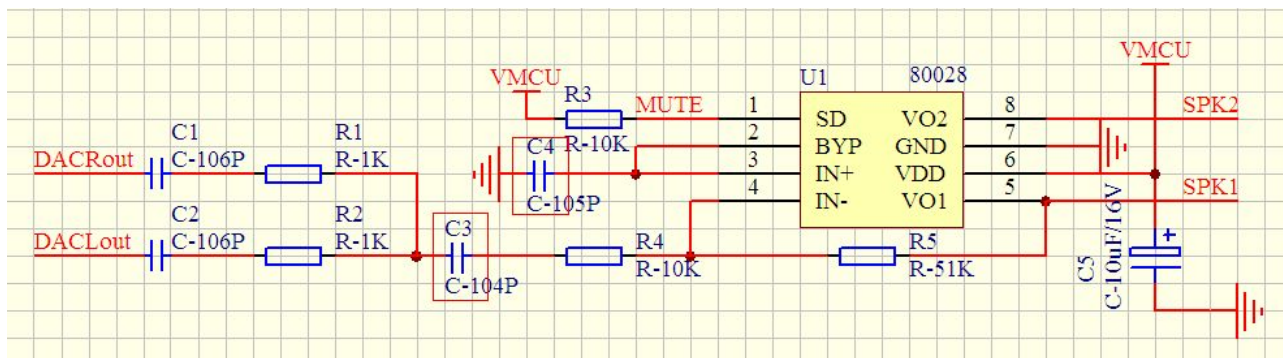
备注:支持 TF 卡或者 U 盘以及 SPIFLASH 三种设备的任意一种。这里的语音是文件夹和文件名指定播放

(5)、指定文件夹文件名播放

播放功能按键	短按	长按	按着不松	按键抬起
S1	01文件夹001文件			02文件夹001文件
S2	01文件夹002文件			02文件夹002文件
S3	01文件夹003文件			02文件夹003文件
S4	01文件夹004文件			02文件夹004文件
S5	01文件夹005文件			02文件夹005文件
S6	01文件夹006文件			02文件夹006文件

备注:支持 TF 卡或者 U 盘以及 SPIFLASH 三种设备的任意一种。这里的语音是文件夹和文件名指定播放

4.3 外接单声道功放



这里功放我们采用的是 8002，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，这样的过程中，就会少很多杂音

4.5 芯片的指示灯说明

1、我们的芯片集成了一个翠绿色的指示灯，用来指示工作状态

芯片状态	下载模式	播放语音	暂停	睡眠
指示灯状态	快闪	慢闪500ms	常亮	灭

备注：正常工作状态指示灯

2、如果芯片上电未检测到 TF 卡或者 U 盘或者 FLASH 在线，指示灯会自动熄灭。这样用户就可以判断芯片外挂的设备是否正常。

3、如果用户不需要此指示灯，可以直接焊下来就可以，不会对芯片的正常工作产生任何影响。

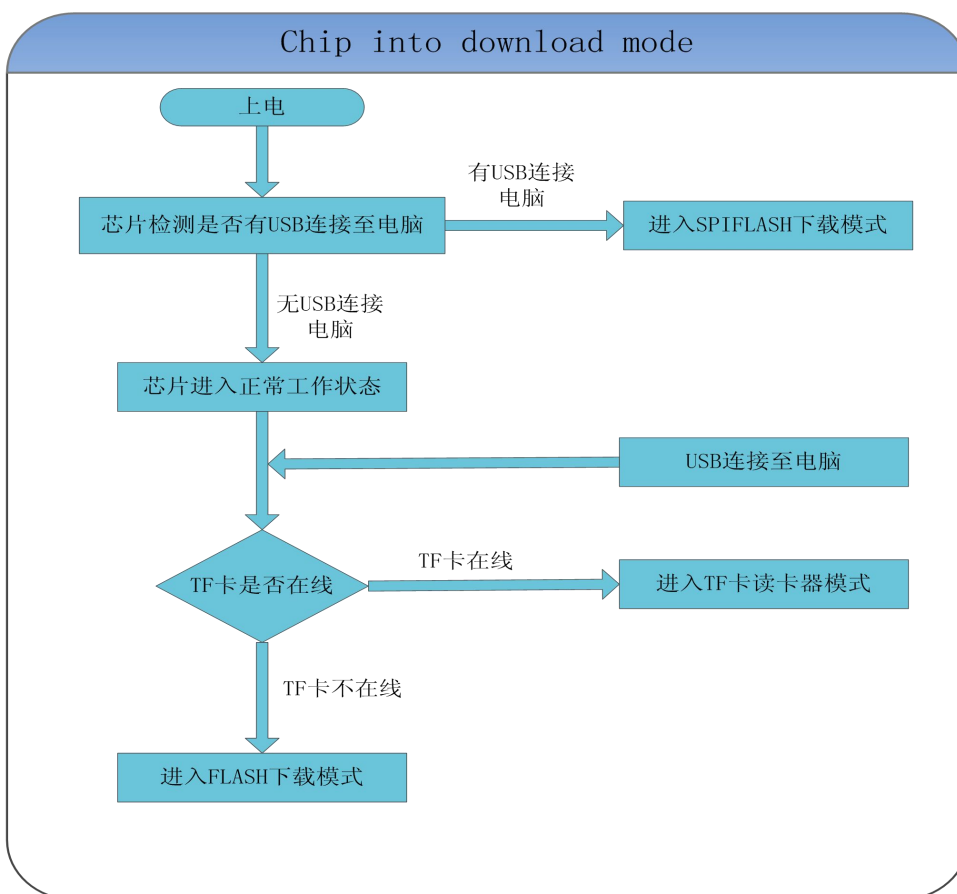
4.7 USB 更新语音说明[业内首创功能]

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。

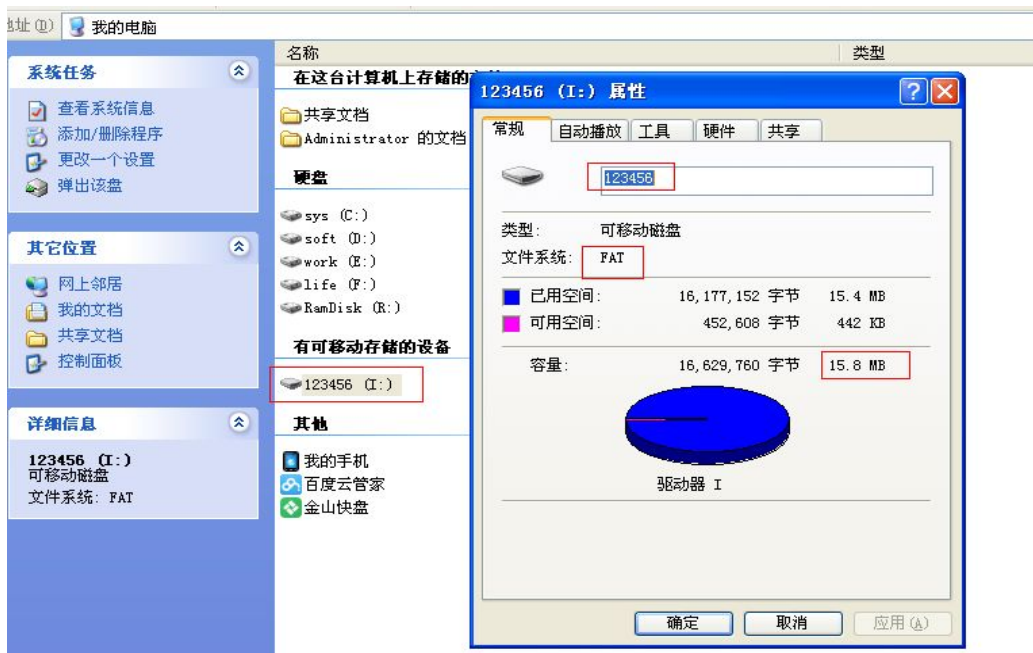


4.7.1 USB 更新 SPIFLASH 的语音详细说明

我们的芯片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

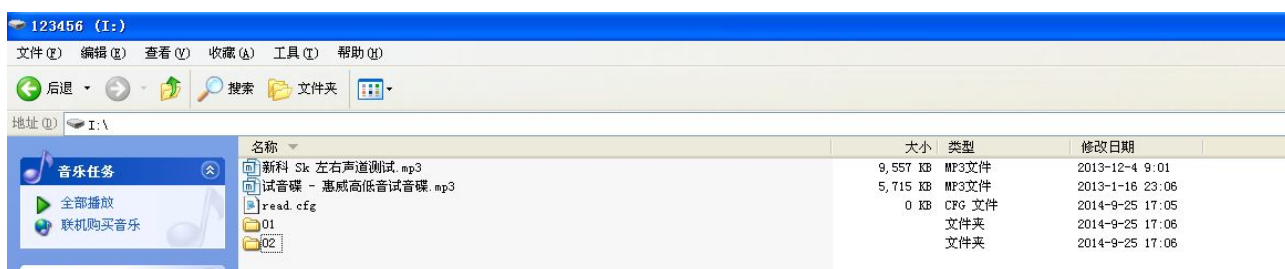
- 可以适配市面上几乎大部分的 FLASH 芯片，兼容性极好
- 无需安装任何软件，直接更新，也不需要专用下载器
- 对音质无任何压缩和损坏，保证更高的音质体验

1、插上我们芯片的 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，我们芯片都是支持的，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已的需求来确定 FLASH 的容量和型号。

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、我们的方案目前最大支持 64M 字节的 FLASH。换算为 FLASH 一般的型号，如：W25Q512。但是市面上目前量产的 FLASH 为 32M 字节，也就是 W25Q256。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]
W25Q256	32Mbyte	SOP16[宽体]
W25Q512	64Mbyte	SOP16[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。将文件系统的列表写入 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. 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
V_{IL}	Low-Level Input Voltage	-0.3	-	$0.3 * V_{DD}$	V	$V_{DD}=3.3V$
V_{IH}	High-Level Input Voltage	$0.7V_{DD}$	-	$V_{DD}+0.3$	V	$V_{DD}=3.3V$
IO 输出特性						
符号	参数	最小	典型	最大	单位	测试条件
V_{OL}	Low-Level Output Voltage	-	-	0.33	V	$V_{DD}=3.3V$
V_{OH}	High-Level Output Voltage	2.7	-	-	V	$V_{DD}=3.3V$

5.2 应用中的注意点

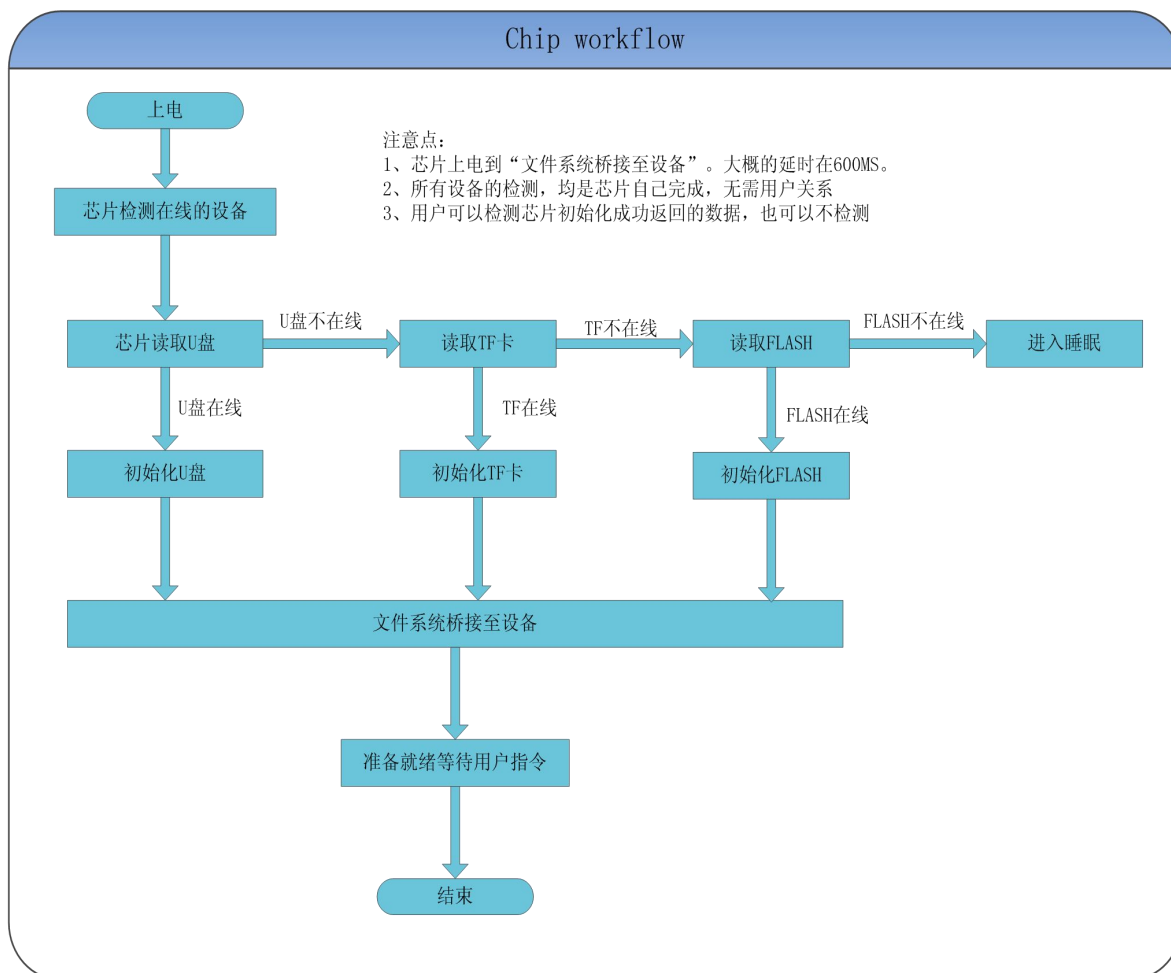
- 1、芯片对外的接口均是 3.3V 的 TTL 电平，所以在硬件电路的设计中，请注意电平的转换问题。另外在强干扰的环境中，请注意电磁兼容的一些保护措施，GPIO 采用光耦隔离，增加 TVS 等等
- 2、串口通信，在一般的使用环境下，注意好电平转换即可。如果强干扰环境，或者长距离的 RS485 应用，那么请注意信号的隔离，严格按照工业的标准设计通信电路。可以联系我们，我们提供设计参考
- 3、我们支持音频文件的采样率最低为 8KHZ。也就是说低于 8KHZ 的音频文件是不支持的，不能正常解码播放。用户可以使用音频处理软件，提高音频文件的采样率来解决这个问题。
- 4、芯片在睡眠状态的电流在 10MA 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，请用户控制芯片或者芯片的供电。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、该芯片支持 MP3、WAV 二种主流的音频格式。
- 8、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睬这些数据
- 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。这样可以使串口产生 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

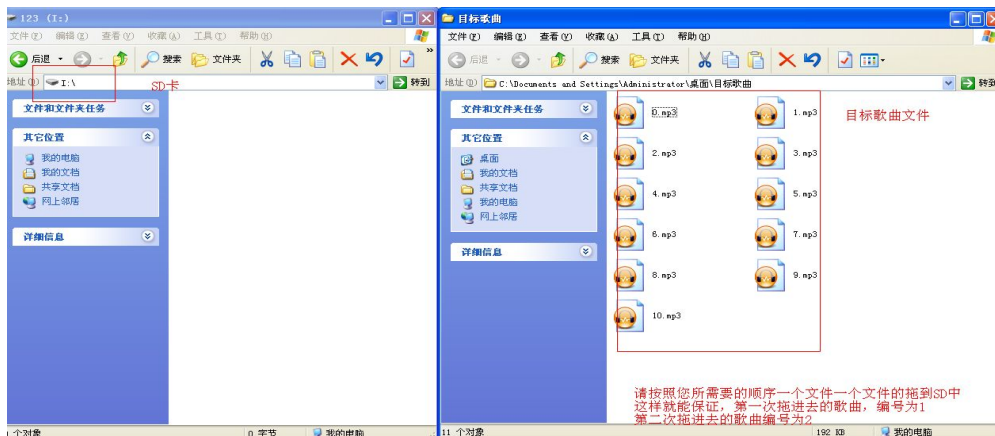
(3)、标准的 51，如：STC89C52 或者 AT89C52 等等都是采用定时器产生波特率的，经过简单的计算就可以算出，12M 晶体做 9600 波特率的误差是 0.16%，正常运行是没有任何问题，但还是需要用户进行全面测试

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]
- 指定曲目播放[按照文件名，必须在 MP3 文件夹下]

1、按照物理顺序指定曲目播放：

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



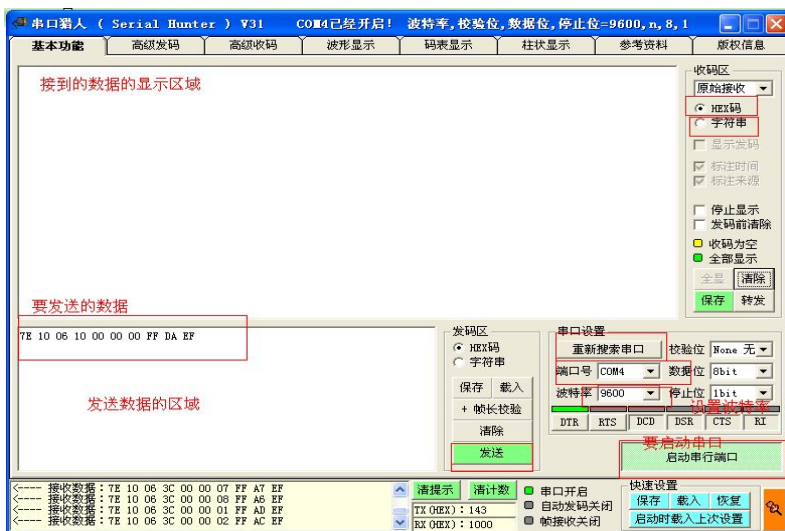
2、指定文件夹和文件名播放

详情请参考芯片数据手册的“3.4.4 指定文件夹播放”章节

3、指定 MP3 文件夹里面的文件名播放

详情请参考芯片数据手册的“3.4.5 指定 MP3 文件夹播放”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
}

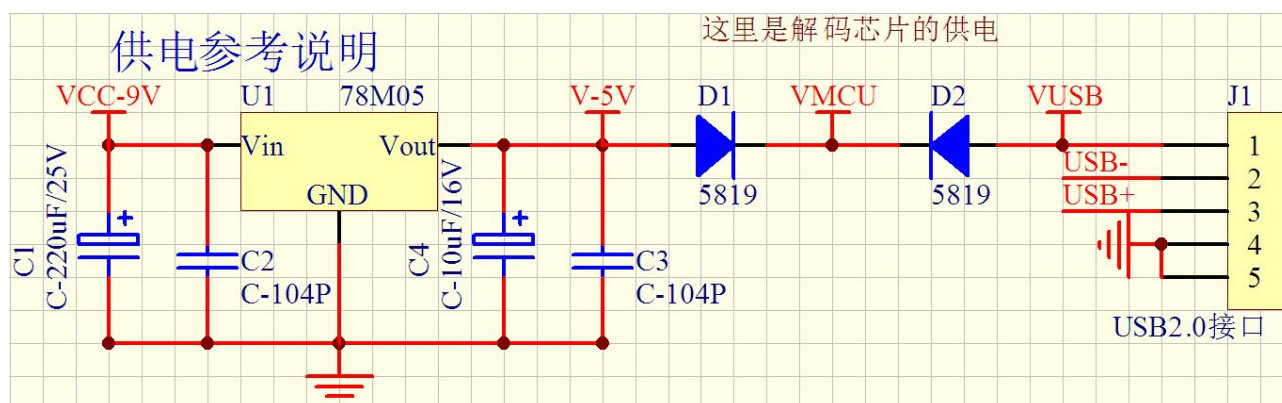
```

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

千乐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

千乐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。千乐系列芯片的 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。

千乐系列芯片的 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。我们将努力完善千乐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

千乐工作室能保留任何时候在不事先声明的情况下对千乐系列产品相关文档的修改权力

■ ESD 静电放点保护

千乐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，安装千乐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

REF.	mm			Inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			2			0.079
A1	0.05			0.002		
A2	1.65	1.75	1.85	0.065	0.069	0.073
b	0.22		0.38	0.009		0.015
c	0.09		0.25	0.004		0.010
D	7.9	8.2	8.5	0.311	0.323	0.335
E	7.4	7.8	8.2	0.291	0.307	0.323
E1	5.00	5.3	5.6	0.197	0.209	0.220
e		0.65			0.0256	
K	0°		8°	0°		8°
L	0.55	0.75	0.95	0.022	0.030	0.037

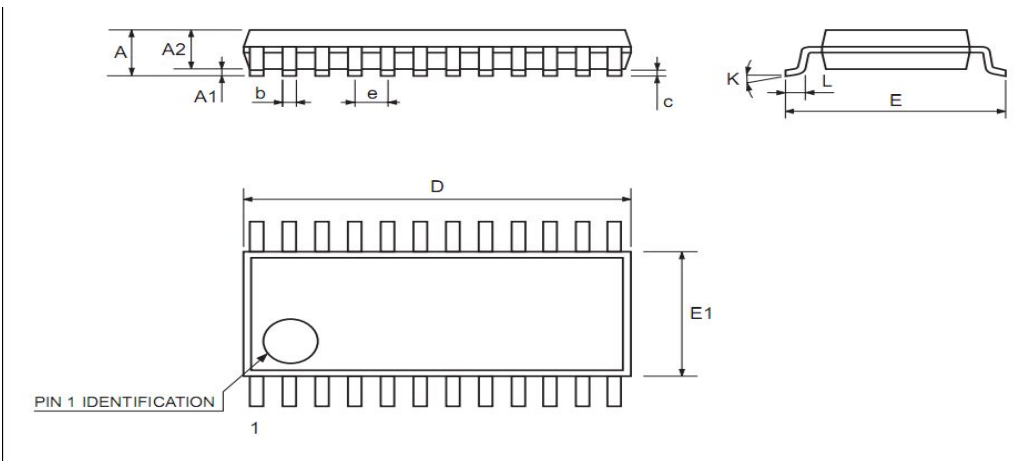
 **PACKAGE AND PACKING INFORMATION**

24-LEAD SHRINK SMALL OUTLINE PACKAGE

Weight: not available



SSOP24



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	
[指定曲目]	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
	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
[循环播放曲目]	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
	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
[指定播放设备]	7E FF 06 09 00 00 01 FE F1 EF	7E FF 06 09 00 00 01 EF	指定播放 UDISK
	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 TF/U 盘总文件夹数	7E FF 06 4F 00 00 00 FE AC EF	7E FF 06 4F 00 00 00 EF	只支持 TF 卡和 U 盘