WT8601M02 V1.1 使用说明书

本资料适用硬件版本 V1.1

支持 config.txt 信息的固件

目　　录

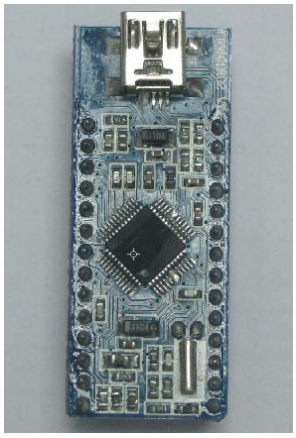
1、产品特性

- 支持 8Kbps～320Kbps 位速率 MP3 格式音频播放；
- 支持 VBR 可变位速率及 CBR 恒定位速率；
- 立体声高品质音频输出；
- 支持上电自动复位，低电压自动复位，外部管脚复位；
- 支持 MP3 控制模式、按键一对一控制模式、随机播放模式、DSA 控制模式；
- 具有上电自动播放、单曲循环、全部循环等功能；
- 采用 NAND-Flash 作为存储中心，存储空间大，语音时间长；
- 最大可支持 2GB byte NAND-Flash；
- 通过 TXT 文件修改控制模式及上电默认音量等功能；
- 全速 USB2.0 数据传送；
- 低功耗运行，续航时间更长；
- 工作电压 DC5V。

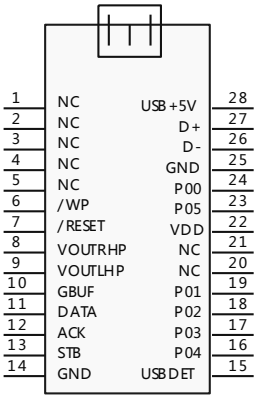
2、应用范围

WT8601M02 V1.1 能应用在高级家用电器，如智能语音导航电冰箱、语音导航空调、语音导航电磁炉等，以及高级玩具、汽车电子系统、长时间放音系统等高音质要求场所。

3、接线图示



WT8601M02 V1.1 正面实物图片



WT8601M02 V1.1 正面框图

WT8601M02 V1.1 框图接线说明

管脚序号	功能	说明	管脚序号	功能	说明
1	NC	空	15	USBDET	USBDET
2	NC	空	16	P04	按键 4
3	NC	空	17	P03	按键 3
4	NC	空	18	P02	按键 2
5	NC	空	19	P01	按键 1
6	/WP	数据写保护	20	NC	空
7	/RESET	复位脚	21	NC	空
8	VOUTRHP	音频 R 声道输出	22	VDD	电源正极
9	VOUTLHP	音频 L 声道输出	23	P05	按键 5
10	GBUF	音频地	24	P00	BUSY 输出
11	DATA	DSA 传送数据及开始标志	25	GND	USB 电源地
12	ACK	DSA 传送应答标志	26	D-	USB_D-
13	STB	DSA 数据保存标志	27	D+	USB_D+
14	GND	电源地	28	USB+5V	USB 电源

注： 1、GBUF 不能直接接电源地。 2、USBDET 为 USB 电压电压检测，模块内部已有处理电路，可以不接。

4、电气参数

环境温度：25℃ 输入电压 DC5V P00 上拉电阻：10KΩ

参数	标记	环境条件	最小值	典型值	最大值	单位
工作电压	V_{DD}	F_{sys}=12MHz	3.5	5.0	6.0	V
工作电流 1	I_{OP1}	没有负载	24.2	26.5	28.0	mA
工作电流 2	I_{OP2}	R_{ROUT}=8Ω R_{LOUT}=8Ω	26.0	27.5	46.6	mA
停止电流	I_{DD2}	没有负载	---	12.0	---	mA
上拉阻	RH	RESETn	---	75	---	KΩ

5、功能设置

当前 WT8601M02 V1.1 可通过在 config.txt 设置以下功能。

在 PC 端新建一个文本文档，重命名为“config.txt”，在文本中输入以下内容：

vl=0～31；

fa=0/1；

pp=0/1；

sn=00001.mp3；

pm=0/1/2；

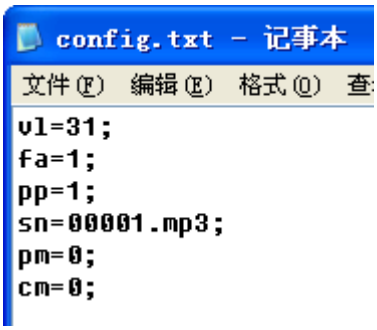
cm=0/1/2；

其中指令的作用如下表所示

序号	命令	说明	数据	功能
1	vl	默认音量设置	0～31	音量设置，可输入音量等级 0～31
2	fa	音乐渐进渐出效果设置	0	取消音乐渐进渐出效果
			1	增加音乐渐进渐出效果
3	pp	上电自动播放设置	0	取消开机自动播放
			1	设置开机自动播放
4	sn	上电自动播放曲目设置	00001.mp3	设置上电播放曲目，如歌曲名中包含有其他文字，则只需要输入前五位数字
5	pm	循环模式设置	0	设置播放完单曲后停止
			1	设置单曲一直循环播放
			2	设置所有曲目循环播放
6	cm	控制模式设置	0	MP3 控制模式
			1	按键一对一控制模式
			2	随机播放模式

注：1、DSA 控制模式在任何时候都存在。

2、如果设置成“随机播放模式”，则要同时将循环模式设置为 2（所有曲目循环）和上电自动播放设置为 1（开机自动播放）。具体设置如图所示。



设置好 config.txt 后，将 config.txt 复制到 WT8601M02 根目录即可。WT8601M02 在上电或者复位后会先执行该 config.txt 中的命令。

6、公共区/隐藏区音乐描述

公共区音乐，指的是可以通过 USB 线直接下载到 NAND-Flash 根目录的 MP3 音乐。

隐藏区音乐，指的是需要订制的，由我司拷贝到 NAND-Flash，并且客户端无法修改及格式化的 MP3 音乐。

注意事项：通过 USB 更新公共区的文件请严格遵守下面操作步骤,否则 WT8601M02 模块的数据不能够自动更新:

1. 通过 WT8601M02 的 USB 接口连接到 PC 机.
2. 在 PC 机的可移动磁盘内更新语音文件(文件名遵循第 7 节之规定).
3. 在 PC 机上移除 USB 设备(可移动磁盘),断开 USB 电源.
4. 重新插入 USB 数据线,重新打开可移动磁盘盘符，此时不要进行任何操作，模块将自动更新存储的语音索引,3 秒钟后再移除设备，断开 USB 电源即可正常使用.

7、音频文件命名规则

音乐文件存放在 WT8603M01 的根目录，以 5 位数字加后缀名的方式命名，如 00001.mp3，00002.mp3，00003.mp3 等。为了方便记忆音频文件的名字及内容，可以采取序号加原文件名的方式命名，如 00001 歌唱祖国.mp3，00002 春天的故事.mp3 等。在控制指定音乐文件进行播放时，只需要发送 5 位数字即可。所有的 MP3 文件都放在根目录下，并且文件编号按文件存放的顺序定义。WT8603M01 按索引播放时是以音频文件先后存放的顺序进行播放，并不是按照文件名排列顺序。可以在先电脑上命名好所有音频文件后，排列好顺序，然后整体复制到 WT8603M01 的根目录。有两种常用的复制方法：一是用快捷键“Ctrl + C”和“Ctrl + V”，但注意鼠标不能点击到任何选中待发送的文件，否则会以鼠标所点击的文件开始发送的。这样就会打乱了文件的顺序了。二是排列好文件文件的顺序，选中所要发送的文件，然后右键点击第一个文件(例如 00001.mp3),选择发送到 WT8603M01 的根目录。

注意:右键点击的是要发送的第一个文件，系统会从此文件开始发送。

录音文件自动保存在 VOICE 文件下（不支持查找目录功能）。

8、控制模式

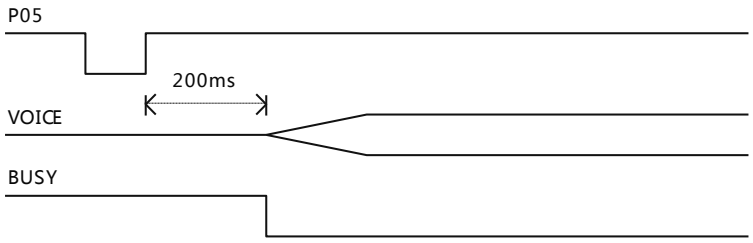
WT8601M02 V1.1 支持 MP3 控制模式、按键一对一控制模式、随机播放控制模式、MCU 控制模式等四种控制模式，控制模式通过 config.txt 更换。

8.1、MP3 控制模式

在 MP3 控制模式下，I/O P01～P05 保持 10ms 的低电平，就能触发相关的功能。各 I/O 所对应的功能如下。P05 短按 1 秒为播放/暂停功能，长按 3 秒为停止功能。P00 为输出口，语音播放过程中为低电平，语音停止时为高电平。

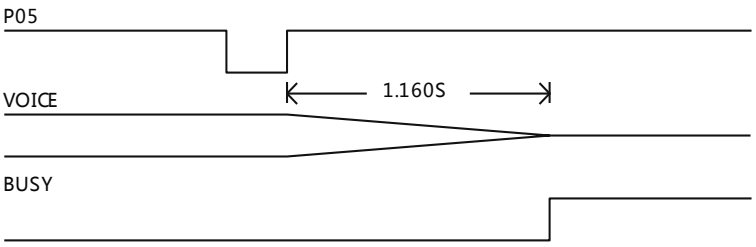
I/O	P00	P01	P02	P03	P04	P05
功能	BUSY	音量+	音量-	上一曲	下一曲	播放/暂停/停止

播放操作



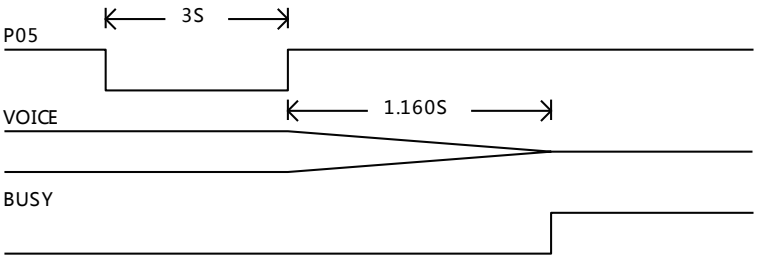
在语音停止状态，用 10ms～1S 的低电平触发 P05，就能触发播放语音。触发后 200ms 开始播放语音，同时 BUSY 转为低电平。语音以渐进放大的功能播放。

暂停操作



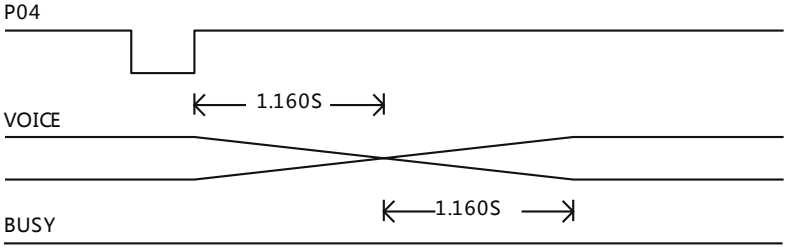
在语音播放状态，用10ms～1S 的低电平触发 P05，就能暂停当前的语音。触发后语音开始逐渐减小，1.160S 后完全停止播放，同时 BUSY 信号转为高电平。

停止操作



在语音播放状态，P05保持3S 的低电平，就能停止播放当前的语音。触发后语音开始逐渐减小，1.160S 后完全停止播放，同时 BUSY 信号转为高电平。

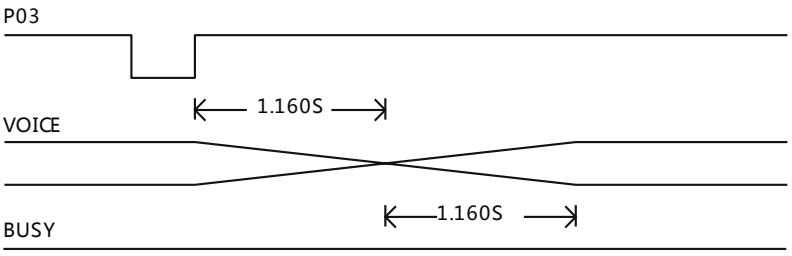
下一曲操作



在语音播放状态，用10ms～1S 的低电平触发 P04，当前语音声音逐渐减小1.160S 后停止播放，切换到下一曲开始播放，语音播

放时声音逐渐增大。在语音播放过程中切换到下一曲语音，BUSY 一直为低电平。

上一曲操作



在语音播放状态，用10ms～1S 的低电平触发 P03，当前语音声音逐渐减小1.160S 后停止播放，切换到上一曲开始播放，语音播放时声音逐渐增大。在语音播放过程中切换到上一曲语音，BUSY 一直为低电平。

8.2、按键一对一控制模式

按键一对一控制模式下，WT8601M02 V1.1 最多只能播放 5 首音乐，且一个 I/O 对应一段音乐。I/O P01～P05 保持 10ms 的低电平，就能触发相关的功能。P00 为输出口，语音播放过程中为低电平，语音停止时为高电平。

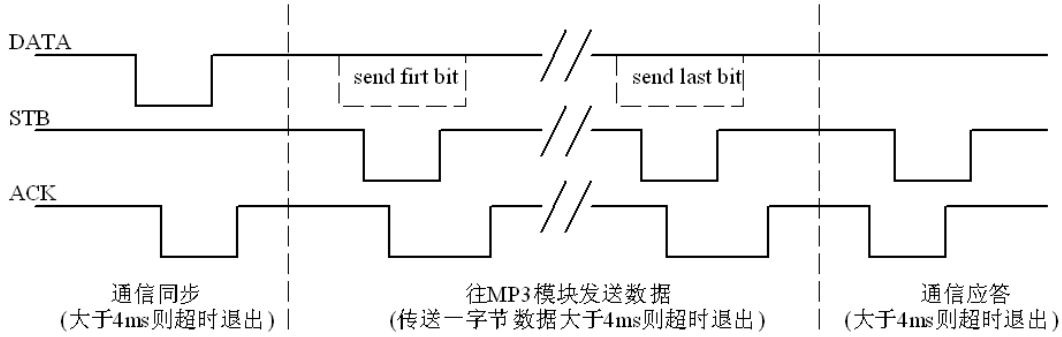
I/O	P00	P01	P02	P03	P04	P05
功能	BUSY	第一首	第二首	第三首	第四首	第五首
对应文件名	无	第一首语音	第二首语音	第三首语音	第四首语音	第五首语音

8.3、MCU 控制模式

MCU 控制模式通过 DSA_DATA、DSA_ACK、DSA_STB 三个端口来控制 WT8601M02 V1.1 工作。本协议以标准 DSA 模式做为基础修改，占用系统资源少，对时间没有严格要求。

8.3.1、控制时序

MCU 发送不带数据返回的命令的时序：



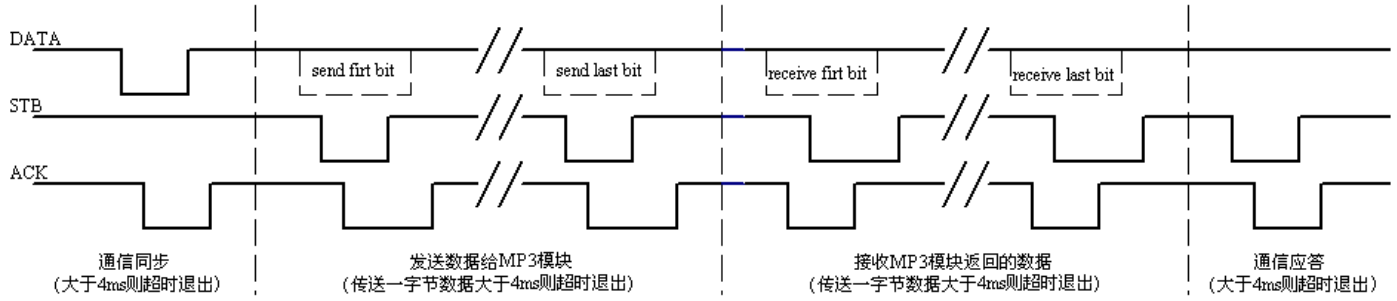
MCU 拉低 DATA 开始通信同步。当 MP3 模块检测到 DATA 为低 ,通过拉低 ACK 回复 MCU。MCU 收到 ACK 为低电平后把 DATA 拉高，当 MP3 模块把 ACK 拉高后，同步完成。

MCU 发送 send first bit 后再发送 STB，检测到 ACK 为低电平后把 STB 拉高，继而把 DATA 也拉高，只有检测到 ACK 为高电平后才能继续发送下一位数据。用同样的方式发送完所有数据（send first bit – send last bit 为整个命令长度），发送完成后 DATA 为高。MCU 拉低 ACK 开始通信应答，等待 MP3 模块拉低 STB 后，MCU 拉高 ACK，等待 MP3 模块拉高 STB，完成通信应答。

如果通信同步和通信应答超出 4ms 则超时退出。数据按字节发送，先发高位再发低位，每字节数据传送时间超出 4ms 则超时退

出。通信过程中发生超时退出则为通信失败，必须重新通信，由通信同步开始。

MCU 发送带数据返回的命令的时序：



MCU 拉低 DATA 开始通信同步。当 MP3 模块检测到 DATA 为低 通过拉低 ACK 回复 MCU。MCU 收到 ACK 为低电平后把 DATA 拉高，当 MP3 模块把 ACK 拉高后，同步完成。

MCU 发送 send first bit 后再发送 STB，检测 ACK 为低电平后把 STB 拉高，继而把 DATA 也拉高，只有检测到 ACK 为高电平后才能继续发送下一位数据。用同样的方式发送完所有数据（send first bit – send last bit 为发送数据长度）。然后转向接收 MP3 模块返回的数据。

MCU 检测到 ACK 为低，读取 DATA 的数据后把 STB 拉低回应 MP3 模块，等待 ACK 为高后把 STB 拉高，然后接收下一位数据（按字节接收，高位在前，低位在后）。接收完最后一位数据 receive last bit，开始通信应答，MCU 等待 STB 为低，拉低 ACK 回应，等待 MP3 模块拉高 STB 后，MCU 拉高 ACK 完成通信应答。

如果通信同步和通信应答超出 4ms 则超时退出。数据按字节发送，先发高位再发低位，每字节数据传送时间超出 4ms 则超时退出。通信过程中发生超时退出则为通信失败，必须重新通信，由通信同步开始。

8.3.2、指定播放公共区/隐藏区索引文件命令码

指定播放公共区索引音乐

索引文件是以存放的顺序决定的，发送此命令可指定公共区索引文件进行播放。

起始码	长度	命令	公共区	曲目高位	曲目低位	结束码
7E	05	AC	00	00	01	7E

范例中的命令为播放公共区第一段索引音乐文件。

指定播放隐藏区索引音乐

索引文件是以存放的顺序决定的，发送此命令可指定隐藏区索引文件进行播放。

起始码	长度	命令	隐藏区	曲目高位	曲目低位	结束码
7E	05	AC	01	00	01	7E

范例中的命令为播放隐藏区第一段索引音乐文件。

8.3.3、公共区命令码

指定公共区音频播放

此命令可以指定公共区某一文件名的文件进行播放，不受文件存放的顺序影响。

起始码	长度	命令	曲目高位	曲目低位	结束码
7E	04	A0	00	01	7E

范例中的命令为指定播放公共区文件名为 00001.mp3 的音频文件。

语音暂停

起始码	长度	命令	结束码
7E	02	A1	7E

第一次发送该指令，则暂停播放音乐，再次发送该数据，则从暂停处继续播放音乐。

语音停止

起始码	长度	命令	结束码
7E	02	A3	7E

发送该指令，停止播放当前正在播放的音乐。

音量控制

音量等级共有 32 级，分别为 00～31，其中 00 为静音，31 级为最大音量，可通过 config.txt 中的 vl 设置默认音量值。

起始码	长度	命令	音量等级	结束码
7E	03	A4	19	7E

范例中为发送最大音量 31 级。

上一曲

起始码	长度	命令	结束码
7E	02	A7	7E

该指令能够触发播放上一曲音乐，在播放最后一曲音乐时，发送该指令可触发播放第一曲音乐。

下一曲

起始码	长度	命令	结束码
7E	02	A6	7E

该指令能够触发播放下一曲音乐，在播放第一曲音乐时，发送该指令可触发播放最后一曲音乐。

8.3.4、读取返回信息

读取公共区语音总数

起始码	长度	命令	曲目高位	曲目低位	结束码
7E	04	C2	XX	XX	7E

红色部分为 WT8601M02 返回的数据。MCU 发送 7E 04 C2，WT8601M02 返回 XX XX 7E，其中 XX XX 为公共区语音曲目总数。

读取隐藏区语音总数

起始码	长度	命令	曲目高位	曲目低位	结束码
7E	04	C3	XX	XX	7E

红色部分为 WT8601M02 返回的数据。MCU 发送 7E 04 C3 后，WT8601M02 返回 XX XX 7E，其中 XX XX 为隐藏区语音曲目总数。

获取当前播放状态

起始码	长度	命令	状态	结束码
7E	03	C4	XX	7E

发送该命令可返回 WT8601M02 当前的播放状态，如 MCU 发送 7E 03 C4，则 WT8601M02 返回 XX 7E，当 XX 为 01，表示正常播放；XX 为 02，表示没有播放，处于停止状态；XX 为 03，表示没有播放，处于暂停状态。

读取当前设置音量

起始码	长度	命令	音量	结束码
7E	03	C5	XX	7E

MCU 发送命令 7E 03 C5，WT8601M02 返回 XX 7E，其中 XX 为当前音量。

读取公共区当前播放语音索引信息

起始码	长度	命令	公共区	曲目高位	曲目低位	结束码
7E	05	C1	00	XX	XX	7E

MCU 发送 7E 04 C1 00，WT8601M02 返回 XX XX 7E， XX XX 表示索引的位置。

读取隐藏区当前播放音乐索引信息

起始码	长度	命令	隐藏区	曲目高位	曲目低位	结束码
7E	05	C1	01	XX	XX	7E

MCU 发送 7E 04 C1 01，WT8601M02 返回 XX XX 7E， XX XX 表示索引的位置。

读取公共区当前指定文件信息

发送该命令可读取公共区当前正在播放的指定文件信息。其中“存在”表示该 U 盘中存在文件，文件名只读取前五位数字，文件容量的单位为 byte，时间长度单位为秒。

起始码	长度	命令	存在	文件名		文件容量（byte）				时间长度（秒）		结束码
				高位	低位	高位	……	……	低位	高位	低位	
7E	0B	C7	XX	XX	XX	XX	XX	XX	XX	XX	XX	7E

MCU 发送 7E 0B C7 后，WT8601M02 返回一些信息(红色部分为 WT8601M02 返回的数据)，如

起始码	长度	命令	存在	文件名		文件容量（byte）				时间长度（秒）		结束码
				高位	低位	高位	……	……	低位	高位	低位	
7E	0B	C7	01	00	01	00	2E	1F	00	01	0A	7E

则表示 WT8601M02 中存在文件，当前的文件为 00001.mp3，容量为 3022592byte，时间长度 266 秒。

9、控制程序范例

9.1、汇编程序

```
;*********************************************************************
;项目名:WT8601M02 模块  DSA 通信测试程序
;功能要求:
```

```
;       1)  通过 RS232 和 PC 进行通信(暂无)
;       2)  通过 DSA 通信协议和 WT8601M02  模块通信
;       3)  实现简易按键控制方式
;硬件配置:
;       1) MCU 型号: AT89C2051
;       2)  外接晶振频率为:11.0592MHz
;       3) I/O  定义:

    ACK       EQU P3.2      ;ACK 反馈信号引脚（通信口不要加其他上拉等，3V 供电）
    STB       EQU P3.3      ;STB 发送引脚      （通信口不要加其他上拉等，3V 供电）
    DAT       EQU P3.4      ;数据引脚          （通信口不要加其他上拉等，3V 供电）
FLAG_10MS     BIT 20H.0     ;10MS 定时标志
comm_flag     bit 20h.1     ;命令发送标志
comm_ok       bit 20h.2     ;dsa 命令成功标志
timeout_flag bit 20h.3      ;时间超出标志
  STARTN      EQU 30H       ;DSA 数据暂存空间
    NUM1      EQU 31H
    NUM2      EQU 32H
    NUM3      EQU 33H
    NUM4      EQU 34H
    NUM5      EQU 35H
    NUM6      EQU 36H
    NUM7      EQU 37H
    ENDN      EQU 38H       ;DSA 数据暂存空间结束

    VOLN      EQU 40H       ;音量级数暂存
dsa_timeout   EQU 41H       ;dsa 时间限制计数器
P0_IN_REG     EQU 60H       ;P0 口按键比较值
   DUMMY      EQU 61H       ;扫描暂存值
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ORG       0000H
        LJMP      START
        ORG       000BH
        LJMP      TIME0
```

```
;          org       0023h
;      LJMP      SERIAL
           ORG       0030H

;;;;;;;;;;;;; 主 程 序 ;;;;;;;;;;;;;;;;
START:
        NOP
        NOP
        MOV      R0,#STARTN
        MOV      R5,#32
CLEAR:MOV       @R0,#0
        INC      R0
        DJNZ     R5,CLEAR          ;清零寄存器
        MOV      VOLN,#16
        MOV      P0_IN_REG,#0FFH
        MOV      20H, #01H         ;初始化
        mov      SCON,#50H
        MOV      TMOD,#21H
        MOV      TH0, #0D8H
        MOV      TL0, #0F0H        ;TIME0   10MS 定时
        mov      TH1, #0F3H
        mov      TL1, #0F3H        ;TIME1   初始化波特率(2400)
        SETB     REN               ;允许串口接收
        SETB     ES                ;开串口中断
        SETB     TR0
        SETB     TR1
        SETB     ET0
        MOV      P3,#0FFH
        SETB     EA
MAIN:
        LCALL    SCAN_KEY                ;按键扫描
        LCALL    dsa_data_transmit   ;数据处理
        LJMP      MAIN
```

```
;;;;;;;;;;;;;;串口中断程序（暂时没用）;;;;;;;;;;;;;;;
SERIAL:
        JNB     RI,SEND_RET
        CLR     RI
        MOV      A,SBUF
        RETI
SEND_RET:
        CLR     TI
SERIAL_END:
        RETI

;;;;;;;;;;;;;;;10MS 定时中断;;;;;;;;;;;;;;;;;;
TIME0:
        PUSH    ACC
        PUSH    PSW
        CLR     TR0
        MOV      TH0,#0D8H
        MOV      TL0,#0F0H
        SETB    FLAG_10MS
        dec     dsa_timeout
        MOV      A,dsa_timeout
        CJNE     A,#0,TIME00
        SETB    timeout_flag
TIME00:
        SETB    TR0
        POP      PSW
        POP      ACC
        RETI

;;;;;;;;;;;;;;;;;;数据处理程序;;;;;;;;;;;;;;;
dsa_data_transmit:
        jb      comm_flag,transmit
                ret
transmit:
```

```
        mov     r0,#startn
        lcall   dsa_syn_start
        jnb     comm_ok,dsa_data_reset

        mov     a,@r0    ;startn
        lcall   dsa_send_byte
        jnb     comm_ok,dsa_data_reset

        inc     r0
        mov     a,@r0    ;num1
        mov     r4,a     ;存放字节长度
        lcall   dsa_send_byte
        jnb     comm_ok,dsa_data_reset

transmit_loop:
        inc     r0
        mov     a,@r0    ;num1
        lcall   dsa_send_byte
        jnb     comm_ok,dsa_data_reset
        DJNZ    R4,transmit_loop

        lcall   dsa_comm_acknowledge
        jnb     comm_ok,dsa_data_reset
dsa_data_reset:
        clr     comm_flag        ;数据发送接收
        SETB    STB
        SETB    ack
        SETB    dat
        ret

;;;;;;;;;;;;;;;按键扫描程序;;;;;;;;;;;;;;;;;;;;;;;;
SCAN_KEY:
        JB      FLAG_10MS,KEY
              RET
```

```
KEY:
        CLR     FLAG_10MS
        MOV       A,P1
        mov       DUMMY,a
        XRL       A,P0_IN_REG           ;第一次初始化为 0FFH
        ANL       A,P0_IN_REG     ;判断为下降沿确定为按键按下
        MOV       P0_IN_REG,DUMMY;存 P0 口本次扫描的值
        CJNE    A,#0H,K1
    RET
K1:
        CJNE    a,#01h,K2
        LCALL   music_one       ;播放第一首
        RET
    K2:
        CJNE    a,#02h,K3
        LCALL   music_pause     ;播放/暂停
        RET
    K3:
        CJNE    a,#04h,K4
        LCALL   music_stop      ;语音停止
        RET
    K4:
        CJNE    a,#10h,K5
        LCALL   music_up        ;上一曲
        RET
    K5:
        CJNE    a,#20h,K6
        LCALL   music_down      ;下一曲
        RET
K6:
        CJNE    a,#40h,K7
        LCALL   vol_inc         ;音量+
        RET
K7:
```

```
        CJNE      a,#80h,KEY_END
        LCALL   vol_dec          ;音量-
                RET
KEY_END:
                RET
;=====================================================================
;功能描述: 赋值对应的控制命令
;=====================================================================
music_one:
        mov       startn,#7eh
        mov       num1,#04h
        mov       num2,#0a5h
        mov       num3,#00h
        mov       num4,#01h
        mov       num5,#7eh
        mov       num6,#00h
        mov       num7,#00h
        mov       endn,#00h
        SETB     comm_flag
        ret
music_pause:
        mov       startn,#7eh
        mov       num1,#02h
        mov       num2,#0a1h
        mov       num3,#7eh
        mov       num4,#00h
        mov       num5,#00h
        mov       num6,#00h
        mov       num7,#00h
        mov       endn,#00h
        SETB     comm_flag
        ret
music_stop:
        mov       startn,#7eh
```

```
        mov     num1,#02h
        mov     num2,#0a3h
        mov     num3,#7eh
        mov     num4,#00h
        mov     num5,#00h
        mov     num6,#00h
        mov     num7,#00h
        mov     endn,#00h
        SETB    comm_flag
        ret
music_up:
        mov     startn,#7eh
        mov     num1,#02h
        mov     num2,#0a7h
        mov     num3,#7eh
        mov     num4,#00h
        mov     num5,#00h
        mov     num6,#00h
        mov     num7,#00h
        mov     endn,#00h
        SETB    comm_flag
        ret
music_down:
        mov     startn,#7eh
        mov     num1,#02h
        mov     num2,#0a6h
        mov     num3,#7eh
        mov     num4,#00h
        mov     num5,#00h
        mov     num6,#00h
        mov     num7,#00h
        mov     endn,#00h
        SETB    comm_flag
        ret
```

```
vol_inc:
        mov     startn,#7eh
        mov     num1,#03h
        mov     num2,#0a4h
        mov     num4,#7eh
        mov     num5,#00h
        mov     num6,#00h
        mov     num7,#00h
        mov     endn,#00h
        mov     a,voln
        CJNE    a,#31,inc_x
inc_x:
        jnc     inc_end
        inc     voln
        mov     num3,voln
inc_end:
        SETB    comm_flag
        ret
vol_dec:
        mov     startn,#7eh
        mov     num1,#03h
        mov     num2,#0a4h
        mov     num4,#7eh
        mov     num5,#00h
        mov     num6,#00h
        mov     num7,#00h
        mov     endn,#00h
        mov     a,voln
        CJNE    a,#0,dec_x
dec_end:
        SETB    comm_flag
        ret
dec_x:
        dec     voln
```

```
        mov     num3,voln
        SETB    comm_flag
        ret

;=====================================================================
;函数名  : dsa_syn_start
;功能描述: 启动 DSA 总线，产生同步信号(250ms 超时)
;输入参数:
;输出参数: 启动是否成功标志，成功 comm_ok=1，失败 comm_ok=0
;=====================================================================
dsa_syn_start:
        mov     dsa_timeout,#25
        CLR     timeout_flag
        SETB    STB
        CLR     DAT

start_ackl:
        JB      timeout_flag,start_err
        JB      ACK,start_ackl    ;等待 ack 为 0

        SETB    DAT
start_ackh:
        JB      timeout_flag,start_err
        JNB     ACK,start_ackh    ;等待 ack 为 1
start_ok:
        SETB    comm_ok
        ret
start_err:
        CLR     comm_ok
        ret

;=====================================================================
;函数名  : dsa_comm_acknowledge
;功能描述: 停止 DSA 总线，产生应答信号(250ms 超时)
```

```
;输入参数:
;输出参数: 停止总线是否成功标志，成功 comm_ok=1，失败 comm_ok=0
;=====================================================================
dsa_comm_acknowledge:
        mov      dsa_timeout,#25
        CLR      timeout_flag
        SETB     STB
        CLR      ack

stop_ackl:
        JB       timeout_flag,stop_err
        JB       STB,stop_ackl      ;等待 STB 为 0

        SETB     ack
stop_ackh:
        JB       timeout_flag,stop_err
        JNB      STB,stop_ackh      ;等待 STB 为 1
stop_ok:
        SETB     comm_ok
        ret
stop_err:
        CLR      comm_ok
        ret

;=====================================================================
;函数名   : dsa_send_byte
;功能描述: DSA 总线写一字节数据(250ms 超时)
;输入参数: acc
;输出参数: 发送 DSA 数据是否成功,成功 comm_ok=1，失败 comm_ok=0
;=====================================================================
dsa_send_byte:
        mov      dsa_timeout,#25
        CLR      timeout_flag
        MOV      R5,#08H
```

```
send_loop:
        RLC     A
        MOV     DAT,C           ;先发送高位

        CLR     STB
send_ackl:
        JB      timeout_flag,send_err
        JB      ACK,send_ackl   ;等待 ack 为 0

        SETB    STB
send_ackh:
        JB      timeout_flag,send_err
        JNB     ACK,send_ackh   ;等待 ack 为 1
        DJNZ    R5,send_loop
send_ok:
        SETB    DAT
        SETB    comm_ok
        ret
send_err:
        CLR     comm_ok
        ret
          END
```

9.2、C 语言程序

DSA 通讯参考例程

MCU: STC10F04

Oscillator Crystal Frequency: 11.0592MHz

```
#include <reg51.h>              /* reg51 definitions*/
#define SUCCESS 1
#define FAILURE 0
#define   DSA_TIMEOUT     16    //250us*16 = 4ms
#define Timer_250us 0x19   //250us @ 11.0592MHz
```

```
/****** DSA 通信接口定义 *****/
sbit DSA_DATA = P2^2;
sbit DSA_STB  = P2^1;
sbit DSA_ACK  = P2^0;

/******* 按键接口定义 ******/
sbit  KEY1 = P3^7;
sbit  KEY2 = P3^6;

unsigned char uc_dsa_timeout;
unsigned char uc_dsa_data;  //保存 DSA 通讯接收的数据
unsigned char dsa_tx_buff[6]  = {0x7E, 0x04, 0xA0, 0x00, 0x02, 0x7E};
unsigned char dsa_tx_buff1[6] = {0x7E, 0x04, 0xC2, 0xff, 0xff, 0x7E};


//==========================================================================
//函数名：  Delay_10ms(); STC10F04 10ms @11.0592MHz
//功能描述：延时函数
//输入参数：无
//输出参数：无
//==========================================================================
void Delay_10ms(unsigned int z)
{
    unsigned int i,j;
    for(i=z;i>0;i--)
    {
        for(j=8450;j>0;j--);
    }
}

//==========================================================================
//函数名：  Init_Timer
//功能描述：初始化定时器
//输入参数：无
```

```
//输出参数：无
//==========================================================================
void Init_Timer()
{
    TMOD = 0x02;        //Timer0 Mode 2: 8-bit auto-reload
    TL0 = Timer_250us;
    TH0 = Timer_250us;

    ET0 = 1;
    EA = 1;
}

//==========================================================================
//函数名：   Timer0_ISR
//功能描述：定时器 0 中断服务函数
//输入参数：无
//输出参数：无
//==========================================================================
void Timer0_ISR() interrupt 1 using 1   //250us 中断一次
{
    TR0 = 0;

    if(uc_dsa_timeout > 0)
    {
        TR0 = 1;
        uc_dsa_timeout--;
    }
}

//==========================================================================
//函数名：   Dsa_Bus_Syn_Start
//功能描述：启动 DSA 总线，产生同步信号
//输入参数：无
//输出参数：启动是否成功标志
```

```
//=============================================================================
unsigned char Dsa_Bus_Syn_Start()
{
    DSA_DATA = 0;     //DATA 拉低，开始同步
    DSA_STB  = 1;
    uc_dsa_timeout = DSA_TIMEOUT;                    //同步超出 4ms 则退出

    TR0 = 1;

    while((DSA_ACK == 1) && (uc_dsa_timeout > 0));     //等待 ACK 拉低

    if(uc_dsa_timeout == 0)
    {
    return(FAILURE);
    }
    DSA_DATA = 1;       //DATA 拉高

    while((!DSA_ACK)&& (uc_dsa_timeout > 0));//等待 ACK 拉高,完成同步
    if(uc_dsa_timeout == 0)
        return(FAILURE);
    return(SUCCESS);
}

//=============================================================================
===============
//函数名：   Dsa_Bus_Ack_Write
//功能描述: 发送完数据,停止 DSA 总线，通讯应答。
//输入参数：无
//输出参数：停止总线是否成功标志
//=============================================================================
===============
unsigned char Dsa_Bus_Ack_Write(void)
{
    DSA_ACK = 0; //ACK 拉低
```

```
    uc_dsa_timeout = DSA_TIMEOUT; //通讯应答超出 4ms 则退出

    while((DSA_STB) && (uc_dsa_timeout > 0)); //等待 STB 拉低
    if(uc_dsa_timeout == 0)
        return(FAILURE);

    DSA_ACK = 1;                                    //ACK 拉高，等待 STB 拉高
    while((!DSA_STB) && (uc_dsa_timeout > 0));      //等待 STB 拉高,完成通讯应答
    if(uc_dsa_timeout == 0)
        return(FAILURE);

    return(SUCCESS);
}

//==========================================================================
===============
//函数名：   Dsa_Bus_Ack_Read
//功能描述：接收完 MP3 模块返回的数据，产生 DSA 应答信号
//输入参数：无
//输出参数：DSA 应答是否成功标志
//==========================================================================
===============
unsigned char Dsa_Bus_Ack_Read(void)
{
    DSA_ACK = 1;
    uc_dsa_timeout = DSA_TIMEOUT; //通讯应答超出 4ms 则退出

    TR0 = 1;

    while((DSA_STB) && (uc_dsa_timeout > 0)); //等待 STB 拉低
    if(uc_dsa_timeout == 0)
        return(FAILURE);

    DSA_ACK = 0;    //ACK 拉低
```

```
    while((!DSA_STB) && (uc_dsa_timeout > 0));       //等待 STB 拉高
    if(uc_dsa_timeout == 0)
        return(FAILURE);
    DSA_ACK = 1;    //ACK 拉高
    return(SUCCESS);
}

//==========================================================================
===============
//函数名：    Dsa_Bus_Byte_Write
//功能描述：DSA 总线写一字节数据（4ms 超时）
//输入参数：待发送的 DSA 数据
//输出参数：发送 DSA 数据是否成功
//==========================================================================
===============
unsigned char Dsa_Bus_Byte_Write(unsigned char temp_data)
{
    unsigned char x;

    uc_dsa_timeout = DSA_TIMEOUT; //传输一字节超出 4ms 则退出

    TR0 = 1;

    for(x = 0; x < 8; x++)
    {
        //把数据输出到 DATA
        if(temp_data & 0x80)
            DSA_DATA = 1;
        else
            DSA_DATA = 0;

       temp_data <<= 0x01;
        DSA_STB = 0;  //拉低 STB,通知接收端读取数据
```

```
        while((DSA_ACK) && (uc_dsa_timeout > 0));//等待接收端读取数据后把 ACK 拉低
        if(uc_dsa_timeout == 0)
            return(FAILURE);

        DSA_STB = 1;  //STB 拉高

        while((!DSA_ACK) && (uc_dsa_timeout > 0));      //等待 ACK 拉高,完成一个位的传送
        if(uc_dsa_timeout == 0)
            return(FAILURE);
    }
    DSA_DATA = 1;
    return(SUCCESS);
}


//=============================================================================
===============
//函数名：   Dsa_Bus_Byte_Read
//功能描述：DSA 总线读一字节数据（4ms 超时）
//输入参数：无
//输出参数：1.读数据是否成功标志
//                2.读出的数据（uc_dsa_data）
//=============================================================================
===============
unsigned char Dsa_Bus_Byte_Read(void)
{
    unsigned char x;

    uc_dsa_data = 0;
    uc_dsa_timeout = DSA_TIMEOUT; //传输一字节超出 4ms 则退出

    TR0 = 1;
```

```
    for(x = 0; x < 8; x++)
    {
        while((DSA_ACK) && (uc_dsa_timeout > 0));//等待 ACK 拉低
        if(uc_dsa_timeout == 0)
            return(FAILURE);

        uc_dsa_data <<= 1;

        if(DSA_DATA)
            uc_dsa_data |= 0x01;

        DSA_STB = 0;  //STB 拉低，DSA 标准时序是 ACK 拉低

        while((!DSA_ACK) && (uc_dsa_timeout > 0));//等待 ACK 拉高
        if(uc_dsa_timeout == 0)
            return(FAILURE);

        DSA_STB = 1;  //STB 拉高
    }
    return(SUCCESS);
}


//================================================================================
===============
//函数名：   Dsa_Bus_Communicate
//功能描述：与 MP3 模块 DSA 通讯函数
//输入参数：dsa_data_buff(通讯命令数据数组)
//输出参数：无
//================================================================================
===============
unsigned char Dsa_Bus_Communicate(unsigned char *dsa_data_buff)
{
    unsigned char dsa_point, dsa_length;
```

```
if(Dsa_Bus_Syn_Start() != SUCCESS) goto dsa_bus_reset;
if(Dsa_Bus_Byte_Write(dsa_data_buff[0]) != SUCCESS) goto dsa_bus_reset;
if(Dsa_Bus_Byte_Write(dsa_data_buff[1]) != SUCCESS) goto dsa_bus_reset;
if(Dsa_Bus_Byte_Write(dsa_data_buff[2]) != SUCCESS) goto dsa_bus_reset;

dsa_point = 0x03;
dsa_length = dsa_data_buff[1] - 1;

if((dsa_data_buff[2] & 0x0F0) != 0x0C0)   //处理不带数据返回的命令
{
    while(dsa_length--)
    {
        if(Dsa_Bus_Byte_Write(dsa_data_buff[dsa_point++]) != SUCCESS)
            goto dsa_bus_reset;
    }
    if(Dsa_Bus_Ack_Write() != SUCCESS)          //通讯应答
        goto dsa_bus_reset;
}
else    //处理带数据返回的命令
{
    while(dsa_length--)
    {
        if(Dsa_Bus_Byte_Read()!= SUCCESS)   //接受 MP3 模块返回的数据
            goto dsa_bus_reset;
        dsa_data_buff[dsa_point++] = uc_dsa_data;
    }
    if(uc_dsa_data != 0x7E)
        goto dsa_bus_reset;
    if(Dsa_Bus_Ack_Read() != SUCCESS)          //通讯应答
        goto dsa_bus_reset;
}
DSA_DATA = 1;
DSA_ACK = 1;
```

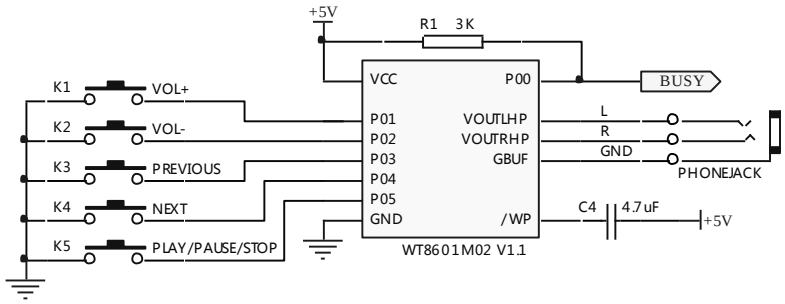
```
    DSA_STB = 1;
    return(SUCCESS);

dsa_bus_reset:  //DSA 总线复位到空闲状态
    {
        DSA_DATA = 1;
        DSA_ACK = 1;
        DSA_STB = 1;

        return(FAILURE);
    }
}

//==========================================================================
===============
//函数名：   Dsa_Bus_Data_Transmit
//功能描述：发送通讯命令，如果不成功，则重复发送 3 次，超过 3 次则不再重新发送
//输入参数：dsa_tx_buff(通讯命令数据数组)
//输出参数：无
//==========================================================================
===============
unsigned char Dsa_Bus_Data_Transmit(unsigned char *dsa_tx_buff)
{
    unsigned char cnt;

    for(cnt = 0;cnt < 3;cnt++)
    {
        if(Dsa_Bus_Communicate(dsa_tx_buff) == SUCCESS)
        {
        return(SUCCESS);
        }
    }

    return(FAILURE);                //3 次通讯不成功，退出
```

```
}


void main()
{
    Init_Timer();

    //DSA 总线初始化
    DSA_DATA = 1;
    DSA_ACK = 1;
    DSA_STB = 1;

    while(1)
    {
        if(KEY1 == 0)
        {
            Dsa_Bus_Data_Transmit(dsa_tx_buff);
        }

        if(KEY2 == 0)
        {
            Dsa_Bus_Data_Transmit(dsa_tx_buff1);
        }
        Delay_10ms(20);
    }
}
```

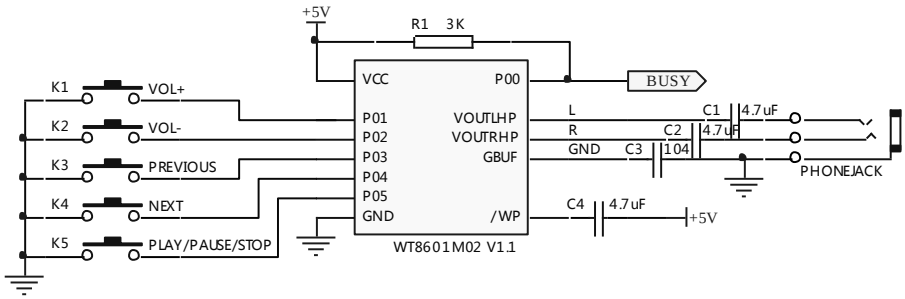
10、应用电路

10.1、MP3 控制模式应用电路

外接耳塞模式

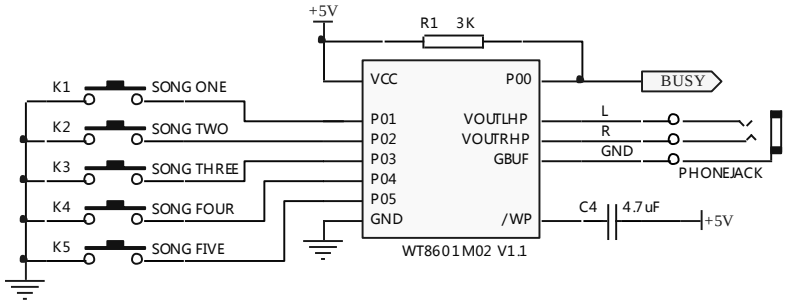


外接功放模式

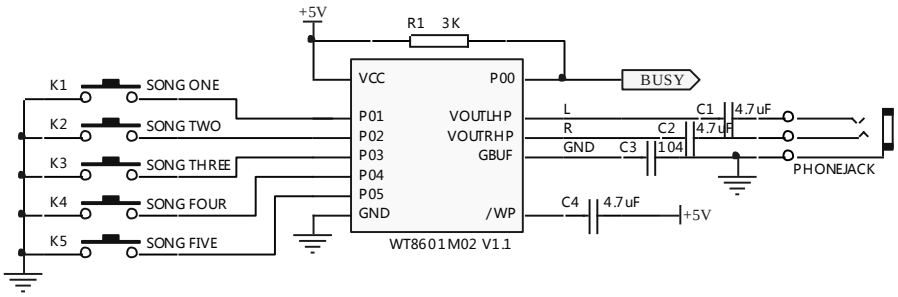


10.2、按键一对一控制模式应用电路

外接耳塞模式



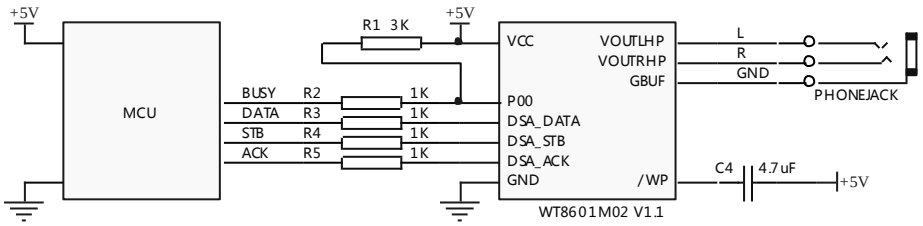
外接功放模式



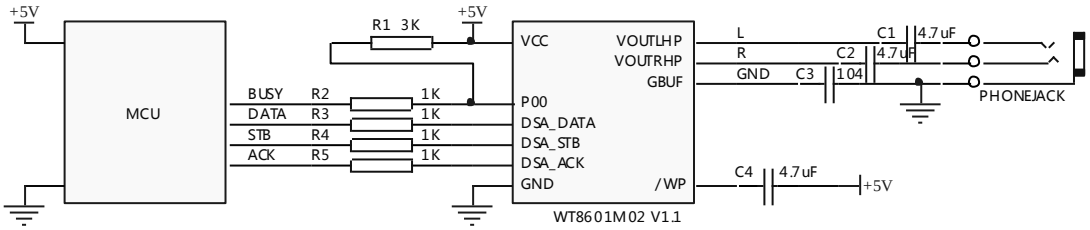
10.3、MCU 控制模式应用电路

在 MCU 控制模式中，P00 为 BUSY 输出端，语音停止播放时，BUSY 为高电平，语音播放过程中，BUSY 为低电平。可以由 MCU 来检测 WT8601M02 V1.1 的语音播放状态。如 MCU 为 3V 供电，则不用接 R2、R3、R4、R5。

外接耳塞模式

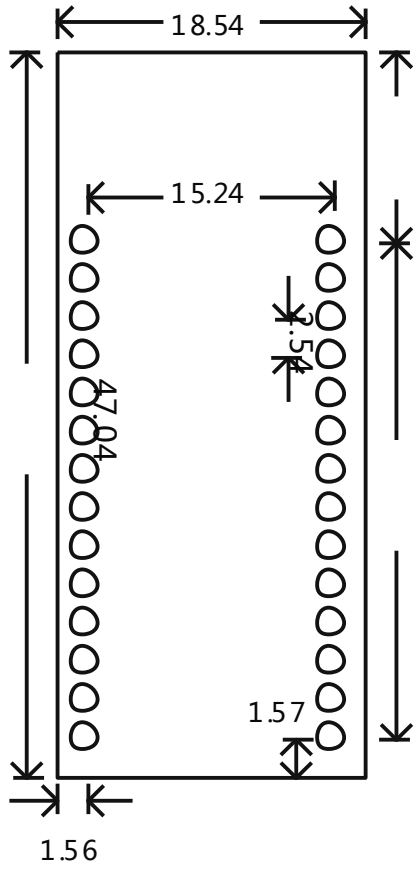


外接功放模式



11、封装尺寸图

单位：mm



12、历史版本记录

版本	日期	描述
V1.0	2009-9-24	原始版本
V1.1	2009-10-12	修正电路图部分
V1.2	2009-10-28	修改部分描述
V1.3	2009-11-13	修正DSA控制程序，完善DSA命令描述，修改应用电路
V1.4	2010-9-4	更新联系方式
V1.5	2010-9-28	针对WT8601M02 V1.1模块更新管脚说明及应用电路
V1.6	2010-10-29	更新DSA命令信息，增加config.txt说明
V1.7	2011-9-22	更新注意事项:更新语音后要重新插拔一次USB,避免数据更新不成功 更新DSA通信描述

广州唯创电子有限公司（原广州唯创科技有限公司）1999 年创立于广州市天河区，是一家集语音芯片研发、语音产品方案设计、语音产品生产、语音编辑上位机软件开发的高新技术公司。业务范围涉及汽车电子、多媒体、家居防盗、通信、家电、医疗器械、工业自动化控制、玩具及互动消费类产品等领域。团队有着卓越的 IC 软、硬件开发实力和设计经验，秉持着「积极创新、勇于开拓、满足顾客、团队合作」的理念，力争打造“语音业界”的领导品牌。

唯创主要生产 WTV 系列语音芯片、WTR 可录音系列语音芯片、WT588D 语音芯片、WTB 系列语音芯片、WTM 系列高音质语音应用模块、WTF 系列的高性价比长时间播放模块，及特约代理的 APLUS 系列语音芯片、ISD 全系列可录放语音芯片等。率先提供最完备、多元化的客需解决方案，节约研发成本，缩短研发周期，使产品在最短的时间内成熟上市。在汽车电子及特种车领域，自主研发的公交车报站器在国内有着很好的市场口碑，为叉车使用安全而开发的叉车超速报警器是国内第一家研发此类产品并大量生产的企业。

唯创坚持“以人为本，不断进行核心技术创新，优良的售后技术跟踪服务”的经营策略，使得唯创能傲立于语音产品行业。WTV 系列语音芯片、WTR 可录音系列语音芯片、WTM 系列高音质语音应用模块、WTF 系列的高性价比长时间播放模块等都是唯创的自主品牌，具有很强的市场竞争优势。产品、模块、编辑软件等的人性化设计，使得客户的使用更方便。于 2006 年新成立的北京唯创虹泰分公司主要以销售完整的方案及成熟产品为宗旨，以便于为国内北方客户提供更好的服务。

唯创持续在研发与技术升级领域大力投资，每年平均提拨超过 20%的营业额作为研发经费，在我们的研发团队中，有超过 90%员工钻研技术及产品发展。并与同行业大厂合作，勇于迈出下一个高峰。

总公司名称：广州市唯创电子有限公司

电话：020-85638557 85638660 38357061 38055581 传真：020-85638637

技术支持 E-mail：sos@1999c.com 网址：http://www.w1999c.com

地址：广东省广州市天河区棠东东路 55 号 3 楼

分公司名称：北京唯创虹泰科技有限公司

电话：010-89756745 传真：010-89750195

E-mail：BHL8664@163.com 网址：www.wcht1998.com.cn

地址：北京昌平区立汤路 186 号龙德紫金 3#902 室

广州唯创电子有限公司深圳办事处

移动电话：0755-36956575 83044339 83555462 传真：0755-83044339

业务支持 E-mail：sos@1999c.com

地址：深圳福田区福华路 29 号 7 层 7E