WT8603M01 Datasheet

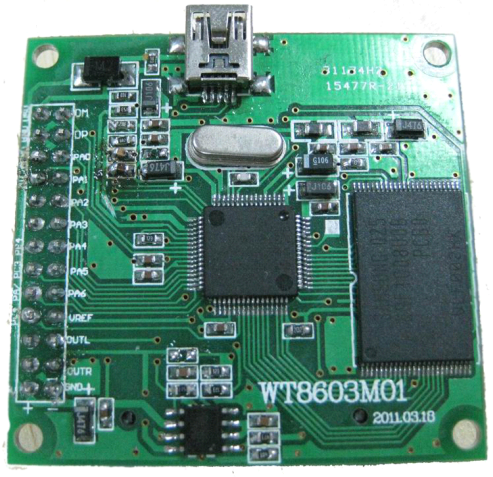
1、Features

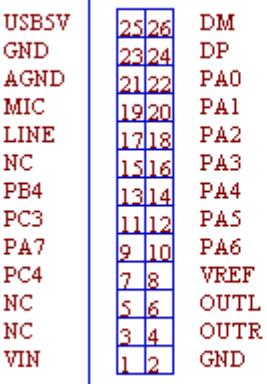
- Support playback of 8Kbps ~ 320Kbps bit rate MP3 audio files
- High quality stereo audio output
- Support MP3 control mode, MP3 one-to-one key control mode，recording parallel control mode, standard recording key control mode, and DSA control mode
- With the functions of automatic playback when power on, single loop, all loop, and random playback
- Support SD card and NAND-Flash as the memories simultaneously, with large storage space and long voice time duration
- Support long time voice recording
- Support 32MB to 32GB SD card and 128MB to 2GB NAND-Flash
- Change the control mode, the default volume level when power on, and other functions through the TXT file
- Full speed USB2.0 data transfer
- Support FAT16 and FAT32 file systems
- Support Win98(needs to install the driver)/2000/XP/VISTA operating systems
- Operating Voltage: DC5V

2、Applications

WT8603M01 is very suitable to be used in high class home appliances, such as intelligent voice navigation refrigerator, voice navigation air conditioner, voice navigation induction cooker, and etc., high class toy, electronic system for automobile as well as the occasions needing high sound quality and long time sound playback.

3、Wiring Illustration





WT8603M01 Front Side Picture

WT8603M01 Front Block Diagram

WT8603M01 Wiring Description

Pin No.	Function	Description	Pin No.	Function	Description
1	GND	Power Ground	14	PB4	BUSY
2	VIN	Power Positive	15	PA3	Key 4/ASK
3	OUTR	Right sound channel audio output	16	NC	Idle
4	NC	Idle	17	PA2	Key 3/DATA
5	OUTL	Left sound channel audio output	18	NC	Idle
6	NC	Idle	19	PA1	Key 2/STB
7	VREF	Audio Ground	20	MIC+	MIC+ input
8	PC4	Idle	21	PA0	Key 1
9	PA6	Key 7	22	AGND	MIC- input
10	PA7	Key 8	23	DP	USB Data Terminal
11	PA5	Key 6	24	GND	Power Ground
12	PC3	Idle	25	DM	USB Data Terminal
13	PA4	Key 5	26	USB+5V	USB Power

4、Electrical Parameters

Environment Temp：25℃ Input Voltage: DC5V

Parameter	Mark	Environmental Condition	Min Value	Typical Value	Max Value	Unit
Operating Voltage	V_{DD}	F_{sys}=12MHz	3.5	5.0	5.5	V
Operating Current1	I_{OP1}	No load	48.5	50.9	51.2	mA

Operating Current2	I_{OP2}	Rout=8ohm	62.5	80	250	mA

5、Function Settings

WT8603M01 can set the following functions through a new built config.txt file on PC. The specific settings are shown as below.

If you do not set the config.txt file, the module will start according to the default mode. The specific command functions are described as follows.

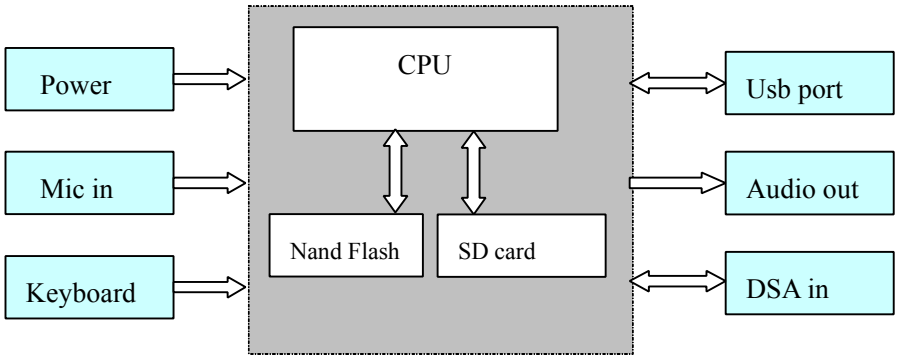
No.	Command	Description	Data	Function
1	vl	Volume setting by default	0～31	Volume setting, can enter volume level 0～31
2	fa	Fade in and fade out effect	0	Cancel fade in and fade out effect
			1	Set fade in and fade out effect(Default)
3	pp	Automatic playback when power on	0	Cancel automatic playback when power on (Default)
			1	Set automatic playback when power on
4	sn	Set the song needed to play automatically when power on	00001.mp3	If the song name contains other words, only need to enter 5 numbers before the words, like 00001my love.mp3, 00002pretty boy.mp3, and so on
5	pm	Play mode	0	Set stop after playing single song(Default)
			1	Set single song loop
			2	Set all songs loop
			3	Set random play mode
6	cm	Control mode	0	MP3 control mode (with MCU mode) (Default)
			1	MP3 one-to-one key control mode (with MCU mode)

			2	Parallel control mode (without MCU mode)
			3	Recording parallel control mode (without MCU mode)
			4	standard recording key control mode (without MCU mode)
7	rs	Recording sampling rate	0	Sampling rate at 16KHZ (Default)
			1	Sampling rate at 8KHZ

After finish setting config.txt, copy it to the root of WT8603M01 based on PC. WT8603M01 will execute the commands in the config.txt firstly after power on or reset.

6、Structure Instructions

This module consists of the CPU, NAND Flash, SD card slot, USB interface, DSA communication interface, audio output, recording input and power input. Below is the general structure diagram.



7、Naming Rule of Audio Files

Music files are stored in the root directory of WT8603M01, and name them in the way of using 5 digits plus the suffix name, such as 00001.mp3, 00002.mp3, 00003.mp3 and etc. In order to memorize audio file names and contents easily, can adopt the method of using the serial number plus the original filename to rename, such as 00001My Love.mp3, 00002Yesterday Once More.mp3, and etc. When you need to play the specified music file, it will be done only send the five digits. Had better put all the MP3 files in the root directory and define the file number according to the file storage order.

The recording files are automatically saved in the VOICE folder.(does not support directory search function)

8、Control Mode

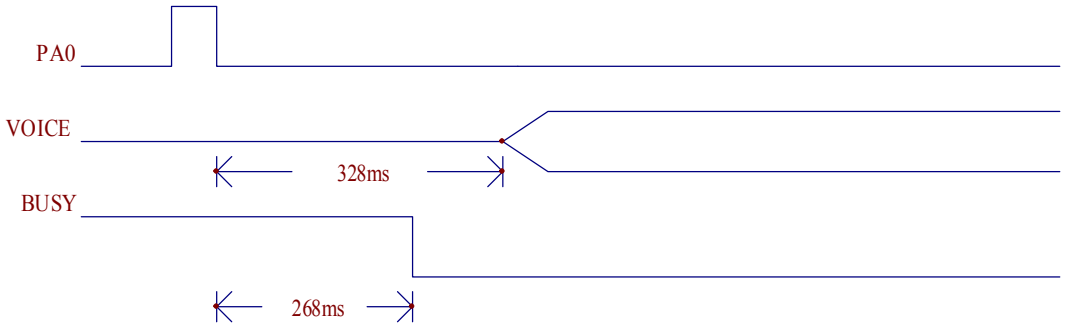
WT8603M01 supports MP3 control mode, one-to-one key control mode, random play control mode, MCU control mode, and etc. The control modes can be changed by sending codes with a MCU.

8.1、MP3 Control Mode

In the MP3 control mode, I/O PA0、PA4~PA7 keep10ms high level to trigger the relative functions. The corresponding function for each I/O is shown as below. PB4 is the output port, which is at low level in the process of audio playback and at high level when playing is stopped.

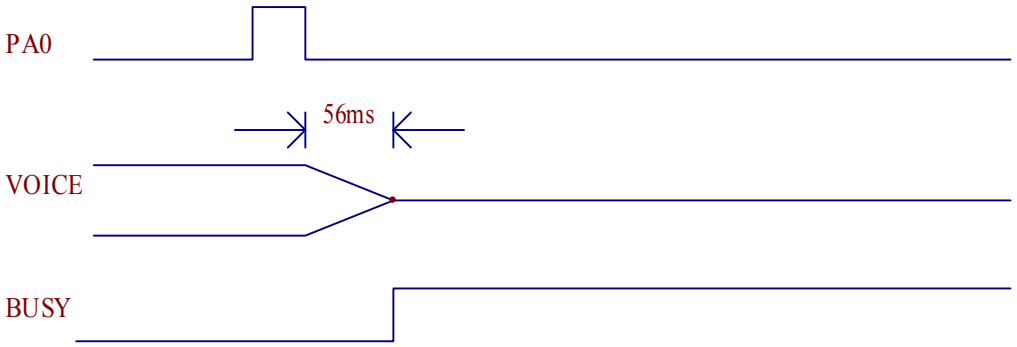
I/O	PA0	PA4	PA5	PA6	PA7	PB4
Function	Play/Pause	Previous	Next	Vol+	Vol-	Busy

PLAY



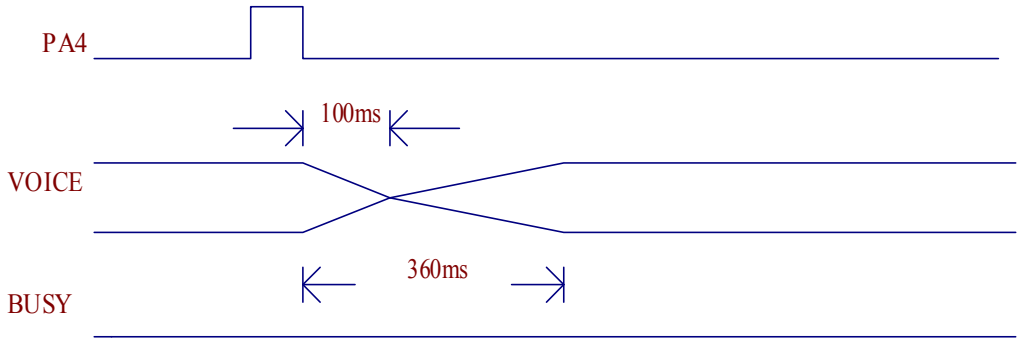
At the state of stop, use 40ms～500 ms low level to trigger PA0. After triggering 268ms, the BUSY signal turns to low level; after triggering 328ms, the audio file is started to play.

STOP



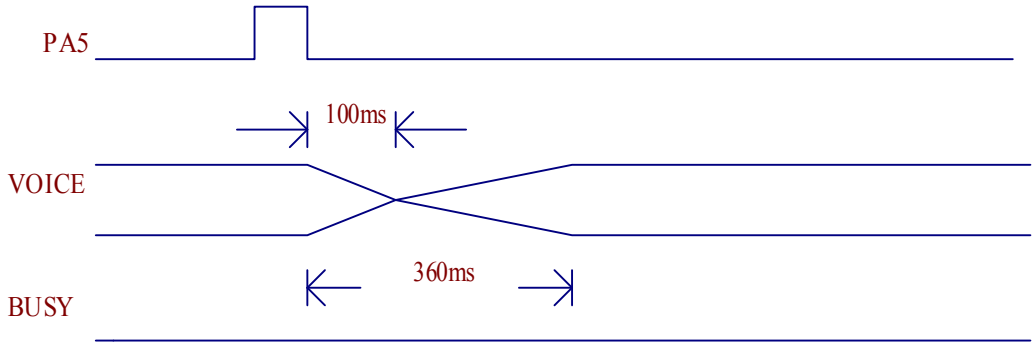
At the state of playing, use 40ms～500 ms high level to trigger PA0 and can stop playing the current voice. After triggering, the volume gradually becomes lower; after 56ms, stops playing completely, and the BUSY signal turns to high level at the same time.

PREVIOUS



At the sate of playing, use 40ms～500ms high level to trigger PA4, and the current volume gradually becomes lower until stop playing after 100ms; then switches to the previous song to play, and the volume gradually becomes louder. During this period, the BUSY always keeps at low level.

NEXT



At the sate of playing, use 40ms～500ms high level to trigger PA5, and the current volume gradually becomes lower until stop playing after 100ms; then switches to the next song to play, and the volume gradually becomes louder. During this period, the BUSY always keeps at low level.

8.2、One-to-one Key Control Mode

In the one-to-one control mode, WT8603M01 only can play maximum 5 songs, and each I/O port is corresponding to each song. The I/O ports PA0, PA4～PA7 keep 40ms~500ms high level to trigger the related functions. PB4 is the output port, which is at low level in the process of audio playback and at high level when playing is stopped.

I/O	PA0	PA4	PA5	PA6	PA7	PB4
Function	Song 1	Song 2	Song 3	Song 4	Song 5	Busy
File Name	Voice of song 1	Voice of song 2	Voice of song 3	Voice of song 4	Voice of song 5	No

8.3、MP3 Parallel Port Control Mode

In the MP3 parallel port control mode, the pin PA0 is defined as the address trigger pin STB; short press STB by 40~500ms for playback, and play the corresponding voice through changing the address bit An. The PB4 is the output port, and the level is low in the sate of playing while the level is high when stop playing.

Group N	Address Pin						
	A6（PA7）	A5（PA6）	A4（PA5）	A3（PA4）	A2（PA3）	A1（PA2）	A0（PA1）
Group1	0	0	0	0	0	0	0
Group2	0	0	0	0	0	0	1
Group3	0	0	0	0	0	1	0
……	……	……	……	……	……	……	……
Group128	1	1	1	1	1	1	1

8.4、Recording Parallel Port Control Mode

In the recording parallel control mode, there are two functional modes, i.e. voice recording mode and recording playback mode, both of which can be switched to each other through PA1. While PA1 is at high level, it is for recording mode, and while PA1 is at low level, it is for recording playback mode.

In the recording mode, you can record maximum 32 groups recordings; PA0 is used to trigger recording and stop recording. At the state of not recording, short press PA0 by 40ms~500ms to start recording, and the BUSY turns to low level from high level. In the state of recording, short press PA0 by 40ms ~ 500ms to stop recording and save, and the BUSY turns to high level from low level.

At the recording playback mode, PA0 is defined as the PLAY button, and play the corresponding voice by changing the address bit An. PA2 is defined as DEL button, short press PA2 by 40ms~500ms to delete the voice corresponding to the address bit An. Long press PA2 by 3~4 seconds to delete all the recording files under the folder VOICE.

Group N	Address Pin				
	A4（PA7）	A3（PA6）	A2（PA5）	A1（PA4）	A0（PA3）
Group1	0	0	0	0	0
Group2	0	0	0	0	1
Group3	0	0	0	1	0
……	……	……	……	……	……

Group64	1	1	1	1	1

8.5、Standard Recording Control Mode

In the standard recording control mode, there are two functional modes, i.e. voice recording mode and recording playback mode, both of which can be switched to each other through PA1. While PA1 is at high level, it is for recording mode, and while PA1 is at low level, it is for recording playback mode. (The playback order depends on the recording order, not the file name order)

I/O	PA0	PA1	PA2	PA3	PA4	PA5	PA6	PA7	PB4
Function	Play/Pause	Switching Port	DEL	REC	Previous	Next	VOL+	VOL-	Busy

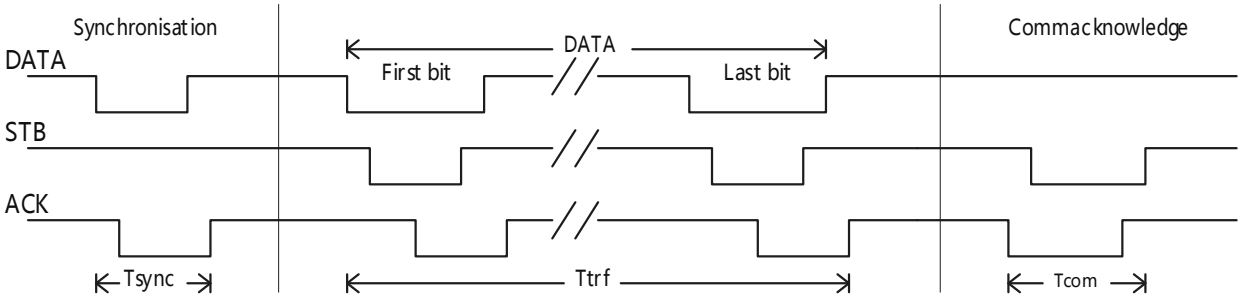
In the recording mode, only PA3 (recording button) and PB4 (Busy) are effective. If the current state is not for recording, short press PA3 by 40~500ms to start recording, and the BUSY turns to low level from high level; If the current state is for recording, short press PA3 by 40~500ms to stop the recording and save, and the BUSY turns to high level from low level.

In the recording playback mode, short press PA0, PA4-PA7 to operate the recoding files normally. Short press PA2 by 40ms-500ms to delete current recording file; while long press PA2 by 3-4s to delete all the recording files in the drive letter/folder "VOICE".

8.6、MCU Control Mode

Use the three ports of DSA_DATA、DSA_ACK、DSA_STB to control WT8603M01 at MCU control mode. This protocol is defined based on the standard DSA mode, taking up less system resource and with no restrict demand to time.

8.6.1、Timing Control



First MCU sends DATA signal pulled low to WT8603M01. After WT8603M01 receives low level signal (will not work when receiving high level signal), it will send return messages to MCU through ACK. After detect ACK is at low level, it will pull high DATA. After detect ACK is at high level, it will send First and then STB. Then after detect ACK is at low level, it will pull high STB and then DATA. Only do detect ACK is at high level, then it is possible to send next data continually. Send Last bit (First bit – Last bit is the whole length of the command) in the same way, and after detect ACK is at high level, switch the status of STB and ACK. Send ACK to WT8603M01 through MCU, and after WT8603M01 pulls low STB, MCU pulls high ACK; then wait WT8603M01 to pull high STB. When this operation is finished, WT8603M01 confirms the data sent is valid.

If any time among Tsync, Ttrf and Tcom exceeds 100ms, WT8603M01 will judge it as failure data. Send high-order digits first and low order digits.

8.6.2、Switching Commands of Function Modes

WT8603M01 has MP3 Playback Mode, Voice Recording Mode, and Recording Playback Mode of three function modes. When you implement the corresponding action, you have to switch to the appropriate mode, otherwise it can not work normally. The functions are different under different modes. (Default Mode: MP3 Playback mode)

Switch to MP3 Playback Mode

Start Code	Length	Command	End Code
7E	02	B1	7E

Switch to Recording Mode

Start Code	Length	Command	End Code
7E	02	B2	7E

Switch to Recording Playback Mode

Start Code	Length	Command	End Code
7E	02	B3	7E

8.6.3、Switching Commands of Playback Modes

Start Code	Length	Command	Parameter	End Code
7E	03	B4	00：Single Non-loop Playback Mode	7E
			01：Single Loop Playback Mode	
			02：All Loop Playback Mode	
			03：Random Mode	

8.6.4、Switching of Current Storage Medium, Function Mode, and Information State Switching Commands of Current Storage Medium

WT8603M01 can only have NAND FLASH as the storage medium, and can also have both NAND FLASH and SD card as the storage media at the same time. Generally, just operate to the NAND FLASH, and if in some special case, SD card can be specified to operate. For example, the voice recording files can be stored in NAND FLASH or SD card.(After switching mode, it is better to check the information of current storage medium to make sure the correct operation). The settings for operative storage medium of MP3 playback mode and voice recording mode are independent, and both of them can be set to use different storage media. Recording playback mode and voice recording mode share one storage medium and one set value. The module has power off memory function, and when powered on again, the module reads the audio files from the storage medium you set last time. But after switch to the operative storage medium in the voice recording mode and recording playback mode, you still switch to the MP3 mode to save the set value of the storage medium, which means it is available to memorize the setting information of the operative storage medium when power off. After setting of the operative storage medium in the MP3 mode, it will automatically save the set value. After the SD card is taken away, it will automatically switch to the master drive NAND FLASH.

Start Code	Length	Command	Parameter	End Code
7E	03	B5	00: Switch to master drive NAND-FLASH 01: Switch to the drive SD card	7E

Returns to Current Function Mode and Information State.

Start Code	Length	Command	Function Mode	Drive Letter	End Code
7E	04	CE	FF	FF	7E

MCU sends 7E 04 CE FF FF 7E, and WT8603M01 returns 7E 04 CE XX XX 7E. The first XX represents the current function mode(B1 is MP3 Playback Mode, B2 is Voice Recording Mode, and B3 is Recording Playback Mode); The second XX represent the current storage medium(00 is NAND Flash and 01 is SD card).

8.6.5、Other Commands in MP3 Playback Mode

Specify a File Name to Play

This command can specify a filename to play, free from the order of files storage.

Start Code	Length	Command	High-order	Low-order	End Code
7E	04	A0	00	01	7E

This command above in the example is specified to play the audio file named 00001.mp3 in the public area.

Specify to Play an Index Music

Index files are based on the storage order, and send the commands below can specify a file to play.

Start Code	Length	Command	High-order	Low-order	End Code
7E	04	A5	00	01	7E

This command in the example above is specified to play the first index music file.

Pause

Start Code	Length	Command	End Code
7E	02	A1	7E

MCU sends this command for the first time to pause. Send it again to continue to play the music.

Stop

Start Code	Length	Command	End Code
7E	02	A3	7E

MCU sends this command to stop playing the current music.

Volume Control

There are totally 32 levels of voice control, i.e. 00-31. Please note here needs hex to represent, i.e.00-1FH. 00(00H) is the mute while 31(1FH) is the max volume. The default volume level can be set through the file config.txt.

Start Code	Length	Command	VOL Level	End Code
7E	03	A4	1F	7E

MCU sends this command to get the max volume level 31 in the above example.

Previous

Start Code	Length	Command	End Code
7E	02	A7	7E

This command triggers to play the previous music. When playing the last music, this command can trigger to play the first one.

Next

Start Code	Length	Command	End Code
7E	02	A6	7E

This command triggers to play the next music. When playing the first music, this command can trigger to play the last one.

Read Current Audio Index Information

Start Code	Length	Command	High-order	Low-order	End Code
7E	04	C1	FF	FF	7E

After MCU sends this command, WT8603M01 returns 7E 04 C1 XX XX 7E，XX XX to tell you the index position.

Read Total Audio Numbers

Start Code	Length	Command	High-order	Low-order	End Code
7E	04	C2	FF	FF	7E

After MCU sends 7E 04 C2 FF FF 7E, WT8603M01 returns 7E 04 C2 XX XX 7E. XX XX means the total audio numbers in the public area.

Read the Current State of Playback

Start Code	Length	Command	State	End Code
7E	03	C4	FF	7E

After MCU sends this command, WT8603M01 returns the current state of playback. For example, MCU sends 7E 03 C4 FF 7E, then WT8603M01 returns 7E 03 C4 XX 7E. If the XX is 01, it means normal playback; if the XX is 02, it means stop; if the XX is 03, it means pause.

Read Current Volume Level

Start Code	Length	Command	VOL	End Code
7E	03	C5	FF	7E

After MCU sends the command 7E 03 C5 FF 7E, WT8603M01 returns 7E 03 C5 XX 7E. Then the XX is the current volume level.

Read Current Specified File Information

MCU sends this command to read current specified file being played in the public area. Existence here means this file is existed, and just reads the first five digits of the file name. The unit of file capacity is byte, and the unit of time length is second.

Start Code	Length	Comm.	Exist.	File Name		File Capacity (byte)				Time Length (sec)		End Code
				High-order	Low-order	High-order	……	……	Low-order	High-order	Low-order	
7E	0B	C7	FF	FF	FF	FF	FF	FF	FF	FF	FF	7E

After MCU sends 7E 0B C7 FF FF FF FF FF FF FF FF FF 7E, WT8603M01 returns the messages below.

Start Code	Length	Comm.	Exist.	File Name		File Capacity (byte)				Time Length(sec)		End Code
				High-o	Low-o	High-o	……	……	Low-o	High-o	Low-o	

				rder	rder	rder			rder	rder	rder	
7E	0B	C7	01	00	01	00	2E	1F	00	01	0A	7E

Then it shows there is a file or more existed in WT8603M01. The current file is 00001.mp3, the capacity is 3022592 bytes, and the time length is 266 seconds.

8.6.6、Other Commands under Voice Recording Mode

Specify a File Name to Start Recording

This command specifies to build an audio file with the specified name under the folder "VOICE", and start recording

in this file.

Start Code	Length	Command	High-order	Low-order	End Code
7E	04	D0	00	01	7E

The command in the example specifies to build the audio file named MICR0001.WAV under the folder "VOICE", and then start recording in this file.

Start and Pause for Recording in Order

Start Code	Length	Command	End Code
7E	02	D2	7E

If the module is at the state of stop, send this command to build a new file arranged a name in sequence and start recording. For example, there is an existing audio file named MICR0003.WAV, and after send the command it will automatically generate a file named MICR0004.WAV and start recording. If the module is at the state of recording or pause, after send the command it will execute the opposite operation (recording suspended or recording recovery).

Stop Recording and Save

Start Code	Length	Command	End Code
7E	02	D3	7E

If the module is at the state of recording, MCU sends this command to stop recording and save.

Setting for Recording Sampling Rate

Start Code	Length	Command	Parameter	End Code
7E	03	D4	00: 16K sampling rate (Default)	7E
			01: 8K sampling rate	

Read Total Space and Free Space in Current Drive (available only under voice recording mode)

Start Code	Length	Command	Total Space				Free Space				End Code
7E	0A	C9	FF	FF	FF	FF	FF	FF	FF	FF	7E

MCU sends 7E 0A C9 FF FF FF FF FF FF FF FF 7E, then suppose WT8603M01 returns 7E 0A C9 00 06 65 77 00 06 5D D7 7E, so the first four pairs of data 00 06 65 77 is the total space (0X66577 * 512 = 214625792bytes = 204.68MB) while the last four pairs of data 00 06 5D D7 is the free space (0X65DD7 * 512 = 213626368bytes = 203.73MB) . The data acquired is the total cluster numbers, and need to multiply 512 to be bytes.

8.6.7、Other Commands under Recording Playback Mode

Play the Specified Recording File

Start Code	Length	Command	High-order	Low-order	End Code
7E	04	E0	00	01	7E

The command in the example is specified to play the recording file named MICR0004.WAV in the folder VOICE.

Replay

Start Code	Length	Command	End Code
7E	02	E1	7E

MCU sends this command to replay the audio.

Stop

Start Code	Length	Command	End Code
7E	02	E3	7E

MCU sends this command to stop playing the audio being played.

Volume Control

There are totally 32 levels of voice control, i.e. 00-31. Please note here needs hex to represent, i.e.00-1FH. 00(00H) is the mute while 31(1FH) is the max volume. The default volume level can be set through the file config.txt.

Start Code	Length	Command	VOL Level	End Code
7E	03	E4	1F	7E

MCU sends this command to get the max volume level 31 in the above example.

Previous

Start Code	Length	Command	End Code
7E	02	E7	7E

This command triggers to play the previous audio file. When playing the last audio file, this command can trigger to

play the first one.

Next

Start Code	Length	Command	End Code
7E	02	E6	7E

This command triggers to play the next audio file. When playing the first audio file, this command can trigger to play the last one.

Delete Current File

Start Code	Length	Command	End Code
7E	02	EC	7E

MCU sends this command to delete the current operation file.

Delete All

Start Code	Length	Command	End Code
7E	02	ED	7E

MCU sends this command to delete all recording files in the current drive.

Read the Current File Being Played

Start Code	Length	Command	High-order	Low-order	End Code
7E	04	C1	FF	FF	7E

After MCU sends this command, the module returns 7E 04 F1 XX XX 7E, and XX XX represent the numbers of the file name. For example, the module returns 7E 04 F1 00 24 7E, then the current file being played is MICR0036.WAV.

Read Total Numbers of Audio Files

Start Code	Length	Command	High-order	Low-order	End Code
7E	04	C2	FF	FF	7E

After MCU sends 7E 04 C2 FF FF 7E, WT8603M01 returns 7E 04 C2 XX XX 7E, then XX XX is the total numbers of audio files.

Read Current Specified File Information

MCU sends this command to read current specified file being played. Existence here means this file is existed in the USB drive, and just reads the first five digits of the file name. The unit of file capacity is byte, and the unit of time length is second.

Start Code	Length	Comm	Exist	File Name	File Capacity (byte)	Time Length	End

										(sec)		Code
				High-order	Low-order	High-order	……	……	Low-order	High-order	Low-order	
7E	0B	C7	FF	FF	FF	FF	FF	FF	FF	FF	FF	7E

After MCU sends 7E 0B C7 FF FF FF FF FF FF FF FF FF 7E, it returns the information below.

Start Code	Length	Comm.	Exist.	File Name		File Capacity (byte)				Time Length (sec)		End Code
				High-order	Low-order	High-order	……	……	Low-order	High-order	Low-order	
7E	0B	C7	01	00	01	00	2E	1F	00	01	0A	7E

Then it shows there is a file or more existed in WT8603M01. The current file is MICR0001.WAV, the capacity is 3022592 bytes, and the time length is 266 seconds.

9、Control Program Example

9.1、Assembly Program

```
;*********************************************************************
;Project: DSA Communication Test Program for WT8603M01
;Function Requirements
;       1). Communicate with PC via RS232
;       2). Communicate with WT8603M01 via DSA communication protocal
;       3). Achieve simple key control mode
;Hardware Configuration
;       1) MCU Model: AT89C2051
;       2) External crystal frequency:11.0592MHz
;       3) I/O Definition:
        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 Programming Language

```
#include <iom8v.h>
#include <macros.h>
#include "define.h"
#include "dsa_ctl.h"

extern uint8     uc_dsa_data;
extern uint8     uc_dsa_timeout;

INT8U dsa_bus_data_transmit(INT8U * dsa_tx_buff);

void delay_x10us(INT8U times)
{
      while(--times);
}

//=====================================================================
//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
//=====================================================================
INT8U dsa_bus_syn_start()
{
     P_DSA_DATA_LOW();
      DSA_DATA_OUTPUT();
     P_DSA_STB_HIGHT();
      DSA_STB_OUTPUT();
      DSA_ACK_INPUT();
     uc_dsa_timeout = DSA_TIMEOUT;                          //同步 250ms 则超时退出]
     while(P_DSA_GET_ACK_PIN() && (uc_dsa_timeout>0));
      if(uc_dsa_timeout == 0) return(FAILURE);
      P_DSA_DATA_HIGHT();
     while((!P_DSA_GET_ACK_PIN()) && (uc_dsa_timeout>0));
```

```
    if(uc_dsa_timeout == 0) return(FAILURE);
    return(SUCCESS);
}

//=====================================================================
//函数名    : dsa_bus_ack_write
//功能描述: 停止 DSA 总线，产生应答信号
//输入参数: 无
//输出参数: 停止总线是否成功标志
//=====================================================================
INT8U dsa_bus_ack_write(void)
{                                                       //250ms 超时
    P_DSA_ACK_LOW();                                    //ACK 拉低，等待 STB 拉低
    DSA_ACK_OUTPUT();
    uc_dsa_timeout = DSA_TIMEOUT/10;
    DSA_STB_INPUT();
    DSA_DATA_INPUT();
    while(P_DSA_GET_STB_PIN() && (uc_dsa_timeout>0));
    if(uc_dsa_timeout == 0) return(FAILURE);
    P_DSA_ACK_HIGHT();                                  //ACK 拉高，等待 STB 拉高
    delay_x10us(10);
    //    while((!P_DSA_GET_STB_PIN()) && (uc_dsa_timeout>0));
    //    if(uc_dsa_timeout == 0) return(FAILURE);
    return(SUCCESS);
}

//=====================================================================
//函数名    : dsa_bus_ack_read
//功能描述: 读 DSA 应答信号
//输入参数: 无
//输出参数: 读 DSA 应答信号是否成功标志
//=====================================================================
INT8U dsa_bus_ack_read(void)                        //250ms 超时
{
    P_DSA_ACK_HIGHT();
```

```
    DSA_STB_INPUT();
    DSA_ACK_OUTPUT();
    uc_dsa_timeout = DSA_TIMEOUT/10;
    while(P_DSA_GET_STB_PIN() && (uc_dsa_timeout>0));
    if(uc_dsa_timeout == 0) return(FAILURE);
//  DSA_STB_OUTPUT();
    P_DSA_ACK_LOW();                                    //pull down DSA_ACK
    while((!P_DSA_GET_STB_PIN()) && (uc_dsa_timeout>0));
    if(uc_dsa_timeout == 0) return(FAILURE);
    P_DSA_ACK_HIGHT();
    return(SUCCESS);
}

//=====================================================================
//函数名    : dsa_bus_byte_write
//功能描述: DSA 总线写一字节数据(125ms 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
//=====================================================================
INT8U dsa_bus_byte_write(INT8U temp_data)
{
    INT8U x;
    uc_dsa_timeout = DSA_TIMEOUT/10;
    for(x = 0; x < 8; x++)
    {
        if(temp_data & 0x80)
            P_DSA_DATA_HIGHT();
        else
            P_DSA_DATA_LOW();

        temp_data <<= 0x01;
        P_DSA_STB_LOW();                                //DSA_STB 拉低,等待 ACK 拉低应答
        DSA_ACK_INPUT();
        while(P_DSA_GET_ACK_PIN() && (uc_dsa_timeout>0));
        if(uc_dsa_timeout == 0) return(FAILURE);
```

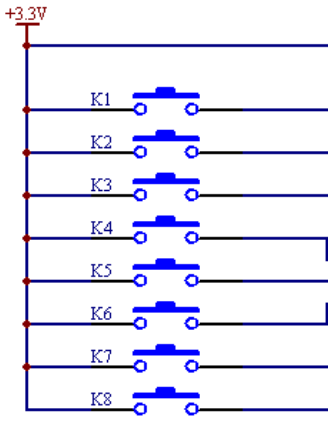
```
            P_DSA_STB_HIGHT();                                          //拉高 DSA_STB,等待 ACK 拉高
            while((!P_DSA_GET_ACK_PIN()) && (uc_dsa_timeout>0));
            if(uc_dsa_timeout == 0) return(FAILURE);
        }
        return(SUCCESS);
}

//=====================================================================
//函数名    : dsa_bus_byte_read
//功能描述: DSA 总线读一字节数据(125ms 超时)
//输入参数: NONE
//输出参数: 1.读数据是否成功标志
//                  2.读出的数据(uc_dsa_data)
//=====================================================================
INT8U dsa_bus_byte_read(void)
{                                                                       //125ms 超时
        INT8U x;
        uc_dsa_data = 0;
        uc_dsa_timeout = DSA_TIMEOUT/10;
        for(x = 0; x < 8; x++)
        {
            DSA_ACK_INPUT();                                            //等待 DSA_ACK 拉低
            while(P_DSA_GET_ACK_PIN() && (uc_dsa_timeout>0));
            if(uc_dsa_timeout == 0) return(FAILURE);
            uc_dsa_data <<= 1;
            if(P_DSA_GET_DATA_PIN())
                uc_dsa_data |= 0x01;
            P_DSA_STB_LOW();                                            //DSA_STB 拉低,等待 ACK 拉高
            while(!P_DSA_GET_ACK_PIN() && (uc_dsa_timeout>0));
            if(uc_dsa_timeout == 0) return(FAILURE);
            P_DSA_STB_HIGHT();
        }
        return(SUCCESS);
}
```

```
//=====================================================================
//函数名    : dsa_bus_communicate
//功能描述:
//输入参数:
//输出参数: NONE
//=====================================================================
INT8U dsa_bus_communicate(INT8U *dsa_data_buff)
{
    INT8U 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]&0xf0) != 0xC0)
    {
        while(dsa_length--)
        {
            if(dsa_bus_byte_write(dsa_data_buff[dsa_point++]) != SUCCESS)
                goto dsa_bus_reset;
        }
        if(dsa_bus_ack_write() != SUCCESS) goto dsa_bus_reset;
    }
    else
    {
        DSA_DATA_INPUT();
        while(dsa_length--)
        {
            if(dsa_bus_byte_read() != SUCCESS) goto dsa_bus_reset;
            dsa_data_buff[dsa_point++] = uc_dsa_data;
        }
        if(uc_dsa_data != 0x7E) goto dsa_bus_reset;
```

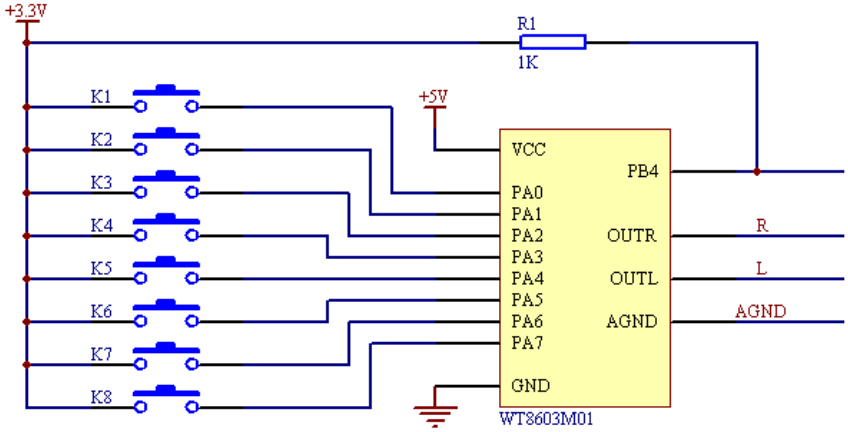
```
        if(dsa_bus_ack_read() != SUCCESS) goto dsa_bus_reset;
    }
    DSA_DATA_INPUT();
    DSA_STB_INPUT();
    DSA_ACK_INPUT();
    return(SUCCESS);
    //-----------------------------------------------------------------
dsa_bus_reset:                                          //DSA 写数据通讯超时推出
    DSA_DATA_INPUT();
    DSA_STB_INPUT();
    DSA_ACK_INPUT();
    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
//输出参数: NONE
//=====================================================================
INT8U dsa_bus_data_transmit(INT8U * dsa_tx_buff)
{
    INT8U 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
}
```

10、Application Circuits

10.1、Application Circuit for MP3 Control Mode

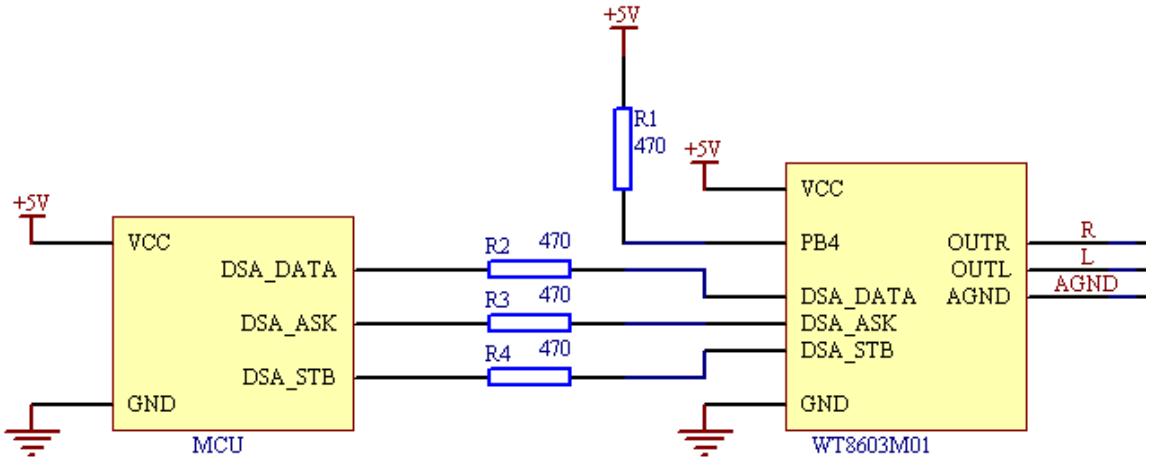


10.2、Application Circuit for One-to-one Key Control Mode

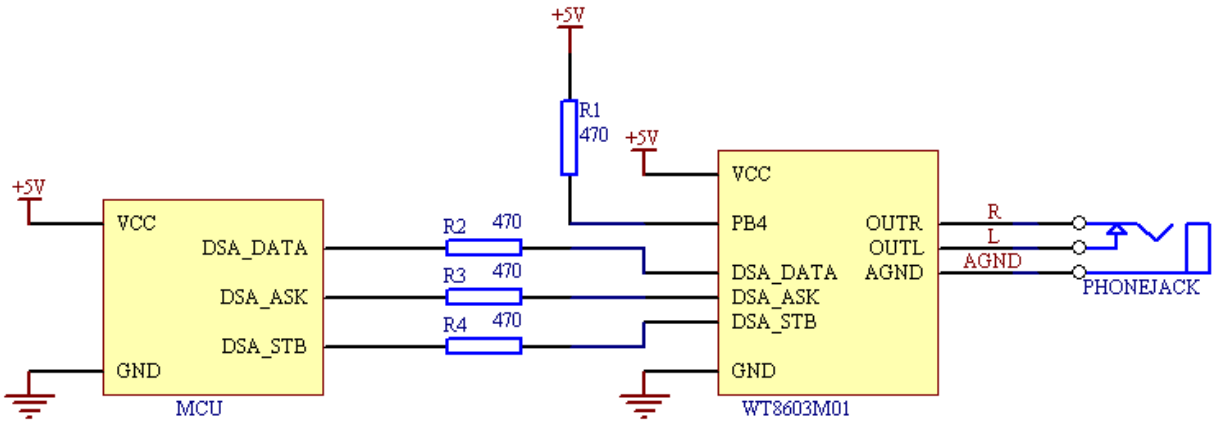


10.3、Application Circuit for MCU Control Mode

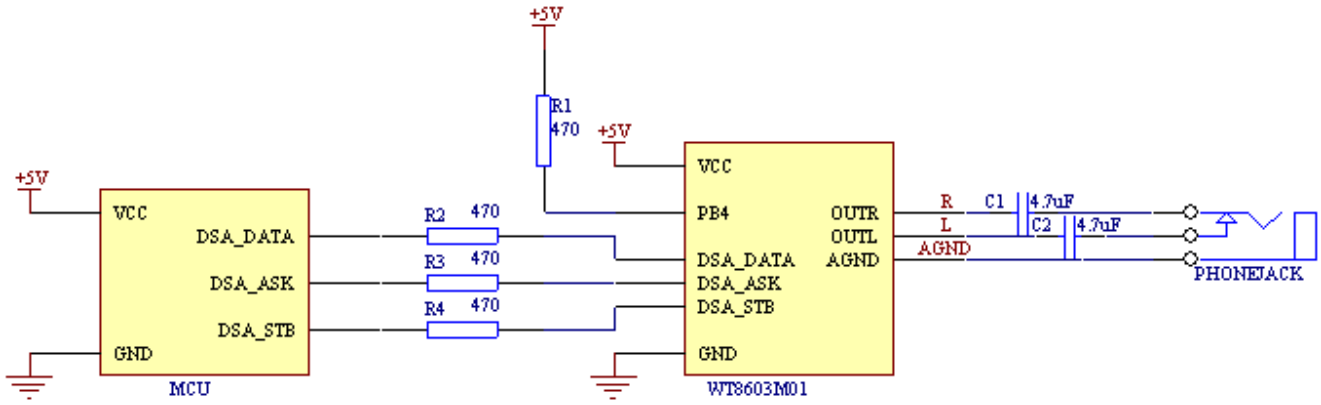
In the MCU control mode, PB4 is BUSY output port.; when stopping voice playback, BUSY is at high level; when playing the voice, BUSY is at low level. You can detect the state of WT8603M01 via MCU.



10.4、Application Circuit for Connecting a Headphone



10.5、Application Circuit for Connecting an External Amplifier



11、Dimension Drawing

Unit：mm