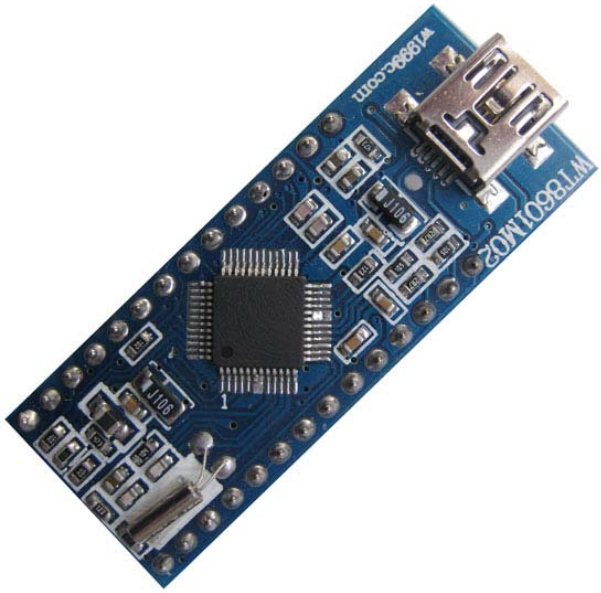
MP3 Module Built-in Memory

WT8601M02



Manual

Data Sheet

Document Date: Apr. 20th, 2009

Document Revision: V1.3

Content of table

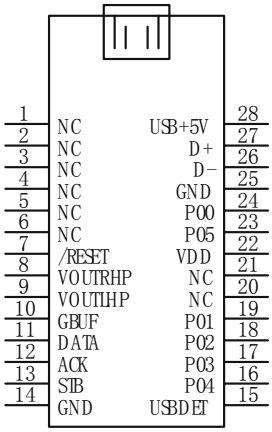
1. Feature

* MP3 format audio players, support 8Kbps ~ 320Kbps bit-rate.
* Support VBR variable bit-rate and CBR constant bit-rate.
* High-quality stereo audio output.
* Self-reset automatically when power-on, low voltage, external pin, etc.
* Control mode: MP3 mode, key mode (one-to-one), random play, DSA, etc.
* With play automatically, single song loop, all of song loop.
* For NAND-Flash as a storage center, a lot of space for storage, long duration time.
* Support 2GB byte NAND-Flash.
* USB2.0 data transfer.
* Low-power operation, longer life.
* Operating Voltage: DC5V.

2. Applications

Home appliances, such as intelligent navigation of refrigerators, air conditioners, induction cooker and so on, as well as advanced toys, automotive electronic systems, long time playback systems require, etc.

3. Connection icon



PINS	FUNCTIONS	ILLUMINATE	PINS	FUNCTION	ILLUMINATE
1	NC	NO	15	USBDET	USBDET
2	NC	NO	16	P04	KEY 4
3	NC	NO	17	P03	KEY 3
4	NC	NO	18	P02	KEY 2

5	NC	NO	19	P01	KEY 1
6	NC	NO	20	NC	NC
7	/RESET	RESET PIN	21	NC	NC
8	VOUTRHP	AUDIO OUTPUT RIGHT CHANNNEL	22	VDD	POWER POSITIVE
9	VOUTLHP	AUDIO OUTPUT LEFT CHANNNEL	23	P05	KEY5
10	GBUF	AUDIO GROUND	24	P00	BUSY OUT POWER
11	DATA	DSA DATA TRANSMISSION AND START SYMBOL	25	GND	USB PWR GND
12	ACK	DSA RESPONDENT SYMBO,L	26	D-	USB_D-
13	STB	DSA SAVE DATA SYMBOL	27	D+	USB_D+
14	GND	POWER GROUND	28	USB+5V	USBPOWER

Notes: 1. GBUF can not connect GND.

2. USBDET is USB voltage detection, the module already within processing circuit, needn't to connect.

4. Parameter

Ambient temperature：25℃ Input Voltage: DC5V P00 pull-up resistor: 10KΩ

PARAMETER	REMARK	CONDITIONS	MIN VALUE	TYPICAL VALUE	MAX VALUE	Unit
OPERATING VOLTAGE	V_{DD}	F_{sys}=12MHz	3.5	5.0	6.0	V
OPERATING	I_{OP1}	NO LOAD	24.2	26.5	28.0	mA

CURRENT 1						
OPERATING CURRENT 2	I_{OP2}	R_{ROUT}=8Ω R_{LOUT}=8Ω	26.0	27.5	46.6	mA
STOP CURRENT	I_{DD2}	NO LOAD	---	12.0	---	mA
PULL-UP RESISTOR	RH	RESETn	---	75	---	KΩ

5. Function settings

Serial numbers	Operating mode	Operating	Descriptions
1	Voice revolving mode	(1). Single cycle	The current voice play loop
		(2). all loop	play loop of all stores
		(3). Singles one play/ stop	Stop playing when the current voice is finished
2	Control mode	(1). MP3 mode	Play/ pause, stop, up, down, volume adjustable
		(2). Key mode (one-to-one)	5 keys to control 5 groups voice respectively.
		(3). random play mode	any key is invalid, play the first when plug in power(default), then play randomly.
		(4). MUC mode	DAS mode, coexist with other modes
3	Power-on mode	(1). play voice when plug in power	play the first song when plug in power
		(2). No playing voice when plug in power	No playing voice when plug in power
4		(1). Start to play	Public zone was defaulted operation zone

	select	from public zone	when plug in power
		(2). Start to play from hidden zone	Hidden zone was defaulted operation zone when plug in power
5	Store loop	(1). Store play loop, respectively	It's valid when all of voice play loop, loop at the current storage zone
		(2). All of storage zone play loop	It's valid when all of voice play loop, loop at public zone and hidden zone

In option of No. 1~5, can be only set one function with a serial number. Only can set one of these control modes (options (1) (2) (3) which in the No. 2), options (4) exist in any control mode.

6. Music description of public zone/ hidden zone

Music description of public zone, defined, the MP3 music, can download via USB cable directly to NAND-Flash.

Music description of hidden zone, need to custom. The MP3 music, which we help customers copied to NAND-Flash, client can not modified and format. In order to protect customer's files from being stolen.

7. Control mode

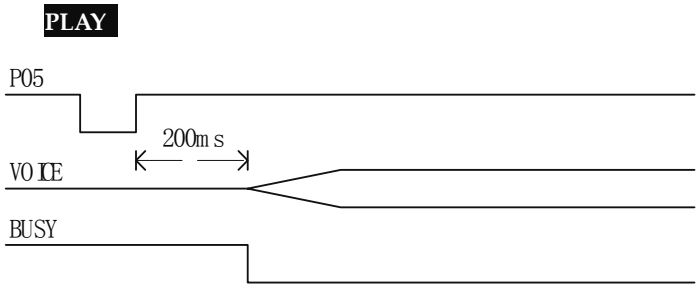
WT860M02 support MP3 mode, key mode (one-to-one), playing randomly mode, MCU control mode, etc. Control can be change by MCU code sending.

.

7.1. MP3 mode

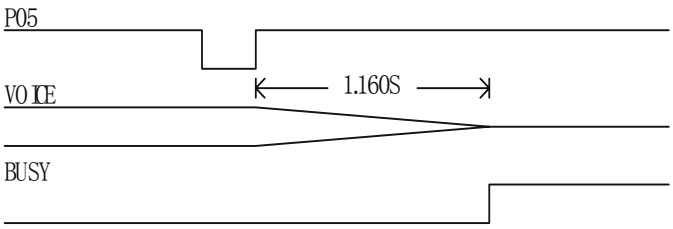
Keeping I/O P01~05 at low level can trigger the related functions. Short-pressed button P05 1 second means "play/ pause", keeping press 3 seconds means "stop". P00 is COMOUT.

I/O	P00	P01	P02	P03	P04	P05
FUNCTION	BUSY	VOLUME+	VOLUME-	UP	DOWN	PLAY/ PAUSE/ STOP



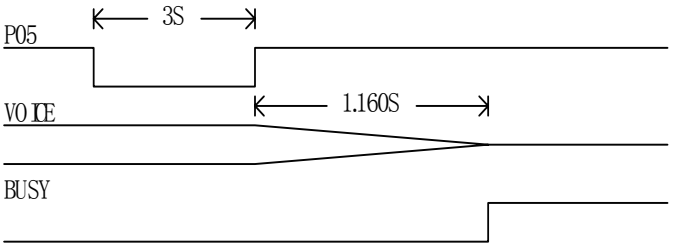
In the state of stop, trigger P05 by 10ms~1s low level, after 200ms will play voice and BUSY turn to low level, volume of the voice progressive high.

PAUSE



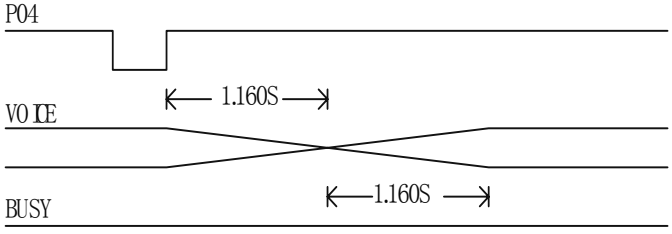
In the state of playing, trigger P05 by 10ms~1s low level, can pause the voice, the volume is becoming lower and stop after 1.160s. At this moment, BUSY signal turns to high level.

STOP



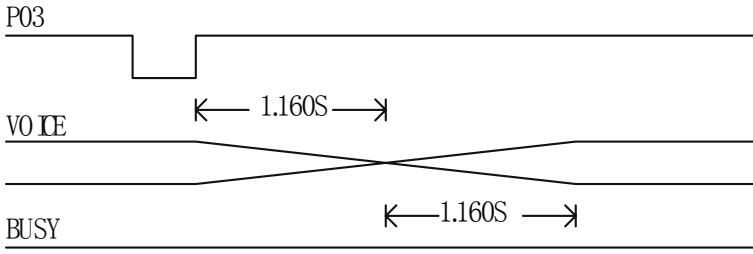
In the state of playing, keep P05 at low level 3s, can pause the voice, the volume is becoming lower and stop after 1.160s. At this moment, BUSY signal turns to high level.

DOWN



In the state of playing, trigger P04 by 10ms~1s law level, the volume is becoming lower and stop after 1.160s. Switchover to play the down voice, and the volume become loudly. At the time, BUSY always at low level.

UP



In the state of playing, trigger P03 by 10ms~1s law level, the volume is becoming lower and stop after 1.160s. Switchover to play the up voice, and the volume become loudly. At the time, BUSY always at low level.

7.2. Key mode (one to one)

In the mode, WT8601M02 can play 5 songs, max. And one I/O to one song. I/O P01~P05 keep 10ms low level can trigger related function. P00 is output port.

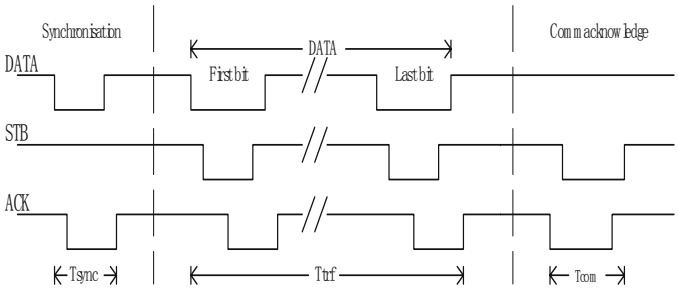
I/O	P00	P01	P02	P03	P04	P05
FUNCTIONS	BUSY	Song 1	Song 2	Song 3	Song 4	Song 5
FILE NAMES	No	Voice of song 1	Voice of song 2	Voice of song 3	Voice of song 4	Voice of song 5

7.3. MCU mode

Control by DSA_DATA、DSA_ACK、DSA_STB. This

module is DSA control, like MCU. Occupy less system resources, and no stringent requirements for time.

7.3.1. Control timing sequence



Sending a data to WT8601M02 from DATA, ACK return information to MCU when WT8601M02 received data. pull DATA after detecting ACK at low level. when the ACK at high level, send First, and then send STB. Pulled STB and DATA after detecting ACK at low level, only after detecting ACK at high level, can send the next data. In the same way to send Last bit. Sending ACK, pulled ACK and STB according to priority after detecting ACK at high level.

If time of any of Tsync, Ttrf, Tcom exceeds 250ms, were judged as failures data by WT8601M02. Data send high level first, the low.

7.3.2. Voice code of hidden zone

Switch play music

START CODE	LENGTH	COMMAND	ENDING CODE
7E	02	B0	7E

Sending this directive, can switchover between the public zone and hidden zone.

7.3.3. Voice code of public zone / hidden zone

The operation is valid for the current zone

PLAY

START CODE	LENGTH	COMMAND	FIXED DATA	TRACKS	ENDING CODE
7E	04	A5	00	01	7E

When send data, play different voice/ music if changing tracks, Track is 16 binary data. Send 7E 04 A5 00 01 7E plays song 1.

PAUSE

START CODE	LENGTH	COMMAND	ENDING CODE
7E	02	A1	7E

Sending this command, to pause the play back, send it again, and continue to play back voice/ music.

STOP

START CODE	LENGTH	COMMAND	ENDING CODE
7E	02	A3	7E

Sending this command, to stop the current play back voice/ music

VOLUME CONTROL

START CODE	LENGTH	COMMAND	VOLUME LEVELS	ENDING CODE
7E	03	A4	31	7E

Total 32 levels volume, which from 00 to 31. “00” is mute, “31” is the MAX volume. Default as level 16 when plug in power. If u want to send level 8, please send “7E 03 A4 08 7E”

UP

START CODE	LENGTH	COMMAND	ENDING CODE
7E	02	A7	7E

DOWN

START CODE	LENGTH	COMMAND	ENDING CODE
7E	02	A6	7E

7.3.4. Read the return info

Read total voice in public zone

Transmitting			Receiving		
START CODE	LENGTH	COMMAND	FIXED DATA	TRACKS	ENDING CODE
7E	04	C2	00	XX	7E

WT8601M02 return "00 XX 7E" after MUC send 7E 04 C2, and "XX" is total of voice tracks in the public.

Read total voice in hidden zone

Transmitting			Receiving		
START CODE	LENGTH	COMMAND	FIXED DATA	TRACKS	ENDING CODE
7E	04	C3	00	XX	7E

WT8601M02 return "00 XX 7E" after MUC send 7E 04 C3, and "XX" is total of voice tracks in the hidden.

Read the current volume

Transmitting			Receiving	
START CODE	LENGTH	COMMAND	VOLUME	ENDING CODE
7E	03	C5	XX	7E

WT8601M02 return "XX 7E" after MUC send 7E 04 C5, and "XX" is the current volume

Read the voice index of current play back

Index: Means the ranked number of voice in memory equipment, in other words, is the corresponding other of voice files, which copied to memory.

Transmitting			Receiving			
START CODE	LENGTH	COMMAND	PLAY AREA	FIXED DATA	TRACKS	ENDING CODE
7E	05	C1	XX	00	XX	7E

WT8601M02 return "XX 00 XX 7E" after MUC send 7E 05 C1. The return info "00" is the public zone, "01" is the hidden zone. The tracks info is voice index of playing in the public zone or hidden zone.

7.4. Command code of functions settings

START CODE	LENGTH	COMMAND	FUNCTION	ENDING CODE
7E	03	AA	XX	7E

MCU send 7E 03 AA XX 7E, can set control functions of the module. "XX" is the code of function settings, as follows:

No.	XX DATA		FUNCTIONS DESCRIPTIONS	REMARKS
1	D7	0	Default volume when starting up	Set the volume of starting up is unavailable at present
		1	Set the volume of starting up	
2	D6	0	all voice, loop play	D6 setting valid when D5 set to " 1 "
		1	single voice, loop play	
3	D5	0	D6 setting valid	
		1	Stop play of single voice	
4	D4	0	MP3 mode	If D4 and D3 combination, DSA mode is exist in any other modes.
		1	Key mode (one to one)	
5	D3	2	Play randomly mode	
		---	DSA control mode	
6	D2	0	each storage plays their own voice respectively, loop	
		1	all of storages play voice, loop	
7	D1	0	Start to play from hidden zone	
		1	Start to play from public zone	
8	D0	0	Don't play voice when boot-up	
		1	Play voice when boot-up	

Modes descriptions of D4, D3

CONTROL MODE	D4	D3
MP3 mode	0	0
Key mode (one to one)	0	1
Play randomly mode	1	0

8. Example of control program

8.1. Assembler

```
;*********************************************************************
;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
```

7.2. C language program

```
#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
}

//====================================================================
```

9. Put voice files into WT8601M02

It'd better change files name for managing more easy.

Firstly, connect WT8601M02 to computer by USB cable.

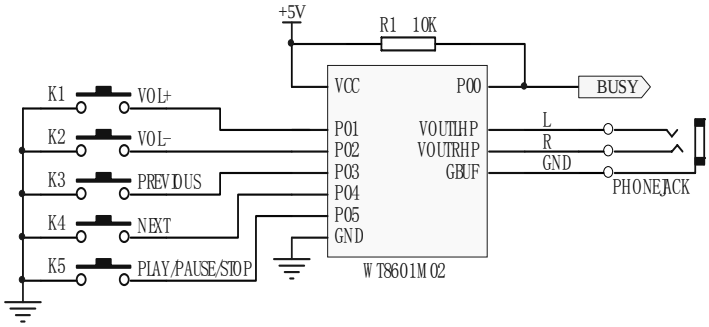
Secondly, change the files name on computer. For example, "love. Mp3", rename as "0001love.mp3", etc. (note: the "0001love.mp3" file will be defined as the voice of 01 address. Send "7E 04 A5 00 01 7E" to play back voice) And create a document, next, put all of renamed files into the document. And then, press "Ctrl+A" on keyboard, then press "Ctrl+C" to copy all mp3 files. Open the memory disc of WT8601M02, and press ""Ctrl+V" to finish putting files to WT8601M02.

Note: Please don't select, copy, paste, etc by mouse.

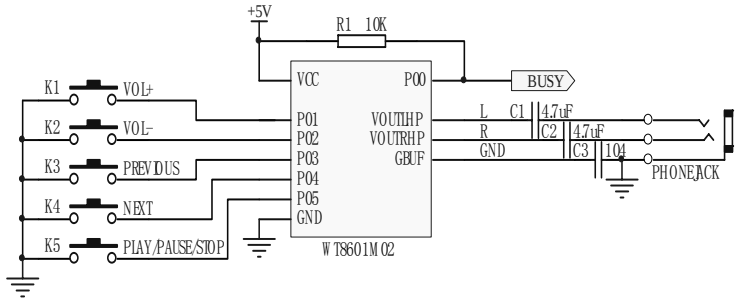
10. Application circuit

10.1. MP3 mode

External earphone

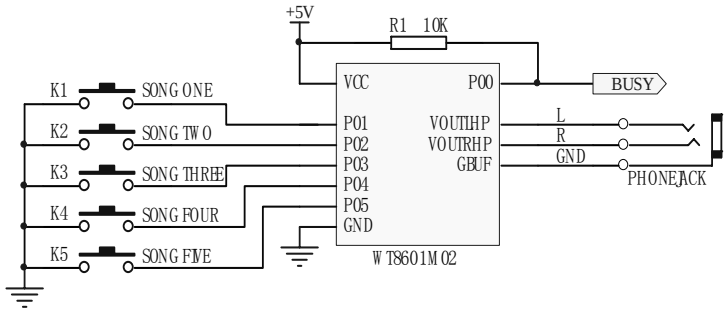


External amplifier

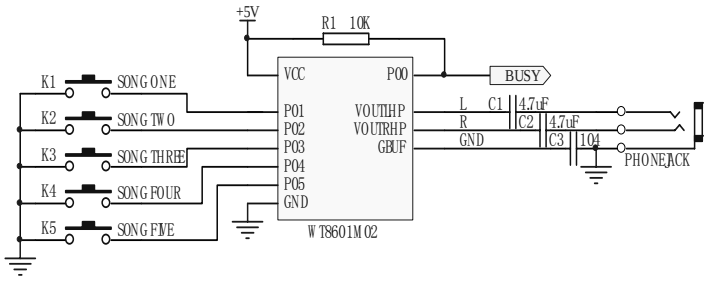


10.2. Key mode (One to one)

External earphone



External amplifier

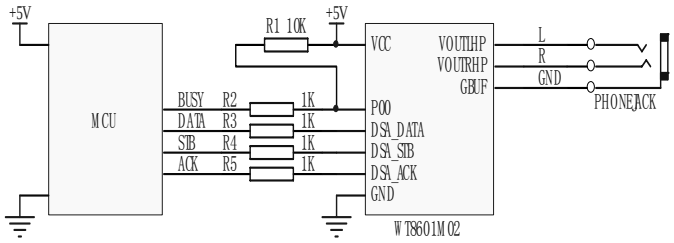


10.3. MCU mode

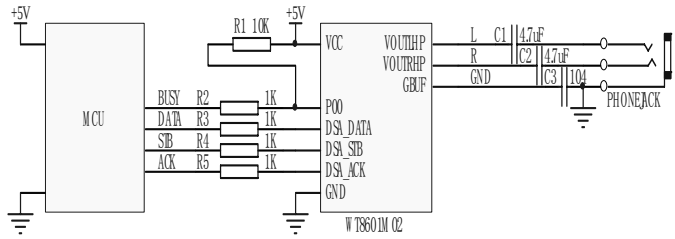
In MCU mode, P00 is the port of BUSY. BUSY at high level when stop playing, on the contrary, at low level. Can use MCU to test the playing state of voice WT8601M02. If the voltage of MCU is 3V, do not need

connect to R2、R3、R4、R5.

External earphone

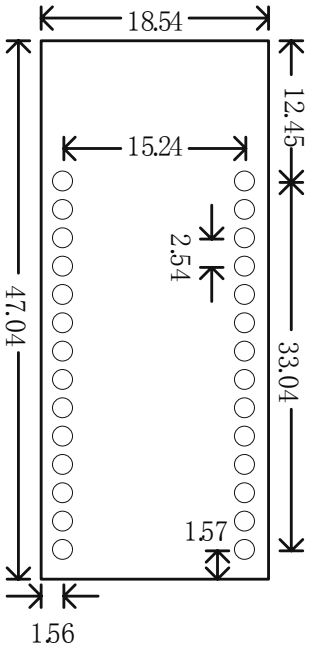


External amplifier



11. Drawing package

Unit: mm



12. History version

Version	Data	Descriptions
V1.0	2009-9-24	Original version
V1.1	2009-10-12	Amending some circuit diagram
V1.2	2009-10-28	Amending some description
V1.3	2009-11-13	Amending DSA control and completing description of DSA commend