Table of contents

1. Hardware description of the module

1. Power supply of the module

The ideal working voltage of the module is 4.2V. Therefore, if the user is powered by a 5V power supply, it is recommended to connect a diode in series

2. The indicator light of the module [power-on state]

工作状态	下载模式	播放语音	暂停	睡眠
状态	快闪	慢闪	常亮	灭

备注：正常工作状态指示灯

3. The indicator light of the module [working status]

工作状态	一对一可打断	抬起停止	一对一不可打断	标准MP3功能
状态	常亮2S	快闪2S[100ms取反]	中慢闪2S[200ms取反]	慢闪2S[500ms取反]

备注：此时指示灯，只在上电初始化的时候，指示2秒

4. Audio output description

1. SPK1 and SPK2 are connected to the speakers, regardless of the positive and negative poles. Note: only speakers below 2W are allowed to be connected.

2. Connect power amplifier or earphone, directly connect both ends of DAC-R and DAC-L, note: common ground (power ground)

5. Module debugging instructions

(1) Our module is plugged into the USB cable by default, and then it enters the download mode. When the user finishes updating the voice, trigger any IO

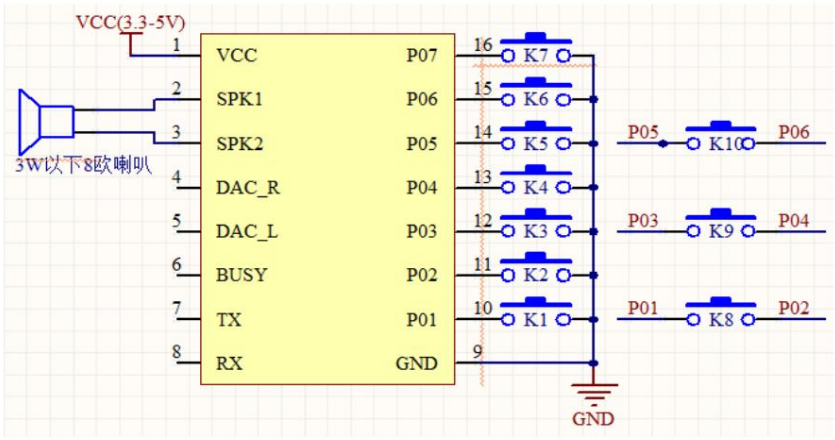
port, you can exit the download mode and return to the normal working state.

(2) Before the module is shipped, the sound has been downloaded and tested. Before the customer gets the module, they should not download the sound in a hurry.

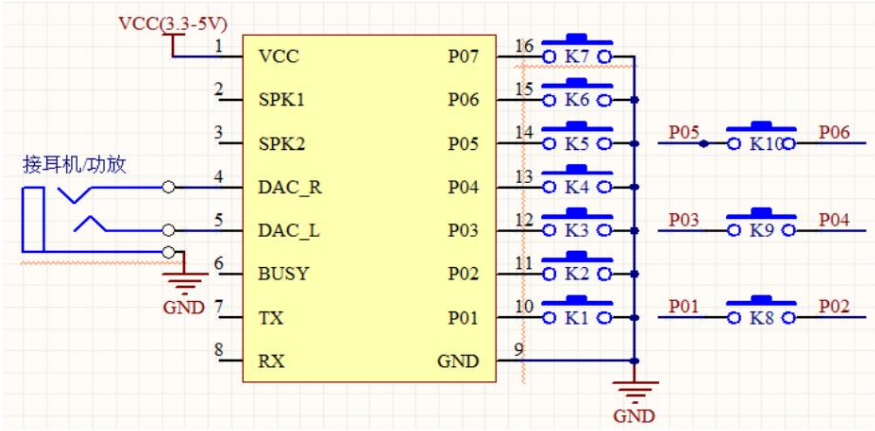
Speaker or connected to amplifier or earphone, plug in USB (can be used as power supply), short-circuit the trigger port to ground to trigger, and then change the sound when there is a sound

2. Module wiring diagram

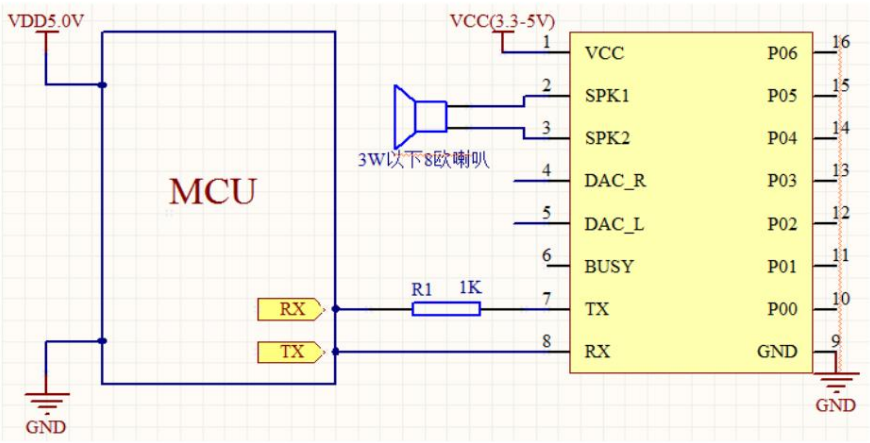
Button Mode - Connect to Small Speaker



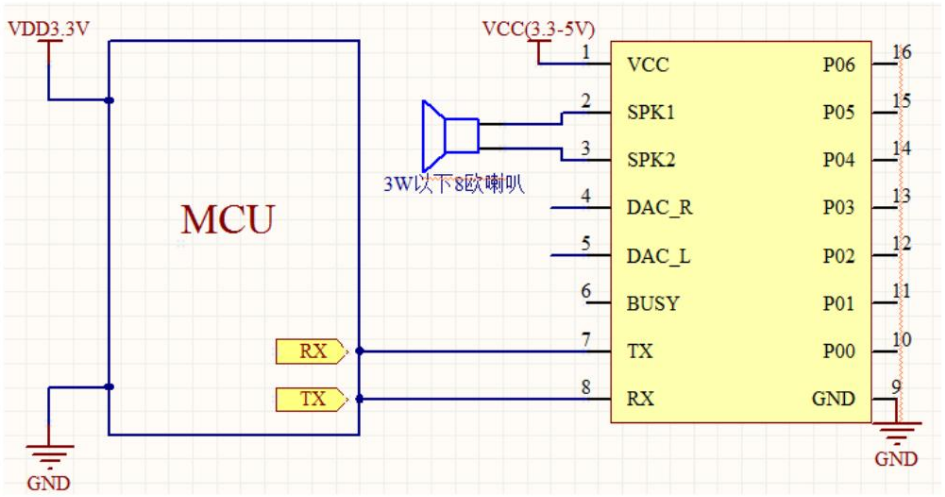
Button mode - connected to amplifier or headphones



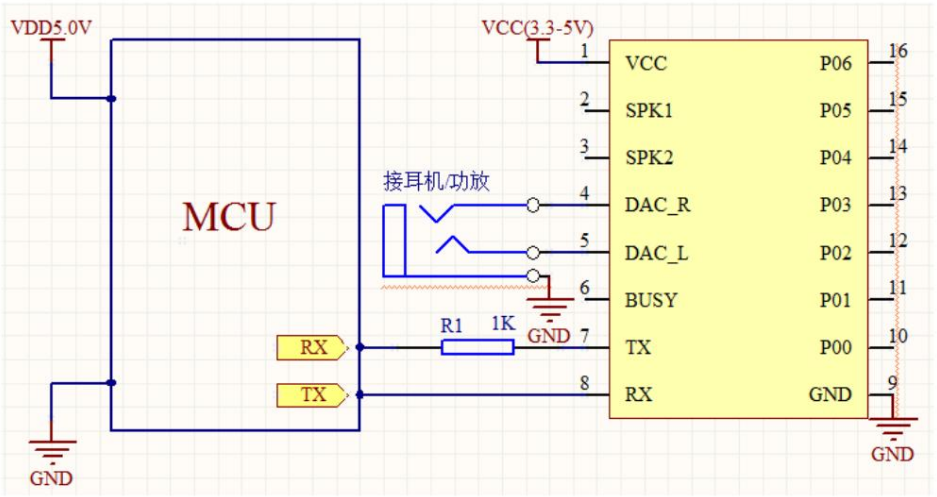
MCU 5V - connected to a small speaker



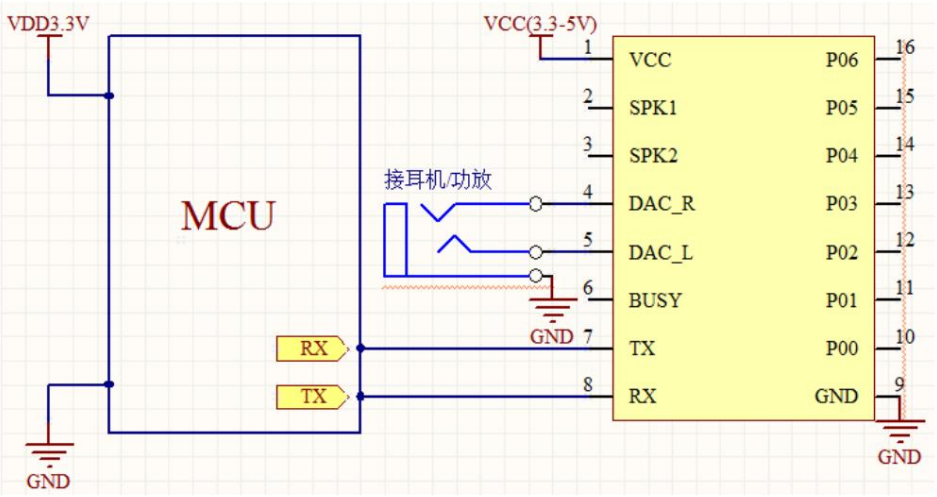
MCU 3.3V - connected to small speaker



MCU 5V - headphone or power amplifier



MCU 5V - headphone or power amplifier



Serial debugging assistant for testing	Sent command [with checksum]	Sent command [without parity]	Remark
[next song]	7E FF 06 01 00 00 00 FE FA EF	7E FF 06 01 00 00 00 EF	
[previous song]	7E FF 06 02 00 00 00 FE F9 EF	7E FF 06 02 00 00 00 EF	
[specified track]	7E FF 06 03 00 00 01 FE F7 EF	7E FF 06 03 00 00 01 EF Specifies the first play of the root directory	
	7E FF 06 03 00 00 02 FE F6 EF	7E FF 06 03 00 00 02 EF Specifies the second song in the root directory	
	7E FF 06 03 00 00 0A FE EE EF	7E FF 06 03 00 00 0A EF specifies the tenth song of the root directory	
volume up	7E FF 06 04 00 00 00 FE F7 EF	7E FF 06 04 00 00 00 EF	
volume down	7E FF 06 05 00 00 00 FE F6 EF	7E FF 06 05 00 00 00 EF	
[Specify volume]	7E FF 06 06 00 00 1E FE D7 EF	7E FF 06 06 00 00 1E EF Specifies the volume to be level 30	
[Specify EQ]	7E FF 06 07 00 00 01 FE F3 EF	7E FF 06 07 00 00 01 EF Reserved	
[loop track]	7E FF 06 08 00 00 01 FE F2 EF	7E FF 06 08 00 00 01 EF Play the first song in a loop	
	7E FF 06 08 00 00 02 FE F1 EF	7E FF 06 08 00 00 02 EF Loop the second song	
	7E FF 06 08 00 00 0A FE E9 EF	7E FF 06 08 00 00 0A EF Play the tenth song in a loop	
[Enter sleep mode]	7E FF 06 0A 00 00 00 FE F1 EF	7E FF 06 0A 00 00 00 EF	
[wake sleep]	7E FF 06 0B 00 00 00 FE F0 EF	7E FF 06 0B 00 00 00 EF	
[chip reset]	7E FF 06 0C 00 00 00 FE EF EF	7E FF 06 0C 00 00 00 EF	
[play]	7E FF 06 0D 00 00 00 FE EE EF	7E FF 06 0D 00 00 00 EF	
[pause]	7E FF 06 0E 00 00 00 FE ED EF	7E FF 06 0E 00 00 00 EF	
[specify folder file name]	7E FF 06 0F 00 01 01 FE EA EF	7E FF 06 0F 00 01 01 EF Folder "01", track "001"	
	7E FF 06 0F 00 01 02 FE E9 EF	7E FF 06 0F 00 01 02 EF Folder "01", track "002"	
Stop play	7E FF 06 16 00 00 00 FE E5 EF	7E FF 06 16 00 00 00 EF Stop software decoding	
Specified folder loop playback 7E FF 06 17 00 02 00 FE E2 EF		7E FF 06 17 00 02 00 EF Specify 02 folder for loop playback	
	7E FF 06 17 00 01 00 FE E3 EF	7E FF 06 17 00 01 00 EF Specify 01 folder for loop playback	
Single loop playback	7E FF 06 19 00 00 00 FE E2 EF	7E FF 06 19 00 00 00 EF Single loop playback on	
	7E FF 06 19 00 00 01 FE E1 EF	7E FF 06 19 00 00 01 EF Single loop play off	
Play with volume	7E FF 06 22 00 1E 01 FE BA EF	7E FF 06 22 00 1E 01 EF 30 level volume play the first song	
	7E FF 06 22 00 0F 01 FE C9 EF	7E FF 06 22 00 0F 01 EF 15-level volume play the first song	
	7E FF 06 22 00 0F 02 FE C8 EF	7E FF 06 22 00 0F 02 EF 15-level volume play the second song	

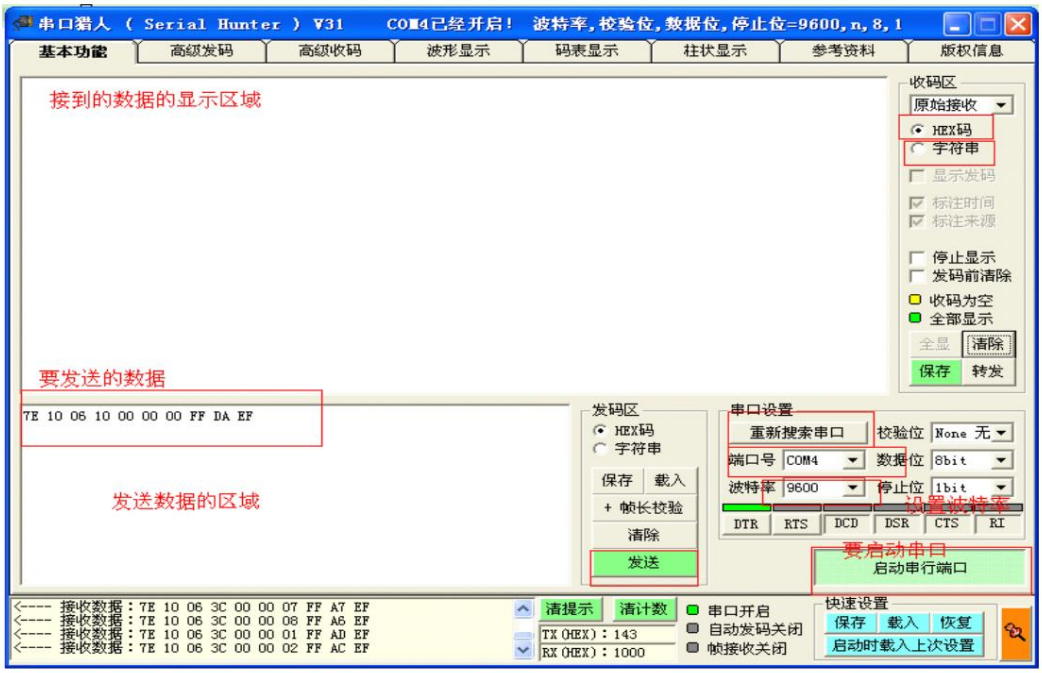
query parameter directive

Serial debugging assistant for testing	Sent command [with checksum]	Sent command [without parity]	Remark

Check online equipment	7E FF 06 3F 00 00 00 FE BC EF 7E FF 06 3F 00 00 00 EF		
query current status	7E FF 06 42 00 00 00 FE B9 EF 7E FF 06 42 00 00 00 EF		
[query volume]	7E FF 06 43 00 00 00 FE B8 EF 7E FF 06 43 00 00 00 EF		
[Query current EQ]	7E FF 06 44 00 00 00 FE B7 EF 7E FF 06 44 00 00 00 EF		
FLASH total number of files	7E FF 06 49 00 00 00 FE B2 EF 7E FF 06 49 00 00 00 EF		
FLASH Current folder track pointer 7E FF	06 4D 00 00 00 FE AE EF 7E FF 06 4D 00 00 00 EF		
Query the total number of tracks in the specified folder 7E FF 06 4E 00 00 01 FE AC EF 7E FF 06 4E 00 01 00 EF			

3. Test method

1. Operation of serial port software



(1), first install the "serial port hunter" software in the data, open the software, first search for the serial port, find After specifying the port, specify the "baud rate", the default baud rate of our module is 9600, and finally "start , the serial port", so the software is configured. There are two concepts here

It needs to be clear that the first is "HEX code", we default to this, this is used to display

data. So you must set here the second is "string", this is used to display the print

Characters, we don't use them here.

(3) After the software configuration is OK, copy the required commands to the sending area. specific reference Please refer to the data sheet of the module

(4) If there is no test instruction in the data sheet of the module, please calculate it by yourself, especially if you need it Note that if the two bytes of the checksum are calculated incorrectly, the module will not accept the instruction of.

The calculation method of the check code:

```
/*********************************************************************************************************
  - 功能描述：  求和校验
  - 隶属模块：
  - 参数说明：
  - 返回说明：
  - 注：        和校验的思路如下
                发送的指令，去掉起始和结束。将中间的6个字节进行累加，最后取反码
                接收端就将接收到的一帧数据，去掉起始和结束。将中间的数据累加，再加上接收到的校验
                字节。刚好为0.这样就代表接收到的数据完全正确。
*********************************************************************************************************/
void DoSum( INT8U *Str, INT8U len)
{
    INT16U xorsum = 0;
    INT8U i;

    for(i=0; i<len; i++)
    {
        xorsum = xorsum + Str[i];
    }
    xorsum      = 0 -xorsum;
    *(Str+i)    = (INT8U)(xorsum >>8);
    *(Str+i+1)  = (INT8U)(xorsum & 0x00ff);
}
```

0 = 24 + x analogy 0000 0000 (0) = 0010 0100 (24) + 1101 1011 (DB+1)

Appendix 1: Comparison Table of SPI-FLASH Capacity and Audio Length

Attached table 1-1 MP3-FLASH-16p module FLASH capacity and audio time length exchange table: (unit: S)

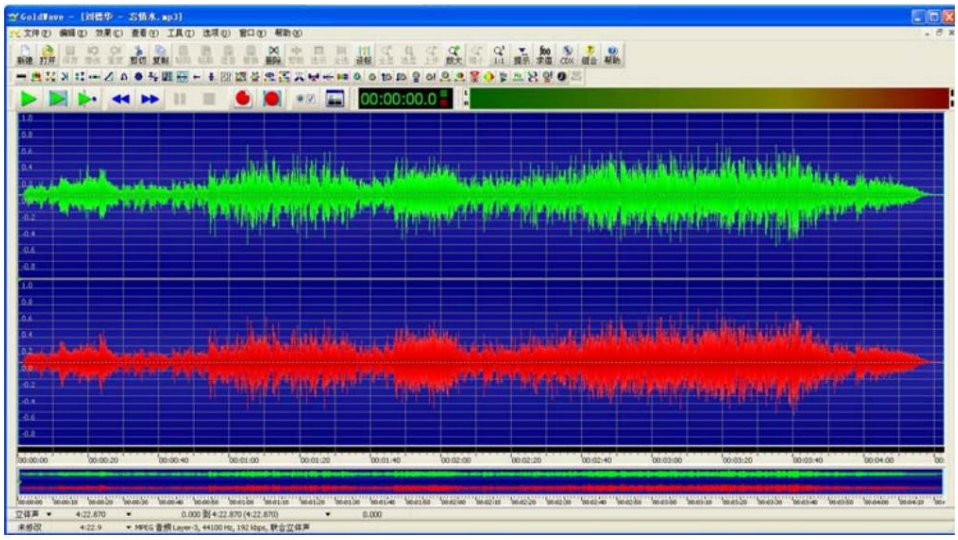
capacity / code rate	4MBit	8MBit	16MBit	32MBit	64MBit	
16Kbps	252	505	1011	2022	4045	
24Kbps	163	327	654	1309	2618	
32Kbps	113	226	453	906	1812	
64Kbps	59	119	239	477	955	
96Kbps	41	81	162	325	651	
128Kbps	31	61	123	246	493	
160Kbps	twenty four	49	97	194	389	
192Kbps	20	40	81	161	323	
256Kbps	15	30	60	120	241	
320Kbps	11	twenty three	47	95	191	

Note: The MP3 file size depends on the bit rate, not the sample rate. It is recommended to use 16Kbps ~ 64Kbps for voice broadcast, and music playback is recommended.

It is recommended to use 32Kbps to 96Kbps.

3. Conversion method of code rate

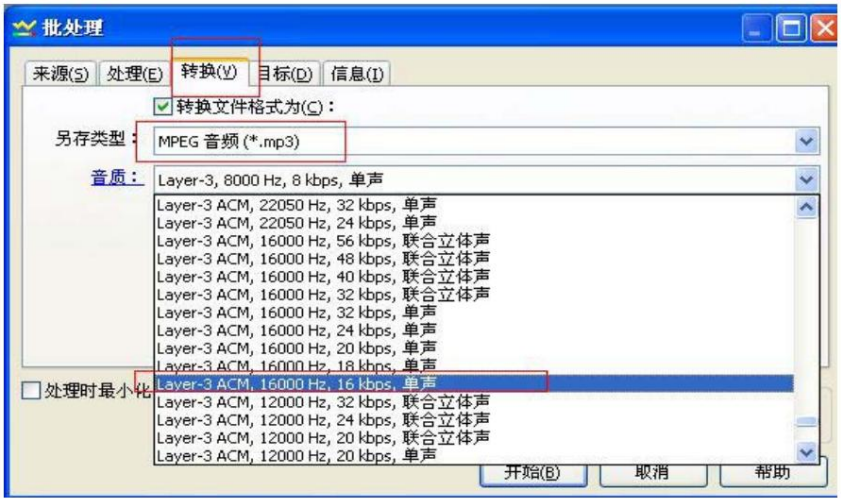
1. Competing for the small capacity and stability of SPIFLASH, we developed the MP3-FLASH-16P module. Update the voice directly through the Microusb line of the mobile phone, but for common MP3 files, most of them are about 4M bytes, using SPIFLASH, Space seems to be exhausted. But we generally use it as a voice broadcast and prompt occasion, and we don't need a high sampling rate at all.



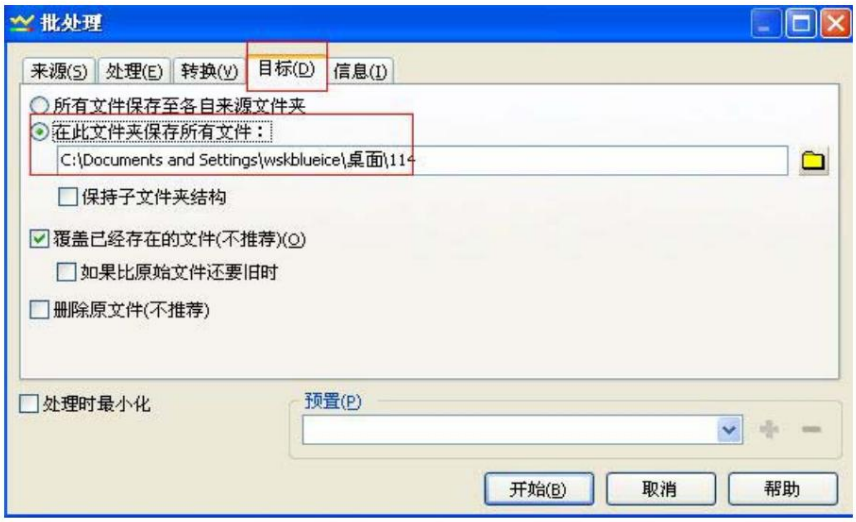
From the lower left corner of the picture above, we can see that the sampling rate of "world first. MP3" is as high as 44100HZ. bit rate 256KBS. This parameter shows that the sound quality of the current song is quite good, so it takes up 4.5M of space. 2. But in reality, we don't need such high sound quality at all, so we can compress it at this time. As follows: Use the software "GoldWave".

Click Batch,

add files



Select "Convert", set the sampling rate to 16000KHZ, and the bit rate to 16KBS.

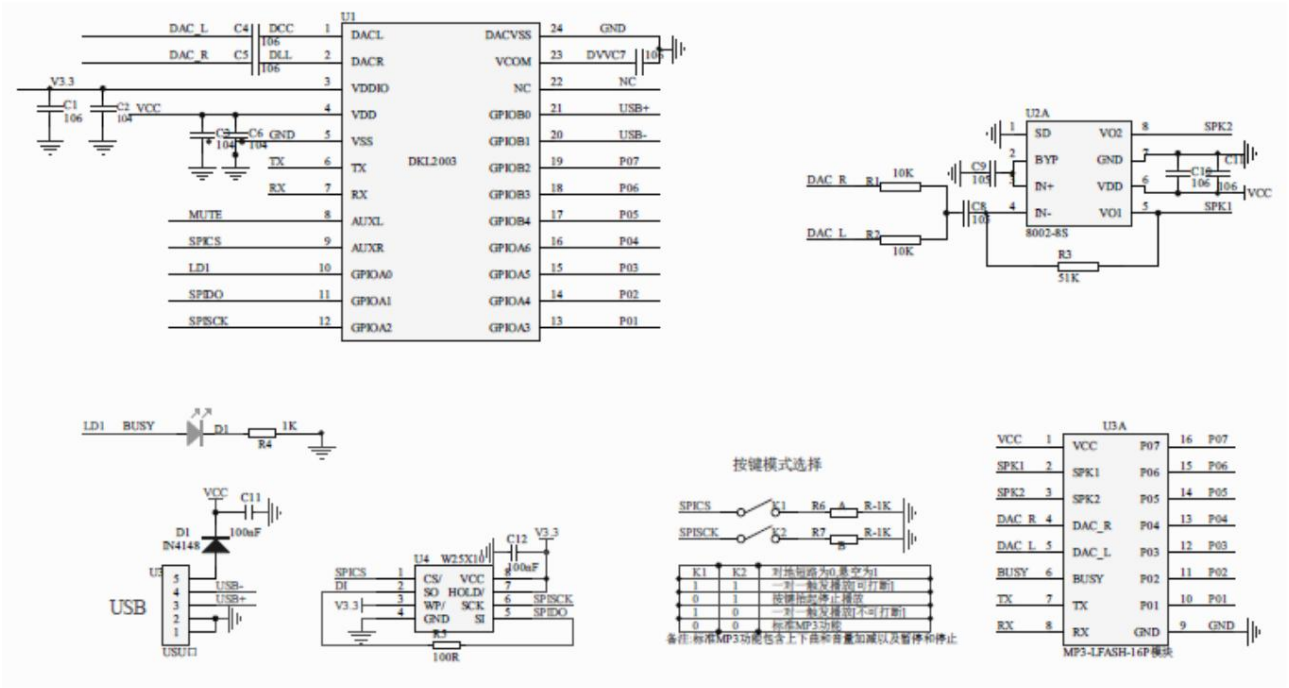


Then specify the path to save the file after conversion.

After compression, the 4.5M file becomes 507K. The steps are like this

Remarks: 1. If it is a wav file, you can also use this software to convert it to MP3. 2. For the effect after conversion, the user can directly audition and listen to the effect on the computer. The effect played on the computer is the same as the effect played by our module. 3. If you feel that the sound quality is not good, you can appropriately increase the two parameters of sampling rate and bit rate. You can try it yourself 4. If you need to modify the volume of the audio source and cut the audio source, you can use this software

Fourth, the schematic diagram of the module



Fourth, the program example

Program example: serial port specified playback

```
/**************************************************** ****************************

  - Realization function: realize the chip is powered on and specify to play the first song and the second song respectively, the basic program is for users to test
  - Date: 2013-05-06 - Operating
  Environment: STC Crystal Oscillator: 11.0592M Baud Rate: 9600 -
  Remarks: Debug OK on Puzhong Technology's 51 development board ---
 STC89C516RD+ 1. The test program must be a module or chip There are devices online in the
program, such as U disk, TF card, FLASH ********************************** ********************************************/
#include "REG52.h"

#define COMM_BAUD_RATE 9600 // Serial port baud rate
#define OSC_FREQ                    11059200 //Run crystal oscillator: 11.05926MHZ
static INT8U Send_buf[10] = {0} ;

void Delay_Ms(INT32U z) {

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

/**************************************************** **************************** - Function description: Serial port 1
  initialization
  - Note: Set to 9600 baud rate
**************************************************** ****************************/
void Serial_init(void) {

      TMOD = 0x20; //                          // set T1 as baud rate generator
      SCON = 0x50;    0101,0000 8 data bits, no parity
      PCON = 0x00;                            //PCON=0;
      TH1=256-(OSC_FREQ/COMM_BAUD_RATE/32/12);//Set to 9600 baud rate
      TL1=256-(OSC_FREQ/COMM_BAUD_RATE/32/12);
       TR1 = 1; = 1; =                    //Timer 1 is on
       REN          1;                    // Serial port 1 receive enable
       ES                                 // Serial port 1 interrupt enable

} void Uart_PutByte(INT8U ch) {

       SBUF = ch;
      while(!TI){;}
       TI = 0;
}

/**************************************************** ****************************

  - Function description: The serial port sends out commands [including control and query] -
  Parameter description: CMD: indicates the control command, please refer to the command table, and also includes the related commands of the query
                         feedback: Whether a response is required [0: no response is required, 1: a response is required]
                         data: transmitted parameters
**************************************************** ****************************/
```

```
void SendCmd(INT8U len) {

        INT8U i = 0 ;
        Uart_PutByte(0x7E); //Start
        for(i=0; i<len; i++)//data
        {
              Uart_PutByte(Send_buf[i]) ;
        }
        Uart_PutByte(0xEF);//End
}

/********************************************************* ****************************
  - Function description: sum check
  - The idea of sum check is as follows:
        The command to send, remove the start and end. Accumulate the middle 6 bytes, and finally take the inverse code. The receiver will receive
 One frame of data, remove the start and end. Accumulate the intermediate data, plus the received check byte. It is exactly 0. This will replace
The data received by the table is completely correct.
********************************************************* ****************************/ void DoSum( INT8U *Str, INT8U len) {


        INT16U xorsum = 0;
        INT8U i;
        for(i=0; i<len; i++) {

              xorsum = xorsum + Str[i];
        }
       xorsum            = 0 -xorsum;
       *(Str+i) = (INT8U)(xorsum >>8);
       *(Str+i+1) = (INT8U)(xorsum & 0x00ff);
}

void Uart_SendCMD(INT8U CMD ,INT8U feedback {                  , INT16U dat)

        Send_buf[0] = 0xff; // reserved bytes
        Send_buf[1] = 0x06; //length
        Send_buf[2] = CMD; //Control command
        Send_buf[3] = feedback;//Whether feedback is needed
        Send_buf[4] = (INT8U)(dat >> 8);//datah
        Send_buf[5] = (INT8U)(dat); //datal //check
        DoSum(&Send_buf[0],6); //Send
        SendCmd(8);                  this frame of data
}

void main() {

       Serial_init() ;//Initialization of serial port registers
       Uart_SendCMD(0x03 , 0          , 0x01) ;//Play the first song
       Delay_Ms(1000);//The delay is about 6S
       Uart_SendCMD(0x03 ,          0 , 0x02) ;//Play the second song
       Delay_Ms(1000);//The delay is about 6S
       Uart_SendCMD(0x03 ,          0 , 0x04) ;//Play the fourth song
       while(1) ;
}
```