FN-BC10 Triggerable MP3 Sound Module

User's Manual

V2.0

Contents

1. Overviews

1.1. Brief Introduction

FN- BC10 is a high quality MP3 sound module developed by Flyron Technology Co., Ltd. Equipped with an on-board 3W amplifier, it is able to drive a 1-3W speaker directly. This sound module can be controlled by 10 separate buttons hooked up to the 'one-on-one' inputs terminals and by UART R232 serial port working with a MCU. Great audio output, industry-grade design and strong anti-jamming capability make it possible to be used for many different applications.

1.2. Features

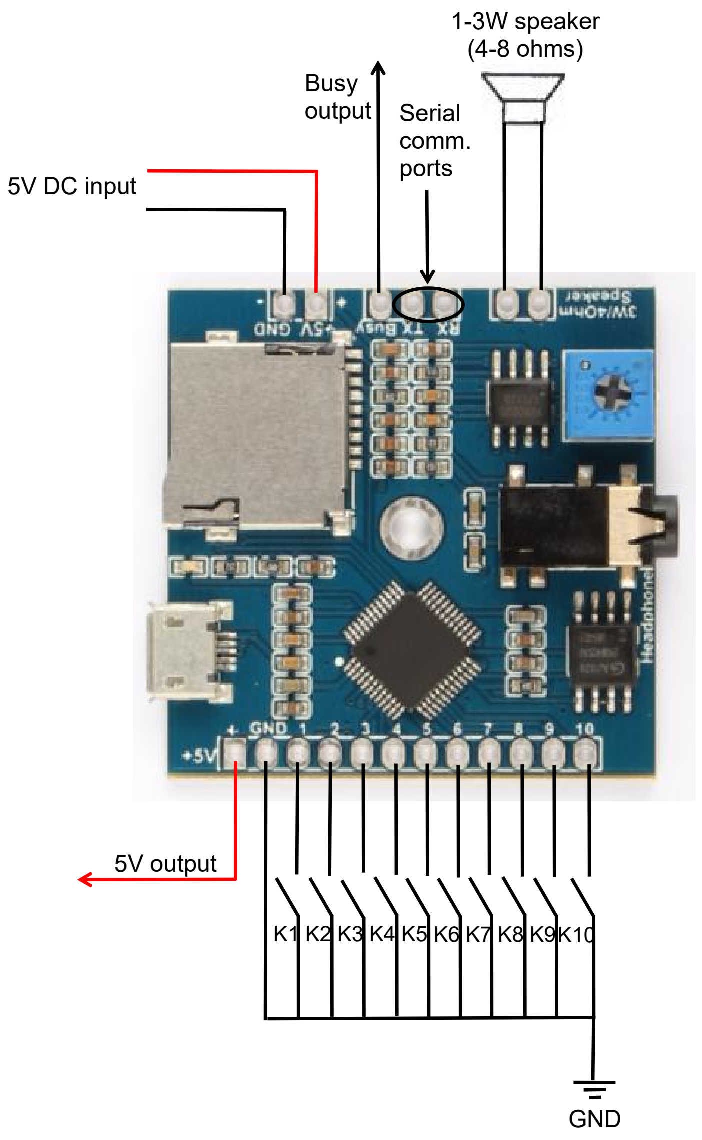
1). Uses a high quality audio decoder, supports MP3 and WAV audio formats.
2). Sampling rates supported: 8/11.025/12/16/22.05/24/32/44.1/48(KHz).
3). 24 bit DAC output and supports dynamic range 90dB and SNR 85dB.
4). Supports one-on-one 10 button trigger control mode and RS232 serial port control mode.
5). In button control mode, it can play back one-on-one 10 sound files, and it also can play multiple sound files in random order on per button.
6). Built-in a 8MBytes(64Mbits) SPI flash memory and supports max. 32GB micro SD card as well.
7). Load audio files to the flash memory/micro SD card directly via the micro USB port connecting with computer. The flash memory/micro SD card works as a USB flash drive on computer.
8). Equipped with a mono 3 watts amplifier that can drive a 1-3 watts(4-8 ohms) speaker directly.
9). Equipped with a 3.5mm audio jack for stereo output that can drive a headphone directly or connect with an external amplifier.
10). Supports detection of indication signal of status change through the Busy output.
11). Adjustable sound volume through the potentiometer.
12). 5V DC power input. Also possible to use 5V power adapter or power bank to supply the power via the micro USB port.
13). 5V power output available that can supply power to an external device.
14). PCB size: 40mmx40mm

1.3. Technical Parameters

1). Working voltage: 5V DC
2). Working current: ≤1000mA
3). Current at playing status: ≤300mA
4). Standby current: ≤10mA
5). Power Consumption: ≤3W
6). On-board flash memory size: 8MBytes(64Mbits)
7). Audio format: MP3 and WAV

2. Connections

K1 to K10 are representing Normally Open (N.O.) manual buttons. TX port and RX port are used to connect with a MCU, through which sending serial commands to control the module. Please refer to the connection example below.



Notes: 1). The Busy port outputs high level at the status of playing while low level at standby.

2). The serial communication ports TX and RX need to be connected with a MCU working at 3.3V TTL level.

3). 5V output can be used to supply power for an external device.

3. Button Control Mode

3.1. Trigger Mode Selection

In button control mode, there are 4 trigger modes available for users to choose according to the actual needs. Any of these 4 trigger modes can be set/acquired through a configuration file named “read.cfg”, which comes from a text file(.txt) originally. Users just need to fill in a digit/parameter that is corresponding to a trigger mode in a new built text file. Save it and rename it “read.cfg”, then put it in the root directory of the flash memory together with the 4 audio files. Please refer to the sheet below about the digits and the associated trigger modes.

Digit in file "read.cfg"	Corresponding Trigger Mode
0	Pulse interruptible playback
1	Level hold loop playback
2	Pulse non-interruptible playback
3	Standard MP3 key mode playback

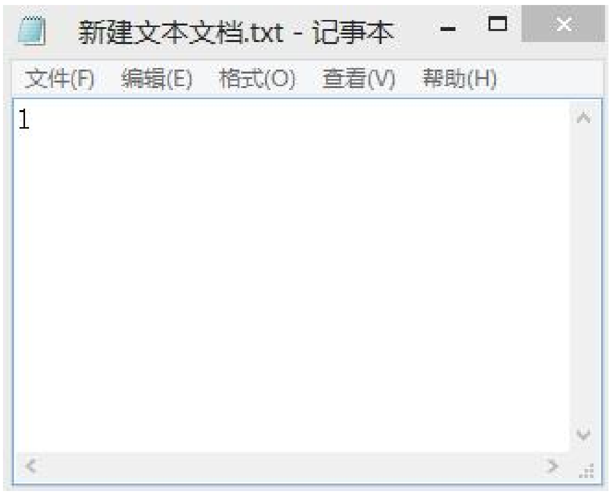
- Pulse interruptible playback: In this mode, a single negative pulse will start playback. It is possible to interrupt the playback by pressing the same button used to activate. Once playback is interrupted, it will automatically restart the audio file immediately. It's also possible to interrupt the play back by pressing any of the other 9 buttons. Once playback is interrupted, it will automatically start the sound that is associated with the button pressed.

- Level hold loop playback: In this mode, the negative pulse must be held/maintained to the sound module trigger for audio file to complete. The audio file will only play back while button, or negative pulse, is held/maintained. Once the button being held, or negative pulse, is removed, the playback will be stopped/ canceled. Once the button is kept holding, when the playback of the audio file is finished, it will start to play it repeatedly(loop playback).

- Pulse non-interruptible playback: In this mode, a single negative pulse will start playback. It's not possible to interrupt the playback by pressing the same button or the other buttons. Once an audio file is triggered, the audio file will not be able to be interrupted/canceled during playback. The playback will only end when the audio file has played its entirety.

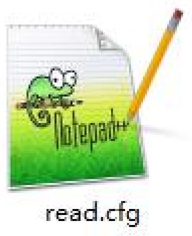
- Standard MP3 key mode playback: In this mode, only the buttons between K1 and K4 are valid. These 4 buttons will be functioned as Previous, Next, Play/pause, and Stop respectively. And the audio files can be placed in folder 01 only.

For example, if the trigger mode of level hold loop playback is needed, firstly build a new text file on the computer, and simply enter the digit "1" as below, and save the file.


新建文本文档.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
1
```

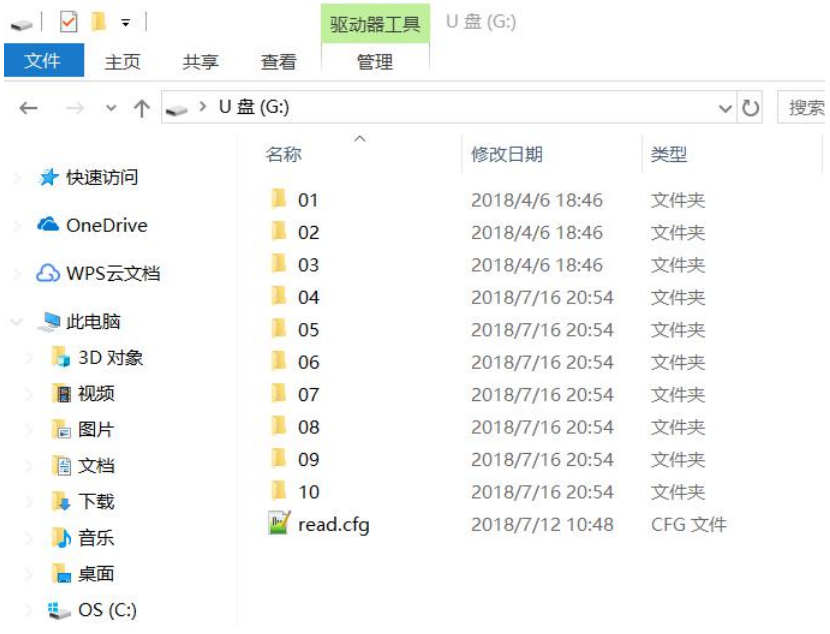
And change the file name "xxx.txt" to "read.cfg" as below, then the configuration file with level hold loop playback mode is made successfully. Please be noted the extension name".txt" of the text file must be changed to ".cfg", otherwise the configuration will not work.



3.2. Audio Files Loading

Before loading audio files onto the module, please check if you need to use a micro SD card. If the on-board flash memory is large enough for your application, then just ignore the micro SD card option. On the contrary, please use a micro SD card instead. Once there is a micro SD card inserted, the module always plays sound files from micro SD card, and when the module is connected to computer through a USB cable, it shows micro SD card there working as a USB flash drive. We suggest users prepare an Android phone purposed USB data transfer cable to connect the module to computer. Please refer to the following steps on how to load/update audio files.

1) Firstly, connect the module to computer and you will see there a removable disk or USB flash drive, and double click to open it.
2) Build 10 folders, and rename them 01, 02, 03...10 respectively. The folder 01 will be associated with K1, the folder 02 will be associated with K2, and so on.
3) Copy one audio file or multiple audio files to each of the 10 folders. And put the prepared configuration file in the root directory together with the 10 folders. Please refer to the image below.



名称	修改日期	类型
01	2018/4/6 18:46	文件夹
02	2018/4/6 18:46	文件夹
03	2018/4/6 18:46	文件夹
04	2018/7/16 20:54	文件夹
05	2018/7/16 20:54	文件夹
06	2018/7/16 20:54	文件夹
07	2018/7/16 20:54	文件夹
08	2018/7/16 20:54	文件夹
09	2018/7/16 20:54	文件夹
10	2018/7/16 20:54	文件夹
read.cfg	2018/7/12 10:48	CFG 文件

4) Safely disconnect the module from computer.
5) Apply power to the module and push any of 10 buttons to play back a sound.

Note: When there is only one audio file in the folder, the module always plays this audio file; and when there are multiple audio files in the folder, the modules play one of the audio files in random after each activation.

4. UART RS232 Serial Control Mode

Serial control mode is provided for users who want to use a MCU to control this module. It's more flexible and is able to realize many more functions to control this module via serial serial commands through the ports TX and RX on the module..

4.1. Naming Rules of Audio Files and Folders

1). Audio files directly stored in the root directory of the storage device(the SPI flash memory) need to be renamed as 0001.mp3/ 0001.wav, 0002.mp3/0002.wav, 0003.mp3/0003.wav, ...
When you copy audio files from computer o SPI flash memory, please refer to the method 3.2.

2). Ordinary folders must be renamed as 01, 02, 03......99, and the audio files must be renamed as 001.mp3/001.wav, 002.mp3/002.wav, 003.mp3/003.wav, ...255.mp3/255.wav. It is also possible to keep the original name when you rename a file. For example, the original name is "Yesterday Once More.mp3", then you can rename it as "001Yesterday Once More.mp3".

Note: In addition, there are two special purposed folders "MP3" and "ADVERT" that can be chosen by users to use or not according to the actual needs. Audio files stored in these two folders need to be renamed as 0001.mp3/0001.wav, 0002.mp3/0002.wav, 0003.mp3/0003.wav, ...3000.mp3/3000.wav.

4.2. Command Format

Supports asynchronous serial communication mode, via which accept serial commands sent by upper PC.
Communication Standard: 9600 bps
Data bits: 1
Checkout: none
Flow Control: none

Format: $S Ver. Length CMD Feedback Para_MSB Param_LSB Check_MSB Check_LSB $O	
$S	Start byte 0x7E
Ver.	Version 0xFF by default
Length	Number of byte from version info to Check_LSB, typically 0x06 (checksum not counted)
CMD	Command Code
Feedback	0x01: Need feedback--send confirmation back to MCU; 0x00: No need feedback
Para_MSB	Most significant byte of parameter
Para_LSB	Least significant byte of parameter
Check_MSB	Most significant byte of checksum
Check_LSB	Least significant byte of checksum
$O	End byte 0xEF

4.3. About Checksum

Regarding to calculating checksum, you can use the following formula to count.
Checksum (2 bytes) = 0xFFFF–(CMD + Feedback + Para_MSB + Para_LSB) + 1

Normally it's okay whether users choose to use checksum or not, our module can receive a serial data with or without checksum, but some of users use a MCU without crystal oscillator, so if in that case we strongly suggest users to add checksum to make sure a stable communication.

4.4. Serial Communication Commands

4.4.1. Control Commands

Command	Function Description	Note
0x01	Play Next	
0x02	Play Previous	
0x03	Specify playback of a track	See 4.7.1 for details
0x04	Increase volume	
0x05	Decrease volume	
0x06	N/A(Reserved)	
0x07	N/A(Reserved)	
0x08	Specify single repeat playback in a folder	See 4.7.2 for details
0x09	N/A(Reserved)	
0x0A	Set Sleep	See 4.7.3 for details
0x0B	Awake from sleep	
0x0C	Reset	
0x0D	Play	
0x0E	Pause	
0x0F	Specify playback a track in a folder	See 4.7.4 for details
0x13	Inter cut an advertisement	See 4.7.5 for details
0x14	N/A(Reserved)	
0x15	Stop playing inter-cut advertisement and go back to play the music interrupted	See 4.7.6 for details
0x16	Stop	
0x17	Specify repeat playback of a folder	See 4.7.7 for details
0x18	Set random playback	See 4.7.8 for details
0x19	Set repeat playback of current track	See 4.7.9 for details
0x1A	Set DAC	See 4.7.10 for details
0x21	Set combination playback(playback of a group)	See 4.7.11 for details

4.4.2.Query Commands

Command	Function Description	Note
0x40	Module returns an error data with this command	
0x41	Module reports a feedback with this command	
0x42	Query current status	See 4.8.1 for details
0x43	Query current volume	
0x49	Query number of tracks	Total number of audio files
0x4D	Query current track	Based on physical order
0x4E	Query number of tracks in a folder	See 4.8.2 for details
0x4F	Query number of folders	See 4.8.3 for details

4.5. Examples of Sending Serial Commands

Commands Description	Serial Commands [with checksum]	Serial Commands [without checksum]	Note
Play Next	7E FF 06 01 00 00 00 FE FA EF	7E FF 06 01 00 00 00 EF	
Play Previous	7E FF 06 02 00 00 00 FE F9 EF	7E FF 06 02 00 00 00 EF	
Specify playback of a track in the root directory	7E FF 06 03 00 00 01 FE F7 EF	7E FF 06 03 00 00 01 EF	Specify playback of the 1st track
	7E FF 06 03 00 00 02 FE F6 EF	7E FF 06 03 00 00 02 EF	Specify playback of the 2nd track
	7E FF 06 03 00 00 0A FE EE EF	7E FF 06 03 00 00 0A EF	Specify playback of the 10th track
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 single repeat playback in a folder	7E FF 06 08 00 01 01 FE F1 EF	7E FF 06 08 00 01 01 EF	Loop playback of track 001 in folder 01
	7E FF 06 08 00 02 01 FE F1 EF	7E FF 06 08 00 02 01 EF	Loop playback of track 001 in folder 02
Set sleep mode	7E FF 06 0A 00 00 00 FE F1 EF	7E FF 06 0A 00 00 00 EF	
Awake from sleep	7E FF 06 0B 00 00 00 FE F0 EF	7E FF 06 0B 00 00 00 EF	
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 playback of a track in a folder	7E FF 06 0F 00 01 01 FE EA EF	7E FF 06 0F 00 01 01 EF	Specify track "001" in the folder “01”
	7E FF 06 0F 00 01 02 FE E9 EF	7E FF 06 0F 00 01 02 EF	Specify track "002" in the folder “01”
Inter cut an advertisement	7E FF 06 13 00 00 01 FE E7 EF	7E FF 06 13 00 00 01 EF	Inter cut track "0001"in the folder “ADVERT”
	7E FF 06 13 00 00 02 FE E6 EF	7E FF 06 13 00 00 02 EF	Inter cut track "0002"in the folder “ADVERT”
	7E FF 06 13 00 00 FF FD E9 EF	7E FF 06 13 00 00 FF EF	Inter cut track "0255"in the folder “ADVERT”
Stop playing inter-cut ad	7E FF 06 15 00 00 00 FE E6 EF	7E FF 06 15 00 00 00 EF	Go back and continue to play the music interrupted
Stop playback	7E FF 06 16 00 00 00 FE E5 EF	7E FF 06 16 00 00 00 EF	Stop software decoding
Specify repeat playback of a folder	7E FF 06 17 00 00 02 FE E2 EF	7E FF 06 17 00 00 02 EF	Specify repeat playback of the folder “02”
	7E FF 06 17 00 00 01 FE E3 EF	7E FF 06 17 00 00 01 EF	Specify repeat playback of the folder “01”
Set random playback	7E FF 06 18 00 00 00 FE E3 EF	7E FF 06 18 00 00 00 EF	

Set single repeat playback	7E FF 06 19 00 00 00 FE E2 EF	7E FF 06 19 00 00 00 EF	Turn on single repeat playback
	7E FF 06 19 00 00 01 FE E1 EF	7E FF 06 19 00 00 01 EF	Turn off single repeat playback
Set DAC	7E FF 06 1A 00 00 00 FE E1 EF	7E FF 06 1A 00 00 00 EF	Turn on DAC
	7E FF 06 1A 00 00 01 FE E0 EF	7E FF 06 1A 00 00 01 EF	Turn off DAC
Query current status	7E FF 06 42 00 00 00 FE B9 EF	7E FF 06 42 00 00 00 EF	
Query current volume	7E FF 06 43 00 00 00 FE B8 EF	7E FF 06 43 00 00 00 EF	
Query number of tracks	7E FF 06 49 00 00 00 FE B2 EF	7E FF 06 49 00 00 00 EF	
Query number of tracks in a folder	7E FF 06 4E 00 00 01 FE AC EF	7E FF 06 4E 00 00 01 EF	Query number of tracks in the folder “01”.
	7E FF 06 4E 00 00 0B FE A2 EF	7E FF 06 4E 00 00 0B EF	Query number of tracks in the folder “11”.
Query number of folders	7E FF 06 4F 00 00 00 FE AC EF	7E FF 06 4F 00 00 00 EF	

4.6. Returned Data from Module

4.6.1 Returned data after the module is powered on

1). After the module is powered on, normally it needs about no more than 500ms to 1500ms(depending on the actual track quantities in the storage device) initialization time. Once the initialization is done, the module returns a data to MCU. If it does not return a data after more than the initialization time, it means there is an error for initialization, and please check the hardware connections.
2). The returned data from module after initialization means the current effective storage device/online equipment. For example, the module returns 7E FF 06 3F 00 00 08 xx xx EF. 0x3F is the returned command by module, and 0x08 represents the SPI flash is effective/online.
3). MCU can not send commands to control the module until the initialization of the module is done and a data is returned, otherwise the commands sent by MCU will be ignored and also this will effect initializing of the module.

4.6.2 Returned data after a track is finished playing

Track Played	Returned Data
1st track in folder 01 is finished playing	7E FF 06 3E 00 01 01 xx xx EF
2nd track on folder 02 is finished playing	7E FF 06 3E 00 02 02 xx xx EF

1). There is a returned data after a track is finished playing. For example, the returned data is 7E FF 06 3E 00 01 01 xx xx EF. 0x3E represents SPI flash memory. 0x01 and 0x01 represents the 1st track in folder 01.
2). Because all of the files(tracks) in the root of the flash memory are read in physical sequence, the returned data still follow the physical sequence, which needs to be noted.

4.6.3 Returned data of feedback from module

Module returns ACK	7E FF 06 41 00 00 00 xx xx EF

1). In order to enhance stability between data communication, the function of a feedback from module is added. Once there is a feedback to MCU from the module, it means the module has successfully received the command that MCU sent out. 0x41 is the returned command by module.
2). Users are free to choose this feedback or not. It’s also fine not to choose this function.

4.6.4 Returned data of errors

Returned Data of Errors	Meaning Description
7E FF 06 40 00 00 01 xx xx EF	Module busy(this info is returned when the initialization is not done)

7E FF 06 40 00 00 02 xx xx EF	Currently sleep mode(supports only specified device in sleep mode)
7E FF 06 40 00 00 03 xx xx EF	Serial receiving error(a frame has not been received completely yet)
7E FF 06 40 00 00 04 xx xx EF	Checksum incorrect
7E FF 06 40 00 00 05 xx xx EF	Specified track is out of current track scope
7E FF 06 40 00 00 06 xx xx EF	Specified track is not found
7E FF 06 40 00 00 07 xx xx EF	Inter-cut error(an inter-cut operation only can be done when a track is being played)
7E FF 06 40 00 00 09 xx xx EF	Initialization error SPI flash memory
7E FF 06 40 00 00 0A xx xx EF	Entered into sleep mode

4.7. Detailed Annotation of Control Commands

4.7.1.Specify playback of a track in the root of SPI flash memory

The available selective tracks is from 0001.mp3/wav to 3000.mp3/wav in the root of SPI flash memory. Actually it can support more, but if we make it support more, the operation speed will become slow. Usually most of applications do not need to support much more files.

1). For example, select the first song played, and send the command 7E FF 06 03 00 00 01 FF E7 EF

7E --- Start byte

FF --- Version Information

06 --- Data length (checksum not included)

03 --- Actual command(specify playback of a track)

00 --- 0x01: need feedback, 0x00:no need feedback

00 --- Most significant byte of the track(MSB of Parameter)

01 --- Least significant byte of the track(LSB of Parameter)

FF --- Most significant byte of checksum(MSB of checksum)

E7 --- Least significant byte of checksum(LSB of checksum)

EF --- End byte

2). Regarding track selection, if choose the 100th song(track), firstly convert 100 to hexadecimal. It is double-byte by default, i.e. 0x0064. MSB=0x00; LSB=0x64

3). If you choose to play the 1000th song(track), firstly convert 1000 to hexadecimal. It is double-byte, i.e. 0x03E8. MSB=0x03; LSB=0xE8

4).And so on in the same way to the other operations, as in the embedded area hexadecimal is the most convenient operation method.

4.7.2. Specify single repeat playback in a folder

Start to repeatedly play the track 001 in folder 01	7E FF 06 08 00 01 01 xx xx EF
Start to repeatedly play the track 002 in folder 01	7E FF 06 08 00 01 02 xx xx EF

1). We added this control command 0x08, to meet the needs that some users need single repeat playback.

2). During single repeat playback, you can still normally execute the operations Play/Pause, Previous, Next, Volume+/-, and so on. You can specify single track playback or make it sleep to turn off single repeat playback status.

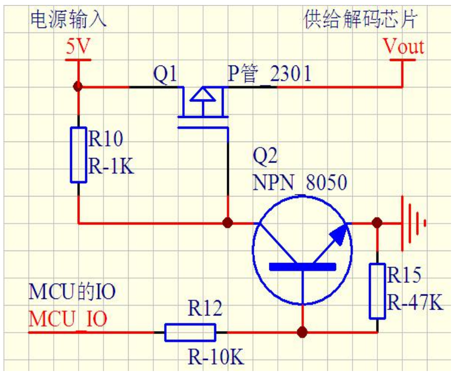
4.7.3 Set sleep mode, awake from sleep and reset

Set sleep mode	7E FF 06 0A 00 00 00 FE F1 EF
Awake from sleep	7E FF 06 0B 00 00 00 FE F0 EF
Reset	7E FF 06 0C 00 00 00 FE EF EF

1). After set the module enter into sleep mode, there is also another way other than sending the command to awake the module that re-power up the module.

2). Regarding the reset, it's a soft reset, and the reset time is 5-8 seconds. It is allowed to send the reset command under any status.

Note: When the module enters into the sleep mode, the standby power consumption is about 10mA. If users are very strict to the power consumption, you can use a MOS and a transistor to control power supply of the module. It is possible to cut off the power supply completely when standby is not necessary. Please refer to the schematic as below.



4.7.4. Specify playback of a track in a folder

Specify playback of track 001 in folder 01	7E FF 06 0F 00 01 01 xx xx EF
Specify playback of track 100 in folder 11	7E FF 06 0F 00 0B 64 xx xx EF
Specify playback of track 255 in folder 99	7E FF 06 0F 00 63 FF xx xx EF

1). The default folders are named as "01", "11", "99" in this way. In order to be with a better system stability, it is made to support maximum 99 folders and maximum 255 tracks in each folder..

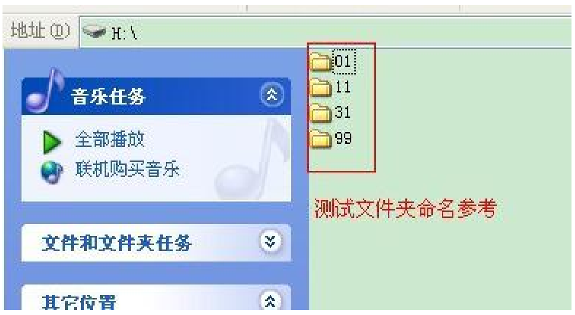
2). For example, if specify to play "100.mp3" in the folder "01", send the command 7E FF 06 0F 00 01 64 xx xx EF

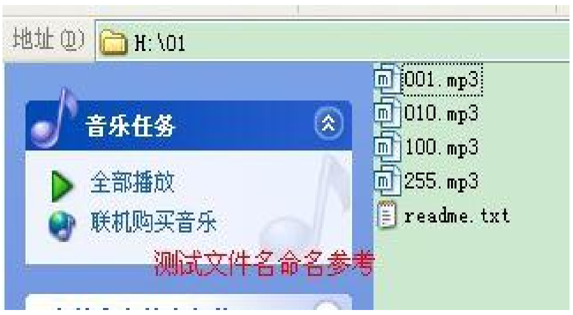
MSB: represents the name of the folder, maximum supports 99 folders from 01 - 99.

LSB: represents the track, maximum supports 255 tracks from 0x01 to 0xFF.

3). You must specify both the folder and the file name to target a track. This feature supports MP3 and WAV audio formats.

4). The following two images illustrates the naming method of folders and files.





4.7.5. Inter cut an advertisement in folder "ADVERT"

Inter cut track "0001"in the folder "ADVERT"	7E FF 06 13 00 00 01 FE E7 EF
Inter cut track "0002"in the folder "ADVERT"	7E FF 06 13 00 00 02 FE E6 EF
Inter cut track "0255"in the folder "ADVERT"	7E FF 06 13 00 00 FF FD E9 EF
Inter cut track "1999"in the folder "ADVERT"	7E FF 06 13 00 07 CF FE 12 EF
Inter cut track "3000"in the folder "ADVERT"	7E FF 06 13 00 0B B8 FE 25 EF

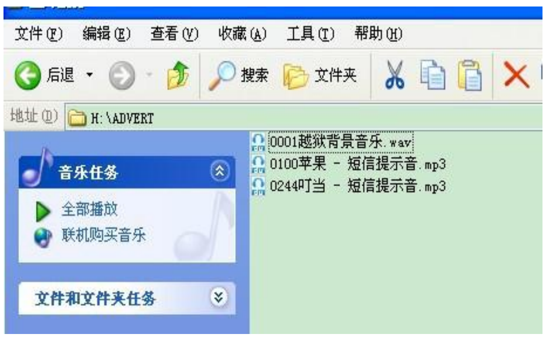
1). This module supports inter-cut advertisements during playback of a track, so that it can meet some special needs for some applications.

2). After sending the command 0x13, the system will save the IDV3 information of the track being played and pause, then it will play the specified inter-cut track(advertisement). When the inter-cut track is finished, the system will go back and continue to play the track that was interrupted until to the end.

3). The setting method is to build a folder named “ADVERT” in the storage device, and put the tracks(ads) you need in the folder, and rename the files as “0001.mp3/wav”, 0002.mp3/wav.

4). If you send an inter-cut command when the module is at Pause status or Stop status, it will not work and there will be an returned error information. In the course of an inter-cut, you can continue to inter cut the other tracks(ads). When the last inter-cut track goes to the end, the system still goes back to the IDV3 position saved at the first time.

5). Audio files stored in this special folder need to be renamed as 0001.mp3/0001.wav, 0002.mp3/0002.wav, 0003.mp3/0003.wav, ...3000.mp3/3000.wav as shown below.



H:\ADVERT

0001越狱背景音乐.wav
0100苹果 - 短信提示音.mp3
0244叮当 - 短信提示音.mp3

4.7.6. Stop

Stop playing advertisement	7E FF 06 15 00 00 00 FE E6 EF
Stop all playback tasks	7E FF 06 16 00 00 00 FE E5 EF

1). During playback of the module, there is two modes to stop. One is to stop playing the inter-cut advertisement, and go back and continue to play the music interrupted, and the other mode is to stop all playback(stop decoding).

2). For example, suppose the module is playing an inter-cut advertisement, and now if send a stop command 0x16, it will stop all playback tasks.

4.7.7. Specify repeat playback of a folder

Specify repeat playback of folder “02”	7E FF 06 17 00 00 02 FE E2 EF
Specify repeat playback of folder “01”	7E FF 06 17 00 00 01 FE E3 EF

1). The folder names must be 01-99, and no more than 99.

2). After sending the command, it repeatedly plays the tracks in the specific folder, and it will not stop until it receives a command to stop.

4.7.8. Set random playback

Random playback of tracks in the whole memory	7E FF 06 18 00 00 00 FE E3 EF

This command is used to randomly play audio files in the SPI flash according to physical sequence and no matter if there is a folder or not. The first audio file that is conducted to be played is the first one copied to the flash memory.

4.7.9. Set repeat playback of current track

Turn on single repeat playback	7E FF 06 19 00 00 00 FE E2 EF
Turn off single repeat playback	7E FF 06 19 00 00 01 FE E1 EF

1). During playback, send the turn-on command, and it will repeatedly play the current track. If the module is at Pause or Stop status, it will not respond to this command.
2). If you need to turn off repeat playback, just send the turn-off command.

4.7.10. Set DAC

Turn on DAC	7E FF 06 1A 00 00 00 FE E1 EF
Turn off DAC(high resistance)	7E FF 06 1A 00 00 01 FE E0 EF

When the module is powered on, DAC is turned on by default. It is not turned off until it is set up by sending the command.

4.7.11. Set combination playback(playback of a group)

1). We added this function to meet some users' special need that when users need to send only one frame data to play multiple tracks one by one without pause. It supports maximum 15 tracks together for combination playback. All of the sound files used for combination playback need to be put in folders(folder 01-folder 99).

2). If MCU sends a frame data as **7E FF 15 21 01 02 01 03 01 04 01 05 01 06 02 01 03 05 04 07 05 09 EF**, see the analysis as below.
Command: 0x21
Number of bytes: 0x15=21 bytes --- **FF 15 21 01 02 01 03 01 04 01 05 01 06 02 01 03 05 04 07 05 09**(two parameters for one track, i.e. the folder number and the track number)
The module will play track 002 in folder 01, track 003 in folder 01, track 004 in folder 01, track 005 in folder 01, track 006 in folder 01, track 001 in folder 02, track 005 in folder 03, track 007 in folder 04, and track 009 in folder 05.

3). During combination playback, it is allowed to Play/Pause and set volume, but not allowed to set Previous and Next. If need to stop, just direct send the stop command. And it is not allowed to play another group of combination during it is working. Users need to send the stop command to stop the current combination playback before start another group of combination playback.

4). If a track specified to be played in combination is not in the folder, it will stop playing at this track position, so please make sure the track specified to play must be available in the folder.

5). If users are very strict to the combination playback, please edit the sound sources with some audio edit software like Adobe Audition or GoldWave to cut off the silence at the beginning and the end of the sound.

6). Due to this frame command data is long, we cut off the byte "Feedback" compared with other commands. Please be noted.

4.8. Detailed Annotation of Main Query Commands

4.8.1 Query current status

Query current status	7E FF 06 42 00 00 00 FE B9 EF

1). There are 4 status(playing, paused playing, stopped playing, and in sleep) that can be queried during the module is decoding. Users can query the current status via sending the command as above(0x42).

2). Interpretation of returned data

Returned Data	Status
7E FF 06 42 00 08 01 xx xx EF	A track in the SPI flash is being played
7E FF 06 42 00 08 02 xx xx EF	A track in the SPI flash is paused playing

7E FF 06 42 00 08 00 xx xx EF	A track in the SPI flash is stopped playing
7E FF 06 42 00 10 00 xx xx EF	Module in sleep

3). MSB and LSB Representations

MSB Representation		**LSB Representation**	
0x08	SPI flash	0x00	Stopped
0x10	Module in sleep mode	0x01	Playing
		0x02	Paused

4.8.2 Query number of tracks in a folder

Query number of tracks in folder 01	7E FF 06 4E 00 00 01 FE AC EF
Query number of tracks in folder 11	7E FF 06 4E 00 00 0B FE A2 EF

If the folder queried is empty without any files, the module will report an error, and the data 7E FF 06 40 00 00 06 FE B5 EF will be returned.

4.8.3 Query number of folders

Query number of folders	7E FF 06 4F 00 00 00 FE AC EF

Users can query the total folder numbers through sending the command above.This just supports to query the folder numbers in the root directory of the device. Not possible to query the sub-folder numbers(Please don’t build any sub-folders in a folder).

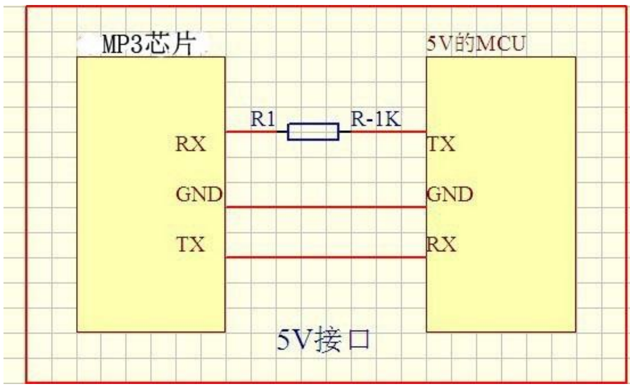
4.9. Example of Serial Program

Code example：specify playback of a track

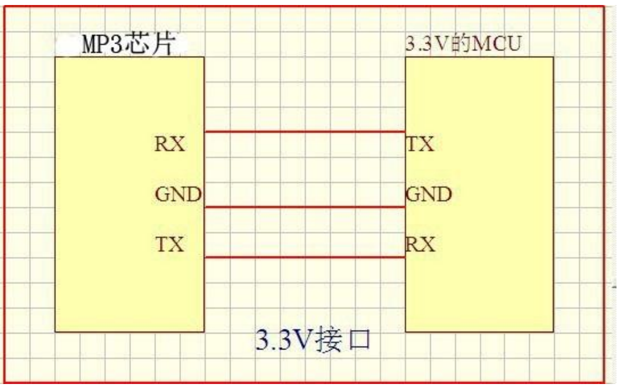
```
/*******************************************************************************
 - 实现功能：实现芯片上电分别指定播放第一曲和第二曲，基本的程序供用户测试
 - 运行环境：STC   晶振：11.0592M     波特率:9600
 - 备注    ：在普中科技的 51 开发板上调试 OK --- STC89C516RD+
1、该测试程序必须是模块或者芯片方案中有设备在线，譬如 U 盘、TF 卡、FLASH
*******************************************************************************/
#include "REG52.h"

#define COMM_BAUD_RATE  9600    //串口波特率
#define OSC_FREQ        11059200    //运行晶振：11.05926MHZ
static INT8U Send_buf[10] = {0} ;

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

/*******************************************************************************
 - 功能描述：  串口 1 初始化
 - 注：          设置为 9600 波特率
*******************************************************************************/
void Serial_init(void)
{
        TMOD = 0x20;            // 设置 T1 为波特率发生器
        SCON = 0x50;            // 0101,0000 8 位数据位, 无奇偶校验
        PCON = 0x00;            //PCON=0;
        TH1=256-(OSC_FREQ/COMM_BAUD_RATE/32/12);//设置为 9600 波特率
        TL1=256-(OSC_FREQ/COMM_BAUD_RATE/32/12);
  TR1    = 1;                         //定时器 1 打开
  REN    = 1;                         //串口 1 接收使能
  ES     = 1;                         //串口 1 中断使能
}
void Uart_PutByte(INT8U ch)
{
  SBUF  = ch;
  while(!TI){;}
  TI = 0;
}

/********************************************************************************
 - 功能描述： 串口向外发送命令[包括控制和查询]
 - 参数说明： CMD:表示控制指令，请查阅指令表，还包括查询的相关指令
           feedback:是否需要应答[0:不需要应答，1:需要应答]
           data:传送的参数
********************************************************************************/
void SendCmd(INT8U len)
{
  INT8U i = 0 ;
  Uart_PutByte(0x7E); //起始
  for(i=0; i<len; i++)//数据
  {
                Uart_PutByte(Send_buf[i]) ;
  }
  Uart_PutByte(0xEF) ;//结束
}
/********************************************************************************
 - 功能描述：求和校验
 - 和校验的思路如下：
  发送的指令，去掉起始和结束。将中间的 6 个字节进行累加，最后取反码。接收端就将接收到的一帧数据，去掉起始和结束。将中间的数据累加，再加上接收到的校验字节。刚好为 0.这样就代表接收到的数据完全正确。
********************************************************************************/
void DoSum( INT8U *Str, INT8U len)
{
  INT16U xorsum = 0;
  INT8U i;
  for(i=0; i<len; i++)
  {
    xorsum  = xorsum + Str[i];
  }
        xorsum     = 0 -xorsum;
        *(Str+i)   = (INT8U)(xorsum >>8);
        *(Str+i+1) = (INT8U)(xorsum & 0x00ff);
}

void Uart_SendCMD(INT8U CMD ,INT8U feedback , INT16U dat)
{
  Send_buf[0] = 0xff;    //保留字节
  Send_buf[1] = 0x06;    //长度
  Send_buf[2] = CMD;     //控制指令
  Send_buf[3] = feedback;//是否需要反馈
```

```
    Send_buf[4] = (INT8U)(dat >> 8);//datah
    Send_buf[5] = (INT8U)(dat);     //datal
    DoSum(&Send_buf[0],6);        //校验
    SendCmd(8);       //发送此帧数据
}

void main()
{
            Serial_init() ;//串口寄存器的初始化设置
    Uart_SendCMD(0x03 , 0 , 0x01) ;//播放第一首
    Delay_Ms(1000) ;//延时大概 6S
    Uart_SendCMD(0x03 , 0 , 0x02) ;//播放第二首
    Delay_Ms(1000) ;//延时大概 6S
    Uart_SendCMD(0x03 , 0 , 0x04) ;//播放第四首
    while(1) ;
}
```

4.10. Connection of UART Serial Port



Connection to a 5V MCU



Connection to a 3.3V MCU

The module uses 3.3V TTL level, so if you use a 5V MCU we suggest you attach a 1K resistor. Please refer to the upper-left diagram.

4.11. About Delay of Serial Programming

1). After the module is powered on, it needs about 500ms to 1500ms (depending on the track quantities in the storage device) to initialize. After that, some data related to initialization returns to MCU. Users can choose to ignore these data.

2). After specifying playback of a device (SD card or SPI flash), it needs 200ms delay before sending the command to execute the relative operation.

3). The module processes a serial data per 10ms, so when MCU continuously sends commands one by one, 20ms delay must be added before sending next command, otherwise the command MCU sends out will not be executed.

4). If specifying playback of a track in a folder, the delay must be longer than 40ms, as it needs time to target a tracking a folder. And even so song as sending the commands related to query a track or a folder, 40ms delay is required.

5. Appendix

Relation table between SPI flash capacity and time duration supported based on different bit rates of MP3 files
unit:seconds

Capacity / Bit rate	4MBits	8MBits	16MBits	32MBits	64MBits
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	24	49	97	194	389
192Kbps	20	40	81	161	323
256Kbps	15	30	60	120	241
320Kbps	11	23	47	95	191

Note:

a). 1Mbyte=4Mbits

b). In order to save more audio files in the SPI flash if in need, we suggest users convert the bit rate of the MP3 files to 16Kbps-32Kbps for voice messages and 32Kbps-96Kbps for music files.

c).We recommend users to use the softwares like COOL EDIT PRO, ADOBE AUDITION, GOLDWAVE or TTPlayer to convert the bit rate of the audio files.