



Voice MCU Workshop V2.4

使用手册

版本 : V1.70 日期 : 2020-03-04

www.holtek.com

目录

1 语音平台使用说明	4
开发平台简介与软体开发	4
特性	4
系统需求及配置	4
系统的组合	5
软体安装	6
工程建立软体操作 (立即上手)	9
打开 Voice MCU Workshop V2.4	9
新建一个工程文档 (Project)	9
打开一个已有的工程	44
硬件电路	45
评估板原理图	45
评估板使用方式	46
平台所支持 flash 型号	48
2 ASM 及 C 库使用说明	49
ASM 调用 Voice library	49
目的说明	49
如何使用	49
所提供函数	51
完整调用 library 的举例	56
C 语言调用 Voice library	72
目的说明	72
如何使用	72
所提供函数	74
具体过程应用举例	79
3 Voice Library 的建立信息及对应仿真器	95
HT66FV130	95
HT66FV140	97
HT66FV150	99
HT66FV160	101
BH67F2262	103
HT45F67	105
HT45F65	107
HT45F3W	109
HT66F4550	111
BA45F5250	113
BA45F5260	115
HT45F23A	117
HT45F24A	119

HT83F02	121
HT86BX0	123
4 Audacity 的快速入门	129
Audacity 的概述	129
Audacity 的处理流程	129
快速上手	130
5 Adobe Audition CS6 简易教程.....	147
简介	147
快速入门	148
对单首音乐的编辑	148
录音功能编辑	152

1 语音平台使用说明

开发平台简介与软体开发

特性

Voice MCU Workshop V2.4 是针对语音系列 MCU 开发而设计的简易开发软体。它将一个工程中的 code 图形化直接进行图形化的逻辑设计，可以帮助开发人员快速地制作样板和完成工程。在语音段较多的工程中 Voice MCU Workshop V2.4 简化开发过程的优势更加明显。最后代码以某种的形式储存于语音系列的 MCU 中而音频文件的压缩编码存储到外部 flash 中。

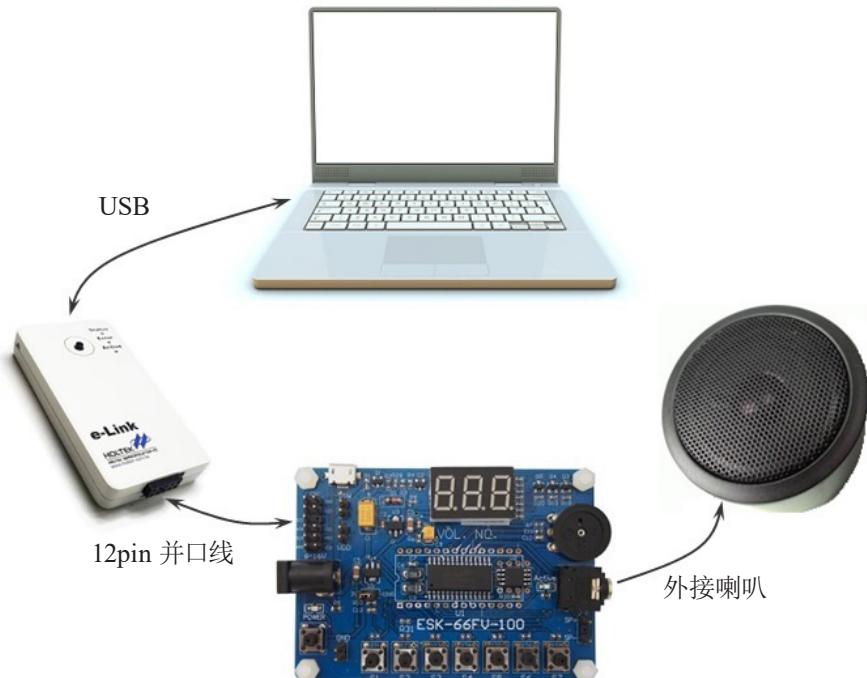
系統需求及配置

Windows 版本	Recommended RAM/ Processor Speed	Minimum RAM/ Processor Speed
Windows 7 (32-bit or 64-bit)	4 GB / 2 GHz	2 GB / 1 GHz
Windows Vista (Home Premium/Business/ Ultimate) (32-bit or 64-bit)	4 GB / 2 GHz	1 GB / 1 GHz
Windows XP (32-bit or 64-bit)	2 GB / 1 GHz	512 MB / 1 GHz

系统的组合

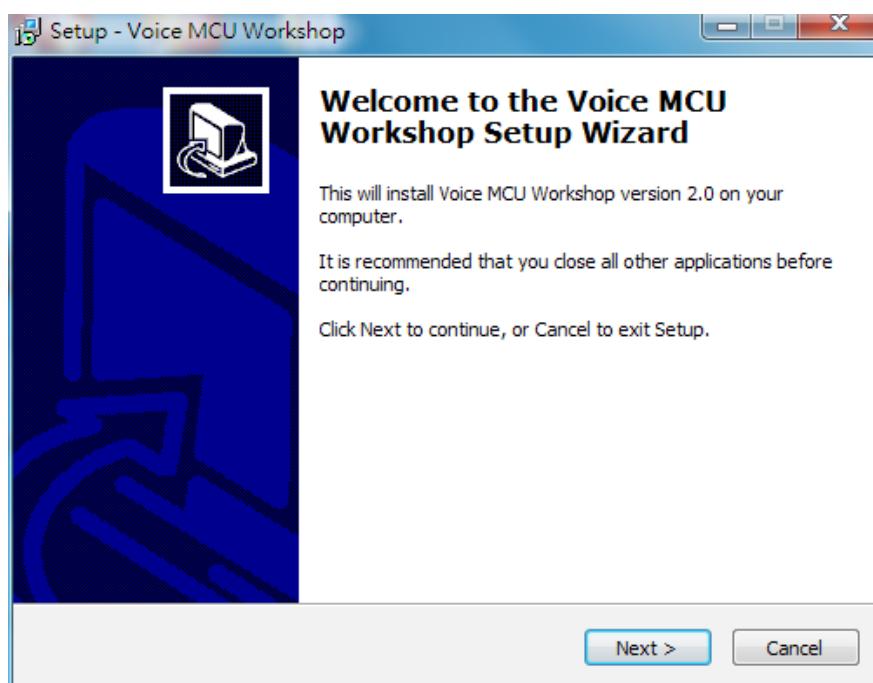
整个系统主要由软体和硬体组成如下所示:

- 软体: Voice MCU Workshop V2.4
- 硬体: ESK-FV160-200 开发板 / ESK-66FV-100 评估板
 - e-Link (需自备)
 - 播音的喇叭 (需自备)

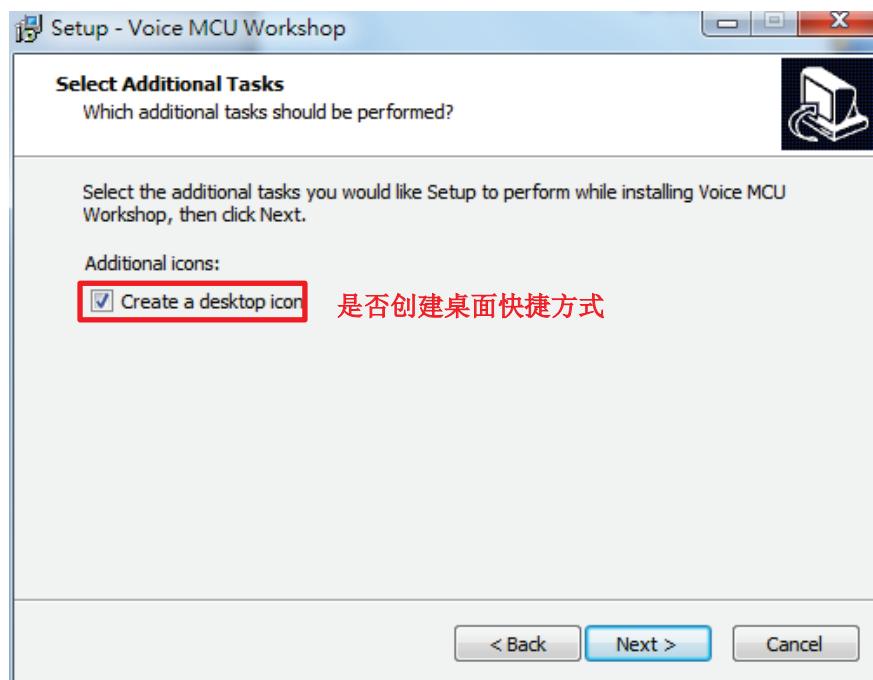


软件安装

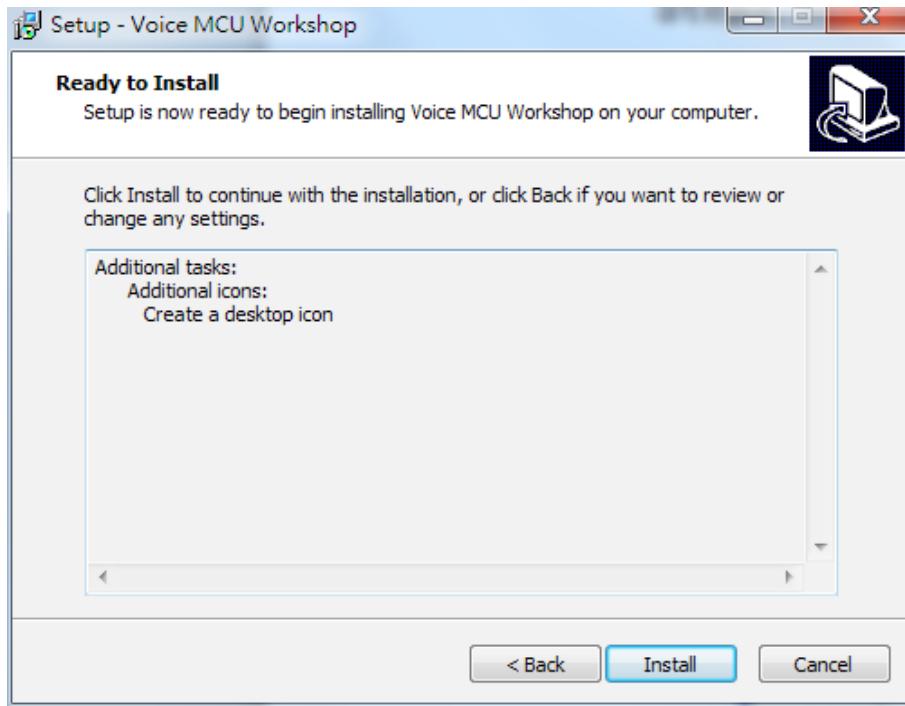
Step1. 双击安装图标 弹出如下图:



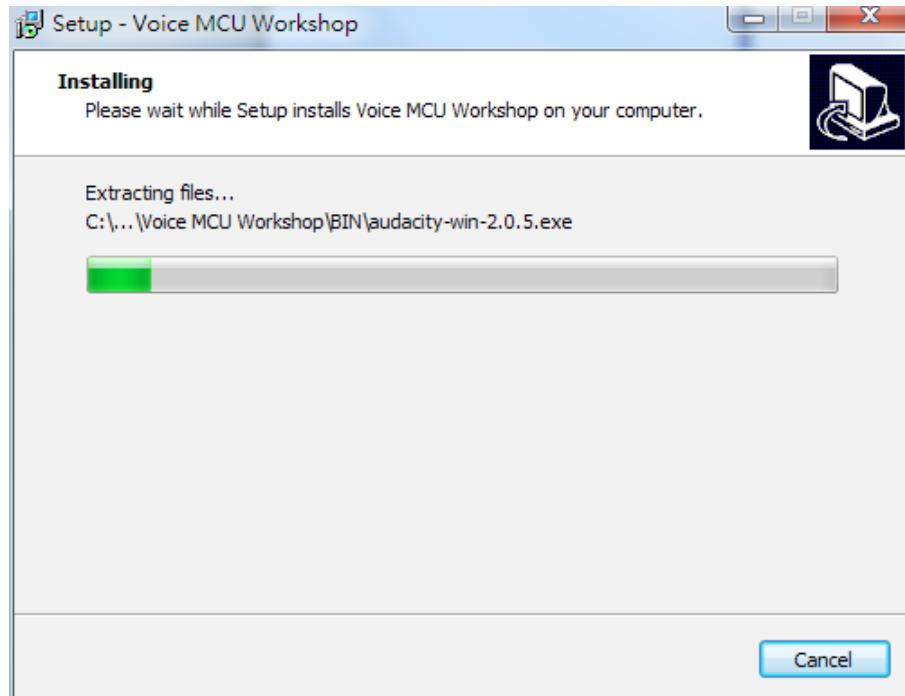
Step2. 点击“Next”，得到下图:



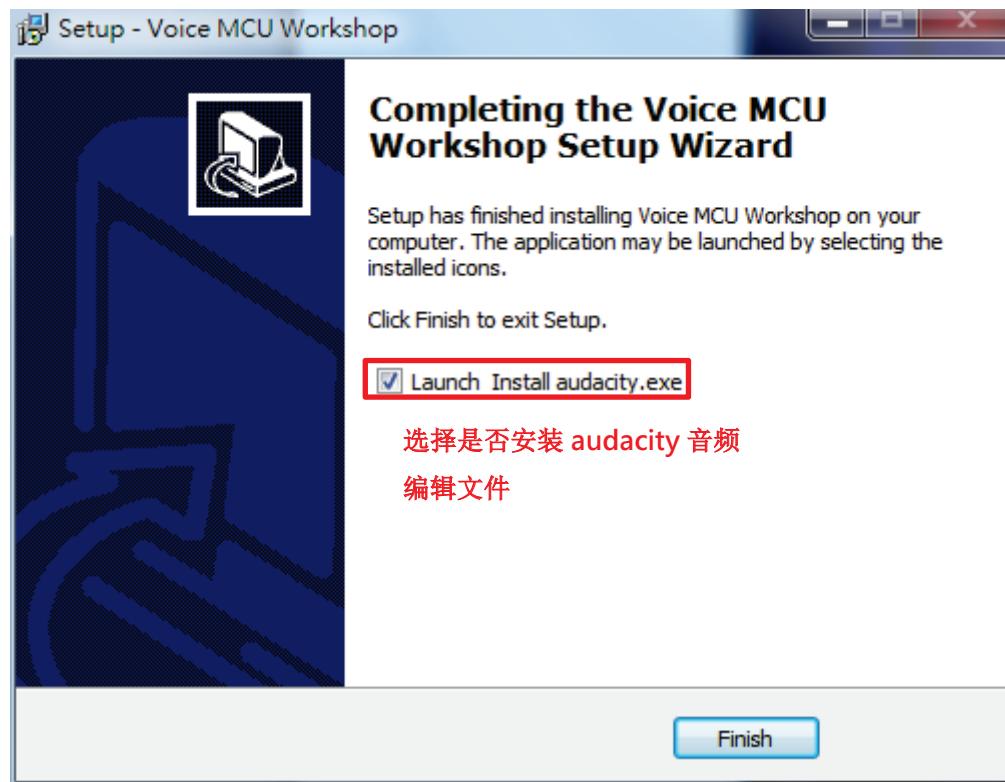
Step3. 点击“Next”，得到下图：



Step4. 选择“Install”进行安装：



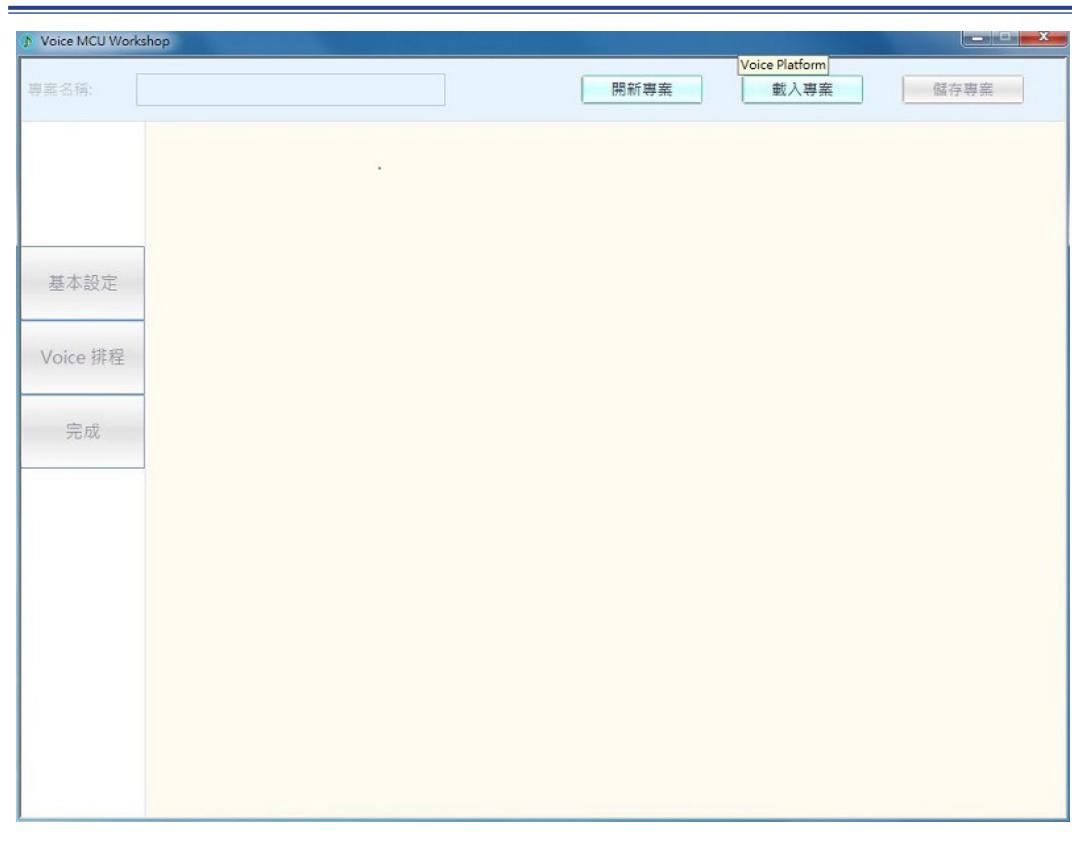
Step5. 点击“Finish”完成安装:



工程建立软体操作 (立即上手)

打开 Voice MCU Workshop V2.4

双击 Voice MCU Workshop V2.4.exe 运行软件，界面如下图所示：



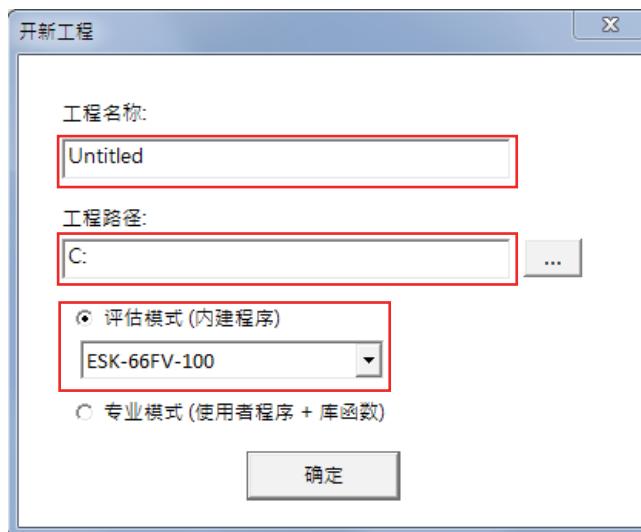
新建一个工程文档 (Project)

评估模式：

Step1. 选择 " 开新專案 " 构建新工程



Step2. 在新建工程后，在工程名称和工程路径中分别输入“工程名称”、“工程路径”，和选择评估模式及评估板，界面如下：



Step3. 点击“确定”后弹出了左边的三个界面可选，其中“基本設定”的界面包括“模式選擇”（选择了评估模式），“可用功能”和“可選用 MCU”，界面如下图所示：



Step4. 四种可用功能的使用与设定:

■ 按键功能:

- 鼠标单击“按键”将该功能载入或撤出 MCU;
- 鼠标单击右边 MCU 中的按键选择用户需要按键的个数。如下图所示:



■ 外接闪存功能:

- 鼠标单击“外接記憶體”将该功能载入或撤出 MCU;
- 鼠标单击右边 MCU 中的外接闪存选择外接闪存的大小。如下图所示:



■ 喇叭驱动功能:

- 鼠标单击“喇叭驅動”将该功能载入或撤出 MCU;
- 鼠标单击 MCU 中喇叭设定驱动模式 (目前只支持 DAC) 输出模式。如下图所示:

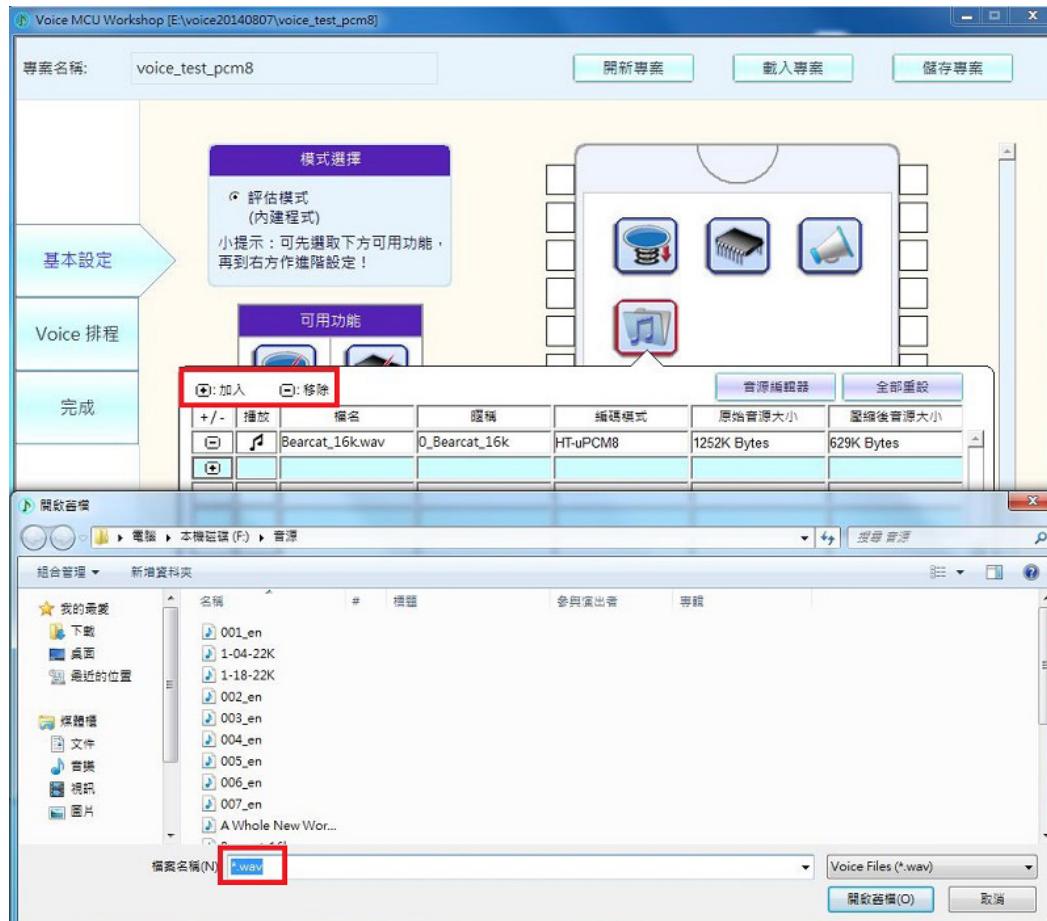


■ 音频来源功能:

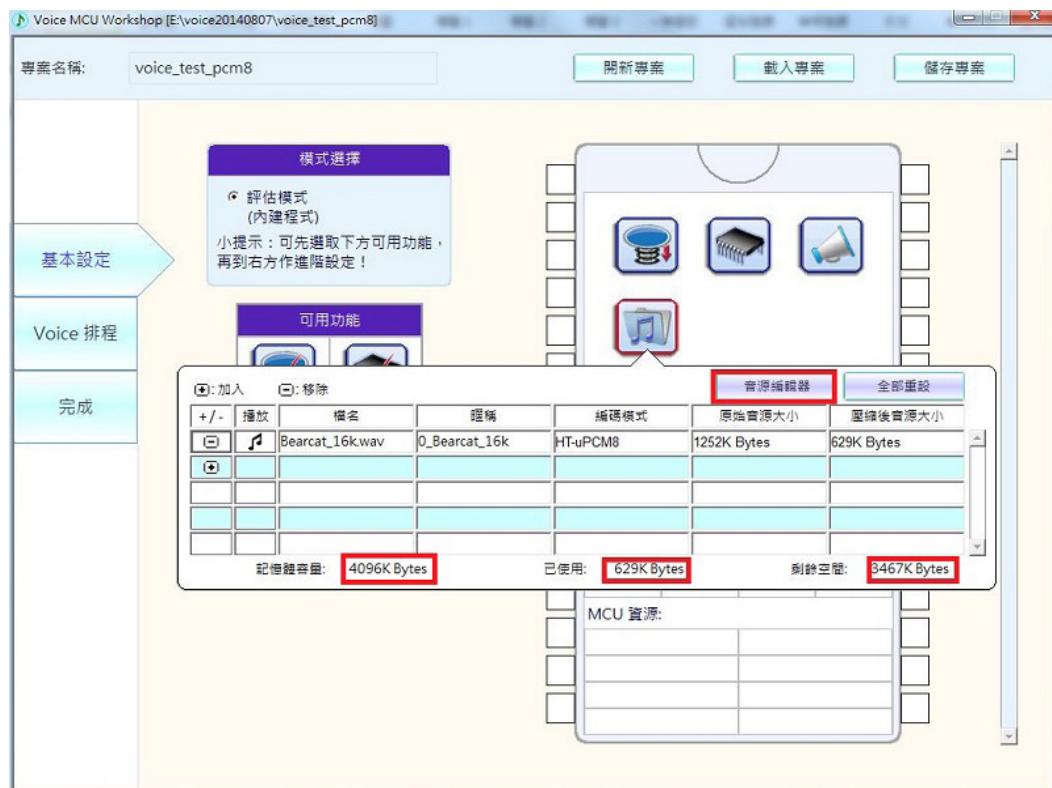
- ①鼠标单击“音频来源”将该功能载入 MCU;



- ②鼠标单击 MCU 中音频来源图标添加或删除 Wav 格式的音频文件。如下图所示：

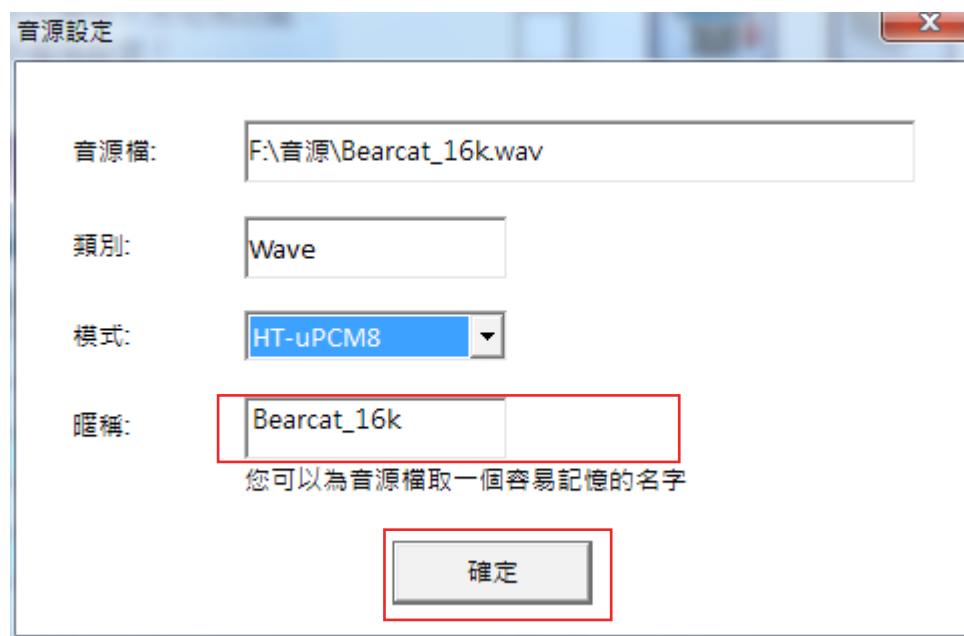
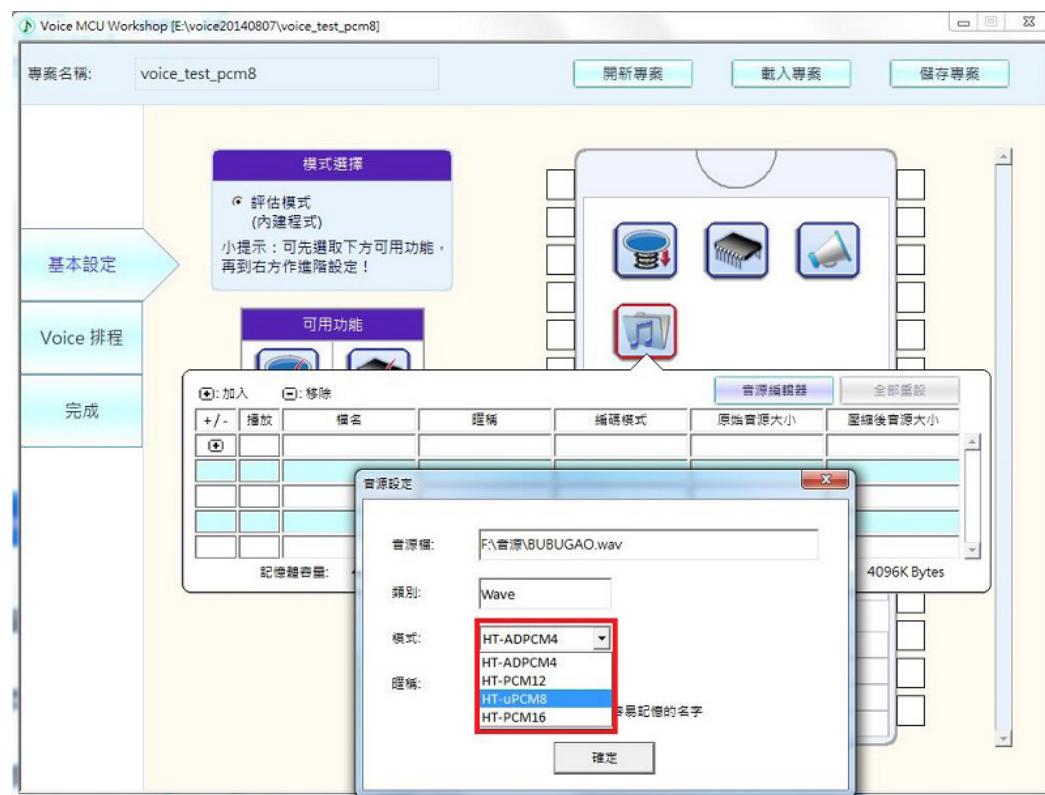


- ③加载音源前可以点击“音频编辑器”连接到“Audacity”软件对音源进行编辑保存后再进行加载（注：首先PC机上得安装Audacity编辑器，可到<http://audacity.sourceforge.net/>下载安装，Audacity应用请查看[Audacity编辑手册](#)），加载后可以显示外接内存、已用内存和剩余内存：

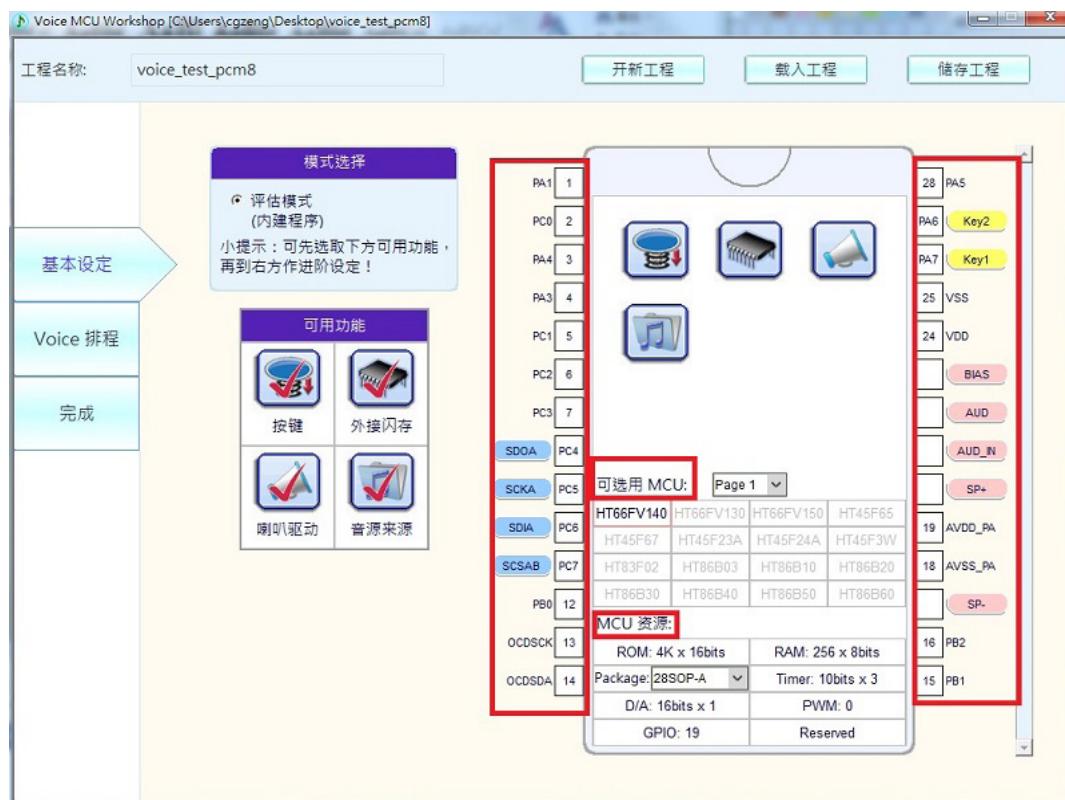


注：可添加最高音源频率可到[library 建立信息](#)章节查看。

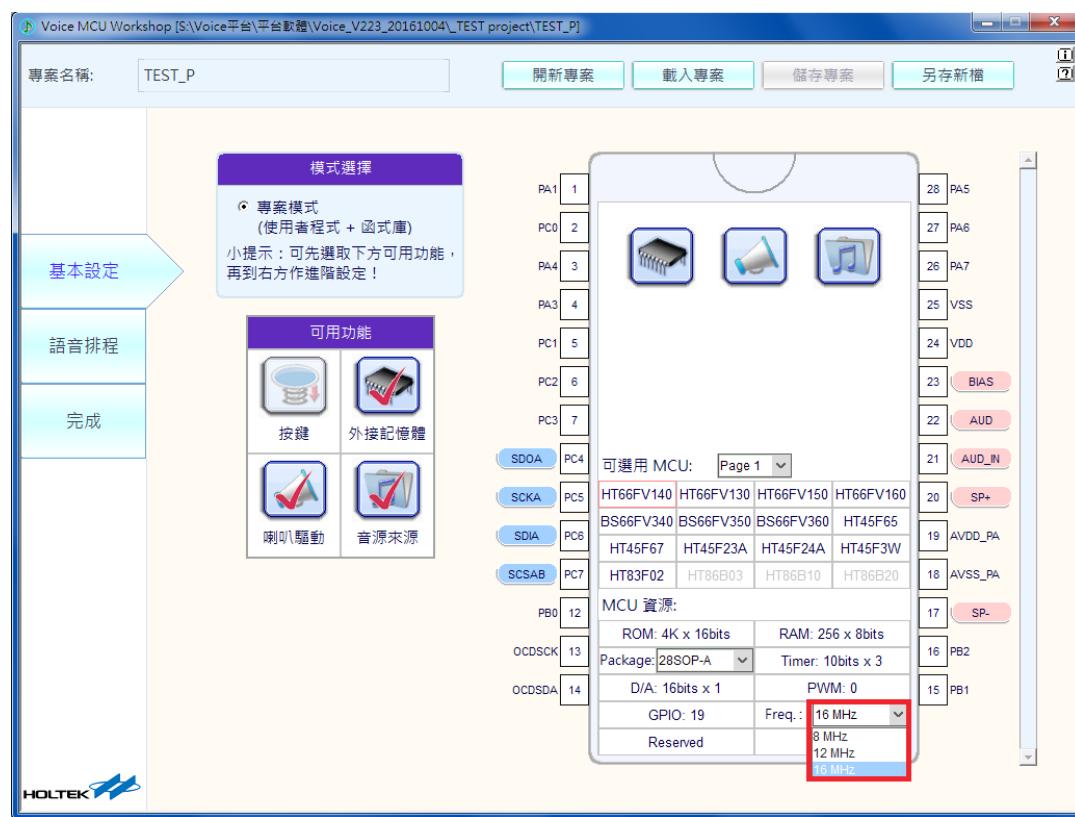
- ④由上面②按下“开启档案”后弹出下面的音频来源信息和音源压缩模式设置等，设置完成后，按下“确定”即可完成音源功能设计。如下图所示：



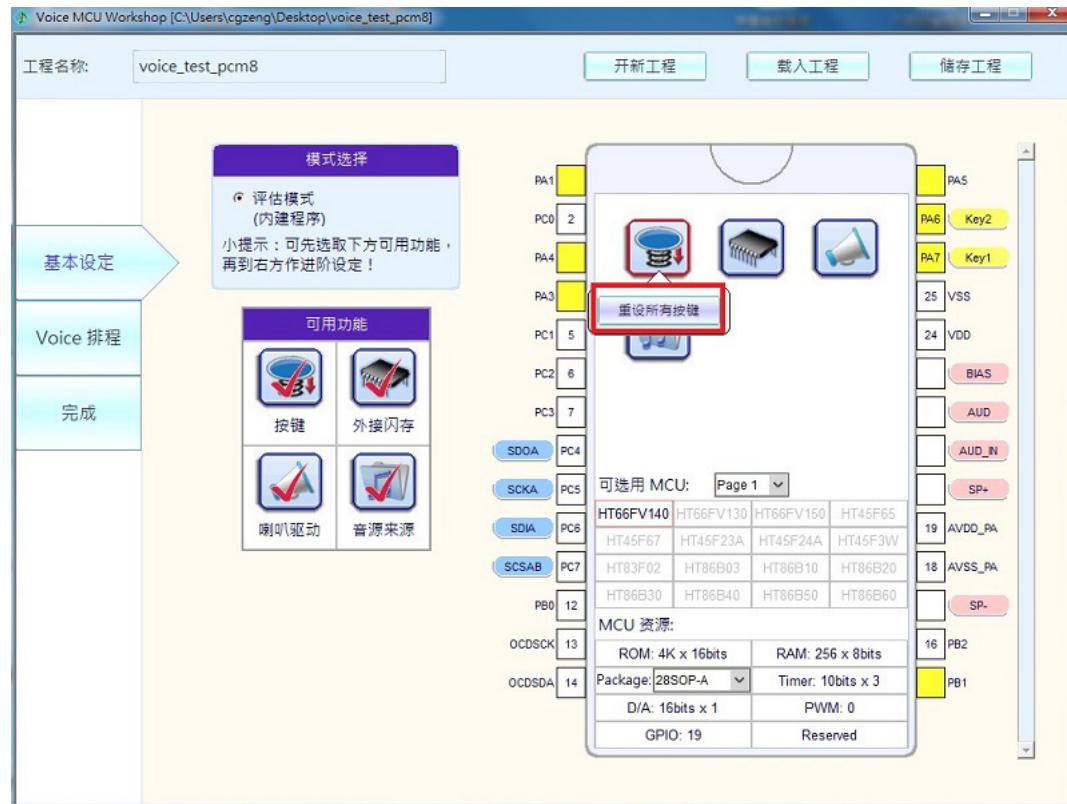
Step5. MCU 的选择了 HT66FV140、相关配对信息的显示 (可用功能对应 MCU 管脚和 MCU 内部资源等), 如下图所示:

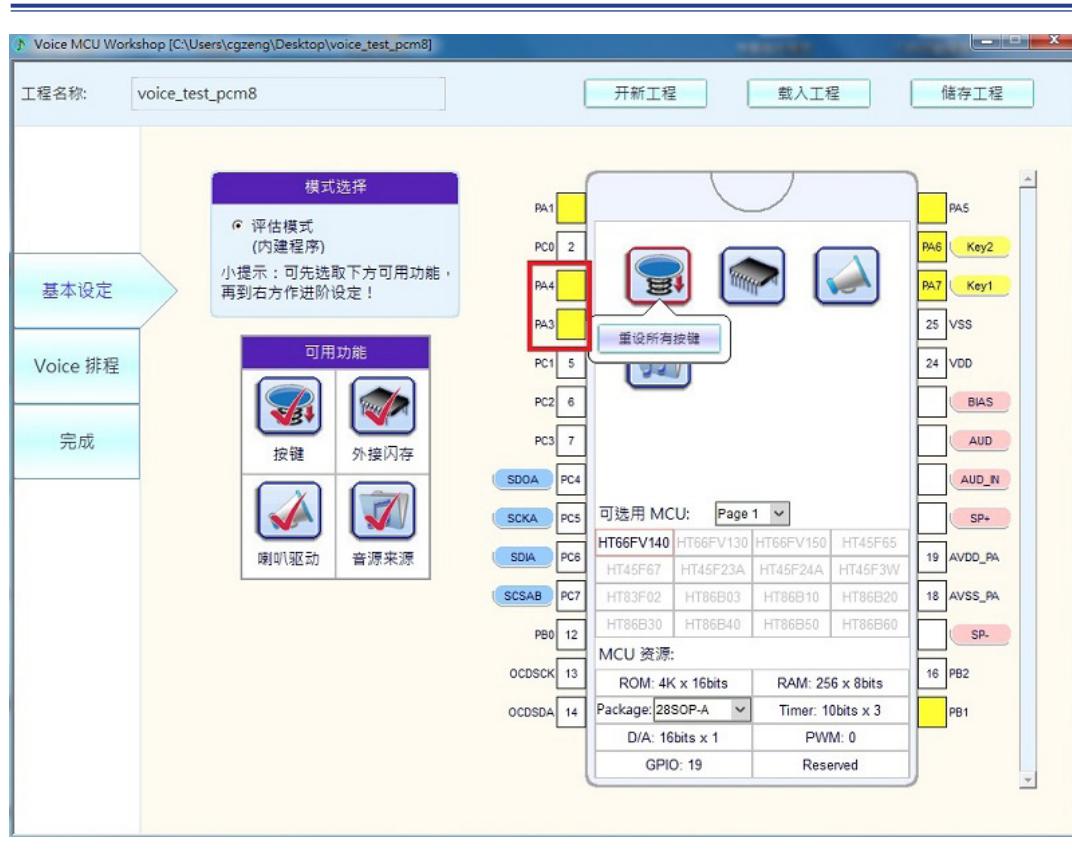


注 1：单击鼠标“右键”可以进行 MCU 频率的选择，结果显示如下图：

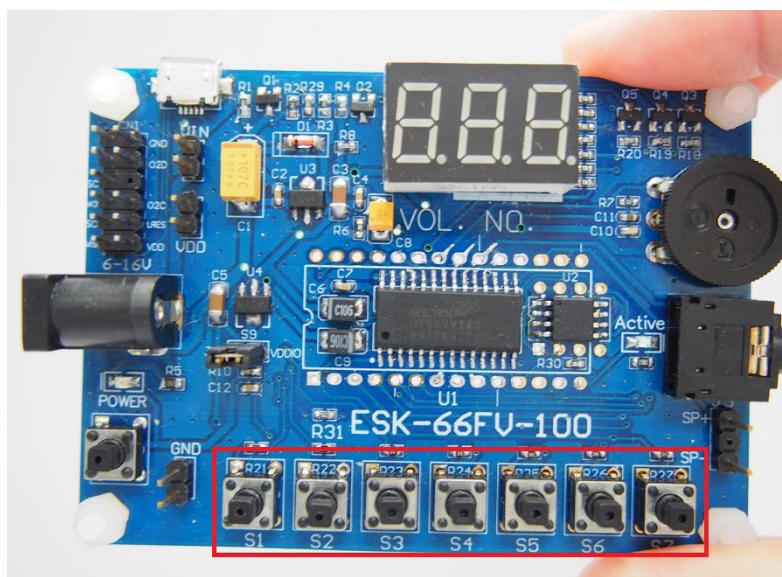


注 2：单击 MCU 中按键的图标出现“重设按键”即可撤销默认按键对应的引脚，再将鼠标移到要想设置的引脚单击即可修改按键对应的脚位如下 2 图所示脚位已修改为 PA4 和 PA3：（注：单独从板子上不看出哪个 KEY 对应对应哪个 IO 口，评估与 MCU 引脚对应如下图：）





评估板上对应引脚如下图所示:



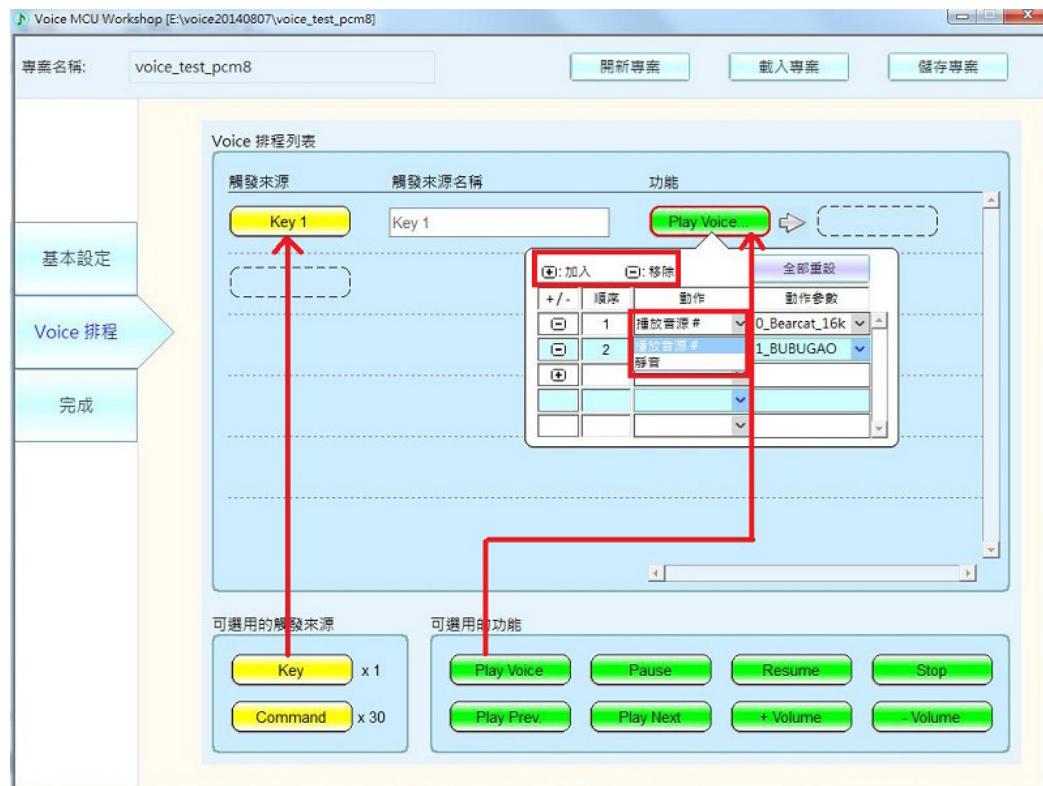
“S1->PA7、S2->PA6、S3->PA5、S4->PA4、S5->PA3、S6->PB1、S7->PA1”

Step6. 基本设定设置完后，点击 voice 排程。

■ 点击“voice 排程”进行工程的逻辑设计，各个设计板块如下图所示：

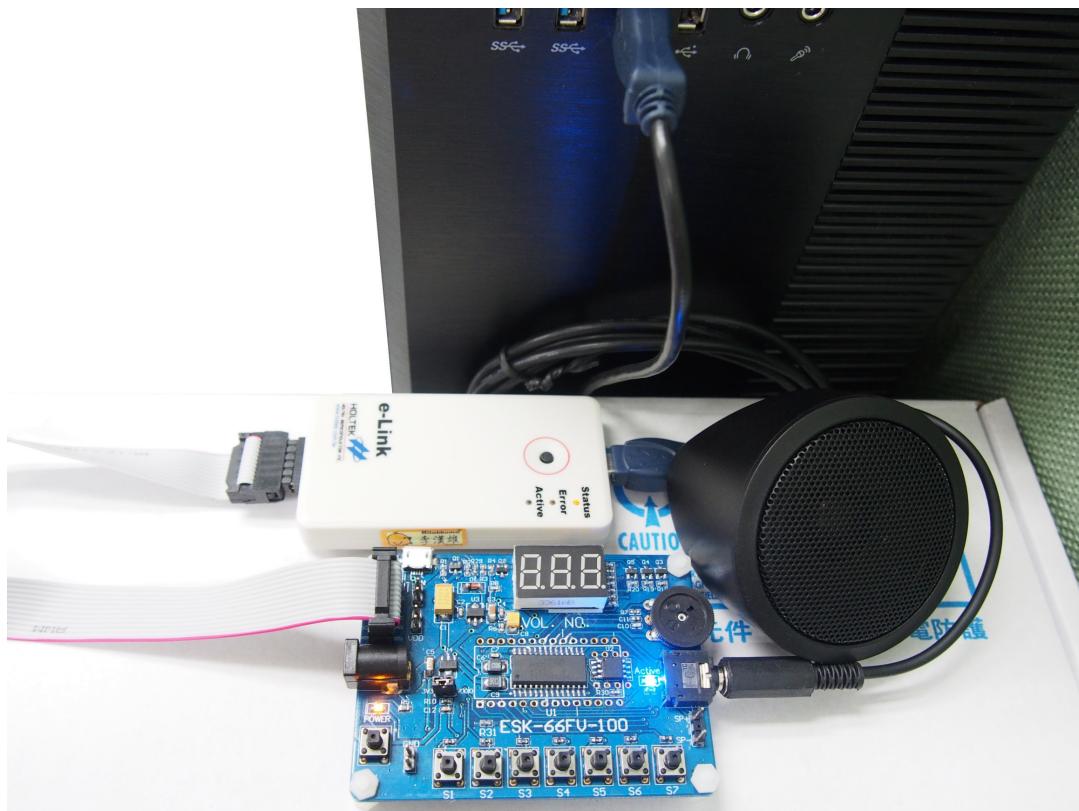


- 单击鼠标左键将可选用触发来源和可选用功能中的模块拖入 voice 排成列表进行逻辑设计如下图所示。(注：“play voice”功能中要添加入要操作的音频选择“播放”或“静音”，并且可以通过“加入”和“移除”构成 sentence)。

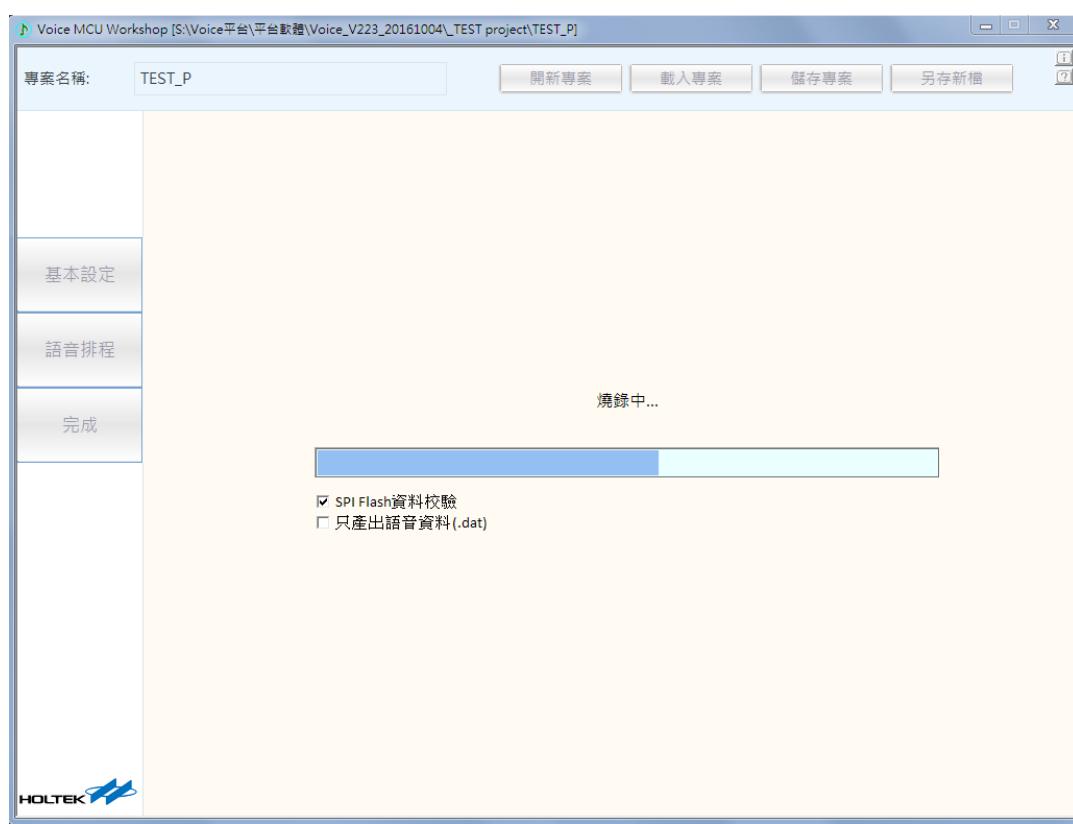


Step7. 连接好 BiCE000ELINK0B (B 板的 e-Link)、评估板和 PC 机。点击“完成”进行程序烧写
(注: 不能插入外接电源), 如下图所示:

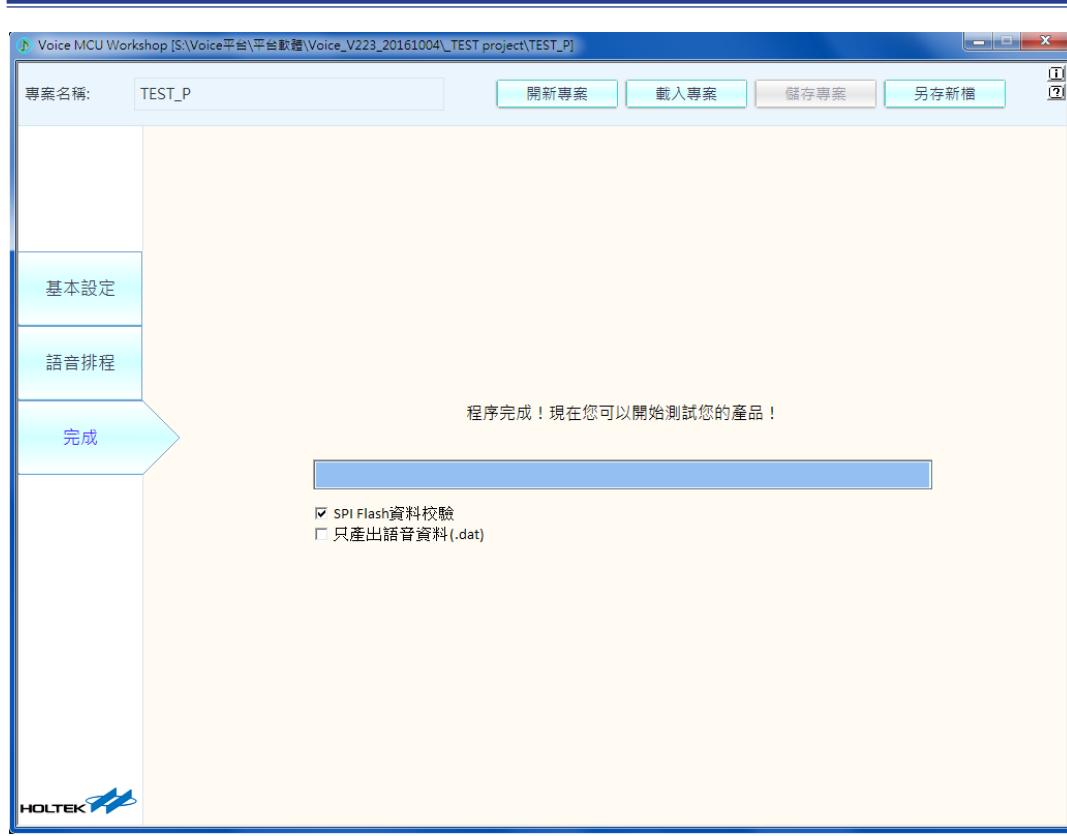
■ 硬件连接



■ 程序烧录中



■ 程序烧录完成



- 直接应用 e-Link 进行供电或拔掉 e-Link 插入外部电源进行供电，外部电源插入后，请按下 POWER 按键，此时 POWER LED 与 Active LED 同时亮起代表可进行功能演示，接入喇叭如下图所示：



专业模式:

Step1. 选择“开新专案”构建新工程



Step2. 在新建工程后，在工程名称和工程路径中分别输入“專案名稱”、“專案路徑”，和“專案模式”，界面如下：



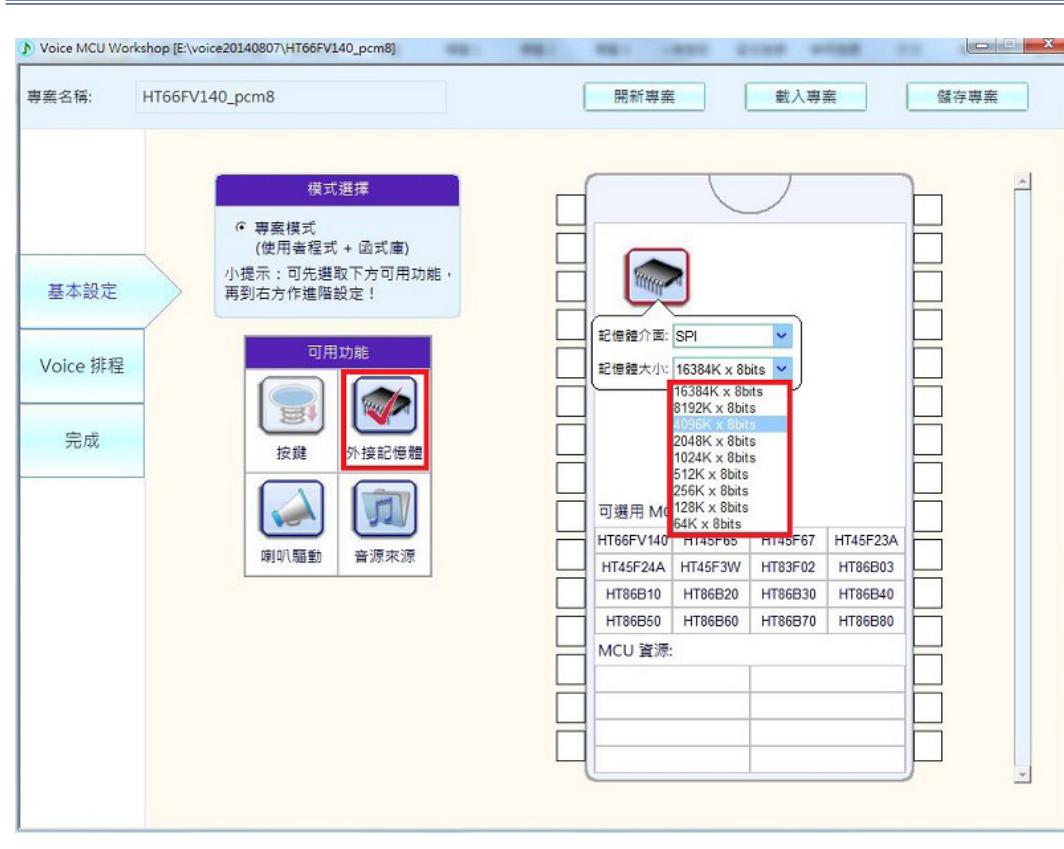
Step3. 点击“确定”后弹出了左边可选择的三个界面，其中“基本設定”的界面包括“模式選擇”（选择了“專案模式”）、“可用功能”（该模式下可用功能中“按鍵”图标变灰无法实现）和“MCU 的選擇”，界面如下图所示：



Step4. 三种可用功能的使用与设定：

■ 外接闪存功能：

- 鼠标单击“外接記憶體”将该功能载入 MCU 或从 MCU 中撤销；
- 鼠标单击右边 MCU 中的外接闪存选择外接闪存的大小。如下图所示：



■ 喇叭驱动功能:

- 鼠标单击“喇叭驅動” 将该功能载入 MCU 或从 MCU 中撤销；
- 鼠标单击 MCU 中喇叭设定驱动模式(目前只支持 DAC 输出模式)。如下图所示：

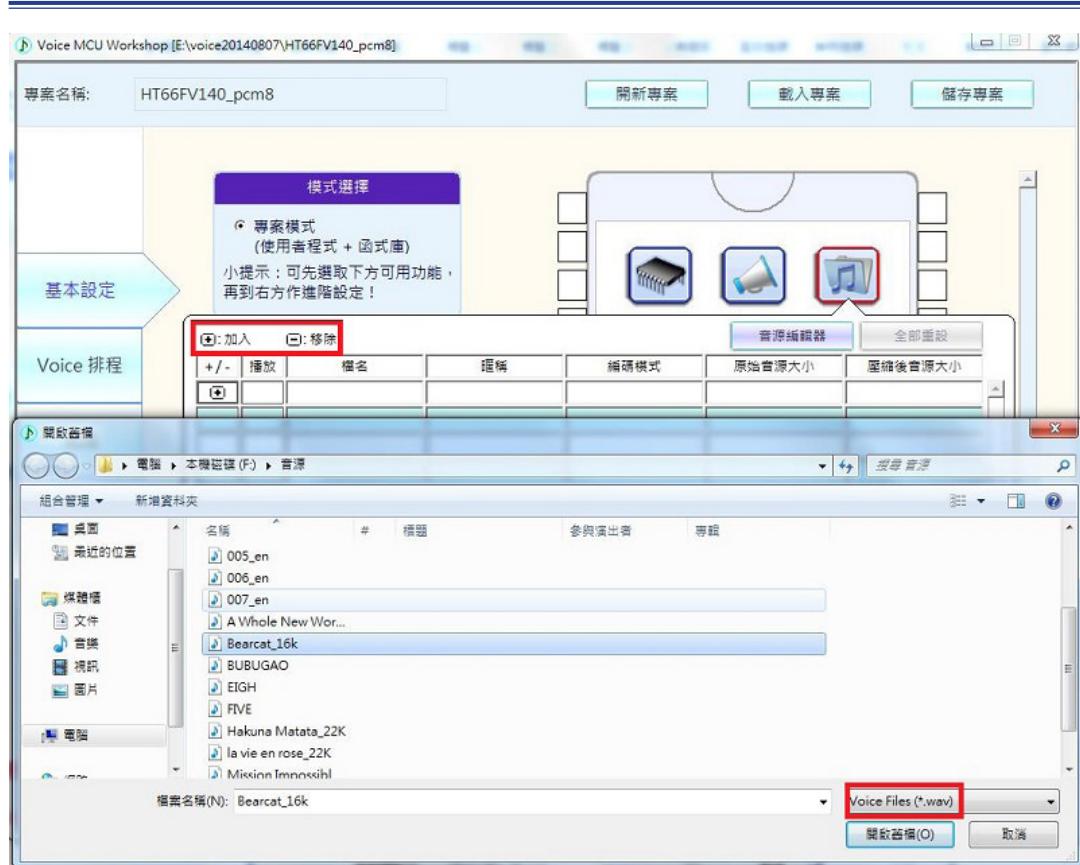


■ 音频来源功能:

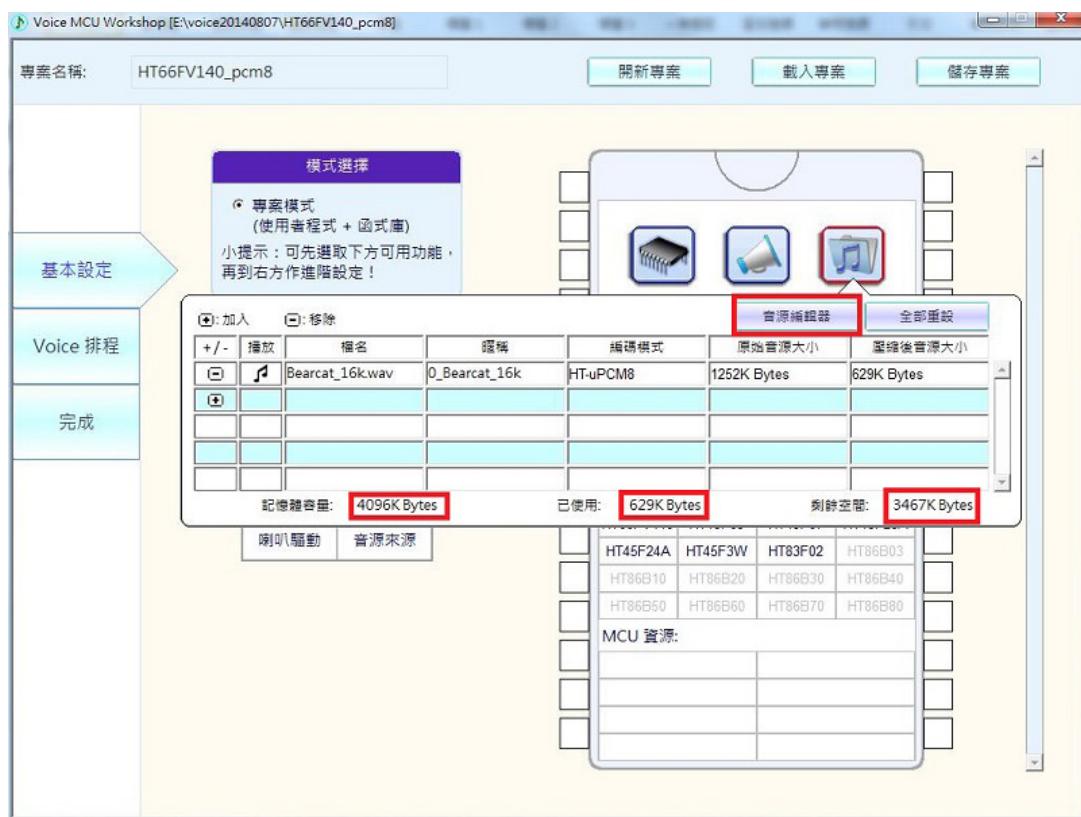
- ①单击“音频来源” 将该功能载入 MCU 或从 MCU 中撤销；



- ②点击MCU中音频来源图标添加或删除Wav格式的音频文件，如下图所示（点击加入）：

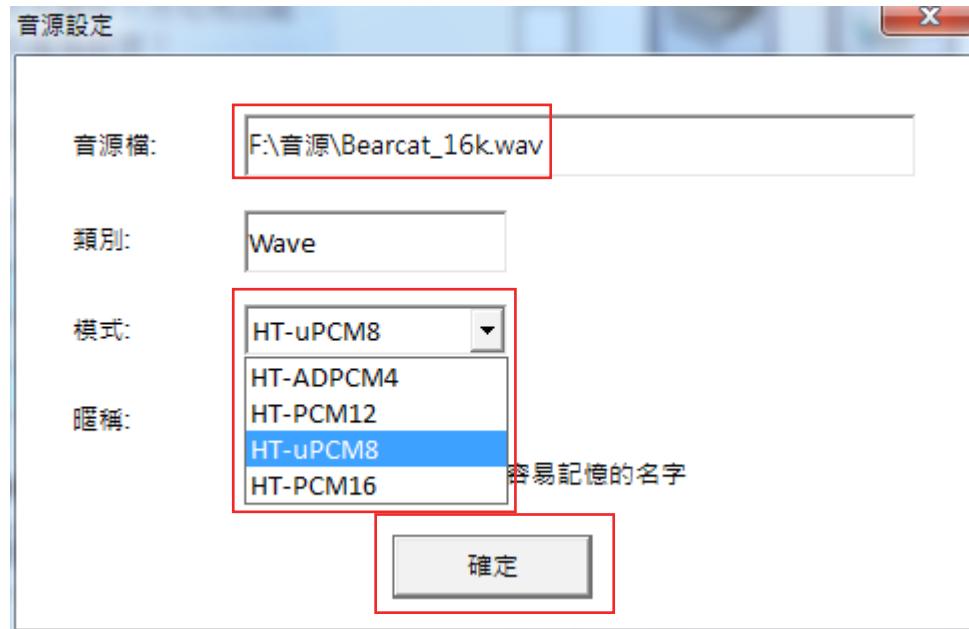


- ③加载音源前可以点击“音频编辑器”连接到“Audacity”软件对音源进行编辑保存后再进行加载（注：首先PC机上得安装Audacity编辑器，可到<http://audacity.sourceforge.net/>下载安装，Audacity应用请查看[Audacity编辑手册](#)），加载后可以显示外接内存、已用内存和剩余内存：



注：可添加最高音源频率可到[library 建立信息](#)章节查看。

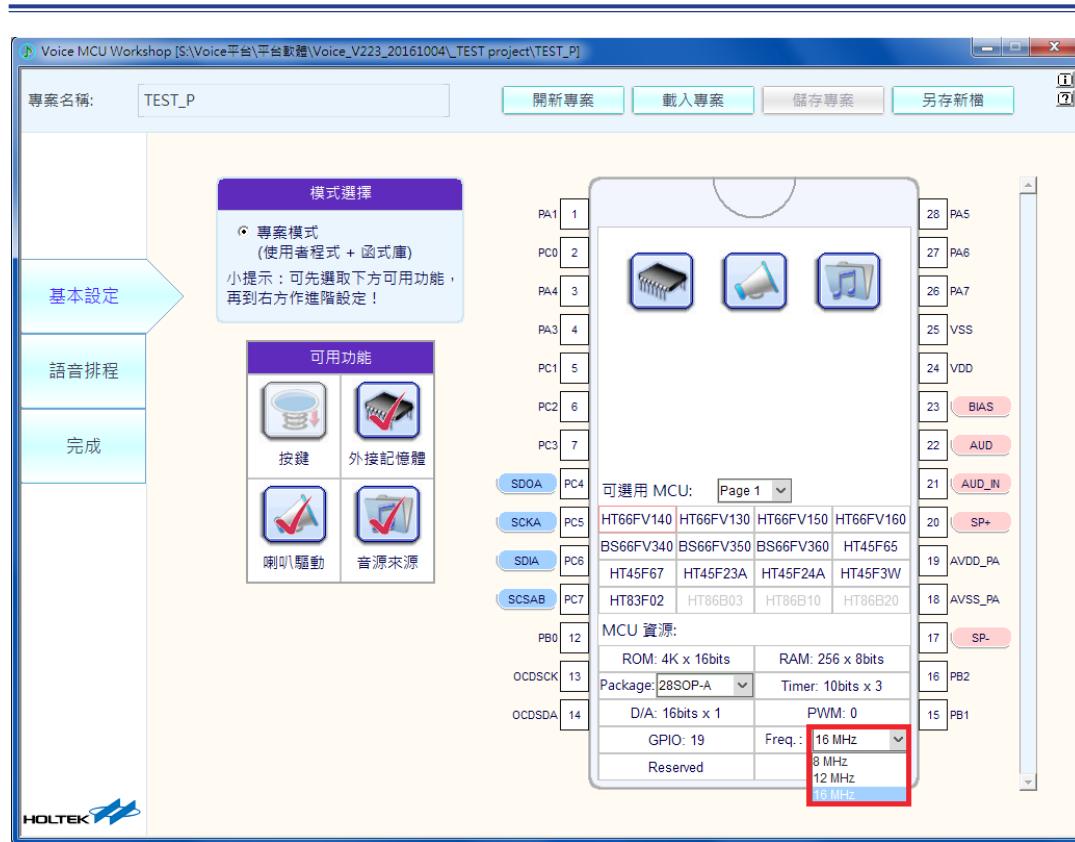
- ④由上面②按下“開啟檔案”后弹出下面的音频来源信息和“模式”，设置完成后，按下“確定”即可完成音源功能设计。如下图所示：



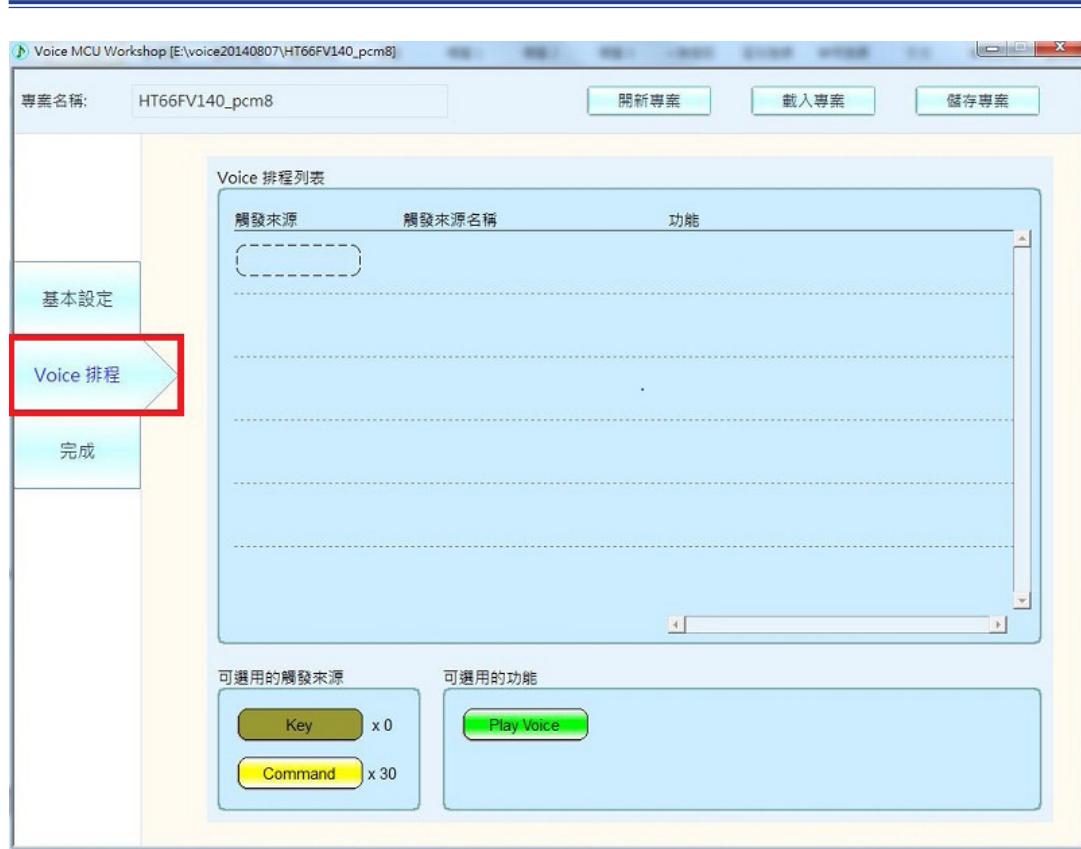
Step5. MCU 的选择、相关配对信息的显示 (可用功能对应 MCU 管脚和 MCU 内部资源等),
如下图所示:



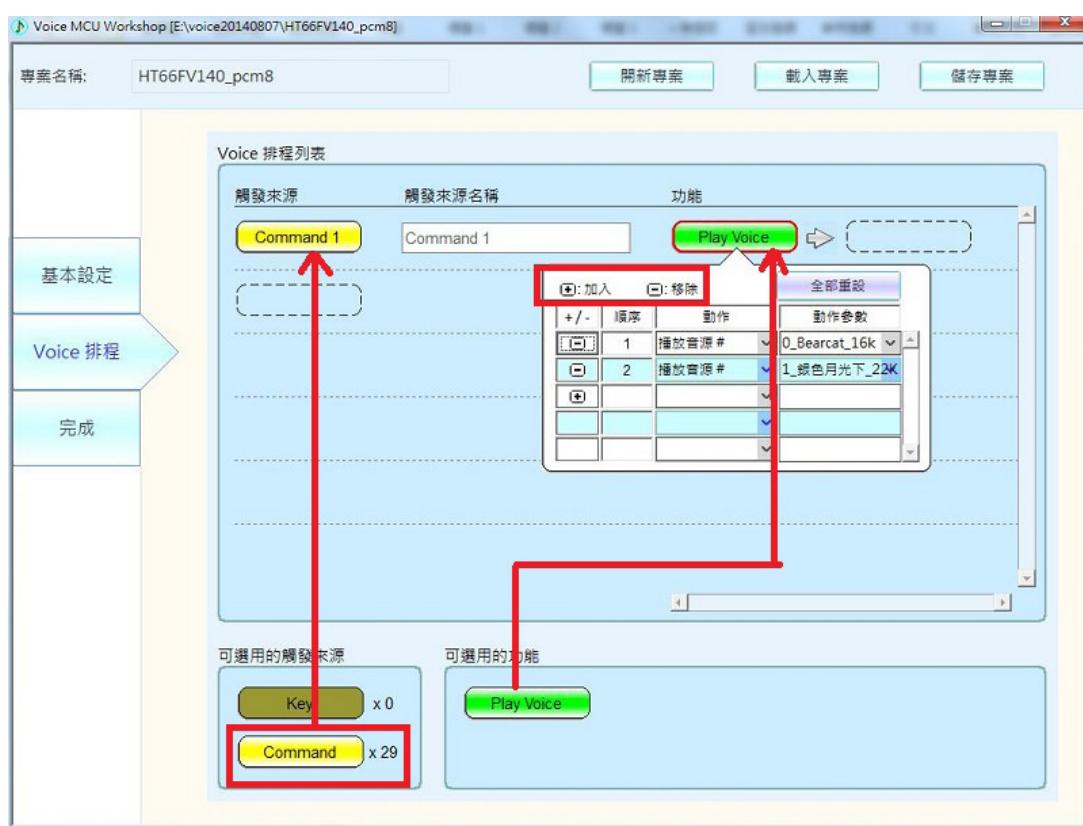
注：单击鼠标“右键”可以进行 MCU 频率的选择，结果显示如下图：



Step6. 设置完基本设定后, 点击“VOICE 排程”, 得到如下图所示:



Step7. 依据需要触发命令个数进行编排 (Command 指的是 play sentence 地址), 如下图所示:

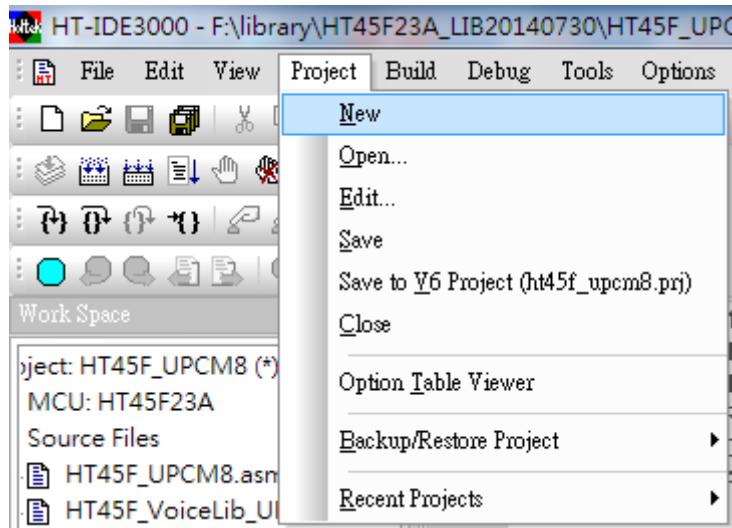


Step8. 编排完成后, 点击“完成”选项, 将 DAT 文件(音频压缩文件)烧录到 FLASH 中, 和在构建的专业模式下生成库和一些相关函数进行调用。

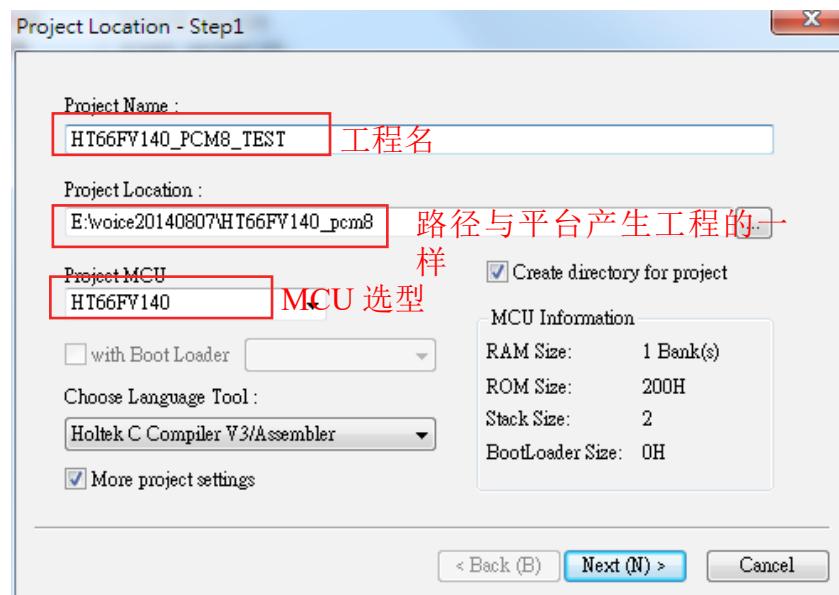


Step9. 在刚刚完成的专业模式工程目录下新建一个 IDE-3000 的工程对相关的库和文件进行调用

- 打开 IDE3000 菜单 “ ‘project’ -> ‘NEW’ ” 新建工程;



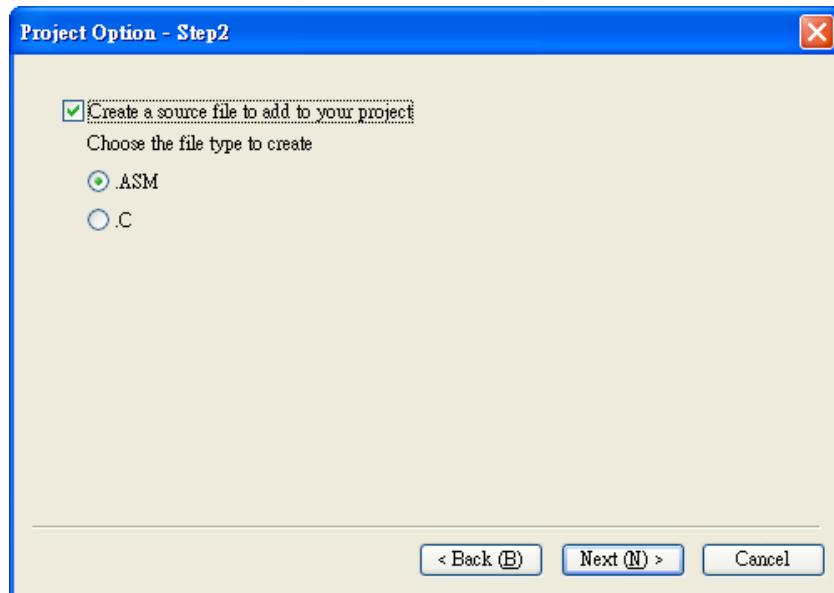
- 点击“NEW”后，填充下列信息:



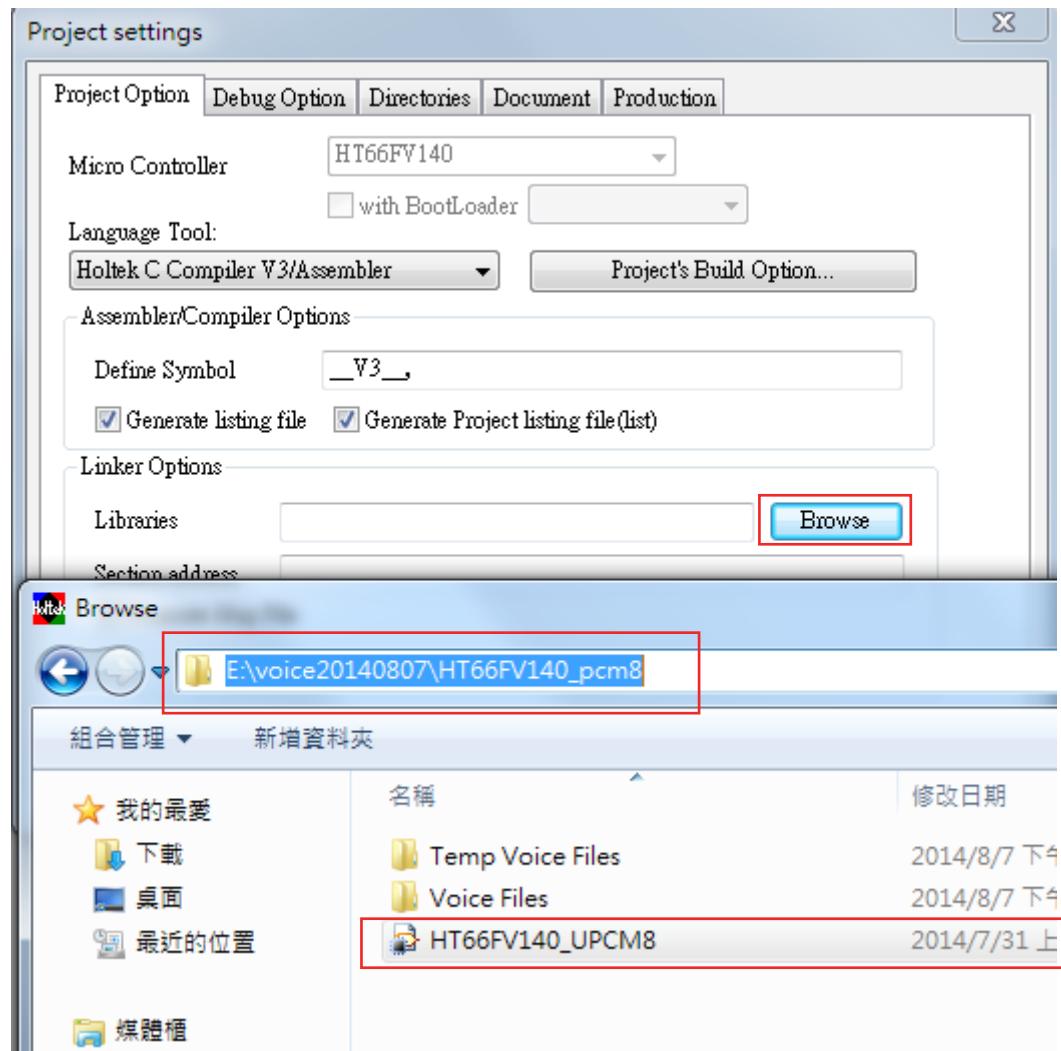
注：新建的 IDE3000 工程路径选择与平台产生的路径主要是为了能够在该工程下调用库（受编译器限制，库要和工程在同一目录下才能被调用），或者也可以两个工程不同一目录也可以但需将

library  HT66FV140_UPCM8
Altium Library
13 KB 这个库文件拷贝到 IDE3000 工程目录下。

- 由上图点击 next，对要构建的工程所采取开发的语言进行选择。



- 添加库文件到新建 IDE3000 的工程中：

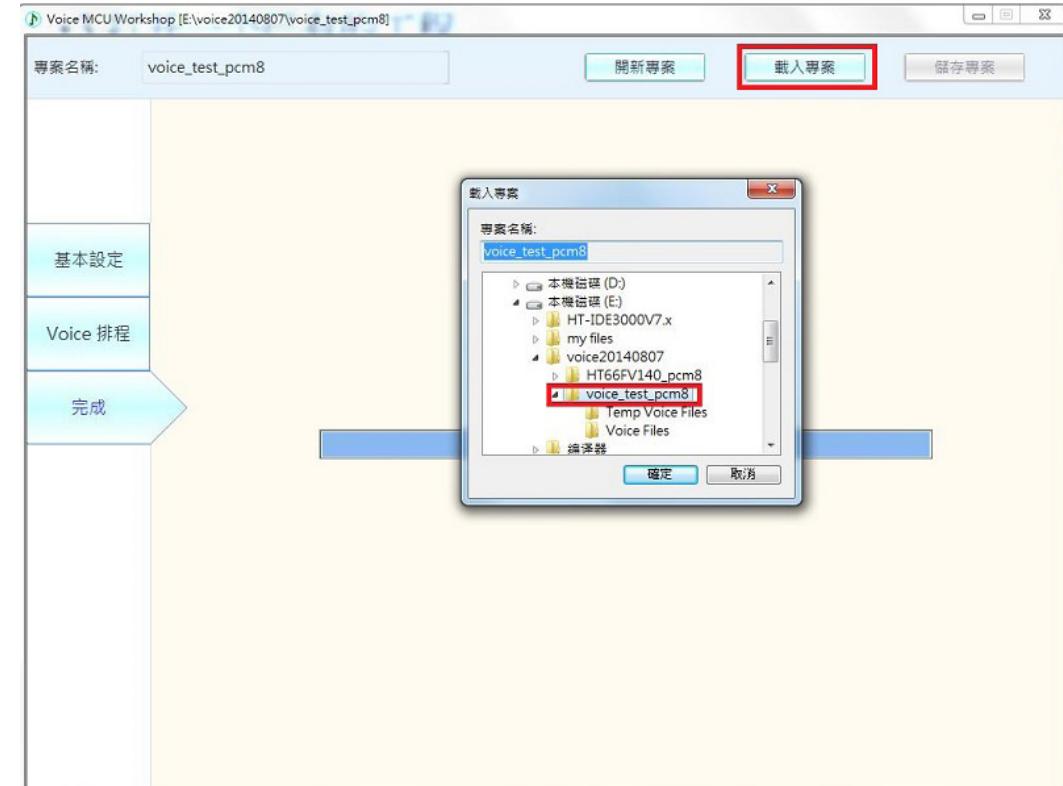


- 参考 [ASM CALL](#) 或 [C CALL](#) (按住 Ctrl 键单击鼠标左键即可跳转到该地方) 进行调用相关函数来完成工程构建。

Step10. 构建好工程后, 将 IDE-3000 工程生成的 .MTP 文件下载到语音 MCU 进行调试及播放。

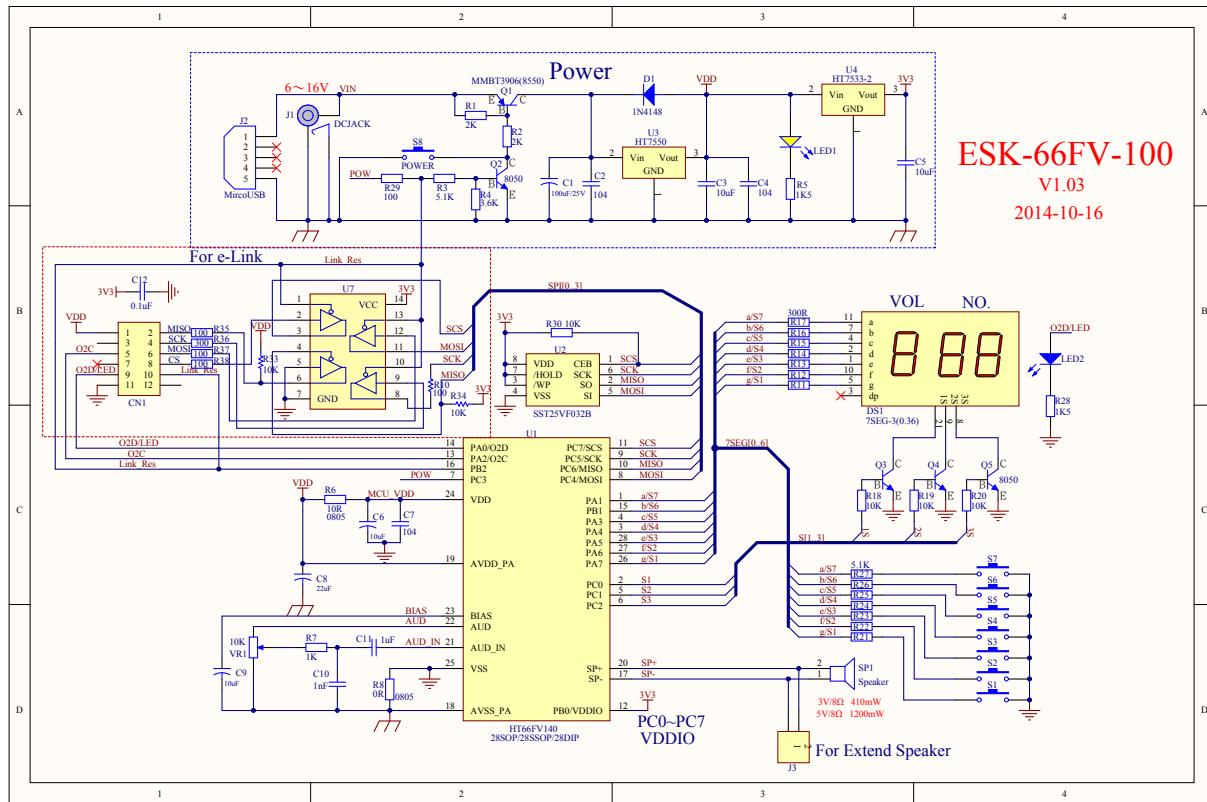
打开一个已有的工程

单击“载入专案”打开已有的工程的路径即可像上述新建工程那样编辑或下载。界面如下图所示：



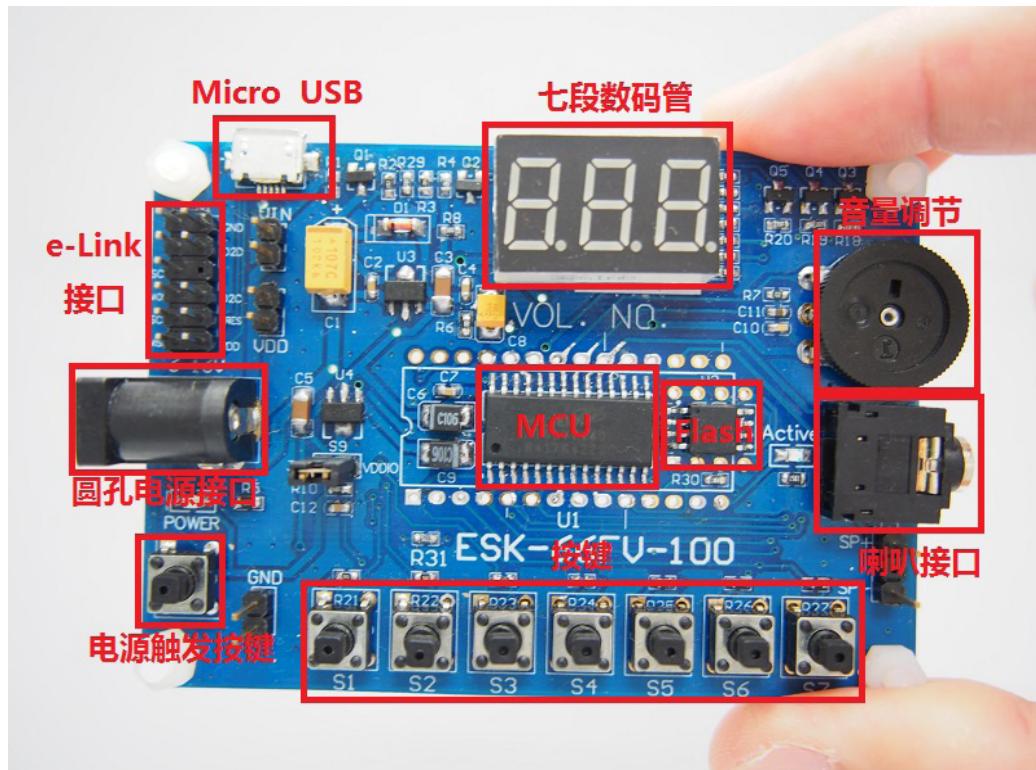
硬件电路

评估板原理图



评估板使用方式

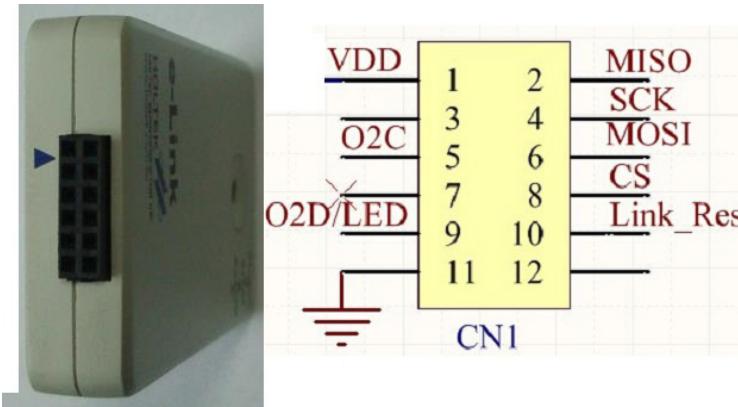
■ 评估板认识



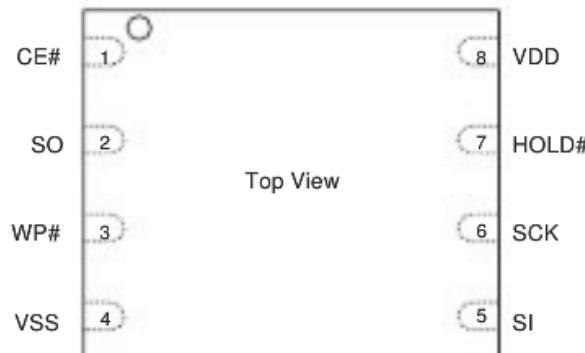
■ 硬件使用步骤 (已烧录程序)

- 接上外接喇叭。
- 通过“圆孔电源接口”接入 6V~16V 电源或通过“微型 USB”接入 5V，并按下“电源打开按键”，或者下载完后直接应用 e-Link 进行供电。
- 通过所设置的“音频控制按键”控制音频播放或可通过旋转“音量调节”改变声音的大小。

- 单独将 DAT 文件烧录到 flash 的接线方法
 - flash 的连线方式



上图中左图所对应的为 e-Link 的管脚图, 右图为实物图三角标志表示第一引脚, 两者是对应的。
下图为 flash 的引脚图。



在进行对 flash 进行单独烧录数据时, e-Link 与 flash 连接线如下所表示:
e-Link 的 VDD→flash 的 VDD; e-Link 的 GND→flash 的 VSS;
e-Link 的 MISO→flash 的 SO; e-Link 的 SCK→flash 的 SCK;
e-Link 的 MOSI→flash 的 SI; e-Link 的 SCS→flash 的 CE#.

平台所支持 flash 型号

MXIC 系列			
128M bits	MX25L12873F		MX25L3206E
	MX25L6406E		MX25L3235E
	MX25L6435E		MX25L3208E
	MX25L6408E		MX25L3273E
	MX25L6473E		MX25L8006E
	MX25L1606E	8M bits	MX25L8035E
	MX25L1633E		MX25L8036E
	MX25L1608E	4M bits	MX25L4006E
	MX25L1635E		MX25L4026E
	MX25L1636E	2M bits	MX25L2006E
	MX25L1673E		MX25L2026E
1M bits	MX25L1006E	512K bits	MX25L512E
	MX25L1026E		
SST 系列			
64M bits	SST26VF064B	8M bits	SST25VF080B
	SST25VF032B		SST25VF040B
32M bits	SST26VF032B	4M bits	
	SST25VF016B		SST25PF020B
16M bits	SST26VF016B	2M bits	SST25VF020B
Winbond 系列			
128Mbits	W25Q128BV		W25Q80CV
	W25Q128FV	8Mbits	W25Q80DV
	W25Q64CV		W25Q80BL
64Mbits	W25Q64FV		W25Q40CL
	W25Q32FV	4Mbits	W25X40CL
32Mbits	W25Q32BV		W25Q20CL
	W25Q16CV	2Mbits	W25X20CL
16Mbits	W25Q32BV	1Mbits	W25X10CL
	W25Q16CL	512Kbits	W25X05CL
GigaDevice 系列			
128M bits	GD25Q128C	4M bits	GD25Q40C
64M bits	GD25Q64C		GD25Q41B
32M bits	GD25Q32C	2M bits	GD25Q20C
16M bits	GD25Q16C	1M bits	GD25D10B
8M bits	GD25Q8C	512K bits	GD25D05B

2 ASM 及 C 库使用说明

ASM 调用 Voice library

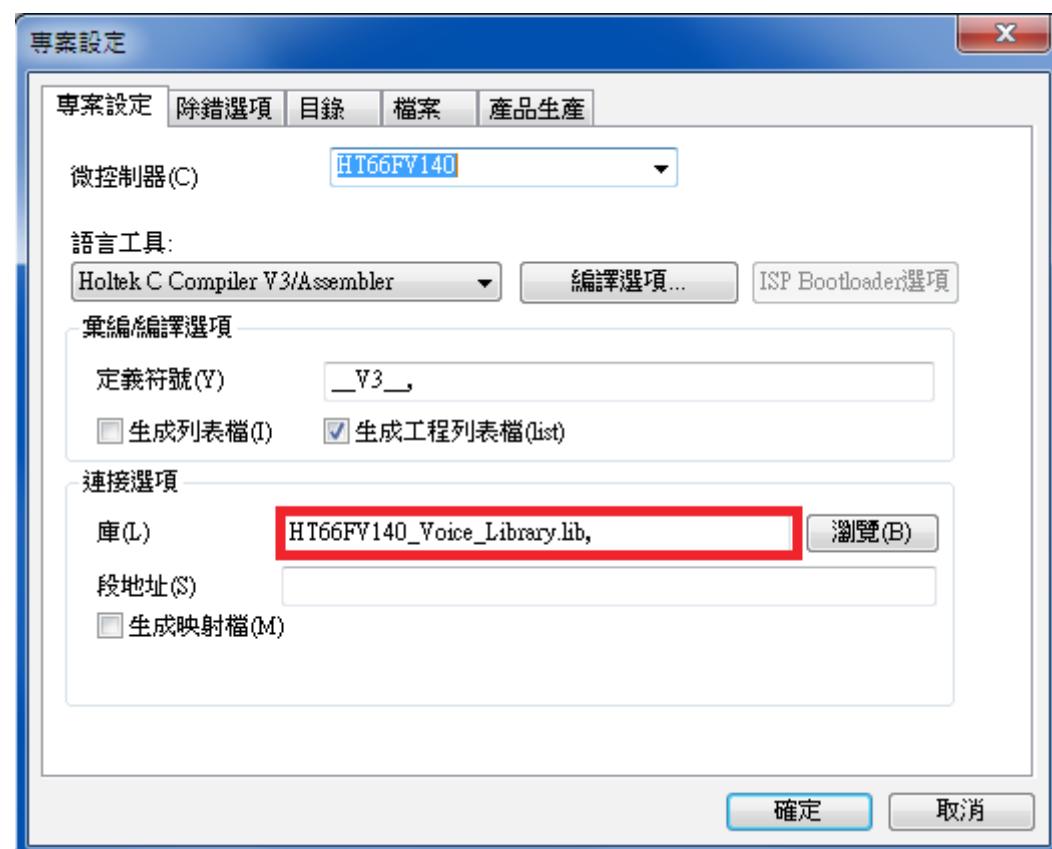
目的说明

主要介绍客户如何使用 ASM 呼叫 Voice library 提供的函数。

如何使用

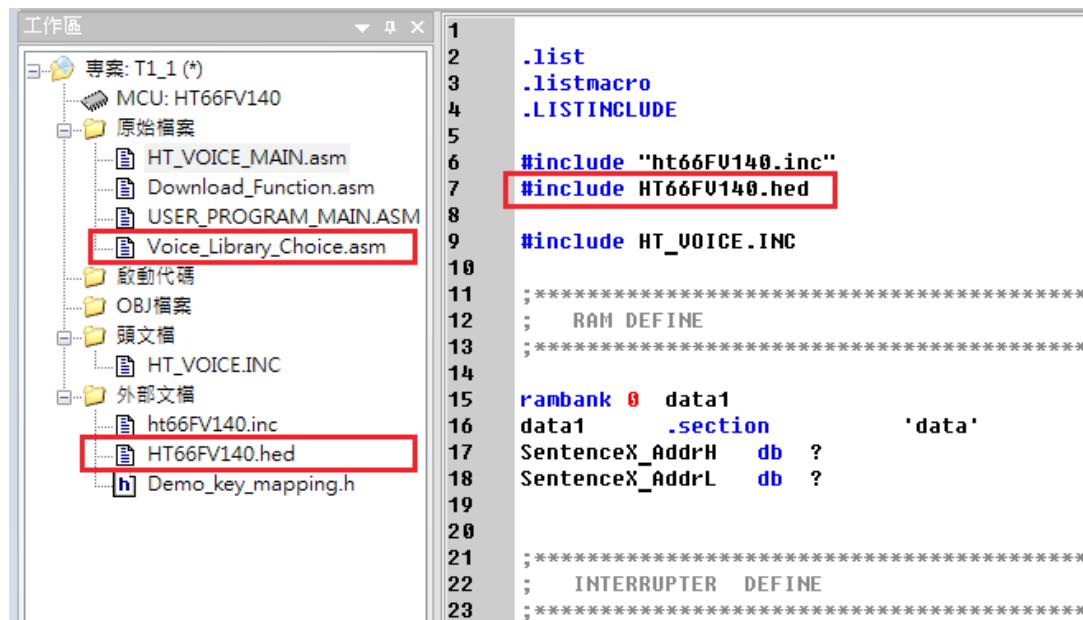
创建 .ASM 工程后：

- 添加库文件



■ 添加头文件

添加库的头文件如： XX.hed & Voice_Library.Choice.asm，以便调用库里面的函数。



The screenshot shows the software interface with the following details:

- Project Structure:**
 - Project: T1_1 (*)
 - MCU: HT66FV140
 - Source Files:
 - HT_VOICE_MAIN.asm
 - Download_Function.asm
 - USER_PROGRAM_MAIN.ASM
 - Voice_Library.Choice.asm** (highlighted with a red box)
 - Object Code
 - OBJ Files
 - Header Files:
 - HT_VOICE.INC
 - External Files:
 - ht66FV140.inc
 - HT66FV140.hed** (highlighted with a red box)
 - Demo_key_mapping.h
- Code Editor:**

```

1   .list
2   .listmacro
3   .LISTINCLUDE
4
5
6   #include "ht66FV140.inc"
7   #include HT66FV140.hed
8
9   #include HT_VOICE.INC
10
11  *****
12  RAM DEFINE
13  *****
14
15  rambank 0 data1
16  data1    .section      'data'
17  SentenceX_AdrH  db  ?
18  SentenceX_AdrL  db  ?
19
20
21  *****
22  INTERRUPTER  DEFINE
23  *****

```

■ 参照举例编写程序 (具体例程举例)

所提供函数

■ _CLRRAM

描述:

ram 的所有 bank 区均被清为 0.

举例:

```
_CLRRAM
```

■ _SYSTEM_INITIALIZATION

描述:

系統 Fsys 设置, SPI 介面设置, timer 初始化等

举例:

```
_CLRRAM
```

```
_SYSTEM_INITIALIZATION
```

■ _DAC_RAMP_UP

描述:

DA 使能, 在 _PLAY_VOICE, _PLAY_SENTENCE, _PLAY_SENTENCE_INDEX; 执行之前需先执行该函数

举例:

```
_DAC_RAMP_UP
```

```
_PLAY_VOICE 0, 0, 0, 7, 0
```

■ _DAC_RAMP_DOWN

描述:

DA 除能, 在 _PLAY_VOICE, _PLAY_SENTENCE, _PLAY_SENTENCE_INDEX; 执行之后, 执行该函数。可以减少不必要的耗电流

举例:

```
_PLAY_VOICE 0, 0, 0, 7, 0
```

```
_DAC_RAMP_DOWN
```

■ _STOP_PLAY

描述:

停止放音, 在任何时刻都可以直接调用该函数。

举例:

```
_STOP_PLAY
```

■ **_VOLUME Volume**

描述:

User 設置音量大小，參照規格書上的音量值，直接寫入

参数:

Volume: 规格书中的音量值

举例:

_VOLUME 0; 音量调节最小

_VOLUME 7; 音量调节最大 (注: 各音量的大小会存在不同设置值及范围, 请参照规格书上的音量值)

注 : HT66FV1X0 系列音量调节为 0 ~ 12

■ **_PLAY_VOICE VoiceNumHigh, VoiceNumLow, Channel, Volume, Reserve**

描述:

播放预先通过 voice 平台將 WAV 音频文件生成的 DAT 存储到 flash 的 voice 文件。

参数:

VoiceNumHigh: Voice NUM high byte

VoiceNumLow: Voice NUM low byte

Channel: 选择音频播放通道 (目前只支持通道 0)

Volume: Voice 音量选择 (0-7)

Reserve: 0

举例:

播放第 1 首音原始文件 (注: UI 介面上的音源序号从 “0” 开始计数, 並不是 “1”)

采用的音量为 7

则: **_DAC_RAMP_UP**

_PLAY_VOICE 0, 0, 0, 7, 0

- `_PLAY_SENTENCE SentenceNumHigh, SentenceNumLow, Channel, Volume, Reserve`
描述:

`play_sentence`

参数:

`SentenceNumHigh: SentenceAddr high byte`

`SentenceNumLow: SentenceAddr low byte`

Channel: 选择音频播放通道 (目前只支持通道 0)

Volume: Sentence 音量选择

Reserve : 0

举例:

播放第 1 首 sentence 文件 , 假设其地址为 0100H, 采用的音量为 7

则: `_DAC_RAMP_UP`

`_PLAY_SENTENCE 01h, 00h, 0, 7, 0`

注: sentence 的各地址在平台生成工程目录下的 Demo_key_mapping.h 文件中得到,
如下图所示: (第一个 sentence 地址为 0100H。)

```
*****
#define COMMANDABLE_NUM 1 ;;
#define COMMAND1_START_ADDRESS 0100H ;;
*****
```

- `_PLAY_SENTENCE_INDEX Reservel, Sentence index, Channel, Volume, Reserve`

描述: 播放第几首 Sentence

参数:

`Reservel: no use`

`Sentence index: index number (1~255)`

Channel: 选择音频播放信道 (目前只支持信道 0)

Volume: Sentence 音量选择

Reserve : 0

举例: 播放第 1 首 sentence 文件 , 采用的音量为 7

则: `_DAC_RAMP_UP`

`_PLAY_SENTENCE_INDEX 0, 1, 0, 7, 0`

注: Sentence index 在语音排程列表中得到, 如下图所示:

语音排程列表		
觸發來源	觸發來源名稱	功能
<code>Sentence 1</code>	Sentence 1	Play
<code>Sentence 2</code>	Sentence 2	Play

■ _MODIFY_SAMPLINGRATE mSamplingRate

描述： 改变当前播放语音的采样率

参数：

mSamplingRate : 指定的采样率值 (Hz)

举例： 改变当前播放语音的采样率为 11025Hz

则：

_MODIFY_SAMPLINGRATE 11025

_PLAY_VOICE 0, 0, 0, 7, 0

■ _PLAY_VOICE_ISR

描述：

根据初始化的时间，当定时中断到达时，就进入该中断函数进 play voice

举例：

ORG XXH; XXH:play voice 定时中断入口地址

_PLAY_VOICE_ISR

■ _PLAY_SENTENCE_ISR

描述：

根据初始化的时间，当定时中断到达，就进入该中断函数进行 play sentence

举例：

ORG XXH; XXH:play sentence 定时中断入口地址

_PLAY_SENTENCE_ISR

注： 下面程序可应用于判断 Play voice or sentence 结束

MOV A,00H

SZ fSentencePlaying; fSentencePlaying =1 代表 Sentence 在播放， 0 播放完

RET

SZ fVoiceStandBy; fVoiceStandBy=0 代表 Voice 在播放， 1 播放完

MOV A,01H; 若 Play voice or sentence 結束， 則 A=1， 否則 A=0， 可通过判断 A 即可得到是否播放完毕。

■ _ENABLE_VDDIO

描述：

使能 MCU 的 VDDIO 功能，使 SPI 引脚的电压来源于 VDDIO; 在 _CLRRAM 之前需调用此函数。

范例：

_ENABLE_VDDIO

_CLRRAM

_SYSTEM_INITIALIZATION

■ _PAUSE

描述:

当一个 Voice 或 Sentence 正在播放时，调用此函数可以暂停播放。

范例:

```
_PLAY_VOICE 0, 0, 0, 3, 0      ; 播放第一个 voice, 音量为 3  
_CALL_DELAY                      ; 延时一段时间, 在声音播放一段时间后暂停  
_PAUSE                            ; 调用暂停函数
```

注: 上述延时函数只是提供一个范例, 语音函数库中并不提供该函数, 具体暂停语音播放的条件由使用者决定。

■ _RESUME

描述:

当调用了 _PAUSE 函数之后, 再调用 _RESUME 函数可以恢复播放。

范例:

```
_PLAY_VOICE 0, 0, 0, 3, 0      ; 播放第一个 voice, 音量为 3  
_CALL_DELAY                      ; 延时一段时间, 在声音播放一段时间后停止  
_PAUSE                            ; 调用暂停函数  
_CALL_DELAY ;  
_RESUME                           ; 恢复播放
```

注: 上述延时函数只是提供一个范例, 语音函数库中并不提供该函数, 具体恢复语音播放的条件由使用者决定。

完整调用 library 的举例

使用播音函数库，需在项目中添加如下档案：

1. Voice_Library_Choice.asm
2. MCUNAME_Voice_Library.lib

■ 【HT66FV130 举例】

```
#INCLUDE HT66FV130.INC
#INCLUDE HT66FV130.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin
ORG 08H
CLR WDT
CLR WDT2
JMP _PLAY_SENTENCE_ISR ;Timer0 interrupt( sentence )
ORG 0CH
CLR WDT
CLR WDT2
JMP _PLAY_VOICE_ISR ;Timer1 interrupt( voice )
ORG 50H
```

Begin:

```
CALL _CLRRAM ;Clear all RAM banks
CALL _SYSTEM_INITIALIZATION ;System initialization
CALL _DAC_RAMP_UP ;Open DAC and do ramp up
_PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
SNZ fVoiceStandBy
JMP $-1 ;Wait play voice finish
_PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
;0100H, volume is 5

SZ fSentencePlaying
JMP $-1 ;Wait play sentence finish
_PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
SZ fSentencePlaying
JMP $-1 ;Wait play sentence finish
CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down
CLR WDT
CLR WDT2
JMP $-2
```

■ 【HT66FV140 举例】

```

#INCLUDE HT66FV140.INC
#INCLUDE HT66FV140.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin
ORG 08H
CLR WDT
CLR WDT2
JMP _PLAY_SENTENCE_ISR ;Timer0 interrupt( sentence )
ORG 0CH
CLR WDT
CLR WDT2
JMP _PLAY_VOICE_ISR ;Timer1 interrupt( voice )
ORG 50H

```

Begin:

```

CALL _CLRRAM ;Clear all RAM banks
CALL _SYSTEM_INITIALIZATION ;System initialization
CALL _DAC_RAMP_UP ;Open DAC and do ramp up
_PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
SNZ fVoiceStandBy
JMP $-1 ;Wait play voice finish
_PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
;0100H, volume is 5

SZ fSentencePlaying
JMP $-1 ;Wait play sentence finish
_PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
SZ fSentencePlaying
JMP $-1 ;Wait play sentence finish
CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down
CLR WDT
CLR WDT2
JMP $-2

```

■ 【HT66FV150 举例】

```

#INCLUDE HT66FV150.INC
#INCLUDE HT66FV150.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin
ORG 08H
CLR WDT
CLR WDT2
JMP _PLAY_SENTENCE_ISR ;Timer0 interrupt( sentence )
ORG 0CH
CLR WDT
CLR WDT2
JMP _PLAY_VOICE_ISR ;Timer1 interrupt( voice )
ORG 50H

```

Begin:

```

CALL _CLRRAM ;Clear all RAM banks
CALL _SYSTEM_INITIALIZATION ;System initialization
CALL _DAC_RAMP_UP ;Open DAC and do ramp up
_PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
SNZ fVoiceStandBy
JMP $-1 ;Wait play voice finish
_PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
;0100H, volume is 5

SZ fSentencePlaying
JMP $-1 ;Wait play sentence finish
_PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
SZ fSentencePlaying
JMP $-1 ;Wait play sentence finish
CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down
CLR WDT
CLR WDT2
JMP $-2

```

■ 【HT66FV160 举例】

```

#INCLUDE HT66FV160.INC
#INCLUDE HT66FV160.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin
ORG 08H
MOV BackupAcc,A
MOV A,PBP
CLR PBP
JMP _PLAY_SENTENCE_ISR ;Timer0 interrupt( sentence )
ORG 0CH
MOV BackupAcc,A
MOV A,PBP
CLR PBP
JMP _PLAY_VOICE_ISR ;Timer1 interrupt( voice )
ORG 50H

```

Begin:

```

CALL _CLRRAM ;Clear all RAM banks
CALL _SYSTEM_INITIALIZATION ;System initialization
CALL _DAC_RAMP_UP ;Open DAC and do ramp up
_PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
CLR WDT
CLR WDT2
SNZ fVoiceStandBy
JMP $-3 ;Wait play voice finish
_PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
;0100H, volume is 5
CLR WDT
CLR WDT2
SZ fSentencePlaying
JMP $-3 ;Wait play sentence finish
_PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
CLR WDT
CLR WDT2
SZ fSentencePlaying
JMP $-3 ;Wait play sentence finish
CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down
CLR WDT
CLR WDT2
JMP $-2

```

■ 【BH67F2262 举例】

```

#INCLUDE    BH67F2262.INC
#INCLUDE    BH67F2262.HED

CODE        .SECTION      AT      0000H      'CODE'
ORG         00H
CLR         WDT
CLR         WDT2
JMP         Begin
ORG         010H
MOV         BackupAcc,A
MOV         A,PBP
CLR         PBP
JMP         _PLAY_SENTENCE_ISR           ;Timer0 interrupt( sentence )
ORG         014H
MOV         BackupAcc,A
MOV         A,PBP
CLR         PBP
JMP         _PLAY_VOICE_ISR            ;Timer1 interrupt( voice )
ORG         50H

Begin:

CALL        _CLRRAM                   ;Clear all RAM banks
CALL        _SYSTEM_INITIALIZATION    ;System initialization
CALL        _DAC_RAMP_UP              ;Open DAC and do ramp up

_PLAY_VOICE 0,0,0,5,0                 ;Play the first audio, volume is 5
CLR         WDT
CLR         WDT2
SNZ         fVoiceStandBy
JMP         $-3                      ;Wait play voice finish
_PLAY_SENTENCE 01H,00H,0,5,0          ;Play the sentence whose address is
                                      ;0100H, volume is 5

CLR         WDT
CLR         WDT2
SZ          fSentencePlaying
JMP         $-3                      ;Wait play sentence finish
_PLAY_SENTENCE_INDEX 0,1,0,5,0       ;Play the sentence which index is 1,
                                      ;volume is 5

CLR         WDT
CLR         WDT2
SZ          fSentencePlaying
JMP         $-3                      ;Wait play sentence finish
CALL        _DAC_RAMP_DOWN            ;Close DAC and do ramp down
CLR         WDT
CLR         WDT2
JMP         $-2

```

■ 【HT45F67 举例】

```

#INCLUDE HT45F67.INC
#INCLUDE HT45F67.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin
ORG 10H
MOV BackupAcc, A
MOV A,BP
CLR BP
JMP _PLAY_VOICE_ISR ;Timer2 interrupt( voice )
ORG 14H
MOV BackupAcc, A
MOV A,BP
CLR BP
JMP _PLAY_SENTENCE_ISR ;Timer1 interrupt( sentence )
ORG 30H

```

Begin:

```

CALL _CLRRAM ;Clear all RAM banks
CALL _SYSTEM_INITIALIZATION ;System initialization
CALL _DAC_RAMP_UP ;Open DAC and do ramp up
_PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
CLR WDT
CLR WDT2
SNZ fVoiceStandBy
JMP $-3 ;Wait play voice finish
_PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
;0100H, volume is 5
CLR WDT
CLR WDT2
SZ fSentencePlaying
JMP $-3 ;Wait play sentence finish
_PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
CLR WDT
CLR WDT2
SZ fSentencePlaying
JMP $-3 ;Wait play sentence finish
CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down
CLR WDT
CLR WDT2
JMP $-2

```

■ 【HT45F65 举例】

```

#INCLUDE HT45F65.INC
#INCLUDE HT45F65.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin
ORG 10H
MOV BackupAcc, A
MOV A,BP
CLR BP
JMP _PLAY_SENTENCE_ISR ;Timer1 interrupt( sentence )
ORG 18H
MOV BackupAcc, A
MOV A,BP
CLR BP
JMP _PLAY_VOICE_ISR ;Timer2 interrupt( voice )
ORG 30H

```

Begin:

```

CALL _CLRRAM ;Clear all RAM banks
CALL _SYSTEM_INITIALIZATION ;System initialization
CALL _DAC_RAMP_UP ;Open DAC and do ramp up
_PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
CLR WDT
CLR WDT2
SNZ fVoiceStandBy
JMP $-3 ;Wait play voice finish
_PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
;0100H, volume is 5
CLR WDT
CLR WDT2
SZ fSentencePlaying
JMP $-3 ;Wait play sentence finish
_PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
CLR WDT
CLR WDT2
SZ fSentencePlaying
JMP $-3 ;Wait play sentence finish
CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down
CLR WDT
CLR WDT2
JMP $-2

```

■ 【HT45F3W 举例】

```

#INCLUDE HT45F3W.INC
#INCLUDE HT45F3W.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin
ORG 0CH
MOV BackupAcc, A
MOV A,BP
CLR BP
JMP _PLAY_SENTENCE_ISR ;Timer1 interrupt( sentence );
ORG 10H
MOV BackupAcc, A
MOV A,BP
CLR BP
JMP _PLAY_VOICE_ISR ;Timer2 interrupt( voice )
ORG 30H

```

Begin:

```

CALL _CLRRAM ;Clear all RAM banks
CALL _SYSTEM_INITIALIZATION ;System initialization
CALL _DAC_RAMP_UP ;Open DAC and do ramp up
_PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
CLR WDT
CLR WDT2
SNZ fVoiceStandBy
JMP $-3 ;Wait play voice finish
_PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
;0100H, volume is 5
CLR WDT
CLR WDT2
SZ fSentencePlaying
JMP $-3 ;Wait play sentence finish
_PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
CLR WDT
CLR WDT2
SZ fSentencePlaying
JMP $-3 ;Wait play sentence finish
CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down
CLR WDT
CLR WDT2
JMP $-2

```

■ 【HT66F4550 举例】

```

#INCLUDE HT66F4550.INC
#INCLUDE HT66F4550.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin

ORG 0CH
CLR WDT
CLR WDT2
JMP _PLAY_SENTENCE_ISR ;Timer interrupt( sentence )

ORG 10H
CLR WDT
CLR WDT2
JMP _PLAY_VOICE_ISR ;Timer interrupt( voice )

ORG 50H

Begin:
CALL _CLRRAM ;Clear all RAM banks
CALL _SYSTEM_INITIALIZATION ;System initialization

CALL _DAC_RAMP_UP ;Open DAC and do ramp up

_PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
SNZ fVoiceStandBy
JMP $-1 ;Wait play voice finish

_PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
;0100H, volume is 5
SZ fSentencePlaying
JMP $-1 ;Wait play sentence finish

_PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
SZ fSentencePlaying
JMP $-1 ;Wait play sentence finish

CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down

CLR WDT
CLR WDT2
JMP $-2

```

■ 【BA45F5250 举例】

```

#INCLUDE    BA45F5250.INC
#INCLUDE    BA45F5250.HED

CODE        .SECTION    AT    0000H    'CODE'
ORG         00H
CLR          WDT
CLR          WDT2
JMP         Begin

ORG         0CH
CLR          WDT
CLR          WDT2
JMP         _PLAY_SENTENCE_ISR           ;Timer interrupt( sentence )

ORG         10H
CLR          WDT
CLR          WDT2
JMP         _PLAY_VOICE_ISR            ;Timer interrupt( voice )

ORG         50H

Begin:
CALL        _CLRRAM                ;Clear all RAM banks
CALL        _SYSTEM_INITIALIZATION   ;System initialization

CALL        _DAC_RAMP_UP           ;Open DAC and do ramp up

_PLAY_VOICE 0,0,0,5,0             ;Play the first audio, volume is 5
SNZ         fVoiceStandBy
JMP         $-1                   ;Wait play voice finish

_PLAY_SENTENCE 01H,00H,0,5,0      ;Play the sentence whose address is
                                  ;0100H, volume is 5
SZ          fSentencePlaying
JMP         $-1                   ;Wait play sentence finish

_PLAY_SENTENCE_INDEX 0,1,0,5,0    ;Play the first sentence, volume is 5
SZ          fSentencePlaying
JMP         $-1                   ;Wait play sentence finish

CALL        _DAC_RAMP_DOWN         ;Close DAC and do ramp down

CLR          WDT
CLR          WDT2
JMP         $-2

```

■ 【BA45F5260 举例】

```

#INCLUDE    BA45F5260.INC
#INCLUDE    BA45F5260.HED

CODE        .SECTION    AT    0000H    'CODE'
ORG         00H
CLR          WDT
CLR          WDT2
JMP         Begin

ORG         0CH
CLR          WDT
CLR          WDT2
JMP         _PLAY_SENTENCE_ISR           ;Timer interrupt( sentence )

ORG         10H
CLR          WDT
CLR          WDT2
JMP         _PLAY_VOICE_ISR            ;Timer interrupt( voice )

ORG         50H

Begin:
CALL        _CLRRAM                ;Clear all RAM banks
CALL        _SYSTEM_INITIALIZATION   ;System initialization

CALL        _DAC_RAMP_UP           ;Open DAC and do ramp up

_PLAY_VOICE 0,0,0,5,0             ;Play the first audio, volume is 5
SNZ         fVoiceStandBy
JMP         $-1                   ;Wait play voice finish

_PLAY_SENTENCE 01H,00H,0,5,0      ;Play the sentence whose address is
                                  ;0100H, volume is 5
SZ          fSentencePlaying
JMP         $-1                   ;Wait play sentence finish

_PLAY_SENTENCE_INDEX 0,1,0,5,0    ;Play the first sentence, volume is 5
SZ          fSentencePlaying
JMP         $-1                   ;Wait play sentence finish

CALL        _DAC_RAMP_DOWN         ;Close DAC and do ramp down

CLR          WDT
CLR          WDT2
JMP         $-2

```

■ 【HT45F23A 举例】

```

#INCLUDE HT45F23A.INC
#INCLUDE HT45F23A.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin

ORG 0CH
CLR WDT
CLR WDT2
JMP _PLAY_SENTENCE_ISR ;Timer0 interrupt( sentence )

ORG 10H
CLR WDT
CLR WDT2
JMP _PLAY_VOICE_ISR ;Timer1 interrupt( voice )

ORG 20H

Begin:
    CALL _CLRRAM ;Clear all RAM banks
    CALL _SYSTEM_INITIALIZATION ;System initialization
    CALL _DAC_RAMP_UP ;Open DAC and do ramp up
    _PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
    SNZ fVoiceStandBy
    JMP $-1 ;Wait play voice finish
    _PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
                                ;0100H, volume is 5
    SZ fSentencePlaying
    JMP $-1 ;Wait play sentence finish
    _PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
    SZ fSentencePlaying
    JMP $-1 ;Wait play sentence finish
    CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down
    CLR WDT
    CLR WDT2
    JMP $-2

```

■ 【HT45F24A 举例】

```

#INCLUDE HT45F24A.INC
#INCLUDE HT45F24A.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin

ORG 0CH
CLR WDT
CLR WDT2
JMP _PLAY_SENTENCE_ISR ;Timer0 interrupt( sentence )

ORG 10H
CLR WDT
CLR WDT2
JMP _PLAY_VOICE_ISR ;Timer1 interrupt( voice )

ORG 20H

```

Begin:

```

CALL _CLRRAM ;Clear all RAM banks
CALL _SYSTEM_INITIALIZATION ;System initialization
CALL _DAC_RAMP_UP ;Open DAC and do ramp up
_PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
SNZ fVoiceStandBy
JMP $-1 ;Wait play voice finish
_PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
;0100H, volume is 5

SZ fSentencePlaying
JMP $-1 ;Wait play sentence finish
_PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
SZ fSentencePlaying
JMP $-1 ;Wait play sentence finish
CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down
CLR WDT
CLR WDT2
JMP $-2

```

■ 【HT83F02 举例】

```

#INCLUDE HT83F02.INC
#INCLUDE HT83F02.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin

ORG 08H
CLR WDT
CLR WDT2
JMP _PLAY_SENTENCE_ISR ;Timer0 interrupt( sentence )

ORG 0CH
CLR WDT
CLR WDT2
JMP _PLAY_VOICE_ISR ;Timer1 interrupt( voice )

ORG 20H

Begin:
    CALL _CLRRAM ;Clear all RAM banks
    CALL _SYSTEM_INITIALIZATION ;System initialization
    CALL _DAC_RAMP_UP ;Open DAC and do ramp up
    _PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
    SNZ fVoiceStandBy
    JMP $-1 ;Wait play voice finish
    _PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
                                ;0100H, volume is 5
    SZ fSentencePlaying
    JMP $-1 ;Wait play sentence finish
    _PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
    SZ fSentencePlaying
    JMP $-1 ;Wait play sentence finish
    CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down
    CLR WDT
    CLR WDT2
    JMP $-2

```

■ 【HT86B03 举例】适用于 HT86B10、HT86B20、HT86B30

```

#INCLUDE HT86B03.INC
#INCLUDE HT86B03.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin

ORG 08H
CLR WDT
CLR WDT2
JMP _PLAY_SENTENCE_ISR ;Timer0 interrupt( sentence )

ORG 0CH
CLR WDT
CLR WDT2
JMP _PLAY_VOICE_ISR ;Timer1 interrupt( voice )

ORG 20H

Begin:
    CALL _CLRRAM ;Clear all RAM banks
    CALL _SYSTEM_INITIALIZATION ;System initialization
    CALL _DAC_RAMP_UP ;Open DAC and do ramp up
    _PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
    SNZ fVoiceStandBy
    JMP $-1 ;Wait play voice finish
    _PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
                                ;0100H, volume is 5
    SZ fSentencePlaying
    JMP $-1 ;Wait play sentence finish
    _PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
    SZ fSentencePlaying
    JMP $-1 ;Wait play sentence finish
    CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down
    CLR WDT
    CLR WDT2
    JMP $-2

```

■ 【HT86B40 举例】适用于 HT86B50、HT86B60、HT86B70、HT86B80、HT86B90

```

#INCLUDE HT86B40.INC
#INCLUDE HT86B40.HED

CODE .SECTION AT 0000H 'CODE'
ORG 00H
CLR WDT
CLR WDT2
JMP Begin
ORG 08H
MOV BackupAcc, A
MOV A,BP
CLR BP
JMP _PLAY_SENTENCE_ISR ;Timer0 interrupt( sentence )
ORG 10H
MOV BackupAcc, A
MOV A,BP
CLR BP
JMP _PLAY_VOICE_ISR ;Timer2 interrupt( voice )
ORG 20H

```

Begin:

```

CALL _CLRRAM ;Clear all RAM banks
CALL _SYSTEM_INITIALIZATION ;System initialization
CALL _DAC_RAMP_UP ;Open DAC and do ramp up
_PLAY_VOICE 0,0,0,5,0 ;Play the first audio, volume is 5
CLR WDT
CLR WDT2
SNZ fVoiceStandBy
JMP $-3 ;Wait play voice finish
_PLAY_SENTENCE 01H,00H,0,5,0 ;Play the sentence whose address is
;0100H, volume is 5
CLR WDT
CLR WDT2
SZ fSentencePlaying
JMP $-3 ;Wait play sentence finish
_PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the first sentence, volume is 5
CLR WDT
CLR WDT2
SZ fSentencePlaying
JMP $-3 ;Wait play sentence finish
CALL _DAC_RAMP_DOWN ;Close DAC and do ramp down
CLR WDT
CLR WDT2
JMP $-2

```

C 语言调用 Voice library

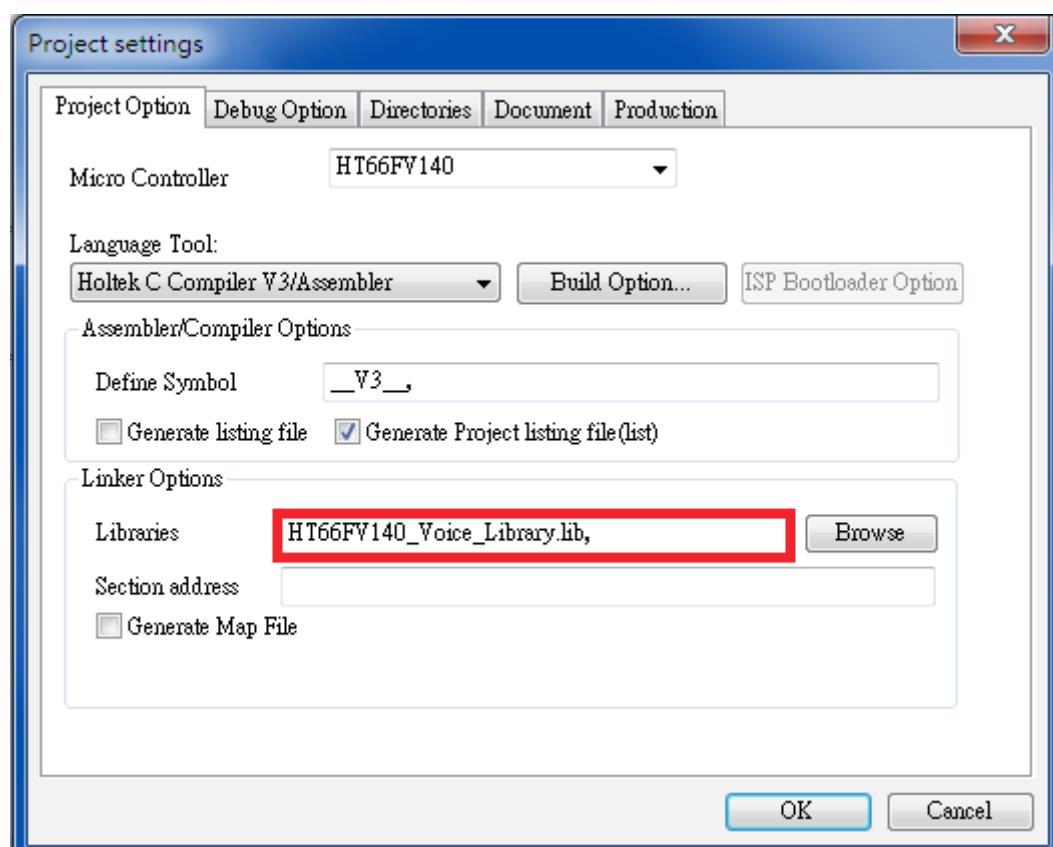
目的说明

主要介紹客戶如何使用 C language 呼叫调用 Voice library 提供的函数.

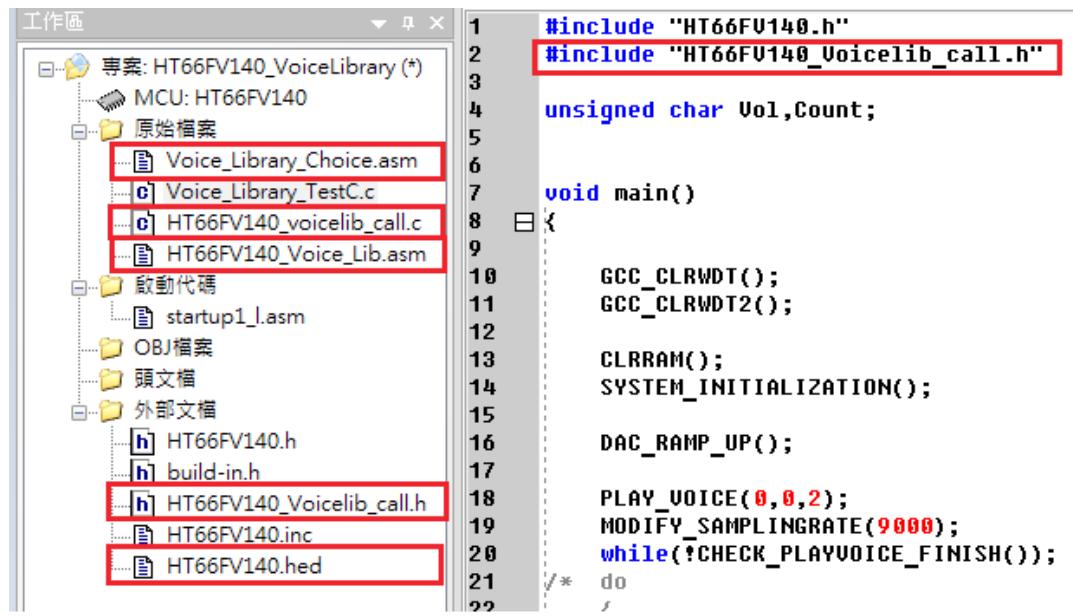
如何使用

创建 C 工程后:

- 添加库文件



- 添加库函数所需的相关文件如下：
Voice_Library_Choice.asm, XX.hed, XX_Voicelib_call.c, XX_Voicelib_call.h , XX_Voice_Lib.asm



The screenshot shows the Holtek Voice MCU Workshop interface. On the left, the '工作區' (Work Area) panel displays the project structure for '專案: HT66FV140_VoiceLibrary (*)' under 'MCU: HT66FV140'. The '原始檔案' (Source Files) section contains several files highlighted with red boxes: 'Voice_Library_Choice.asm', 'Voice_Library_TestC.c', 'HT66FV140_voicelib_call.c', 'HT66FV140_Voice_Lib.asm', 'HT66FV140.h', 'build-in.h', 'HT66FV140_Voicelib_call.h', 'HT66FV140.inc', and 'HT66FV140.hed'. To the right, the code editor shows the main assembly code:

```

1 #include "HT66FV140.h"
2 #include "HT66FV140_Voicelib_call.h"
3
4 unsigned char Vol,Count;
5
6
7 void main()
8 {
9
10    GCC_CLRWDT();
11    GCC_CLRWDT2();
12
13    CLRRAM();
14    SYSTEM_INITIALIZATION();
15
16    DAC_RAMP_UP();
17
18    PLAY_VOICE(0,0,2);
19    MODIFY_SAMPLINGRATE(9000);
20    while(!CHECK_PLAYVOICE_FINISH());
21
22    /* do
23 */

```

- 添加 XX_voicelib_call.h 於工程目录中，並在要调用的 C 文件里
“#include “XX_voicelib_call.h” ”，其中就是被所有 C call 函数的声明
■ [参考 C 的举例建立工程 \(具体工程举例\)](#)。

所提供函数

■ void CLRRAM();

描述:

ram bank0, bank1, 00h~FFh 均被清为 0.

举例:

```
CLRRAM( );
```

■ void SYSTEM_INITIALIZATION();

描述:

系統 Fsys 设置, SPI 介面设置, timer 初始化等

举例:

```
CLRRAM( );
```

```
SYSTEM_INITIALIZATION( );
```

■ void DAC_RAMP_UP();

描述:

DA 使能, 在 PLAY_VOICE(), _PLAY_SENTENCE(), PLAY_SENTENCE_INDEX(); 执行之前需先执行该函数

举例:

```
DAC_RAMP_UP( );
```

```
PLAY_VOICE( );
```

■ void DAC_RAMP_DOWN();

描述:

DA 除能, 在 PLAY_VOICE(), _PLAY_SENTENCE(), PLAY_SENTENCE_INDEX(); 执行之后, 执行该函数。可以减少不必要的耗电流。

举例:

```
PLAY_VOICE( );
```

```
DAC_RAMP_DOWN( );
```

■ void STOP_PLAY();

描述:

停止放音, 在任何时刻都可以直接调用该函数。

举例:

```
STOP_PLAY( );
```

■ **Void VOLUME_CHOICE(unsigned char vol);**

描述:

User 设置音量大小，参照规格书上的音量值，直接写入
参数:

Vol: 规格书中的音量值

举例:

```
VOLUME_CHOICE(0x67);
```

■ **void PLAY_VOICE(unsigned char Voicenumh,unsigned char Voicenuml,unsigned char vol_voice);**

描述:

play_voice

参数:

Voicenumh:Voice NUM high byte

Voicenuml :Voice NUM low byte

vol_voice :Voice 音量选择

举例:

播放第 1 首音原始文件 (注: UI 介面上的音源序号从“0”开始计数，并不是“1”)

采用的音量为增益 =6DB (参照规格书为 0x0C)

则: DAC_RAMP_UP();

```
PLAY_VOICE(0,0,0xc);
```

注: 其中参数 Voicenumh、Voicenuml、vol_voice 可以为变数形式, PLAY_VOICE(A,B,C);

■ **Void PLAY_SENTENCE(unsigned char SentenceAddrH, unsigned char SentenceAddrL,
unsigned char vol_sentence)**

描述:

play_sentence

参数:

SentenceAddrH : SentenceAddr high byte

SentenceAddrL : SentenceAddr low byte

vol_sentence : Sentence 音量选择

注: SentenceAddr 定义: 二代 VOICE 平台, UI 介面上的被选择为
“play_voice” Function 的地址。

见平台 SW 产出的 Demo_key_mapping.h 文件。

举例:

播放第 1 首 sentence 档, 假定其地址为“0100h”

采用的音量为增益 =6DB (参照规格书为 0x0C)

则: DAC_RAMP_UP();

```
PLAY_SENTENCE (0x01,0x00,0x0c);
```

注: 其中参数 SentenceAddrH、SentenceAddrL、vol_sentence 可以为变数形式如:

```
PLAY_SENTENCE(A,B,C);
```

■ Void PLAY_SENTENCE_INDEX (unsigned char Reserve1, unsigned char SentenceIndex,
unsigned char vol_sentence)

描述：播放第几首 Sentence

参数：

Reserve1: no use

SentenceIndex: Sentence number (1~255)

vol_Sentence : Sentence 音量选择

举例：播放第 1 首 sentence 文件，采用的音量为 7

则：DAC_RAMP_UP();

PLAY_SENTENCE_INDEX (0, 1, 7);

注：Sentence Index 在语音排程列表中得到，如下图所示：

語音排程列表		
觸發來源	觸發來源名稱	功能
Sentence 1	Sentence 1	Play
Sentence 2	Sentence 2	Play

■ **unsigned char CHECK_PLAYVOICE_FINISH();**

描述:

用于判断 play_voice, play_sentence 结束否

返回值:

若为 1: play finish

0: play unfinished

举例:

```
do
{
    GCC_CLRWDT();
    GCC_CLRWDT2();
}while(!CHECK_PLAYVOICE_FINISH());
```

■ **void MODIFY_SAMPLINGRATE(unsigned int mSamplingRate)**

描述:

改变当前播放语音的采样率

参数:

mSamplingRate: 指定的采样率值 (Hz)

举例:

改变当前播放语音的采样率为 11025Hz

则:

MODIFY_SAMPLINGRATE(11025);

PLAY_VOICE(0,0,7);

注: 要使用该函数, 需将 USE_MODIFY_SAMPLINGRATE 设为 1, 该参数位于 xxx_voicelib_call.h

```
#define USE_MODIFY_SAMPLINGRATE 1 // =1:use MODIFY_SAMPLINGRATE() function
```

■ **void ENABLE_VDDIO ();**

描述 :

使能 MCU 的 VDDIO 功能, 使 SPI 引脚的电压来源于 VDDIO; 在 CLRRAM () 之前需调用此函数。

范例 :

ENABLE_VDDIO ();

CLRRAM ();

SYSTEM_INITIALIZATION ();

■ void PAUSE();

描述:

当一个 Voice 或者 Sentence 正在播放时，调用此函数可以暂停播放。

范例:

```
PLAY_VOICE(0, 0, 3);           // 播放第一个 voice, 音量为 3
DELAY();                      // 延时一段时间，在声音播放一段时间后暂停
PAUSE();                      // 调用暂停函数
```

注：上述延时函数只是提供一个范例，语言函数库中并不提供该函数，具体暂停语音播放的条件由使用者决定。

■ void RESUME();

描述:

当调用了 PAUSE() 函数之后，再调用 RESUME() 函数可以恢复播放。

范例:

```
PLAY_VOICE(0, 0, 3);           // 播放第一个 voice, 音量为 3
DELAY();                      // 延时一段时间，在声音播放一段时间后暂停
PAUSE();                      // 调用暂停函数
DELAY();                      // 恢复播放
RESUME();
```

注：上述延时函数只是提供一个范例，语音函数库中并不提供该函数，具体恢复语音播放的条件由使用者决定。

具体过程应用举例

使用播音函数库，需在项目中添加如下档案：

1. Voice_Library_Choice.asm
2. MCUNAME_Voice_Library.lib
3. MCUNAME.hed
4. MCUNAME_Voice_Lib.asm
5. MCUNAME_Voicelib_call.c
6. MCUNAME_Voicelib_call.h

■ 【HT66FV130 举例】

```
#include "HT66FV130.h"
#include "HT66FV130_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();

    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                        //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}
```

■ 【HT66FV140 举例】

```

#include "HT66FV140.h"
#include "HT66FV140_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();

    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);           //Play the first sentence, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}

```

■ 【HT66FV150 举例】

```
#include "HT66FV150.h"
#include "HT66FV150_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();

    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}
```

■ 【HT66FV160 举例】

```

#include "HT66FV160.h"
#include "HT66FV160_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();

    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}

```

■ 【BH67F2262 举例】

```

#include "BH67F2262.h"
#include "BH67F2262_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();

    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address
                                            //is 0100H, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}

```

■ 【HT45F67 举例】

```

#include "HT45F67.h"
#include "HT45F67_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();
    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}

```

■ 【HT45F65 举例】

```

#include "HT45F65.h"
#include "HT45F65_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();
    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}

```

■ 【HT45F3W 举例】

```

#include "HT45F3W.h"
#include "HT45F3W_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();
    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}

```

■ 【HT66F4550 举例】

```

#include "HT66F4550.h"
#include "HT66F4550_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();

    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}

```

■ 【BA45F5250 举例】

```

#include "BA45F5250.h"
#include "BA45F5250_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();

    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}

```

■ 【BA45F5260 举例】

```

#include "BA45F5260.h"
#include "BA45F5260_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();

    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}

```

■ 【HT45F23A 举例】

```

#include "HT45F23A.h"
#include "HT45F23A_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();

    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}

```

■ 【HT45F24A 举例】

```
#include "HT45F24A.h"
#include "HT45F24A_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();

    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}
```

■ 【HT83F02 举例】

```

#include "HT83F02.h"
#include "HT83F02_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();

    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                       //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}

```

■ 【HT86B03 举例】适用于 HT86B10、HT86B20、HT86B30

```
#include "HT86B03.h"
#include "HT86B03_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();

    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization

    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                        //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}
```

■ 【HT86B40 举例】适用于 HT86B50、HT86B60、HT86B70、HT86B80、HT86B90

```

#include "HT86B40.h"
#include "HT86B40_voicelib_call.h"

void main()
{
    GCC_CLRWDT();
    GCC_CLRWDT2();
    CLRRAM();                                //Clear all RAM banks
    SYSTEM_INITIALIZATION();                  //System initialization
    DAC_RAMP_UP();                           //Open DAC and do ramp up

    PLAY_VOICE(0,0,5);                      //Play the first audio, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish

    PLAY_SENTENCE(0x01,0x00,5);             //Play the sentence whose address is
                                            //0100H, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    PLAY_SENTENCE_INDEX(0,1,5);            //Play the first sentence, volume is 5
    do
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();

    }while(!CHECK_PLAYVOICE_FINISH()); //Wait play sentence finish

    DAC_RAMP_DOWN();                      //Close DAC and do ramp down

    while(1)
    {
        GCC_CLRWDT();
        GCC_CLRWDT2();
    }
}

```

3 Voice Library 的建立信息及对应仿真器

HT66FV130

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8	HT-PCM16
PROM(Word)	584/2048(27%)	241/2048(11%)	51/2048(2%)	316/2048(15%)	17/2048(1%)
RAM(Byte)			37/128(28%)		
压缩译码数组固定在 PROM 中存储地址		500H-578H	没有译码数组	600H-6FDH 700H-701H	没有译码数组
其它程式固定在 PROM 中存储地址	57EH-582H 583H-58CH	579H-57DH		6FEH-6FFFH 702H-703H	
Stack (层)			2		
寄存器使用记录	SPI1: SPIC0、SPIC1、SPID D/A: USVC、DAH、DAL Timer: PTM0C0、PTM0C1、PTM0AL、PTM0AH、CTM0C0、CTM0C1、 CTM0AL、CTM0AH General: ACC、MP1、IAR1、TBLP、TBLH、TBHP、PCL、STATUS I/O: PCS1、PCPU、PBS0				

注: 1. 用户代码不能占用已经固定存储译码数组的空间
2. 计算所耗用 PROM 空间 : Default + 所选取之压缩模式 (可支持混合压缩模式)

■ MCU 各功能模块使用说明:

- SPI1 应用为对外部 flash 的控制 (使用引脚: SCS、SCK、MISO、MOSI)
- PTM0 定时器中断应用为 play voice (中断入口地址为: 0CH)
- CTM0 定时器中断应用为 play sentence (中断入口地址: 08H)
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: AUD、AUDIN)
- 功率放大器模块 (使用引脚: SP+、SP-)

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	26
_PLAY_SENTENCE	25
_PLAY_SENTENCE_INDEX	30
_VOLUME	19
_MODIFY_SAMPLINGRATE	11
_ENABLE_VDDIO	3
_PAUSE	3
_RESUME	3
_SLEEP_INIT	13

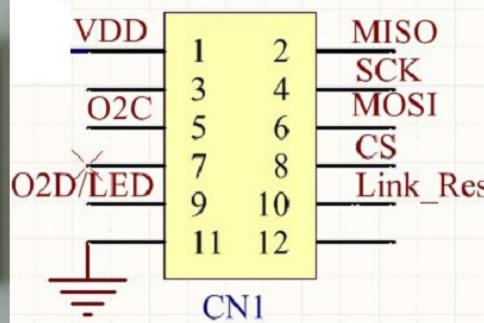
■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示：

系统频率 压缩模式	8MHz	12MHz	16MHz
HT-ADPCM4	13kHz	20kHz	27kHz
HT-PCM12	12kHz	18kHz	24kHz
HT-uPCM8	11kHz	17kHz	22kHz
HT-PCM16	13kHz	19kHz	26kHz

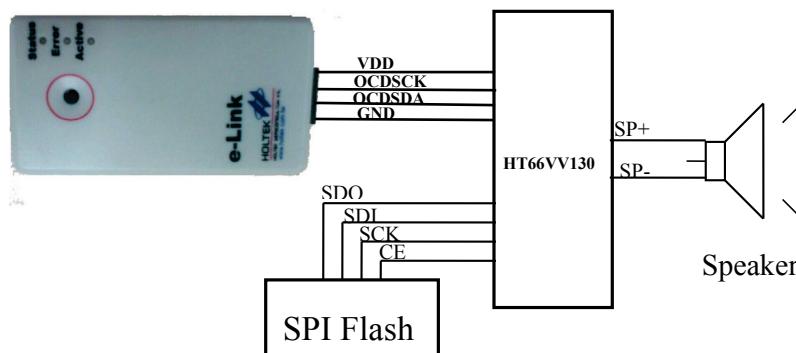
■ 仿真器及连线

这颗 MCU 所使用的是 e-Link 仿真器和 EV 芯片 HT66VV130 进行仿真调试，除此之外还需客户自行搭建 SPI Flash。

- e-Link 引脚图对应如下所示：



- HT66VV130 的 VDD、GND、OCDSCK、OCDSDA 引脚分别对应连 e-Link。



注：SPI flash 连线及烧录请看“[单独将 DAT 文件烧录到 flash 的接线方法](#)”章节。

HT66FV140

■ 资源使用信息表

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8	HT-PCM16
PROM(Word)	647/4096 (15%)	241/4096 (6%)	51/4096 (1%)	316/4096 (8%)	17/4096 (1%)
RAM(Byte)			37/256(14%)		
压缩译码数组固定在 PROM 中存储地址		500H~578H	没有译码数组	600H~6FDH 700H~701H	没有译码数组
其它程序固定在 PROM 中存储地址	57EH~582H 583H~58CH	579H~57DH		6FEH~6FFH 702H~703H	
Stack (层)			2		
寄存器使用记录	SPI1: SPIC0、SPIC1、SPID D/A: USVC、DAH、DAL Timer: PTM0C0、PTM0C1、PTM0AL、PTM0AH、CTM0C0、CTM0C1、 CTM0AL、CTM0AH General: ACC、MP1、IAR1、TBLP、TBLH、TBHP、PCL、STATUS I/O: PCS1、PCPU、PBS0				

注：1. 用户代码不能占用已经固定存储译码数组的空间
2. 计算所耗用 PROM 空间 : Default + 所选取之压缩模式 (可支持混合压缩模式)

■ MCU 各功能模块使用说明：

- SPI1 应用为对外部 flash 的控制 (使用引脚：SCS、SCK、MISO、MOSI)
- PTM0 定时器中断应用为 play voice (中断入口地址为：0CH)
- CTM0 定时器中断应用为 play sentence (中断入口地址：08H)
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚：AUD、AUDIN)
- 功率放大器模块 (使用引脚：SP+、SP-)
- 实现了对 RAM 的 BANK0 区进行优化 (BANK0: 20/128(15%); BANK1: 17/128(13%))

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表：

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	28
_PLAY_SENTENCE	27
_PLAY_SENTENCE_INDEX	36
_VOLUME	19
_MODIFY_SAMPLINGRATE	11
_ENABLE_VDDIO	3
_PAUSE	3
_RESUME	3
_SLEEP_INIT	13

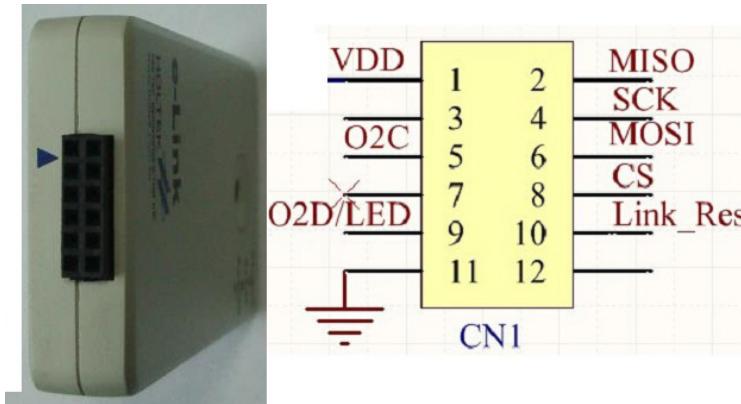
■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示：

系统频率 压缩模式	8MHz	12MHz	16MHz
HT-ADPCM4	13kHz	20kHz	27kHz
HT-PCM12	12kHz	18kHz	24kHz
HT-uPCM8	11kHz	17kHz	22kHz
HT-PCM16	13kHz	19kHz	26kHz

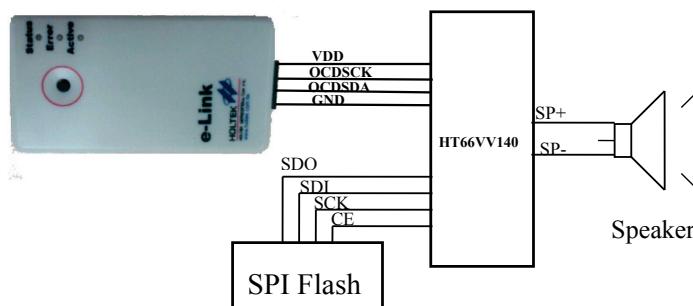
■ 仿真器及连线

这颗 MCU 所使用的是 e-Link 仿真器和 EV 芯片 HT66VV140 进行仿真调试，除此之外还需客户自行搭建 SPI Flash。

- e-Link 引脚图对应如下所示：



- HT66VV140 的 VDD、GND、OCDSCS、OCDSDA 引脚分别对应连 e-Link。



注：SPI flash 连线及烧录请看“[单独将 DAT 文件烧录到 flash 的接线方法](#)”章节。

HT66FV150

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8	HT-PCM16
PROM(Word)	646/8192 (7%)	241/8192 (3%)	51/8192 (1%)	316/8192 (4%)	17/8192 (1%)
RAM(Byte)			37/512(7%)		
压缩译码数组固定在 PROM 中存储地址		500H–578H	没有译码数组	600H–6FDH 700H–701H	没有译码数组
其它程序固定在 PROM 中存储地址	57EH–582H 583H–58CH	579H–57DH		6FEH–6FFFH 702H–703H	
Stack (层)			2		
寄存器使用记录	SPI1: SPIC0、SPIC1、SPID D/A: USVC、DAH、DAL Timer: PTM0C0、PTM0C1、PTM0AL、PTM0AH、CTM0C0、CTM0C1、 CTM0AL、CTM0AH General: ACC、MP1、IAR1、TBLP、TBLH、TBHP、PCL、STATUS I/O: PCS1、PCPU、PBS0				

注: 1. 用户代码不能占用已经固定存储译码数组的空间
2. 计算所耗用 PROM 空间 : Default + 所选取之压缩模式 (可支持混合压缩模式)

■ MCU 各功能模块使用说明:

- SPI1 应用为对外部 flash 的控制 (使用引脚: SCS、SCK、MISO、MOSI)
- PTM0 定时器中断应用为 play voice (中断入口地址为: 0CH)
- CTM0 定时器中断应用为 play sentence (中断入口地址: 08H)
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: AUD、AUDIN)
- 功率放大器模块 (使用引脚: SP+、SP-)
- 实现了对 RAM 的 BANK0 区进行优化 (BANK0: 20/128(15%); BANK1: 17/128(13%))

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	28
_PLAY_SENTENCE	27
_PLAY_SENTENCE_INDEX	36
_VOLUME	19
_MODIFY_SAMPLINGRATE	11
_ENABLE_VDDIO	3
_PAUSE	3
_RESUME	3
_SLEEP_INIT	17

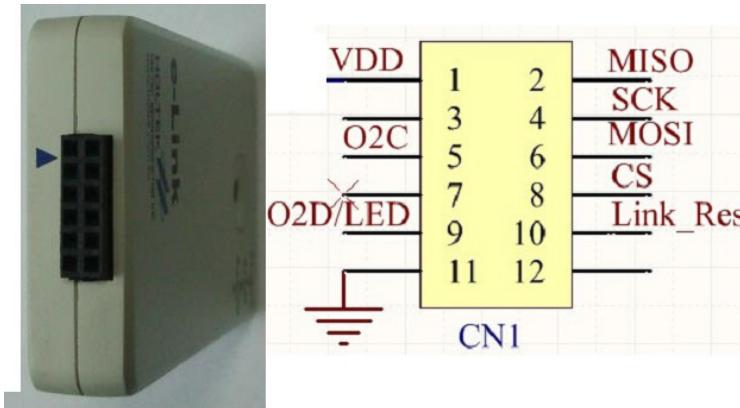
■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示：

系统频率 压缩模式	8MHz	12MHz	16MHz
HT-ADPCM4	13kHz	20kHz	27kHz
HT-PCM12	12kHz	18kHz	24kHz
HT-uPCM8	11kHz	17kHz	22kHz
HT-PCM16	13kHz	19kHz	26kHz

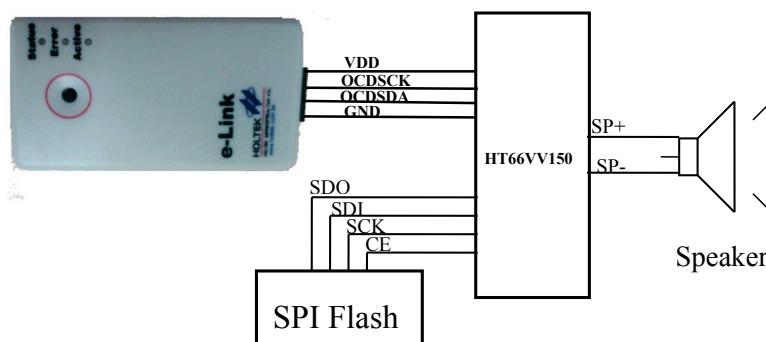
■ 仿真器及连线

这颗 MCU 所使用的是 e-Link 仿真器和 EV 芯片 HT66VV150 进行仿真调试，除此之外还需客户自行搭建 SPI Flash。

- e-Link 引脚图对应如下所示：



- HT66VV150 的 VDD、GND、OCDSCK、OCDSDA 引脚分别对应连 e-Link。



注：SPI flash 连线及烧录请看“[单独将 DAT 文件烧录到 flash 的接线方法](#)”章节。

HT66FV160

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8	HT-PCM16
PROM(Word)	648/16384 (3%)	241/16384 (2%)	51/16384 (1%)	316/16384 (2%)	17/16384 (1%)
RAM(Byte)			38/1024(3%)		
压缩译码数组固定在 PROM 中存储地址		500H-578H	没有译码数组	600H-6FDH 700H-701H	没有译码数组
其它程式固定在 PROM 中存储地址	57EH-582H 583H-58CH	579H-57DH		6FEH-6FFH 702H-703H	700H-704H
Stack(层)			2		
寄存器使用记录	SPI1: SPIC0、SPIC1、SPID D/A: USVC、DAH、DAL Timer: PTM0C0、PTM0C1、PTM0AL、PTM0AH、CTM0C0、CTM0C1、 CTM0AL、CTM0AH General: ACC、MP1、IAR1、TBLP、TBLH、TBHP、PCL、STATUS I/O: PCS1、PCPU、PBS0				

注: 1. 用户代码不能占用已经固定存储译码数组的空间
2. 计算所耗用 PROM 空间 : Default + 所选取之压缩模式 (可支持混合压缩模式)

■ MCU 各功能模块使用说明:

- SPI1 应用为对外部 flash 的控制 (使用引脚: SCS、SCK、MISO、MOSI)
- PTM0 定时器中断应用为 play voice (中断入口地址为: 0CH)
- CTM0 定时器中断应用为 play sentence (中断入口地址: 08H)
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: AUD、AUDIN)
- 功率放大器模块 (使用引脚: SP+、SP-)
- 实现了对 RAM 的 BANK0 区进行优化 (BANK0: 21/128(16%); BANK1: 17/128(13%))

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	28
_PLAY_SENTENCE	27
_PLAY_SENTENCE_INDEX	36
_VOLUME	19
_MODIFY_SAMPLINGRATE	11
_ENABLE_VDDIO	3
_PAUSE	3
_RESUME	3
_SLEEP_INIT	17

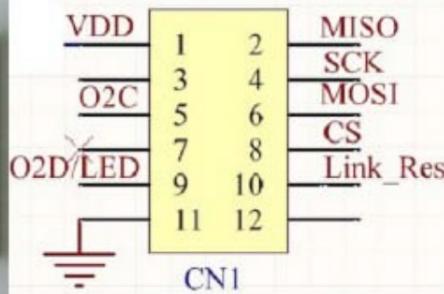
■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示：

系统频率 压缩模式	8MHz	12MHz	16MHz
HT-ADPCM4	13kHz	20kHz	26kHz
HT-PCM12	11kHz	17kHz	23kHz
HT-uPCM8	11kHz	16kHz	22kHz
HT-PCM16	12kHz	19kHz	25kHz

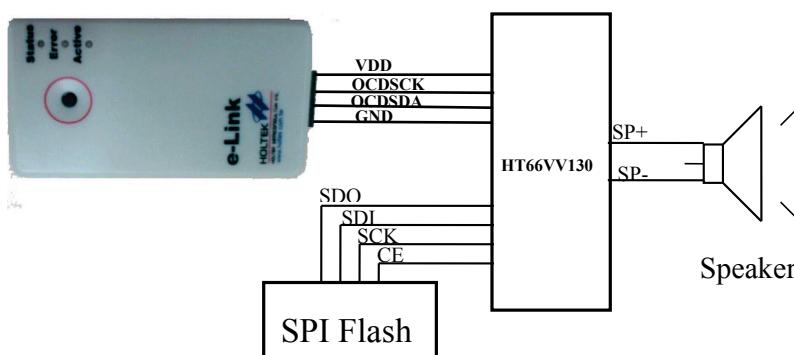
■ 仿真器及连线

这颗 MCU 所使用的是 e-Link 仿真器和 EV 芯片 HT66VV160 进行仿真调试，除此之外还需客户自行搭建 SPI Flash。

- e-Link 引脚图对应如下所示：



- HT66VV130 的 VDD、GND、OCDSCK、OCDSDA 引脚分别对应连 e-Link。



注：SPI flash 连线及烧录请看“单独将 DAT 文件烧录到 flash 的接线方法”章节。

BH67F2262

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	PCM12	UPCM8	PCM16
PROM(Word)	650/16384 (4%)	241/16384 (2%)	51/16384 (1%)	316/16384 (2%)	17/16384 (1%)
RAM(Byte)			38/512(7%)		
压缩译码数组固定在 PROM 中存储地址		500H-578H	没有译码数组	600H-6FDH 700H-701H	没有译码数组
其它程式固定在 PROM 中存储地址	57EH-582H 583H-58CH	579H-57DH		6FEH-6FFH 702H-703H	
Stack(层)			2		
寄存器使用记录	SPI1: SPIC0、SPIC1、SPID PWM: PWMC、USVC、PLADH、PLADL Timer: PTM1C0、PTM1C1、PTM1AL、PTM1AH、PTM0C0、PTM0C1、 PTM0AL、PTM0AH General: ACC、MP1、IAR1、TBLP、TBLH、TBHP、PCL、STATUS I/O: PMPS0、PBPU、PBS1、PGS1				

注：用户代码不能占用已经固定存储译码数组的空间。

■ MCU 各功能模块使用说明:

- SPI1 应用为对外部 flash 的控制 (使用引脚: SPISCSB、SPISCK、SPISDO、SPISDI)
- PTM1 定时器中断应用为 play voice(中断入口地址为: 14H)
- PTM0 定时器中断应用为 play sentence(中断入口地址: 10H)
- PWM 模块应用为对读取到 flash 音频数据进行转换 (使用引脚: PWM1、PWM2)
- 实现了对 RAM 的 BANK0 区进行优化 (BANK0: 21/128(16%); BANK1: 17/128(13%);
BANK2: 0/128(0%); BANK3: 0/128(0%))

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	28
_PLAY_SENTENCE	27
_PLAY_SENTENCE_INDEX	36
_VOLUME	19
_MODIFY_SAMPLINGRATE	16
_ENABLE_VDDIO	6
_PAUSE	4
_RESUME	4

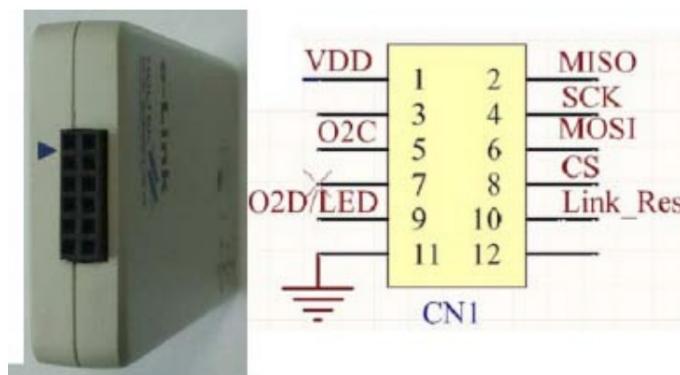
■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示：

系统频率	8MHz	12MHz	16MHz
压缩模式	理论最高采样率		
HT-ADPCM4	13kHz	20kHz	26kHz
HT-PCM12	11kHz	17kHz	23kHz
HT-uPCM8	11kHz	16kHz	22kHz
HT-PCM16	12kHz	19kHz	25kHz

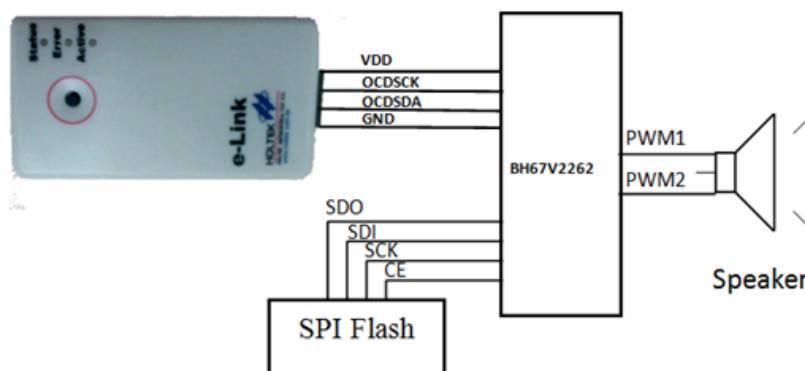
■ 仿真器及连线

这颗 MCU 所使用的是 e-link 仿真器和 EV 芯片 BH67V2262 进行仿真调试，除此之外还需客户自行搭建 SPI Flash。

- e-Link 引脚图对应如下所示：



- BH67V2262 的 VDD、GND、OCDSCK、OCDSDA 引脚分别对应连 e-Link



注：SPI flash 连线及烧录请看“[单独将 DAT 文件烧录到 flash 的接线方法](#)”章节。

HT45F67

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-uPCM8
PROM(Word)	707/32768(2%)	241/32768 (1%)	98/32768 (1%)	316/32768 (1%)
RAM(Byte)		40/512(7%)		
压缩译码数组固定在 PROM 中存储地址		500H-578H	没有译码数组	600H-6FDH 700H-701H
其它程式固定在 PROM 中存储地址	57EH-582H 583H-58AH	579H-57DH	704H-72CH	6FEH-6FFH 702H-703H
Stack (层)		2		
寄存器使用记录	SPI1: SPI1C0、SPI1C1、SPI1D D/A: ADAC、ADAH、ADAL Timer: TM2C0、TM2C1、TM2AL、TM2AH、TM1C0、TM1C1、TM1AL、 TM1AH General: ACC、MP1、IAR1、BP、STATUS、TBLP、TBLH、TBHP I/O: PHPU			

注: 1. 用户代码不能占用已经固定存储译码数组的空间
2. 计算所耗用 PROM 空间: Default + 所选取之压缩模式 (可支持混合压缩模式)

■ MCU 各功能模块使用说明:

- SPI1 应用为对外部 flash 的控制 (使用引脚: SDI1、SDO1、SCK1、SCS1B0)
- Timer2 定时器中断应用为 play voice (中断入口地址为: 10H)
- Timer1 定时器中断应用为 play sentence (中断入口地址: 14H)
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: AUD)
- 实现了对 RAM 的 BANK0 区进行优化 (BANK0: 23/128(17%); BANK1: 17/128(13%))

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	20
_PLAY_SENTENCE	19
_PLAY_SENTENCE_INDEX	24
_VOLUME	9
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

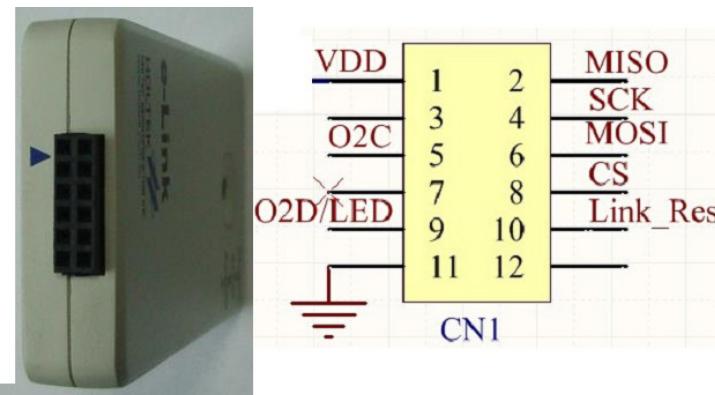
■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示:

系统频率 压缩模式	4MHz	8MHz	12MHz
HT-ADPCM4	6kHz	13kHz	20kHz
HT-PCM12	5kHz	11kHz	17kHz
HT-uPCM8	5kHz	11kHz	16kHz

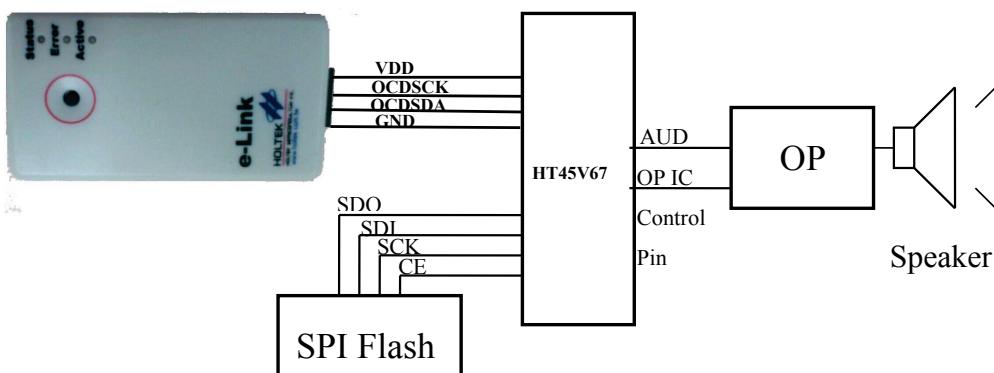
■ 仿真器及连线

这颗 MCU 所使用的是 e-Link 仿真器和 EV 芯片 HT45V67 进行仿真调试，除此之外还需客户自行搭建 SPI Flash 和运放电路模块。

- e-Link 引脚图对应如下所示：

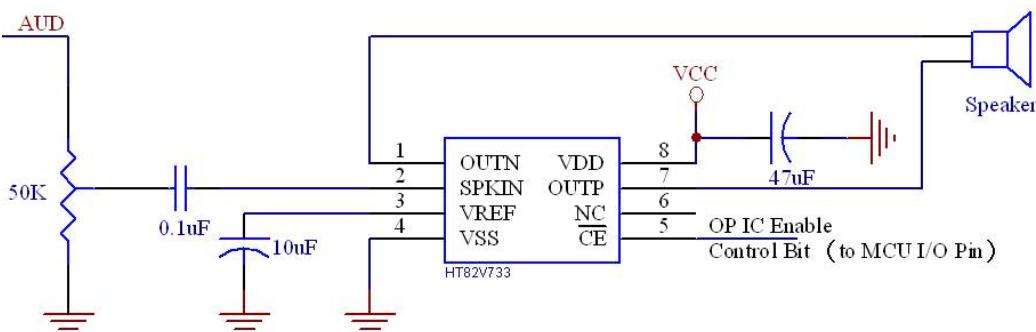


- HT45V67 的 VDD、GND、OCD SCK、OCD SDA 引脚分别对应连 e-Link。



注 1：SPI flash 连线及烧录请看“[单独将 DAT 文件烧录到 flash 的接线方法](#)”章节。

2：应用 HT82V733 搭建运放参考电路如下所示：



HT45F65

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-uPCM8
PROM(Word)	708/8192 (8%)	241/8192 (2%)	98/8192 (1%)	316/8192 (3%)
RAM(Byte)		40/256 (15%)		
压缩译码数组固定在 PROM 中存储地址		500H–578H	没有译码数组	600H–6FDH 700H–701H
其它程式固定在 PROM 中存储地址	57EH–582H 583H–58AH	579H–57DH	704H–72CH	6FEH–6FFH 702H–703H
Stack (层)		2		
寄存器使用记录	SPI1: SPI1C0、SPI1C1、SPI1D D/A: ADAC、ADAH、ADAL Timer: TM2C0、TM2C1、TM2AL、TM2AH、TM1C0、TM1C1、TM1AL、 TM1AH General: ACC、MP1、IAR1、BP、STATUS、TBLP、TBLH、TBHP I/O: PCPU、PDPU			

注: 1. 用户代码不能占用已经固定存储译码数组的空间
2. 计算所耗用 PROM 空间: Default + 所选取之压缩模式 (可支持混合压缩模式)

■ MCU 各功能模块使用说明:

- SPI1 应用为对外部 flash 的控制 (使用引脚: SDI1、SDO1、SCK1、SCS1B0)
- Timer2 定时器中断应用为 play voice (中断入口地址为: 18H)
- Timer1 定时器中断应用为 play sentence (中断入口地址: 10H)
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: AUD)
- 实现了对 RAM 的 BANK0 区进行优化 (BANK0: 23/128(17%); BANK1: 17/128(13%))

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	20
_PLAY_SENTENCE	19
_PLAY_SENTENCE_INDEX	24
_VOLUME	9
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

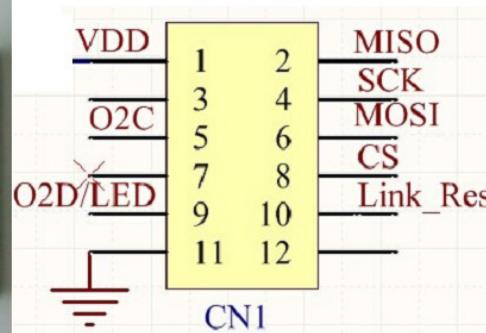
■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示:

系统频率 压缩模式	4MHz	8MHz	12MHz
HT-ADPCM4	6kHz	13kHz	20kHz
HT-PCM12	5kHz	11kHz	17kHz
HT-uPCM8	5kHz	11kHz	16kHz

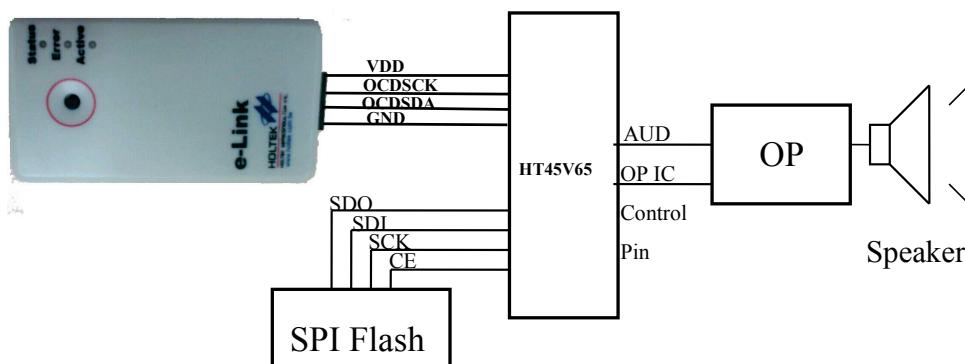
■ 仿真器及连线

这颗 MCU 所使用的是 e-Link 仿真器和 EV 芯片 HT45V65 进行仿真调试，除此之外还需客户自行搭建 SPI Flash 和运放电路模块。

- e-Link 引脚图对应如下所示：

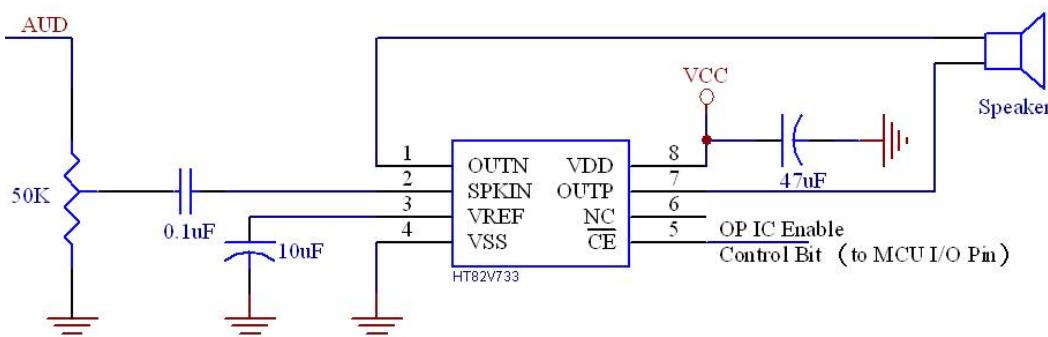


- HT45V65 的 VDD、GND、OCDsck、OCDsda 引脚分别对应连 e-Link。



注 1：SPI flash 连线及烧录请看“单独将 DAT 文件烧录到 flash 的接线方法”章节。

2：应用 HT82V733 搭建运放参考电路如下所示：



HT45F3W

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8
PROM(Word)	717/16384 (4%)	241/16384 (2%)	98/16384 (1%)	316/16384 (2%)
RAM(Byte)		40/512(7%)		
压缩译码数组固定在 PROM 中存储地址		500H-578H	没有译码数组	600H-6FDH 700H-701H
其它程式固定在 PROM 中存储地址	57EH-582H 583H-58AH	579H-57DH	704H-72CH	6FEH-6FFFH 702H-703H
Stack (层)		2		
寄存器使用记录	SPI1: SPI1C0、SPI1C1、SPI1D D/A: VOL、DAH、DAL Timer: TM2C0、TM2C1、TM2AL、TM2AH、TM1C0、TM1C1、TM1AL、 TM1AH General: ACC、MP1、IAR1、BP、STATUS、TBLP、TBLH、TBHP I/O: PBPU			

注: 1. 用户代码不能占用已经固定存储译码数组的空间
2. 计算所耗用 PROM 空间 : Default + 所选取之压缩模式 (可支持混合压缩模式)

■ MCU 各功能模块使用说明:

- SPI1 应用为对外部 flash 的控制 (使用引脚: S1DI、S1DO1、S1CK、S1CS)
- Timer2 定时器中断应用为 play voice (中断入口地址为: 10H)
- Timer1 定时器中断应用为 play sentence (中断入口地址: 0CH)
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: AUD)
- 实现了对 RAM 的 BANK0 区进行优化 (BANK0: 23/128(17%); BANK1: 17/128(13%))

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	20
_PLAY_SENTENCE	19
_PLAY_SENTENCE_INDEX	24
_VOLUME	9
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

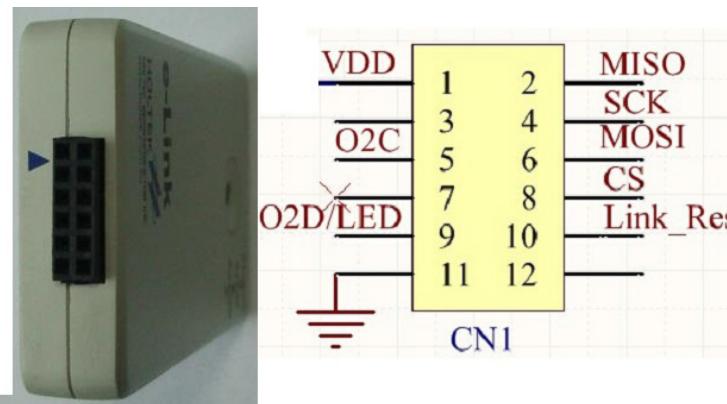
■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示:

系统频率 压缩模式	4MHz	8MHz	12MHz
HT-ADPCM4	6kHz	13kHz	20kHz
HT-PCM12	5kHz	11kHz	17kHz
HT-uPCM8	5kHz	11kHz	16kHz

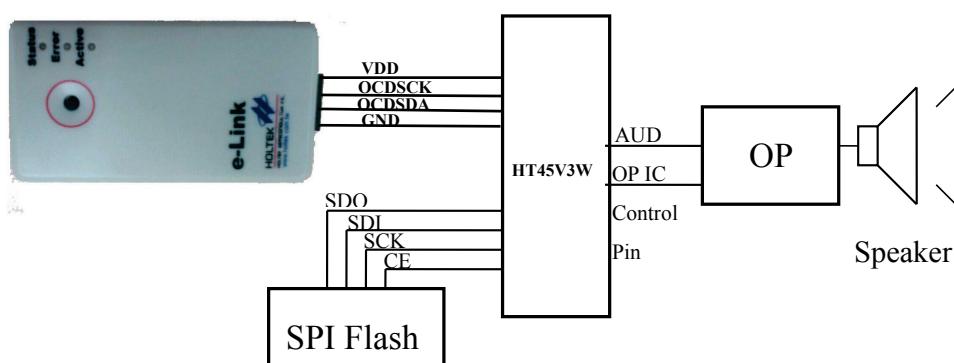
■ 仿真器及连线

这颗 MCU 所使用的是 e-Link 仿真器和 EV 芯片 HT45V3W 进行仿真调试，除此之外还需客户自行搭建 SPI Flash 和运放电路模块。

- e-Link 引脚图对应如下所示：

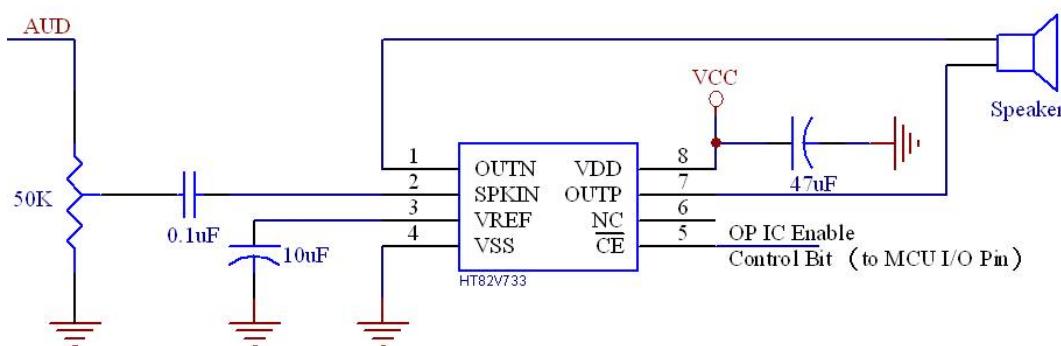


- HT45V3W 的 VDD、GND、OCDSDCK、OCDSDA 引脚分别对应连 e-Link。



注 1：SPI flash 连线及烧录请看“[单独将 DAT 文件烧录到 flash 的接线方法](#)”章节。

2：应用 HT82V733 搭建运放参考电路如下所示：



HT66F4550

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8	HT-PCM16
PROM(Word)	754/8192(9%)	243/8192 (3%)	30/8192 (1%)	316/8192 (4%)	18/8192 (1%)
RAM(Byte)			45/384(9%)		
压缩译码数组固定在 PROM 中存储地址		500H-578H	没有译码 数组	600H-6FDH 700H-701H	没有译码 数组
其它程式固定在 PROM 中存储地址	57EH-582H 583H-58CH	579H-57DH		6FEH-6FFFH 702H-703H	
Stack (层)			2		
寄存器使用记录	SPI1: SPIC0、SPIC1、SPID D/A: DAH、DAL Timer: STM1C0、STM1C1、STM1AL、STM1AH、PTM0C0、PTM0C1、 PTM0AL、PTM0AH General: ACC、MP1、IAR1、TBLP、TBLH、TBHP、PCL、STATUS I/O: PAS0、PAS1、PCS0、IFS、PAPU、PCPU				

注: 用户代码不能占用已经固定存储译码数组的空间

■ MCU 各功能模块使用说明:

- SPI1 应用为对外部 flash 的控制 (使用引脚: SCS(PC0)、SCK(PA4)、SDI(PA1)、SDO(PC1))
- STM0 定时器中断应用为 play voice (中断入口地址为: 10H)
- PTM0 定时器中断应用为 play sentence (中断入口地址: 0CH)
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: DAC0)
- 实现了对 RAM 的 BANK0 区进行优化 (BANK0: 28/128(21%); BANK1: 17/128(13%); BANK2: 0/128(0%))

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	14
_PLAY_SENTENCE	13
_PLAY_SENTENCE_INDEX	30
_VOLUME	4
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

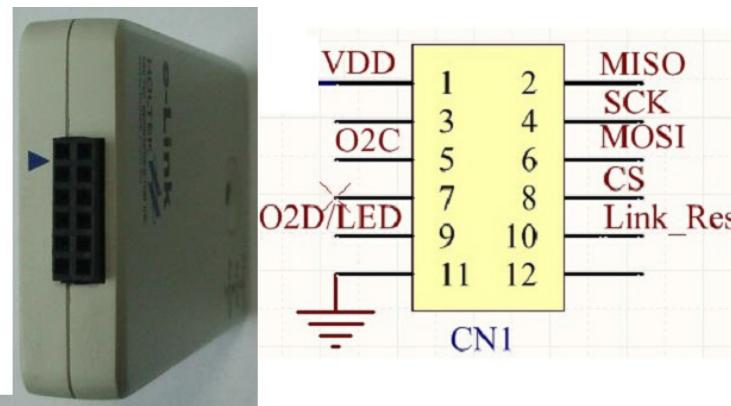
■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示:

系统频率 压缩模式	2MHz	4MHz	8MHz
HT-ADPCM4	2kHz	4kHz	8kHz
HT-PCM12	2kHz	5kHz	11kHz
HT-uPCM8	2kHz	5kHz	10kHz
HT-PCM16	2kHz	5kHz	11kHz

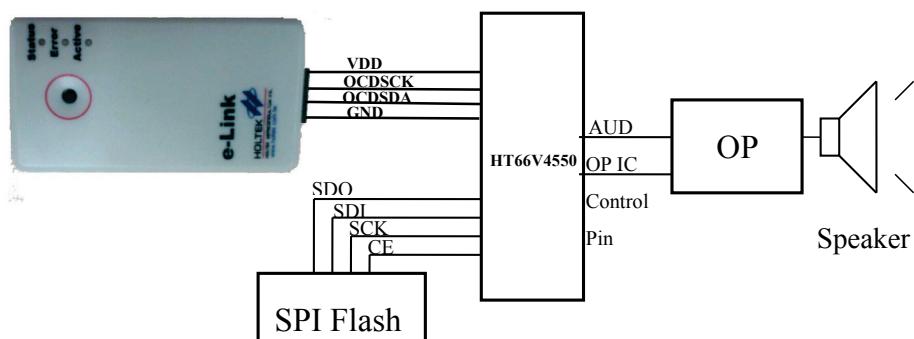
■ 仿真器及连线

这颗 MCU 所使用的是 e-Link 仿真器和 EV 芯片 HT66V4550 进行仿真调试，除此之外还需客户自行搭建 SPI Flash 和运放电路模块。

- e-Link 引脚图对应如下所示：

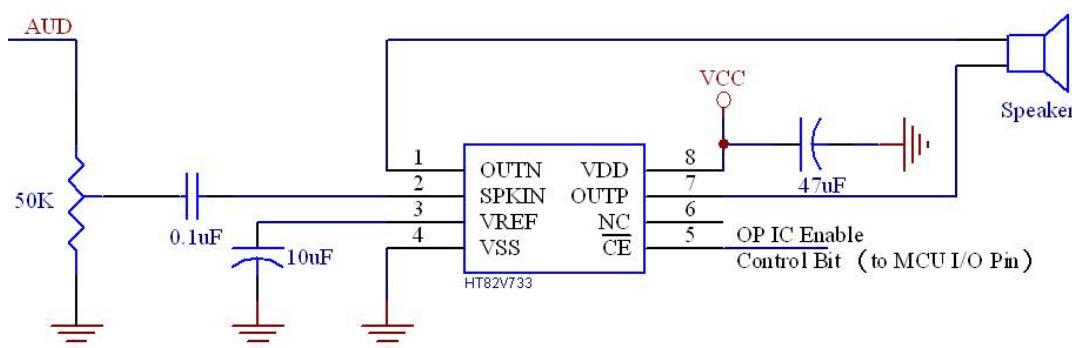


- HT66V4550 的 VDD、GND、OCDSCCK、OCDSDA 引脚分别对应连 e-Link。



注 1：SPI flash 连线及烧录请看“[单独将 DAT 文件烧录到 flash 的接线方法](#)”章节。

2：应用 HT82V733 搭建运放参考电路如下所示：



BA45F5250

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT- ADPCM4	PCM12	UPCM8	PCM16
PROM(Word)	745/8192 (9%)	243/8192 (3%)	30/8192 (1%)	316/8192 (3%)	18/8192 (1%)
RAM(Byte)			45/1024(4%)		
压缩译码数组固定在 PROM 中存储地址		500H–578H	没有译码数组	600H–6FDH 700H–701H	没有译码数组
其它程式固定在 PROM 中存储地址	57EH–582H 583H–58CH	579H–57DH		6FEH–6FFFH 702H–703H	
Stack (层)			2		
寄存器使用记录	SPI1: SIMC0、SIMC2、SIMD D/A: DAH、DAL Timer: STM1C0、STM1C1、STM1AL、STM1AH、STM0C0、STM0C1、 STM0AL、STM0AH General: ACC、MP1、IAR1、TBLP、TBLH、TBHP、PCL、STATUS I/O: PBS0、PBS1、IFS、PBPU				

注: 1. 用户代码不能占用已经固定存储译码数组的空间。
2. 计算所耗用 PROM 空间: Default + 所选取之压缩模式 (可支持混合压缩模式)。

■ MCU 各功能模块使用说明:

- SPI1 应用为对外部 flash 的控制 (使用引脚: SCS(PB4)、SCK(PB2)、SDI(PB3)、SDO(PB1));
- STM1 中断应用为 play voice (中断入口地址为: 3CH);
- STM0 中断应用为 play sentence (中断入口地址: 2CH);
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: DACO(PB0)) ;
- 实现了对 RAM 的 BANK0 区进行优化 (BANK0: 28/128 (21%); BANK1: 17/128 (13%); BANK2: 0/128 (0%))。

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	14
_PLAY_SENTENCE	13
_PLAY_SENTENCE_INDEX	30
_VOLUME	4
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

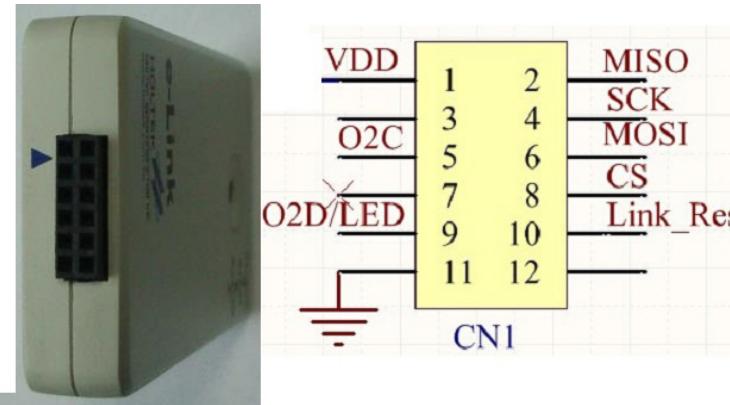
■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示 :

系统频率 压缩模式	2MHz	4MHz	8MHz
HT-ADPCM4	2KHz	4KHz	8KHz
HT-PCM12	2KHz	5KHz	11KHz
HT-uPCM8	2KHz	5KHz	10KHz
HT-PCM16	2KHz	5KHz	11KHz

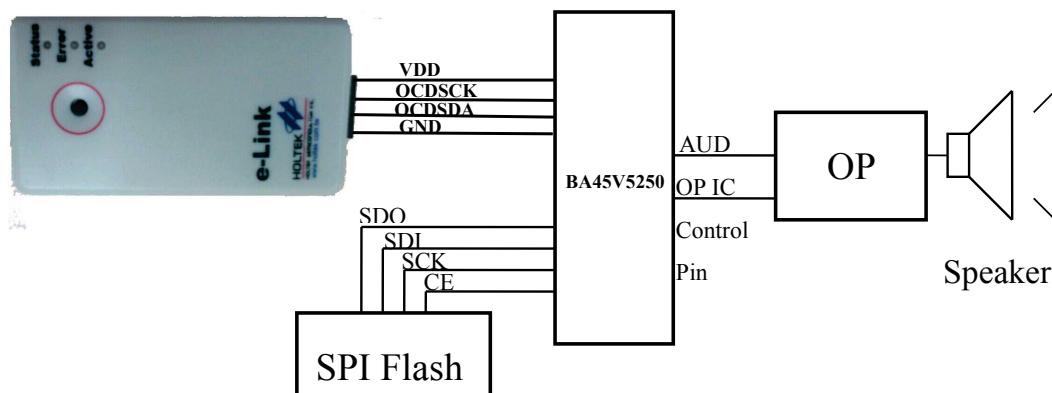
■ 仿真器及连线

这颗 MCU 所使用的是 e-Link 仿真器和 EV 芯片 BA45V5250 进行仿真调试，除此之外还需客户自行搭建 SPI Flash 和运放电路模块。

- e-Link 引脚图对应如下所示：

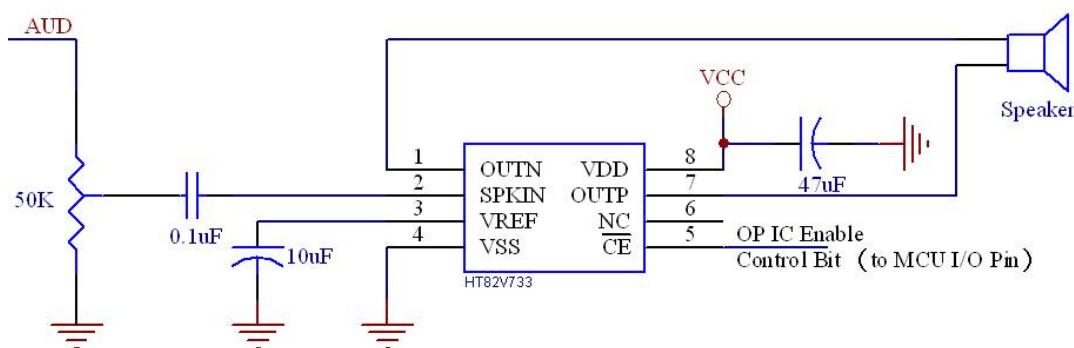


- BA45V5250 VDD、GND、OCDsck、OCDsda 引脚分别对应连 e-Link。



注 1：SPI flash 连线及烧录请看“[单独将 DAT 文件烧录到 flash 的接线方法](#)”章节。

2：应用 HT82V733 搭建运放参考电路如下所示：



BA45F5260

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT- ADPCM4	PCM12	UPCM8	PCM16
PROM(Word)	759/8192 (9%)	243/8192 (3%)	30/8192 (1%)	316/8192 (3%)	18/8192 (1%)
RAM(Byte)			45/2048(2%)		
压缩译码数组固定在 PROM 中存储地址		500H-578H	没有译码数组	600H-6FDH 700H-701H	没有译码数组
其它程式固定在 PROM 中存储地址	57EH-582H 583H-58CH	579H-57DH		6FEH-6FFFH 702H-703H	
Stack (层)			2		
寄存器使用记录	SPI1: SIMC0、SIMC2、SIMD D/A: DAH、DAL Timer: PTM0C0、PTM0C1、PTM0AL、PTM0AH、STM0C0、STM0C1、 STM0AL、STM0AH General: ACC、MP1、IAR1、TBLP、TBLH、TBHP、PCL、STATUS I/O: PBS0、PBS1、IFS、PBPU				

注: 1. 用户代码不能占用已经固定存储译码数组的空间。
2. 计算所耗用 PROM 空间: Default + 所选取之压缩模式 (可支持混合压缩模式)。

■ MCU 各功能模块使用说明:

- SPI1 应用为对外部 flash 的控制 (使用引脚: SCS(PB4)、SCK(PB2)、SDI(PB3)、SDO(PB1));
- STM0 中断应用为 play voice (中断入口地址为: 20H);
- PTM0 断应用为 play sentence (中断入口地址: 1CH);
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: DACO(PB0));
- 实现了对 RAM 的 BANK0 区进行优化 (BANK0: 28/128 (21%); BANK1: 17/128 (13%); BANK2: 0/128 (0%))。

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	14
_PLAY_SENTENCE	13
_PLAY_SENTENCE_INDEX	30
_VOLUME	4
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

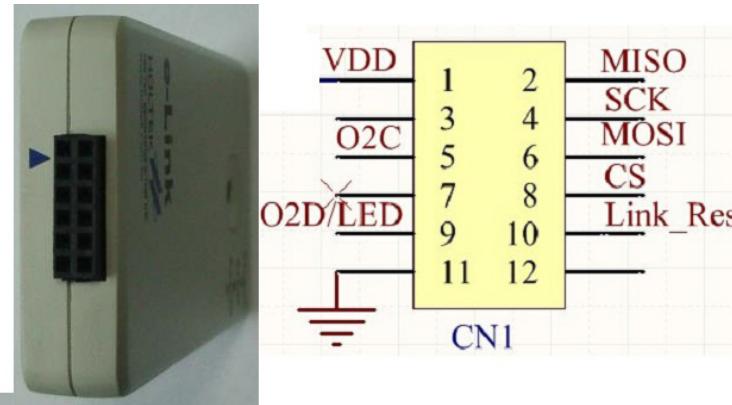
■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示 :

系统频率 压缩模式	2MHz	4MHz	8MHz
HT-ADPCM4	2KHz	4KHz	8KHz
HT-PCM12	2KHz	5KHz	11KHz
HT-uPCM8	2KHz	5KHz	10KHz
HT-PCM16	2KHz	5KHz	11KHz

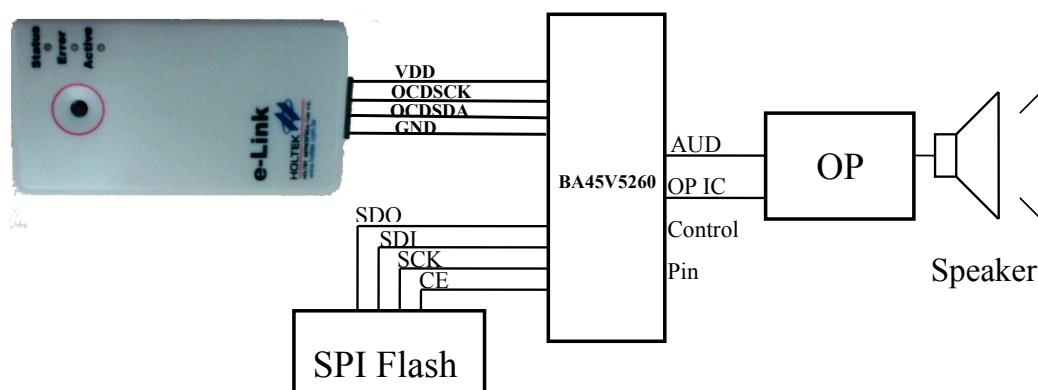
■ 仿真器及连线

这颗 MCU 所使用的是 e-Link 仿真器和 EV 芯片 BA45V5260 进行仿真调试，除此之外还需客户自行搭建 SPI Flash 和运放电路模块。

- e-Link 引脚图对应如下所示：

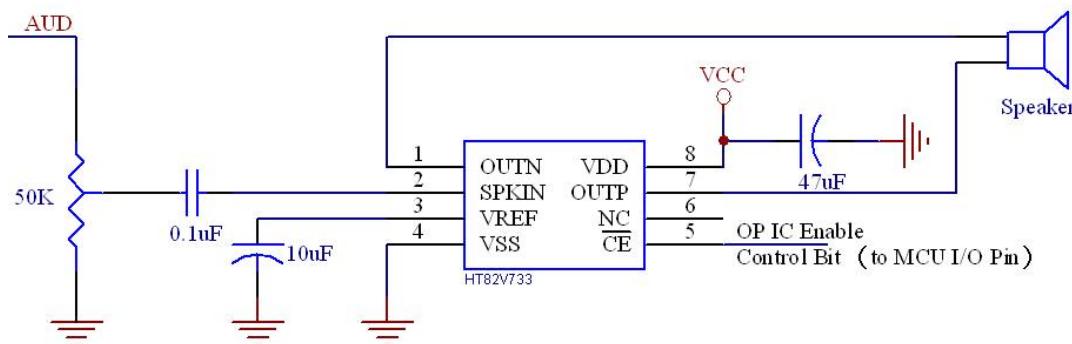


- BA45V5260 VDD、GND、OCDsck、OCDsda 引脚分别对应连 e-Link。



注 1：SPI flash 连线及烧录请看“[单独将 DAT 文件烧录到 flash 的接线方法](#)”章节。

2：应用 HT82V733 搭建运放参考电路如下所示：



HT45F23A

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-uPCM8
PROM(Word)	467/2048(23%)	241/2048 (12%)	98/2048 (5%)	316/2048 (15%)
RAM(Byte)			37/128(28%)	
压缩译码数组固定在 PROM 中存储地址		500H-578H	没有译码数组	600H-6FDH 700H-701H
其它程式固定在 PROM 中存储地址	57EH-582H 583H-58AH	579H-57DH	704H-72CH	6FEH-6FFH 702H-703H
Stack (层)			2	
寄存器使用记录	SPI1: SIMC0、SIMC1、SIMD D/A: DACTRL、DAH、DAL Timer: TMR0、TMR0C、TMR1H、TMR1L、TMR1C General: ACC、MP1、IAR1、INTC0、TBHP、INTC1、TBLP、TBLH、TBHP I/O: PBPU			

注: 1. 用户代码不能占用已经固定存储译码数组的空间
2. 计算所耗用 PROM 空间 : Default + 所选取之压缩模式 (可支持混合压缩模式)

■ MCU 各功能模块使用说明:

- SPI1 应用为对外部 flash 的控制 (使用引脚: SDI、SDO、SCK、SCS)
- Timer1 定时器中断应用为 play voice (中断入口地址为: 10H)
- Timer0 定时器中断应用为 play sentence (中断入口地址: 0CH)
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: AUD)

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

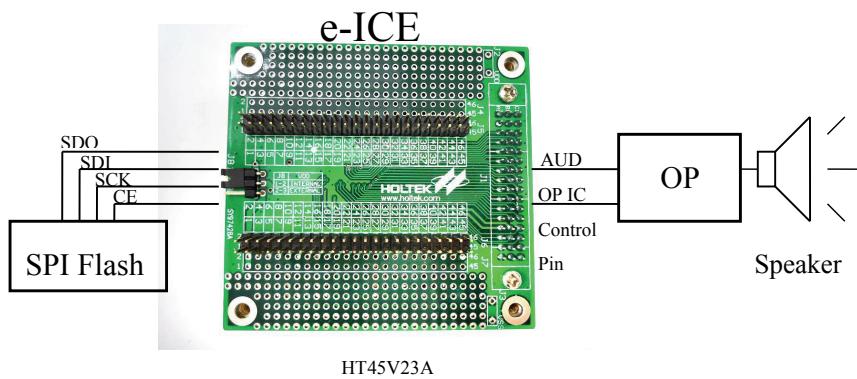
Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	13
_PLAY_SENTENCE	12
_PLAY_SENTENCE_INDEX	17
_VOLUME	9
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示:

系统频率 压缩模式	2MHz	4MHz	8MHz
HT-ADPCM4	3kHz	6kHz	13kHz
HT-PCM12	2kHz	5kHz	11kHz
HT-uPCM8	2kHz	5kHz	11kHz

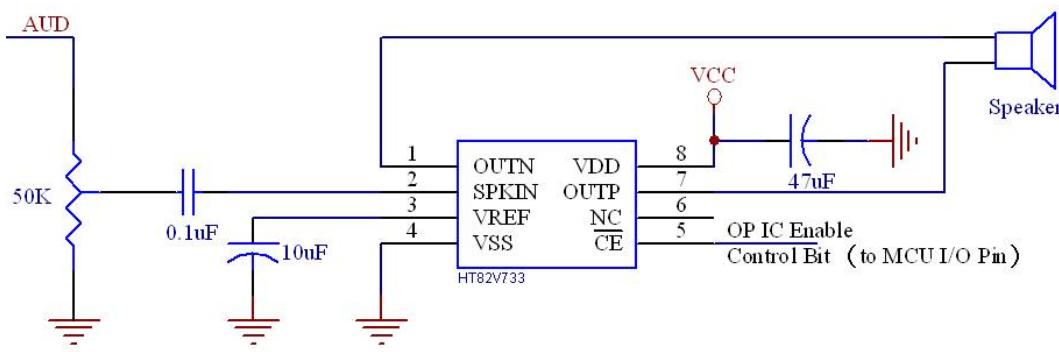
■ 仿真器及连线

这颗 MCU 所使用的是 e-ICE (M1001D+D1088A) 进行仿真调试，除此之外还需客户自行搭建 SPI Flash 和运放电路模块，连线如下图所示：



注 1：SPI flash 连线及烧录请看“[单独将 DAT 文件烧录到 flash 的接线方法](#)”章节。

2：应用 HT82V733 搭建运放参考电路如下所示：



HT45F24A

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-uPCM8
PROM(Word)	467/4096(11%)	241/4096 (6%)	98/4096 (2%)	316/4096(8%)
RAM(Byte)		37/192(18%)		
压缩译码数组固定在 PROM 中存储地址		500H–578H	没有译码数组	600H–6FDH 700H–701H
其它程式固定在 PROM 中存储地址	57EH–582H 583H–58AH	579H–57DH	704H–72CH	6FEH–6FFH 702H–703H
Stack (层)		2		
寄存器使用记录	SPI1: SIMC0、SIMC1、SIMD D/A: DACTRL、DAH、DAL Timer: TMR0、TMR0C、TMR1H、TMR1L、TMR1C General: ACC、MP1、IAR1、INTC0、TBHP、INTC1、TBLP、TBLH、 TBHP I/O: PBPU			

注: 1. 用户代码不能占用已经固定存储译码数组的空间
2. 计算所耗用 PROM 空间: Default + 所选取之压缩模式 (可支持混合压缩模式)

■ MCU 各功能模块使用说明:

- SPI1 应用为对外部 flash 的控制 (使用引脚: SDI、SDO、SCK、SCS)
- Timer1 定时器中断应用为 play voice (中断入口地址为: 10H)
- Timer0 定时器中断应用为 play sentence (中断入口地址: 0CH)
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: AUD)

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

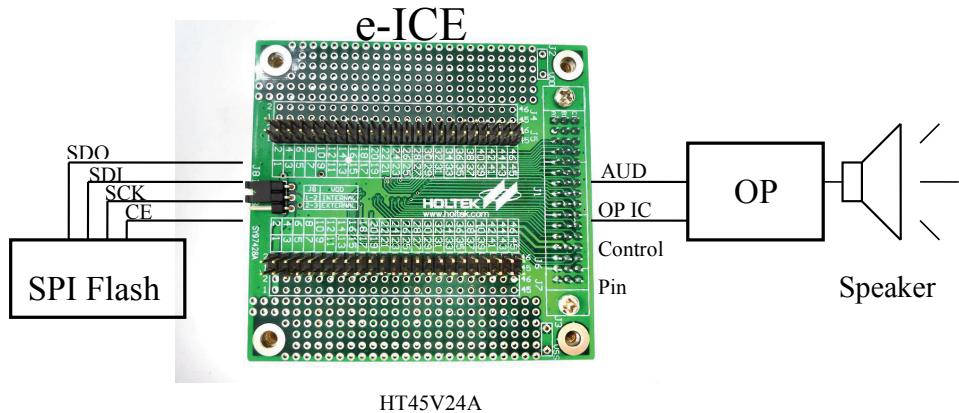
Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	13
_PLAY_SENTENCE	12
_PLAY_SENTENCE_INDEX	17
_VOLUME	9
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示:

系统频率 压缩模式	2MHz	4MHz	8MHz
HT-ADPCM4	3kHz	6kHz	13kHz
HT-PCM12	2kHz	5kHz	11kHz
HT-uPCM8	2kHz	5kHz	11kHz

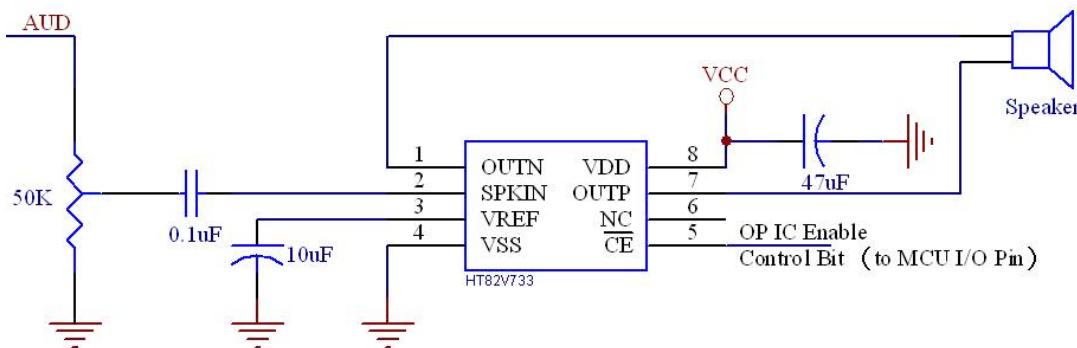
■ 仿真器及连线

这颗 MCU 所使用的是 e-ICE (M1001D+D1095A) 进行仿真调试，除此之外还需客户自行搭建 SPI Flash 和运放电路模块，连线如下图所示：



注 1：SPI flash 连线及烧录请看“[单独将 DAT 文件烧录到 flash 的接线方法](#)”章节。

2：应用 HT82V733 搭建运放参考电路如下所示：



HT83F02

■ 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-uPCM8
PROM(Word)	478/2048(25%)	237/2048(11%)	97/2048(4%)	310/2048(15%)
RAM(Byte)		36/208(16%)		
压缩译码数组固定在 PROM 中存储地址		500H-578H	没有译码数组	600H-6FDH 700H-701H
其它程式固定在 PROM 中存储地址	57EH-582H 583H-58AH	579H-57DH	704H-72CH	6FEH-6FFH 702H-703H
Stack (层)		2		
寄存器使用记录	SPI: SIMC0A、SIMC2A、SIMDRA、SIMDRB D/A: DACTRL、DAH、DAL Timer: TMR0、TMR0C、TMR1、TMR1C General: ACC、MP1、IAR1、INTC、TBLP、TBLH			

注: 1. 用户代码不能占用已经固定存储译码数组的空间
2. 计算所耗用 PROM 空间 : Default + 所选取之压缩模式 (可支持混合压缩模式)

■ MCU 各功能模块使用说明:

- SPIA 应用为对外部 flash 的控制 (使用引脚: SDAA、SCLA、SDIA、SCSA)
- Timer1 定时器中断应用为 play voice (中断入口地址为: 0CH)
- Timer0 定时器中断应用为 play sentence (中断入口地址: 08H)
- DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: AUD)

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表:

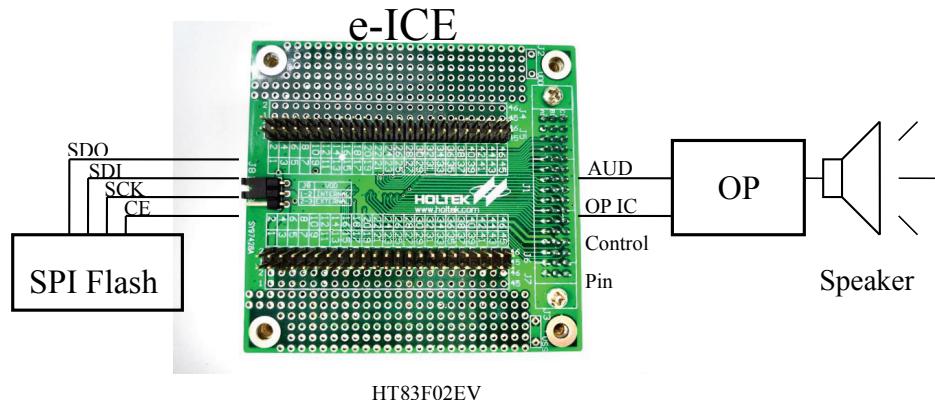
Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	14
_PLAY_SENTENCE	13
_PLAY_SENTENCE_INDEX	17
_VOLUME	6
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示:

系统频率 压缩模式	4MHz	8MHz	12MHz
HT-ADPCM4	7kHz	14kHz	21kHz
HT-PCM12	5kHz	11kHz	17kHz
HT-uPCM8	6kHz	12kHz	18kHz

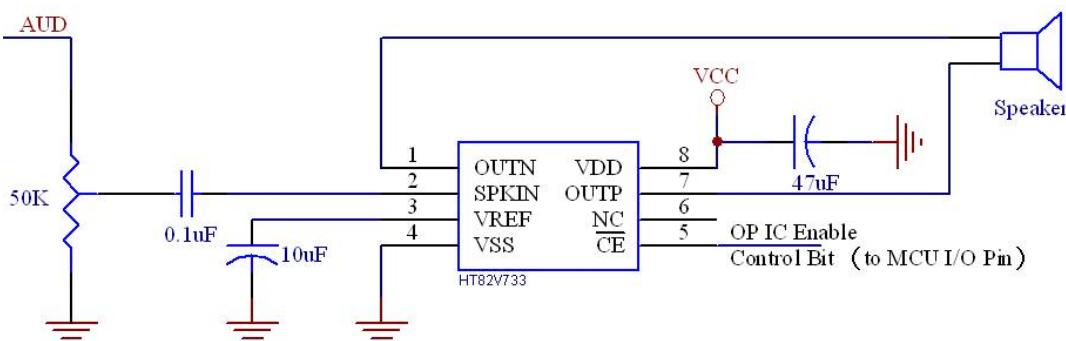
■ 仿真器及连线

这颗 MCU 所使用的是 e-ICE (M1001D+D1026A) 进行仿真调试，除此之外还需客户自行搭建 SPI Flash 和运放电路模块，连线如下图所示：



注 1：SPI flash 连线及烧录请看“[单独将 DAT 文件烧录到 flash 的接线方法](#)”章节。

2：应用 HT82V733 搭建运放参考电路如下所示：



HT86BX0

■ 资源使用信息表：

- HT86B03 资源使用信息表：

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8
PROM(Word)	470/4096 (11%)	241/4096 (6%)	98/4096 (2%)	316/4096 (8%)
RAM(Byte)			37/192(18%)	
压缩译码数组固定在 PROM 中存储地址		500H-578H	没有译码数组	600H-6FDH 700H-701H
其它程式固定在 PROM 中存储地址	57EH-582H 583H-58AH	579H-57DH	704H-72CH	6FEH-6FFH 702H-703H
Stack (层)			2	
寄存器使用记录	Voice rom data: VOICEC、LATCH0L、LATCH0M、LATCH0H、LATCHD D/A: DACTRL、DAH、DAL、VOL Timer: TMR0、TMR0C、TMR1、TMR1C General: ACC、MP1、IAR1、INTC、TBHP、TBLP、TBLH			

注：1. 用户代码不能占用已经固定存储译码数组的空间
2. 计算所耗用 PROM 空间：Default + 所选取之压缩模式（可支持混合压缩模式）

- HT86B10、HT86B20、HT86B30 资源使用信息表：

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8
PROM(Word)	470/8192(6%)	241/8192 (3%)	98/8192 (1%)	316/8192 (4%)
RAM(Byte)			37/192(18%)	
压缩译码数组固定在 PROM 中存储地址		500H-578H	没有译码数组	600H-6FDH 700H-701H
其它程式固定在 PROM 中存储地址	57EH-582H 583H-58AH	579H-57DH	704H-72CH	6FEH-6FFH 702H-703H
Stack (层)			2	
寄存器使用记录	Voice rom data: VOICEC、LATCH0L、LATCH0M、LATCH0H、LATCHD D/A: DACTRL、DAH、DAL、VOL Timer: TMR0、TMR0C、TMR1、TMR1C General: ACC、MP1、IAR1、INTC、TBHP、TBLP、TBLH			

注：1. 用户代码不能占用已经固定存储译码数组的空间
2. 计算所耗用 PROM 空间：Default + 所选取之压缩模式（可支持混合压缩模式）

● HT86B40、HT86B50、HT86B60、HT86B70、HT86B80、HT86B90 资源使用信息表:

压缩模式 已使用的资源	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8
PROM(Word)	648/8192 (16%)	241/8192 (3%)	98/8192 (1%)	316/8192 (4%)
RAM(Byte)	40/384(9%)			
压缩译码数组固定在 PROM 中存储地址		500H-578H	没有译码数组	600H-6FDH 700H-701H
其它程式固定在 PROM 中存储地址	57EH-582H 583H-58AH	579H-57DH	704H-72CH	6FEH-6FFFH 702H-703H
Stack (层)	2			
寄存器使用记录	Voice rom data: VOICEC、LATCH0L、LATCH0M、LATCH0H、LATCHD D/A: DACTRL、DAH、DAL、VOL Timer: TMR0、TMR0C、TMR2H、TMR2L、TM2C General: ACC、MP1、IAR1、BP、INTC、TBHP、TBLP、TBLH			

注: 1. 用户代码不能占用已经固定存储译码数组的空间
 2. 计算所耗用 PROM 空间 : Default + 所选取之压缩模式 (可支持混合压缩模式)

■ MCU 各功能模块使用说明:

- HT86B03、HT86B10、HT86B20、HT86B30:
 - ◆ Timer0 定时器中断应用为 play sentence (中断入口地址为: 08H)
 - ◆ Timer1 定时器中断应用为 play voice (中断入口地址: 0CH)
 - ◆ DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: AUD)
- HT86B40、HT86B50、HT86B60、HT86B70、HT86B80、HT86B90:
 - ◆ Timer0 定时器中断应用为 play sentence (中断入口地址为: 08H)
 - ◆ Timer2 定时器中断应用为 play voice (中断入口地址: 10H)
 - ◆ DAC 模块应用为对读取到 flash 音频数据进行 D/A 转换 (使用引脚: AUD)

■ 每调用一次下面子函数所耗 PROM 空间也将相应增加如下表：

- HT86B03、HT86B10、HT86B20、HT86B30

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	14
_PLAY_SENTENCE	13
_PLAY_SENTENCE_INDEX	18
_VOLUME	10
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

- HT86B40、HT86B50、HT86B60、HT86B70、HT86B80、HT86B90

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	20
_PLAY_SENTENCE	19
_PLAY_SENTENCE_INDEX	24
_VOLUME	10
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

■ MCU 相应频率对应不同压缩格式音源所能达到最大采样频率如下表所示：

- HT86B03 / HT86B10 / HT86B20 / HT86B30

系统频率 压缩模式	4MHz	6MHz	8MHz
HT-ADPCM4	6kHz	10kHz	13kHz
HT-PCM12	5kHz	8kHz	11kHz
HT-uPCM8	5kHz	8kHz	11kHz

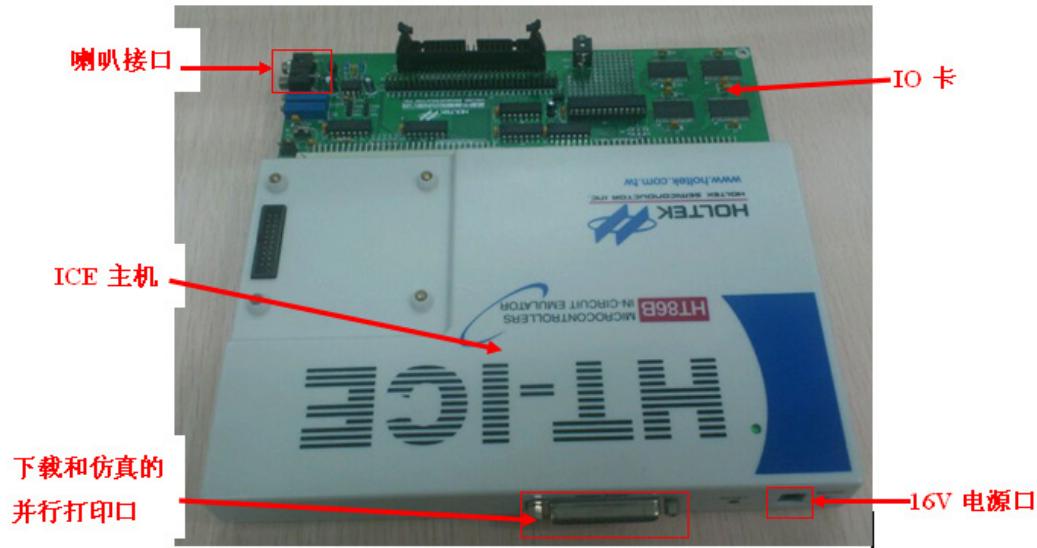
- HT86B40 / HT86B50/ HT86B60 / HT86B70 / HT86B80 / HT86B90

系统频率 压缩模式	4MHz	6MHz	8MHz
HT-ADPCM4	6kHz	10kHz	13kHz
HT-PCM12	5kHz	8kHz	11kHz
HT-uPCM8	5kHz	8kHz	11kHz

■ 仿真器及连线

这颗 MCU 所使用的是 HT-ICE 仿真器 HT86B 和 IO 板卡进行仿真调试，具体流程如下：

- 硬体的认知

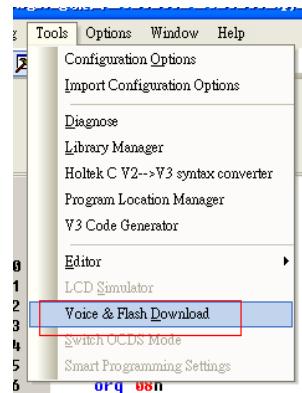


Print Cable

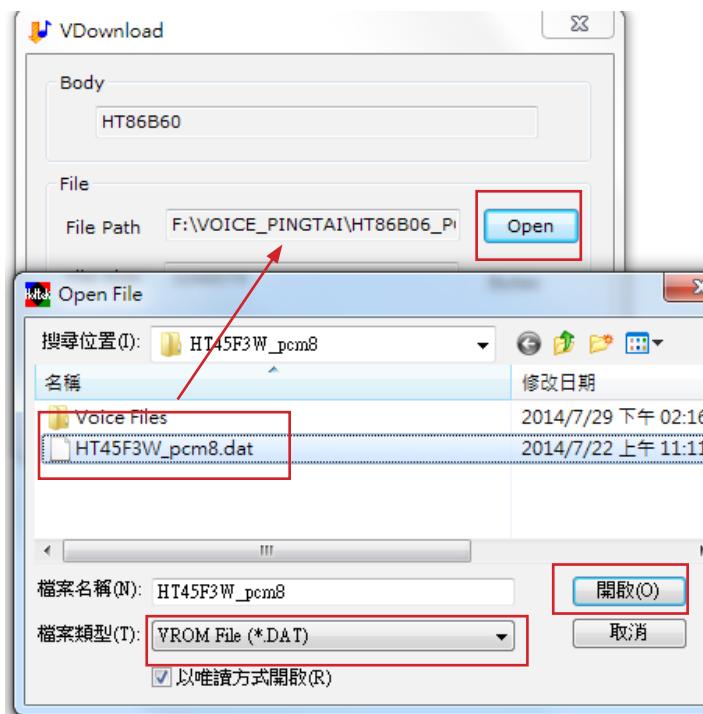


USB 转 Print Cable

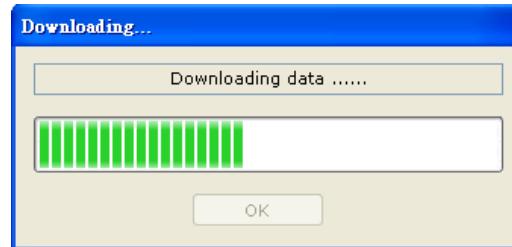
- 由 voice 平台产生 .dat 音频档；
- 接上 16V 直流电源；应用 Print Cable (若电脑没有 Print 接口可使用 USB 转 Print Cable) 连接到 PC 机下载和仿真；
- 新建或打开已有 HT86BX0 的 IDE3000 工程 (以便 PC 可以链接硬体)，在工程菜单栏打开 tools 里面的 voice&flash download 下载音频信号，如下图所示：



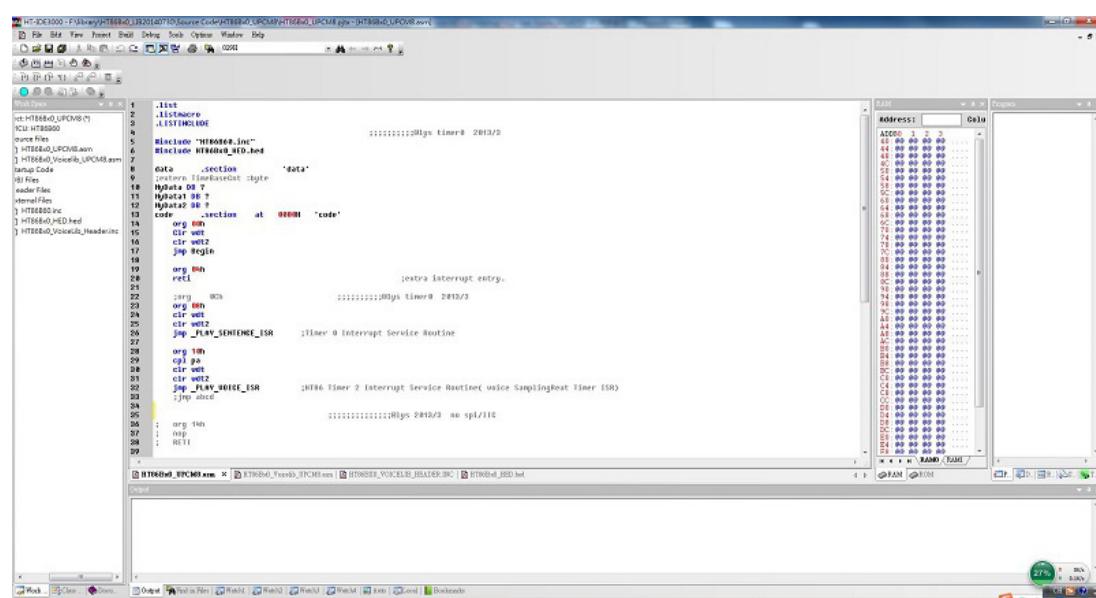
- 点击 ‘open’ 将平台产生的 dat 文件加载到 VDownload 中进行下载，如下图所示：



- Download 图示:



- IDE 对工程进行仿真



4 Audacity 的快速入门

Audacity 的概述

Audacity 是一款免费的、易于使用的音频编辑器和录音器，可运行于 Windows、Mac OS X、GNU/Linux 及其它操作系统上。您可将 Audacity 用于：

- 现场录音
- 将磁带和录音带装变为数字录音或 CD
- 编辑 Ogg Vorbis、MP3 及 WAV 音频文件
- 剪切、拷贝、接合及混音
- 改变录音的速率或音高

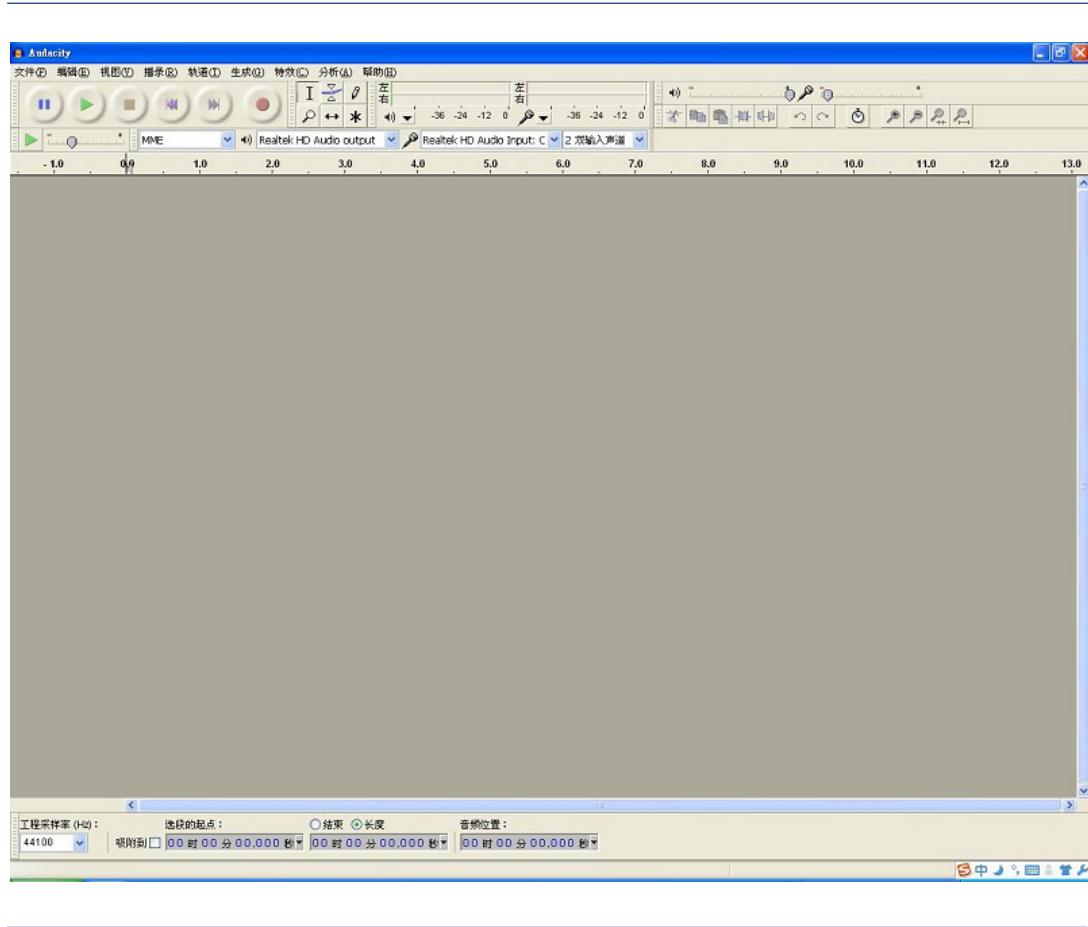
注：Audacity 可于 <http://audacity.sourceforge.net> 免费下载使用。

Audacity 的处理流程

- 声音的输入
有音乐 CD 输入，转为 WAV 格式或由 WAV/AIFF/OGG/MP3 文件输入
直接开启使用或录音。
- 音乐基本处理
基本剪接（删除、插入、复制）音量控制（envelope/amplify）
Fade in/Fade out、消除噪音
插入固定长度静音、混音、改变音调
- 输出
输出成 wav/aiff/mp3/ogg 文件 烧录成音乐 CD

快速上手

■ 鼠标右键双击  打开 Audacity 软件界面如下图所示：



- Audacity 界面工具的认识



- ◆ Audacity 中的控制工具列认识

	跳到最前的位置。		暂停播放或錄音。再按一下继续。
	播放。如果只选取了某部分，则只播放该部分。		停止播放或录音。
	录音，由光标位置开始录音		跳到最后位置。

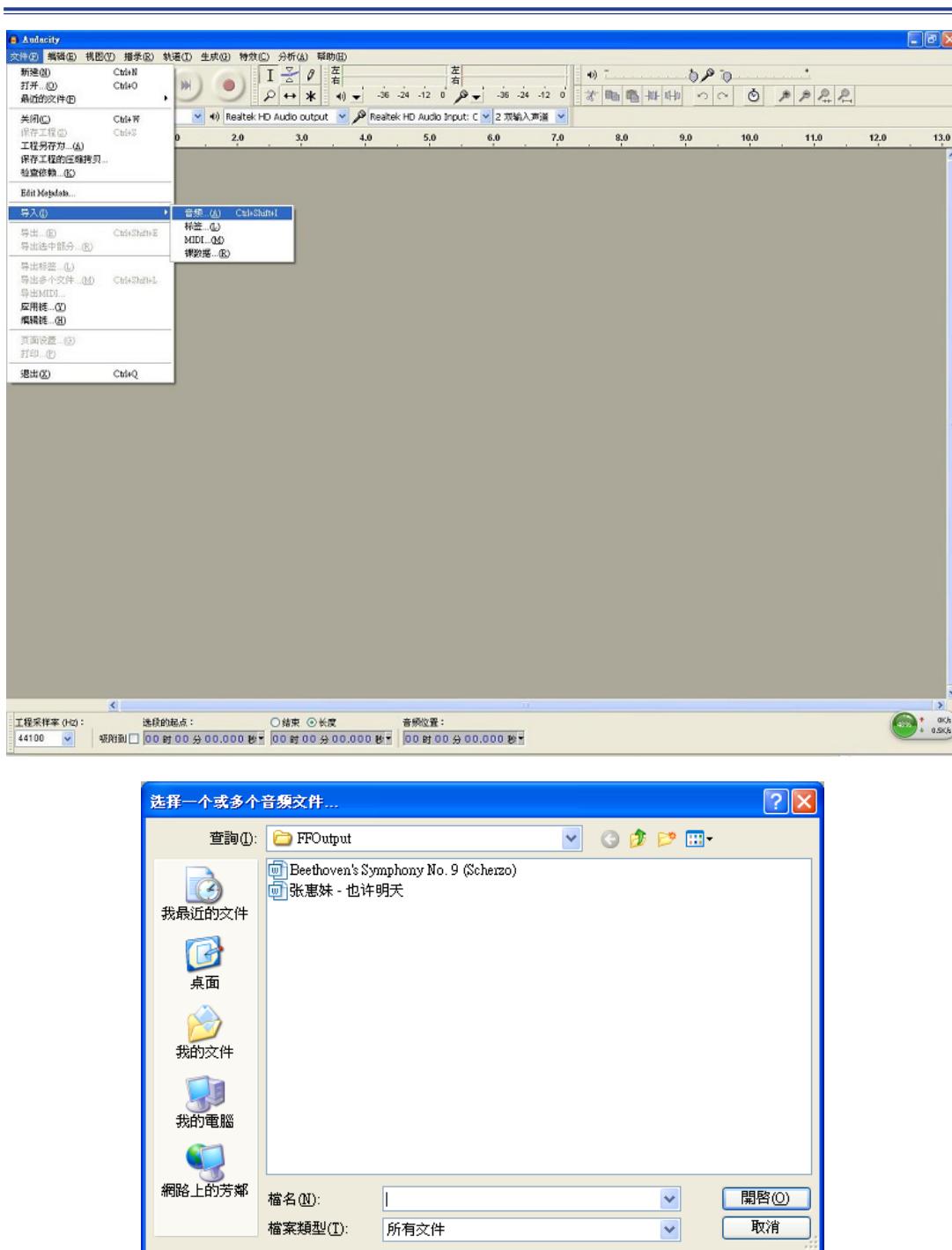
- ◆ Audacity 中的编辑工具列认识

	选择工具 – 可用來选择你的声音部分，以便可以进行编辑或聆听。		缩放工具 – 放大、缩小音轨。
	包络线工具 – 改变音轨的音量		时间移动工具 – 可把音轨左右移动。
	绘图工具 – 改变个别音乐样本		多选工具模式 – 电脑会依鼠标在不同的位置或所按下的键自动选取适当的编辑工具。

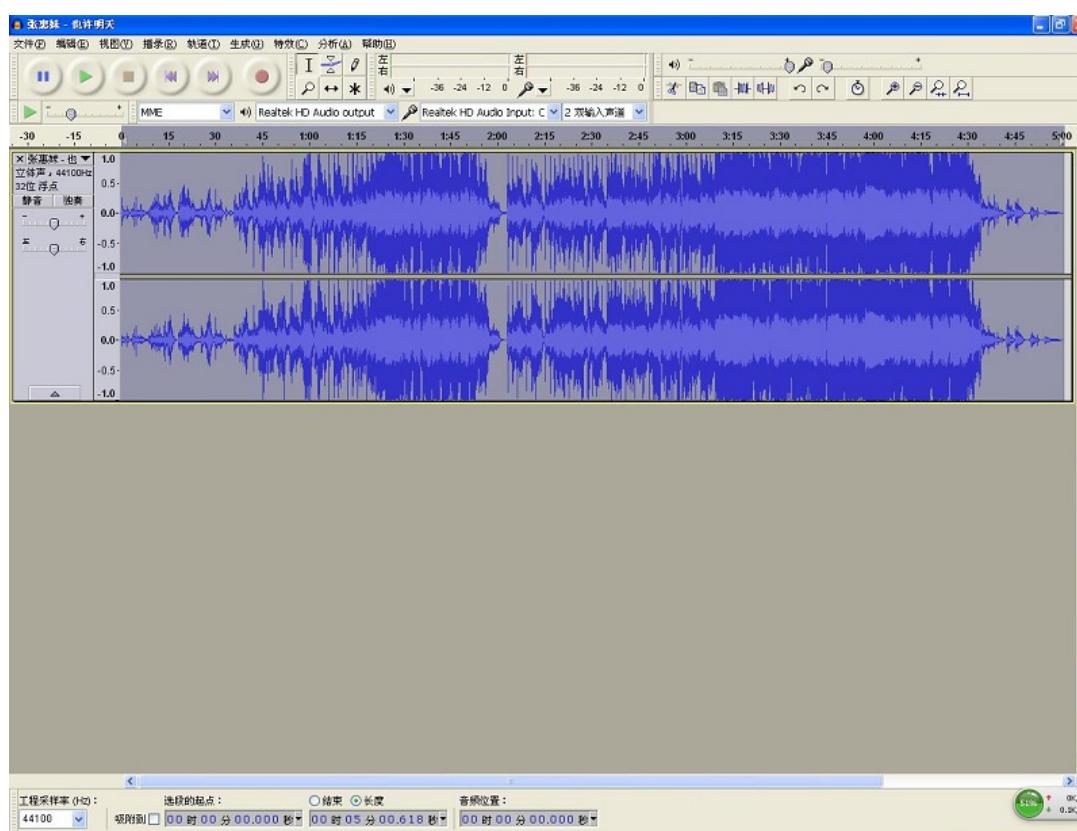
■ 音乐的载入

- 通常有以下三种况
 - ◆ 音乐 CD 输入入，转为 wav 格式
 - ◆ 音音，需有适当软件设备（如麦克风录音软件）
 - ◆ wav / aiff / ogg / mp3 文件输入入，直接开启使用

如下图所示：“文件”→“导入”→“音频”→“音频路径选择”。



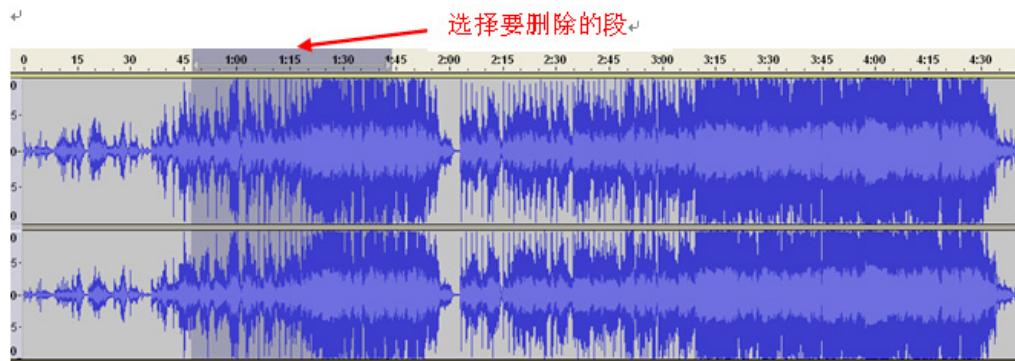
● 载入音频文件后的界面



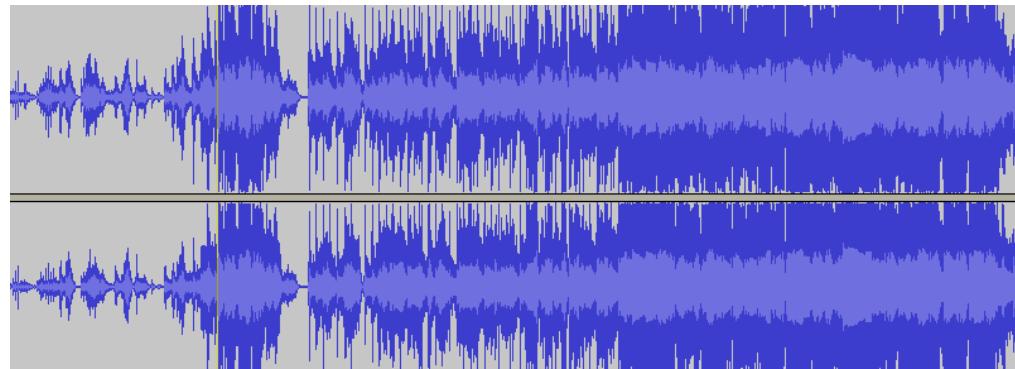
■ 对载入音乐的基本处理

- 基本剪接(删除、插入、复制)
 - ◆ 删除: 选择需要删除的部分(按下滑鼠不放, 拖拉出所要选择的范围), 然后按Delete 键删除所选内容。

删除前的选择的图:



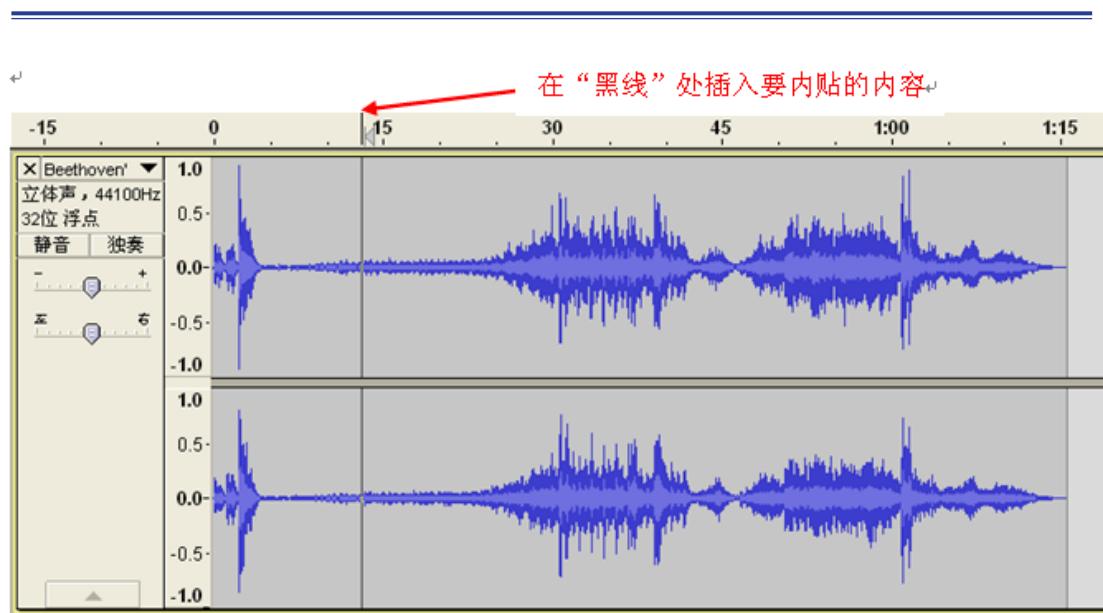
删除后:



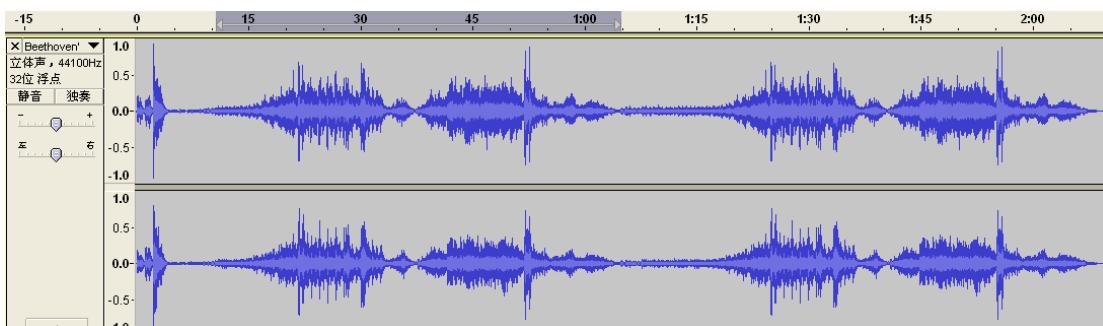
- ◆ 复制黏贴：选取一小段音轨后  下「复制」按钮，然后再按需要插入的位置  按钮。

注：如需由另一个文件复制音轨，请先打开该文件，File → Open。打开后，你将看到两个 Audacity 窗口，复制你所选取的部分，复制后于第一个窗口的适当位置「贴上」 。

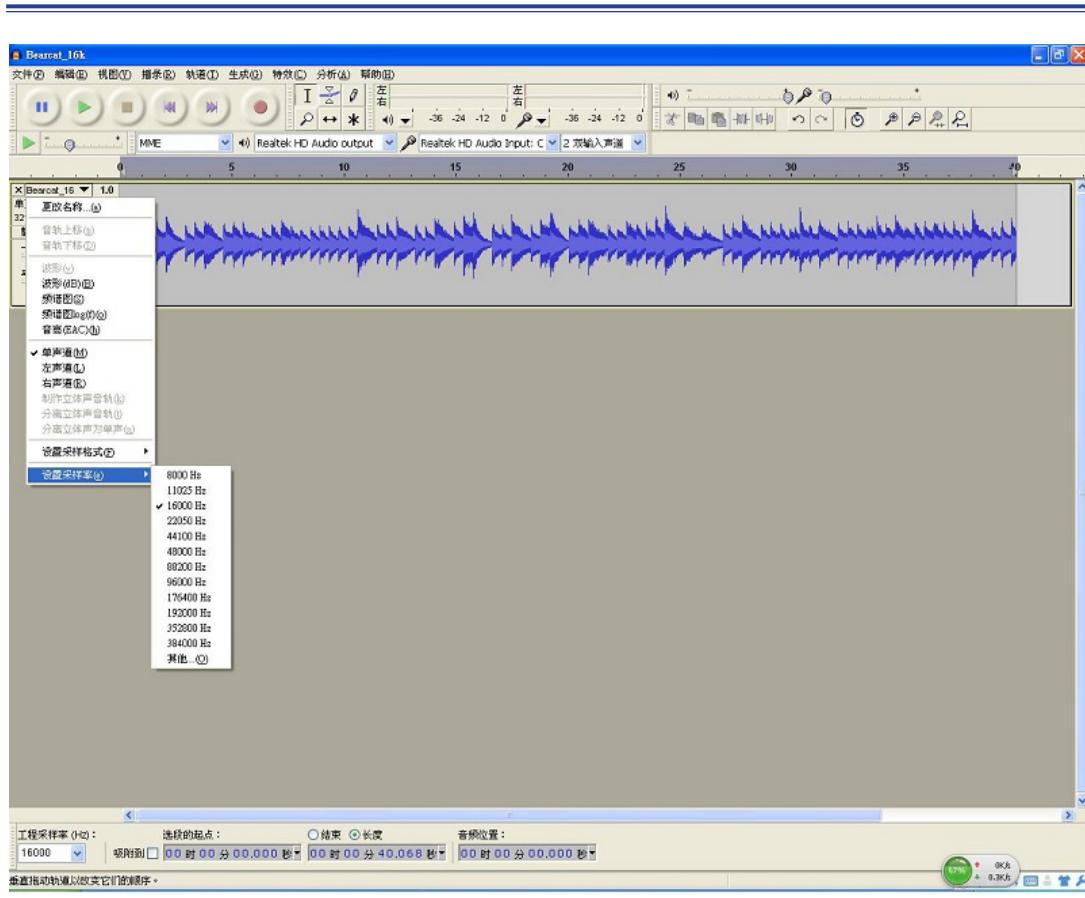
复制前：



复制后：



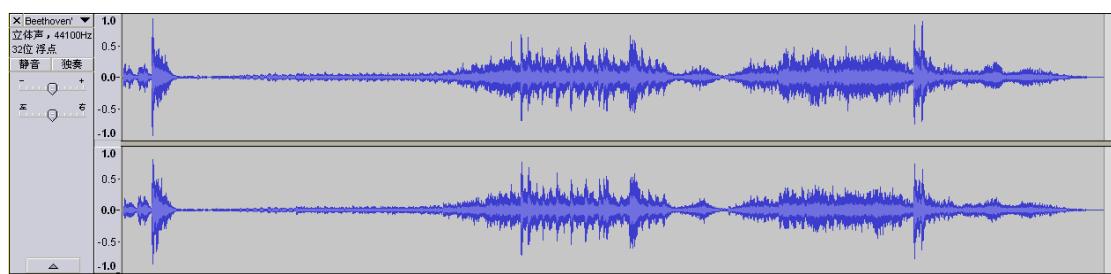
- 修改音源的采样频率，如下图所示：



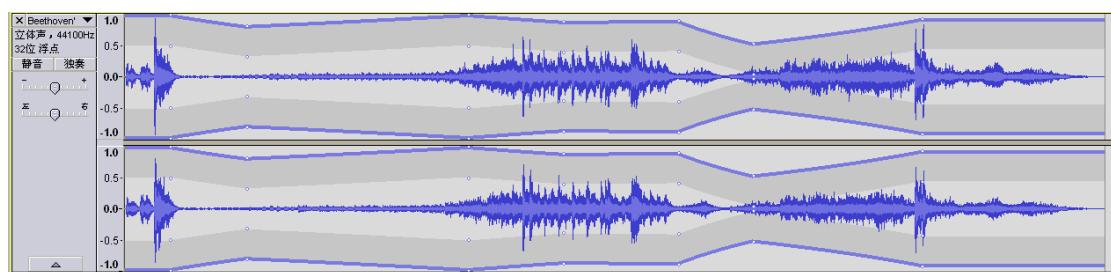
- 音量控制 (envelope)

使用 Envelope 工具  在音频轨道上点击就可看见“白点”不同位置设定这些点即可上下拖拉改变音量大小。

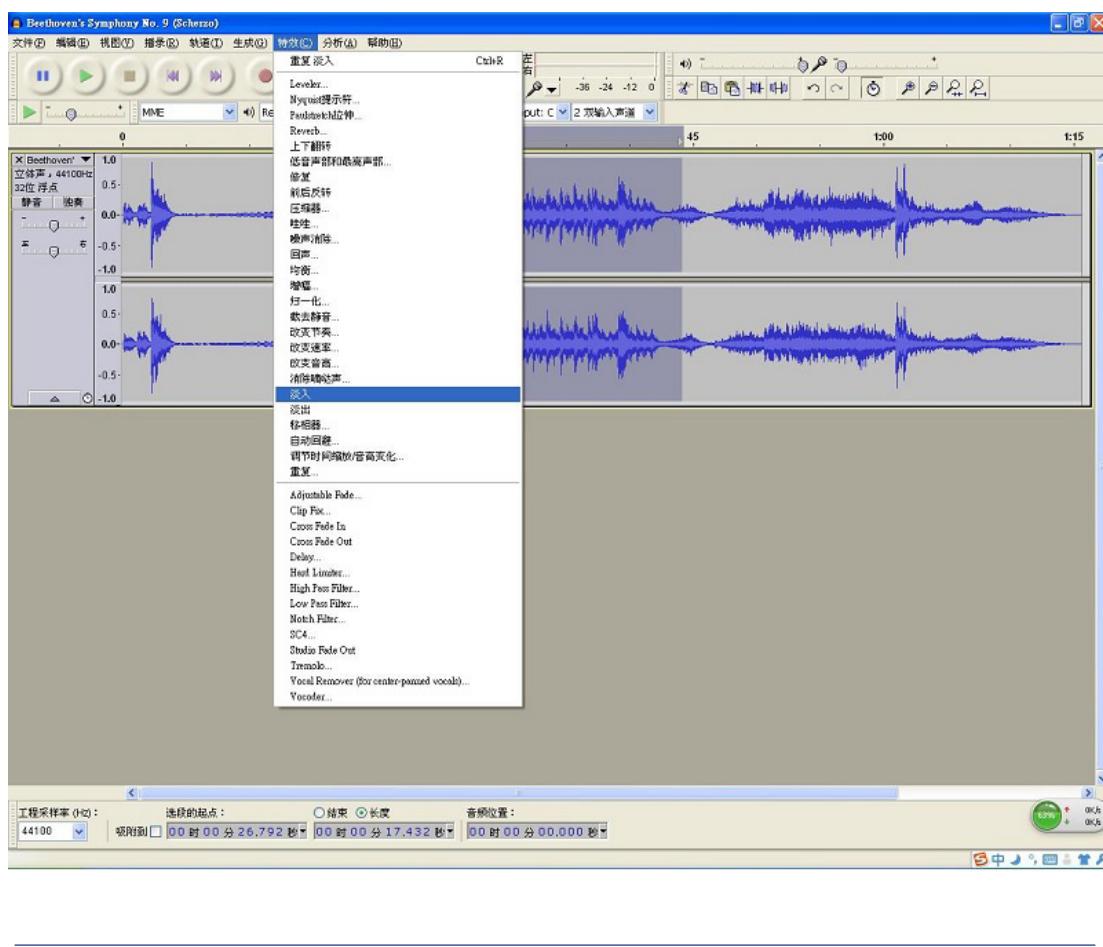
音量改变前



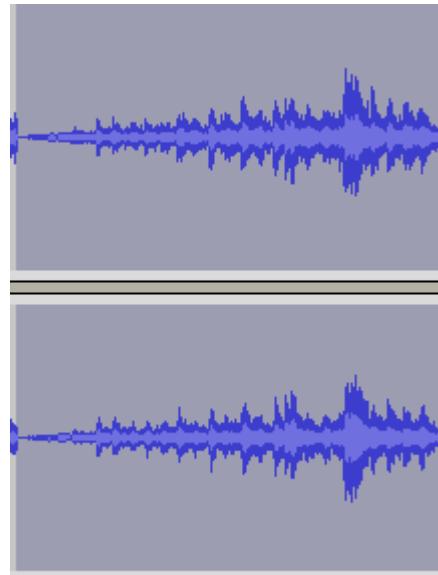
音量改变后



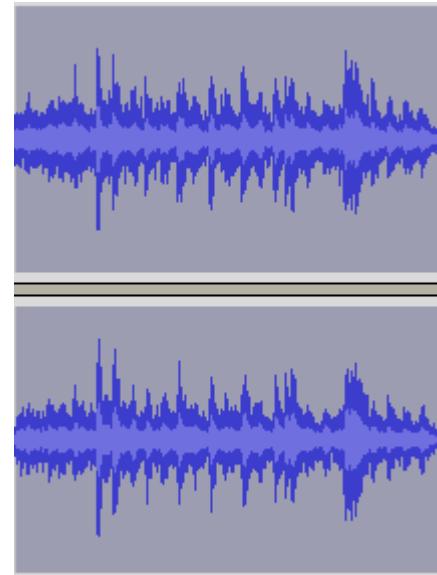
- 音频的特效：
- 点击“特效”即可选择如图以下的一些效果：



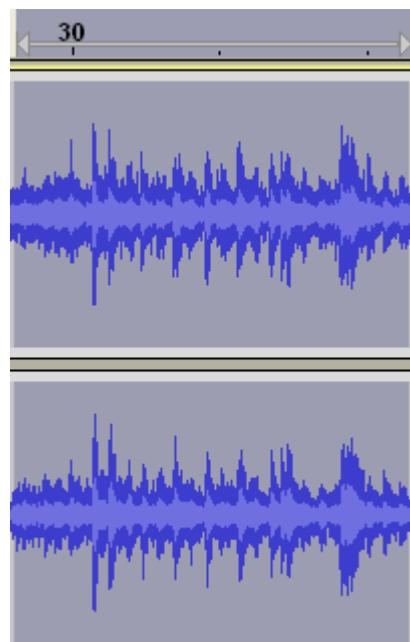
◆ 淡入 / 淡出功能：



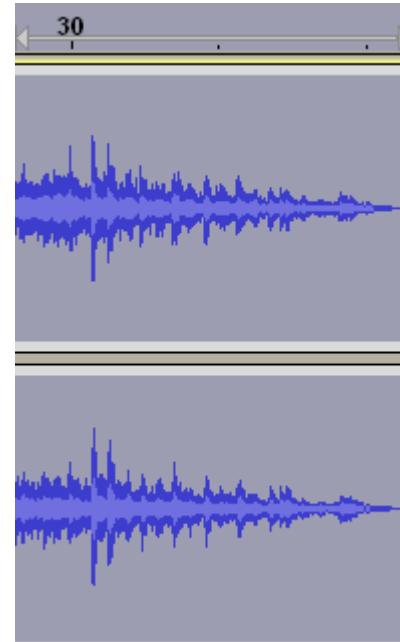
淡入前



淡入后



淡出前

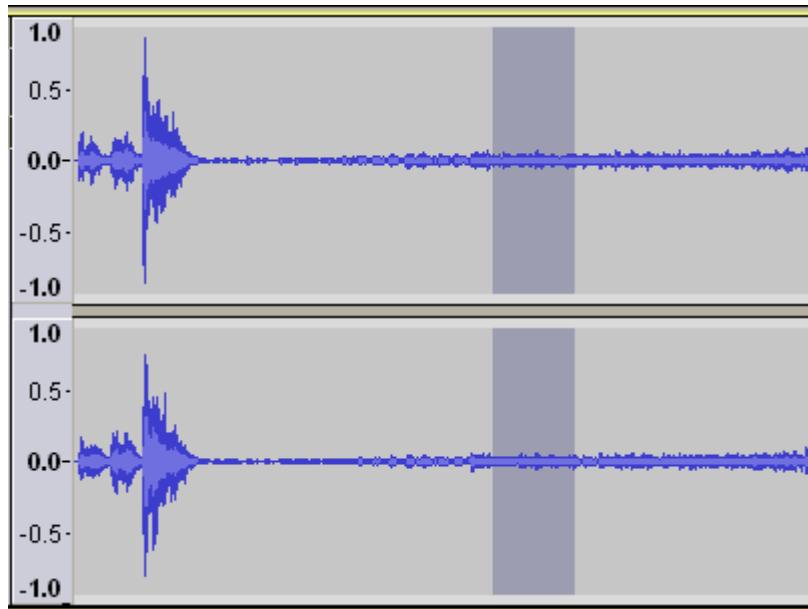


淡出后

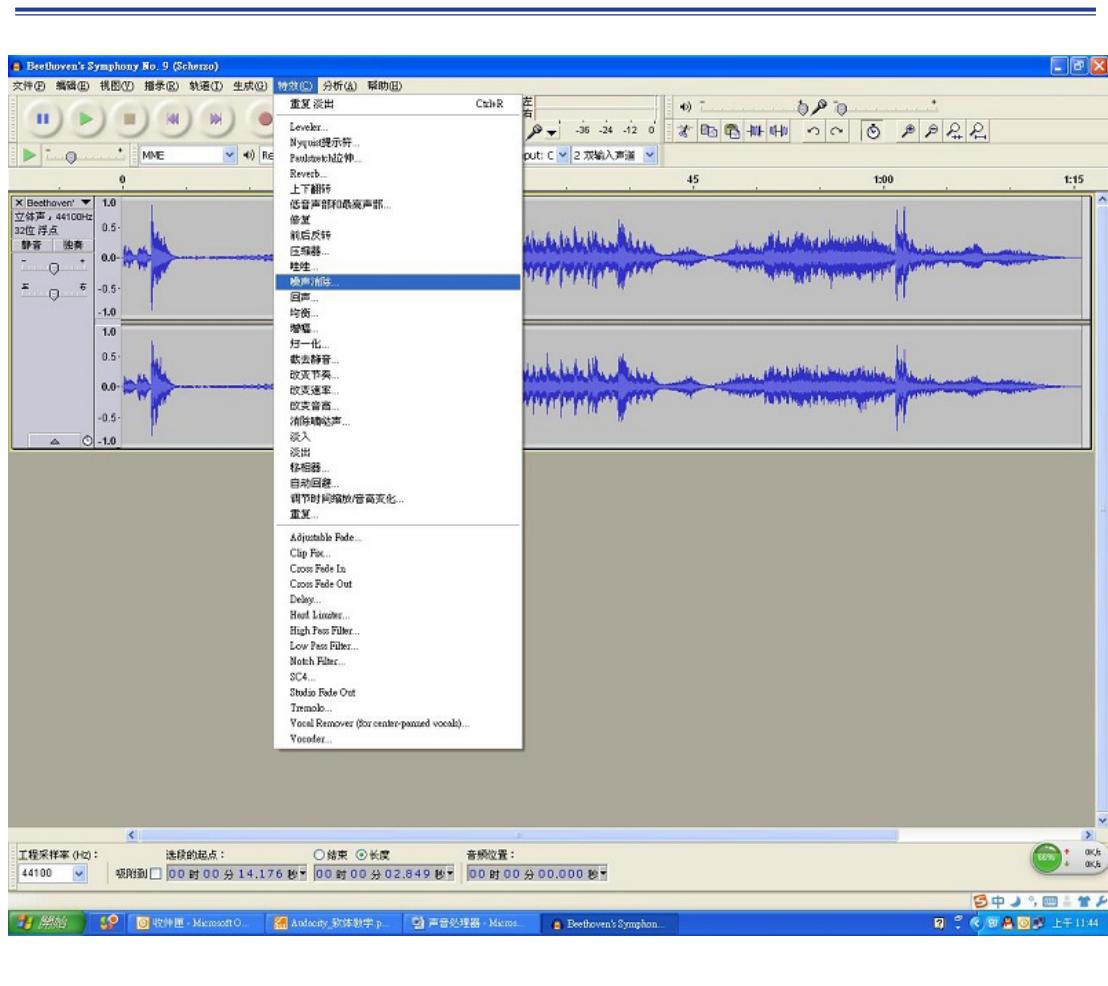
◆ 消除噪音

可以有效消除持续重复出現的背景噪音等。

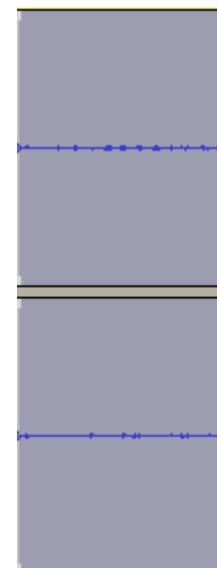
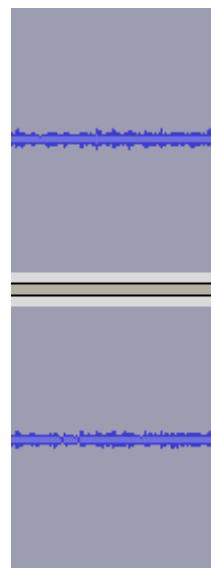
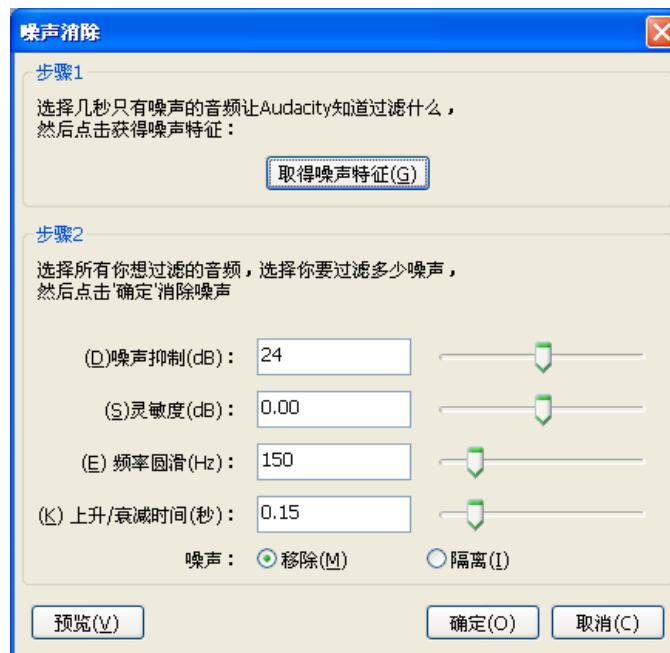
- a. 先告诉电脑哪些是「噪音」——选取噪音样本。于音轨上选择「噪音」的一小段
(最理想的大約是 0.5 至 2 秒)。



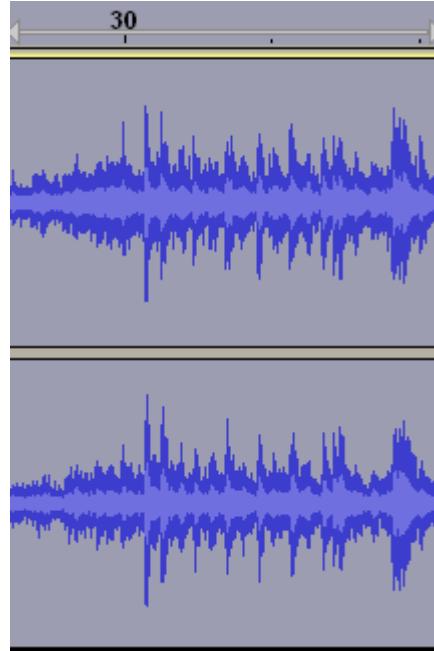
b. 点击菜单栏中“特效”->“消除噪音”如下图所示：



c. 点击“噪音消除”后弹出了下面一些参数设置后按确定就可以看到轨道波形的效果：



- ◆ 把选择部分变为静音：
选择需要静音的部分 -> 点击静音的按钮  即可：



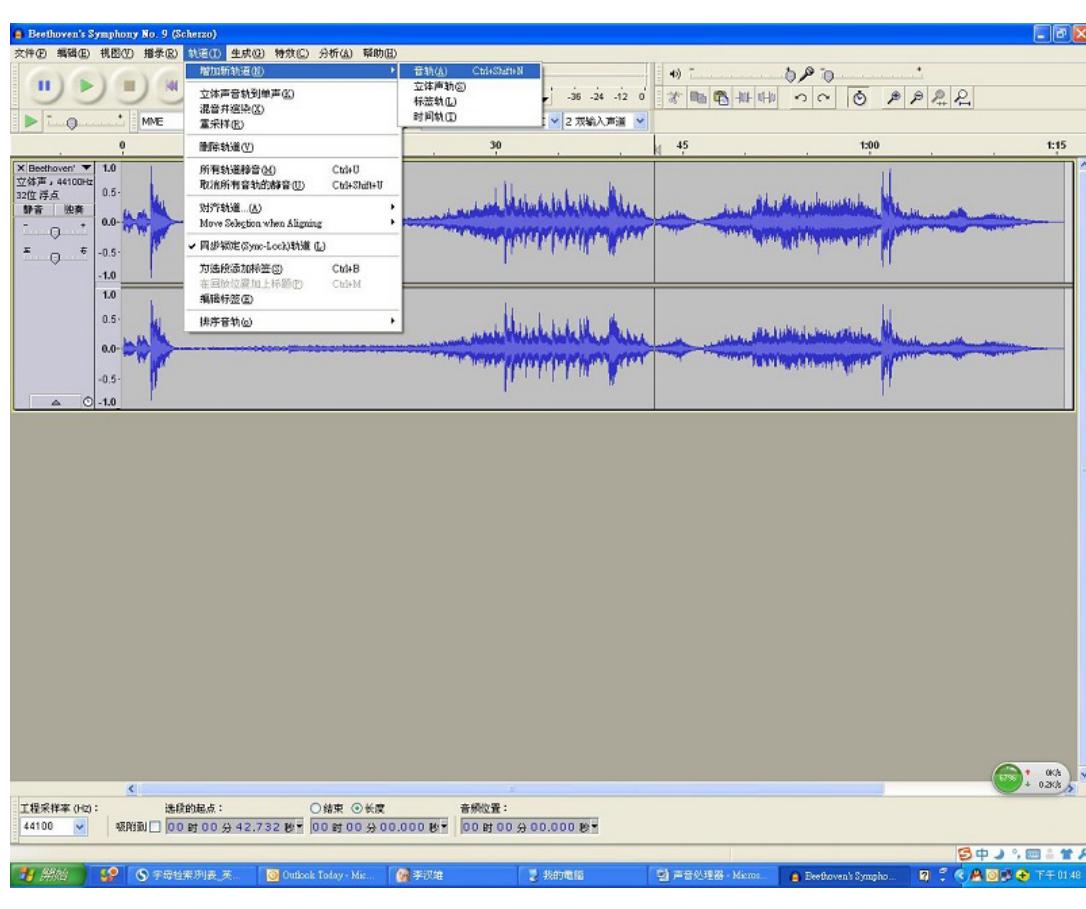
静音处理前



静音处理后

◆ 混音

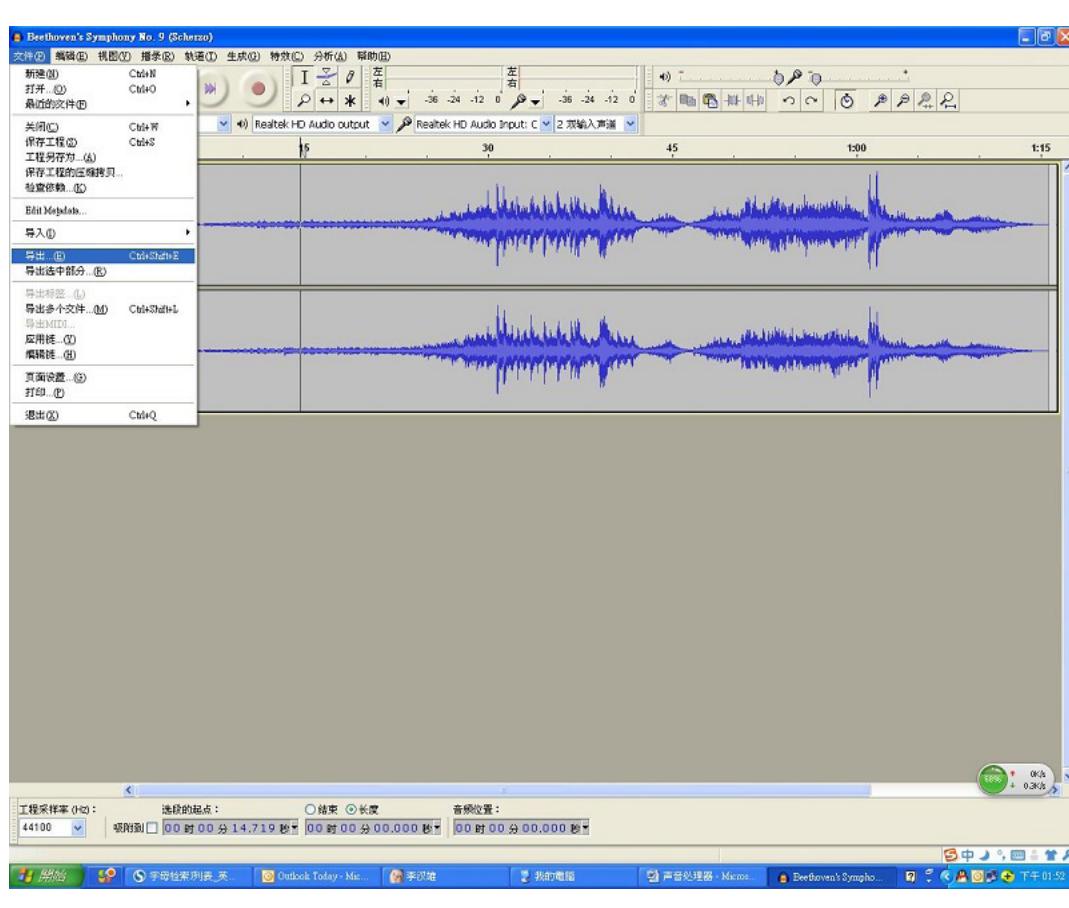
混音即是把兩段或以上的音轨混合在一起，例如把声音音轨和音乐音轨合在一起，使有背景音乐的效果。如要加入另一立音轨，“请选择轨道” → “增加新轨道” à “音轨”，然後在新加入的音轨上贴上所需內容。

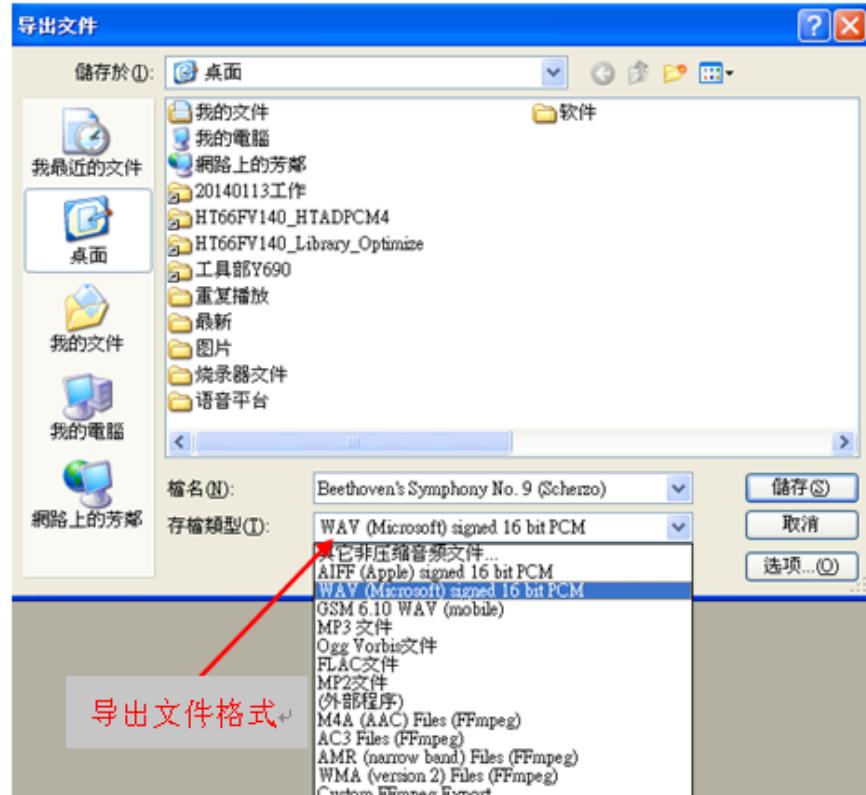


◆ 声音输出

输出成 wav / aiff / mp3 / ogg 文件

完成声音处理后，可以直接把文件输出为以上的格式。（注：Voice 平台只支持 WAV 音频格式）。“文件”->“导出”->“选择导出路径和格式”





5 Adobe Audition CS6 简易教程

简介

Adobe 推出 Adobe Audition 软件, 这是一个完整的、应用于运行 Windows 系统的 PC 机上的多音轨唱片工作室。该产品此前叫做 Cool Edit Pro 2.1, 在 2003 年 5 月从 Syentrillium Software 公司成功购买。 Adobe Audition 提供了高级混音、编辑、控制和特效处理能力, 是一个专业级的音频工具, 允许用户编辑个性化的音频文件、创建循环、引进了 45 个以上的 DSP 特效以及高达 128 个音轨。

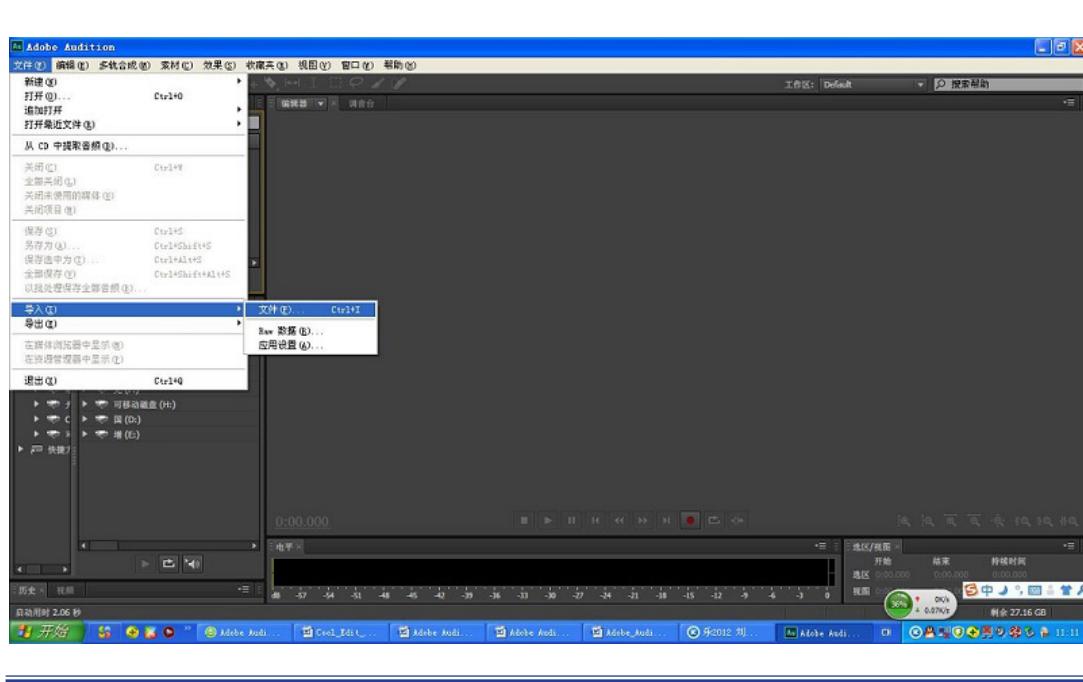
Adobe Audition 拥有集成的多音轨和编辑视图、实时特效、环绕支持、分析工具、恢复特性和视频支持等功能, 为音乐、视频、音频和声音设计专业人员提供全面集成的音频编辑和混音解决方案。用户可以从允许他们听到即时的变化和跟踪 EQ 的实时音频特效中获益匪浅。它包括了灵活的循环工具和数千个高质量、免除专利使用费 (royalty-free) 的音乐循环, 有助于音乐跟踪和音乐创作。

Adobe Audition 提供了直觉的、客户化的界面, 允许用户删减和调整窗口的大小, 创建一个高效率的音频工作范围。一个窗口管理器能够利用跳跃跟踪打开的文件、特效和各种爱好, 批处理工具可以高效率处理诸如对多个文件的所有声音进行匹配、把它们转化为标准文件格式之类的日常工作。Adobe Audition 为视频项目提供了高品质的音频, 允许用户对能够观看影片重放的 AVI 声音音轨进行编辑、混合和增加特效。广泛支持工业标准音频文件格式, 包括 WAV、AIFF、MP3、MP3PRO 和 WMA, 还能够利用达 32 位的位深度来处理文件, 取样速度超过 192 kHz, 从而能够以最高品质的声音输出磁带、CD、DVD 或 DVD 音频。

快速入门

对单首音乐的编辑

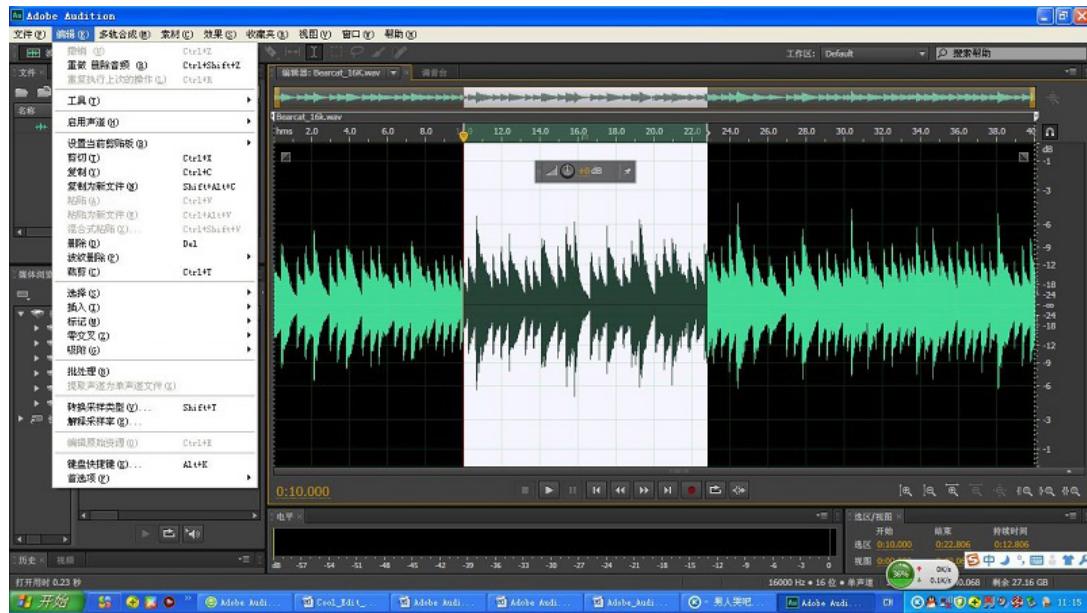
■ 打开软件后，首先我们点击“文件”->“导入”->“文件”。如下图所示：



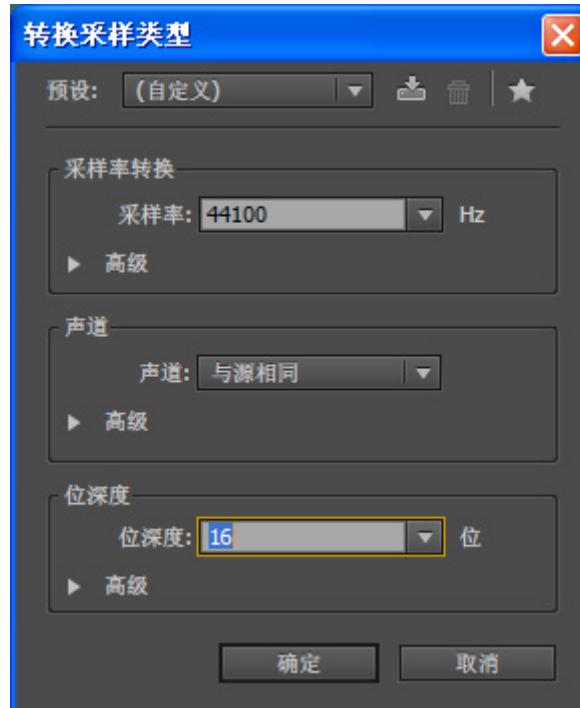
■ 导入音乐后如下图所示：



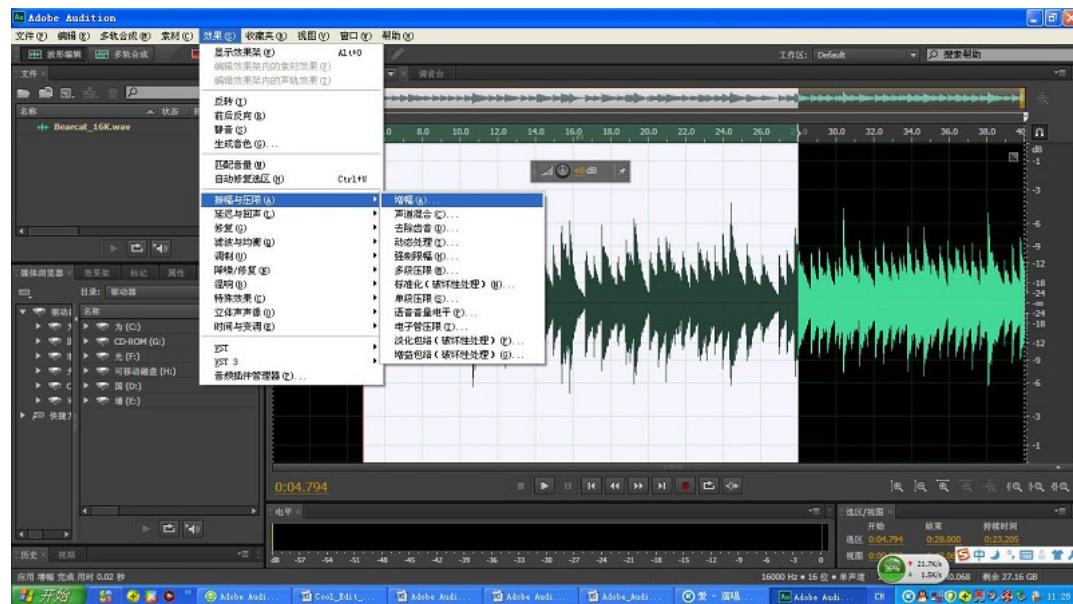
■ 选择音频段，点击“编辑”选择删除、剪切、复制、黏贴等处理。



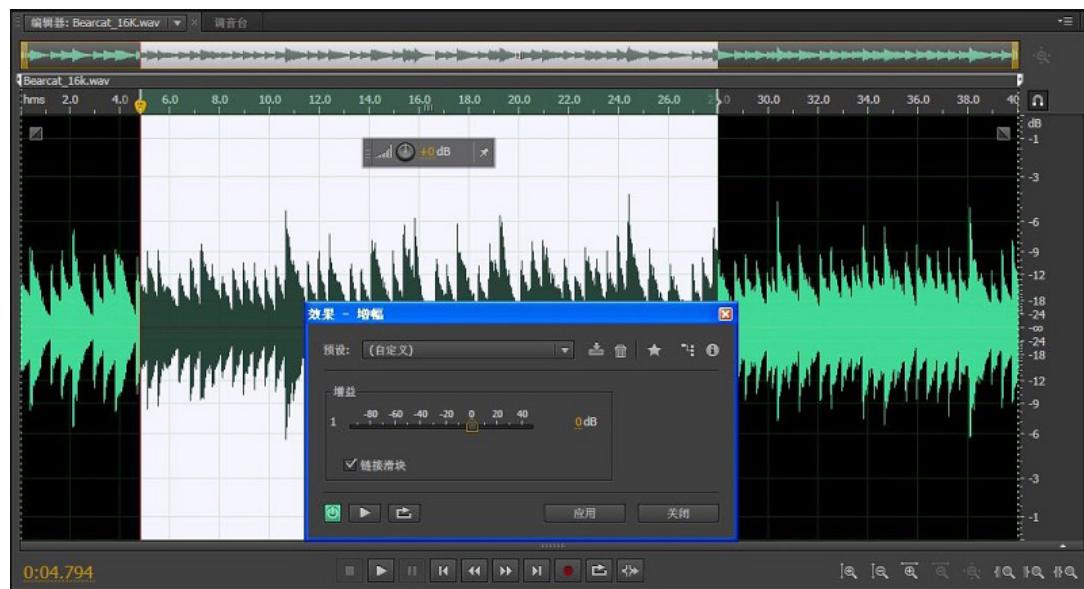
■ “编辑”中选择“转换频率”对“采样频率”、“位深度”进行设置。



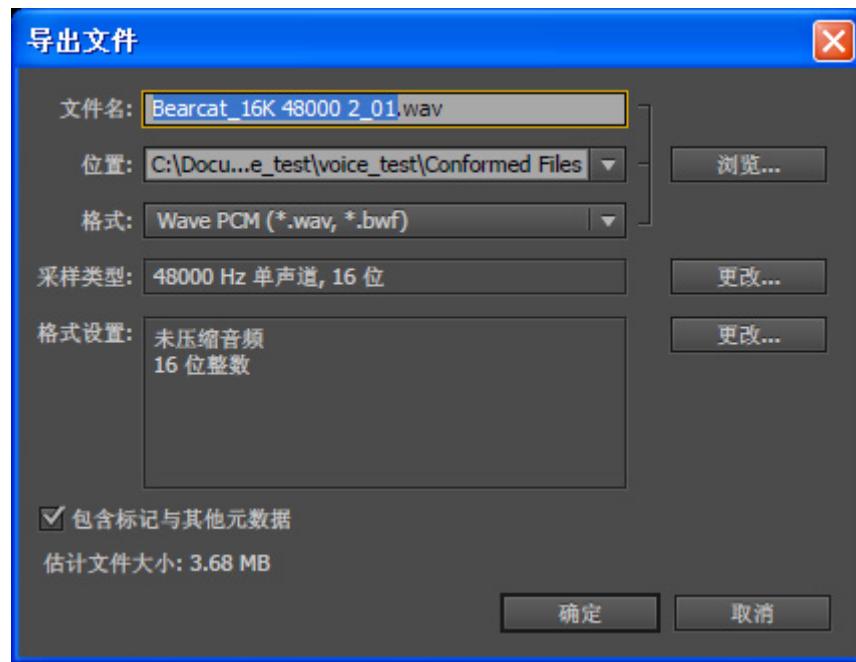
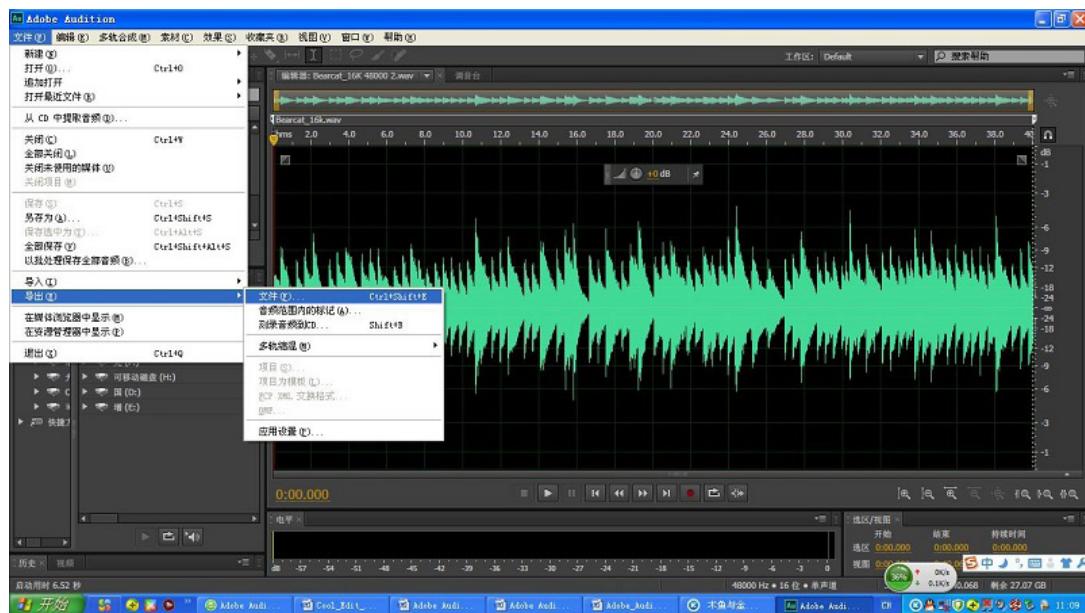
■ 增加音频效果，包括很多效果，可以根据自己的意愿进行选择，如下面改变某一段的音量。选择要改变的音频点击“效果”->“振幅与压限”->“振幅”。



点击“振幅”后弹出下面窗口修改后点击“应用”即可改变声音大小。

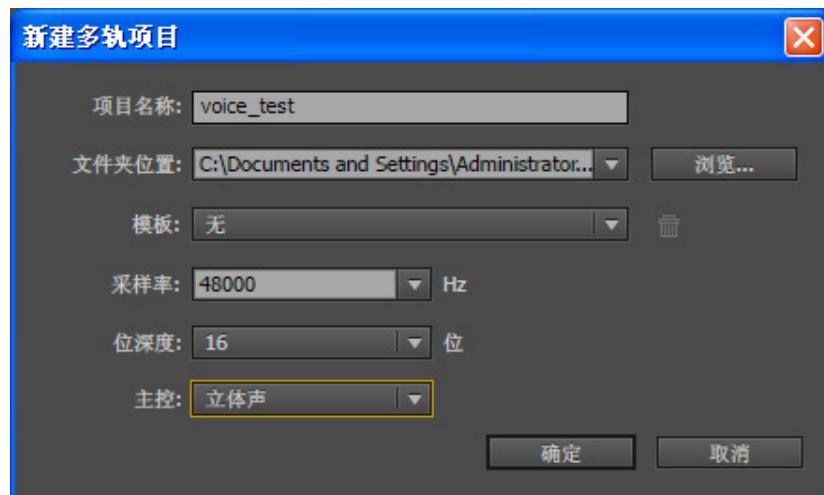
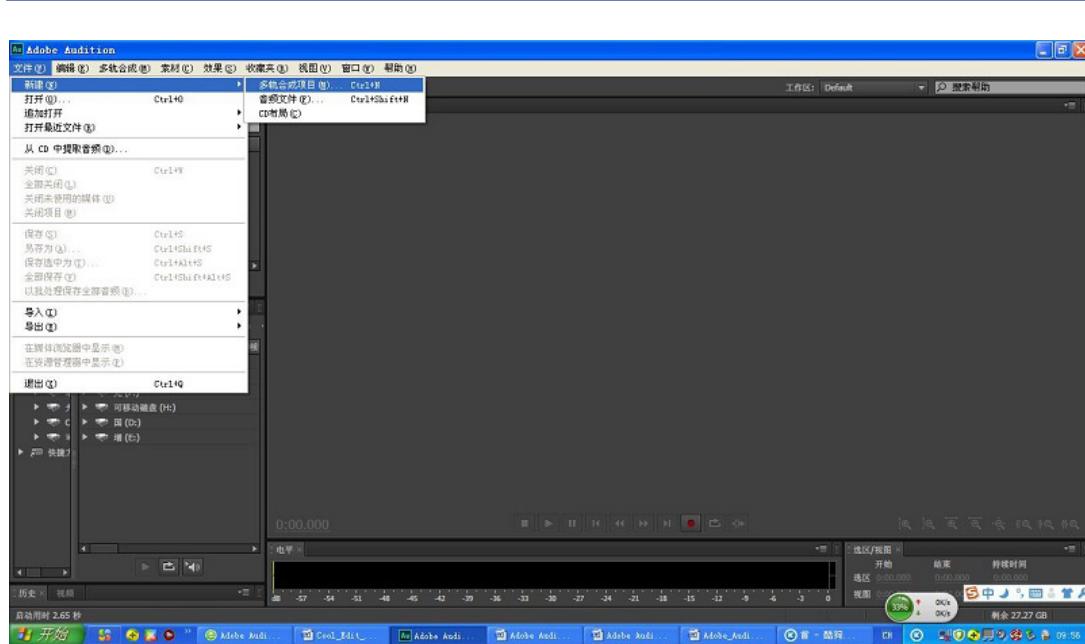


■ 编辑完成后，我们点击“文件”，选择“导出”->“文件”你会看到，在该窗口中可以查看或更改保存文件的具体参数，选择好参数后，点击“确定”就完成文件的导出了。

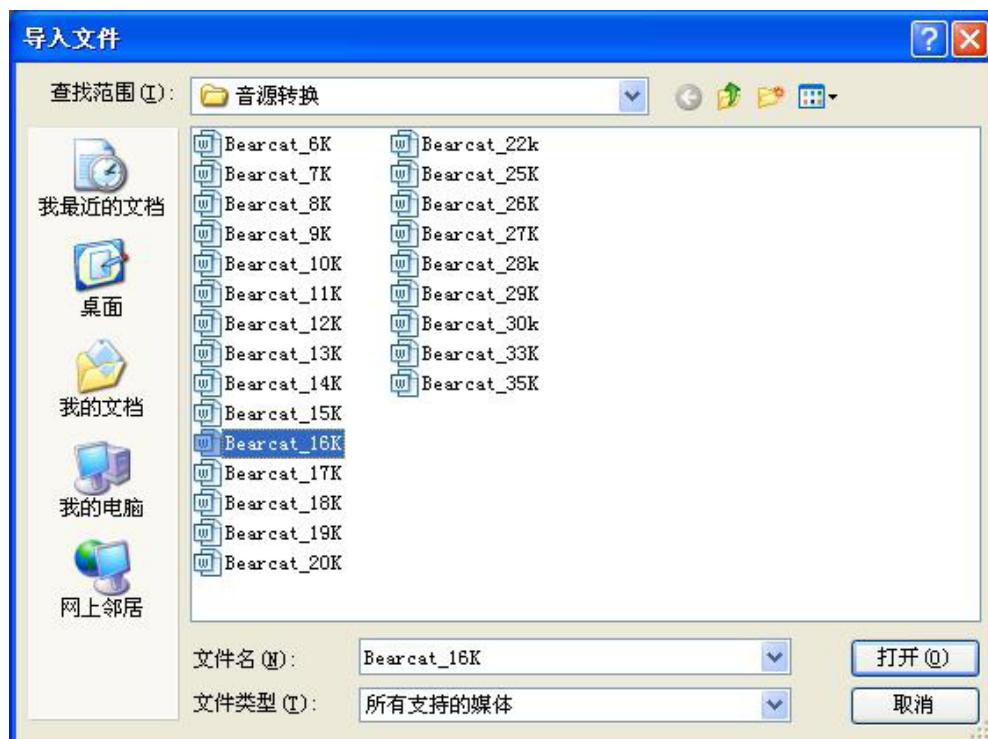
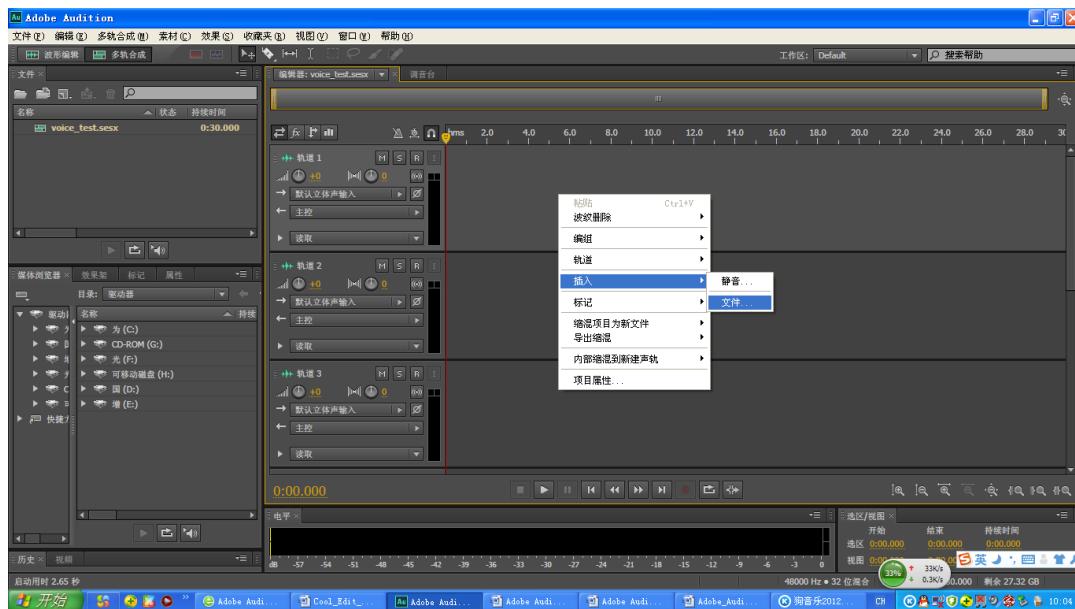


录音功能编辑

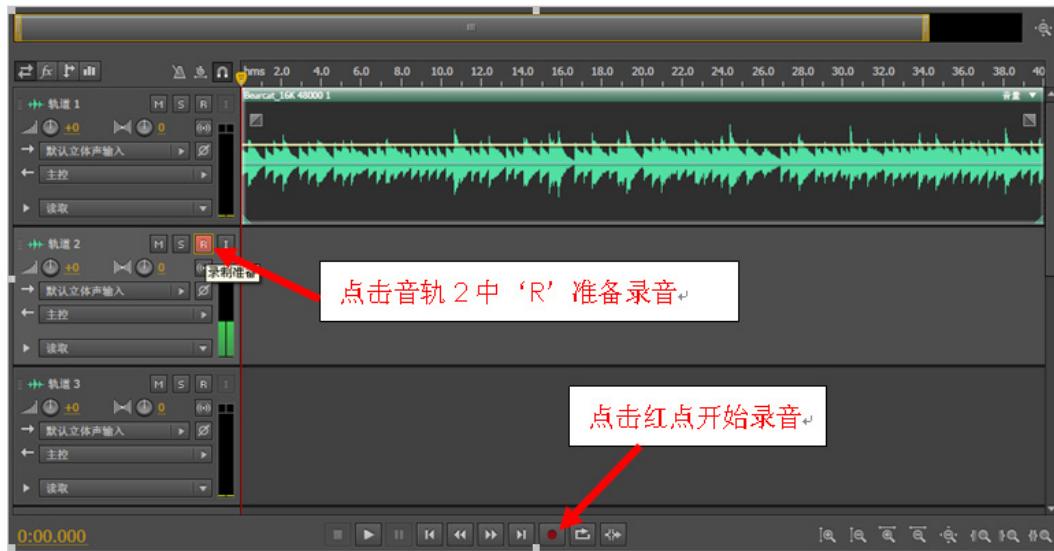
- 打开软件后，首先我们点击“文件”，选“新建”创建一个多轨混音项目，在里面设置项目参数，如采样率、声道等，如下图所示。



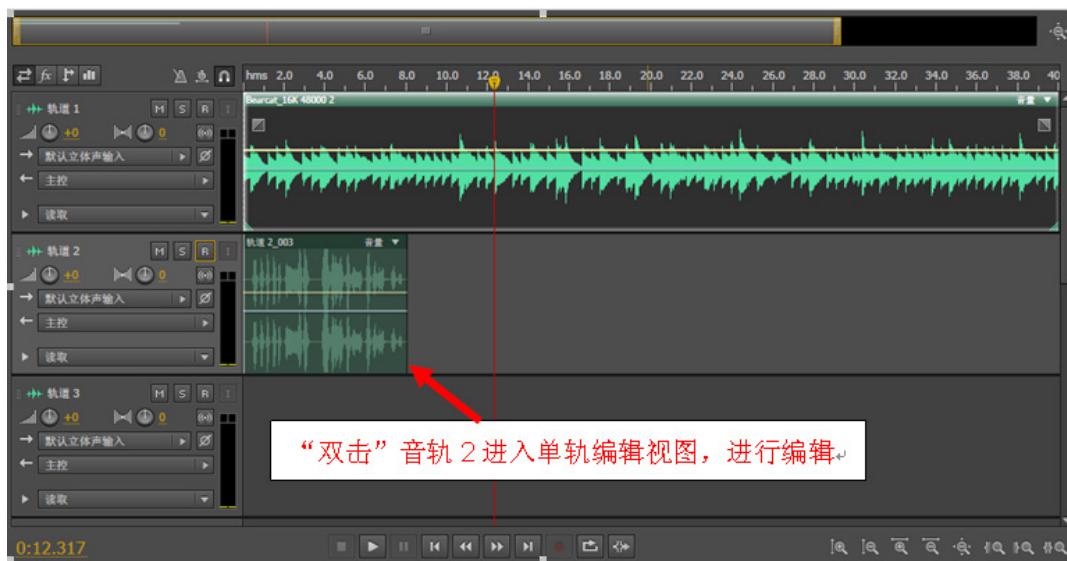
■ 创建项目后，那我们先插入一首背景音乐，在背景音乐的烘托下进行人声的录制效果更好。如下图所示，我们把背景音乐插入到指定的空音轨中吧，示例为空音轨 1，右键点击音轨 1 选择“插入” – “文件”。如果需要插入几首音乐或音效，也是一样，只是要注意音乐不要放置重叠在一起，自己想要达到的效果除外。



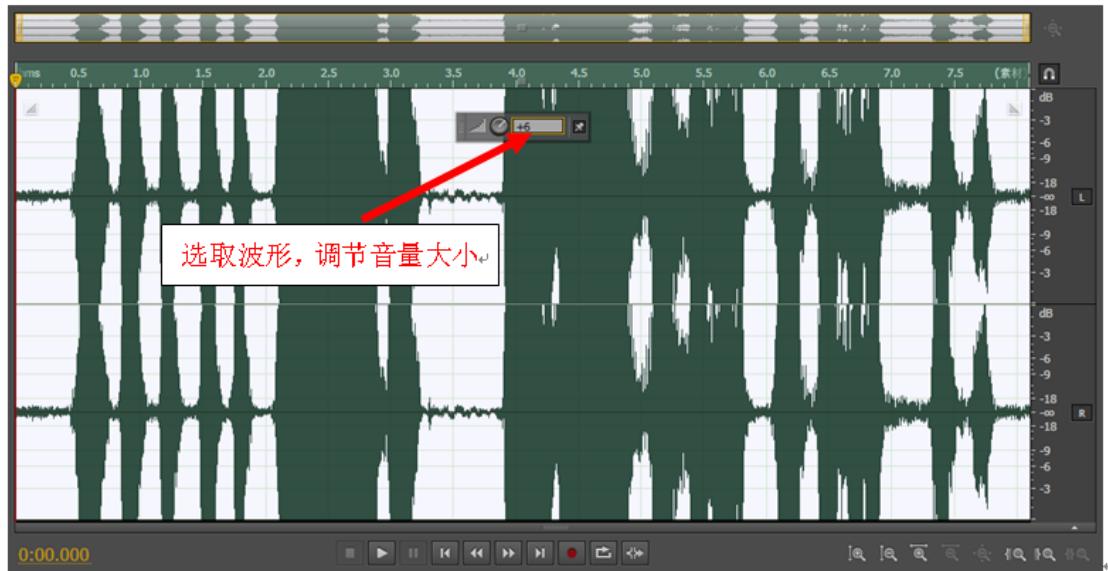
- 插入背景音乐后，我们点开音轨 2 中的红色录制准备按钮“R”，即表示将人声录制到该音轨 2 中。当然，你也可以将人声录制在其他的某一个空音轨中，在此示例为音轨 2。如下图所示，点开红色录制按钮就可以开始与背景音乐一起同步录制了。



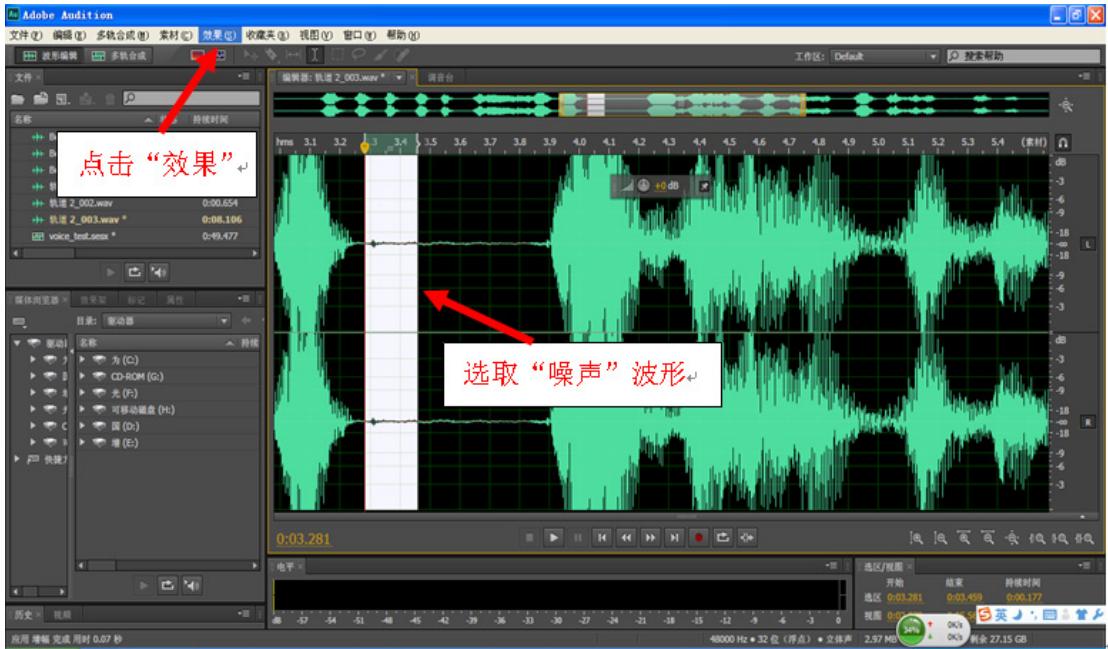
- 录制好人声后，那我们就开始处理人声吧，如图所示我们双击音轨 2 中的人声波形，即可进入单轨编辑视图中。



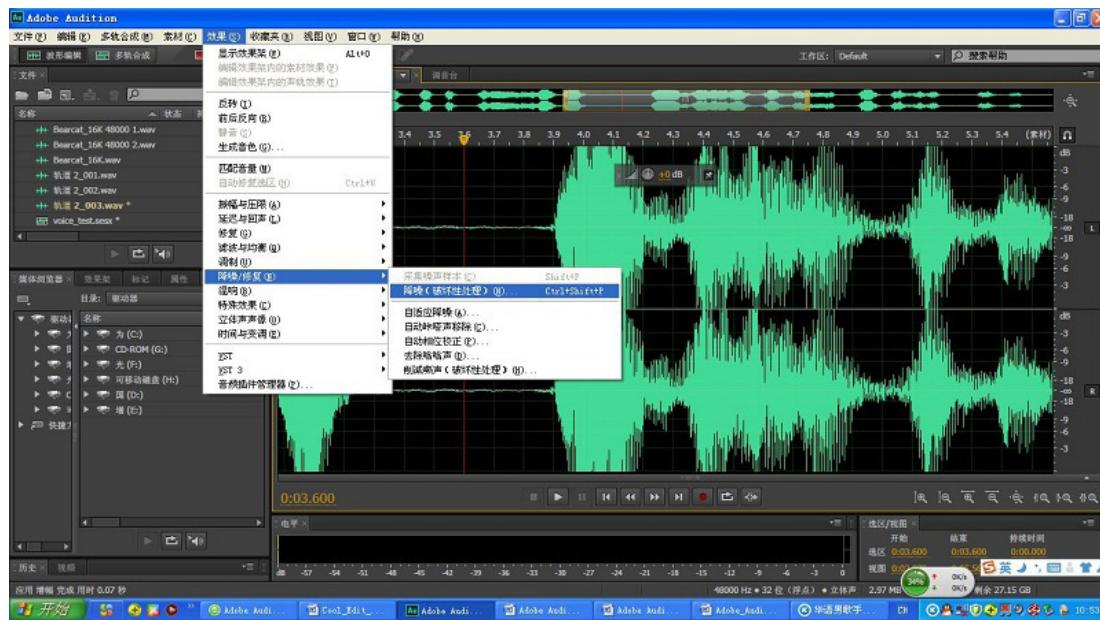
- 在单轨编辑视图中，我们可以试听人声音量的大小，观察波形的大小也可以判断，我们可以选中具体某段波形后对其进行音量的调整，如下图所示。



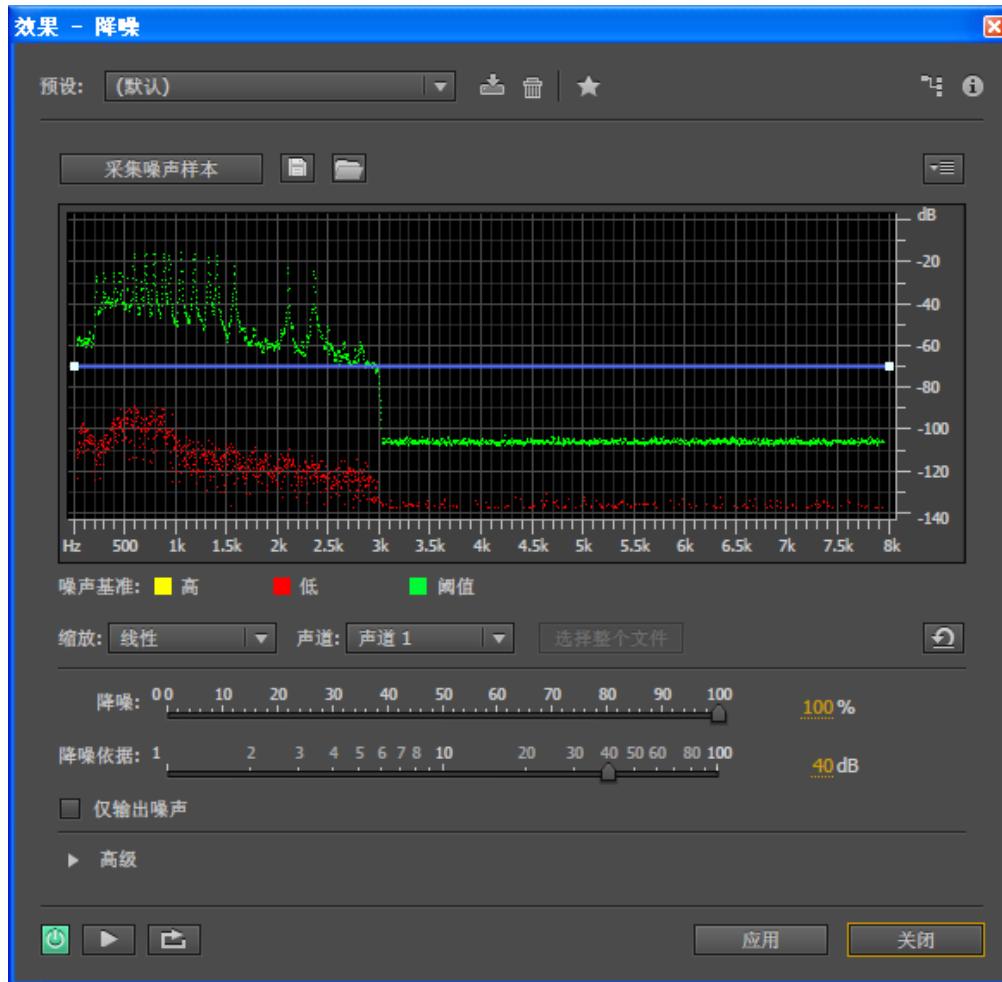
- 如何对人声进行降噪呢？其实也很简单，我们可以先选取一段噪音（电流声等内部环境音）波形，注意，不要选取人声部分哦。其他的环境音如（鼠标的点击声、咳嗽声等）可以选取该波形后，按删除键即可。选取好噪音波形后，我们点击“效果”，如下图所示。



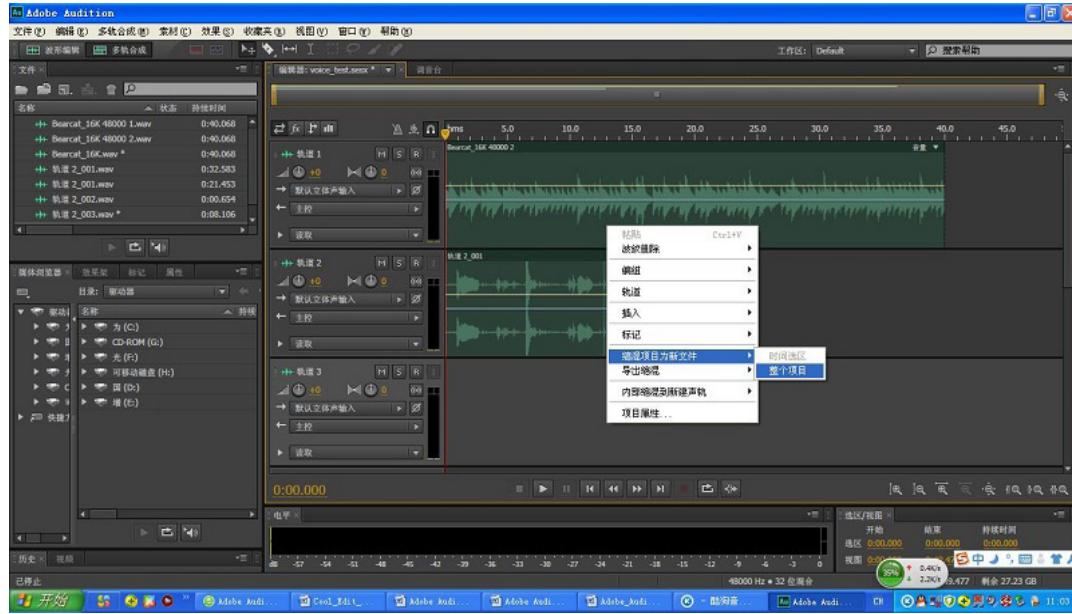
■ 点击“效果”后，我们找到并选择“降噪 / 恢复” – “降噪(处理)”。



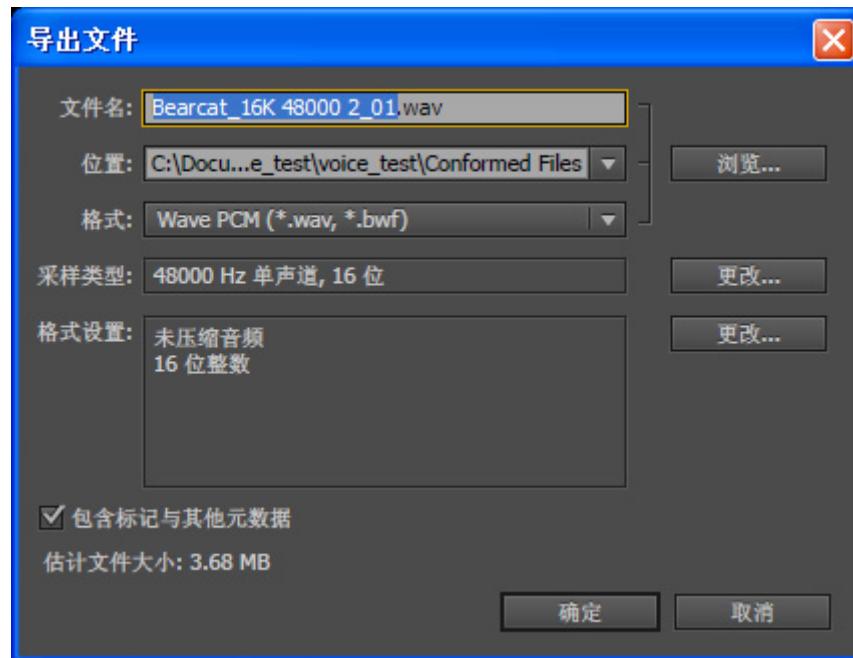
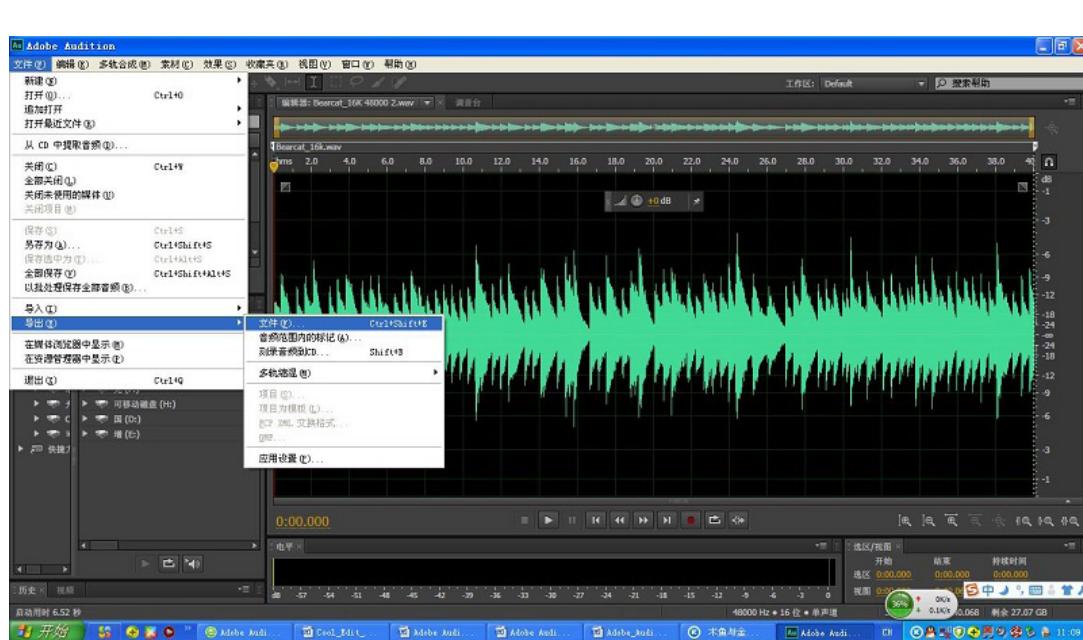
- 打开窗口后，我们首先采集刚才所选取的噪音，点击“捕捉噪音样本”。采集后就可以应用到全部波形文件中了，接着点击“选择完整文件”，最后点击“应用”，就完成人声的基础降噪处理。



■ 当然，对人声的处理还有很多其他的功能，比如混响、回声、变调变速、压限等功能都可以在“效果”中去实现。我们先看看如何将背景音乐和人声合成在一起吧。如下图所示，右键点击任意空音轨，选择“混缩混音项目为新文件” – “完成混音”。



- 合成完成后，会自动切换到单轨视图，如下图所示，我们点击“文件”，选择“导出”->“文件”你会看到，在该窗口中可以查看或更改保存文件的具体参数，选择好参数后，点击“确定”就完成文件的导出了。



Copyright® 2020 by HOLTEK SEMICONDUCTOR INC.

使用指南中所出现的信息在出版当时相信是正确的，然而 Holtek 对于说明书的使用不负任何责任。文中提到的应用目的仅仅是用来看说明，Holtek 不保证或表示这些没有进一步修改的应用将是适当的，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。Holtek 产品不授权使用于救生、维生从机或系统中做为关键从机。Holtek 拥有不事先通知而修改产品的权利，对于最新的信息，请参考我们的网址 <http://www.holtek.com/zh/>。