Voice MCU Workshop V2.3
User Guide

Revision: V1.50 Date: September 15, 2017

www.holtek.com

Table of Contents

1 Voice MCU Workshop Description

Development platform Introduction and Software Installation

Characteristics

The HOLTEK Voice MCU Workshop V2.3 is a software development platform for Voice MCU product development. Using a simple graphical user interface, it allows users to easily integrate the project code with their audio files and complete their audio product designs in an easy and efficient manner. The code will be automatically generated and stored in a voice MCU in a certain format with the compressed audio files stored in external flash.

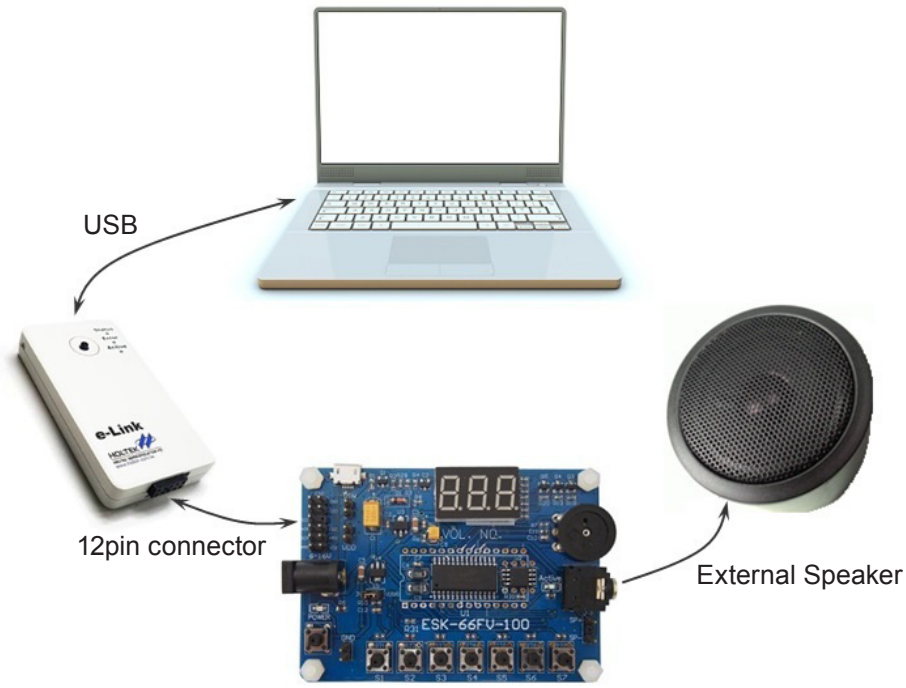
System Requirements

Windows Version	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

System Configuration

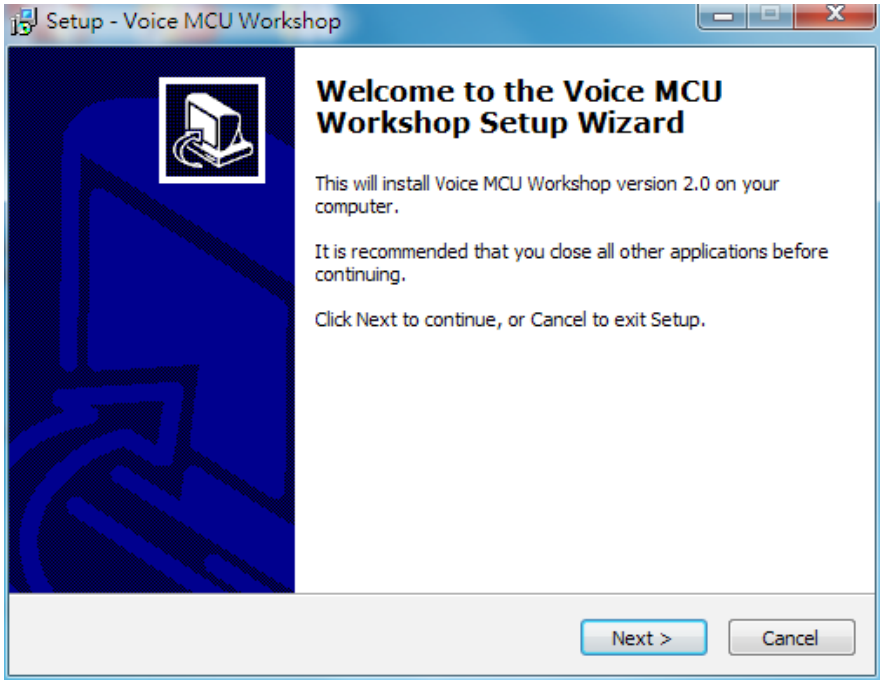
The complete system has both software and hardware components:

- S/W: Voice MCU Workshop V2.3
- H/W: ESK-66FV-100 EV board
  - e-Link – user provided
  - Speaker for broadcasting – user provided

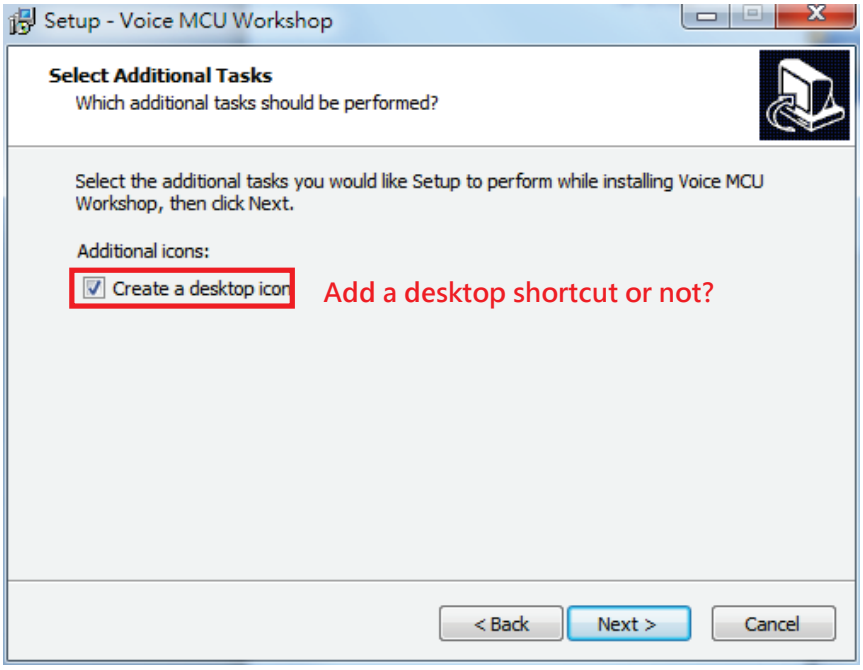


S/W Installation

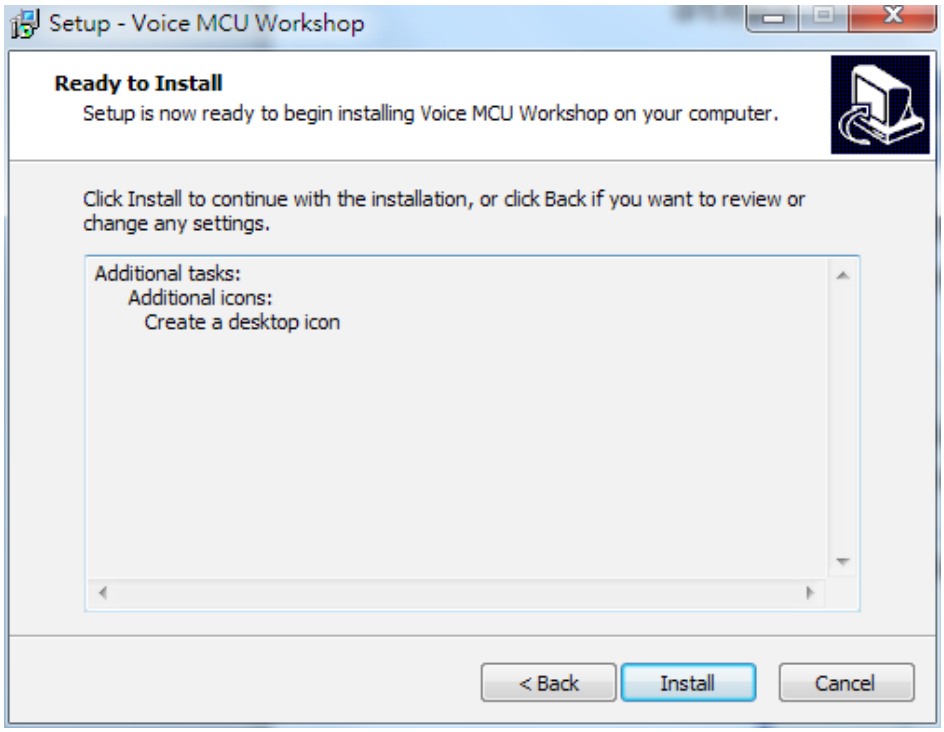
Step1. Double-click on the install icon  and the following screen will appear:



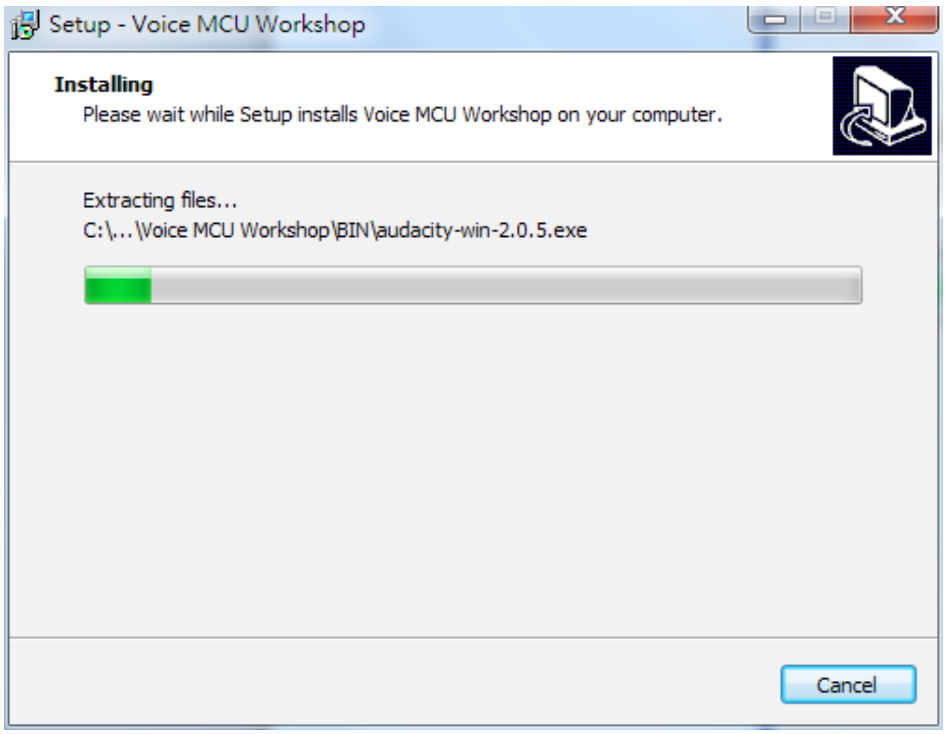
Step2. Click the "Next" button and the following screen will appear:



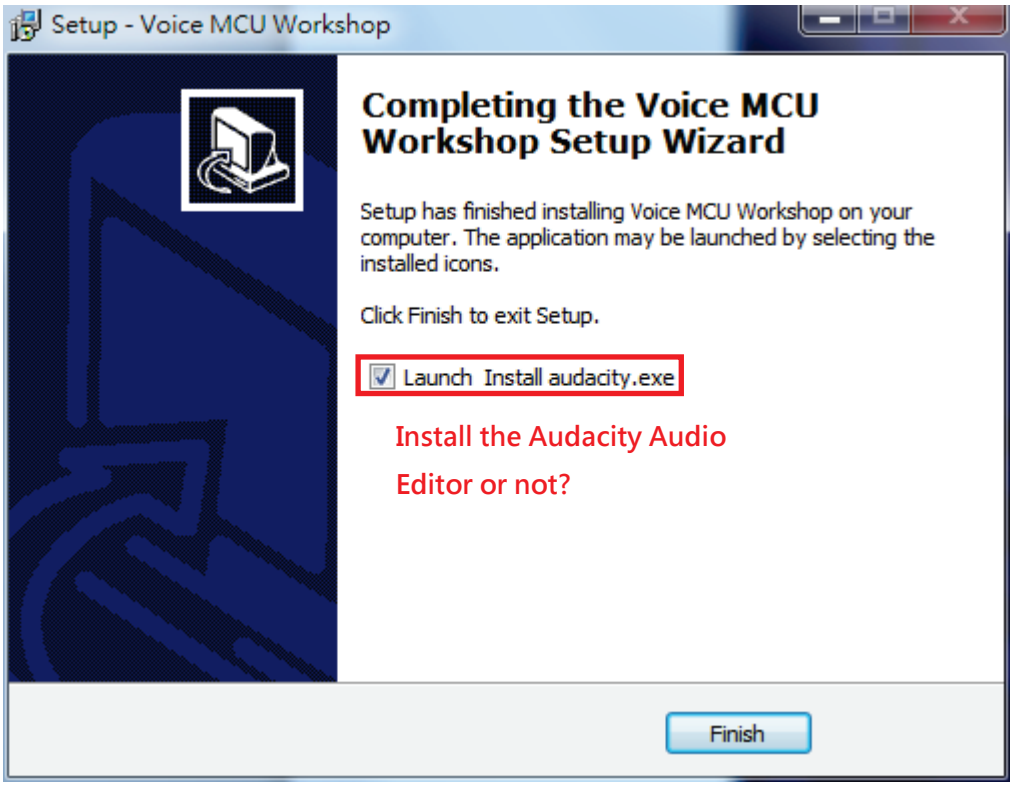
Step3. Click “Next” and the following screen will appear:



Step4. Click "Install” to continue with the installation.



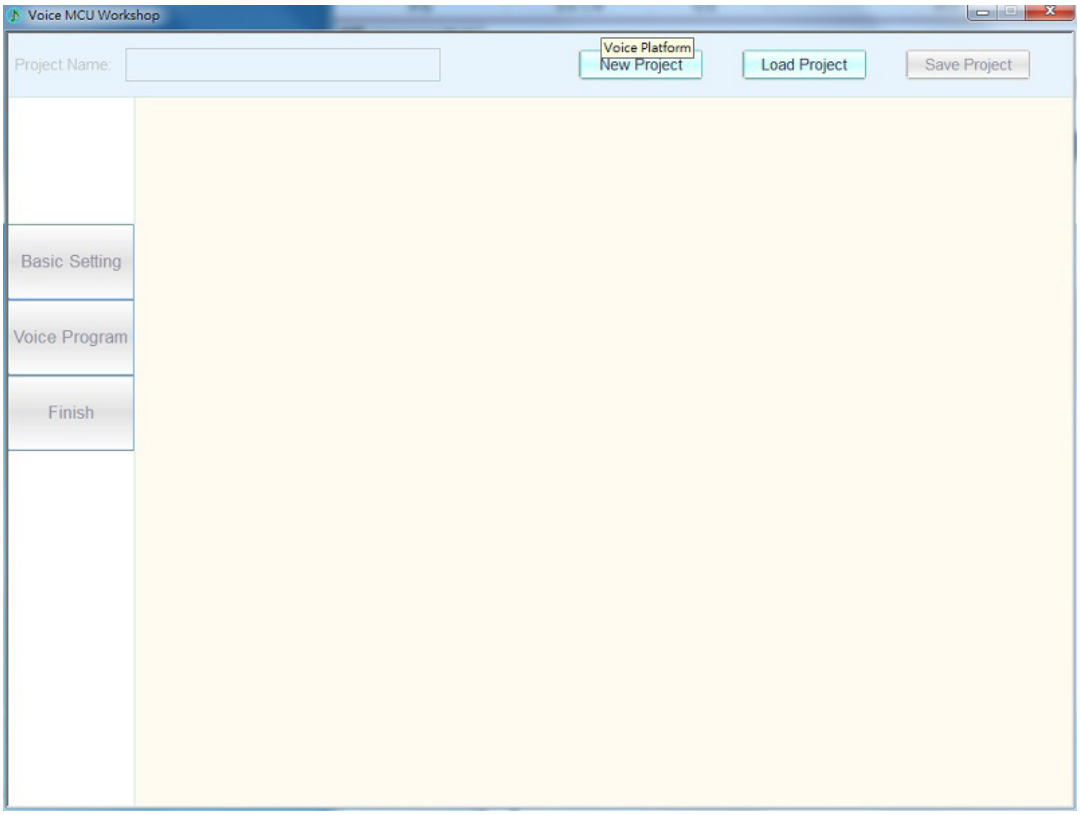
Step5. Press the “Finish” button to finish the setup.



S/W Operation Quick Start

Start the Voice MCU Workshop V2.3

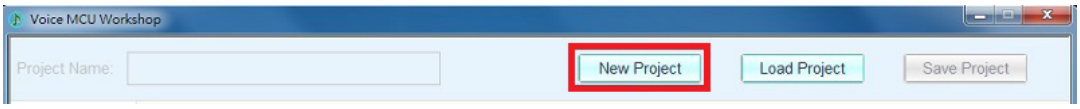
Double-click the “Voice MCU Workshop V2.3.exe” icon, the following screen will appear:



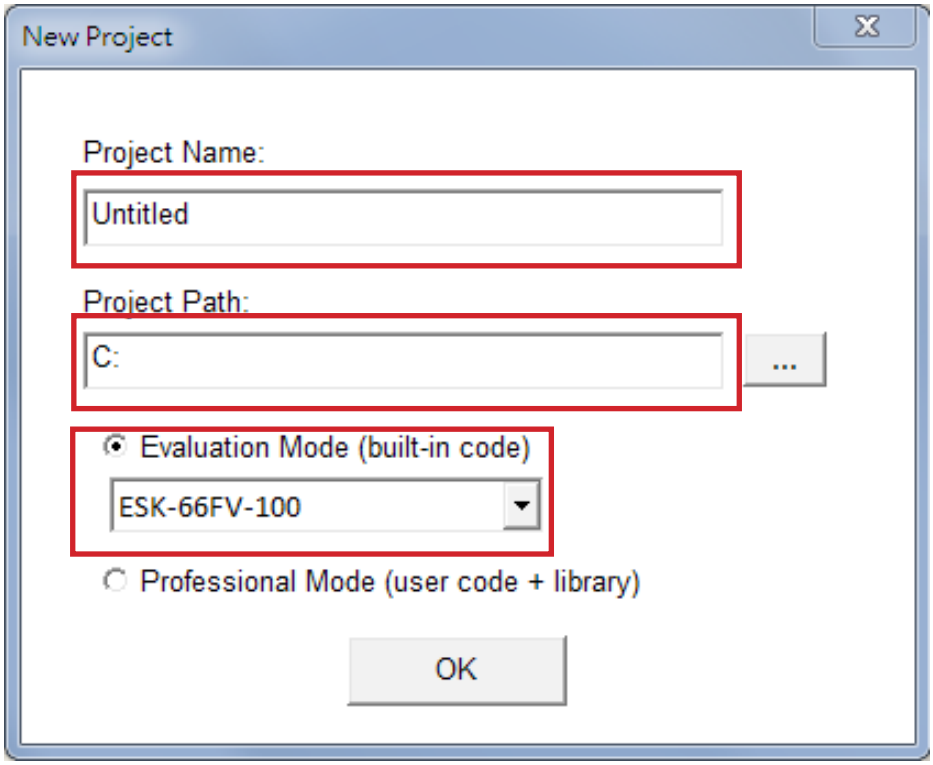
Create a New Project

Evaluation Mode:

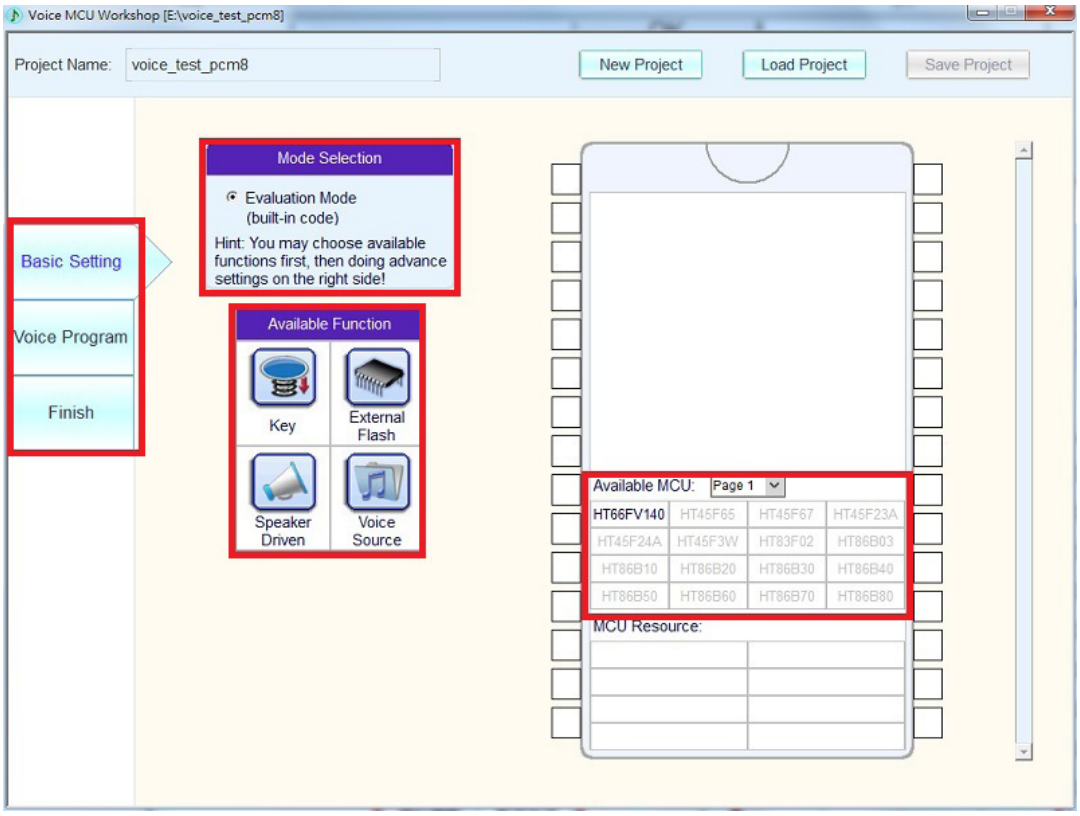
Step1. Press the "New Project" button to create a new project



Step2. Enter a project name and project path, check the Evaluation Mode and Evaluation Mode built-in code, as shown below.



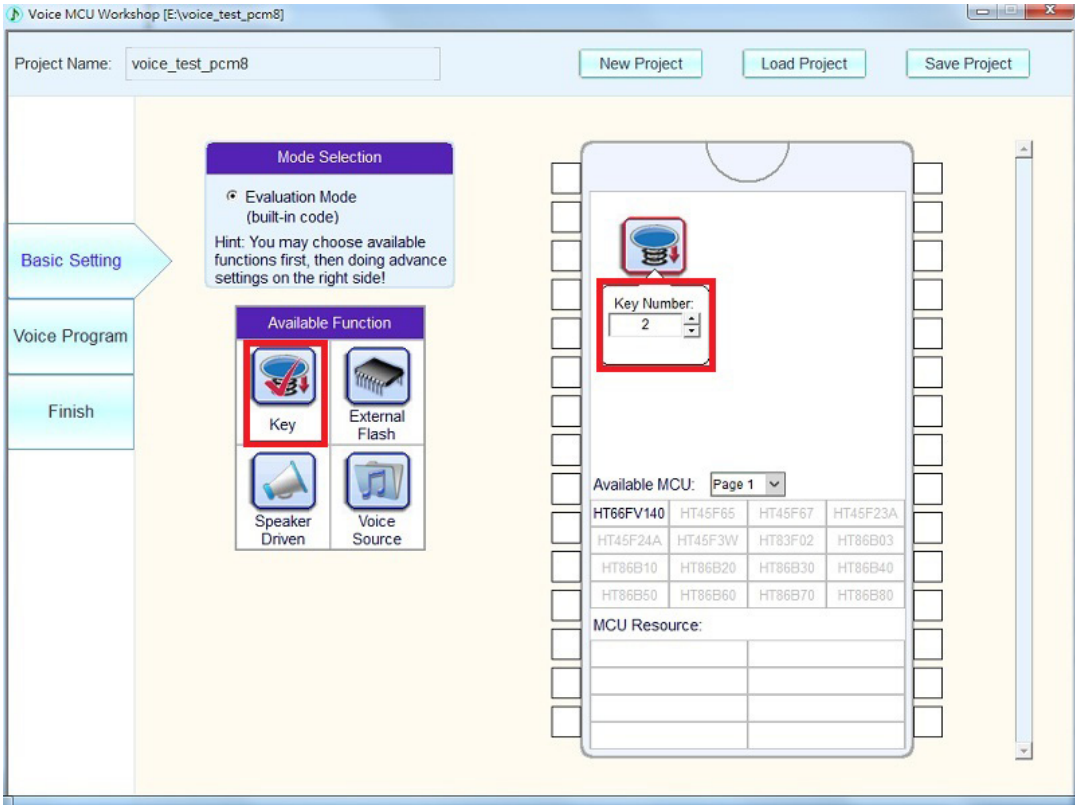
Step3. Click “OK” and then the following window, which has three selectable pages on the left, will appear. The Basic Setting page includes a “Mode Selection” box. Here we have already selected the Evaluation Mode, “Available Function” box and “Available MCU” box, as shown below:



Step4. Use and setup the four available functions:

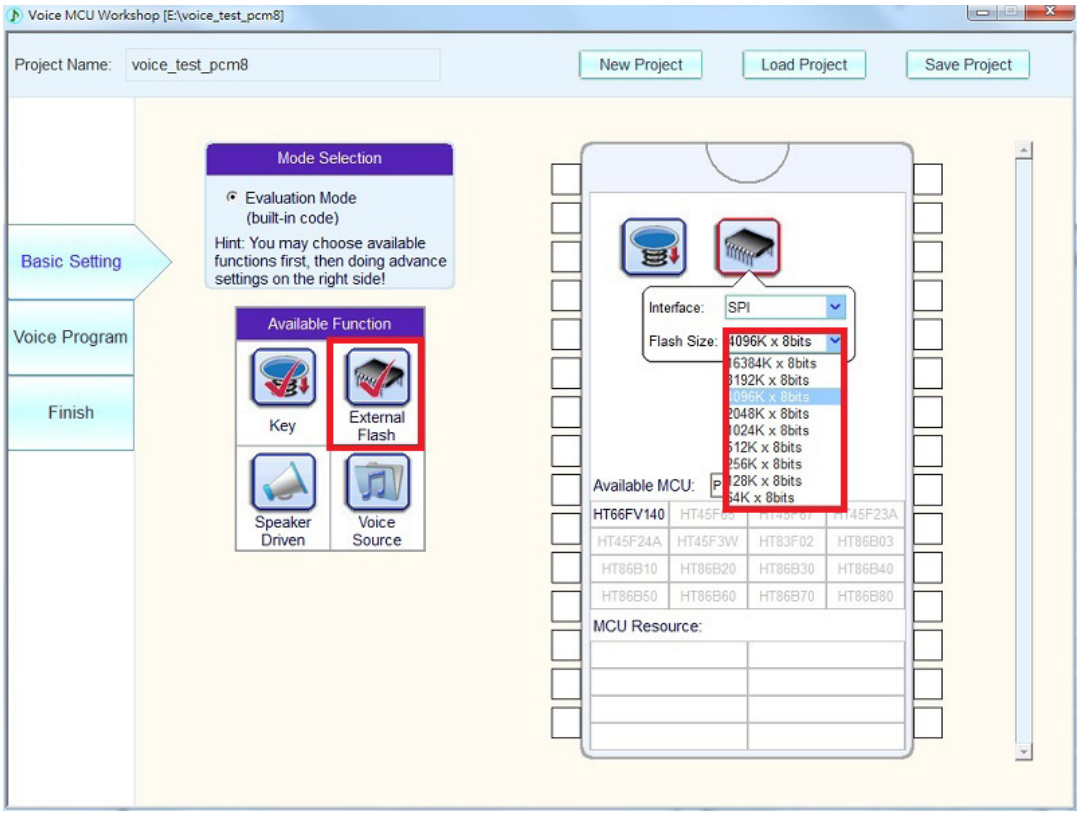
■ Key Function:

- Click the "Key" button to load/remove the function to/from the MCU.
- Click the "Key" icon in the MCU block on the right to setup the required key number, as shown below:



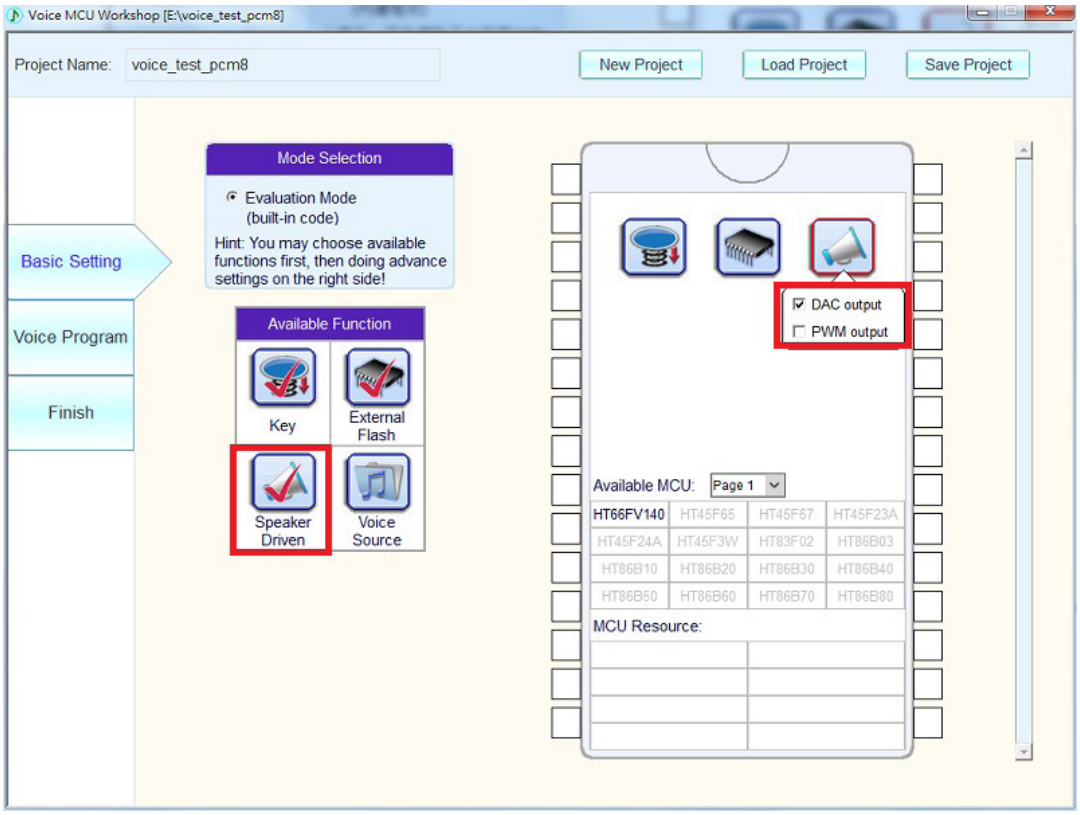
■ External Flash Function:

- Click the "External Flash" button to load/remove the function to/from the MCU.
- Click the "External Flash" icon in the MCU block on the right to select the Flash size, as shown below:



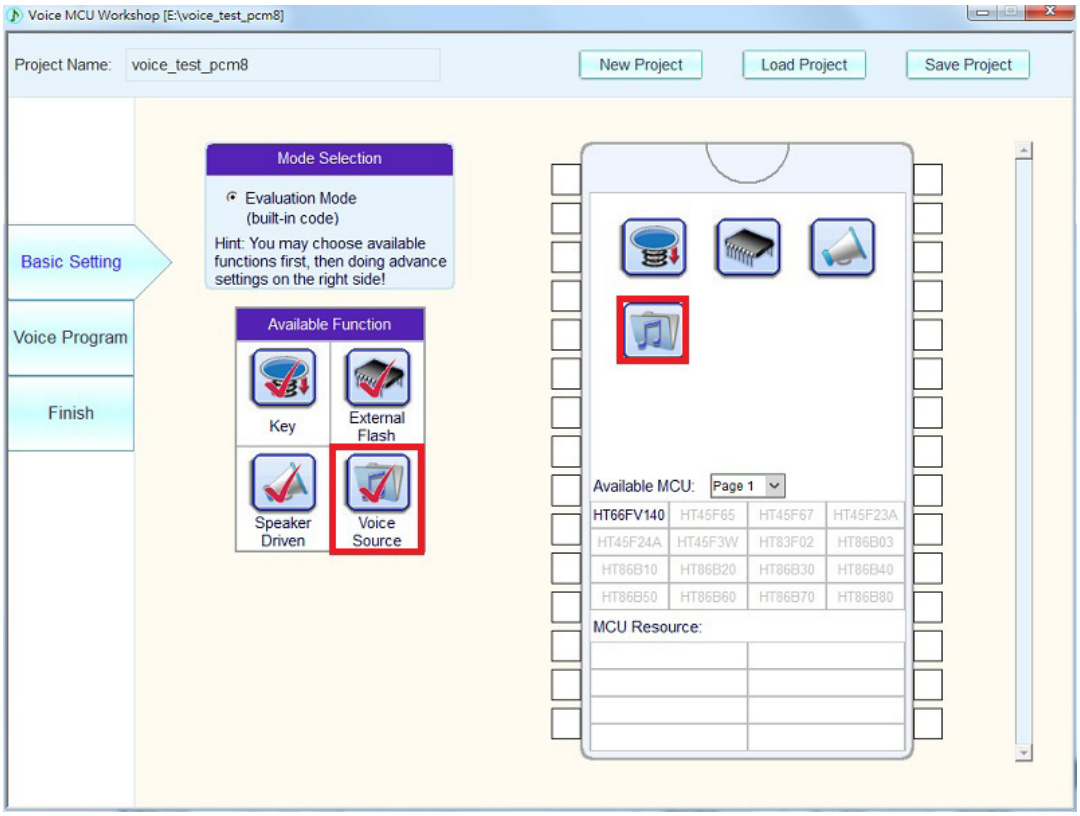
■ Speaker Driven Function:

- Click the "Speaker Driven" button to load/remove the function to/from the MCU.
- Click the "Speaker Driven" icon in the MCU block on the right to setup the drive mode - at the present time only the DAC output mode is supported- as shown below:

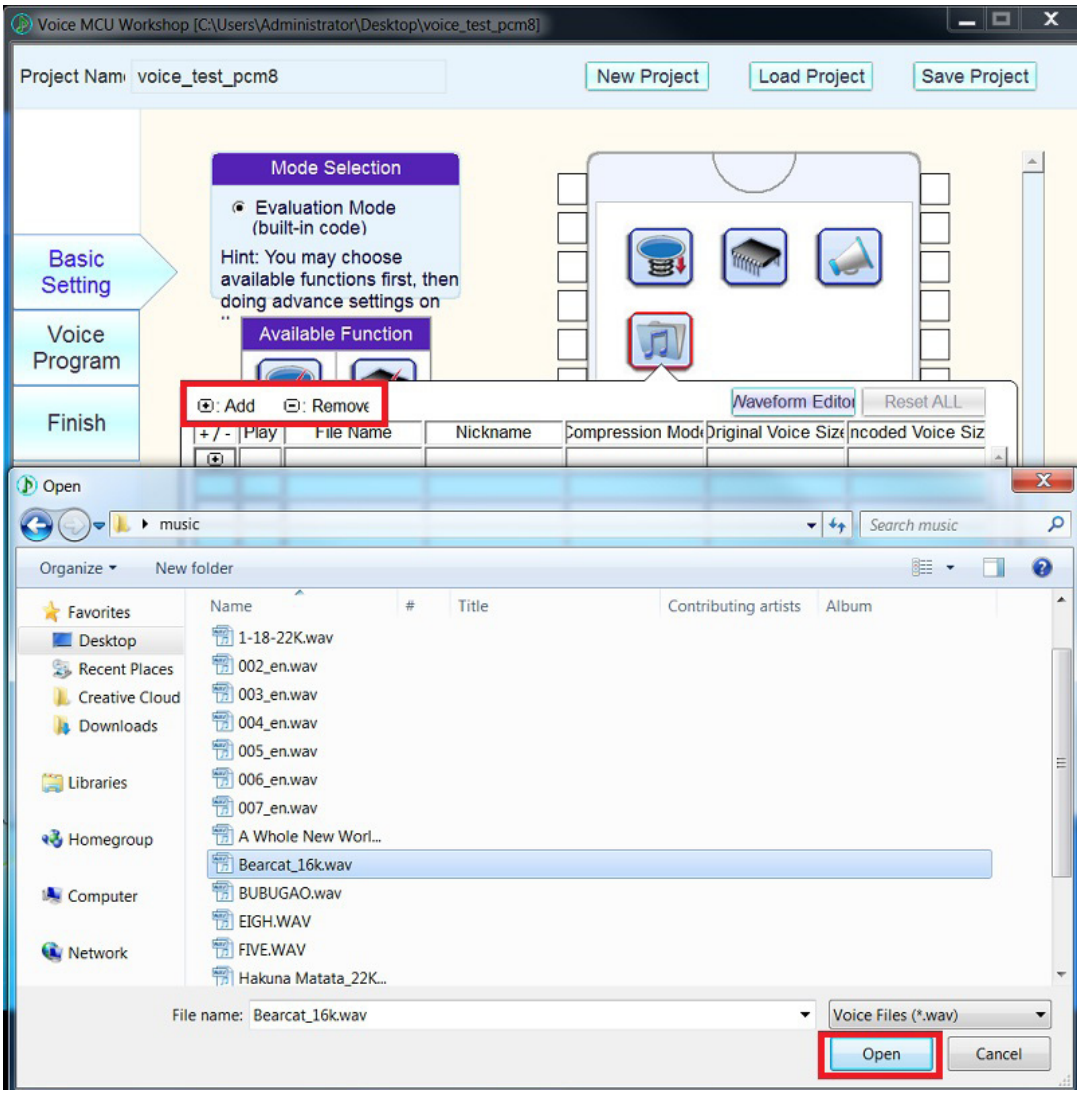


■ Voice Source Function:

- ① Click the "Voice Source" button to load the function to the MCU.

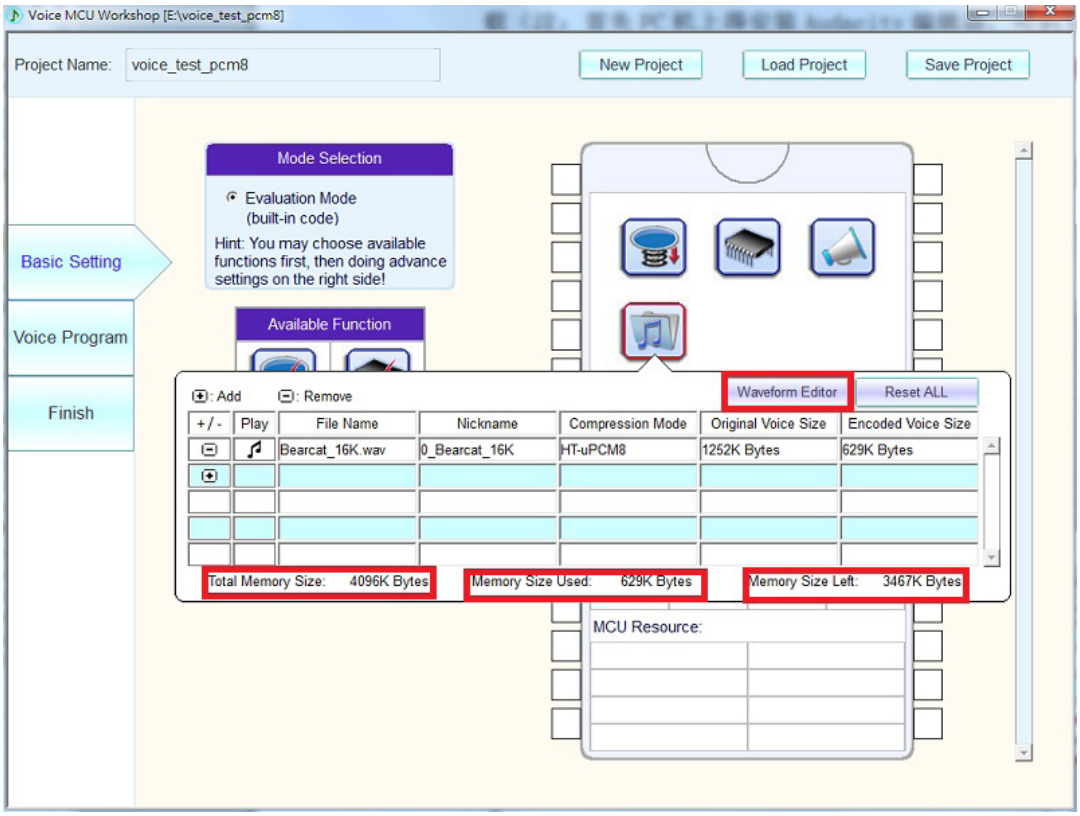


- ② Click the "Voice Source" icon in the MCU block on the right to add or remove ".wav" files, as shown below:

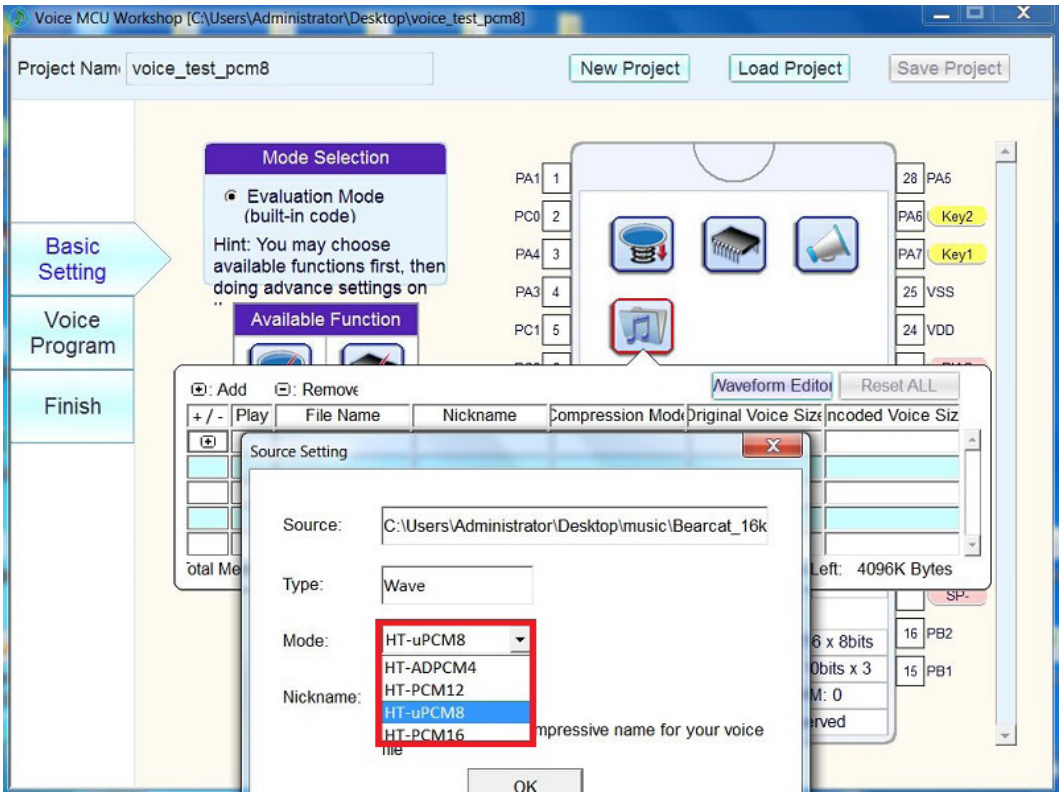


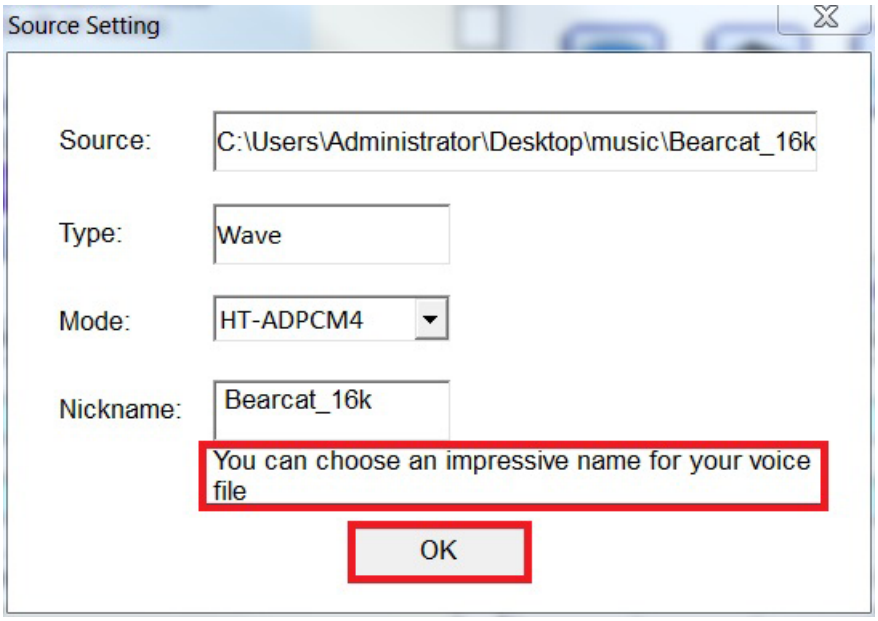
Note: For the maximum frequency limit for the added voice source, refer to the Library Establishment Information section.

- ③ Before loading the voice source file, first click the "Waveform Editor" button to connect to the "Audacity" Audio Editor to process the voice source file after which it should be saved. Ensure that the Audacity software has been installed in advance, otherwise it must first be downloaded from the website http://audacity.sourceforge.net/, Refer to the Audacity Quick Start for Audacity application details. After loading the file successfully, the "Total Memory Size", "Memory Size Used", "Memory Size Left" information is displayed, as shown below.

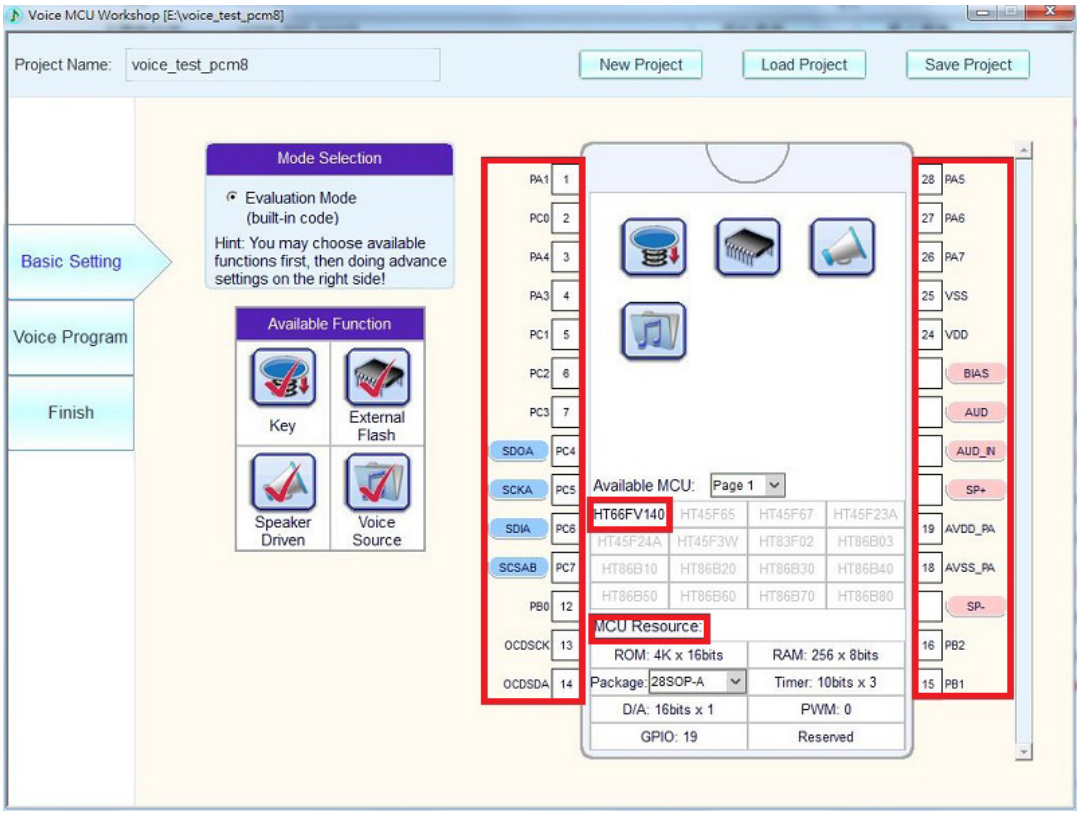


- ④ In above ② , press "Open file" and the source setting dialog box, including voice source information, compression mode settings, etc., appears. After adjusting the settings, click "OK" to complete the voice source design. See the figure below.

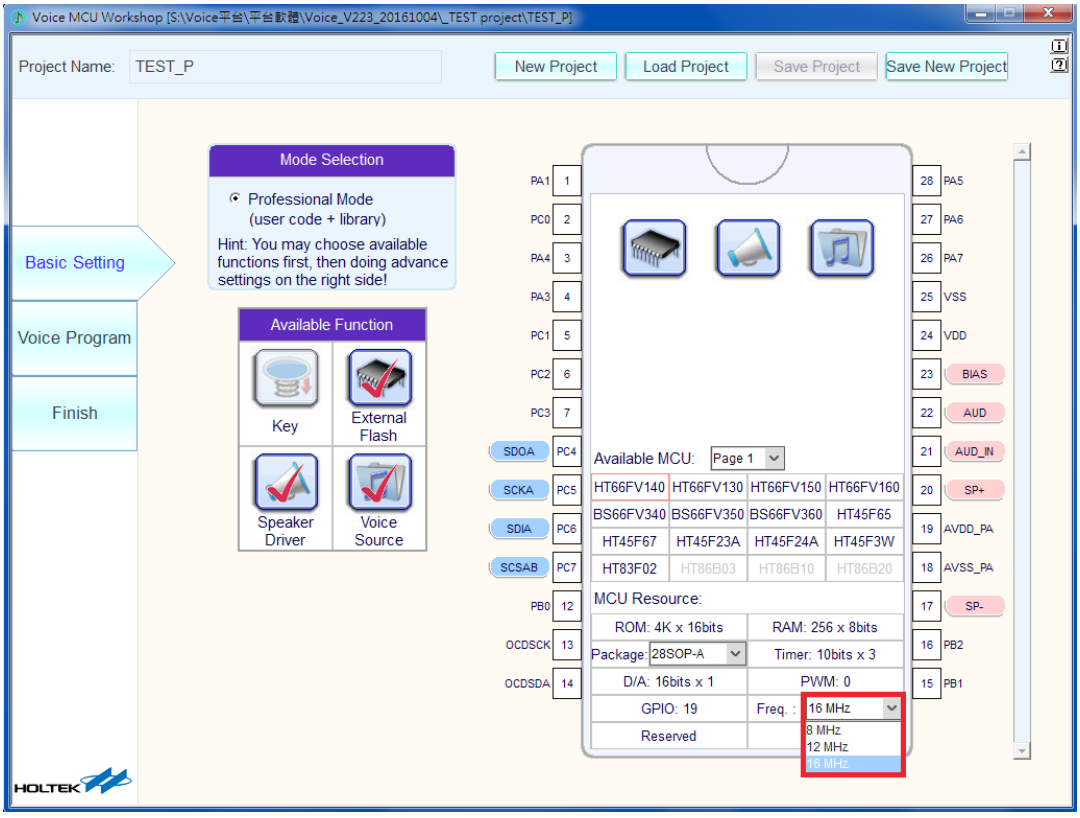




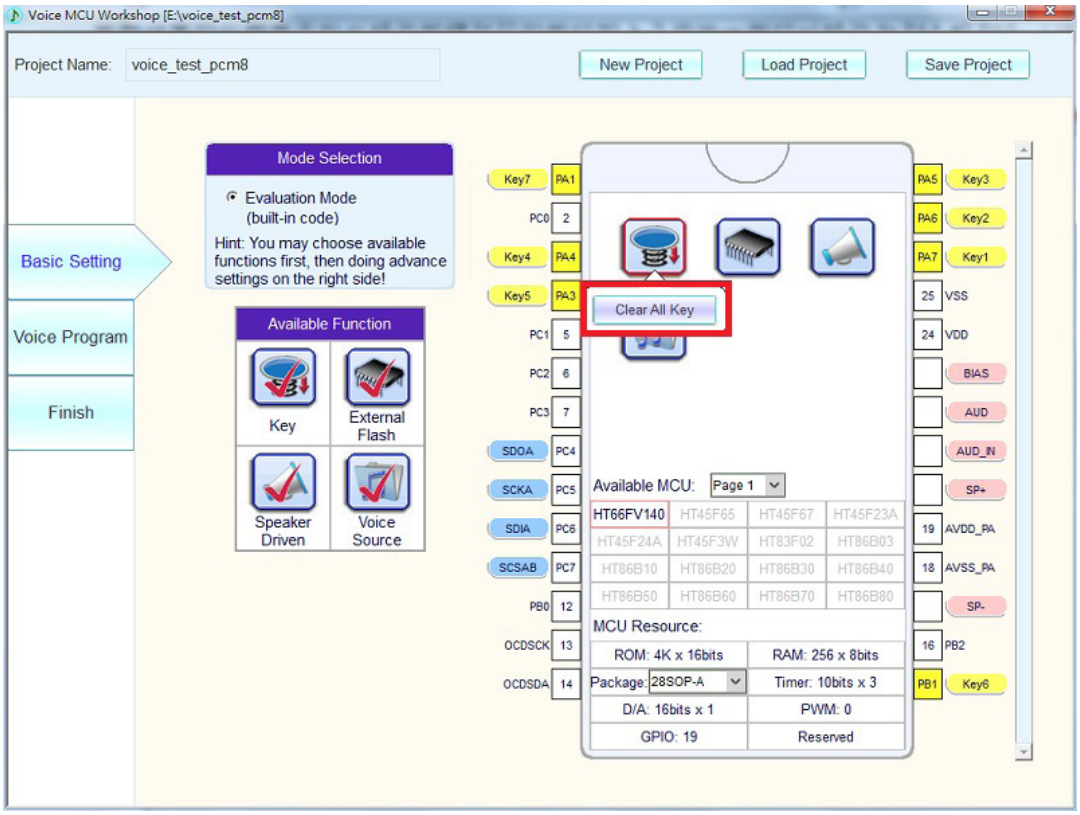
Step5. In this example, we choose the HT66FV140 MCU. When this is done the related selected MCU information including the MCU pins and internal resources as used by the available functions are displayed, as shown in the following figure.

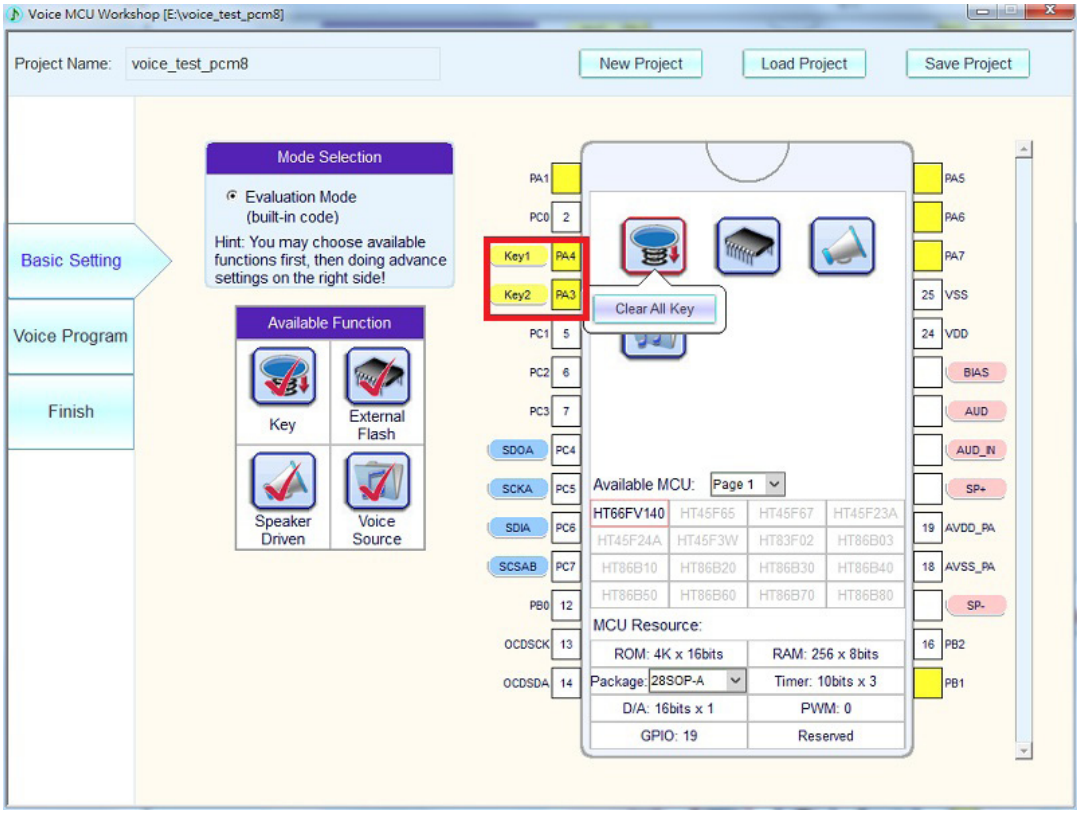


Note 1: Right-click the mouse button and select the MCU frequency, as shown below.

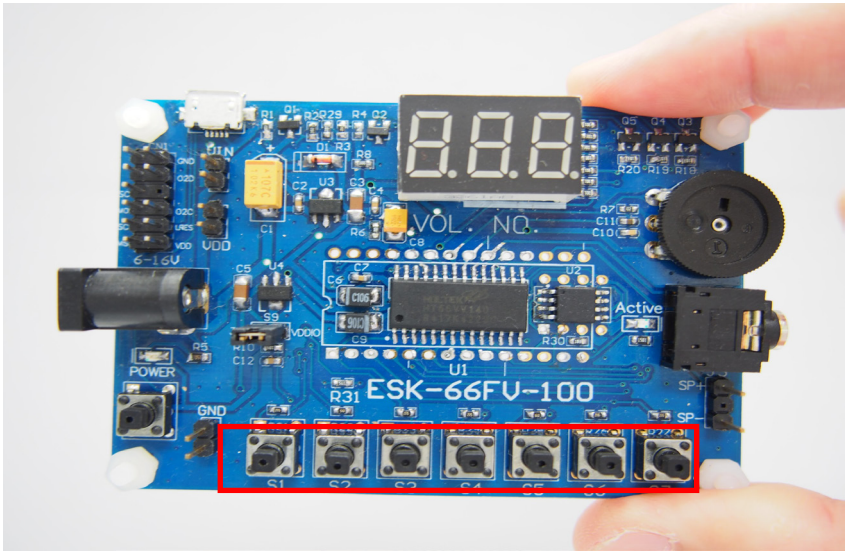


Note 2: Click the Key icon in the MCU block, then the “Clear All Key” selection appears. If it is selected the default key pins will be cancelled. You can move the mouse to the pin position you want to use and click, the pin will then be set as a key pin. In the following figures, the key pins have been set to PA4 and PA3. Note: Apart from the board, it is not possible to see the pin connection relationship. The evaluation board pins and the corresponding MCU pins are shown in the figure below.





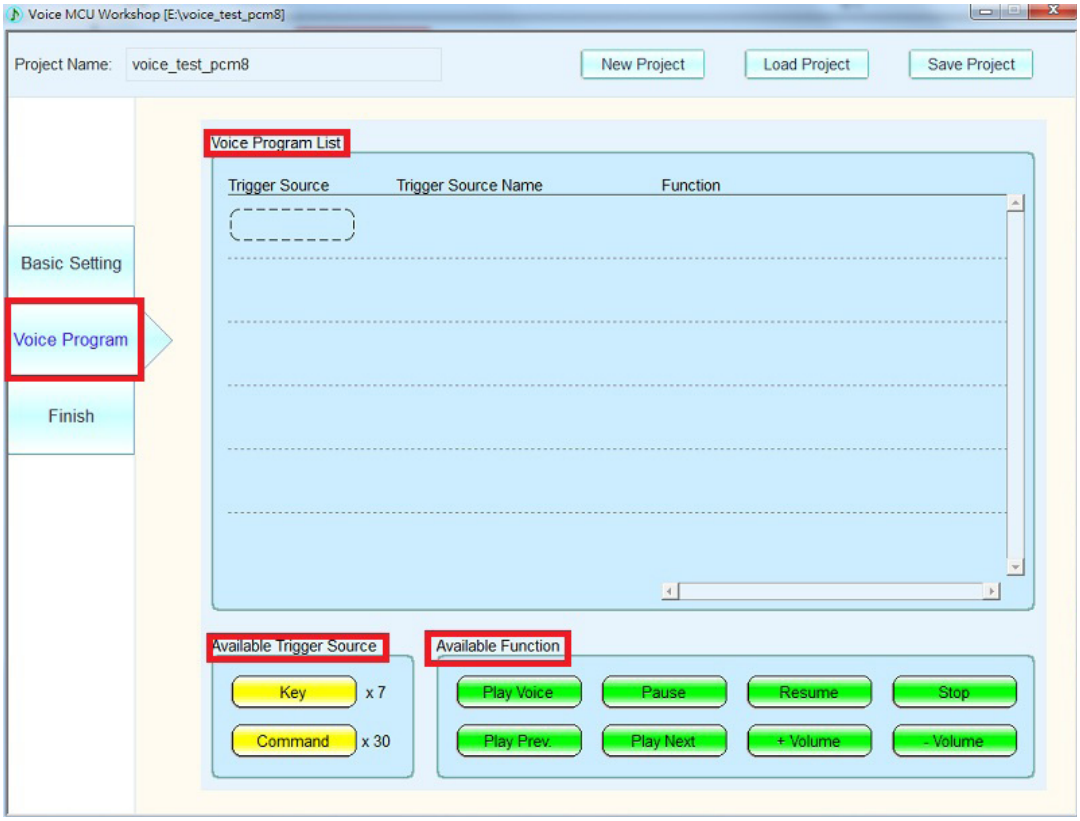
The following shows the Evaluation Board keys and the corresponding MCU pins:



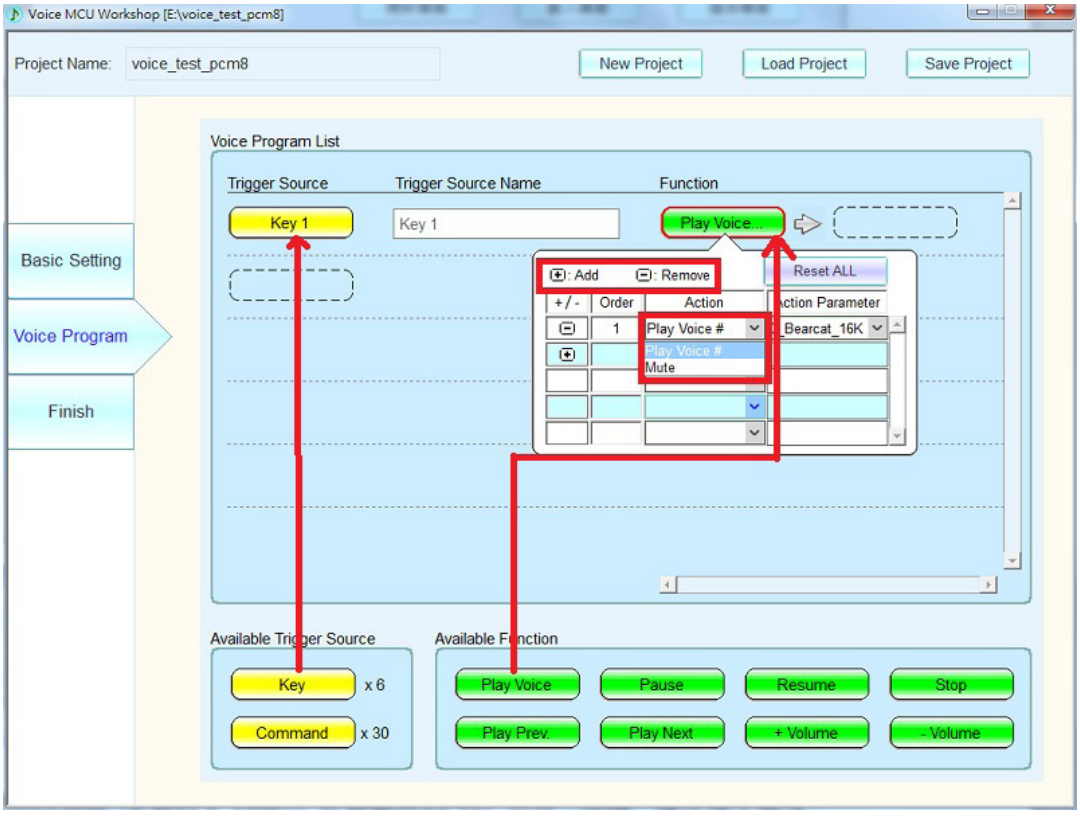
"S1->PA7, S2->PA6, S3->PA5, S4->PA4, S5->PA3, S6->PB1, S7->PA1"

Step6. After finishing the basic setting, switch to the voice program Page.

■ Click “Voice Program” to enter the project logical design page. The screen is shown below.



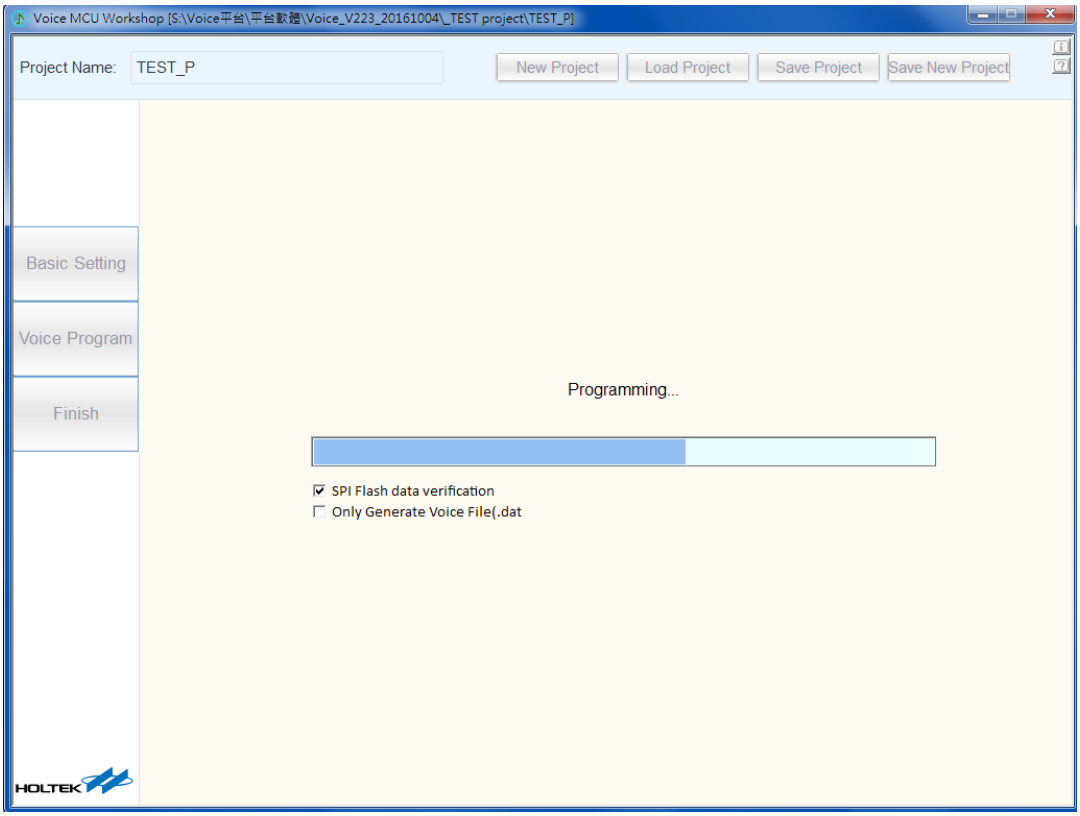
■ Select an available trigger source and an available function, click and then drag it to the voice program list appropriate position as shown below. Note that in the "Play Voice" function, it is necessary to choose between "Play Voice" or "Mult" for added voices. In this way we can make a sentence through the "add" and "remove" operations.



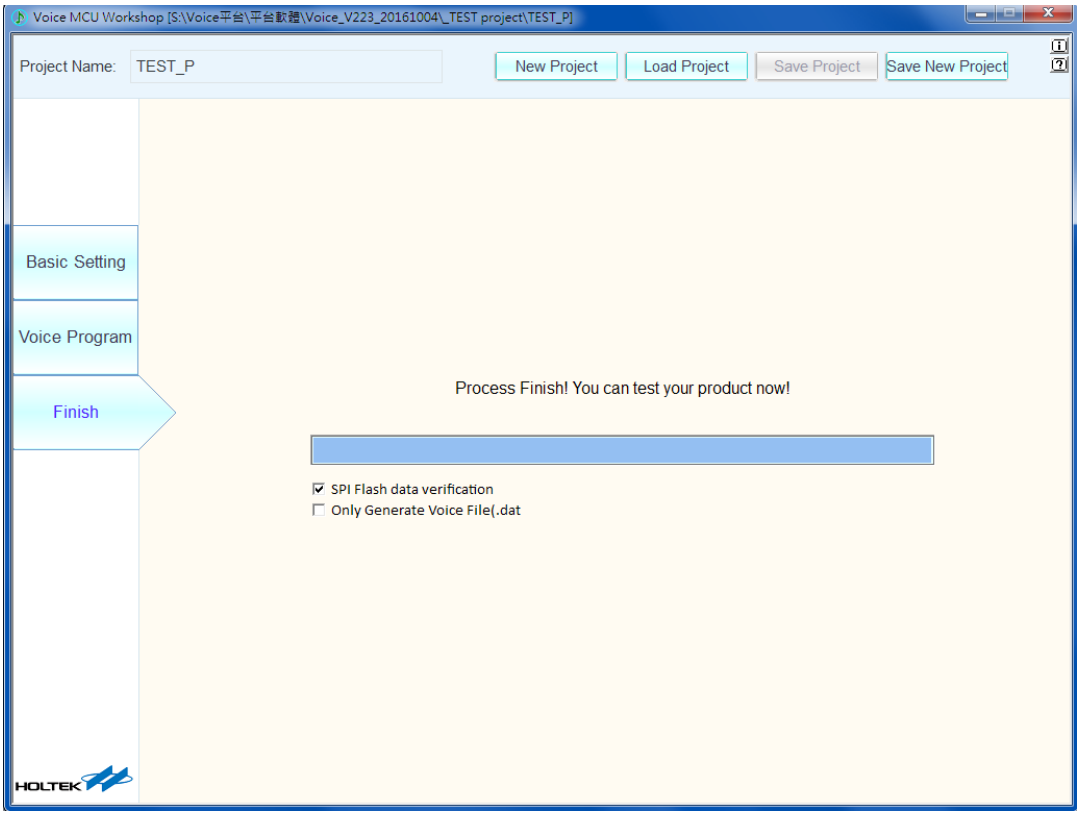
Step7. After completing the connection of the BiCE000ELINK0B (B Board e-Link), Evaluation Board and PC, click “OK” to start programming. Note that an external power source is not allowed here.

■ Hardware connection:

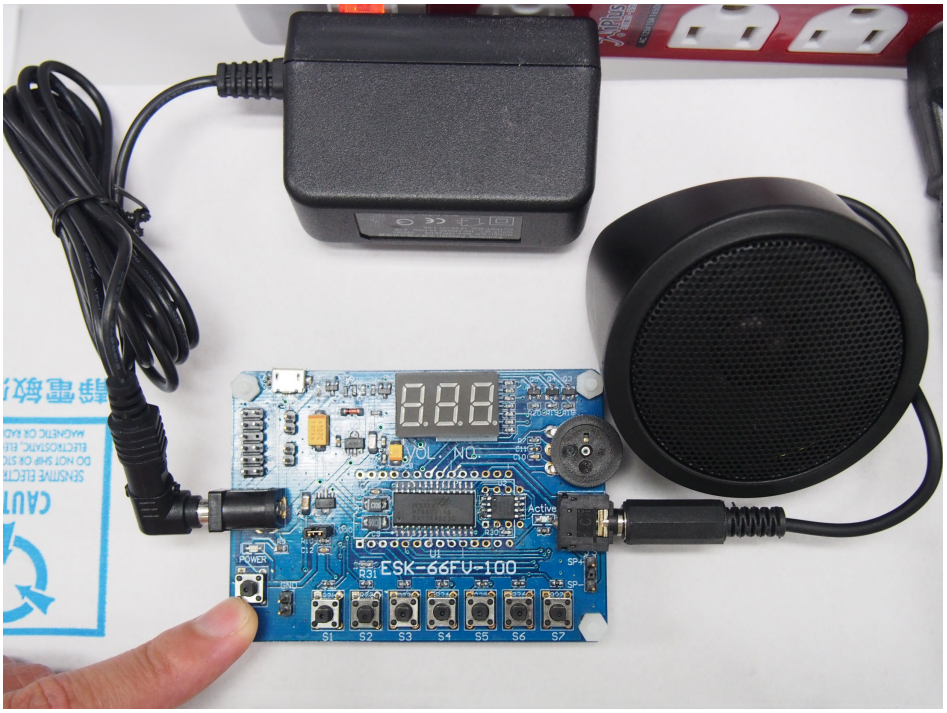
■ Programming



■ Programming finish

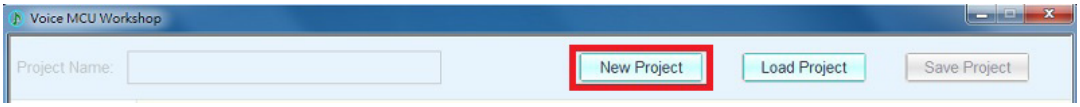


■ Plug in a speaker and power up either using the e-Link or unplug the e-Link and connect to an external power supply, External POWER supply after insert, please press the POWER button, the POWER LED and Active LED lights up at the same time on behalf of the function can be demo.

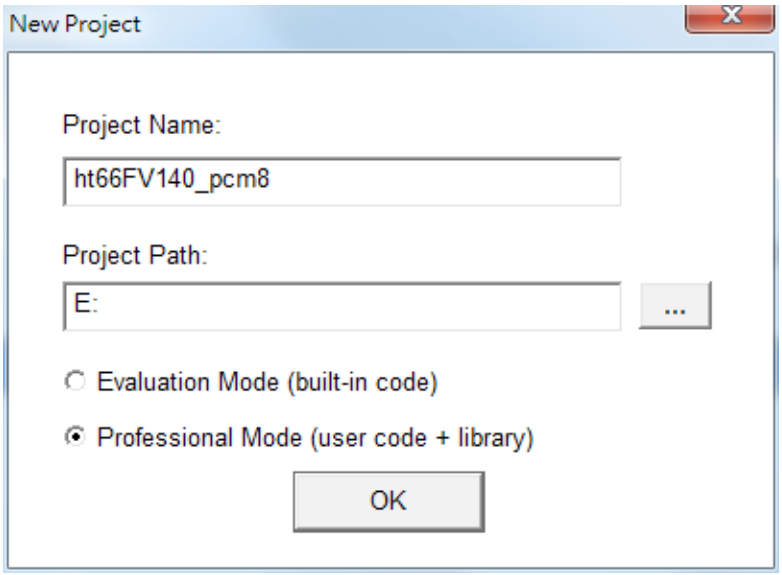


Professional Mode:

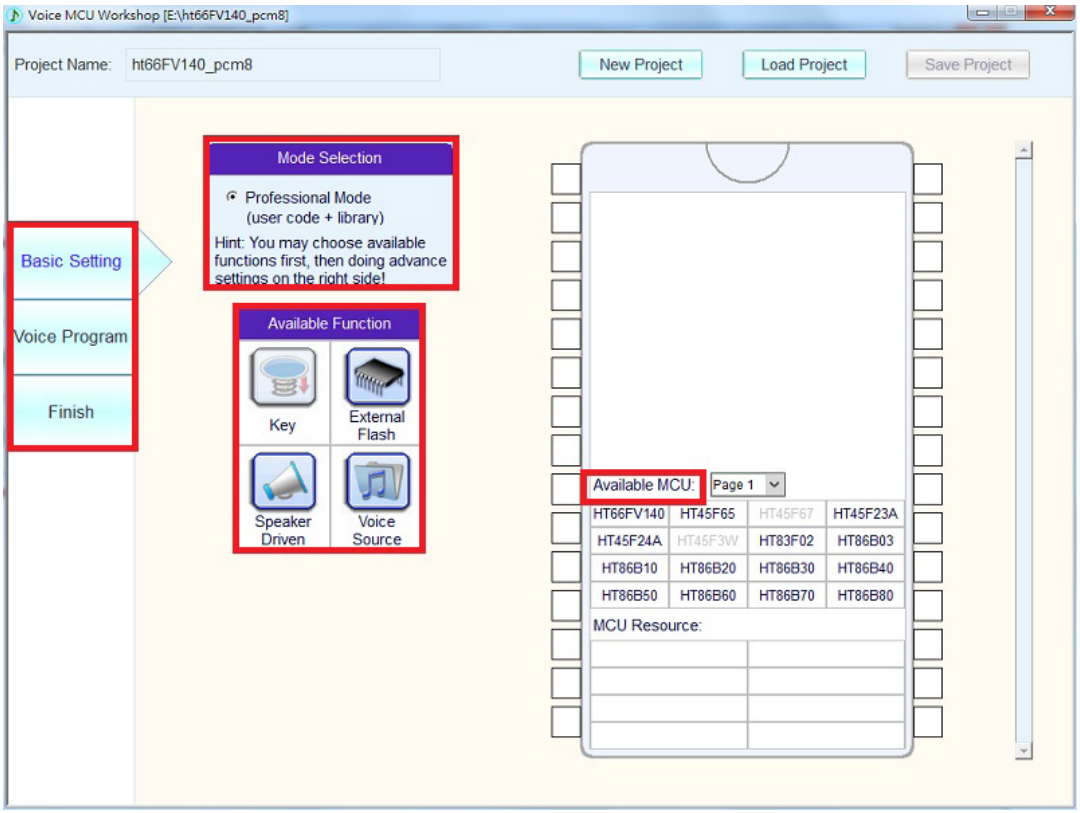
Step1. Select "New Project" to create a new project



Step2. In the New Project setting window, enter “Project Name”, “Project Path” and check the “Professional Mode” as shown below.



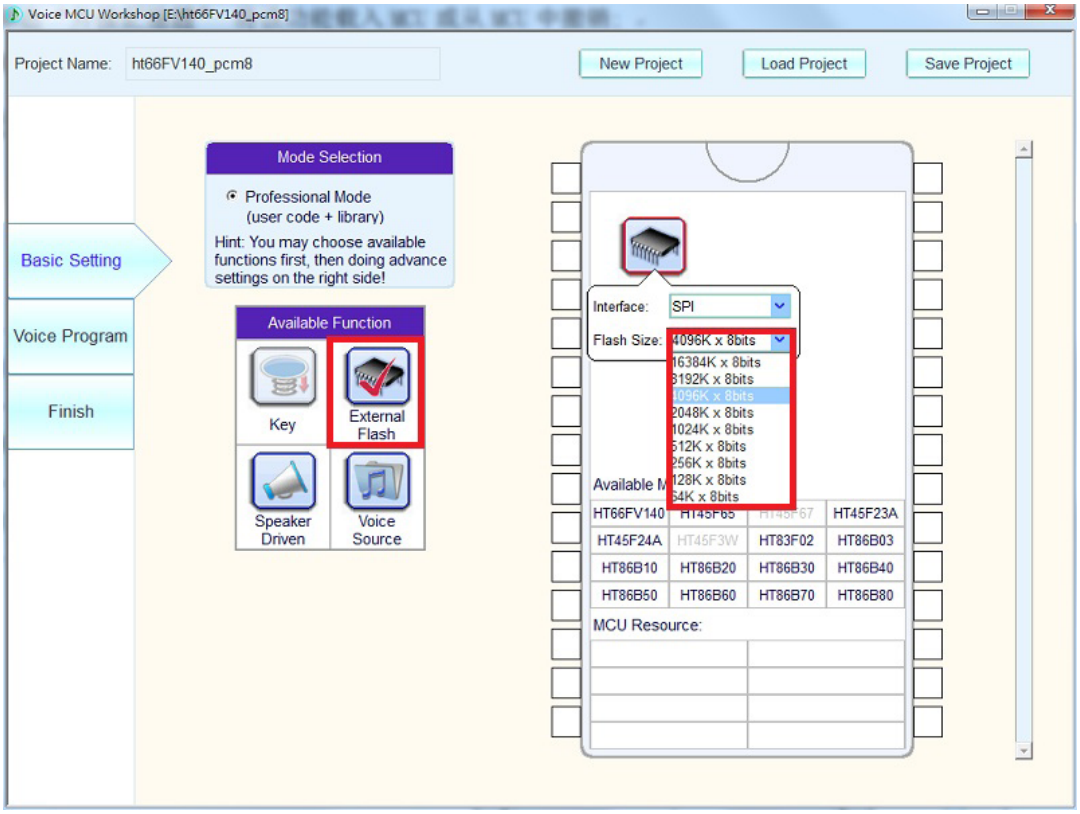
Step3. Click “OK” and then a window which has three optional pages on the left will appear. The Basic Setting page includes “Mode Selection” in which we have selected the Professional Mode, “Available Function” and “Available MCU” selection boxes, as shown below.



Step4. Use and setup the three available functions:

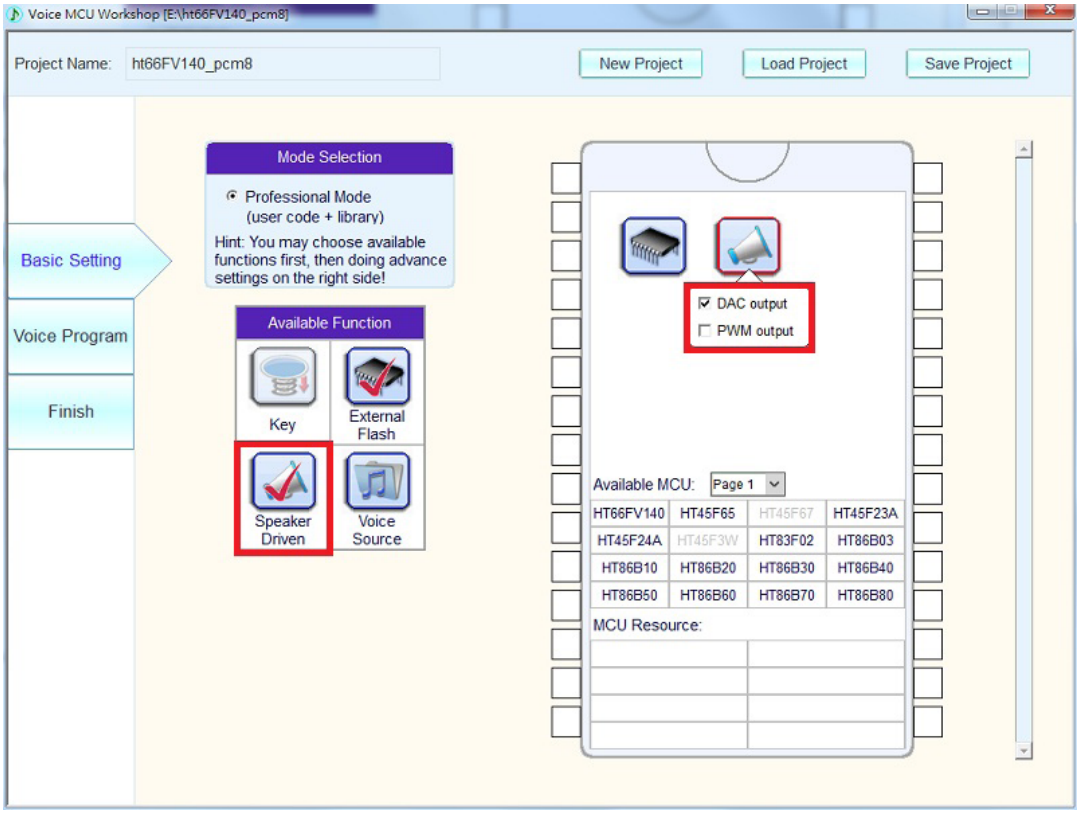
■ External Flash Function:

- Click the "External Flash" button to load/remove the function to/from the MCU.
- Click the "External Flash" icon in the MCU block on the right to select the Flash size, as shown below:



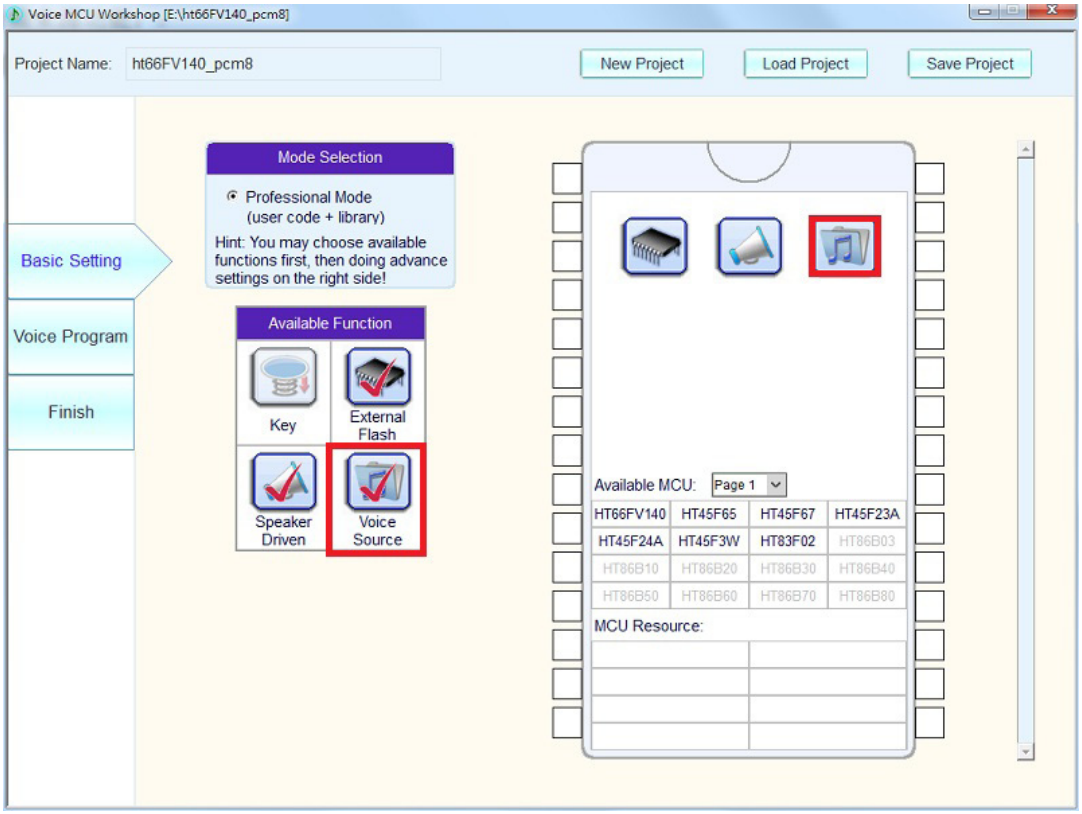
■ Speaker Driven Function:

- Click the "Speaker Driven" button to load/remove the function to/from the MCU.
- Click the "Speaker Driven" icon in the MCU block on the right to setup the driver mode. At the present time only the DAC output mode is supported as shown below:

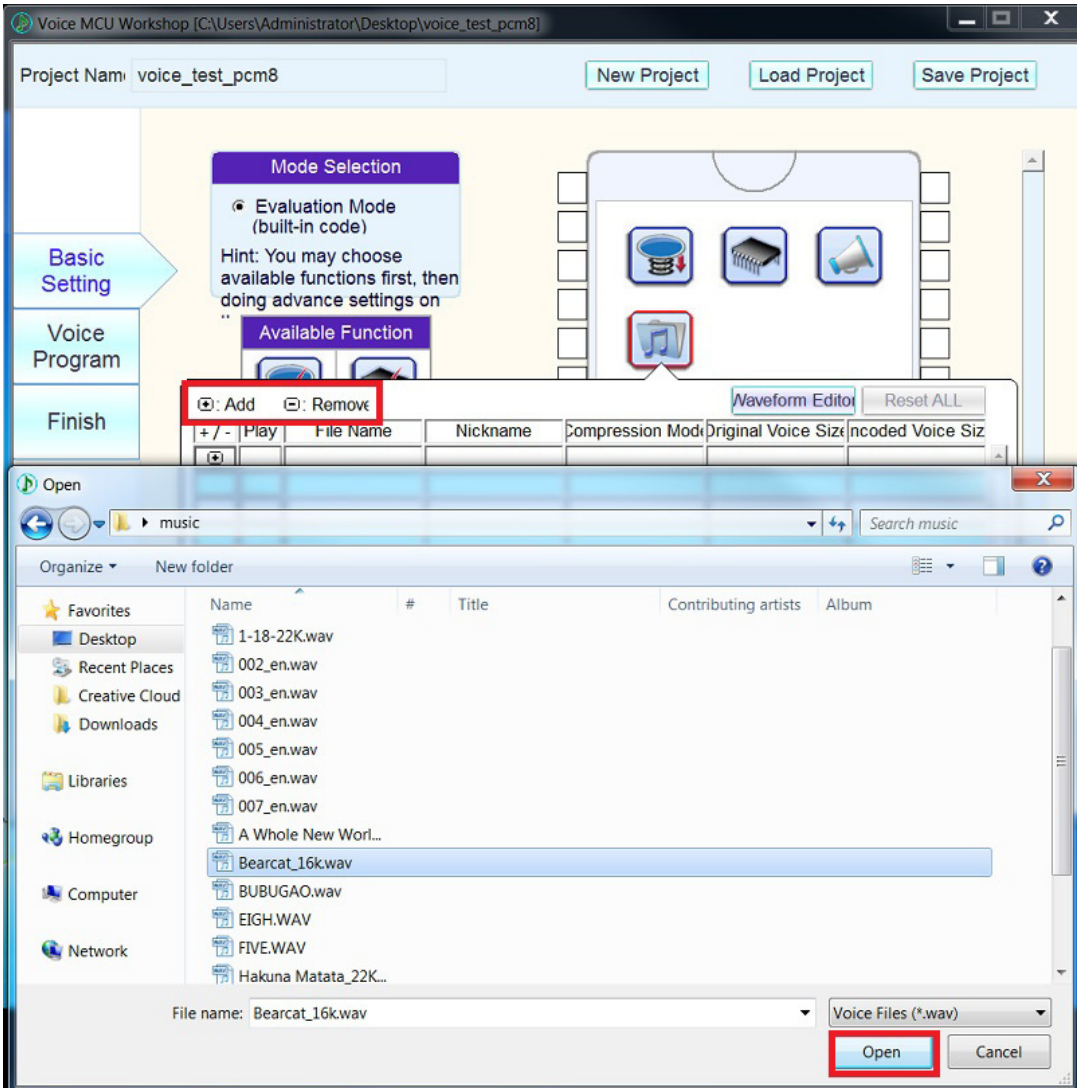


■ Voice Source Function

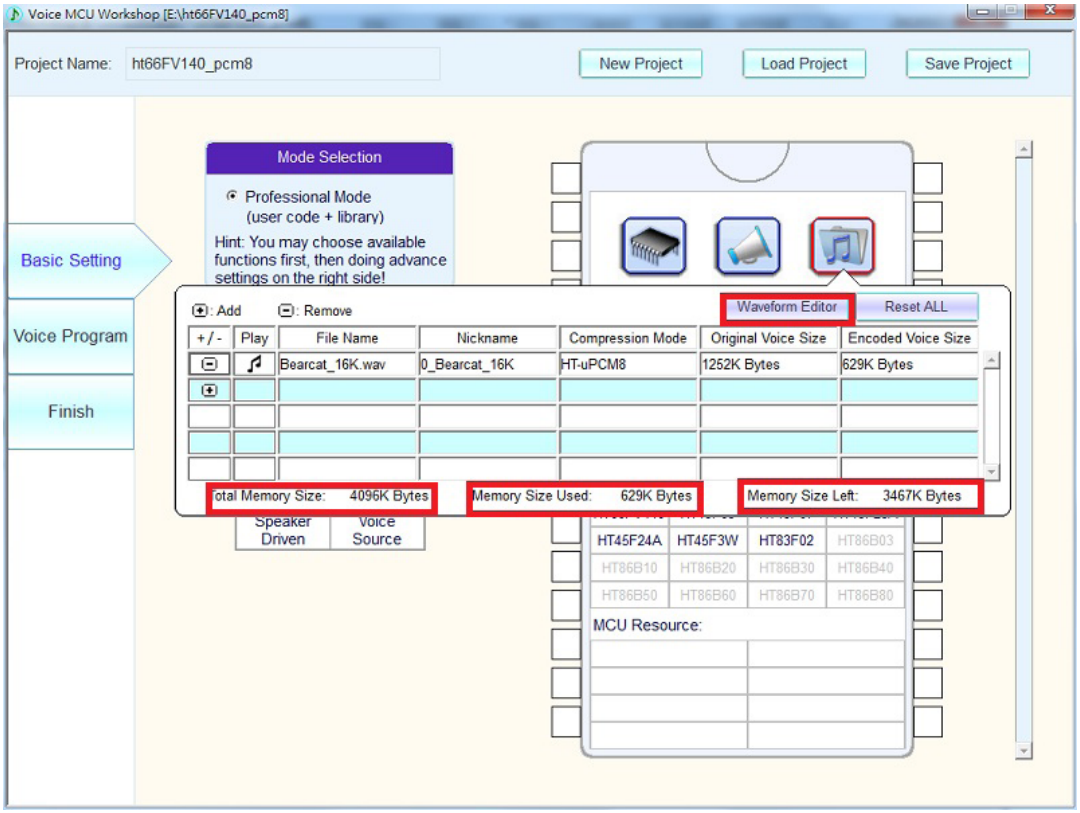
- ① Click the “Voice Source” button to load/remove the function to/from the MCU.



- ② Click the "Voice Source" icon in the MCU block on the right to add or remove ".wav" files, as shown below:

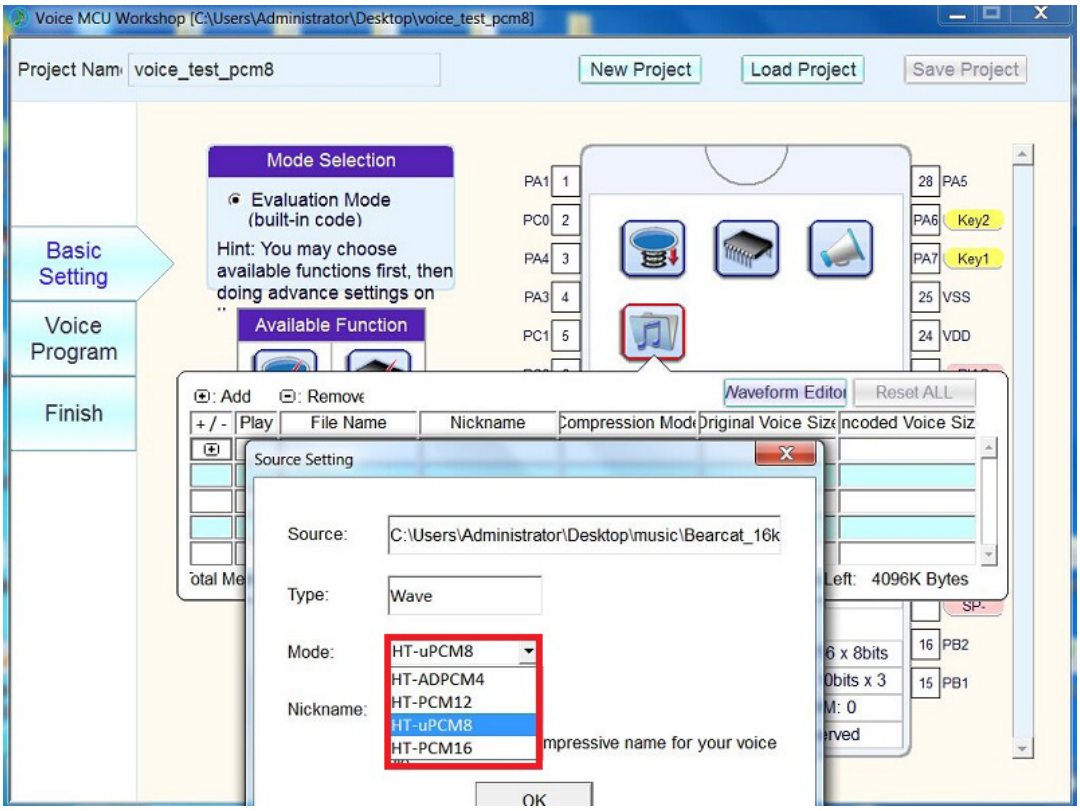


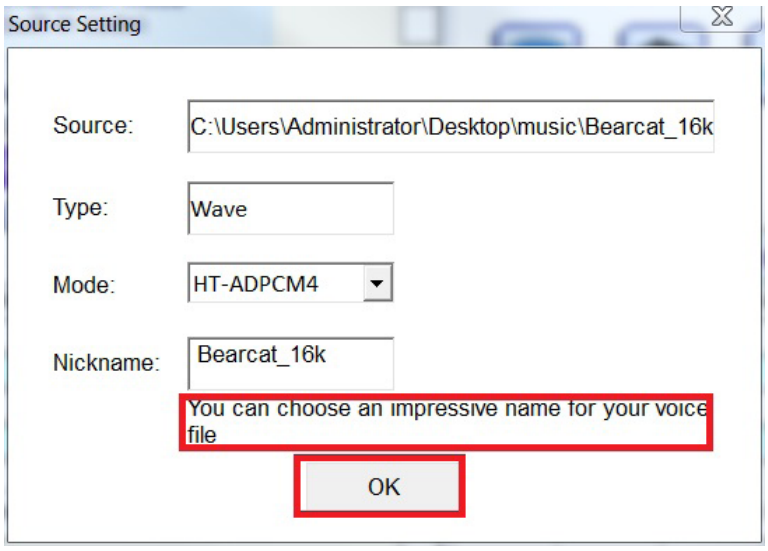
- ③ Before loading the voice source file, you can first click the "Waveform Editor" button to connect to the "Audacity" Audio Editor to process the voice source file after which it can be saved. Note: ensure that the Audacity software is installed, otherwise it must first be downloaded from the website http://audacity.sourceforge.net/. Then refer to Audacity Quick Start for the Audacity application details. After loading the file successfully, the "Total Memory Size", "Memory Size Used", "Memory Size Left" information is displayed, as shown below.



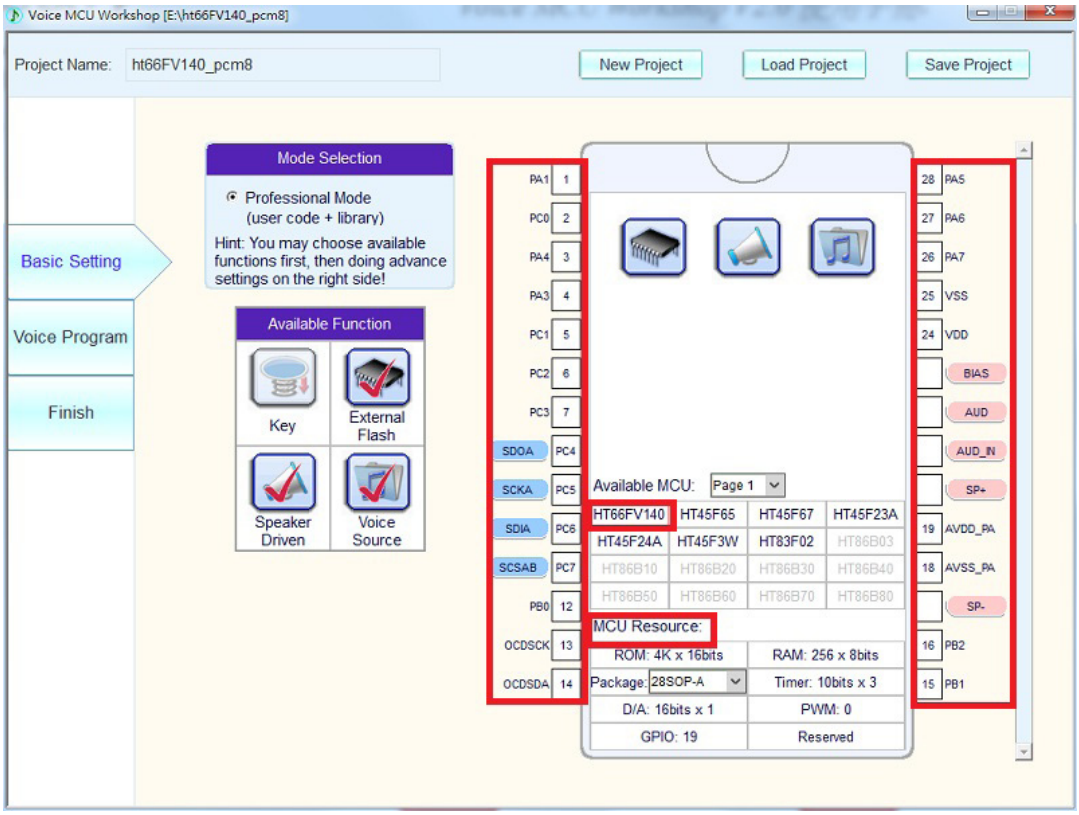
Note: For the maximum frequency limit for the added voice source, refer to the Library Establishment Information section.

- ④ In above ② , press **"Open file" and a source setting dialog box, including voice source in**formation, compression mode setup, etc., appears. After these have been setup, click "OK" to complete the voice source design. See the figure below.

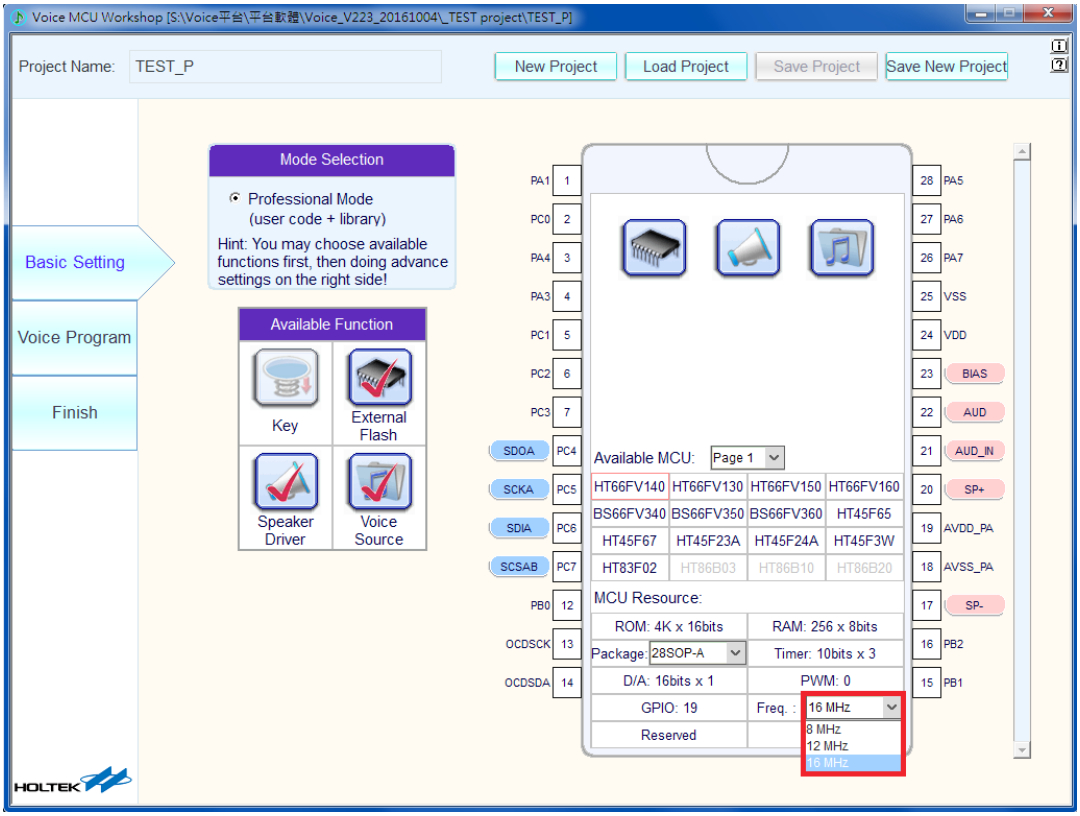




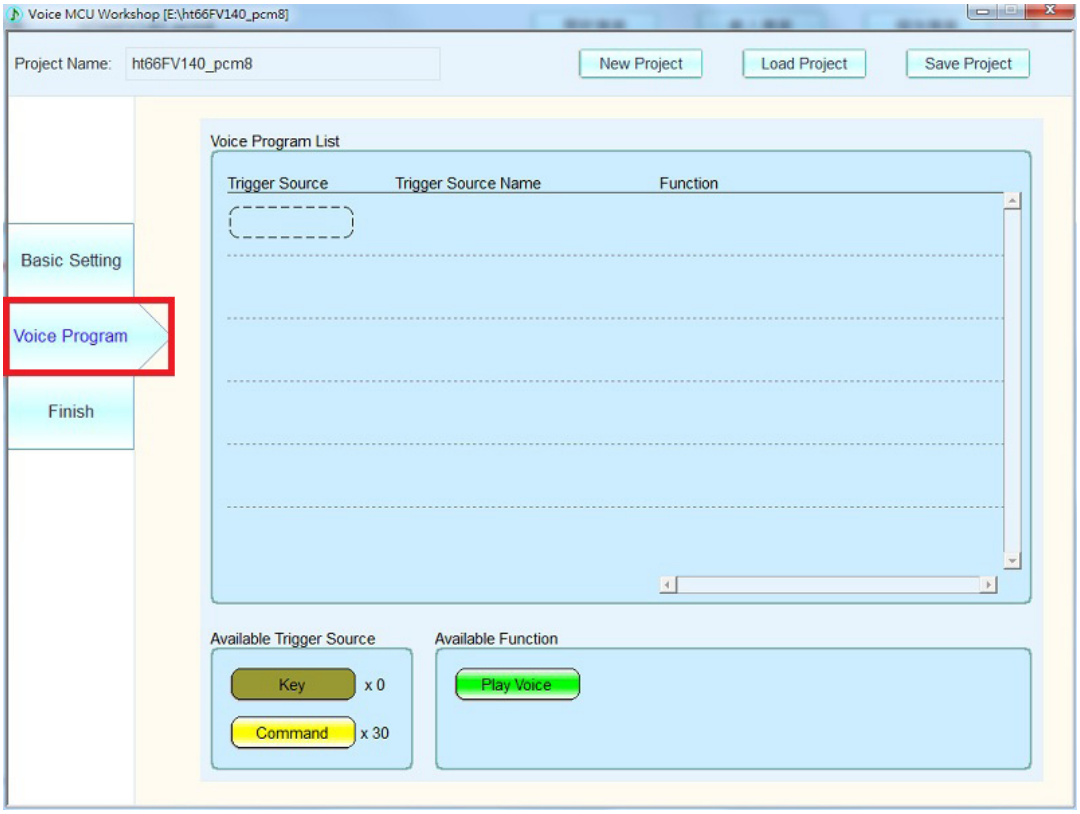
Step5. If an MCU has been selected, then the MCU related information including the MCU pins and internal resources that can be used by the available functions are displayed as shown in the following figure.



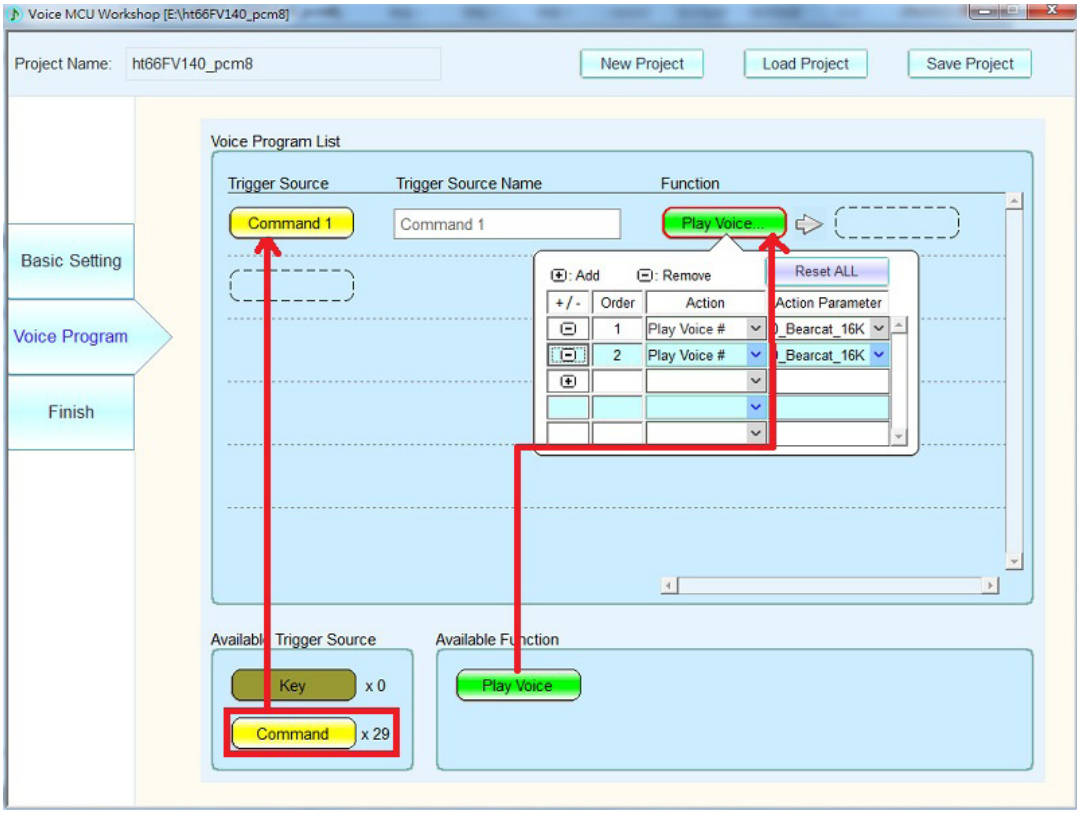
Note: Right-click the mouse button, select the MCU frequency, as shown below.



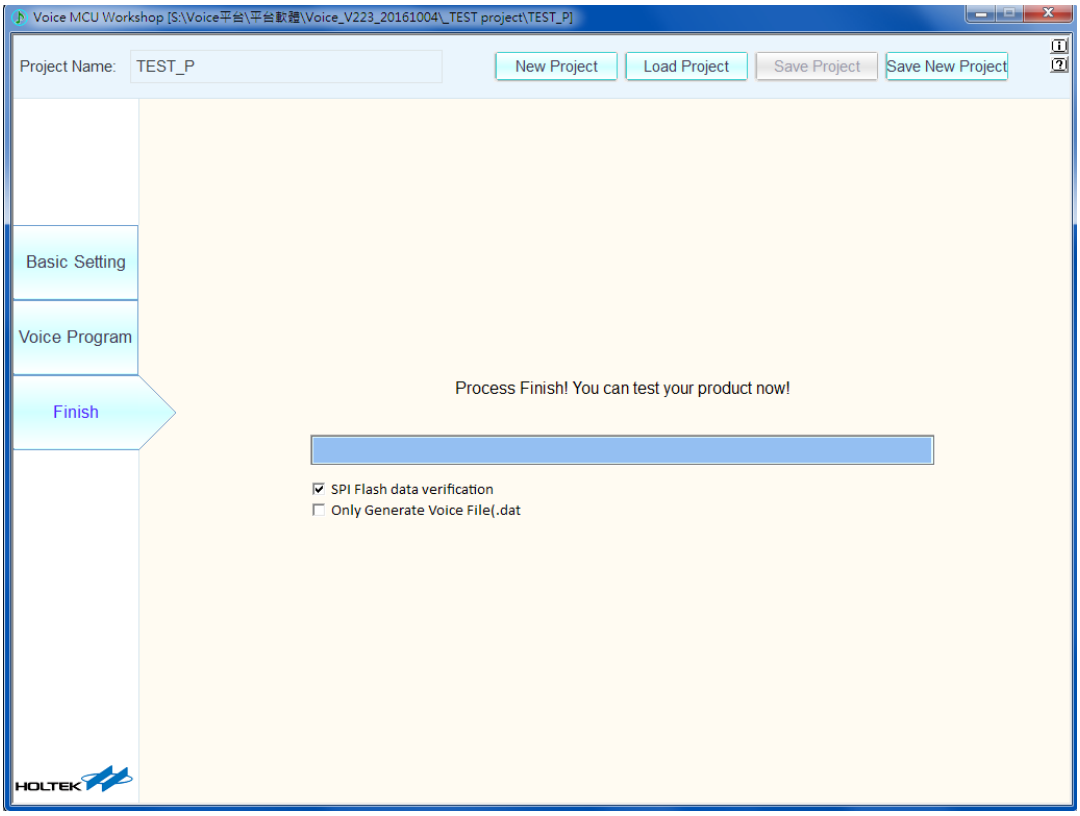
Step6. After completing the basic settings, switch to the voice program page as shown in the following figure.



Step7. According to the number of required trigger commands now arrange the program. Here Command means play sentence address, as shown below:

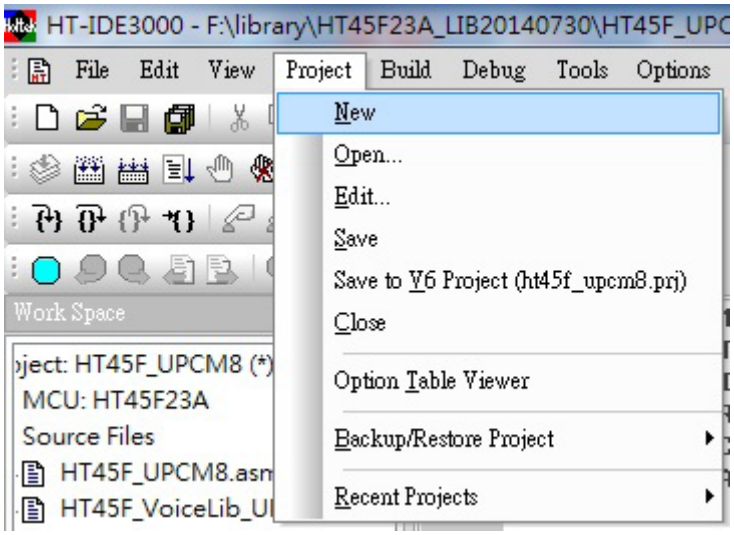


Step8. After finishing the voice program page setup, click "OK" and program the DAT file (audio compressed file) into the Flash memory. The stored data then can be called in the same way as the generated library under the professional mode and some related functions.

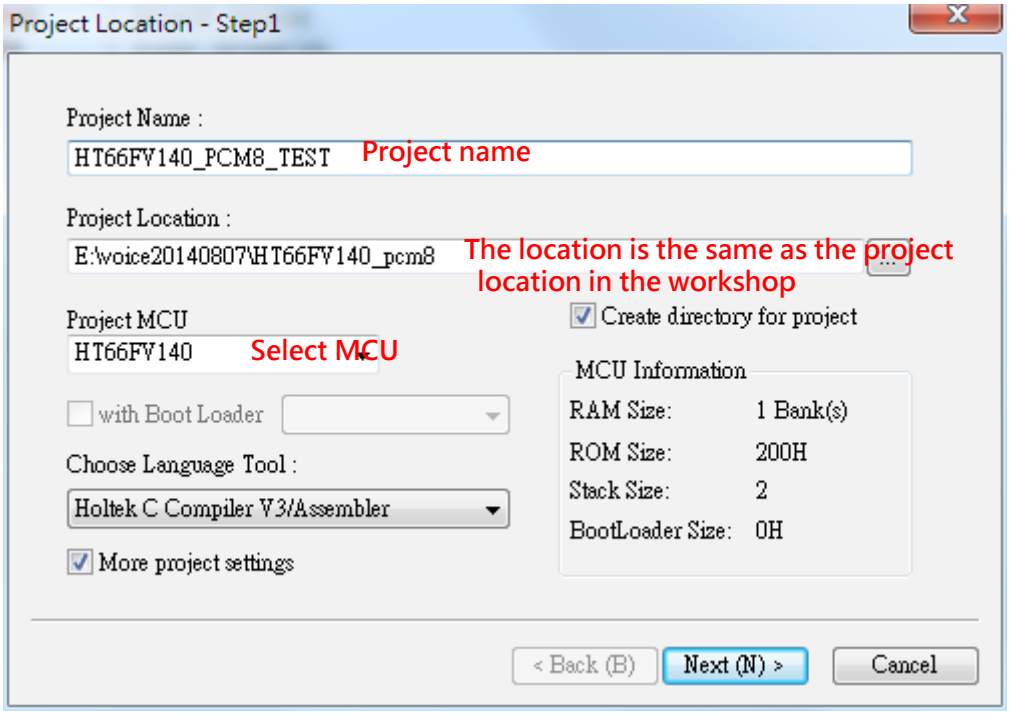


Step9. Create a new IDE-3000 project within the professional mode project directory that was just created for calling related libraries and files.

■ Choose IDE3000 "'Project'-> "New" to create a new project



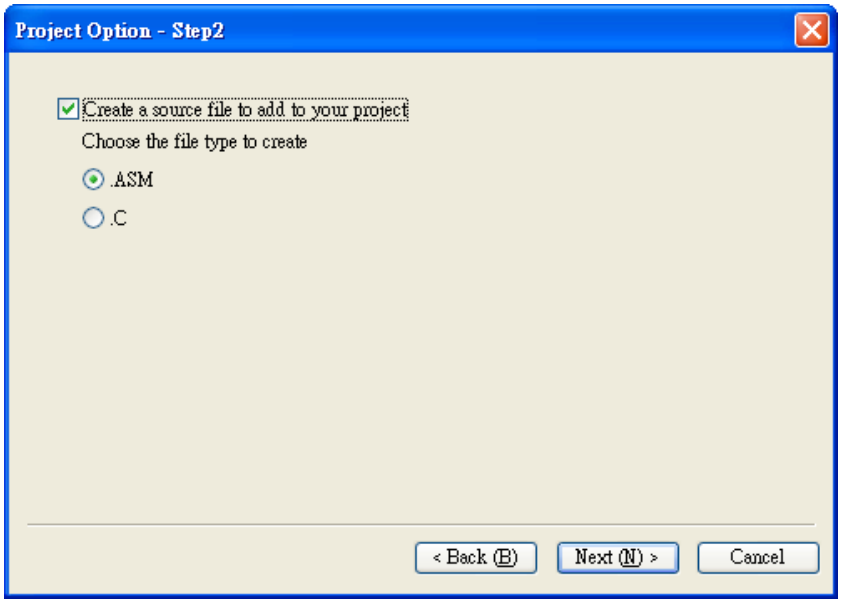
■ After clicking "New", a dialog box will appear after which the project information can be entered.



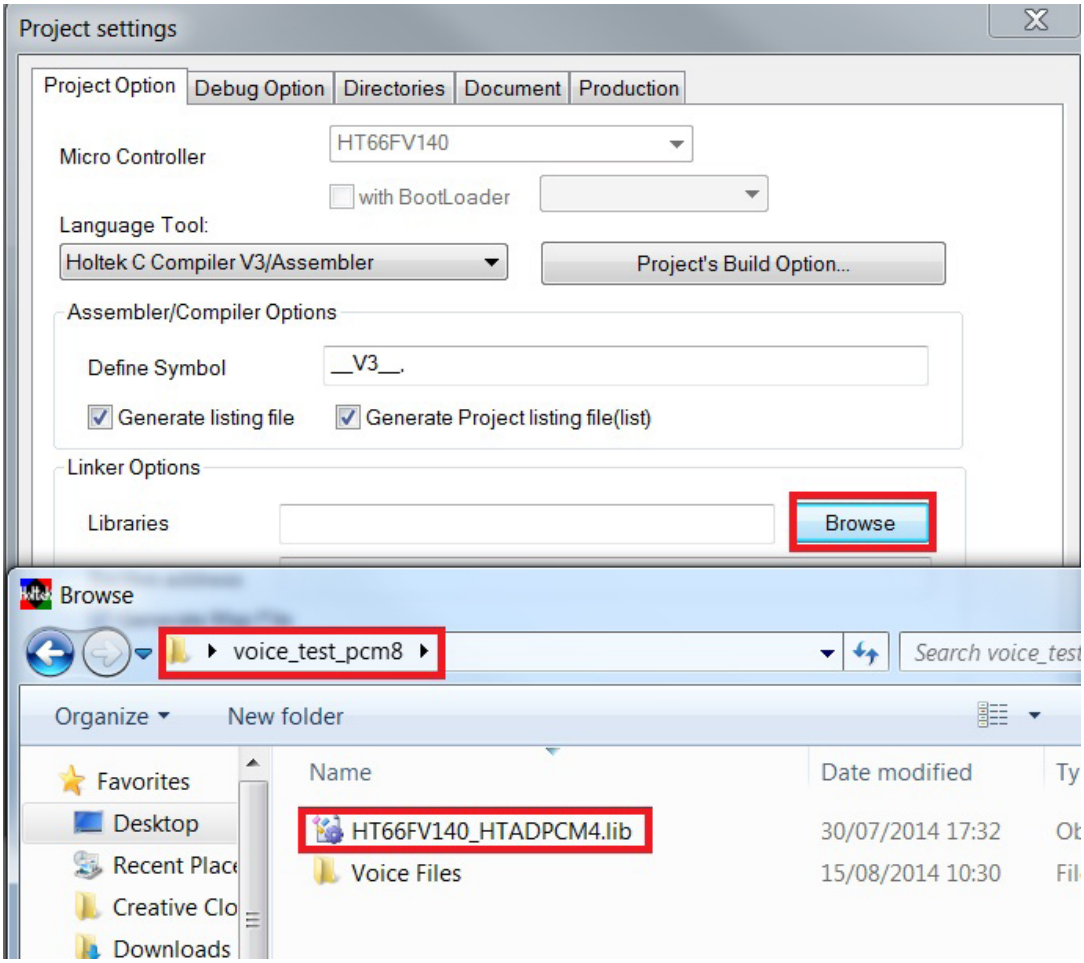
Note: due to Compiler requirements, the library file must be in the same directory as the project. Therefore the new IDE3000 project location must be the same as the platform project location for the called library. If the two projects are in different directories, it is necessary to copy the library file 

into the IDE3000 project directory.

■ Click “next” and then choose the development language.



■ Add the library file to the new IDE3000 project

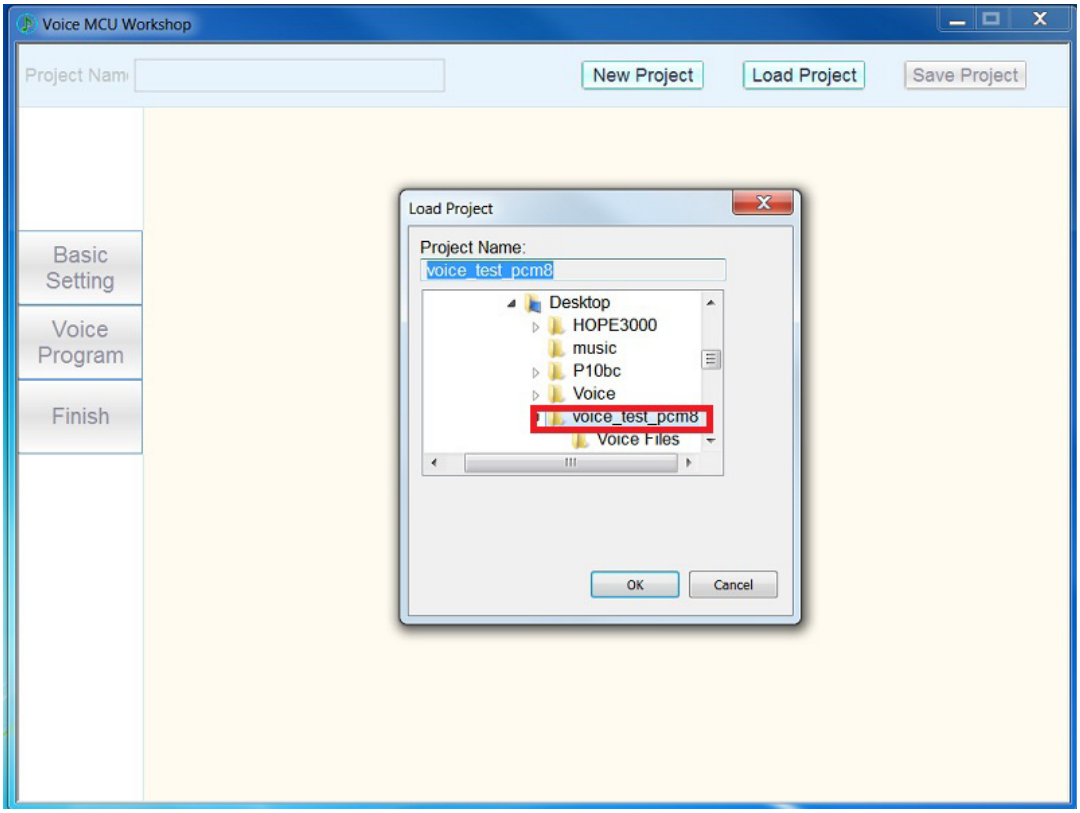


■ Refer to the ASM CALL or C CALL section (press Ctrl key and click the link to jump there)to learn how to call functions for building projects.

Step10. After creating the project, download the .MTP file generated by IDE-3000 to the voice MCU for debugging and playing.

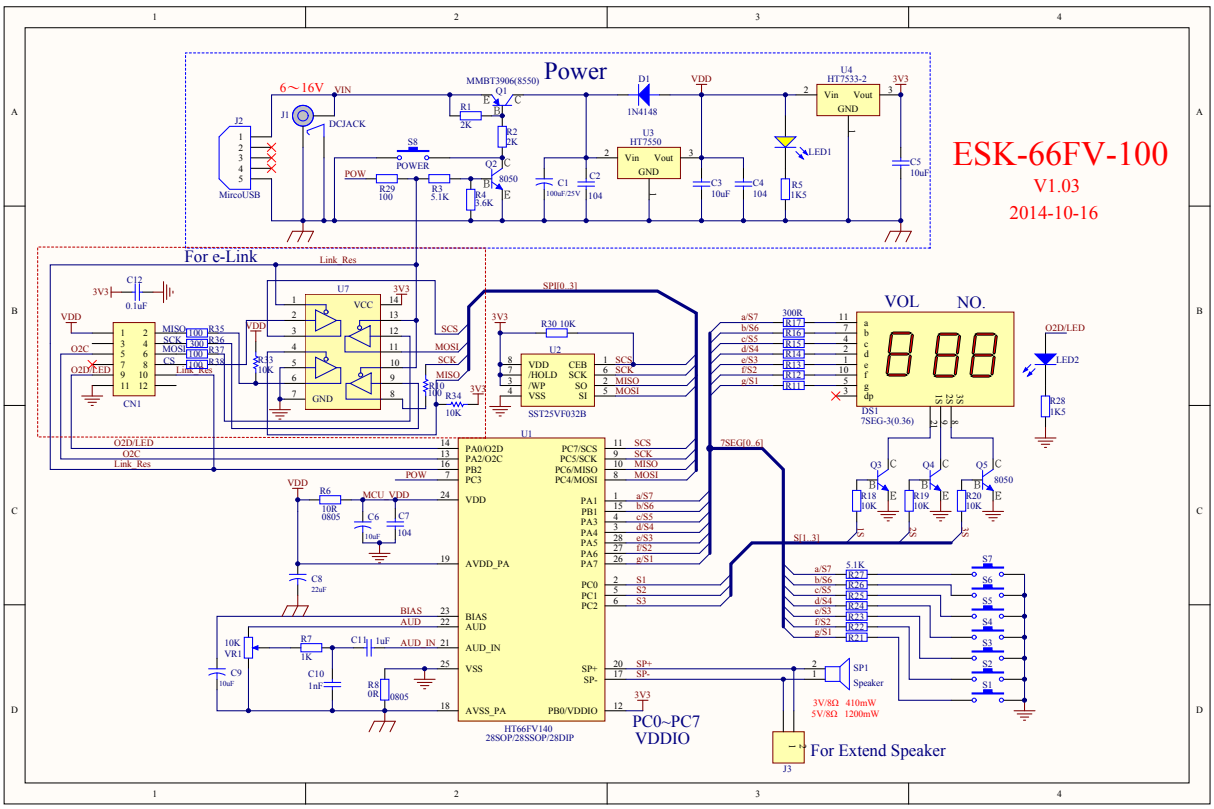
Open an existing project

Click on "Load Project" to open an existing project location. Then edit or download it just in the same way as creating a new project. The interface is shown in the following figure.



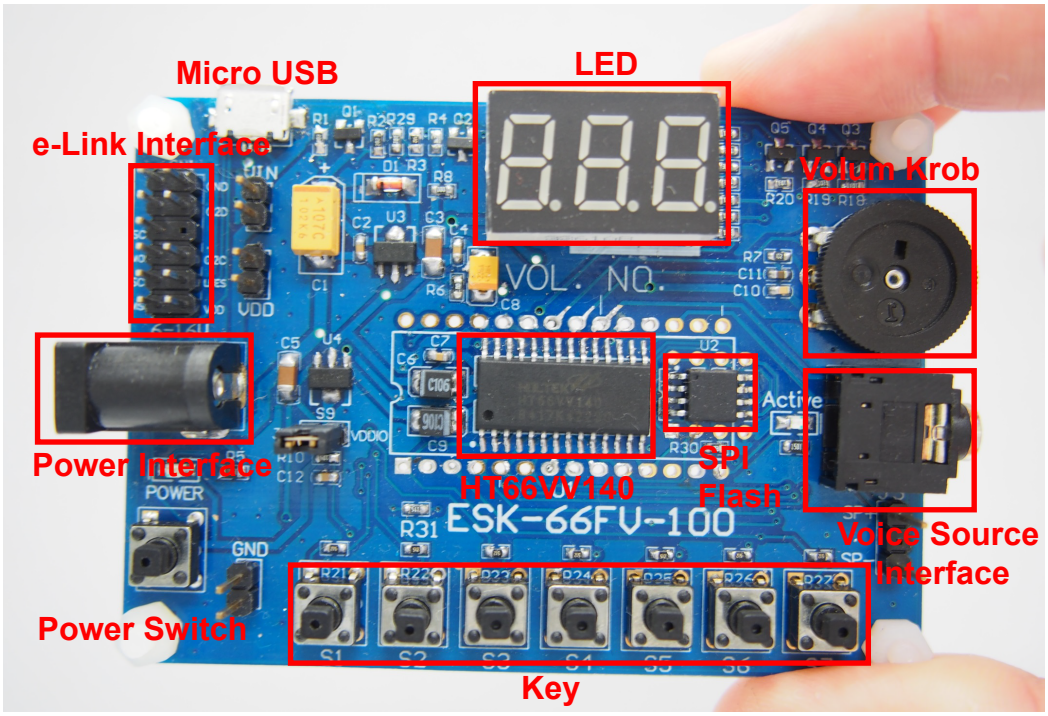
Hardware Circuit

Evaluation Board Schematic Diagram



Using the Evaluation Board

■ Evaluation board introduction

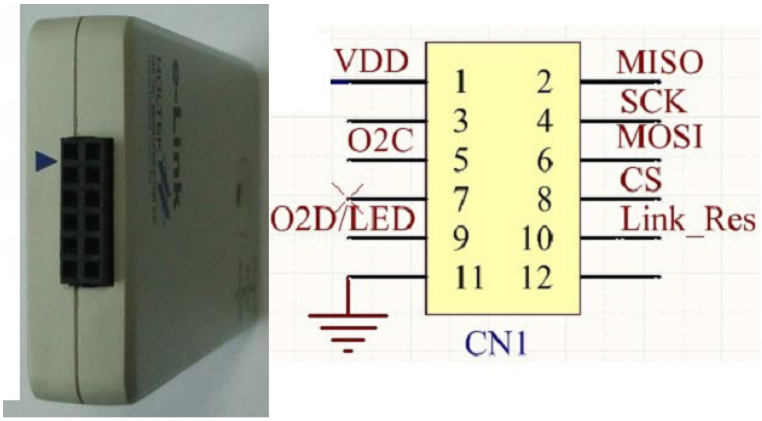


■ Hardware setting steps - Evaluation board has been programmed

- Connect the external speaker
- Connect to a 6V~16V power using the “power interface” or connect to a 5V power via the “micro USB port” and turn on the “power switch”. Another solution is to allow the e-Link to supply the power.
- Adjust the “audio control keys” to control audio playback and then turn the “volume knob” to change the volume.

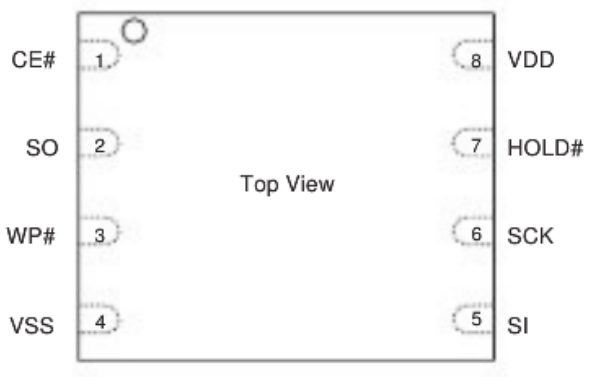
■ Flash Memory DAT File Programming Connections

- Flash connections



The figure shows the e-Link pin assignment and the actual device in which the triangle points to Pin 1. The pins in the two pictures directly correspond.

The following shows the flash pin assignment.



When programming the flash memory, the e-Link pins and flash pins should be connected as follows:

e-Link VDD–>flash VDD; e-Link GND–>flash VSS;

e-Link MISO–>flash SO; e-Link SCK–>flash CK;

e-Link MOSI–>flash SI; e-Link SCS–>flash CE#.

Supported Flash series

MXIC Series			
128M bits	MX25L12873F	32M bits	MX25L3206E
64M bits	MX25L6406E		MX25L3235E
	MX25L6435E		MX25L3208E
	MX25L6408E		MX25L3273E
	MX25L6473E	8M bits	MX25L8006E
16M bits	MX25L1606E		MX25L8035E
	MX25L1633E		MX25L8036E
	MX25L1608E	4M bits	MX25L4006E
	MX25L1635E		MX25L4026E
	MX25L1636E	2M bits	MX25L2006E
	MX25L1673E		MX25L2026E
1M bits	MX25L1006E	512K bits	MX25L512E
	MX25L1026E		
SST Series			
64M bits	SST26VF064B	8M bits	SST25VF080B
32M bits	SST25VF032B	4M bits	SST25VF040B
	SST26VF032B		
16M bits	SST25VF016B	2M bits	SST25PF020B
	SST26VF016B		SST25VF020B
Winbond Series			
128M bits	W25Q128BV	8M bits	W25Q80CV
	W25Q128FV		W25Q80DV
64M bits	W25Q64CV		W25Q80BL
	W25Q64FV	4M bits	W25Q40CL
32M bits	W25Q32FV		W25X40CL
	W25Q32BV	2M bits	W25Q20CL
16M bits	W25Q16CV		W25X20CL
	W25Q32BV	1M bits	W25X10CL
	W25Q16CL	512K bits	W25X05CL
GigaDevice Series			
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 and C library Instructions

Call Voice library Functions using ASM

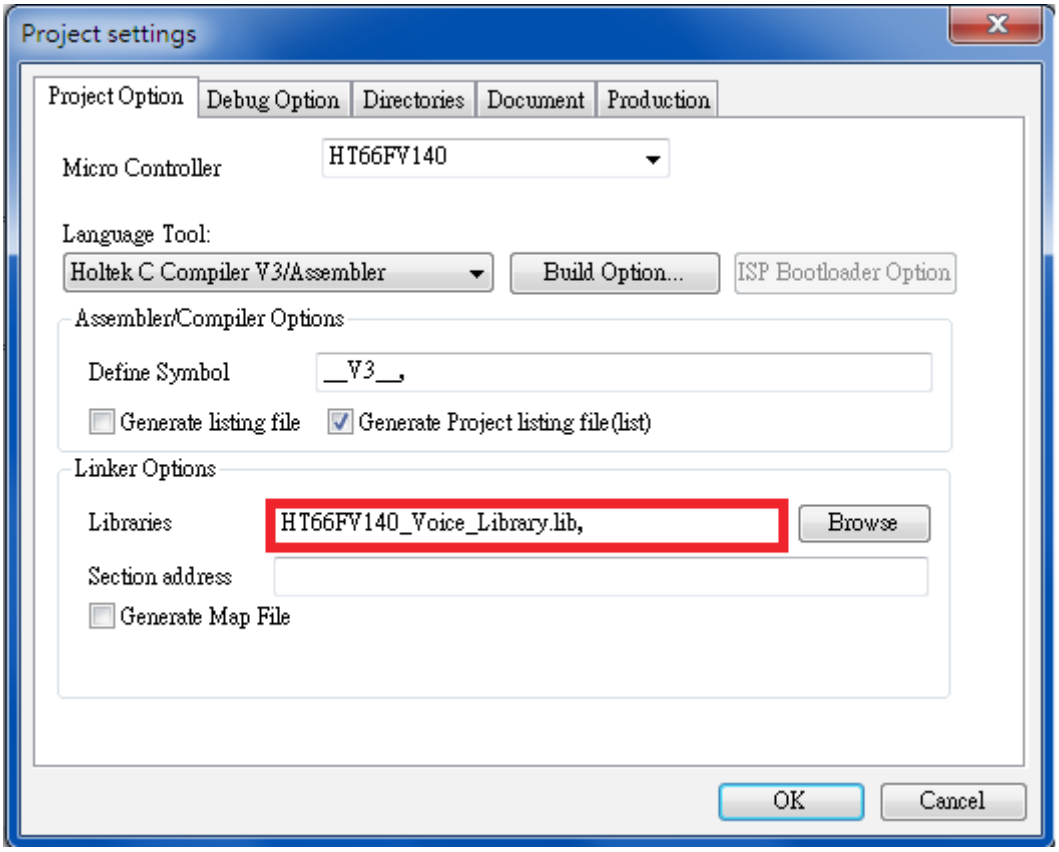
Summary

This chapter will introduce how to call the Voice library functions using ASM.

Usage Instructions

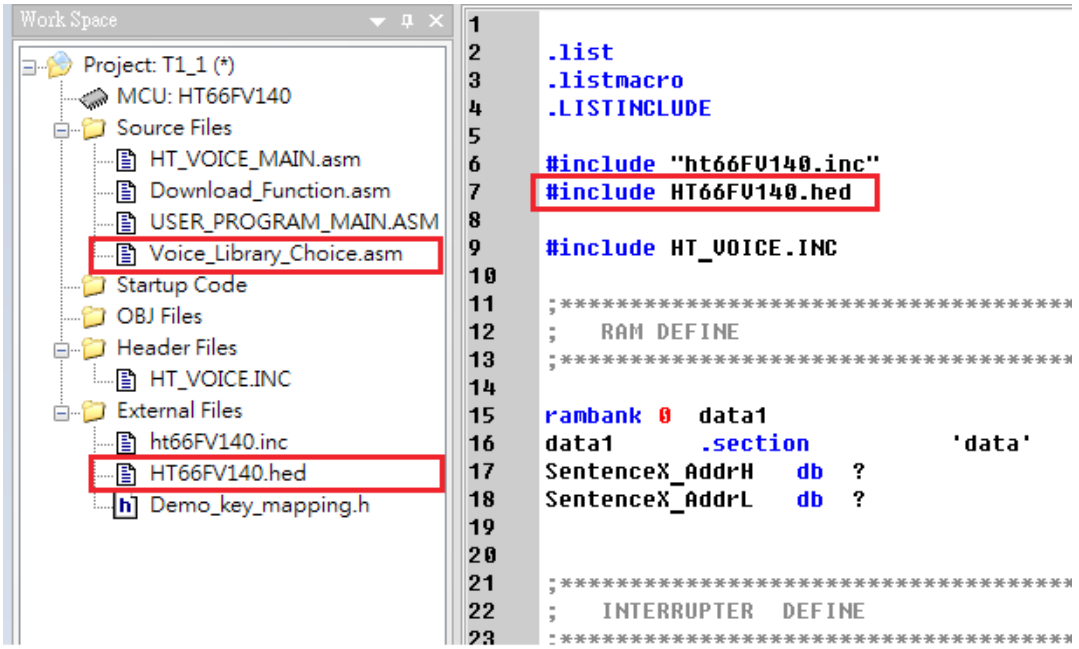
After creating the .ASM project:

■ Add the library file



■ Add the header file

Add the library header file, XX.hed & Voice_Library_Choice.asm, in order to call the library functions.



■ Refer to the Program Example for programming.(ASM Program Example)

ASM Library Functions

■ _CLRRAM

Description:

Clear all the ram banks.

Example:

```
_CLRRAM
```

■ _SYSTEM_INITIALIZATION

Description:

Setup the system frequency f_{SYS}, SPI interface configuration, timers initialization, etc.

Example:

```
_CLRRAM
_SYSTEM_INITIALIZATION
```

■ _DAC_RAMP_UP

Description:

Enable DA function. After the function is executed, then call the "_PLAY_VOICE , _PLAY_SENTENCE , _PLAY_SENTENCE_INDEX " functions.

Example:

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

■ _DAC_RAMP_DOWN

Description:

Disable the DA function. After the "_PLAY_VOICE , _PLAY_SENTENCE , _PLAY_SENTENCE_INDEX" functions is executed then call the function to reduce unnecessary power consumption.

Example:

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

■ _STOP_PLAY

Description:

Stop playing. Call this function directly at any time.

Example:

```
_STOP_PLAY
```

■ _VOLUME Volume

Description:

Set the volume level. Write the volume value with reference to the specification.

Parameter:

Volume: The specification volume value.

```
Example: _VOLUME   0   ; Set the volume to minimum.
         _VOLUME   7   ; Set the volume to maximum. Note that for different volume values,
                       ; there are different settings scopes, so refer to the specification for
                       ; the volume value.
```

Note: volume of 0 ~ 12 HT66FV1X0 series

■ _PLAY_VOICE VoiceNumHigh, VoiceNumLow, Channel, Volume, Reserve

Description:

Play the voice file and the DAT generated by the WAV voice file saved to the flash with the Voice Workshop in advance.

Parameter:

VoiceNumHigh:	Voice NUM high byte
VoiceNumLow:	Voice NUM low byte
Channel :	Voice channel selection(now only support channel 0)
Volume :	Voice volume selection(0-7)
Reserve :	0

Example:

Play the first audio source original file (Note: on the UI, the first audio source number is 0 instead of 1)

Select volume 7

Then: _DAC_RAMP_UP

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

■ _PLAY_SENTENCE SentenceNumHigh, SentenceNumLow, Channel, Volume, Reserve

Description:

play_sentence

Parameter:

SentenceNumHigh:	SentenceAddr high byte
SentenceNumLow:	SentenceAddr low byte
Channel :	Voice channel selection(now only support channel 0)
Volume :	Sentence voice volume
Reserve :	0

Example:

Play the first sentence file, assume the address is 0100H and set the volume as 7

then: _DAC_RAMP_UP

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

Note: entence addresses can be seen in the Demo_key_mapping.h file within the Workshop project directory, as shown below (the first sentence address is 0100H)

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

■ _PLAY_SENTENCE_INDEX Reservel, Sentence index, Channel, Volume, Reserve

Description:

Which the first Sentence

Parameter:

Reservel: no use
What index, the index number (1-255)
Channel: choose the sound audio broadcast Channel (currently only support Channel 0)
Volume: what the Volume option
Reserve: 0

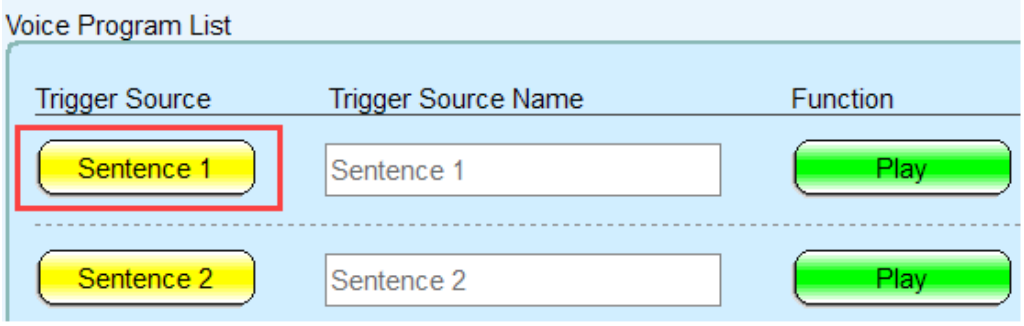
Example:

play the first sentence 1 file, with the volume of 7

then: _DAC_RAMP_UP

_PLAY_SENTENCE_INDEX 0, 1, 0, 7, 0

Note: what index in voice schedule list, as shown in the figure below:



■ _MODIFY_SAMPLINGRATE mSamplingRate

Description:

Change the current broadcast voice sampling rate

Parameter:

MSamplingRate: specify the sampling rate value (Hz)

Example:

Change the current broadcast voice sampling rate of 11025 hz

then: _MODIFY_SAMPLINGRATE 11025

_PLAY_VOICE 0, 0, 0, 7, 0

■ _PLAY_VOICE_ISR

Description:

According to the initialization time, when the timer interrupt arrived, enter into the interrupt function to play voice.

Example:

```
ORG XXH          ; XXH: play voice timer interrupt entry
_PLAY_VOICE_ISR
```

■ _PLAY_SENTENCE_ISR

Description:

According to the initialization time, when the timer interrupt is generated, enter the interrupt subroutine to play a sentence.

Example:

```
ORG XXH          ; XXH: play sentence timer interrupt entry
_PLAY_SENTENCE_ISR
```

Note: The program is used to determine whether a voice or sentence is playing or has been played

```
MOV  A,00H
SZ fSentencePlaying  ;fSentencePlaying  =1 means Sentence is playing, 0 means played
RET
SZ      fVoiceStandBy ;fVoiceStandBy  =0 means Voice is playing, 1 means played
MOV  A,01H           ;if Play voice or sentence has finished, then A=1, or A=0. Through the
                     ;A value, to determine if voice or sentence has played.
```

■ _ENABLE_VDDIO

Description:

Call this function to enable the MCU VDDIO function and the voltage on the SPI pins will be sourced from the VDDIO. After the function is executed then call the "_CLRRAM" function.

Example:

```
_ENABLE_VDDIO
_CLRRAM
_SYSTEM_INITIALIZATION
```

■ _PAUSE

Description:

Call this function to pause playing when a voice or sentence is playing.

Example:

```
_PLAY_VOICE  0, 0, 0, 3, 0        ; Play the first voice, the volume level is 3
_CALL_DELAY                       ; Delay function, pause after the voice is played for a while
_PAUSE                            ; Call the "_PAUSE" function
```

Note: The delay function is only an example, which is not provided in the voice library. The specific condition of the voice play pause is determined by the user.

■ _RESUME

Description:

After the "_PAUSE" function is executed then call the "_RESUME" function to resume play.

Example:

```
_PLAY_VOICE  0, 0, 0, 3, 0        ; Play the first voice, the volume level is 3
_CALL_DELAY                       ; Delay function, pause after the voice is played for a while
_PAUSE                            ; Call the "_PAUSE" function
_CALL_DELAY
_RESUME                           ; Resume play
```

Note: The delay function is only an example, which is not provided in the voice library. The specific condition of the voice play resume is determined by the user.

ASM Program Example

Using a voice library, must add the following files in the project:

1. Voice_Library_Choice.asm
2. MCUNAME_Voice_Library.lib

■ [Application example – HT66FV130]

```
#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
```

■ [Application example – HT66FV140]

```
#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
```

■ [Application example – HT66FV150]

```
#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
```

■ [Application example – HT66FV160]

```
#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
```

■ [Application example – BH67F2262]

```
#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 inedx 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
```

■ [Application example – HT45F67]

```
#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
```

■ [Application example – HT45F65]

```
#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
```

■ [Application example – HT45F3W]

```
#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
```

■ [Application example – HT66F4550]

```
#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          ;Timer0 interrupt( sentence )

    ORG    10H
    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 0,0,0,0,0              ;Play the first audio
    SNZ    fVoiceStandBy
    JMP    $-1                         ;Wait play voice finish

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

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

    CALL   _DAC_RAMP_DOWN              ;Close DAC and do ramp down

    CLR    WDT
    CLR    WDT2
    JMP    $-2
```

■ [Application example – HT45F23A]

```
#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
```

■ [Application example – HT45F24A]

```
#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
```

■ [Application example – HT83F02]

```
#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
```

■ [Application example – HT86B03] (Suitable for 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
```

■ [Application example – HT86B40] (Suitable for 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
```

Call Voice library Functions by C

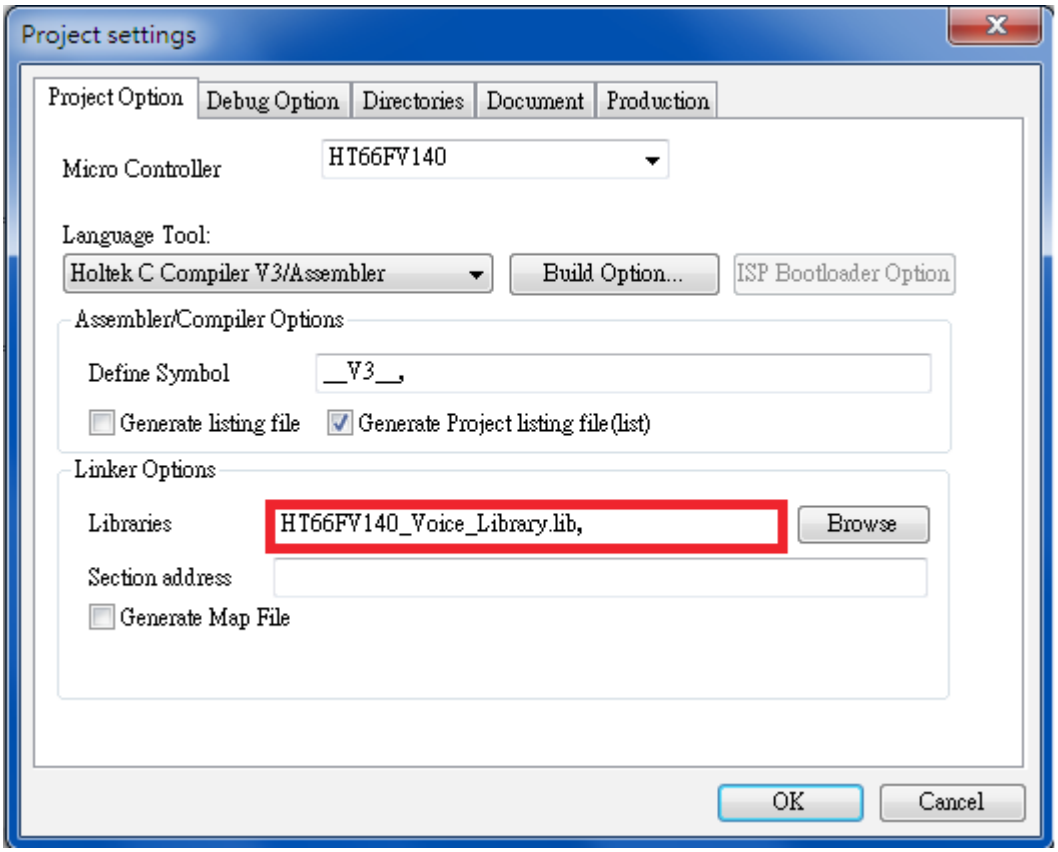
Summary

This chapter will introduce how to call the Voice library functions using C language.

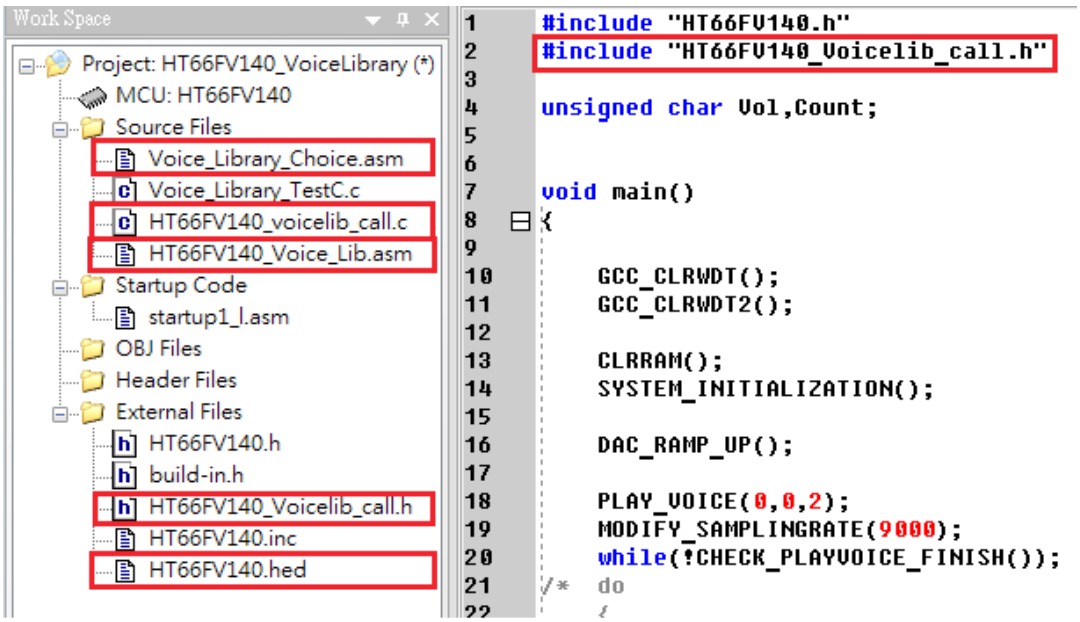
How to use

After creating the .C project:

■ Add the library file



■ The related files needed to add the library function is as follows:
Voice_Library_Choice. Asm, XX. Hed XX_Voicelib_call. C, XX_Voicelib_call. J h, XX_Voice_Lib. Asm



■ Add “XX_voicelib_call.h” file to the project directory, and place it within the C file “ #include "XX_voicelib_call.h" “which will be called. It is declaration of all functions which will called.

■ Refer to the C Program Example for programming (C Program Example).

C Library Functions

■ void CLRRAM( );

Description:

ram bank0, bank1, 00h~FFh are cleared to zero.

Example:

```
CLRRAM( );
```

■ void SYSTEM_INITIALIZATION( );

Description:

Setup the system frequency f_{SYS}, SPI interface configuration, timer initialization, etc.

Example:

```
CLRRAM( );
SYSTEM_INITIALIZATION( );
```

■ void DAC_RAMP_UP( );

Description:

Enable DA function. After the function is executed then call the " PLAY_VOICE() , _PLAY_SENTENCE() , PLAY_SENTENCE_INDEX() " function.

Example:

```
DAC_RAMP_UP( );
PLAY_VOICE( );
```

■ void DAC_RAMP_DOWN( );

Description:

Disable DA function. After the “PLAY_VOICE() , _PLAY_SENTENCE() , PLAY_SENTENCE_INDEX()” functions is executed then call the function to reduce unnecessary power consumption.

Example:

```
PLAY_VOICE( );
DAC_RAMP_DOWN( );
```

■ void STOP_PLAY( );

Description:

Stop playing. Call this function directly at any time.

Example:

```
STOP_PLAY( );
```

■ Void VOLUME_CHOICE(unsigned char vol);

Description:

Set the volume level. Write the volume value with reference to the specification.

Parameter:

Vol: The volume value in the specification

Example:

VOLUME_CHOICE(0x67);

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

Description:

play_voice

Parameter:

Voicenumh: Voice NUM high byte

Voicenuml : Voice NUM low byte

vol_voice : Voice volume selection

Example:

Play the first audio source original file (Note: on the UI, the first audio source number is 0 instead of 1)

Select volume Gain=6DB(0x0C in the specification)

Then: DAC_RAMP_UP( );

PLAY_VOICE(0,0,0xc);

Note: The parameter Voicenumh, Voicenuml, vol_voice for variable form, PLAY_VOICE (A, B, C);

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

Description:

play_sentence

Parameter:

SentenceAddrH: SentenceAddr high byte

SentenceAddrL : SentenceAddr low byte

vol_sentence : Sentence volume selection

Note: SentenceAddr: Selected "play_voice" Function address on the UI of the Voice WorkshopV2.3 version platform.

Refer to the Workshop S/W generated file "Demo_key_mapping.h"

Example:

Play the first sentence file, assume the address is 0100H

Select volume Gain =6DB (0x0C in the specification)

Then: DAC_RAMP_UP( );

PLAY_ SENTENCE (0x01,0x00,0x0c);

Note: The parameter SentenceAddrH, SentenceAddrL, vol_sentence variables to form such as: PLAY_SENTENCE (A, B, C);

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

Description:

play which in the first Sentence

Parameters:

Reservel: no use

SentenceIndex: what number (1-255)

Vol_Sentence: what the volume option

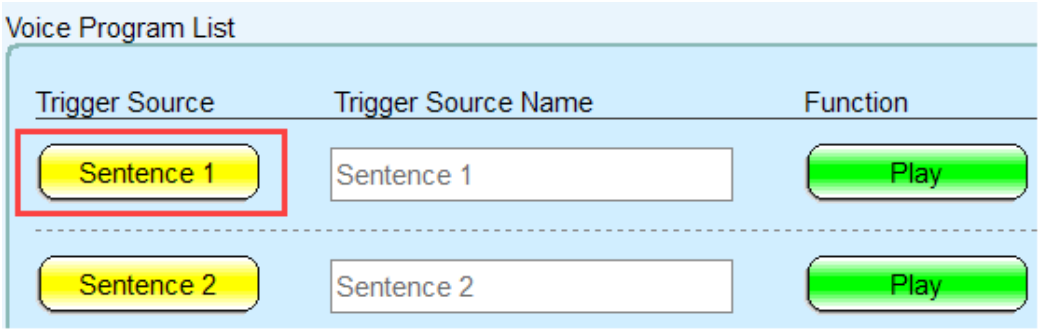
Example:

play the first sentence 1 file, with the volume of 7

Then: DAC_RAMP_UP( );

PLAY_ SENTENCE_INDEX (0, 1, 7);

Note: what Index in voice schedule list, as shown in the figure below:



■ unsigned char CHECK_PLAYVOICE_FINISH( );

Description:

Determine if the "play_voice" or "play_sentence" has finished or not

Return value:

1:play finished

0:play unfinished

Example:

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

■ void MODIFY_SAMPLINGRATE (unsigned int mSamplingRate)

Description:

Change the current broadcast voice sampling rate

Parameters:

MSamplingRate: specify the sampling rate value (Hz)

Example:

Change the current broadcast voice sampling rate of 11025Hz

Then: MODIFY_SAMPLINGRATE (11025);

PLAY_VOICE (0,0,7);

Note: this function is used, the USE_MODIFY_SAMPLINGRATE must be set to 1, the parameters in xxx_voicelib_call. H

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

■ void ENABLE_VDDIO ( );

Description:

Call this function to enable the MCU VDDIO function and the voltage on the SPI pins will be sourced from the VDDIO. After the function is executed then call the "CLRRAM ( )" function.

Example:

```
ENABLE_VDDIO ( );
CLRRAM ( );
SYSTEM_INITIALIZATION ( );
```

■ void PAUSE ( );

Description:

Call this function to pause playing when a voice or sentence is playing.

Example:

```
PLAY_VOICE (0, 0, 3);          //Play the first voice, the volume level is 3
DELAY ( );                     //Delay function, pause after the voice is played for a while
PAUSE ( );                     //Call the "PAUSE ( )" function
```

Note: The delay function is only an example, which is not provided in the voice library. The specific condition of the voice play pause is determined by the user.

■ void RESUME ( );

Description:

After the "PAUSE ( )" function is executed than call the "RESUME ( )" function to resume play.

Example:

```
PLAY_VOICE(0, 0, 3)            //Play the first voice, the volume level is 3
DELAY ( )                      //Delay function, pause after the voice is played for a while
PAUSE ( )                      //Call the "PAUSE ( )" function
DELAY ( )
RESUME ( )                     // Resume play
```

Note: The delay function is only an example, which is not provided in the voice library. The specific condition of the voice play resume is determined by the user.

C Program Example

Using a voice library, must add the following files in the project:

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

■ [Application example – HT66FV130]

```
#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();
    }

}
```

■ [Application example – HT66FV140]

```
#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();
    }

}
```

■ [Application example – HT66FV150]

```
#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();
    }
```

■ [Application example – HT66FV160]

```
#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();
    }
}
```

■ [Application example – BH67F2262]

```
#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();
    }
}
```

■ [Application example – HT45F67]

```
#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();
    }
}
```

■ [Application example – HT45F65]

```
#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();
    }
}
```

■ [Application example – HT45F3W]

```
#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();
    }
}
```

■ [Application example – HT66F4550]

```
#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(0,0,0);                 //Play the first audio
    while(!CHECK_PLAYVOICE_FINISH());  //Wait play voice finish

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

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

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

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

■ [Application example – HT45F23A]

```
#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();
    }
}
```

■ [Application example – HT45F24A]

```
#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();
    }

}
```

■ [Application example – HT83F02]

```
#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();
    }

}
```

■ [Application example – HT86B03] (Suitable for 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();
    }

}
```

■ [Application example – HT86B40] (Suitable for 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 Establishment and Emulator

HT66FV130

■ Resource Usage Table:

Resources Used \ Compression Mode	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%)				
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H	No decoding array
Other programs fixed memory address in the PROM	57EH–582H 583H–58CH	579H–57DH		6FEH–6FFH 702H–703H	
Stack (layers)	2				
Registers used	SPI1: SPIC0, SPIC1, SPID D/A: USVC, DAH, DAL Timer: TM1C0, TM1C1, TM1AL, TM1AH, TM0C0, TM0C1, TM0AL, TM0AH General: ACC, MP1, IAR1, TBLP, TBLH, TBHP, PCL, STATUS I/O: PCS1, PCPU, PBS0				

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

■ MCU function module usage description:

- SPI1is used for controlling the external flash – used pin: SCS, SCK, MISO, MOSI
- Timer1interrupt is used for play voice operation – interrupt entry address: 0CH
- Timer0 interrupt is used for the play sentence operation – interrupt entry address: 08H
- DAC module is used for the flash audio data D/A converter – used pin: AUD, AUDIN
- Power amplifier module – used pin:SP+, SP-

■ Different function calls require different PROM sizes, as shown in the table below:

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

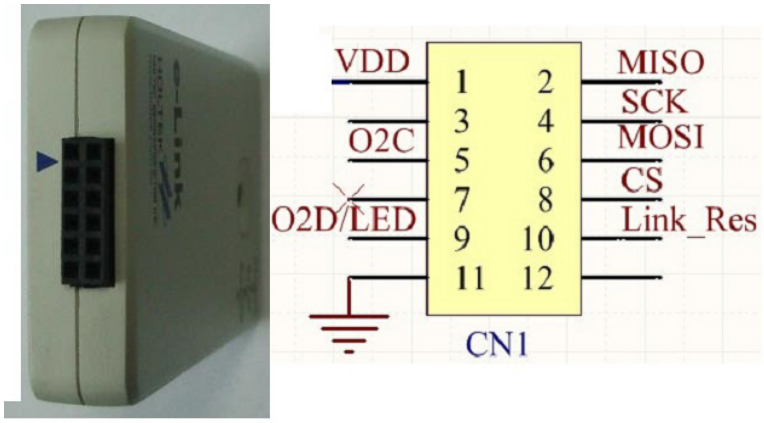
■ Using a specified MCU system frequency, the maximum audio source sampling rate using different compressed formats is shown in the following table:

Compression Mode \ System Frequency	8MHz	12MHz	16MHz
HT-ADPCM4	13kHz	20kHz	27kHz
HT-PCM12	12kHz	18kHz	24kHz
HT-uPCM8	11kHz	17kHz	22kHz
HT-PCM16	13kHz	19kHz	26kHz

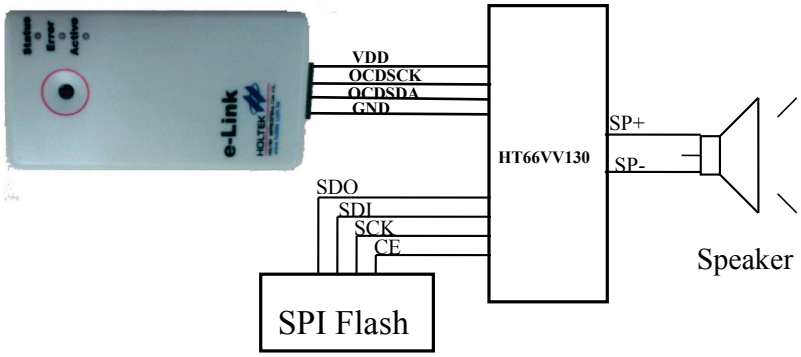
■ Emulator and Connection

This MCU uses the e-Link simulator and the EV chip HT66VV130 for simulating and debugging. In addition an external SPI Flash is needed.

- e-Link Pin Assignment:



- HT66VV130 VDD, GND, OCDSCK, OCDSDA pin connection to the e-Link.



Note: Refer to "Connection for Programming DAT File to the Flash" section for SPI flash connection and programming.

HT66FV140

■ Resource Usage Table:

Resources Used \ Compression Mode	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%)				
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H	No decoding array
Other programs fixed memory address in the PROM	57EH–582H 583H–58CH	579H–57DH		6FEH–6FFH 702H–703H	
Stack (layers)	2				
Registers used	SPI1: SPIC0, SPIC1, SPID D/A: USVC, DAH, DAL Timer: TM1C0, TM1C1, TM1AL, TM1AH, TM0C0, TM0C1, TM0AL, TM0AH General: ACC, MP1, IAR1, TBLP, TBLH, TBHP, PCL, STATUS I/O: PCS1, PCPU, PBS0				

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

■ MCU function module usage description:

- SPI1is used for controlling the external flash – used pins: SCS, SCK, MISO, MOSI
- Timer1interrupt is used to play voice operations – interrupt entry address: 0CH
- Timer0 interrupt is used for the play sentence operation – interrupt entry address: 08H
- DAC module is used for the flash audio data D/A converter – used pins: AUD, AUDIN
- Power amplifier module – used pins:SP+, SP-
- Implements the optimize the RAM BANK0 area (BANK0:20/128 (15%); the BANK1:17/128 (13%))

■ Different function calls require different PROM sizes – see below:

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

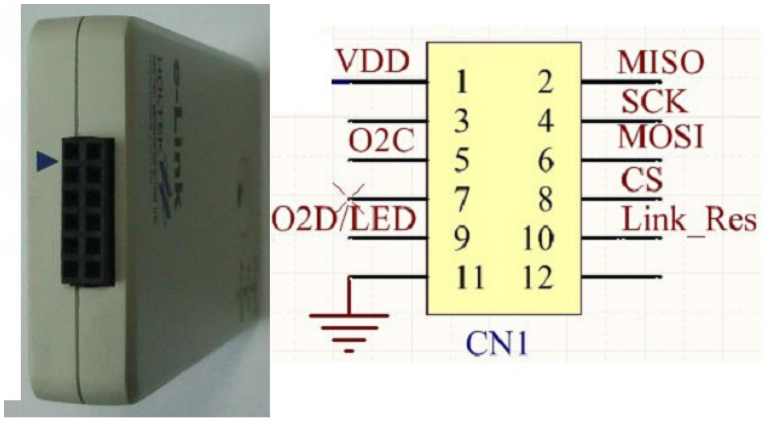
■ Using a specified MCU system frequency, the maximum audio source sampling rate using different compressed formats is shown in the following table:

System Frequency / Compression Mode	8MHz	12MHz	16MHz
HT-ADPCM4	13kHz	20kHz	27kHz
HT-PCM12	12kHz	18kHz	24kHz
HT-uPCM8	11kHz	17kHz	22kHz
HT-PCM16	13kHz	19kHz	26kHz

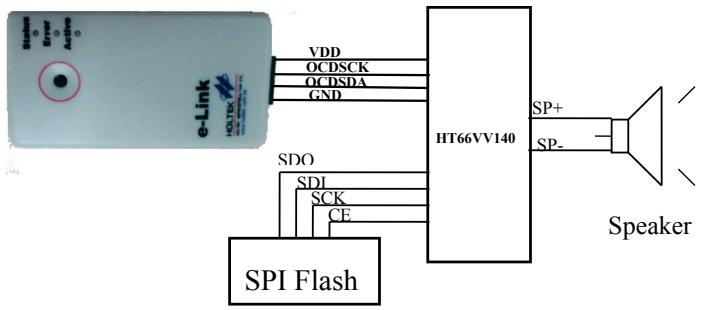
■ Emulator and Connection

This MCU uses the e-Link simulator and the EV chip HT66VV140 for simulating and debugging. In addition an external SPI Flash is needed.

- e-Link Pin Assignment:



- HT66VV140 VDD, GND, OCDSCK, OCDSDA pins connection to the e-Link.



Note: Refer to "Connection for Programming DAT File to the Flash" section for the SPI flash connection and programming.

HT66FV150

■ Resource Usage Table:

Resources Used \ Compression Mode	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%)				
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H	No decoding array
Other programs fixed memory address in the PROM	57EH–582H 583H–58CH	579H–57DH		6FEH–6FFH 702H–703H	
Stack (layers)	2				
Registers used	SPI1: SPIC0, SPIC1, SPID D/A: USVC, DAH, DAL Timer: TM1C0, TM1C1, TM1AL, TM1AH, TM0C0, TM0C1, TM0AL, TM0AH General: ACC, MP1, IAR1, TBLP, TBLH, TBHP, PCL, STATUS I/O: PCS1, PCPU, PBS0				

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

■ MCU function module usage description:

- SPI1is used for controlling the external flash – used pin: SCS, SCK, MISO, MOSI
- Timer1interrupt is used for play voice operation – interrupt entry address: 0CH
- Timer0 interrupt is used for the play sentence operation – interrupt entry address: 08H
- DAC module is used for the flash audio data D/A converter – used pin: AUD, AUDIN
- Power amplifier module – used pin:SP+, SP-
- Implements the optimize the RAM BANK0 area (BANK0:20/128 (15%); the BANK1:17/128 (13%))

■ Different function calls require different PROM sizes, see the table below:

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

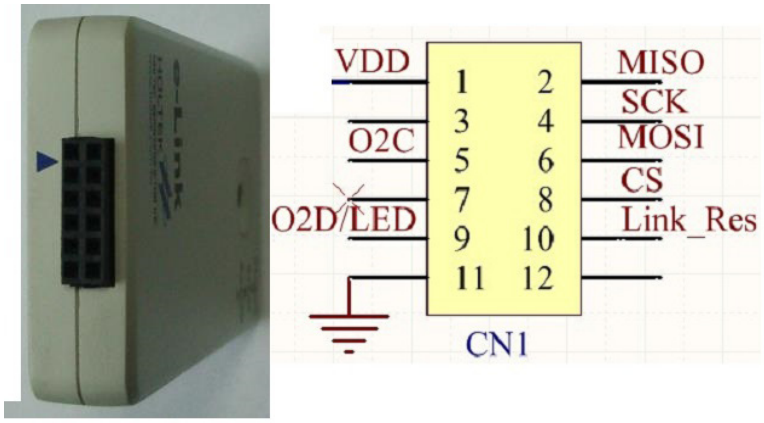
■ Using a specified MCU system frequency, the maximum audio source sampling rate using different compressed formats is shown in the following table:

Compression Mode \ System Frequency	8MHz	12MHz	16MHz
HT-ADPCM4	13kHz	20kHz	27kHz
HT-PCM12	12kHz	18kHz	24kHz
HT-uPCM8	11kHz	17kHz	22kHz
HT-PCM16	13kHz	19kHz	26kHz

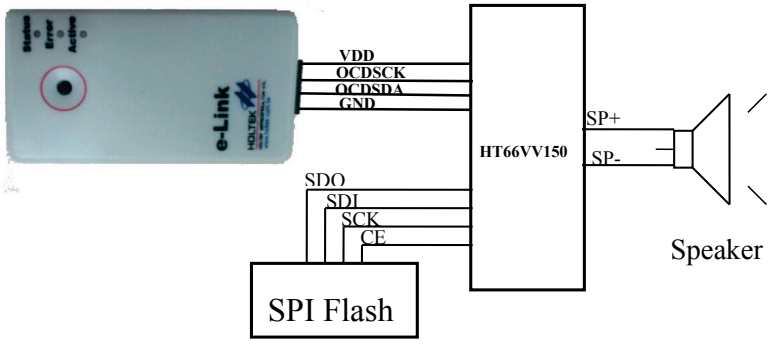
■ Emulator and Connection

This MCU uses the e-Link simulator and the EV chip HT66VV150 for simulating and debugging. In addition an external SPI Flash is needed.

- e-Link Pin Assignment:



- HT66VV150 VDD, GND, OCDSCK, OCDSDA pins connection to the e-Link.



Note: Refer to "Connection for Programming DAT File to the Flash" section for SPI flash connection and programming.

HT66FV160

■ Resource Usage Table:

Compression Mode / Resources Used	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%)				
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H	No decoding array
Other programs fixed memory address in the PROM	57EH–582H 583H–58CH	579H–57DH		6FEH–6FFH 702H–703H	700H–704H
Stack (layers)	2				
Registers used	SPI1: SPIC0, SPIC1, SPID D/A: USVC, DAH, DAL Timer: TM1C0, TM1C1, TM1AL, TM1AH, TM0C0, TM0C1, TM0AL, TM0AH General: ACC, MP1, IAR1, TBLP, TBLH, TBHP, PCL, STATUS I/O: PCS1, PCPU, PBS0				

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

■ MCU function module usage description:

- SPI1is used for controlling the external flash – used pin: SCS, SCK, MISO, MOSI
- Timer1interrupt is used for play voice operation – interrupt entry address: 0CH
- Timer0 interrupt is used for the play sentence operation – interrupt entry address: 08H
- DAC module is used for the flash audio data D/A converter – used pin: AUD, AUDIN
- Power amplifier module – used pin:SP+, SP-
- Implements the optimize the RAM BANK0 area (BANK0:21/128 (16%); the BANK1:17/128 (13%))

■ Different function calls require different PROM sizes, as shown in the table below:

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

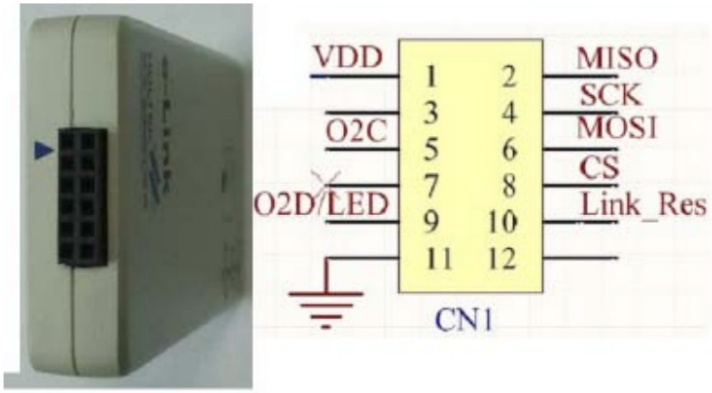
■ Using a specified MCU system frequency, the maximum audio source sampling rate using different compressed formats is shown in the following table:

System Frequency / Compression Mode	8MHz	12MHz	16MHz
HT-ADPCM4	13kHz	20kHz	26kHz
HT-PCM12	11kHz	17kHz	23kHz
HT-uPCM8	11kHz	16kHz	22kHz
HT-PCM16	12kHz	19kHz	25kHz

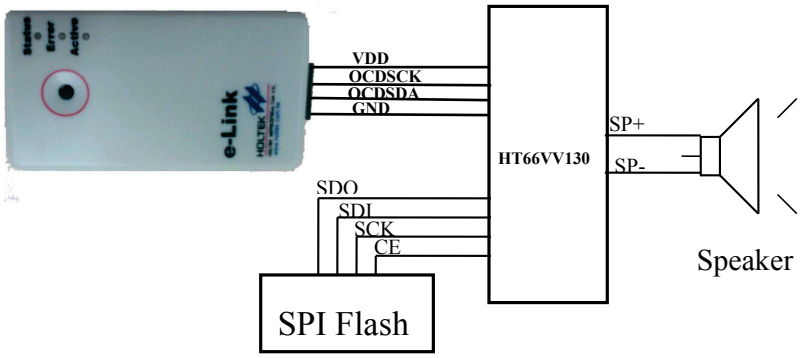
■ Emulator and Connection

This MCU uses the e-Link simulator and the EV chip HT66VV160 for simulating and debugging. In addition an external SPI Flash is needed.

- e-Link Pin Assignment:



- HT66VV130 VDD, GND, OCDSCK, OCDSDA pin connection to the e-Link.



Note: Refer to "Connection for Programming DAT File to the Flash" section for SPI flash connection and programming.

BH67F2262

■ Resource Usage Table:

Compression Mode / Resources Used	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%)				
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H	No decoding array
Other programs fixed memory address in the PROM	57EH–582H 583H–58CH	579H–57DH		6FEH–6FFH 702H–703H	
Stack(layers)	2				
Registers used	SPI1: SPIC0, SPIC1, SPID PWM: PWMC, USVC, PLADH, PLADL Timer: TM1C0, TM1C1, TM1AL, TM1AH, TM0C0, TM0C1, TM0AL, TM0AH General: ACC, MP1, IAR1, TBLP, TBLH, TBHP, PCL, STATUS I/O: PMPS0, PBPU, PBS1, PGS1				

Note: The user code cannot occupy the space specified for the decoding array.

■ MCU function module usage description:

- SPI1 is used for controlling the external flash – used pin: SPISCSB, SPISCK, SPISDO, SPISDI
- Timer1 interrupt is used for the play voice operation – interrupt entry address: 14H
- Timer0 interrupt is used for the play sentence operation – interrupt entry address: 10H
- PWM: module is used for the flash audio data converter – used pin: PWM1, PWM2
- Implements the optimization for RAM BANK0 area (BANK0: 21/128 (16%); BANK1: 17/128 (13%); BANK2: 0/128(0%); BANK3: 0/128(0%))

■ Different function calls require different PROM sizes, as shown in the table below:

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

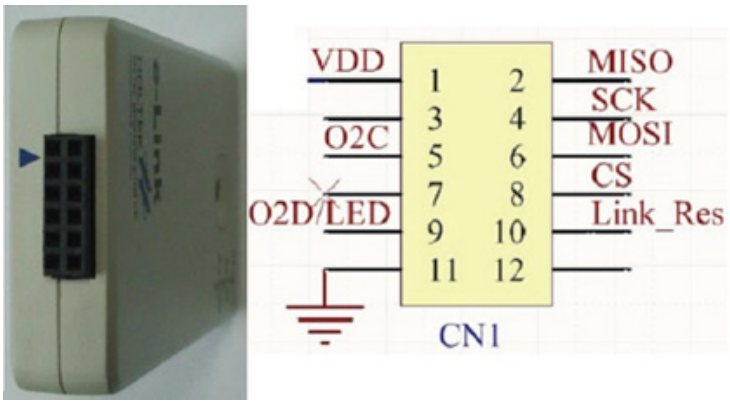
■ Using a specified MCU system frequency, the maximum audio source sampling rate using different compressed formats is shown in the following table:

System Frequency / The Maximum Audio Source Sampling Rate / Compression Mode	8MHz	12MHz	16MHz
HT-ADPCM4	13kHz	20kHz	26kHz
HT-PCM12	11kHz	17kHz	23kHz
HT-uPCM8	11kHz	16kHz	22kHz
HT-PCM16	12kHz	19kHz	25kHz

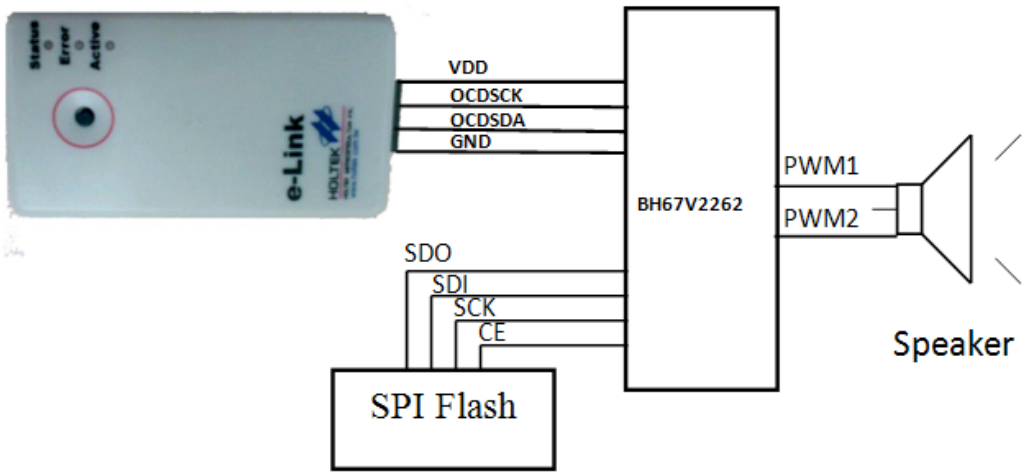
■ Emulator and Connection

This MCU uses the e-Link simulator and the EV chip BH67V2262 for simulating and debugging. In addition an external SPI Flash is needed.

- e-Link Pin Assignment



- BH67V2262 pins, VDD, GND, OCDSCK, OCDSDA, are relevantly connected to the e-Link



Note: Refer to "Connection for Programming DAT File to the Flash" section for SPI flash connection and programming.

HT45F67

■ Resource Usage Table:

Resources Used \ Compression Mode	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%)			
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H
Other programs fixed memory address in the PROM	57EH–582H 583H–58AH	579H–57DH	704H–72CH	6FEH–6FFH 702H–703H
Stack (layers)	2			
Registers used	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			

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

■ MCU function module usage description:

- SPI1is used for controlling the external flash – used pin: SDI1, SDO1, SCK1, SCS1B0
- Timer2 interrupt is used for play voice operation – interrupt entry address: 10H
- Timer1 interrupt is used for the play sentence operation – interrupt entry address: 14H
- DAC module is used for the flash audio data D/A converter – used pin: AUD
- Implements the optimize the RAM BANK0 area (BANK0:23/128 (17%); the BANK1:17/128 (13%))

■ Different function calls require different PROM sizes, shown in the table below:

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

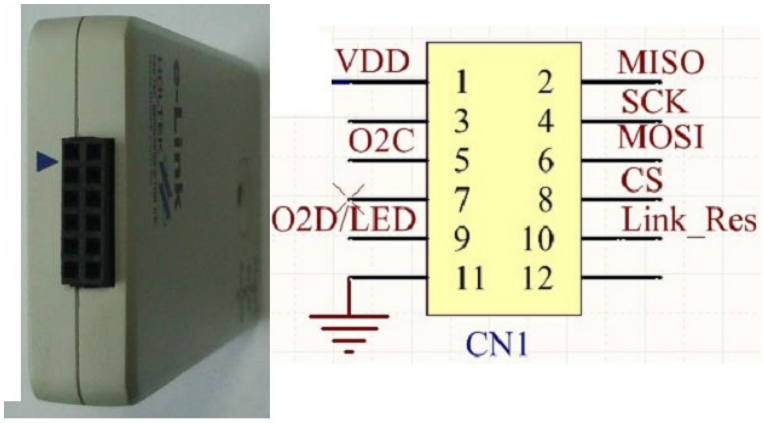
■ Using a specified MCU system frequency, the maximum audio source sampling rate using different compressed format is shown in the following table:

Compression Mode \ System Frequency	4MHz	8MHz	12MHz
HT-ADPCM4	6kHz	13kHz	20kHz
HT-PCM12	5kHz	11kHz	17kHz
HT-uPCM8	5kHz	11kHz	16kHz

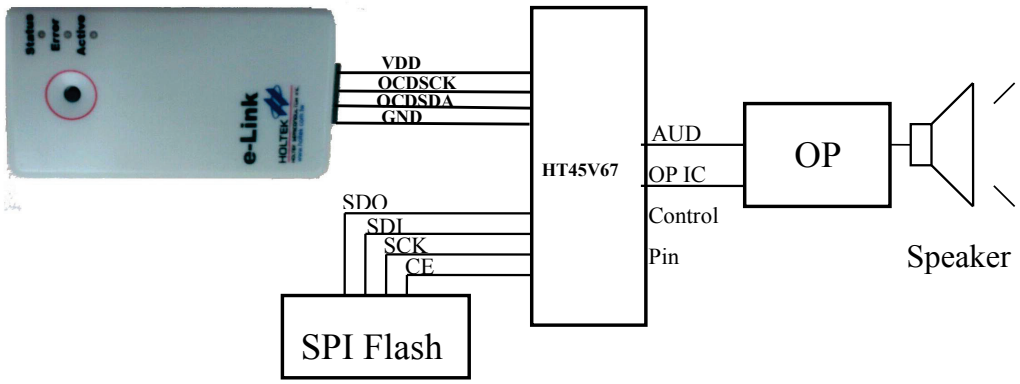
■ Emulator and Connection

This MCU uses the e-Link simulator and the EV chip HT45V67 for simulating and debugging. In addition an external SPI Flash and audio amplifier circuit module are needed.

- e-Link Pin Assignment:

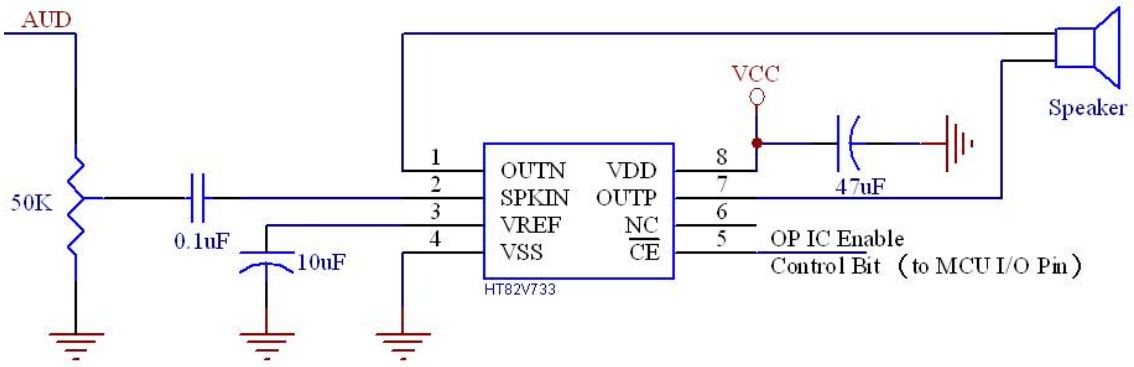


- HT45V67 VDD, GND, OCDSCK, OCDSDA pins connection to the e-Link.



Note1: Refer to "Connection for Programming DAT File to the Flash" section for SPI flash connection and programming.

Note2: The following figure shows an audio amplifier reference circuit using the HT82V733:



HT45F65

■ Resource Usage Table:

Resources Used \ Compression Mode	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%)			
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H
Other programs fixed memory address in the PROM	57EH–582H 583H–58AH	579H–57DH	704H–72CH	6FEH–6FFH 702H–703H
Stack (layers)	2			
Registers used	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			

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

■ The MCU function module usage description:

- SPI1 is used for controlling the external flash – used pin: SDI1, SDO1, SCK1, SCS1B0
- Timer2 interrupt is used for play voice operation – interrupt entry address:18H
- Timer1 interrupt is used for the play sentence operation – interrupt entry address:10H
- DAC module is used for the flash audio data D/A converter – used pin: AUD
- Implements the optimize the RAM BANK0 area (BANK0:23/128 (17%); the BANK1:17/128 (13%))

■ Different function calls require different PROM sizes, shown in the table below:

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

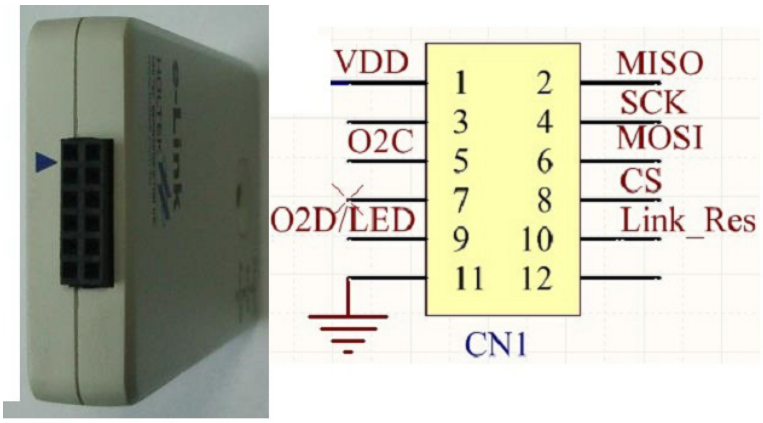
■ Using a specified MCU system frequency, the maximum audio source sampling rate using different compressed format is shown in the following table:

System Frequency / Compression Mode	4MHz	8MHz	12MHz
HT-ADPCM4	6kHz	13kHz	20kHz
HT-PCM12	5kHz	11kHz	17kHz
HT-uPCM8	5kHz	11kHz	16kHz

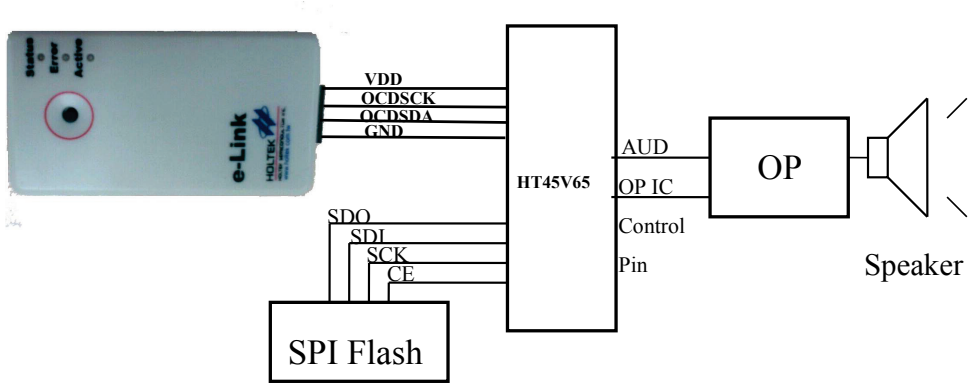
■ Emulator and Connection

This MCU uses the e-Link simulator and the EV chip HT45V65 for simulating and debugging. In addition an external SPI Flash and audio amplifier circuit module are needed.

- e-Link Pin Assignment:

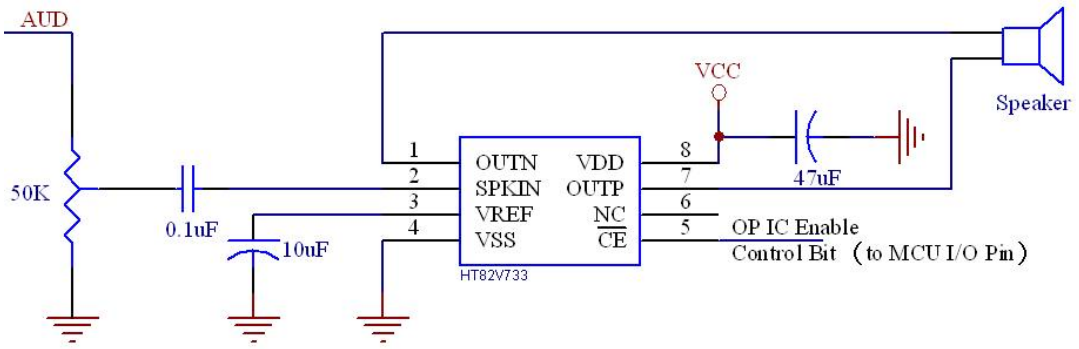


- HT45V65 VDD, GND, OCDSCK, OCDSDA pins connection to the e-Link.



Note1: Refer to "Connection for Programming DAT File to the Flash" section for SPI flash connection and programming.

Note2: The following figure shows an audio amplifier reference circuit using the HT82V733:



HT45F3W

■ Resource Usage Table:

Resources Used \ Compression Mode	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%)			
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H
Other programs fixed memory address in the PROM	57EH–582H 583H–58AH	579H–57DH	704H–72CH	6FEH–6FFH 702H–703H
Stack (layers)	2			
Registers used	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			

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

■ MCU function module usage description:

- SPI1 is used for controlling the external flash – used pin: S1DI, S1DO1, S1CK, S1CS
- Timer2 interrupt is used for play voice operation – interrupt entry address: 10H
- Timer1 interrupt is used for the play sentence operation – interrupt entry address: 0CH
- DAC module is used for the flash audio data D/A converter – used pin: AUD
- Implements the optimize the RAM BANK0 area (BANK0:23/128 (17%); the BANK1:17/128 (13%))

■ Different function calls require different PROM sizes, shown in the table below:

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

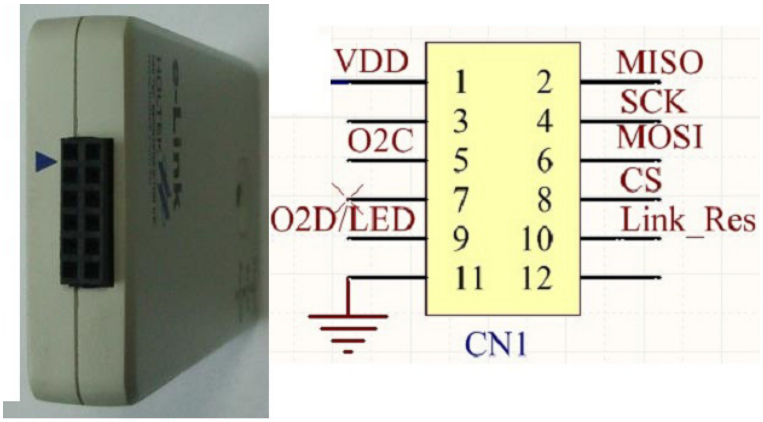
■ Using a specified MCU system frequency, the maximum audio source sampling rate using different compressed formats is shown in the following table:

Compression Mode \ System Frequency	4MHz	8MHz	12MHz
HT-ADPCM4	6kHz	13kHz	20kHz
HT-PCM12	5kHz	11kHz	17kHz
HT-uPCM8	5kHz	11kHz	16kHz

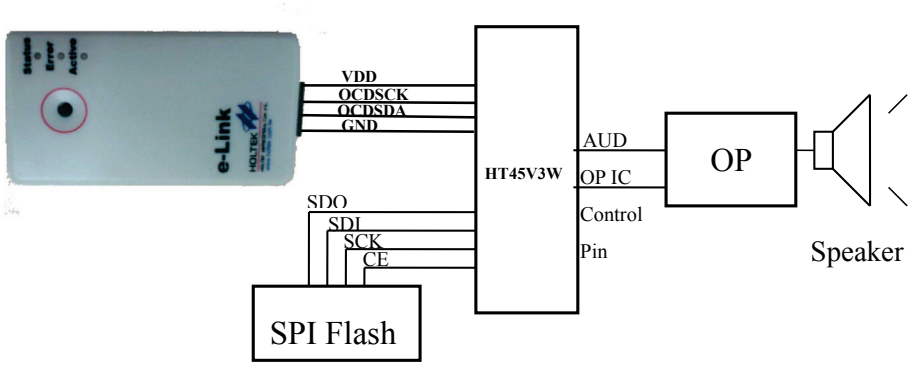
Emulator and Connection

This MCU uses the e-Link simulator and the EV chip HT45V3W for simulating and debugging. In addition an external SPI Flash and audio amplifier circuit module are needed.

- e-Link Pin Assignment:

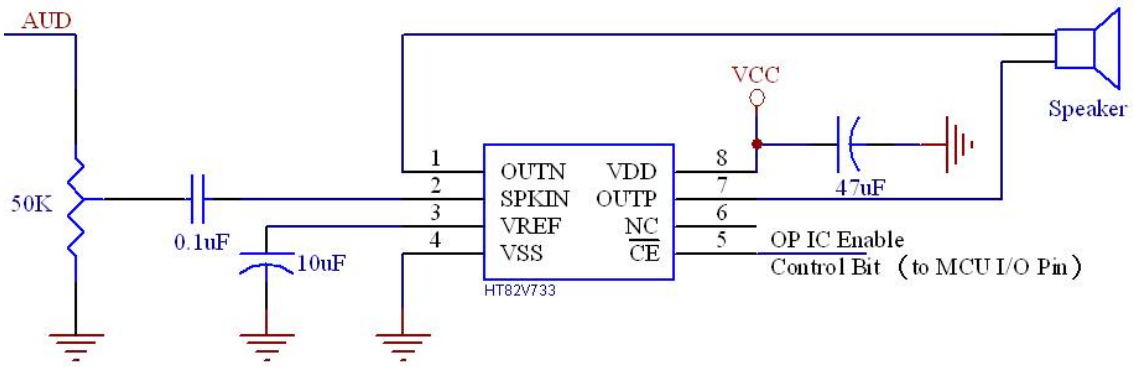


- HT45V3W VDD, GND, OCDSCK, OCDSDA pins connection to the e-Link.



Note1: Refer to "Connection for Programming DAT File to the Flash" section for SPI flash connection and programming.

Note2: The following figure shows an audio amplifier reference circuit using the HT82V733:



HT66F4550

■ Resource Usage Table:

Resources Used \ Compression Mode	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8	HT-PCM16
PROM(Word)	632/8192 (7%)	241/8192 (3%)	29/8192 (1%)	316/8192 (4%)	17/8192 (1%)
RAM(Byte)	37/384(9%)				
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H	No decoding array
Other programs fixed memory address in the PROM	57EH–582H 583H–58CH	579H–57DH		6FEH–6FFH 702H–703H	
Stack (layers)	2				
Registers used	SPI1: SPIC0, SPIC1, SPID D/A: DAH, DAL Timer: TM1C0, TM1C1, TM1AL, TM1AH, TM0C0, TM0C1, TM0AL, TM0AH General: ACC, MP1, IAR1, TBLP, TBLH, TBHP, PCL, STATUS I/O: PAS0, PAS1, PCS0, IFS, PAPU, PCPU				

Note: The user code cannot occupy the space specified for the decoding array.

■ MCU function module usage description:

- SPI1 is used for controlling the external flash – used pin: SCS(PC0), SCK(PA4), SDI(PA1), SDO(PC1)
- Timer1 interrupt is used for play voice operation – interrupt entry address: 10H
- Timer0 interrupt is used for the play sentence operation – interrupt entry address: 0CH
- DAC module is used for the flash audio data D/A converter – used pin: DACO
- Implements the optimize the RAM BANK0 area (BANK0: 20/128(15%); the BANK1: 17/128(13%); the BANK2: 0/128(0%))
- As without internal digital volume control, it requires external circuit to adjust the volume

■ Different function calls require different PROM sizes, shown in the table below:

Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	10
_PLAY_SENTENCE	9
_PLAY_SENTENCE_INDEX	18
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

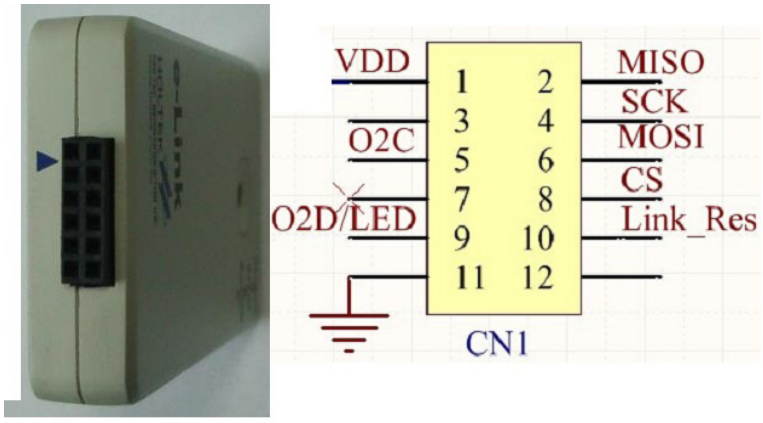
■ Using a specified MCU system frequency, the maximum audio source sampling rate using different compressed formats is shown in the following table:

System Frequency / Compression Mode	2MHz	4MHz	8MHz
HT-ADPCM4	3KHz	6KHz	13KHz
HT-PCM12	3KHz	6KHz	13KHz
HT-uPCM8	2KHz	5KHz	11KHz
HT-PCM16	3KHz	6KHz	13KHz

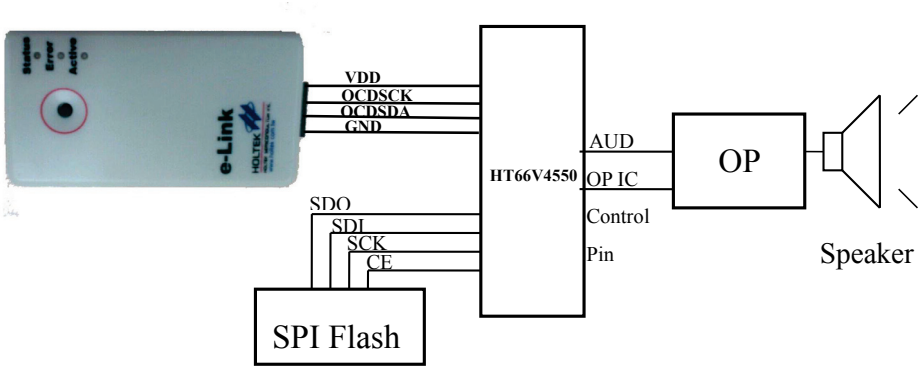
■ Emulator and Connection

This MCU uses the e-Link simulator and the EV chip HT66V4550 for simulating and debugging. In addition an external SPI Flash and audio amplifier circuit module are needed.

- e-Link Pin Assignment:

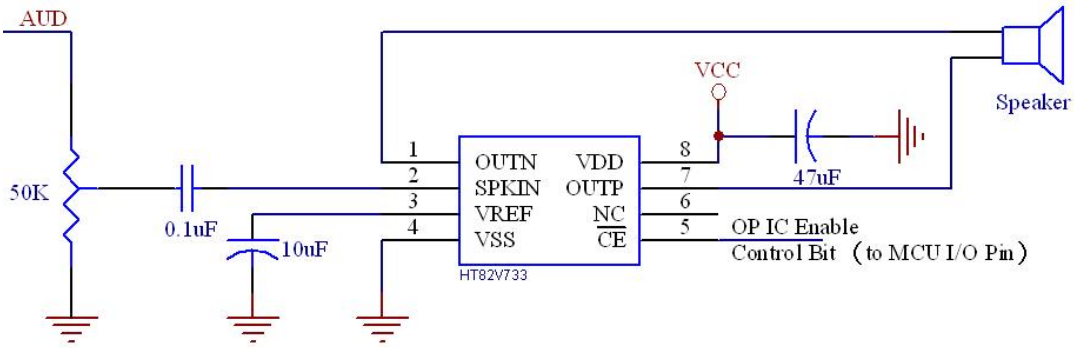


- HT66V4550 VDD, GND, OCDSCK, OCDSDA pins connection to the e-Link.



Note1: Refer to "Connection for Programming DAT File to the Flash" section for SPI flash connection and programming.

Note2: The following figure shows an audio amplifier reference circuit using the HT82V733:



HT45F23A

■ Resource Usage Table:

Resources Used \ Compression Mode	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%)			
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H
Other programs fixed memory address in the PROM	57EH–582H 583H–58AH	579H–57DH	704H–72CH	6FEH–6FFH 702H–703H
Stack (layers)	2			
Registers used	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			

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

■ MCU function module usage description:

- SPI1 is used for controlling the external flash – used pin: SDI, SDO, SCK, SCS
- Timer1 interrupt is used for play voice operation – interrupt entry address: 10H
- Timer0 interrupt is used for the play sentence operation – interrupt entry address: 0CH
- DAC module is used for the flash audio data D/A converter – used pin: AUD

■ Different function calls require different PROM sizes, shown in the table below:

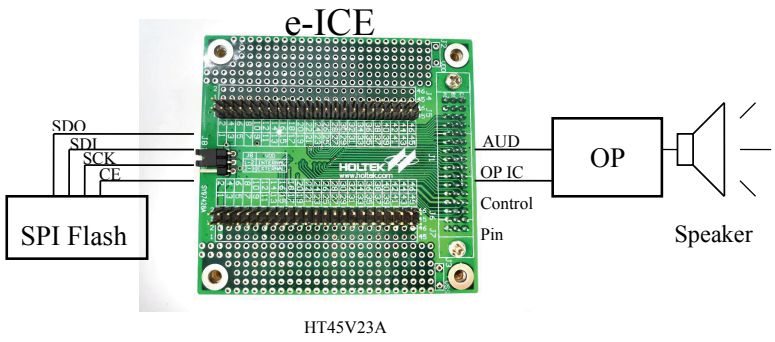
Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	13
_PLAY_SENTENCE	12
_PLAY_SENTENCE_INDEX	17
_VOLUME	9
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

■ Under a specified MCU system frequency, the maximum audio source sampling rate using different compressed format is shown in the following table:

Compression Mode \ System Frequency	2MHz	4MHz	8MHz
HT-ADPCM4	3kHz	6kHz	13kHz
HT-PCM12	2kHz	5kHz	11kHz
HT-uPCM8	2kHz	5kHz	11kHz

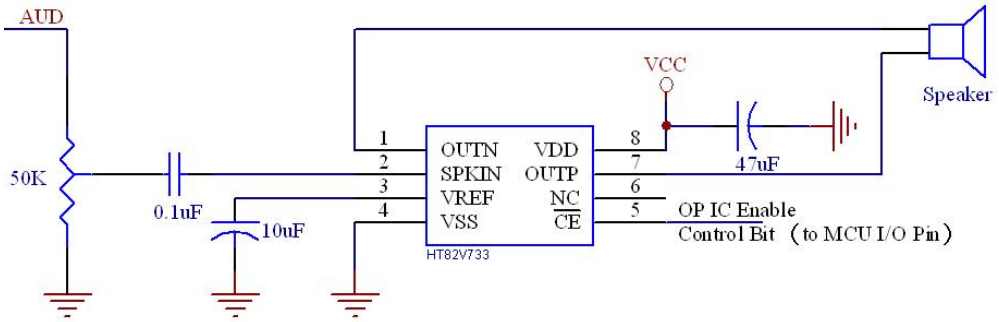
■ Emulator and Connection

This MCU uses the e-ICE (M1001D+D1088A) for simulating and debugging. In addition an external SPI Flash and audio amplifier circuit module are needed.



Note1: Refer to "Connection for Programming DAT File to the Flash" section for SPI flash connection and programming.

Note2: The following figure shown an audio amplifier reference circuit using the HT82V733:



HT45F24A

■ Resource Usage Table:

Resources Used \ Compression Mode	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%)			
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H
Other programs fixed memory address in the PROM	57EH–582H 583H–58AH	579H–57DH	704H–72CH	6FEH–6FFH 702H–703H
Stack (layers)	2			
Registers used	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			

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

■ MCU function module usage description:

- SPI1 is used for controlling the external flash – used pin: SDI, SDO, SCK, SCS
- Timer1 interrupt is used for play voice operation – interrupt entry address: 10H
- Timer0 interrupt is used for the play sentence operation – interrupt entry address: 0CH
- DAC module is used for the flash audio data D/A converter – used pin: AUD

■ Different function calls require different PROM sizes, shown in the table below:

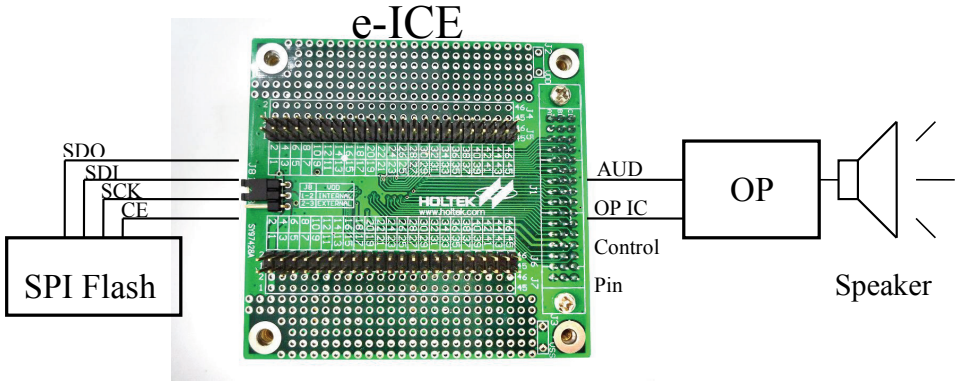
Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	13
_PLAY_SENTENCE	12
_PLAY_SENTENCE_INDEX	17
_VOLUME	9
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

■ Using a specified MCU system frequency, the maximum audio source sampling rate using different compressed format is shown in the following table:

Compression Mode \ System Frequency	2MHz	4MHz	8MHz
HT-ADPCM4	3kHz	6kHz	13kHz
HT-PCM12	2kHz	5kHz	11kHz
HT-uPCM8	2kHz	5kHz	11kHz

■ Emulator and Connection

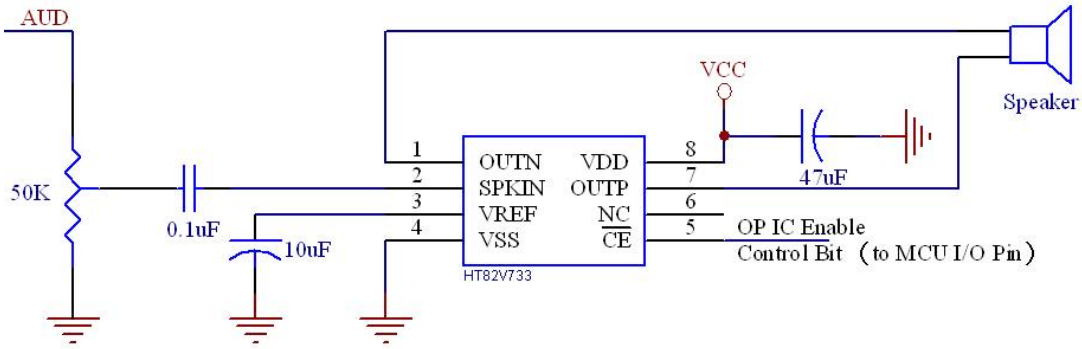
This MCU uses the e-ICE (M100D+D1095A) for simulating and debugging. In addition an external SPI Flash and audio amplifier circuit module are needed.



HT45V24A

Note1: Refer to "Connection for Programming DAT File to the Flash" section for SPI flash connection and programming.

Note2: The following figure shows an audio reference circuit built using the HT82V733:



HT83F02

■ Resource Usage Table:

Resources Used \ Compression Mode	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%)			
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H
Other programs fixed memory address in the PROM	57EH–582H 583H–58AH	579H–57DH	704H–72CH	6FEH–6FFH 702H–703H
Stack (layers)	2			
Registers used	SPI: SIMC0A, SIMC2A, SIMDRA, SIMDRB D/A: DACTRL, DAH, DAL Timer: TMR0, TMR0C, TMR1, TMR1C General: ACC, MP1, IAR1, INTC, TBLP, TBLH			

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

■ MCU function module usage description:

- SPIA is used for controlling the external flash (used pin: SDAA, SCLA, SDIA, SCSA)
- Timer1 interrupt is used for play voice operation (interrupt entry address: 0CH)
- Timer0 interrupt is used for the play sentence operation (interrupt entry address: 08H)
- DAC module is used for the flash audio data D/A converter. (used pin:AUD)

■ Different functions require different PROM sizes, shown in the table below:

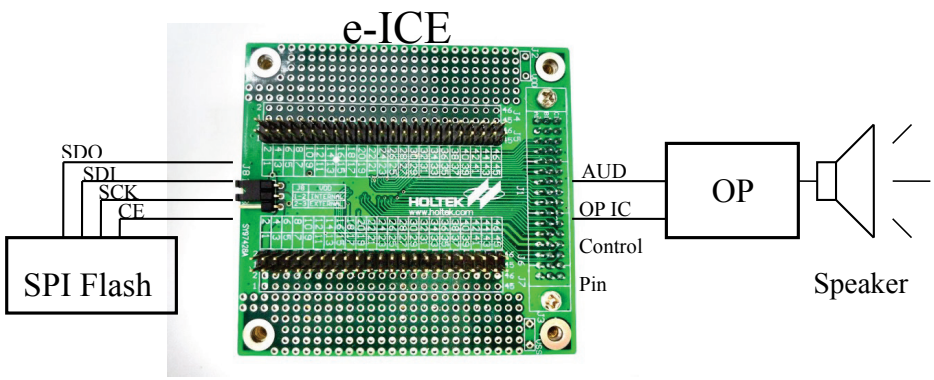
Macro Name	PROM size cost per call (Unit: Word)
_PLAY_VOICE	14
_PLAY_SENTENCE	13
_PLAY_SENTENCE_INDEX	17
_VOLUME	6
_MODIFY_SAMPLINGRATE	11
_PAUSE	3
_RESUME	3

■ Using a specified MCU system frequency, the maximum audio source sampling rate using different compressed format is shown in the following table:

Compression Mode \ System Frequency	4MHz	8MHz	12MHz
HT-ADPCM4	7kHz	14kHz	21kHz
HT-PCM12	5kHz	11kHz	17kHz
HT-uPCM8	6kHz	12kHz	18kHz

■ Emulator and Connection

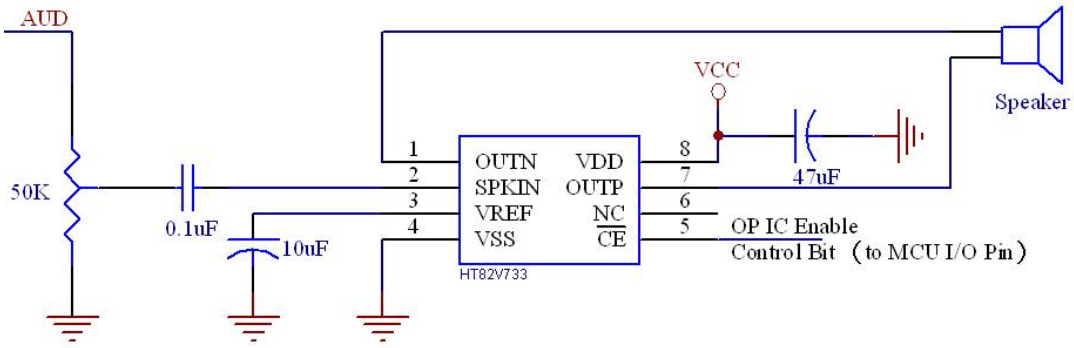
This MCU uses the e-ICE (M1001D+D1026A) for simulating and debugging. In addition an external SPI Flash and audio amplifier circuit module are needed.



HT83F02EV

Note1: Refer to "Connection for Programming DAT File to the Flash" section for SPI flash connection and programming.

Note2: The following figure shows an audio amplifier reference circuit using the HT82V733:



HT86BX0

■ Resource Usage Table:

- HT86B03 Resource Usage Table:

Resources Used \ Compression Mode	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%)			
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H
Other programs fixed memory address in the PROM	57EH–582H 583H–58AH	579H–57DH	704H–72CH	6FEH–6FFH 702H–703H
Stack (layers)	2			
Registers used	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			

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

- HT86B10, HT86B20, HT86B30 Resource Usage Table:

Resources Used \ Compression Mode	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%)			
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H
Other programs fixed memory address in the PROM	57EH–582H 583H–58AH	579H–57DH	704H–72CH	6FEH–6FFH 702H–703H
Stack (layers)	2			
Registers used	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			

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

- HT86B40, HT86B50, HT86B60, HT86B70, HT86B80, HT86B90 Resource Usage Table:

Resources Used \ Compression Mode	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%)			
Compressed decoding array stored address in PROM		500H–578H	No decoding array	600H–6FDH 700H–701H
Other programs fixed memory address in the PROM	57EH–582H 583H–58AH	579H–57DH	704H–72CH	6FEH–6FFH 702H–703H
Stack (layers)	2			
Registers used	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			

Note: 1. The user code cannot occupy the space specified for the decoding array.
2. Calculate cost PROM space: the Default + the selection of compression mode (can support mixed compression mode)

■ MCU function module usage description:

- HT86B03, HT86B10, HT86B20, HT86B30:
  - ♦ Timer0 interrupt is used for the play sentence operation – interrupt entry address: 08H
  - ♦ Timer1 interrupt is used for play voice operation – interrupt entry address: 0CH
  - ♦ DAC module is used for the flash audio data D/A converter – used pin: AUD
- HT86B40, HT86B50, HT86B60, HT86B70, HT86B80, HT86B90:
  - ♦ Timer0 interrupt is used for the play sentence operation – interrupt entry address: 08H
  - ♦ Timer2 interrupt is used for play voice operation – interrupt entry address: 10H
  - ♦ DAC module is used for the flash audio data D/A converter – used pin: AUD

■ Different functions require different PROM sizes, shown in the table below:

- 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

Using a specified MCU system frequency, the maximum audio source sampling rate using different compressed format is shown in the following table:

- HT86B03 / HT86B10 / HT86B20 / HT86B30

System Frequency / Compression Mode	4MHz	6MHz	8MHz
HT-ADPCM4	6kHz	10kHz	13kHz
HT-PCM12	5kHz	8kHz	11kHz
HT-uPCM8	5kHz	8kHz	11kHz

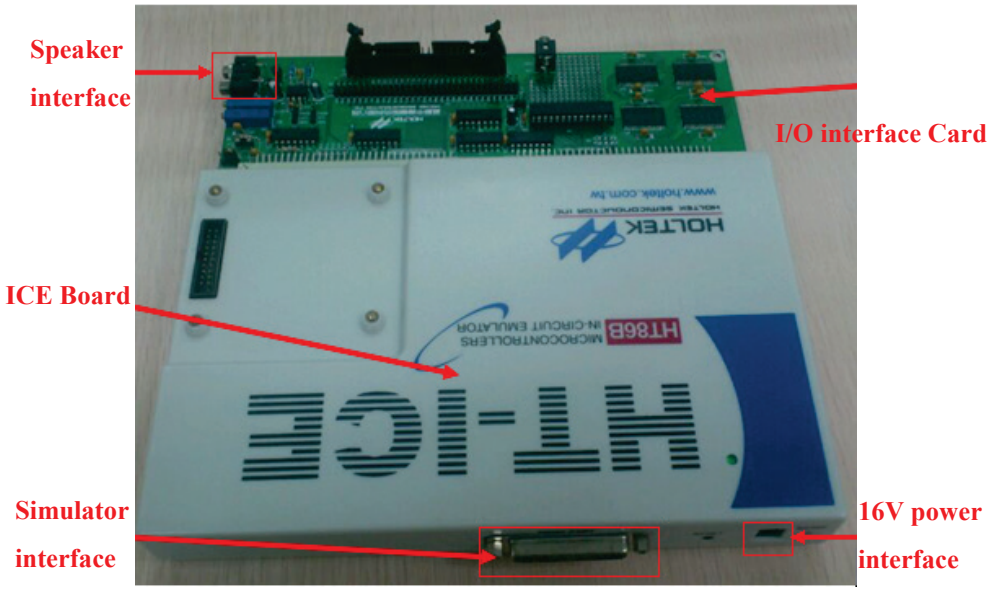
- HT86B40 / HT86B50/ HT86B60 / HT86B70 / HT86B80 / HT86B90

System Frequency / Compression Mode	4MHz	6MHz	8MHz
HT-ADPCM4	6kHz	10kHz	13kHz
HT-PCM12	5kHz	8kHz	11kHz
HT-uPCM8	5kHz	8kHz	11kHz

■ Emulator and Connection

This MCU uses the HT-ICE simulator HT86 Band the IO card for simulating and debugging. The specific process is as following:

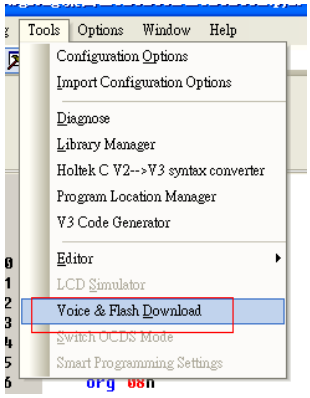
- Hardware Introduction



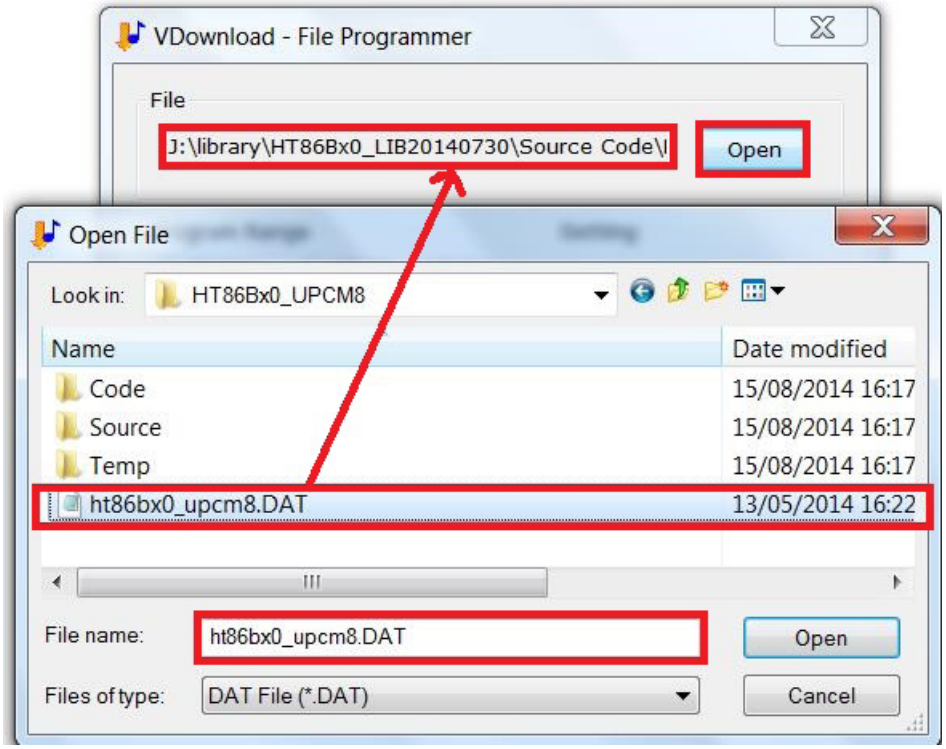
Print Cable

USB/Print converting Cable

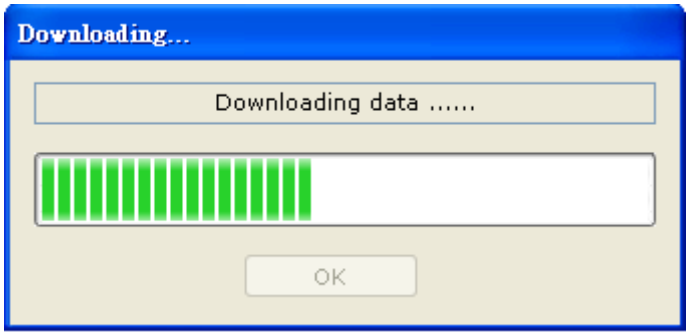
- Voice Workshop generates a .dat audio file.
- Connect to 16V power and use the Print cable to connect to PC for downloading and simulating. If there is no a Print interface on the PC, a USB/Printer conversion cable is needed.
- Create a new or open the existing HT86BX0 ICE 3000 project which used for the pc and hardware conversation, open the project Menu "Tools"→ "Voice & Flash Download", download voice signals, as shown below:



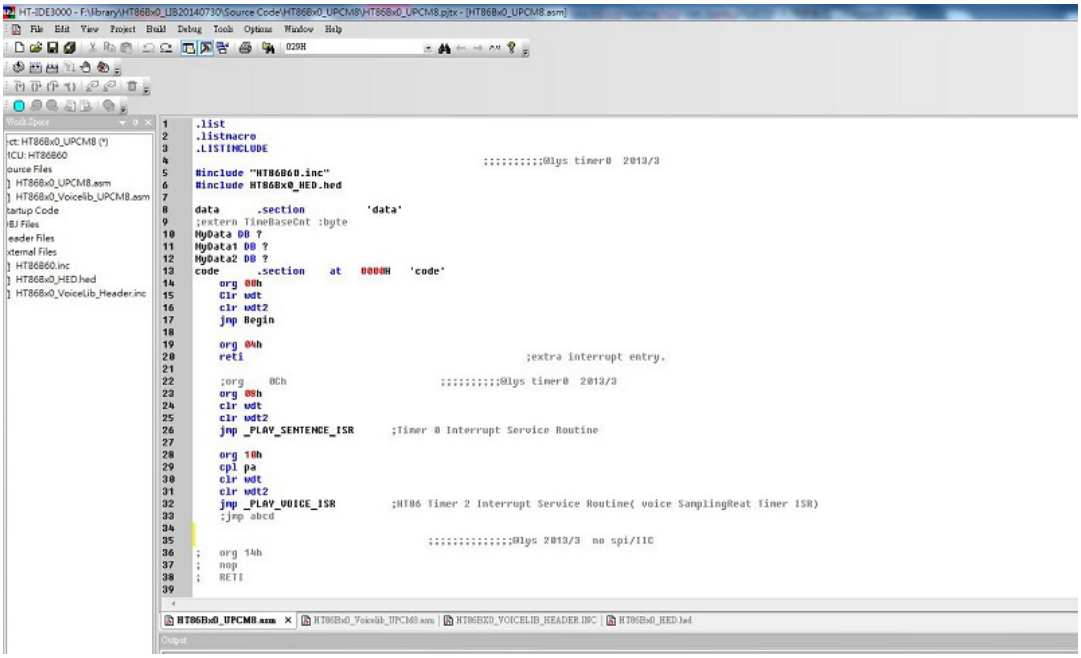
- Click 'open', load the dat file generated by the Workshop to the VDownload, as shown below:



- download data:



- Using IDE for the project emulation



4 Audacity Quick Start

Audacity Summary

Audacity is a free, open source (cross-platform) digital audio editor, recorder and mixer. The software can run on Windows, Mac OS X, GNU/Linux and other operating systems. It is a mature software application that comes with a long list of features such as:

- Recording
- Change tapes to digital recording or CD
- Edit Ogg Vorbis, MP3 and WAV files
- Cut, copy, paste and multitrack mixing
- Change the recording rate or pitch

Note: you can download the Audacity software for free on the website: http://audacity.sourceforge.net

Audacity Processing Flow

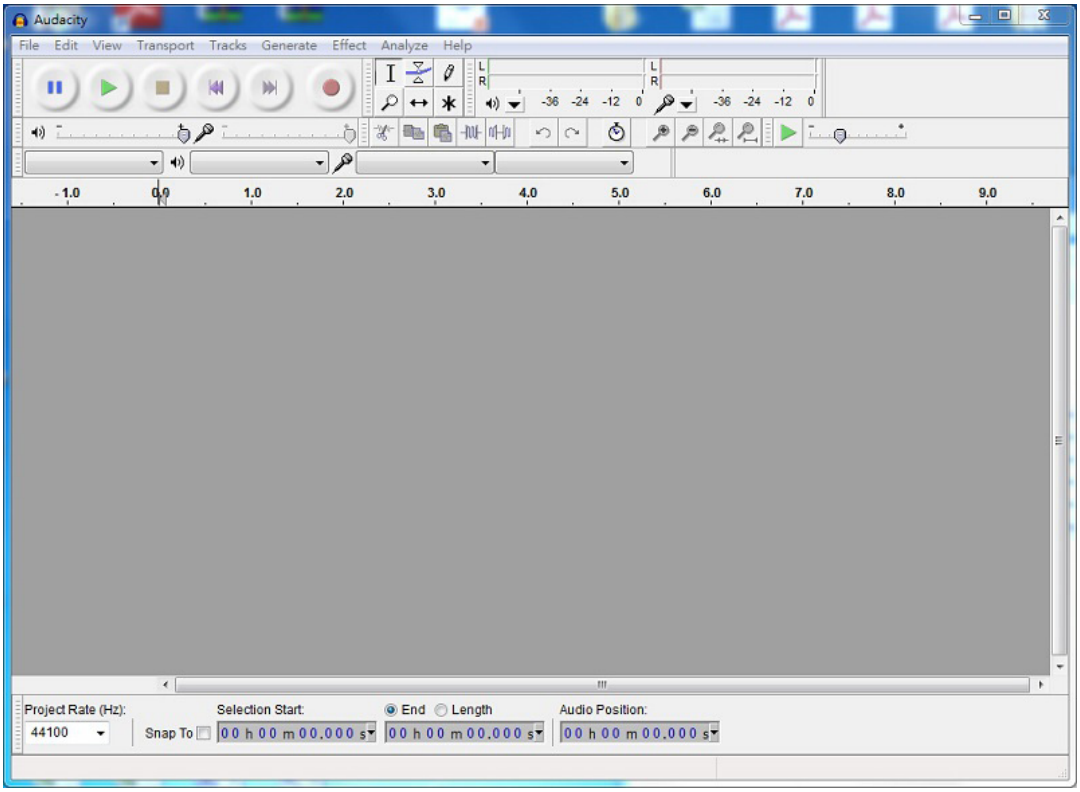
- Importing Audio
  Extract audio CDs to WAV format or import WAV, AIF9F, OGG or MP3 files into Audacity for direct use or recording

- Basic Audio Processing Operation
  Basic splicing(delete, insert, copy) volume control(envelope/amplify)
  fade in/fade out, noise removal
  insert a fixed length, silence a track, mix tracks, change the pitch

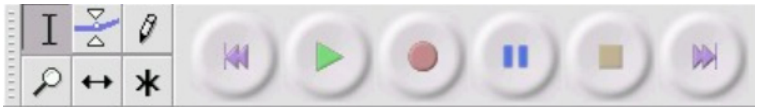
- Exporting audio files
  To export as wav, aiff, mp3 or ogg file and burn to Audio CDs.

Quick Start

■ Right double-click the icon  to open the Audacity software and the following interface appears:



- Audacity Interface Toolbars Overview



① Audacity Transport Toolbar Description

	Skip to start		Temporarily pauses playing or recording without losing the present location. Click Pause a second time to resume.
	Standard-speed playback. If an area of track is selected, only that selection will be played.		Stop playing or recording immediately.
	Start recording at the current cursor position.		Skip to End.

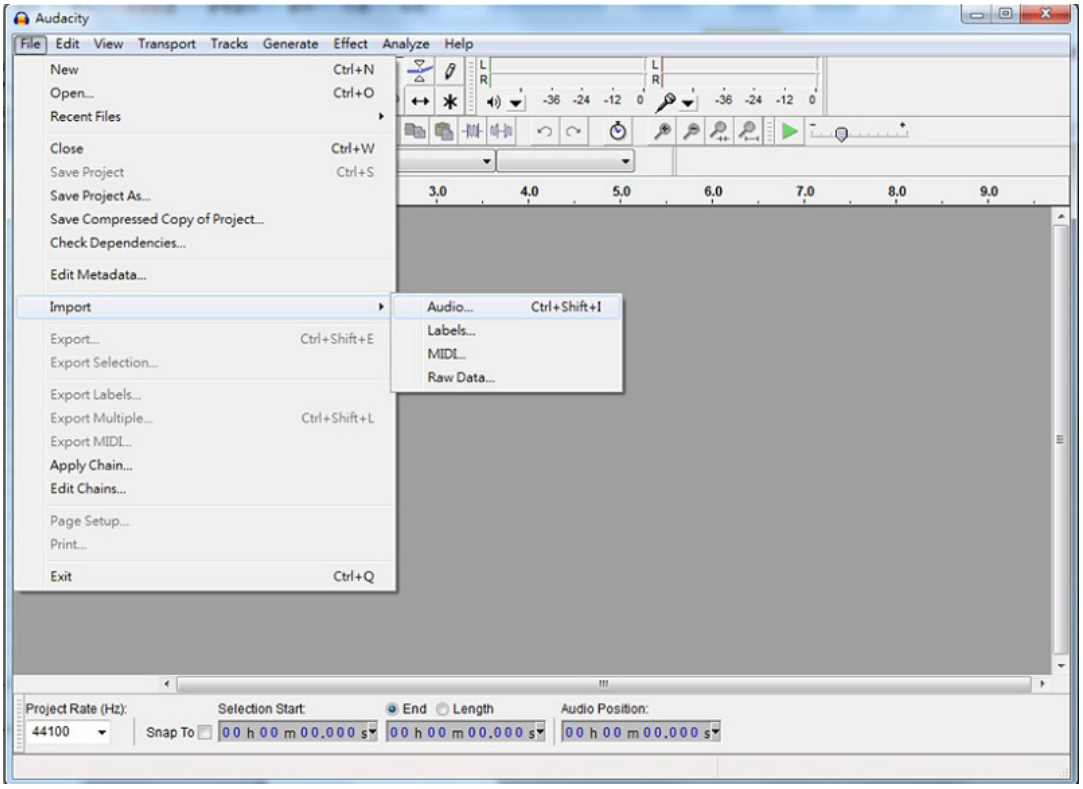
② Audacity Tools Toolbar Description

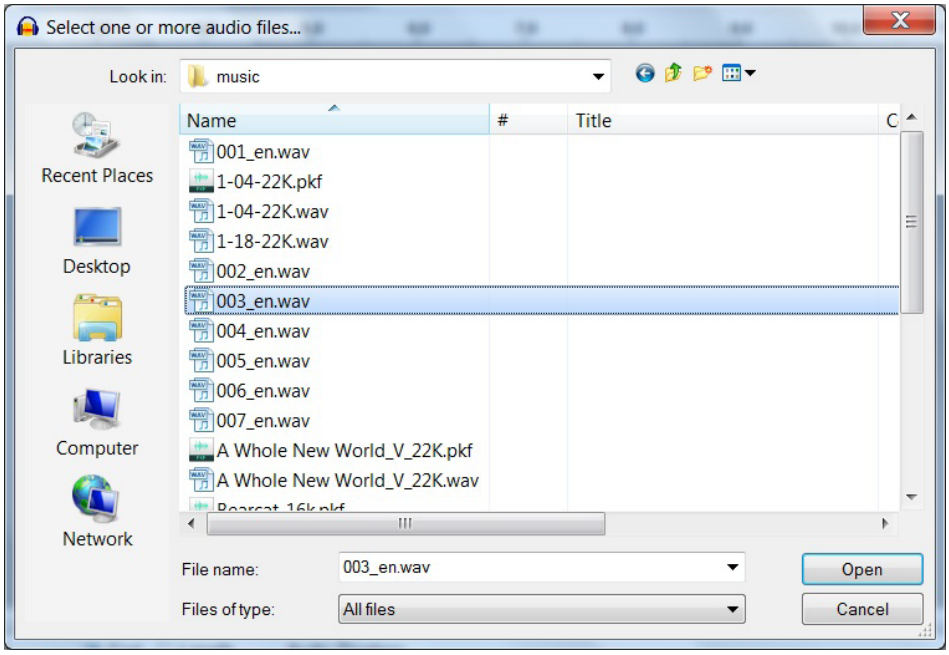
	Selection- click and drag to select a range of audio to play or edit		Zoom- zoom in or zoom out the track
	Envelope- made smooth volume change over the length of a track		Time shift- drag audio tracks left or right
	Draw- adjust the volume level of individual audio samples		Multi- Combine several tools into one. One tool is available at a time according to the mouse position or the pressed button.

■ Importing audio:

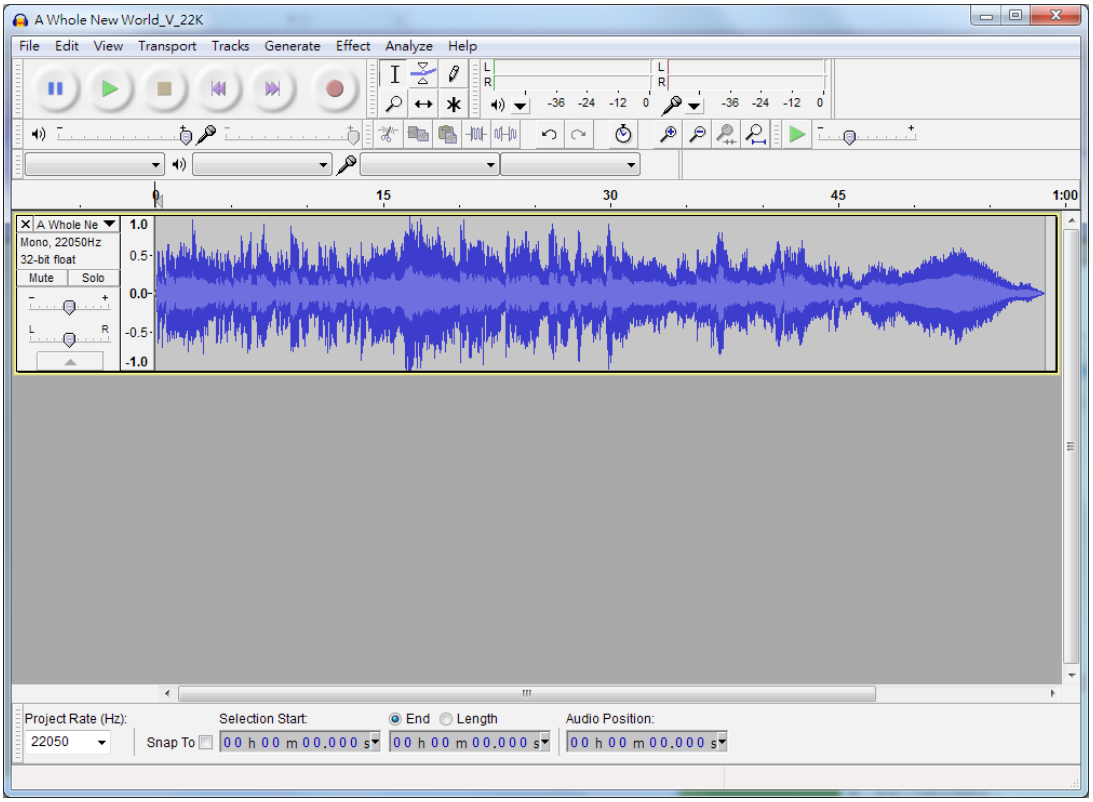
- There are usually the following three conditions:
  - ♦ Import music on an audio CD - necessary to “rip” the music into an audio file in a wav format first.
  - ♦ Import a recording – necessary to use appropriate software such as microphone recording software.
  - ♦ Import wav, aiff, ogg or mp3 file - directly open and use.

  Choose"File"–>"Import"–>"Audio"and select the audio file on your PC.



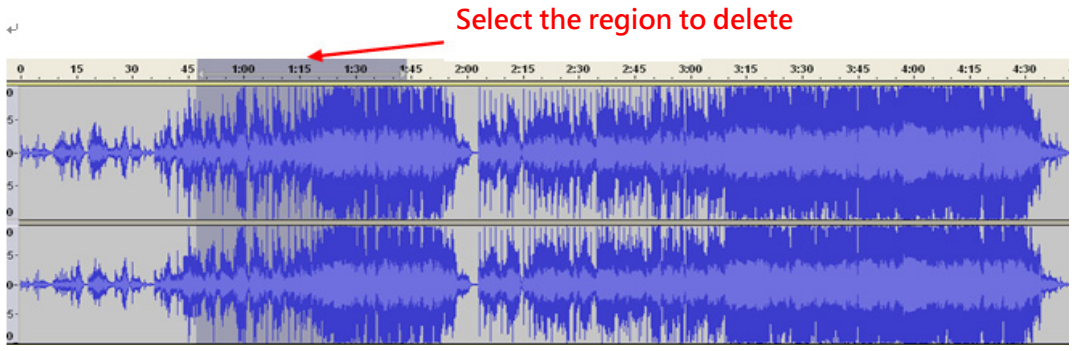


- The following interface appears after importing the audio file:

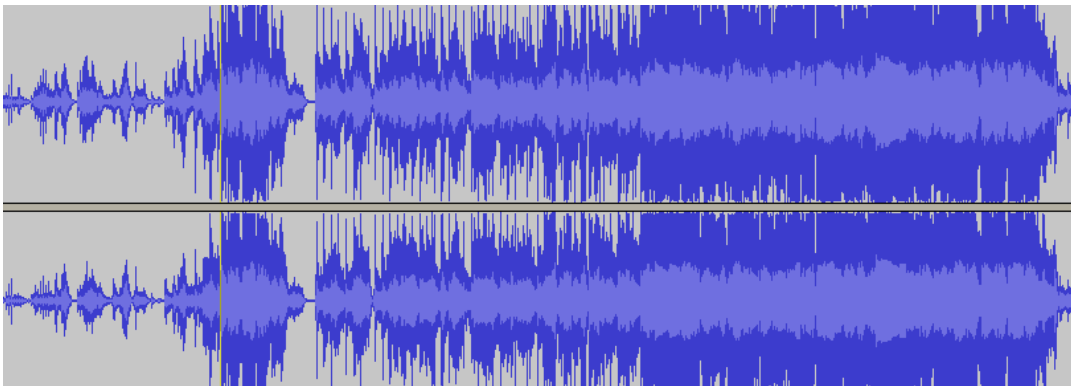


- Basic processing for the imported audio:
  - Basic splicing - delete, insert and copy
    - Delete:select an audio range - click the left mouse button and drag to the other edge of your selection and release, then click the Delete button to remove the selection.

      Before deleting the selection:



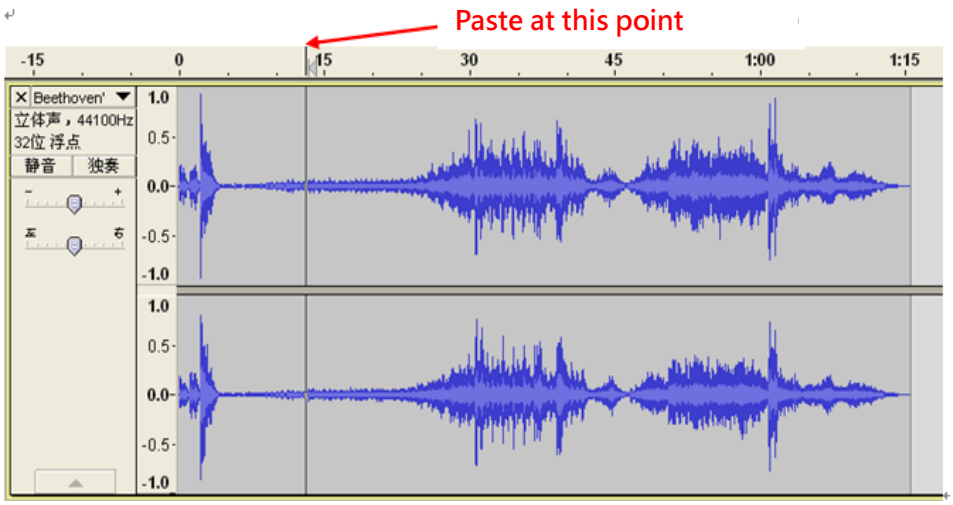
After deleting the selection:



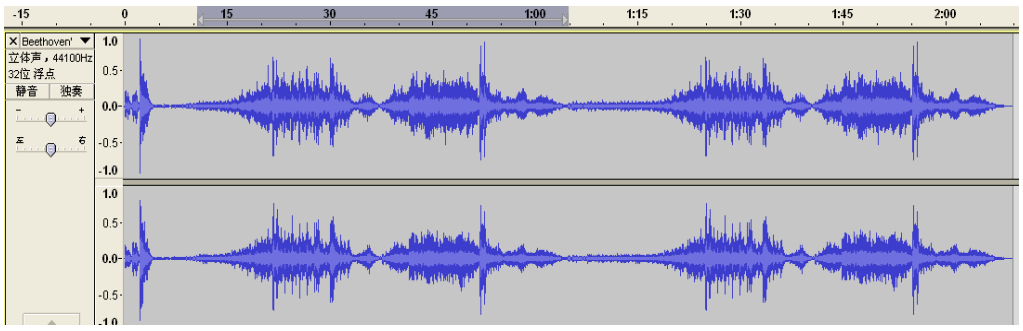
♦ Copy and paste:select a track range then press the Copy button  , click the mouse at the point where to insert the clip and then press the Paste button  .

Note: IF copying the audio track from another file first you need to open the file, File →Open. After this, two Audacity windows are shown, copy the selection, and paste  it at the point where you want it located in the first window.

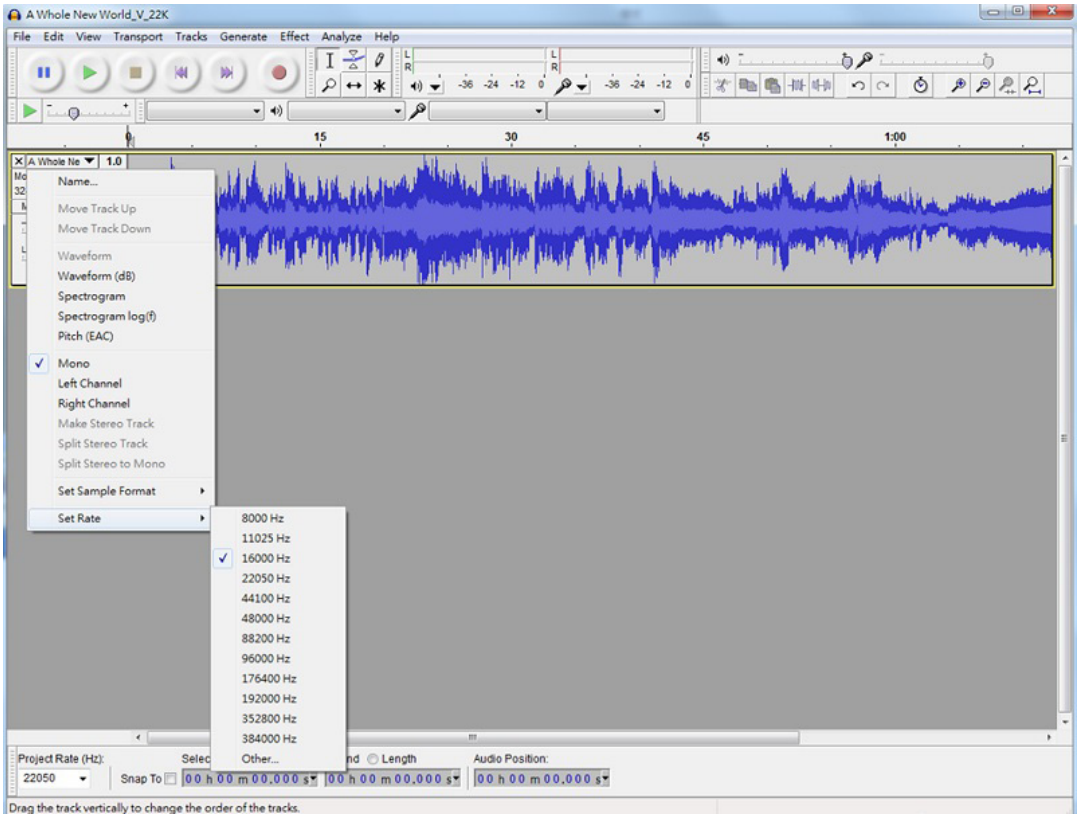
Before copy and paste:



After copy and paste:



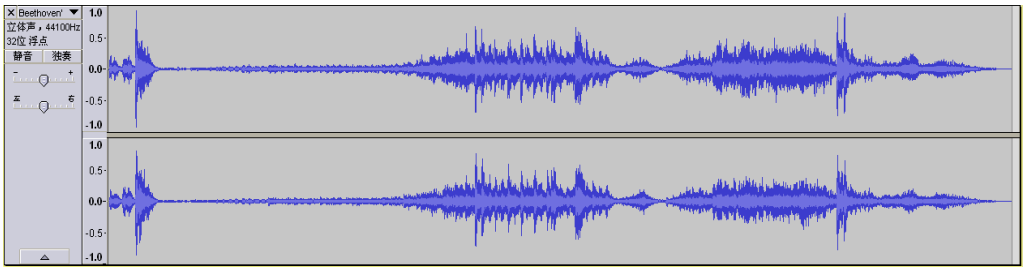
- Change the sampling rate of the voice source, as shown below:



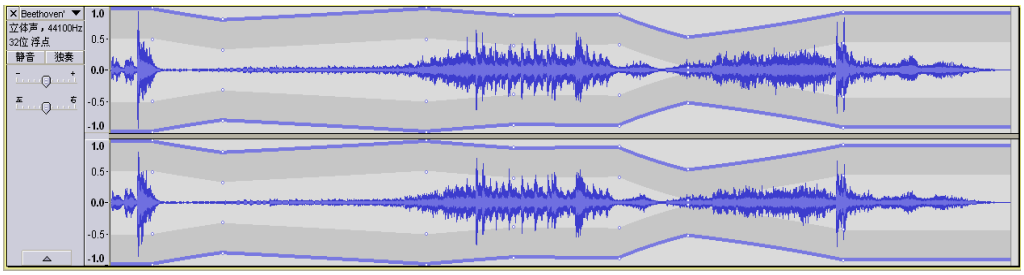
- Volume control - envelope

  After selecting the Envelope tool  , by clicking in the track you can see some “white points”. Then set the volume of that point by dragging one of its four vertically arranged “handles”.

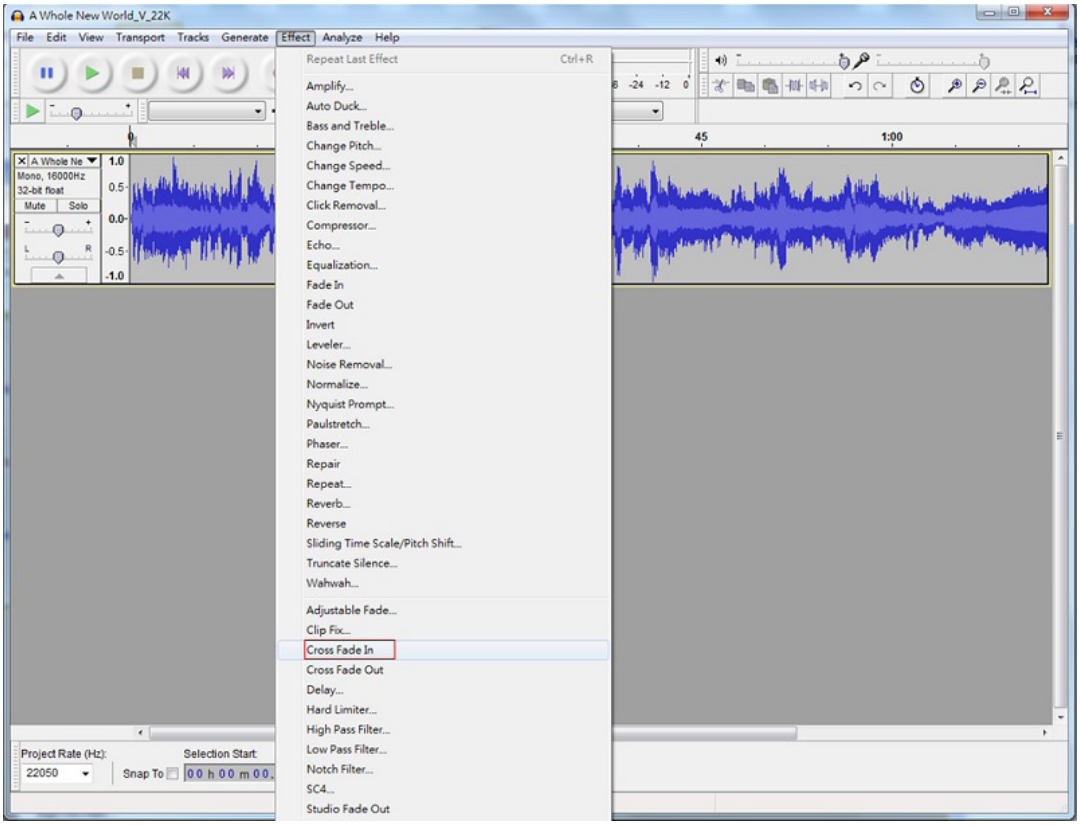
  Before changing volume:



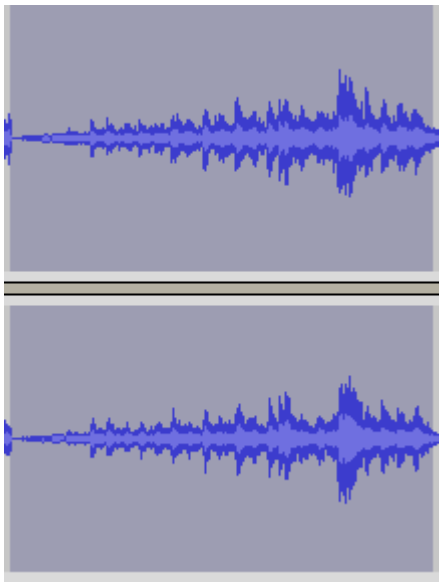
  After changing volume



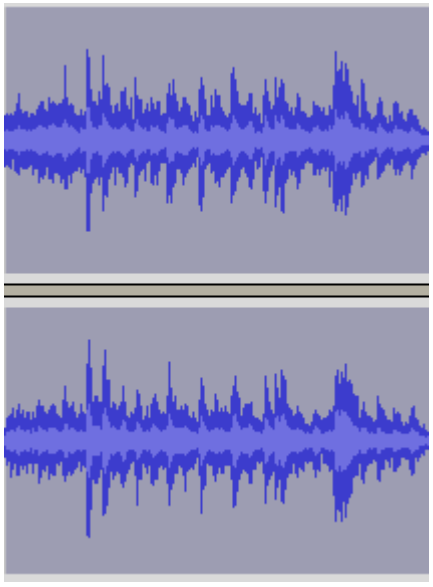
- Effects:
  Click "Effect" to choose the following effects:



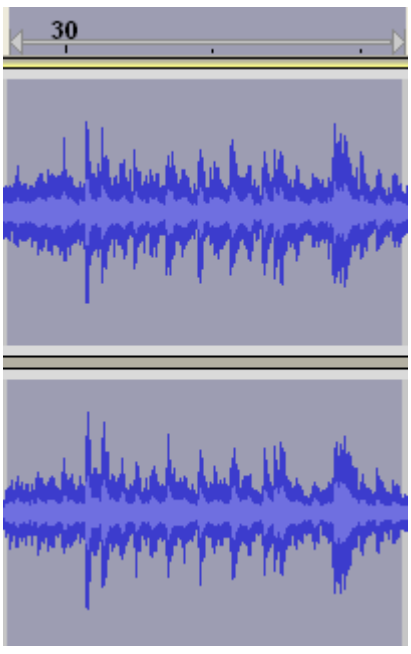
♦ Cross Fade In/Cross Fade Out function:



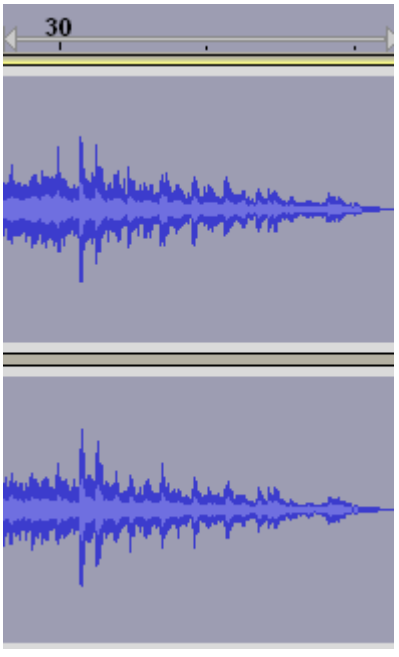
Before a Cross Fade In



After a Cross Fade In



Before a Cross Fade Out

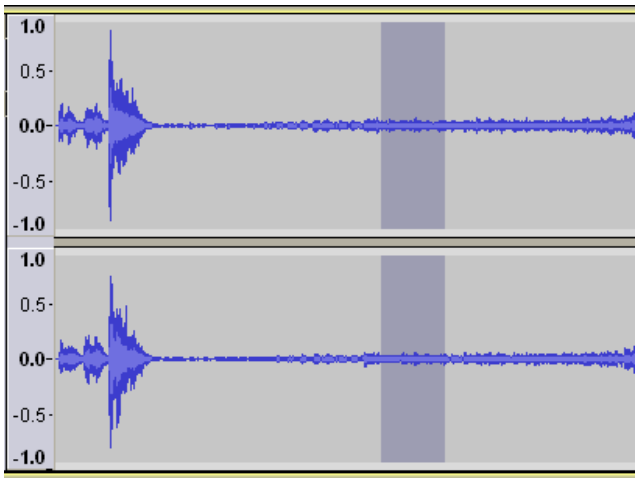


After a Cross Fade Out

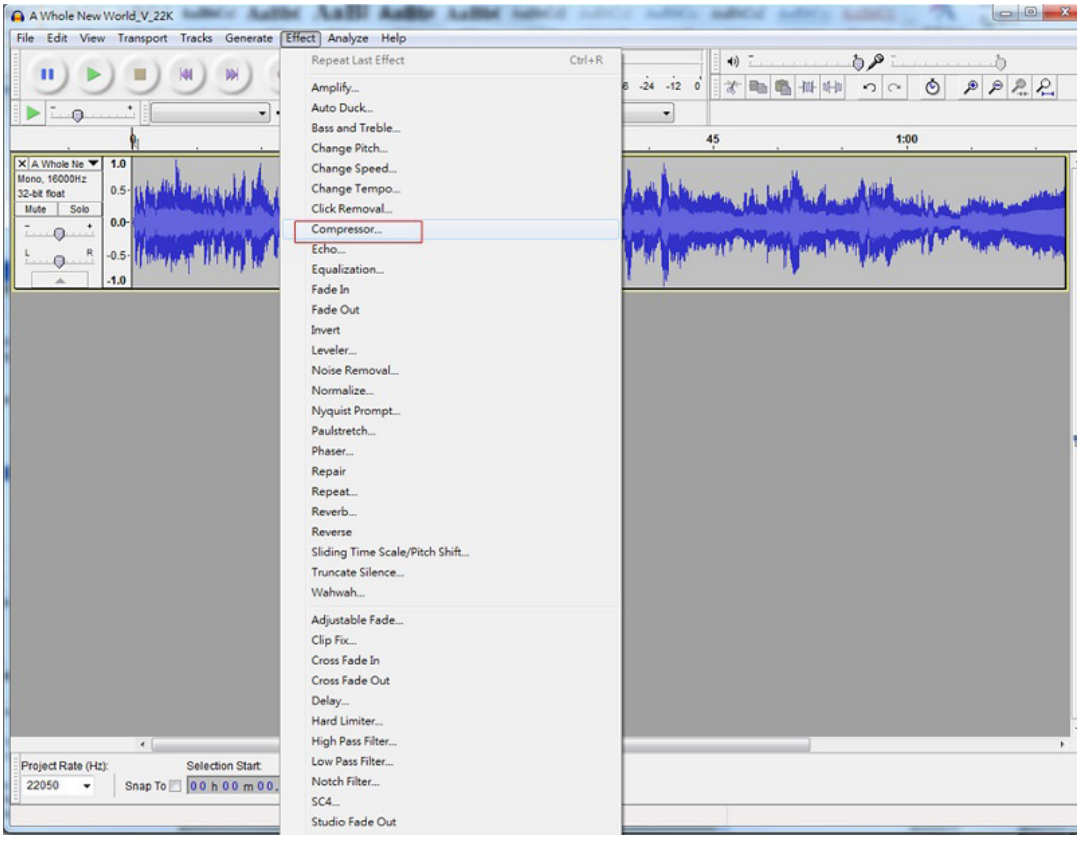
♦ Noise Removal

Noise Removal can reduce constant background sounds.

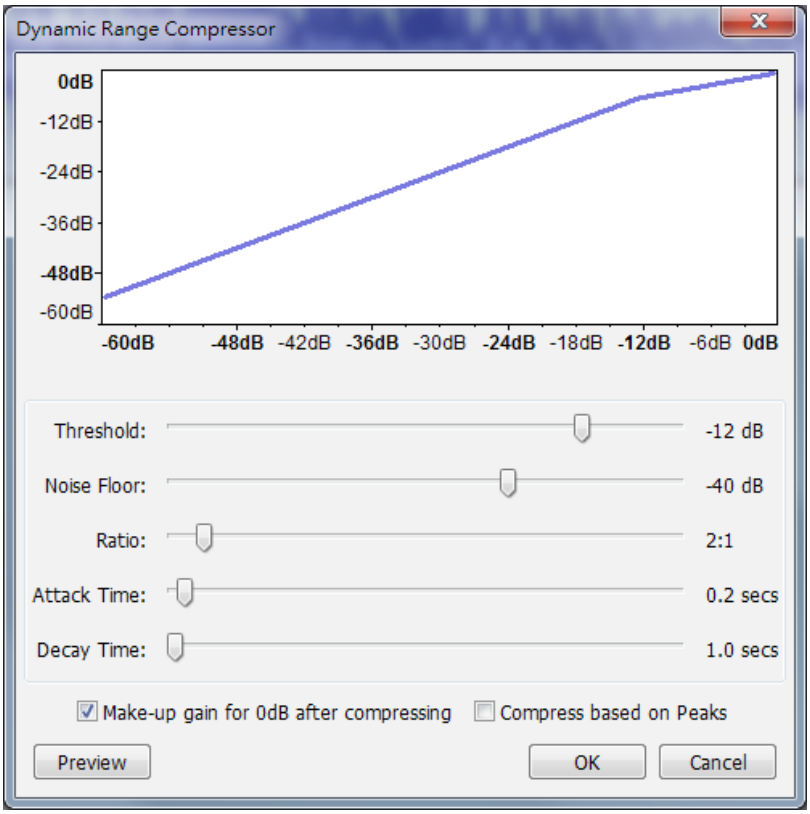
a. Select a track region - about 0.5s~2s long is ideal - which contains only noise to let Audacity know what to filter out.

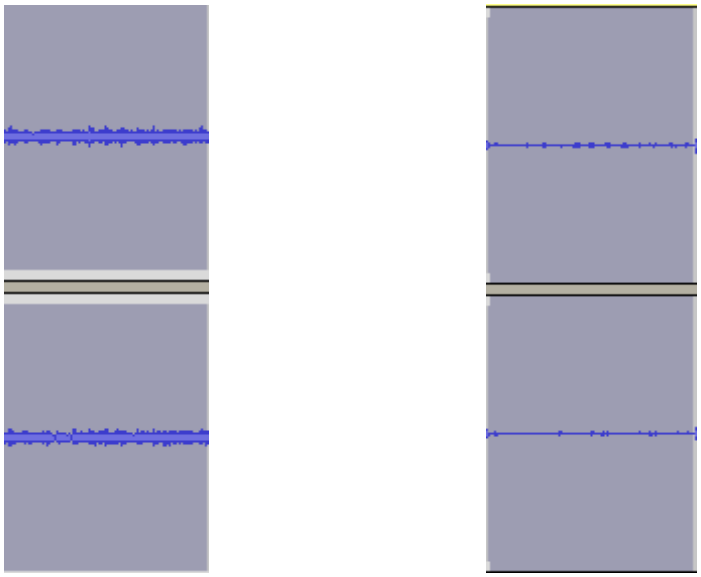


b. Click "Effect" --> "Noise Removal" :



c. After clicking "Noise Removal" and setting up some related parameters, click “OK” and the processed waveform can be seen:

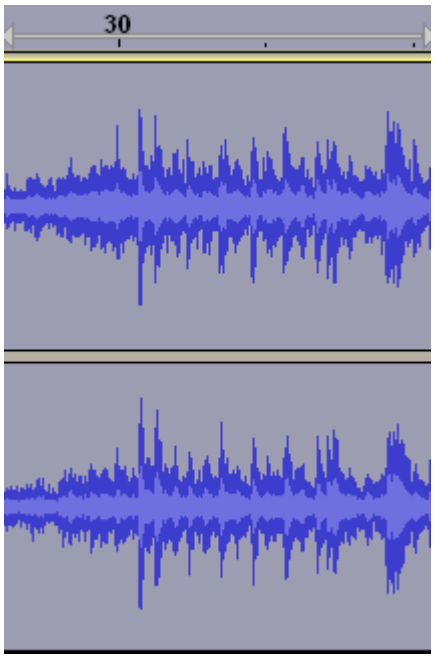




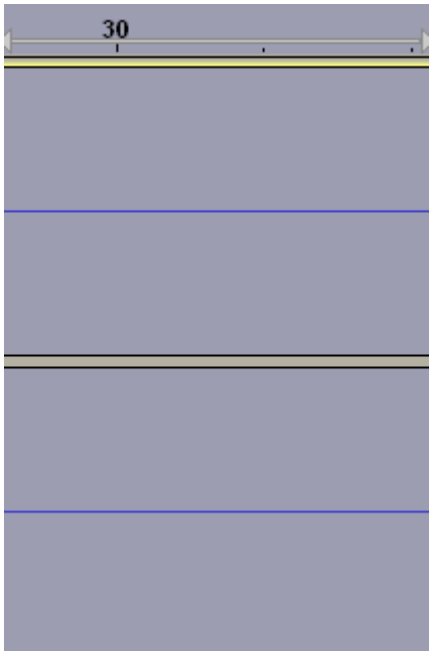
Before Noise Removal **After Noise Removal**

♦ Silence the selection:

Select the track region you want to silence then click the Mult Button  :



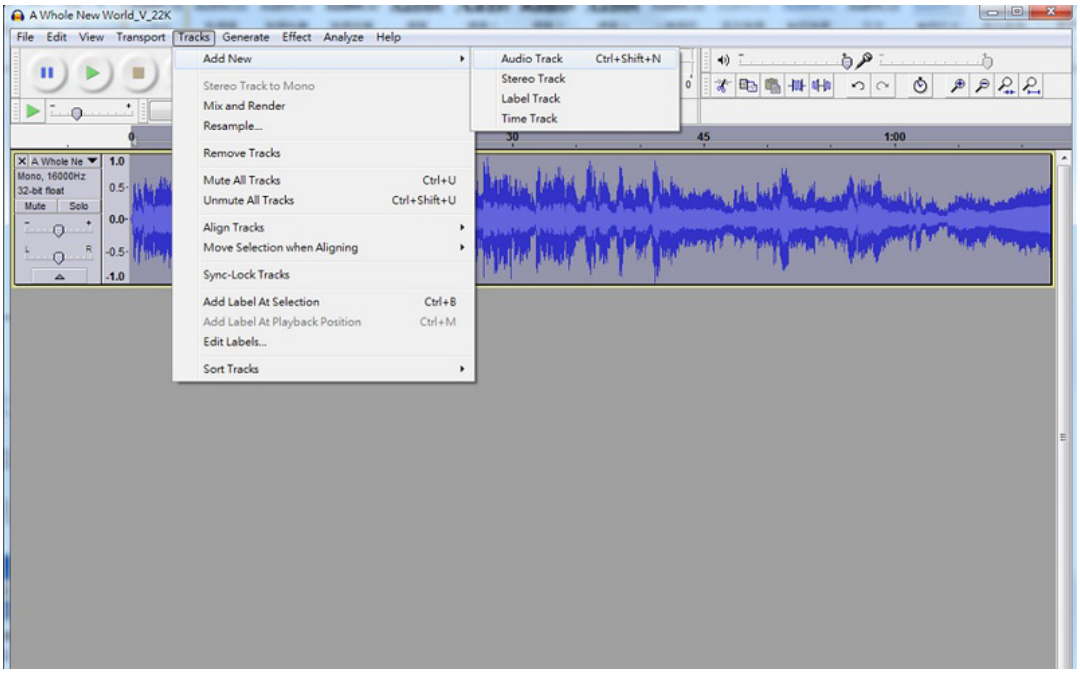
Before Mult



After Mult

♦ Mixing Audio Tracks:

Mixing refers to the process of combining multiple Audacity tracks into a single track. For example, mixing a voice with music to add a background musical effect. If you want to add another track, choose "Track" → "Add New" → "Audio Track" , and then paste the clip you need onto the new track.

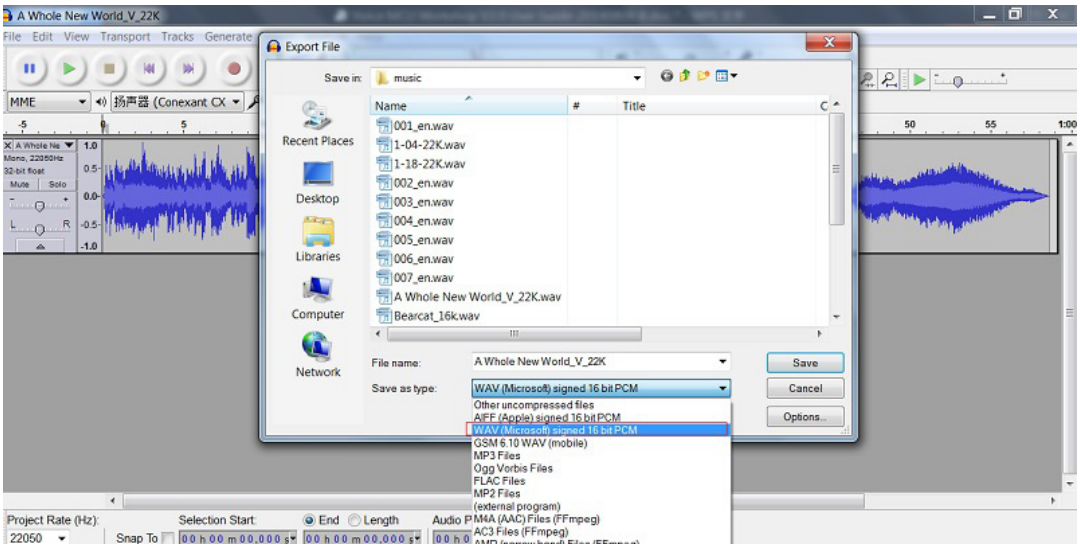


♦ Exporting Audio

Export a wav / aiff / mp3 / ogg file.

After completing the audio processing, Audacity can export the file in the above formats. (Note:The Voice platform only supports WAV audio format)

Choose"File"-> "Export" and then select the folder location and audio format.



5 Adobe Audition CS6 Brief Tutorial

Introduction

The Adobe Audition (formerly Syntrillium Cool Edit Pro) software is a complete multitrack recording studio for Windows-based PCs. Adobe purchased Cool Edit Pro from Syntrillium Software Company in May 2003 and then changed the name of Cool Edit Pro to “Adobe Audition”. Adobe Audition is a professional audio editing environment which offers advanced audio multi track, mixing, editing, controlling and effects processing capabilities. It can mix up to 128 tracks, edit individual audio files, create loops and import more than 45 DSP (digital signal processing) effects.

Adobe Audition provides a fully-integrated audio editing and mixing solution for music, video, radio, and sound design professionals with integrated multitrack and edit views, real-time effects, looping support, analysis tools, restoration features, and video support. Users benefit from real-time audio effects that allow them to hear changes and track EQ instantaneously. Flexible looping tools and thousands of high-quality royalty-free music loops are included to assist in soundtrack and music creation.

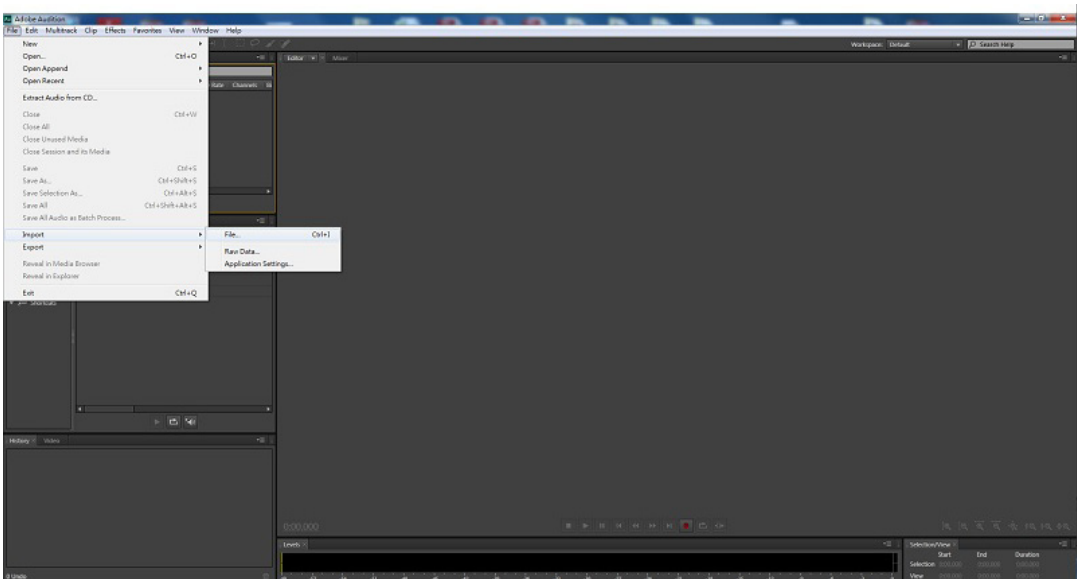
The intuitive, customizable interface allows users to dock and resize windows to create an efficient audio workspace. An organizer window uses tabs to track open files, effects and favorites. Batch processing tools streamline everyday tasks, such as matching the overall loudness of multiple files or converting them to a standard file format.

Adobe Audition provides quality audio for video projects by allowing users to edit, mix and add effects to AVI soundtracks while watching movie playback. Providing extensive support for industry-standard audio file formats including WAV, AIFF, MP3, MP3PRO and WMA, Adobe Audition can also handle files with bit depths of up to 32-bit and sample rates in excess of 192 kHz. This enables export to tape, CD, DVD or DVD-audio, with the highest-quality sound.

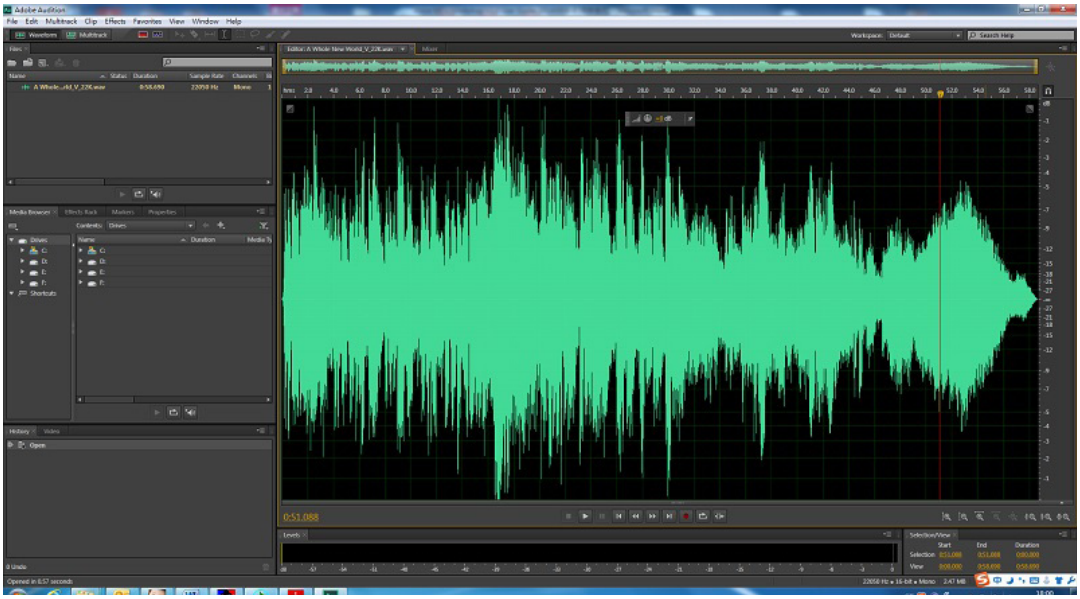
Quick Start

Edit a single audio file

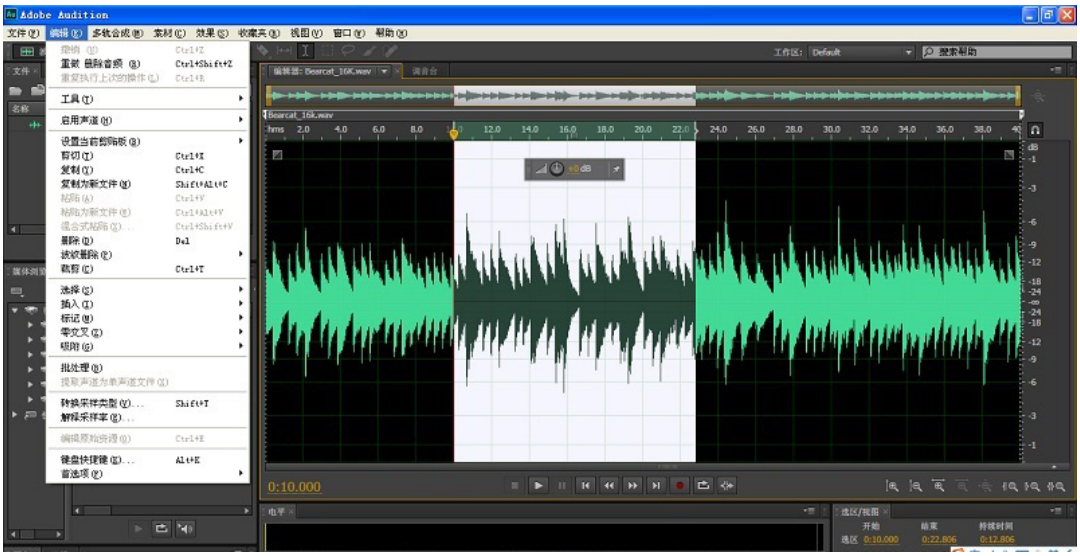
■ Open the software and choose "File" → "Import" → "File", as shown below:



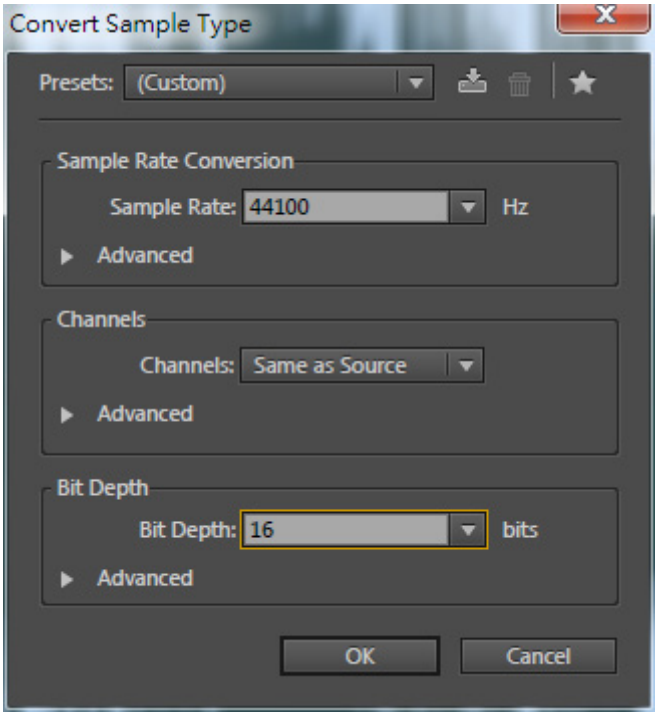
■ After importing an audio file:



■ Select the region you want to process and choose "Edit" → "Delete"/ "Cut"/ "Copy" / "Paste"… depending upon the required action.

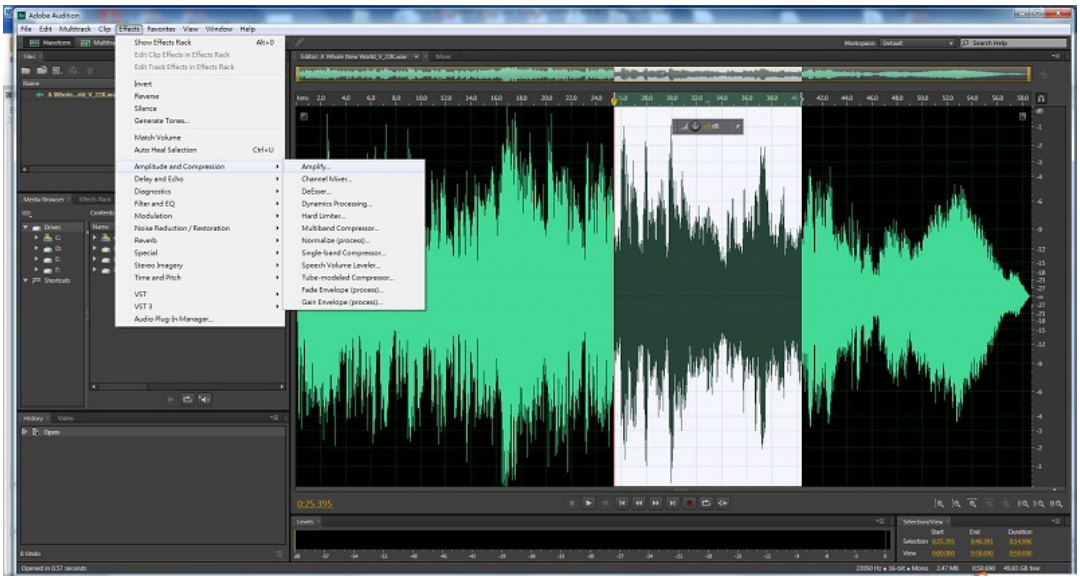


■ Choose "Edit" → "Convert Sample Type" and set the required Sample Rate and Bit Depth .

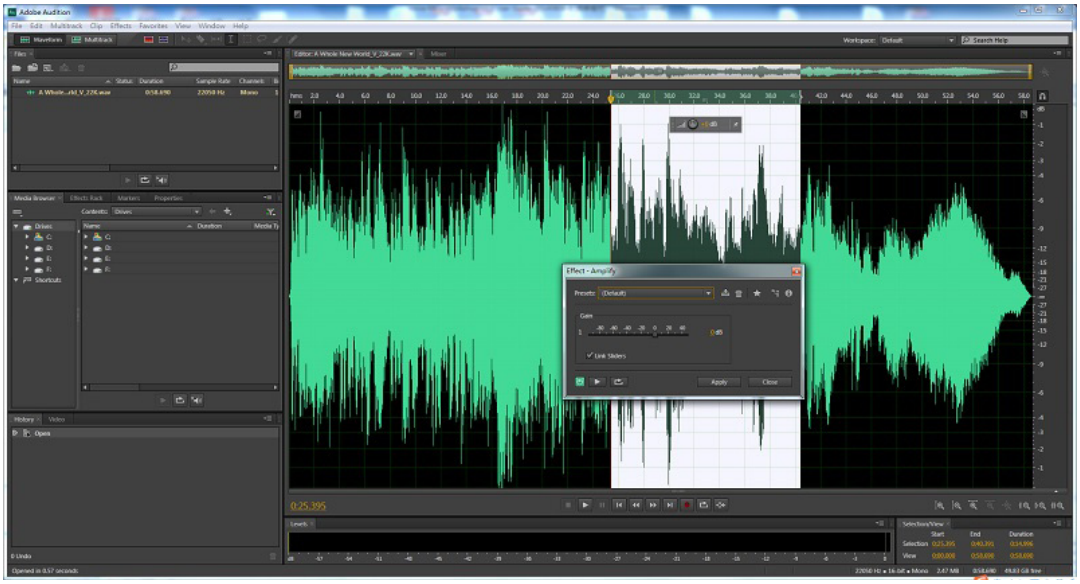


■ There are many effects can be added to the audio clip according to the user's specific desire. The following is an example of how to change the audio clip volume.

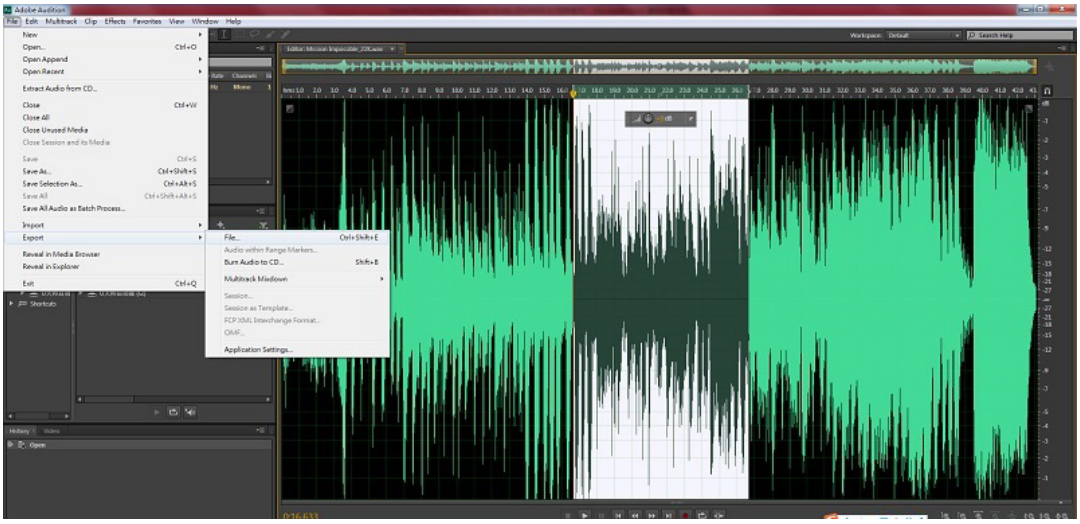
Select the audio you want to change and choose "Effects"-> "Amplitude And Compression" -> "Amplify".

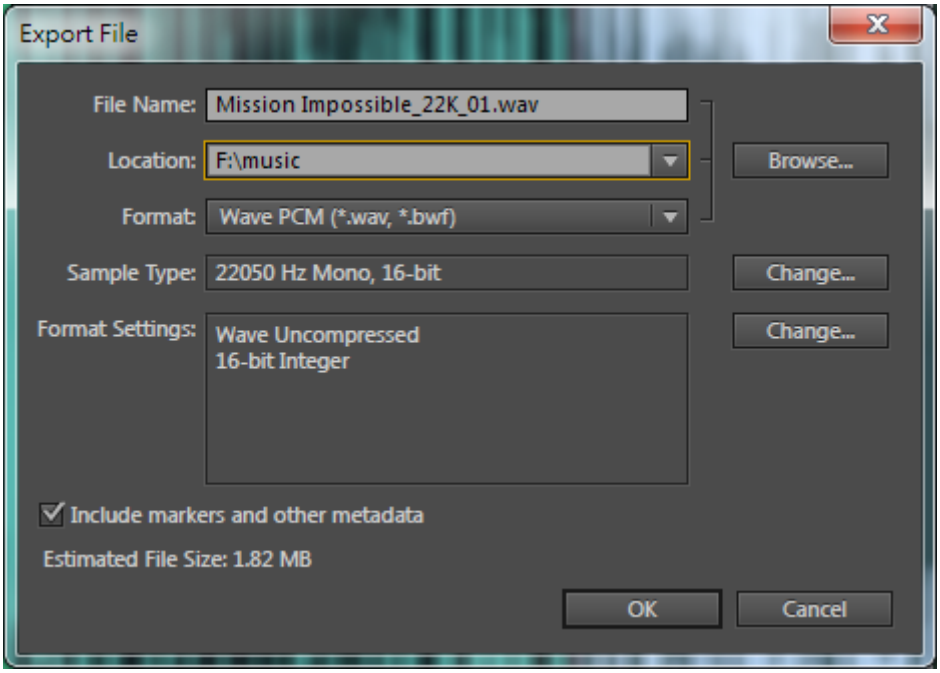


Click "Amplify" after which the following window will appear. Change as required and then click "Apply". The volume will then be changed.



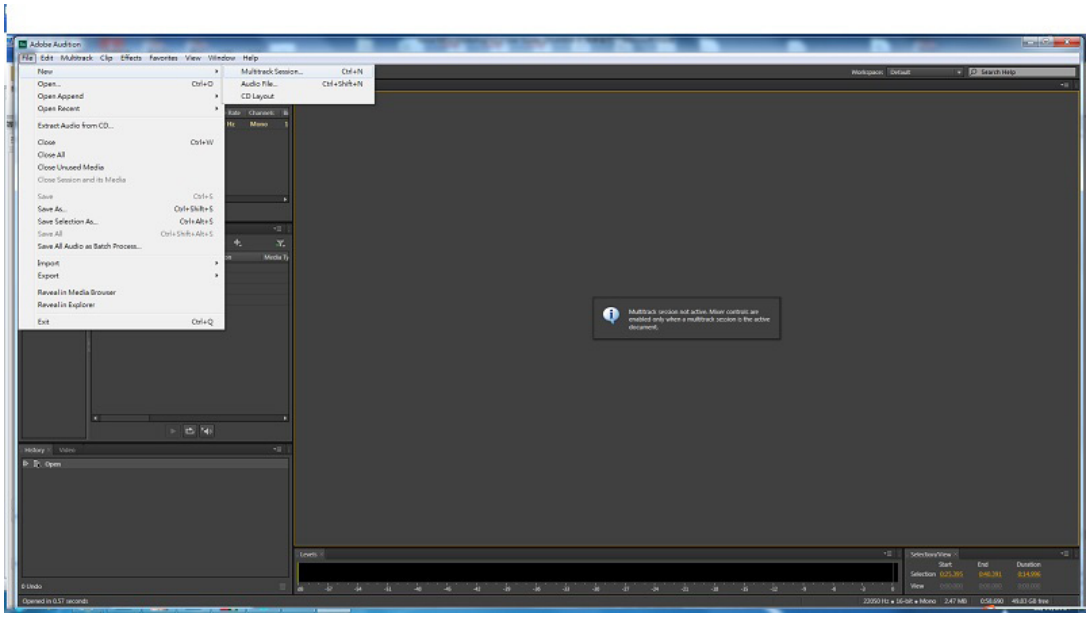
■ After the Editing is finished, click “File”, choose “Export” -> “File”. In the following Export file dialog box, you can view or adjust the saved file parameters. Finally, click “OK” when you have confirmed the setting options.

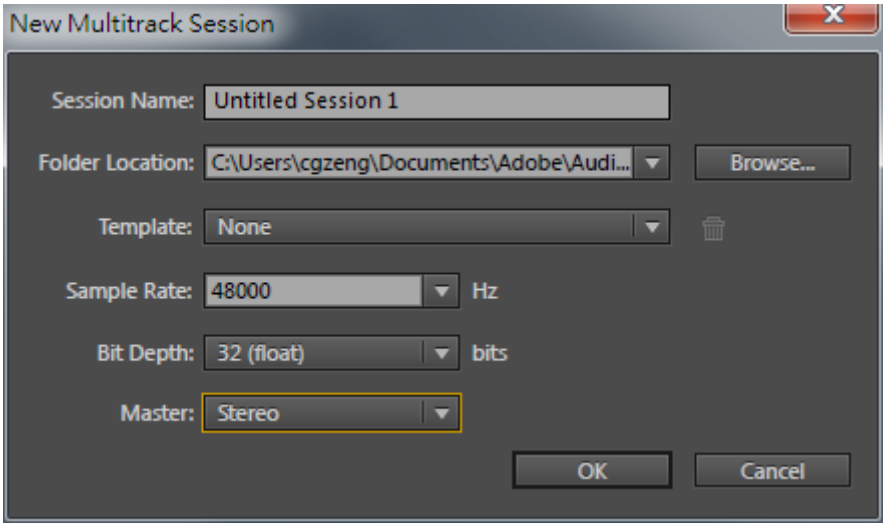




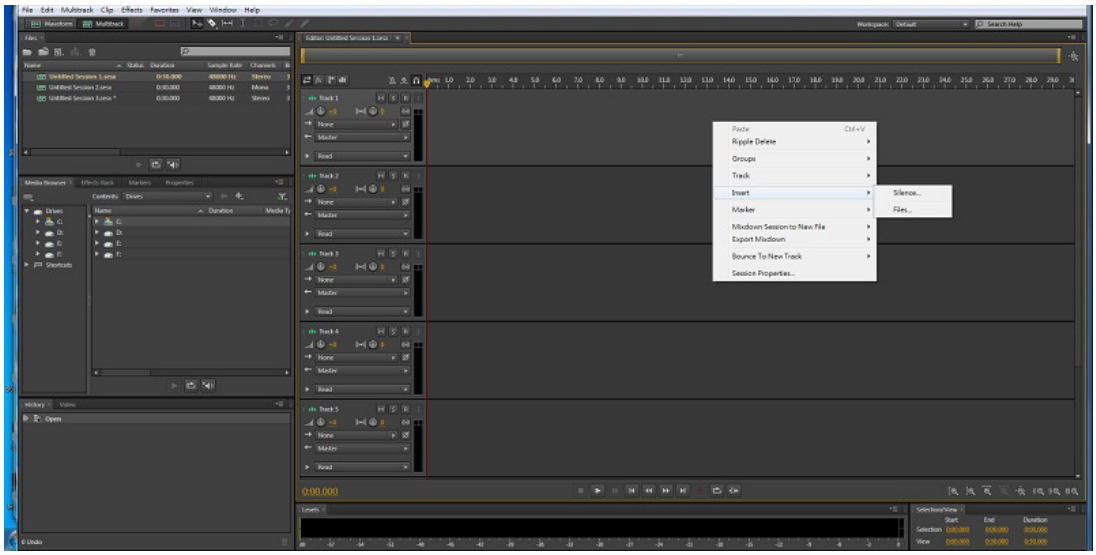
Edit Recording

■ Open the software and choose "File"-> “New” -> “Multitrack Session “, set the options for the new multitrack session, such as the Sample Rate and the Master, as shown below.

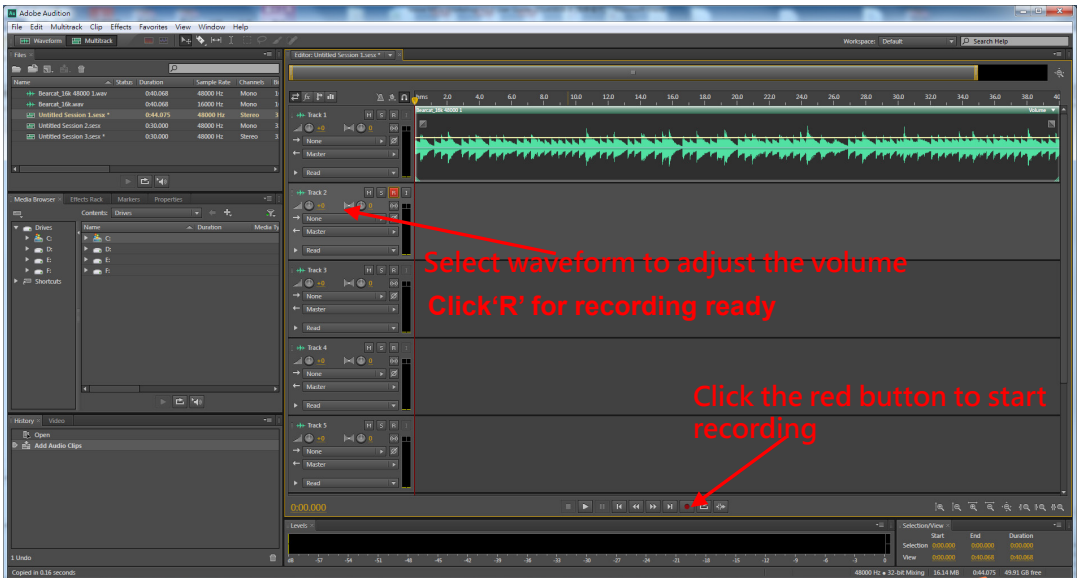




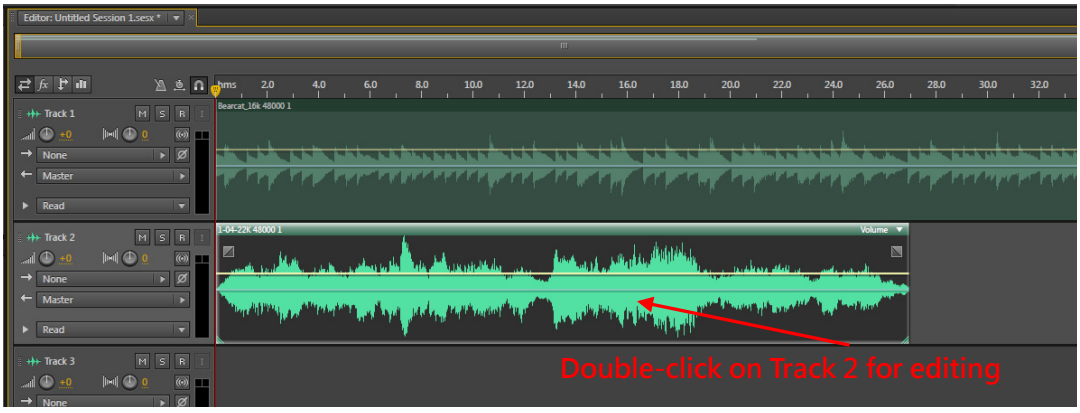
■ After creating the Session, we insert background music, to achieve the effect of mixing a vocal recording with background music. As shown in the figure below, we insert the background music in the specified empty track, here we use Track 1. Right-click on Track 1, and choose “Insert” -> “Files”. If you need insert a few pieces of music or sound effects, repeat the steps. However take care not to locate files where the music overlaps, unless these effects are required.



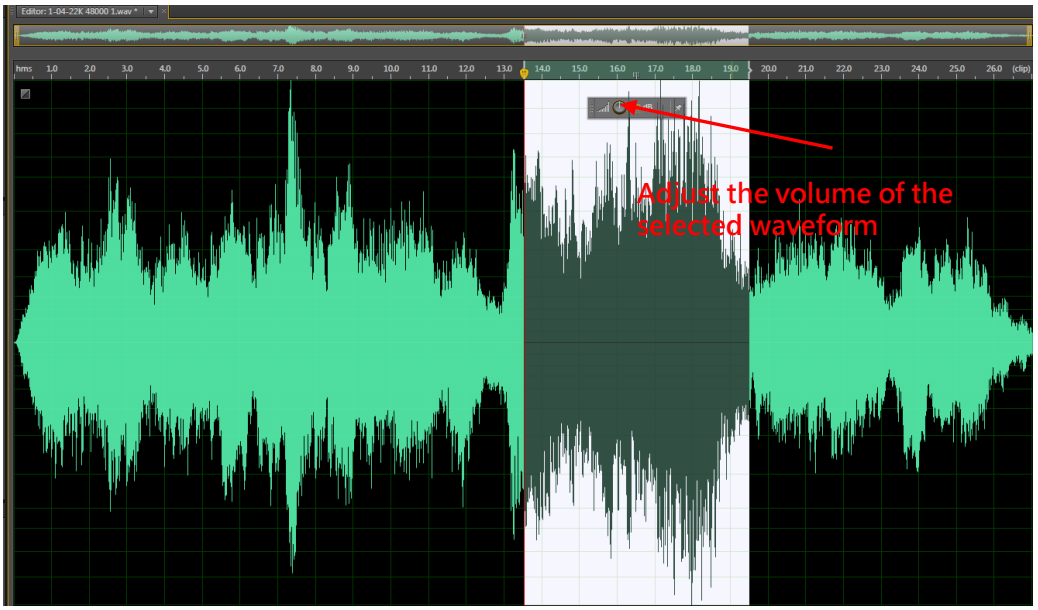
■ After inserting the background music, click the red Record button "R" on Track 2, which means we will record the voice onto Track 2. Of course, you can also record the voice onto other tracks, however in this example we use Track 2. As shown in the figure below, click the red Record button to start recording with the background music simultaneously.



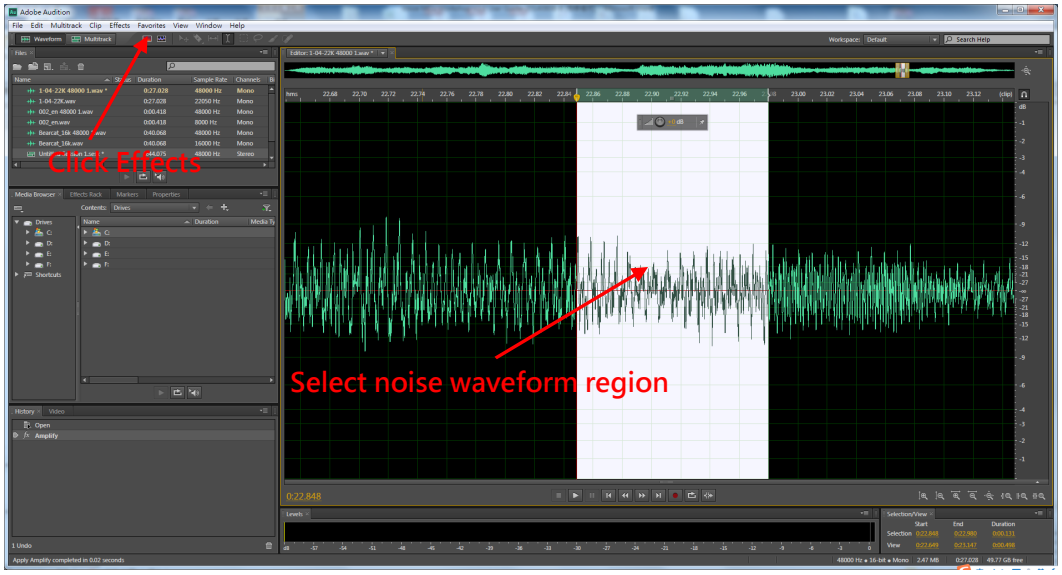
■ Now we have a recording so let's begin to edit the audio. As shown below, double-click the voice waveform in Track 2 and enter the single track edit view.



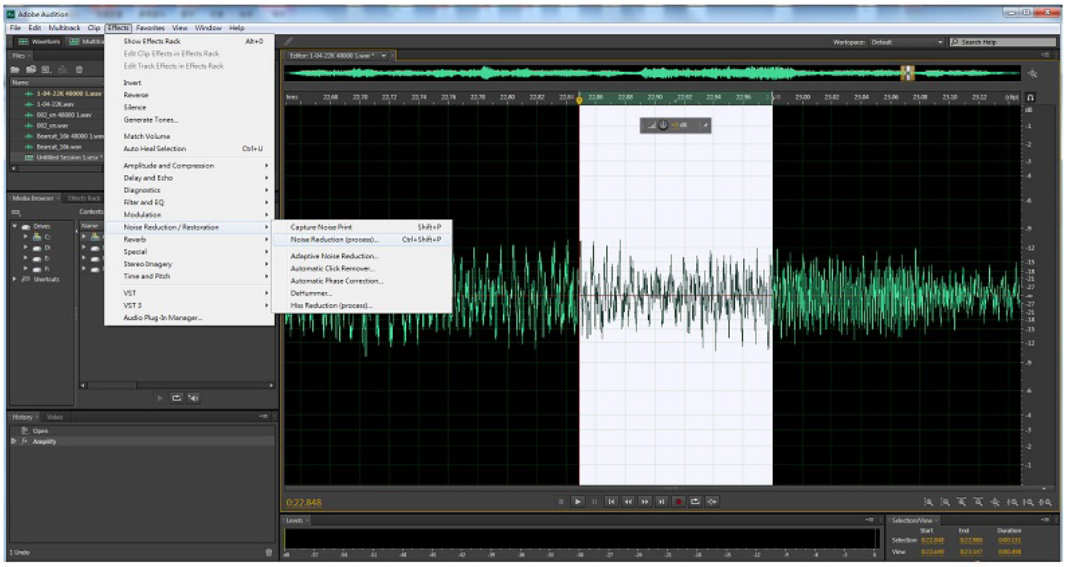
- In the single track editor, we can obtain the audio signal loudness by listening to it or by watching the waveform amplitude. We can select a specified range to adjust the volume, as shown below:



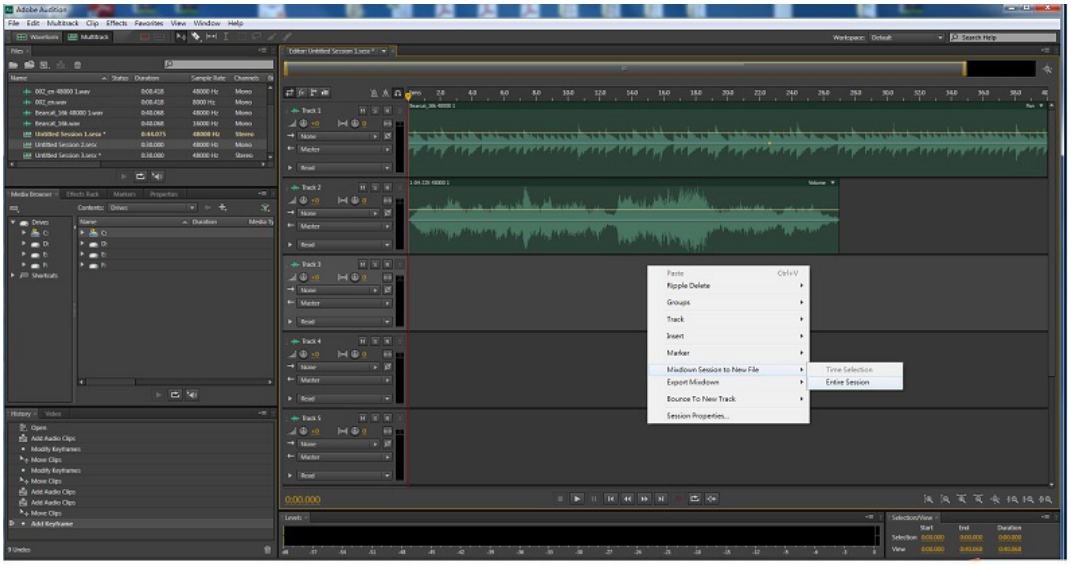
- Removing any noise is actually very simple. For some external environment noise like mouse clicks, coughs, we can select the noise waveform and directly delete it. For other internal environmental noise which can be power-line hum noise or others which may not be in the voice, first we select a range for this noise. After the selection, click “Effects”, as shown in the following figure:



■ Choose "Effects”→ "Noise Reduction/Restoration”→
"Noise Reduction (process)”



■ There are also many other voice process functions including reverb, echo, time and pitch manipulation effect, etc., which can all can be obtained in the “Effects” options. Now let’s learn how to create an entire composition of background music and voice. As shown below, right-click on any empty track, select “Mixdown Session to New File”→ “Entire Session” .



■ After you finish mixing a session, it switches to the single track editor, as shown below. Then choose “File”→ “Export” → “File”. In the following Export file dialog box, you can view or adjust the specific parameters about the saved file. Finally, click “OK” when you have confirmed the setting options.