

Voice MCU Workshop V2.3 User Guide

Revision: V1.50 Date: September 15, 2017



Table of Contents

1	Voice MCU Workshop Description	4
	Development platform Introduction and Software Installation	4
	Characteristics	
	System Requirements	
	System Configuration	5
	S/W Installation	6
	S/W Operation Quick Start	9
	Start the Voice MCU Workshop V2.3	
	Create a New Project	
	Open an existing project	
	Hardware Circuit	
	Evaluation Board Schematic Diagram	
	Using the Evaluation Board	
	Supported Flash series	
2	ASM and C library Instructions	51
	Call Voice library Functions using ASM	51
	Summary	
	Usage Instructions	51
	ASM Library Functions	
	ASM Program Example	
	Call Voice library Functions by C	72
	Summary	
	How to use	
	C Library Functions	
	C Program Example	
3	Voice Library Establishment and Emulator	
	HT66FV130	
	HT66FV140	
	HT66FV150	
	HT66FV160	
	BH67F2262	
	HT45F67	103
	HT45E65	105
	HT45E3W	108
		110
	H145F24A	115
	HT83F02	117
	HT86BX0	119



4 Audacity Quick Start	124
Audacity Summary	
Audacity Processing Flow	124
Quick Start	125
5 Adobe Audition CS6 Brief Tutorial	140
5 Adobe Audition CS6 Brief Tutorial Introduction	140 140
5 Adobe Audition CS6 Brief Tutorial Introduction Quick Start	
5 Adobe Audition CS6 Brief Tutorial Introduction Quick Start Edit a single audio file	140 140 141 141



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:

Ready to Install Setup is now ready to begin installing Voice N	1CU Workshop on you	ur computer.	
Click Install to continue with the installation, change any settings.	or click Back if you wa	ant to review or	
Additional tasks: Additional icons: Create a desktop icon			*
. ◄		4	Ŧ
	< Back	nstall C	ancel

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

Setup - Voice MCU Workshop	
Installing Please wait while Setup installs Voice MCU Workshop on your computer.	
Extracting files C:\\Voice MCU Workshop\BIN\audacity-win-2.0.5.exe	
	Cancel



😼 Setup - Voice MCU Works	shop
	Completing the Voice MCU Workshop Setup Wizard
	Finish

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:

Voice MCU Works	shop	 		
Project Name:		Voice Platform New Project	Load Project	Save Project
Basic Setting				
Voice Program				
Finish				



Create a New Project

Evaluation Mode:

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

Load Project Save Project

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

lew	Project	
	Project Name:	
	Untitled	
	Project Path:	
	C:	
	 Evaluation Mode (built-in code) ESK-66FV-100 	
	C Professional Mode (user code + library)	
	ОК	



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:

oject Name: voice_te	st_pcm8 Mode Selection	New Proje	ct (Load Pro	iject	Save P	roject
Basic Setting Finish Finish 							
		Available M	CU: Page	1 🗸			
	Speaker Voice	HT66FV140	H145F65	H145F67	H145F23A		
	Shiven Source	HT86B10	HT86B20	HT86B30	HT86B40	F	
		HT86B50	HT86B60	HT86B70	HT86B80	H	
		MCU Reso	urce:				



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:

roject Name: voice_test_pcm8	New Project	ct	Load Pro	ject	Save	Project
Assic Setting bice Program Finish Automatical and a setting of the setting of	Available Mi HT66FV140 HT66FV140 HT65F24A HT86B10 MCU Resou	СU: Раде НТ45F65 НТ45F3W НТ86B20 НТ86B60 игсе:	1 V HT45F67 HT83F02 HT86B30 HT86B70	НТ45F23А НТ96В03 НТ96В40 НТ96В80		



- 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:

Moice MCU Workshop [E:\v	voice_test_pcm8]	
Project Name: voice_te	est_pcm8	New Project Load Project Save Project
Basic Setting Voice Program Finish	Hode SelectionGraduation Mode (built-in code)Thirty on may choose available functions first, then doing advance settings on the right side!Intry to may choose available functions first, then doing advance settings on the right side!Intry to may choose available functions first, then doing advance 	Interface: SPI Flash Size: 1096K x 8bits 1036K x 8bits 1036K x 8bits 1046K x 8bits 1036K x 8bits 1046K x 8bits 124 x 8bits 124 x 8bits 124 x 8bits 125 x 8bits 146 x 8bits 126 x 8bits



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:

oject Name: voice_te	st_pcm8	New Project Load Project Save Project
Basic Setting Dice Program	Adde Selection• Evaluation Mode (built-in code)Hint: You may choose available functions first, then doing advance settings on the right side!Image: Additional strength of the settings 	Available MCU: Page 1 HT66FV140 HT45F65 HT66FV140 HT66F0 HT66FV140 HT66F0 HT66FV140 HT66F0 HT66FV140 HT66F0 HT66F



- Voice Source Function:
 - (1) Click the "Voice Source" button to load the function to the MCU.

roject Name:	voice_tes	st_pcm8	New Proje	ect	Load Pro	ject	Save F	Project
asic Setting ice Program Finish		Mode SelectionIf Evaluation Mode (built-in code)Hint: You may choose available functions first, then doing advance settings on the right side!Image: Available FunctionImage: Available	Available M HT66FV140 HT45F24A HT86B10 HT86B50 MCU Reso	CU: Page HT45F65 HT45F65 HT45F3W HT86B20 HT86B60 urce:	1 ▼ НТ45F67 НТ85F02 НТ86B30 НТ86B70	НТ45F23А НТ86В03 НТ86В40 НТ86В80		
							戶	-



- _ 0 X Voice MCU Workshop [C:\Users\Administrator\Desktop\voice_test_pcm8 Project Nam voice_test_pcm8 New Project Load Project Save Project Mode Selection Evaluation Mode (built-in code) Hint: You may choose Basic available functions first, then Setting doing advance settings on Available Function Voice Program Waveform Editor Reset ALL . Add ⊡: Remove Finish Compression Mode Driginal Voice Size ncoded Voice Siz Nickname +/-Play File Name Ð X D Open 🕒 🕞 🗢 📕 🕨 music ✓ ← Search music Q New folder 0 Organize • 💷 🔹 🔲 Title Contributing artists Album Name # + Favorites 1-18-22K.wav E Desktop 1002_en.wav S Recent Places 1003_en.wav L Creative Cloud 1004_en.wav bownloads 1005_en.wav 1006_en.wav Eibraries 📆 007_en.wav A Whole New Worl... 🜏 Homegroup Bearcat_16k.wav BUBUGAO.wav Computer EIGH.WAV FIVE.WAV Network Hakuna Matata_22K... File name: Bearcat_16k.wav Voice Files (*.wav) -• Cancel Open
- 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 <u>Library Establishment</u> <u>Information section</u>.



• 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.

Voice MCU Workshop [f	E:(voice_test_pcm8] _test_pcm8		New Project	Load Project	ct Save P	roject
Basic Setting foice Program	Mode Selection C Evaluation Mode (built-in code) Hint: You may choose available functions first, then doing advar settings on the right side! Available Function C Add C Remove			Waveform Editor	Reset ALL	
	+/- Play File Name Image: Constraint of the state of th	Nickname Con LBearcat_16K HT-u	PCM8	Original Voice Size 1252K Bytes	Encoded Voice Size 629K Bytes	
	Total Memory Size: 4096K Bytes	Memory Size Used	629K Bytes	Memory Size L	eft: 3467K Bytes	



• 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.

Voice MCU Works	hop [C:\Users\Administrator\Desktop\voice_test	_pcm8]		x
Project Name voi	ce_test_pcm8	New Project	Load Project Save Project	t
Basic Setting Voice Program Finish	Mode Selection • Evaluation Mode (built-in code) Hint: You may choose available functions first, then doing advance settings on • Available Function • Available Function • Source Setting • Nick • Node: • HT-uPCM8 • HT-ADPCM4 • Nickname: • HT-PCM16	PA1 1 PC0 2 PA4 3 PA3 4 PC1 5 Pompression Mode prigin ministrator\Desktop\music\Bearca	At 16k 6x 8bits 15 PB1	*
Sour	ce Setting	0		
	Source: C:\Users\Adm Type: Wave	iinistrator\Desktop\mu	isic∖Bearcat_16k	
	Mode: HT-ADPCM4	•		
	Nickname: Bearcat_16k			
	You can choo file	oK	le for your voice	



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.

Mode Selection ⁹ Evaluation Mode (built-in code) ⁹ Pal- ¹ Pal-	roject Name: vo	ice_test_	pcm8			New Proje	ect	Load Pro	iject	Sa	ve Projec
Speaker Driven Voice Source Tto6FV140 HT45F55 HT45F57 HT45F23A 19 AVD0_PA State Cc6 HT45F24A HT45F3W HT85F02 HT86B03 18 AVS5_PA P80 12 HT86B50 HT86B60 HT86B70 HT86B80 SP OCDSCK 14 ROM: 4K x 16bits RAM: 256 x 8bits 15 PB1 D/A: 16bits x 1 PVM: 0 D/A: 16bits x 1 PVM: 0 15 PB1	Basic Setting /oice Program Finish		Mode S C Evaluation N (built-in code Hint: You may ch functions first, th settings on the ri Available Key Key	election lode e) oose available en doing advance ght side! Function External Flash	PA1 1 PC0 2 PA4 3 PA3 4 PC1 5 PC2 6 PC3 7 SDOA PC4 SCKA PC5	Available M	ICU: Page			28 27 26 25 24	PA5 PA6 PA7 VSS VOD BIAS AUD AUD AUD SP+
Source Proprint <			Speaker	Voice	SDIA PC6	HT66FV140	HT45F65	HT45F67	HT45F23A	19	AVDD_PA
PB0 12 HT86B50 HT86B50 HT86B70 HT86B80 OCDSCK 13 ROM: 4K x 16bits RAM: 256 x 8bits 16 PB2 OCDSDA 14 Package: 28SOP-A Timer: 10bits x 3 15 PB1 D/A: 16bits x 1 PWM: 0			Driven	Source	SCSAB PC7	HT86B10	HT86B20	HT86B30	HT86B40	18	AVSS_PA
OCDSCK 13 OCDSCA 14 MCU Resource: 16 PB2 ROM: 4K x 16bits RAM: 256 x 8bits 16 PB2 Package: 28SOP-A Timer: 10bits x 3 D/A: 16bits x 1 PWM: 0					P80 12	HT86B50	HT86B60	HT86870	HT86B80	F	SP.
OCDSCK 13 ROM: 4K x 16bits RAM: 256 x 8bits 16 P82 OCDSDA 14 Package: [28SOP-A Timer: 10bits x 3 15 P81 D/A: 16bits x 1 PWM: 0					100 12	MCU Reso	urce:			H	
OCDSDA 14 Package: 28SOP-A Timer: 10bits x 3 15 PB1 D/A: 16bits x 1 PWM: 0					OCDSCK 13	ROM: 4H	< x 16bits	RAM: 2	56 x 8bits	16	PB2
D/A: 16bits x 1 PVWI: 0					OCDSDA 14	Package: 28	SOP-A 🗸	Timer: 1	10bits x 3	15	PB1
000 10 0						D/A: 16	ibits x 1	PW	/M: 0		





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.

Project Name:	voice_tes	st_pcm8	I	New Proje	ct	Load Pro	ject	Sa	ve Project
Basic Settin Voice Progra Finish	n	Mode SelectionIf Evaluation Mode (built-in code)Unit: You may choose available functions first, then doing advance setings on the right side!Variable FunctionImage: Second colspan="2">Image: Second colspan="2">Evaluation for the right side!Variable FunctionImage: Second colspan="2">Image: Second colspan="2">Evaluation for the right side!Image: Second colspan="2">Image: Second colspan="2" Image: Second colspan="2"	Key7 PA1 PC0 2 Key4 PA4 Key5 PA3 PC1 5 PC2 6 PC3 7 SD0A PC4 SCKA PC5 SDIA PC6 SCSAB PC7 PB0 12 OCDSDA 14	Available M HT66FV140 HT66FV140 HT86B10 HT86B50 MCU Reso ROM: 4k Package: 28 D/A: 16 GPI0	CU: Page HT45F65 HT45F65 HT45F65 HT86B20 HT86B20 HT86B20 Urce: (x 16bits S0P-A v bits x 1 D: 19	1 C HT45F67 HT86B30 HT86B70 RAM: 2: Timer: 1 PW Res	HT45F23A HT86B03 HT86B40 HT86B80 HT86B80 S66 x 8bits S66 x 8bits S66 x 8bits MX: 0 erved	PA5 PA6 PA7 25 24 19 18 18 16 PB1	Key3 Key1 VSS VDD BIAS AUD AUD AUD_N SP+ AVDD_PA AVDD_PA PB2 Key6





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.

						[
oject Name:	voice_test	_pcm8		New Project	Load Project	Save Project
		Voice Program List				
		Trigger Source	Trigger Source Name	Function		
		()				Ē
asic Setting						
eo Drogran						
ce i rogran						
12/02/2017						
Finish						
	_					
						*
				4		•
		Available Trigger Source	Available Function			
		Key	7 Play Voice	Pause	Resume	Stop
		Command	30 Play Prev.	Play Next	+ Volume	- Volume
					1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1	



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.

Voice MCU Works Project Name: v	hop [E:\voice_test_pcm8]
Basic Setting Voice Program Finish	Voice Program List Trigger Source Trigger Source Name Key 1 Play Voice Image: Source Name Image: Source Name Image: Source Nam Image: Source Name <
	Available Trigger Source Available Function
	Key x 6 Command x 30 Play Voice Pause Resume Stop Play Next + Volume - Volume



- 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

() Voice MCU Work	sshop [S:\Voice平台\平台勤赠\Voice_V223_20161004_TEST project\TEST_P]
Project Name:	TEST_P New Project Load Project Save Project Save New Project
Basic Sotting	
Dasic Octiling	
Voice Program	
Finish	Programming
	SPI Flash data verification
	Conly Generate Voice File(.dat
HOLTEK	



Programming finish

	1011/ TT /11TT /125301/ 1 1/000 001/1001/ TECT 1 1/17CT 01	
Voice MCU Workshop	pp[S:\Voice平台\平台軟體\Voice_V223_20161004_IESTproject\IEST_P]	
Project Name: TES	ST_P New Project Load Project Save Project Save New	Project
Basic Setting Voice Program Finish	Process Finish! You can test your product now! SPI Flash data verification Only Generate Voice File(.dat	



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

Yoice MCU Workshop	
Project Name:	New Project Load Project Save Project

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

New Project	
Project Name:	
ht66FV140_pcm8	
Project Path:	
E:	
C Evaluation Mode (built-in code)	
Professional Mode (user code + library)	
ОК	



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.

Project Name:	ht66FV140_pcm8	New Project Load Project	Save Project
Basic Settin Voice Progra Finish	Mode Selection •• Professional Mode (user code + library) Hint: You may choose available functions first, then doing advance settinos on the right side! Varilable Function Image: Neg Professional Mode (Section 2014) Image: Neg Professional Mode (Neg Profession 2014) Image:	Available MCU Page 1 HT66FV140 HT45F65 HT45F23A HT45F65 HT86B10 HT86B20 HT86B50 HT86B30 HT86B50 HT86B60 MCU Resource:	



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:

Project Name: ht66FV140_pcm8 Mode Selection	New Project Load Project Save Project
 Professional Mode (user code + library) Basic Setting Available Function functions first, then doing advance settings on the right side! Available Function (Key) External Flash Speaker Driven 	Interface: SPI Flash Size: 40956K x 8bits 16384K x 8bits 16384K x 8bits 192K x 8bits 1024K x 8bits 1024K x 8bits 1024K x 8bits 1024K x 8bits 128K x 8bits 128K x 8bits 1480 - 67 1456F04 1145F35 14786B10 1145F35 14786B10 1186B20 1786B50 1786B60 1786B50 1786B60 1786B50 1786B60 1786B50 1786B70 1786B50 1786B70 1786B50 1786B70 1786B50 1786B70 1786B70 1786B70 1786B70 1786B70 1786B70 1786B70 1786B70 1786B70 1786B70 1786B70 1786B70 1786B70



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:

Project Name: ht66FV1	40_pcm8	New Proje	ct	Load Pro	ject	Save	Project
Basic Setting /oice Program Finish	Mode SelectionIntroductions first, then doing advance settings on the right side!Valiable FunctionImage: Settings on the right side!Image: Settings on the	Available M HT66FV140 HT766FV140 HT786B10 HT86B50 MCU Resou	CU: Page HT45F65 HT45F65 HT86B60 Jurce:	0 output 1 output 1 output HT45F67 HT83F02 HT86B30 HT86B30	HT45F23A HT86E03 HT86E80 HT86E80		



- Voice Source Function
 - ① Click the "Voice Source" button to load/remove the function to/from the MCU.

Mode Selection • Professional Mode (user code + library) Hint: You may choose available functions first, then doing advance setings on the right side! ice Program Finish Finish Finish Speaker Driven				<u>آلگر</u>	
	vailable MCU: F66FV140 HT IT45F24A HT	U: Page 1 HT45F65 HT45F3W	1 V HT45F67 HT83F02	HT45F23A HT86B03	
HT	-TT86B10 H1	HT86B20	HT86B30	HT86B40	
	4T86B50 H1	HT86B60	HT86B70	HT86B80	
	CU Resource	ce:			F
					H
					H
					μ



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

Voice MCU Workshop	[C:\Users\Administrator\Desktop\	voice test pcm81			_ 🗆 X
Project Nam voice_	test_pcm8	rolecter competitio)	New Project	Load Project	Save Project
Basic Setting Voice Program	Mode Selection • Evaluation Mode (built-in code) Hint: You may choose available functions first, t doing advance settings c · Available Function	then			
Finish	Add Remove + / - Play File Name	Nickname	Compression ModeDr	iginal Voice Size ncode	ed Voice Siz
Den 🕑					×
C . musi	c		1	 ✓ 4 Sea 	rch music 🔎
Organize - New f	folder				· · •
Favorites Desktop Recent Places Creative Cloud Downloads Libraries	Name # 1:18-22K.wav # 0:02_en.wav # 0:03_en.wav # 0:04_en.wav # 0:05_en.wav # 0:06_en.wav #	Title	Contrib	uting artists Album	E
🔧 Homegroup	1007_en.wav 2010 A Whole New Worl 2010 Bearcat_16k.wav				
ika Computer	BUBUGAO.wav				
Ketwork	FIVE.WAV				-
File	name: Bearcat_16k.wav			▼ Voice Fil	es (*.wav) 🔹



• 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.

Project Name:	ht66FV	/140_pc	m8			(New Proje	ct	Load Pro	ject	Save	Project
Basic Setting		Hi	Prof (use nt: You nctions ettings	Mode Selection fessional Mode er code + library) u may choose availal s first, then doing ad on the right side!	ble vance							
oice Program		(•): Ad	bt	Eila Name	Misland		managing Ma		vavelorm Eul		Veise Cir	
olce Flogram			Piay	Bearcat 16K way	0 Bearcat 16K	HT-u	PCM8	1252K	Bytes	629K Bvt	es	e
		Ð]					-,			
Finish]								
										_		
		Tota	al Mem	orv Size: 4096K By	tes Memory Siz	ze Used	: 629K B	/tes	Memory Size	e Left: 34	57K Bytes	~
			Sp	eaker Voice								
				riven Source		H	HT45F24A	HT45F3W	HT83F02	HT86B03	H	
						Ц		HI86B20	H186830	HI86B40		
						Ц	MCII Reco	HIGODOO	1100070	HIGODOU	Ш	
							MOO Reso	urce.	1			
						F					F	
						H					H	

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



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

roject Nam	voice_test_pcm8	New Project	Load Project	Save Project
Basic Setting Voice Program	Mode Selection Evaluation Mode (built-in code) Hint: You may choose available functions first, then doing advance settings on Available Function C: Add C: Remove	PA1 1 PC0 2 PA4 3 PA3 4 PC1 5	Waveform Editor	28 PA5 PA6 Key2 PA7 Key1 25 VSS 24 VDD
	+ / - Play File Name Nickn Source Setting Source: C:\Users\Adm identifye: Wave Mode: HT-uPCM8 HT-ADPCM4 Nickname: HT-PCM16	ame pompression Mod pri-	cat_16k ur voice	D96K Bytes
	Source Setting			2
	Source: C:\Users\Admin	nistrator\Desktop\musi	c∖Bearcat_16k	
	Type: Wave			
	Mode: HT-ADPCM4	•		
	Nickname: Bearcat_16k You can choose file	e an impressive name OK	tor your voice	


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.

Voice MCU Wo	kshop [E:\ht66	5FV140_pcm8]		New Project		
roject Name:	nt66FV140	_pcm8		New Project	Load Project	Save Project
		Voice Program List				
		Trigger Source	Trigger Source Name	Function		
Basic Setting		:				
bice Program	n					
Finish						
	-					
				4		
		Available Trigger Sourc	e Available Function			
		Key Command	x 0 Play Voice			



roject Name: ht66F\	/140_pcm8		New Project	Load Project	Save Project
asic Setting	Voice Program List Trigger Source Command 1	Trigger Source Nam	e Function	ice	[_]
hice Program				Reset ALL Action Parameter D_Bearcat_16K Learcat_16K C	
			4		• •
	Availabl Trigger Source	x 0 x 29	ion ce		

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.

() Voice MCU Worl	kshop [S:\Voice平台\平台軟體	\Voice_V223_20161004_TEST project\TEST_P]	
Project Name:	TEST_P	New Project Load Project Save Project Save Ne	w Project
Basic Setting			
Voice Program	n		
Finish		Process Finish! You can test your product now!	
		✓ SPI Flash data verification ○ Only Generate Voice File(.dat	



- 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.

Project Location : E:woice20140807\HT66FV140_pcm8 The location is the same as the proj location in the workshop Project MCU HT66FV140 Select MCU with Boot Loader RAM Size: 1 Bank(s) ROM Size: 200H	roject Location : E:woice20140807\HT66FV140_pcm8 The location is the same as the project location in the workshop roject MCU Information with Boot Loader I Bank(s) Choose Language Tool : Holtek C Compiler V3/Assembler I Bank Size: 2 Boot Loader Size: 0H	HT66FV140 PCM8 TEST Project	name
Project Location : E:Woice20140807\HT66FV140_pcm8 Project MCU HT66FV140 Select MCU with Boot Loader The location is the same as the proj location in the workshop Create directory for project MCU Information RAM Size: 1 Bank(s) BOM Size: 200H	roject Location : E:/woice20140807\HT66FV140_pcm8 The location is the same as the project location in the workshop roject MCU HT66FV140 Select MCU with Boot Loader with Boot Loader Holtek C Compiler V3/Assembler MCU Information RAM Size: 1 Bank(s) ROM Size: 200H Stack Size: 2 Bootl coder Size: 0H	11001 / 140_10100_1EB1	
E:woice20140807\HT66FV140_pcm8 Inelocation is the same as the projoce in the workshop is the same as the projoce in the workshop in the workshop is the same as the project in the workshop in the workshop is the same as the projoce in the workshop in the workshop is the same as the projoce in the workshop in the workshop is the same as the projoce in the workshop in the workshop is the same as the projoce in the workshop in the workshop is the same as the projoce in the workshop in the workshop in the workshop is the same as the projoce in the workshop in the workshop is the same as the projoce in the workshop is the same as the projoce in the workshop in the workshop is the workshop is the workshop in the workshop is the workshop is the workshop is the workshop in the workshop is the workshop in the workshop is the workshop in the workshop is the wor	E:woice20140807\HT66FV140_pcm8 The location is the same as the project location in the workshop roject MCU Treate directory for project HT66FV140 Select MCU with Boot Loader Tool: Choose Language Tool: Holtek C Compiler V3/Assembler	Project Location :	
Project MCU Image: Construction of the workshop Image: HT66FV140 Select McCU Image: with Boot Loader Image: MCU Information Image: MCU Information RAM Size: 1 Bank(s) Image: MCU Information ROM Size: 200H	Init the workshop roject MCU With Boot Loader with Boot Loader Choose Language Tool : Holtek C Compiler V3/Assembler	E:\voice20140807\HT66FV140_pcm8	The location is the same as the project
Project MCU Create directory for project HT66FV140 Select MCU with Boot Loader RAM Size: 1 Bank(s) ROM Size: 200H	roject MCU Create directory for project HT66FV140 Select MCU with Boot Loader MCU Information with Boot Loader RAM Size: 1 Bank(s) Choose Language Tool : ROM Size: 200H Holtek C Compiler V3/Assembler Stack Size: 2		location in the workshop
HT66FV140 Select MCU with Boot Loader RAM Size: 1 Bank(s) ROM Size: 200H	HT66FV140 Select MCU with Boot Loader RAM Size: 1 Bank(s) Choose Language Tool : Holtek C Compiler V3/Assembler Size: 0H	Project MCU	V Create directory for project
with Boot Loader RAM Size: 1 Bank(s)	with Boot Loader RAM Size: 1 Bank(s) Choose Language Tool: ROM Size: 200H Holtek C Compiler V3/Assembler Stack Size: 2	HT66FV140 Select MCU	MCU Information
ROM Size: 200H	Choose Language Tool : Holtek C Compiler V3/Assembler	with Boot Loader	RAM Size: 1 Bank(s)
	Choose Language Tool : Norm Size: 20011 Holtek C Compiler V3/Assembler Stack Size: 2		ROM Size: 200H
Choose Language Tool : Robin Shot Size: 2	Holtek C Compiler V3/Assembler	Choose Language Tool :	Shak Size 2
Holtek C Compiler V3/Assembler	BootLoader Size: 0H	Holtek C Compiler V3/Assembler	V SIBUK SIZE. 2
BootLoader Size: OH			BootLoader Size: 0H
Choose Language Tool : Norm Size: 2 Holtek C Compiler V3/Assembler BootLoader Size: 0H		Choose Language Tool : Holtek C Compiler V3/Assembler	KOM Size: 200H Stack Size: 2 BootLoader Size: 0H

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

HT66FV140_UPCM8 Altium Library into the IDE3000 project directory. 13 KB



Click "next" and then choose the development language.

Create a source file to add to	your project
Choose the file type to create	
💿 .ASM	
O.C	



Add the library file to the new IDE3000 project

oject settings								25
Project Option	Debug Option	Directories	Document	Production				
Micro Controlle	r	HT66FV140		*				
Language Too	l:	with BootL	oader		*			
Holtek C Com	piler V3/Assen	nbler	•	Projec	ct's Build Op	tion		
Assembler/Co	mpiler Options	5						
Define Sym	bol	V3						
Generate	e listing file	Generate	Project listin	g file(list)				
Linker Options	-							
Libraries						Browse		
Province								-
Browse	- North	_						
30-	voice_tes	t_pcm8 ►				▼ \$\$	Search voi	ce_tes
Organize 🔻	New folde	er						•
☆ Favorites	- Na	ame	*			Date mo	odified	Т
E Desktop		HT66FV14	0 HTADPC	M4.lib		30/07/20	014 17:32	O
딇 Recent F	Place 🗍	Voice File	s	180		15/08/20	014 10:30	Fi
📙 Creative	Clo ≡							
Downloa	ads							

- Refer to the <u>ASM CALL</u> or <u>C CALL</u> 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.

Voice MCU Workshop	
Project Nam	New Project Load Project Save Project
Basic Setting Voice Program Finish	Load Project Project Name: Voice test porms Image: Desktop Image:



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		MX25L3206E			
	MX25L6406E		MX25L3235E			
	MX25L6435E	32IVI DIIS	MX25L3208E			
64IVI DIts	MX25L6408E		MX25L3273E			
	MX25L6473E		MX25L8006E			
	MX25L1606E	8M bits	MX25L8035E			
	MX25L1633E		MX25L8036E			
	MX25L1608E		MX25L4006E			
TOW DIS	MX25L1635E	4IVI DITS	MX25L4026E			
	MX25L1636E		MX25L2006E			
	MX25L1673E	ZIVI DITS	MX25L2026E			
	MX25L1006E	E40K hits				
TIVI DITS	MX25L1026E	512K DIts	MX25L512E			
SST Series						
64M bits	SST26VF064B	8M bits	SST25VF080B			
	SST25VF032B					
32IVI DILS	SST26VF032B	4IVI DILS	33123VF040D			
1CM bits	SST25VF016B		SST25PF020B			
TOW DILS	SST26VF016B		SST25VF020B			
Winbond Series						
129M bito	W25Q128BV		W25Q80CV			
120IVI DILS	W25Q128FV	8M bits	W25Q80DV			
GAM bito	W25Q64CV	-	W25Q80BL			
64IVI DILS	W25Q64FV		W25Q40CL			
20M bito	W25Q32FV	4IVI DILS	W25X40CL			
SZIVI DILS	W25Q32BV	2M bito	W25Q20CL			
	W25Q16CV		W25X20CL			
16M bits	W25Q32BV	1M bits	W25X10CL			
	W25Q16CL	512K bits	W25X05CL			
	GigaDevi	ce Series				
128M bits	GD25Q128C	4M bito	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

F	Project settings					x
	Project Option	Debug Option	Directories	Document	Production	
	Micro Control	ler H.	C66FV140		•	
	Language Too Holtek C Con Assembler/Co	l: npiler V3/Assem ompiler Options	bler ,	- Build	Option	ISP Bootloader Option
	Define Sym	bol 📃	73, Generate Pro	oject listing fi	le (list)	
	Linker Option Libraries Section add:	ns HT66F ress e Map File	V140_Voice_	Library.lib,		Browse
						OK Cancel



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.(<u>ASM Program Example</u>)



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, Res	serve
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 ;;	
777	

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:

Voice Program List		
Trigger Source	Trigger Source Name	Function
Sentence 1	Sentence 1	Play
Sentence 2	Sentence 2	Play



 _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



Description:

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

Example:

_

1	
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

[Applicat	tion 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	OCH	
CLR	WDT	
CLR	WDT2	
JMP	_PLAY_VOICE_ISR	;Timer1 interrupt(voice)
ORG	50H	

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	ООН					
CLR	WDT					
CLR	WDT2					
JMP	Begin					
ORG	08H					
CLR	WDT					
CLR	WDT2					
JMP	_PLAY_SEN	TENCE_	ISR	;Timer0	interrupt(sentence)
ORG	OCH					
CLR	WDT					
CLR	WDT2					
JMP	_PLAY_VOI	CE_ISR		;Timer1	interrupt(voice)
ORG	50H					

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]

HT66FV150	.INC				
HT66FV150	.HED				
.SECTION	AT	0000H	'CODE'		
00H					
WDT					
WDT2					
Begin					
08H					
WDT					
WDT2					
_PLAY_SEN	TENCE_	ISR	;TimerO	interrupt(sentence)
OCH					
WDT					
WDT2					
_PLAY_VOI	CE_ISR		;Timer1	interrupt(voice)
50H					
	HT66FV150 HT66FV150 .SECTION 00H WDT WDT2 Begin 08H WDT WDT2 _PLAY_SEN 0CH WDT WDT2 _PLAY_VOI0 50H	HT66FV150.INC HT66FV150.HED .SECTION AT 00H WDT WDT2 Begin 08H WDT WDT2 _PLAY_SENTENCE_ 0CH WDT WDT2 _PLAY_VOICE_ISR 50H	HT66FV150.INC HT66FV150.HED .SECTION AT 0000H 00H WDT WDT2 Begin 08H WDT WDT2 _PLAY_SENTENCE_ISR 0CH WDT WDT2 _PLAY_VOICE_ISR 50H	HT66FV150.INC HT66FV150.HED .SECTION AT 0000H 'CODE' 00H WDT WDT2 Begin 08H WDT WDT2 _PLAY_SENTENCE_ISR ;Timer0 0CH WDT WDT2 _PLAY_VOICE_ISR ;Timer1 50H	HT66FV150.INC HT66FV150.HED .SECTION AT 0000H 'CODE' OOH WDT WDT2 Begin 08H WDT WDT2 _PLAY_SENTENCE_ISR ;Timer0 interrupt(OCH WDT WDT2 _PLAY_VOICE_ISR ;Timer1 interrupt(50H

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]

#I1	ICLUDE	HT66FV160	.INC				
#INCLUDE HT66FV160.HED							
COI	DE	.SECTION	AT	0000H	'CODE'		
	ORG	00H					
	CLR	WDT					
	CLR	WDT2					
	JMP	Begin					
	ORG	08H					
	MOV	BackupAcc	, A				
	MOV	A,PBP					
	CLR	PBP					
	JMP	_PLAY_SEN	TENCE_	ISR	;TimerO	interrupt(sentence)
	ORG	0CH					
	MOV	BackupAcc	, A				
	MOV	A,PBP					
	CLR	PBP					
	JMP	_PLAY_VOI	CE_ISF	R	;Timer1	interrupt(voice)
	ORG	 50н	_				

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]

BH67F2262.INC #INCLUDE #INCLUDE BH67F2262.HED .SECTION AT 0000H 'CODE' CODE ORG 00H WDT CLR CLR WDT2 JMP Begin ORG 010H MOV BackupAcc,A MOV A,PBP CLR PBP JMP PLAY SENTENCE ISR ;Timer0 interrupt(sentence) 014H ORG MOV BackupAcc,A MOV A,PBP CLR PBP JMP PLAY VOICE ISR ;Timer1 interrupt(voice) 50H ORG Begin: _CLRRAM ;Clear all RAM banks CALL SYSTEM INITIALIZATION CALL ;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 SΖ fSentencePlaying JMP ;Wait play sentence finish \$-3 _PLAY_SENTENCE_INDEX 0,1,0,5,0 ;Play the sentence which inedx is 1, ;volume is 5 WDT CLR CLR WDT2 SΖ fSentencePlaying \$-3 ;Wait play sentence finish JMP CALL DAC RAMP DOWN ;Close DAC and do ramp down WDT CLR CLR WDT2 JMP \$-2



■ [Application example – HT45F67]

#INCLUDE #INCLUDE	HT45F67.INC HT45F67.HED		
CODE ORG CLR CLR JMP ORG MOV MOV	.SECTION AT 00H WDT WDT2 Begin 10H BackupAcc, A A,BP BB	0000H	'CODE'
JMP ORG MOV MOV CLR	PLAY_VOICE_ISR 14H BackupAcc, A A,BP BP		;Timer2 interrupt(voice)
JMP ORG	_PLAY_SENTENCE_I 30H	SR	;Timerl interrupt(sentence)
Begin:			
CALL CALL CALL PLAY_ CLR CLR	_CLRRAM _SYSTEM_INITIALI _DAC_RAMP_UP VOICE 0,0,0,5,0 WDT WDT2	ZATION	;Clear all RAM banks ;System initialization ;Open DAC and do ramp up ;Play the first audio, volume is 5
SNZ JMP _PLAY_	<pre>iVolceStandBy \$-3 SENTENCE 01H,00H,</pre>	0,5,0	;Wait play voice finish ;Play the sentence whose address is ;0100H, volume is 5
CLR CLR SZ JMP	WDT WDT2 fSentencePlaying \$-3		;Wait play sentence finish

\$-2

JMP



■ [Application example – HT45F65]

#INCLUDE #INCLUDE	HT45F65.IN HT45F65.HE	C D				
CODE	.SECTION	AT	0000H	'CODE'		
ORG	00H					
CLR	WDT					
CLR	WDT2					
JMP	Begin					
ORG	10H					
MOV	BackupAcc,	А				
MOV	A,BP					
CLR	BP					
JMP	_PLAY_SENT	ENCE_	ISR	;Timer1	interrupt(sentence)
ORG	18H					
MOV	BackupAcc,	А				
MOV	A,BP					
CLR	BP					
JMP	_PLAY_VOIC	E_ISF	ξ	;Timer2	interrupt(voice)
ORG	30H					

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	0000н	'CODE'		
ORG	00н					
CLR	WDT					
CLR	WDT2					
JMP	Begin					
ORG	0CH					
MOV	BackupAcc,	A				
MOV	A,BP					
CLR	BP					
JMP	PLAY SENTE	NCE	ISR	;Timer1	interrupt(ser	ntence);
ORG	 10н	_	-		1	
MOV	BackupAcc,	A				
MOV	A,BP					
CLR	BP					
JMP	PLAY VOICE	ISF		;Timer2	interrupt(vo	ice)
ORG	 ЗОН	_			1	

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 #INCLUDE	HT66F4550 HT66F4550	.INC .HED				
CODE	.SECTION	AT	0000H	'CODE'		
ORG	00H					
CLR	WDT					
CLR	WDT2					
JMP	Begin					
ORG	OCH					
CLR	WDT					
CLR	WDT2					
JMP	_PLAY_SEN	TENCE	ISR	;Timer0	interrupt(sentence)
ORG	10H					
CLR	WDT					
CLR	WDT2					
JMP	_PLAY_VOI	CE_ISF	ł	;Timer1	interrupt(voice)
ORG	50H					

CALL	_CLRRAM	;Clear all RAM banks
CALL	_SYSTEM_INITIALIZATION	;System initialization
CALL	_DAC_RAMP_UP	;Open DAC and do ramp up
PLAY SNZ	VOICE 0,0,0,0,0 fVoiceStandBy	;Play the first audio
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 SZ	SENTENCE_INDEX 0,1,0,0,0	;Play the first sentence
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.HE	ED					
CODE	.SECTION	AT	0000H	'CODE'			
ORG	00H						
CLR	WDT						
CLR	WDT2						
JMP	Begin						
ORG	OCH						
CLR	WDT						
CLR	WDT2						
JMP	PLAY SENTE	ENCE I	SR	;Timer0	interrupt(sentence)	
		_					
ORG	10H						
CLR	WDT						
CLR	WDT2						
JMP	PLAY VOICE	I ISR		;Timer1	interrupt(voice)	
		_					
ORG	20Н						

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.IN	NC							
#INCLUDE	HT45F24A.HB	HT45F24A.HED							
0000	CTOTAN	2.00	000011						
CODE	.SECTION	A'l'	UUUUH	'CODE'					
ORG	00H								
CLR	WDT								
CLR	WDT2								
JMP	Begin								
ORG	OCH								
CLR	WDT								
CLR	WDT2								
JMP	_PLAY_SENTH	ENCE	ISR	;Timer0	interrupt(sentence)			
ORG	10H								
CLR	WDT								
CLR	WDT2								
JMP	_PLAY_VOICH	E_ISR		;Timer1	interrupt(voice)			
ORG	20н								

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.IN	IC				
#INCLUDE	HT83F02.HE	D				
CODE	.SECTION	AT	0000H	'CODE'		
ORG	00H					
CLR	WDT					
CLR	WDT2					
JMP	Begin					
ORG	08H					
CLR	WDT					
CLR	WDT2					
JMP	_PLAY_SENI	ENCE	ISR	;TimerO	interrupt(sentence)
		_	-			
ORG	OCH					
CLR	WDT					
CLR	WDT2					
JMP	PLAY VOIC	E ISF	2	;Timer1	interrupt(voice)
		_				
ORG	20н					

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.II	NC				
#INCLUDE	HT86B03.HI	ED				
CODE	.SECTION	AT	0000H	'CODE'		
ORG	OOH					
CLR	WDT					
CLR	WDT2					
JMP	Begin					
ORG	08H					
CLR	WDT					
CLR	WDT2					
JMP	_PLAY_SEN	TENCE_	ISR	;Timer0	interrupt(sentence)
ORG	OCH					
CLR	WDT					
CLR	WDT2					
JMP	_PLAY_VOI	CE_ISF	R	;Timer1	interrupt(voice)
		_				
ORG	20Н					

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	2				
#INCLUDE	HT86B40.HEI)				
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_SENTE	ENCE_]	ISR	;Timer0	interrupt(sentence)
ORG	10H					
MOV	BackupAcc,	A				
MOV	A,BP					
CLR	BP					
JMP	_PLAY_VOICE	E_ISR		;Timer2	interrupt(voice)
ORG	20H					

```
Begin:
```

CALL	CLRRAM	;Clear all RAM banks
CALL		;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

Project settings			x		
Project Option	Debug Option	Directories Document Production			
Micro Control	ler H	T66FV140 -			
Language Too Holtek C Cor	l: npiler V3/Assem	nbler 🔹 Build Option ISP Bootloader Option			
-Assembler/Co	ompiler Options				
Define Sym	bol _	_V3,			
📃 Generati	e listing file 🛛 👽	🛙 Generate Project listing file(list)			
-Linker Optio	ns				
Libraries	HT66F	7V140_Voice_Library.lib, Browse			
Section add	ress				
🔄 Generate Map File					
		OK Cau	ncel		


■ 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 (<u>C Program Example</u>).



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() , 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);



Voice Program List		
Trigger Source	Trigger Source Name	Function
Sentence 1	Sentence 1	Play
Sentence 2	Sentence 2	Play



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 w	hen 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 ex condition of the voice play paus	ample, which is not provided in the voice library. The specific se 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();
                                       //Clear all RAM banks
    CLRRAM();
    SYSTEM INITIALIZATION();
                                       //System initialization
    DAC_RAMP_UP();
                                       //Open DAC and do ramp up
                                       //Play the first audio, volume is 5
    PLAY VOICE(0,0,5);
    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();
                                      //Clear all RAM banks
   CLRRAM();
   SYSTEM INITIALIZATION();
                                      //System initialization
   DAC RAMP UP();
                                      //Open DAC and do ramp up
                                      //Play the first audio, volume is 5
    PLAY VOICE(0,0,5);
   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 the first sentence, volume is 5
    PLAY SENTENCE INDEX(0,1,5);
    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
                                     //Close DAC and do ramp down
    DAC 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();
                                      //Clear all RAM banks
   CLRRAM();
    SYSTEM INITIALIZATION();
                                      //System initialization
                                      //Open DAC and do ramp up
   DAC RAMP UP();
                                      //Play the first audio, volume is 5
    PLAY VOICE(0,0,5);
    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();
                                      //Clear all RAM banks
   CLRRAM();
    SYSTEM INITIALIZATION();
                                      //System initialization
                                      //Open DAC and do ramp up
   DAC RAMP UP();
                                      //Play the first audio, volume is 5
    PLAY VOICE(0,0,5);
    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();
    }
```



ASM and C library Instructions

```
■ [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();
                                      //Clear all RAM banks
   CLRRAM();
    SYSTEM INITIALIZATION();
                                      //System initialization
                                      //Open DAC and do ramp up
   DAC RAMP UP();
                                      //Play the first audio, volume is 5
    PLAY VOICE(0,0,5);
    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();
                                       //Clear all RAM banks
    CLRRAM();
    SYSTEM INITIALIZATION();
                                       //System initialization
                                       //Open DAC and do ramp up
    DAC RAMP UP();
                                       //Play the first audio
    PLAY VOICE(0,0,0);
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish
    PLAY SENTENCE (0 \times 01, 0 \times 00, 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();
                                       //Clear all RAM banks
   CLRRAM();
    SYSTEM INITIALIZATION();
                                       //System initialization
                                       //Open DAC and do ramp up
   DAC RAMP UP();
                                       //Play the first audio, volume is 5
    PLAY VOICE(0,0,5);
    while(!CHECK_PLAYVOICE_FINISH()); //Wait play voice finish
    PLAY SENTENCE (0 \times 01, 0 \times 00, 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 the first audio, volume is 5
    PLAY VOICE(0,0,5);
   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
                                     //Close DAC and do ramp down
    DAC 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 the first audio, volume is 5
    PLAY VOICE(0,0,5);
   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
                                     //Close DAC and do ramp down
    DAC 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
                                      //Close DAC and do ramp down
    DAC 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
                                       //Open DAC and do ramp up
    DAC 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 the first sentence, volume is 5
    PLAY SENTENCE INDEX(0,1,5);
    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:

Compression Resources Mode Used	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:

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 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 "<u>Connection for Programming DAT File to the Flash</u>" section for SPI flash connection and programming.



HT66FV140

Resource Usage Table:

Compression Resources Mode Used	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 "<u>Connection for Programming DAT File to the Flash</u>" section for the SPI flash connection and programming.

HT66FV150

Resource Usage Table:

Compression Resources Mode Used	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
- Timerlinterrupt 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:

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 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 "<u>Connection for Programming DAT File to the Flash</u>" section for SPI flash connection and programming.



HT66FV160

Resource Usage Table:

Compression Resources Mode 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
- Timerlinterrupt 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 "<u>Connection for Programming DAT File to the Flash</u>" section for SPI flash connection and programming.



BH67F2262

Resource Usage Table:				
DEFAULT	HT-ADPCM4	PCM12	UPCM8	PCM16
650/16384 (4%)	241/16384 (2%)	51/16384 (1%)	316/16384 (2%)	17/16384 (1%)
		38/512(7%)		
	500H–578H	No decoding array	600H–6FDH 700H–701H	No decoding array
57EH–582H 583H–58CH	579H–57DH		6FEH–6FFH 702H–703H	
2				
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				
	ble: DEFAULT 650/16384 (4%) 57EH–582H 583H–58CH SPI1: SPIC0, SP PWM: PWMC, U Timer: TM1C0, T General: ACC, M I/O: PMPS0, PBF	DEFAULT HT-ADPCM4 650/16384 (4%) 241/16384 (2%) 650/16384 (4%) 241/16384 (2%) 575H-582H 500H-578H 57EH-582H 579H-57DH 57EH-58CH 579H-57DH SPI1: SPIC0, SPIC1, SPID PWM: PWMC, USVC, PLADH, PL Timer: TM1C0, TM1C1, TM1AL, TI General: ACC, MP1, IAR1, TBLP, I/O: PMPS0, PBPU, PBS1, PGS1	DEFAULT HT-ADPCM4 PCM12 650/16384 (4%) 241/16384 (2%) 51/16384 (1%) 650/16384 (4%) 241/16384 (2%) 51/16384 (1%) 650/16384 (4%) 241/16384 (2%) 51/16384 (1%) 575EH-582H 5500H-578H No decoding array 57EH-582H 579H-57DH S79H-57DH 571: SPIC0, SPIC1, SPID 2 SPI1: SPIC0, SPIC1, SPID 2 SPIN: PWMC, USVC, PLADH, PLADL 3 Timer: TM1C0, TM1C1, TM1AL, TM1AH, TM0C0, TH 3 General: ACC, MP1, IAR1, TBLP, TBLH, TBHP, PC 3 I/O: PMPS0, PBPU, PBS1, PGS1 3	DEFAULT HT-ADPCM4 PCM12 UPCM8 650/16384 (4%) 241/16384 (2%) 51/16384 (1%) 316/16384 (2%) 650/16384 (4%) 241/16384 (2%) 51/16384 (1%) 316/16384 (2%) 650/16384 (4%) 241/16384 (2%) 51/16384 (1%) 316/16384 (2%) 550/16384 (4%) 241/16384 (2%) 58/512(7%) 38/512(7%) 5500H-578H No decoding array 600H-6FDH 700H-701H 57EH-582H 583H-58CH 579H-57DH 6FEH-6FFH 702H-703H 57EH-582H 583H-58CH 579H-57DH 2 SPI1: SPIC0, SPIC1, SPID PWM: PWMC, USVC, PLADH, PLADL Timer: TM1C0, TM1C1, TM1AL, TM1AH, TM0C0, TM0C1, TM0AL, TM 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:

Compression Resources Mode Used	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8
PROM(Word)	707/32768(2%)	241/32768 (1%)	98/32768 (1%)	316/32768 (1%)
RAM(Byte)		40/51	2(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	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		AL, TM1AH	

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:

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 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 "<u>Connection for Programming DAT File to the Flash</u>" section for SPI flash connection and programming.

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





HT45F65

	Resource	Usage	Table:
--	----------	-------	--------

Compression Resources Mode Used	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	6 (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 "<u>Connection for Programming DAT File to the Flash</u>" section for SPI flash connection and programming.

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





HT45F3W

Resource Usage Table:

Compression Resources Mode Used	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:

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 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 "<u>Connection for Programming DAT File to the Flash</u>" section for SPI flash connection and programming.

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





HT66F4550

	Resource	Usage	Table:
--	----------	-------	--------

Compression Resources Mode Used	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:





- VDD OCDSCK OCDSDA GND SDO SDO SDO SDO SDO SDO CE CE SPI Flash
- HT66V4550 VDD, GND, OCDSCK, OCDSDA pins connection to the e-Link.

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







HT45F23A

Resource Usage Table:

Compression Resources Mode Used	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	8(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	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:

System Frequency Compression Mode	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 "<u>Connection for Programming DAT File to the Flash</u>" section for SPI flash connection and programming.



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



HT45F24A

Resource Usage Table:

Compression Resources Mode Used	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	2(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	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:

System Frequency Compression Mode	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.





- **Note1:** Refer to "<u>Connection for Programming DAT File to the Flash</u>" section for SPI flash connection and programming.
- Note2: The following figure shows an audio reference circuit built using the HT82V733:





HT83F02

Resource Usage Table:

Compression Resources Mode Used	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	3(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	2	
Registers used	SPI: SIMC0A, SIMC2 D/A: DACTRL, DAH, Timer: TMR0, TMR00 General: ACC, MP1,	2A, SIMDRA, SIMDRB DAL C, TMR1, TMR1C IAR1, INTC, TBLP, TE	3 BLH	

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:

System Frequency Compression Mode	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.



Note1: Refer to "<u>Connection for Programming DAT File to the Flash</u>" 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:

	e						
Compression Resources Mode Used	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8			
PROM(Word)	470/4096 (11%)	470/4096 (11%) 241/4096 (6%) 9		4096 (2%) 316/4096 (8%)			
RAM(Byte)		37/192	2(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_LAR1_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:

Compression Resources Mode Used	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	2(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:

Compression Resources Mode Used	DEFAULT	HT-ADPCM4	HT-PCM12	HT-UPCM8			
PROM(Word)	648/8192 (16%) 241/8192 (3%)		98/8192 (1%)	316/8192 (4%)			
RAM(Byte)		40/38	4(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 JAR1 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





- 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:

File		
J:	\library\HT86Bx0_LIB20140730\So	urce Code\l Open
Open File		
Look in: 🔒	HT86Bx0_UPCM8	- G Ø ▷ □.
Name		Date modified
📙 Code		15/08/2014 16:1
📕 Source		15/08/2014 16:1
📕 Temp		15/08/2014 16:1
ht86bx0_	upcm8.DAT	13/05/2014 16:2
•	III]
File name:	ht86bx0_upcm8.DAT	Open



• download data:

Downloading dat	э
ок]

• 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 way, aiff, mp3 or ogg file and burn to Audio CDs.



Quick Start

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

Audacity						. 0			<u>.</u>	<u>ل</u>	/- D	X
File Edit View	Transport	Tracks Genera	te Effect Ar	alyze Help		_	_	-	_	_	_	-
⋓⋗					4) ↓ -3€	-24 -12 0	L R ♪	-36 -24 -1:	2 0			
•)	·····b/	Ø <u></u>		● ● ·₩	nHn 🗠	○	PPI	₽ <u>₽</u>	I	·····		
	- +)		- 10		-	•						
- 1.0	9.9	1.0	2.0	3.0	4.0	5.0	6.	0	7.0	8.0	9.0	
												E
	•											F
Project Rate (Hz):	an mar	Selection Start:	0	End C Lengt	th	Audio Position	n:	-				
44100 -	Snap To	00 h 00 m 0	0.000 s • 0	0 h 0 0 m 0 0	.000 s*	00 h 00 m	00.000 s	1				
	_		-									ai



• Audacity Interface Toolbars Overview



① Audacity Transport Toolbar Description

(41)	Skip to start	11	Temporarily pauses playing or recording without losing the present location. Click Pause a second time to resume.
Þ	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

Ι	Selection- click and drag to select a range of audio to play or edit	P	Zoom- zoom in or zoom out the track
D d	Envelope- made smooth volume change over the length of a track	↔	Time shift- drag audio tracks left or right
Ø	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.

New	Ctrl+N	2 0 L		R				
Open	Ctrl+O	↔ * •>	36 -24	-12 0	-36 -24	-12 0		
Recent Files	,	₽1 1 1 1 1 1 1 1	000	0 1	0 0 0 0			
Close	Ctrl+W			•		V		
Save Project	Ctrl+S					70		
Save Project As		3.0	4.0	5,0	6.0	7.0	8,0	9.0
Save Compressed Copy of Project								
Check Dependencies								
Edit Metadata								
Import	•	Audio	Ctrl+Shift+I					
Export	Ctrl+Shift+E	Labels						
Export Selection		MIDL						
Event Labels	L	Raw Data						
Export Multiple	Ctrl+Shift+1							
Export MIDI	Curronnere							
Apply Chain								
Edit Chains								
Page Setup								
Print								
Euie	CHLO							
•			"	1				



Look in:	📙 music		- 3 🕫 😕 🛄	•
Pro	Name	#	Title	С
	1001_en.wav			
Recent Places	1-04-22K.pkf			
_	1-04-22K way			
	1-18-22K way			
Desktop				
Desktop	DO2_en.wav			
	1 003_en.wav			
	004_en.wav			
Libraries	1005_en.wav			
	📆 006_en.wav			
	📅 007_en.wav			
Computer	A Whole New World_V_22K.pkf			
	A Whole New World_V_22K.wav			
	Poprost 16k pkf			
Network	•			P
	File name: 003_en.wav		•	Open
	Files of type:		_	Cancel

• 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 \mathbf{k} , 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

30

Before a Cross Fade Out

30 Т

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" :



0dB					
-12dB					
-24dB -					
-36dB -					
-48dB-					
-60dB				,	
-60dB	-48dB	-42dB -36dB	-30dB -24dB	-18dB -12	dB -6dB 0d
Threshold:	,			-0	-12 dB
Noise Floor:	,		0		-40 dB
Ratio:	-0				2:1
Attack Time:	-0				0.2 se
					1.0.55
Decay Time: Make- Preview	up gain for 0	dB after compre	essing 🔲 Com	opress based	on Peaks
Decay Time: Make- Preview	up gain for 00	dB after compre	essing 🔲 Com	OK	on Peaks Cancel
Decay Time: Make- Preview	up gain for 00	dB after compre	essing Com	OK	on Peaks Cancel
Decay Time:	up gain for 00	dB after compre	essing Com	OK	on Peaks Cancel
Decay Time:	up gain for 00	dB after compre	essing Com	OK	on Peaks Cancel
Decay Time:	up gain for 0	dB after compre	essing Com	OK	on Peaks Cancel
Decay Time:	up gain for 0	dB after compre	essing 🔲 Corr	OK	on Peaks Cancel
Decay Time: Make- Preview	up gain for 0	dB after compre	essing Corr	OK	on Peaks Cancel
Decay Time: Make- Preview	up gain for 0	dB after compre	essing Corr	OK	on Peaks Cancel
Decay Time:	up gain for 0	dB after compre	essing Com	press based OK	on Peaks Cancel
Decay Time:	up gain for 0	dB after compre	essing Com	press based OK	on Peaks Cancel

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



• Silence the selection:

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





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" \rightarrow "Add New" \rightarrow "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" \rightarrow "Import" \rightarrow "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" \rightarrow "Convert Sample Type" and set the required Sample Rate and Bit Depth .

Convert Sample Type	
Sample Rate Conversion Sample Rate: 44100 Hz Advanced	
Channels Channels: Same as Source 💌 Advanced	
Bit Depth Bit Depth: 16 ▼ bits ► Advanced	
OK Cancel	

Þ

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.

AdoteAutition	
Allo Audors Carlo Audors Carlo Carl	
n Hang too	
Export File	
File N	ame: Mission Impossible_22K_01.wav
Loca	tion: F:\music
Fo	mat: Wave PCM (*.wav, *.bwf)
Sample	Sype: 22050 Hz Mono, 16-bit Change
Format Set	ings: Wave Uncompressed Change 16-bit Integer

OK

🗹 Include markers and other metadata

Estimated File Size: 1.82 MB


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.



New Multitrack Se	ession	×
Session Name:	Untitled Session 1	
Folder Location:	C:\Users\cgzeng\Documents\Adobe\Audi	Browse
Template:	None	
Sample Rate:	48000 V Hz	
Bit Depth:	32 (float) 💌 bits	
Master:	Stereo 🔻	
	ОК	Cancel



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.

Ne Edit Multitrack Clip Effects Favorites View Window Help				
🔠 Henders 📰 Haldback 👘 🖬 🗈 🏊 🇞 (HH 🗎 🖂 🖓	2		Notopace Detaut	D Search Help
clast -1	fatur untited Section Local (# 1			
🖿 📾 🖬 . Á. 12				
Name A Status Duration Sample Rate Charmets B				
In United Session Large 0.18.000 48000 Hz Stereo 3 Int United Session Zarga 0.00.000 48000 Hz Manu 5	🚅 🎢 🕼 🖄 🖄 👷 🗋 ở 👬 🗤 20 30 40 50 60 70 60 90 930 910 92			
tee Unit Bed Session Ruless * 0.00.000 40.000 Hz Steveo B				
	Hore , g	Pasze	Ctd+V	
	- Maday -	Ripple Delete		
। सम्ब	Fired *	Groups	· · ·	
		Tradk		
E Celeta Dars + + + Y	JOH HOI HE	Insert	> Silence	
T m Drives Hame A Durator Media Tu	** Hore + pf	Marker	> Fiet.	
• 🚣 a 🔹 🖡 🛓 a	- Holy -	Mixdown Session to New Ne		
	 Read 	Export Madown		
Fat Fat		Bounce To New Track		
P Stotuti		Session Properties		
	+ Nore + st			
	- NOTA			
	 Read 			
	JOH HOI H			
100 m m m m m m m m m m m m m m m m m m	+ Hone + 2f			
· · · · ·	• blain ·			
F 10 141	 Real 			
History II Valeo *8				
P B. Open				
	→ Hone > gf			
	- blan			
	* Tout w			
	000000			e la 🖬 🗄 🔶 ra sa na
			and Descention	
	Lods A		- Sdelon's	ter A 10 bundlon
			Selection	
0 Lindo		a a a a a a	-3 6 View (90.800 0.36.000 0.31.000



■ 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".

Z Adobe Audition		
File Edit Multitack Clip Effects Favorites View Window Help		
🖼 Warton 🗮 Mattax 👘 💷 🔤 🐂 🏷 Hel I 😳 🖓 .		Workspace: Default - Difficult -
No X	Editor: Unitied Section 1.5co.*	· - ·
e e a d e 🖸		
Name - Status Duration Sample Rate Channels 8		
40 002_en 40000 1.way 200.405 48000 Hz Mono	計研算機 法点日 ames 28 48 60 80 380 328 340 360 3	10 208 220 240 260 288 900 520 340 968 980 42
+ 002_en.wav 808.428 8000 Hz Mono	A Taraba and a family and an and a family and and a family and a famil	
He Beanut 318 48000 Lwar D48.088 48000 Hz Mono		
In United Sector Loca" D4L075 (8000 H) Store		where the second s
Let Unitited Section Zuesz Bill.000 48000 Hz Mono		The construction of the second s
🔛 Untitled Section Long * 0.10.000 40000 Hz Steven	Early a key a second start with a key and	and the state of the second second second second second second second
5 H M	N Real A	
		Mine *
Media Browner Effects Rack Marken Properties -		
en, Contento: Drives v + +, Y,		
V Doubon Media 1	+ Motor +	and prove the second part of the second s
	a per se a la compara de la	
hat hat		
1 m 1 m 1	+ Tax3 H x H :	
• / ¹⁰ Statuts		data b
	→ Nose + β	
	Meter + Groups	•
	> feet iv	•
	Internal Wild Party State	
	John Maler Hard	
	+ Meter +	Intersound to New Fall
- - - -	Experimental Control of Control o	Crime Sellion
	P Red P	o New Track •
History Video -	Senior P	roperies
E Open		
Si Add Audio Clips	→ Nove x 21	
*+ More Clips	+ Mater +	
Modily Exyltance	a line of	
Ap Move Cips		
Ei Artifacto Cipo		5 FF 液质变变液液液
*+ Nove Clas	1 and	-T Katematika -T
D • Add Keylvane		Sat Ind Durities
		Selection BORIDO BORIDO BORIDO
3 Under 🛛		18 18 12 4 4 4 5 Wew 606000 04004



■ 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.



Export File		×
File Name:	Mission Impossible_22K_01.wav	Т
Location:	F:\music	Browse
Format	Wave PCM (*.wav, *.bwf)	
Sample Type:	22050 Hz Mono, 16-bit	Change
Format Settings:	Wave Uncompressed 16-bit Integer	Change
Include marke	rs and other metadata	
Estimated File Siz	ze: 1.82 MB	
	ОК	Cancel



Copyright[©] 2017 by HOLTEK SEMICONDUCTOR INC.

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek's products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at http://www.holtek.com/en/.