Audio ToolBox™

The Digital Audio Utility Kit

Revision A (September 1996)

Version 3.0

Contents

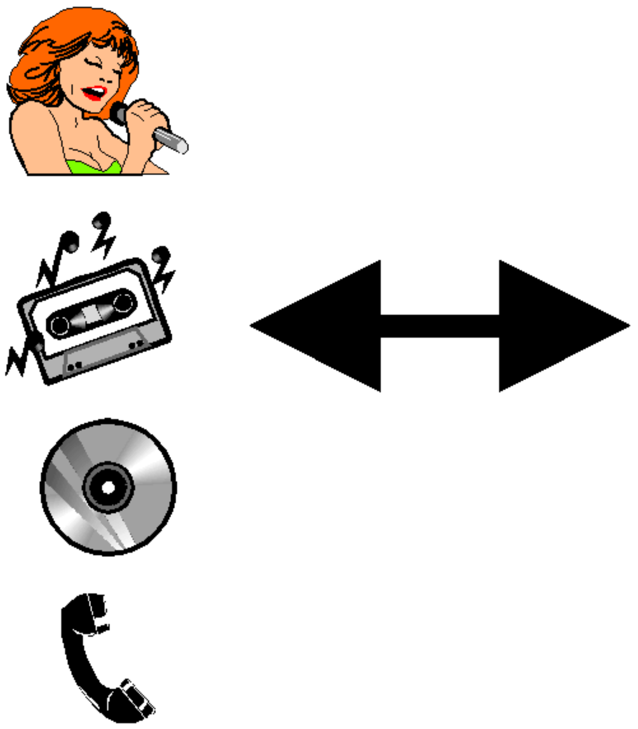
♦ User's Guide

Chapter 1, ***Introduction***, provides a brief overview of ***Audio ToolBox***™ — what it does and how it works. This chapter covers hardware and software requirements as well as ***Audio ToolBox*** technical capabilities.

Chapter 2, ***Sample Session***, gives you a step-by-step, hands-on introduction to using ***Audio ToolBox***.

If you're a new Windows user (or maybe just a little rusty), Chapter 3, ***Windows (Quick) Review*** discusses topics important for the ***Audio ToolBox Apprentice*** user.

Introduction



Dialogic 4 bit (OKI)

Dialogic 8 bit (u-Law)

C-ITU (CCITT) G.711 u-Law 8 bit

C-ITU (CCITT G.711 a-Law 8 bit

C-ITU (CCITT G.721 4 bit ADPCM

Multimedia 8 bit

Multimedia 16 bit

NewVoice CVSD 1 bit

Perception Technology CVSD 1 bit

Wave Multimedia

Audio ToolBox™ lets you convert from one audio format to another.

Save time & sound great! ***Audio ToolBox*** lets you batch process and convert your digital audio files to and from industry standard formats. ***Audio ToolBox*** is a professional quality conversion utility for the MS-DOS® and Microsoft Windows® operating system.

Do You Need To:

- To convert to and from a wide variety of audio formats?
- To “chop” a sequence of recordings into separate files?
- To adjust the volume across all your voice prompts?
- To automatically crop silence from the beginning and ending of files?
- To filter touch-tones from your files?

- To add conversion capabilities to your Windows and DOS programs?

If your answer is YES!, you can make your job a lot easier — and the results sound better — with ***Audio ToolBox.***

A Must for Developers

Are you having problems working with sounds recorded in different formats on different devices? Do you want to improve the way you produce your voice prompts? Would you like to simplify the maintenance of incoming prompts? Would you like to improve the quality of your existing prompts?

Audio ToolBox makes converting and cleaning up your sound files a breeze. It can clean up all of your prompts with just one command, leveling out the volumes, removing dead air and background noise. Most voice developers find the ***Audio ToolBox*** a must when creating a voice system. It makes the system sound like a million bucks — with very little effort on your part.

You can automatically chop a single recording session into separate files, normalize the volume across multiple files, and recombine them into an indexed file. You can add dynamic compression / expansion for improved sound, auto-crop silence from both ends, even filter DTMF Touch-Tones to eliminate "talk-off" application errors. And ***Audio ToolBox*** handles files of any length — in less than 400K of memory!

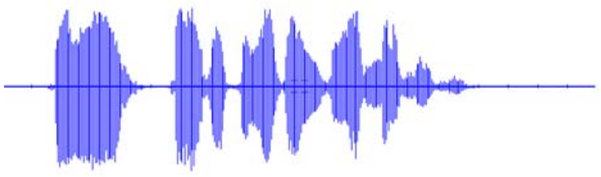
You can even record, edit and review prompts with your multimedia system, then batch process and transfer them for enhanced fidelity!

Audio ToolBox is the ONLY professional quality conversion utility for the MS-DOS® and Microsoft Windows® operating systems — and the only tool that does all this!

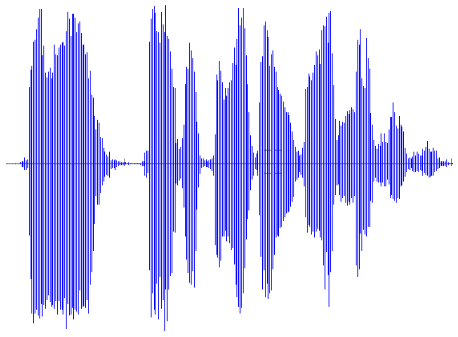
Automatically Improve Your Recordings

The basic reason to use ***Audio ToolBox*** is that you simply want your audio presentations to sound good. If you develop for voice processing, or if you only change, fix, and improve your voice prompts, you'll find ***Audio ToolBox*** just about indispensable.

Let's say you're running a computerized dating service. Your callers leave messages, perhaps thousands of them each day, and you need to ensure that the messages sound as good as possible. With ***Audio ToolBox***, simply run the program every evening to automatically adjust the volume and trim the dead space from each message — without any intervention from you.



Sound with low volume level and unwanted silence at end



Same sound with volume normalized and silence removed

More than 50 *Audio ToolBox* Options

With the six main tools and 50+ command options, and the almost uncountable combinations you can make, you have virtually unlimited ability to shape your messages. But all to what end?

With *Audio ToolBox* You Can Do It

This innovative program gives you the capability to do with spoken words, music, and sound effects what you can do with a word processor macro command. Word processor macro languages equip you to automatically cut, insert, replace, and change *written words*. ***Audio ToolBox*** lets you to do the same with *spoken words*, only more. A lot more.

You've probably discovered that downloading a cool sound file off the Internet can be a problem when it comes to using it. There are so many sound file formats, and if your system can't use the one you've downloaded, you're out of luck. With ***Audio ToolBox***, you can convert those files with a single command.

And our ***ToolBox Apprentice*** simplifies the entire process. Developed at the suggestion of our customers, ***ToolBox Apprentice*** gives you a simple way to see and change complex command lines with ease.

You want your spoken messages to *sound* as good as your written work *looks*. And they will: You'll be able to do professional level work right at home, or in your office, without having to call in expensive outside help.

Skills

Although skill in using DOS and Windows is useful, you can still do creditable work without real computer literacy. The sample sessions provide guidance in creating DOS command lines. The ***Windows (Quick) Review*** chapter presents brush-up exercises in Windows, and should help if you're not current.

Hardware Requirements

To run ***Audio ToolBox*** you'll need an IBM PC (or compatible), with a hard drive, and at least 512K of memory. A 486 processor (or math coprocessor) and 640K is recommended. ***ToolBox Apprentice*** requires Windows 3.1x or higher, 8 megabytes of memory, and at least 4 megabytes of space on your hard disk.

Audio ToolBox works with or without the VISI Accelerator™ module connected to the parallel printer port. If the accelerator is not installed, however, file conversions are limited to approximately 8 seconds in length. To install, unpack the accelerator module and follow the simple connection instructions in Appendix A, ***Installation and Setup***.

Using a Math Coprocessor

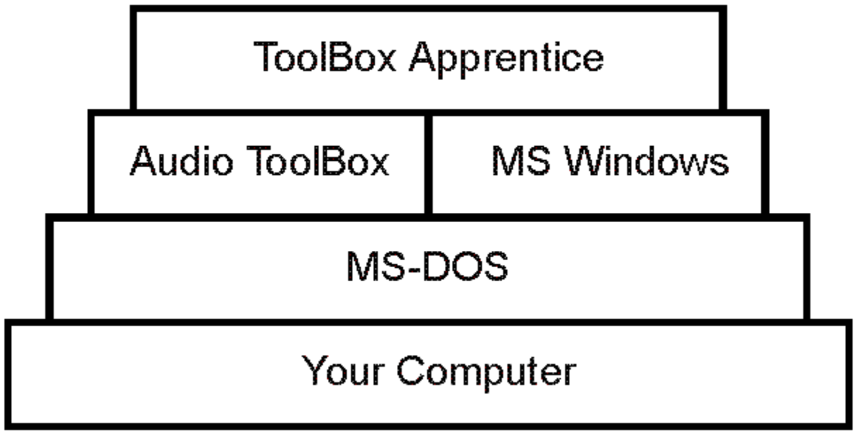
Audio ToolBox functions can benefit greatly from the use of a math coprocessor. ***Audio ToolBox*** will automatically detect and use an 80287/80387 (or compatible) math coprocessor if one is installed. Similarly, ***Audio ToolBox*** will automatically detect the presence of the coprocessor built into chips such as the Intel 80486DX or Pentium® processor. Filtered conversions will typically operate eight to twelve times faster with a coprocessor. The speed of indexed file packing and unpacking operations will not change significantly.

The following benchmarks were made on a zero wait-state 10 MHz 286, a 33 MHz 386, and a 33 MHz 486 based computer. Here are the following times required for conversions performed on a 10–second long 11 kHz multimedia wave file:

PCMCvt (processor)	No Filter (time in sec)	Filtered (time in sec)
80286	20	700
80386	1	200
80386/387	1	10
80486	1	10

Software Requirements

To run ***Audio ToolBox*** you'll need DOS 3.3 or higher. To run ***ToolBox Apprentice,*** the graphical user interface, you'll need Microsoft Windows 3.x or Windows 95.



Notational Conventions

Here are a few notes about terms and typographical conventions used in this manual.

The names of keys are spelled out in capital letters, and are shown within less-than "<" and greater-than ">" signs. For example, an enter key (or return key) is shown as <ENTER>. On your keyboard the keys may be marked with an abbreviation or be named a little differently.

- ***Simultaneous Activation***: When two or more keys must be pressed at the same time, they will be shown joined without punctuation. These combinations are shown as "<KeyA><KeyB>". For example, holding down the CONTROL key while tapping the BREAK key is shown as <CTRL><BREAK>.
- ***Sequential Activation***: If keys are to be pressed one at a time and released, then they will be separated by a comma. These key sequences are shown as "<KeyA>,<KeyB>". For example, holding down the ALTERNATE key while tapping the D key, followed by tapping the H key is shown as <ALT><D>,<H>.

Audio ToolBox Technical Capabilities

Audio ToolBox is the **only** native-mode, direct-to-disk, batch processor and converter for the MS-DOS and Microsoft Windows® environments. Native-mode means you can process compressed PCM instantly and retain fidelity without unnecessary conversions. Direct-to-disk capability lets you process files of any size. Why? So you can create perfect voice prompts for your telephony and multimedia applications.

Main Features:

- ***Industry Standard Formats:*** Convert to and from industry standard formats such as Dialogic 4 & 8 bit, C-ITU (CCITT) G.711 & G.721, NewVoice CVSD, Perception Technology CVSD, Multimedia Wave — and more — from 100 to 65,000 Hz!
- ***Fast Conversion:*** 10x conversion speed with normal filtering; better than 1x conversion speed with high fidelity filtering!
- ***Unlimited File Length:*** Our disk caching technology lets you convert any length message — yet uses as little as 400K of memory!
- ***High Fidelity Conversion:*** Optional FFT re-sampling interpolation filters; sixth order time domain and default second order anti-aliasing filters!
- ***Improved Dynamics:*** Use our dynamic compressor / expander for improved dynamic range!
- ***Volume Normalize / Adjust:*** Normalize the volume levels across multiple files; adjust the volume level up or down!
- ***Auto-Crop:*** Automatically remove silence from both ends of an audio file!

- ***Frequency Equalize:*** Programmable 32 band graphic equalizer!
- ***Foreign file formats:*** Specify starting byte position to skip custom file headers; set input file length for embedded data; select byte and bit ordering options for reading files from non-Intel computers!
- ***Index Chop & Pack:*** Separate indexed files for batch processing, recombine when complete!
- ***Index Rebuild:*** Add empty index entries to an existing file, merge voice or text into a new or existing index position, and rebuild damaged indexed files!
- ***Index List:*** Lists header, voice and text information for any indexed file!
- ***Sound Chop:*** Chop a single audio file into separate files using periods of silence as separators. Use this to separate a long recording session into individual files!
- ***Windows & DOS Users:*** Point and click to create the appropriate command line. Execute immediately or make a batch file, add wildcards for bulk conversion!
- ***Software Development Kit:*** Add conversion capability to your Windows & DOS audiotex applications with our ToolBox SDK!

Enhancements for Version 3.0:

PCMCvt

- Source and destination frequencies now default to those implied by PCM data type selection.
- Changed minimum source frequency to 100 Hz.
- Better equalizer command line support.
- Improved all filter characteristics.
- Improved CVSD fidelity.
- Improved clipping on all volume adjust operations.
- Improved speed on 386-class machines when dynamic compressor/expander is inhibited.
- Improved equalizer for all memory conditions.

- Improved DTMF filter.
- Improved IIR filter.
- Volume normalize default changed to 0.2% over range.
- Added frequency equalizer to audio conversion.
- Added 6th order filters to audio conversion.
- Fixed wildcard destination files for drive root directory usage.

SndChp

- Accepts destination directory argument like "\temp\.vox"
- Now supports all native PCM types.

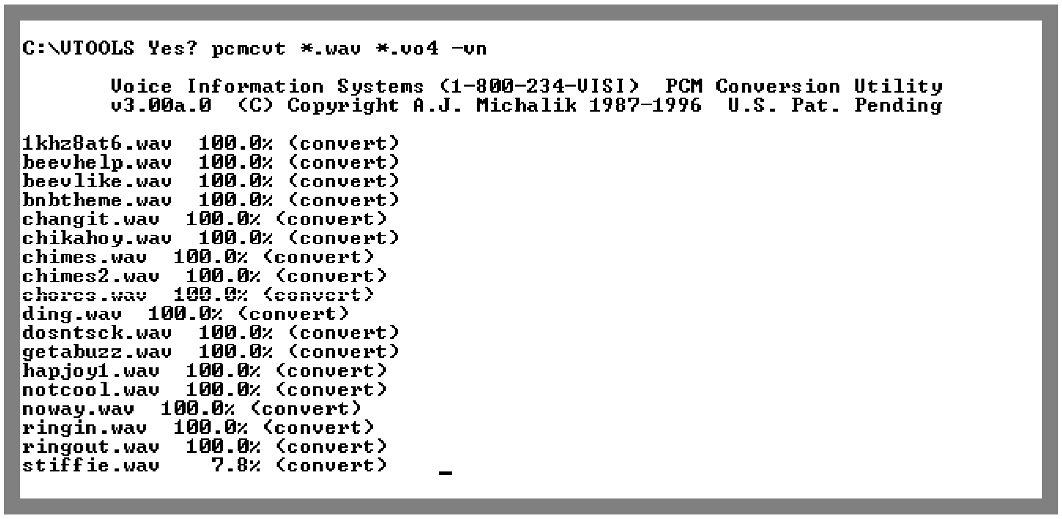
ToolBox Apprentice

- Added units to all frequency labels.
- Reduced overall size for VGA screen types.
- Audio Conversion: Volume normalize default changed to 0.2% over range.
- Audio Conversion: Source and destination data type saved in .INI file.
- Sound Chop: default destination file extension set to input file extension.
- Sound Chop: -f switch moved after destination extension.
- Setup: query before deleting old program group

ToolBox SDK

- Added SDK support for all ToolBox functions.
- Increased size of function blocks, added initialization functions.
- Added sample "C" programs for all SDK functions.

Sample Session



```
C:\VTOOLS Yes? pcmcvt *.wav *.vo4 -vn

       Voice Information Systems (1-800-234-VISI)  PCM Conversion Utility
       v3.00a.0  (C) Copyright A.J. Michalik 1987-1996  U.S. Pat. Pending

1khz8at6.wav  100.0% (convert)
beevhelp.wav  100.0% (convert)
beevlike.wav  100.0% (convert)
bnbtheme.wav  100.0% (convert)
changit.wav  100.0% (convert)
chikahoy.wav  100.0% (convert)
chimes.wav  100.0% (convert)
chimes2.wav  100.0% (convert)
chores.wav  100.0% (convert)
ding.wav  100.0% (convert)
dosntsck.wav  100.0% (convert)
getabuzz.wav  100.0% (convert)
hapjoy1.wav  100.0% (convert)
notcool.wav  100.0% (convert)
noway.wav  100.0% (convert)
ringin.wav  100.0% (convert)
ringout.wav  100.0% (convert)
stiffie.wav    7.8% (convert)    _
```

Audio ToolBox™ DOS Sample Session

Start-up Checklist

This chapter contains a step-by-step, hands-on introduction to using ***Audio ToolBox***. Since you will be working with the program, be sure that you have:

♦ Installed the ***Audio ToolBox*** software; Appendix A, ***Installation and Setup,*** tells you how to set up ***Audio ToolBox*** on your computer.

If you are planning to use ***Audio ToolBox Apprentice***, be sure that you have:

♦ Installed the ***Audio ToolBox Apprentice*** software; Appendix A, ***Installation and Setup,*** tells you how to set up ***Audio ToolBox Apprentice*** on your computer.

♦ Mastered basic Windows concepts; Chapter 3, ***Windows (Quick) Review*** discusses topics important for new Windows user.

What We'll Do

During installation we loaded some sample files onto your hard disk for you to use. We'll tell you how to read what is on your screen in both the DOS and Windows environments, and how to convert voice files from one format to another. We’ll also familiarize you with the various tools and command options available to you in ***Audio ToolBox*** and ***ToolBox Apprentice***.

Some customers prefer to use ***Audio ToolBox*** from the DOS environment, some from the Windows environment, and of course, some use both. If you prefer to use only the ***ToolBox Apprentice*** and work solely from the Windows environment, simply skip to the section named ***Audio Toolbox Apprentice***.

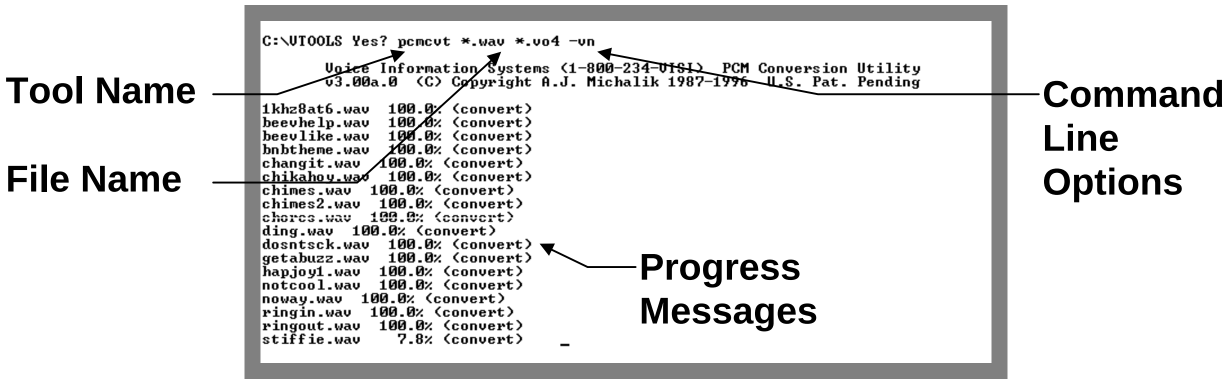
Running *Audio ToolBox* from DOS

First, call up a DOS session and move into the ***Audio ToolBox*** directory. For example, if you installed ***Audio ToolBox*** in the `c:\VTools` directory, enter:

```
C: <ENTER>
CD c:\VTools <ENTER>
```

This will tell DOS to make the `c:\VTools` directory your current working directory.

Audio ToolBox DOS Screen Description



```
C:\VTOOLS Yes? pcmcvt *.wav *.vo4 -vn

      Voice Information Systems (1-800-234-VISI)  PCM Conversion Utility
      v3.00a.0  (C) Copyright A.J. Michalik 1987-1996  U.S. Pat. Pending

1khz8at6.wav  100.0% (convert)
beevhelp.wav  100.0% (convert)
beevlike.wav  100.0% (convert)
bnbtheme.wav  100.0% (convert)
changit.wav  100.0% (convert)
chikahoy.wav  100.0% (convert)
chimes.wav  100.0% (convert)
chimes2.wav  100.0% (convert)
chorcs.wav  100.0% (convert)
ding.wav  100.0% (convert)
dosntsck.wav  100.0% (convert)
getabuzz.wav  100.0% (convert)
hapjoy1.wav  100.0% (convert)
notcool.wav  100.0% (convert)
noway.wav  100.0% (convert)
ringin.wav  100.0% (convert)
ringout.wav  100.0% (convert)
stiffie.wav    7.8% (convert)    _
```

The ***Tool Name*** appears immediately following the DOS command prompt, and is the name of the executable file that performs a particular function.

The ***File Name*** appears immediately following the ***Tool Name***, and is the name of the file that you wish to process.

The ***Command Options*** control the operation of the program, and are used to tell the particular tool how to perform its job. When run, ***Audio ToolBox*** checks the command line for these user-specified options.

Converting a File

As we said earlier, we're going to offer a few "just out of the box" exercises. With this short initiation you'll become familiar with ***Audio ToolBox***, and find you're having fun too!

PCMCvt lets you convert to and from Microsoft Multimedia Wave (8, 16 & MS ADPCM), linear 16 and unsigned 8 bit, Dialogic 4 and 8 bit plus other industry standard formats. **PCMCvt** lets you convert files recorded at 22 kHz, 11 kHz, 8 kHz and 6 kHz - and provides filters for proper re-sampling. Optionally choose volume adjustments, dynamic compression / expansion and fast or high fidelity filtering and re-sampling algorithms.

Convert the sample ***11 kHz 16 bit Multimedia Wave*** file PCMCVT.WAV into a ***Dialogic 6 kHz 4 bit ADPCM*** file by entering:

```
    PCMCvt Sample\PCMCvt.Wav AccTst.v04 <ENTER>
```

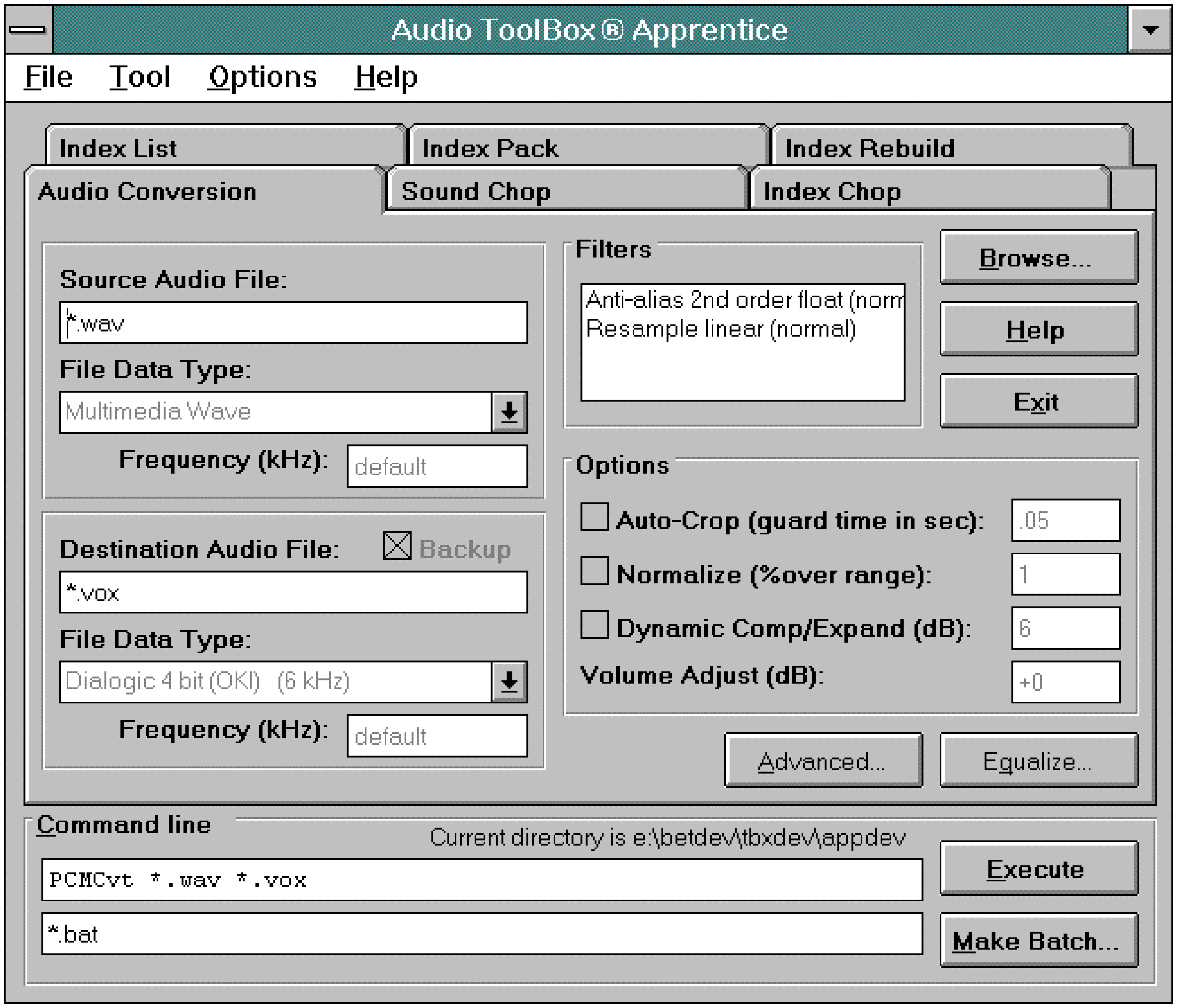
You have now created ACCTST.V04, a ***Dialogic 6 kHz 4 bit ADPCM*** audio file. You can play the file with a playback utility such as VISI's *Scribe Transcription Playback Utility*, or edit with a Dialogic compatible editor such as VISI's *VFEdit*® *Professional Prompt Editor*. Note that since this sample command uses the default conversion settings, you only specified the old and new file names with no additional command line parameters.

Now use the command line switches to convert the sample ***Dialogic 6 kHz 4 bit ADPCM*** file PAK001.VOX into an ***11 kHz 8 bit Multimedia Wave*** file by entering:

```
    PCMCvt Sample\pak001.vox AccTst.Wav -diDLG004 -fi6
    -doWAV008 -fo11.025 <ENTER>
```

You have now created ACCTST.WAV, an ***11 kHz 8 bit Multimedia Wave*** audio file. You may now play the file with a playback utility such as VISI's *Scribe Transcription Playback Utility*, or edit with a multimedia compatible editor such as VISI's *VFEdit*® *Professional Prompt Editor*. Note that since this sample command does not use the default conversion settings, you specified additional command line parameters along with the old and new file names.

Audio ToolBox Apprentice



Source Audio File: *.wav
File Data Type: Multimedia Wave
Frequency (kHz): default

Destination Audio File: *.vox [✓ Backup]
File Data Type: Dialogic 4 bit (OKI) (6 kHz)
Frequency (kHz): default

Filters
Anti-alias 2nd order float (norm
Resample linear (normal)

Options
☐ Auto-Crop (guard time in sec): .05
☐ Normalize (%over range): 1
☐ Dynamic Comp/Expand (dB): 6
Volume Adjust (dB): +0

Command line
Current directory is e:\betdev\tbxdev\appdev

```
PCMCvt *.wav *.vox
```

*.bat

Audio ToolBox Apprentice Sample Session

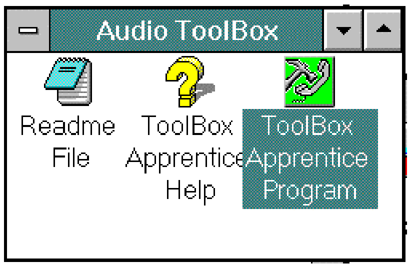
Audio ToolBox Apprentice is a Windows graphical user interface for the ***Audio ToolBox*** Digital Audio Processor & Converter. Developed at the suggestion of our customers, ***ToolBox Apprentice*** provides a simple way to see and change complex command lines with ease. With ***ToolBox Apprentice*** you can:

- Point and click to create the appropriate command line.
- Execute the command immediately.
- Create batch files to save the commands for later use.

If you prefer to use ***Audio ToolBox*** from the DOS environment and not use ***ToolBox Apprentice***, simply skip this section.

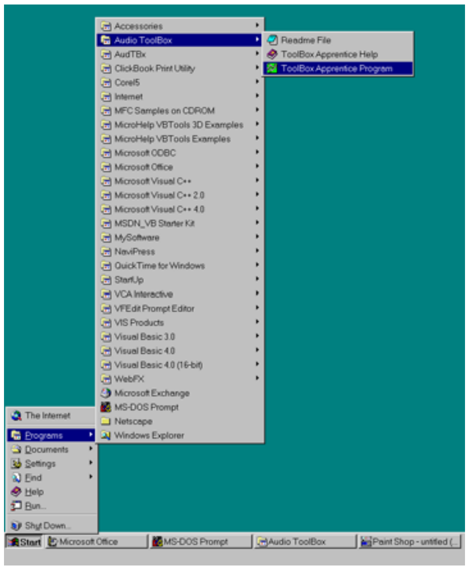
Starting ToolBox Apprentice

If you are running ***Audio ToolBox Apprentice*** under Microsoft Windows 3.x, simply double-click on the ***ToolBox Apprentice*** icon in the ***Audio ToolBox*** Group.



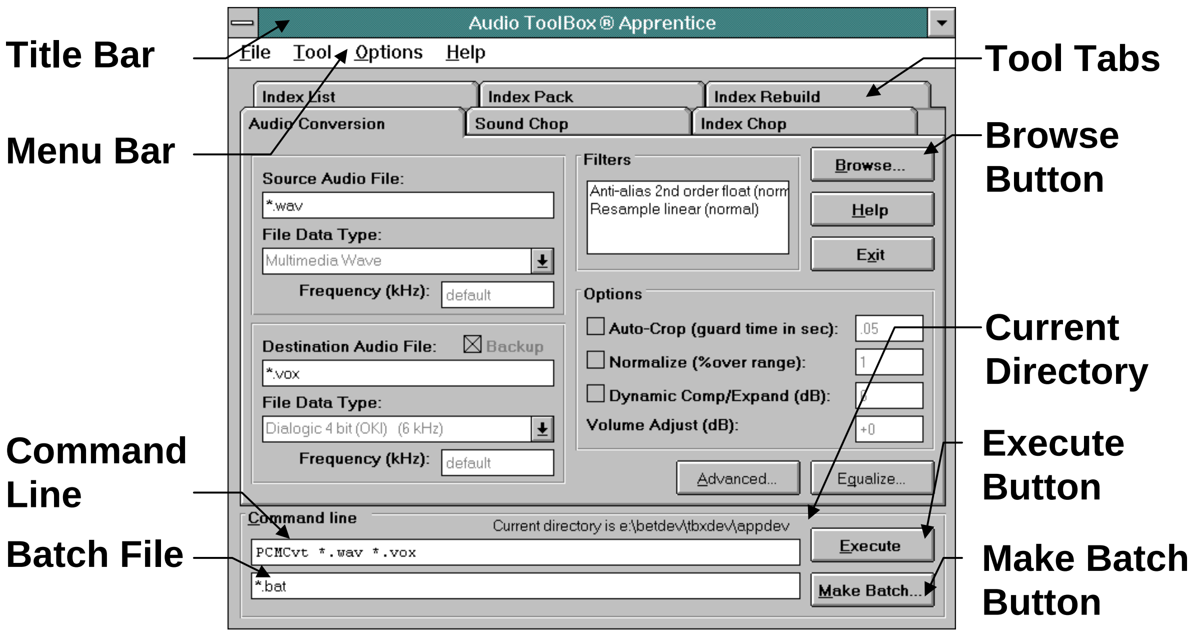
Windows 3.x ***Audio ToolBox*** Program Group

If you are running ***Audio ToolBox Apprentice*** under Microsoft Windows 95, use the Start Programs menu item to select the ***Audio ToolBox Program*** entry.



Windows 95 Start Menu

ToolBox Apprentice Screen Description



Source Audio File: *.wav
File Data Type: Multimedia Wave
Frequency (kHz): default
Destination Audio File: *.vox ☒ Backup
File Data Type: Dialogic 4 bit (OKI) (6 kHz)
Frequency (kHz): default

Filters
Anti-alias 2nd order float (norm
Resample linear (normal)

Options
☐ Auto-Crop (guard time in sec): .05
☐ Normalize (%over range): 1
☐ Dynamic Comp/Expand (dB): 0
Volume Adjust (dB): +0

Command line
Current directory is e:\betdev\tbxdev\appdev

```
PCMCvt *.wav *.vox
```

*.bat

The ***Title Bar*** appears across the top of the main window, and displays the ***Audio ToolBox Apprentice*** program name.

The ***Menu Bar*** appears directly below the title bar, and provides pull-down menus for processing functions. Menu commands may be selected using the keyboard or mouse. These functions are covered in detail in the chapters that follow.

The ***Tool Tabs*** appears directly below the Menu Bar, and provides tabs for rapid access to tools. These tabs may only be selected using the mouse.

The ***Browse Button*** is active whenever the cursor is placed in a field that requires a file name. This button displays a dialog box that allows you to select a file from the standard Windows list of drives, directories, and files.

The ***Command Line*** displays the current command line options so that you can watch the ***Audio ToolBox Apprentice*** actually build the request as you work.

The ***Batch File*** text box allows you to specify the name of a batch file in which to store the current command for later use.

The ***Current Directory*** shows you the name of your current default directory. If you specify a source or destination file with a partial directory specification, ***Audio ToolBox*** will look for these files in your current directory.

The ***Execute Button*** causes the current command line to be executed.

The ***Make Batch Button*** saves the current command line for future use.

Converting a File

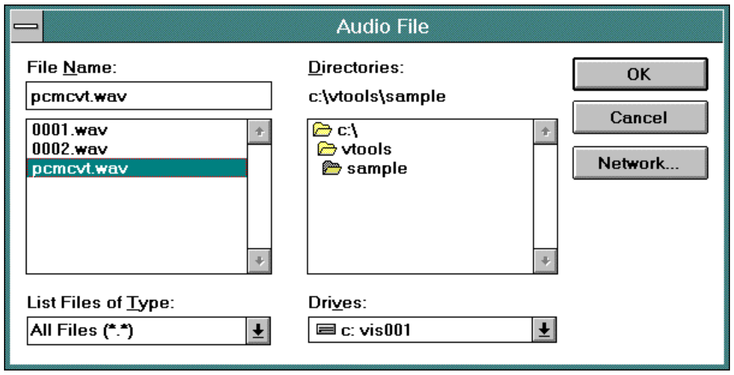
As we said earlier, we're going to offer a few "just out of the box" exercises. With this short initiation you'll become familiar with ***Audio ToolBox Apprentice***, and find you're having fun too!

When you installed ***Audio ToolBox***, the installation program asked you to select a default file type. Since we selected a file type of "Dialogic" during our installation, our screen displays may look slightly different (but not very much different) from yours. Use the drop down menus to change the file data type selections if you want your screen displays to exactly match ours.

Audio ToolBox lets you build standard command lines from the Windows environment. Let's repeat the DOS audio conversion example from above using the ***ToolBox Apprentice***.

Use ***ToolBox Apprentice*** to build the command line switches to convert the sample ***11 kHz 16 bit Multimedia Wave*** file PCMCVT.WAV into a ***Dialogic*** 6 kHz 4 bit ADPCM file by:

- Selecting the source audio file by positioning your cursor in the "Source Audio File" field and entering "SAMPLE\PCMCVT.WAV". Or use the "Browse" button to select a file from the list of files.



File Name: pcmcvt.wav
0001.wav
0002.wav
pcmcvt.wav
Directories: c:\vtools\sample
c:\
vtools
sample
List Files of Type: All Files (*.*)
Drives: c: vis001

Source Audio File Browse Dialog Box

- Selecting the destination audio file by positioning your cursor in the "Destination Audio File" field and entering "ACCTST.V04".

You have now created the following command line:

```
PCMCvt c:\vtools\sample\pcmcvt.wav AccTst.v04
```

Click on the ***Execute Button*** and you will convert PCMCVT.WAV into ACCTST.V04, a ***Dialogic 6 kHz 4 bit ADPCM*** audio file. You can play the file with a playback utility such as VISI’s *Scribe Transcription Playback Utility*, or edit with a Dialogic compatible editor such as VISI’s *VFEdit*® *Professional Prompt Editor*.

Now use ***ToolBox Apprentice*** to build the command line switches to convert the sample ***Dialogic 6 kHz 4 bit ADPCM*** file PAK001.VOX into an ***11 kHz 8 bit Multimedia Wave*** file by:

- ♦ Selecting the source audio file by positioning your cursor in the “Source Audio File” field and entering “PAK001.VOX”. Or use the “Browse” button to select a file from the list of files.
- ♦ Selecting the source “Data Type” by clicking the drop down menu and selecting “Dialogic 4 bit (OKI) (6 kHz)
- ♦ Setting the source frequency to 6 kHz by positioning your cursor in the source “Frequency” field and entering “6”.
- ♦ Selecting the destination audio file by positioning your cursor in the “Destination Audio File” field and entering “ACCTST.WAV”
- ♦ Selecting the destination “Data Type” by clicking the drop down menu and selecting “Wave Multimedia Format 8 bit (11.025 kHz).
- ♦ Setting the destination frequency to 11.025 kHz by positioning your cursor into the destination “Frequency” field and entering “11.025”.

You have now created the following command line:

```
PCMCvt c:\vtools\sample\pak001.vox Acctst.wav
-diDLG004 -fi6 -doWAV008 -fo11.025
```

Click on the ***Execute Button*** and you will convert PAK001.VOX into ACCTST.WAV, an ***11 kHz 8 bit Multimedia Wave*** audio file. You may now play the file with a playback utility such as VISI’s *Scribe Transcription Playback Utility*, or edit with a multimedia compatible editor such as VISI’s *VFEdit*® *Professional Prompt Editor*.

Browse Button

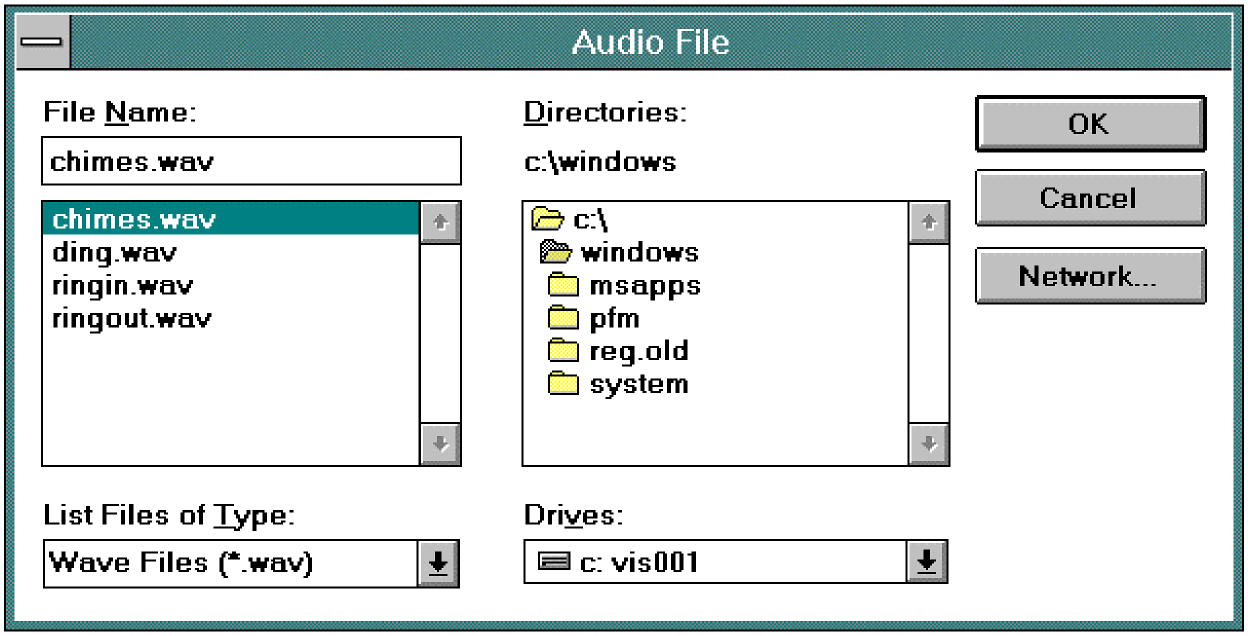
The **Browse** button is the gateway into and out of ***Audio ToolBox*** and your disk storage areas; it retrieves and displays previously saved audio files, using filenames and pathnames which you specify. You can think of filenames and pathnames as mailing addresses pointing to where you saved your recordings, allowing the disk drive to access them for you. The files themselves are the actual recordings of your audio information, in one's and zeroes on magnetic media.

When you select the **Browse** command, the file selection dialog box appears. Here you can select the name of the file you wish to convert.

This dialog box looks like the usual Windows File Open dialog. To select a file from the list of files, type the filename in the File Name text box; you can also double-click a file name with the mouse or use <TAB>, position the highlight bar, and press <ENTER>.

If you type in only a file name, ***Audio ToolBox*** assumes that the file is located in the directory shown to the right of the file name. To specify a different directory, move the cursor to the File Name text box and enter the new pathname. Press <ENTER> and a new list of files will appear.

Press OK or <ENTER> and ***Audio ToolBox*** will display the sound file name in the **Source Audio File** text box. Press Cancel to return to the main screen.



The Browse button displays an existing file.

File Name Characters

File names are limited to a total of eight characters and can be any combination of letters and / or numbers. The program does not differentiate between capital and lower case letters. You can have as few as one character, but you must have at least one character.

The following characters can be used in a file name:

```
a to z, 0 to 9, ! @ # $ % ( ) - ‘ { } ^
```

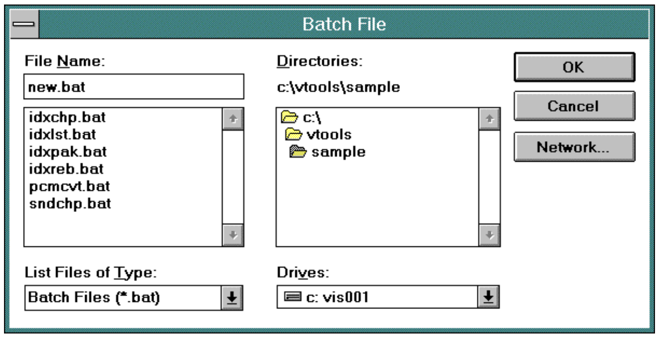
A period can only be used to separate the first part of the file name from the file name extension.

The following characters cannot be used to create a file name:

```
* + = [ ] : ;” ~ < > ? \ / , or a space.
```

Make Batch Button

Let’s say that you want to save the current command line so that you can have a friend or an automated program run it at another time. Click the ***Make Batch Button*** to save the current command line in a batch file for future use.



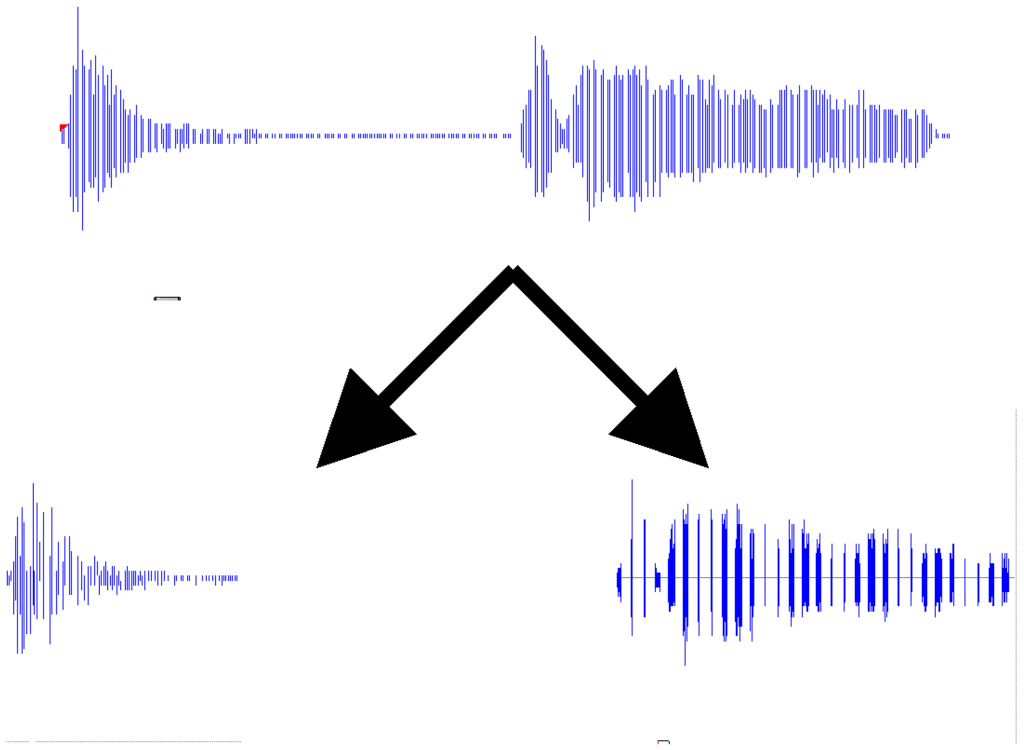
Make Batch Dialog Box

Exit Button

You've now completed the basic exercises for using ***ToolBox Apprentice***. Use the ***Exit Button*** to end the program, or continue with the sample session for using additional tools.

Sound Chop

Audio ToolBox allows you to record a continuous stream of prompts into a single file, and then to automatically break the single file into individual files. **SndChp** separates the single file into individual files using periods of silence as separators. The resulting individual files can then be converted, volume normalized, and packed or converted using other ***Audio ToolBox*** utilities.


Chopping a file at a silence point into separate files

Running SndChp

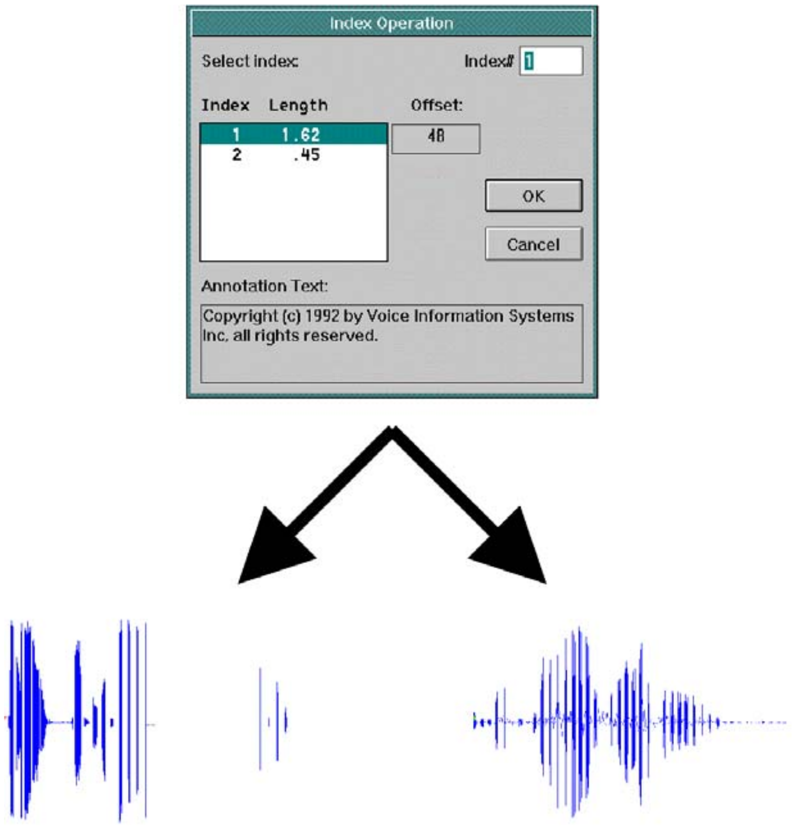
Use the DOS command line or ***ToolBox Apprentice*** to create and execute the following command:

```
    SndChp PCMCvt.Wav <ENTER>
```

You have now created 0001.WAV and 0002.WAV (as shown above), which contain the active audio portions of the original file. You may now process the file further using other ***Audio ToolBox*** utilities.

Index Chop

Audio ToolBox supports both pure (raw amplitude data) and indexed format files. Pure files contain only digitized sound; indexed files contain groups of digitized sound, plus format and annotation text. **IdxChp** allows you to "chop" indexed files into their components: pure audio and text files that can then be batch processed, converted and edited.



Select index: Index# 1

Index	Length
1	1.62
2	.45

Offset: 48

Annotation Text:

Copyright (c) 1992 by Voice Information Systems Inc. all rights reserved.

Chopping an indexed file into its component parts

Running IdxChp

Use the DOS command line or ***ToolBox Apprentice*** to chop the sample ***Dialogic 6 kHz 4 bit ADPCM indexed*** file IDXTST.VAP into its component pure audio and text files:

```
    Idxchp IdxTst.vap <ENTER>
```

You have now created 0001.VOX, 0001.TXT, and 0002.VOX, a set of pure ***Dialogic 6 kHz 4 bit ADPCM*** audio files and their associated annotation text files. You can play the audio files with a playback utility such as VISI’s *Scribe Transcription Playback Utility*, or edit with a Dialogic compatible editor such as VISI’s *VFEdit*® *Professional Prompt Editor*. The text files can be viewed and edited with any standard DOS editor.

Index List

Indexed audio files have a built-in “Table of Contents” — a great time-saver in finding which audio segment you want. Each audio segment can have an attached annotation. The **IdxLst** lets you view and list the header and content information of indexed files.

Running IdxLst

Use the DOS command line or ***ToolBox Apprentice*** to list the contents of the sample ***Dialogic 6 kHz 4 bit ADPCM indexed*** file IDXTST.VAP and route the output to a text file for future reference:

```
IdxLst IdxTst.vap >list.txt <ENTER>
edit list.txt <ENTER>
```

You have now created LIST.TXT, a text file containing header, audio extents and annotation text information for IDXTST.VAP. You can review and edit the information file with any standard text editor.

Index Pack

After the files have been processed individually, as either audio or text data, you then need to combine the individual files back into a single indexed file. Use **IdxPak** when you want to “pack” pure audio and text files into an indexed file.

Running IdxPak

Use the DOS command line or ***ToolBox Apprentice*** to pack the sample ***Dialogic 6 kHz 4 bit ADPCM pure*** files PAK00?.VOX and PAK00?.TXT into a composite indexed file:

```
IdxPak Pak00?.vox IdxTst.vap -t*.txt <ENTER>
```

You have now created IDXTST.VAP, a composite indexed file containing groups of digitized sound, plus format and annotation text. You can load the file into your indexed file compatible voice response system or edit it with a Dialogic compatible editor such as VISI’s VFEdit® Professional Prompt Editor.

Index Rebuild

You may find, over time, that the indexed files you are using become fragmented (i.e., data is scattered throughout), or that you need to change the size of the file (either shorter or longer), or need to perform and integrity check after a hard disk failure. The **index rebuild** tool lets you perform these types of maintenance operations on indexed files. Use this tool when you need to lengthen, shorten, replace entries and "rebuild" indexed files.

Running IdxReb

Use the DOS command line or ***ToolBox Apprentice*** to rebuild the sample ***Dialogic 6 kHz 4 bit ADPCM indexed*** file IDXTST.VAP and add 98 new empty index entries to the end (indicated by 9999); reset the frequency specified in the header to 6053.

```
    IdxReb IdxTst.vap -i98 -@9999 -f6053 <ENTER>
```

You have now modified IDXTST.VAP, increasing the total entry count to 100 indexes and changing the frequency specified in the header to 6053 Hz.

Working with decibels (dB)

Some of the ***Audio ToolBox*** tools use parameters specified in decibels. For example, the audio conversion tool lets you perform a volume adjustment as part of the conversion process. To understand how to set these parameters, you need to understand this unit of measurement.

The decibel (dB) is a unit of measure for expressing the ratio between two amounts of electric voltages. The decibel is used because of the wide range in sensitivity of the human ear. In our usage one dB equals 20 times the common logarithm of the ratio of the old signal voltage versus the new signal voltage.

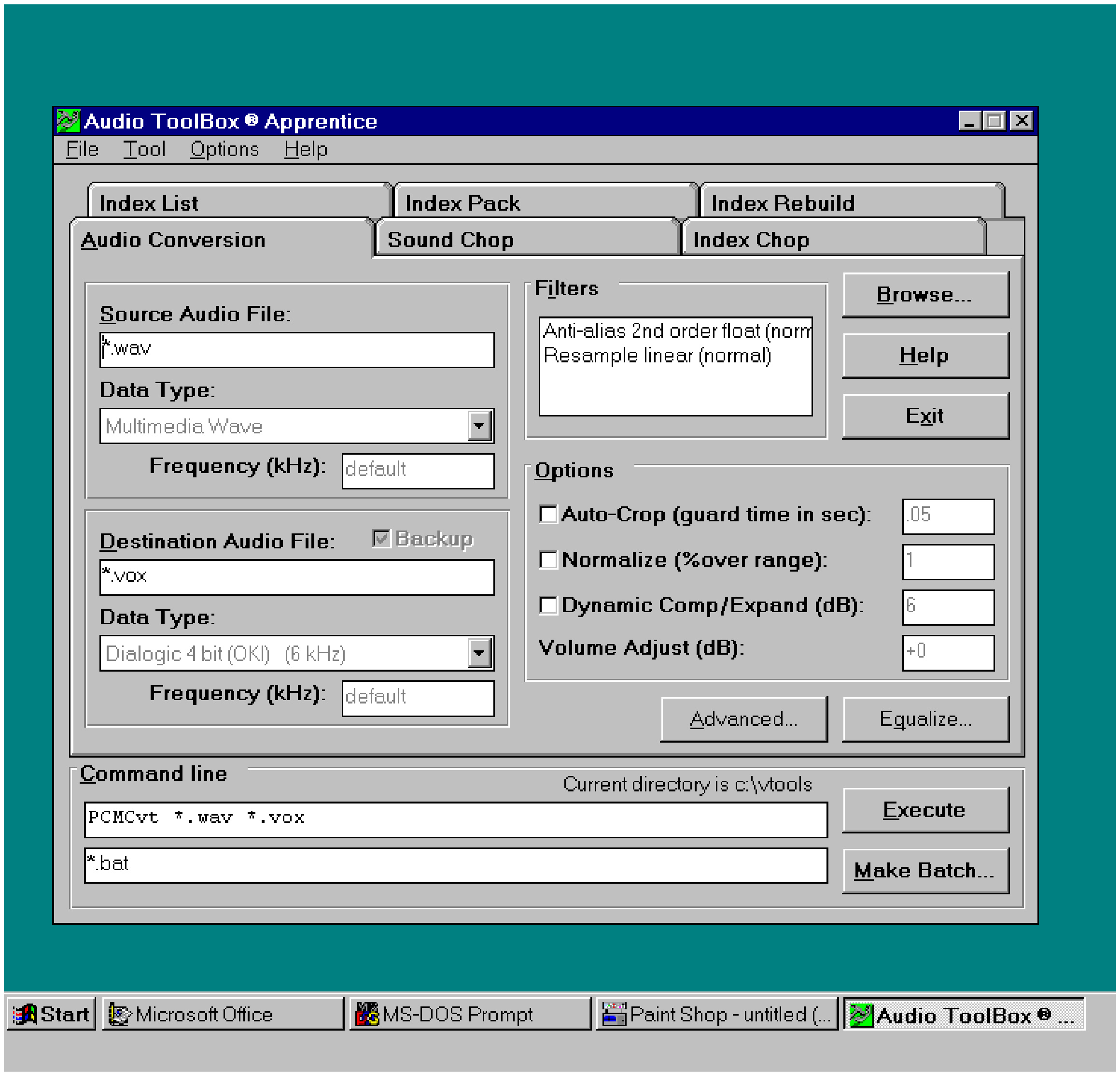
Since the human ear is sensitive to the power of the signal, the intensity of the sound is equal to the square of the voltage level. Thus doubling the intensity of the sound means an increase of a little more than 3 dB.

For example, an audio segment that is to be made four times as loud would increase by:
dB = 20 * log(New Voltage/ Old Voltage)
dB = 10 * log(New Power / Old Power)
dB = 10 * log (4/1)
dB = 10 * +.60
dB = 6.0

Similarly, an audio segment that is to be made one-fourth as loud would decrease by:

dB = 20 * log(New Voltage/ Old Voltage)
dB = 10 * log(New Power / Old Power)
dB = 10 * log (1/4)
dB = 10 * -.60
dB = -6.0

Windows (Quick) Review



Source Audio File: *.wav
Data Type: Multimedia Wave
Frequency (kHz): default

Destination Audio File: *.vox　　☑ Backup
Data Type: Dialogic 4 bit (OKI)　(6 kHz)
Frequency (kHz): default

Filters
Anti-alias 2nd order float (norm
Resample linear (normal)

Options
☐ Auto-Crop (guard time in sec): .05
☐ Normalize (%over range): 1
☐ Dynamic Comp/Expand (dB): 6
Volume Adjust (dB): +0

Command line　　Current directory is c:\vtools

```
PCMCvt *.wav *.vox
```

*.bat

Windows 95 Sample Screen

We presume you have some familiarity with Windows and how to operate the program, but just in case you're a new user (or maybe just a little rusty), we offer this very quick review. This explanation is anything but exhaustive, and if you find you need more thorough instruction, you may want to consult any number of useful texts. Also, classroom instruction is available from many local schools. Those resources failing, you may find it helpful to contact Microsoft Corporation directly.

Arcane Language

Mindful of the difficulties we encountered when we were first learning to cope with computers and their brethren, we’ve tried to define—and make clear—all terms that have computer-peculiar meanings.

Is Your System Ready to Work?

At this point we have to assume that all hardware and software are properly installed, connected, and running correctly. We also assume that you have Microsoft Windows running on your system. If not, please refer to your Microsoft Windows installation guide or contact your system administrator.

Windows Menus and Commands

Windows supports a large number of pointing devices such the mouse, trackball, touch pad, and even a stubby little control stick. We’ll use the generic “mouse” to cover all these possible devices.

Although Windows can be used with the keyboard only, the mouse is usually more convenient and often faster than arrow keys and other methods. The mouse is an important factor in the success and ease of use of Windows and applications. Users often find that a combination of both keyboard and mouse works well for many commands. Here we’ll cover both keyboard and basic mouse commands.

There are three ways to use the keyboard to select menus on the menu bar and choose commands from the menus. First you’ll be introduced to the basic method that works with any Windows application. Once you’re familiar with the basic method, you’ll learn the direct-access method, which uses the underlined letters you see in the command and menu names. Finally, you’ll learn to use accelerators to quickly access a menu command.

The following sections discuss selecting menus, commands, and objects using both the mouse and keyboard.

Using the Mouse

Microsoft Windows can be used with either a single-button or a multiple-button mouse. If you have a mouse with more than one button, use the leftmost button.

The following definitions will help you begin to use your mouse:

- ***Pointer Display***: The program controls the shape of the displayed pointer, and you move it around with the pointer control. Shapes range from the familiar arrow to the vertical I-beam (or sometimes called a “toothpick”) used in text. Pointers can also be shaped like a little hand with an extended finger. (We’re sure the folks at Microsoft had fun with that one!)
- ***Point***: Move the mouse until the tip of the pointer rests on the thing you wish to work on.
- ***Click***: The basic actuation of the pointer is the *click*. **Point** the pointer over a selected object on the display, then **click** (quickly press and release) the left mouse button to execute a command or select an object. To click on an object (an icon or a menu name, for example) means to point to that object on the screen and click the mouse button.
- ***Drag***: **Point** the pointer over the selected object on the display, then **hold** down (don’t click) the button and move the pointer to a new location. **Release** the button and the dragged object remains where it’s placed.
- ***Double-click***: Same operation as the click, except the button is rapidly pressed and released twice. This is used to select an object.
- ***Cancel***: To cancel a command with the mouse simply point to a blank portion of the screen outside the menu bar, and **click** the mouse button.

To select menus and choose commands with the mouse, simply click the menu item you want on the menu bar, then click on the command you want from the drop-down menu that appears under the menu item. To cancel a command, click on a blank part of the screen outside the menu.

Basic Keyboard Menus and Commands

The basic method of selecting menus and choosing commands uses the ARROW keys to move across the Windows menu bar and up and down the menus.

- The first step is always the same: press the <ALT> key. Then press the <RIGHT ARROW> or <LEFT ARROW> keys to select a menu on the menu bar.
- To choose commands from the menu, you use the <UP ARROW> and <DOWN ARROW> keys.

To choose a command, therefore, you do the following:

- ♦ Press the <ALT> key.
- ♦ Press the <RIGHT ARROW> or <LEFT ARROW> key to select a menu on the menu bar.
- ♦ Press the <ENTER> key to display the menu.
- ♦ Press the <UP ARROW> or <DOWN ARROW> key to move up or down the command list.
- ♦ Choose a command from the menu, using the <UP ARROW> or <DOWN ARROW> key to select the command name, then press the <ENTER> key.
- ♦ Press the <ESCAPE> key to cancel the menu selection.

Direct-Access Keyboard Menus and Commands

With the direct-access method, you can select any of the menus on the menu bar by using the <ALT> key with the underlined letter in the menu name. For example, to take a look at a typical File menu:

Press the <ALT> key.

Press the <F> key, the underlined letter in the File-menu name.

The File menu drops down from the menu bar.

Take a look at the other menus listed in the menu bar. Remember to press the <ESCAPE> key to clear a menu from the screen.

Now you can use the commands in the menus. Just press the letter that is underlined in the command’s name.

Accelerator Key Commands

With the accelerator keys, you can select certain common commands with a single key combination. For example, to take a look at a typical File menu accelerator:

Press the <ALT> key.

Press the <F> key, the underlined letter in the File-menu name.

The File menu drops down from the menu bar.

Note that some frequently used menu commands are followed by a key combination, such as “Exit Alt+F4”. Take a look at the other accelerators listed on the menu. Remember to press the <ESCAPE> key to clear a menu from the screen.

Now you can use the accelerators listed in the menus. Just press the keyboard sequence listed after the command’s name.

♦ Reference Guide

Audio ToolBox™ affords wide flexibility in handling data with the six main processing tools: **Audio Conversion**, **Sound Chop**, **Index Chop**, **Index List**, **Index Pack**, and **Index Rebuild**. For each main tool there are command options yielding numerous variations in ways the files and their included data can be manipulated.

This guide covers specific information relating to the use of each tool. In addition, we have included a brief introduction to the principles behind digital audio.

About Digital Audio

As recording engineers have known for years, it takes a little art, and more science, to capture sound and store it for later playback. You may already have tried to do some recording of your own, so you've probably noticed that home recordings often don't sound as good as your favorite compact disc. Professional recordings typically sound clean and free of both distortion and background noise.

If you'll settle for fuzzy tones, distortions, random pops, thumps, buzzes, and hums, then just about any old microphone and recording technique will work. On the other hand, if you prefer digital recordings that are pleasing to the ear, then you'll want to acquaint yourself with some essential concepts. But before jumping into digital audio, it's important to have an understanding of the physics and the perception of sound, as well as the process by which sound can get into and out of your computer.

Like a pebble in a pond (...sort of)

Sound consists of vibrations, either in the air or in some other medium. When a sound is created, waves of vibration spread out from a source, much the way waves spread out on the surface of a pond when you throw a pebble into it.

Waves have an up-and-down motion in pools of water. Sound waves, however, consist of variations in *air pressure*. The "crest" of each wave is a region where the air molecules are packed more closely together, and the "trough" is a region where the molecules are farther apart. This idea is illustrated in Figure 1.

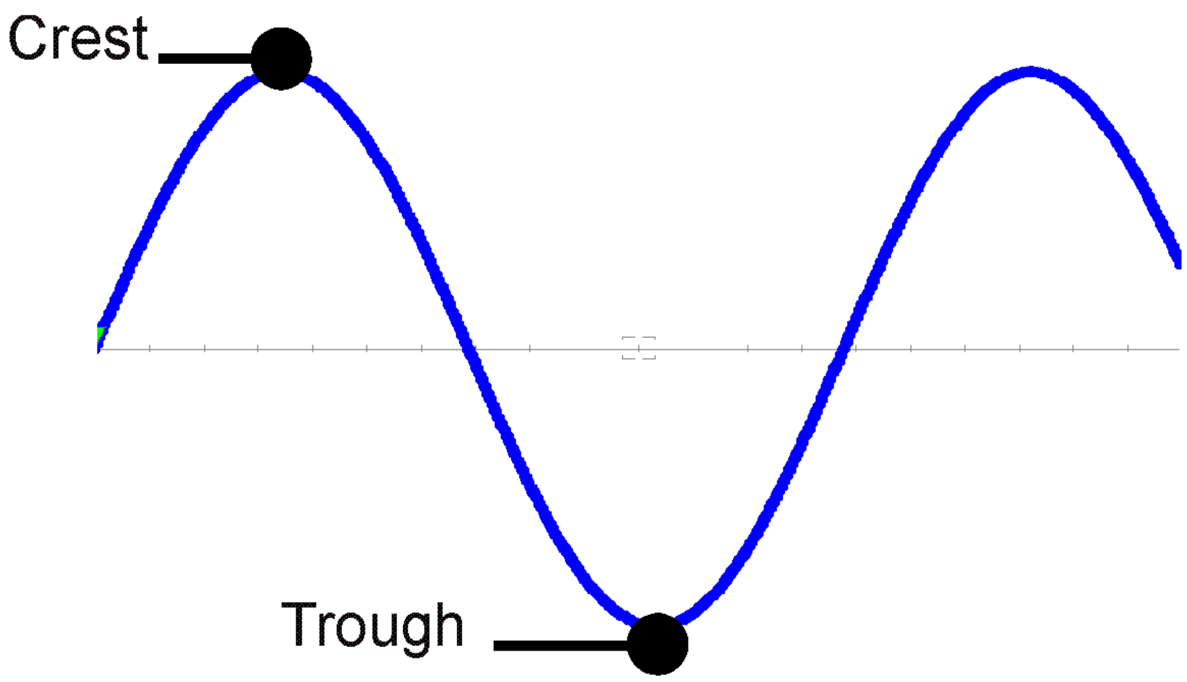


Figure 1: Sound consists of compressed (crest) and uncompressed (trough) air.

Waves are busy, but don’t go anywhere

In a pond, the waves travel outward, but the water itself doesn’t go anywhere. All that happens is that the water molecules near the surface of the pond move up and down. You can see this if you float a dry leaf on the surface and then toss in a pebble. The leaf will not travel away from the point where the pebble entered the water; it will only bob up and down. In the same way, when sound waves travel outward from a sound source, the air molecules only press against one another while staying pretty much where they are.

Sound waves travel considerably faster than the waves in a pond: About 1,000 feet per second in air. And, if you’ve ever watched a marching band outdoors, you’ve probably noticed that you can see the cymbal players’ cymbals crash together well before you hear them. That’s because light travels a *lot* faster than sound. (About 8,900,000 times faster!)

Consider a loudspeaker. When it starts to make sounds, the speaker diaphragm is first pushed outward, compressing the air molecules. Then the diaphragm is pulled back, reducing the air pressure momentarily. As it is moved forward and backward again and again, sound is generated. The key question is this: How quickly does the sound source move back and forth? If it vibrates rapidly, the peaks of the sound waves will be close together (high notes); if it vibrates more slowly, the peaks will be farther apart (low notes), as in Figure 2.

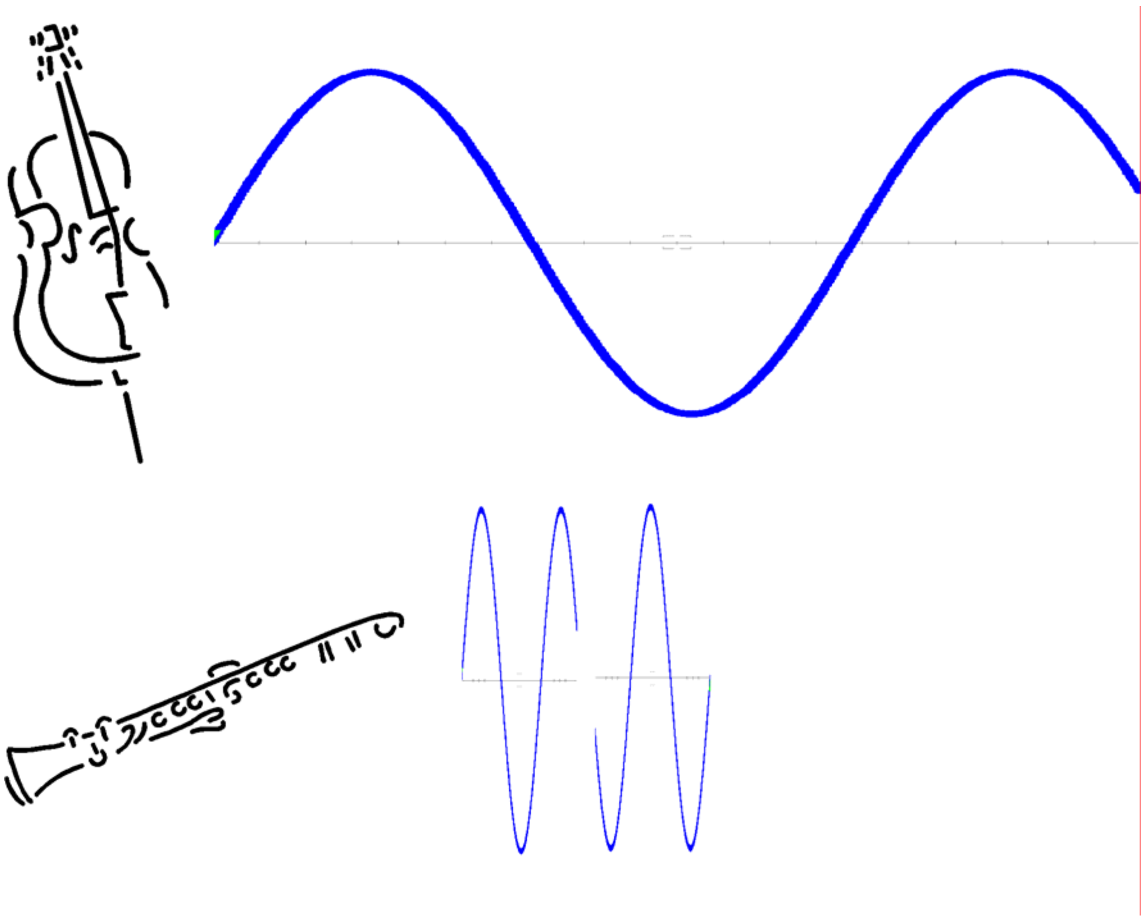


Figure 2: The bass viola has a low frequency and the clarinet has a high frequency.

Our ears discern subtleties

Human ears can very easily detect how closely spaced sound waves are when they arrive at our eardrums. When the sound source is vibrating rapidly, we say that the sound has a high *frequency* (high note), because lots of waves reach our ears in a short period of time. When the vibrations are slow, the sound has a low frequency (low note), because fewer waves strike our eardrums in the same amount of time. A clarinet, for example, produces higher frequency sound, while a bass viola produces much lower frequency sound.

Frequency is measured in cycles per second. The technical term for this measurement is *hertz* (abbreviated Hz). One cycle per second equals one hertz. The low range of human hearing is about 20 Hz; at the upper end of the range, children and adults with acute hearing can hear sounds as high as 20,000 Hz. (abbreviated as 20 kHz. The “k” is for *kilo*, or “thousand.” (Since you use computers, you probably knew that already.)

The range of human hearing, then, is usually considered to be about 20 Hz to 20 kHz. Many adults, however, have a reduced ability to hear high frequencies. They may not be able to perceive sounds above the 12 kHz to 15 kHz range.

Music

Frequency and musical pitch are closely related. Each musical note consists of vibrations at a specific frequency. For example, when you play a Middle C, you’ll hear a sound whose frequency is almost exactly 261.653 Hz. The A above Middle C, which is often used as a tuning reference by orchestras, vibrates at 440 Hz. (Hence, it’s called A-440.)

The musical scale sounds the way it does because the frequency doubles each time pitch rises by an octave. For example, the A directly below Middle C has a frequency of 220 Hz, and the next A below it vibrates at 110 Hz. Moving up the keyboard, the C above Middle C has a frequency of about 523.3 Hz, the next C has a frequency of 1,046.6 Hz. But if a clarinet, a bass viola, and a banjo each sound a note whose pitch is Middle C, how is it that our ears can instantly tell which instrument is playing?

Overtones

Consider the body of a banjo. When a note is played on a banjo, everything vibrates: the front of the banjo vibrates, the sides and back vibrate, the neck and fingerboard vibrate, and so on. The sound of a banjo, then, doesn’t consist of a pure sound wave at a single frequency. Instead, each part tends to vibrate in a different way, and the sound produced by the instrument is a composite, or blend, of a number of different vibrations at different frequencies. If the sound of the banjo is recorded and examined on a computer screen, it won’t look at all liked the simple sound waves shown in Figure 2. It will have a much more complex shape. Most sounds in the real world exhibit complexities of this kind.

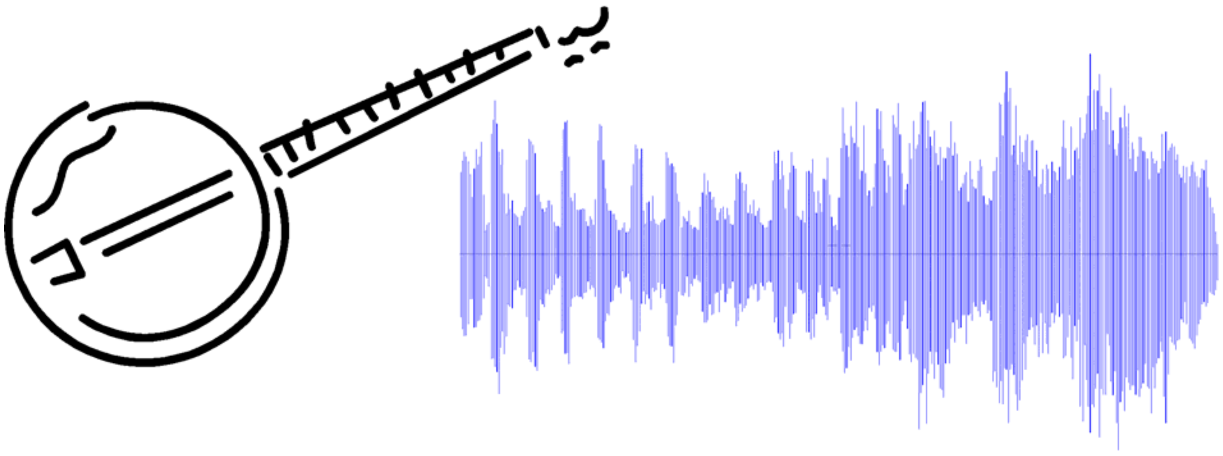


Figure 3: Most sounds in the real world are complex waves.

A mathematically pure tone is called a *sine wave*. Scientifically, it's possible to analyze a complex wave as being the sum of a number of different sine waves, each with its own frequency and amplitude (loudness). This is called a *Fourier* analysis" (pronounced"FOOR-ee-aye") analysis. The body of a banjo produces a complex wave containing a number of separate tones at different frequencies, all at the same time.

These are called *overtones*, and virtually all sounds in nature contain numerous overtones. One way our ears can instantly tell the sound of a banjo from the sound of a clarinet or bass viola is by noticing the relative loudness of the overtones at various frequencies.

Usually, the loudest of the overtones is the one vibrating at the nominal pitch of the note (440 Hz if the violinist is playing an A above Middle C). This frequency is called the *fundamental*.

With this understanding of what sound is, and what happens when it's propagated through the air, let's look at the process of capturing it on your computer.

Capturing Sound

A microphone is a form of *transducer*, a device that can take energy in one form and translates it into another form. Changes in air pressure arrive at a microphone, and are translated into changes in electrical voltage. (During playback, a loudspeaker, driven by electrical voltage, transforms the voltage back into air pressure—so a loudspeaker is a transducer too.)

Mounted inside a microphone is a small, thin, sensitive piece of material called the diaphragm. As sound waves strike the diaphragm, it vibrates at the same frequencies as the sound. The diaphragm, and its associated parts, translate these sound movements into fluctuating voltages. The microphone must be built with precision so it can be as sensitive as possible to slight sound vibrations. Professional recording studios think nothing of spending $1,000 or more for a good microphone. You may not want to rely on the free microphone shipped with your sound card. Many of these microphones are for recording sounds to jazz up your computer operating system, but that's about it.

Analog to Digital Conversion

A microphone works much like our ear. When sound strikes the diaphragm of a microphone, it produces an electrical wave that is analogous to the air pressure that produces the sound. Because of this correspondence between the air pressure and the voltage coming out of the microphone, the wave is termed an *analog* signal. Analog is used to characterize some common types of audio components, such as tape recorders, etc., to distinguish that equipment from the kind that incorporates digital circuitry.

Computers are designed to be operated in the digital realm, where everything is a series of **on** or **off** voltages, formatted as bits. To store a sound in the computer we must convert the continuously changing analog audio signal into digital data. The circuitry that transforms data from analog to digital, and vice-versa is termed the *DAC* (digital to analog converter).

At regular intervals a DAC instantaneously freezes the audio signal voltage and holds it steady while another circuit selects the binary code that most closely represents the sampled voltage. The DAC outputs a number in binary format that represents the input signal at any given instant in time.

Digital-to-analog conversion (for playback) is the exact opposite. In digital-to-analog conversion, the digital data is converted to a continuously changing series of voltage levels. The shape of this continuously changing stream of voltage levels approximates the shape of the original wave. This signal is then passed through a low-pass filter, which removes the digital “switching noise.”

Once in digital form, the audio is essentially immune to degradation caused by system noise or defects in the storage or transmission medium (unlike older analog systems). The digitized audio signal is easily stored on a hard disk drive, where is can be kept indefinitely without loss of fidelity.

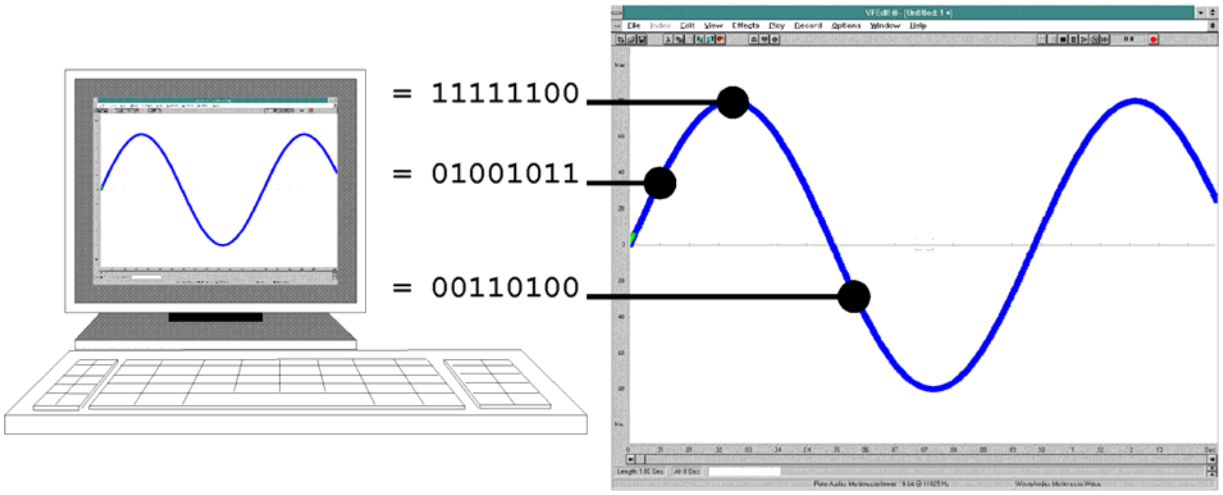


Figure 4: Analog to Digital Conversion

Sampling Resolution

Sampling resolution refers to the number of discrete levels that are used in the analog-to-digital (and digital-to-analog) conversion processes.

Sampling resolution is measured in *bits*, which refer to the amount of memory required to store each individual sample. The number of different values that can be stored in a collection of bits is equal to 2 raised to the “bit’th” power. For example, an eight-bit sample is digitized to 2^8, or 256 different levels. A sixteen-bit sampling system, on the other hand, senses 2^{16}, or 65,536 different levels.

Obviously, the more bits of resolution that are used, the more closely the sampled signal will represent the analog signal, which has an essentially infinite resolution. However, higher resolutions require more storage, so some tradeoffs must be made. Generally, 8 bits is accepted as the lowest sampling resolution that can be used to obtain reasonable results, while 16-bit resolution is preferred for professional applications.

The resolution of a sampling system is almost always determined by the hardware. The minimum resolution permitted for multimedia hardware is eight bits. Compact disks and digital audio tapes use 16-bit samples, although many playback units only use 14 bits in their output circuitry. The telephone network typically uses 14 bits, but with an enhancement called *compression*.

Compression

When the DAC outputs a binary number representing the input signal, this number must be stored in a convenient form for later retrieval. This conversion of a binary number for storage purposes is called *encoding*.

Audio encoding techniques can be broadly categorized into two classes: those for encoding analog waveforms as faithfully as possible, and those for minimizing (or *compressing*) the computer storage requirements. The two most common techniques used to encode an audio waveform are pulse code modulation (PCM) and delta modulation (DM).

Linear pulse code modulation (PCM) associates a particular binary number with every voltage level of the incoming analog signal. As the incoming signal increases, the binary number goes up in value proportionally. Similarly, as the analog voltage goes down, the binary number decreases. A multimedia 16 bit wave file is an example of linear PCM.

Non-linear PCM allows the computer to store fewer bits per sample by dropping the bits for signals that require less sensitivity. Non-linear PCM associates a particular binary number with a voltage *range* of the incoming analog signal. By carefully choosing the ranges, 14 bits of resolution can be packed into an 8 bit sample with minimal loss of fidelity. 8 bit *u*-law (properly pronounced “MEW-law”, for the Greek letter *u*) is an example of a non-linear PCM encoding technique. 8 bit *u*-law is used to carry audio through the North American phone system.

Delta Modulation (DM) is a data compression method where only the *difference* between subsequent samples is stored. Since voice signals are relatively stable from sample to sample, the number of bits required to faithfully reproduce the signal can be reduced. One type of delta modulation is Continuously Variable Slope Delta modulation (CVSD), where a single bit is used to indicate whether the signal is increasing or decreasing. Another common type of DM is Adaptive Delta Pulse Code Modulation (ADPCM) where a constantly changing table of multiplier values enables the encoder to *adapt* to various types of signals. ***Dialogic™*** 4 bit files are an example of ADPCM.

Linear predictive coding (LPC) extracts perceptually significant features of speech directly from a time domain speech waveform to produce a time varying model of the human vocal tract excitation and transfer function. A synthesizer on the decoding reverses the process. C-ITU (CCITT) G.728 is an example of a LPC compression algorithm.

All you really need to know is that compression techniques exist and that their sole purpose is to reduce the number of bits required to store audio while simultaneously retaining as much fidelity as possible.

Frequency of Sampling

If every single number produced by the DAC were used to fully represent the sound, it would be impossible to store all that data in memory. This problem is handled by sampling the output from the DAC at a regular rate. This is called the *sample rate*, and is an important factor in determining the quality of digital sound. The sample rate is also called the *sampling frequency*, because it too can be measured in hertz.

Because of the physics of electronic filtering, it is necessary only to sample a wave twice during each cycle to get an accurate representation. This principle holds true even for very complex sound waves. As we saw earlier, even the most complex wave is composed of the sum of sinusoids at varying frequencies and amplitudes. Therefore, if you sample at a rate that is at least twice the highest frequency in your input signal, the content will be accurately captured.

If a signal is insufficiently sampled, new and unwanted frequencies are generated and added to the sampled sound. These are related to both the input frequencies and the sample frequency in such a way that they are virtually guaranteed to sound unpleasant. This is called *aliasing*.

Fortunately, aliasing is not a serious problem with most modern systems. Audio equipment incorporates special circuits, called *anti-aliasing filters,* that automatically restrict the bandwidth (frequency content) of the input signal based on the sample rate. For example, if the telephony card is commanded to sample a sound at 8 kHz, then the anti-aliasing filter is adjusted to reject frequencies over 4 kHz. A similar filter is used on the output when the sound is played back to smooth off the “rough edges” created when the analog data is digitized.

The sample rate used by a particular sampling system can usually be set through software, though the upper value is limited by the hardware. The Multimedia PC document issued by Microsoft specifies that the sound hardware be at least capable of sampling at 11.025 kHz and 22.050 kHz. Compact discs are recorded with a fixed sample rate of 44.1 kHz. Telephone quality boards, such as ***Dialogic*** boards, are capable of sampling at 6 kHz or 8 kHz, using compression techniques that result in 4 or 8 bits per sample.

Sometimes we need to convert files recorded at one frequency for use on another system at another frequency. This process is called *re-sampling*. Re-sampling requires that the original signal be reconstructed, filtered, and then chopped back up into samples at the new frequency. The techniques used to perform this process are referred to as *re-sampling algorithms*. The most common re-sampling algorithm is based on a linear approximation of the original signal. Faster algorithms simply skip or add samples, which results in lower fidelity. More time-consuming algorithms use Fourier analysis to more accurately reproduce the original audio waveform.

At a relatively low sampling rate of 6 or 8 kHz (typical for telephone quality voice) far fewer code bits are produced each second than, for example, at the 44.1 kHz sampling rate used for compact discs. For a two-channel 16 bit signal at a 44.1 kHz sampling rate, 11 million bytes are generated each minute. That’s why you’ll need at least an 800 million byte hard disk to record an hour of compact disc quality music. On the other hand, a 60 second segment of compressed telephony audio, sampled at 8,000 samples per second, using 1 byte samples, translates into 500 *thousand* bytes of data, thereby requiring significantly less storage. Even with compression, though, you can see that most recordings take up quite a lot of disk space.

Putting it all Together: How Audio Converters Work

Now that you’ve stored the audio signal onto your computer hard disk, what can you do with it? Well, just as a word processor lets you manipulate the words and pictures that make up a document, an audio editor lets you edit sound in much the same way. You can make a sound “bold” by increasing its volume; make it “italic” by changing the pitch. Figure 5 provides a block diagram of how an audio converter interacts with the components you’ve seen so far.

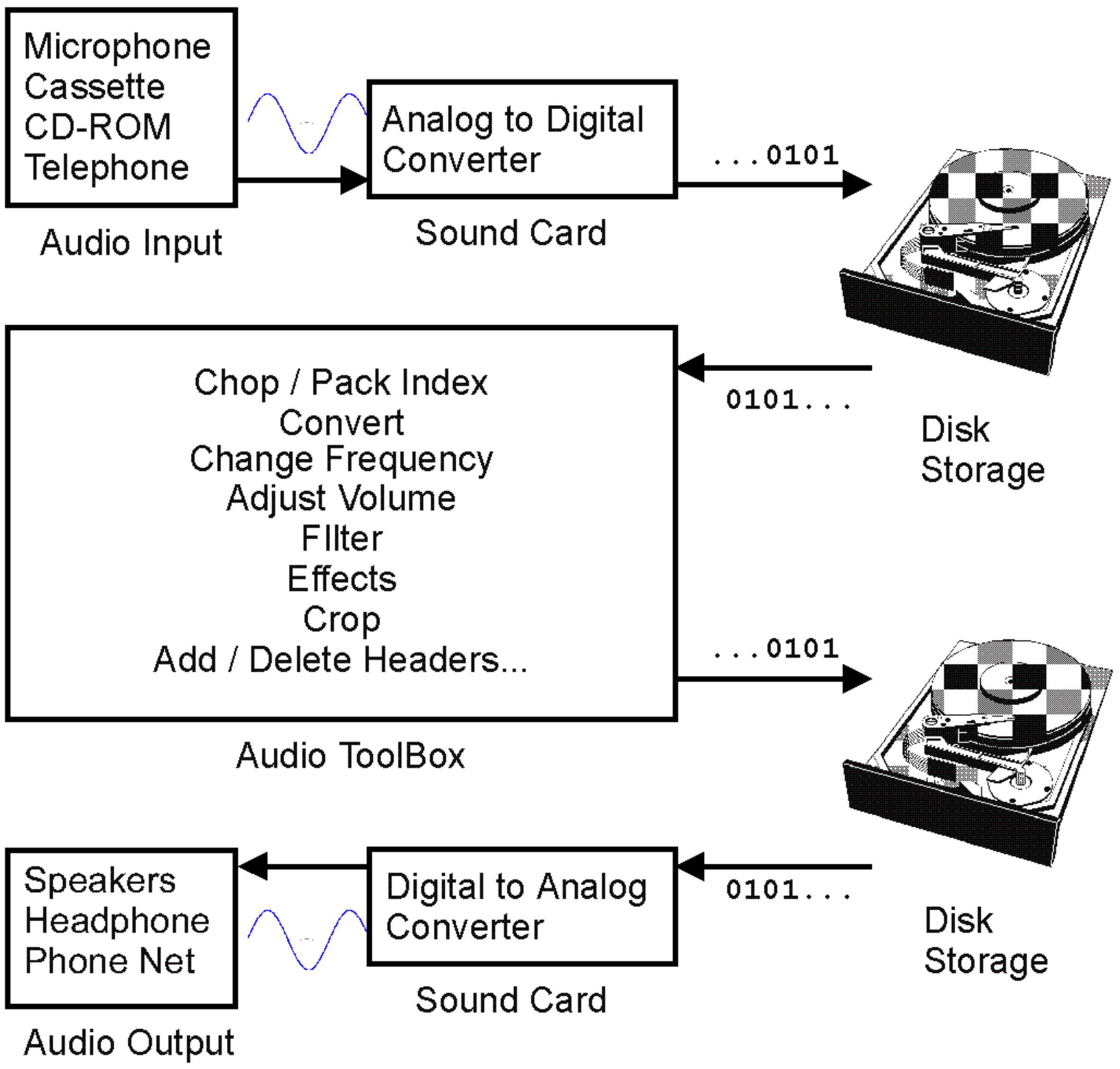
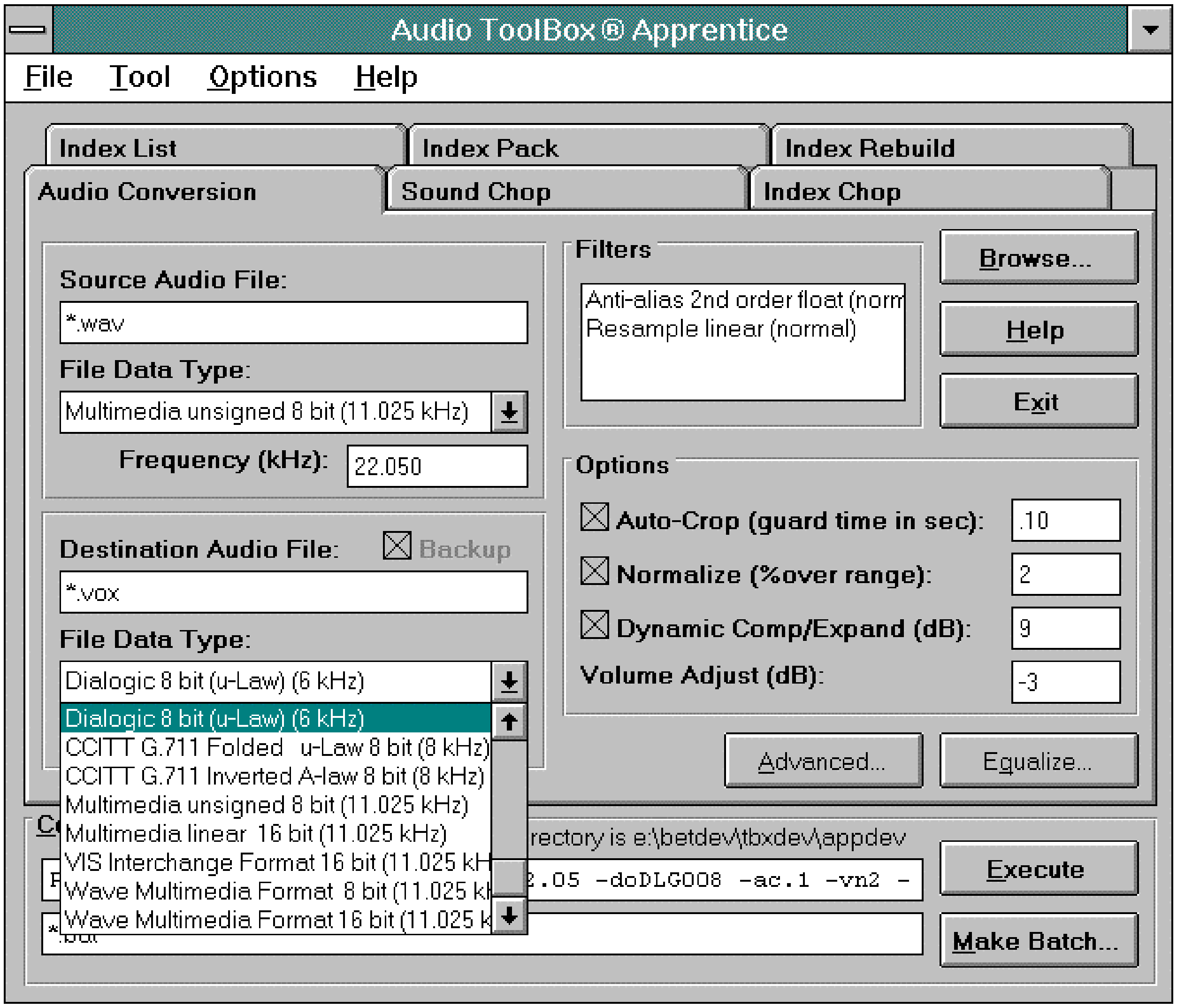


Figure 5: How an Audio Converter Works

What is *Audio ToolBox*™?

Audio ToolBox is a set of utilities that let you modify the numbers that have been stored on your computer hard disk by your telephony or multimedia hardware. ***Audio ToolBox*** knows how to translate these digitized and encoded audio signals from one format to another. Similarly, ***Audio ToolBox*** knows how to take common editing commands such as “Trim Silence” or “Adjust Volume” and perform the requested operation on the data. ***ToolBox Apprentice*** provides a convenient user interface for creating the ***Audio ToolBox*** commands to bulk process batches of audio files. And ***Audio ToolBox*** has been specifically designed to deal with the specialized hardware and audio formats used by telephony systems.



Source Audio File: *.wav

File Data Type: Multimedia unsigned 8 bit (11.025 kHz)

Frequency (kHz): 22.050

Destination Audio File: *.vox ☒ Backup

File Data Type: Dialogic 8 bit (u-Law) (6 kHz)

- Dialogic 8 bit (u-Law) (6 kHz)
- CCITT G.711 Folded u-Law 8 bit (8 kHz)
- CCITT G.711 Inverted A-law 8 bit (8 kHz)
- Multimedia unsigned 8 bit (11.025 kHz)
- Multimedia linear 16 bit (11.025 kHz)
- VIS Interchange Format 16 bit (11.025 kH
- Wave Multimedia Format 8 bit (11.025 kH
- Wave Multimedia Format 16 bit (11.025 k

Filters

Anti-alias 2nd order float (norm
Resample linear (normal)

Options

☒ Auto-Crop (guard time in sec):	.10
☒ Normalize (%over range):	2
☒ Dynamic Comp/Expand (dB):	9
Volume Adjust (dB):	-3

rectory is e:\betdev\tbxdev\appdev

```
.05 -doDLG008 -ac.1 -vn2 -
```

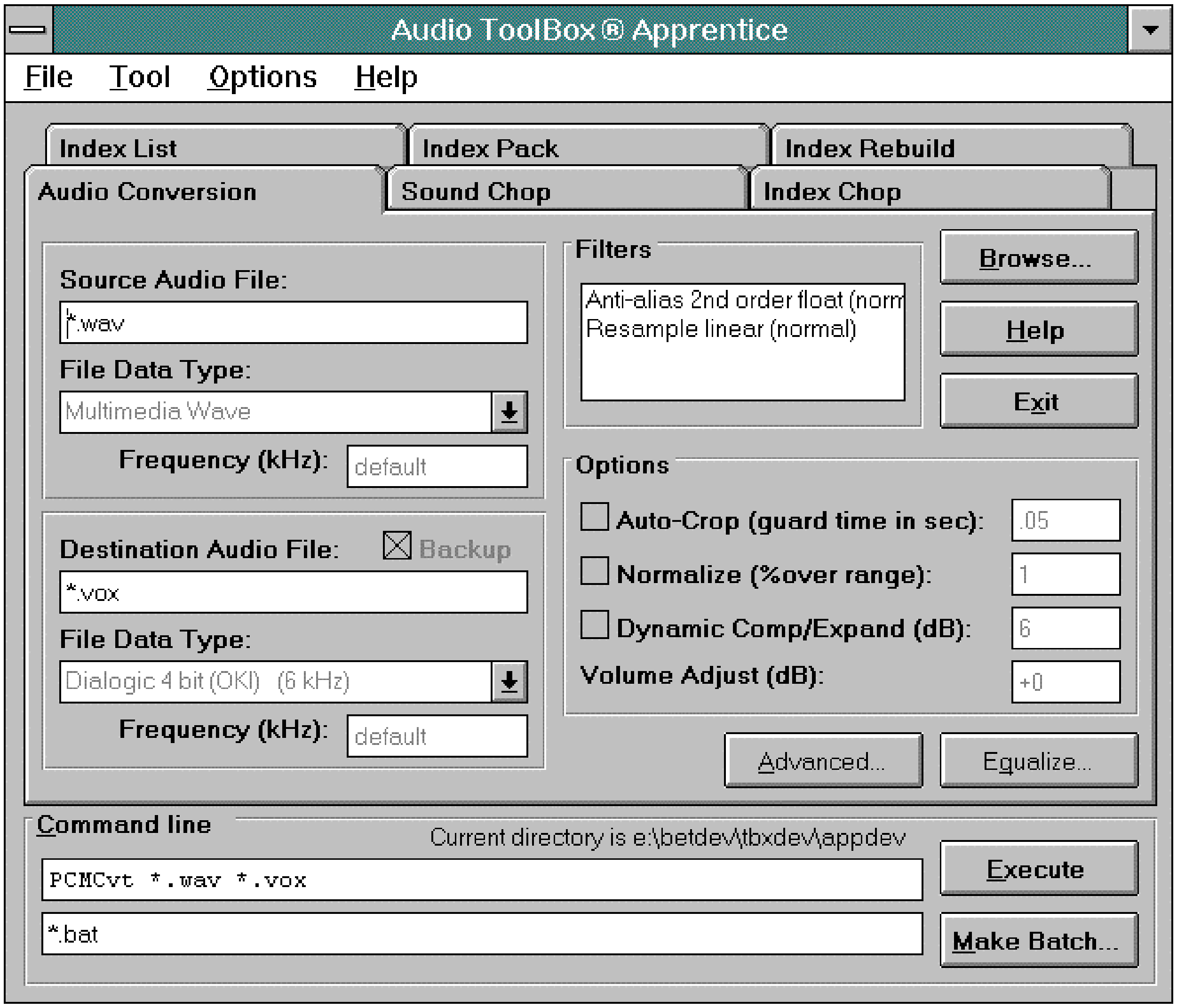
*.bat

Figure 6: ***Audio ToolBox***: The Digital Processor and Converter

Let’s say you’re running a Internet voice messaging service. Your callers send multimedia files containing messages, perhaps thousands of them each day, and you need to convert them to a telephony compatible format and insure that the messages sound as good as possible. With ***Audio ToolBox***, you can simply run the program every evening to automatically convert, adjust the volume and trim the dead space from each message — without any intervention from you.

The basic reason to use ***Audio ToolBox*** is that you simply want your audio presentations to sound good. If you develop for voice processing, or if you only change, fix, and improve your voice prompts, you’ll find ***Audio ToolBox*** just about indispensable.

Audio Conversion Tool



Source Audio File: *.wav
File Data Type: Multimedia Wave
Frequency (kHz): default

Destination Audio File: *.vox [✓ Backup]
File Data Type: Dialogic 4 bit (OKI) (6 kHz)
Frequency (kHz): default

Filters
Anti-alias 2nd order float (norm
Resample linear (normal)

Options
Auto-Crop (guard time in sec): .05
Normalize (%over range): 1
Dynamic Comp/Expand (dB): 6
Volume Adjust (dB): +0

Command line
Current directory is e:\betdev\tbxdev\appdev

```
PCMCvt *.wav *.vox
```

*.bat

Audio Conversion Tool

Sound waves, converted to binary-number data, are stored in computer memory as files by the audio hardware. These files are interpreted by ***Audio ToolBox***™ as audio waveforms, the shapes of which are analogous to the characteristics of the recorded sound. Thus ***Audio ToolBox*** lets you change the binary numbers that have been stored on your computer by your audio hardware. Just as the contents of a spreadsheet file are made intelligible and useable by a spreadsheet program, so are the binary amplitude values made useful by ***Audio ToolBox***. ***Audio ToolBox*** stores these large amounts of data on files on your computer hard disk. Unlike most other converters, ***Audio ToolBox*** does not load a copy of its data into memory when you process a sound file — all processing is done directly on the data stored on the hard disk. This allows ***Audio ToolBox*** to manipulate sound files of any size — up to your available disk space.

When Would You Use Convert?

The **audio conversion** tool lets you convert to and from Microsoft Multimedia Wave (8, 16 & MS ADPCM), linear 16 and unsigned 8 bit, Dialogic 4 and 8 bit, plus other industry standard formats. Optionally choose volume adjustments, dynamic compression/ expansion and fast or high fidelity filtering and re-sampling algorithms.

The following sections refer to options as seen on the ***ToolBox Apprentice*** screen. If you prefer to use ***Audio ToolBox*** from the DOS environment, and not use ***ToolBox Apprentice***, simply skip to the section "*Conversion Command Line Options*".

Convert Source Audio File

The Source Audio File field lets you enter the name of your source file; you can also choose a source file by using the Browse Button (be sure to position your cursor in the Source Audio File field before clicking the Browse Button)

File Data Type

Audio ToolBox supports many different file types and data formats. You may need to review information provided by your system vendor to determine which one is best for your particular needs.

Information required includes:

- ♦ File Data Type: be sure to consider the following:
    - What kind of sound file? These files may contain pure digitized data or digitized data combined with format, control, and annotation information. What kind of data is stored in the file?
    - Which of the many specialized encoding methods does it use?
- ♦ Frequency: Do you want telephony sampling frequency or higher fidelity?

File Types

The **audio conversion** tool supports two different basic file types: Pure files and Wave files.

- Pure files contain only the digitized sound.
- Wave files contain raw audio data plus additional format, control, and annotation information.

Data Types

Following are some industry standard PCM data types and a brief description. Some of the data types are pretty sophisticated, so it's difficult to provide much more information given the scope of this document. Your version of the **audio conversion** tool supports many, but not all, of these data types. Please contact VISI regarding specialized or custom formats.

- Pika AVA: Pika Technology 4 bit Adaptive Delta PCM, similar to Dialogic 4 bit ADPCM.
- Brooktrout CVSD: Brooktrout Technology 1 bit Continuously Variable Slope Delta Modulation.
- Dialogic ADPCM: Dialogic Corporation 4 bit Adaptive Delta PCM.
- Dialogic PCM: Dialogic Corporation 8 bit *u*-law similar to the C-ITU (CCITT) G.711 *u*-law.
- Float: 32 bit floating point data with a 24 bit mantissa and 8 bit exponent. This is an IEEE standard.
- FFT Real: A 32 bit floating point frequency representation of the real portion of an audio signal.
- C-ITU (CCITT) G.711: A-Law (inverted A-Law) or *u*-law (folded *u*-law): An international standard 14 bit to 8 bit compression.
- C-ITU (CCITT) G.721: An international standard 4 bit sub-band encoded Adaptive Delta PCM.
- C-ITU (CCITT) G.723 (ITU-T) 3 bit ADPCM: An international standard 3 bit sub-band encoded Adaptive Delta PCM.
- Harris CVSD: Harris Semiconductor 1 bit Continuously Variable Slope Delta Modulation.

- ♦ Multimedia unsigned 8 bit: unsigned 8 bit (0 to 255) linear data samples.
- ♦ Multimedia linear 16 bit: signed 16 bit normalized (+/- 32,767) linear two’s complement data samples. This format is an industry standard for storing digitized sound.
- ♦ MS Wave ADPCM: Microsoft Adaptive Delta PCM.
- ♦ New Voice CVSD: New Voice 1 bit Continuously Variable Slope Delta Modulation.
- ♦ Perception CVSD: Perception Technology 1 bit Continuously Variable Slope Delta Modulation.
- ♦ Rhetorex 3 bit ADPCM: Rhetorex Corporation super secret proprietary Adaptive Delta PCM.
- ♦ Rhetorex 4 bit ADPCM: Rhetorex Corporation super secret proprietary Adaptive Delta PCM.
- ♦ TTI 8 bit: Talking Technology 8 bit unsigned PCM similar to Multimedia unsigned 8 bit.

Files Converted with the Wrong File Data Type

If you convert a file with the wrong File Data Type ***Audio ToolBox*** will try to detect this condition and report an error. Some data types, however, cannot be accurately detected and the converted file will sound like static, or have sections of “static” at the beginning, end, and possibly intermixed. This “static” is actually header, format and audio information misinterpreted as audio data. Simply re-convert the file with the proper file data type.

Since ***Audio ToolBox*** supports many different file data types, you may need to review information provided by your system vendor to determine which one will work for you.

File Frequency

The source file frequency specifies the sample rate, or “recording speed” used to create the file. This number is similar to the speed of vinyl records: LP’s were typically recorded at 33 RPM while other types were recorded at 45 or even 78 RPM. Just as your LP’s won’t sound right when played at 45 RPM, so will your audio files sound strange if you use the wrong file frequencies.

Files Converted with the Wrong Frequency

If you convert a file with the wrong Frequency, the file will usually sound fast or slow but otherwise sound intelligible. Simply re-convert the file with the proper frequency. Since the source and destination frequencies are related, the following may help you determine how to change your frequency settings:

- ♦ If the file sounds too fast when played back, the **Source Frequency** is set too **high** or the **Destination Frequency** is set too **low**.
- ♦ If the file sounds too slow when played back, the **Source Frequency** is set too **low** or the **Destination Frequency** is set too **high**.

Since ***Audio ToolBox*** supports virtually any frequency, you may need to review information provided by your system vendor to determine the correct frequency for your particular application.

Indexed Files

As we've said earlier, ***Audio ToolBox*** supports Pure (raw audio data) and Multimedia Wave (headered) files. Indexed (VBase) files are groups of related audio segments which are accessed by specifying an index number. When converting an indexed file, you must first use the **index chop** tool to separate it into its component parts. Then use the **index pack** tool to put it back together.

Convert Destination Audio File

The Destination Audio File field lets you enter the name of your destination file. Just as in the Source Audio File field, you can also choose a destination file by using the Browse Button (be sure to position your cursor in the Destination Audio File field before clicking the Browse Button).

Backup Checkbox

By default, ***Audio ToolBox*** creates a backup copy of your file if you perform an "in place" conversion (i.e., do not specify a destination audio file). This protects your original file if you *accidentally* forget to specify a destination file (and wish to retain the original), or if you do not like the resulting conversion and wish to go back to the original. If this checkbox is set, ***Audio ToolBox*** backs up the most recent copy of your source file and names the backup with the extension ".BK1". Turn this checkbox off if your are short of disk space or do not wish to retain a backup copy of your original file.

Convert Filters

During conversion, ***Audio ToolBox*** will convert files recorded at any frequency to match the destination file frequency.

- ♦ Anti-aliasing Filter: The anti-aliasing filter removes unwanted high frequencies when converting from a higher to a lower frequency.

  The Low-pass 2nd order anti-aliasing filter default is on. Use the default on setting for standard fidelity. Use fast for standard fidelity with improved speed on 286/386 class computers. Use off for maximum conversion speed.

  The Low-pass 6th order anti-aliasing filter default is off. Use the on setting for improved fidelity.

  The Low-pass anti-aliasing FFT filter default is off. Use the on setting for improved fidelity.

- ♦ Re-sample Algorithm: The re-sampling algorithm determines how audio data will be generated or removed to adjust a file for the new frequency.

  The default algorithm is Linear (normal). Use the default on setting for standard fidelity. Use the re-sample FFT filter for best fidelity.

Convert Options

Convert options let you specify additional parameters to your Convert command so that you can finely tune the results to your liking.

Convert Auto-Crop

The **Convert Auto-Crop** command option looks at the selected sound segment and trims the quiet sections from the beginning and end, automatically, without accidentally erasing any audible part of the file.

The guard time parameter sets the amount of time to leave at either end of the cropped sound segment. Increase the guard time for more silence at the beginning and end, decrease for less time.

Convert Normalize

Convert Normalize scans the selected audio segment and tries to set the maximum volume level “To” an ideal target level with a permissible number of “Over” exceptions. The ideal “To” level is the volume level that would be achieved if the signal were a perfectly symmetrical wave; the “Over” exceptions compensate for the fact that most real audio waveforms are not symmetrical.

The “To” level is expressed as a percentage of the maximum amplitude possible with the current data format. The default value of 80% should be appropriate for most applications.

In the “Over” parameter, you allow the volume to go over the target range some percentage of times in order to achieve a better overall volume level. Set this parameter to zero to force no portion of this signal to go over the target level.

When you normalize a file, ***Audio ToolBox*** sets the levels of the file to an optimum level based upon a theoretical ideal. If you normalize multiple files, one at a time, the end result will be that they are each equal to an ideal, thus they each equal each other. You may remember from your school mathematics that if A=Z and B=Z, then A=B. This is the same basic principle used for equalizing multiple files, one at a time.

When you **Convert Normalize** to adjust the overall volume level of a group of files, the files will sound roughly equal in volume.

Convert Dynamic Compress / Expand

Convert Dynamic Compress / Expand lets you improve dynamic range of the audio. The dynamic compressor / expander works by analyzing the audio signal and increasing the volume level on sections that are too soft while decreasing the volume level on sections that are too loud.

The “Maximum” parameter limits the amount that the compressor / expander will increase or decrease the volume level of the audio. Increase this number (expressed in decibels) for a more noticeable effect.

Use this command to improve the “depth” and “range” of the recorded voice.

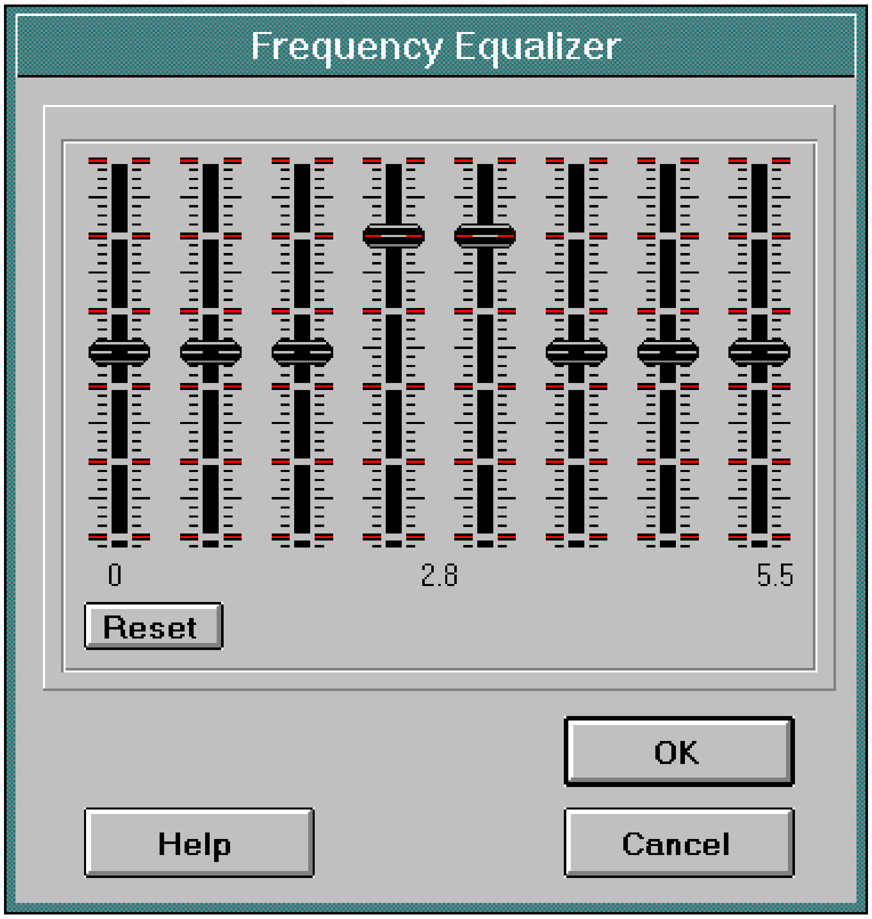
Convert Volume Adjust

The **Convert Volume Adjust** command permanently changes the loudness (volume level) of a previously recorded file. Use this command when an existing recording is too loud, or when a portion of a recording is not loud enough.

The **Convert Volume Adjust** command adjusts the amplitude of the selected sound segment by the number of decibels that you have chosen. A value of 0 dB specifies that the volume level will remain the same. Negative values specify that the level should be quieter; positive values specify that the volume level should be increased.

Technically speaking, **Convert Volume Adjust** increases or decreases the signal voltage by the specified decibel level.

Convert Equalize



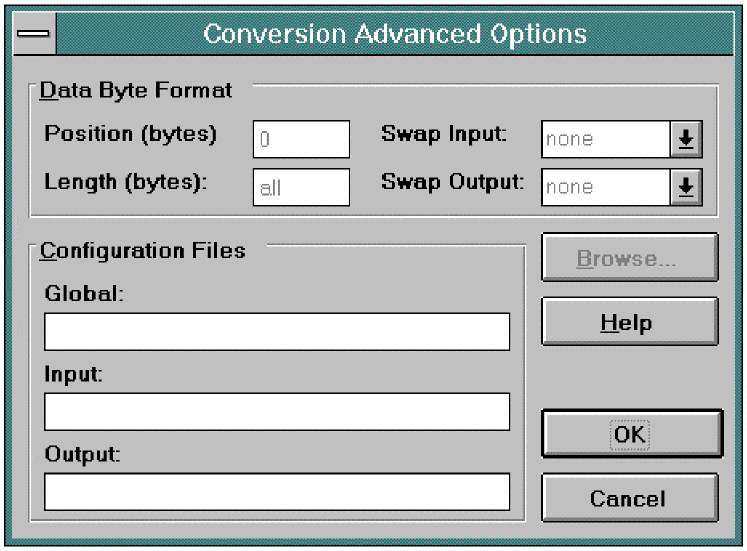
Convert Equalize Dialog Box

The **Equalize** button lets you fine tune the sound of your words, to change the quality of the vibrations themselves. While a microphone records the whole range of vibration in sound levels, the equalizer can be used to make certain ranges softer and other ranges louder, so that the net effect makes you sound different.

When you select **Equalize**, you will be presented with a dialog box that lets you adjust the relative levels of frequency bands. By moving the controls up or down from the center point, you will increase or decrease the volume level of the audio in those frequency bands. After adjusting the frequency band controls, press “OK” to apply the changes to the recorded audio during the conversion process.

Convert Advanced

Audio ToolBox offers additional features which you may find useful in handling foreign file formats and specialized conversion tasks. You can specify a starting byte position to skip custom file headers, set input file length for extracting embedded data, even select byte and bit ordering options for reading files from non-Intel computers!


Conversion Advanced Options

Data Byte Format
Position (bytes): 0
Length (bytes): all
Swap Input: none
Swap Output: none

Configuration Files
Global:
Input:
Output:

Convert Advanced Dialog Box

Position

This is the position of the input file, measured in bytes, where you will begin your conversion (default is byte 0). Use this setting to skip a portion of the beginning of the source audio file.

Length

This is the length of the input file, measured in bytes (default is entire file). Use this setting to convert only a portion of the source audio file and leave some portion of the end of the source audio file unconverted.

Swap Input

Use this setting to swap input bits, nibbles, bytes or words (default is none). Use this setting when converting files from non-Intel based computer.

Swap Output

Use this setting to swap output bits, nibbles, bytes or words (default is none). Use this setting when converting files for non-Intel based computer.

Configuration Files

Lists the name of global, input and output configuration file (default is none). Use this setting to override internal conversion variables. For VISI technical support use only.

Conversion Command Line Options

The following tables describe in detail the **audio conversion** command line parameters and their usage. If you prefer to use only the ***ToolBox Apprentice***, and work solely from the Windows environment, simply skip this section.

PCMCvt

PCMCvt [-help] OldFile [NewFile] [-ac -b -di -do -fi -fo -li -ng -ni -no -pi -qa -qd -qe -ql -qr -q2 -si -so -va -vd -vn]

Convert to and from Microsoft Multimedia Wave (8, 16 & MS ADPCM), linear 16 and unsigned 8 bit, Dialogic 4 and 8 bit, plus other industry standard formats. Optionally choose volume adjustments, dynamic compression/ expansion and fast or high fidelity filtering and re-sampling algorithms.

Parameters	Description
-h	Displays abbreviated help screen.
OldFile	Source file name specification.
NewFile	Destination output file name specification. If not specified, ***Audio ToolBox*** will overwrite the input file.
-acX.x	Auto-crop (default sound guard time is 0.05 sec).Use this setting to automatically remove silence from the beginning and

end of the file. Increase the guard time for more silence at the beginning and end.

`-b`	Inhibits creation of a backup file when no output file is specified. Use this setting to save disk space.
`-di``XXX`	Optional input data format; default is Wave Multimedia. Use this setting to specify the format of the source audio file.

Value	Meaning
DLG004	Dialogic 4 bit (OKI). Default frequency is 6 kHz.
DLG008	Dialogic 8 bit (u-Law). Default frequency is 6 kHz.
G11F08	C-ITU (CCITT) G.711 (Folded) u-Law 8 bit. Default frequency is 8 kHz.
G11I08	C-ITU (CCITT) G.711 (Inverted) a-Law 8 bit. Default frequency is 8 kHz.
G21004	C-ITU (CCITT) G.721 4 bit ADPCM. Default frequency is 8 kHz. (C-ITU (CCITT) installation only)
MPC008	Multimedia 8 bit. Default frequency is 11.025 kHz.
MPC016	Multimedia 16 bit. Default frequency is 11.025 kHz.
NWV001	NewVoice CVSD 1 bit. Default frequency is 32 kHz. (New Voice installation only)
PTC001	Perception Technology CVSD 1 bit. Default frequency is 32 kHz. (Perception Technology installation only)
VIS016	VIS Interchange format 16 bit. Frequency specified in file header.
WAV000	Wave Multimedia (input default). Frequency specified in file header.

`-do``XXX` Optional output data format; user selects default when running the setup program. Use this setting to specify a destination audio file format that differs from the default.

Value	Meaning
DLG004	see input data format.
DLG008	see input data format.
G11F08	see input data format.
G11I08	see input data format.
G21004	see input data format.
MPC008	see input data format.
MPC016	see input data format.
NWV001	see input data format.
PTC001	see input data format.
WAV008	Wave Multimedia 8 bit. Default frequency is 11.025 kHz.
WAV016	Wave Multimedia 16 bit. Default frequency is 11.025 kHz.
VIS016	VIS Interchange format 16 bit. Default frequency is 11.025 kHz.

`-fi``X.x` Frequency of input file. Use this setting to override the default sample frequency of the source audio file.

Value	Meaning
X.x	Frequency in X.x kHz (default is 11.025).

`-fo``X.x` Frequency of output file. Use this setting to override the sample frequency of the destination audio file.

Value	Meaning
X.x	Frequency in X.x kHz (see input data format for default frequency).

`-li``XXX` Length of input file in bytes (default is entire file). Use this setting to convert only a portion of the source audio file.

-ngXXX	Name of global configuration file (default is none). Use this setting to override internal conversion variables. For VISI technical support use only.
-niXXX	Name of input configuration file (default is none). Use this setting to override internal conversion variables. For VISI technical support use only.
-noXXX	Name of output configuration file (default is none). Use this setting to override internal conversion variables. For VISI technical support use only.
-piXXX	Position of input file beginning (default is byte 0). Use this setting to skip a portion of the source audio file.
-qa	Quality All: DTMF touch-tone and low pass anti-aliasing Fast Fourier Transform (FFT) filters on, FFT re-sampling on. (default is none). Use this setting for improved fidelity.
-qdXXX	DTMF touch-tone FFT filter on/ off (default is “off”). Use this setting to remove touch-tone frequencies during conversion.
-qeXXX	Equalizer FFT filter on/ off (default is “off”). Use this setting to perform frequency equalization. Default equalizer setting are optimized for voice band boost.

Comma delimited custom parameters may follow to indicate: Gain, low frequency, upper frequency, band settings (up to 32 bands): -qeG,L,H,B1,B2...B32

Value	Meaning
G	Overall gain setting.
L	Low end of equalizer band range. Default is 0 Hz.
H	High end of equalizer band range. Default of 0 sets high range to maximum (i.e., Nyquist) frequency of source file.
B1...B32	Gain levels (in dB) of bands spread evenly between low and high equalizer band range.

-qlXXX	Low pass anti-aliasing FFT filter on/ off (default is “off”). Use the “on” setting for improved fidelity.

-qrXXX	Re-sample FFT filter on/ off (default is “off”). Use the “on” setting for improved fidelity.
-q2XXX	Low pass 2nd order anti-aliasing filter off/ on/ fast (default is “on”). Use the default “on” setting for standard fidelity. Use “fast” for standard fidelity with improved speed on 286/386 class computers. Use “off” for maximum conversion speed.
-q6XXX	Low pass 6th order anti-aliasing filter off/ on/ fast (default is “off”). Use the “on” setting for improved fidelity. Use “fast” for improved fidelity with improved speed on 286/386 class computers.
-siXXX	Swap input bit, nibble, byte or word (default is none). Use this setting when converting files from non-Intel based computer.
-soXXX	Swap output bit, nibble, byte or word (default is none). Use this setting when converting files for non-Intel based computer.
-vaX.x	Volume adjust (+/-) (default is 0 dB). Use this setting to increase or decrease the volume level of the destination audio file.
-vdX.x	Volume dynamic compress/ expand (default is 6.0 dB). Use this setting to improve the dynamic range of the destination audio file.
-vnX.x	Volume normalize (default is 1.0% over range). Use this setting to adjust the volume levels across multiple files.

Example

Convert the sample ***11 kHz 16 bit Multimedia Wave*** file PCMCVT.WAV into a ***Dialogic 6 kHz 4 bit ADPCM*** file by entering:

```
PCMCvt PCMCvt.Wav AccTst.v04 <ENTER>
```

Convert the sample ***Dialogic 6 kHz 4 bit ADPCM*** file ACCTST.V04 into the ***11 kHz 16 bit Multimedia Wave*** file PCMNEW.WAV. Auto-crop silence, activate the low pass anti-aliasing and re-sampling FFT filters, activate the dynamic compressor/expander and normalize volume.

```
PCMCvt AccTst.v04 PCMNew.Wav -diDLG004 -fi6 -
doMPC016 -fo11.025 -ac -qlON -qrON -vd -vn <ENTER>
```

If we were to specify the additional equalizer command:

```
    -qe-3,0,3,12,-12,0,0,0,0,0,0
```

we would reduce the overall volume by 3 db (the first -3), set the equalizer range to between 0 and 3 kHz (the 0,3), increase the first band at 350 Hz by 12 dB (the 12), reduce the band at 700 Hz by 12 dB (the -12), and leave the rest of the bands alone (the 0 dB settings). This example shows the settings for an eight band equalizer; you can specify up to 32 bands and ***Audio ToolBox*** will adjust the equalizer accordingly.

Remarks

C-ITU (CCITT) G.721, New Voice and Perception Technology input and output formats are only available if the user selects them when running the setup program.

Generalized File Conversion

Audio ToolBox provides general purpose switches for converting additional file formats.

The following example shows how to convert Apple™ McIntosh™ AIFF 16 bit linear files recorded at 48 kHz and transferred to the IBM PC compatible computer.

```
    pcmcvt In.aif Out.vox -dimpc016 -dodlg004 -sibyte
    -pi128 -fi44.1 -fo6
```

The previous command line tells the PCMCvt program to:

- ♦ Convert the input file “IN.AIF” to the output file “OUT.VOX”
- ♦ Read the input file as “Multimedia 16 bit linear” data; write the output file as “Dialogic 4 bit” data.
- ♦ Swap the input file bytes to convert from Apple “high-low” byte order to PC “Low-high” byte order.
- ♦ Skip the first 128 bytes of the input file AIFF header.
- ♦ Read the input file frequency as 44.1 kHz (samples/ second), write the output file as 6 kHz.

The following example shows how to convert PIKA Technology 4 bit ADPCM files recorded at 9.3 kHz into Dialogic 4 bit ADPCM at 6 kHz.

```
    pcmcvt In.vox Out.vox -diDlg004 -dodlg004 -sinibble
    -fi9.3 -fo6
```

The previous command line tells the conversion program to:

- ♦ Convert the input file “IN.VOX” to the output file “OUT.VOX”
- ♦ Read the input file as “Dialogic 4 bit” data; write the output file as “Dialogic 4 bit” data.
- ♦ Swap the input file nibbles to convert from PIKA “first-last” byte order to Dialogic “last-first” nibble order.
- ♦ Read the input file frequency as 9.3 kHz (samples/ second), write the output file as 6 kHz.

Similarly, the following command will convert from Talking Technology 8 bit 6.757 kHz (skipping the 32 bit header) into the default Dialogic 4 bit at 6 kHz:

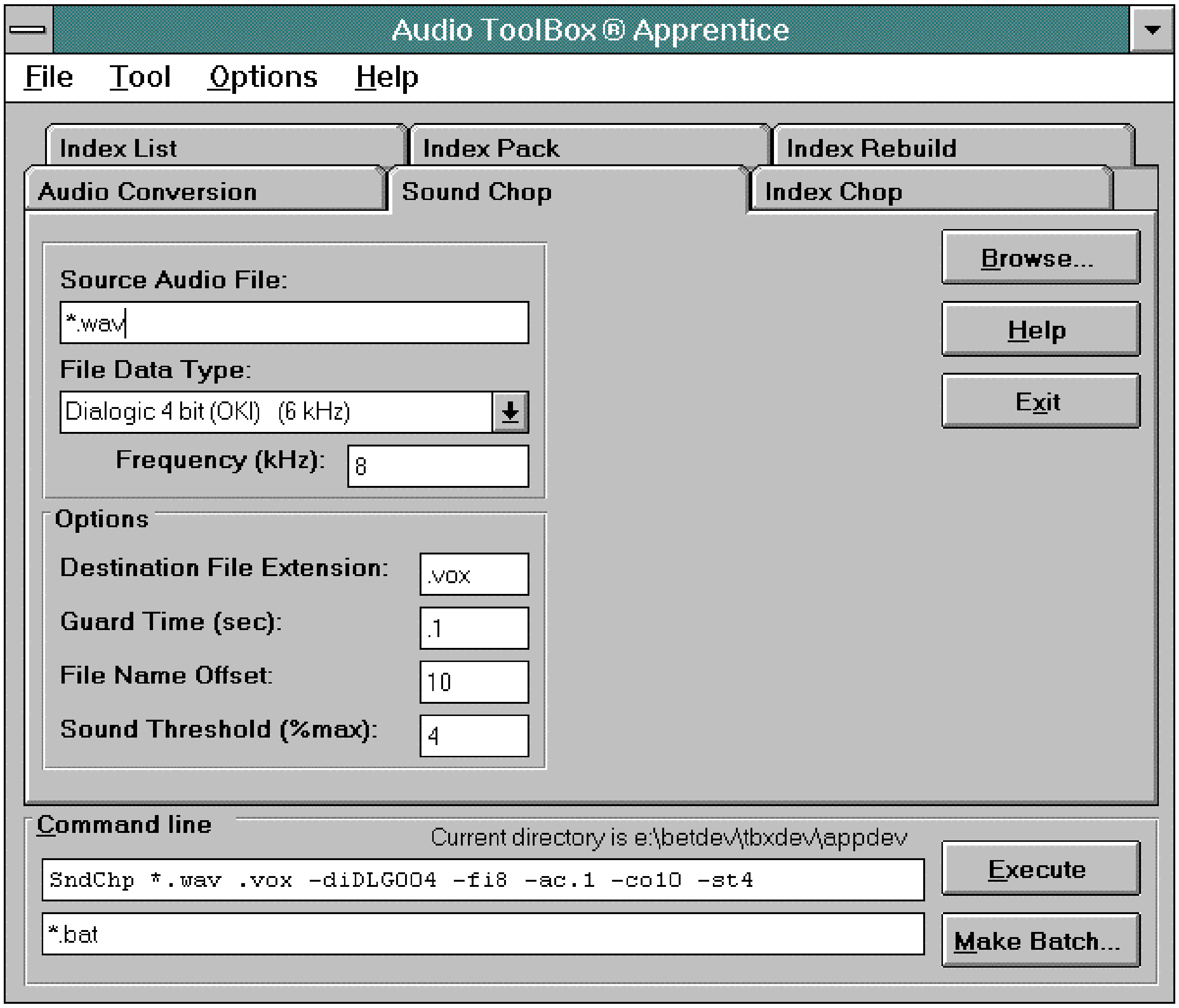
```
PCMCvt In.TTI Out.Dlg -diMPC008 -fi6.757 -pi32
```

In addition, many commercial formats have an equivalent **PCMCvt** data type. For example:

Industry Name	PCMCvt Type
AIFF	Multimedia 16 bit with header.
BiCom ADPCM	Dialogic OKI with “-swapnibble” switch
Pika Technology	Dialogic OKI with “-swapnibble” switch
Sun .au	C-ITU (CCITT) G.711 *u*-Law
Talking Technology	Multimedia 8 bit
Vynet CVSD	Perception Technology CVSD

These techniques can be used to convert from virtually any file format and data type to a supported audio format.

Sound Chop Tool



Source Audio File: *.wav

File Data Type: Dialogic 4 bit (OKI) (6 kHz)

Frequency (kHz): 8

Options

Destination File Extension: .vox

Guard Time (sec): .1

File Name Offset: 10

Sound Threshold (%max): 4

Command line

Current directory is e:\betdev\tbxdev\appdev

```
SndChp *.wav .vox -diDLG004 -fi8 -ac.1 -co10 -st4
```

*.bat

Sound Chop Tool

Audio ToolBox allows you to record a continuous stream of prompts into a single file, and then automatically break the single file into individual files. **SndChp** separates a single file into individual files using periods of silence as separators. The resulting individual files can then be converted, volume normalized, and packed using other ***Audio ToolBox*** utilities.

When Would You Use Sound Chop?

Suppose that you need to record the numbers from one to one hundred. Rather than starting and stopping the recording process each time, you can simply record all the numbers into a single file — leaving a few seconds of silence between each value — and then use the **sound chop** tool to separate, crop silence, and store each message into a separate file numbered from one to one hundred.

The following sections refer to options as seen on the ***ToolBox Apprentice*** screen. If you prefer to use ***Audio ToolBox*** from the DOS environment, and not use ***ToolBox Apprentice***, simply skip to the section "*Sound Chop Command Line Options*".

Sound Chop Source Audio File

The Sound Chop Source Audio File field allows you to type in your desired source file; you can also choose a source file by using the Browse Button (making sure that your cursor is positioned in the Source Audio File field before clicking the Browse Button).

File Data Type and File Frequency

Use this settings to specify the data type and frequency for your source audio file. ***Audio ToolBox*** supports many different file types and data formats. You may need to review information provided by your system vendor to determine which one is best for your particular needs. For a detailed discussion on file data types and frequencies, please refer to *Audio Conversion Tool, File Data Type,* page 52.

Files Chopped with the Wrong File Type

If you chop a file with the wrong File Data Type ***Audio ToolBox*** will try to detect this condition and report an error. Some data types, however, cannot be accurately detected and the file will be interpreted as "constant sound", i.e., static. Sometimes, however, the file be interpreted as normal but with sections of "static" at the beginning, end, and possibly intermixed. This "static" is actually header and format information being read as audio data, and will be treated as sound and possibly placed into a separate file. Simply re-chop the file with the proper file data type.

Since ***Audio ToolBox*** supports many different file data types, you may need to review information provided by your system vendor to determine which one will work for you.

Files Chopped with the Wrong Frequency

If you chop a file with the wrong Frequency, the file processing will usually sound normal, but the various processing parameters (expressed in seconds) will be shifted. In most cases, these discrepancies will have little noticeable effect (slightly larger or smaller regions of silence at the beginning or end, measured in hundredths of a second). For best results, however, you should review information provided by your system vendor to determine the correct frequency for your particular application.

Sound Chop Options

These options let you specify additional parameters to your **sound chop** command line so you can more accurately control the result.

Sound Chop Destination File Extension

Sound chop produces numbered files such as “0000.Vox”, “0001.Vox”, etc. Use this parameter to change the “.Vox” file extension name.

Sound Chop Guard Time

The guard time parameter sets the amount of time to leave at either end of chopped sound segments. Increase the guard time for more silence at the beginning and end, decrease for less time.

Sound Chop File Name Offset

Sound chop produces numbered files such as “0000.Vox”, “0001.Vox”, from 0 to the total number of file sound sections. Use this parameter to change the starting count value to produce starting with a value other than zero.

Sound Chop Sound Threshold

Sound threshold level maximum percentage (default is 2% of max). Use this setting to change the audio level interpreted as “sound”. Higher values result in higher immunity to noise, but lower sensitivity to sound. If your audio segments are recorded so that they fade in or fade out, you may want to decrease this setting so that sound chop retains the quieter beginnings and endings of segments. Similiarly, if the silence between segments contains “noise”, you may want to increase this setting so that sound chop accurately interpretes the separators as silence.

Sound Chop Command Line Options

The following tables describe in detail the **sound chop** command line parameters and their usage. If you prefer to use only the ***ToolBox Apprentice***, and work solely from the Windows environment, simply skip this section.

SndChp

SndChp [-help] SndFile [VoxExt -ac -co -di -fi - fl -fr -st -sa -sd]

Chops the specified file into individual files using periods of silence as separators.

Parameters	Description
-h	Displays abbreviated help screen.
SndFile	Source file name specification.
VoxExt	Destination audio file extension (default is .VOX).
-acX.x	Auto-crop (default sound guard time is 0.05 sec).Increase the guard time for more silence at the beginning and end.
-co	Destination file name counter offset (default is +0).
-di Format	Optional input data format; default is Wave Multimedia. Use this setting to specify the format of the source audio file.

Value	Meaning
DLG004	Dialogic 4 bit (OKI). Default frequency is 6 kHz.
DLG008	Dialogic 8 bit (u-Law). Default frequency is 6 kHz.
G11F08	C-ITU (CCITT) G.711 (Folded) u-Law 8 bit. Default frequency is 8 kHz.
G11I08	C-ITU (CCITT) G.711 (Inverted) a-Law 8 bit. Default frequency is 8 kHz.
G21004	C-ITU (CCITT) G.721 4 bit ADPCM. Default frequency is 8 kHz. (C-ITU (CCITT) installation only)
MPC008	Multimedia 8 bit. Default frequency is 11.025 kHz.

MPC016	Multimedia 16 bit. Default frequency is 11.025 kHz.
NWV001	NewVoice CVSD 1 bit. Default frequency is 32 kHz. (New Voice installation only)
PTC001	Perception Technology CVSD 1 bit. Default frequency is 32 kHz. (Perception Technology installation only)
VIS016	VIS Interchange format 16 bit. Frequency specified in file header.
WAV000	Wave Multimedia (input default). Frequency specified in file header.

-fiX.x	Frequency of input override in kHz. Use this setting to override the default sample frequency of the source audio file.
-flX.x	Frame length for silence detector (default is 1.0 sec). Use this setting to change the length of silence considered as separating new files.
-frX.x	Frame resolution for silence detector (default is 0.01 sec). Use this setting to change the level of detail used when analyzing a file. For VISI technical support use only.
-stXXX	Sound threshold level maximum percentage (default is 2% of max). Use this setting to change the audio level interpreted as "sound". Higher values result in higher immunity to noise, but lower sensitivity to sound.
-saXXX	Sensitivity attack ratio for silence detector (default is 20% of frame). Use this setting to change the amount of sound that must be detected to recognize a transition from a region of silence to a region of active audio. Higher values result in higher immunity to noise, but increase the possibility of losing a portion of the beginning of active audio.
-sdXXX	Sensitivity decay ratio for silence detector (default is 10% of frame). Use this setting to change the amount of sound that must be detected to recognize a transition from a region of active audio to a region of silence. Higher values result in

higher immunity to noise, but increase the possibility of leaving a larger portion of silence at the end of active audio.

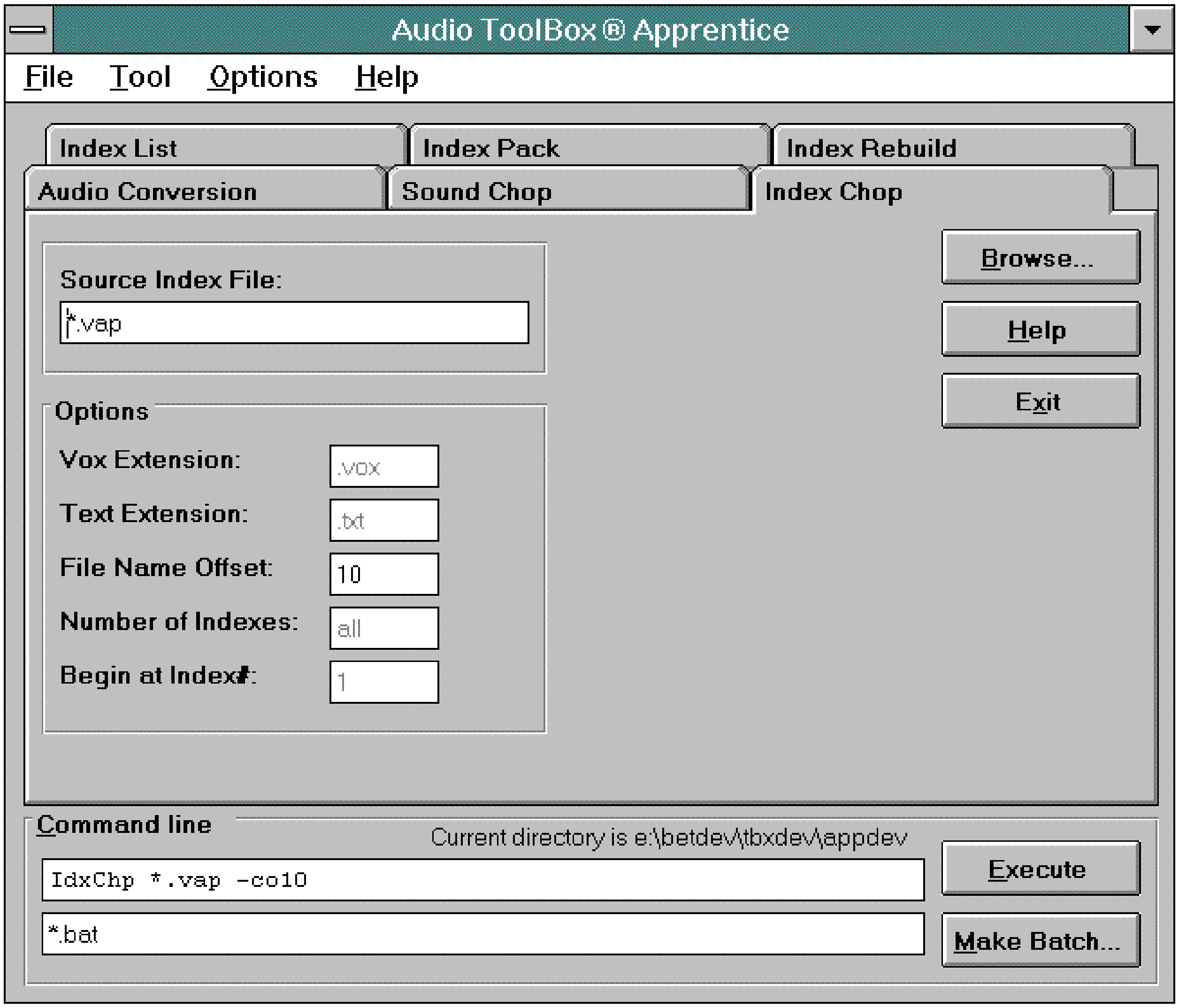
Example

Chop the sample silence delimited ***11 kHz 16 bit Multimedia Wave*** file PCMCVT.WAV into separate ***Multimedia Wave*** files by entering:

```
SndChp PCMCvt.Wav<ENTER>
```

You have now created 0001.WAV and 0002.WAV, which contain the active audio portions of the original file.

Index Chop Tool



Source Index File: *.vap

Options

Vox Extension: .vox

Text Extension: .txt

File Name Offset: 10

Number of Indexes: all

Begin at Index#: 1

Command line

Current directory is e:\betdev\tbxdev\appdev

```
IdxChp *.vap -co10
```

```
*.bat
```

Index Chop Tool

Related audio files can be grouped into a single larger file known as an “indexed file”. Each of the individual file segments or “indexes” is assigned an “index number”. You can access any audio segment by simply referring to its index number. It’s a lot like a compact disk, where each piece of music can be accessed by its number. Indexed files are very useful in organizing and managing large numbers of related files.

Indexed files also give you the option of associating a text file with each specific index in the file. You can use the text file to jot down a description of the index or indicate a special name for that index. It's a great way to remember what is in an index without actually playing it. Indexed files store all of this special information in a file header that is saved with the file.

The **index chop** tool allows you to "chop" indexed files into their components: pure audio and text files that can then be batch processed, converted and edited.

When Would You Use Index Chop?

Indexed files are popular with many voice mail systems and software vendors. Since these files contain a mixture of both audio and text information, it is sometimes necessary to extract the "audio only" portions of the file. Use the **index chop** tool to separate the indexed file into component parts for further processing.

The following sections refer to options as seen on the ***ToolBox Apprentice*** screen. If you prefer to use ***Audio ToolBox*** from the DOS environment, and not use ***ToolBox Apprentice***, simply skip to the section "*Index Chop Command Line Options*".

Index Chop Source Index File

The Source Index File field lets you enter the name of your source file; you can also choose a source file by using the Browse Button (be sure to position your cursor in the Source Index File field before clicking the Browse Button).

Index Chop Options

The following options lets you adjust **index chop** processing parameters to suite the needs of your particular task and environment.

Index Chop Vox File Extension

This is where you specify the destination audio file extension (default is .vox). Use this setting to change the file name extension of the audio file you are creating.

Index Chop Text File Extension

This is where you specify the destination text file extension (default is .txt). Use this setting to change the file name extension of the text file you are creating.

Index Chop File Name Offset

The individual files in your index file will be numbered sequentially when you use **index chop**, and ***Audio ToolBox*** will automatically name the first one "0001". Use the File Name Offset to start numbering the audio and text files you're creating with whatever number you choose.

Index Chop Number of Indexes

Here you can specify the number of indexes to chop (default is all). Use this setting to limit the extraction of indexes to a specific number.

Index Chop Begin at Index#

Here you can specify where to begin chopping indexes. ***Audio ToolBox*** will automatically start at index position 1; you can start at any number up to 9999. Use this setting to limit the extraction of indexes to a specific range.

Index Chop Command Line Options

The following tables describe in detail the **index chop** command line parameters and their usage. If you prefer to use only the ***ToolBox Apprentice***, and work solely from the Windows environment, simply skip this section.

IdxChp

IdxChp [-help] IdxFile [VoxExt TxtExt -co -n -@]

Chops indexed files into pure audio and text files that can be batch processed, converted, and edited.

Parameters	Description
`-h`	Displays abbreviated help screen.
`IdxFile`	Source file name specification.
`VoxExt`	Destination audio file extension (default is .vox). Use this setting to change the file name extension of the audio files you are creating.
`TxtExt`	Destination text file extension (default is .txt). Use this setting to change the file name extension of the text files you are creating.

`-co`XXX	Destination file name counter offset (default is “0001”). Use this setting to change the beginning numeric name of the audio and text files you area creating.
`-n`XXXX	Number of indexes to chop (default is all). Use this setting to limit the extraction of indexes to a specific number.
`-@`XXXX	Chop @ index position (default is 1). Use this setting to start the extraction of indexes at a specific index number. you can set this number at any value up to 9999.

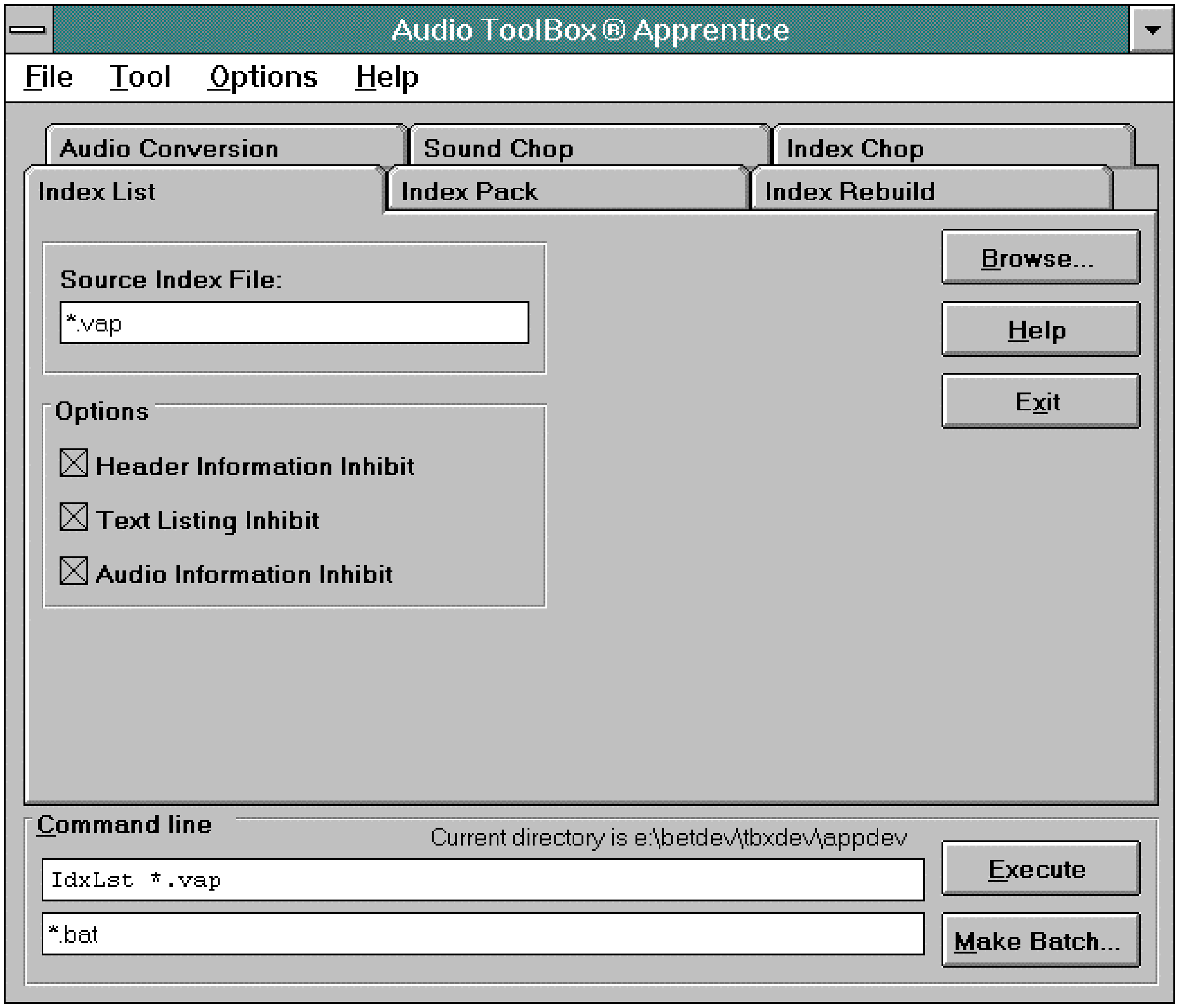
Example

Chop the sample file IDXTST.VAP into its component audio and text files beginning with the first index, for only one index, and change the default base name and extension of the destination files:

```
    IdxChp IdxTst.vap *.vvv *.ttt -@1 -n1 -co11 <ENTER>
```

You have now created 0011.VVV and 0011.TTT, which contain the audio and annotation text from the first index entry.

Index List Tool



Source Index File: *.vap

Options
☒ Header Information Inhibit
☒ Text Listing Inhibit
☒ Audio Information Inhibit

Command line
Current directory is e:\betdev\tbxdev\appdev

```
IdxLst *.vap
```

*.bat

Index List Tool

Related audio files can be grouped into a single larger file known as an “indexed file”. Each of the individual file segments or “indexes” is assigned an “index number”. You can access any audio segment by simply referring to its index number. It’s a lot like a compact disk, where each piece of music can be accessed by its number. Indexed files are very useful in organizing and managing large numbers of related files.

Indexed files also give you the option of associating a text file with each specific index in the file. You can use the text file to jot down a description of the index or indicate a special name for that index. It’s a great way to remember what is in an index without actually playing it. Indexed files store all of this special information in a file header that is saved with the file.

The **index list** tool lets you “see” the text contents of an indexed file, and provides information on the length and physical position of individual entries within the file.

When Would You Use Index List?

Indexed audio files have a built-in “Table of Contents” — a great time-saver in finding which audio segment you want. Each audio segment can have an attached annotation. The **index list** tool lets you view and list the header and content information of indexed files.

The following sections refer to options as seen on the ***ToolBox Apprentice*** screen. If you prefer to use ***Audio ToolBox*** from the DOS environment, and not use ***ToolBox Apprentice***, simply skip to the section “*Index List Command Line Options*”.

Index List Source Index File

The Index List Source Audio File field lets you enter the name of your source file; you can also choose a source file by using the Browse Button (be sure to position your cursor in the Source Index File field before clicking the Browse Button).

Index List Options

The following options lets you adjust **index list** processing parameters to suite the needs of your particular task and environment.

Index List Header Information Inhibit

Inhibits the general header information listing. Use this setting if you plan to ignore the header information.

Index List Text Listing

Inhibits the text listing. Use this setting if you plan to ignore the text content of the indexed file.

Index List Audio Information Inhibit

Inhibits the Vox audio data listing. Use this setting if you plan to ignore the audio offset and length information of the indexed file.

Index List Command Line Options

The following tables describe in detail the **index list** command line parameters and their usage. If you prefer to use only the ***ToolBox Apprentice***, and work solely from the Windows environment, simply skip this section.

IdxLst

IdxLst [-help] IdxFile [-g -t -v]

List the header and content information of an indexed file.

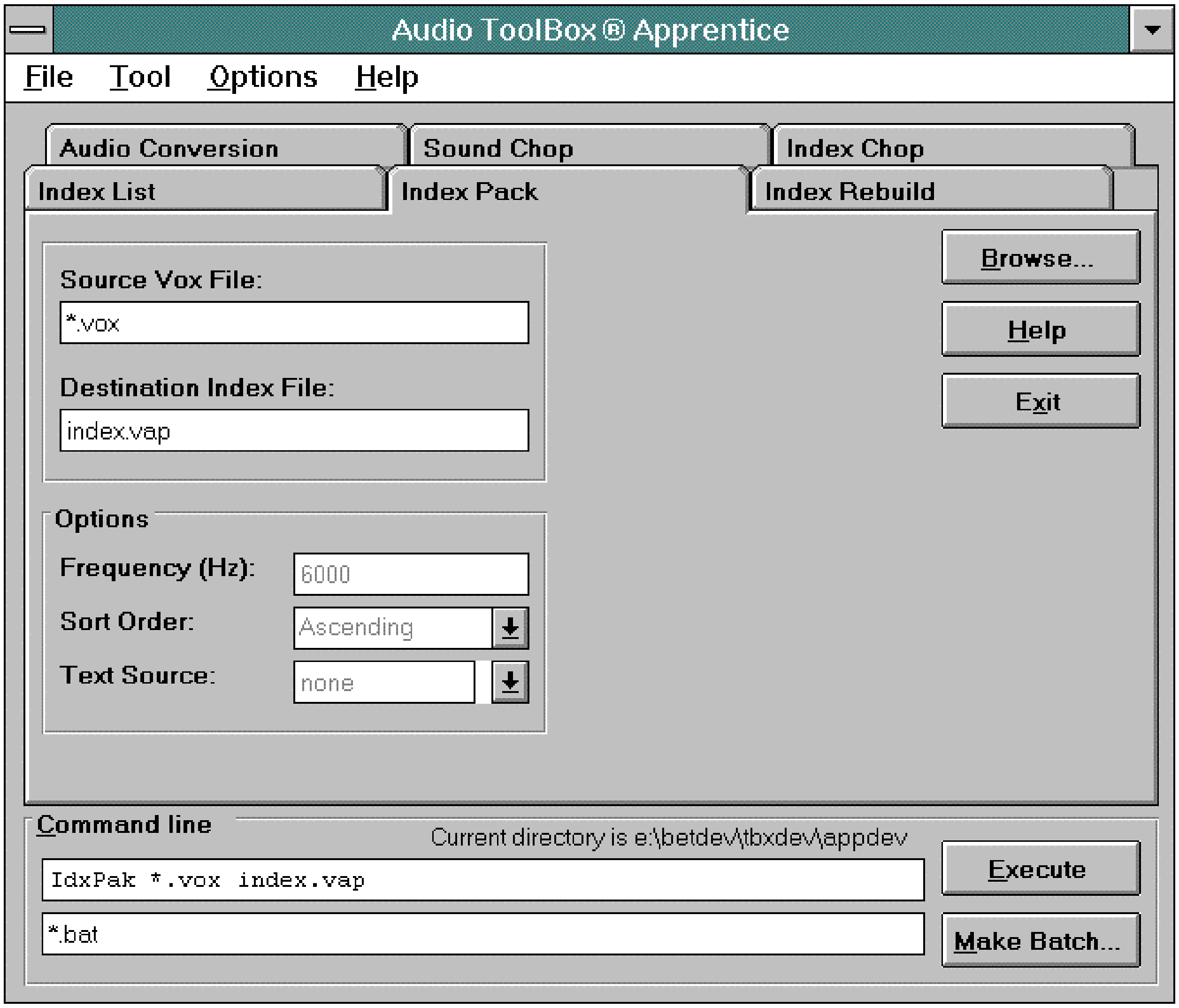
Parameters	Description
-h	Displays abbreviated help screen.
IdxFile	Source file name specification.
-g	General header information listing inhibited. Use this setting to ignore the header information.
-t	Text listing inhibited. Use this setting to ignore the text content of the indexed file.
-v	Vox audio data listing inhibited. Use this setting to ignore the audio extent information of the indexed file.

Example

List the contents of the sample indexed file IDXTST.VAP and route the output to a text file. View the listing with the DOS editor.

```
    IdxLst IdxTst.vap >list.txt <ENTER>
    edit list.txt <ENTER>
```

Index Pack Tool



Source Vox File: *.vox

Destination Index File: index.vap

Options

Frequency (Hz): 6000

Sort Order: Ascending

Text Source: none

Command line

Current directory is e:\betdev\tbxdev\appdev

```
IdxPak *.vox index.vap
```

*.bat

Index Pack Tool

Related audio files can be grouped into a single larger file known as an “indexed file”. Each of the individual file segments or “indexes” is assigned an “index number”. You can access any audio segment by simply referring to its index number. It’s a lot like a compact disk, where each piece of music can be accessed by its number. Indexed files are very useful in organizing and managing large numbers of related files.

Indexed files also give you the option of associating a text file with each specific index in the file. You can use the text file to jot down a description of the index or indicate a special name for that index. It's a great way to remember what is in an index without actually playing it. Indexed files store all of this special information in a file header that is saved with the file.

The **index pack** tool lets you create these collections of numbered sound segments by combining individual files into a single indexed file.

When Would You Use Index Pack?

Indexed files are popular with many voice mail systems and software vendors. Since these files contain a mixture of both audio and text information, it is sometimes necessary to extract the "audio only" portions of the file. Use the **index chop** tool to separate the indexed file into component parts for further processing.

After the files have been processed individually, as either audio or text data, you then need to combine the individual files back into a single indexed file. Use the **index pack** tool when you want to "pack" pure audio and text files into an indexed file. You can later work with the combined file using an indexed file compatible editor such as VISI's *VFEdit*$^{®}$ *Professional Prompt Editor*.

The following sections refer to options as seen on the ***ToolBox Apprentice*** screen. If you prefer to use ***Audio ToolBox*** from the DOS environment, and not use ***ToolBox Apprentice***, simply skip to the section "*Index Pack Command Line Options*".

Index Pack Source Vox File

The Index Pack Source Vox File field lets you enter the name of your source file; you can also choose a source file by using the Browse Button (be sure to position your cursor in the Source Vox File field before clicking the Browse Button).

Index Pack Destination Index File

The Destination Index File field lets you enter the name of your destination file. Just as in the Source Vox File field, you can also choose a destination file by using the Browse Button (be sure to position your cursor in the Destination Index File field before clicking the Browse Button).

Index Pack Options

The following options lets you adjust **index pack** processing parameters to suite the needs of your particular task and environment.

Index Pack Frequency

Here you can set the sample frequency as a number in the format NNNN (default is 6000). Use this setting to change the frequency stored in the header of the indexed file. This setting DOES NOT change the audio data.

Index Pack Sort Order

This option lets you pack files in ascending or descending alphabetical order, or to pack them, unsorted, in their current directory order. (default is ascending order).

Index Pack Text Source

Text file specification with * wildcard (default is no text). Use a setting such as “*.txt” to include annotation text data in the indexed file.

Indexed audio Files have a built-in “Table of Contents” — a great time-saver in finding which audio segment you want. Each audio segment can have an attached annotation. The **index pack** tool lets you add text along with the audio data.

Index Pack Command Line Options

The following tables describe in detail the **index pack** command line parameters and their usage. If you prefer to use only the ***ToolBox Apprentice***, and work solely from the Windows environment, simply skip this section.

IdxPak

IdxPak [-help] VoxFile IdxFile [-f -s -t]

Pack pure audio and text files into indexed files.

Parameters	Description
`-h`	Displays abbreviated help screen.
`VoxFile`	Source file name specification.
`IdxFile`	Destination file name specification.

`-fNNNN`	Set sample frequency to value NNNN (default is 6000). Use this setting to change the frequency stored in the header of the indexed file. This setting DOES NOT change the audio data.
`-sX`	Sort files; + up, - down, u unsorted. (default is +). Use this setting to pack the files in descending or unsorted directory order.
`-tSPEC`	Text file specification with * wildcard (default is no text). Use this setting to include annotation text data in the indexed file.

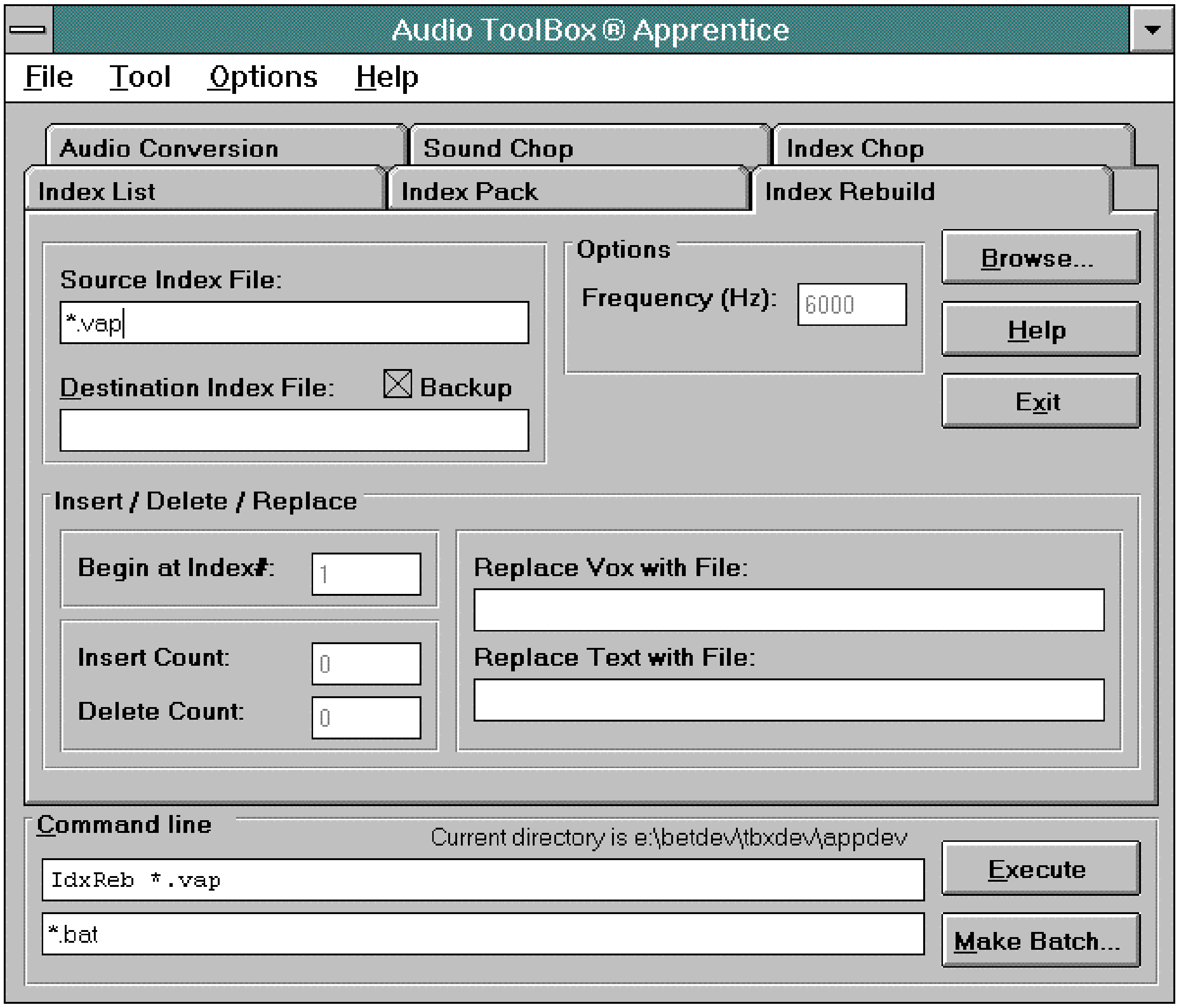
Example

Pack the sample files PAK00?.VOX and PAK00?.TXT into an indexed file in ascending order and set the header sample frequency to 6053:

```
IdxPak Pak00?.vox IdxTst.vap -t*.txt -s+ -f6053
<ENTER>
```

You have now created IDXTST.VAP, an indexed file containing the raw audio and text from PAK001.VOX, PAK001.TXT and PAK002.VOX. The frequency specified in the header has been set to 6053.

Index Rebuild Tool



Source Index File: *.vap

Destination Index File:

☒ Backup

Options

Frequency (Hz): 6000

Insert / Delete / Replace

Begin at Index#: 1

Insert Count: 0

Delete Count: 0

Replace Vox with File:

Replace Text with File:

Command line

Current directory is e:\betdev\tbxdev\appdev

```
IdxReb *.vap
```

*.bat

Index Rebuild Tool

Related audio files can be grouped into a single larger file known as an “indexed file”. Each of the individual file segments or “indexes” is assigned an “index number”. You can access any audio segment by simply referring to its index number. It’s a lot like a compact disk, where each piece of music can be accessed by its number. Indexed files are very useful in organizing and managing large numbers of related files.

Indexed files also give you the option of associating a text file with each specific index in the file. You can use the text file to jot down a description of the index or indicate a special name for that index. It's a great way to remember what is in an index without actually playing it. Indexed files store all of this special information in a file header that is saved with the file.

The **index rebuild** tool is your maintenance kit for these collections of numbered sound segments.

When Would You Use Index Rebuild?

You may find, over time, that the indexed files you are using become fragmented (i.e., data is scattered throughout), or that you need to change the size of the file (either shorter or longer), or need to perform and integrity check after a hard disk failure. The **index rebuild** tool lets you perform these types of maintenance operations on indexed files. Use this tool when you need to lengthen, shorten, replace entries and "rebuild" indexed files.

The following sections refer to options as seen on the ***ToolBox Apprentice*** screen. If you prefer to use ***Audio ToolBox*** from the DOS environment, and not use ***ToolBox Apprentice***, simply skip to the section "*Index Rebuild Command Line Options*".

Index Rebuild Source Index File

The Source Index File field lets you enter the name of your source file; you can also choose a source file by using the Browse Button (be sure to position your cursor in the Source Index File field before clicking the Browse Button)

Index Rebuild Destination Index File

The Destination Index File field lets you enter the name of your desired destination index file. Just as in the Source Index File field, you can also choose a destination file by using the Browse Button (be sure to position your cursor in the Destination Index File field before clicking the Browse Button).

Backup Checkbox

Audio ToolBox creates a backup copy of your file if you perform an "in place" rebuild (i.e., do not specify a destination audio file). This protects your original file if you *accidentally* forget to specify a destination file (and wish to retain the original), or if you do not like the resulting reconstruction and wish to go back to the original. If this checkbox is set, ***Audio ToolBox*** backs up the most recent copy of your source file and names the backup with the extension ".BK1". Turn this checkbox off if your are short of disk space or do not wish to retain a backup copy of your original file.

Index Rebuild Options

The following options lets you adjust **index rebuild** processing parameters to suite the needs of your particular task and environment.

Index Rebuild Frequency

Here you can set the sample frequency as a number in the format NNNN (default is 6000). This setting changes the frequency stored in the header of the indexed file and DOES NOT change the audio data.

Index Rebuild Insert / Delete / Replace

The following options let you lengthen, shorten and replace specific index entries.

Begin at Index#

The **Begin at Index#** value lets you select where to begin processing within the indexed file. This parameter lets you control which audio segments will be affected by the rebuild operations. (default is 1).

Insert Count

The **Insert Count** option lets you insert a new (empty) audio segment into the list of audio segments. Use this option to insert a specific number of new indexes (expressed in the format NNNN, default is none). Adds empty index entries before the selected position as specified by the **Begin at Index#**.

Delete Count

The **Delete Count** options lets you delete an audio segment from the indexed file. You can delete a specific number of indexes (expressed in the format NNNN, the default being none), including the index specified by the **Begin at Index#**. Use this setting to remove index entries and the associated audio and text.

Replace Vox with File

Use this setting to change the audio data in the indexed file. Replaces audio data (default is none) at the specified position. Replaces the audio contents of the index specified by the **Begin at Index#**.

Replace Text with File

Use this setting to change the annotation text data in the indexed file. Replaces annotation text data (default is none) at the specified position. Replaces the annotation text contents of the index specified by the **Begin at Index#**.

Index Rebuild Command Line Options

The following tables describe in detail the **index rebuild** command line parameters and their usage. If you prefer to use only the ***ToolBox Apprentice***, and work solely from the Windows environment, simply skip this section.

IdxReb

IdxReb [-help] OldFile NewFile [-b -d -f -i -rt -rv -@]

Rebuilds an existing index file. You can optionally delete or insert indexes and change the audio and text contents.

Parameters	Description
-h	Displays abbreviated help screen.
OldFile	Source file name specification.
NewFile	Destination file name specification, if different from OldFile.

Parameters	Description
-b	Inhibit creation of a backup file when no output file is specified. Use this setting to save disk space when no destination audio file is specified.
-dNNNN	Delete NNNN indexes, after and including @ position (default is none). Use this setting to remove index entries and the associated audio and text.

`-fNNNN`	Set sample frequency to value NNNN. Use this setting to change the frequency stored in the header of the indexed file. This setting DOES NOT change the audio data.
`-iNNNN`	Insert NNNN new indexes before @ position (default is none). Use this setting to add empty index entries.
`-rtFILE`	Replace annotation text @ position (default is none). Use this setting to replace the annotation text contents of the specified index.
`-rvFILE`	Replace audio data @ position (default is none). Use this setting to replace the audio contents of the specified index.
`-@XXXX`	Perform operations @ index position (default is 1, end is 9999). Use this setting to limit operations to a range of indexes.

Example

Rebuild the sample file IDXTST.VAP and add 98 new empty index entries to the end; reset the frequency specified in the header to 6053.

```
    IdxReb IdxTst.vap -i98 -@9999 -f6053 <ENTER>
```

Now replace the audio and annotation text in the last index:

```
    IdxReb IdxTst.vap -rtPak001.txt -rvPak001.vox -
    @9999 <ENTER>
```

Now return the file to the original condition by deleting the last 98 entries and resetting the header sample frequency:

```
    IdxReb IdxTst.vap -d98 -@3 -f6000 <ENTER>
```

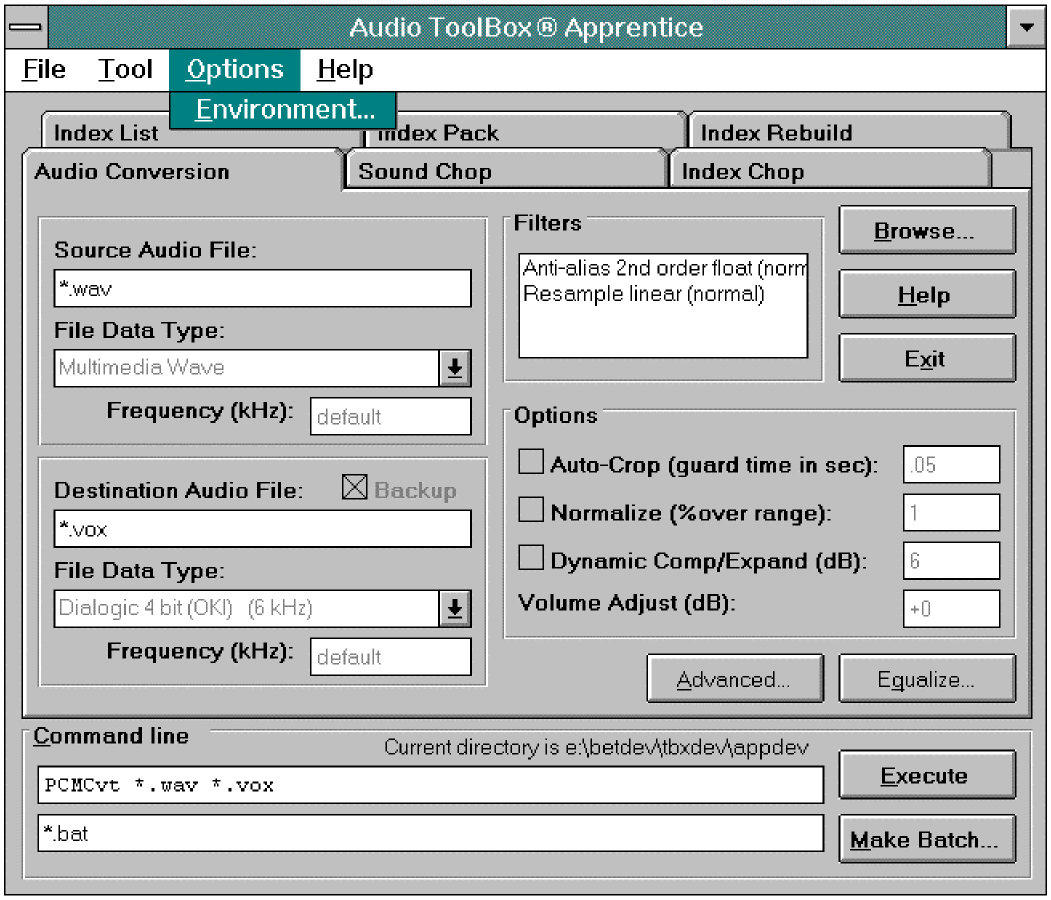
As a final example, you can use the DOS “FOR” command to insert file 0001.VOX into position one of an existing indexed file, 0002.VOX into position two, and so on. Simply place the following line into a file called REBMUL.BAT:

```
    for %%f in (*.vox) do idxreb %1 -rv%%f -@%%f
```

To use this batch file, change to the directory containing the individual .VOX files, and enter:

```
    REBMUL File.vap
```

Options Menu



Source Audio File: *.wav
File Data Type: Multimedia Wave
Frequency (kHz): default
Destination Audio File: *.vox [✓ Backup]
File Data Type: Dialogic 4 bit (OKI) (6 kHz)
Frequency (kHz): default

Filters
Anti-alias 2nd order float (norm
Resample linear (normal)

Options
Auto-Crop (guard time in sec): .05
Normalize (%over range): 1
Dynamic Comp/Expand (dB): 6
Volume Adjust (dB): +0

Command line
Current directory is e:\betdev\tbxdev\appdev

```
PCMCvt *.wav *.vox
```

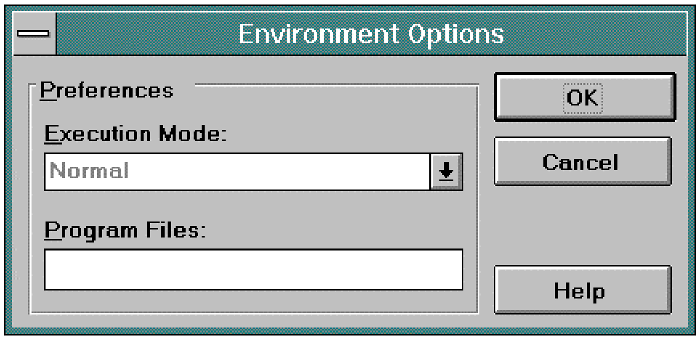
*.bat

Options Menu Items

The **Options Menu** lets you control ***Audio ToolBox***™ operational preferences and features of your audio hardware. Use the Options Menu to:

- Control the execution mode of the conversion processing tasks.
- Control the location of executable files.

Options Environment



Options Environment Dialog Box

These settings control preferences to customize operation for your environment. ***Audio ToolBox*** lets you set these preferences but you will normally not need to change these settings.

Execution Mode

This will affect what is displayed when you press the Execute button for your ***Audio ToolBox*** command.

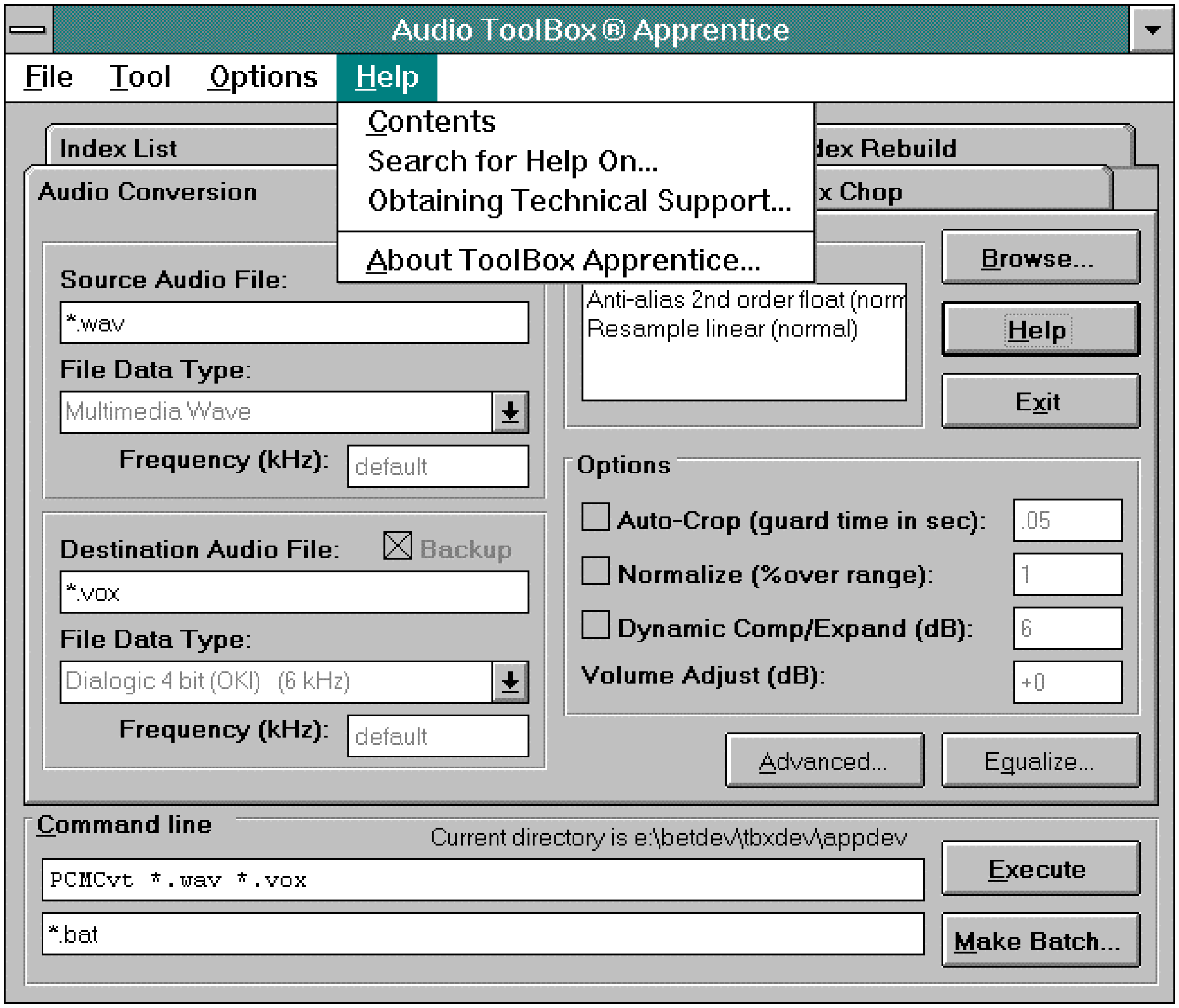
- Normal: This setting tells ***Audio ToolBox*** to show a windowed DOS session to display the progress and display messages during processing. This setting lets you view any messages and monitor progress during time-consuming operations.
- Minimized: This setting tells ***Audio ToolBox*** to show the conversion DOS session as a minimized icon. This setting lets you monitor progress (by double-clicking on the icon) but does not distract from your other work with a regular screen. Use this setting after you are comfortable with the ***Audio ToolBox*** functionality, and know the requested operation will complete as expected.
- Hidden: This setting tells ***Audio ToolBox*** to completely hide the conversion DOS session. This setting lets you perform the ***Audio ToolBox*** operations, but does not distract from your other work with any visual indicator. Use this setting after you are comfortable with the ***Audio ToolBox*** functionality, and know the requested operation will complete as expected.

Program Files

- Program Files: This setting specifies the location of the ***Audio ToolBox*** tools. These are the executable files that end with the EXE extension. Change this parameter if you are on a disk-less workstation or running from a networked drive. You can also use this setting if you do not have the ***Audio ToolBox*** installation directory in your DOS PATH environment variable.

  Note that the `TMP=` variable in the DOS environment controls the settings for temporary work files.

Help Menu



Source Audio File: *.wav

File Data Type: Multimedia Wave

Frequency (kHz): default

Destination Audio File: *.vox [✓ Backup]

File Data Type: Dialogic 4 bit (OKI) (6 kHz)

Frequency (kHz): default

Anti-alias 2nd order float (norm
Resample linear (normal)

Options

Auto-Crop (guard time in sec): .05

Normalize (%over range): 1

Dynamic Comp/Expand (dB): 6

Volume Adjust (dB): +0

Command line

Current directory is e:\betdev\tbxdev\appdev

```
PCMCvt *.wav *.vox
```

*.bat

Help Menu Items

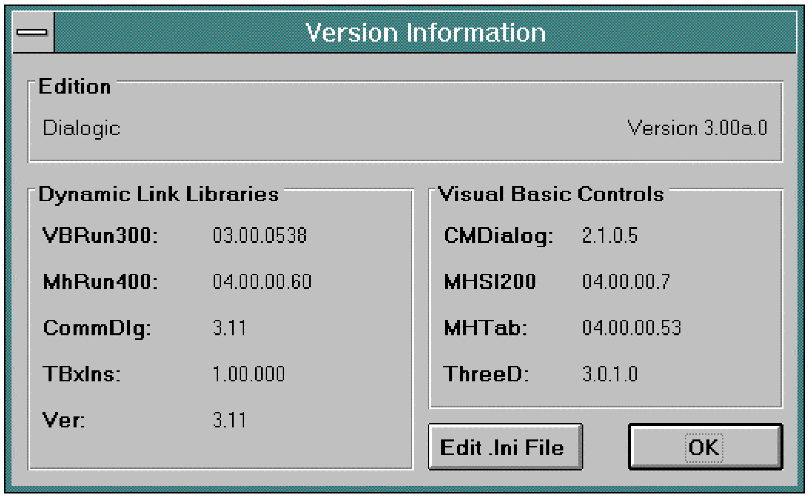
You can get help in several ways. You can use the Help button on dialog boxes or choose Search for Help On from the Help menu. Use the **Help menu** to:

- Display the contents list of the help file.
- Search for help by key word.
- Get detailed information on obtaining technical support.

About *ToolBox Apprentice*

The **Help About ToolBox Apprentice** menu item displays detailed information about the company and the product. Use this command to:

- Get VISI contact information.
- Get detailed system information.


Version Information

Edition
Dialogic   Version 3.00a.0

Dynamic Link Libraries	
VBRun300:	03.00.0538
MhRun400:	04.00.00.60
CommDlg:	3.11
TBxIns:	1.00.000
Ver:	3.11

Visual Basic Controls	
CMDialog:	2.1.0.5
MHSI200	04.00.00.7
MHTab:	04.00.00.53
ThreeD:	3.0.1.0

The Help About ToolBox Apprentice Version Information Dialog Box

The **Help About ToolBox Apprentice** dialog “System Info” button provides detailed information about currently loaded software modules. VISI technical staff may ask that you refer to an item on this dialog box during a support session.

Audio ToolBox SDK

Overview

VISI's ***Audio ToolBox Software Development Kit (SDK)*** lets you process and convert audio file formats from your MS Windows and MS DOS applications. The ***Audio ToolBox SDK*** contains sample programs that use the libraries and DLLs. These programs were created using Visual Basic and Microsoft C version 8.0, and demonstrate usage under both Windows and DOS. The sample PCM conversion program performs a simple .WAV to .VOX conversion.

System Requirements

Audio ToolBox SDK is designed to run on a minimum system configuration through careful design and use of system resources. A minimum system configuration for ***Audio ToolBox SDK*** is as follows:

- ♦ An IBM PC/AT or compatible with a hard drive, graphics monitor (color or monochrome) and at least 640K of memory.
- ♦ MS-DOS 3.3 or above.
- ♦ Microsoft Windows 3.x and above (for DLL version).

Software Installation

Insert the diskette labeled ***Audio ToolBox SDK*** in the A: drive. Change to the A: drive by entering:

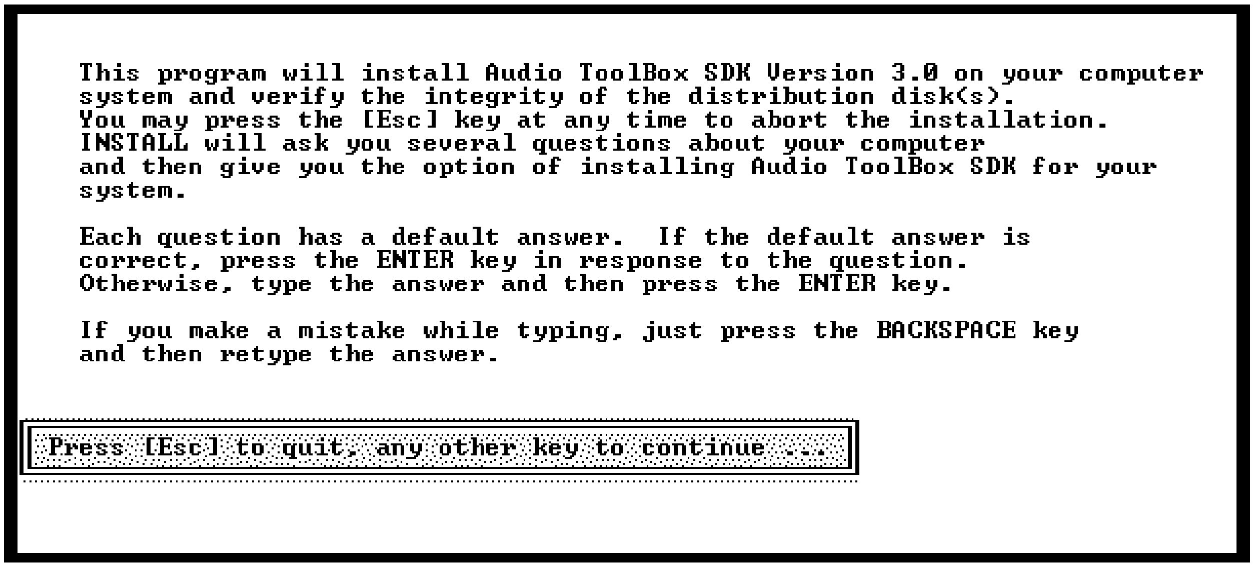
```
A: <ENTER>
```

To install ***Audio ToolBox SDK*** on your system, enter the following:

```
INSTALL <ENTER>
```

Audio ToolBox SDK supports many popular data formats. The Install program will ask you to specify the data format you use. Just follow the instructions on the screen!

After running Install, store your original ***Audio ToolBox SDK*** disks in a safe place. You're now ready to work with the ***Audio ToolBox SDK***.



```
This program will install Audio ToolBox SDK Version 3.0 on your computer
system and verify the integrity of the distribution disk(s).
You may press the [Esc] key at any time to abort the installation.
INSTALL will ask you several questions about your computer
and then give you the option of installing Audio ToolBox SDK for your
system.

Each question has a default answer.  If the default answer is
correct, press the ENTER key in response to the question.
Otherwise, type the answer and then press the ENTER key.

If you make a mistake while typing, just press the BACKSPACE key
and then retype the answer.


Press [Esc] to quit, any other key to continue ...
```

Audio ToolBox SDK Install program screen

The *Audio ToolBox SDK* Directories

The ***Audio ToolBox SDK*** directories contains all the files necessary to convert your files to and from a variety of audio file formats. In addition, the distribution diskette contains audio files you can use for sample conversions. The ***Audio ToolBox*** installation directory is organized into subdirectories. The following briefly describes the files and their uses:

♦ The MSCCvt directory contains a sample MS Windows and DOS program “CVTTST.C”. To build both the Windows and DOS versions, run the batch file “MAKEME.BAT”. The associated makefile assumes that Microsoft C 8.0 tools are installed and accessible.

♦ The CvtLib directory contains the “C” header and library files for various PCM formats. The libraries are named PPPMMM.LIB to designate the formats. PPP = DLG for Dialogic, G21 for C-ITU (CCITT) G.721, NWV for NewVoice, and PTC for Perception Technology. MMM = MLE for Microsoft Large memory model with floating point emulator, and MWE or Microsoft Windows model with floating point emulator.

♦ The MVBCvt directory contains a sample MS Windows Visual Basic project,”CVTTST.MAK”.

♦ The WINDLL directory contains the DLLs necessary for the MS Windows executable versions.

The CvtTst Program

The sample program performs a simple .WAV to .VOX conversion. To build the Microsoft Visual C CvtTst programs:

♦ Make sure the Microsoft C 8.0 tools are installed and accessible.

♦ Run the batch file MakeMe.bat to build both MS Windows and MS DOS versions of the sample program.

Visual Basic Interface

To build the Microsoft Visual Basic CvtTst project:

♦ Make sure Microsoft Visual Basic 3.0 or above is installed and accessible.

♦ Use “File... Open Project” to open “MVBCVT\CVTTST.MAK”.

♦ Use “File... Make Exe File...” to build the sample program.

Audio ToolBox SDK Functions

The following pages describe, in alphabetical order, the functions in the ***Audio ToolBox SDK***. The discussion of each function is divided into the following sections:

♦ Description: Illustrates the function calling syntax and summarizes the routine’s effect.

♦ Parameter: Describes function arguments and purpose.

♦ Returns: Specifies the function return value, if any.

♦ Remarks: Provides more detailed information on side effects and usage.

Audio Conversion Functions

Audio ToolBox SDK lets you convert to and from Microsoft Multimedia Wave (8, 16 & MS ADPCM), linear 16 and unsigned 8 bit, Dialogic 4 and 8 bit, plus other industry standard formats. Optionally choose volume adjustments, dynamic compression/ expansion and fast or high fidelity filtering and re-sampling algorithms.

The following descriptions provide detailed information for calling these functions from your application. The definitions of these functions, structures and constants are defined in the files GENCVT.H (for "C/C++" programmers) and GENCVT.TXT (for Visual Basic programmers).

CvtCfgIni

WORD FAR PASCAL CvtCfgIni (CVTBLK FAR *lpCvtBlk);

Convert Configuration Initialize: Initializes the Audio Conversion function block. Call this function to set all CVTBLK parameters to zero. Use this function for compatibility with future releases.

Parameter	Description
lpCvtBlk	Conversion data block.

Returns

0 if successful.

CvtPCMFil

DWORD FAR PASCAL CvtPCMFil (short sSrcHdl, short sDstHdl, LPCSTR szSrcRsv, LPCSTR szDstRsv, LPSTR lpWrkRsv, WORD usSizRsv, CVTBLK FAR *lpCvtBlk);

Convert PCM File: Convert to and from Microsoft Multimedia Wave (8, 16 & MS ADPCM), linear 16 and unsigned 8 bit, Dialogic 4 and 8 bit, plus other industry standard formats. Optionally choose volume adjustments, dynamic compression/ expansion and fast or high fidelity filtering and re-sampling algorithms.

Parameter	Description
sSrcHdl	Source file handle.

sDstHdl	Destination file handle
szSrcRsv	Reserved for future use. (Set to NULL)
szDstRsv	Reserved for future use. (Set to NULL)
lpWrkRsv	Reserved for future use. (Set to NULL)
usSizRsv	Reserved for future use. (Set to 0L)
lpCvtBlk	Conversion data block.

The conversion data block, CVTBLK, contains the following parameters for controlling processing options:

CVTBLK Parameter	Description
usSrcFmt	Source file format. Use this parameter to define the type of header information used by the input file. File formats supported in this release are:

Value	Meaning
ALLPURFMT	Pure (raw audio data).
WAVHDRFMT	Multimedia wave.

usSrcPCM	Source PCM type. Use this parameter to define the type of audio data contained in the input file. Note that this parameter is ignored (and may be set to zero) when a valid PCM type can be determined from the source file header. PCM types supported in this release are:

Value	Meaning
DLGPCM004	Dialogic 4 bit (OKI).
DLGPCM008	Dialogic 8 bit (u-Law).
G11PCMF08	C-ITU (CCITT) G.711 u-Law 8 bit.
G11PCMI08	C-ITU (CCITT) G.711 a-Law 8 bit.
G21PCM004	C-ITU (CCITT) G.721 4 bit ADPCM.
MPCPCM008	Multimedia 8 bit.
MPCPCM016	Multimedia 16 bit.
NWVPCM001	NewVoice CVSD 1 bit.
PTCPCM001	Perception Technology CVSD 1 bit.

usSrcChn	Source file channel count. Use this parameter to define the number of audio channels contained within the audio data. Note that this parameter is ignored (and may be set to zero) when a valid file channel count can be determined from the source file header.
usSrcBIO	Source bit encoding. Use this parameter to control bit and byte ordering requirements of the input file. Indicate that this option is active by logically “OR-ing” FIOENCSWP with one or more of the following bit masks:

Value	Meaning
FIOSWPBIT	Swap single bits.
FIOSWPNIB	Swap 4-bit nibbles.
FIOSWPBYT	Swap 8-bit bytes.
FIOSWPWRD	Swap 16-bit words

ulSrcFrq	Source sample frequency in Hertz. Use this parameter to define the sample frequency of the input file. Note that this parameter is ignored (and may be set to zero) when a valid sample frequency can be determined from the source file header. May be set to zero to use default frequency as defined by the file header or implied by the PCM data type.
ulSrcOff	Source byte offset in bytes. Use this parameter to skip over a section of the input file.
ulSrcLen	Source byte length in bytes. Use this parameter to limit the length of the input data. Set to hexadecimal FFFFFFF for whole file.
mhSrcCfg	Source configuration memory buffer. Reserved for future use.
usDstFmt	Destination file format. Use this parameter to define the type of header information used by the output file. File formats supported in this release are:

Value	Meaning
ALLPURFMT	Pure (raw audio data).
WAVHDRFMT	Multimedia wave.

usDstPCM	Destination PCM type. Use this parameter to define the type of audio data contained in the output file. PCM types supported in this release are:

Value	Meaning
DLGPCM004	Dialogic 4 bit (OKI).
DLGPCM008	Dialogic 8 bit (u-Law).
G11PCMF08	C-ITU (CCITT) G.711 u-Law 8 bit.
G11PCMI08	C-ITU (CCITT) G.711 a-Law 8 bit.
G21PCM004	C-ITU (CCITT) G.721 4 bit ADPCM.
MPCPCM008	Multimedia 8 bit.
MPCPCM016	Multimedia 16 bit.
NWVPCM001	NewVoice CVSD 1 bit.
PTCPCM001	Perception Technology CVSD 1 bit.

usDstChn	Destination file channel count. Reserved for future use.
usDstBIO	Destination bit encoding. Use this parameter to control bit and byte ordering requirements of the output file. Indicate that this option is active by logically “OR-ing” FIOENCSWP with one or more of the following bit masks:

Value	Meaning
FIOSWPBIT	Swap single bits.
FIOSWPNIB	Swap 4-bit nibbles.
FIOSWPBYT	Swap 8-bit bytes.
FIOSWPWRD	Swap 16-bit words

ulDstFrq	Destination sample frequency in Hertz. Use this parameter to define the sample frequency of the output file. May be set to zero to use default frequency as implied by the PCM data type.
ulDstOff	Destination byte offset in bytes. Reserved for future use.
ulDstLen	Destination byte length in bytes. Use this parameter to limit the length of the output data. Set to hexadecimal FFFFFFF for whole file.

mhDstCfg	Destination configuration memory buffer. Reserved for future use.
flCrpFrm	Frame length for auto-crop silence detector (typical value is 0.50 sec, defined by `CRPFRMDEF`). Use this setting to change the length of silence considered as separating distinct audio segments.
flCrpRes	Frame resolution for auto-crop silence detector (typical value is 0.01 sec, defined by `CRPRESDEF`). Use this setting to change the level of detail used when analyzing a file.
flCrpSnd	Auto-crop sound threshold level maximum percentage (typical value is 2% of max, defined by `CRPSNDDEF`). Use this setting to change the audio level interpreted as “sound”. Higher values result in higher immunity to noise, but lower sensitivity to sound.
usCrpAtk	Sensitivity attack ratio for auto-crop silence detector (typical value is 20% of frame, defined by `CRPATKDEF`). Use this setting to change the amount of sound that must be detected to recognize a transition from a region of silence to a region of active audio. Higher values result in higher immunity to noise, but increase the possibility of losing a portion of the beginning of active audio.
usCrpDcy	Sensitivity decay ratio for auto-crop silence detector (typical value is 10% of frame, defined by `CRPDCYDEF`). Use this setting to change the amount of sound that must be detected to recognize a transition from a region of active audio to a region of silence. Higher values result in higher immunity to noise, but increase the possibility of leaving a larger portion of silence at the end of active audio.
flCrpGrd	Auto-crop sound guard time (typical value 0.05 sec, defined by `CRPGRDDEF`). Increase the guard time for more silence at the beginning and end.
flVolAdj	Volume adjust multiplier (+/-) (typical value is 0 dB). Use this setting to increase the volume level of the destination audio file.
usNrmLvl	Volume normalize (typical value is 1.0% over range, defined by `NRMLVLDEF`). Use this setting to adjust the volume levels across multiple files.

flNrmExc	Volume normalize exception threshold (typical value is 0.2%, defined by `NRMLEXCDEF`). Since most real audio waveforms are not symmetrical, this parameter allows the volume level to go over target range some percentage of the time in order to achieve a better overall level. Set this parameter to zero to force no portion of this signal to go over the target level.
usNrmMax	Volume normalize maximum adjustment level (typical value is 6.0 dB,defined by `NRMMAXDEF`). This parameter limits the maximum level that the volume can be increased or decreased during the normalization process
flDCELvl	Volume dynamic compressor/ expander level (typical value is 6.0 dB, defined by `DCELVLDEF`). Use this setting to improve the dynamic range of the destination audio file.
usCmpThr	Volume dynamic compressor/ expander compressor threshold (typical value is 80% of max, defined by `CMPTHRDEF`). Use typical value for best results.
usCmpAtk	Volume dynamic compressor/ expander compressor attack time (typical value is 50 Hz, defined by `CMPATKDEF`). Use typical value for best results.
flCmpMin	Volume dynamic compressor/ expander compressor minimum (typical value is 6 dB, defined by `DCELVLDEF`). Use typical value for best results.
usExpThr	Volume dynamic compressor/ expander expander threshold (typical value is 30% of max, defined by `EXPTHRDEF`). Use typical value for best results.
usExpAtk	Volume dynamic compressor/ expander expander attack time (typical value is 5 Hz, defined by `EXPATKDEF`). Use typical value for best results.
flExpMax	Volume dynamic compressor/ expander expander maximum (typical value is 6 dB, defined by `DCELVLDEF`). Use typical value for best results.
usCE_Dcy	Volume dynamic compressor/ expander decay time (typical value is 5Hz, defined by `CE_DCYLVLDEF`). Use typical value for best results.

usNoiThr	Volume dynamic compressor/ expander noise gate threshold. (typical value is 2% of max, defined by `NOITHRDEF`). Use typical value for best results.
flNoiAtt	Volume dynamic compressor/ expander noise gate attenuation (typical value is 3 dB, defined by `NOIATTDEF`). Use typical value for best results.
usTDLAAF	Time domain anti-aliasing filter order (typical value is 2nd order). Use this parameter to set the quality level of the time domain anti-aliasing filter.

Value	Meaning
AAF_Q2TYP	2nd order filter.
AAF_Q6TYP	6th order filter.

bfTDLFst	Time domain anti-aliasing filter fast math on/off (typical value is `FALSE`). Set `TRUE` for improved speed on 286/386 class computers.
bfFFTDTM	DTMF touch-tone FFT filter on/ off (typical value is `FALSE`) Set `TRUE` to remove touch-tone frequencies during conversion.
bfFFTAAF	Low pass anti-aliasing FFT filter on/ off (typical value is `FALSE`). Set `TRUE` for improved fidelity.
bfFFTFEq	FFT Frequency Equalize on/off (typical value is `FALSE`). Set `TRUE` to activate frequency equalizer parameters described below.
bfFFTRes	Re-sample FFT filter on/ off (typical value is `FALSE`). Use the `TRUE` setting for improved fidelity.
usFFTOrd	FFT filter size (typical value is 9, defined by `FFTORDDEF`). Use typical value for best results.
flFEqGai	Frequency equalizer overall gain adjust (typical value is 0 dB). Use this setting to adjust the overall volume level during and equalization request.
ulFEqBot	Frequency equalizer bottom frequency (typical value is 0 Hz). Use this parameter to set the bottom band of the frequency equalizer data points settings.

ulFEqTop	Frequency equalizer top frequency (typical value of 0 Hz uses top Nyquist frequency of file). Use this parameter to set the top band of the frequency equalizer data points settings.
usFEqCnt	Frequency equalizer point count. Number of data points in the frequency equalizer `flFEqPnt` data array.
flFEqPnt	Frequency equalizer data points (typical value is between ±12 dB for each band). Use this setting to build a series of band adjustment levels.
mhGloCfg	Global configuration memory buffer. Reserved for future use.
lpPolDsp	Conversion poll display procedure. Reserved for future use.
ulRsvPol	Reserved for future use.
lpMsgDsp	Reserved for future use.
ulRsvErr	Reserved for future use.
ucRsv001	Reserved for future use.
usCmpCod	Conversion completion code. Reserved for future use.
ucRsv002	Reserved for future use.

Returns

Number of bytes converted.

Remarks

This function supports all the conversion options available in the command line version of the PCMCvt program. See the C and Visual Basic sample files for additional information on the CVTBLK parameters.

CvtPCMIni

WORD FAR PASCAL CvtPCMIni (WORD usReqTyp, DWORD ulPrm001, DWORD ulPrm002);

Convert PCM Initialize: Initialize support for the conversion functions of the ***Audio ToolBox SDK***. This function must be called before using any audio conversion processing functions.

Parameter	Description
usReqTyp	Reserved for future use. (Set to 0)

ulPrm001	Reserved for future use. (Set to 0L)
ulPrm002	Reserved for future use. (Set to 0L)

Returns

0 if successful. The following pre-defined values can be used to determine the return code status:

Value	Meaning
SI_CVTNO_ERR	Success
SI_CVTKEYERR	Key/License error, 8 second limit
SI_CVTVERERR	Module version mismatch

CvtPCMTrm

WORD FAR PASCAL CvtPCMTrm (void);

Convert PCM Terminate: Terminate support for the conversion functions of the ***Audio ToolBox SDK***. This function should be called prior to program termination.

Parameters

None

Returns

0 if successful.

CvtPCMVer

WORD FAR PASCAL CvtPCMVer (void);

Convert PCM Version: Return the ***Audio ToolBox SDK*** audio conversion version number. This function may be called at any time.

Parameters

None

Returns

The version number as hexadecimal 0xAABB where AA equals the major version, and BB equals the minor version number.

Sound Chop Functions

Audio ToolBox allows you to record a continuous stream of prompts into a single file, and then automatically break the single file into individual files. Sound chop separates a single file into individual files based upon periods of silence used as a separator. The resulting individual files can then be converted, volume normalized, and packed or converted using other ***Audio ToolBox*** utilities.

The following descriptions provide detailed information for calling these functions from your application. The definitions of these functions, structures and constants are defined in the files GENCHP.H (for "C/C++" programmers) and GENCHP.TXT (for Visual Basic programmers).

ChpCfgIni

WORD FAR PASCAL ChpCfgIni (CHPBLK FAR *lpChpBlk);

Chop Configuration Initialize: Initializes the Sound Chop function block. Call this function to set all CHPBLK parameters to zero. Use this function for compatibility with future releases.

Parameter	Description
lpChpBlk	Sound chop data block.

Returns

0 if successful.

ChpSndFil

DWORD FAR PASCAL CvtPCMFil (short sSrcHdl, LPSTR szSrcRsv, CHPOPNCBK lpOpnCBk, CHPCOPCBK lpCopCBk, CHPCLSCBK lpClsCBk, DWORD ulCBkDat, CHPBLK FAR *lpChpBlk);

Chop Sound File: ***Audio ToolBox*** allows you to record a continuous stream of prompts into a single file, and then automatically break the single file into individual files. This function separates a single file into individual files based upon periods of silence used as a separator. The resulting individual files can then be converted, volume normalized, and packed or converted using other ***Audio ToolBox*** utilities.

Parameter	Description
sSrcHdl	Source file handle.
szSrcRsv	Reserved for future use. (Set to NULL)
lpOpnCBk	Pointer to a function that opens a file and provides the sound chop function with the file handle of the file to contain the next sound segment.
lpCopCBk	Reserved for future use. (Set to NULL)
lpClsCBk	Pointer to a function that closes the current file containing the most recent sound segment.
ulCBkDat	A user defined 32-bit value that is passed to the “lpOpnCBk” and “lpClsCBk” functions. Use this parameter to pass instance specific data to the callback functions.
lpChpBlk	Sound chop data block.

The sound chop data block, CHPBLK, contains the following parameters for controlling processing options:

CHPBLK

Parameter	Description
usSrcFmt	Source file format. Use this parameter to define the type of header information used by the input file. File formats supported in this release are:
usSrcPCM	Source PCM type. Use this parameter to define the type of audio data contained in the input file. Note that this parameter is ignored (and may be set to zero) when a valid PCM type can be determined from the source file header. PCM types supported in this release are:

usSrcFmt values:

Value	Meaning
ALLPURFMT	Pure (raw audio data).
WAVHDRFMT	Multimedia wave.

usSrcPCM values:

Value	Meaning
DLGPCM004	Dialogic 4 bit (OKI).
DLGPCM008	Dialogic 8 bit (u-Law).
G11PCMF08	C-ITU (CCITT) G.711 u-Law 8 bit.

G11PCMI08	C-ITU (CCITT) G.711 a-Law 8 bit.
G21PCM004	C-ITU (CCITT) G.721 4 bit ADPCM.
MPCPCM008	Multimedia 8 bit.
MPCPCM016	Multimedia 16 bit.
NWVPCM001	NewVoice CVSD 1 bit.
PTCPCM001	Perception Technology CVSD 1 bit.

usSrcChn Source file channel count. Use this parameter to define the number of audio channels contained within the audio data. Note that this parameter is ignored (and may be set to zero) when a valid file channel count can be determined from the source file header.

usSrcBIO Source bit encoding. Use this parameter to control bit and byte ordering requirements of the input file. Indicate that this option is active by logically “OR-ing” FIOENCSWP with one or more of the following bit masks:

Value	Meaning
FIOSWPBIT	Swap single bits.
FIOSWPNIB	Swap 4-bit nibbles.
FIOSWPBYT	Swap 8-bit bytes.
FIOSWPWRD	Swap 16-bit words

ulSrcFrq Source sample frequency in Hertz. Use this parameter to define the sample frequency of the input file. Note that this parameter is ignored (and may be set to zero) when a valid sample frequency can be determined from the source file header. May be set to zero to use default frequency as defined by the file header or implied by the PCM data type.

ulSrcOff Source byte offset in bytes. Use this parameter to skip over a section of the input file.

ulSrcLen Source byte length in bytes. Use this parameter to limit the length of the input data. Set to hexadecimal FFFFFFF for whole file.

mhSrcCfg Source configuration memory buffer. Reserved for future use.

usDstFmt	Destination file format. Use this parameter to define the type of header information used by the output file. File formats supported in this release are:

Value	Meaning
ALLPURFMT	Pure (raw audio data).
WAVHDRFMT	Multimedia wave.

flChpFrm	Frame length for silence detector (typical value is 1.0 sec, defined by `CHPFRMDEF`). Use this setting to change the length of silence considered as separating distinct audio segments.
flChpRes	Frame resolution for silence detector (typical value is 0.01 sec, defined by `CHPRESDEF`). Use this setting to change the level of detail used when analyzing a file. Use typical value for best results.
flChpSnd	Sound threshold level maximum percentage (typical value is 2% of max, defined by `CHPSNDDEF`). Use this setting to change the audio level interpreted as “sound”. Higher values result in higher immunity to noise, but lower sensitivity to sound.
usChpAtk	Sensitivity attack ratio for silence detector (typical value is 20% of frame, defined by `CHPATKDEF`). Use this setting to change the amount of sound that must be detected to recognize a transition from a region of silence to a region of active audio. Higher values result in higher immunity to noise, but increase the possibility of losing a portion of the beginning of active audio.
usChpDcy	Sensitivity decay ratio for silence detector (typical value is 10% of frame, defined by `CHPDCYDEF`). Use this setting to change the amount of sound that must be detected to recognize a transition from a region of active audio to a region of silence. Higher values result in higher immunity to noise, but increase the possibility of leaving a larger portion of silence at the end of active audio.
flChpGrd	Auto-crop sound guard time (typical value 0.05 sec, defined by `CHPGRDDEF`). Increase the guard time for more silence at the beginning and end.

mhGloCfg	Global configuration memory buffer. Reserved for future use.
lpPolDsp	Sound chop poll display procedure. Reserved for future use.
ulRsvPol	Reserved for future use.
lpMsgDsp	Reserved for future use.
ulRsvErr	Reserved for future use.
ucRsv001	Reserved for future use.
usCmpCod	Reserved for future use.
ucRsv002	Reserved for future use.

Returns

Number of bytes of audio data written to the output file(s).

ChpSndIni

WORD FAR PASCAL ChpSndIni (WORD usReqTyp, DWORD ulPrm001, DWORD ulPrm002);

Chop Sound Initialize: Initialize support for the sound chop functions of the ***Audio ToolBox SDK***. This function must be called before using any sound chop processing functions.

Parameter	Description
usReqTyp	Reserved for future use. (Set to 0)
ulPrm001	Reserved for future use. (Set to 0L)
ulPrm002	Reserved for future use. (Set to 0L)

Returns

0 if successful. The following pre-defined values can be used to determine the return code status:

Value	Meaning
SI_CHPNO_ERR	Success
SI_CHPKEYERR	Key/License error, 8 section limit
SI_CHPVERERR	Module version mismatch

ChpSndTrm

WORD FAR PASCAL ChpSndTrm (void);

Chop Sound Terminate: Terminate support for the sound chop functions of the ***Audio ToolBox SDK***. This function should be called prior to program termination.

Parameters

None

Returns

0 if successful.

ChpSndVer

WORD FAR PASCAL ChpSndVer (void);

Chop Sound Version: Return the ***Audio ToolBox SDK*** sound chop version number. This function may be called at any time.

Parameters

None

Returns

The version number as hexadecimal 0xAABB where AA equals the major version, and BB equals the minor version number.

Indexed File Functions

Audio ToolBox supports both pure (raw amplitude data) and indexed format files. Pure files contain only digitized sound; indexed files contain groups of digitized sound, plus format and annotation text. These functions let you convert to and from pure and indexed file formats, as well as provide file maintenance capabilities.

The following descriptions provide detailed information for calling these functions from your application. The definitions of these functions, structures and constants are defined in the files GENIDX.H (for "C/C++" programmers) and GENIDX.TXT (for Visual Basic programmers).

IdxCfgIni

Reserved for future use.

IdxFillni

WORD FAR PASCAL IdxFillni (WORD usReqTyp, DWORD ulPrm001, DWORD ulPrm002);

Indexed File Initialize: Initialize support for the indexed file functions of the ***Audio ToolBox SDK***. This function must be called before using any indexed file processing functions.

Parameter	Description
usReqTyp	Reserved for future use. (Set to 0)
ulPrm001	Reserved for future use. (Set to 0L)
ulPrm002	Reserved for future use. (Set to 0L)

Returns

0 if successful. The following pre-defined values can be used to determine the return code status:

Value	Meaning
SI_IDXNO_ERR	Success
SI_IDXKEYERR	Key/License error, 8 second limit
SI_IDXVERERR	Module version mismatch

IdxFilTrm

WORD FAR PASCAL IdxFilTrm (void);

Indexed File Terminate: Terminate support for the indexed file functions of the ***Audio ToolBox SDK***. This function should be called prior to program termination.

Parameters

None

Returns

0 if successful.

IdxFilVer

WORD FAR PASCAL IdxFilVer (void);

Indexed File Version: Return the ***Audio ToolBox SDK*** indexed file version number. This function may be called at any time.

Parameters

None

Returns

The version number as hexadecimal 0xAABB where AA equals the major version, and BB equals the minor version number.

Index Chop Functions

Index chop functions allow you to “chop” indexed files into pure audio and text files that can be batch processed, converted and edited. The following function descriptions provide detailed information for calling these functions from your application.

ChpIdxIni

WORD FAR PASCAL ChpIdxIni (CHIBLK FAR *lpChIBlk);

Chop Index Initialize: Initializes the Index Chop function block. Call this function to set all CHIBLK parameters to zero. Use this function for compatibility with future releases.

Parameter	Description
lpChIBlk	Index chop data block.

Returns

0 if successful.

ChpIdxFil

WORD FAR PASCAL ChpIdxFil (short sSrcHdl, LPSTR szVoxSpc, LPSTR szTxtSpc, CHICHKCBK lpChkCBk, LPVOID lpRsv001, CHIENDCBK lpEndCBk, DWORD ulCBkDat, CHIBLK FAR *lpChIBlk)

Chop Index File: This function allows you to “chop” indexed files into pure audio and text files that can be batch processed, converted and edited.

Parameter	Description
sSrcHdl	Source file handle.
szVoxSpc	Destination audio file extension (typical value is “.vox”). Use this setting to change the file name extension of the newly created audio files.
szTxtSpc	Destination text file extension (typical value is “.txt”). Use this setting to change the file name extension of the newly created text files.
lpChkCBk	Pointer to a function that is called to indicate current index file segment that is being processed. This function should return TRUE for future compatibility.
lpRsv001	Reserved for future use. (Set to NULL)
lpEndCBk	Pointer to a function that is called to indicate current index file segment has completed processing. This function should return TRUE for future compatibility.
ulCBkDat	A user defined 32-bit value that is passed to the “lpChkCBk” and “lpEndCBk” functions. Use this parameter to pass instance specific data to the callback functions.
lpChIBlk	Index chop data block.

The conversion data block, CHIBLK, contains parameters for controlling processing options.

CHIBLK

Parameter	Description
lNamOff	Destination file name counter offset (typical value is 0). Use this setting to change the beginning numeric name of the newly created audio and text files.
ulIdxPos	Chop @ index position (beginning of file is 1, end is 9999). Use this setting to limit the extraction to a range of indexes.
ulIdxCnt	Number of indexes to chop (0 indicates all indexes). Use this setting to limit the extraction to a range of indexes.
lpPolDsp	Index chop poll display procedure. Reserved for future use.
ulRsvPol	Reserved for future use.
lpMsgDsp	Reserved for future use.
ulRsvErr	Reserved for future use.
ucRsv001	Reserved for future use.

Returns

0 if successful.

Index List Functions

Index list functions allow you to list the header and content information of an indexed file. The following function descriptions provide detailed information for calling these functions from your application.

LstIdxIni

WORD FAR PASCAL LstIdxIni (LSIBLK FAR *lpLsIBlk);

List Index Initialize: Initializes the Index List function block. Call this function to set all LSIBLK parameters to zero. Use this function for compatibility with future releases.

Parameter	Description
lpLsIBlk	Index list data block.

Returns

0 if successful. The following pre-defined values can be used to determine the return code status:

Value	Meaning
SI_IDXNO_ERR	Success
SI_IDXKEYERR	Key/License error, 8 second limit
SI_IDXVERERR	Module version mismatch

LstIdxFil

DWORD FAR PASCAL LstIdxFil (short sSrcHdl, LPCSTR szSrcRsv, LPSTR lpWrkBuf, WORD usBufSiz, LSIBLK FAR *lpLstBlk)

List Index File: This function allows you to list the header and content information of an indexed file to the standard output device.

Parameter	Description
sSrcHdl	Source file handle.
szSrcRsv	Reserved for future use. (Set to NULL)
lpWrkBuf	Temporary work buffer. (recommended size is 1024 bytes).
usBufSiz	Size of lpWrkBuf temporary work buffer (typical value is 1024).
lpLsIBlk	Index List data block.

The conversion data block, LSIBLK, contains parameters for controlling processing options.

LSIBLK

Parameter	Description
usLstFlg	List request flags
lpMsgDsp	Reserved for future use.

Value	Meaning
GENINHFLG	Inhibit general information display flag.
VOXINHFLG	Inhibit vox information display flag.
TXTINHFLG	Inhibit text information display flag.

ulRsvErr	Reserved for future use.
ucRsv001	Reserved for future use.

Returns

0 if successful.

Index Pack Functions

Index pack functions allow you to “pack” pure audio and text files into an indexed file. The following function descriptions provide detailed information for calling these functions from your application.

PakIdxIni

WORD FAR PASCAL PakIdxIni (PKIBLK FAR *lpPkIBlk);

Pack Index Initialize: Initializes the Index Pack function block. Call this function to set all PKIBLK parameters to zero. Use this function for compatibility with future releases.

Parameter	Description
lpPkIBlk	Index pack block.

Returns

0 if successful.

PakIdxFil

WORD FAR PASCAL PakIdxFil (LPSTR szVoxSpc, LPSTR szTxtSpc, LPSTR szDstSpc, PKICHKCBK lpChkCBk, LPVOID lpRsv001, PKIENDCBK lpEndCBk, DWORD ulCBkDat, PKIBLK FAR *lpPkIBlk)

Pack Index File: This function allows you to “pack” pure audio and text files into an indexed file.

Parameter	Description
szVoxSpc	Destination audio file extension (typical value is “.vox”). Use this setting to change the file name extension of the newly created audio files.

szTxtSpc	Destination text file extension (typical value is ".txt"). Use this setting to change the file name extension of the newly created text files.
szDstSpc	Destination file specification (typical value is "Index.Vap"). Use this setting to set the output file drive, path and file name.
lpChkCBk	Pointer to a function that is called to indicate current index file segment that is being processed. This function should return TRUE for future compatibility.
lpRsv001	Reserved for future use. (Set to NULL)
lpEndCBk	Pointer to a function that is called to indicate current index file segment has completed processing. This function should return TRUE for future compatibility.
ulCBkDat	A user defined 32-bit value that is passed to the "lpChkCBk" and "lpEndCBk" functions. Use this parameter to pass instance specific data to the callback functions.
lpPkIBlk	Index pack data block.

The conversion data block, PKIBLK, contains parameters for controlling processing options.

PKIBLK Parameter	Description
sSrtTyp	Sort files; +1 up, -1 down, 0 unsorted. (typical value is +1). Use this setting to pack the files in descending or unsorted directory order.
ulSmpFrq	Set header sample frequency to value. (typical value is 6000). Use this setting to change the frequency stored in the header of the indexed file. This setting DOES NOT change the audio data.
lpPolDsp	Index pack poll display procedure. Reserved for future use.
ulRsvPol	Reserved for future use.
lpMsgDsp	Reserved for future use.
ulRsvErr	Reserved for future use.
ucRsv001	Reserved for future use.

Returns

0 if successful.

Index Rebuild Functions

Index rebuild functions allow you to lengthen, shorten, replace entries and "rebuild" indexed files. The following function descriptions provide detailed information for calling these functions from your application.

RebIdxIni

WORD FAR PASCAL RebIdxIni (RBIBLK FAR *lpRbIBlk);

Rebuild Index Initialize: Initializes the Index Rebuild function block. Call this function to set all RBIBLK parameters to zero.

Parameter	Description
lpRbIBlk	Index rebuild block.

Returns

0 if successful.

RebIdxFil

DWORD FAR PASCAL RebIdxFil (short sSrcHdl, short sDstHdl, LPCSTR szSrcRsv, LPCSTR szDstRsv, LPSTR lpWrkBuf, WORD usBufSiz, RBIBLK FAR *lpRbIBlk);

Rebuild Index File: This function allows you to lengthen, shorten, replace entries and "rebuild" indexed files.

Parameter	Description
sSrcHdl	Source file handle.
sDstHdl	Destination file handle.
szSrcRsv	Reserved for future use. (Set to NULL)
szDstRsv	Reserved for future use. (Set to NULL)
lpWrkBuf	Temporary work buffer. (recommended size is 1024 bytes).
usBufSiz	Size of lpWrkBuf temporary work buffer (typical value is 1024).

lpRbIBlk	Index Rebuild data block.

The conversion data block, RBIBLK, contains parameters for controlling processing options.

RBIBLK

Parameter	Description
sShfCnt	Add (+ value) or remove (- value) the specified number of indexes (typical value is none).
usAt_Pos	Perform operations @ index position (typical value is 1, end is 9999). Use this setting to limit operations to a range of indexes.
ulSmpFrq	Set header sample frequency to value NNNN position (typical value of 0 leaves original header frequency setting unchanged). Use this setting to change the frequency stored in the header of the indexed file. This setting DOES NOT change the audio data.
ucRepVox	Replacement vox filename character string. Replace audio data @ position (typical value of 0 leaves original audio data unchanged). Use this parameter to replace the audio contents of an index with the raw data from the specified vox file.
ucRepTxt	Replacement text filename character string. Replace annotation text @ position (typical value of 0 leaves original text data unchanged). Use this parameter to replace the annotation text contents of an index with the raw data from the specified text file.
lpPolDsp	Index rebuild poll display procedure. Reserved for future use.
ulRsvPol	Reserved for future use.
lpMsgDsp	Reserved for future use.
ulRsvErr	Reserved for future use.
ucRsv001	Reserved for future use.

Returns

Number of bytes written to the new “rebuilt” file.

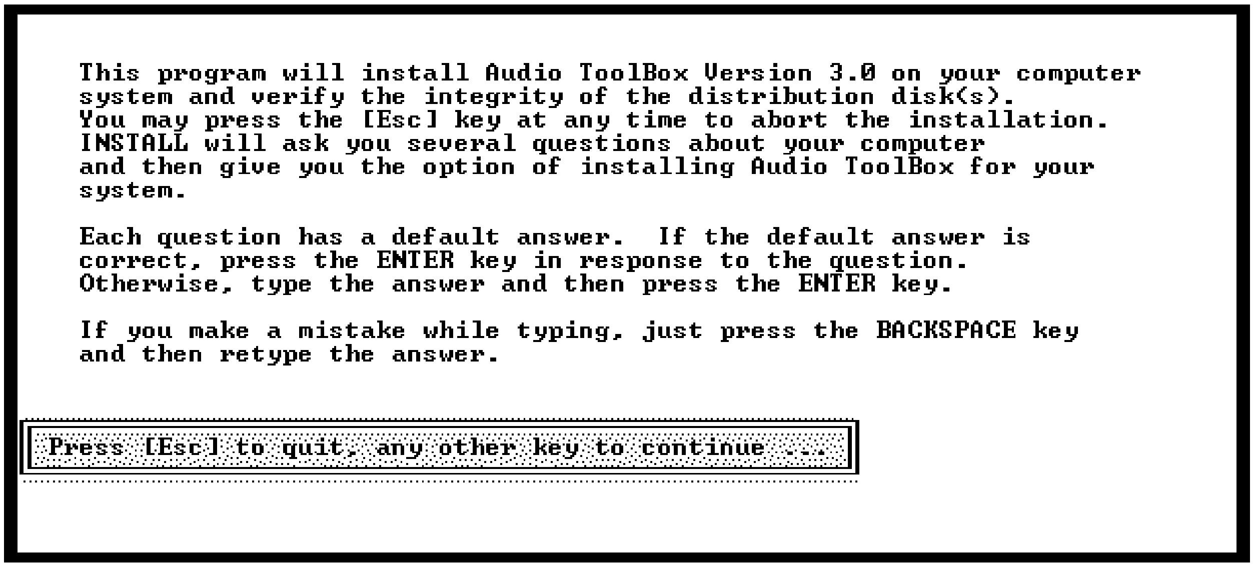
♦ Appendix

Appendix A, ***Installation and Setup*** tells you how to set up ***Audio ToolBox*** on your computer. This chapter covers system requirements, and prepares you for a sample processing session.

Appendix B, ***Configuration***, lists options that can be set in the ***Audio ToolBox*** initialization file.

Appendix C, ***Troubleshooting & Recovery***, contains information to help diagnose and correct problems if something unexpected occurs.

Installation & Setup



```
This program will install Audio ToolBox Version 3.0 on your computer
system and verify the integrity of the distribution disk(s).
You may press the [Esc] key at any time to abort the installation.
INSTALL will ask you several questions about your computer
and then give you the option of installing Audio ToolBox for your
system.

Each question has a default answer.  If the default answer is
correct, press the ENTER key in response to the question.
Otherwise, type the answer and then press the ENTER key.

If you make a mistake while typing, just press the BACKSPACE key
and then retype the answer.

Press [Esc] to quit, any other key to continue ...
```

Audio ToolBox Install program screen

This section provides basic instructions for installing ***Audio ToolBox***™. You must use the ***Audio ToolBox*** Install and Setup programs to install ***Audio ToolBox*** and ***ToolBox Apprentice***. The files on the program disks are compressed; you can't just copy them to your hard drive.

Audio ToolBox Software Installation

Insert the diskette labeled VISI ***Audio ToolBox*** in the A: drive. Change to the A: drive by entering:

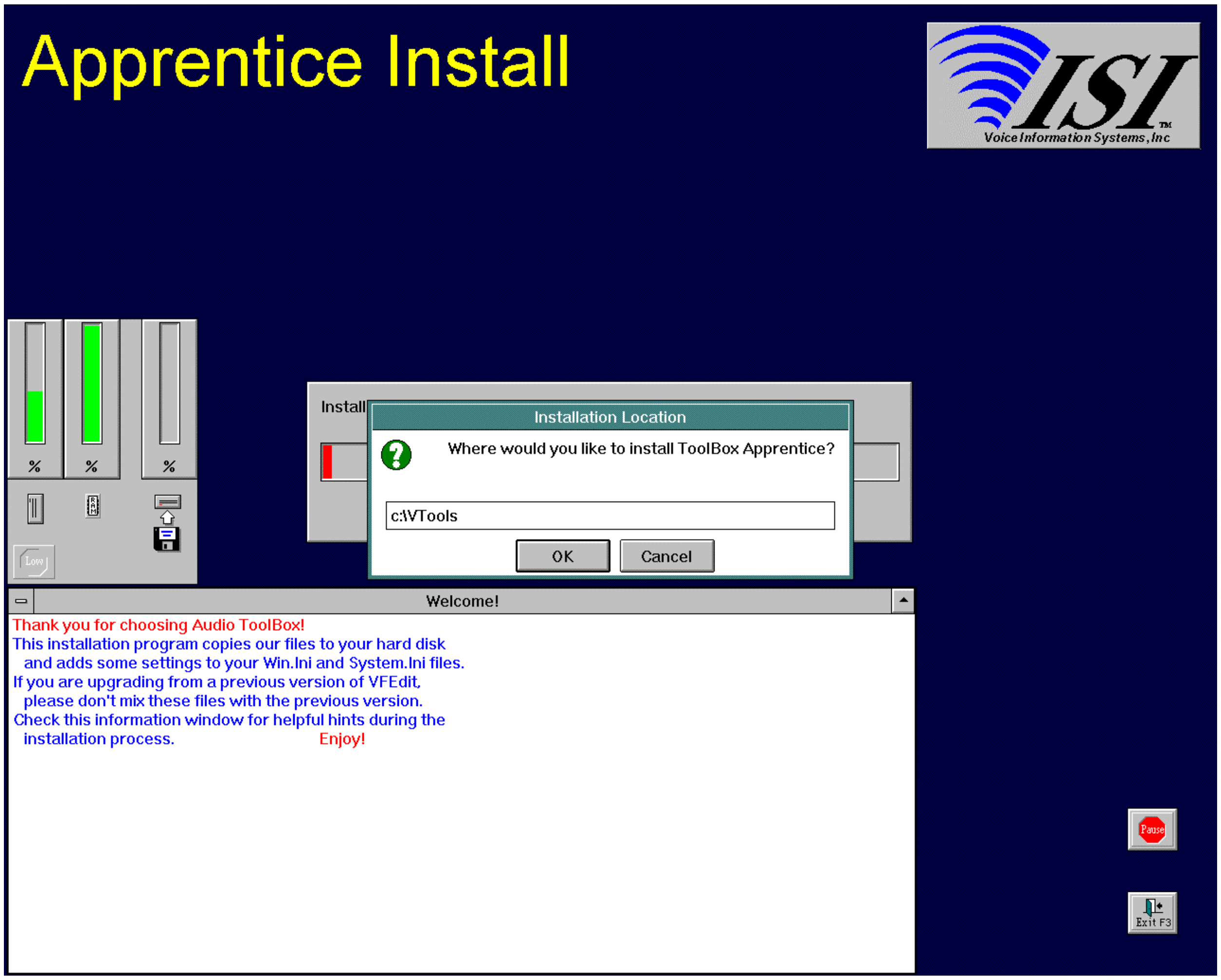
```
A: <ENTER>
```

To install ***Audio ToolBox*** on your system, enter the following:

```
INSTALL <ENTER>
```

Audio ToolBox supports many popular data formats. The install program, INSTALL, will ask you to specify the data format you use. Just follow the instructions on the screen!

After running INSTALL, store your original ***Audio ToolBox*** disks in a safe place. You're now ready to run ***Audio ToolBox***.



Where would you like to install ToolBox Apprentice?

c:\VTools

Thank you for choosing Audio ToolBox!
This installation program copies our files to your hard disk
and adds some settings to your Win.Ini and System.Ini files.
If you are upgrading from a previous version of VFEdit,
please don't mix these files with the previous version.
Check this information window for helpful hints during the
installation process. Enjoy!

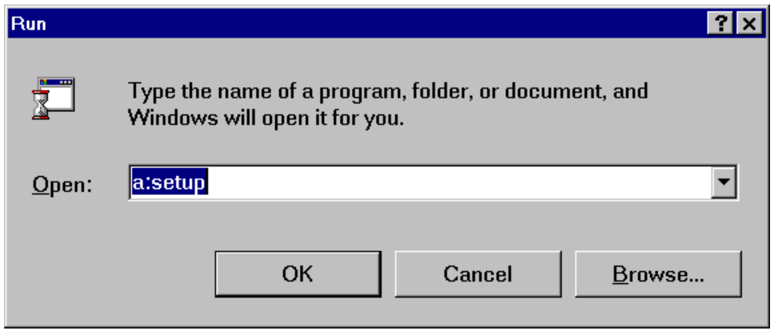
ToolBox Apprentice Setup program screen

ToolBox Apprentice Software Installation

Follow these basic steps to install ***Audio ToolBox Apprentice***. You can quit Setup at any time.

- Start Windows.
- Insert the diskette labeled "***Audio ToolBox Apprentice***" in drive A or drive B..
- Select "File Run" from the Windows 3.x Program Manager menu, or select "Start Run" from the Windows 95 task bar.
- Enter "A:\Setup" in the Windows dialog box.

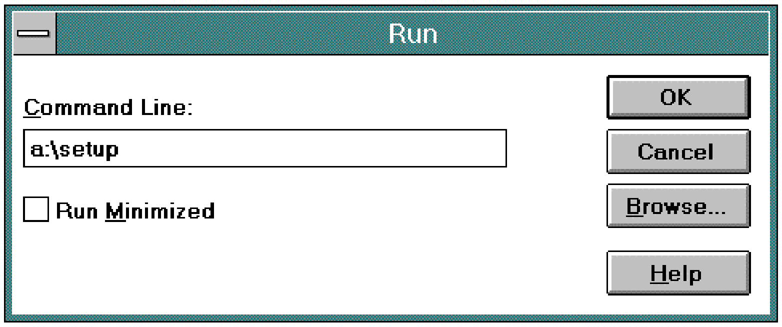
♦ Follow the Setup program instructions.


Run

Type the name of a program, folder, or document, and Windows will open it for you.

Open: a:setup

Windows 95 Start Run Dialog Box


Run

Command Line: a:\setup

Run Minimized

Windows 3.x File Run Dialog Box

After completing the installation procedure, the ***Audio ToolBox*** icons will appear within a new application group. After running Setup, store your original ***Audio ToolBox*** disks in a safe place.

Installing *Audio ToolBox* Hardware

Audio ToolBox works with or without the VISI Accelerator™ module connected to the parallel printer port. If the accelerator is not installed, however, file conversions are limited to approximately 8 seconds in length. To install, unpack the accelerator module and follow these simple connection instructions.

VISI Accelerator™ module

The VISI Accelerator module is a small plastic and metal box with sockets on both sides.

First find the parallel printer port on your computer. It may be shown as “LPT1” in you computer manual. If you already have a printer plugged in, temporarily unplug it, and plug the VISI Accelerator module in its place. Now plug the printer into the socket in the back of the adapter module. Make sure all the connections are tight. Most IBM compatible printer cables are equipped with screws to hold the plug in place. You may want to take a minute and tighten them securely; a loose plug may cause an access error.

Note: If your printer is plugged into the accelerator module, the printer must be turned on while the ***Audio ToolBox*** is operating. Otherwise, conversions may not complete properly.

The *Audio ToolBox* Directories

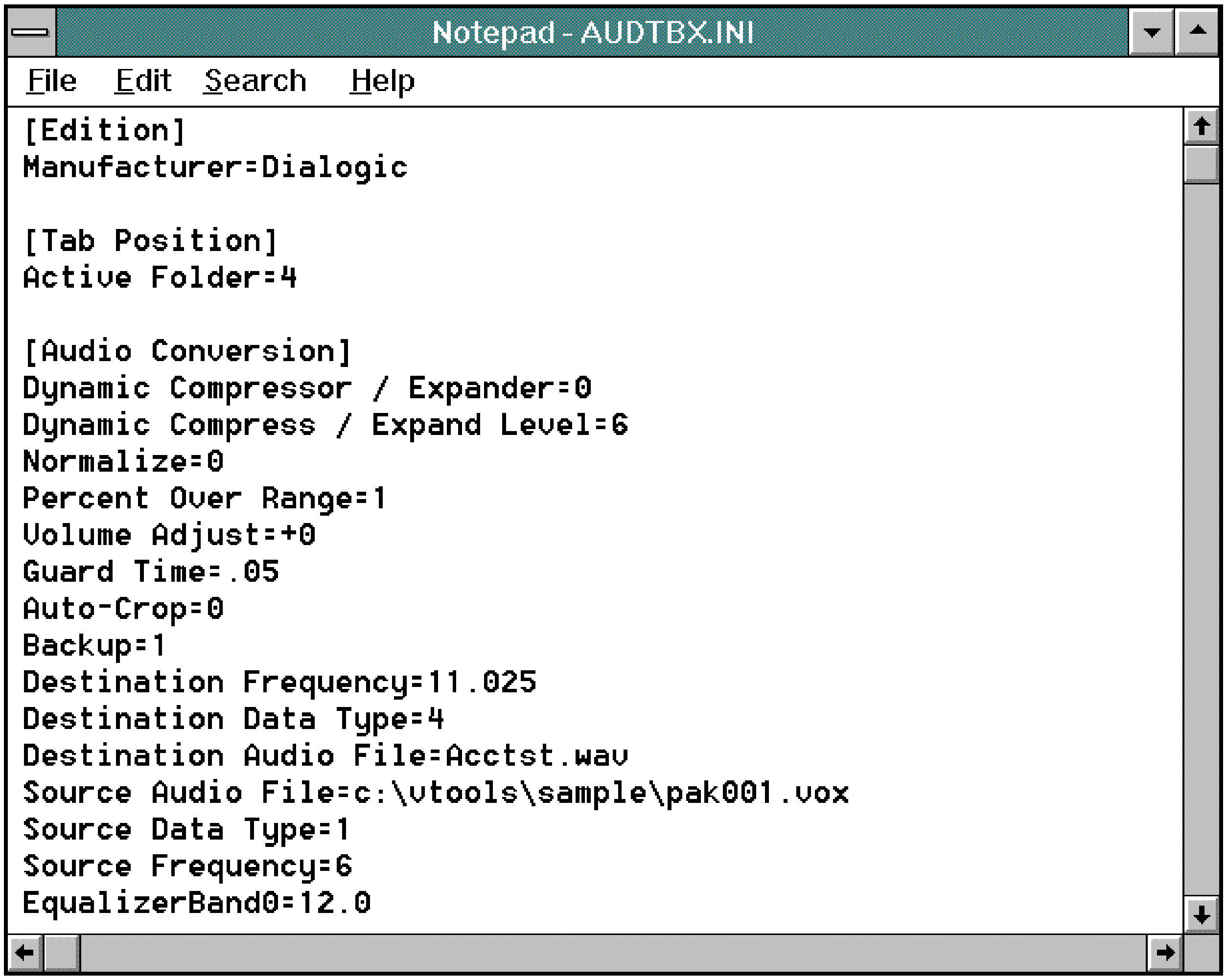
The ***Audio ToolBox*** directories contain all the files necessary to convert your files to and from the industry standard audio file formats. In addition, the installation directories contain audio files you can use for sample conversions.

The ***Audio ToolBox*** installation directory is organized into subdirectories. The following briefly describes the files and their uses:

- *.exe: Executable files that perform conversions.
- README.TXT: Information made available subsequent to the publication of this manual.
- SAMPLE*.*: Sample Dialogic and Multimedia audio files created with ***Audio ToolBox***; sample batch files.

After installing ***Audio ToolBox*** store your original disks in a safe place. You’re now ready to run ***Audio ToolBox***.

Configuration



```
[Edition]
Manufacturer=Dialogic

[Tab Position]
Active Folder=4

[Audio Conversion]
Dynamic Compressor / Expander=0
Dynamic Compress / Expand Level=6
Normalize=0
Percent Over Range=1
Volume Adjust=+0
Guard Time=.05
Auto-Crop=0
Backup=1
Destination Frequency=11.025
Destination Data Type=4
Destination Audio File=Acctst.wav
Source Audio File=c:\vtools\sample\pak001.vox
Source Data Type=1
Source Frequency=6
EqualizerBand0=12.0
```

Sample AUDTBX.INI Notepad Edit Session

The following sections discuss configuration issues for customizing operation of ***ToolBox Apprentice***.

- Changing AUDTBX.INI: Describes the purpose, structure and editing of the Microsoft Windows AUDTBX.INI files.
- ***ToolBox Apprentice*** Settings: Describes changes to the Microsoft Windows AUDTBX.INI file and when to make them.

AUDTBX.INI Settings

ToolBox Apprentice reads settings from the AUDTBX.INI file every time you start ***ToolBox Apprentice***. You can make changes directly to your AUDTBX.INI file by using any ASCII text editor. If using a word processor, be sure to save the file as an ASCII text file (unformatted).

When you edit AUDTBX.INI you will see a series of sections that appear in the following format:

```
[section name]
keyword1=setting1 setting2 ...
keyword2=setting1 setting2 ...
```

The section name precedes the setting for a particular Windows setting or application. The parameters for Screen Position, for example, are preceded by"[Screen Position]".

VISI technical staff may ask that you refer to an item on the following list during a support session. The following list itemizes profile sections and key strings and is for reference purposes only.

[Edition]

```
Manufacturer =
```

[Screen Position]

```
WindowState =
PositionLeft =
PositionTop =
```

[Tab Position]

```
Active Folder =
```

[Audio Conversion]

```
Dynamic Compressor / Expander =
Dynamic Compress / Expand Level =
Normalize =
Percent Over Range =
Volume Adjust =
Guard Time =
Auto-Crop =
Backup =
Destination Frequency =
Destination Data Type =
Destination Audio File=
Source Audio File =
Source Data Type =
Source Frequency =
EqualizerBand0 =
EqualizerBand1 =
EqualizerBand2 =
EqualizerBand3 =
EqualizerBand4 =
EqualizerBand5 =
EqualizerBand6 =
EqualizerBand7 =
Swap Input =
Swap Output =
Byte Position =
Byte Length =
Configuration Global =
Configuration Input =
Configuration Output =
```

[Index Rebuild]

```
Begin at Index # =1
Replace Text with File =
Replace Vox with File =
Delete Count =
Insert Count =
    Frequency =
    Backup =
    Destination Index File =
    Source Index File =
```

[Index List]

```
    Source Index File =
    Header Information Inhibit =
    Text Listing Inhibit =
    Audio Information Inhibit =
```

[Index Chop]

```
    Source Index File =
    Text Extension =
    Vox Extension =
    Number of Indexes =
    File Name Offset =
    Begin at Index# =
```

[Index Pack]

```
    Frequency =
    Sort Order =
    Text Source =
    Destination Index File =
    Source Vox File =
```

[Sound Chop]

```
    Sound Threshold =
    Destination File Extension =
    Guard Time =
    Source Frequency =
    Source Data Type =
    Source Audio File =
```

[Audio ToolBox]

```
Command Line Batch File =
```

[Environment Options]

```
Program Files =
Execution Mode =
```

[Module Verification]

```
(file name of software module) =
```

Troubleshooting & Recovery

System Configuration

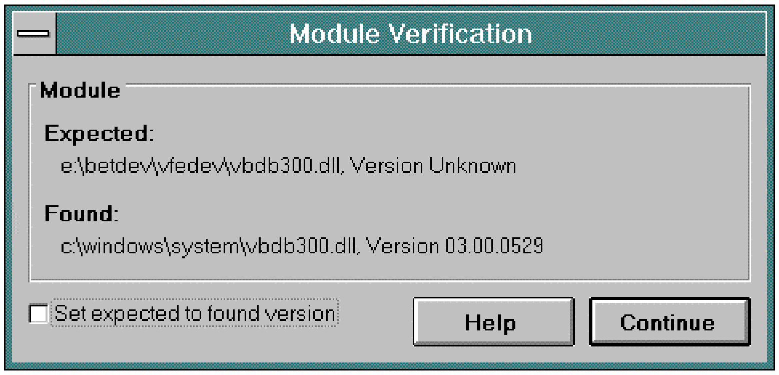
Audio ToolBox is tested using a wide variety of system configurations (both hardware and software), and every attempt is made to ensure that ***Audio ToolBox*** will work correctly with popular displays, TSR's, memory boards, and related products.

If you experience difficulty in running ***Audio ToolBox***, you may want t o double-check your system configuration. Because of the number and variety of add-on devices and software packages, it is possible to get side effects resulting from the interaction of these entities.

The most unusual errors occur as a result of conflicts between the audio board and the hardware/software interrupts of other system components. Double (and triple) check to ensure that the audio boards and drivers are not contending for these important resources. Peculiar and bizarre system behaviors have been reported when this occurs—and is often not detected by diagnostic software because of the intermittent nature of the device conflicts.

Quickest results are obtained by removing any extraneous boards, TSR's, and device drivers. Double check your system documentation to assure that the system boards do not conflict with any other audio hardware interrupts. This usually solves the problem. If the problem persists, please provide us with as much information as possible so that we can reproduce and help you eliminate the problem.

Module Verification


Module Verification

Module

Expected:

e:\betdev\vfedev\vbdb300.dll, Version Unknown

Found:

c:\windows\system\vbdb300.dll, Version 03.00.0529

☐ Set expected to found version

Module Verification dialog box

Audio ToolBox has detected a mismatch between the version numbers of shareable software components. ***Audio ToolBox*** installs shareable software components into its own directory on your hard disk. Some other program has loaded a different version of this component into memory. In many cases, the difference between versions will not affect normal operation. You should, however, note the version numbers of the modules found in memory if the product behaves in a peculiar manner.

♦ Module Expected: This is the expected version of the shareable software component.

- Found: This is the version of the shareable software component currently loaded in memory. Please note this version number for future reference.
- Set expected to found version: Once you have verified that the product is behaving normally, you set the loaded version number to the expected version number. That will prevent this message from appearing when the found version is already loaded in memory.

File Recovery

Unlike most other conversion utilities, ***Audio ToolBox*** does not load a copy of its data into memory when you process a sound file—all conversion is done directly on the data stored on the hard disk. This allows you to recover any audio that would otherwise be lost if you experience a power failure or system error. In addition, ***Audio ToolBox*** creates a backup file of the any file that is to be overwritten unless you specifically ***Audio ToolBox*** not to do so.

All ***Audio ToolBox*** backup audio file names end with the letters “BK1”. You can re-open any one of these files and re-use them. Be sure to use the same data format and frequency as that of the original file.

Known System Conflicts

There are no known system conflicts at this time.

Technical Support

VISI offers a number of comprehensive technical support options for users. For more information, call 1-310-392-8780. VISI support services are subject to VISI prices, terms, and conditions in place at the time the service is used.

When you call...

When you call, you should be at your computer and have the appropriate product documentation at hand. Be prepared to give the following information:

- The version number of the VISI product you are using
- The type of hardware you are using
- The exact wording of any messages that appeared on your screen.

- What happened and what you were doing when the problem occurred.
- How you tried to solve the problem.

Setup and Installation

For setup and installation assistance with VISI products, dial 310-392-6266. This service is designed to get you up and running quickly with our products.

How-to and in-depth support

For how-to assistance and in-depth support for advanced issues, you can obtain support by calling our credit card line for at the rate of $20 per call. To use this service, call 1-800-469-4874. Please have your Visa or MasterCard ready.

Download Service

Access technical notes and supplementary files covering common VISI product support issues are available on our website at **www.vinfo.com** and via modem on the VISI Support BBS at 310-392-6610. These services are available 24 hours a day, seven days a week.

♦ Glossary

adaptive delta pulse code modulation (ADPCM): An audio compression algorithm for digital audio based on describing level differences between adjacent samples.

aliasing: Undesired frequencies that are produced when harmonic components of an audio signal being sampled by a digital recording device (or harmonic components generated within a digital sound source) lie above the Nyquist frequency. Aliasing differs from some other types of noise in that its pitch changes radically when the pitch of the intended sound changes.

amplitude: The amount of a signal. Amplitude is measured by determining the amount of fluctuation in air pressure (of a sound), voltage (of an electrical signal), or numerical data (in a digital application). When the signal is in the audio range, amplitude is perceived as loudness.

analog: Of, relating to, or being a mechanism in which data is represented by continuously variable physical quantities

analog-to-digital converter (ADC): A device that changes the continuous fluctuations in voltage from an analog device (such as a microphone) into digital information that can be stored or processed in a sampler, digital signal processor, or digital recording device.

application: A sub-program in the Windows environment, such as ***VFEdit***®, or Microsoft Word for Windows.

attenuate: To reduce the level of a signal.

b: An abbreviation for byte.

B: An abbreviation for bel.

batch process: A technique that permits more than one file or operation to be performed with a single command.

bel: Named for Alexander Graham Bell, who did the original scientific investigations; Also see decibel.

byte: A group of eight binary digits processed as a unit by a computer and used especially to represent an alpha-numeric character; also see word.

carrier: A signal that is modulated by some other signal, as in FM synthesis.

CCITT: Consultative Committee on International Telephone and Telegraph. The CCITT was an international standards committee now known as the ITU.

C-ITU: Committee of the International Telecommunication Union. The International Telecommunication Union (ITU) is group that sets international communications standards. They sponsor committees to provide detailed technical specifications for those standards.

clangorous: Sounds containing partials that are not part of the natural harmonic series. Clangorous tones often sound bell-like.

companding: A type of signal processing in which the signal is compressed on input and expanded back to its original form on output. Digital companding enables a device to achieve a greater apparent dynamic range with fewer bits per sample word.

cutoff frequency: The point in the frequency spectrum beyond which a synthesizer filter attenuates the audio signal being sent through it.

dB: See decibel.

decibel: A unit of measurement used to indicate audio power level. Technically, a decibel is a logarithmic ratio of two numbers, which means that there is no such thing as a dB measurement of a single signal. To measure a signal in decibels, you need to know what level it is referenced to. Commonly used reference levels are indicated by such symbols as dBm, dBV, and dBu.

default: A selection automatically used by a computer program in the absence of a choice made by the user.

Dialogic: Trade name of a telephony board manufacturer.

digital: Relating to an audio recording method in which sound waves are represented digitally (as on magnetic disk) so that in recordings wow and flutter are eliminated and background noise is reduced; also see analog. (Informal definition: Digital sound is what you get when you electronically chop up a continuous analog sound.)

digital-to-analog converter (DAC): A device that changes the sample words put out by a digital audio device into analog fluctuations in voltage that can be sent to a mixer or amplifier. All digital synthesizers, samplers, and effects devices have DAC at their outputs to create audio signals.

dry: Consisting entirely of the original, unprocessed sound. The output of an effects device is 100% dry when only the input signal is being heard, with none of the effects created by the processor itself; also see wet.

digital signal processing: Broadly speaking, all changes in sound that are produced within a digital audio device, other than changes caused by simple cutting and pasting of sections of a waveform, are created through DSP. A digital echo is a typical DSP function.

DSP: See digital signal Processing.

feedback: See resonance.

fast Fourier transform: A quick method of performing a Fourier analysis on a sound; see Fourier analysis.

FFT: See fast Fourier transform.

file format: A description of the disk file format or contents; also see pure, indexed, wave.

file type: A description of the disk file format or contents; also see pure, indexed, wave.

filter: A device for eliminating selected frequencies from the sound spectrum of a signal and perhaps (in the case of a resonant filter) increasing the level of other frequencies.

FM: See frequency modulation.

formant: A resonant peak in a frequency spectrum. For example, the variable formants produced by the human vocal tract are what give vowels their characteristic sounds.

Fourier analysis: A technique, usually performed with a digital signal processing (DSP) algorithm, that allows complex dynamically changing audio waveforms to be described mathematically as sums of sine waves at various frequencies and amplitudes; also see DSP.

frequency modulation (FM): A change in the frequency (pitch) of a signal. At low modulation rates, FM is perceived as vibrato or some type of trill. When the modulation wave is in the audio range (above 20 Hz, or so) FM is perceived as a change in tone color. FM synthesizers, commonly found on sound cards, create sounds using audio-range frequency modulation.

G: for giga, i.e., a billion, as in 1 GHz = 1,000,000,000 (10^9) hertz, or Gb, a billion bytes; [both ‘g’ in giga are pronounced as in ‘go’, *not* as in jelly].

direct-to-disk recording: A computer-based form of tapeless recording in which incoming audio is converted into digital data and stored on a hard disk.

harmonic: A frequency that is a whole-number multiple of the fundamental frequency. For example, if the fundamental frequency of a sound is 440 Hz, the first two harmonics are 880 Hz and 1,320 Hz (1.32 kHz). See overtone; also see inharmonic.

headroom: The amount of additional signal above the nominal input level that can be sent into a module before clipping distortion occurs.

hertz: A unit of frequency equal to one cycle per second. Named for Heinrich R. Hertz who did the original scientific investigations.

high-pass filter: A filter that attenuates the frequencies below its cutoff frequency.

Hz: abbreviation for hertz

indexed file: A file format that permits you to group related audio segments, then access those segments by specifying an index number.

inharmonic: Containing frequencies that are not whole-number multiples of the fundamental; also see harmonic.

interactive voice response (IVR): Computer systems that answer (or dial) a phone, play or record a message, often under control of the user’s touch-tone phone. A voice mail system is an example of a basic IVR system.

IRQ: Interrupt Request

ITU: International Telecommunication Union. A group that sets international communications standards. The ITU was previously known as the CCITT.

k: abbreviation for kilo (thousand): 1,000 or 10^3, as in 1 kHz.

low-frequency oscillator (LFO): An oscillator whose output is below the audible frequency range, typically used as a control source for modulating the sound to create vibrato, tremolo, trills, and so on.

m: abbreviation for milli (one thousandth): 1/1000 or 10^{-3}, as in 1 mW (0.001 W).

M: abbreviation for mega (million): 1,000,000 or 10^6, as in 1 MHz.

Nyquist frequency: The highest frequency that can be reproduced accurately when a signal is digitally encoded at a given sample rate. Theoretically, the Nyquist frequency is half of the sampling rate. For example, when a digital recording uses a sampling rate of 11 kHz, the Nyquist frequency is 5.5 kHz. If a signal being sampled contains frequency components that are above the Nyquist limit, aliasing will be introduced in the digital representation of the signal unless those frequencies are filtered out prior to digital encoding; see aliasing, low-pass filter.

OEM: Original equipment manufacturer. A company that produces an end product as opposed to a sub-assembly.

oscillator: An electronic sound source. In an analog synthesizer, oscillators typically produce regularly repeating fluctuations in voltage—that is, they oscillate. In a digital synthesizer, an oscillator more typically plays back a complex waveform by reading the numbers in a wavetable.

overtone: A whole-number multiple of the fundamental frequency.

frequency of a tone: The overtones define the harmonic spectrum of a sound. See Fourier analysis; also see partial.

partial: One of the sine-wave components (the fundamental, an overtone, or a tone at some other frequency) of a complex tone; see overtone.

PC: Personal computer.

pole: A portion of a filter circuit. The more poles a filter has, the more abrupt its cutoff slope will be. Each pole causes a slope of 6 dB per octave; typical filter configurations are two-pole (12 dB/oct) and four-pole (24 dB/oct). See rolloff.

pot: See potentiometer.

potentiometer: A device used to adjust the amplitude of the signal passing through it. The amplitude can usually be set to any value between full (no attenuation) and zero (infinite attenuation). Pots can be either rotary or linear (sliders), and can also be either hardware or “virtual sliders” on a computer display.

prompts: Common term for outgoing messages recorded for your voice messaging system. These messages typically “prompt” the user to press a key for a specific person or service.

pure file: Files containing only the digitized sound.

Q: See resonance.

quantization noise: One of the types of errors introduced into an analog audio signal by encoding it in digital form. The digital equivalent of tape hiss, quantization noise is caused by the small differences between the actual amplitudes of the points being sampled and the bit resolution of the analog-to-digital converter.

quantized: Set up to produce an output in discrete steps.

reconstruction filter: A low-pass filter on the output of a digital-to-analog converter that smoothes the staircase-like changes in voltage produced by the converter in order to eliminate clock noise from the output.

regeneration: See resonance; feedback.

resonance: A function of a filter in which a narrow band of frequencies (the resonant peak) becomes relatively more prominent. If the resonant peak is high enough, the filter will begin to oscillate, producing an audio output even in the absence of input. Filter resonance is also known as emphasis, and as Q. It is also referred to in some older instruments as *regeneration* or *feedback*, because feedback was used in the circuit to produce a resonant peak.

rolloff slope: The acuity of a filter cutoff frequency. Rolloff is generally measured in decibels (dB) per octave. A shallow slope, such as 6 dB per octave, allows some frequency components beyond the cutoff frequency to be heard, but at a reduced volume. When the rolloff slope is steep (on the order of 24 dB/oct), frequency components very close to the cutoff frequency are reduced in volume so much that they fall below the threshold of audibility. See filter, and pole.

SDK: Software development kit.

sidebands: Frequency components outside the natural harmonic series, generally introduced to the tone by using an audio-range wave for modulation. See clangorous.

sine wave: A signal put out by an oscillator in which the voltage, or equivalent, rises and falls smoothly and symmetrically, following the trigonometric formula for the sine function. Sub-audio sine waves are used to modulate other waveforms to produce vibrato and tremolo. Audio-range sine waves contain only the fundamental frequency, with no overtones, and thus can form the building blocks for more complex sounds.

sound card: Add-on hardware typically placed inside the computer and used to record and play digital audio. These devices usually interface to standard audio equipment such as microphones and speakers.

telephony card: Add-on hardware typically placed inside the computer and used record and play digital audio. These devices usually interface to standard telephone lines and often can handle multiple phone lines per board; also see voice board.

total harmonic distortion (THD): An audio measurement specification used to determine how accurately a device can reproduce an input signal at its output. THD describes the cumulative level of the harmonic overtones the device being tested adds to an input sine wave. THD+n is a specification that includes both harmonic distortion of the sine wave and nonharmonic noise.

TI\F DLL: VISI's Telephone Interface Dynamic Link Library.TI/F DLL is a set of programs that manage the telephony audio Terminate-and-Stay-Resident (TSR) driver and enables it to communicate with other Windows programs.

transient: Any of the non-sustaining, non-periodic frequency components of a sound, usually of brief duration and higher amplitude than the sustaining components, and occurring near the onset of the sound (attack transients).

tremolo: A periodic change in amplitude, usually controlled by an LFO, with a periodicity of less than 20 Hz; also see vibrato.

vap: A common file extension specifying an indexed file format. Typically encountered as FILENAME.VAP

VBase: A type of indexed files containing raw audio data plus additional format, control, and annotation information. These files permit you to group related audio segments, then access those segments by specifying an index number.

vibrato: A periodic change in frequency, often controlled by an LFO, with a periodicity of less than 20 Hz; also see tremolo.

VISI: Voice Information Systems, Inc. The really great company that makes this product.

voice board: Add-on hardware typically placed inside the computer and used record and play digital audio. These device usually interface to standard telephone lines and often can handle multiple phone lines per board; also see telephony card.

vox: A file extension specifying a generic telephony audio format. Typically encountered as FILENAME.VOX

wav: A file extension specifying an industry standard multimedia audio format. Typically encountered as FILENAME.WAV

wave: An industry standard file format that contains raw audio data plus additional format, control, and annotation information.

waveform: A signal, either sampled (digitally recorded) or periodic, being generated by an oscillator. Also, the graphic representation of this signal, as on a computer screen. Each waveform has its own unique harmonic content. See oscillator.

wet: Consisting entirely of processed sound. The output of an effects device is 100% wet when only the output of the processor itself is being heard, with none of the dry (unprocessed) signal; also see dry.

word: A number of bytes processed as a unit and conveying a quantum of information in communication and computer work. Also, a single number (sampled word) that represents the instantaneous amplitude of a sampled sound at a particular moment. In 8-bit recording, a sample word contains one byte; in 16-bit recording, each word is a two-byte number.

♦ Index