VICTOR

Audio Input Option

104691-01

Audio Tool Kit II

First VICTOR printing August, 1983.
Second VICTOR printing December, 1983.

ISBN 0-88182-104-7 Printed in U.S.A.

IMPORTANT SOFTWARE DISKETTE INFORMATION

For your own protection, do not use this product until you have made a backup copy of your software diskette(s). The backup procedure is described in the user's guide for your computer.

Please read the DISKID file on your new software diskette. DISKID contains important information including:

- The part number of the diskette assembly.
- The software library disk number (for internal use only).
- The date of the DISKID file.
- A list of files on the diskette, with version number, date, and description for each one.
- Configuration information (when applicable).
- Notes giving special instructions for using the product.
- Information not contained in the current manual, including updates, any known bugs, additions, and deletions.

To read the DISKID file onscreen, follow these steps:

1. Load the operating system.
2. Remove your system diskette and insert your new software diskette.
3. Enter —

   TYPE DISKID

   and press Return.
4. The contents of the DISKID file is displayed on the screen. If the file is large (more than 24 lines), the screen display will scroll. Type ALT-S to freeze the screen display; type ALT-S again to continue scrolling.

CONTENTS

APPENDIXES

CHAPTERS

1

INTRODUCTION

OVERVIEW 1.1

The Audio Tool Kit allows you to store your own voice on disk and retrieve it from application programs. Your computer can read these stored words and sentences. The phrases you enter are "spoken" via internal speakers and coder/decoder (CODEC) hardware.

With the Audio Tool Kit, you can record human voices, sounds, and music through the microphone, transfer them to disk file libraries, and edit the libraries. The sounds saved on disk are accessed with programming languages (such as Pascal and BASIC) that run under MS-DOS.

The Audio Tool Kit consists of:

- The *Audio Input Option*—the hardware interface between your computer and your voice; it consists of an audio preamplifier board and a microphone.
- The *Voice Editor*—a sophisticated and easy-to-use software package that lets you record, store, edit, and reproduce sounds from the Audio Input Option; you can also edit the sound files already saved on your disk.
- The *Voice Kernel*—the subroutine library and loader programs used to interface the Audio Tool Kit to most of the programming languages supported by MS-DOS.

The hardware portion is described in the *Audio Input Option* booklet. Once the board and microphone are installed in your computer, you use the microphone just like the microphone on a tape recorder.

1.2 AUDIO TOOL KIT FEATURES

Many application programs require you to watch many different locations on your screen to keep track of what you are doing. Very few computers provide more than a "beep" for audio interaction. With the Audio Tool Kit, however, you can make any computer speak in clear, human voices that you record and edit.

For example, a data base management system can actually tell users to save files, instead of requiring them to look for a message on their screens. It can also respond during data entry ("Please enter the part name"), and give audio results of calculations ("The total is ten percent higher"). You can also give users reminders ("Please read your mail") and error messages ("The disk is full").

All voice information is kept in disk files. You can edit, modify, and re-record the data at any time with the Voice Editor. The data files are accessed from your programs with the Voice Kernel program libraries.

The Voice Editor provides an easy way to record words, phrases, sentences, and sounds from the microphone of the Audio Input Option. The editor uses most of the function keys, and displays representations of sounds on your screen as you edit them. Each voice file can contain many different sounds that are individually accessible.

The Voice Kernel is a set of libraries that contains the voice input and output functions of the Audio Tool Kit. It also provides functions for playing musical tones on the speaker in the computer. The Voice Kernel libraries are loaded from easy-to-use programs, and support these languages:

- MS-BASIC (both Interpreter and Compiler)
- GW-BASIC (with graphics)
- MS-Pascal
- PL/M
- Assembly language

The Voice Kernel allows you to use more than one voice file in an application program. Although you are limited to the amount of memory a voice file can take, you can switch between files based on the needs of your program. You can also use the Voice Kernel to record and play sounds. For example, you can create a voice mail system in which short spoken messages are passed between users who have the audio input option.

INSTALLING THE AUDIO TOOL KIT 1.3

Follow the directions in the *Audio Input Option* booklet to install the hardware portion of the Audio Tool Kit. When you are using the microphone, be sure that the on-off switch is set "on," or no sounds will be recorded. You should also ensure that the microphone's battery is working.

The Audio Tool Kit diskette contains these files:

- VOICE.EXE —Voice Editor
- VOICE.DAT —Help screens for the Voice Editor
- CODECBAS.COM —Voice Kernel subroutine loader for interpreted MS-BASIC
- CODECGW —Voice Kernel subroutine library for GW-BASIC
- CODEC*.LIB —Libraries for the compiled languages for the Voice Kernel
- BUFF.OBJ —Object file for the compiled languages for the Voice Kernel
- BUFF.ASM —Source code for BUFF.OBJ
- CALC.VOC —Sample voice file, used with CALC.BAS
- CALC.KEY —Key file for CALC.VOC
- CALC.BAS —Example application program using the Voice Kernel
- SAMPLEGW.BAS —Sample GW-BASIC Audio initialization system

The Voice Editor requires 256K of RAM memory and the Audio Input Option hardware to run. The other software will operate on unmodified 128K systems. There is no special installation required for any of the software.

2

USING THE VOICE EDITOR

The Voice Editor is the major software portion of the Audio Tool Kit. It allows you to store and retrieve words in your own voice on diskette, arrange libraries of words and phrases, and edit the words and phrases in a library.

The program is menu-driven; it comes with extensive help messages, available by pressing function key 7. Each voice library you set up is maintained by the Voice Editor.

STARTING THE PROGRAM 2.1

The Voice Editor is run by giving the command:

VOICE

in the MS-DOS operating system. Your screen clears, and the program prompts:

```
From which disk do you want your voice files to be read ?
          ---› Type a letter from A to O ‹---
```

Enter the letter of the drive that contains the voice files you want to edit, or the drive on which you want to create the files. The screen clears, and a list of all of the voice files on that disk (and the words <NEW FILE>) is displayed. Use the cursor control keys to move the highlighted block to the name of the file you want to edit, and press function key 2 (OPEN).

If you are entering a new file, you are prompted at the top of the screen for the file name. Enter a legal file name (up to 8 characters with no spaces), and press function key 2 (OK).

You are now presented with the main menu (see Section 2.3). The name of the file you are editing is shown in the upper left corner of the screen.

2.2 HINTS FOR USING THE VOICE EDITOR

The Voice Editor maintains two types of files for each voice library:

- Voice files contain the data for reproducing the recorded sounds; these files have a file type of .VOC.
- Key files contain information about the sounds in their associated voice file; these files have a file type of .KEY.

You can keep as many as 128 different sounds in each voice file, and you can give each sound a name (up to 32 characters) with the Voice Editor. Thus, all of the sounds for a given application can reside in both one file and many different files.

When you are editing a sound, the Voice Editor displays the sound graphically on your screen. The sound is shown in eleven horizontal bars as patches of light and dark areas. The sounds are represented by the mixed light and dark patterns, and the areas in these bars with no mixed patches are silent. Although only eleven bars are displayed on the screen, a total of 20 bars are available by scrolling the screen up and down with the cursor control keys. (Each bar represents 2.5K of RAM or disk memory.)

There are two markers on the screen display—"S" and "E"—which represent the start and end of the current sound. You can move these markers to shorten or lengthen a sound saved on the disk. The start and end markers are set in the SOUND, KEYS, and EDIT menus by moving the cursor to the position you want with the cursor control keys, and then pressing the START or END function keys.

When the Voice Editor prompts you for a file name or a new voice name, type the name (without spaces); and then finish the name by pressing function key 2 (OK). When the Voice Editor prompts you with a list of the current file names or current key names, move the cursor to the file name you want with the cursor arrow keys, and then press the desired function key.

Never remove the disk that contains the current sound files until you have exited from the Voice Editor. Also, before leaving the program, save the current sound to disk if you want to keep it. (If you have selected CH DISK in the open menu, you can change disks. This procedure is described later.)

By pressing function key 7 (HELP) you can always receive help on your current status. By pressing function key 1 (MENU), you can go to the previous menu at any time. See Appendix A for a list of all the menus. A walk-through example of using the Voice Editor is given in Chapter 4.

2.3 THE MAIN MENU

The main menu is the first menu displayed after you specify the file you want to edit with the Voice Editor. The screen has eleven horizontal bars across it, alternated with blank lines. The bars hold the visual representation of the sounds you are editing, and the blank lines hold the starting and ending markers of the sound.

The choices offered by the main menu are selected by using the function keys. They are described in the next sections:

1 **EXIT** —Exit back to the operating system

2 **SOUND** —Play a sound on the speaker, or record a sound from the microphone

3 **CONTROL** —Adjust the speed of recording and playback

4 **DISK** —Save the current sound to the library file, retrieve a previously saved sound from the current library file, open a new library file, or delete a sound or library file

5 **EDIT** —Edit the current sound (add, erase, and delete sounds)

7 **HELP** —Bring up the main menu's help screen

2.4 EXITING FROM THE VOICE EDITOR

Function key 1 (EXIT) on the main menu returns you to the operating system, after prompting:

```
Are you sure you want to exit to the Operating System?
```

Press function key 2 (YES) if you do, or function key 1 (NO) to get back to the main menu. When you exit, the currently open sound file and key file are closed and written to the disk. Do not remove the disk with the sounds you are recording and editing before you are at the operating system level.

PLAYING OR RECORDING THE CURRENT SOUND 2.5

Use function key 2 (SOUND) on the main menu to play the sound that you are currently editing, or to record a new sound. The sound menu is displayed:

1 **MENU** —Return to the main menu

2 **START** —Set the starting point for the sound

3 **END** —Set the ending point for the sound

4 **PLAY** —Play the current sound between the starting and ending points

5 **RECORD** —Record a new sound from the microphone between the starting and ending points

7 **HELP** —Get help for this menu

The PLAY and RECORD options play or record only between the starting and ending markers (marked "S" and "E" on the screen). To move either marker, move the block cursor on the screen with the cursor control keys. Press function key 2 (START) or function key 3 (END) to move the markers.

Press function key 4 (PLAY) to play the current sound on the speaker of your computer. The message:

```
Space bar to begin, "A" to abort
```

appears at the upper left corner of the screen. The sound begins to play when you press the space bar, and continues to play until you press a key, or when the end pointer is reached. Type "A" instead of a space if you do not want to hear the sound.

Press function key 5 (RECORD) to record a new sound between the start and end pointers. The message:


```
Space bar to begin, "A" to abort
```

appears at the upper left corner of the screen, just as when you are listening to a sound. The microphone begins to record when you press the space bar, and continues to record until you press a key, or when the end pointer is reached. Type "A" instead of a space if you do not want to record a new sound.

If no sound is appearing on your screen when you record, the microphone may not be plugged in, or the switch on the microphone may be set to "off". If the sound quality is poor, or the microphone does not record at all, check the microphone's internal battery.

2.6 SETTING THE SPEED

Press function key 3 (CONTROL) from the main menu to change the speed at which the CODEC records or plays back sounds. Each sound saved on the disk has an associated speed. The default speed is 32, and the range is 1 (fast) to 240 (slow). The menu for this option is:

1 **MENU** —Return to the main menu

2 **FASTER**

3 **SLOWER**

The CONTROL menu lets you change these parameters with function keys 2 (FASTER) and 3 (SLOWER). To test the results of your changes, go to the main menu (with function key 1) and play the sound from the SOUND menu (function key 2).

Recording sounds at faster rates causes clearer sound reproduction, but takes up more space in memory and on disk.

SAVING AND LOADING SOUNDS 2.7

The DISK menu (function key 5 on the main menu) is used to open and close voice files, or to load and save sounds in the current file. The menu for this option is:

1 **MENU** —Return to the main menu

2 **KEYS** —Sub-menu that loads and saves sounds

3 **FILES** —Sub-menu that opens and closes sound libraries

2.7.1 SOUNDS WITHIN A FILE

The options under the KEYS sub-menu are:

1 **MENU** —Return to the DISK menu

2 **START** —Set the starting point for the sound

3 **END** —Set the ending point for the sound

4 **LOAD** —Load a new sound from the current library, overwriting the current sound

5 **SAVE** —Save the sound between the start and end pointers into the library; you are prompted for a key name for the sound

6 **DELETE** —Delete a sound from the current library

7 **HELP** —Get help for this menu

Choices MENU, START, END, and HELP (function keys 1, 2, 3, and 7) are the same as in the SOUND menu described previously.

The LOAD option (function key 4) clears the screen and lists the current sounds in the library. Use the cursor control keys to choose the sound you want to load, then press function key 3 (LOAD). The sound is loaded from the disk to the area between the pointers, and the graphic representation of the sound is displayed.

Once you have edited a sound (with the CONTROL and EDIT functions from the main menu), save it to the library with the SAVE option (function key 5). You are prompted in the upper left corner of the screen for the name of the sound, which you type in and finish by pressing function key 2 (OK). The Voice Editor prevents you from giving two different sounds the same name; you can use up to 32 characters for the name.

You can delete sounds from the library with function key 6 (DELETE), which presents you with the list of sounds in the library. Choose the sound to be deleted with the cursor control keys, then press function 3 (DELETE). Deleting a sound takes longer than loading or saving one since the voice file is compacted to retrieve disk space.

The options under the FILES sub-menu are:

1 **MENU** —Return to the DISK menu

2 **OPEN** —Open a sound file, or create a new sound file

3 **HEADER** —Change the header information in the current sound file

4 **DELETE** —Delete a sound file and its key file from the disk

7 **HELP** —Get help on this sub-menu

The OPEN procedure (function key 2) is the same as the one you perform when you first start the Voice Editor. You can select one of these function keys:

1 **MENU** —Return to the FILES sub-menu

2 **OPEN** —Open another file

3 **CH DISK** —Allows you to change the disk drive to use

4 **CH DIR** —Allows you to change the current subdirectory on the current drive

7 **HELP** —Gets help for this sub-menu

Each sound file keeps information about the origin of each sound in the key file. To organize your different sound files, you should keep "origin" information current with the HEADER option (function key 3). The header information consists of the following fields:

- Voice Editor format (filled in by the system).
- Version of the file (1 digit, filled in by you).
- Originator (your name, up to 16 characters).
- Comment (brief description of the file, up to 32 characters).
- Date (up to 8 numbers).
- File size, in bytes (filled in by the system).

The Voice Editor lets you change the contents of the originator, comment, and date fields. First move with the cursor control keys to each field you want to fill in; then, enter your information. Finish the entry by pressing function key 2 (OK). If you are changing an entry, and you decide to keep the old entry, press function key 6 (UNDO) to restore the previous entry.

The DELETE option (function key 4) deletes files from your disk. The prompt for deleting is similar to opening a file. After selecting this option, a list of files is presented; position the cursor onto the desired file name, then press function key 4 (DELETE) to delete the sound and key files from the disk.

2.8 EDITING SOUNDS

The EDIT menu (function key 5 from the main menu) lets you edit the sounds in your sound libraries. For example, if a library contains one sound with the words "Please press the", and another with the words "space bar", you can make one long sound with the two phrases combined.

The choices on the EDIT menu are:

1 **MENU** —Return to the main menu

2 **START** —Set the starting point for the sound

3 **END** —Set the ending point for the sound

4 **COPY** —Copy the sound between the start pointer and end pointer to after the cursor

5 **ERASE** —Erase the sound between the start and end pointers; this fills the space with silence

6 **DELETE** —Deletes the sound between the start and end pointers, and moves the pointers together

7 **HELP** —Get help on the edit menu

Choices MENU, START, END, HELP (function keys 1, 2, 3, and 7) are the same as described previously.

Function key 4 (COPY) is used to copy a part of the sound to another area. For example, to turn the phrase "one hundred and" into "one hundred and one", move the starting pointer to the beginning of the buffer, the ending pointer to the area between "one" and "hundred" (the silent zone where there are no mixed light and dark spots), and the cursor to the end of the word "and". Press function key 4, and "one" is copied to the area directly after the cursor.

Use the ERASE option (function key 5) to fill the area between the start and end pointers with silence. You use the DELETE option (function key 6) to remove sound from the buffer. For example, if the phrase in the buffer is "the TAB key", and the start and end pointers are around the word "TAB":

- The ERASE function produces "the key" with a pause between "the" and "key".
- The DELETE function produces "the key" without a pause between the words.

3

USING THE VOICE KERNEL

The Voice Kernel is the software interface between application programs and the CODEC hardware. Voice libraries created and edited with the Voice Editor can be played from your application program. You can also create your own files from your application program with the Audio Input Option, as well as generate musical tones and other sounds from your program.

The two steps required to use the Voice Kernel are:

1. Prepare the kernel routines in memory by running the CODECBAS program (MS-BASIC), using the BLOAD command (GW-BASIC), or calling the initialization routine, AUDIO_INIT (Pascal, PL/M, and assembly language).
2. Call the CODEC routines with the CALL statement in your programming language.

An example of using the Voice Kernel in an application program is given in Chapter 4.

3.1 LOADING THE VOICE KERNEL

The Voice Kernel subroutines must be loaded into memory before they are called. Different languages require the subroutines to be loaded in different fashions.

MS-BASIC —Run the CODECBAS program before running MS-BASIC. Your program must determine the offset address for the subroutines; this is described in Section 3.4.

3

GW-BASIC —Your program must load the subroutines with a BLOAD statement; this is described in Section 3.5.

Compiled languages —In your program, an initialization subroutine must be called to initialize the Voice Kernel (except in BASIC Compiler). An object file and a library must be linked into your program with the LINK command.

Before running your program, you must determine the amount of memory you want to allocate for the sounds, and the maximum number of voice keys you will use in the application.

Since the Voice Kernel has internal memory management, the total size of the sounds you use could be larger than the amount you allocate. The range for the amount of memory is 1 to 50 kilobytes; 50 is the default. If you attempt to access a sound that is larger than this parameter, the Voice Kernel gives you an error message.

Each key takes up 56 bytes of memory, and you can specify between 0 and 128 keys; 128 is the default. If you attempt to open a voice file that has more keys than is specified in this parameter, the Voice Kernel gives an error message.

Determine the amount of memory reserved for voice and key files based on your application and the amount of available RAM memory. At a minimum, the amount of memory must be sufficient to contain the largest sound to be produced. You should figure about 2.5K of memory for each sound bar that the voice editor displays. If a play phrase or a play/record sound is performed and the buffer is not large enough, the Voice Kernel issues you an error message. You may want extra memory so that less disk access is required when reproducing multiple sounds. If you use both the Voice Kernel and the GRAFIX package in the same application, you must load the GRAFIX Kernel first.

CALLING THE VOICE KERNEL 3.2

Call the Voice Kernel subroutines with the CALL statement of your programming language. In the formats below, "Pn" indicates a parameter, "Fn" indicates a function number, and "Fname" indicates the function name.

MS-BASIC	—CALL AUDIO(P1, P2, ... , Pn, Fn) All parameters must be variables, not constants.
GW-BASIC	—CALL AUDIO(P1, P2, ... , Pn, Fn) All parameters must be variables, not constants.
BASIC Compiler	—CALLS AUDIO(P1, P2, ... , Pn, Fn)
Pascal	—Fname(P1, P2, ... , Pn) Any parameters that will be passed back to the function must be declared type VAR.
PL/M	—CALL Fname(P1, P2, ... , Pn)

3.3 VOICE KERNEL FUNCTIONS

The functions in the Voice Kernel are:

FUNCTION	NAME
0	GET_KEY_NAME
1	PLAY_NOTE
2	RECORD_SND
3	PLAY_SND
4	LOAD_LIBRARY
5	PLAY_PHRASE
6	FREE_LIBRARY
7	PAUSE
8	CHANGE_CLOCK

Use the function numbers for MS-BASIC, BASIC Compiler, or GW-BASIC, and the function names for all other languages.

Functions 2 and 3, RECORD_SND and PLAY_SND, do not use the voice files you created with the Voice Editor. Instead, RECORD_SND creates files in its own format, which is only playable with the PLAY_SND function. These functions allow you to record users' voices and play them again, although you cannot edit or modify the files created. If you are going to use only .SND files in your applications, you can specify 0 keys when you load the Voice Kernel.

Only one library can be open at a time. No libraries can be open while doing a record or playback. Free the current library with function 6, FREE_LIBRARY, before you call functions 2, 3, or 4 (RECORD_SND, PLAY_SND, or LOAD_LIBRARY). An error message is printed on your screen if you do not free memory before calling these functions.

Function 0—GET_KEY_NAME

Returns the name of a sound, given its key number. The key names in a voice library are in alphabetical order. The key number associated with a key name is the position number of the key name in the alphabetized list. This function

can be used with function 5, PLAY_PHRASE. The name is returned as an array corresponding to the ASCII codes of the letters in its name. This function is useful if you do not know the name of the sounds in a file.

Parameters passed:

- Key number
- Name of integer array (32 items) to receive name

Parameters returned:

- The key name associated with the key number

Related functions:

- 4 (LOAD_LIBRARY)
- 5 (PLAY_PHRASE)
- 6 (FREE_LIBRARY)

Function 1—PLAY_NOTE

Plays a note on your computer's speaker.

Parameters passed:

- Frequency, in Hertz (20-20000)
- Duration, in milliseconds (0-65535)

Parameters returned:

- None

Related functions:

- 7 (PAUSE)

Function 2—RECORD_SND

Records sounds from the microphone to the disk. Recording starts when you press the space bar, and ends when you press any other key (or when the sound fills up the memory assigned when the Voice Kernel was loaded). The

data is stored in the named file (with extension .SND), including a 16-byte header with the clock rate (which is set by function 8, CHANGE_CLOCK). This file is different from the sound files created with the Voice Editor, and can be played only with function 3, PLAY_SND.

Parameters passed:

- Filename with no extension (drive letter optional)
- Name of an integer variable for error return

Parameters returned:

- Error status (0 = normal, 1 = buffer full)

Related functions:

- 3 (PLAY_SND)
- 8 (CHANGE_CLOCK)

Function 3—PLAY_SND

Loads a sound file from disk and plays it on your computer's speaker. The file type of this file is .SND, and the file is usually created with function 2, RECORD_SND. If the file is larger than the amount of memory specified when the Voice Kernel was loaded, an error message is printed on the screen.

Parameters passed:

- Filename with no extension (drive letter optional)

Parameters returned:

- None

Related functions:

- 2 (RECORD_SND)
- 8 (CHANGE_CLOCK)

Function 4—LOAD_LIBRARY

Loads a voice file and the corresponding key file into memory. The function must be used before calling functions 0, 5, or 6 (GET_KEY_NAME, PLAY_PHRASE, or FREE_LIBRARY). These library files are usually created with the Voice Editor. If the file being loaded has more keys than specified when the Voice Kernel was loaded (or if a library is already loaded), an error message is printed on the screen.

Parameters passed:

- Filename without extension (drive letter optional)

Parameters returned:

- None

Related functions:

- 0 (GET_KEY_NAME)
- 5 (PLAY_PHRASE)
- 6 (FREE_LIBRARY)

Function 5—PLAY_PHRASE

Plays the specified sounds on the speaker of your computer. The names are either specified directly, or returned from function 0, GET_KEY_NAME. If any sound is larger than the amount of memory specified when the Voice Kernel is loaded, an error message is printed on the screen.

To play many sounds in succession, pass their names to PLAY_PHRASE delimited with a single space. If more than one space is used between names, a pause of .25 seconds is added for each extra space. For example, passing to PLAY_PHRASE the string

```
SAVING DISK    TIME
```

will cause the phrase SAVING to be played, immediately followed by the phrase DISK, then a pause of .5 seconds, then the phrase TIME.

Parameters passed:

- Character string name

Parameters returned:

- None

Related functions:

- 0 (GET_KEY_NAME)
- 4 (LOAD_LIBRARY)

Function 6—FREE_LIBRARY

Frees the current voice library. Use this function before you access a different voice library, or before you use either function 2 or 3 (PLAY_SND or RECORD_SND) after opening a voice library.

Parameters passed:

- None

Parameters returned:

- None

Related functions:

- 4 (LOAD_LIBRARY)

Function 7—PAUSE

Turns off the CODEC, which silences the speaker, and waits for the specified amount of time.

Parameters pssed:

- Duration, in milliseconds (0-32767)

Parameters returned:

- None

Related function:

- 1 (PLAY_NOTE)

Function 8—CHANGE_CLOCK

Changes the sampling rate used to record and play back sound files (functions 2 and 3, RECORD_SND and PLAY_SND). A higher sampling rate gives better sound quality and reproduction, but takes more RAM and disk space. The default value for the rate is 32.

Parameters passed:

- Clock rate (1-240, fast to slow)

Parameters returned:

- None

Related functions:

- 2 (RECORD_SND)
- 3 (PLAY_SND)

MS-BASIC INTERFACE 3.4

Before you run MS-BASIC, you must load the subroutines by running the CODECBAS program. The format of the CODECBAS command is:

```
CODECBAS /Sn /Kn
```

The /S and /K parameters are optional, and the "n" indicates a number. (If you do not specify the /S or /K parameters, then the Audio Kernel is loaded with 50K of sound buffer and space for 128 keys.) The options specify:

- /S—The amount of RAM memory (in kilobytes) to be reserved for the sounds used. This number is the maximum size that an individual sound can take up in memory.
- /K—The maximum number of keys to be used in this application.

For example, to initialize the Voice Kernel for a BASIC program that is going to use 40K of memory, and a maximum of 24 keys, give the command:

```
CODECBAS /S40 /K24
```

To use the MS-BASIC CALL statement to access the CODEC, you must define the segment and offset of the Voice Kernel subroutines. The address of the interface is found in entry 220 of the interrupt vector table, which is bytes 880 through 883 of segment 0.

For example, this program loads both AUDIOSEG and AUDIO with the segment and offset address of the subroutines, and issues the correct DEF SEG command:

```
10 '
20 '  get AUDIO entry point from interrupt vector entry 220
30 '
40 DEF SEG = 0
50 LOWOFF = PEEK(880)                  ' Low byte offset address
60 HIOFF  = PEEK(881)                  ' High byte offset address
70 LOWSEG = PEEK(882)                  ' Low byte segment address
80 HISEG  = PEEK(883)                  ' High byte segment address
90 AUDIOSEG = (256*HISEG) + LOWSEG     ' Full segment address
100 AUDIO   = (256*HIOFF) + LOWOFF     ' Full offset address
110 DEF SEG = AUDIOSEG                 ' Define segment for CALL
```

These BASIC statements play a note at 440 Hertz for one second:

```
130 PLAY.NOTE% = 1                     ' Function number
140 FREQ% = 440                        ' Frequency
150 DURATION% = 1000                   ' Duration
160 CALL AUDIO(FREQ%, DURATION%, PLAY.NOTE%)
```

This example illustrates two more important points. First, all numeric parameters must be defined as integers. The variable can be declared as integer by using a percent sign (%) as the last character of the name (as shown here), or by using the DEF INT statement. Second, all of the parameters must be variables, since BASIC does not pass constants correctly. Thus, the following statement does not work:

```
150 CALL AUDIO(440,1000,1)
```

GW-BASIC INTERFACE 3.5

Loading the subroutines in GW-BASIC is easier than in MS-BASIC because you do not need to run a separate program before you run your application. You load the procedures into a free segment of memory with the BLOAD command.

You must first find a free segment for the subroutines; this is done with a small assembly language program embedded in your BASIC code. You then switch to that segment with a DEF SEG command; use the BLOAD command to load the file CODECGW. This program—which is supplied on the Audio Tool Kit in the file SAMPLEGW.BAS—finds the segment and loads the subroutines:


```
10 '
20 ' find a free segment for the CODEC subroutines
30 ' and load them
40 DIM GETDS%(8)
50 FOR I = 0 TO 8
60     READ GETDS%(I)
70     NEXT I
80 DATA &H8B55, &H8CEC, &H05D8, &H1000, &H5E8B, &H8906
90 DATA &H5D07, &H02CA, &H0000
100 '
110 DEF SEG
120 DATA.SEG% = 0
130 GET.CODEC.BASE% = VARPTR(GETDS%(0))
140 CALL GET.CODEC.BASE%(DATA.SEG%)
150 '
160 DEF SEG = DATA.SEG%
170 BLOAD "CODECGW",0
180 AUDIO% = 0
```

These GW-BASIC statements play a note at 440 Hertz for one second:

```
190 PLAY.NOTE% = 1                    ' Function number
200 FREQ% = 440                       ' Frequency
210 DURATION% = 1000                  ' Duration
220 CALL AUDIO%(FREQ%, DURATION%, PLAY.NOTE%)
```

3.6 INTERFACES TO THE COMPILED LANGUAGES

Calling the Voice Kernel subroutines in the compiled languages is very similar to calling them in MS-BASIC. The main difference is that you use the name of the subroutine, not the number. The other difference is that you need to call an initializing subroutine, AUDIO_INIT, before you call any of the Voice Kernel subroutines. This subroutine sets the amount of RAM memory and maximum number of keys (described in Section 3.1). Do not call AUDIO_INIT from BASIC Compiler; it is done automatically.

To link the Voice Kernel into the compiled languages, link the module BUFF.OBJ after your program modules; then, link the correct library (CODECPAS for Pascal, CODECPLM for PL/M, CODECASM for assembly language, and CODECBAS for BASIC Compiler). For example, to link a Pascal program with object modules PROG1 and PROG2, and the main Pascal library:

- When the linker prompts you for the object modules, type:

  PROG1 + PROG2 + BUFF

- When the linker prompts you for the libraries, type:

  CODECPAS + PASCAL

Be sure that BUFF is listed last in your linker commands. BUFF comes with the default amount of memory (50K) and maximum number of keys (128). To specify a different amount of memory or number of keys, modify BUFF.ASM and re-assemble the file. The equates that contain these variables are NUM_KEYNAMES and BUF_SIZE, located near the end of the file. The number specified in BUF_SIZE is in paragraphs (1 paragraph is 16 bytes), not kilobytes, so multiply the amount of Kbytes you require by 64; that is, XK * 1024/K * 1 Paragraph/16 bytes.

Each Pascal procedure must be defined as external. The variable "key" is defined as LSTRING(32), "file_name" is defined as LSTRING(10), and "talk_string" as LSTRING(255). Be sure to call AUDIO_INIT before calling any other Audio Kernel routine.

```
Procedure Get_key_name(key_num:  INTEGER; VAR key: LSTRING);
    External;

Procedure Play_note(freq, duration: WORD); External;

Procedure Record_snd(CONST file_name: LSTRING; VAR error_val:
    INTEGER); External;

Procedure Play_snd(CONST file_name: LSTRING); External;

Procedure Load_library(CONST file_name: LSTRING); External;

Procedure Play_phrase(CONST talk_string: LSTRING); External;

Procedure Free_library; External;

Procedure Pause(duration: INTEGER); External;

Procedure Change_clock(clock_rate: INTEGER); External;

Procedure Audio_init; External;
```

3.6.2 PL/M INTERFACE

The procedures in PL/M are defined as:

```
Get_key_name : procedure (key_num, key_ptr) external;
    declare   key_num          Integer;
    declare   key_ptr          Pointer;
end Get_key_name;

Play_note : procedure (freq, duration) external;
     declare   freq             Word;
     declare   duration         Word;
end Play_note;

Record_snd : procedure (file_name_ptr, error) external;
     declare   file_name_ptr    Pointer;
     declare   error            Integer;
end Record_snd;

Play_snd : procedure (file_name_ptr) external;
     declare   file_name_ptr    Pointer;
end Play_snd;

Load_library : procedure (file_name_ptr) external;
     declare   file_name_ptr    Pointer;
end Load_library;

Play_phrase : procedure (talk_string_ptr) external;
     declare   talk_string_ptr Pointer;
end Play_phrase;

Free_library : procedure external;
end Free_library;

Pause : procedure (duration) external;
     declare   duration         Word;
end Pause;

Change_clock : procedure (clock_rate) external;
     declare   clock_rate       Word;
end Change_clock;

Audio_int : procedure external;
end Audio_init;
```

Here are descriptions of the structures passed to these PL/M subroutines:

```
declare
     key_struc structure
          ( length              byte,
            key_name (32)       byte ),
      filename_struc structure
          ( length              byte,
            file_name (10)      byte ),
     talk_string_struc structure
          ( length              byte,
            talk_string (255)   byte );
```

The user application program in PL/M must be MEDIUM Model. Be sure to call AUDIO_INIT before calling any other Audio Kernel routine.

ASSEMBLY LANGUAGE INTERFACE 3.6.3

The assembly language interface requires that each parameter of an Audio Kernel subroutine be pushed onto the stack before calling the subroutine. The parameters should be pushed onto the stack in the same order as they are defined in this document. Integer and word parameters should be passed by pushing the desired value on the stack as a word. String parameters should be passed by address; push the segment address of the string, then the offset address of the string, and finally the length of the string. The following sample code illustrates a call to the LOAD_LIBRARY routine, as well as a call to AUDIO_INIT, which must be called before any other Audio Kernel routine.

```
CALL    AUDIO_INIT      ; Initialize Audio Kernel
LEA     AX, LIBRARY
PUSH    DS              ; Pass segment of library name
PUSH    AX              ; Pass offset of library name
MOV     AX, LEN
PUSH    AX              ; Pass length of library name
CALL    LOAD.LIBRARY    ;Load the library
```

The stack will be restored to its correct value by the Voice Kernel subroutines. In GET_KEY_NAME, the ES:AX register pair will point to the key name if it was found; AX will equal −1 if it was not found. In RECORD_SOUND, AX will be returned as 0 if a key was pressed while recording, and as 1 if the sound buffer became full while recording.

The Audio Kernel subroutines that you use must be declared as external routines in your program, by using the assembler EXTRN directive. They must also be declared FAR in the external declaration. The external declarations must come before the segment definitions in your program in order for the Audio Kernel to be linked to your program correctly.

The following example shows the correct ordering of the external directives and segment definitions.

```
EXTRN      AUDIO_INIT      : FAR
EXTRN      PLAY_PHRASE     : FAR
EXTRN      LOAD_LIBRARY    : FAR
EXTRN      FREE_LIBRARY    : FAR

MY_PROG SEGMENT
ASSUME CS:  MY_PROG

( Your program goes here )

MY_PROG ENDS
        END
```

3.6.4 COMPILED BASIC INTERFACE

The interface for BASIC Compiler is the same as the one for MS-BASIC, except that all calls are made with the CALLS statement instead of the CALL statement. (You do not need to call AUDIO_INIT in BASIC Compiler.)

UNLOADING THE VOICE KERNEL 3.7

If you are using MS-BASIC, you can unload the Voice Kernel and the memory taken up by the voice and key files by pressing the RESET button and re-booting the operating system. If you do not press RESET, the memory taken by the Voice Kernel (4K) and the memory taken by the voice and key files is not available to you.

If you are using one of the language compilers or GW-BASIC, the memory for the Voice Kernel, the voice, and the key files are set free when you exit from the program.

4

WRITING APPLICATION PROGRAMS

The combination of the Voice Editor and Voice Kernel provides a powerful applications tool for including voice and sounds in your programs. This chapter lists the steps you take to design a program for the CODEC, input and edit the sounds, and incorporate the sounds into an application program.

For simplicity, the example given here uses only a few of CODEC's features. It is a small MS-BASIC program that adds numbers together, and announces keystrokes and results on the speaker as you type them.

DESIGNING THE APPLICATION 4.1

Most already-existing applications can be easily modified to use the Voice Kernel. Only these steps are necessary to add voice to your current applications, or to applications that you are writing:

1. Record and edit the sounds you want to use in your application with the Voice Editor.
2. Add the Voice Kernel calls to your program. You must call LOAD_LIBRARY (function 4) before you call other functions that relate to your voice library; and you should call FREE_LIBRARY (function 6) before exiting the program.
3. Explain to the user how to load the Voice Kernel before running the application program, and how to unload it after leaving the program.

When you design your application, you should define the user of the application, the amount of memory in his or her computer, and the type of disk storage (floppy or hard) to be used. Since the Voice Kernel and the voice and key files can take up to 50K of RAM, this could influence the amount of memory you have for your application program. The less memory you allocate, the more often the Voice Kernel must go to disk; on a floppy-based system, this can slow down the application program.

Before recording the voices you are going to use in your application, write down all of the phrases. Then, determine if you can save disk (and RAM) space by combining some small phrases into larger ones in your application instead of with the Voice Editor. For example, the phrases "New file", "The file is empty", and "Saving the file" could be stored as "New", "file", "the", "is empty", and "Saving"; then, they could be combined in your application program. This combination process saves a significant amount of disk space, although the application program may have to access the disk more often for the different words.

4.2 THE CALC PROGRAM

The CALC program is a simple MS-BASIC program that adds two numbers together and prints the results. The user hears each prompt as it is given, and each number as it is typed in. Each time the user finishes a number, the program plays a short set of tones. The program is listed in Appendix C; it is included on your distribution diskette as CALC.BAS.

The design of the program calls for the following phrases to be spoken to the user:

- "Please enter operand 1"
- "Please enter operand 2"
- "The sum is"
- "Invalid operand"
- "One", "Two", etc.

To save disk space, these were broken into the following phrases:

- "Please enter"
- "Operand"
- "The sum is"
- "Invalid"
- "One", "Two", etc.

This saves the space of the word "operand" twice, and the words "one" and "two" once, which is about 15K in this application.

ENTERING THE PHRASES FOR CALC 4.3

Since CALC is a small applications package, you should put all of the sounds used in the program into one sound file. To do this, first run the Voice Editor, choose a disk drive on which to write the file, and select the <NEW FILE> option for the sound file. Name the file CALC to be consistent with the CALC program.

You are now presented with a set of blank stripes on the screen, indicating that you haven't loaded a sound yet. Choose a phrase from the list you are going to use, and a keyname for that phrase. This section shows how to record the phrase "Please enter". Repeat this process for each of the phrases listed in Section 4.2.

RECORDING YOUR VOICE 4.3.1

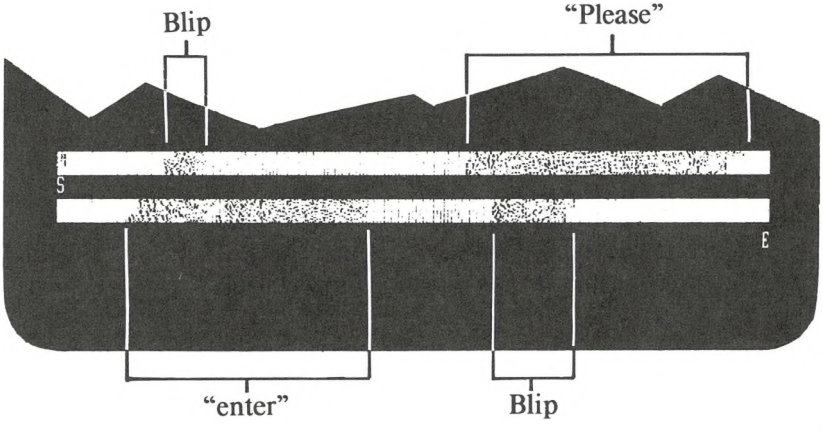
First you must record the sound. Your starting and ending pointers are at the top and bottom of the screen, which is a long period of time even at fast recording rates. Since the phrase you are recording is short, you do not need to change the pointers.

To make a high-quality recording, you need to set the clock rate faster than the default of 32; a value of 8, even though it takes up four times as much memory, provides clear voice reproduction. To get this faster clock rate:

- ► Select CONTROL (function key 3) on the main menu.
- ► The screen tells you that the current setting is 32. Press FASTER (function key 2) until the speed is 8.
- ► Press MENU (function key 1) to get back to the main menu.

Now record yourself speaking the words "Please enter". You must speak close to the microphone for good recording quality.

- ► Select SOUND (function key 2) on the main menu. This gets you to the record and play-back menu.
- ► Since the start and end pointers are already set far apart, you do not need to reset them. Press RECORD (function key 5).
- ► Press the space bar, say "Please enter" into the microphone, and press the space bar again. You now see mixed light and dark patches where you spoke, and only light where you did not speak. Although no two voices are alike, your pattern may look something like this:



There is silence at the beginning, then a small area of sound (which corresponds to the sound of hitting the space bar), a pause, an area of mixed ("Please"), another pause (the space between the two words), another area of mixed ("enter"), another pause, and a final mixed area (hitting the space bar again).

▶ To hear what this sounds like, press PLAY (function key 4), then the space bar. You hear the "blip" at the beginning, then your voice, then the "blip" at the end, then silence.

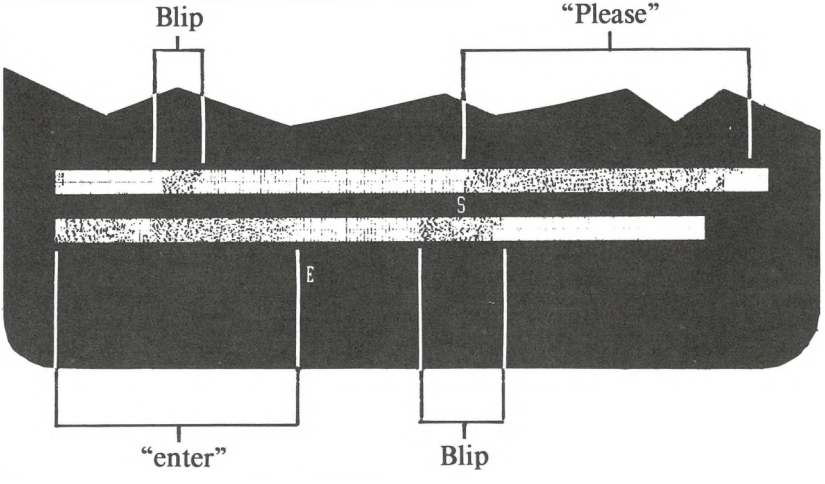
Since you do not want the user to hear the blips or the long pause after the end of the phrase, you must now edit the sound, and then save it on disk. Get back to the main menu by pressing MENU (function key 1).

EDITING YOUR PHRASE 4.3.2

You use the editing functions for a number of tasks. In this case, you want to reset the start and end of the sound (so that no silence takes up space on the disk or in memory), and you want to skip over the "blips" caused by hitting the space bar. For practice, some of the silence is taken out between the words "Please" and "enter" (thus saving memory and disk space).

1. Select EDIT (function key 5) on the main menu.
2. Move the cursor, which is between the sound bars, with the cursor control keys. First move it to just after the sound that corresponds to the word "Please". Be sure to place it after the word, not after the "blip" caused by the space bar.
3. Press START (function key 2). You see the "S" jump to the position of the cursor. The sound being deleted starts at the starting pointer, not the beginning of the screen
4. Move the cursor a few spaces to the right, but still before the beginning of the mixed patch which corresponds to the word "enter". Press END (function key 3) to set the end marker here. This delineates the area to be removed from the phrase.
5. Press DELETE (function key 6), and you see the silence between the start and end pointers disappear.

6. Move the cursor to just before the beginning of the word "Please" and press START. Move the cursor to just after the word "enter" and press END. This positions the pointers so that when you save the sound, you do not save the "blips" on either side. Your screen now shows:



With your sound edited, you are ready to save it on disk. If you want to check it to be sure that you have placed the start and end pointers correctly, and have the correct amount of space between the words, press MENU (function key 1), then SOUND (function key 2), and play the sound the same way you did before.

SAVING YOUR SOUND ON DISK 4.3.3

When you are satisfied with the sound you have recorded, save it on disk. The name for the phrase you just recorded is "PLEASE". To save the sound in the CALC.VOC file:

- Press DISK (function key 4) from the main menu. The DISK menu appears; your choices are MENU, KEYS, FILES, and HELP.
- Select KEYS (function key 2), since you want to save a new key in the current file. You have already set the start and end pointers in the last section, although you can reset them in this menu as well.
- Press SAVE (function key 5), and you are prompted in the upper left corner of the screen:

  Enter KEYNAME:

  Enter the word PLEASE, then press OK (function key 2). Your sound is entered on the disk. The key file, however, is not updated until you exit from the Voice Editor, or until you change voice files.

You should keep the "header" information in the file current. To do this, press MENU (function key 1) to get to the DISK menu, then select FILES (function key 3), then select the HEADER option (function key 3), and fill in the areas specified by the Voice Editor.

4.4 APPLICATION PROGRAMS WITH THE VOICE KERNEL

The CALC program (listed in Appendix C) is a simple example of the power of the Voice Kernel. Even if you plan to use other programming languages with the Voice Kernel, this example is applicable because it shows the basic requirements. Due to some of the limitations of MS-BASIC, however, there are some steps (such as setting constants to the values for the CALL statements) that are not required in languages such as Pascal and PL/M.

Most applications you write will only use sounds previously stored on disk with the Voice Editor; and you will know the names of these sounds when you write the program. Thus, the functions you use could be limited to LOAD_LIBRARY, PLAY_PHRASE, and FREE_LIBRARY (functions 4, 5, and 6). The CALC program uses two more functions, PLAY_NOTE and PAUSE (functions 1 and 7), to make a little music and to pause between speaking the prompt and speaking the numbers entered.

Before you use a sound library, you must open the library with the LOAD_LIBRARY function, as in line 370 of the CALC program. Once a library is open, you can send any of its sounds out the speaker with the PLAY_PHRASE function (such as lines 480 and 490). Be sure to close the library before returning to the operating system (as on line 920), since the libraries remain in memory until the Voice Kernel is unloaded.

You do not need to open voice libraries to use the PLAY_NOTE function (as in line 830 through 850).

LINES	DESCRIPTION
10–20	Print a banner at the beginning of the program.
50–150	Initialize the constants that will hold the names of the keys. The keys in the voice file and the words they contain are:

SAY0	"Zero"
SAY1	"One"
SAY2	"Two"
..	..
SAY9	"Nine"
PLEASE	"Please enter"
OPERAND	"Operand"
INVALID	"Invalid"
SUM	"The sum is"

SAYNUM$ is an array whose index corresponds to the key for that number. That is, SAYNUM$(0) = "SAY0", SAYNUM$(1) = "SAY1", etc.

Also, 130-150 combine some of the phrases with spaces between them, so that you can say a combination of the phrases with only one CALL.

180–270	Initialize the constants that hold the function numbers (PLAY.NOTE%, LOAD.LIB%, etc.), and the numeric constants used in calling PAUSE and PLAY. Also specify the tones used with PLAY_NOTE.
280–350	Initialize the segment for the AUDIO routines. This is the same code that is given in Chapter 3.
360–370	Open the voice file ("CALC").
400–410	Initialize the interim variables and the sum.

450–510 Request the first operand by printing a message on the screen, and speaking "Please enter operand one." Be sure to pause for half a second (in case the user is typing ahead), or else the word "one" will run together with the first number typed. Call the subroutine to accept operand 0, and place the result in operand 1.

520–580 Repeat the previous steps for operand 2.

590–700 Find the sum, type and say "The sum is", pause for half a second, and type and say each number of the sum. Line 660 starts at position 2 of SUM$ since the first character of the string is always a space. When all the numbers are printed and spoken, start the loop again.

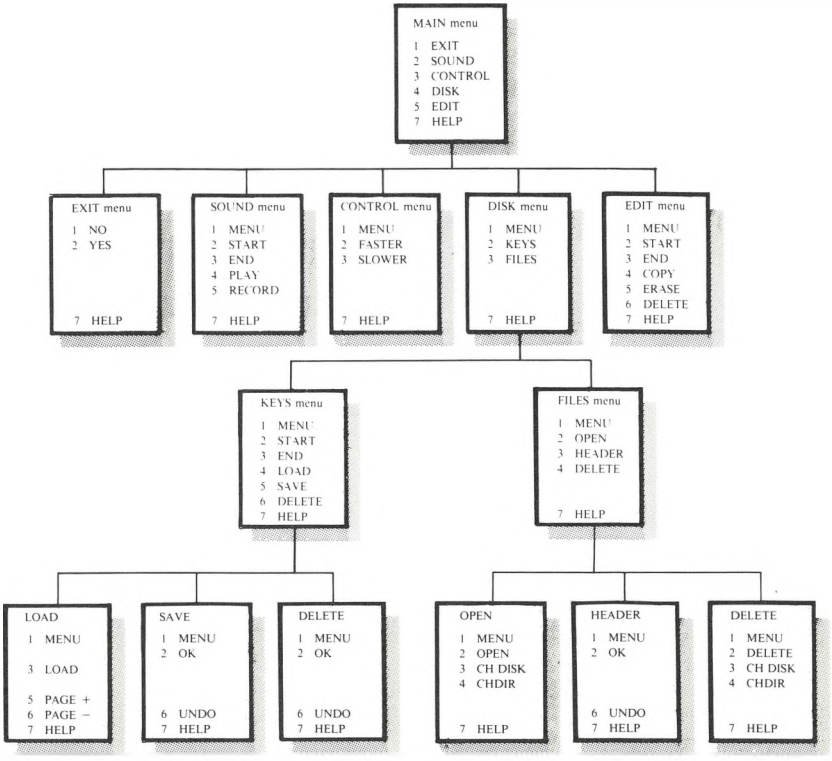
730–930 This subroutine accepts characters typed, and decides what the user wants. The number given is placed in operand 0.

If a number is typed, operand 0 is multiplied by 10 and the number is added to it. The number is printed and spoken, and the subroutine waits for the next character. If a <RETURN> is typed, indicating the end of the number, a short set of tones (the notes C, E, and G) is played, and the subroutine exits. If an "X" or "x" is typed, the CALC voice library is freed and the program ends. If any other character is typed, the words "Invalid operand" are spoken and typed, the CALC voice library is freed, and the program exits.

A

THE VOICE EDITOR MENU STRUCTURE

Here are all the menu prompts available in the Voice Editor. They are described in Chapter 2. If you need help while you are running the Voice Editor, use function key 7 (HELP) from any menu.



B

STRUCTURE OF VOICE, KEY, AND SOUND FILES

VOICE AND KEY FILES B.1

Each voice file has an associated key file (with a file type of .KEY). This key file contains a record for each sound in the sound file (with a file type of .VOC), and is maintained by the Voice Editor. The key file has a record in it for each key. The structure of each record is:

NAME	LENGTH IN BYTES	DESCRIPTION
Keyname	32	Name of the sound
Offset	4	Address of the sound in the voice file; offset in bytes from the beginning of the file
Length	4	Length of the sound (in bytes)
Unused	1	Reserved for expansion
Clock	1	Clock rate of sound at time of save
Unused	6	Reserved for expansion

The voice file has a 128-byte header record which contains information about the file. This record has user-definable information filled in by the Voice Editor. The structure of the voice file is:

NAME	LENGTH IN BYTES	DESCRIPTION
Voice file designator	1	"V" Identifies file as a voice file created by the Voice Editor
Format	1	Format of sound, contains a "0"
Revision	1	Filled in by user
Originator	16	Filled in by user
Comment	32	Filled in by user
Date	8	Filled in by user
File size	4	Number of bytes in file (including the header)
Unused	65	Reserved for expansion

The sound data follows this header.

B

B.2 SOUND FILES

The sound file holds the data recorded from within an application program, using the Voice Kernel. The structure of the sound file is:

NAME	LENGTH IN BYTES	DESCRIPTION
Clock Rate	1	The CODEC clock rate that was used during recording of this file
Reserved	15	Reserved for future use

CODEC data follows this header.

THE CALC PROGRAM LISTING

```
10 PRINT : PRINT "Verbal Adding Machine"
20 PRINT: PRINT "Enter an X to exit"
30 '
40 '   Initialize the CODEC strings
50 DIM SAYNUM$(9)
60 FOR I% = 0 TO 9
70     SAYNUM$(I%) = "SAY" + MID$(STR$(I%), 2, 1)
80     NEXT I%
90 PLEASE$ = "PLEASE"          ' name of "Please enter"
100 OPERAND$ = "OPERAND"       ' name of "Operand"
110 INVALID$ = "INVALID"       ' name of "Invalid"
120 THE.SUM.IS$ = "SUM"        ' name of "The sum is"
130 PLEASE.1$ = PLEASE$ + " " + OPERAND$ + " " + SAYNUM$(1)
140 PLEASE.2$ = PLEASE$ + " " + OPERAND$ + " " + SAYNUM$(2)
150 INVAL.OPER$ = INVALID$ + " " + OPERAND$
160 '
170 '   Initialize the CODEC
180 PLAY.NOTE% = 1             ' Function 1
190 LOAD.LIB% = 4              ' Function 4
200 SAY%  = 5                  ' Function 5
210 FREE.LIB% = 6              ' Function 6
220 PAUSE% = 7                 ' Function 7
230 PAUSE.LEN% = 500           ' Duration of PAUSE
240 PLAY.DUR% = 50             ' Duration of PLAY_NOTE
250 PLAY.C% = 262              ' C note (in Hertz)
260 PLAY.E% = 330              ' E note (in Hertz)
270 PLAY.G% = 392              ' G note (in Hertz)
280 DEF SEG = 0
290 LOWOFF = PEEK(880)
300 HIOFF  = PEEK(881)
310 LOWSEG = PEEK(882)
320 HISEG  = PEEK(883)
330 CODECSEG = (256 * HISEG) + LOWSEG
340 CODEC    = (256 * HIOFF) + LOWOFF
350 DEF SEG  = CODECSEG
360 FILE.NAME$ = "CALC"
370 CALL CODEC(FILE.NAME$, LOAD.LIB%)
380 '
390 '    Initialize the calculator
400 DIM OPERAND%(2)
```

```
410 OPERAND%(1) = 0 : OPERAND%(2) = 0 : SUM% = 0
420 '
430 '    Main loop
440 '
450 PRINT : PRINT "Enter operand 1: ";
460 '
470 '    Say "Please enter operand 1"
480 CALL CODEC(PLEASE.1$, SAY%)
490 CALL CODEC(PAUSE.LEN%, PAUSE%)
500 GOSUB 710
510 OPERAND%(1) = OPERAND%(0)
520 PRINT : PRINT "Enter operand 2: ";
530 '
540 '    Say "Please enter operand 2"
550 CALL CODEC(PLEASE.2$, SAY%)
560 CALL CODEC(PAUSE.LEN%, PAUSE%)
570 GOSUB 710
580 OPERAND%(2) = OPERAND%(0)
590 SUM% = OPERAND%(1) + OPERAND%(2)
600 PRINT : PRINT "The sum is:     ";SUM%
610 '
620 '    Say each piece of the sum
630 SUM$ = STR$(SUM%)
640 CALL CODEC(THE.SUM.IS$, SAY%)
650 CALL CODEC(PAUSE.LEN%, PAUSE%)
660 FOR I% = 2 TO LEN(SUM$)
670     NUMBER% = VAL(MID$(SUM$, I%, 1))
680     CALL CODEC(SAYNUM$(NUMBER%), SAY%)
690     EXT I%
700 GOTO 450      '   Go to main loop
710
720 '   Get the operand into OPERAND%(0)
730 OPERAND%(0) = 0
740 INPUT.CH$ = INPUT$(1)
750 '   Did they hit a number?
760 IF INPUT.CH$ < "0" OR INPUT.CH$ > "9" THEN GOTO 810
770 OPERAND%(0) = ( OPERAND%(0) * 10 ) + VAL(INPUT.CH$)
780 PRINT INPUT.CH$;
790 CALL CODEC(SAYNUM$(VAL(INPUT.CH$)), SAY%)
800 GOTO 740
810 '   Did they hit <RETURN>?
820 IF ASC(INPUT.CH$) <> 13 THEN GOTO 870
830 CALL CODEC(PLAY.C%, PLAY.DUR%, PLAY.NOTE%)
840 CALL CODEC(PLAY.E%, PLAY.DUR%, PLAY.NOTE%)
850 CALL CODEC(PLAY.G%, PLAY.DUR%, PLAY.NOTE%)
860 RETURN
870 '   Did they hit X or x?
```

```
880 IF ASC(INPUT,CH$) = 88 OR ASC(INPUT,CH$) = 120 THEN
    GOTO 920
890 '   Must have entered an invalid operand
900 PRINT "  Invalid operand"
910 CALL CODEC(INVAL,OPER$, SAY%)
920 CALL CODEC(FREE,LIB%)
930 END
```

D

VOICE KERNEL ERROR MESSAGES

When an error is detected by the Voice Kernel, one of the following error messages will be displayed on your screen. Possible causes and solutions to these errors are described here.

Bad Disk specified

A file name was supplied that had an invalid disk drive specified.

CODEC clock must be between 1 and 240

A CODEC clock value outside of the legal range was specified.

Disk is Full

While writing to the .SND file, the disk became full. Delete some files or use a different disk.

File Creation Error - Disk Directory Full

In the RECORD function, the .SND file could not be created because the disk directory is full. Delete a file or use a different disk.

Frequency must be between 20Hz and 20,000Hz

Only notes of frequencies in the range of 20Hz to 20,000Hz can be played.

Library Entry Not Found

The specified key name or number could not be found in the currently loaded library. Ensure that the key name is correct and that the correct library is loaded.

Library VOC or KEY File Not Found

In Load_Library, the specified .VOC or .KEY file could not be found on the disk. Make sure that the library name is correct and that you are using the right disk.

No Voice Library Loaded

A voice library must be loaded before any calls to Get_Key_Name, Play_Phrase, or Free_Library are executed.

Not enough keys specified

An attempt was made to load a library that has more key names than will fit in the memory allocated for keys. Reload the program with a larger number of keys.

Sound Buffer Too Small

A Sound in your .VOC file is larger than the Sound buffer that you allocated. Allocate more memory for the Sounds, or make the Sound smaller.

Specified File Not Found

In the PLAYBACK function, the specified .SND file could not be found on the disk. Ensure that the name is correct and that you are using the correct disk.

Voice Library is already loaded

Only one Voice Library can be loaded at a time. You must free the currently loaded library before loading another library.

CP/M-80 SYSTEM
User's Guide

VICTOR Publications
380 El Pueblo Road
Scotts Valley, CA 95066
(408) 438-6680

First VICTOR printing March, 1983.

ISBN 0-88182-052-0 Printed in U.S.A.

CONTENTS

SECTION DIVIDERS

1

INTRODUCTION

This guide explains how to run the CP/M-80 operating system using the CP/M-80 interface card. For a more detailed description of CP/M-80, refer to the CP/M version 2.2 User's Manual supplied by Digital Research.

This guide is written for a reader with experience in the areas described below. You are expected to be able to:

- Install the CP/M-80 card in the computer.
- Select port addresses using switches on the CP/M-80 card.
- Be familiar with CP/M and Z80 assembly language.
- Select I/O device codes.
- Install a Corvus hard disk.
- Configure the hard disk.

THE CP/M-80 CARD 1.1

The CP/M-80 card gives you access to 64K CP/M-80 version 2.2 running on a Z80 processor at 6MHz (with no wait states). This allows you to run virtually any CP/M-80 software. By typing a command you can switch between CP/M-86 and CP/M-80, and both operating systems can read or write files on the same diskette.

The CP/M-80 card also has a built-in hard disk port which allows you to connect a 5, 10, or 20 megabyte high-speed hard disk drive. Up to 64 computers can share a single hard disk by using a multiplexing unit.

1.2 CP/M-80

CP/M-80 is a disk operating system (DOS). The DOS manages your program and data files on floppy or hard disks. CP/M-80 provides a standard environment for developing and running application programs.

Many software packages are available for CP/M-80. Languages include CIS-COBOL, FORTRAN, C, APL, PL/1, FORTH, and Pascal. Also available are Microsoft's extended BASIC compiler and interpreter, Cross assemblers, and Macro assemblers. Business packages include WordStar™, dBASE II™, and Magic Wand™.

Most applications written for CP/M-80 run on any computer system supporting CP/M-80, provided that sufficient memory is available. The maximum is typically 60K bytes of RAM on an 8080 or Z80 based system. A library of software is available to the CP/M-80 user from a wide variety of suppliers.

2

INSTALLING CP/M-80 ON YOUR SYSTEM

INPUT/OUTPUT ADDRESSING 2.1

The CP/M-80 card occupies one of the 65,536 available I/O port addresses. It is normally strapped to work with I/O port 0. This can be changed if it conflicts with other I/O cards plugged into your computer's expansion slots.

You can change the switches on the CP/M-80 card (shown below) to change the port address. You must also change the port address in the CP/M-80 software (using the NEWSYS program). Consult your dealer if you require more information on input/output addressing.

Switch 1 (nearest center of card)

1	I/O address bit 0
2	I/O address bit 1
3	I/O address bit 2
4	I/O address bit 3
5	I/O address bit 4
6	I/O address bit 5
7	I/O address bit 6
8	I/O address bit 7
9	I/O address bit 8
10	I/O address bit 9

Switch 2 (furthest from center of card)

1	I/O address bit 10
2	I/O address bit 11
3	I/O address bit 12
4	I/O address bit 13
5	I/O address bit 14
6	I/O address bit 15
7	Reserved for future use
8	Reserved for future use
9	Reserved for future use
10	Reserved for future use

2.2 INSTALLING THE CARD

The CP/M-80 card can reside in any of the four expansion slots inside the computer.

WARNING: Do not attempt to install the card with power applied to the computer, or serious damage might occur to the computer or card.

Be sure to plug the card in correctly: *the component side of the card should be facing outward, away from the floppy disk drives.* Ensure also that the edge connector tracks on the card line up properly with the pins of the expansion socket.

When you have inserted the card correctly, replace the cover of the computer and turn on the power.

2.3 MAKING A WORKING COPY OF THE CP/M-80 DISKETTE

You should next make a working copy of the CP/M-80 master diskette. Do not use the original copy as a work diskette, because it contains serialized software that cannot be replaced.

Use DCOPY or PIP to copy the diskette or its files. Your dealer can provide you with manuals that describe these programs. When this is done, remove the CP/M-80 master diskette and put it in a safe place.

RUNNING CP/M-80 2.4

To try out your CP/M-80 card, type the command:

```
80 <cr>
```

The sign-on message:

```
64K CP/M-80 Version 2.2
©1982 Victor Technologies Inc.
Revision xx-month-198x
A>
```


indicates that the CP/M-80 card and the CP/M-80 operating system have been invoked. List the directory by typing DIR. Files are stored identically under CP/M-86 and CP/M-80. This is important because files created or edited under CP/M-86 can be accessed under CP/M-80 and vice versa.

DIFFERENCES BETWEEN CP/M-80 AND CP/M-86

While CP/M-86 is designed to be similar to CP/M-80, there are some important differences between the two systems. The main differences are:

1. CP/M-80 commands have file type COM, and CP/M-86 commands have file type CMD. Command files created for one operating system will NOT work on the other. Many source files, such as Microsoft BASIC programs, are interchangeable between CP/M-80 and CP/M-86.
2. The amount of RAM addressable under CP/M-80 is limited to 64K because, like most 8-bit CPU's, the Z80 is designed to address 65536 bytes of memory.
3. While the two operating systems are very similar from the user's point of view, there are major differences in internal organization. This reflects the different architecture of the 8080 and the 8086 CPU families. For example, BDOS calls are made by calling location 5 under

CP/M-80; CP/M-86 uses an interrupt 224 instruction. More details of CP/M-80 internal operation are given in Chapter 4.

4. Assembly language programming is different under CP/M-80. There is no memory segmentation on the Z80 and programs always start at location 100 hex. The programs ASM, LOAD, and DDT are used to assemble an 8080 source program, generate a command file, and debug 8080 programs, respectively. Command (.COM) files under CP/M-80 are binary code images loaded in at location 100 hex and executed. There are no headers as in CP/M-86 CMD files.

RETURNING TO CP/M-86

To get back to CP/M-86, type:

```
86 <cr>
```

You can switch between CP/M-80 and CP/M-86 by using the "80" and "86" commands. I/O device assignments and the current disk drive are unaltered when you switch operating systems. If the computer fails to recognize a CP/M-80 or a CP/M-86 command, you are probably in the wrong operating system.

AUTOLOAD COMMANDS

The NEWSYS program, described in Appendix B, allows you to insert an autoload command into a CP/M-80 operating system. An autoload command is performed automatically when that copy of CP/M-80 is run.

For example, you can enter:

```
dir <cr>
```

to print out a directory listing each time the CP/M-80 system is loaded. You can also make CP/M-80 directly load a BASIC interpreter, or your own turnkey application program.

The A (autoload) option in NEWSYS displays the current autoload command. A NEWSYS prompt asks if you want to change or remove the autoload command. If you do not want to make changes, type Return to return to the main menu. If you want to insert an autoload command, type Y and a Return. Another prompt asks you to enter the new autoload command, just as it would be typed, in reply to the CP/M prompt. To remove the existing autoload command, type Return at this prompt.

PIP COMMAND 2.5

When you use PIP to copy CMD files under CP/M-80, or to copy COM files under CP/M-86, always use the [O] option. This option tells PIP that the files being copied are object files rather than ASCII text files, and files will not be truncated if an ALT-Z (1A hex) character is found.

3

CP/M-80 UTILITIES

MEMTEST 3.1

MEMTEST is a memory test program for the CP/M-80 card memory. It is supplied on your CP/M-80 master disk.

To invoke the memory test, type:

```
memtest <cr>
```


MEMTEST performs a complete cycle of tests on the 64K bytes of RAM on the CP/M-80 card. Ten different tests are performed (passes 1 to 10). After each test, a cumulative error count is printed. After the tenth pass, the cycle is repeated (passes 11 to 20) and so on until you press a key or perform a hardware reset.

If errors occur, the cumulative address range of the faulty locations is displayed. A data byte is also displayed, indicating the faulty bit positions with 1's. The display is used by your dealer or service engineer.

DUMP 3.2

The DUMP utility is a file dump that displays the contents of a file on the console screen. The dump is sixteen bytes per line, and the absolute address is displayed on the left in hex. The syntax is:

```
DUMP drive designation:filename.ext <cr>
```

4

CP/M-80 PROGRAMMING

The following information may be helpful to users writing application programs in machine code for the CP/M-80 card.

Z80 MEMORY MAP 4.1

Page zero is reserved by CP/M:

0000 to 0002	Jump to warm boot routine at FA03
0003	IOBYTE
0004	Current drive #
0005 to 0007	Jump to BDOS entry point
0008 to 003F	Reserved by CP/M for the 8080 interrupt vectors. CP/M-80 card does not use any interrupts but some CP/M application programs may use the restart vectors for code efficiency. The CP/M debuggers use restart vector 7 for their breakpoints.
0040 to 005B	Used internally by the CP/M-80 BIOS
005C to 007F	CP/M's 36-byte default file control block (FCB)
0080 to 00FF	128 byte default buffer for disk transfers. Also used as a console buffer for passing command parameters.
0100 to E3FF	Transient Program Area (TPA). This is a general purpose RAM area. CP/M application programs (transients) load in here, and execution starts at 100 hex.
E400 to EBFF	CP/M's Console Command Processor (CCP) gets loaded here after a cold or warm boot. This is the program that processes CP/M command lines typed at the console, and executes them. The CCP may be overwritten during execution of a user program.
EC00 to F9FF	CP/M's Basic Disk Operating System (BDOS) gets loaded here after a cold or warm boot. This is a collection of routines to perform disk or I/O related functions

required by the CCP. The BDOS can also be called by user programs, or overwritten if it is not needed.

FA00 to FFFF — The CP/M-80 Basic Input/Output System (BIOS) gets loaded here on cold boot only. It should not be overwritten by user programs. This is the hardware-dependent part of CP/M-80 and comprises a collection of low-level I/O routines such as console, printer and disk I/O. These routines are used by the BDOS and can also be called directly from user programs.

4.2 THE IOBYTE

The IOBYTE (CP/M memory location 3) is a byte that determines which physical I/O device is linked to each of the four CP/M logical devices. The I/O Byte setting can be examined and set by using the STAT command. Physical devices in brackets are not used:

bits 0-1 (least significant) are the console device:
00 = TTY: 01 = CRT: [10 = BAT: 11 = UC1:]

bits 2-3 are the reader device:
00 = TTY: 01 = PTR: [10 = UR1: 11 = UR2:]

bits 4-5 are the punch device:
00 = TTY: 01 = PTP: [10 = UP1: 11 = UP2:]

bits 6-7 are the list device:
00 = TTY: 01 = CRT: [10 = LPT: 11 = UL1:]

5

ACCESSING RAM AND I/O

When running under CP/M-80 you may need to access the 8088's memory and I/O space:

- To write directly to the screen or character RAM.
- To write to an I/O card.
- To make use of the RAM for extra storage.

WRITING TO RAM 5.1

The specified number of bytes are written to the specified memory address.

```
MVI   A, 0DH
CALL FA33H
MVI   A, low address
CALL FA36H
MVI   A, high address
CALL FA36H
MVI   A, low byte count
CALL FA36H
MVI   A, high byte count
CALL FA36H
MVI   A, byte1
CALL FA36H
MVI   A, byte2
CALL FA36H
.
.
.
```

5.2 READING FROM RAM

The specified number of bytes are read from the specified memory address.

```
MVI   A, 0CH
CALL  FA33H
MVI   A, low address
CALL  FA36H
MVI   A, high address
CALL  FA36H
MVI   A, low byte count
CALL  FA36H
MVI   A, high byte count
CALL  FA36H
CALL  FA39H
STA   byte1
CALL  FA39H
STA   byte2
.
.
.
```


5.3 EXECUTE AN 8088 SUBROUTINE IN RAM

Control is temporarily transferred to the 8088 processor of the computer, at the specified memory address. Control can be returned to the Z80 by executing a "RETF" instruction.

```
MVI   A, 0EH
CALL  FA33H
MVI   A, low address
CALL  FA36H
MVI   A, high address
CALL  FA36H
```

FIND THE MAXIMUM RAM ADDRESS 5.4

This function returns a 16-bit segment value representing the top of memory available to the user. Memory is available from address 02C00 hex up to this address during CP/M-80 operation. To calculate the number of bytes available, subtract 02C00H from the number returned by this function.

```
MVI   A, 16H
CALL  FA33H
CALL  FA39H
STA   low max address
CALL  FA39H
STA   high max address
```

SET THE SEGMENT VALUE 5.5

This function sets up the 16-bit base segment value for read, write, and execute operations in memory. It defaults initially to F000 hex (the start of the screen RAM).

```
MVI   A, 12H
CALL  FA33H
MVI   A, low segment value
CALL  FA36H
MVI   A, high segment value
CALL  FA36H
```

5.6 WRITE TO AN I/O PORT

An 8-bit value is output to the 16-bit 8088 I/O port address.

```
MVI   A, 15H
CALL FA33H
MVI   A, low port address
CALL FA36H
MVI   A, high port address
CALL FA36H
MVI   A, data value
CALL FA36H
```

5.7 READ FROM AN I/O PORT

An 8-bit data value is input from the specified 16-bit 8088 I/O port address.


```
MVI   A, 14H
CALL FA33H
MVI   A, low port address
CALL FA36H
MVI   A, high port address
CALL FA36H
CALL FA39H
[data value is in A]
```

5.8 ACCESSING THE CORVUS DRIVE DIRECTLY

The Corvus ports are:

Data - Z80 port #1
Status in - Z80 port #0 (bit 0 = drive ready, bit 1 = drive active)

Examples of Corvus commands are provided by the SEMA4, MIRROR, PUTGET, and CDIAGNOS utility programs whose source code is provided on the CP/M-80 disk.

A

NOTES ON CP/M-80 PACKAGES

INTRODUCTION A.1

The main application for your CP/M-80 card is to run the commercially available CP/M software packages. When you receive this manual and diskette, you should first make a back-up copy of the master diskette for the package.

To format a diskette you use the FORMAT utility running under CP/M-86. Your dealer can provide you with a manual that describes this utility.

Running your application package on the CP/M-80 card is just like running it on any 8080/Z80 CP/M 2.2 system.

WORDSTAR A.2

Use the "X" terminal option (Heath H89/H19) when configuring. Also alter the delay bytes at hex memory locations 2AE and 2AF to zero for maximum operating speed.

B

CORVUS HARD DISK INTERFACE

INTRODUCTION B.1

The hard disk interface on the CP/M-80 card can drive a Corvus 5, 10, or 20 megabyte hard disk. CP/M-80 operation using a Corvus drive is similar to operation using floppy disks, but is much faster and offers more on-line storage capacity.

DRIVE ACCESS TIME:	10 or 20 Mbyte drive:	80ms maximum 40ms average
	5 Mbyte drive:	240ms maximum 125ms average
5Mb Winchester	file space (2 × 2788 Kbytes):	5576 Kbytes
	block size:	8 Kbytes
	directory entries (2 × 256):	512
10Mb Winchester	file space (2 × 4704 Kbytes):	9408 Kbytes
	block size:	8 Kbytes
	directory entries (2 × 256):	512
20Mb Winchester	file space (4 × 4704 Kbytes):	18816 Kbytes
	block size:	8 Kbytes
	directory entries (4 × 256):	1024

During normal operation you generally want drive A: to be on the Corvus drive, because it is used most. Because CP/M-80 version 2.2 only supports drives up to 8 Mbytes, the Corvus drive is divided into logical CP/M-80 drives of about 5 Megabytes each. These are typically:

5 Mbyte drive	CP/M-80 drives A: and B:
10 Mbyte drive	CP/M-80 drives A: and B:
20 Mbyte drive	CP/M-80 drives A:, B:, C: and D:

B.2 SETUP PROCEDURE FOR A NEW DRIVE

With the power turned off, connect the Corvus drive to the CP/M-80 card using the 34-pin ribbon connector. *Be sure that the cable orientation is correct, or it may damage the card or the drive.* The CP/M-80 card should be positioned so the red stripe of the ribbon cable is in the top left corner of the card. The red stripe on the Corvus drive is nearest the right side of the socket marked "processor" when the drive is viewed from the rear.

> **WARNING:** On the revision B Corvus drives, inspect the row of four switches located under the front panel of the drive, below the Busy, Ready and Fault lights. Reading from left to right, these switches are the LSI-11, Constellation, Format, and Reset switches.
>
> If a Constellation unit is not part of the configuration, all four switches should be in the left position during normal operation. If a Constellation unit is attached, the CONSTELLATION SWITCH ONLY (second switch from the left) should be switched to the right.
>
> The Format switch must remain in the left position during normal use or the drive controller firmware may be overwritten. The drive reset switch should never be used when a disk operation is in progress, or data may be destroyed.

Power up the computer and the Corvus disk drive. The Busy and Fault lights on the drive should go out after a few seconds, leaving the Ready light on. This indicates that the drive is up to speed. You need to reconfigure a copy of your CP/M-80 operating system to work with the hard disk (use the NEWSYS program). *Never reconfigure your master of the CP/M-80 disk.* Place the disk to be reconfigured in drive A and load CP/M-80. Type:

```
NEWSYS <cr>
```

The NEWSYS program displays the prompt:

```
Reconfigure "80.CMD" on which drive (A-P) ?
```

Reply with:

```
A <cr>
```

When the system to be reconfigured is loaded into memory, NEWSYS displays a menu of options:

```
A - autoload command
D - disk drive assignments
I - change I/O port number
M - drive M setup
S - save reconfigured system
E - execute reconfigured system
Q - quit to CP/M-80
```

Type D and a Return. CP/M-80 displays several choices of disk drive assignments:

```
1. A, B = floppy
2. A, B = floppy,  C, D       =  5 Mbyte Corvus
3. A, B = floppy,  C, D       = 10 Mbyte Corvus
4. A, B = floppy,  C, D, E, F = 20 Mbyte Corvus
5. C, D = floppy,  A, B       =  5 Mbyte Corvus
6. C, D = floppy,  A, B       = 10 Mbyte Corvus
7. E, F = floppy,  A, B, C, D = 20 Mbyte Corvus
8. Special drive assignment (not recommended)
```

B

Select option 2, 3, or 4 (depending on drive size) and a Return. Then type:

```
E <cr>
```

to execute the reconfigured system with drives C, D (E, F) as the hard disk.

Next, clear the directory on the hard disk by typing:

```
CLEAR <cr>
```

The directory clearing program asks for a disk drive name. Type:

```
C <cr>
```

The program asks if you are sure, and then clears the directory on drive C. Next, the program prompts you for another drive name. At this point, type:

```
D <cr>
```

to clear the drive D directory. If you have a 20-Mbyte drive, clear the directories on drives E and F in the same way. After all the directories have been cleared, type a Return to exit from the directory clearing program.

Now you should be able to read and write CP/M-80 files on the hard disk as if it were a floppy disk. Copy all the files from your master CP/M-80 disk with the command:

```
PIP C: =*.*[o]
```

When the copying has finished, you can run programs from the hard disk instead of the floppy disk. To run a program from the hard disk, indicate the drive name of the hard disk before typing the program name. For example,

```
C:STAT
```

runs STAT.COM from drive C.

When you are satisfied that the hard disk is functioning properly, the final step is to reconfigure the system with the Corvus as the main system drive—namely, drive A:. Use NEWSYS to reconfigure your system again, but this time specify drive configuration 5, 6 or 7 (depending on drive size), and use the S command to save the reconfigured system on the floppy disk.

DRIVE M OPERATION

Drive M is a feature that allows you to treat expansion RAM cards as disk drives. You can transfer frequently used programs and files into RAM (drive M) at the start of a session. The RAM card behaves as a high-speed, volatile, CP/M disk drive. All of the normal CP/M commands can be used with drive M (STAT, PIP, ERA, DIR, etc.).

CAUTION: When all of available RAM is used, the system may hang up. Use this feature with caution.

CARE OF THE DISK DRIVE

The Corvus drive is considerably more durable than a floppy disk unit, but observe the following points to avoid damaging your drive or data:

1. Always power the drive up last and switch it off first.
2. Do not obstruct the air holes on the drive or keep it in an enclosed space without adequate ventilation.
3. Keep the drive away from strong magnetic fields.
4. Ensure that the front panel switches are always kept in their correct positions, as described in Section B.2.
5. Avoid excessive mechanical shocks, such as dropping the drive.

CAUTION: Run a format check on any newly shipped drive before any valuable data is stored on it (see Section B.3, "Diagnostic Utility").

CORVUS UTILITY PROGRAMS B.3

The CP/M-80 master disk contains Corvus documentation files, which can be displayed using the TYPE command. It also contains support programs for the hard disk. These programs are designed to run under CP/M-80, and include:

- Diagnostic utility.
- Mirror utility.

DIAGNOSTIC UTILITY

The Corvus diagnostic program includes a format check function. A format check should be run on new hard disk drives to determine whether the hard disk was damaged in shipping.

To use FORMAT CHECK, run the CDIAGNOS program from the utility disk and answer the menu prompt with the command "2" (FORMAT

CHECK). Specify drive number 1 to start the format check. While FORMAT CHECK is in operation, pressing any key produces the display:

```
FORMAT CHECK IN PROGRESS . . .
```

When the check is finished, the program displays the number of bad sectors found. If there are no bad sectors, press ALT-C to return to CP/M-80.

If bad sectors are found, attempts are made to correct them without destroying the data. Repeat the format check until no bad sectors are found.

MIRROR UTILITY

The Corvus Mirror option supported by the CP/M-80 card provides a fast and inexpensive way to back up the contents of Corvus hard disks on a commercially available video cassette recorder. Two speeds, Normal or Fast, record the data with quadruple or dual redundancy.

The Normal backup speed is about 7.5 Kbytes per second (recording with quadruple redundancy). A 10 Mbyte drive can be copied in about 20 minutes. The data capacity of a video cassette is about 100 Mbytes.

There are four commands in the Mirror utility: BACKUP, VERIFY, RESTORE, and IDENTIFY.

B.4 HARD DISK ERRORS

The message "Hard disk error" is displayed during CP/M-80 operation when a disk read/write failure occurs. If this occurs while a file is being read/written with the hard disk, the transferred file is bad. A format check should be run (see the section on the diagnostic utility), and the file operation which caused the error should be re-run if possible. Hard disk errors are rare and are usually caused during shipping or by a power supply failure.

PC Comm User's Guide

First VICTOR printing March, 1983.

ISBN 0-88182-048-2 Printed in U.S.A.

CONTENTS

CHAPTERS

IMPORTANT SOFTWARE DISKETTE INFORMATION

For your own protection, do not use this product until you have made a backup copy of your software diskette(s). The backup procedure is described in the user's guide for your computer.

Please read the DISKID file on your new software diskette. DISKID contains important information including:

- The product name and revision number.
- The part number of the product.
- The date of the DISKID file.
- A list of the files on the diskette, with a description and revision number for each one.
- Configuration information (when applicable).
- Release notes giving special instructions for using the product.
- Information not contained in the current manual, including updates, additions, and deletions.

To read the DISKID file onscreen, follow these steps:

1. Load the operating system.
2. Remove your system diskette and insert your new software diskette.
3. Enter —

   TYPE DISKID

   and press Return.
4. The contents of the DISKID file is displayed on the screen. If the file is large (more than 24 lines), the screen display will scroll. Type ALT-S to freeze the screen display; type ALT-S again to continue scrolling.

1

INTRODUCTION

With PC COMM, your computer can communicate with the IBM Personal Computer. You can receive files, both text and object, from the IBM PC. The interface connection is made through the IBM Parallel Printer Adapter (PPA) and the parallel port on your computer. The PC COMM eliminates the need for a serial interface board for the IBM machine. Since the PPA contains only outputs and status inputs, this interface connection ensures that your files cannot be retrieved by the IBM machine.

PC COMM has the following capabilities.

CENTRONICS CONNECTION:

The capabilities of the parallel port on your computer let the interface be accomplished with a Centronics printer connection. You don't need to buy a serial connector board for the IBM Personal Computer.

PROGRAMMING LANGUAGE:

BASIC is used for implementing the IBM side of the interface, allowing you to access and enhance the software with the standard IBM Personal Computer package.

BINARY TRANSFER PROTOCOL:

Any type of file can be transferred with this interface.

CONTINUED DATA RETRIEVAL:

Multiple files can be transferred with only one initiation of the program.

COMPLETE ERROR CHECKING:

An error-checking procedure checks byte errors and position-bit errors, providing you with ample security in transmission.

PROGRAM MONITORING:

You can watch the progress of the transfer by using the built-in program flow monitoring feature.

USER ABORT CAPABILITIES:

1

You can abort the transfer at any time during the process. Type an "a" (for Abort) to terminate the transfer, close the files on both computers, and delete the receiving file.

2

USING PC COMM

OVERVIEW 2.1

PC COMM lets you transfer files from an IBM Personal Computer (PC). The PC COMM package contains two programs: one for your computer and one for the IBM PC. PC COMM transmits existing files containing any type of data from the IBM PC to a file on your computer by using a parallel communications interface. An error-checking protocol combines an additive checksum with byte rotating and complementing manipulations to find bit-position and block transfer errors.

USER INTERFACE 2.2

PHYSICAL CONNECTIONS 2.2.1

The hardware configuration of your computer's parallel port enables a direct connection to the IBM machine. The connection accesses your computer's parallel port and the PPA on the IBM PC.

METHOD OF OPERATION 2.2.2

The PC COMM interface sends and receives multiple files after one initiation of the program. To do this, you build a text file with an editor program containing the following: the drive, file name, and extension of the file being sent from the IBM PC; and the drive, file name, and

extension assigned to the file on your computer's disk. The name of each file to be transmitted must be followed by a carriage return/line feed sequence. The attributes for the two files should be separated by a space. If a file has the same attributes on your computer's disk as on the IBM disk, you don't need to repeat the attributes. If no disk drive is specified for your computer's file, however, the default drive is always assumed.

To begin the program, you enter an "@" symbol, followed by the name of the file containing the attributes of the files to be transferred. If only one file is being transferred, only the file attributes need to be entered following the program prompt (keeping the same format as specified for the text file).

You can stop file transmission at any time during the transfer by typing "a" (for Abort). When a transfer is aborted, only the file currently being transmitted is affected. The files on the IBM and on your computer are closed, and the receiving file is deleted.

2.3 PROGRAM INITIALIZATION

Initialize the communications interface by typing the name of the receive program at your keyboard:

PCCOMM

Then, copy BASICA.COM onto your PC COMM diskette (IBM) and at the IBM PC keyboard type:

BASICA PCXMIT

The program then prompts you for the name of the file being sent from the IBM PC:

```
INPUT FILENAME TO BE TRANSFERRED
```

You could enter the following to transfer several files contained in the text file MULTIFIL:

```
@MULTIFIL
```

If only the single file "ONE.EXT" is being transferred, enter the following:

```
B:ONE.EXT B:ONECOPY.EXT
```


Synchronization of these two programs is handled by the software; you need only make sure that your computer's program is initialized first.

When both programs are initialized, the program name, version number, and release date of PC COMM appear on the screen. This tells you that the program has been initialized and the file transfer is in progress.

```
PCCOMM   VERSION 1.0   xx-xx-xx
Transmission of 'filename' initiated
PCXMIT   VERSION 1.0   xx-xx-xx
```

Now, the sending/receiving process starts. If there is not enough available space on your disk, PC COMM tells you that the data transfer cannot be completed:

```
INSUFFICIENT SPACE ON DISK FOR FILE
```

PC COMM monitors the transfer process and informs you of the current transfer status. The first item displayed is the number of blocks being transferred (a block contains 128 bytes). As each block is transferred, the block count is incremented and displayed. (Extra bytes that do not fill a block are considered a block in the monitoring scheme.)

The IBM PC response looks like this:

```
THE NUMBER OF BLOCKS TO BE TRANSMITTED----3
BLOCK BEING TRANSMITTED---1[2,3]
```

Your computer's response looks like this:


```
THE NUMBER OF BLOCKS TO BE RECEIVED----3
BLOCK BEING RECEIVED---1[2,3]
```

At the end of the transfer, a "successful completion" message appears on the screen along with the total number of bytes that were transferred:

```
TRANSMISSION OF "FILENAME" COMPLETE (TOTAL BYTES: 515)
```

(for your computer)

```
TRANSMISSION OF "FILENAME" COMPLETE (TOTAL BYTES: 515)
```

(for the IBM PC)

A

ERROR MESSAGES

ERRORS ON YOUR COMPUTER A.1

THE DESTINATION FILE CANNOT BE CREATED

The destination disk is full. Place disk with available space into the drive and retry process.

THE DESTINATION FILE CANNOT BE OPENED

This error is caused by a system error when opening the destination file.

THE DESTINATION FILE CANNOT BE CLOSED

This error is caused by a system error when closing the destination file.

DISK ERROR: INSUFFICIENT SPACE ON DISK

The software tried to write the data to the file but didn't find enough space on disk. The file is closed and deleted from the disk directory.

DATA CHECKSUM ERROR/
HEADER CHECKSUM ERROR

The data was written to the file but errors were incurred on the communication lines during the process. Retry.

CANNOT DELETE FILE

This keeps you from overwriting an existing file. The transfer process is aborted. To use this file name, you need to delete the existing file with that name from your computer's disk before the transfer is restarted.

RESPONSE TIMEOUT

Sufficient time was given for the sender/receiver (the computer not receiving the message) to respond and no response occurred. This error happens when too much time passes between the initiation of the IBM program and your computer's program. This error message keeps you from hanging the system in an infinite wait loop.

A.2 ERRORS ON THE IBM PC

TRANSMIT FILE NOT FOUND

The specified file name could not be found. Check spelling of file name and make sure that the correct disk is being accessed.

TRANSMIT FILE CANNOT BE OPENED

Same as the "DESTINATION FILE CANNOT BE OPENED" error on your computer.

TRANSMIT FILE CANNOT BE CLOSED

Same as the "DESTINATION FILE CANNOT BE CLOSED" error on your computer.

Audio Input Option

VICTOR Publications
380 El Pueblo Road
Scotts Valley, CA 95066
(408) 438-6680

Second VICTOR printing April, 1983.

ISBN 0-88182-050-4 Printed in U.S.A.

CONTENTS

CHAPTERS

1

INTRODUCTION

The audio option kit contains the following component parts:

audio preamp board with attached microphone input cable
PCB mounting standoff
microphone input jack with mounting plate
condenser microphone
audio input extension cable
mounting screw and nut
three plastic standoffs

After you unpack the components, follow the instructions in this booklet to prepare the audio option for use. The instructions describe how to:

- ► Remove the rear panel and top cover from the processor.
- ► Install the audio preamp board on the CPU board.
- ► Install the audio extension cable (internal hard disk).
- ► Route the audio connector cable through the rear panel.
- ► Connect the audio input jack to the rear panel.
- ► Prepare the microphone for use.

2

UNPACKING INSTRUCTIONS

Figure 1 shows component parts as packaged. Figure 2 shows microphone parts as packaged. Component parts in both figures are labeled. Carefully remove components from the packing material. Save the microphone package for storing the microphone.

Figure 1: Audio Input Option Components, As Packaged

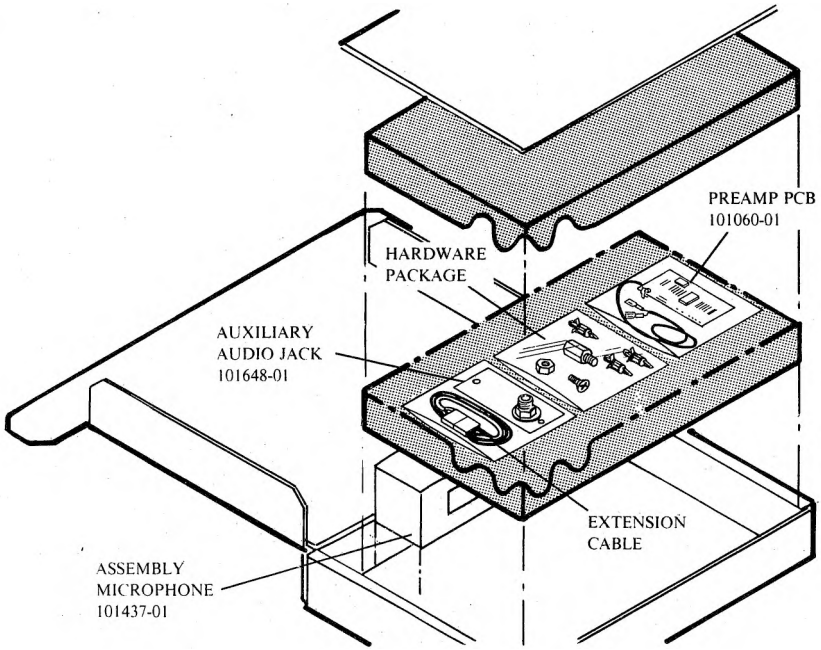
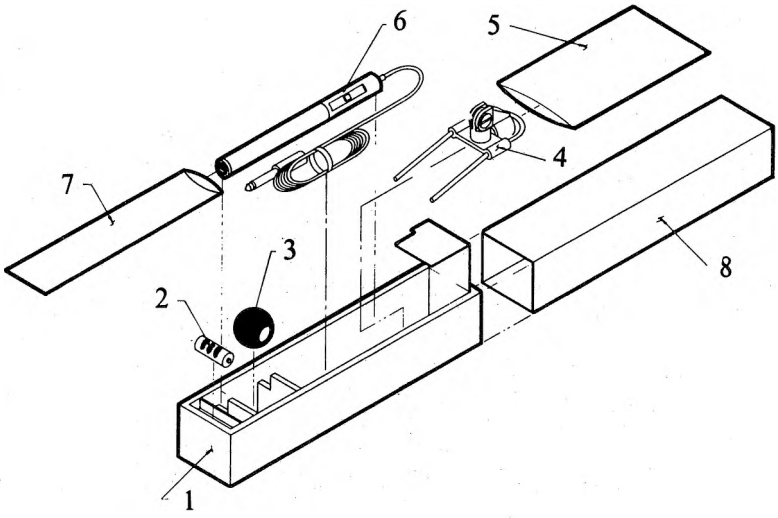


Figure 2: Microphone Components, As Packaged



1. BOX
2. BATTERY, N-TYPE, 1.5V
3. FOAM, WIND SCREEN
4. STAND, MICROPHONE
5. BAG, PLASTIC
6. MICROPHONE, CONDENSER
7. BAG, PLASTIC
8. BOX SLEEVE

3

SYSTEM DISASSEMBLY

To install the audio input option you must first disassemble the system. The following steps apply to all models:

1. Turn off the power and unplug the power cord.
2. Disconnect and carefully set aside the CRT and keyboard.
3. Unscrew and remove the rear panel cover (4 screws).
4. Slide the top cover back and out of the front cover.
5. Remove the power supply.

4

AUDIO PREAMP BOARD INSTALLATION

This section describes how to install the audio preamp board. Three configurations are described:

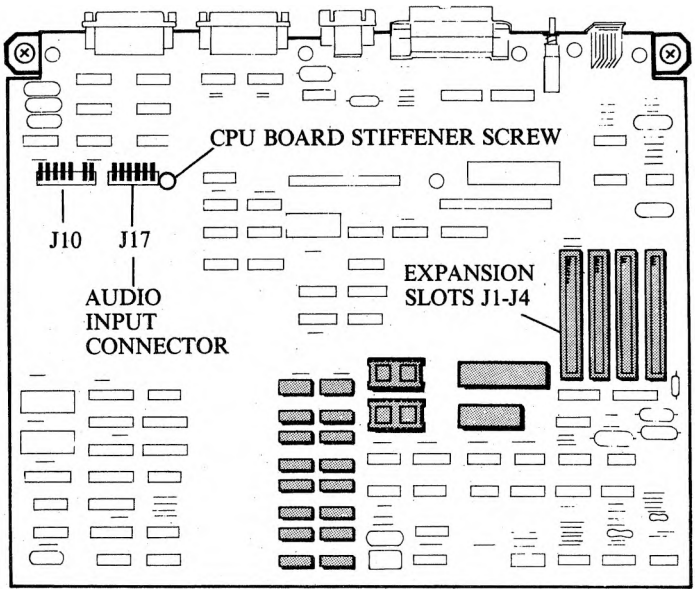
- ► 128K CPU board with dual floppy drives.
- ► 256K CPU board or 256 CPU board with gate arrays.
- ► 128K CPU board with internal hard disk.

128K CPU BOARD WITH DUAL FLOPPY DRIVES 4.1

Figure 3 shows where to connect the audio preamp board on the CPU board. To install the audio preamp board:

1. Remove the CPU board stiffener screw (Figure 3).
2. Install the threaded end of the standoff in the stiffener screw position.
3. Plug the preamp header into the audio input connector (Figure 3).
4. Install the stiffener screw (removed in step 1) through the preamp board and into the standoff.
5. The three plastic standoffs, kepnut, screw, and extension cable are not used in this configuration and may be discarded.

Figure 3: Audio Preamp Board Location — 128K CPU Board

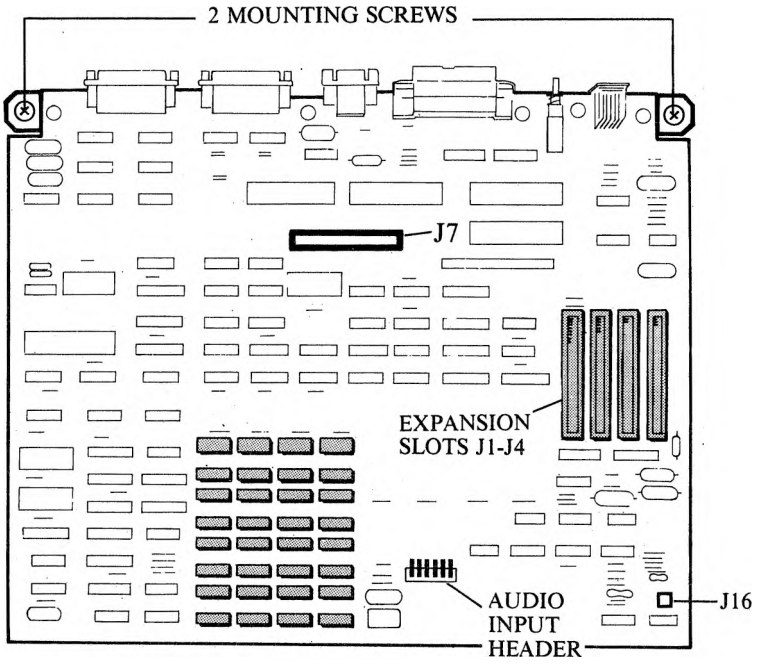


4.2 256K CPU BOARD OR 256K CPU BOARD WITH GATE ARRAYS

Figure 4 shows how to remove the CPU board from the mainframe. To remove the CPU board:

- ► Unplug ribbon cable P7 on the CPU board (to the disk drive).
- ► Loosen the 2 mounting screws on the rear of the CPU board.
- ► Unplug the speaker connector P16 on the CPU board.
- ► Slide the CPU board out of the mainframe.

Figure 4: 256K CPU Board Removal



Refer to Figure 5 as you do the following steps to install the audio preamp on the 256K CPU board without gate arrays.

1. Insert the threaded end of the metal standoff through the hole between J17 and C93. Screw the kepnut onto the threaded end of the standoff from the bottom of the CPU board.

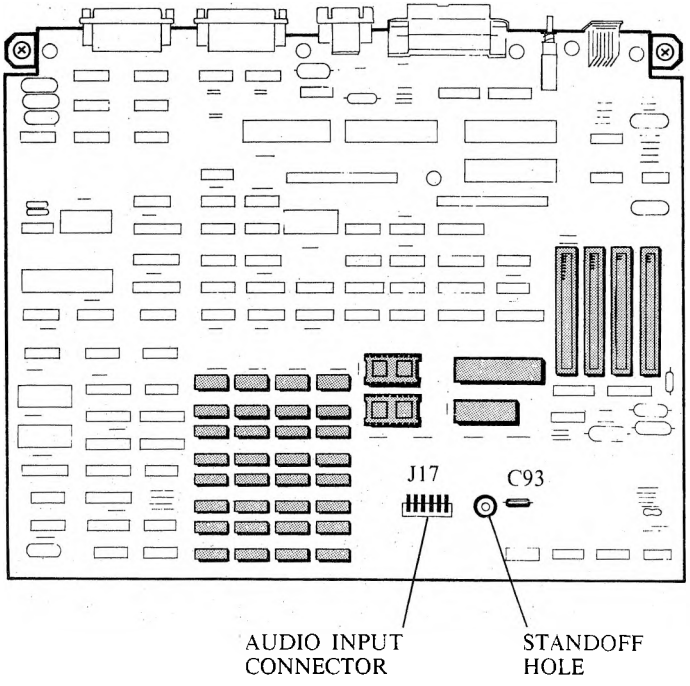
2. Plug the preamp header into the audio input connector. Be sure that the hole in the preamp board is directly over the standoff.

3. Install the screw provided through the preamp board hole and into the standoff.

4. Reinstall the CPU board into the mainframe.

5. The three plastic standoffs and extension cable are not used in this configuration and may be discarded.

Figure 5: Audio Preamp Board Location — 256K Board Without Gate Arrays



Refer to Figure 5A as you do the following steps to install the audio preamp on the 256K CPU board with gate arrays.

1. Remove the CPU board stiffener screw.

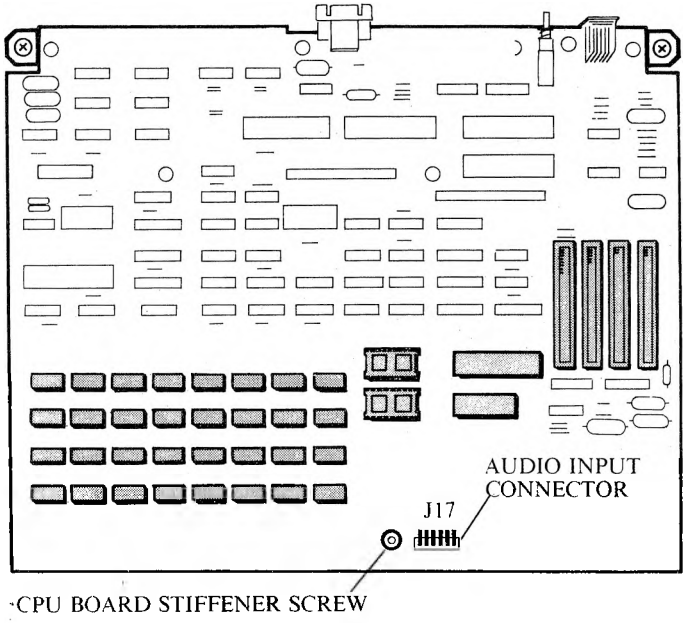
2. Install the threaded end of the standoff in the stiffener screw position.

3. Plug the preamp header into the audio input connection.

4. Install the stiffener screw through the preamp board and into the standoff.

5. The three plastic standoffs, kepnut, screw and extension cable are not used in this configuration and may be discarded.

Figure 5A: Audio Preamp Board Location — 256K CPU Board With Gate Arrays

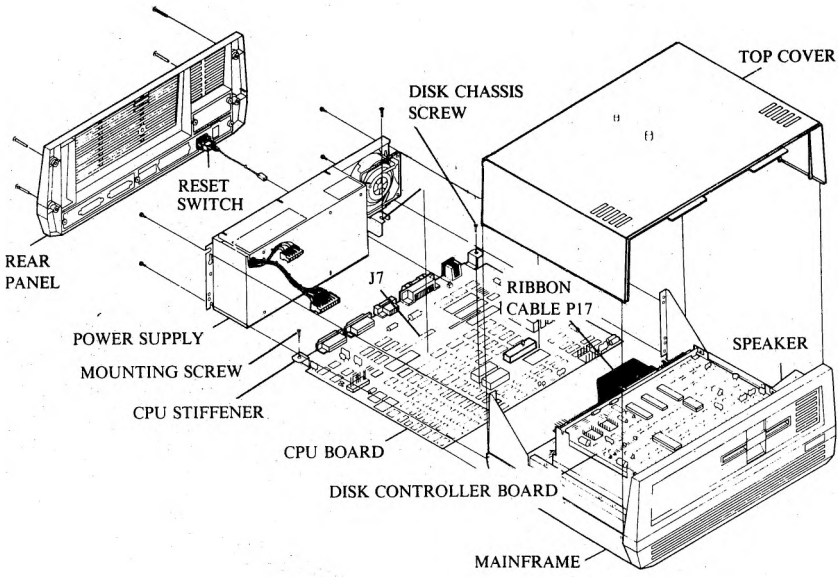


128K CPU BOARD WITH INTERNAL HARD DISK DRIVE 4.3

Refer to Figure 6 as you do the following steps to remove the drive chassis assembly from the mainframe.

1. Loosen the 4 screws holding the drive chassis assembly to the mainframe.
2. Unplug ribbon cable P17 from the drive board (to the CPU board).
3. Unplug the speaker connector P16 from the CPU board.
4. Slide the drive chassis assembly forward and remove it from the mainframe.

Figure 6: Hard Disk — Drive Chassis Assembly Removal



AUDIO INPUT EXTENSION CABLE

Figure 7 and Figure 8 show how to install the extension cable. The cable connects the audio output from the preamp board to either type of CPU board. Do the following steps to install the extension cable.

1. Unplug the ends of the extension cable from each other.

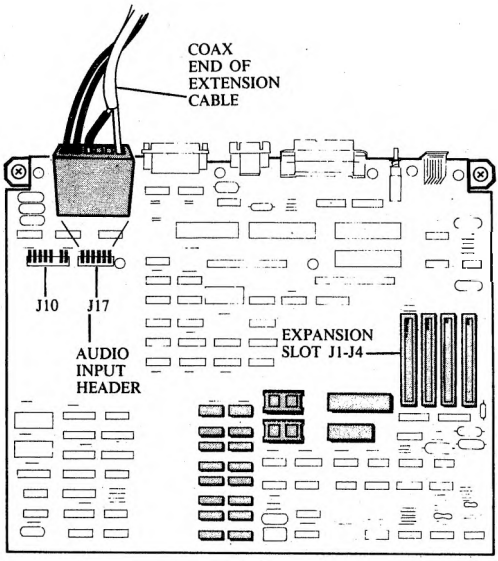
2. Note the orientation of the cable: the coax wire in the socket (female) end of the cable goes toward the expansion slots.

3. Insert the socket end of the cable into the audio input header (J17) See Figure 7.

4. Note the orientation of the plug (male) end of the cable. The coax wire in the plug goes toward J2 on the preamp board.

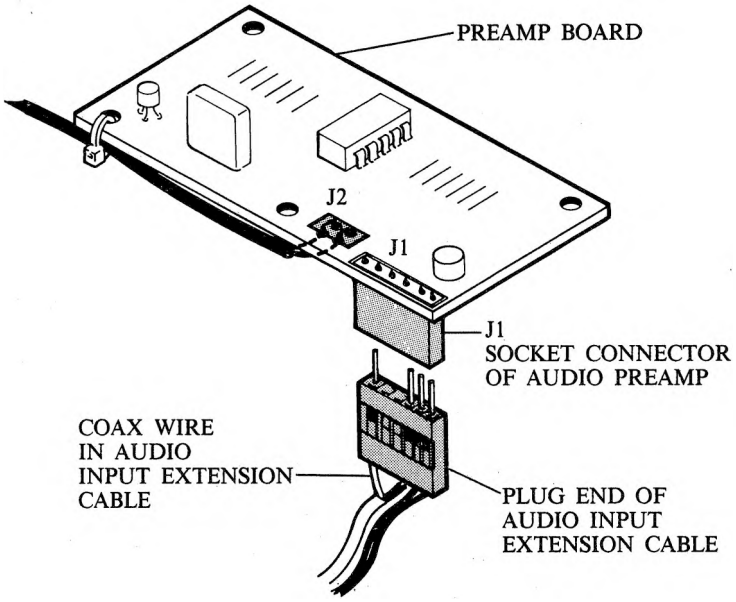
5. Insert the plug end of the cable into socket connector J1 of the preamp board. See Figure 8.

Figure 7: Extension Cable Installation — CPU Board



128K BYTE CPU BOARD (100470)

Figure 8: Extension Cable Installation — Preamp Board



AUDIO PREAMP BOARD INSTALLATION

When used with an internal hard disk, the preamp board is plugged into the underside of the disk drive assembly. Refer to Figure 9 and the steps below to install the preamp board.

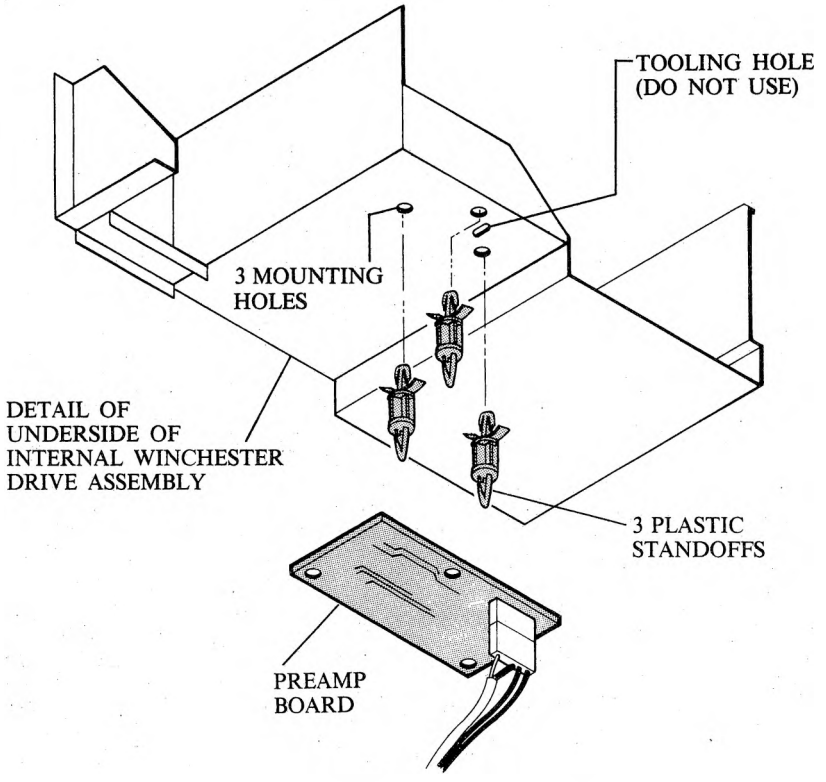
1. Plug the pointed ends of the three plastic standoffs into the holes on the component side of the preamp board.

2. Plug the opposite ends of the standoffs through the holes under the disk drive assembly. The component side of the preamp board should be toward the drive chassis.

3. Reinstall the disk drive assembly onto the mainframe.

4. The metal standoff, kepnut, and screw are not used in this configuration and may be discarded.

Figure 9: Preamp Installation — Internal Hard Disk



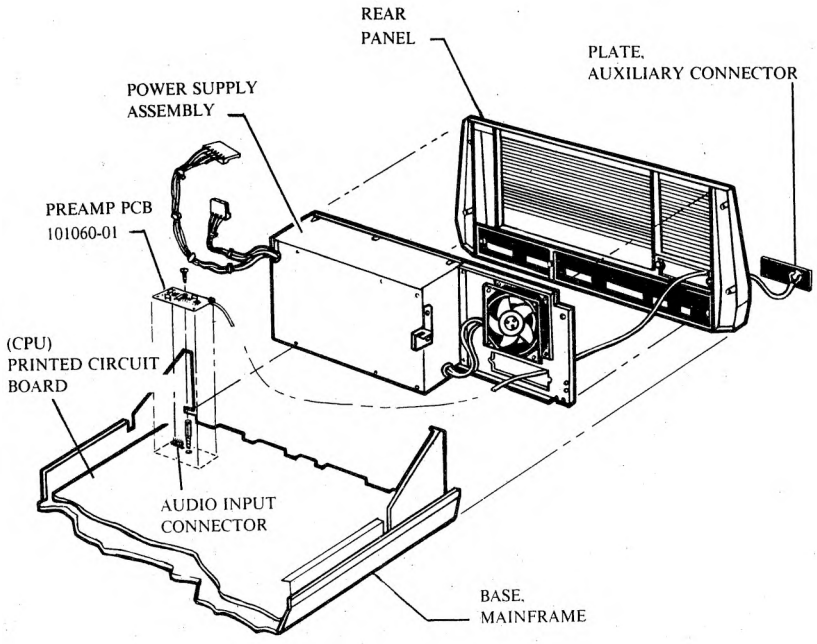
5

AUDIO INPUT JACK

Figure 10 shows the audio input jack installed on the 128K CPU board. To install the audio input jack (the auxiliary connector):

1. Remove and discard the blank plate on the rear panel.
2. Install the mounting plate assembly (supplied) on the rear panel.
3. Route the audio preamp cable through the power supply plate access opening and connect it to the audio input jack.
4. Reinstall the power supply.
5. Reinstall the top cover under the front cover.
6. Reinstall the rear panel cover (4 screws).
7. Carefully replace and connect the CRT and keyboard.
8. Plug in the power cord.

Figure 10: Audio Preamp Board and Input Jack Installation



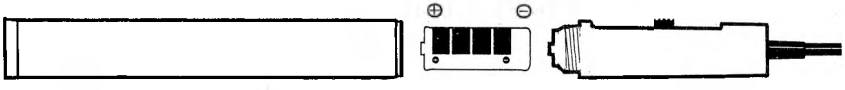
MICROPHONE

To prepare the microphone for use:

1. Unscrew the microphone cartridge.
2. Insert the positive end of the N-type battery (supplied with the microphone) into the front of the microphone cartridge (Figure 11.)
3. Screw the microphone cartridge back on.
4. Place the microphone in the stand and adjust the angle and height.
5. Set the microphone switch to ON.

NOTE: Do not turn the microphone switch on and off during a recording session. The switch does not turn sound on and off. The switch disconnects power from the condenser microphone element. (You may hear a "pop" when you turn the microphone on and off.) Leave the switch off when not in use to conserve battery power.

Figure 11: Microphone Battery Installation



CARTRIDGE BATTERY CASE

6.1 MICROPHONE OPERATION

Proper microphone placement is very important for good recording pickup. If you place the microphone too close to the sound source, it may pick up unwanted noises. Use the foam wind/pop screen when recording close to the sound source. This reduces wind sounds and voice pops.

Remove the battery when the microphone is not to be used for an extended period of time. Normal battery life is about one year. Replace the battery when microphone sensitivity decreases. Use only an alkaline N-type battery.

Do not expose the microphone to high heat or humidity.

6.2 MICROPHONE SPECIFICATIONS

The microphone package contains an N-type battery, foam-type wind/pop screen-filter, and a desk stand.

Electrical specifications follow:

► Type:	Electret Condenser Microphone
► Frequency response:	20-13000 Hertz
► Output impedance:	600 Ohms ± 30%
► Sensitivity:	–71 db ± 4 db (0 db = 1 V/u bar)
► Directional:	Omni-directional
► Power supply:	N-type alkaline battery
► Minimum operating voltage:	1.0 Vdc