ED 409 690	EC 305 737
TITLE	A Talking Computers System for Persons with Vision and Speech Handicaps. Final Report.
INSTITUTION	Visek & Maggs, Urbana, IL.
SPONS AGENCY	Department of Education, Washington, DC.
PUB DATE	84
NOTE	239p.
CONTRACT	300-83-0271
PUB TYPE	Guides - Non-Classroom (055) -- Reports - Descriptive (141)
EDRS PRICE	MF01/PC10 Plus Postage.
DESCRIPTORS	*Artificial Speech; Assistive Devices (for Disabled); *Blindness; *Computer Interfaces; Computer Software; *Computer Software Development; *Electromechanical Aids; *Speech Impairments
IDENTIFIERS	Apple (Computer); Computer Manuals; IBM Personal Computer

ABSTRACT

This final report contains a detailed description of six software systems designed to assist individuals with blindness and/or speech disorders in using inexpensive, off-the-shelf computers rather than expensive custom-made devices. The developed software is not written in the native machine language of any particular brand of computer, but in the universal language "C," which will run on all present and future brands of computers. The report provides user manuals describing the software for individuals with blindness, "Screen-Voice," adapted to three types of computer in use in 1984, the Apple IIe (the most popular computer in schools); IBM compatible personal computers (the most popular office computer, available in tabletop and portable versions); and the TRS-80 Model 100 (a portable computer). The report includes user manuals for "Talkhelper," a program designed to allow persons with speech disabilities to use a personal computer with a speech synthesizer as a substitute voice. Manuals are also provided for Apple IIe, IBM-compatible computers, and for the TRS-80 Model 100 portable computer. Finally, the source code of the most elaborate version of the software, that of the IBM-compatible personal computer, is reproduced in full. (CR)

A TALKING COMPUTERS SYSTEM FOR PERSONS WITH VISION AND SPEECH HANDICAPS

FINAL REPORT

submitted under:

United States Department
of Education Contract
No. 300-83-0271

submitted by:

Visek & Maggs
608 W. Pennsylvania Ave.
Urbana, IL 61801
(217) 367-5027

TABLE OF CONTENTS

INTRODUCTION

This is the final substantive report submitted by Visek & Maggs, contractor under United States Department of Education Contract No. 300-83-0271, "A Talking Computer System for Persons with Vision and Speech Contracts." This report contains a detailed description of the six software/hardware systems designed and tested by Visek & Maggs in accordance witht the terms of this contract.

The basic philosophy of the contract proposal and the Design Report developed under the contract was that, with appropriate software, persons with vision and speech handicaps could use inexpensive off-the-shelf computers rather than expensive custom made devices. This philosophy was based on the tremendous success of previous work by the two general partners in the contractor organization in designing the speech and blind user software for the Echo II Synthesizer for the Apple computer, software that was instrumental in making artificial speech available on a mass-produced basis at a highly affordable price.

However, despite its wide acceptance, this early software had two serious drawbacks. First it required substantial additional programming to be of any use to speech-handicapped persons. Second, it was written in machine language for the obsolescent Apple II computer and could not easily be transferred to other machines.

To meet the requirements of the speech-handicapped, contractors developed a highly sophisticated authoring and customization system, allowing the speech-handicapped person or an assistant to customize the software to meet the particular combination of handicaps and needs of the particular individual.

The software developed previously is not suitable for the IBM-compatible personal computers that have become the standard in many business offices, nor for the most popular brand of portable computer, the TRS-80 Model 100, nor for the many new types of computers sure to appear in the next few years in this area of rapidly changing technology. The software developed under the present contract is written not in the native machine language of the Apple II or any other particular brand of computer, but in the universal language "C", which will run on all present and future brands of computers. Thus this software can still bring benefits to persons with vision and speech handicaps long after the present generation of computers is obsolete.

The material which follows describes in detail the software for blind users, Screen-Voice, as adapted to three types of computer: the Apple IIe (the most popular computer in schools), IBM-compatible personal computers (the most popular office computers, available in tabletop and portable versions), and the TRS-80 Model 100 (the most possible portable computer). Finally the source code of the most elaborate versions of the software, those for the IBM-compatible personal computers is reproduced in full.

SCREEN-VOICE

Software to Adapt the IBM Personal Computer for Blind Users

USER'S MANUAL

This manual and the accompanying software were developed under United States Department of Education Contract No. 300-83-0271.

Visek & Maggs
608 W. Pennsylvania Ave.
Urbana, IL 61801

TABLE OF CONTENTS

PREFACE

This user's manual tells how to set the switches on a speech synthesizer and how to install the Screen-Voice software for the IBM Personal Computer, and provides instructions for blind users of the software. This manual assumes the IBM Personal Computer is set up and ready to operate, and that the speech synthesizer has been connected to the computer in accordance with the instructions supplied with the synthesizer.

Purposes of The Manual

This manual has five purposes:

1. To tell you how to set up your computer and speech synthesizer for use with the Screen-Voice software.
2. To guide you in getting the Screen-Voice software working.
3. To explain the options for reading from the computer screen.
4. To explain the options for controlling the types of output.
5. To help you set up special pronunciation tables.

INTRODUCTION

The Screen-Voice software for the IBM Personal Computer is a program that provides two essential services for the blind user. First, it sends to the speech synthesizer a copy of everything that appears on the screen, to be spoken as it appears on the screen. Second, it allows the user to select parts of the contents of the screen for review, replaying individual lines, words, or letters.

SYSTEM REQUIREMENTS

This software is designed for use with the IBM Personal Computer and any of the following brands of speech synthesizer: Echo GP, Intextalker, Microvox, or Votrax Personal Speech System. The software may work with some IBM-compatible personal computers and can be adapted for other synthesizers as explained in the final section of this manual.

USING SOFTWARE DEVELOPED FOR SIGHTED USERS OF IBM PERSONAL COMPUTERS

Many, but by no means all, software programs developed for sighted users of the IBM Personal Computer can be used with Screen-Voice. For instance, blind users can write and run programs in the BASIC programming language supplied with the IBM Disk Operating System. Many commercially available programs will also work. Some, however, will not work.

There are three possible types of problems in using commercially available programs. First, some commercial programs are supplied on diskettes which incorporate special features to prevent illegal copying. Some of these programs will incorrectly identify an attempt to use Screen-Voice as an attempt to make an illegal copy and may refuse to cooperate with Screen-Voice. Second, some programs write information directly to the computer screen, bypassing the standard IBM screen output software built into the computer and its operating system. This technique gives faster performance for these programs, but makes it impossible for Screen-Voice to capture abd speak the programs' output. Third, some programs, because they depend heavily on graphics or screen location of text, are difficult to describe verbally. Thus they are unsuitable for blind users.

Screen-Voice will work well with software developed especially for blind users. This software, for purposes such as word processing and data base management, makes use of the power of the computer to search for the words that blind users want to change or the information they want to get.

HOW MUCH OF THIS MANUAL DO I NEED TO READ?

The most essential part of the manual deals with screen review and output control features of Screen-Voice. You must thoroughly familiarize yourself with these features. However, don't worry about forgetting some of them -- you can have the computer help you while you are running a program, without interfering with the operation of the program.

If you are planning to install your synthesizer and Screen-Voice yourself, you must study the sections of this manual on installation before you try to use Screen-Voice. If someone else does the installation, they must read those sections.

The custom pronunciation table feature is a part of the manual that can be left until later. You may find that the brand of synthesizer you have selected mispronounces some words -- your last name, for instance. The custom pronunciation table will let you enter the pronunciation you want for these words.

Unless you intend to use a brand of synthesizer other than the Echo II, Intextalker, Microvox, or Votrax Personal Speech System, you can completely ignore the section of the manual on adapting Screen-Voice for use with other synthesizers.

SETTING THE SPEECH SYNTHESIZER SWITCHES

The speech synthesizer must be attached to the computer in accordance with the instructions provided by the synthesizer manufacturer in the manual that comes with the synthesizer. If the manufacturer provides a demonstration diskette, it must be used to test the speech synthesizer once it is installed. A blind user will find it very convenient to plug the computer, speech synthesizer, and the video monitor (if he or she has one) into a singl power strip so that they may all be turned on with one switch.

Most brands of speech synthesizer have optional switch settings. These switches must be set as follows:

Echo GP:

The switches are located on the bottom of the synthesizer near the middle of the back. (The back is the side where the wires come out.) There is a square hole in the back of the synthesizer, and in this hole are four tiny switches. The part of each switch toward the back of the synthesizer must be pushed inward. This will turn these switches on and will set the synthesizer for 9600 baud, which means high-speed communication of information from the computer to the synthesizer.

Intextalker and Microvox:

The hardware of these two synthesizers is identical. The switches are located inside the synthesizer. Make sure the synthesizer is unplugged from the electric power line before attempting to change the switches! Get a Phillips screwdriver.

Turn the synthesizer upside down find the heads of two screws. One is located in the middle of the right side of the bottom of the synthesizer, just in from the edge; the other screw is located in the middle of the left side of the bottom of the synthesizer, just in from the edge. Remove the two screws, turn the synthesizer right side up and remove the top cover.

Now place the synthesizer with the side with the volume control knob toward you. There are two sets of eight switches that will have to be set. The first set is located halfway between the right and left edges of the circuit board inside the synthesizer, about one inch back from the front of the board. Switch number 4, which is the fourth switch from the rear of the board on this set of switches, must be pushed to the right. The other switches must be pushed to the left. This will set the communications rate to 1200 baud or medium speed.

The second set of switches is located very near the right hand edge of the board, about three inches back from the front of the board. On this set of switches, switches 1, 2, and 8 (the first, second, and eighth switches from the rear of the board) must be set to the right. The others must be set to the left. This will provide for what is called "hardware handshaking" -- sending electronic signals to coordinate the activity of the synthesizer and the computer. If you have the Intex upgrade for improved pronunciation, switch settings may be different, as described in the instructions that come with this upgrade.

Votrax Personal Speech System:

Turn the synthesizer so the volume control is away from you. On the part of the synthesizer that is now toward you, about three inches leftward from the right side is an oblong hole, about one-third of an inch high and one inch wide. A set of eight switches is recessed about one-half an inch behind this hole. The switches must all be pushed down, except switch 6 (the sixth from the left), which must be pushed up. In case you are interested, the technical terms for these switch settings are: 9600 baud, RTS serial port communication, 7 bit word with ignored parity bit, power-up message spoken, serial port used as primary input port, and self-test off.

GETTING STARTED

We will first list and then explain in detail the steps in getting started with your Screen-Voice software. Experienced IBM users may read the headings to determine which sections they already know. The installation procedure described below will inform Screen-Voice of the brand of synthesizer you are using, so it can customize itself for this brand. It will also copy some copyrighted IBM software from the IBM DOS diskette to the Screen-Voice diskette. (The copyright laws do not allow distribution of this software on the Screen-Voice diskette, but they do allow you to make a copy of it for your personal use if you own a genuine copy of IBM DOS.) Finally, the manual will explain how to make working copies of your Screen-Voice diskette so that the original can be kept in a safe place as a backup copy.

PREPARATIONS

Make sure your synthesizer is plugged in, turned on, and connected to the computer, and that the computer is plugged in but turned off.

STARTING UP THE COMPUTER

Get your DOS (Disk Operating System) disk. You must have IBM DOS 2.0 or a higher number -- DOS 1.1 for instance will not work. If you do not have the right version of DOS, show these instructions to your IBM dealer, who will be delighted to sell you the appropriate version of DOS.

Now we will explain for new IBM users how to put this disk in the computer and how to start the computer running. In the front of the box that encloses the main computer unit there will be one or two disk drives. The disk drive or drives will be located in a recessed area about 12 inches long and three inches high. The disk drive you want is located in the left half of this recessed area. It is called Drive A by IBM. If you feel the front of this drive, you will feel two thin horizontal slots, one on the right and one on the left. Between them is the door of the diskette drive. If the door is closed there will still be a recess below it. If the door is open, the recess will extend above and below the slots and you will be able to feel the open door near the top of the middle of the disk drive. Practice opening and closing the door and then leave it open.

Now pick up the DOS diskette, which will be in a paper cover. Handle the diskette only by the edges, so as to avoid damaging its sensitive recording surfaces. Remove the diskette from the paper cover and feel around the edges. You will find two very small notches on one side and one larger notch on another side. Now hold this diskette horizontally, so that the two small notches are on the side toward the computer and the larger notch is to the left. Making sure the door is open, insert the diskette in the slot in Drive A and push it all the way in. Do not force it. Now close the door. Do not force the door closed if something is blocking it or you could damage the diskette.

Now, turn on the computer. The diskette unit will whir for a while and then fall silent. At this point the computer will not be talking, but unless it has an automatic clock, it will be writing a message on the screen to ask the time. To bypass the request for the time, do the following. Without pressing any keys, put your finger at the right top corner key on the keyboard. Still without pressing any keys, move your finger two keys to the left of the right top corner key. Now move your finger to the key below this. Your finger will be on the ENTER key. Press it twice.

INSTALLING SCREENREADER

The following procedures will install Screen-Voice and tell it what brand of synthesizer you are using. Note: If you change brands of synthesizer you will need to repeat this procedure. During the process parts of DOS will be copied automatically to your Screen-Voice diskette so that you can later use that diskette to start up your computer.

First we will give a summary for experienced IBM users; then we will give a much more detailed explanation for new users.

Summary for Experienced Users

Make sure that your synthesizer has been turned on, so that you will hear any message Screen-Voice may send to it. If you have two floppy diskette drives, leave your DOS disk in drive A and insert the Screen-Voice diskette into drive B. Type "b:" followed by the first letter of the name of your synthesizer. For Intextalker, type "i"; for Microvox, type "m"; for Echo GP, type "e"; for Votrax Personal Speech System, type "v". After a moment, the synthesizer should say "Screen-Voice by Visek and Maggs" and "configured for" followed by the brand name of your synthesizer.

If you have only one floppy diskette drive (with or without a hard disk drive), leave the DOS diskette inserted and type "b:" followed by the letter corresponding to your synthesizer as given above. Since the installation program needs to copy some DOS files onto your Screen-Voice diskette, you will need to switch diskettes and press the ENTER key each time you year the disk drive stop. After several switches, the synthesizer should say "Screen-Voice by Visek and Maggs" and "configured for" followed by the name of your synthesizer.

Instructions for New Users

First make sure that your synthesizer is turned on, so that you will hear any message Screen-Voice sends to it. The procedure beyond this point depends upon what type of disk drives your computer has. The following instructions will cover in turn computers with two floppy diskette drives and computers with one floppy diskette (with or without a hard disk drive).

If you have one floppy diskette drive and one hard disk drive, skip to the section below called "Instructions for New Users with Only One Floppy Diskette Drive." If you have two floppy diskette drives, continue reading.

Instructions for New Users with Two Floppy Diskette Drives

The next step is to place the Screen-Voice diskette in the right hand drive, drive B. Now what you do depends on what brand of synthesizer you have:

If you have an Intextalker, type:

```
b:i
```

and then press the ENTER key.

If you have a Microvox, type:

```
b:m
```

and then press the ENTER key.

If you have an Echo GP, type:

```
b:e
```

and then press the ENTER key.

If you have a Votrax personal speech system, type:

```
b:v
```

and then press the ENTER key.

By typing "b:" you have told the computer to look on the diskette in the right hand drive, drive B, for the file specified by the letter corresponding to your synthesizer. The diskette drives will whir and when they stop whirring the synthesizer should say "Screen-Voice by Visek and Maggs," and "configured for", followed by the name of your synthesizer.

Instructions for Users with Only One Floppy Diskette Drive

If you have only one floppy diskette drive, with or without a hard disk drive, the procedures are more complicated. Remove the DOS diskette and insert the Screen-Voice diskette. Type the following depending upon your brand of synthesizer:

If you have an Intextalker, type:

b:i

and then press the ENTER key.

If you have a Microvox, type:

b:m

and then press the ENTER key.

If you have an Echo GP, type:

b:e

and then press the ENTER key.

If you have a Votrax personal speech system, type:

b:v

and then press the ENTER key.

The diskette drive will whir. When it stops, remove the DOS diskette, insert the Screen-Voice diskette and press the ENTER key. The drive will whir again. When it stops, remove the Screen-Voice diskette and insert the DOS diskette. Press ENTER. The diskette drive will again whir and stop. Switch diskettes and then press ENTER each time the drive stops, until the synthesizer says "Screen-Voice by Visek and Maggs," and "configured for" followed by the name of your synthesizer.

MAKING WORKING COPIES OF YOUR SCREENREADER DISKETTE

If you are an experienced IBM Personal Computer user, make some working copies of the Screen-Voice diskette using the DISKCOPY command. (The Screen-Voice diskette is not copy-protected.) Experienced users may skip the following detailed instructions on making extra copies.

Now we will explain how to make working copies of your Screen-Voice diskette. It is important to make several working copies so that you can put the original diskette and one working copy in a safe place as a backup in case the copy you are using is accidently damaged or erased or physically wears out.

Remove the Screen-Voice diskette and cover the large notch with one of the special write-protect stickers provided with each box of blank diskettes. These stickers are about one inch by one-quarter inch. (Note: They are not the bigger stickers provided for use as labels.) When the notch is covered, the diskette drive cannot alter the contents of the diskette, so by covering the notch, you can protect your diskette from accidental modification. Now get a blank diskette to use in making a copy of Screen-Voice. The copy procedure depends on the number of floppy disk drives you have.

Making Working Copies if You Have Two Diskette Drives

If you have two floppy diskette drives, insert the DOS diskette in drive A (the left hand drive) and a blank diskette in drive B (the right hand drive). Now type the following and then press the ENTER key once:

```
diskcopy a: b:
```

When Drive A stops whirring, remove the DOS diskette from drive A and place the Screen-Voice diskette in drive A. Press the ENTER key. The drives will whir for quite a while as they copy all the information from the Screen-Voice diskette to the new diskette you have in Drive B. When the whirring stops, the copying is done.

Making Working Copies if You Have One Floppy Diskette Drive

Now we will explain how to make working copies of your Screen-Voice diskette if you have only one floppy diskette drive. Insert the DOS diskette in the floppy diskette drive. Then type the following and then press the ENTER key once:

```
diskcopy a: b:
```

The Screen-Voice diskette is called the "source" diskette and the blank diskette is called the "target" diskette. The synthesizer will tell you when to insert a diskette or press a key. It will notify you when the copy is done and ask if you want to make another copy. Press the "Y" or "N" keys, depending on what you want to do.

USING SCREENREADER

Once you have configured your Screen-Voice diskette and made working copies of it, using Screen-Voice is very simple. All you will need to do is to put a working copy in drive A before turning on the computer. When the computer is turned on, the Screen-Voice software appropriate for your synthesizer will be loaded automatically into your computer and the synthesizer will say "Screen-Voice ready." You may then replace the working copy of the Screen-Voice diskette with one of your program diskettes.

If you turn the computer off, you will have to put a copy of the Screen-Voice diskette in drive A before you turn it on again.

OUTPUT CONTROL FEATURES

When your computer is turned on and Screen-Voice is activated, it automatically selects a number of standard or "default" features for reading what is sent to the screen. However, for special purposes you may want to choose other features. This chapter explains the standard features and your options for varying them.

The standard features for Screen-Voice attempt to imitate what a sighted person would say if asked to read what was written on the screen. The special features allow various types of more detailed reading, for tasks such as proofreading. They also allow less detailed reading to enable rapid skimming of the text on the screen.

How to Select an Output Control Feature

To select an output control feature, hold down the Control key (the key immediately to the left of the A key), and while holding it down press the E key. This combination is called Control-E. The next step is to release these two keys and press the command key given below for the feature desired. In some cases it will also be necessary to type a number. For these commands, upper case and lower case letters have the same specified effect.

An attempt has been made to choose command keys so that the letters on the keys are related to the first letters of the names of the commands, but this has not always been possible. Single-key commands rather than whole-word commands are used because, although they are harder to remember, they are much faster to use after some practice. One can issue any number of Screen-Voice commands in a row and commands may be issued at any time the computer is accepting key presses.

GENERAL PRONUNCIATION FEATURES

Most speech synthesizers allow commands to be sent by the computer to control synthesizer pitch, volume, speed, and intonation. Details vary from synthesizer to synthesizer. Thus the commands described here for varying synthesizer voice characteristics may have different effects with different synthesizers. If a particular brand of synthesizer lacks certain pronunciation features, the command may have no effect, but will do no harm.

Intonation

Most synthesizers allow the user to select either an imitation of human speech intonation or robot-like flat speech. Some people prefer the flat speech because no synthesizer selling for less than $4000 offers good intonation. Others find flat speech boring and prefer inaccurate intonation. Screen-Voice sets the synthesizer initially to provide intoned speech. To switch to flat speech type Control-E and then I. To switch back to intoned speech, type Control-E and I again.

Remember: I is for Intonation on or off.

Speech Rate

Most synthesizers allow the user to control the rate of speech. Blind persons who are experienced with speech synthesizers usually like to run them as fast as possible, so as to increase their effective reading speed. However, even experienced users sometimes need to slow down the synthesizer for difficult passages. To set the synthesizer speech rate, type Control-E, then R, and then a letter from A to Z. The letter A stands for the slowest rate, the letter Z for the fastest rate and other letters for rates in between. If your synthesizer has only one rate, this command will have no effect. If your synthesizer has only two rates, letters in the first half of the alphabet will select the slower rate, and letters in the second half of the alphabet will select the faster rate. The initial rate is programmed into the synthesizer by its manufacturer. For some synthesizers, such as the Intextalker or Microvox, changing the rate may also change the pitch.

Remember: R is for Rate;

the following letter selects the rate.

Volume

Some synthesizers allow users to control speech volume through the computer. Others offer a convenient volume control knob on the synthesizer. Still others have both options. To set the volume through the computer, type Control-E, then V, and then a letter from A to Z. If the synthesizer has computer-controllable volume, the letter A will produce the softest volume, the letter Z the loudest, and other letters will produce intermediate volume levels. If the synthesizer has less than 26 different levels of controllable volume, letters close together in the alphabet may not produce different volumes. On most synthesizers, use of the volume control knob on the synthesizer itself is a better way to change volume. The initial volume is programmed into the synthesizer by its manufacturer.

Remember: V is for Volume;
the following letter selects the volume.

Pitch

Some synthesizers allow computer control of speech pitch. It may lessen fatigue to vary the pitch from time to time so as not to listen to the same voice for hours on end. To set the pitch control through the computer type Control-E, then P, and then a letter from A to Z. If the synthesizer has computer-controllable pitch, the letter A will produce the softest pitch, the letter Z the loudest, and other letters will produce intermediate pitch levels. If the synthesizer has less than 26 different levels of controllable pitch, letters close together in the alphabet may not produce different pitch levels. The initial pitch is programmed into the synthesizer by the synthesizer manufacturer. For some synthesizers, such as the Intextalker or Microvox, pitch can also be varied by changing the rate.

Remember: P is for Pitch;
the following letter selects the pitch.

Pauses

Scientific research shows that it is easier to understand speech synthesizers if they are programmed to pause briefly between words. However, the pauses slightly slow down the reading speed. When Screen-Voice starts running, it does not add pauses between words; however there is an option to add pausing. Some synthesizers, such as the Votrax Personal Speech System, automatically provide a short pause between words. With these synthesizers, if Screen-Voice is set for pausing, there will be an extra long pause between words. To switch between not pausing and pausing, press Control-E, and then letter E. To switch back, use the same command.

Remember: E is for Enunciation on or off.

Custom Pronunciation

Sometimes synthesizers mispronounce words or fail to handle abbreviations properly. Sometimes users get tired of hearing long words pronounced in full over and over again. The custom pronunciation table feature, discussed in more detail in a later chapter, allows the user to specify the desired pronunciation for each word in a list. When Screen-Voice is activated, the substitutions provided in this table are automatically put into effect whenever screen-Voice is in normal speech mode. (When the synthesizer is spelling letter by letter or with the "Alpha, Bravo, Charlie" alphabet, custom pronunciation is temporarily suspended, because it is assumed that the user is using these modes to find out in detail what is actually on the screen.) To switch between custom and regular pronunciation, type Control-E followed by C. To switch back, use the same command.

Remember: C is for Custom pronunciation on or off.

FEATURES FOR MORE DETAILED READING

The usual way of reading text does not bring out all the detail necessary for proofreading. The proofreading features of Screen-Voice allow the user to have just the amount of extra detail he or she wants for the particular task at hand.

Spelling or Whole Words

When Screen-Voice is activated it reads by whole words. For proofreading it may be necessary to have individual words spelled out, since two words may be spelled differently but pronounced the same (for instance "t h e i r" and "t h e r e"). A word could be spelled wrong but pronounced correctly (for instance "t h a i r" will be pronounced "there") or spelled correctly but pronounced wrong due to a shortcoming of the speech synthesizer. To have words spelled out, type Control-E, and then L. To return to normal whole word pronunciation press Control-E and then N.

Some people have trouble understanding a synthesizer when it is spelling letter by letter. To have spelling done not with letters, but with the "Alpha, Bravo, Charlie" alphabet, press Control-E and then type B.

Remember: L for Letter by letter spelling;
B is for Alpha, Bravo, Charlie spelling;
N is for Normal (whole word) pronunciation.

Capital Letters

Proofreading may require distinguishing upper case from lower case letters. To have upper case letters indicated, press Control-E and then U. The computer will then say "cap" before each upper case letter. To go back to regular reading, press Control-E and then U again.

Remember: U is for Upper case on or off.

Punctuation

In normal reading, special symbols such as "$" and "+" are usually read out, but common punctuation marks such as period and comma are not pronounced. When Screen-Voice starts, it is in normal reading mode. All special symbols and all punctuation on the screen are pronounced except the following: dash, period, comma, semicolon, exclamation point, question mark, star, double quotes, apostrophe, tilde, grave accent, vertical line, backslash, left parenthesis, right parenthesis, left bracket, right bracket, left brace, and right brace. When reading numbers, a decimal point will be pronounced as point. If a combination of digits, dollar signs, commas, and periods is encountered that is not a well-formed number or monetary amount, all characters will be pronounced to alert the user to the unusual situation.

This option is called "Some" punctuation. If you have switched to another punctuation option, you can return to "Some" punctuation by typing Control-E and then S.

When proofreading, it is usual to pronounce all punctuation but not to indicate spaces. This option is called "Most" punctuation. It is available by pressing Control-E and then M.

Finally, for some purposes it is necessary to know exactly what is on the screen, including spaces. This option, called "All" punctuation, is available by pressing Control-E and then A.

None of these modes provides an indication of video cursor movement about the screen. Positions of this cursor may be checked at any time the computer is accepting input, by using the video cursor command described below.

Remember: S is for Some punctuation;
M is for Most punctuation;
A is for All punctuation.

Numbers

The reading of numbers also differs in ordinary speech and in proofreading. In normal speech, for instance, a three followed by a four followed by a five is pronounced "three hundred forty five." When proofreading, the same number is pronounced digit by digit, as "three four five." When Screen-Voice starts, numbers are pronounced in normal speech mode, except for combinations of digits, commas, dollars signs and periods that constitute neither well-formed numbers nor proper monetary amounts. These combinations, such as $3,4.5 are spoken character-by-character. To switch between normal speech mode and digit-by-digit mode type Control-E and then D. To switch back, use the same command.

Remember: D is for Digits on or off.

FEATURES FOR LESS DETAILED READING

The following options allow text to be read faster by skipping portions of it. (Remember also that setting the synthesizer to a fast speech rate, as explained earlier, will speed up reading without loss of any text.)

Skimming

Often a user wishes to skim through text to see if it is of any interest. A skimming option is available that just reads the long words in the text, those with seven or more letters. Numbers and strings of punctuation and other symbols are skipped regardless of length. To just read these big words, type Control-E and then J. To return to ordinary reading, type Control-E and J again.

Remember: J is for Just reading long words.

Ignore Top Lines

Normally Screen-Voice reads everything that appears on the screen. However, some programs continually write status information on top line of the screen. This information, such as the number of the column currently being written, is of little value to the blind user, and is highly annoying when it interrupts spoken text. To set the computer to ignore lines at the top of the screen, type Control-E, then T, then a letter from A to Y. The letter indicates the first line that will be pronounced. If the letter is A, the whole screen will be pronounced. If the letter is B, pronunciation will start at the second line on the screen, if it is C, pronunciation will start at the third line on the screen. The letter Y turns off speech on all but the bottom line. To turn all the lines back on again, type Control-E , then T, then A.

Remember: T is for Top line of screen;
the following letter specifies the first line to read.

Columnar Reading

A feature that is useful both for skimming and for other purposes is columnar reading. One use of this feature is to read a column of numbers in a table. Another is to skim text, for instance, by reading only the first twenty characters on each line. There are eighty columns on the screen. The leftmost column is considered to be column 00 and the rightmost column is considered to be column 79. To set which columns are to be read, press Control-E, then type the letter O. Next, type in two two-digit numbers, each between 00 and 79. If you make a mistake, the computer will tell you and give you a chance to correct it. To return to normal reading of the whole screen press Control-E, type the letter O, and press the ENTER key.

Remember: O is for Only certain columns;
the next 2 digits give the left column;
the last 2 digits give the right column.

Count Repeating Punctuation

A feature of some programs that is useful for sighted persons but annoying for blind users is repeating punctuation. One word processing program, for instance, indicates page breaks by putting a row of seventy-nine dashes on the screen. Few people want to hear the synthesizer say "dash" seventy-nine times. This option deals with this problem.

When Screen-Voice is activated it is in "Some" punctuation mode (see explanation above), where some punctuation characters will be pronounced and some will be ignored. In addition, Screen-Voice initially does not count repeating punctuation--each instance of the punctuation to be spoken will be individually pronounced, no matter how often it repeats.

Pressing Control-E and then G switches to a mode where repeating punctuation signs are counted and the count is spoken. The user will only hear the count for those characters that the current punctuation mode (All, Most or Some) is supposed to pronounce. All other characters will continue to be ignored. Using this feature in Most or All modes, seventy-nine dashes in a row would result in the synthesizer saying the word "dash", followed by the count "seventy-nine", followed by the word "times". To return to saying each instance of the repeating punctuation, type Control-E followed by the letter G again.

Remember: G is for Get rid of repeating punctuation (on or off).

Stopping Speech

Often the computer will display a screenful of text, but as soon as Screen-Voice starts saying it, the user will realize that he does not want to hear it. The zap feature allows the user to have the synthesizer stop speaking the text it is preparing to say. The effectiveness of the zap feature depends upon the brand of synthesizer used -- some cannot be stopped once text has been sent to them. The zap command can be entered only if the running program is accepting key input. Sometimes, even when a program will not take regular key input, it can be stopped by pressing Control-C. Then the zap command can be given. To stop speech, type Control-E, then Z. In help mode, the computer will send the phrase "clear buffer" to the synthesizer immediately after the "zap" command; however, this phrase may not be spoken since some synthesizer remain inactive for a few moments after receiving a "zap" command.

Remember: Z is for zap the speech buffer.

Input Modes

Whenever typing a key causes the corresponding letter, number, or symbol to appear on the screen, Screen-Voice will immediately read the letter, number, or symbol from the screen, without waiting for the end of the line. This is called key echoing. Good typists often feel that their time is being wasted listening to the synthesizer echo each key. To turn off key echo, type Control-E and then K. To turn it on again, type Control-E and then K again.

Remember: K is for Key echo on or off.

HELP FEATURES

Reporting Command Key Functions

There are so many options that it may be hard to remember which options have been set or which key affects which options. Typing Control-E and then H, will put the computer in a mode where the function of each key is spoken if the key is pressed. If the same command key switches a mode on and off, the new status will be spoken. To turn off the help messages, type Control-E and then H again.

Remember: H is for Help on or off.

Position of Video Cursor

Most programs put a cursor or small flashing symbol at the point on the screen where they expect to receive text. To hear the location of this point, press Control-E and then Y. The coordinates will be spoken as first a letter corresponding to the line on the screen and then a number from 0 to 79 corresponding to the column position on the screen.

Remember: Y is for video cursor position.

Changing the Command Control Character

If the word processor or other program you are using employs Control-E as a command, you will need to select a different control key by using the following procedure. Instead of pressing a letter key to choose one of the special features, press a different control key (ie. press a different letter key while holding down the control key). You may select any key except C (because Control-C is used by DOS for escaping from programs), H (because Control-H is Backspace), M (because Control-M is ENTER), or the key currently being used to start screen review, which will be discussed below. In selecting a key, avoid control keys that have important functions in software you plan to use. The new control key can then be used to initiate Screen-Voice output control commands.

Remember: Control keys change the command control character.

SCREEN REVIEW FEATURES

Screen-Voice provides a wide variety of options for reviewing what the computer has put on the screen. You may want a particular line of the screen replayed because you have forgotten what it said. You might have had trouble understanding the speech synthesizer and want a few words repeated or spelled out.

The video screen of the IBM Personal Computer has 25 lines of 80 characters each. For purposes of screen review, the twenty-five lines are called lines A through Y -- the first twenty-five letters of the alphabet. The character positions are called position zero through position seventy-nine. Programs running on the IBM personal computer keep track of the location where the next character will be written. Usually they indicate this location to the user by putting a flashing spot of light on the screen. This flashing spot is called the "video cursor." When used for screen review, Screen-Voice keeps track of where it is reading from the screen. The next location to be read from is specified by the "audio cursor."

Starting Screen Review

To start screen review, hold down the Control key and press the L key. If the word processor, data base system, or other program running on the personal computer is accepting key input, its operation will be suspended so you can review the screen. You may now may select various options and continue reviewing until you decide to return to the program you were using. To return to that program, press the Escape key. While you are in screen review mode, you do not need to type Control-L before selecting each option. Just type the appropriate letters.

While you are in screen review mode, all of the output control modes previously selected are still in effect. If you wish to change one of Screen-Voice's output control features from within screen review mode, you may press Control-E (or whatever control key you have previously selected as the command key) and the usual characters needed to change the feature in question. Note that in screen review mode, all key presses will be interpreted as Screen-Voice commands. No keys will be passed to any other program.

When you enter screen review mode, the audio cursor will be set to the same position as the video cursor. If the video cursor is outside the boundaries you specified using the column or top of screen commands, however, the audio cursor will be set to the upper left corner of the screen window. Screen-Voice will notify you of this alternate position. The video cursor will not be moved during screen review mode.

Remember: Control-L initiates screen review;
Escape ends screen review;
all output control options are still available.

Going to a Particular Line

The lines on the screen are designated by the letters from A at the top to Y at the bottom. To go to a particular line, type G and then the letter corresponding to the line.

Remember: G to Go to a line;

followed by a letter to specify the line.

Saying Rest of Line

To say the rest of the line the audio cursor is on, press the ENTER key. When the line has been read, the audio cursor will be moved to the start of the same line. Thus you can hear the line again by pressing the ENTER key.

Remember: ENTER to say the rest of the line.

Moving the Audio Cursor

You can move up, down, right, or left on the screen by pressing the U, D, R, and L keys. If you are in the word-by-word mode, pressing the R key will cause the next word or group of one or more punctuation symbols to the right to be pronounced, and pressing the L key will cause the next word or punctuation group to the left to be pronounced. If you are in the letter-by-letter mode, pressing these keys will cause the next letter or character to be pronounced. It is usually a good idea to switch into "all punctuation pronounced" mode before reviewing to the left or right. Otherwise pronunciation of many punctuation symbols and of spaces will be suppressed. Pressing the U key will move the audio cursor up to the start of the next higher line on the screen. Pressing the D key will move the audio cursor down to the start of the next lower line on the screen. If you hit the right or left end of a line or the top or bottom of the screen, the computer will tell you and no harm will be done. When you hit the right end of a line the audio cursor will be repositioned automatically to the start of the same line.

Remember: R is for Right;
L is for Left;
U is for Up;
D is for Down.

Audio Cursor Position

You may wish to know exactly where the audio cursor is on the screen. (The audio cursor starts in the same place as the video cursor when you enter screen review mode, but you may have moved the audio cursor.) To have the location of the audio cursor spoken press the A key. As usual, the position is given by a letter representing the line and a number representing the column.

Remember: A is for Audio cursor.

Mark a Screen Position and Go Back to it

You may want to mark a position on the screen so that you can go back to it later. Press the M key to mark the current audio cursor position. Press the B key to go back to it. The marked position will stay the same unless you specifically change it. Thus you can mark a position, leave screen review mode and do other things. When you come back to screen review mode the marker will be at the same place.

Remember: M is for Mark;

B is for Back.

Finding a String of Characters on the Screen

To find a string of characters on the screen, type F. Next type the string you want to find and press the ENTER key. (Note: If the string is over 19 characters long, the twentieth and following characters will be ignored.

At this point, the synthesizer will pronounce the string you are looking for. Then it will start at the top of the screen and look for the string you want to find. If you have used the column or top of screen commands, the search will be limited to the screen window you have specified. If Screen-Voice finds the string, the synthesizer will say the line of text containing the string, and then set the audio cursor to the start of the string it has found. If it does not find the string, the synthesizer will say "not found" and the position of the audio cursor will be unchanged. If you find an instance of the string and want to look for another, press the space bar. When you are done looking, press any other command key.

Remember: F to Find a string;

followed by the string and ENTER;

space bar to try to find it again.

Changing the Screen Review Control Character

If the word processor or other program you are using employs Control-L as one of its commands, you will need to select another control key by using the following procedure. Instead of pressing a letter key to choose one of the screen review features, press a different control key (ie. press a different letter key while holding down the control key). You may select any key except C, H, M, or the key currently being used to start output control commands.

Remember: Control keys change the screen review character.

End Screen Review Mode

Finally, you will want to escape from review mode to get back to the program you were running. Press the Escape key to end screen review.

Remember: Escape to end screen review.

BUILDING A CUSTOM PRONUNCIATION TABLE

The synthesizer may make some annoying pronunciation errors or there may be words or phrases that the user wants pronounced differently for some reason. The custom pronunciation table allows the user to use his own pronunciation for any word that appears on the screen. We have already given directions for activating and deactivating this table. We will now describe its structure so that the user may modify it.

The custom pronunciation table must always be contained in an ASCII file called "table". It consists of a series of lines, one for each word which will have a custom pronunciation. Each line must have the following elements in the order listed:

(1) The word to be specially pronounced;

(2) One space;

(3) The string to be substituted (zero to forty-eight characters)

As an example, many synthesizers pronounce the word "typewriter" as "type uh writer". The following line in the custom pronunciation table will ensure correct pronunciation with most synthesizers:

```
typewriter type writer
```

The treatment of upper case letters when custom pronunciation is in effect depends upon whether you are having capitals indicated or not. If you are not having capitals indicated, custom pronunciation will be provided regardless of capitalization. If you are having capitals indicated, the situation is more complicated. If only the first letter of a word is capitalized, capitalization will be indicated and the custom table substitution will be made. If any other letters of a word are capitalized, all capitalized letters will be indicated, thus breaking the word into pieces. In this case custom substitution will not be made, because it would generally interfere with indication of capitalization.

A sample custom pronunciation table is contained in a file called "table" on the Screen-Voice diskette. This table may be altered using any word processor that is capable of creating an ASCII text file without special word processing control characters. (For instance, the "n" option in WordStar works very well for editing the custom pronunciation table.) A desperate user may use the atrocious Edlin text editor supplied by IBM on the DOS disk. There are limits on the size of table entries and the table as a whole. In each table entry, the string to be substituted may not be over 48 characters long. The total size of the table may not be over 800 characters long (counting spaces and ends of lines as characters). If the table or an entry in it is too long, Screen-Voice will discard the excess characters and say a warning message.

ADAPTING SCREENREADER TO WORK WITH OTHER SYNTHESIZERS

The Screen-Voice diskette includes tables of information about the characteristics of the following brands of speech synthesizer: Echo GP, Intextalker, Microvox, or Votrax Personal Speech System. If you are using one of these synthesizers, you need read no further. If you want to use some other brand of synthesizer, read on. You may need the aid of someone experienced in working with an IBM personal computer, since the following description is quite technical.

If a new brand of speech synthesizer is to be used, a table must be created with information about it, and Screen-Voice must be reconfigured to use this table. Further adaptation must be made if the synthesizer does not meet certain minimum hardware requirements.

The files on the Screen-Voice disk labeled "e.syn", "i.syn", "m.syn" and "v.syn" contain tables of synthesizer parameters needed by Screen-Voice for use with with the Echo GP (e.syn), Intextalker (i.syn), Microvox (m.syn), and Votrax Personal Speech System (v.syn). To use Screen-Voice with another synthesizer a table must be created for that synthesizer. This table must be created as an ASCII file named "n.syn" using a word processor or text editor. It may be necessary to enter unprintable ASCII characters in the file. These characters must be given their equivalent numeric value in the form of a three digit decimal (base 10) number preceeded by a star (*); for instance Control-C would be *003. (To put a star itself into the table, use *042, the equivalent of the ASCII code for star.) The table must contain the following entries listed below with one entry per line:

First line: the string of characters necessary to initialize the synthesizer. (If no initialization is needed, this line must just be a RETURN.);

Second line: the string of characters necessary to make the synthesizer pause (for use between words). (If the synthesizer lacks this feature, this line must just be a RETURN);

Third line: the string of characters necessary to make the synthesizer stop speaking and throw away any speech data received but not yet spoken. (If the synthesizer lacks this feature, this line must just be a RETURN.).

Next twenty-six lines: the strings of characters to be sent to the synthesizer to implement Screen-Voice's 26 rate settings, slowest rate first, fastest rate last. (If the synthesizer has less than 26 possible rate settings, some of these settings will have to be the same; if it has more, 26 settings will have to be chosen.)

Next twenty-six lines: the strings of characters to be sent to the synthesizer to implement Screen-Voice's 26 volume settings, lowest volume first, highest volume last. (If the synthesizer has less than 26 possible volume settings, some of these strings will have to be the same; if it has more, 26 settings will have to be chosen.)

Next twenty-six lines: the strings of characters to be sent to the synthesizer to implement Screen-Voice's 26 pitch settings in flat intonation mode, lowest pitch first, highest pitch last. (If the synthesizer has less than 26 possible pitch settings, some of these settings will have to be the same; if it has more, 26 settings will have to be chosen.)

Next twenty-six lines: the strings of characters to be sent to the synthesizer to implement Screen-Voice's 26 pitch settings in inflected intonation mode, lowest pitch first, highest pitch last. (If the synthesizer has less than 26 possible pitch settings, some of these settings will have to be the same; if it has more, 26 settings will have to be chosen.)

After creating file n.syn, files n.bat and autoexec.n must be edited to set the "mode" command each file contains to the proper parameters for the synthesizer used. File n.bat must be edited to say the name of the synthesizer brand.

Once n.syn, n.bat, and autoexec.n are ready, Screen-Voice must be configured as described earlier in the section of this manual on "Installing Screen-Voice Software." To indicate the type of synthesizer, the person doing the configuration must type:

```
b:n
```

at the point in the procedure for informing Screen-Voice of the type of synthesizer used.

If the speech synthesizer plugs into the serial port (COM1), accepts ordinary English text, is capable of storing at least 256 characters of text, and says that text each time a carriage return is sent, no further adaptation is necessary. If the synthesizer fits in a slot rather than the serial port, or if it requires resident text-to-speech software or a custom driver program, the procedure is much more complex and the assistance of a highly skilled IBM Personal Computer Assembler Language programmer will be needed.

This programmer must be able to understand the following technical jargon. He or she must write a synthesizer interface program as a .COM program to be loaded before Screen-Voice. The program must do the following. It must make a copy of the Communications Interrupt (INT 14H) Vector by using DOS Function 35H. It then must replace the Communications Interrupt (INT 14H) Vector with the address of the synthesizer interface program's own output handling routine by using DOS Function Call 25H. It then must quit and stay resident by using INT 27H. All communications interrupts will thus be sent to the special output handling routine. The output handling routine must redirect communication meant for ports other than COM1 to the BIOS Communications Interrupt Routine whose address (the Int 14H Vector) it has saved. Characters of English text sent by Screen-Voice to COM1 must be intercepted by the output handling routine and stored in a buffer. When a carriage return is sent out by Screen-Voice, it must be detected and the accumulated text must be processed as necessary and then sent to the synthesizer to be spoken. The synthesizer interface program is responsible for saving and restoring all registers it changes and for preserving the integrity of information returned in registers by BIOS calls. It must create and maintain its own stack.

The autoexec.n program must be changed to load the resident software supplied with the synthesizer and to run the synthesizer interface program before it runs Screen-Voice.

COMMAND SUMMARY

OUTPUT CONTROL FEATURES

Before typing each of the following commands, type Control-E or whatever Control key you have selected for output control.

A -- All punctuation pronounced.

B -- Spell with Alpha, Bravo, Charlie alphabet.

C -- Custom pronunciation on/off.

D -- Digits versus numbers.

E -- Enunciation (pauses on/off).

F -- (unused)

G -- Get rid of repeating punctuation on/off.

H -- Help on/off.

I -- Intonation on/off.

J -- Just speak big words (skim).

K -- Key echo on/off.

L -- Letter by letter spelling.

M -- Most punctuation pronounced.

N -- Normal speech.

O -- Only certain columns (must be followed by two two-digit numbers).

P -- Pitch (must be followed by a letter).

R -- Rate (must be followed by a letter).

S -- Some punctuation pronounced.

T -- Top line of screen to be spoken (must be followed by a letter).

U -- Upper case on/off.

V -- Volume (must be followed by a letter).

W -- (unused)

X -- (unused)

Y -- Video cursor position.

Z -- Zap.

SCREEN REVIEW COMMANDS

Type Control-L or whatever Control key you have selected to enter screen review mode. When you are done press ESCAPE to exit. While in screen review mode, you do not have to type Control-L again -- just type the letters listed below. (Note: From screen review you can use, in the usual manner, the output control features listed above.)

A -- Audio cursor position.

B -- Back to marked position.

D -- Down one line.

F -- Find (must be followed by typing the string of characters to be found and then pressing ENTER.)

G -- Go to line (must be followed by a letter).

L -- Say what is to the left.

M -- Mark a place to return.

R -- Say what is to the right.

U -- Up one line.

ENTER -- Say rest of line.

ESCAPE -- Leave screen review.

SCREEN-VOICE

Software to Adapt the Apple Personal Computer for Blind Users

USER'S MANUAL

This manual and the accompanying software were developed under United States Department of Education Contract No. 300-83-0271.

Visek & Maggs
608 W. Pennsylvania Ave.
Urbana, IL 61801

TABLE OF CONTENTS

PREFACE

In 1981, when we designed the blind user interface for use with the Apple Computer and the Echo II speech, the Apple Computer had only very limited memory, a maximum of 64K bytes, and the Echo II was the only inexpensive speech synthesizer available for the Apple. This first blind user interface was designed to take a minimum of memory and would work only with the Echo II. Minimizing memory meant hand-coding the blind user interface in assembly language to squeeze out every last byte. Minimizing memory also meant, unfortunately, minimizing the features available. Even so, the low cost of the Echo II synthesizer, the fact that our blind user interface was supplied free with every Echo II, and the ready availabity of Apple computers made a talking computer an affordable reality for many blind persons.

In 1985, Apple announced the Apple II RAM Expansion card, with deliveries to start in December 1985 or January 1986. The RAM expansion card comes with 256 bytes of RAM, but may easily be expanded to one megabyte. It contains ROM firmware and bank switching hardware to allow it to be used as either a large RAM disk or as supplemental bank switched memory. A number of companies have started producing excellent synthesizers for the Apple, including synthesizers by Micromint, Votrax, and Artic incorporating the new SSI-263 (or SC02) speech synthesis chip. Street Electronics has begun marketing a substantially improved text to speech program for the Echo II. Digital Equipment Company has made the outstanding DECTALK speech synthesizer available at reduced prices to educational institutions. (The DECTALK may be easily interfaced to the Apple, using the Apple Superserial card.) Our new blind user interface has been designed to take advantage of the vastly increased memory resources and the many brands of synthesizer available. No longer is it necessary to restrict the features available in order to save memory. Rather, we have attempted to incorporate all the features that blind users have indicated that they might like added to the limited set of options in the original software we designed. Memory availability has also meant that we could write our program not in assembly language, which is hard to maintain and impossible to transport to other brands of computer, but rather in a higher level language (We choose the popular "C" language) that makes programs easy to maintain, easy to transport to other computers, and easy to adapt to other synthesizers, but which does require more memory.

The blind user interface we have designed will operate on existing models of Apple IIe computers. However, the memory it requires significantly limits the user memory available on a 64K Apple IIe; thus most users with only 64K memory will probably want to continue to use our earlier program to save memory. We see our program as the program for the Apples of the future with their megabyte of RAM each.

This user's manual tells how to install the speech synthesizer and the Screen-Voice software for the Apple IIe Personal Computer, and provides instructions for blind users of the software. This manual assumes the Apple IIe Personal Computer is set up and ready to operate.

Purposes of The Manual

This manual has five purposes:

1. To tell you how to set up your computer and speech synthesizer for use with the Screen-Voice software.
2. To guide you in getting the Screen-Voice software working.
3. To explain the options for reading from the computer screen.
4. To explain the options for controlling the types of output.

INTRODUCTION

The Screen-Voice software for the Apple IIe Personal Computer is a program that provides two essential services for the blind user. First, it sends to the speech synthesizer a copy of everything that appears on the screen, to be spoken as it appears on the screen. Second, it allows the user to select parts of the contents of the screen for review, replaying individual lines, words, or letters.

SYSTEM REQUIREMENTS

This software is designed for use with the Apple IIe Personal Computer with at least 64K memory and the Street Electronics Echo Synthesizer. It may be adapted to take advantage of Apple II series computers with additional memory, for instance the Apple II RAM Expansion Card, and for other synthesizers by following the instructions in the Appendix to this manual.

USING SOFTWARE DEVELOPED FOR SIGHTED USERS OF THE APPLE IIe PERSONAL COMPUTER

Many, but by no means all, software programs developed for sighted users of the Apple IIe Personal Computer can be used with Screen-Voice. For instance, blind users can write and run programs in the BASIC programming language supplied with the Apple Disk Operating System. Many commercially available programs will also work. Some, however, will not work.

There are three possible types of problems in using commercially available programs. First, some commercial programs are supplied on diskettes which incorporate special features to prevent illegal copying. Some of these programs will incorrectly identify an attempt to use Screen-Voice as an attempt to make an illegal copy and may refuse to cooperate with Screen-Voice. Second, some programs write information directly to the computer screen, bypassing the standard Apple IIe screen output software built into the computer and its operating system. This technique gives faster performance for these programs, but makes it impossible for Screen-Voice to capture and speak the programs' output. Third, some programs, because they depend heavily on graphics or screen location of text, are difficult to describe verbally. Thus they are unsuitable for blind users.

Screen-Voice will work well with software developed especially for blind users. This software, for purposes such as word processing and data base management, makes use of the power of the computer to search for the words that blind users want to change or the information they want to get.

HOW MUCH OF THIS MANUAL DO I NEED TO READ?

The most essential part of the manual deals with screen review and output control features of Screen-Voice. You must thoroughly familiarize yourself with these features. However, don't worry about forgetting some of them -- you can have the computer help you while you are running a program, without interfering with the operation of the program.

The section on "Adapting Screen-Voice to Work with Other Synthesizers is meant for experienced programmers who want to alter Screen-Voice to work with other synthesizers. Ordinary users can skip this section.

GETTING STARTED

We will first list and then explain in detail the steps in getting started with your Screen-Voice software. Experienced Apple users may read the headings to determine which sections they already know.

SYNTHESIZER PREPARATIONS

Make sure the Street Electronics Echo Synthesizer is plugged into a slot in the computer in accordance with the manufacturer's instructions. Run the demonstration routine provided with the synthesizer to make sure it is working correctly.

PREPARING WORKING COPIES OF SCREEN-VOICE

Get the Apple DOS 3.3 diskette supplied with your computer. Follow the instructions in the DOS Manual to make a copy of the Screen-Voice diskette (Use the COPYA program). Put the original Screen-Voice diskette in a safe place in case your working copies of the program are damaged. Now use the FILEM program on the DOS diskette to copy the file "TEXTALKER.RAM" & TEXTALKER.RAM.OBJ from the Echo diskette to the copy you have just made of the Screen-Voice diskette. Finally use the COPYA program to make several working copies of the diskette you have just made. Put one of these working copies in a safe place as a backup copy.

USING SCREEN-VOICE

Once you have made working copies of your Screen-Voice diskette (with "TEXTALKER.RAM" added), using Screen-Voice is very simple. All you will need to do is to put a working copy in Drive I before turning on the computer. When the computer is turned on, the Screen-Voice software will be loaded automatically into your computer and the synthesizer will say "Screen-Voice ready." You may then replace the working copy of the Screen-Voice diskette with one of your program diskettes. If you turn the computer off, you will have to put a copy of the Screen-Voice diskette in Drive I before you turn it on again.

OUTPUT CONTROL FEATURES

When your computer is turned on and Screen-Voice is activated, it automatically selects a number of standard or "default" features for reading what is sent to the screen. However, for special purposes you may want to choose other features. This chapter explains the standard features and your options for varying them.

The standard features for Screen-Voice attempt to imitate what a sighted person would say if asked to read what was written on the screen. The special features allow various types of more detailed reading, for tasks such as proofreading. They also allow less detailed reading to enable rapid skimming of the text on the screen.

How to Select an Output Control Feature

To select an output control feature, hold down the Control key, and while holding it down press the E key. This combination is called Control-E. The next step is to release these two keys and press the command key given below for the feature desired. In some cases it will also be necessary to type a number. For these commands, upper case and lower case letters have the same specified effect.

An attempt has been made to choose command keys so that the letters on the keys are related to the first letters of the names of the commands, but this has not always been possible. Single-key commands rather than whole-word commands are used because, although they are harder to remember, they are much faster to use after some practice. One can issue any number of Screen-Voice commands in a row and commands may be issued at any time the computer is accepting key presses.

GENERAL PRONUNCIATION FEATURES

Most speech synthesizers, including the Echo synthesizer, allow commands to be sent by the computer to control synthesizer pitch, volume, speed, and intonation. Details vary from synthesizer to synthesizer. Thus the commands described here for varying synthesizer voice characteristics may have different effects with different synthesizers. If a particular brand of synthesizer lacks certain pronunciation features, the command may have no effect, but will do no harm.

Intonation

Most synthesizers allow the user to select either an imitation of human speech intonation or robot-like flat speech. Some people prefer the flat speech because no synthesizer selling for less than $4000 offers good intonation. Others find flat speech boring and prefer inaccurate intonation. Screen-Voice sets the synthesizer initially to provide intoned speech. To switch to flat speech type Control-E and then I. To switch back to intoned speech, type Control-E and I again.

Remember: I is for Intonation on or off.

Speech Rate

Most synthesizers allow the user to control the rate of speech. Blind persons who are experienced with speech synthesizers usually like to run them as fast as possible, so as to increase their effective reading speed. However, even experienced users sometimes need to slow down the synthesizer for difficult passages. To set the synthesizer speech rate, type Control-E, then R, and then a letter from A to Z. The letter A stands for the slowest rate, the letter Z for the fastest rate and other letters for rates in between. If your synthesizer has only one rate, this command will have no effect. If your synthesizer, for instance the Echo has only two rates, letters in the first half of the alphabet will select the slower rate, and letters in the second half of the alphabet will select the faster rate. The initial rate is programmed into the synthesizer by its manufacturer. For some synthesizers, such as the Intextalker or Microvox, changing the rate may also change the pitch.

Remember: R is for Rate;

the following letter selects the rate.

Volume

Some synthesizers allow users to control speech volume through the computer. Others offer a convenient volume control knob on the synthesizer. Still others have both options. To set the volume through the computer, type Control-E, then V, and then a letter from A to Z. If the synthesizer has computer-controllable volume, the letter A will produce the softest volume, the letter Z the loudest, and other letters will produce intermediate volume levels. If the synthesizer has less than 26 different levels of controllable volume, letters close together in the alphabet may not produce different volumes. On most synthesizers, but not the Echo, use of the volume control knob on the synthesizer itself is a better way to change volume. The initial volume is programmed into the synthesizer by its manufacturer.

Remember: V is for Volume;

the following letter selects the volume.

Pitch

Some synthesizers allow computer control of speech pitch. It may lessen fatigue to vary the pitch from time to time so as not to listen to the same voice for hours on end. To set the pitch control through the computer type Control-E, then P, and then a letter from A to Z. If the synthesizer has computer-controllable pitch, the letter A will produce the softest pitch, the letter Z the loudest, and other letters will produce intermediate pitch levels. If the synthesizer has less than 26 different levels of controllable pitch, letters close together in the alphabet may not produce different pitch levels. The initial pitch is programmed into the synthesizer by the synthesizer manufacturer. For some synthesizers, such as the Intextalker or Microvox, pitch can also be varied by changing the rate.

Remember: P is for Pitch;
the following letter selects the pitch.

Pauses

Scientific research shows that it is easier to understand speech synthesizers if they are programmed to pause briefly between words. However, the pauses slightly slow down the reading speed. When Screen-Voice starts running, it does not add pauses between words; however there is an option to add pausing. To switch between not pausing and pausing, press Control-E, and then letter E. To switch back, use the same command.

Remember: E is for Enunciation on or off.

FEATURES FOR MORE DETAILED READING

The usual way of reading text does not bring out all the detail necessary for proofreading. The proofreading features of Screen-Voice allow the user to have just the amount of extra detail he or she wants for the particular task at hand.

Spelling or Whole Words

When Screen-Voice is activated it reads by whole words. For proofreading it may be necessary to have individual words spelled out, since two words may be spelled differently but pronounced the same (for instance "t h e i r" and "t h e r e"). A word could be spelled wrong but pronounced correctly (for instance "t h a i r" will be pronounced "there") or spelled correctly but pronounced wrong due to a shortcoming of the speech synthesizer. To have words spelled out, type Control-E, and then L. To return to normal whole word pronunciation press Control-E and then N.

Some people have trouble understanding a synthesizer when it is spelling letter by letter. To have spelling done not with letters, but with the "Alpha, Bravo, Charlie" alphabet, press Control-E and then type B.

Remember: L for Letter by letter spelling;
B is for Alpha, Bravo, Charlie spelling;
N is for Normal (whole word) pronunciation.

Capital Letters

Proofreading may require distinguishing upper case from lower case letters. To have upper case letters indicated, press Control-E and then U. The computer will then say "cap" before each upper case letter. To go back to regular reading, press Control-E and then U again.

Remember: U is for Upper case on or off.

Punctuation

In normal reading, special symbols such as "$" and "+" are usually read out, but common punctuation marks such as period and comma are not pronounced. When Screen-Voice starts, it is in normal reading mode. All special symbols and all punctuation on the screen are pronounced except the following: dash, period, comma, semicolon, exclamation point, question mark, star, double quotes, apostrophe, tilde, grave accent, vertical line, backslash, left parenthesis, right parenthesis, left bracket, right bracket, left brace, and right brace. When reading numbers, a decimal point will be pronounced as point. If a combination of digits, dollar signs, commas, and periods is encountered that is not a well-formed number or monetary amount, all characters will be pronounced to alert the user to the unusual situation.

This option is called "Some" punctuation. If you have switched to another punctuation option, you can return to "Some" punctuation by typing Control-E and then S.

When proofreading, it is usual to pronounce all punctuation but not to indicate spaces. This option is called "Most" punctuation. It is available by pressing Control-E and then M.

Finally, for some purposes it is necessary to know exactly what is on the screen, including spaces. This option, called "All" punctuation, is available by pressing Control-E and then A.

None of these modes provides an indication of video cursor movement about the screen. Positions of this cursor may be checked at any time the computer is accepting input, by using the video cursor command described below.

Remember: S is for Some punctuation;
M is for Most punctuation;
A is for All punctuation.

Numbers

The reading of numbers also differs in ordinary speech and in proofreading. In normal speech, for instance, a three followed by a four followed by a five is pronounced "three hundred forty five." When proofreading, the same number is pronounced digit by digit, as "three four five." When Screen-Voice starts, numbers are pronounced in normal speech mode, except for combinations of digits, commas, dollars signs and periods that constitute neither well-formed numbers nor proper monetary amounts. These combinations, such as $3,4.5 are spoken character-by-character. To switch between normal speech mode and digit-by-digit mode type Control-E and then D. To switch back, use the same command.

Remember: D is for Digits on or off.

FEATURES FOR LESS DETAILED READING

The following options allow text to be read faster by skipping portions of it. (Remember also that setting the synthesizer to a fast speech rate, as explained earlier, will speed up reading without loss of any text.)

Skimming

Often a user wishes to skim through text to see if it is of any interest. A skimming option is available that just reads the long words in the text, those with seven or more letters. Numbers and strings of punctuation and other symbols are skipped regardless of length. To just read these big words, type Control-E and then J. To return to ordinary reading, type Control-E and J again.

Remember: J is for Just reading long words.

Ignore Top Lines

Normally Screen-Voice reads everything that appears on the screen. However, some programs continually write status information on top line of the screen. This information, such as the number of the column currently being written, is of little value to the blind user, and is highly annoying when it interrupts spoken text. To set the computer to ignore lines at the top of the screen, type Control-E, then T, then a letter from A to Y. The letter indicates the first line that will be pronounced. If the letter is A, the whole screen will be pronounced. If the letter is B, pronunciation will start at the second line on the screen, if it is C, pronunciation will start at the third line on the screen. The letter Y turns off speech on all but the bottom line. To turn all the lines back on again, type Control-E , then T, then A.

Remember: T is for Top line of screen;
the following letter specifies the first line to read.

Columnar Reading

A feature that is useful both for skimming and for other purposes is columnar reading. One use of this feature is to read a column of numbers in a table. Another is to skim text, for instance, by reading only the first twenty characters on each line. There are forty columns on the screen. The leftmost column is considered to be column 00 and the rightmost column is considered to be column 79. To set which columns are to be read, press Control-E, then type the letter O. Next, type in two two-digit numbers, each between 00 and 39. If you make a mistake, the computer will tell you and give you a chance to correct it. To return to normal reading of the whole screen press Control-E, type the letter O, and press the RETURN key.

Remember: O is for Only certain columns;
the next 2 digits give the left column;
the last 2 digits give the right column.

Count Repeating Punctuation

A feature of some programs that is useful for sighted persons but annoying for blind users is repeating punctuation. One word processing program, for instance, indicates page breaks by putting a row of seventy-nine dashes on the screen. Few people want to hear the synthesizer say "dash" seventy-nine times. This option deals with this problem.

When Screen-Voice is activated it is in "Some" punctuation mode (see explanation above), where some punctuation characters will be pronounced and some will be ignored. In addition, Screen-Voice initially does not count repeating punctuation--each instance of the punctuation to be spoken will be individually pronounced, no matter how often it repeats.

Pressing Control-E and then G switches to a mode where repeating punctuation signs are counted and the count is spoken. The user will only hear the count for those characters that the current punctuation mode (All, Most or Some) is supposed to pronounce. All other characters will continue to be ignored. Using this feature in Most or All modes, seventy-nine dashes in a row would result in the synthesizer saying the word "dash", followed by the count "seventy-nine", followed by the word "times". To return to saying each instance of the repeating punctuation, type Control-E followed by the letter G again.

Remember: G is for Get rid of repeating punctuation (on or off).

Stopping Speech

Often the computer will display a screenful of text, but as soon as Screen-Voice starts saying it, the user will realize that he does not want to hear it. The zap feature allows the user to have the synthesizer stop speaking the text it is preparing to say. The effectiveness of the zap feature depends upon the brand of synthesizer used -- some cannot be stopped once text has been sent to them. The zap command can be entered only if the running program is accepting key input. Sometimes, even when a program will not take regular key input, it can be stopped by pressing Control-C. Then the zap command can be given. To stop speech, type Control-E, then Z. In help mode, the computer will send the phrase "clear buffer" to the synthesizer immediately after the "zap" command; however, this phrase may not be spoken since some synthesizer remain inactive for a few moments after receiving a "zap" command.

Remember: Z is for zap the speech buffer.

Input Modes

Whenever typing a key causes the corresponding letter, number, or symbol to appear on the screen, Screen-Voice will immediately read the letter, number, or symbol from the screen, without waiting for the end of the line. This is called key echoing. Good typists often feel that their time is being wasted listening to the synthesizer echo each key. To turn off key echo, type Control-E and then K. To turn it on again, type Control-E and then K again.

Remember: K is for Key echo on or off.

HELP FEATURES

Reporting Command Key Functions

There are so many options that it may be hard to remember which options have been set or which key affects which options. Typing Control-E and then H, will put the computer in a mode where the function of each key is spoken if the key is pressed. If the same command key switches a mode on and off, the new status will be spoken. To turn off the help messages, type Control-E and then H again.

Remember: H is for Help on or off.

Position of Video Cursor

Most programs put a cursor or small flashing symbol at the point on the screen where they expect to receive text. To hear the location of this point, press Control-E and then Y. The coordinates will be spoken as first a letter corresponding to the line on the screen and then a number from 0 to 79 corresponding to the column position on the screen.

Remember: Y is for video cursor position.

Changing the Command Control Character

If the word processor or other program you are using employs Control-E as a command, you will need to select a different control key by using the following procedure. Instead of pressing a letter key to choose one of the special features, press a different control key (ie. press a different letter key while holding down the control key). You may select any key except C (because Control-C is used by DOS for escaping from programs), H (because Control-H is Backspace), M (because Control-M is RETURN), or the key currently being used to start screen review, which will be discussed below. In selecting a key, avoid control keys that have important functions in software you plan to use. The new control key can then be used to initiate Screen-Voice output control commands.

Remember: Control keys change the command control character.

SCREEN REVIEW FEATURES

Screen-Voice provides a wide variety of options for reviewing what the computer has put on the screen. You may want a particular line of the screen replayed because you have forgotten what it said. You might have had trouble understanding the speech synthesizer and want a few words repeated or spelled out.

The video screen of the Apple IIe Personal Computer has 24 lines of 40 characters each. For purposes of screen review, the twenty-four lines are called lines A through X -- the first twenty-four letters of the alphabet. The character positions are called position zero through position thirty-nine. Programs running on the Apple personal computer keep track of the location where the next character will be written. Usually they indicate this location to the user by putting a flashing spot of light on the screen. This flashing spot is called the "video cursor." When used for screen review, Screen-Voice keeps track of where it is reading from the screen. The next location to be read from is specified by the "audio cursor."

Starting Screen Review

To start screen review, hold down the Control key and press the L key. If the word processor, data base system, or other program running on the personal computer is accepting key input, its operation will be suspended so you can review the screen. You may now may select various options and continue reviewing until you decide to return to the program you were using. To return to that program, press the Escape key. While you are in screen review mode, you do not need to type Control-L before selecting each option. Just type the appropriate letters.

While you are in screen review mode, all of the output control modes previously selected are still in effect. If you wish to change one of Screen-Voice's output control features from within screen review mode, you may press Control-E (or whatever control key you have previously selected as the command key) and the usual characters needed to change the feature in question. Note that in screen review mode, all key presses will be interpreted as Screen-Voice commands. No keys will be passed to any other program.

When you enter screen review mode, the audio cursor will be set to the same position as the video cursor. If the video cursor is outside the boundaries you specified using the column or top of screen commands, however, the audio cursor will be set to the upper left corner of the screen window. Screen-Voice will notify you of this alternate position. The video cursor will not be moved during screen review mode.

Remember: Control-L initiates screen review;
Escape ends screen review;
all output control options are still available.

Going to a Particular Line

The lines on the screen are designated by the letters from A at the top to Y at the bottom. To go to a particular line, type G and then the letter corresponding to the line.

Remember: G to Go to a line;

followed by a letter to specify the line.

Saying Rest of Line

To say the rest of the line the audio cursor is on, press the RETURN key. When the line has been read, the audio cursor will be moved to the start of the same line. Thus you can hear the line again by pressing the RETURN key.

Remember: RETURN to say the rest of the line.

Moving the Audio Cursor

You can move up, down, right, or left on the screen by pressing the U, D, R, and L keys. If you are in the word-by-word mode, pressing the R key will cause the next word or group of one or more punctuation symbols to the right to be pronounced, and pressing the L key will cause the next word or punctuation group to the left to be pronounced. If you are in the letter-by-letter mode, pressing these keys will cause the next letter or character to be pronounced. It is usually a good idea to switch into "all punctuation pronounced" mode before reviewing to the left or right. Otherwise pronunciation of many punctuation symbols and of spaces will be suppressed. Pressing the U key will move the audio cursor up to the start of the next higher line on the screen. Pressing the D key will move the audio cursor down to the start of the next lower line on the screen. If you hit the right or left end of a line or the top or bottom of the screen, the computer will tell you and no harm will be done. When you hit the right end of a line the audio cursor will be repositioned automatically to the start of the same line.

Remember: R is for Right;

L is for Left;

U is for Up;

D is for Down.

Audio Cursor Position

You may wish to know exactly where the audio cursor is on the screen. (The audio cursor starts in the same place as the video cursor when you enter screen review mode, but you may have moved the audio cursor.) To have the location of the audio cursor spoken press the A key. As usual, the position is given by a letter representing the line and a number representing the column.

Remember: A is for Audio cursor.

Mark a Screen Position and Go Back to it

You may want to mark a position on the screen so that you can go back to it later. Press the M key to mark the current audio cursor position. Press the B key to go back to it. The marked position will stay the same unless you specifically change it. Thus you can mark a position, leave screen review mode and do other things. When you come back to screen review mode the marker will be at the same place.

Remember: M is for Mark;

B is for Back.

Finding a String of Characters on the Screen

To find a string of characters on the screen, type F. Next type the string you want to find and press the RETURN key. (Note: If the string is over 19 characters long, the twentieth and following characters will be ignored.

At this point, the synthesizer will pronounce the string you are looking for. Then it will start at the top of the screen and look for the string you want to find. If you have used the column or top of screen commands, the search will be limited to the screen window you have specified. If Screen-Voice finds the string, the synthesizer will say the line of text containing the string, and then set the audio cursor to the start of the string it has found. If it does not find the string, the synthesizer will say "not found" and the position of the audio cursor will be unchanged. If you find an instance of the string and want to look for another, press the space bar. When you are done looking, press any other command key.

Remember: F to Find a string;
followed by the string and RETURN;
space bar to try to find it again.

Changing the Screen Review Control Character

If the word processor or other program you are using employs Control-L as one of its commands, you will need to select another control key by using the following procedure. Instead of pressing a letter key to choose one of the screen review features, press a different control key (ie. press a different letter key while holding down the control key). You may select any key except C, H, M, or the key currently being used to start output control commands.

Remember: Control keys change the screen review character.

End Screen Review Mode

Finally, you will want to escape from review mode to get back to the program you were running. Press the Escape key to end screen review.

Remember: Escape to end screen review.

ADAPTING SCREEN-VOICE FOR THE APPLE II RAM EXPASION CARD

The specifications for the Apple II RAM Expansion card have not yet been released. It is expected that it will implement a "bank select option," to be activated by accessing certain memory locations. The following changes would need to be made in Screen-Voice software. (These changes will be simple for any experienced Apple machine-language programmer to make; impossible for anyone else to make.) First, provision should be made to load the software (file SV on the diskette) into the RAM Expansion card. Second, the short assembler language interface program "APY" on the diskette should be altered so that it will reside in the RAM Expansion card and connect with file SV there.

ADAPTING SCREEN-VOICE TO WORK WITH OTHER SYNTHESIZERS

The Screen-Voice source code includes a tables of information about the characteristics of the Echo speech synthesizer. If you are using this synthesizer, you need read no further. If you want to use some other brand of synthesizer, read on. The "C" source code supplied on should be altered by supplying new values for variables *volstr, *ratestr, *fpstr, *ipstr, *initstr, *paustr, and *stapstr appropriate to the given synthesizer in as described in detail below. File TEXTALKER.RAM should not be used--this is the initialization program for the Echo synthesizer only. Instead, the initialization program for automatic speaking of output directed to the screen should be run for the syntehsizer selected. If necessary, changes should be made in routine INIT in file APY.

The "C" program variables should be initialized as follows;

*initstr: the string of characters necessary to initialize the synthesizer. (If no initialization is needed, this string must just be a RETURN.);

*pausestr: the string of characters necessary to make the synthesizer pause (for use between words). (If the synthesizer lacks this feature, this string must just be a space);

*zapstr: the string of characters necessary to make the synthesizer stop speaking and throw away any speech data received but not yet spoken. (If the synthesizer lacks this feature, this line must just be a RETURN.).

*ratestr: the strings of characters to be sent to the synthesizer to implement Screen-Voice's 26 rate settings, slowest rate first, fastest rate last. (If the synthesizer has less than 26 possible rate settings, some of these settings will have to be the same; if it has more, 26 settings will have to be chosen.)

*volstr: the strings of characters to be sent to the synthesizer to implement Screen-Voice's 26 volume settings, lowest volume first, highest volume last. (If the synthesizer has less than 26 possible volume settings, some of these strings will have to be the same; if it has more, 26 settings will have to be chosen.)

*fpstr: the strings of characters to be sent to the synthesizer to implement Screen-Voice's 26 pitch settings in flat intonation mode, lowest pitch first, highest pitch last. (If the synthesizer has less than 26 possible pitch settings, some of these settings will have to be the same; if it has more, 26 settings will have to be chosen.)

*ipstr: the strings of characters to be sent to the synthesizer to implement Screen-Voice's 26 pitch settings in inflected intonation mode, lowest pitch first, highest pitch last. (If the synthesizer has less than 26 possible pitch settings, some of these settings will have to be the same; if it has more, 26 settings will have to be chosen.)

COMMAND SUMMARY

OUTPUT CONTROL FEATURES

Before typing each of the following commands, type Control-E or whatever Control key you have selected for output control.

A -- All punctuation pronounced.

B -- Spell with Alpha, Bravo, Charlie alphabet.

C -- (unused)

D -- Digits versus numbers.

E -- Enunciation (pauses on/off).

F -- (unused)

G -- Get rid of repeating punctuation on/off.

H -- Help on/off.

I -- Intonation on/off.

J -- Just speak big words (skim).

K -- Key echo on/off.

L -- Letter by letter spelling.

M -- Most punctuation pronounced.

N -- Normal speech.

O -- Only certain columns (must be followed by two two-digit numbers).

P -- Pitch (must be followed by a letter).

R -- Rate (must be followed by a letter).

S -- Some punctuation pronounced.

T -- Top line of screen to be spoken (must be followed by a letter).

U -- Upper case on/off.

V -- Volume (must be followed by a letter).

W -- (unused)

X -- (unused)

Y -- Video cursor position.

Z -- Zap.

SCREEN REVIEW COMMANDS

Type Control-L or whatever Control key you have selected to enter screen review mode. When you are done press ESCAPE to exit. While in screen review mode, you do not have to type Control-L again -- just type the letters listed below. (Note: From screen review you can use, in the usual manner, the output control features listed above.)

A -- Audio cursor position.

B -- Back to marked position.

D -- Down one line.

F -- Find (must be followed by typing the string of characters to be found and then pressing RETURN.)

G -- Go to line (must be followed by a letter).

L -- Say what is to the left.

M -- Mark a place to return.

R -- Say what is to the right.

U -- Up one line.

RETURN -- Say rest of line.

ESCAPE -- Leave screen review.

SCREEN-VOICE

Software to Adapt the TRS-80 Model 100 Portable Computer

for Blind Users

USER'S MANUAL

This manual and the accompanying software

were developed under United States Department

of Education Contract No. 300-83-0271.

Visek & Maggs
608 W. Pennsylvania Ave.
Urbana, IL 61801

TABLE OF CONTENTS

PREFACE

This user's manual tells how to the Screen-Voice software for the TRS-80 Model 100 Portable Computer and provides instructions for blind users of the software. This manual assumes the TRS-80 Model 100 Portable Computer is set up and ready to operate, and that the speech synthesizer has been connected to the computer in accordance with the instructions supplied with the synthesizer.

INTRODUCTION

The Screen-Voice software for the TRS-80 Model 100 Portable Computer is a program that provides four essential services for the blind user. First, it sends to the speech synthesizer a copy of everything that appears on the screen, to be spoken as it appears on the screen. Second, it allows the user to select whether text should be spoken as words or spelled. Third, it allows the user to control what punctuation will be spoken. Fourth, it provides a notetaking and review system.

THE COMPUTER AND THE SYNTHESIZER

THE COMPUTER

This software is designed to function with the Radio Shack TRS-80 Model 100 Portable Computer. Unless you have considerable computer expertise, you should buy a new computer from a local authorized Radio Shack Dealer so that you can get qualified help in setting up the computer and warranty repairs if necessary. If you do have computer expertise, you may be able to save money by ordering a computer by mail or by buying a second-hand computer.

To be used with this program, the TRS-80 Portable Computer must have 32K of RAM memory. (Any Radio Shack dealer will be happy to sell and install additional memory if your computer comes with less than 32K of RAM.)

THE SPEECH SYNTHESIZER

This software requires a special speech synthesizer for the TRS-80 Model 100. You should follow the instructions supplied with the synthesizer to install it.

MAKING WORKING COPIES OF THE SCREEN-VOICE SOFTWARE

Before configuring the Screen-Voice software you should make several working copies of it. Screen-Voice is supplied on two cassettes. One contains a copy of a program called TALK, the other contains a program called NOTE.

To make copies of TALK, place the TALK casette in the cassette recorder, ready the recorder, select BASIC from the menu, then type:

CLEAR 45000,256 <ENTER>

LOADM "TALK"

When SCREEN is successfully loaded insert and rewind a blank data cassette in the recorder and type:

SAVEM "TALK",450$\overset{56}{\cancel{00}}$,58181

To make an additional backup copy, once again insert and rewind a blank data cassette in the recorder and type:

SAVEM "TALK",450$\overset{56}{\cancel{00}}$,58181

BEST COPY AVAILABLE

To make copies of NOTE, place the NOTE casette in the cassette recorder, ready the recorder, select BASIC from the menu, then type:

```
LOAD "NOTE"
```

When NOTE is successfully loaded insert and rewind a blank data cassette in the recorder and type:

```
SAVE "NOTE"
```

To make an additional backup copy, once again insert and rewind a blank data cassette in the recorder and type:

```
SAVE "NOTE"
```

USING SCREEN-VOICE

TALKING BASIC SYSTEM

Program TALK, the Talking Basic System is designed to allow you to run programs written in BASIC and to write programs in BASIC. To activate the Talking BASIC System type:

```
CLEAR 45000,256
```

and then, being sure that the diskette with the TALK program is ready in your cassette recorder, type:

```
RUNM "TALK"
```

At this point Talking BASIC is loaded, but not activated. To activate it type:

```
CALL 45059
```

(You may deactivate the Talking BASIC at any time by typing:

```
CALL 45062
```

You may then reactivate it at any time by typing:

CALL 45059

Warning -- you must deactivate Talking BASIC before leaving BASIC to go to the main menu or options on the main menu will not work. If you accidently exit to the main menu, with Talking BASIC still active, you must restart the computer by turning off the master power switch on the bottom of the computer, removing any 110 volt adapter, waiting a minute, and then turning the computer on again.

SPEECH OPTIONS

The standard features for Talking BASIC attempt to imitate what a sighted person would say if asked to read what was being written on the screen.

Spelling or Whole Words

When Talking BASIC is activated it reads by whole words. For proofreading it may be necessary to have individual words spelled out, since two words may be spelled differently but pronounced the same (for instance "t h e i r" and "t h e r e"). A word could be spelled wrong but pronounced correctly (for instance "t h a i r" will be pronounced "there") or spelled correctly but pronounced wrong due to a shortcoming of the speech synthesizer. To have words spelled out, type CALL 45065. To return to normal whole word pronunciation press type CALL 45068.

Remember: CALL 45065 for letter by letter spelling;

CALL 45068 for whole word pronunciation.

Punctuation

In normal reading, special symbols such as "$" and "+" are usually read out, but common punctuation marks such as period and comma are not pronounced. When Screen-Voice starts, it is in normal reading mode. All special symbols and all punctuation on the screen are pronounced except the following: dash, period, comma, semicolon, exclamation point, question mark, star, double quotes, apostrophe, tilde, grave accent, vertical line, backslash, left parenthesis, right parenthesis, left bracket, right bracket, left brace, and right brace. When reading numbers, a decimal point will be pronounced as point. If a combination of digits, dollar signs, commas, and periods is encountered that is not a well-formed number or monetary amount, all characters will be pronounced to alert the user to the unusual situation.

This option is called "Some" punctuation. If you have switched to another punctuation option, you can return to "Some" punctuation by typing CALL 45074.

When proofreading, it is usual to pronounce all punctuation but not to indicate spaces. This option is called "Most" punctuation. It is available by typing CALL 45077.

Finally, for some purposes it is necessary to know exactly what is on the screen, including spaces. This option, called "All" punctuation, is available by typing CALL 45071.

Remember: CALL 45074 is for some punctuation;
CALL 45077 is for most punctuation;
CALL 45071 is for all punctuation.

NOTE-TAKING AND REVIEW SYSTEM

The Note-taking and review system is a program in BASIC that lets you take notes, read the notes back, and transfer the notes to another computer. To activate this system, first activate Talking Basic. Then put the cassette with program NOTE in the recorder and type:

```
RUN "NOTE"
```

When program NOTE is running, anything typed on the keyboard will be saved in a large, 8000 character storage area in memory. If this area becomes full, the computer will beep every time you try to type something.

You have a number of options while running program the Notaking and Review System. Each option is selected by pressing one of the function keys under the screen. The options are as follows:

Function Key 1

Pressing this key sends the entire file out the serial port. The serial port should be connected to the receiving computer through a "modem eliminator." The receiving computer should be set to receive a text file at 300 baud, with even or ignored parity bit, one stop bite, and no X-on, X-off protocol. A shareware program, such as PC-TALK for the IBM PC, is ideal for this purpose.

Function Key 2

Pressing this key saves the entire file on a cassette with the name "NOTES".

Function Key 3

Pressing this key reads a text file named "NOTES" from cassette.

Function Key 4

Reserved for future expansion.

Function Key 5

Reserved for future expansion.

Function Key 6

Reserved for future expansion.

Function key 7

Speaks out entire contents of note file.

Function key 8

Erases entire note file. To prevent accidental erasure, this key beeps when pressed. Then if you really want to erase the file, press "Y". Otherwise press any other key and the file will remain unharmed.

Altering the Notetaking and Review System

Since the system is written in ordinary BASIC, you (or a friend who is used to programming in BASIC) can easily alter it to add any features you want.

TALKHELPER

Software to Adapt the IBM Personal Computer for

Use by Persons with Speech Disabilities

USER'S MANUAL

This manual and the accompanying software

were developed under United States Department

of Education Contract No. 300-83-0271

Visek & Maggs
608 W. Pennsylvania Ave.
Urbana, IL 61801

TABLE OF CONTENTS

PREFACE

Talkhelper is a program designed to allow non-speaking persons to use a personal computer with a speech synthesizer as a substitute voice. Many non-speaking persons have other physical or mental handicaps. Talkhelper offers flexibility in selection of input devices and screen displays so that it may be customized for each user to make maximum use of his or her abilities and to compensate for his or her disabilities.

This manual is for use with Talkhelper for the IBM Personal Computer. (Other versions of Talkhelper are available for the Apple IIe personal computer and the Radio Shack Model 100 portable computer.)

PURPOSES OF THIS MANUAL

This manual has four purposes:

1. To tell you how to set up the computer, speech synthesizer, and input device for use with the Talkhelper software;
2. To help you configure the Talkhelper software for the computer, synthesizer, input device, and options you want;
3. To explain how to create and edit screen menus;
4. To explain how to use Talkhelper as a speech aid.

DO I HAVE TO READ ALL OF THIS MANUAL?

If you are planning to install the speech synthesizer, input devices and Talkhelper yourself, you should study the sections of this manual on installation before you try to install the synthesizer, input devices, and Talkhelper. If someone else has installed the synthesizer, input devices, and Talkhelper for you, you can ignore these sections of the manual.

Unless you intend to use a brand of synthesizer other than one of those listed below, you can ignore the section of the manual on adapting Textalker for use with other synthesizers.

All of the rest of the manual is important, regardless of what synthesizer you use and regardless of who installs it.

SELECTION OF COMPUTER, SPEECH SYNTHESIZER, AND INPUT DEVICES

COMPUTER SELECTION

This software is designed to function with the IBM Personal Computer. Unless you have considerable computer expertise, you should buy a new computer from a local authorized IBM Dealer so that you can get qualified help in setting up the computer and warranty repairs if necessary. If you do have computer expertise, you may be able to save money by ordering a computer by mail or by buying a second-hand computer.

All models of the IBM Personal Computer come equipped with the diskette drive and memory needed for this program. If you plan to use picture menus, the computer must have the IBM Color/Graphics Monitor Adapter. (A color monitor is desirable, but not necessary for picture menu displays -- you can use a black and white, green screen, or amber screen monitor with picture menus.

SPEECH SYNTHESISER SELECTION

Any of the following synthesizers may be used with the IBM Personal Computer: Echo GP, Intextalker, Microvox, Votrax Personal Speech System.

The speech quality of all these synthesizers is about equal. However, you may have a personal preference, so listen to several before buying. None of these synthesizers provides truly high quality speech -- for high quality speech the best choice is the very, very expensive Dectalk synthesizer from Digitial Equipment Company.

INPUT DEVICE SELECTION

Some persons with speech handicaps can use the computer keyboard very well. However, since many persons with speech handicaps also have other disabilities which prevent them from typing on the regular computer keyboard, Talkhelper allows the use of different input devices so that each user can use the form of input best adapted to his or her disabilities.

The possible devices are: the regular IBM Personal Computer keyboard or special switches connected to the hand control connector on the computer. The following discussion is meant to help you choose an input device. Often it is obvious which device should be used. Sometimes it is difficult to determine which device is best, and it may be necessary to try several types of input device and to seek the advice of an appropriate health care or education professional. Some users may be able to advance to more complex input devices that provide faster control of the computer. The discussion below will consider factors in choice of input devices for users with speech disabilities only,or those with speech disabilities and with poor motor control, poor vision, poor or no reading ability, or poor memory. The various possible types of input device and the ways in which the can be used will be considered in turn.

COMPUTER KEYBOARD INPUT

The computer keyboard has two great advantages and two great disadvantages. The two great advantages are that a keyboard comes at no extra charge with every IBM Personal Computer and that a good typist can enter any English phrase at high speed using the keyboard. The two great disadvantages are that the keyboard is hard to use for persons with physical disabilities and hard to understand for persons with cognitive disabilities. Talkhelper offers several ways to use the keyboard to allow customization for different types of users.

Full Keyboard Input

Full keyboard input is ideal for a person with speech disabilities, but no other disabilities. With a relatively small investment of time such a person should be able to learn typing, a skill with many applications, and should be able to reach a typing speed of over 50 words a minute. Such a person will be able to use Talkhelper to say anything he or she wants at a speed of at least 50 words a minute, as compared to a normal speech rate of 80 to 100 words a minute.

Full keyboard input may also be possible for a person with a moderate level of physical disability. The disability may reduce the accuracy with which keys are pressed, the speed with which they can be reached, or may prevent use of the shift key. Even with these problems, it may be possible to get satisfactory results from Talkhelper. Often accuracy can be improved substantially by the use of a keyguard or an oversized keyboard. (These are available from companies supplying computer products for handicapped users.) If the disability substantially reduces the speed with which keys can be selected, the user will probably wish to supplment typed text with phrase selection input (see the section below entitled "Phrase Selection").

Restricted Keyboard Input

Users with poor accuracy, limited reach, or inability to understand the complexities of a full typewriter keyboard may benefit from an option of Talkhelper that allows the assigning of the whole keyboard to be treated as a single switch or of the spacebar and the rest of the keyboard each to be considered as a switch.

SWITCH INPUT

Switch input also has advantages and disadvantages. The advantages of switches are that they are easy to buy and attach to the computer and that they can be used by persons with physical handicaps that prevent pressing keys on the computer keyboard. One disadvantage is that input is much slower than with full keyboard typing. Another disadvantage is the difficulty users with limited cognitive ability may have in associating closing switches with what is happening on the screen.

Talkhelper can be configured to work with either one or two switches. The two-switch input mode offers better control and faster input than the one-switch mode. Therefore it is the preferred mode for persons who can operate more than one switch.

SETTING THE SPEECH SYNTHESIZER SWITCHES AND CONTROLS

Speech synthesizers have switch settings which must be made before the synthesizer is ready to use. The synthesizer switches should be set as follows:

Echo GP:

The switches are located on the bottom of the synthesizer near the middle of the back. (The back is the side where the wires come out.) There is a square hole in the back of the synthesizer, and in this hole are four tiny switches. The part of each switch toward the back of the synthesizer should be pushed inward. This will set the synthesizer for 9600 baud.

Intextalker and Microvox:

The hardware of these two synthesizers is identical. The switches are located inside the synthesizer. Make sure the synthesizer is unplugged from the electric power line before attempting to change the switches! The first step is to turn the synthesizer upside down and remove the two long screws that hold the top on. One screw is located in the middle of the right and the left side of the bottom of the computer, just inside the edge. You will need a Phillips screwdriver to remove the screws. After removing the screws, turn the synthesizer right side up, and place it with side with the volume control knob facing you. There are two sets of eight switches that will have to be set. The first set is located half-way between the right and left edges of the circuit board, about one inch back from the front of the board. Switch number 4, which is the fourth switch from the rear of the board on this set of switches, should be pushed to the right, and the other switches should be pushed to the left. This will set the communications rate to 1200 baud. The second set of switches is located very near the right hand edge of the board, about three inches back from the front of the board. On this set of switches, Switches 1, 2, and 8 (the first, second and eighth switches from the rear of the board) should be set to the right, and the others should be set to the left. This will provide for hardware handshaking. If you have the Intex upgrade for improved pronunciation, switch settings may be different, as described in the instructions that come with this upgrade.

Votrax Personal Speech System:

Turn the synthesizer so the volume control is away from you. On the part of the synthesizer that is now toward you, about three inches leftward from the right side is an oblong hole, about a third of an inch high and an inch wide. A set of eight switches can be reached through this hole. The switches should all be pushed down, except switch 6 (the sixth from the left), which should be pushed up. This will set the synthesizer to 9600 baud, RTS serial port communication, 7-bit word with ignored parity bit, power-up message spoken, serial port used as primary input port, and self-test off.

INSTALLING THE SPEECH SYNTHESIZER

The synthesizer will have to be attached by cable to the cable to the COMM1 serial port on the IBM Personal Computer. If possible, you should buy the necessary cable from the synthesizer manufacturer. If this is not possible, you should be prepared to pay up to $100 to have a local computer store make a cable from the instructions in the IBM and synthesizer manuals. If the manufacturer provides a demonstration diskette, it should be used to test the speech synthesizer once it is installed. Users with physical disabilities will find it very convenient to plug the computer, speech synthesizer (if it has a separate plug), and the video monitor into a single power strip so that they may all be turned on with one switch.

INSTALLING THE INPUT DEVICES

Keyboard

The keyboard comes with the IBM computer, so no special purchase or installation is necessary if you plan to use the standard keyboard as the only input device. If a user cannot use the standard keyboard, because of poor motor control, he may be able to use the keyboard with a metal keyguard (These are available from companies supplying aids for the handciapped.)

Switches

Talkhelper may be used with either one or two switches. Two switches allow much better control and faster communication than one switch. The switches should be attached to an IBM game control adapter card plugged into the IBM computer. Switches, the necessary connectors, and installation instructions are available from companies specializing in devices for the physically handicapped.

CONFIGURING THE TALKHELPER SOFTWARE

Before configuring the Talkhelper software you should make several working copies of it. To make a working copies follow the instructions in the IBM DOS manual for copying a disks with the DISKCOPY command. Then put the original Talkhelper disk and one working copy in a safe place to use in case your other copies are damaged or wear out. Now you are ready to configure the working copy for a particular hardware configuration and a particular user. If different persons will be using Talkhelper, for instance various students in a special education classroom, a separate working copy of the disk will be have to be configured to meet the needs of each student. Configuration consists of indicating the kind of synthesizer that will be used.

If you have two floppy diskette drives, leave put a DOS diskette in drive A and insert the Talkhelper diskette into drive B. Type "b:" followed by the first letter of the name of your synthesizer. For Intextalker, type "i"; for Microvox, type "m"; for Echo GP, type "e"; for Votrax Personal Speech System, type "v".

If you have only one floppy diskette drive (with or without a hard disk drive), insert a DOS diskette inserted and type "b:" followed by the letter corresponding to your synthesizer as given above. Since the installation program needs to copy some DOS files onto your Talkhelper diskette, you will need to switch diskettes and press return each time the diskette drive stops, until the the computer prints "A>" to show that configuration is done.

DESIGNING SCREEN DISPLAYS

Once the computer has been configured for type of synthesizer, type of pronuciation, and input mode, it is necessary to design what will be displayed on the screen and what will be output from the computer to the speech synthesizer. The following discussion will first discuss the theory behind the various options for screen display for speech-handicapped persons, and their advantages and disadvantages, and then will indicate exactly how to design Talkhelper screen displays.

THEORY OF SCREEN DISPLAY

Screen display serves five purposes. First, it can present a list of alternatives to prompt the user who is unable to remember the effect of a complex set of keypresses or switch closures. Second, it can provide feedback on input errors, so that the user can correct them before a phrase is spoken. Third, it can enable users of stepping or scanning modes to coordinate their input actions with the stepping or scanning. Fourth, it can provide information to a person helping or training a user on what the user is trying to do. Fifth, it can provide necessary feedback and information during the process of selecting alternatives and editing menu selections.

The Screen Display as an Aid to Memory

The screen display aids the user who cannot remember the effects of various keypresses. This use of the screen display can be compared to the function of the menu in the typical Chinese restaurant in the United States. Usually this menu will contain a list of dishes, each with a number and a short name, e.g. "4. Sweet and Sour Pork." The customer can tell the waiter the numbers for the selections he wants and soon a delicious Chinese meal will be served. There are really three aspects to this menu. The name of the dish on the menu ("Sweet and Sour Pork") serves as a short description to remind the American customer of what dishes are available. The number serves as a means of communication with the Chinese waiter (who may not understand English). The waiter then may yell the order back to the kitchen in Chinese, "Please prepare one order of Sweet and Sour Pork." Similarly the screen display system of Talkhelper has three parts. Each item on the screen has three parts: (1) a designator -- a letter or a number that serves the same function as the number next to the name of the dish on the Chinese restaurant menu; (2) a label -- a word or phrase that serves the function of the name of the dish on the Chinese restaurant menu; (3) in the computer memory, but not normally appearing on the screen, the actual action that selecting the particular designator will cause the computer to make, corresponding to the action that will result from a request to a waiter in the restaurant.

Take the following example. There may be an entry on the screen that appears as follows:

W. Water

When the user presses the designator "W" that is next to the label "Water", the the action the computer will take is to say the corresponding phrase that has been previously entered, e.g., "I'm thirsty, please get me a glass of water."

The menu is necessary in a Chinese restaurant because most people cannot remember the huge variety of dishes in the Chinese cuisine, and because it provides a simple and fast method of communication even if the waiter does not understand English.. Likewise, once a computer is programmed to say a large number of phrases, a menu may be necessary to help the user remember all the phrases and to provide a fast and simple means of communication with the computer, which, like many foreign waiters, does not understand English. However one menu or one screen display may not be enough. In a restaurant, the menu might get too big if the owner tried to put everything on it. So the menu may say, "Please ask if you want to see the banquet menu, the cocktail menu, or the children's menu." Customers may then obtain these menus from the waiter. The need for such supplemental "menus" or displays is far greater on the computer, since a single screen display can hold much less information than a typical restaurant menu. Thus one of the actions the user must be able to command from the computer is to display additional screens of information with additional options.

Usually in a restaurant a patron will want to order a whole meal at once. Likewise someone using a computer as a speech aid may want to make a whole series of statements at once. Thus a mechanism is needed to allow the computer to save up several statements and say them all at once.

Screen Feedback on Input Errors

Whether a user is typing in words to be spoken, or using a screen designator and label system to select phrases, the user will undoubtedly make some errors. The screen display must show what words are typed or what phrases are selected so the user can see any errors and then must show corrections as they are made.

Coordination of Input Actions with Item Selection In Scanning and Stepping Modes

In scanning and stepping modes, the screen display provides coordination of user input actions with selection of items. For instance, in a single switch system, a cursor or spot of light may move from item to item on the screen. The user needs the information on the screen to be able to close the switch at the right time to select a particular phrase. In stepping mode, the user closes one switch to step the cursor from item to item and then presses a different switch to select a phrase. Here again, screen feedback is needed to allow the user to see which phrase is currently indicated by the cursor.

Information for the Person Helping or Training the User

Some users may need considerable help in learning to use Talkhelper. The screen display will enable persons helping users to see exactly what the user is doing, and thus to provide the necessary assistance.

HOW TO USE TALKHELPER

The Sample Programs

There are sample text and picture menu programs on the Talkhelper Diskette called sample. You should read these instructions through and then work for a while with the programs called text and picture before you try to create your own set of menus. To run the text menu program, just type "text" and press the ENTER key. To run the picture menu program, just type "picture" and press the ENTER key.

After you have practiced with these programs, you will be ready to create and edit menus for a particular user.

Every user will have different needs in terms of what the user wants to say and the type of labels the user needs as memory aids in order to be able to make selections. Talkhelper is not based upon any preconceived model of what the user should want to say or how much memory assistance the user needs. Rather it is left to the user or someone helping the user to create memory aids and phrases to be spoken that are appropriate to the particular user.

Each menu has a sets of what we will call labels, designators, and associated phrases or actions. When a text menu is displayed on the screen, the display will look something like the following (actually there will usually be more entries):

HOME

A.	Open Door	D.	Open Window
B.	Close Door	E.	Close Window
C.	Upstairs	F.	Downstairs

The word "HOME" at the top of the screen is the name of the menu. We call the letters A, B, C, D, E, F designators. They are used for selection of particular menu entries. The short text items, "Open Door," "Close Door", etc., are called labels. They serve to remind the user of what will happen if a particular entry is selected. Finally, associated with each entry is a phrase or action that is not ordinarily visible on the screen. Thus for instance, assocated with "A. Open Door" may be the phrase, "Please open the door." Associated with "C. Downstairs" may be the action command *DOWNSTAIRS, which instructs the computer to display a different menu called "DOWNSTAIRS" if entry "C" is selected.

The editing process allows creation of new menus, the changing of old menus, and setting of various items affecting all the menus.

Picture menus are exactly like text menus, except that a picture is associated with each picture menu and the picture is normally displayed whenever the menu is accessed. To edit a picture menu, hold down the control key and press "E" (this is called typing "Control-E"). The text menu with which picture is associated will then appear on the screen for editing. To go back to the picture, hold down the control key and press "R" (this is called typing "Control-R"). The picture will again be displayed.

Menu Operations

The user may also wish to take actions affecting the menus as a whole or a particular menu. To take such actions, press the slash ("/") key and a list of alternatives will appear on the screen. The alternatives are: adding a menu, deleting a menu, editing a menu entry, input mode selection, number of entries selection, renaming a menu, scanning speed selection, writing out all menus, doing nothing. Each of these will be discussed in turn.

Adding a Menu

New menus may be added. However there may not be more than a total of 35 menus. To add a menu, first press slash ("/") and then the letter "a" or "A" (capital or small letters always have the same effect). The computer will then ask for the name of the new menu. If there is some problem, for instance if a menu with the same name already exists, an appropriate message will appear on the screen. If there is no problem, you will be asked the number of entries you want in the menu.

Before trying to display a picture menu, you must have put on the diskette a file containing a picture. The file must have the same name as that you plan to give to the picture menu. Details of how to create the picture file will be discussed below under "Picture Creation."

Deleting a Menu

A menu may no longer be needed. To get rid of a menu, press first slash ("/") and then the letter "d". Instructions for deleting a menu will appear on the screen. If there is an error, for instance an attempt to delete a menu that does not exist, or an attempt to delete the menu called MAIN, which cannot be deleted, an appropriate message will appear on the screen. There always must be a menu named "MAIN", which will be the first menu to appear on the screen when the program is run. If you want to delete the current "MAIN" menu, the program will ask you for the name of some other menu to take its place. WARNING!!!!!! If a menu is deleted it is gone. Be careful about deleting a menu. If you delete a menu by mistake, no harm is done as long as you do not write out the menus to the diskette, provided that the menu deleted was on the diskette when you turned on the computer. Just take out the diskette and turn off the computer and all the changes made will be forgotten. To be safe, before editing manus, make backup copies of diskettes containing important menus.

Editing a Menu Entry

To enter or change a label and the corresponding phrase or action the user should press slash ("/") and then "e". (As a shortcut, the user can press period ("."). This has the same effect as pressing slash and then "e".) Then the user should select the designator for which the user wishes to supply information. The user will then be asked for a label to appear after the designator and for a phrase or action to be selected when the designator is pressed. The label can be up to 30 characters long. The phrase can be as short as a single letter or number or as long as 75 characters. A user capable of typing on a regular keyboard should enter the phrase or action just by typing and then pressing the ENTER key. A special scanning and stepping entry mode (described below) is available for uses who cannot use a regular keyboard.

In addition to generating a phrase to be spoken, an entry can be used to command the computer to take the action of plotting another menu on the screen. This is done by typing as the desired action an asterisk ("*") followed by the name of the menu to be put on the screen. Thus if the user wanted the result of pressing M to be to put the menu called MATH on the screen the user would type: *MATH as the action to be taken.

Text menus for use in scanning and stepping modes should be set up with the following five entries as the first three entries, since the user will not be able to select the necessary special keys by directly typing on the keyboard:

A. ENTER (the corresponding text should be "&ENTER")

B. ESCAPE (the corresponding text should be "&!ESCAPE")

C. BACKSPACE (the corresponding text should be "&BACKSPACE")

D. SLASH (the corresponding text should be "&/")

E. PERIOD (the corresponding text should be "&.")

Input Mode Selection

The system provides 5 basic and 5 supplemental input modes. The basic input modes are: (1) normal keyboard input; (2) scanned input using the keyboard as a switch; (3) scanned input using a switch attached to a game control adapter card; (4) stepped input using the space key as one switch and the other keys as a second switch; (5) stepped input using two switches attached to a game control adapter card. The supplemental input modes all use a list of letters and charaters displayed on the screen. They vary by method of selection: (1) normal keyboard typing; (2) scanned input with the keyboard as a switch; (3) scanned input using a single switch attached to a game control adapter card; (4) stepped input with the space bar as one switch and the other keys as a second switch; (5) stepped input using two switches attached to a game control adapter card.

The basic input modes are used to make selections from a menu. The supplemental mode is used to allow a handicapped user to type in entries to edit menus and select program features. To select an input mode for a menu, type slash ("/") and then "i" and follow the instructions that appear on the screen. The supplemental mode for the menu will be set to correspond to the basic mode. Thus, for instance, if scanned input is selected for a menu, supplemental input will be by scanning a list of characters displayed on the screen.

Renaming a Menu

You may want to change the name of a menu. For instance you might want to change the name of the menu called "PHYSICS" to "SCIENCE". Press slash ("/") and then press "r" and follow the instructions to rename the menu. Since there must always be a menu named "MAIN," if you try to rename "MAIN," you will be asked for the name of another menu to substitute for "MAIN."

Scanning Rate Selection

The speed with which menus should be scanned depends both on the degree of handicap of the user and the user's experience with the system. Permissible speeds range from 1 (very slow) to 30 (very fast). When the program begins, the scanning rate is set at rate 1. Many users will want to select a faster rate. An appropriate speed can be found only by trial and error. Obviously it is best to start with a slow speed and then adjust the speed faster as the user's skill develops. To select a speed for scanning, type slash ("/") and then "s" and follow the instructions that appear on the screen. When the menus are written out and saved, the scanning rate associated with them will also be saved.

Number of Menu Entries

The user can choose the number of entries for each menu. For a text menu, the maximum number of entries is 34. As explained below in the section on creating pictures, pictures may have 4, 9, 16, or 25 frames. To select the number of entries for a menu, type slash ("/") and then "n". The computer will ask the number of entries desired. Type in the appropriate number and press return. For text menus, numbers below 1 or above 34 will not be accepted. For picture menus, numbers other than 4, 9, 16, or 25 will not be accepted -- the computer will keep asking until it is given a "legal" number. WARNING -- if you reduce the number of entries in a menu, you will lose the last entries in that menu. For instance if a menu had length 34 and you reduce the length to 24, you will lose anything you have in menus 25 through 34.

Type of Menus

You may choose to have all menus be text menus or all menus be picture menus. To change the type of menus, press slash ("/") and then "t". If you change menus to picture menus, all menus must be legal lengths for picture menus.

Write Out All Menus

Once all desired editing changes have been made, you may want to write all the menus out to a diskette. If you write them to a diskette that already has menus on it you will destroy those menus. Thus if you want to save all your old menus, put a fresh backup copy of your Talkhelper diskette into drive A when you give the instruction to write out menus. WARNING!!! If you have removed all references to a menu, it will be lost when the menus are written out, unless you add at least one reference to the menu before writing out the menus. To write out all the menus, press slash ("/") and type "w".

Do Nothing

If you press slash ("/") by accident or if you change your mind and do not want to take any action to with respect to menus, press "x" to return to where you were before.

SPECIAL EDITING MODES

Stepping or Scanning Keylist

A handicapped user may want to edit his own menus, but may lack the physical capacity to use the keyboard, and so will have to use the scanning or stepping mode.

Sometimes a handicapped user will lack the physical or mental ability to edit his own menus. This will sometimes be the case with respect to text menus and will be assumed to always be the case with respect to picture menus. In this case a helper will be needed to enter the menus. The helper will want to use typed input and in editing the text to be spoken and actions to be taken with a picture menu will want to see and edit typed information on the screen rather than a picture. There is a special command to allow this type of editing. While holding down the Control Key, press E (for Edit). This will convert the entry mode to typing and will display the associated text if a picture menu is accessed. To return to the normal settings, while holding down the Control Key, press R (for Regular).

CREATING PICTURES

As mentioned above, a menu may have an associated picture that is displayed on the screen when the menu is accessed. In this case the menu is not normally seen (but it always can be accessed by pressing Control-E as described above). Pictures are edited with a "mouse" and picture-drawing software. A mouse is a small device that when moved around on a table causes its position to be input to the computer. Picture-drawing software works with a mouse to facilitate drawing and editing pictures. Pictures can originally be created either with a mouse and picture-drawing software or with a television camera attached to the computer. (Mouse Systems and Microsoft each sell a mouse for the IBM Personal Computer; Koala sells a picture drawing tablet and software; Micron Technology and Micromint sell digital television cameras; Chorus sells interfaces for ordinary television cameras. All these companies advertise regularly in Byte and in magazines for IBM computer owners.)

Unfortunately, handicapped persons who need picture menus generally lack the physical coordination and cognitive ability needed to create pictures. Therefore it is assumed that picture menus will be created by a parent, teacher, or other helper. The first step is to become thoroughly aquainted with the mouse and picture-drawing software by reading the manual that accompanies them and going through the introductory information in the manual. For each picture menu, the helper should decide on the number of picture frames to appaear on the screen. Options available are: 2 x 2 (4 squares), 3 x 3 (9 squares), 4 x 4 (16 squares), and 5 x 5 (25 squares). On the Talkhelper disk are four ready made picture templates, P4, P9, P16, P25, corresponding to these options. The helper should load the appropriate template into the picture-drawing software system. Pictures may be drawn into the squares using the mouse and the picture-drawing software. Pictures may be read in from other picture files already on the disk, using the "cut and paste" option of the picture-drawing software to transfer them to the template. Words may be added to the template in combination with pictures or separate from them, depending upon the reading ability of the user. (The picture-drawing software systems provide very convenient facilities for adding words to pictures.)

The resulting picture should be saved on a working copy of the Talkhelper diskette with a name identical to that of the menus with which it is to be associated.

An alternative to drawing the pictures is to photograph them with a television camera attachment and store them on a diskette. The mouse and the picture-drawing software system can then be used with the "cut and past" option to combine pieces of these photographs into the final picture.

USING TALKHELPER AS A SPEECH AID

SELECTION OF PHRASE OR ACTION WITH A TEXT MENU

Full Keyboard Selection Mode

To select the phrase or action corresponding to a label on the screen press the key corresponding to the letter or number to the left of the label. Phrases selected will gradually build up on the screen. To remove the most recent phrase selected, press the BACKSPACE KEY. To remove all the phrases from the screen, press the ESCAPE key. To remove all text press the ENTER key.

Keyboard Stepping Mode

When it reaches the desired menu entry, press any key other than the space bar. To get the effects of the BACKSPACE, ESCAPE, and

ENTER keys, select the menu items with those names.

Two Switch Stepping Mode

An asterisk will appear above the left hand column of menu entries. To select the right hand column, press the first switch, and the asterisk will move to above the right hand column. (If you want to go back to the left hand column, again press the first switch again.) When the asterisk is above the column you want, press the second switch bar to move the asterisk down to the first entry. Now you can press the second switch from menu entry to menu entry. When it reaches the desired menu entry, press the second switch. To get the effects of the BACKSPACE, ESCAPE, ENTER, SLASH, and PERIOD keys, select the menu items with those names.

Keyboard Scanning Mode

An asterisk will appear above the left hand column of menu entries. To select the right hand column, promptly press any key and the asterisk will move above the right hand column. (If you want to go back to the left hand column, again promptly press any key other than the space bar.) when the asterisk is above the column you want, wait, and it will begin to go downward from entry to entry, pausing at each entry. While it is next to the entry you want, press any key to select that entry. To get the effects of the BACKSPACE, ESCAPE, and ENTER keys, select the menu items with those names.

Single Switch Scanning Mode

An asterisk will appar above the left hand columnof menu entries. To select the right hand column, promptly press the switch and the asterisk will move above the right hand column. (If you want to go back to the left hand column again, promptly press the switch.) When the asterisk is above the column you want, wait, and it will begin to go downward from entry to entry, pausing at each entry. While it is next to the entry you want, press the switch to select that entry. To get the effects of the BACKSPACE, ESCAPE, and ENTER keys, select the menu items with those names.

SELECTION OF PHRASE OR ACTION WITH A PICTURE MENU

Full Keyboard Selection Mode

To select the phrase or action corresponding to picture frame on the screen, press the key corresponding to the letter in the frame. If a phrase is selected it will be spoken. If a new menu is selected, the program will put that menu on the screen.

Keyboard Stepping Mode

A lighted inner frame will flash inside the first picture frame. Press the space bar to move the lighted frame from frame to frame. When it reaches the desired frame, press any key other than the space bar.

Two Switch Stepping Mode

A lighted inner frame will flash inside the first picture frame. Press the first switch to move the lighted inner frame from frame to frame. When it reaches the desired frame, press the second switch.

Keyboard Scanning Mode

A lighted inner frame will jump from frame to frame, pausing at each frame. While it is around the entry you want, press any key to select that entry.

Single Switch Scanning Mode

A lighted inner frame will jump from frame to frame, pausing at each frame. While it is around the entry you want, press the switch to select that entry.

SUPPLEMENTAL ENTRY MODES

The following supplemental entry modes are available to let a handicapped person unable to use the normal keyboard make entries using a scanning or stepping mode. They are available only with text menus and are automatically actived if the text menu is in the corresponding mode. When the computer asks a question as part of the editing process, the following list of keys will appear on the bottom of the screen:

```
  ? < * ! & 0 1 2 3 4 5 6 7 8 9
  E T A O I N H N S R L U D Y W G M C B F K P V J X Z
```

If the menu in one of the stepping modes, a stepping cursor will appear at the left of the first row of keys. If it is in a scanning mode, a scanning cursor will appear at the left of the first row of of keys. The operation of stepping and scanning modes are exactly as described above, except that instead of choosing between vertical columns, the user can choose between horizontal rows. Selecting a letter, number, symbol is the same as typing the corresponding key, except that selecting "?" is equivalent to pressing the BACKSPACE key, selecting "<" is equivalent to pressing the ENTER key, and selecting "!" is equivalent to pressing the ESCAPE key.

TALKHELPER

Software to Adapt the Apple IIe Personal Computer for Use by Persons with Speech Disabilities

USERS' MANUAL

This manual and the accompanying software were developed under United States Department of Education Contract No. 300-83-0271

Visek & Maggs
608 W. Pennsylvania Ave.
Urbana, IL 61801

TABLE OF CONTENTS

INTRODUCTION

Talkhelper is a program designed to allow non-speaking persons to use a personal computer with a speech synthesizer as a substitute voice. Many non-speaking persons have other physical or mental handicaps. Talkhelper offers flexibility in selection of input devices and screen displays so that it may be customized for each user to make maximum use of his or her abilities and to compensate for his or her disabilities.

This manual is for use with Talkhelper for the Apple Computer. (Other versions of Talkhelper are available for the IBM Personal Computer and the Radio Shack Model 100 portable computer.)

PURPOSES OF THIS MANUAL

This manual has four purposes:

1. To tell you how to set up the computer, speech synthesizer, and input device for use with the Talkhelper software;
2. To help you configure the Talkhelper software for the computer, synthesizer, input device, and options you want;
3. To explain how to create and edit screen menus;
4. To explain how to use Talkhelper as a speech aid.

DO I HAVE TO READ ALL OF THIS MANUAL?

If you are planning to install the speech synthesizer, input devices and Talkhelper yourself, you should study the sections of this manual on installation before you try to install the synthesizer, input devices, and Talkhelper. If someone else has installed the synthesizer, input devices, and Talkhelper for you, you can ignore these sections of the manual.

All of the rest of the manual is important, regardless of what synthesizer you use and regardless of who installs it.

SELECTION OF COMPUTER, SPEECH SYNTHESIZER, AND INPUT DEVICES

COMPUTER SELECTION

This software is designed to function with the Apple IIe Personal Computer. Unless you have considerable computer expertise, you should buy a new computer from a local authorized Apple dealer so that you can get qualified help in setting up the computer and warranty repairs if necessary. If you do have computer expertise, you may be able to save money by ordering a computer by mail or by buying a second-hand computer.

All models of the Apple IIe Personal Computer come equipped with the diskette drive and memory needed for this program. A color monitor is desirable, but not necessary for picture menu displays -- you can use a black and white, green screen, or amber screen monitor with picture menus.

SPEECH SYNTHESIZER SELECTION

You should use the Street Electronics Echo synthesizer for the Apple IIe.

INPUT DEVICE SELECTION

Some persons with speech handicaps can use the computer keyboard very well. However, since many persons with speech handicaps also have other disabilities which prevent them from typing on the regular computer keyboard, Talkhelper allows the use of different input devices so that each user can use the form of input best adapted to his or her disabilities.

The possible devices are: the regular Apple IIe Personal Computer keyboard or special switches connected to the joystick connector on the computer. The following discussion is meant to help you choose an input device. Often it is obvious which device should be used. Sometimes it is difficult to determine which device is best, and it may be necessary to try several types of input device and to seek the advice of an appropriate health care or education professional. Some users may be able to advance to more complex input devices that provide faster control of the computer. The discussion below will consider factors in choice of input devices for users with speech disabilities only, or those with speech disabilities and with poor motor control, poor vision, poor or no reading ability, or poor memory. The various possible types of input device and the ways in which the can be used will be considered in turn.

COMPUTER KEYBOARD INPUT

The computer keyboard has two great advantages and two great disadvantages. The two great advantages are that a keyboard comes at no extra charge with every Apple IIe Personal Computer and that a good typist can enter any English phrase at high speed using the keyboard. The two great disadvantages are that the keyboard is hard to use for persons with physical disabilities and hard to understand for persons with cognitive disabilities. Talkhelper offers several ways to use the keyboard to allow customization for different types of users.

Full Keyboard Input

Full keyboard input is ideal for a person with speech disabilities, but no other disabilities. With a relatively small investment of time such a person should be able to learn typing, a skill with many applications, and should be able to reach a typing speed of over 50 words a minute. Such a person will be able to use Talkhelper to say anything he or she wants at a speed of at least 50 words a minute, as compared to a normal speech rate of 80 to 100 words a minute.

Full keyboard input may also be possible for a person with a moderate level of physical disability. The disability may reduce the accuracy with which keys are pressed, the speed with which they can be reached, or may prevent use of the shift key. Even with these problems, it may be possible to get satisfactory results from Talkhelper. Often accuracy can be improved substantially by the use of a keyguard or an oversized keyboard. (These are available from companies supplying computer products for handicapped users.) If the disability substantially reduces the speed with which keys can be selected, the user will probably wish to supplment typed text with phrase selection input (see the section below entitled "Phrase Selection").

Restricted Keyboard Input

Users with poor accuracy, limited reach, or inability to understand the complexities of a full typewriter keyboard may benefit from an option of Talkhelper that allows the assigning of the whole keyboard to be treated as a single switch or of the spacebar and the rest of the keyboard each to be considered as a switch.

SWITCH INPUT

Switch input also has advantages and disadvantages. The advantages of switches are that they are easy to buy and attach to the computer and that they can be used by persons with physical handicaps that prevent pressing keys on the computer keyboard. One disadvantage is that input is much slower than with full keyboard typing. Another disadvantage is the difficulty users with limited cognitive ability may have in associating closing switches with what is happening on the screen.

Talkhelper can be configured to work with either one or two switches. The two-switch input mode offers better control and faster input than the one-switch mode. Therefore it is the preferred mode for persons who can operate more than one switch.

INSTALLING THE SPEECH SYNTHESIZER

The synthesizer should be placed in a slot in the Apple IIe in accordance with the instructions in the synthesizer manual. It should be tested using the software provided by the synthesizer manufacturer.

INSTALLING THE INPUT DEVICES

Keyboard

The keyboard comes with the Apple IIe computer, so no special purchase or installation is necessary if you plan to use the standard keyboard as the only input device. If a user cannot use the standard keyboard, because of poor motor control, he may be able to use the keyboard with a metal keyguard (These are available from companies supplying aids for the handciapped.)

Switches

Talkhelper may be used with either one or two switches. Two switches allow much better control and faster communication than one switch. The switches should be attached to the Apple IIe joystick port in accordance with the instructions provided by the switch manufacturer. Switches and connecting cables are available from companies specializing in devices for the physically handicapped.

MAKING WORKING COPIES OF THE TALKHELPER SOFTWARE

Before configuring the Talkhelper software you should make several working copies of it. To make a working copies follow the instructions in the Apple DOS manual for copying a disks with the COPYA program. Then put the original Talkhelper disk and one working copy in a safe place to use in case your other copies are damaged or wear out. Now you are ready to configure the working copy for particular hardware and a particular user. If different persons will be using Talkhelper, for instance various students in a special education classroom, a separate working copy of the disk will be have to be configured to meet the needs of each student. The following discussion will aid you in deciding how to configure the software for suers with particular handicaps.

DESIGNING SCREEN DISPLAYS

Once the computer has been configured for type of synthesizer, type of pronunciation, and input mode, it is necessary to design what will be displayed on the screen and what will be output from the computer to the speech synthesizer. The following discussion will first discuss the theory behind the various options for screen display for speech-handicapped persons, and their advantages and disadvantages, and then will indicate exactly how to design Talkhelper screen displays.

THEORY OF SCREEN DISPLAY

Screen display serves five purposes. First, it can present a list of alternatives to prompt the user who is unable to remember the effect of a complex set of keypresses or switch closures. Second, it can enable users of stepping or scanning modes to coordinate their input actions with the stepping or scanning. Third, it can provide information to a person helping or training a user on what the user is trying to do. Fourth, it can provide necessary feedback and information during the process of selecting alternatives and editing menu selections.

The Screen Display as an Aid to Memory

The screen display aids the user who can ot remember the effects of various keypresses. This use of t e screen display can be compared to the function of the menu i the typical Chinese restaurant in the United States. Usu lly this menu will contain a list of dishes, each with a number nd a short name, e.g. "4. Sweet and Sour Pork." The customer an tell the waiter the numbers for the selections he wants and s on a delicious Chinese meal will be served. There are reall three aspects to this menu. The name of the dish on the menu "Sweet and Sour Pork") serves as a short description to remin the American customer of what dishes are available. The n mber serves as a means of communication with the Chinese waite (who may not understand English).. The waiter then may yel the order back to the kitchen in Chinese, "Please prepare one o der of Sweet and Sour Pork." Similarly the screen display sys em of Talkhelper has three parts. Each item on the screen ha three parts: (1) a designator -- a letter or a number that ser es the same function as the number next to the name of th dish on the Chinese restaurant menu; (2) a label -- a w rd or phrase that serves the function of the name of the dish o the Chinese restaurant menu; (3) in the computer memory, but not normally appearing on the screen, the actual action th t selecting the particular designator will cause the computer to make, corresponding to the action that will result rom a request to a waiter in the restaurant.

Take the following example. There may b an entry on the screen that appears as a picture as a glass of milk and in one corner the label:

M. Milk

When the user presses the designator "M" that is next to the label "Milk", and on the picture of the glass of milk the the action the computer will take is to say the corresponding phrase that has been previously entered, e.g., "I'm thirsty, please get me a glass of milk."

The menu is necessary in a Chinese restaurant because most people cannot remember the huge variety of dishes in the Chinese cuisine, and because it provides a simple and fast method of communication even if the waiter does not understand English.. Likewise, once a computer is programmed to say a large number of phrases, a menu may be necessary to help the user remember all the phrases and to provide a fast and simple means of communication with the computer, which, like many foreign waiters, does not understand English. However one menu or one screen display may not be enough. In a restaurant, the menu might get too big if the owner tried to put everything on it. So the menu may say, "Please ask if you want to see the banquet menu, the cocktail menu, or the children's menu." Customers may then obtain these menus from the waiter. The need for such supplemental "menus" or displays is far greater on the computer, since a single screen display can hold much less information than a typical restaurant menu. Thus one of the actions the user must be able to command from the computer is to display additional screens of information with additional options.

Coordination of Input Actions with Item Selection In Scanning and Stepping Modes

In scanning and stepping modes, the screen display provides coordination of user input actions with selection of items. For instance, in a single switch system, a lighted frame may move from item to item on the screen. The user needs the information on the screen to be able to close the switch at the right time to select a particular phrase. In stepping mode, the user closes one switch to step the frame from item to item and then presses a different switch to select a phrase. Here again, screen feedback is needed to allow the user to see which phrase is currently indicated by the frame.

Information for the Person Helping or Training the User

Some users may need considerable help in learning to use Talkhelper. The screen display will enable persons helping users to see exactly what the user is doing, and thus to provide the necessary assistance.

HOW TO USE TALKHELPER

The Sample Programs

There are sample menus programs on the Talkhelper Diskette. You should read these instructions through and then work for a while with the sample menus before you try to create your own set of menus. To run TALKHELPER, just type RUN TH and press the RETURN key.

After you have practiced with these menus, you will be ready to create and edit menus for a particular user.

Every user will have different needs in terms of what the user wants to say and the type of pictures and labels the user needs as memory aids in order to be able to make selections. Talkhelper is not based upon any preconceived model of what the user should want to say or how much memory assistance the user needs. Rather it is left to the user or someone helping the user to create memory aids and phrases to be spoken that are appropriate to the particular user.

Editing Menus

To create a new set of menus or alter an existing set, first use the COPYA program on the DOS 3.3 diskette to make a copy of your working copy of the Talkhelper master diskettes (for a new set of menus), or make a copy of the diskettes with the set of menus you wish to alter. Now type RUN EDIT and press RETURN.

Each set of menus has two settings that apply to the whole set. These are the number of frames (or boxes) per screen and the type of input. In addition, scanning menus have a scanning speed setting. The first step with the EDIT program is to answer questions about these settings.

If you have a 4 frame per screen system, there will be an initial menu display with 4 frames. These can be used to select 4 submenus, each of which will also have four frames. These submenus are called "A", "B", "C", and "D". Follow the instructions to choose a submenu. Each submenu will have four phrases to be spoken, one associated with each frame as follows:

```
A   B

C   D
```

Choose the line of text you wish to edit and enter its new value.

If you have a 9 frame per screen system, there will be an initial menu display with 9 frames. These can be used to select 9 submenus, each of which will also have four frames. These submenus are called "A", "B", "C", "D", "E", "F", "G", "H", and "I". Follow the instructions to choose a submenu. Each submenu will have nine phrases to be spoken, one associated with each frame as follows:

A	B	C
D	E	F
G	H	I

Choose the line of text you wish to edit and enter its new value.

If you have a 16 frame per screen system, there will be an initial menu display with 16 frames. These can be used to select 16 submenus, each of which will also have four frames. These submenus are called "A", "B", "C", "D". "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", and "P". Follow the instructions to choose a submenu. Each submenu will have sixteen phrases to be spoken, one associated with each frame as follows:

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

Choose the line of text you wish to edit and enter its new value.

CREATING PICTURES

Pictures are best edited with the "Apple Mouse", the Gibson Light Pen", or the "Koala Pad" and the picture-drawing software supplied with them. Pictures can originally be created either with these devices and their picture-drawing software or with a Micron Eye television camera attached to the computer

Unfortunately, handicapped persons who need picture menus generally lack the physical coordination and cognitive ability needed to create pictures. Therefore it is assumed that picture menus will be create by a parent, teacher, or other helper. The first step is to become thoroughly aquainted with the Gibson Light Pen or the Koala Pad and the picture-drawing software by reading the manual that accompanies them and going through the introductory information in the manual. For each picture menu, the helper should decide on the number of picture frames to appaear on the screen. Options available are: 2 x 2 (4 squares), 3 x 3 (9 squares), and 4 x 4 (16 squares). On the Talkhelper disk are four ready made picture templates, $\overset{\text{MOD}}{\cancel{\text{P}}}$4, $\overset{\text{MOD}}{\cancel{\text{P}}}$9, and $\overset{\text{MOD}}{\cancel{\text{P}}}$16, corresponding to these options. The helper should load the appropriate template into the picture-drawing software system, following the instructions in the picture-drawing software manual for loading a screen image. (The pictures may then be drawn into the squares using Apple Mouse, the Gibson Light-Pen, or the Koala Pad and the picture-drawing software. Pictures may be read in from other picture files already on the disk, using the "cut and paste" option of the picture-drawing software to transfer them to the template. Words may be added to the template in combination with pictures or separate from them, depending upon the reading ability of the user. (The picture-drawing software systems provide very convenient facilities for adding words to pictures.) If full keyboard input is to be used, it is important to put a letter ("A", "B", "C", "D", etc., in each frame corresponding to the key that will need to be pressed to say the message represented by the picture in the frame.

The resulting picture should be saved on a working copy of the Talkhelper diskette in a file named with a letter identical to that of the menu with which it is to be associated.

An alternative to drawing the pictures is to photograph them with a television camera attachment and store them on a diskette. The Gibson Light Pen or the Koala Pad and the picture-drawing software system can then be used with the "cut and paste" option to combine pieces of these photographs into the final picture.

USING TALKHELPER AS A SPEECH AID

SELECTION OF PHRASE OR MENU

Full Keyboard Selection Mode

To select the phrase or menu corresponding to picture frame on the screen, press the key corresponding to the letter in the frame. If a phrase is selected it will be spoken. If a new menu is selected, the program will put that menu on the screen.

Keyboard Stepping Mode

A lighted inner frame will flash inside the first picture frame. Press the space bar to move the lighted frame from frame to frame. When it reaches the desired frame, press any key other than the space bar.

Two Switch Stepping Mode

A lighted inner frame will flash inside the first picture frame. Press the first switch to move the lighted inner frame from frame to frame. When it reaches the desired frame, press the second switch.

Keyboard Scanning Mode

A lighted inner frame will jump from frame to frame, pausing at each frame. While it is around the entry you want, press any key to select that entry.

Single Switch Scanning Mode

A lighted inner frame will jump from frame to frame, pausing at each frame. While it is around the entry you want, press the switch to select that entry.

TALKHELPER

Software to Adapt the TRS-80 Model 100 Personal Computer for Use by Persons with Speech Disabilities

USER'S MANUAL

This manual and the accompanying software were developed under United States Department of Education Contract No. 300-83-0271

Visek & Maggs
608 W. Pennsylvania Ave.
Urbana, IL 61801

TABLE OF CONTENTS

INTRODUCTION

Talkhelper is a program designed to allow non-speaking persons to use a personal computer with a speech synthesizer as a substitute voice.

This manual is for use with Talkhelper for the TRS-80 Model 100 Portable Computer. (Other versions of Talkhelper are available for the IBM Personal Computer and the Apple IIe Personal Computer.)

THE COMPUTER AND THE SYNTHESIZER

THE COMPUTER

This software is designed to function with the Radio Shack TRS-80 Model 100 Portable Computer. Unless you have considerable computer expertise, you should buy a new computer from a local authorized Radio Shack Dealer so that you can get qualified help in setting up the computer and warranty repairs if necessary. If you do have computer expertise, you may be able to save money by ordering a computer by mail or by buying a second-hand computer.

To be used with this program, the TRS-80 Portable Computer must have 32K of RAM memory. (Any Radio Shack computer dealer will be happy to sell and install additional memory, if necessary to bring the total memory up to 32K.)

THE SPEECH SYNTHESIZER

This software requires a special speech synthesizer for the TRS-80 Model 100. You should follow the instructions supplied with the synthesizer to install it.

MAKING WORKING COPIES OF THE TALKHELPER SOFTWARE

Before configuring the Talkhelper software you should make several working copies of it. Talkhelper is supplied on two cassettes. One contains a copy of a program called TALK. The other contains a copy of a program called HELP.

To make copies of TALK place the TALK cassette in the cassette recorder, ready the recorder, select BASIC from the menu, then type:

CLEAR 45000,256 <ENTER>

LOADM "TALK"

When TALK is successfully loaded insert and rewind a blank data cassette in the recorder and type:

SAVEM "TALK",45~~000~~056,58181

To make an additional backup copy, once again insert and rewind a blank data cassette in the recorder and type:

SAVEM "TALK",45~~000~~056,58181

To make copies of HELP, place the HELP casette in the cassette recorder, ready the recorder, select BASIC from the menu, then type:

LOAD "HELP"

When HELP is successfully loaded, insert and rewind a blank data cassette in the recorder and type:

SAVE "HELP"

To make an additional backup copy, once again insert and rewind a blank data cassette in the recorder and type:

SAVE "HELP"

USING TALKHELPER AS A SPEECH AID

STARTUP

To start Talkhelper, you must first load the TALK, the Talking BASIC System from the cassette. The Talking BASIC System is designed to allow you to run talking programs written in BASIC and to write talking programs in BASIC. To load the Talking BASIC System, type:

```
CLEAR 45000,256
```

and then, being sure that the diskette with the TALK program is ready in your cassette recorder, type:

```
RUNM "TALK"
```

At this point the Talking BASIC Program is loaded, but not activated. Special talking programs for the Model 100 will activate it automatically. Details on how you can write your own talking programs are given in the Appendix.

To run Talkhelper, once TALK is loaded, put the rewound cassette with the HELP program in the recorder and type:

```
RUN "HELP"
```

The computer is now ready to function as a speech aid. The Talkhelper program may be left functioning indefinitely, by keeping the computer plugged into a 110 volt house current adapter when it is not being carried around.

To use Talkhelper, merely type in any English word or phrase and press the ENTER key. If you wish to say the phrase again, just press the ENTER key.

APPENDIX

WRITING TALKING PROGRAMS

Once Talking BASIC is loaded, you may activate it in your own BASIC programs by executing the following command:

```
CALL 45059
```

When Talking BASIC is activated, anything printed on the scren will be spoken.

(You may deactivate the Talking BASIC at any time by executing:

```
CALL 45062
```

You may then reactivate it at any time by executing:

```
CALL 45059
```

Warning -- you must deactivate Talking BASIC before leaving BASIC to go to the main menu or options on the main menu will not work. If you accidently exit to the main menu with Talking BASIC activated, you must restart the computer by turning off the master power switch on the bottom of the computer, removing any 110 volt adapter, waiting a minute, and then turning the computer on again.

SPEECH OPTIONS

The standard features of Talking BASIC attempt to imitate what a person would say if he were asked to read what was being written to the screen. However, there are a number of options to alter how text is read from the screen.

Spelling or Whole Words

When Talking BASIC is activated, it reads by whole words. some uses, you may want words spelled letter by letter. To have words spelled out, type CALL 45065 To return to normal whole word pronunciation press type CALL 45068.

Remember: CALL 45065 for letter by letter spelling;
CALL 45068 for whole word pronunciation.

Punctuation

In normal reading, special symbols such as "$" and "+" are usually read out, but common punctuation marks such as period and comma are not pronounced. When Talking BASIC starts, it is in normal reading mode. All special symbols and all punctuation on the screen are pronounced except the following: dash, period, comma, semicolon, exclamation point, question mark, star, double quotes, apostrophe, tilde, grave accent, vertical line, backslash, left parenthesis, right parenthesis, left bracket, right bracket, left brace, and right brace. When reading numbers, a decimal point will be pronounced as point. If a combination of digits, dollar signs, commas, and periods is encountered that is not a well-formed number or monetary amount, all characters will be pronounced to alert the user to the unusual situation.

This option is called "Some" punctuation. If you have switched to another punctuation option, you can return to "Some" punctuation by typing CALL 45074.

When proofreading, it is usual to pronounce all punctuation but not to indicate spaces. This option is called "Most" punctuation. It is available by typing CALL 45077.

Finally, for some purposes it is necessary to know exactly what is on the screen, including spaces. This option, called "All" punctuation, is available by typing CALL 45071.

Remember: CALL 45074 is for some punctuation;
CALL 45077 is for most punctuation;
CALL 45071 is for all punctuation.

SOURCE CODE FOR SCREEN-VOICE FOR THE IBM PERSONAL COMPUTER

```
/* file:  DEFS.C
   last edited by pm
   3:34 p.m., May 12, 1985
*/

/*************************************/

/* DEFINITIONS */

/************************************/

/* logical values: */
#define EOLC  '\015'      /* end of line char */
#define EOF -1            /* end of file */

/* End of phrase character (carriage return) for
   ECHO-PC, Type-'N-Talk, Intex-Talker, Micro-Vox */
#define EOP 13

/* screen size: */
#define MAXLINE 24
#define MAXCOL 79

/* table sizes for user custom pronunciation tables: */
#define TSIZE 40

/* string space for user tables: */
#define USIZE 800

/* size of buffer for synthesizer-specific parameters (file "synth"): */
#define SBFSIZE 1000


/* toupper(c) and tolower(c) defined for ASCII character set */
#define TOUPPER(c) ((c >= 'a' && c <= 'z') ? (c - 32) : c)
#define TOLOWER(c) ((c >= 'A' && c <= 'Z') ? (c + 32) : c)

/* alphabetic(c) and numeric(c) defined for ASCII character set */
#define ALPHABETIC(c) (c >= 'a' && c <= 'z')
#define NUMERIC(c) (c >= '0' && c <= '9')


/* general defines: */

#define  NULL 0
#define  TRUE 1
#define  FALSE 0
#define  BACKSPACE 0x08
#define  NEWLINE 0x0d
#define  DELETE 0x7f
#define  FAKESPACE  0x1f

/***********************8/

/* The following definitions are used for our commands: */

/* default for initiating Screen Review:*/
#define SCREENKEY '\014'      /* Control-L */

/* default for initiating Mode Change: */
 efine MODEKEY '\05'       /* control-E */

   commands while in screen review mode: */
```

BEST COPY AVAILABLE

```
#define AUDIOKEY 'A'        /* hear position of audio cursor */
#define BACKKEY 'B'         /* back to marked position */
#define DOWNKEY 'D'         /* cursor down 1 line */
#define FINDKEY 'F'         /* find specified string */
#define LINEKEY 'G'         /* go to specified line */
#define LEFTKEY 'L'         /* move & say left unit */
#define RIGHTKEY 'R'        /* move & say right unit */
#define MARKKEY 'M'         /* mark current position */
#define UPKEY 'U'           /* cursor up 1 line */
#define RESTKEY '\015'      /* carriage return--say rest of line */
#define ESCKEY '\033'       /* Escape gets out of Screen Review Mode */


/* commands to change general modes: */

#define ALLKEY 'A'          /* all punct */
#define BRAVOKEY 'B'        /* spell saying "alpha, bravo", etc. */
#define CUSTKEY 'C'         /* custom pronunciation on/off */
#define NUMKEY 'D'          /* digits vs. numbers */
#define PAUSEKEY 'E'        /* pause between words on/off */
/* #define       'F'           NOW UNUSED */
#define RPUNCKEY 'G'        /* count repeating punct on/off */
#define HELPKEY 'H'         /* echo function keys on/off */
#define INTONKEY 'I'        /* intoned vs. flat */
#define SKIMKEY 'J'         /* say only big words */
#define ECHOKEY 'K'         /* key echo on/off */
#define LETTERKEY 'L'       /* spell using normal letter names */
#define MOSTKEY 'M'         /* most punct */
#define NORMALKEY 'N'       /* speak normally, ie. in words */
#define COLKEY 'O'          /* say only certain columns */
#define PITCHKEY 'P'        /* change synthesizer pitch */
/* #define       'Q'           NOW UNUSED */
#define RATEKEY 'R'         /* change synthesizer speech rate */
#define SOMEKEY 'S'         /* some punct */
#define TOPKEY 'T'          /* change first line of screen to speak */
#define CAPKEY 'U'          /* capital letters indicated or not */
#define VOLKEY 'V'          /* change synthesizer volume */
/* #define       'W'           NOW UNUSED */
/* #define       'X'           NOW UNUSED */
#define CURSKEY 'Y'         /* speak cursor position */
#define ZAPKEY 'Z'          /* empty speech buffer */

/*****************************************/

/* SUBROUTINES THAT RETURN NON-INTEGERS */

char getscr();
char *numberchk();
char src();
char *syalloc();
char *synstr();


/*****************************************/

/* VARIABLES: */

/*****************************************/

/* These variables hold information as to what modes we are in.
   Some are toggles, ie. TRUE or FALSE.  Others can have more than
   two values, depending on which keys affect that mode.
```

BEST COPY AVAILABLE

```
char talkmode, oldtalkmode;  /* values: BRAVOKEY, LETTERKEY,
char prmode;    /* values:  ALLKEY, SOMEKEY, MOSTKEY (default SOMEKEY) */

int captog;  /* TRUE if capital letters pronounced (default FALSE) */

int helptog;  /* TRUE if function keys echoed (default FALSE) */

int numtog;  /* TRUE if numbers pronounced normally rather than as
                digits (default TRUE) */

int synchtog;  /* TRUE if screen & voice synchronized (default FALSE) */

int pausetog;  /* TRUE if pause between words (default FALSE) */

int echotog;   /* TRUE if input keys echoed (default FALSE) */

int rpunctog;  /* TRUE if repeating punctuation counted (default FALSE) */

int intontog;  /* TRUE if intoned rather than flat (default TRUE) */

int custog;  /* TRUE if user table is to be used (default TRUE) */

/******************************************/

/* These variables contain information on other aspects of
   our current status.
*/

char lineky;  /* control key used to initiate screen review */
char modeky;  /* char used to start a command from the user to us */
char delch1; /* first leftward character that user is trying to delete */
char delch2; /* second leftward character that user may be trying to delete */
char bscount; /* number of characters since BACKSPACE or DELETE */

int top;  /* first line of screen to pronounce */

int line, column; /* position of audio cursor */

int lcol;  /* boundaries for reading specified columns, eg. in a table */
int rcol;

int pitch; /* pitch to be used in setpitch() and setinton() */

/******************************************/

/* The following hold command strings needed to tell the synthesizer
   to perform specific functions.  They are assigned particular values
   when file "synth" is read.  "Synth" must be configured by the user
   for his particular brand of synthesizer.

   Note:  Since different synthesizers may have command strings of
   different lengths, these variables are merely pointers to objects
   of indeterminate length.  The correct amount of storage will be
   allocated by subroutine "syalloc" during initialization.
*/

char *initstr;  /* to initialize the synthesizer */

char *pausestr;  /* to tell it to pause between words */

char *zapstr;  /* to tell it to empty its buffer of anything unspoken */

  ar *volstr[26];  /* to tell it to change volume */

  ar *ratestr[26];  /* to tell it to change the rate of speech */
```

```
char *fpstr[26];    /* 26 pitches to be used with flat intonation */

char *ipstr[26];    /* 26 pitches to be used with normal intonation */

/****************************************/

/* Other global variables: */

int scount; /* counts number of characters sent to screen */

int in;  /* used by subroutine "saychk" to keep track of what mode
            we've been in (alphabetic, numeric or punctuation). */

int markcol, markline;  /* used by subroutine "marker" (called by
       "linerev") to hold the cursor position the user wants marked */

int ncomm;    /* holds the number of the communications port the user
                 has the synthesizer plugged into */

int keyz;     /* "Mykey" sets this to the last "real" key (ie. not part
                 of one of our commands) which we have passed along to
                 the user's program. "Myscreen" then looks at keyz to
                 determine if the screen char is a key echo. */

char oldc;    /* This is used by subroutine "src" (suppress repeating
                 characters) to hold the last character. */

int srcct;  /* "suppress repeating character count"--needed when
               RPUNCKEY is activated (in "modeset") */

/* The following 4 variables are needed for the user's custom
   pronunciation table.  */

char *ctab[TSIZE]; /* original words to find and match */

char *utab[TSIZE]; /* substitute words to replace original ones */

int ltab;  /* actual length of table */

char space[USIZE];  /* our program builds the user table in this area */


char *sp; /*working pointer to where we are in "space" */
char *se;  /* pointer end of space */

char sbf[SBFSIZE]; /* storage for syalloc to use when we read in
                    synthesizer parameters from file "synth" */

char *sbfp = sbf; /* next free position for "syalloc" */

char *sbfe = &sbf[SBFSIZE]; /* last free position */

/* Note:  These 2 variables must be at the end of the unitialized
   variable position.  "Xmine.asm" uses the position of "endram"
   to tell the operating system where our code and variables end,
   so that we will be protected from trashing by others.
*/

int dummy[30];

int endram; /* Used by xmine.asm to protect ram.
               Note:  Must be last unitialized variable. */

/****************************************/
```

```
/* The following are used as defines and allow various ascii strings
   to be pronounced differently, according to the user's commands.
   Certain commands cause the elements of one of these arrays to be
   placed into the appropriate spots in array "ascii", which specifies
   the current pronunciation of each ascii character.
*/

char eos = '\0';  /* end of string (nothing said) */

char *digits[] = {
  " zero "," one "," two "," three "," four "," five "," six ",
  " seven "," eight "," nine "," ten "
};

char *teens[] = {
  " ten "," eleven "," twelve "," thirteen "," fourteen "," fifteen ",
  " sixteen "," seventeen "," eighteen "," nineteen "
};

char *tens[] = {
  " zero "," ten "," twenty "," thirty "," forty "," fifty "," sixty ",
  " seventy "," eighty "," ninety "
};

/* These are put in array "ascii" when the user wants capital letters: */

char *alpha[] = {
  " cap a "," cap b "," cap c "," cap d "," cap e "," cap f",
  " cap g "," cap h "," cap i "," cap j "," cap k "," cap l",
  " cap m "," cap n "," cap o "," cap p "," cap q "," cap r",
  " cap s "," cap t "," cap u "," cap v "," cap w "," cap x",
  " cap y "," cap z "
};

/* These are put in array "ascii" when the user wants letters spelled
   in 'bravo' mode, with capitals indicated: */

char *bravo[] = {
  " cap alpha ",
  " cap bravo ",
  " cap charlie ",
  " cap delta ",
  " cap echo ",
  " cap foxtrot ",
  " cap golf ",
  " cap hotel ",
  " cap india ",
  " cap juliet ",
  " cap kilo ",
  " cap lima ",
  " cap mike ",
  " cap november ",
  " cap oscar ",
  " cap papa ",
  " cap quebec ",
  " cap romeo ",
  " cap sierra ",
  " cap tango ",
  " cap uniform ",
  " cap victor ",
  " cap whiskey ",
  " cap x-ray ",
  " cap yankee ",
  " cap zulu "
```

```
/* This array holds pointers to the current pronunciation for each
   ascii character.  The entries may be changed by certain user commands. */

char *ascii[] = {
  &eos,   /* no pronunciation for control characters */
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  &eos,
  " ",  /* hex 1F used when we want a space not pronounced in 'all' mode */
  " ",     /* space */
  " ",
  " ",     /* quotes */
  " number sign ",
  " dollar sign ",
  " percent ",
  " and ",
  "\'",
  "  ",
  "  ",
  " star ",
  " plus ",
  " ",     /* comma */
  &eos,
  " ",
  " slash ",
  " zero ",
  " one ",
  " two ",
  " three ",
  " four ",
  " five ",
  " six ",
  " seven ",
  " eight ",
  " nine ",
  " ",
  " ",    /* semicolon */
  " less than ",
    " greater than ",
    " ",
    " at ",
    &eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,
    &eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,
    " ",
    " ",
    " ",
    " up arrow ",
    " ",
    " ",
    &eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,
    &eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,&eos,
    " ",
    " ",
    " ",
    " ",
    &eos
};

/****************************************/
```

```
/* file:  MAIN.C
   last edited by pm at
   4:16 p.m., May 14, 1985
*/

/*****************************************/

vima(status,accumulator)
int status;
int accumulator;

/* This routine is effectively the "main" part of our C program.
   It is called by our machine language routine that intercepts
   operating system calls.

   "Vima" is called once with status == 0 on boot-up.

   "Vima" is called with status == 1 each time a user program
   asks the operating system to get a key from the keyboard,
   with the call coming before the key has been gotten.

   "Vima" is then responsible for returning with the requested key
    in the keybuffer.

   "Vima" is called with status == 2 each time a user program
   asks the operating system to put a character on the screen,
   with the call coming after the character has been put on the screen.

*/

{
 char tempch;

 if (status == 0)     /* boot-up */
 {
   init();
   return;
 }

 if (status == 1)       /* get a key */
 {
   mykey();      /* see file "key" for code */
   return;
 }

 if (status == 2)       /* put character on screen */
 {
   tempch = accumulator & 0x007f;
   myscreen(tempch);
   return;
 }

}     /* end "vima" */

/*********************************/

/* The following routines (in alphabetical order) are associated
   with "vima" status = 0 or 2, ie. subroutines "init" and
   "myscreen" and all the major routines they call.
*/

/*********************************/
```

```
/* Boot-up initialization of program. */

{
 static int i;

 ncomm = 0;
 keyz = -2;

 /* initialize command characters: */
 lineky = SCREENKEY;
 modeky = MODEKEY;

 /* initialize other variables & default modes. */
 setup();

 saychk('\01');

 send(pausestr);
 say("screen reeder by veesek and maggs");
 send(pausestr);

}      /* end "init" */

/***********************************/

myscreen(a)
char a;

/* This routine is called only by subroutine "vima", and only when
   "vima" has been called with status = 2.  This happens just after
   a character is plotted on the screen.

   Since "myscreen" implements key echoing, it must determine if
   the character just plotted is the last key the user pressed
   (saved by "mykey" in variable "keyz").  Note that some control
   characters are echoed to the screen as 2 characters:  up-arrow
   and a letter.
*/

{
 static int temp;
 static int ncol, ocol;    /* new column & old column */
 static int nline, oline; /* new line and old line */
 static int cpos; /* cursor position */
 static int i; /* counter */

 static char savecho; /* save echoed key here to check for '^' */

 /* Did we just move to a new column that is to the left of the
    old column, (other than by BACKSPACE or DELETE)?
    Such a situation would indicate that a new line has been
    started and we should therefore say the old line.
 */

 cpos = gccp();
 ncol = cpos % 256;
 nline = cpos / 256;

 if (ncol < ocol && keyz != BACKSPACE && keyz != DELETE)
     saychk(FAKESPACE);

 * Check up and report on what (if anything) happens on screen
    after user presses backspace or delete
 */
```

```
   if ((keyz == BACKSPACE || keyz == DELETE) && nline == oline && ncol <= ocol)
   {
     ++ bscount;

     say("zap");

     if (bscount == 1)
     {
       onall();
       say(ascii[delch1]);
       if (prmode == 'S')
           onsome();
       if (prmode == 'M')
           onmost();
     }

     if (bscount == 2 && delch2 == '^')
     {
       onall();
       say(ascii[delch2]);
       if (prmode == 'S')
           onsome();
       if (prmode == 'M')
           onmost();
     }
   }
   ocol = ncol;
   oline = nline;

  /*  If we are in echo mode then we do the following to see if what
      was sent to the screen is an echo of a keypress.

      Was a key just pressed -- if so echo next character on screen
      if it is the same as the key

                              or

      if the user press a control key, e.g. Control-B, most software
      will echo it as ^B, etc.  In such cases we want to echo both the
      ^ and the B if we are in echo mode.
   */

   ++scount; /* one more character to screen -- set to zero in mykey */

   /* find out if we have a key echo by ibm */
   if
   (
    (scount == 1 && keyz == a) /* first screen char is same as key */
    ||
    (
      (keyz >= 0 && keyz < 0x20) /* keypress was a control key */
      &&
      (
        (scount == 1 && a == '^') /* start of echo as ^B, etc. */
        ||
        (scount == 2 && savecho == '^') /* second character of echo in ^B */
      )
    )
   )

     /* this character is an echo by ibm */
   {
    savecho = a; /* save to check for '^' */

    if (echotog) /* if echotog is on say its name
```

```
      onall();    /* turn on all mode so that punctuation will be echoed */
      say(ascii[a]);

      if (prmode == 'S') /* restore old mode */
        onsome();
      else if (prmode == 'M')
        onmost();
   }
   else
      ;  /* if key echo is not on we throw character away */
  }

  else /* not a key echo, just a plain old character sent to screen */
  {
   savecho = '\0';

  /* check for space after BACKSPACE used for erasing */
    if (scount == 1 && (keyz == BACKSPACE || keyz == DELETE))
        return;

    /* check for top of screen */
    temp = gccp();
    temp = temp / 256;
    if (temp >= top)
    {
       /* check column mode */
       if (lcol > 0 || rcol < MAXCOL)    /* say only certain columns */
       {
          temp = gccp();
          temp = temp % 256;
          if (temp < lcol || temp >  rcol)
             return;
       }

       prechk(a);

    }   /* end if (temp >= top) */

  }  /* end big else */

}    /* end "myscreen" */

/**********************************/

mystrcpy(s,t)     /* copy t to s; pointer version */
char *s, *t;
{
  while(*s++ = *t++);
}

/************************************/

char *numberchk(number)
char *number;

/* Complicated routine which accepts a string of digits and
   returns a pointer to their word equivalent. */

{
  static char dollar,units,whole[20],fraction[20];
  static char t[200];
  static char *np, *wp, *fp;
  static char last;
  static int j, k;
```

```
      return(ascii['.']);

   if (!numtog
        ||
          (number[0] == '0' && number[1] != '.')
        ||
          (number[0] == '$' && number[1] == '\0')
        ||
          number[0] == ','
        ||
          (number[0] == '$' && number[1] == '.' && number[2] == '\0'))
   {
     explode(t, number);
     return(t);
   }

   np = number;
   dollar = *np;
   if (dollar == '$')
      ++np;
   wp = whole;
   fp = fraction;
   k = 0;

   while (*np != '\0'&& *np != '.')
   {
     if (*np == ',')
     {
       if (*(np + 1) < '0' || *(np + 2) < '0' || *(np + 3) < '0'
           || (*(np + 4) >= '0' && *(np + 4) <= '9'))
       {
         explode (t, number);
         return(t);
       }
     }

     else
     {
       *wp++ = *np;
       ++k;
     }

    ++np;

   }  /* end while */

   last = *np;

   *wp = '\0';
   j = 0;

   if (*np == '\0')
      *fp = '\0';
   else
   {
     ++np;
     while (*fp++ = *np++)
     {
       if (*(fp - 1) < '0')
       {
         explode(t, number);
         return(t);
       }
       ++j;
     }
```

```
if (dollar == '$' && (j == 1 || j > 2))
{
  explode(t, number);
  return(t);
}

*t = '\0';
wp = whole;
units = ' ';

do
{
  switch (k)
  {
    case 1:
    case 4:
    case 7:
      if (wp != whole && *wp == '0')
         break;
      if (wp != whole && *(wp - 1) == '1')
         break;
      if (k == 1)
         units = *wp;
      mystrcat(t,digits[*wp - '0']);
      if (k == 4)
         mystrcat(t," thousand ");
      if (k == 7)
         mystrcat(t," million ");
      break;

    case 2:
    case 5:
    case 8:
      if (wp != whole && *wp == '0')
         break;
      if (*wp == '1')
      {
        mystrcat(t,teens[*(wp+1) - '0']);
        if (k == 5)
           mystrcat(t," thousand ");
        if (k == 8)
           mystrcat(t," million ");
      }
      else
      {
        mystrcat(t,tens[*wp - '0']);
        if (*(wp + 1) == '0')
        {
          if (k == 5)
             mystrcat(t," thousand ");
          if (k == 8)
             mystrcat(t," million ");
        }
      }
      break;

    case 3:
    case 6:
    case 9:
      if (wp != whole && *wp == '0')
         break;
      mystrcat(t,digits[*wp - '0']);
      mystrcat(t," hundred ");
      break;
```

```
        default:
          break;

    }    /* end switch */

    --k;
    ++wp;

  }  while (*wp != '\0');     /* end do */

  fp = fraction;

  if (dollar == '$')
  {
    if (*whole != '\0')
    {
      if (units == '1')
         mystrcat(t," dollar ");
      else
         mystrcat(t," dollars ");
    }

    if (*fp == '\0')
    {
      if (last = '.')
          mystrcat(t,ascii['.']);
      return(t);
    }

    if (*fp == '0' && *(fp + 1) == '0')
    {
      if (*whole == '\0')
         return(" zero cents ");
      if (*whole == '0' && *(whole+1) == '\0')
         return (" zero dollars and zero cents");
      return(t);
    }

    if (*whole != '\0')
       mystrcat(t," and ");
    if (*fp == '1')
    {
      mystrcat(t,teens[*(fp+1) - '0']);
      mystrcat(t," cents ");
      return(t);
    }

    if (*fp != '0')
       mystrcat(t,tens[*fp - '0']);
    ++fp;
    if (*fp != '0')
       mystrcat(t,digits[*fp - '0']);
    if (*(fp - 1) == '0' && *fp == '1')
       mystrcat(t," cent ");
    else
       mystrcat(t," cents ");
    return(t);

  }   /* end if (dollar == '$') */

  if (*fp == '\0')
  {
   if (last == '.')
       mystrcat(t,ascii['.']);
   return(t);
```

```
 else
    mystrcat(t," point ");

 do
 {
   mystrcat(t,digits[*fp - '0']);
   ++fp;
 } while (*fp != '\0');

 return(t);

}     /* end "numberchk" */

/***************************************/

prechk(a)
char a;
{
/* Are we saying each character? */

if (talkmode == LETTERKEY || talkmode == BRAVOKEY)
{
      if (rpunctog)       /* count repeated punctuation */
          a = src(a);
      say(ascii[a]);
}

else
  {
   if (captog && a>= 'A' && a<= 'Z')
     {
       saychk(FAKESPACE);
       saychk('c');
       saychk('a');
       saychk('p');
       saychk(FAKESPACE);
     }

     a = TOLOWER(a);
     saychk(a);
 }
} /*end of prechk*/

/***************************************/

readin()

/* reads in ctab and utab pairs (custom pronunciation) until end of file */

{
 static int i,j,k;
 static char string[30];

 myopen();

 i = 0;

 do          /* read in ctab word */
 {
   ctab[i] = sp;

   while ((k = mygetc()) < ' ' && k != EOF)     /* pass end of line */
      ;
   if (k == EOF)
      break;
    /* get ctab entry */
    *sp++ = TOLOWER(k);
    if (sp >= se)
       return(i+1);

    while ((k = mygetc()) != ' ')
    {
      *sp++ = TOLOWER(k);
      if (sp >= se)
         return(i+1);
    }

    *sp++ = '\0';
    if (sp >= se)
       return(i+1);

    /* get utab word or phrase (may be null string)*/
    utab[i++] = sp;

    j = 0;
    while((k = mygetc()) >= ' ') /* read right hand side */
    {
      *sp++ = k;
      if (sp >= se)
         return(i);
      ++j;

      if (j > 48) /* we just put in 49th character */
      {
        say("custom table entry");
        itoa(i,string);
        say(numberchk(string));
        say("too long");
        while (mygetc() >= ' ') /* throw away rest of too long line */
                ;
        --sp;
        --sp;  /* we will zap 49th character */
        break;
      }

    }

    *sp++ = '\0';
    if (sp >= se)
       return(i);

  } while (TRUE);      /* end do */

  ltab = i; /* this will be length of table for later binary search */

  return(0); /*all entries fit in (some may be truncated)*/

 }  /* end "readin" */

 /****************************************/

 rsyn()

 /* Reads in synthesizer control parameters from file "synth". */

 {
  static int i;

  itstr = synstr();
  usestr = synstr();
 for (i = 0; i < 26; i++)
     volstr[i] = synstr();

 for (i = 0; i < 26; i++)
     ratestr[i] = synstr();

 for (i = 0; i < 26; i++)
     fpstr[i] = synstr();

 for (i = 0; i < 26; i++)
     ipstr[i] = synstr();

 if (sygetc() != EOF)
       say(" too many entries in file sinth ");

}  /* end "rsyn" */

/*****************************************/

saychk(a)
char a;

/* This routine is the heart of the code.  It checks every screen
   character and handles it according to what mode we have been in
   and what type of character it is.

   The mode is indicated by the variable "in", which can take on
   3 possible values:
        'P' for punctuation;
        'A' for alphabetic;
        'N' for numeric.
*/

{
 static char phrase[150], word[50];
 static char *wp, *ap;

 if (a == '\001')     /* zapping the buffer */
 {
   wp = word;
   *wp = '\0';
   *phrase = '\0';
 }

 if (rpunctog && talkmode != SKIMKEY)       /* count repeated punctuation */
    a = src(a);

 if (mystrlen(word) > 48) /* ridiculously long word */
 {
  say(phrase);
  *phrase = '\0';
  say(word);
  *word = '\0';
 }

 if (in == 'P')      /* previously in punctuation mode */
 {
   if (ALPHABETIC(a) || (prmode == 'S' && (a == '\'' || a == '-')))
   {
     in = 'A';      /* now alphabetic mode */
     wp = word;
     *wp++ = a;
     return;
   }
```

```
    if (NUMERIC(a) || a == '$' || a == '.')
    {
      in = 'N';     /* now numeric mode */
      wp = word;
      *wp++ = a;
      return;
    }

    /* Neither alphabetic nor numeric so must be punctuation character: */

    if (talkmode != SKIMKEY)    /* SKIMKEY means only speak big words */
       say(ascii[a]);

    *phrase = '\0';

    return;

  }   /* end if (in == 'P')     ie. previously in punct. mode */


  if (in == 'A')       /* previously in alphabetic mode */
  {
    if (ALPHABETIC(a) || (prmode == 'S' && (a == '\'' || a == '-')))
    {
      *wp++ = a;
      return;
    }

    else
    {
      *wp = '\0';

      if (custog)       /* user wants custom pronunciation */
           userchk(word);

      if (mystrlen(word) < 25)
           abbrchk(word);

      if (talkmode == SKIMKEY)
      {
        if (mystrlen(word) < 7)
          *word = '\0';
        else
          mystrcat(word," ");
      }
      if ((mystrlen(wp) + mystrlen(phrase)) >= sizeof(phrase))
      {
        say(phrase);
        *phrase = '\0';
      }

      mystrcat(phrase,word);
      wp = word; /* reinitialize wp to start of word */

      if (a == ' ')
      {
        if (talkmode != SKIMKEY)
           mystrcat(phrase,ascii[' ']);
        return;
      }

      if (a == '-')
      {
        if (talkmode != SKIMKEY)
           mystrcat(phrase,ascii['-']);
      }

      if (NUMERIC(a) || a == '$' || a == '.')
      {
        in = 'N';
        *wp++ = a;
        return;
      }

      /* All other cases fail so we have a punctuation character: */

      in = 'P';      /* now in punct mode */
      say(phrase);
      *phrase = '\0';

      if (talkmode != SKIMKEY)
         say(ascii[a]);
      return;

    }   /* end else */

  }     /* end if (in == 'A'),   ie. previously in alphabetic mode */


  /* Only other choice is (in == 'N'), ie. previously in numeric mode */

  if (NUMERIC(a) || a == '$' || a == '.' || a == ',')
  {
    *wp++ = a;
    return;
  }

  else
  {
    *wp = '\0';

    if (talkmode != SKIMKEY)
       wp = numberchk(word);

    if ((mystrlen(wp) + mystrlen(phrase)) >= sizeof(phrase))
    {
      say(phrase);
      *phrase = '\0';
    }

    mystrcat(phrase,wp);
    wp = word;

    if (a == ' ')
    {
      if (talkmode != SKIMKEY)
         mystrcat(phrase,ascii[' ']);
      return;
    }

    if (ALPHABETIC(a) || (prmode == 'S' && (a == '\'' || a == '-')))
    {
      in = 'A';    /* now in alphabetic mode */
      *wp++ = a;
      return;
    }

    /* All previous tests fail so we have a punctuation character */

    in = 'P';    /* now in punctuation mode */
```

```
      *phrase = '\0';
      if (talkmode != SKIMKEY)
         say(ascii[a]);
      return;

    }     /* end else */

}     /* end "saychk" */


/*******************************************/

setup()

/* Initialize variables and default modes.
   Called only by subroutine "init" on boot-up.
*/

{
  static int temp;
  static int s[30];

  in = 'P';
  srcct = 0;
  oldc = '\0';

  /* Variables associated with what we read on the screen: */

  top = 0;
  markcol = MAXCOL + 1;
  lcol = 0;
  rcol = MAXCOL;

  /* Initialize modes represented by toggles: */

  rpunctog = FALSE;
  helptog = FALSE;
  pausetog = FALSE;
  numtog = TRUE;
  custog = TRUE;
  captog = FALSE;
  echotog = TRUE;
  synchtog = FALSE;
  intontog = TRUE;

  /* Initialize other modes: */

  talkmode = NORMALKEY;
  prmode = SOMEKEY;

  pitch = 'M' - 'A';  /* must be set for possible use in setinton() */

  /* read synthesizer specific parameters & initialize synthesizer: */

  syopen();
  rsyn();
  send(initstr);

  /* initialize for "some" punctuation and normal letters (no caps): */
  onsome();
  offbravo();

   p = space;
   e = &space[USIZE-1];
   emp = readin();
  {
     say("ran out of space for user table at entry");
     itoa(temp,s);
     say(numberchk(s));
     ltab = temp - 1; /* discard offending entry */
  }
  shell(ctab,utab,ltab);
}    /* end "setup" */

/********************************************/

shell(s,t,n)
char *s[],*t[];
int n;

/* Sort table of pointers to chars s and related table t,
    both of length n */

{
  static int gap,i,j;
  static char *temp, *temp1;

  for (gap = n/2; gap > 0; gap /= 2)
    for (i = gap; i < n; i++)
      for (j = i - gap; j >= 0; j -= gap)
      {
        if (mystrcmp(s[j],s[j+gap]) <= 0)
           break;
        temp = s[j];
        temp1 = t[j];
        s[j] = s[j + gap];
        t[j] = t[j + gap];
        s[j + gap] = temp;
        t[j + gap] = temp1;
      }

}   /* end "shell" */

/************************************/

char *syalloc(n)     /* return pointer to n characters*/

{
  if (sbfp + n < sbfe)  /* fits */
  {
    sbfp += n;
    return(sbfp - n);
  }
  else return(NULL);
}

/***************************************/

sygetc()       /* get a character from file "synth" */

{
 static int z;

 z = syread();

 if (z == '*')
 {
   z = syread() - '0';
   z = 10*z + syread() - '0';
   z = 10*z + syread() - '0';
 if (z == 26)
    z = EOF;
 return(z);

}     /* end "sygetc" */

/***************************/

char *synstr()      /* read in a line from file "synth" */

{
 static char *s;
 static char t[30];
 static int i,k;
 static char string[30];

 i = 0;

 while ((k = sygetc()) != '\n' && k != EOF)
      t[i++] = k;

 if (k == EOF)
 {
   say(" missing entry in file sinth ");
   return(NULL);
 }

 t[i] = '\0';
 s = syalloc(i+1);
 mystrcpy(s,t);
 return(s);

}  /* end "synstr" */
```

```
/* file: KEY.C
   last edited by pm at
   410:16 a.m., May 17, 1985.
*/

/*******************************************/

/* This file contains the major routines associated with
   "vima" status = 1, ie. "mykey" and the major subroutines
   it calls.
*/

/********************************************/

mykey()

/* This routine is called only by vima and only when vima is called
   with status == 1 (meaning the user program wants a key).

   "Mykey" loops until a key has been pressed.  Then it looks at the
   key in the keyboard buffer without removing it.
   If it is one of our command keys, it removes it from the buffer
   and handles it.

   This process continues until the user presses an ordinary key,
   destined for the user's program.  At that point, we store the key
   in "keyz" (for later use by "myscreen") and return (via "vima" and
   our assembly language code) to the BIOS routine which will get
   the character and pass it to the user program.
*/

{
 static int k;

 while ((k = zkey()) == 255)    /* wait until a key is pressed */
      ;

 do      /* wait for key & process our commands */
 {

   if (k == lineky)     /* screen review mode */
      linerev();

   if (k == modeky)      /* normal command mode */
   {
     gkey(); /* remove command key from buffer */
     modeset();
   }
   while ((k = zkey()) == 255)    /* wait until a key is pressed */
      ;

 } while (k == lineky || k == modeky);


 /* At this point, the keyboard buffer holds a key to be transferred
    to the user program.
 */

 if (k == BACKSPACE || k == DELETE)
    kback();            /* save what might get erased. */

 keyz = k;  /* save for "myscreen" to check */
 [illegible]count = 0; /* nothing has appeared on screen yet -- for "myscreen" */
```

```
}     /* end "mykey" */

/**********************************/

linerev()

/* Implements screen review mode. */

{
 static int k,c;

 gkey();      /* remove lineky from buffer */

 if (helptog)
    say("review");

 movecursor();  /* audio cursor = video cursor */

 /* get commands and perform them: */

 while ((k = gkey()) != ESCKEY)
 {
   k = TOUPPER(k);
   switch (k)
   {
     case AUDIOKEY:    /* speak current audio cursor position */
       if (helptog)
          say("audio cursor at");
       audiocursor();
       break;

     case BACKKEY:  /* move audio cursor to previously saved position */
       if (helptog)
          say("back");
       review();
       break;

     case DOWNKEY:    /* move cursor down 1 line */
       if (helptog)
          say("down");
       movedown();
       break;

     case FINDKEY:    /* find specified string */
       if (helptog)
          say("find");
       search();
       break;

     case LEFTKEY:     /* move cursor 1 unit (char/word) to left & say it */
       if (helptog)
          say("left");
       if (column > lcol)
       {
         if (talkmode == LETTERKEY || talkmode == BRAVOKEY)
         {
           prechk(getscr());
           --column;
         }
         else
             slastword();
       }
       else
         say("start of line");
       break;
```

```
case LINEKEY:      /* move audio cursor to start of given line */
  if (helptog)
     say("line");
  c = gkey();
  c = TOUPPER(c);
  if (c >= top + 'A' && c <= ('A' + MAXLINE))
  {
    line = c - 'A';
    if (helptog)
       say(ascii[c]);    /* line letter */
    column = lcol;
  }
  else
    say("illegal line");
    if (c < top + 'A')
         say("above top");
  break;

case MARKKEY:    /* save current position of audio cursor */
  if (helptog)
     say("mark");
  marker();
  break;

case RESTKEY:    /* say rest of line & move cursor to next line */
  if (helptog)
     say("rest of line");
  if (column <= rcol)
     sayrest();
  else
     say("at end");
  break;

case RIGHTKEY:    /* move cursor 1 unit (char/word) to right & say it */
  if (helptog)
     say("right");
  if (talkmode == LETTERKEY || talkmode == BRAVOKEY)
  {
    prechk(getscr());
    ++column;
  }
  else
    saynextword();

  if (column > rcol)
  {
    send(pausestr);
    say("line done");
    column = lcol;
  }
  break;

case UPKEY:   /* move cursor up 1 line */
  if (helptog)
     say("up");
  moveup();
  break;

default:
  if (k == modeky)  /* command key to access other set of commands */
  {
    modeset();
    if (column < lcol || column > rcol || line < top)
    {
      column = lcol;
```

```
           say("audio cursor now out of bounds");
           say("moving audio cursor to top left");
         }
       }

       else if (k > 0 && k < 27)   /* control chars */
       {
          if (k == 3|| k == 8 || k == 13 || k == modeky)
             say("cannot use that key");
          else
          {
             lineky = k;
             if (helptog)
             {
                say("review now control");
                say(alpha[k - 1] + 4);
             }
          }
       }
       else   /* not control char */
          say("illegal key");
       break;

   }   /* end switch */

  }    /* end while */

 if (helptog)
    say("end review");

}   /* end "linerev" */

/*******************************************/

modeset()

/* Changes modes as specified by user command. */

{
 static int k;

 if (helptog)
    say("mode");

 k = gkey();    /* get command letter */

 k = TOUPPER(k);

 switch(k)
 {
   case ALLKEY:    /* pronounce all punctuation */
     if (helptog)
        say("all punctuation");
     prmode = k;
     onall();      /* change entries in array "ascii" */
     break;

   case BRAVOKEY:    /* spell letters using 'alpha', 'bravo', etc. */
     if (helptog)
        say("bravo");
     talkmode = k;
     onbravo();      /* change entries in array "ascii" */
     break;

   case CAPKEY:      /* indicate capital letters */
  if (captog)
  {
     if (helptog)
        say("capitals on");
     upcase();    /* change entries in array "ascii" */
  }
  else
  {
     if (helptog)
        say("capitals off");
     lowcase();
  }
  break;

case COLKEY:        /* only speak certain columns */
  if (helptog)
     say("column");
  setcol();
  break;

case CURSKEY:       /* speak cursor position */
  if (helptog)
     say("video cursor at");
  saycursor();
  break;

case CUSTKEY:       /* search user table for custom pronunciations */
  custog = !custog;
  if (helptog)
  {
     say("custom");
     if (custog)
        say("on");
     else
        say("off");
  }
  break;

case ECHOKEY:      /* echo key presses */
  echotog = !echotog;
  if (helptog)
  {
     say("key echo");
     if (echotog)
        say("on");
     else
        say("off");
  }
  break;

case HELPKEY:       /* say effect of commands as user gives them */
  helptog = !helptog;
  if (helptog)
     say("help on");
  else
     say("help off");
  break;

case INTONKEY:      /* flat vs. intoned */
  intontog = !intontog;
  setinton();
  if (helptog)
  {
     say("intonation");
     if (intontog)
```

```
        else
           say("off");
     }
     break;

  case LETTERKEY:    /* spell using regular letter names */
     if (helptog)
        say("letter");
     talkmode = k;
     offbravo();     /* change entries in array "ascii" */
     break;

  case MOSTKEY:      /* pronounce most punctuation */
     if (helptog)
        say("most punctuation");
     prmode = k;
     onmost();       /* change entries in array "ascii" */
     break;

  case NORMALKEY:    /* normal pronunciation, ie. words */
     if (helptog)
        say("normal");
     talkmode = k;
     offbravo();     /* change entries in array "ascii" */
     break;

  case NUMKEY:       /* pronounce digits as either numbers or digits */
     numtog = !numtog;
     if (helptog)
     {
        if (numtog)
           say("numbers");
        else
           say("digits");
     }
     break;

  case PAUSEKEY:     /* pause between words */
     pausetog = !pausetog;
     if (helptog)
     {
        say("pause");
        if (pausetog)
           say("on");
        else
           say("off");
     }
     break;

  case PITCHKEY:     /* change pitch of synthesizer voice */
     if (helptog)
        say("pitch");
     setpitch();
     break;

  case RATEKEY:      /* change synthesizer speech rate */
     if (helptog)
        say("rate");
     setrate();
     break;

  case RPUNCKEY:     /* count repeating punctuation */
     rpunctog = !rpunctog;
     if (helptog)
     {
        if (rpunctog)
           say("on");
        else
           say("off");
      }
      break;

   case SKIMKEY:      /* only pronounce words at least 7 letters long */
      if (helptog)
         say("skim");
      talkmode = k;
      offbravo();     /* change entries in array "ascii" */
      break;

   case SOMEKEY:      /* pronounce some punctuation */
      if (helptog)
         say("some punctuation");
      prmode = k;
      onsome();       /* change entries in array "ascii" */
      break;

   case TOPKEY:       /* specify top line of screen to be spoken */
      if (helptog)
         say("top");
      settop();
      break;

   case VOLKEY:       /* change synthesizer volume */
      if (helptog)
         say("volume");
      setvol();
      break;

   case ZAPKEY:       /* empty speech buffer */
      zap();
      if (helptog)
         say("clear buffer");
      break;

   default:
      if (k >0 && k < 27)   /* control char means new command char */
      {
         if (k == 3 || k == 8 || k == 13 || k == lineky)
            say("cannot use that key");
         else
         {
            modeky = k;
            if (helptog)
            {
               say("now control");
               say(alpha[k - 1] + 4);
            }
         }
      }
      else
         say("illegal key");
      break;

 }    /* end switch */

}    /* end "modeset" */
```

```
/***********************************/
```

Major routines (in alphabetical order) needed to implement

```
*/

/*************************************/

audiocursor()

{
 static char string[50];

 say(ascii['A' + line]);
 itoa(column,string);
 say(numberchk(string));
}

/************************************/

lowcase()

/* turns off spoken indication of upper case */

{
 static int i;

 if (talkmode == BRAVOKEY)
    for (i = 'A'; i <= 'Z'; i++)
        ascii[i] = bravo[i-'A'] + 4;
 else
    for (i = 'A'; i <= 'Z'; i++)
        ascii[i] = alpha[i-'A'] + 4;

}    /* end "lowcase" */

/*************************************/

marker()

{
 markcol = column;
 markline = line;
}

/************************************/

movedown()

{
 column = lcol;

 if (line == MAXLINE)
    say("at the bottom");
 else
    ++line;
}

/****************************************/

moveup()

{
 column = lcol;

 if (line == top)
    say("at the top");
 else
    --line;
```

```
/***********************************************/

offbravo()

{
 static int i;

 if (captog)
 {
   for (i = 'A'; i <= 'Z'; i++)
   {
     ascii[i] = alpha[i-'A'];
     ascii[i+32] = alpha[i-'A'] + 4;
   }
 }

 else
   for (i = 'A'; i <= 'Z'; i++)
     ascii[i+32] = ascii[i] = alpha[i-'A'] + 4;

}   /* end "offbravo" */

/****************************************/

onall()

/* all punctuation pronounced */

{
 onmost();
 ascii[0x20] = " space ";
}

/****************************************/

onbravo()

/* initiates alpha bravo pronunciation of letters */

{
 static int i;

 if (captog)
 {
   for (i = 'A'; i <= 'Z'; i++)
   {
     ascii[i] = bravo[i-'A'];
     ascii[i+32] = bravo[i-'A'] + 4;
   }
 }

 else
   for (i = 'A'; i <= 'Z'; i++)
       ascii[i+32] = ascii[i] = bravo[i-'A'] + 4;

}   /* end "onbravo" */

/*****************************************/

onmost()

   cii['?'] = " question mark ";
   cii['!'] = " exclamation point ";
 ascii[')'] = " right paren ";
 ascii['_'] = " underline ";
 ascii['-'] = " dash ";
 ascii['{'] = " left brace ";
 ascii['}'] = " right brace ";
 ascii['['] = " left bracket ";
 ascii[']'] = " right bracket ";
 ascii['~'] = " tilda ";
 ascii[':'] = " colon ";
 ascii[';'] = " semicolon ";
 ascii['"'] = " quote ";
 ascii['\''] = " apostrophe ";
 ascii['`'] = " grave accent ";
 ascii['|'] = " vertical line ";
 ascii['\\'] = " backslash ";
 ascii[','] = " comma ";
 ascii['.'] = " period ";
 ascii[0x20] = " ";

}   /* end "onmost" */

/****************************************/

onsome()

/* only punctuation pronounced is @#$%&*+=/<> */

{
 ascii['?'] = " ";
 ascii['!'] = " ";
 ascii['('] = " ";
 ascii[')'] = " ";
 ascii['_'] = " ";
 ascii['-'] = " ";
 ascii['{'] = " ";
 ascii['}'] = " ";
 ascii['['] = " ";
 ascii[']'] = " ";
 ascii['~'] = " ";
 ascii[':'] = " ";
 ascii[';'] = " ";
 ascii['"'] = " ";
 ascii['\''] = "\'";
 ascii['`'] = " ";
 ascii['|'] = " ";
 ascii['\\'] = " ";
 ascii[','] = " ";
 ascii['.'] = " ";
 ascii[0x20] = " ";

}    /* end "onsome" */

/************************************/

review()

{
 if (markcol < lcol || markcol > rcol || markline < top)
 {
   say("out of bounds");
   return;
 }

   lumn = markcol;
   ne = markline;
```

```
/************************************/

saycursor()

/* say position of cursor */

{
 static int j;
 static char string[50];
 char *numberchk();

 /* get cursor position with call to machine language program */
 j = gccp();

 say(ascii['A' + j / 256]);     /* line */
 itoa(j % 256,string);
 say(numberchk(string));     /* column */

}    /* end "saycursor" */

/************************************/

saynextword()
/* assumes that on entry line and column are legal, i.e. between
   top and MAXLINE and between lcol and rcol respectively
*/

{
 static char string[MAXCOL+4];
 static char *stp;
 static char c;

 stp = string;

 if (!alphanumeric(c = getscr()))
 {
   do
   {
     *stp++ = c;
     ++column;
   } while (column <= rcol && !alphanumeric(c = getscr()));
 }

 else
 {
   while (column <= rcol && alphanumeric(c = getscr()))
   {
     *stp++ = c;
     ++column;
   }
 }

 *stp = '\0';
 stp = string;
 while (*stp != '\0')
    prechk(*stp++);
 prechk('\0');

}    /* end "saynextword" */

/************************************/

rest()
```

```
 static char string[81];
 static char *stp;

 stp = string;

 for (; column <= rcol; column++)
     *stp++ = getscr();

 *stp = '\0';
 stp = string;

 while (*stp != '\0')
    prechk(*stp++);

 prechk('\0');

 column = lcol;

}     /* end "sayrest" */

/*******************************************/

saywholeline()

{
 static char string[MAXCOL+4];
 static char *stp;

 stp = string;

 for (column = lcol; column <= rcol; column++)
     *stp++ = getscr();

 *stp = '\0';
 stp = string;

 while (*stp != '\0')
    prechk(*stp++);

 prechk('\0');

}    /* end "saywholeline" */

/*****************************************/

search()

/* get string, search for it, and say line containing it */

{
 static char sstring[20];
 static int i,j,k,l,c;
 static char *ssp;

 i = 0;

 while ((k = gkey()) != NEWLINE && i < 19)
    sstring[i++] = k;
 if (i == 19)    /* throw away extra keys up to delimiter */
     while (gkey() != NEWLINE)
           ;
 sstring[i] = '\0';
   y(sstring);

   = top;
 c = lcol;

 while(l <= MAXLINE)
 {
   c = lcol;

   while (c <= (rcol - i + 1))
   {
     j = c;
     ssp = sstring;

     while ((getsc(256*l + j) % 256) == *ssp)
     {
       j++;
       ssp++;
     }

     if (*ssp == '\0')
     /* SUCCESS!! */
     {
       line = l;
       saywholeline();
       column = c; /* set audio cursor to start of word */

       /* user presses space bar to look for another instance */
       /* any other key ends search and is the start of a new */
       /* command to be interpreted elsewhere */

       while ((k = zkey()) == 255)
          ;
       if (k != ' ')
         return; /* return with key still in key buffer */
       gkey();  /* get space out of key buffer */
       c = rcol + 3; /* pop out of loop for this line */
     }

     /* either still looking for first instance or start looking */
     /* for next instance */
     ++c;

   }       /* end while (c <= (rcol - i + 1)) */

   /* not on this line -- try next line */
   ++l;

 }         /* end (while l <= MAXLINE) */

 say("not found");

}   /* end "search" */

/************************************/


setcol()

/* set up to read text by columns */

{
 static int ltemp, rtemp, lnum, rnum;
 static char s[30];

 while (zkey() == 255)
       ;
 f (zkey() == NEWLINE)
```

```
   lcol = 0;
   rcol = MAXCOL;
   if (helptog)
      say("all columns");
 }

 else /* get column numbers */
 {
   if ((ltemp = getnum()) != ESCKEY
       && (lnum = getnum()) != ESCKEY
       && (rtemp = getnum()) != ESCKEY
       && (rnum = getnum()) != ESCKEY)
   {
     ltemp = 10*ltemp + lnum;
     rtemp = 10*rtemp + rnum;

     if (rtemp <= MAXCOL && ltemp <= rtemp)
     {
       lcol = ltemp;
       rcol = rtemp;
       if (helptog)
       {
         say("left");
         itoa(ltemp,s);
         say(numberchk(s));
         say("right");
         itoa(rtemp,s);
         say(numberchk(s));
       }
     }

     else /* illegal */
     {
       say("you tried to put left column at");
       itoa(ltemp,s);
       say(numberchk(s));
       say("and right column at");
       itoa(rtemp,s);
       say(numberchk(s));

       if (rtemp > MAXCOL)
           say("right column is too far right");
       if (ltemp > MAXCOL)
           say("left column is too far right");
       if (ltemp > rtemp)
           say("left column may not be to the right of right column");
     }  /* end else illegal */

   } /* end if ltemp = getnum(), etc. */

 }    /* end else column numbers */

}    /* end "setcol" */

/*****************************************/

setinton()

{
 if (intontog)
    send (ipstr[pitch]);
 else
    send(fpstr[pitch]);
```

```
setpitch()
{
 /* pitch must be global and initialized because it is used in setinton()*/
 static int kk;

 kk = gkey();
 kk = TOUPPER(kk) - 'A';

 if (kk < 0 || kk > 25)
 {
   say("illegal pitch");
   return;
 }

 pitch = kk;

 if (intontog)
    send(ipstr[pitch]);
 else
    send(fpstr[pitch]);
 if (helptog)
    say(ascii[pitch + 'A']);
}   /* end "setpitch" */

/*******************************************/

setrate()

{
 static int rate;

 rate = gkey();
 rate = TOUPPER(rate);

 if (rate < 'A' || rate > 'Z')
 {
   say("illegal rate");
   return;
 }

 send(ratestr[rate-'A']);

 if (helptog)
    say(ascii[rate]);
}    /* end "setrate" */

/*******************************************/

settop()

/* sets to ignore anything written on designated top screen lines */

{
 static int oldtop;

 oldtop = top;
 top = gkey();
 top = TOUPPER(top) - 'A';
 if (top < 0 || top > MAXLINE)
 {
   say("illegal top");
   top = oldtop;
```

```
else if (helptog)
```

```
}

/*************************************/

setvol()

{
 static volume;

 volume = gkey();
 volume = TOUPPER(volume);

 if (volume < 'A' || volume > 'Z')
 {
   say("illegal volume");
   return;
 }

 send(volstr[volume-'A']);
 if (helptog)
    say(ascii[volume]);

}  /* end "setvol" */

/*************************************/

slastword()

{
 static char string[MAXCOL+4];
 static char *stp;
 static char c;

 stp = string;

 --column;

 if (!alphanumeric(c = getscr()))
 {
   do
   {
     *stp++ = c;
     --column;
   } while (column >= lcol && !alphanumeric(c = getscr()));
 }

 else
 {
   while (column >= lcol && alphanumeric(c = getscr()))
   {
     *stp++ = c;
     --column;
   }
 }
 ++column;

 *stp = '\0';
 reverse(string);
 stp = string;
 while (*stp != '\0')
    prechk(*stp++);
 prechk('\0');

}  /* end "slastword" */
```

```
/******************************************/

upcase()

/* forces spoken indication of upper case */

{
 static int i;

 if (talkmode == BRAVOKEY)
    for (i = 'A'; i <= 'Z'; i++)
        ascii[i] = bravo[i-'A'];
 else
    for (i = 'A'; i <= 'Z'; i++)
        ascii[i] = alpha[i-'A'];
}

/******************************************/

zap()
{
  saychk('\001');
  send(zapstr);
}
```

```
/* file:  sts.c
   last edited by pm
   2:16 p.m., May 10, 1985
*/


#include  defs.c
#include  main.c
#include  key.c

/****************************************

           UTILITY ROUTINES:

****************************************/

abbrchk(word)
char word[];

/* Sees if we have an abbreviation (all consonants). */

{
 static char *wp, *np, newword[50];

 wp = word;

 while (*wp != '\0')
 {
   if (*wp == 'a' || *wp == 'e' || *wp == 'i' || *wp == 'o'
           || *wp == 'u' || *wp == 'y')
      return;
   ++wp;
 }

 np = newword;
 wp = word;

 while (*wp != '\0')
 {
   *np++ = *wp++;
   *np++ = ' ';
 }

 *np = '\0';
 mystrcpy(word,newword);

}   /* end "abbrchk" */

/********************************/

alphanumeric(testchar)
char testchar;

{
 if ((testchar >= 'a' && testchar <= 'z') ||
         (testchar >= 'A' && testchar <= 'Z') ||
         (testchar >= '0' && testchar <= '9'))
    return(TRUE);

 else
    return(FALSE);
```

```
/****************************************/

binary(word,tab,n)          /* binary search */
char *word, *tab[];
int n;

/* find word in tab[0]...tab[n-1], return index */

{
 static int low, high, mid, cond;

 low = 0;
 high = n - 1;

 while (low <= high)
 {
   mid = (low + high) / 2;

   if ((cond = mystrcmp(word,tab[mid])) < 0)
      high = mid - 1;
   else if (cond > 0)
      low = mid + 1;
   else
      return(mid);
 }

 return(-1);

}    /* end "binary" */

/****************************************/

explode(s, t)      /* expand number in t characterwise to s */
char *s, *t;

{
 *s = '\0';

 while (*t != '\0')
 {
   if (*t == '$')
      mystrcat(s, " dollar sign ");
   else if (*t == '.')
           mystrcat(s, " point ");
   else if (*t == ',')
           mystrcat(s, " comma ");
   else
        mystrcat(s,digits[*t - '0']);

   ++t;
 }

}    /* end "explode" */

/**************************************/

getnum()

{
 static int num;

 while (((num = gkey()) < '0' || num > '9') && num != ESCKEY)
      say("press a digit or press escape");

    (num == ESCKEY)
      return(ESCKEY);
```

```
        return(num - '0');
}

/**********************************/

char getscr()
{
 char a;

 a = getsc(column+256*line);
 return(a);
}

/************************************/

itoa(n, s)    /* convert n to characters in s */
char s[];
int n;

{
 static int itn,iti;

 itn = n;
 if (itn < 0)
    itn = -itn;         /* make n positive */
 iti = 0;

 do     /* generate digits in reverse order */
 {
    s[iti++] = itn % 10 + '0';   /*get next digit*/
 }  while ((itn /= 10) > 0); /*delete it*/

 if (n < 0)
    s[iti++] = '-';

 s[iti] = '\0';
 reverse(s);

}    /* end "itoa" */

/****************************************/

movecursor()

/* move audio cursor to same point as video cursor */

{
 static int j;

 j = gccp();
 column = j % 256;
 line = j / 256;

 if (column < lcol || column > rcol || line < top)
 {
   column = lcol;
   line = top;
   say("video cursor out of current bounds");
   say("moving audio cursor to top left");
 }
}

   **************************************/

mygetc()
```

```
{
 static int z;

 z = myread();
 if (z == 26)
      z = EOF;
 return(z);
}

/***********************************/

mystrcat(s, t) /* concatenate t to end of s */
char *s, *t;

{
 while (*s++)       /* find end of s */
     ;

 --s;

 while (*s++ = *t++)        /* put t on end of s */
     ;
}

/*************************************/

mystrcmp(s, t)    /* return <0 if s < t, 0 if s == t, >0 if s > t */
char *s, *t;

{
  for ( ; *s == *t; s++, t++)
      if (*s == '\0')
         return(0);

  return (*s - *t);
}

/************************************/

mystrlen(s)     /* return length of string s */
char *s;

{
 static char *strp;

 strp = s;
 while (*strp)
     strp++;
 return(strp - s);
}

/***************************************/

reverse(s)   /* reverse string s in place */
char s[];

{
 char rc;
 static int ri, rj;

 for (ri = 0, rj = mystrlen(s) - 1; ri < rj; ri++, rj--)
 {
   rc = s[ri];
   s[ri] = s[rj];
   s[rj] = rc;
}

/*********************************/

say(phr)
char phr[];

/* Says the phrase pointed to by phr. */

{
 static int ich;
 static char *php;
 static char savech = '\0';

 if (phr[0] == ' ' && phr[1] == '\0' && savech == ' ')
     return;

 if (*phr == '\0')
      return;
 php = phr;

 while (*php != '\0')
 {
   ich = *php++;
   ich = TOLOWER(ich);
   if (pausetog && ich == ' ')
           send(pausestr);
   putcm(ich,ncomm);
 }

 savech = ich;

 putcm(EOF,ncomm);

}    /* end "say" */

/*************************************/

kback() /* save characters that may be backspaced over to say them later */
{
 static int p, j, c;

 p = gccp();
 c = p % 256;

 if (c != 0)
 {
   j = getsc(p - 1);
   delch1 = j % 256;
 }

 c = c - 1;

 if (c != 0)
 {
   j = getsc(p - 2);
   delch2 = j % 256;
 }
   bscount = 0; /* no output since backspace pressed */

}    /* end "kback" */

***************************/
```

```
send(phr)
char phr[];

/* Send out unchanged the string pointed to by phr. */

{
 static int kk;
 static char cc, *phpl;

 phpl = phr;

 while (*phpl != '\0')
 {
   cc = *phpl++;
   kk = cc;
   putcm(kk,ncomm);
 }

 putcm(EOP,ncomm);

}   /* end "send" */

/*********************************/

char src(c)
char c;

{
 char *numberchk();
 static char string[50];

 if (c == oldc) /*one more of the same*/
 {
   if (ALPHABETIC(c) || NUMERIC(c))
      return(c);
   ++srcct;
   return('\0');
 }

 /* at this point we have a new character */

 /* now we test to see if the previous characters were repeated */

 if (srcct != 0)
 {
   /* deal with repeated characters */
   /* if old character was pronounced say count, otherwise say nothing */

   if (*ascii[oldc] != '\0'
       && (*ascii[oldc] != ' ' || *(ascii[oldc] + 1) != '\0')
      )
   {
     ++srcct; /* adjust count to include first character */
     itoa(srcct,string);
     say(numberchk(string));
     say("times");
   }
   srcct = 0;
 }

   oldc = c;
   return(c);
}   /* end "src" */

/*************************************/
```

```
userchk(word)
char word[];

{
 int n;

 n = binary(word,ctab,ltab);
 if (n != -1)
      mystrcpy(word,utab[n]);
} /*end "userchk" */
```

```
;last edited by p.m.
;4:18 p.m., Apr. 26, 1985
;This is file "xbef.asm"
;
;We have split the assembly code into two files so that "endram_"
;will be put as the last uninitialized ram space.
;
codeseg segment para public 'code'
assume cs:codeseg
 extrn myinit:near
;
;initialize
$begin proc near
 jmp myinit
$begin endp
;
codeseg ends
  end
```

```
;last edited by p.m.
;4:18 p.m., Apr. 26, 1985
;This is file "xaft.asm"
;
;This file must be linked at the very end so that endram will
;be at the top of unitialized ram
;
codeseg segment para public 'code'
assume cs:codeseg
 extrn vima_:near
 extrn endram_:near
;
;**********************************************************
;
; Come here on a display interrupt.  First do the display
; and then call vima with ax = 2.  This will cause vima
; to call myscreen().
;
my10 proc near ;display interrupt
  jmp my10j
saveax dw ?
sav2ax dw ?
savesp dw ?
savess dw ?
filler dw 90 dup(?)
mystack dw ?
my10j:
  mov cs:saveax,ax
  call int10  ;do display
  mov cs:sav2ax,ax
  mov cs:savess,ss
  mov cs:savesp,sp
  cli
  mov ax,cs
  mov ss,ax
  mov ax,offset mystack
  mov sp,ax
  sti
  mov ax,cs:sav2ax
  pushf
  push ax
  mov ax,cs:saveax
  cmp ah,9 ;write attribute/character at current cursor position
  jb my10r
  cmp ah,10 ;write character only at current cursor position
  ja my10r
  push bx
  push cx
  push dx
  push ds
  push es
  push si
  push di
  push bp
  mov ah,0
  push ax
  mov ax,cs
  mov ds,ax
  mov es,ax
  mov ax,2
   ush ax
   all vima_
  pop ax
  pop ax
  pop bp
  pop di
  pop si
  pop es
  pop ds
  pop dx
  pop cx
  pop bx
my10r:
  pop ax
  popf
  cli
  mov ss,cs:savess
  mov sp,cs:savesp
  sti
  iret
my10 endp
;
;
;**********************************************************
;
; We save original BIOS display interrupt address here
;
int10 proc near ;display interrupt
  pushf
  db 09ah ;inter-segment direct call
i10off dw 0
i10seg dw 0
  ret
int10 endp
;
;**********************************************************
;
; We come here on key interrupt, call vima with ax = 1
; which results in a call to mykey(), and then come
; back and get the key and return it to the calling program
;
my16 proc near ;key interrupt
  pushf
  cmp ah,0
  je vi16
  popf
  db 0eah ;inter-segment direct jump
my16off dw 0
my16seg dw 0
;
vi16:
  mov cs:saveax,ax
  mov cs:savess,ss
  mov cs:savesp,sp
  cli
  mov ax,cs
  mov ss,ax
  mov ax,offset mystack
  mov sp,ax
  sti
  mov ax,cs:saveax
  push ax
  push bx
  push cx
  push dx
  push bp
  push si
  push di
  push ds
  push es
  push ax
  mov ax,cs
  mov ds,ax
  mov es,ax
  mov ax,1
  push ax
  call vima_ ;on return ax will have result of int16
  add sp,4
  pop es
  pop ds
  pop di
  pop si
  pop bp
  pop dx
  pop cx
  pop bx
  pop ax
  cli
  mov ss,cs:savess
  mov sp,cs:savesp
  sti
  popf   ; on old stack
  db 0eah ;inter-segment direct jump
my16of dw 0
my16se dw 0
;
my16 endp
;
;***********************************************
;
;  We save the original BIOS key interrupt
;  vector here.
;
int16 proc near ;key interrupt
  pushf
  db 09ah ;inter-segment direct call
i16off dw 0
i16seg dw 0
  ret
int16 endp
;
;*****************************************
;
; This routine is called with a character
; and a Comm port number.  It sends this character
; out the specified Comm port.  It is used to
; send data to the synthesizer
;
  public putcm_
putcm_ proc near
  mov ax,0
  call $sav
  mov ax,word ptr 8[bp]
ptcm_:
  mov ah,1
  mov dx,word ptr 10[bp]
  int 14h
  ret
putcm_ endp
;
;************************************
;
;  This gets the current video cursor position
;  and returns it to the calling program.
;  See IBM Bios description for Int 10 for details
```

```
  public gccp_
gccp_ proc near
;get current cursor position
 mov ax,0
 call $sav
;get current cursor position into ax
 mov ah,3
 mov bh,0
 push bp
 call int10
 mov ax,dx
 pop bp
 ret
gccp_ endp
;
;********************************
;
; This routine gets a character from
; a screen location specified by the parameters
; sent by the calling program
; See IBM BIOS description for Int 10 for details.
;
  public getsc_
getsc_  proc near
 mov ax,0
 call $sav
;read current cursor position
 mov ah,3
 mov bh,0
 push bp
 call int10
 pop bp
 push dx ;save result
;set cursor to desired read position
 mov dx,word ptr 8[bp]
 mov ah,2
 mov bh,0
 push bp
 call int10
 pop bp ;dx is still on stack
;read attribute/char at new cursor position
 mov ah,8
 mov bh,0
 push bp
 call int10
 pop bp
 pop dx ;get old cursor location
 push ax        ;save attribute/char
;restore cursor position
 mov ah,2
 mov bh,0
 push bp
 call int10
 pop bp
 pop ax ;get back attribute/char
 ret
getsc_  endp
;
; ******************************
;
;  This routine looks for a key.  If found it leaves key in
;  buffer and returns ascii value of key.  If no key found
   it returns 255

  public zkey_
```

```
zkey_ proc near
  mov ah,1
  call int16
  jz short zky ;if zero, no code available
  mov ah,0
  ret  ;ax has code
zky:
  mov ax,255
  ret
zkey_ endp
;
;*****************************
;
;  This routine waits until there is a key in the buffer
;  and then removes the key and returns with it.
;
  public gkey_
gkey_ proc near
  mov ah,0
  call int16
  mov ah,0
  ret
gkey_ endp
;
;**********************************
;
;  Routine syopen_ opens a file called synth.
;  Routine syread_ reads a character from it.
;
  public syopen_
  public syread_
syopen_ proc    near
        jmp     short syjmp
;
sydta   db      0
;
syfcb   db      0
        db      'SYNTH   '
        db      '   '
syblock dw      0
sysize  dw      0
        db      16 dup(?)
syrec   db      0
        db      4 dup (?)
syjmp:
        push    ds
        mov     bx,cs
        mov     ds,bx
        mov     dx,offset sydta
        mov     ah,1ah          ;set disk transfer address
        int     21h
        mov     dx,offset syfcb
        mov     ah,0fh          ;open
        int     21h
        mov     syblock,0       ;set up starting point
        mov     sysize,1        ;one-byte records
        mov     syrec,0
        pop     ds
        mov     ah,0
        ret
;
syread_:
        push    ds
        mov     bx,cs
        mov     ds,bx
        mov     dx,offset syfcb
        mov     ah,14h
        int     21h
        mov     al,byte ptr sydta
        mov     ah,0
        pop     ds
        ret
;
syopen_ endp
;
;
;**********************************
;
;  Routine myopen_ opens a file called table.
;  Routine myread_ reads a character from it.
;
  public myopen_
  public myread_
myopen_ proc    near
        jmp     short myjmp
;
mydta   db      0
;
myfcb   db      0
        db      'TABLE   '
        db      '   '
myblock dw      0
mysize  dw      0
        db      16 dup(?)
myrec   db      0
        db      4 dup (?)
myjmp:
        push    ds
        mov     bx,cs
        mov     ds,bx
        mov     dx,offset mydta
        mov     ah,1ah              ;set disk transfer address
        int     21h
        mov     dx,offset myfcb
        mov     ah,0fh              ;open
        int     21h
        mov     myblock,0           ;set up starting point
        mov     mysize,1            ;one-byte records
        mov     myrec,0
        pop     ds
        mov     ah,0
        ret
;
myread_:
        push    ds
        mov     bx,cs
        mov     ds,bx
        mov     dx,offset myfcb
        mov     ah,14h
        int     21h
        mov     al,byte ptr mydta
        mov     ah,0
        pop     ds
        ret
;
myopen_ endp
;
;********************************8
;
;These routines are used by the c compiler for entry to and
;exit from called subroutines
;
```

```
  public $sav
  public $ret
$sav  proc  near
  pop  bx
  push  di
  push  si
  push  bp
  mov  bp,sp
  add  sp,ax
  call  bx
$ret:
  mov  sp,bp
  pop  bp
  pop  si
  pop  di
  test  ax,ax
  ret
$sav  endp
;
;****************************************
;
; This routine is used to compile the switch statement
;
  public $swt
$swt  proc  near
  pop  bx
  mov  cx,word ptr cs:[bx]
  add  bx,2
  jcxz  eswt
swtloop:
  cmp  ax,word ptr cs:[bx]
  je  found
  add  bx,4
  loop  swtloop
eswt:
  jmp  word ptr cs:[bx]
found:
  jmp  word ptr cs:2[bx]
$swt  endp
;
;************************************
;
; This routine is jumped to by the first statement in xbef.asm
; save the BIOS vectors, substitute our vectors
; and then leaves and stay resident.
;
  public myinit
;
myinit proc near
;get int 10h and int 16h vectors
;
  mov ah,35h ;get vector for
  mov al,10h ;interrupt number 10h
  int 21h
;now es:bx has vector for int 10h
;save in our int 10h call area
  mov i10off,bx
  mov i10seg,es
;
;now put vector to my10 in BIOS int 10h area
;
;put address of my10 in ds:dx
 mov ax,cs
 mov ds,ax
 mov dx,offset my10
 mov ah,25h ;put vector for
 mov al,10h ;interrupt number 10h
 int 21h
;
  mov ah,35h ;get vector for
  mov al,16h ;interrupt number 16h
  int 21h
;now es:bx has vector for int 16h
;save in our int 16h call area
  mov i16off,bx
  mov i16seg,es
  mov my16off,bx
  mov my16seg,es
  mov my16of,bx
  mov my16se,es
;
;now put vector to my16 in BIOS int16 area
;
;put address of my16 in ds:dx
 mov ax,cs
 mov ds,ax
 mov dx,offset my16
 mov ah,25h ;put vector for
 mov al,16h ;interrupt number 16h
 int 21h
;
;initialize vima_
;
 mov ax,cs
 mov ds,ax
 mov ax,0
 push ax
 push ax
 call vima_
 pop ax
 pop ax
;
;now quit
;
  mov dx,offset endram_
  int 27h
;
myinit endp
;
codeseg ends
  end
```

SOURCE CODE FOR TALKHELPER FOR THE IBM PERSONAL COMPUTER

```
/*this is file speech.c*/
/*last edited by p.m.*/
/*9:38 a.m., May 27, 1985*/

/* defines for speech program */
#include "sdefs"

/* picture definitions for graphics screen paramenters */
#include "pdefs"

/* standard i/o and file definitions supplied by Aztec C */
#include "stdio.h"
#include "fcntl.h"

#define TRUE 1
#define FALSE 0

/* ++++++++++++++++++++++++++++++++++++++++++++++++ */

/* Machine specific routines coded in assembler by p.m.*/

/* check for a key, return 255 if no key, ascii key value if */
/* there is a key, leave the key in keybuffer */
int zkey();

/* get a key from keyboard waiting as long as necessary */
/* return with ascii code */
int gkey();

/* send a character out serial port (COM1 on IBM) to synthesizer */
int putcm();

/* put screen in text mode */
int texton();

/* put screen in graphics mode */
int graphon();

/* +++++++++++++++++++++++++++++++++++++++++++ */

/* move block to graphics screen memory -- an Aztec C library routine */
int graphmv();

/* +++++++++++++++++++++++++++++++++++++++++++ */
/* working variables */

/* black and white data to allow quick changes by block moves to */
/* graphics screen -- will be initialized in program */
/* ??? used for building scanning inner frames on picture menu */

char black[40],white[40];


/* SCREEN STATUS VARIABLES */

int keyson = FALSE; /* is scanning keylist on bottom of screen */

int line; /*line where next writing goes on screen*/

int col; /*column where next writing goes on screen*/


/* VARIABLES RELATED TO CURRENT MENU */
```

```
int menu; /*number of current menu*/

int nentry; /*number of current entry in current menu */

char edmode; /*'E' for special editing, 'R' for regular mode */


/* VARIABLES AFFECTING ALL MENUS */

char mmark[MAXMENUS]; /* used to mark active menus for garbage collection */

/* The following information will be read in from the menu file, but may be */
/* altered by the user.  Changes may optionally be written out */

/* global information about menus */

/* number of menus -- may not exceed MAXMENUS */
int menuct; /*number of menus*/

/* scan rate -- a number between 1 and 30 */
int grate;

/* input mode:

   K -- Key selection

   A -- All keys as a single switch

   S -- Single switch attached to joystick button connector

   D -- Divided keyboard as two switches

   T -- Two switches attached to joystick connector

*/
char ginput;


/* type of menus:  'P' == picture;  'T' == text */
char gtype;

/* arrays of menu-specific information */

char *mname[MAXMENUS]; /*names of menus*/

/* Number of entries in each menu */
/* For text menu may not exceed MAXENTRIES. */
/* For picture menu must be 4, 9, 16, or 25 */
int msize[MAXMENUS];

char *mlabel[MAXMENUS][MAXENTRIES]; /*entry labels */

char *mtext[MAXMENUS][MAXENTRIES]; /*entry phrase or action */


/* VARIABLES FOR STRING MANAGEMENT */

char sbuff[WIDTH + 2]; /*buffer for string input*/

char cs[MAXBUF]; /*buffer for strings selected by user*/

int eend[50]; /* locations of ends of phrases in "to say" buffer */

    eendi; /* index for eend, e.g. eend[eendi] */
```

```
char buf[PSIZE]; /*buffer for reading in a picture*/

/* STRING CONSTANTS */

/* end of string -- used for setting pointers */
char eos = '\0';

/* deignators for left and right columns */
char lrdes[] = LRDES;

/* special function keys */
char special[] = {UNDKEY,DISKEY,EXITKEY,SAYKEY,EEKEY,EMKEY,EDKEY,REGKEY,'\0'};

/* keyset to display at bottom of screen for handicapped people */
char keys1[] = "? < * &   0 1 2 3 4 5 6 7 8 9 ";
char keys2[] = "E T O A I H N S R L U D Y W G M C B F K P V J X Z ";

/********************************/

main()
/*read in menus, then go into loop getting and obeying user commands*/
{
 static int choice;
 unsigned int i,j,k;

/* lazy way to initialize to values for IBM graphics adapter */
 for(i = 0; i < 40; i++) {
   black[i] = '\000';
   white[i] = '\377';
 }

 /* initial modes */
 edmode = 'R';
 texton(); /* probably not needed */
 readmenus();
 showmenu("MAIN"); /*put main menu on screen*/

 /* initizialize "to say" buffer */
 cs[0] = '\0';
 eendi = 0;
 eend[eendi++] = 0;

 /* where we start writing on screen */
 line = BOTTOM - 4;
 col = 1;

 /* MAIN PROGRAM LOOP */

 while (TRUE) {

   /* make a valid choice -- ignore illegal choices */
   /* getin() will return a large number for illegal entries */
   /* getin() will return a negative number for command keys */
   /* getin() will return a positive number for entry selection */

   while((choice = getin()) >= msize[menu])
           ;

   /* deal with special handicapped input mode for command keys */
   /* non-typing user can select command keys off menu */
   /* (If first character in the phrase is "&", this means the */
   /*  next character signifies a command character chosen by */
   /*  the user.) */
```

```
[illegible] (choice >= 0 && mtext[menu][choice][0] == '&') {
    if (mtext[menu][choice][1] == '<') choice = -SAYKEY;
    else if (mtext[menu][choice][1] == '!') choice = -DISKEY;
    else if (mtext[menu][choice][1] == '?') choice = -UNDKEY;
    else choice = - mtext[menu][choice][1];
  }

  /*take appropriate action based on choice*/

  switch (choice) {
    case -EXITKEY: /*exit to computer operating system*/
      texton();
      EDU();
      exit(0);

    case -DISKEY: /*clear bottom of screen & "to say" in text mode*/
      clrbot();
      cs[0] = '\0';
      eendi = 0;
      eend[eendi++] = 0;
      line = BOTTOM - 4;
      col = 1;
      CUP(line,col);
      break;

    case -SAYKEY: /*say string created in text mode*/
      say(cs);
      break;

    case -UNDKEY: /*erase most recent phrase selected in text mode*/
      --eendi;
      --eendi;
      cs[eend[eendi++]] = '\0';
      replot();
      break;

    case -EDKEY: /* enter helper edit mode with keyboard entry & text screen*/
      edmode = 'E';
      showmenu(mname[menu]);
      clrbot();
      break;

    case -REGKEY: /* exit helper edit mode -- go back to mode for this menu */
      edmode = 'R';
      showmenu(mname[menu]);
      if (gtype == 'T')
           clrbot();
      break;

    case -EMKEY: /* all options except changing label or phrase */
      editmenu();
      clrbot();
      replot();
      break;

    case -EEKEY:
      emenu();
      clrbot();
      replot();
      break;

    default: /*make a selection from menu*/
      nentry = choice;

      /* new menu selected */
      if (*mtext[menu][nentry] == '*') {
         [illegible]
         if (gtype == 'T') {
           line = BOTTOM - 3;
           col = 1;
           showtext(cs,BOTTOM,WIDTH);
         }
       }

       /*selection made from within a picture menu*/
       /*say it at once*/
       else if (gtype == 'P') {
         say(mtext[menu][nentry]);
         if (edmode == 'R') { /*wait for speech then flip box back*/
           j = strlen(mtext[menu][nentry])*MSPEED*200;
           for (i = 0; i < j; i ++) k = j / 7;
           flip(msize[menu],nentry,black);
         }
       }
       /*selection made from within a text menu*/
       /*put it on the screen to build a phrase to say later*/
       else {
         if (col != 0) {
           strcat(cs,mtext[menu][nentry]);
           strcat(cs," ");
           eend[eendi++] = strlen(cs);
           showtext(mtext[menu][nentry],BOTTOM,WIDTH);
           showtext(" ",BOTTOM,WIDTH);
         }
       }
       break;
   }
 }
}

/***************************************/

replot() /*clear bottom of screen and show text in cs*/
{
 clrbot();
 line = 22;
 col = 1;
 CUP(BOTTOM - 4,1);
 showtext(cs,BOTTOM,WIDTH);
}

/***************************************/

showtext(s,b,w)

/* shows text in a window down through line b, of width w */
/* on entry line has number of next line to write on */
/* characters that do not fit in the window are not shown */
/* returns index in s of last character shown */

int b; /*bottom line of window*/
int w; /* width of window */
char s[]; /*string to be shown */

{
 char *sl; /* points to end of string */
 char *sb; /* for where unwritten part of string starts in s */
 char *si; /* working string pointer */

    = s;
    = sb + strlen(s);
    /* skip initial blank in column 1 */
    if (col == 1 && *sb == ' ') ++sb;

    /* if string will all fit on this line just print it */
    if (sl - sb < w - col) {
      CUP(line,col);
      printf("%s",sb);
      col = col + sl - sb;
      sb = sl + 1;
    }

    else  { /* won't fit */

      /* try to break string at a blank; if impossible print w characters */
      for (si = sb + w - col - 1; si != sb; --si) {
      if (*si == ' ') break;
      }

      if (*si == ' ') {
        *si = '\0';
        CUP(line,col);
        printf("%s",sb);
        *si = ' ';
        sb = si + 1;
      } /* end if (*si == ' ') */

      ++line;
      col = 1;
    } /* end else */
  } /* end while */
} /* end showtext */

/*********************************/

say(s) /*say the string pointed to by s*/
char *s;
{
 int k;

 putcm(32,0);
 while (*s != '\0') {
   k = *s++;
   putcm(k,0);
 }
 putcm(32,0);
 putcm(13,0);
}

/*********************************/

getin() /*get an input command*/
{
 int k, m;

 m = ginput;
 if (edmode == 'E') m = 'E';

 switch (m) {
   case 'K':  /*Key selection*/
   case 'E':
     return(keyin());
   case 'A':  /*All keys as a single switch*/
     if (gtype == 'P') k = pscan('A');
     else k = tscan('A');
```

```
      case 'S':  /*Single switch (joystick button on Apple/IBM)*/
        if (gtype == 'P') k = pscan('S');
        else k = tscan('S');
        break;
      case 'D':  /*Double switch -- keyboard space bar vs all others*/
        if (gtype == 'P') k = pstep('D');
        else k = tstep('D');
        break;
      case 'T':  /*Joystick socket double switch*/
        if (gtype == 'P') k = pstep('T');
        else k = tstep('T');
        break;
   }
   return(k);
  }

  /**********************************/

  keyin() /*get a key selection from keyboard*/
  {
   int k;

   k = toupper(gkey());
   if (mindex(special,k) != -1) return(-k);
   return (mindex(lrdes,k));
  }

  /**********************************/

  button() /* get joystick button as a switch */
  /* if first button is closed returns 0; if second button is closed returns 1*/
  /* if both or no buttons are closed returns 255 */
  {
   char c;

   outportb(0x201,0xff);
   c = inportb(0x201);
   c = c & 0x30;
   if (c == 0x10) return(0);
   if (c == 0x20) return(1);
   return(255);
  }

  /**********************************/

  debounce() /* make sure button has been up (switch open) a good while */
  {
   int i;

   i = 0;
   do {
     if (button() == 255) i++;
     else i == 0;
   } while (i < 200);
  }

  /**********************************/

  pscan(d) /*scan with a picture menu*/
  char d; /*'A' -- all keys as one switch, 'S' -- single switch*/
  {
   int b;

   entry = 0;
```

```
    flip(msize[menu],nentry,white);

    if (nentry < 5)
       b = pause(grate,15,d);
    else
       b = pause(grate,10,d);

    flip(msize[menu],nentry,black);

    if (d == 'S')
    {
      debounce();
      if (zkey() != 255 && gkey() == EDKEY) return(-EDKEY);
      if (b == 0) break;
    }
    else /* d == 'A' */
    {
      if (zkey() != 255)
      {
        if (gkey() == EDKEY) return(-EDKEY);
        break;
      }
    }

    ++nentry;

    if (nentry == msize[menu]) nentry = 0;

  }

  return(nentry);

}

/**********************************/

tscan(d) /*scan with a text menu*/
char d;
{
  int b, col, line;

  nentry = -1;

  if (d == 'S') debounce();

  /* put cursor on screen */

  while (TRUE) {
    if (nentry == -2) {
      col = WIDTH/2;
      line = 1;
    }
    else if (nentry < (msize[menu]+1)/2) {
      col = 1;
      line = nentry + 2;
    }
    else {
      col = WIDTH/2;
      line = nentry - (msize[menu]+1)/2 + 2;
    }
    CUP(line,col);
    printf("*");
    if (nentry < 0)
      b = pause(grate,25,d);
  [illegible]
    b = pause(grate,15,d);
  else
    b = pause(grate,10,d);

  CUP(line,col);
  printf(" ");

  if (d == 'S')
  {

    /* throw away all keys except EDKEY.  Return if EDKEY hit */
    if (zkey() != 255 && gkey() == EDKEY)
          return(-EDKEY);

    if (b == 0)  /* the button has been pressed */
    {
      if (nentry >= 0)
        break;
      if (nentry == -1)
        nentry = -2;
      else
        nentry = -1;
    }
    else
    {
    if (nentry == -2)
      nentry = (msize[menu] - 1)/2;
    ++nentry;
      if (nentry == msize[menu]) nentry = -1;
    }

  } /* end if (d == 'S') */

  else /* d == 'A' */
  {
    if (zkey() != 255)  /* a key has been pressed */
    {
      if (gkey() == EDKEY)
        return(-EDKEY);

      if (nentry >= 0)
        break;
      if (nentry == -1)
        nentry = -2;
      else
        nentry = -1;
    }
    else
    {
      if (nentry == -2)
        nentry = (msize[menu] - 1)/2;
      ++nentry;
      if (nentry == msize[menu]) nentry = -1;
    }

  } /* end else [ d == 'A' ] */

} /* end while */

return(nentry);

}
```

```
*********************************/
```

```
pstep(d) /*step with picture mode*/
char d; /*'D' for keyboard as double switch, 'T' for 2 switches*/
{
 int k;

 nentry = 0;
 while(TRUE) {

   flip(msize[menu],nentry,white);

   if (d == 'T')
   {
    debounce();
    while ((k = button()) != 0 && k != 1)
      if(zkey() != 255 && gkey() == EDKEY) return(-EDKEY);
    if (k == 0) break;
   }
   else /* d == 'D' */
   {
     if ((k = gkey()) == EDKEY) return(-EDKEY);
     if (k != ' ') break;
   }

   flip (msize[menu],nentry,black);

   ++nentry;

   if(nentry == msize[menu]) nentry = 0;

 }

 return(nentry);
}

/**********************************/

tstep(d) /*step with a text menu*/
char d; /*'D' for keyboard, 'T' for switches*/
{
 int col, line, k;

 nentry = -1;

 k = 254;

 while (TRUE) {
   if (nentry == -2) {
     col = WIDTH/2;
     line = 1;
   }
   else if (nentry < (msize[menu] + 1)/2) {
     col = 1;
     line = nentry + 2;
   }
   else {
     col = WIDTH/2;
     line = nentry - (msize[menu] + 1)/2 + 2;
   }
   CUP(line,col);
   printf("*");

   if (d == 'T')
        debounce();

   if (d == 'D')
      if((k = gkey()) == EDKEY) return(-EDKEY);
    if (d == 'T') while ((k = button()) != 0 && k != 1)
       if(zkey() != 255 && gkey() == EDKEY) return(-EDKEY);

    if ((d == 'D' && k != ' ') || (d == 'T' && k == 0)) {
       if (nentry >= 0)
          break;
       if (nentry == -1)
          nentry = -2;
       else
          nentry = -1;
    }
    else
    {
      if (nentry == -2)
        nentry = (msize[menu] - 1)/2;
      ++nentry;
      if (nentry == msize[menu]) nentry = -1;
    }
    CUP(line,col);
    printf(" ");
  }
 CUP(line,col);
 printf(" ");
 if (zkey() != 255) k = gkey();
 if (k == EDKEY) return(-EDKEY);
 return(nentry);
}

/**********************************/

flip(size,box,color)
/*flips black/white for box on screen with size boxes*/
int size,box;
char *color;
{
 static int x4[4] = {PX4};
 static int y4[4] = {PY4};
 static int h4[4] = {PH4};
 static int v4[4] = {PV4};

 static int x9[9] = {PX9};
 static int y9[9] = {PY9};
 static int h9[9] = {PH9};
 static int v9[9] = {PV9};

 static int x16[16] = {PX16};
 static int y16[16] = {PY16};
 static int h16[16] = {PH16};
 static int v16[16] = {PV16};

 static int x25[25] = {PX25A,PX25B};
 static int y25[25] = {PY25A,PY25B};
 static int h25[25] = {PH25A,PH25B};
 static int v25[25] = {PV25A,PV25B};

 int x,y,h,v;
 int offset,start,row;

 switch (size) {

   case 4:
     x = x4[box];
     y = y4[box];
     h = h4[box];
     v = v4[box];
    break;
    case 9:
      x = x9[box];
      y = y9[box];
      h = h9[box];
      v = v9[box];
      break;

    case 16:
      x = x16[box];
      y = y16[box];
      h = h16[box];
      v = v16[box];
      break;

    case 25:
      x = x25[box];
      y = y25[box];
      h = h25[box];
      v = v25[box];
      break;

    default:
      err("bad size");
      return(-1);
  };

  for (row = y; row <= y + 3; row++) {
    if ((row & 0x0001) == 1) offset = POFFSET; /*odd row*/
    else offset = 0; /*even row*/
    start = offset + 80*(row>>1) + x;
    /*send line for top of box*/
    graphmv(start,color,h);
  }
  for (row = y+4; row <= y+v-4; row++) {
  if ((row & 0x0001) == 1) offset = POFFSET; /*odd row*/
    else offset = 0; /*even row*/
    start = offset + 80*(row>>1) + x;
    /*send edges to screen*/
    graphmv(start,color,1);
    graphmv(start+h-1,color,1);
  }
  for (row = y+v-3; row <= y + v; row++) {
    if ((row & 0x0001) == 1) offset = POFFSET; /*odd row*/
    else offset = 0; /*even row*/
    start = offset + 80*(row>>1) + x;
    /*send line for bottom of  box*/
    graphmv(start,color,h);
  }
}

/*********************************/

kscan(d) /*scan with simulated keyset*/
char d;
{
 int b,i,j,kline,kcol,klen1,klen2;

 klen1 = strlen(keys1);
 klen2 = strlen(keys2);

 i = -2;
 kline = BOTTOM - 1;
 kcol = 1;
 while (TRUE) {

   CUP(kline,kcol);

   if (i < 5 && kline == BOTTOM - 1)
        b = pause(grate,10,d);
   else
        b = pause(grate,10,d);

   if (d == 'S')
         debounce();

   if (zkey() != 255 || (d == 'S' && b == 0))
   {
     if (zkey() != 255 && gkey() == EDKEY) return(-EDKEY);

     if (i >= 0)
       break;
     if (i == -2) {
       i = -4;
       kline = BOTTOM;
     }
     else {
       i = -2;
       kline = BOTTOM - 1;
     }
   }
   else
   {
     if (i == -4) {
       i = -2;
     }

     i += 2;
     kcol += 2;

     if (i == klen1 && kline == BOTTOM - 1) {
        kline = BOTTOM;
        kcol = 1;
        i = -4;
     }
     else if (i == klen2 && kline == BOTTOM) {
       kline == BOTTOM - 1;
       kcol = 1;
       i = -2;
     }
  }
 }
 if (kline == BOTTOM)
    j = keys2[i];
 else
    j = keys1[i];

 if (j == '<') return(13);
 if (j == '?') return(8);
 return(j);

}

/***********************************/

kstep(d) /*step with simulated keyset*/
char d;
  t i,j,k,kline,kcol,klen1,klen2;
```

```
klen1 = strlen(keys1);
klen2 = strlen(keys2);

i = -2;
kline = BOTTOM - 1;
kcol = 1;
while (TRUE) {
  CUP(kline,kcol);

  if (d == 'D')
     if((k = gkey()) == EDKEY) return(-EDKEY);

  if (d == 'T') {
    debounce();
    while ((k = button()) == 255)
       if (zkey() != 255 && gkey() == EDKEY) return(-EDKEY);
  }
  if ((d == 'D' && k != ' ') || (d == 'T' && k == 0))
  {
    if (i >= 0)
       break;
    if (i == -2)
    {
      i = -4;
      kline = BOTTOM;
    }
    else
    {
      i = -2;
      kline = BOTTOM - 1;
    }
  }
  else
  {
    if (i == -4) {
      i = -2;
    }

    i += 2;
    kcol += 2;

    if (i == klen1 && kline == BOTTOM - 1) {
       kline = BOTTOM;
       kcol = 1;
       i = -4;
    }
    else if (i == klen2 && kline == BOTTOM) {
       kline = BOTTOM - 1;
       kcol = 1;
       i = -2;
    }
  }
 }

if (kline == BOTTOM)
   j = keys2[i];
else
   j = keys1[i];

if (j == '<') return(13);
if (j == '?') return(8);
return(j);
```

```
/***********************************/
```

```
getkey()
{
 if (ginput == 'K' || edmode == 'E') return(gkey());
 if (keyson == FALSE) {
   CUP(BOTTOM - 1, 3);
   printf(keys1);
   CUP(BOTTOM, 3);
   printf(keys2);
   keyson == TRUE;
 }
 if (ginput == 'A' || ginput == 'S')
      return(kscan(ginput));
 else return(kstep(ginput));
}

/***********************************/

getstrin(m) /* returns a string in sbuff -- uses current input mode */
int m; /* maximum string to return */
{
 int c,i,j,gline,gcol,scol,limit;

 printf(" >");

 scr_loc(&gline,&gcol);

 limit = WIDTH - gcol - 1;

 if (limit > m)
   limit = m;

 for (j = gcol; j <= gcol + limit; j++);
      printf(" ");

 CUP(gline,gcol + limit);
 printf("<");

 if (ginput != 'K' && edmode != 'E' && keyson == FALSE) {
   CUP(BOTTOM - 1,3);
   printf(keys1);
   CUP(BOTTOM,3);
   printf(keys2);
   keyson == TRUE;
 }

 CUP(gline,gcol);

 ngets(limit);

}

nkey() /* returns a key from appropriate input */
{
 int c, sline, scol;

 if (ginput == 'K' || edmode == 'E')
    return(gkey());

 scr_loc(&sline,&scol); /* save screen cursor position */

 if (ginput == 'A' || ginput == 'S')
      c = kscan(ginput);
   se
      c = kstep(ginput);
 CUP(sline,scol); /* restore screen cursor position */

 return(c);

}

/*********************************/

ngets(n) /* gets a string from stdin of maximum n characters */
/* string goes into sbuff */
int n;
{
 int t,i,l,c;

 t = 0;

 while ((i = nkey()) != 13) {
   if (i == 8) { /* backspace */
     if (t > 0) {
       scr_loc(&l,&c);
       --c;
       CUP(l,c);
       printf(" ");
       CUP(l,c);
       --t;
     }
   }
   else if (t < n) {
     printf("%c",i);
     sbuff[t++] = i;
   }
 }
 sbuff[t] = '\0';
}

/*********************************/

pause(rate,n,d) /* pause specified length of time in scanning mode */
int rate; /* user specified rate */
int n; /* program specified pause constant */
char d; /* input mode */
{
 unsigned int i,j,k,m;
 int r;

 r = rate;
 if (r < 1)
      r = 1;
 if (r > 30)
      r = 30;
 r = 31 - r;

 /* MSPEED is chosen for each machine to make pausing machine independent */
 j = n * MSPEED * r;
 k = j;

 if (d == 'A')
   for (i = 0; i < j; i++) if ((m = zkey()) != 255) return(m);

 if (d == 'S')
   for(i = 0; i < k; i++) if ((m = button()) != 255) return(m);

   turn(255);
```

```
/**********************************/

reverse(s)  /* reverse string s in place */
char s[];
{
  int c, i, j;

  for (i = 0, j = strlen(s)-1; i < j; i++, j--) {
    c = s[i];
    s[i] = s[j];
    s[j] = c;
  }
}

/**********************************/

itoa(n, s)       /* convert n to characters in s */
char s[];
int n;
{
 int i, sign;

 if ((sign = n) < 0) /*record sign*/
      n = -n;         /*make n positieve*/
 i = 0;
 do {  /*generate digits in reverse order*/
     s[i++] = n % 10 + '0';  /*get next digit*/
 } while ((n /= 10) > 0); /*delete it*/
 if (sign < 0)
   s[i++] = '-';
 s[i] = '\0';
 reverse(s);
}

/**********************************/

mindex(s, k)  /* return index of t in s, -1 if none */
char s[];
int k;
{
  int i;

  for (i = 0; s[i] != '\0'; i++)
   if (s[i] == k) return(i);
  return(-1);
}

/**********************************/

mcheck(s) /*recursively check menus for references*/
char *s; /*name of menu to be checked*/
{
 int m,e; /*menu and entry we are now checking*/

 /*find number of this menu*/
 m = 0;
 while(strcmp(mname[m],s)) {
   m++;
   if (m == menuct) err("menu not on list");
 }

 /*if this menu is already marked we are done*/
    (mmark[m] == 'R') return;

  mark this menu*/
 /*check all entries*/
 for (e = 0; e < msize[menu]; e++)
   if (mtext[m][e][0] == '*') mcheck(mtext[m][e]+1);
}

/*********************************/

readmenus()
{
 FILE *fopen();
 FILE *menufile;
 char s[MAXSTRING];

 menufile = fopen("MENUS","r");

 /*read type of menus */
 fgets(s,MAXSTRING,menufile);
 gtype = s[0];


 /*read input mode*/
 fgets(s,MAXSTRING,menufile);
 ginput = s[0];

 /* read scan rate */
 fgets(s,MAXSTRING,menufile);
 s[strlen(s) - 1] = '\0'; /*zap final '\n'*/
 grate = atoi(s);

 /*read number of menus*/
 fgets(s,MAXSTRING,menufile);
 s[strlen(s) - 1] = '\0'; /*zap final '\n'*/
 menuct = atoi(s);

 /* read individual menus */
 for(menu = 0; menu < menuct; menu++)  {

   /* read size of this menu */
   fgets(s,MAXSTRING,menufile);
   s[strlen(s) - 1] = '\0'; /*zap final '\n'*/
   msize[menu] = atoi(s);

   /*read name of menu*/
   fgets(s,MAXSTRING,menufile);
   s[strlen(s) - 1] = '\0';
   mname[menu] = malloc(strlen(s) + 1);
   if (mname[menu] == NULL) err("insufficient memory for menus");
   strcpy(mname[menu],s);
   for (nentry = 0; nentry < msize[menu]; nentry++)  {
     /*read abbreviation - entry pair*/
     fgets(s,MAXSTRING,menufile);
     s[strlen(s) - 1] = '\0';
     mlabel[menu][nentry] = malloc(strlen(s) + 1);
     if (mlabel[menu][nentry] == NULL) err("insufficient memory for menus");
     strcpy(mlabel[menu][nentry],s);
     fgets(s,MAXSTRING,menufile);
     s[strlen(s) - 1] = '\0';
     mtext[menu][nentry] = malloc(strlen(s) + 1);
     if (mtext[menu][nentry] == NULL) err("insufficient memory for menus");
     strcpy(mtext[menu][nentry],s);
   }
 
 lose(menufile);
}
```

```
/**********************************/

showmenu(menuname)
char *menuname;
{
 int temp;
 menu = 0;
 while (strcmp(mname[menu],menuname))  {
   menu++;
   if (menu == menuct) { /* can't find menu on list */
     clrbot();
     CUP(BOTTOM - 3, 19);
     printf("Cannot find a menu named: %s.",menuname);
     pak();
     replot();
     return;
   }
 }


 if (gtype == 'P' && edmode == 'R') display(menu);

 if(gtype == 'T' || edmode == 'E') typeout(menu);

}

/**********************************/

typeout(mnp) /*types out menu whose name is mname[mnp]*/
int mnp;
{
 texton();
 EDU();
 /*write name of menu centered on line 1*/
 CUP(0,WIDTH/2 - strlen(mname[mnp])/2);
 printf("%s",mname[mnp]);
 /*write designators and abbreviations*/
 for (nentry = 0; nentry < (msize[menu]-1)/2; nentry++) {

  /* print left column entry */

  CUP(2 + nentry, 3);
  printf("%c. %s",lrdes[nentry],mlabel[mnp][nentry]);

  /* print right column entry */

  CUP(2 + nentry, WIDTH/2+2);
  printf("%c. %s",lrdes[(msize[menu] + 1)/2+nentry],
          mlabel[mnp][nentry + (msize[menu]+1)/2]);
 }

 /* last line */

 /* print left column entry */

 CUP(2 + nentry, 3);
 printf("%c. %s",lrdes[nentry],mlabel[mnp][nentry]);

  /* print right column entry */

 if ((msize[menu] & 1) == 0) {  /*even number of entries*/
   CUP(2 + nentry, WIDTH/2+2);
   printf("%c. %s",lrdes[msize[menu]/2+nentry],
          mlabel[mnp][nentry + msize[menu]/2]);
 CUP(BOTTOM - 4, 1);

}

/*********************************/

display(mnp) /*displays picture menu whose name is mname[mnp]*/
int mnp;
{
 int fides, r;

 fides = open(mname[mnp],O_RDONLY);
 if (fides == -1) {
    texton();
    EDU();
    CUP(10,10);
    printf("Unable to find picture file called %s",mname[mnp]);
    exit(1);
 }

 /* read past header */
 if (HEADER != 0)
     read(fides,buf,HEADER);

 /* read picture */
 r = read(fides,buf,PSIZE);
 if (r == -1) {
   texton();
   EDU();
   printf("Unable to load picture file called %s",mname[mnp]);
   exit(1);
 }

 close(fides);
 graphon();
 graphmv(0,buf,PSIZE); /*?? calling parameters*/
}

/*********************************/

pak()
{
 CUP(BOTTOM - 1,1);
 if (ginput == 'K' || ginput == 'A' || ginput == 'D' || edmode == 'E') {
   printf("                       Press any key to continue                  ");
   gkey();
 }
 else {
   printf("                     Press any switch to continue                 ");
   while(button() == 255)
           ;
 }
 CUP(BOTTOM - 1);
 ELI();
}

/*********************************/

editmenu()
/* all changes other than to individual entries */
{
 int key;

 rbot();
 CUP(BOTTOM - 4,1);
```

```
printf("Options: A(add a menu); D(delete this menu); E(edit entries); ");
printf("I(input mode);");
CUP(BOTTOM - 3,1);
printf("N(number of entries); R(rename this menu); S(scan speed); ");
printf("T(type of menu);");
CUP(BOTTOM - 2,1);
printf("W(write out all menus); X(do nothing)");
do
{
  CUP(BOTTOM - 2,1);
  printf("W(write out all menus); X(do nothing)");
  /* get choice */
  getstrin(1);
  key = sbuff[0];
  key = toupper(key);

  switch (key) {

    case 'A':
       amenu();
       return;

    case 'D':
       dmenu();
       return;

    case 'E':
       emenu();
       return;

    case 'I':
       imenu();
       return;

    case 'N':
       nmenu();
       showmenu(mname[menu]);
       return;

    case 'R':
       rmenu();
       return;

    case 'S':
       smenu();
       return;

    case 'T':
       tmenu();
       return;

    case 'W':
       wmenu();
       return;

    case 'X':
       return;

    default:
       CUP(BOTTOM - 2, 1);
       ELI();
       break;
  }
} while (TRUE);
```

```
/**********************************/

amenu() /*accepts and performs user request to add menu*/
{
 int key, savemenu, result;
 char type;
 char *sb;

 do {
   clrbot();
   CUP(BOTTOM - 3,1);
   printf("New menuname");
   getstrin(MAXMNAME);
   sb = sbuff;
   while (*sb != '\0')
        *sb++ = toupper(*sb);
   result = addmenu(sbuff);
   clrbot();
   CUP(BOTTOM - 3,1);
   switch (result) {
     case 0:
      printf("Don't forget to refer to the new menu and save all menus");
      break;
     case 1:
      printf("Sorry, no room for another name");
      break;
     case 2:
      printf("Name already exists.");
      break;
    default:
      printf("Addname failure");
   }
   pak();
 } while (result > 1);
  savemenu = menu;
  menu = menuct - 1;
  nmenu();
  menu = savemenu;
}

/**********************************/

dmenu() /*accepts and performs user request to delete menu*/
{
 int result;
 char s[255];

 clrbot();
 CUP(BOTTOM - 3,1);
 if (strcmp(mname[menu],"MAIN") == 0) {
   printf("You cannot delete MAIN");
   pak();
   return(0);
 }
 CUP(BOTTOM - 2,1);
 printf("Do you really want to PERMANENTLY DESTROY this menu? (y/n)");
 getstrin(1);
 if (sbuff[0] == 'y' || sbuff[0] == 'Y') {
   delmenu(mname[menu]);
   showmenu("MAIN");
   return(0);
 
 clrbot();
```

```
/**********************************/

emenu()
/*allows editing of an entry in a menu that is displayed on*/
/*the screen*/
{
 int line, col;

 clrbot();

/* get number of entry to change */
 CUP(BOTTOM - 3,1);
 printf("Select entry to change > ");

 while((nentry = getin()) == -1)
         ;

/* get new label */
 CUP(BOTTOM - 3,1);
 ELI();

 CUP(BOTTOM - 3,1);
 printf("Old label:  %s\n",mlabel[menu][nentry]);
 CUP(BOTTOM - 2,1);
 printf("New label");
 getstrin(MAXLABEL);

 /* add new label to memory */
 mlabel[menu][nentry] = malloc(strlen(sbuff) + 1);
 if (mlabel[menu][nentry] == NULL) err("out of memory space");
 strcpy(mlabel[menu][nentry],sbuff);

/* write new label on screen at appropriate spot*/
 if (nentry < (msize[menu]+1)/2)  {
   col = 6;
   line = nentry + 2;
 }
 else  {
   col = WIDTH/2 + 5;
   line = nentry - (msize[menu]+1)/2 + 2;
 }
 CUP(line, col);
 printf(EBLANKS);
 CUP(line, col);
 printf("%s",sbuff);

 /* get new phrase or action */
 clrbot();
 CUP(BOTTOM - 3,1);
 printf("Old phrase or menu name:  %s\n",mtext[menu][nentry]);
 CUP(BOTTOM - 2,1);
 printf("New");
 getstrin(MAXTEXT);

 /* add new text to menu */
 mtext[menu][nentry] = malloc(strlen(sbuff) + 1);
 if (mtext[menu][nentry] == NULL) err("out of memory");
 strcpy(mtext[menu][nentry],sbuff);

}

/**********************************/

imenu() /* change input mode */
```

```
  char c;

  do {
    clrbot();
    CUP(BOTTOM - 4, 1);
    printf("Input Modes: A(all keys as one switch); D(divided keyboard);");
    CUP(BOTTOM - 3, 1);
    printf("K(keyboard); S(single switch); T(two switches)");
    CUP(BOTTOM - 2,1);
    printf("Old mode was %c; New mode",ginput);
    getstrin(1);
    c = sbuff[0];
    c = toupper(c);
  } while (c != 'A' && c != 'D' && c != 'K' && c != 'S' && c!= 'T');
  ginput = c;
}

/*********************************/

nmenu() /* number of entries in current menu */
{
 int oldsize, e, i;

 clrbot();

 oldsize = msize[menu];

 CUP(BOTTOM - 4, 1);
 if (msize[menu] != 0) {
    printf("Warning -- You will lose last entries permanently if you ");
    printf("reduce size.");
    CUP(BOTTOM - 3, 1);
    printf("Old Size = %d",oldsize);
 }

 if (gtype == 'P')  {
   do {
     CUP(BOTTOM - 2,1);
     printf("Choose a size (4, 9, 16, or 25)");
     getstrin(2);
   } while ( !(sbuff[0] == '4' && sbuff[1] == '\0')
          && !(sbuff[0] == '9' && sbuff[1] == '\0')
          && !(sbuff[0] == '1' && sbuff[1] == '6' && sbuff[2] == '\0')
          && !(sbuff[0] == '2' && sbuff[1] == '5' && sbuff[2] == '\0')
         );
 }

 else {

   do {
     CUP(BOTTOM - 2,1);
     printf( "Choose a size (1 to %d)",MAXENTRIES);
     getstrin(2);

     if (sbuff[0] >= '1' && sbuff[0] <= '9'
         &&
         (sbuff[1] == '\0'
          || (sbuff[2] == '\0' && sbuff[1] >= '0' && sbuff[1] <= '9'))
        )
          i = atoi(sbuff);
     else
          i = MAXENTRIES + 1;

   } while (i > MAXENTRIES);
 msize[menu] = atoi(sbuff);

 if (msize[menu] > oldsize)    /* initialize new entries */
    for (e = oldsize; e < msize[menu]; e++)
         mlabel[menu][e] = mtext[menu][e] = &eos;

}

/********************************/

rmenu() /*accepts and performs user request to rename menu*/
{
 int result;
 char *sb;

 clrbot();
 CUP(BOTTOM - 3,1);
 printf("New name of menu (press ENTER to keep old name)");
 getstrin(MAXMNAME);
 sb = sbuff;
 while (*sb != '\0')
     *sb++ = toupper(*sb);
 if (sbuff[0] == '\0') {
 clrbot();
 return;
 }

 result = rnmenu(sbuff);
 CUP(BOTTOM - 2,10);
 switch (result) {

   case 0:
        break;

   case 1:
        printf("Both names are the same.");
        break;

   case 2:
        printf("No new name specified.");
        break;

   case 3:
        printf("New name already exists.");
        break;

   case 4:
        printf("Illegal to rename MAIN.");
        break;

   case 5:
        printf("Not enough memory to store new name.");
        break;

   default:
        printf("rename failure");
        break;

 }
 if (result == 0 && (gtype == 'T' || edmode == 'E')) {
   /*write name of menu centered on line 0*/
   CUP(0,1);
   ELI();
   CUP(0,WIDTH/2 - strlen(sbuff)/2);
```

```
  }

 pak();

}

/*********************************/

smenu() /* change scan rate */
{
  int i;

  do {
    clrbot();
    CUP(BOTTOM - 3, 1);
    printf("Old speed is %d:  New Speed (1 to 30)",grate);
    getstrin(2);
    i = atoi(sbuff);
  } while (i < 1 || i > 30);
  grate = i;
}

/*****************************/

tmenu() /* changes types of all menus */
{
 int m;

 clrbot();
 CUP(BOTTOM - 3,1);

 if (gtype == 'T')
   for (m = 0; m < menuct; m++)
      switch (msize[m]) {

        case 4:
        case 9:
        case 16:
        case 25:
           break;

        default:
           printf("Illegal size menu for conversion to picture: ");
           printf("%s", mname[m]);
           pak();
           return;
      }

 printf(" You now have ");
 if (gtype == 'P')
      printf("picture");
 else
      printf("text");
 printf(" menus; do you want to change to ");
 if (gtype == 'P')
      printf("text");
 else
      printf("picture");
 printf(" menus? (y/n)");
 getstrin(3);

 if (sbuff[0] != 'y' && sbuff[0] != 'Y')
      return;

 if (gtype == 'T')
```

```
        gtype = 'F';
  else
        gtype = 'T';

 }

/*********************************/

wmenu() /*write out menu*/
{
  int result;

  clrbot();

  CUP(BOTTOM - 3, 1);
  printf("Writing out will overwrite the menus now on the disk.");
  CUP(BOTTOM - 2, 1);
  printf("Are you ready to write? (y/n)");
  getstrin(3);

  if (sbuff[0] != 'y' && sbuff[0] != 'Y')
       return;

  result = outmenus("menus");

  clrbot();

  CUP(BOTTOM - 3,1);
  switch (result) {

    case 0:
         printf("Menus written successfully to file MENUS");
         break;

    case 1:
         printf("Unable to open file MENUS");
         break;

    case 2:
         printf("Unable to finishing writing file MENUS");
         break;

    default:
         printf("Error in writing menus to disk");
         break;
  }
  pak();
 }

/*********************************/

addmenu(newname,type) /*adds a new menu*/
char newname[];
{
  int entnum, mnum;

  /*does a menu already exist with this name?*/
  for (mnum = 0; mnum < menuct; mnum++)
    if (strcmp(newname,mname[mnum]) == 0) return(1); /*name already exists*/

  /*put new menu on list */
  mname[menuct] = malloc(strlen(newname) + 1);
  if (mname[menuct] == NULL) return(2);
   rcpy(mname[menuct],newname);

  /* set entries to empty string */
```

```
    mlabel[menuct][entnum] = mtext[menuct][entnum] = &eos;

  /*We now have one more menu*/
  ++menuct;
  return(0); /*succeeded*/
 }

/**********************************/

delmenu(menuname)
/*delete name from menu list*/
/*delete all references to menu*/
char menuname[];
{
  int mnum, entnum;

  /*illegal to delete MAIN menu*/
  if(strcmp(menuname,"MAIN") == 0) return (1);

  /*delete from list; error if not on list*/
  for (mnum = 1; mnum < menuct; mnum++) {
    if (strcmp(menuname,mname[mnum]) == 0) break;
  }
  if (mnum == menuct) return (2);
  --menuct;
  while(mnum < menuct) {
    mname[mnum] = mname[mnum + 1];
    for (entnum = 0; entnum < msize[menu]; entnum++) {
      mlabel[entnum][mnum] = mlabel[entnum][mnum+1];
      mtext[entnum][mnum] = mtext[entnum][mnum+1];
    }
    ++mnum;
  }

  /*remove all references to menuname*/
  for(mnum = 0; mnum < menuct; mnum ++) {
    for(entnum = 0; entnum < msize[menu]; entnum++) {
      if (mtext[mnum][entnum][0] == '*'
           && strcmp(menuname,mtext[mnum][entnum]+1) == 0)
         mlabel[mnum][entnum] = mtext[mnum][entnum] = &eos;
    }
  }
}

/**********************************/

rnmenu(newname) /*rename menu and all references to it*/
char newname[];
{
  int mnum, entnum;
  char *starname;

  /*check for errors*/

  /* both the same?*/
  if (strcmp(mname[menu],newname) == 0) return(1);

  /*empty string?*/
  if (strlen(mname[menu]) == 0 || strlen(newname) == 0) return(2);

  /*newname already exists?*/
  for (mnum = 0; mnum < menuct; mnum++)
    if (strcmp(newname, mname[mnum]) == 0) return(3);

  /*illegal to rename "MAIN"*/
 /*set pointers to name and name with star*/
 starname = malloc(strlen(newname) + 2);
 if (starname == NULL) return(5);
 starname[0] = '*';
 strcpy(starname + 1, newname);

 /*replace each reference to oldname with a reference to newname*/
 for (mnum = 0; mnum < menuct; mnum++)
   for (entnum = 0; entnum < msize[menu]; entnum++)
     if ((mtext[mnum][entnum][0] == '*')
          && strcmp(mname[menu], mtext[mnum][entnum] + 1) == 0)
          mtext[mnum][entnum] = starname;

 mname[menu] = starname + 1;

 return(0);

}

/***********************************/

outmenus(filename)  /*collects garbage & outputs menus to filename*/
char filename[];
{
 FILE *ofp;
 int mct, entnum, mnum;
 char s[5];

 /*tenatively mark all menus as unreachable*/
 for (mnum = 0; mnum < menuct; mnum++) mmark[mnum] = 'U';

 /*now mark all menus reachable from main*/
 mcheck("MAIN");

 /*now write out menus*/
 ofp = fopen(filename,"w");
 if (ofp == NULL) return(1);

 /* write out type */
 aputc(gtype,ofp);
 aputc('\n',ofp);

 /* write out input mode */
 aputc(ginput,ofp);
 aputc('\n',ofp);

 /* write out scan rate */
 itoa(grate,s);
 fputs(s,ofp);
 aputc('\n',ofp);

 /*count number of referenced menus and write out the count*/
 mct = 0;
 for (mnum = 0; mnum < menuct; mnum++) if (mmark[mnum] = 'R') ++mct;
 itoa(mct,s);
 fputs(s, ofp);
 aputc('\n',ofp);

 /*write out referenced menus*/
for (mnum = 0; mnum < mct; mnum++) {
   if (mmark[mnum] == 'R') {

    /* write out size */
    itoa(msize[mnum],s);
```

```
      aputc('\n',ofp);

      /* write out menu name */
      fputs(mname[mnum],ofp);
      aputc('\n',ofp);

      /* write out menu entries */
      for (entnum = 0; entnum < msize[menu]; entnum++) {
        fputs(mlabel[mnum][entnum],ofp);
        aputc('\n',ofp);
        fputs(mtext[mnum][entnum],ofp);
        aputc('\n',ofp);
      }
    }
  }
 fclose(ofp);
}

/**********************************/

clrbot()
/*clears bottom four lines of screen*/
{
 CUP(BOTTOM - 4,1);
 ELI();
 CUP(BOTTOM - 3,1);
 ELI();
 CUP(BOTTOM - 2,1);
 ELI();
 CUP(BOTTOM - 1,1);
 ELI();
 CUP(BOTTOM,1);
 ELI();
 keyson = FALSE;
}

/*********************************/

err(s)
char s[];
{
 texton();
 CUP(BOTTOM - 1,1);
 ELI();
 printf("%s",s);
 exit(1);
}

/*********************************/
```

```
; Copyright (C) 1983  Thomas Fenwick
; :ts=8
;Copyright (C) 1983 by Manx Software Systems
Cg      group   CODESEG
CODESEG segment word public 'code'
        assume  cs:Cg
        public graphmv_
graphmv_ proc near
;
;syntax is like blockmv, but moves stuff to graphics screen segment
;
        mov     bx,sp
        pushf
        cld
        push    es
        push    si
        push    di
        mov     ax,0b800h
        mov     es,ax
        mov     di,2[bx]
        mov     si,4[bx]
        mov     cx,6[bx]
        mov     dx,cx
        shr     cx,1
        jz      mv_skip
rep     movs    word ptr [di], word ptr [si]
mv_skip:
        test    dl,1
        jz      done
        movs    byte ptr [di], byte ptr [si]
done:
        pop     di
        pop     si
        pop     es
        popf
        ret
graphmv_        endp
;
        public  graphon_
graphon_ proc   near    ;turns on med-res b & w graphics
        push    si
        push    di
        push    bp
        mov     ah,0
        mov     al,5
        int     10h
        pop     bp
        pop     di
        pop     si
        ret
graphon_ endp
;
        public  texton_
texton_ proc    near    ;turns on text (b & w)
        push    si
        push    di
        push    bp
        mov     ah,0
        mov     al,2
        int     10h
        pop     bp
        pop     di
        pop     si
```

```
texton_  endp
;
CODESEG  ends
         end
```

```
;last edited by p.m.
;12:06 p.m., Dec. 11, 1984
Cg       group    CODESEG
CODESEG segment word public 'code'
        assume  cs:Cg
        extrn   $sav:near
        public  gscan_
gscan_  proc    near
        mov     ax,0
        call    $sav
        mov     ah,0
        int     16h
        mov     al,ah   ;we want scan code
        mov     ah,0
        ret
gscan_  endp
;
  public zkey_
zkey_ proc near
  mov ax,0
  call $sav
  mov ah,1
  int 16h
  jz short zky ;if zero, no code available
  mov ah,0
  ret  ;ax has code
zky:
  mov ax,255
  ret
zkey_ endp
;
  public gkey_
gkey_ proc near
  mov ax,0
  call $sav
 mov ah,0
  int 16h
  mov ah,0
  ret
gkey_ endp
;
        public  putcm_
putcm_  proc    near
        mov     ax,0
        call    $sav
        mov     ax,word ptr 8[bp]
        mov     ah,1
        mov     dx,word ptr 10[bp]
        int     14h
        ret
putcm_  endp
;
CODESEG ends
        end
```