To the Graduate Council:

I am submitting herewith a thesis written by Jane H. Kieninger entitled "Problems in the Design and Installation of a Microcomputer Based Speech Output System for Handicapped Students and Faculty." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

J. Michael Moshell, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Vice Chancellor
Graduate Studies and Research

PROBLEMS IN THE DESIGN AND INSTALLATION OF A MICROCOMPUTER BASED SPEECH OUTPUT SYSTEM FOR HANDICAPPED STUDENTS AND FACULTY

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Jane H. Kieninger

June 1981

3051854

ACKNOWLEDGMENTS

The author wishes to thank Dr. Michael Moshell, Dr. David Straight, and Dr. Robert Shelton for serving as committee members, and Mike Sappington for his assistance in the University of Tennessee's Mini Lab. A special thanks goes to Kathy Sowder for all her time and assistance which made this project possible.

ABSTRACT

The purpose of this thesis paper is to provide a review of the current state of microcomputer based speech output systems and to relate the problems encountered in attempting to install one on a micro computer at the University of Tennessee so that it could be used as an aid to handicapped students and faculty there. It is hoped that the information provided within this thesis and appendixes will serve as a guideline for a future successful installation.

TABLE OF CONTENTS

SECTION		PAGE
I.	VOICE SYNTHESIZER REVIEW	1
1.	Voice Analysis	1
2.	Synthesis by Rule	3
3.	Applications	11
II.	METHODS AND PROCEDURES/PROBLEMS ENCOUNTERED	13
1.	Obtaining a Voice Synthesizer	13
2.	Installing and Testing Synthesizer/SPEAKIT	16
3.	TALKER Program: Loading and Cross Assembling	20
4.	Downloading to IMSAI/TENCOM, DUMBT and TELNET	23
5.	TALKER Program: Modifying and Testing	29
6.	VSK - DEC 10 Communication	33
III.	DISCUSSION	35
1.	University Survey	35
2.	Devices for the Blind	37
3.	Voice Output Quality	37
4.	VS-6 at UT	38
5.	TALKER Improvements	39
IV.	CONCLUSIONS	41
LIST OF REFERENCES		42
APPENDIXES		45
APPENDIX A.	VENDOR ADDRESSES AND TELEPHONE NUMBERS	46
APPENDIX B.	CROSS/COLON PROGRAMS	47
APPENDIX C.	PROBLEMS WITH TENCOM, DUMBT, AND PASCAL	51
APPENDIX D.	TELNET - CHECKLIST FOR EXECUTION	55
APPENDIX E.	TALKER - CHANGES AND INTERNAL PROCEDURES	57
APPENDIX F.	TALKER USER'S GUIDE	63
APPENDIX G.	BLIND STUDENT QUESTIONNAIRE	66
APPENDIX H.	UNIVERSITY CONTACTS	68
APPENDIX I.	MISCELLANEOUS USEFUL INFORMATION	70
VITA		71

I. VOICE SYNTHESIZER REVIEW

Speech output systems available on today's market are one of two basic types, both of which are referred to as voice synthesizers. The first type is called a Voice Analysis or a Waveform Encoding synthesizer. This type takes voice input via microphone, digitalizes it and stores the speech for later recall. The second type is referred to as Synthesis by Rule or Parameter Encoded Speech synthesizers. This type is driven by phonetically coded parameters which are provided directly by the user or supplied through some English text to phonetic code software.

1. VOICE ANALYSIS

The Voice Analysis technique, also referred to as Stored Speech or Automatic Speech Recognition (ASR), currently provides the best quality of speech. This system is speaker dependent and requires a lot of storage for the amount of speech input. The storage requirement results in a limited vocabulary being stored and very little flexibility in the use of the system.

The Voice Analysis technique attempts to reproduce natural speech by generating a waveform similar to that of the speech input. The concept uses an analog-to-digital converter to sample analog speech waveforms and convert it to a digitalized format. This format is stored in memory and played back through a digital-to-analog converter and a low-pass filter. (7)

Voice Analysis is sometimes referred to as Phrase or Word Concatenation since complete phrases or individual words are usually stored. If complete phrases are stored, the pitch, rhythm and loudness of the speech is stored as recorded. The result is very natural speech. A large amount of storage is required for Phrase Concatenation, and no amount of flexibility is allowed within a phrase. Word Concatenation is the same concept except that it makes use of a large stored word dictionary. This technique allows for more flexibility, but proper rhythm, pitch, inflexion and loudness may be lost. In both cases storage is a problem. Storage requirements vary from 300 to 12 K bytes per 1 second of speech depending on the technique used and the amount of noise permitted. (7)

When surveying the commercial market to find samples of voice input synthesizers and parameter encoded synthesizers, it was difficult to determine the type of synthesizer described. It was discovered that unless the literature specifically mentioned voice input or the use of a microphone that the synthesizer being described would most likely require phonetic parameter input or input into an English text-to-phonetic front end processor. Below is a sampling of voice input synthesizers currently on the market. Addresses and telephone numbers of the vendors are listed in Appendix A.

MIMIC Electronics sells the DATA-BOY Speech Processor. It is a speech digitizer that gives natural sounding speech output. The user speaks into a microphone to store the speech. A standard floppy (240,000 bytes) can store approximately 240

seconds (approximately 240 words) of continuous speech. This system can interface to any input/output port which uses TTL-compatible signals. (1)

The SuperTalker is a voice input speech synthesizer designed for the Apple Computer and made by Mountain Hardware Inc. Spoken words are input into Random Access Memory (RAM) through a microphone and at that point they can be accessed like any other stored data. The SuperTalker package includes a SuperTalker card with plugs into a peripheral slot on the Apple II, a microphone, a loudspeaker, operating system software and documentation, and two "ready to run" programs. This system stores about 50 seconds (about 50 words) on an Apple II diskette. The 1980 cost was $279 and can be purchased through most micro computer stores. (11)

Master Specialties sells a voice input synthesizer called a Model 1700 voice readout system. It can store 16 to 32 words plus digits and a few other items. Its Model 1650 is a solid state model. It has a programmable read out system and can store up to 160 words.

2. SYNTHESIS BY RULE

Synthesis by Rule synthesizers do not yet supply the quality voice output that the voice analysis synthesizers can, but the flexibility and small storage requirements make this type synthesizer the more useable. Since the synthesizer is parameter driven, it is not associated with a particular voice and is therefore speaker independent. The vocabulary available is

infinite since each synthesizer system is driven by a limited number of discrete phonetic codes which the manufacturers proclaim can reproduce most speech utterances. Storing a phonetic word takes up considerably less space than its digitalized vocal equivalent. A word can be stored in its phonetic format or can be supplied by the user in English text which would be converted to phonetic codes by a software interface before being processed by the synthesizer. The quality of the synthesis by rule synthesizer is not as good as a voice analysis synthesizer since the first one tries to simulate the human voice track rather than duplicate the voice waveform as the second does.

Texas Instruments was really the forerunner in the commercial market for voice output systems with its production of the SPEAK & SPELL educational aid for children. The aid speaks words for children to spell and sells for $40 to $50. The device uses a TMC 0280 speech synthesizer, TMC 0350 ROM, and TMC 0270 controller. The system uses "canned" words and phrases that are stored in 2 128K byte ROMS in the exact format necessary for the synthesizer. SPEAK & SPELL has a vocabulary of over 200 words. The synthesizer chip duplicates human speech on the integrated circuit level using a technique called Linear Predictive Coding described below under "TI Voice Systems." This synthesizer chip is not for sale. (17)

Texas Instruments Inc. (TI), VOTRAX, and COMPUTALKER are the major producers of synthesis-by-rule synthesizers. VOTRAX sells much of their equipment at the chip or board level and

in volumes such that it can be incorporated into a final product sold by another vendor. Therefore, many voice output systems carrying their own manufacturer's label may actually contain a VOTRAX synthesizer. For example, Radio Shack's voice synthesizer for its TRS80 computer is actually a low quality VOTRAX synthesizer made specifically for the TRS80 (sells for approximately $400). The Kurzweil Reading Machine discussed later in this paper also uses a VOTRAX synthesizer. (16)

What these systems have in common is the concept of converting English text into a phonetic form and then further process the phonetic form to drive the voice synthesizers. In VOTRAX synthesizers an English word must be converted into phonemes by the user, where phonemes are considered to be the "smallest fundamental units of speech." (16) The VS6 (one of the top of the line VOTRAX models) uses input from 61 possible phonemes. (16) A COMPUTALKER synthesizer system requires similar phoneme input called "linguistic feature codes." TI chooses to break the phoneme down further into allophones. A phoneme is actually a set of allophones. Since several sounds which although normally perceived and categorized as the same phoneme were actually slightly different when used in a specific context or environment, TI chose to use allophones to capture this subtle difference and get a more accurate sound. (7) TI has 128 possible allophones. Stress or inflection may be added to these phonemes or allophones to improve the quality of speech.

TI Voice Systems

TI voice systems now offer a text-to-speech translator (software program) that analyzes English text input, converts it to allophone codes and then adds stress and intonation. The allophones then pick up parameter information related to them that is stored in a (3K approximately) allophone library. The parameters of the allophones that are concatenated are then smoothed to make sound transitions less abrupt. The result is then fed to the TMS 5200 speech synthesizer.

The TMS 5200 speech synthesizer uses the Linear Predictive Coding Method (LPC). This technique employs a mathematical model of the human vocal tract and is implemented as a filter network. (7) The parameters supplied from the allophone library are the coefficients used in the model. The model outputs the information needed to determine the signal's pitch, amplitude and correct voicing (16). The results feed a digital to analog converter connected to an amplifier and speaker. TI's TMS 5200 can operate on any standard 8-bit microprocessor.

COMPUTALKER Voice Systems

COMPUTALKER offers its voice synthesizer and its synthesis by rule software (CSR 1) to Apple II users (CT-1A), to TRS80 users (CT-1T) and to its s100 bus based computers (CT-1) for $695 (1980 price). This price includes hardware and software and a manual. You must add a speaker or headphone. The CSR1 software includes the object code as well as the 8080

source code. 16K of RAM is required to operate the system but 32 K is recommended.

The input required for the CSR 1 to drive the synthesizer are phonetic codes representing the English text. COMPUTALKER's phonetic codes are ASCII characters which closely resemble ordinary English spelling. The CSR 1 software then takes these codes and converts them into a "linguistic feature code." A set of English phonetic rules then modifies the codes in accordance with the phonetic structure of the English language. The results are then fed to a numerical algorithm which generates a smooth sequence of data values. The output is 9 control parameters which drive 9 real time data channels into the synthesizer. Each channel "specifies an acoustic aspect of the speech signal" (2). For S100 buss based computers a front end text processor that converts English text into phonetic codes is available at the 1980 costs of $55 for the software and $25 for the documentation (user's manual).

The synthesizer board contains a series of oscillators and filters that attempt to be an electronic simulation of the sounds made by the larnyx and mouth. The synthesizer board alone sells for $400 (1979). (3)

VOTRAX Voice Systems

VOTRAX offers its VS 6.0 Electronic Voice System for a basic cost of from $3,500 to $4,500 (1979 prices). The VS 6.4 is an improved version of the VS 6.0 (it naturalizes voice

output) and sells for about $5,500 (1979 prices). Both systems offer a large selection of computer interface types, and phoneme coded input is required to drive the synthesizer.

VOTRAX has just recently introduced the "Talking Chip" (February, 1981), the new SC-01. The SC-01 contains its latest synthesizer. The synthesizer expects a dictionary of stored phonetic vocabulary to reside in memory. The chip alone if bought in large quantities is about $12 (1981 prices). If bought as part of a new CDS II Development System the price is $15,000. This system includes a 6808 Motorola microprocessor, a SC-01 Speech chip, English text-to-phonetic speech algorithm, a CDS operating system stored in 24K bytes of ROM and 2K bytes of RAM work space. It also includes a Centronics 730 printer and a RS-232 interface for interfacing to a host computer. The drawback to this system is that the phonetic output to this system must be output to a host computer or some other mechanism so that the codes can be burned into PROM which the synthesizer will access. This system is intended for commercial users who wish to access non-volitile pre-stored speech. It also gives them a mechanism for creating and storing the phonetic codes. (7)

Maryland Computer Services sells top of the line VOTRAX synthesizers with Maryland's software interface. They have just recently (Spring, 1981) announced a speech terminal called TOTALTALK which includes the actual CRT and keyboard as well as synthesizer, memory, English text to phoneme programs and other software. It sells for $5,995 (1981 prices).

This new system reportedly has significantly improved voice quality over other systems currently on the market.

The VOTRAX synthesizers are formant type synthesizers. Formants are the "major resonant frequencies of speech" (5) and are produced by analog bandpass filters. VOTRAX uses a phoneme translator to translate the phoneme codes to parameters that drive a vocal tract model. The model contains a "variable frequency oscillator for simulating the vocal chords and a pseudorandom signal generator that simulates the sound of rushing air." (5) The output of this passes into a bank of four analog bandpass filters which feed an audio amplifier and finally an external amplifier. (5)

A. I. Cybernetics Voice System

A. I. Cybernetics also has a Model 100 Speech Synthesizer which is compatible to the Altair, IMSAI and Poly-88 bus structured computers. The input is phonetically encoded and the output is a robot sounding voice. (13)

See Appendix A for the addresses and telephone numbers of the above vendors and for additional vendors, see (16).

English Text Translator Programs

In order to successfully use any of these phonetically driven synthesizers, an English text to phonetic code translator is essential. H. Elovitz and group (4) developed a translator that was tested with a VS 6 (VOTRAX) but was designed to be portable to any phonetically driven synthesizer. They devised a set of 329 letter-to-sound rules where each

rule specifies the pronunciation of one or more letters in a given text. These rules take English text and using basic linguistic rules translate it into a set of standard phonetic codes (IPA-International Phonetic Alphabet). The context of the letters plays an important part in the evaluation of a letter or group of letters pronunciation. Many text to speech systems include a pronouncing dictionary for common words--if a lookup fails, the rules are examined. Adding a sizeable dictionary requires considerably more storage. The more sophisticated systems examine the part of speech of the word to help determine the correct pronunciation. After the words are translated into phonetic codes, another set of rules is then applied to convert these IPA codes to the codes required by a specific voice synthesizer--in their case a VS 6. (4)

Cassette Demo Tapes

Cassette Demo Tapes are available from several of the synthesizer manufacturers and must be scrutinized carefully when being used to evaluate the quality of the product. Several demo tapes were listened to. The tape of COMPUTALKER was contrasted to that of a VOTRAX. The quality of some of the COMPUTALKER demo is excellent but it is of their voice input synthesizer not their parameter driven synthesizer. The section which demonstrated the use of their parameter driven synthesizer was only of moderate quality and compared almost equally to the moderate quality of the VOTRAX demo. If the

demo contains some text with which you are familiar (a nursery rhyme, the alphabet, for example), then the speech is understandable; otherwise the robot-like speech is difficult to understand. It was found that if the speech was hand edited that the quality improved. (2) (19)

3. APPLICATIONS

Voice input and output equipment and software for microprocessor based systems are now readily available on the commercial market. One reason for this availability is the expanding realm of practical applications. One of the major application areas is in aiding the handicapped. The Kurzweil Reading Machine gives blind individuals the ability to read any book, magazine or typed transcript. The typed page is scanned by a high resolution camera which converts the image to its ASCII character representation. This data is then passed on to the same type of text-to-speech technology employed in this project. The Kurzweil Report (6) describes a Talking Terminal for the blind which is now being produced. The terminal is actually a microcomputer with the text-to-speech software, synthesizer and speaker hardware included, and (must be) can be attached to any computer terminal. The cost is approximately $3,700 (1979). The article does not indicate whether this terminal includes a communication interface to a large mainframe.

Mark Dahmke, while a student at the University of Nebraska, set up a system that would allow a fellow student, Bill Rush,

a quadraplegic with no speech, to talk. With a long stick attached to a headband, Bill could type on a computer terminal. Mark's system was an 8080 based system that used COMPUTALKER's CT-1 synthesizer and their CSR1 synthesis by rule interface. The CSR1 requires phonetic input which Mark's program supplied from an internally stored table or which Bill input through the keyboard. The table was a table of English words and their phonetic equivalent so that Bill could enter in the English word instead of the phonetic code. The system allowed Bill to speak a full sentence at a time. (3)

The new voice technology opens the way for a touch tone telephone to be used as a terminal to any variety of computers. Data input via the tones can be digitalized and initiate a program whose output could then be vocalized. Banks could use this technology to allow their customers to inquire on the current balance of their accounts, pay bills, or transfer funds. Stock brokers could give their customers stock exchange information.

Anything a visual mechanism passively displays can now be vocalized. (19) Security breaches, or industrial accidents or warnings could employ a voice system that would instantaneously indicate the problem. The system could even include a telephone message and dial up procedure to alert the police or fire stations.

II. METHODS AND PROCEDURES/PROBLEMS ENCOUNTERED

In the spring of 1980, The University of Tennessee had 2 blind students enrolled in the Computer Science program and one blind math professor using some of the computer facilities. All were completely dependent on the use of readers to access the computer. At the same time, speech output systems for microcomputers were becoming more readily available. The primary objective of this project was then to attempt to add speech capability to one of UT's microcomputers and thus give our blind students and faculty members readerless access to a microcomputer. It was hoped that the micro could also be interfaced to UT's DEC 10 and become a "talking terminal" for DEC 10 access.

1. OBTAINING A VOICE SYNTHESIZER

The first step in this project was to obtain a voice output system. In order to do that, it was necessary to review the types of systems on the market, their prices, and their compatibility to UT's microcomputers. It was decided that a voice output system using a voice analysis synthesizer would not be appropriate for this thesis project. Voice analysis systems require that all letters and words or phrases to be used must be stored in an on-line dictionary, and this would cause storage problems on a micro based system. Limiting the dictionary would result in very little flexibility in speech capability. Additionally, voice analysis systems are

speaker dependent. Since the input into the system is via a microphone the same individual would be required to input all speech data in order to achieve any consistency in vocal output.

Parameter encoded synthesizers did not impose either a speaker dependency or the same storage problem, and offered additional flexibility. Since the input into this type of synthesizer is through phonetic codes it is in no way related to a speaker. Storing the phonetic code data for a word takes significantly less space than that needed for storing the wavelength form data (in some cases 8 to 10 bytes versus 1,000 bytes of data); and thus storage requirements are much less.

This type of synthesizer also offered the flexibility needed to operate as a talking terminal to a mainframe computer. It was determined that if an English text to phonetic code software was a part of this type of synthesizer then the data being transmitted to the DEC 10 as well as the data being returned from the DEC 10 could be vocalized.

UT currently supports the operation of an Apple II and 3 IMSAI micro computers in its Mini Lab. The IMSAI's are S100 bus based microprocessors with a Z80 chip which can process 8080 as well as Z80 assembled code. The voice output system to be selected had to be capable of interfacing with one of these two types of computers.

COMPUTALKER's CT-1 synthesizer and English text to phonetic code processor software would interface with the IMSAI

at a cost of approximately $800. COMPUTALKER's CT-1A synthesizer could have been interfaced to the Apple II but the text processor software was not available. At this initial state of the project (Spring, 1980), TI had not made available their TMS5200 synthesizer and text processor software to the commercial market and therefore was not considered.

VOTRAX was contacted and would sell their VS6 synthesizer to operate with the IMSAI for a cost of $3,500 to $4,500. This price did not include the price of an interface board needed to operate on the IMSAI or any English text to phonetic code processor software. VOTRAX would sell an RS232 interface board for an additional $4,400. Diane deHaven of VOTRAX recommended contacting Mr. Dean Blazie of Maryland Computer Services (301-879-3366) for a better price on a VOTRAX synthesizer. She said that Computer Services had purchased a large volume of synthesizer boards at a reduced rate and was reselling them at a cost of approximately $800. This price again did not include an interface board. She also recommended contacting Peter Maggs at the University of Illinois (217-333-6711) since he had been working with an English text to phonetic code translator using a VOTRAX synthesizer on a TRS80 computer.

Peter Maggs was contacted and he offered to lend UT a VOTRAX synthesizer which could be plugged directly into the S100 buss of the IMSAI without the need for an interface board. He also offered to send us the 8080 code for an English text to phonetic code translator program on a DEC 10

tape. The synthesizer offered was a VSK module which is of lower quality than the VS-6 and offers no stress or inflection capability--it is comparable to the quality of the one used in the Radio Shack synthesizer. The translator program was written to operate with a VS-6 synthesizer and a TRS80 computer (Radio Shack). It would require slight modifications to interface with the VSK as well as changes in the input/output procedures and screen handler to conform to the IMSAI hardware configuration. He would include documentation on the changes needed to operate with the VSK synthesizer. The copy of the translator tape would be sent on a DEC 10 tape since the IMSAI operating systems could not read a disk created under a TRS80 operating system. The tape would then have to be downloaded from the DEC 10 to the IMSAI through a communication interface program.

The decision was made to accept Peter Maggs' offer. A thesis project had just recently been completed that included a program to transfer files to and from the DEC 10 (TENCOM), and this program would give the communication interface needed to download the translator program onto the IMSAI.

2. INSTALLING AND TESTING SYNTHESIZER/SPEAKIT

The VSK 11 synthesizer and a tape containing the translator software program was received at UT in the fall of 1980. Kathy Sowder, a lab assistant in the UT Micro Lab, purchased a speaker, connected it to the synthesizer and plugged the synthesizer into a S100 buss slot in IMSAI #1. The synthesizer

was found to actually need the space for 2 slots to allow room for the speaker plug. The synthesizer and speaker were later installed on the IMSAI #2 due to continuing disk drive problems on the IMSAI #1.

VSK Synthesizer

The VSK synthesizer module operates on the IMSAI's S-100 buss as a single I/O port (Hex address DF). Input from this port provides the status of the synthesizer while phonemes to be vocalized are output to the port. Bit 7 (highest order bit) is the busy status of the input port byte. Bit 7 will be zero when the synthesizer is busy and will become one when it is ready to receive another phoneme. The last phoneme of any string must be a null character (Hex FF) to "turn off" the synthesizer. The synthesizer will repeat the last phoneme every 350 microseconds unless the null phoneme has been transmitted. There is a significant time delay between the time the synthesizer receives the phoneme and alters the busy status bit to reflect the receipt. To prevent transmitting another phoneme before the busy bit has been set (or before the phoneme has been processed) it is necessary for the user program to loop, waiting on the status to go busy, immediately after transmitting a phoneme.

The synthesizer is expecting a stream of 1 byte phonemes with pauses and null characters transmitted as phonemes. The phonemes are processed and spoken serially as they are received. The VSK on the IMSAI expects the phoneme codes in

hexidecimal form in the low order 6 bits of the output port byte. The top 2 bits are used in the VS-6 VOTRAX synthesizer and other models to add stress and inflection but are not operable on this model. The codes are obtained from a table of VSK phonetic codes in the "MODEL S100-VSK SYNTHESIZER INTERFACE SPECIFICATIONS" (July 1, 1978). Before presenting the output port with one of these phonemes the top 2 bits must be made zero and the entire byte complemented.

SPEAKIT

A program was then written to test the voice synthesizer. The program called SPEAKIT was written in Z80 source code and tested under USCD Pascal's operating system. Since TENCOM, the program that would transfer over the translator program, was written in Pascal, the USCD Pascal operating system would have to be used to run this program and the file created would be accessible only through Pascal. Therefore, for conformity's sake, the test program was also stored and tested under Pascal. The program was written in Z80 rather than 8080 source code since it was discovered at that time that the version of Pascal currently at UT did not have an 8080 assembler even though Pascal would execute the binary file of either Z80 or 8080 source code. The program was later modified to assemble and execute under the CP/M operating system as well (see 4. Downloading to IMSAI).

SPEAKIT allows a string of phonemes to be entered through the IMSAI keyboard and, when a period is entered, the

string is vocalized. The program begins with a colon, ":", prompt, then expects a series of two character hexidecimal (hex) phonetic codes separated by a single space. The space indicates the end of the code--if more than 2 hex characters are entered before a space is encountered, all but the last 2 characters are ignored. A comma following a space will indicate the end of the word and a pause phoneme will be transmitted by the program. A period following a space indicates the end of a phrase and a null character will be transmitted. If no invalid hex codes are received, the complete phrase will be vocalized at this point. By entering a carriage return as the first character in a string the phrase will be revocalized. A sample input for "one, two, three" would look like:

:2D 23 31 31 0D 0D ,2A 36 36 37 37 3E ,39 39 2B 2C 29.

Pause phonemes need not be entered since the program transmits them.

Up to four separate phrases can be entered and temporarily stored during the program session. By entering a Control A, B, C, or D as the first character in a string, the string that is next entered will be stored in one of four locations (A, B, C or D). To recall a phrase stored, a control letter should be entered followed by a carriage return. The carriage return will cause the phrase stored at that location to be vocalized. Note that unless another control A, B, C or D is entered, the next phrase entered would overlay the one just recalled.

When the string is first entered through the keyboard, it is stored in a buffer to allow for deleting or backspacing over characters. Before the hex characters are converted to the 6-bit binary hex code needed for the synthesizer it is echoed a second time to the screen and a "?" will be printed if an invalid hexidecimal code is entered. The phrase will not be vocalized until all phonemes are checked for errors and converted to binary. The program is stored on a floppy disk in UT's Mini Lab.

3. TALKER PROGRAM: LOADING AND CROSS ASSEMBLING

While the test program for the synthesizer was being developed and tested, the tape sent from Peter Maggs containing the English text to phonetic code translation program was being loaded and assembled onto the DEC 10. Neither loading nor assembling was a simple task. Professor Maggs, when offering the program on the DEC 10 tape, had stated that the tape was a standard DEC 10 tape and had been created through the DEC 10. It is doubtful that this was the case. To load the tape, a special alien tape number had to be obtained from UT's Computer Center (UTCC). Then the tape had to be submitted over the counter at UTCC. A tape is held for processing at UTCC only 12 hours and then is returned to a tape rack to be picked up by the user (the user can request its return before that time). After several attempts to load the tape it was discovered that the tape did not contain

standard DEC 10 ASCII characters and therefore could not have been created through the DEC 10. Charles Boyd of the UTCC patiently gave his assistance by manipulating the data and testing for all possible ASCII configurations. The characters were finally determined to be Hewlett Packard defined 8-bit ASCII characters. The tape was most likely created through a Hewlett Packard to DEC 10 interface and not created on a DEC 10.

A small SOS (MIC) file was created on the DEC 10 containing the following code to copy the tape onto a DEC 10 disk area:

```
DSKC:TALKER.DAT=MT:/BLOCK:110/RECORD:110/MODE:HPASCII/
       NOCRLF/INDUST
```

TALKER.DAT is the name of the file to be created from the tape. The tape contained 110 character records unblocked and was industrial standard (INDUST). This code was executed using the following commands issued under DEC 10 monitor mode:

```
.MOUNT U6801
.R CHANGE
```

The first instruction contained the special alien tape number given to us by UTCC and the second instruction was "RUN CHANGE" where CHANGE was the name of the SOS file created above. The copy was successful.

An 8080 source code cross assembler called CROSS was said to exist on the DEC 10. Initially it seemed to be a good idea to make all the changes to Maggs' translation

program (TALKER) on the DEC 10, assemble it on the DEC 10 and then download the smaller binary file to the IMSAI. The TALKER source file takes up approximately 191 blocks on the DEC 10 or about 50k bytes and would be a time-consuming task to assemble on the IMSAI. It would most likely have to be assembled in parts on the IMSAI. When it was discovered that the version of USCD Pascal available for use by our IMSAI did not contain an 8080 source code assembler (had Z80 assembler instead), it became mandatory that the file be converted to its binary format before downloading with the (Pascal) TENCOM communicating software available.

Using the cross assembler was not a simple task either. First, it does not currently reside on the DEC 10 for public use nor is it supported in any fashion by UTCC personnel. Through the assistance of Kathy Sowder, the program was found to reside on a DEC 10 disk area assigned to Dennis Reed of UT's Engineering Department. Dennis gave permission to copy the program and related files, and this was done.

Second, it was found that the Cross assembler does not expect the same general assembler directives (pseudo operations) or label format as are in the TALKER program. When TALKER was initially assembled with CROSS, this fact was not known and the errors were extensive. It was first thought that the program was expecting Z80 instead of 8080 source code since the switch code expected by the program was the same for 8080 or Z80 code. Later a pattern of errors was recognized and it was discovered that CROSS expected MACRO-11

directives which differed significantly from the ones in TALKER. Additionally, CROSS required colons to follow labels (not done in TALKER) and all hexidecimal literal expressions had to have a "$" preceding them instead of an "H" following the expression as used in TALKER. A program called COLON was written and executed that added the needed colons to all labels in the TALKER Program. All other necessary changes were made to one test section of TALKER and that section was assembled successfully into binary code. See Appendix B for specific changes made and operating instructions for CROSS as well as documentation on COLON.

4. DOWNLOADING TO IMSAI/TENCOM, DUMBT AND TELNET

TENCOM

Once the synthesizer had been tested successfully with the SPEAKIT program, testing began on the TENCOM communication program in preparation for downloading of the TALKER program from the DEC 10. From the start, problems plagued the testing of TENCOM. The problems began with trying to find a copy of the program and its documentation, and included having to recompile the program, having to modify the Pascal operating system, having to work with a bad Pascal Filer routine, and having to work around IMSAI hardware changes and problems. See Appendix C for a full description of the problems encountered.

After all the above problems had been surmounted, the testing with TENCOM was still unsuccessful. The program

would test successfully up to the logon procedure. The program would get no response from the DEC 10. Another small assembler routine called DUMBT, was written to communicate with the DEC 10 in hopes of bypassing the elaborate logon procedure in TENCOM with a call to this routine. This routine did not test successfully either. Data would transfer from the DEC 10 but not to the DEC 10. See Appendix C for an additional description of DUMBT.

After this time-consuming test period, work with TENCOM had to be abandoned. It was discovered through the dissection of the TENCOM program that TENCOM was not capable of transferring a binary file. It could transfer the 8080 source file but the USCD Pascal operating system did not have an 8080 assembler available to assemble the file once it was transferred. See Appendix C for details of this problem.

TELNET

During the course of the preceding events, another communication program called TELNET was discovered. TELNET is written in "C" programming language, and can execute under the CP/M operating system. It is capable of uploading or downloading ASCII files of a limited size. When TELNET was discovered, it was not given serious consideration since it was hoped that the large source file of the TALKER program could be modified and assembled on the more powerful DEC 10 and then downloaded as a binary file to the IMSAI using the TENCOM program. TELNET also was not capable of downloading

the entire TALKER source program at one time. Neither did it do any kind of I/O error checking nor monitor the DEC 10 input for control characters stopping or starting data transfer.

TELNET essentially turns the IMSAI into a "dumb" communicating terminal with a few additional capabilities available by keying in one of 8 special characters. One special character directs output to the computer with which it is communicating (the DEC 10) to originate from an IMSAI disk file instead of the IMSAI console. A combination of 3 other special characters initiate routines which first, turn on text collection allowing the input from the MODEM (DEC 10) to be stored directly into an IMSAI memory buffer area; then second, turn off text collection; and then finally, to dump the memory buffer to disk. Another special character routine will flush the buffer in readiness for additional data if necessary. The size of IMSAI's memory limits the size of the memory buffer. IMSAI II's buffer area is set at 32 K (maximum size of a file that can be transferred at one time).

TELNET Versus TENCOM

TELNET and TENCOM are vastly different types of programs although they accomplish essentially the same task. TENCOM is a very sophisticated program that transmits and monitors all input and output to and from the DEC 10. The user has no access to the DEC 10 monitor. When downloading a file, it stores the DEC file directly onto an IMSAI (Pascal) floppy

disk file without any intermediary steps. TENCOM is completely dependent on a predictable response from the DEC 10 monitor or SOS program. For example, TENCOM depends on character string recognition for the LOG-ON routine, end-of-text signal routine and many other routines. It also expects a particular DEC 10 echo pattern in all transmission. Any changes to the DEC 10 system could cause the program to abort or transmit erroneously. In comparison, TELNET is a very primitive program that primarily polls both the input port from the DEC 10 (LST port) or the console port, waiting for data to be transmitted. The DEC 10 has priority over data coming from the console. The user communicates directly with the DEC 10 with this method. To transfer files, the special character routines just redirect whatever data is being transmitted to a memory buffer or a file. See Appendix K for TELNET operating instructions. TENCOM, when uploading a file to the DEC 10 has I/O error checking routines that retransmit a line if any character in the line was not transmitted correctly. Also messages will print if a bad block was written or other error condition such as "no more space" occurs. TENCOM monitors the DEC 10 for XOFF (control "S"), a stop sending data signal, or XON (control "Q"), a restart signal. TELNET has none of this error checking or monitor ability.

Testing TELNET

When it was discovered that TENCOM could not transfer a binary file and that Pascal could not assemble an 8080

source code file, work with TENCOM and Pascal was abandoned and TELNET was given serious consideration. A closer look at the downloading capability of TENCOM versus TELNET revealed that the TENCOM I/O error checking and error message capability applied to uploading (transmitting) a file from the IMSAI to the DEC 10 and did not apply to downloading (receiving) a file from the DEC 10 to the IMSAI as was needed to transfer the TALKER program. TENCOM monitors the DEC 10 for XON and XOFF characters only in its TRANSMIT (uploading) routine, too; but when a file is being downloaded, it should not be necessary to monitor for these control characters. The intent of these characters is to stop or start data being transmitted from the IMSAI, and in a downloading routine the data is being transmitted from the DEC 10, not the IMSAI. The TENCOM program seems to have distinct advantages over TELNET in uploading a file from the IMSAI to the DEC 10 but both programs appear to be equivalent in their downloading capability. The only exception is that the source file will have to be transmitted in two sections under TELNET due to the limited buffer size. Therefore, not much seems to be lost by abandoning TENCOM for TELNET to download the TALKER file.

TELNET was then tested with the IMSAI II, connecting it first with the DEC 10 and second with the Spinwriter. Testing was unsuccessful with both. Communication again seemed to go in one direction, from the IMSAI II to the other

device, but not vice versa. Since TELNET runs successfully on the APPLE II it seems very likely that there is a hardware problem with IMSAI II. It is possible that there is a wiring problem or something wrong with the SMB (Source Management Board) and that the port logic is not working as it should. If either is the case it would explain why all communication testing in this thesis project was unsuccessful. At this point in the project, the SMB board has not been repaired (nothing has been found wrong as yet) and the wiring has not been tested.

By this time, the IMSAI I had received new disk drives, and a communications problem which prevented it from transferring data out of its List port had been corrected. The IMSAI I had some areas in high memory that were bad, but the SMB board was assumed to be good. TELNET was then tested with this IMSAI but the test was unsuccessful. A problem was found in the communication wiring and work with the IMSAI I had to be abandoned.

Mike Sappington of the UT Micro Lab offered the use of his home IMSAI computer to test TELNET. The test was successful and a small section of the large TALKER program was successfully downloaded and stored on a floppy disk. The transmission rate on his IMSAI was extremely slow (about 110 baud), there were some data transmission errors, and the TELNET buffer overflowed, making it difficult to save the memory buffer on disk. This test showed that TELNET could communicate in both directions and could be used to download

the TALKER file. It also showed the need for downloading small sections of code at a time. The max size that can be transferred at one time seems to be about 30 DEC blocks. With 191 blocks to be transferred, it would take 7 transfer sessions. Each section transferred would then have to be checked against the source listing for transmission errors.

Work on this thesis project was then terminated. At this point, neither of UT's IMSAI's were capable of bi-directional communication nor was there any indication of the IMSAI's being repaired in the near future. The following two sections (Section 5 and Section 6) contain the information needed to proceed with this project once the IMSAI's are repaired and the TALKER program has been successfully transferred to the IMSAI.

5. TALKER PROGRAM: MODIFYING AND TESTING

Modifications for Use with IMSAI and VSK

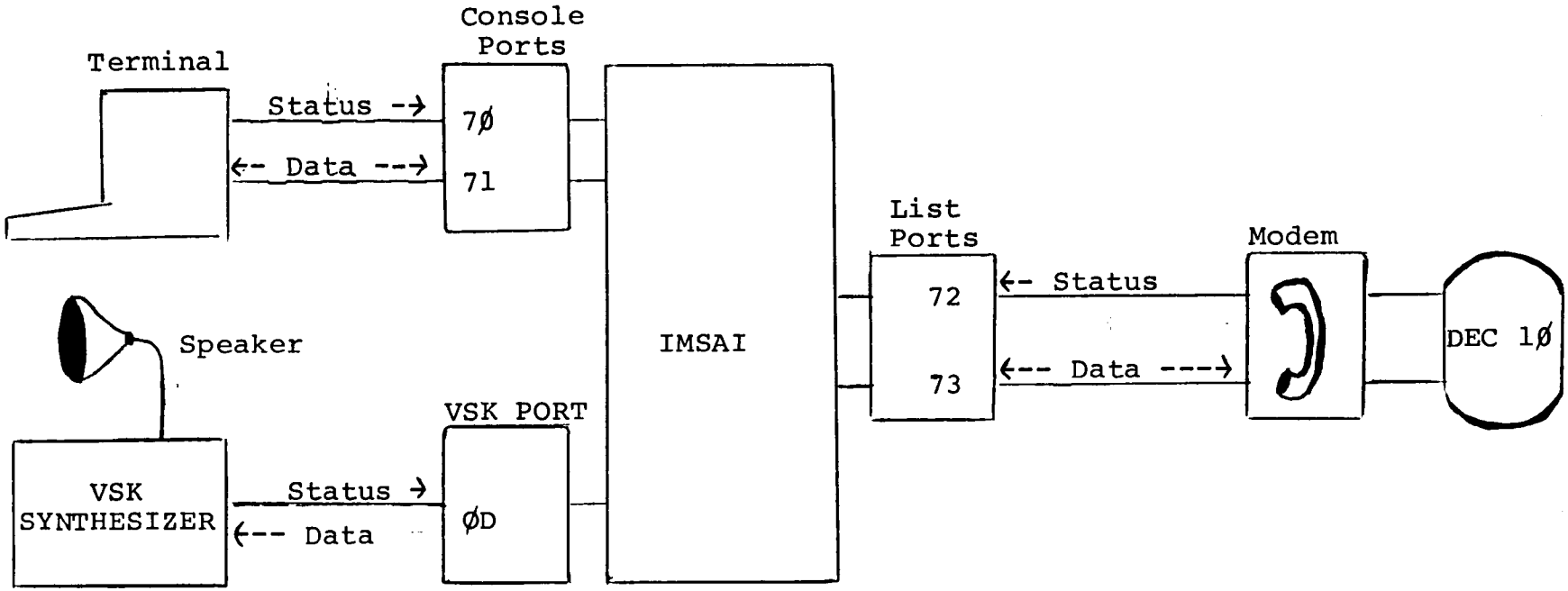
Once the TALKER program has been successfully downloaded onto a disk file on the IMSAI, the next step will be to modify the TALKER program to operate properly on the IMSAI with the VSK synthesizer. The clear screen routine will have to be modified as well as the routine which gets information from the keyboard and sends it to the video screen. The TALKER is set up for use with a VS-6 synthesizer and modifications must be made to the I/O (input and output) routines to achieve the proper handshaking convention. Additionally, the top 2 bits of a phoneme byte that are used for pitch and

inflection on the VS-6, must be masked off for use with the VSK (the VSK does not have the pitch and inflection capability). The input and output ports used in the program must be modified to reflect those used with the IMSAI and VSK synthesizer. See Figure 1 for a port diagram. The keyboard character used to initiate the special speech features in the TALKER program must be modified to be a standard ASCII key.

The specific changes recommended can be found in Appendix G. The paragraph number refers to the corresponding number in an annotated copy of the TALKER source program stored in UT's Mini Lab. Appendix E also includes documentation on some of the internal procedures and labels used in the TALKER program. This documentation may be of assistance in later modifications or improvements to the program. Also see references (8) and (18) for more detailed information concerning the program changes. These changes could be made on the DEC 10 under SOS before the file is downloaded, or could be made under TECO, TED or other CP/M editor on the IMSAI after the file has been transferred.

Assembling with MASM80

If the changes mentioned above have been made, then the program can be assembled by the Macro 80 assembler (under CP/M - MASM80). MASM80 expects the first statement to be "ASEG" and the next to be .8080 or .Z80, to signal 8080 versus Z80 source code. All labels must be followed with a ":" and "*" comment indicators must be changed to "Ø;" (where "Ø" is



Console and List Ports: Bit 0 (LSB) = 1 when there is data waiting to be transmitted.
Bit 1 = 1 when line is not busy (ready to receive data.

VSK Status Port: Bit 7 (MSB) = 1 when line is not busy (ready to receive a phoneme).

Figure 1. Port Diagram

a blank space). All other pseudo commands (assembler directives - EQU, DS, DB, etc.) seem to be identical to those used in the TALKER program, and they should assemble without problem. TALKER is written in six sections which may have to be compiled separately and hand linked. The CP/M commands to assemble, link and execute program using MASM80 are:

```
A MASM80 ,=TALKER.ASM/L/R              (involves assembler)
A LINK80 TALKER.RE./N/E,TALKER.REL     (links)
A TALKER                               (executes TALKER.COM)
```

The assembler creates a list file and a relocatable file. The linker creates a com file (binary, executable file) ready for execution.

Testing

The TALKER program should be tested under CP/M. The program expects to transfer data to and from a mainframe computer, in our case the DEC 10. Anything entered through the keyboard will be sent out the serial port of the IMSAI to the computer, and anything that comes from the computer will be spoken. To communicate with the DEC 10, the serial ports in the TALKER programs will be modified to be the list ports of the IMSAI as indicated in Appendix E. Data will be transferred from the IMSAI List port through a telephone modem to the DEC 10. The DEC 10 communicates in a full duplex mode. Every character entered through the keyboard will be echoed back to the IMSAI from the DEC 10. In this way, data entered through the keyboard as well as responses from the DEC 10 will be spoken.

See "Section K. VSK - DEC 10 Communication" for additional communication information.

To test the program without having to communicate with the DEC 10, the IMSAI could be plugged into another CRT in the Mini Lab through the patch panel. The data typed into this CRT would appear to be coming from another computer and would be vocalized.

See Appendix F. "Talker User's Guide," for operating instructions for this TALKER program.

6. VSK - DEC 10 COMMUNICATION

The TALKER program operates with the DEC 10 as a "dumb" terminal. It constantly polls the input line from the DEC 10 and stores the data into a buffer. Some problems may arise from this polling technique. First, the polling is software controlled, and some characters being sent from the DEC 10 may be lost while the TALKER is in one of its translate routines. There does not appear to be any mechanism in the TALKER program to temporarily stop and start DEC 10 transmissions (as with an XOFF and XON). Neither does there appear to be any checking for XON and XOFF characters being sent from the DEC 10 to slow down transmission from the IMSAI. Since the transmission rate is only 300 baud, perhaps slowing down DEC 10 transmission may not be necessary. Checking for XON and XOFF may not be necessary either since data input will not be echoed back to the screen and thus not spoken if an XOFF has been issued by the DEC 10. A blind user would

not have the immediate feedback of the absence of a screen echo, but the absence of vocal transmission of the line of data would indicate the problem. See (12) and (15) for additional notes on DEC 10 communication that might aid in implementing the TALKER program if communication problems arise.

III. DISCUSSION

1. UNIVERSITY SURVEY

A survey was made of selected colleges and universities who have Computer Science departments to see what technology was being used on other campuses to aid the blind or speech impaired Computer Science (or related) students and faculty. Of the 10 universities surveyed, only 4 had operational equipment. Three had voice synthesizers attached to micro processor computers while the fourth had a Braille printer. The 2 blind students at a fifth university had their own opticons (a hand held camera with a tactile image). The other 5 universities indicated that they had neither any blind Computer Science students or faculty members, nor did they have any equipment available to aid them should they enroll (or be hired). See Table 1 for the universities surveyed and a summary of the information obtained. See Appendix G for a copy of the questionnaire and Appendix H for contact names and addresses at these universities.

This survey was in no way meant to be an exhaustive survey or even a random sample survey. The intent was purely to get a general feeling of the technology on other campuses.

Computer Science (CS) can be an attractive field of study for the blind, but unless aids are made available, the students will be completely dependent on readers to accompany them constantly during the time consuming lab sessions required to complete the vast amount of programming necessary

TABLE 1. Survey of Computer Equipment Available for Blind or Speech Impaired Users at Selected Universities.

University name	No. of Blind Students/Faculty	Equipment Available
University of Texas (Austin)	3	Braile Printer
University of Minnesota	2	Students have own opticons
Texas A & M	0	Plans for Apple synthesizer to AMDAHL system.
University of Colorado	0	None
University of Georgia	0	None
Pennsylvania State University	0	Obsolete Braile printer
Georgia Institute of Technology	0	None
University of Illinois (Urbana)	several	SOL computers with VS-6 synthesizers
University of Nebraska	1+	Computalker CT-1
Vanderbilt University	many	TRS80 with Radio Shack synthesizer, Paperless Braille Recorder (Versabraile)

for a CS degree. This may explain why the 5 universities that have no equipment for the blind have not attracted any blind students or faculty.

2. DEVICES FOR THE BLIND

The primary devices currently on the commercial market to aid blind computer users are talking terminals (voice synthesizers), opticons, Braille printers, and paperless Braille recorders. The last one, paperless Braille recorders, is one of the newest and has been found very useful by blind users who are good Braille readers. It is essentially a 20 character display board with pins to present the tactile image of the Braille character. Braille characters can be input to the computer by pushing these pins and the Braille data output from the computer is displayed with these pins as well. Telesensory Inc., sells their VERSABRAILLE recorder for $6,800. It has an RS232 board which gives it the capability to interface with almost any computer and has a 1,000 character buffer which can be scanned.

3. VOICE OUTPUT QUALITY

The voice output quality of the voice synthesizer and related software will determine whether the voice output system is useable in a practical sense for blind users. Any voice system which uses an English text translator program will require several hours of listening to acquire complete comprehension. The quality of the VSK synthesizer used for

this project and that of the Radio Shack is such that it is questicnable whether it is of high enough quality to be acceptable for continuous use by blind users. Susan Williams of Vanderbilt University (Peabody College) has been working with blind grade school and high school students at the Tennessee School of the Blind (see Appendix H). Her office has a Radio Shack TRS 80 and voice synthesizer that can communicate with other computers. The system has not been well accepted by the students since the quality of the feedback is not accurate enough. The feedback from the VSK used in this project does not appear to be accurate enough for blind users either, and it is unlikely that listening time will drastically affect its usefulness.

VOTRAX's VS-6 synthesizer is of a much better quality than that of the VSK or Radio Shack synthesizer. It is currently being used successfully with SOL computers in the offices of blind professors at the University of Illinois. Also, a VS-6 is being used successfully with an INTEL 8080 computer by a blind programmer working for NIH (National Institute of Health) in Bethesda, Maryland (14). These two examples indicate that voice synthesizer output systems can be useful and practical if the quality of the synthesizer is high.

4. VS-6 AT UT

The UT Mini Lab has just recently received (April, 1981) on loan from Goddard Space Center a VS-6 VOTRAX synthesizer. If this synthesizer could be installed on the IMSAI and used

with the TALKER program described in this thesis, the result would be a much higher quality speech output system than that originally proposed in this thesis.

The VS-6 received uses 2 parallel ports instead of the single serial port used in the VSK and requires an interface board to be used with the IMSAI. If purchased through a kit, the interface board would cost between $150 and $200. An assembled board from VOTRAX (RS232) would cost an estimated $4,400. It is also possible that an interface could be borrowed. Contact Kathy Sowder (UT Mini Lab) for more information.

The TALKER program was originally written for a VS-6, so no modifications to the input/output routines for the synthesizer would be necessary. The other changes would be required as noted in Appendix E, as well as changes in the synthesizer port addresses as required by the VS-6.

5. TALKER IMPROVEMENTS

To further improve the quality of the voice output system implemented with the VS-6 or VS-K synthesizer, the TALKER program could be modified for the specific system with which it was to be used. A table could be entered as a front end to the TALKER program which would contain all the commonly used vocabulary and the phoneme based voice output desired for that vocabulary. If no match was found in the dictionary, the program would call the TALKER program. For example, when communicating with the DEC 10, the table could contain the

DEC 10 prompts and the desired speech for each prompt. A disk could be created for every system that would contain a tailor-made table for that system and the TALKER program.

IV. CONCLUSIONS

Once UT's IMSAI #2 computer is repaired such that bi-directional communication is again possible, there is no reason why the TALKER program with the changes indicated in this thesis could not be installed and made available for blind student and faculty use. The quality of the voice system using the VSK synthesizer may unfortunately make this use very limited.

The VSK is on loan to UT's Computer Science Department indefinitely or until we can get a better synthesizer installed (per Peter Maggs, 4/29/81). It is not known how long the VS-6 is on loan to our department. It is therefore recommended that work continue with installing the TALKER program with the VSK synthesizer in hopes that an interface will become available so that the VS-6 synthesizer could replace the VSK. Once the VS-6 is installed, the quality should greatly improve and the system would be much more useful to UT's blind students and faculty members. The VS-6 eventually would have to be obtained permanently by UT. By demonstrating the speaking capabilities of the system and determining the usefulness of it as an aid for the blind, perhaps funds would come available for its purchase, or perhaps the VS-6 could be purchased at a reduced price from Goddard.

The VS-6 with an English text translator program has been proved successful for use by the blind at NIH and the University of Chicago. Hopefully, the University of Tennessee will soon be able to proclaim success as well.

LIST OF REFERENCES

LIST OF REFERENCES

1. Anderson, J., "DATA-BOY Speech Processor", Dr. Dobb's Journal of Computer Calesthenics and Orthodontia, July-August 1979, pp. 30-32.

2. Computalker Consultants, Inc., "Special Demo Cassette", a cassette recording dated 7/15/78.

3. Dahmke, M., "A Voice for Bill", Computing Inc., Winter 1979, pp. 48-54.

4. Elowitz, H. S., "Letter-to-Sound Rules for Automatic Translation of English Test to Phonetics", IEEE Transactions on Acoustics, Speech, and Signal Processing, December 1976, pp. 446-458.

5. Gargaguano, T. A., "Text Translation Builds Vocabulary for Speech Chip", Electronics, February 10, 1981, pp. 118-121.

6. Kurzweil, R., "Kurzweil Talking Terminal Announced; First Unlimited Vocabulary, Text-to-Speech Device", The Kurzweil Report, Spring 1979.

7. Lin, Kun-Shan, "Software Rules Give Personal Computer Real Word Power", Electronics, February 10, 1981, pp. 117-125.

8. Maggs, P., a letter to J. Michael Moshell at the University of Tennessee, Knoxville, from the author at the University of Illinois at Urbana-Champaign, dated September 29, 1980, and now in the possession of Jane Kieninger, Knoxville, Tennessee.

9. Maggs, P., "Talking Terminal -- Preliminary Release", documentation of a talking terminal program, provided by the author at the University of Illinois at Urbana-Champaign, received April, 1980.

10. Maggs, P., "Talking Terminal Software for Use in the Radio Shack TRS-80 Computer", documentation of talking terminal program, provided by the author at the University of Illinois at Urbana-Champaign, 1980 copyright.

11. Mountain Hardware, Inc., "SUPERTALKER", a brochure from Mountain Hardware, Inc., 3000 Harvey West Blvd, Santa Cruz, CA, 95060, obtained Spring 1980.

12. Nayak, H. R., Telecommunication Processing Link between DEC System 10 and CP/M Based Operating System, University of Tennessee, Knoxville, Tenn., 1979.

13. Rice, D. L., "Friends, Humans and Country Robots, Lend Me Your Ears", reprint from BYTE, August 1976.

14. Songco, D., "How Computers Talk to the Blind", IEEE Spectrum, May 1980, PP. 34-38.

15. Sundarem, M., UNIX to DEC-10: A Simple Teleprocessing Link, University of Tennessee, Knoxville, Tenn., 1979.

16. Tega, E. R., "Voice Input and Output", EDN, November 20, 1979, pp. 159-167.

17. "TI's Module Speaks", Electronic Engineering Times, May 12, 1979, p 1.

18. VOTREX Division of Federal Screw Works, "Model S100-VSK Synthesizer Interface Specifications", Manual Part Number M100 from the author at 500 Stephenson Highway, Troy, Michigan, July 1, 1978.

19. VOTREX Division of Federal Screw Works, "VSK4 Synthesizer - Voice Interface Application Samples", a cassette recording dated June 1977.

APPENDIXES

APPENDIX A. VENDOR ADDRESSES AND TELEPHONE NUMBERS

A. I. Cybernetic Systems
P. O. Box 4691
University Park, NM 88003

COMPUTALKER Consultants
821 Pacific Street
Santa Monica, CA 90404
(213) 828-6546

Maryland Computer Services
502 Rock Spring
Belair, MD 21014
(301) 879-3366

Master Specialties
1640 Monrovia
Costa Mesa, CA 92627
(714) 642-2427

MIMIC Electronics
P. O. Box 921
Action, MA 01720

Texas Instruments, Inc.
8600 Commerce Park Drive
Suite 100, MS 653
Houston, TX 77036
(713) 778-6511

VOTRAX
Division of Federal Screw Works
500 Stephenson Highway
Troy, MI 48084
(313) 588-2050

APPENDIX B. CROSS/COLON PROGRAMS

TALKER Modifications for CROSS

The following changes had to be made to the TALKER program to get it to assemble using the CROSS cross-assembler on the DEC 10:

1. All comment lines that began with an "*" in column one had to be changed to "␣;". Where "␣" is a black space.

2. All references to hexidecimal literal expressions had to be modified. Instead of following the expression with an "H", the expression had to be preceeded with a "$". For example, "MVI A,59H" had to be changed to "MVI A,$59".

3. All assembler storage directives had to be changed. "DS" was changed to ".BLKB", and "DB" was changed to ".BYTE".

4. All equate statements had to be changed. "EQU" had to be replaced with "=".

5. All references to "ORG" had to be changed to ".=".

6. References to "LISTING" had to be changed to ".LST".

7. All labels had to be immediately followed with a colon.

The errors most often encountered during assembly were:

D - Doubly defined symbol reference

Q - Questionable syntax, Arguments are missing or too many

U - Undefined symbol

A - Addressing error

These error messages can be found in a Macro 11 assembler manual.

Cross Execution

To execute this program, a copy of the program must be copied over into the user's area. A copy can be obtained from Doug Birdwell (EE) or perhaps from Kathy Sowder (UT Mini Lab). To copy, log on to the DEC 10 and issue the following commands:

```
COPY CROSS.EXEC=DSKB:CROSS.EXEC[ # , # ]
COPY CROSS.HLP=DSKB:CROSS.HLP[ # , # ]
COPY CROSS.TXT=DSKB:CROSS.TXT[ # , # ]
```

Where DSKB should be changed to the disk where the program resides and [# , #] should be changed to reflect the programmer ID and project ID of the person from whom you are copying the program. The first file is the executable file, the second and third are documentation on the program.

To execute CROSS, type:

```
.R DSKC:CROSS.EXEC
*PROG1.BIN,PROG1.LST=PROG1.SRC/M80
```

Where DSKC is where your copy of the program resides. This will read PROG1.SRC (arbitrary name) source program and if the assembler encounters no errors, it will create a PROG1.BIN binary file and a PROG1.LST file with the assembled listing. The "/M80" is a switch specifying 8080 or Z80 source code.

COLON Program

The program "COLON" was written in SNOBOL by Charles Boyd of UTCC to modify all labels in the TALKER program to

have colons as required by CROSS.

To run, enter the following commands to the DEC 10 monitor:

```
.R SITBOL
=COLON.STB
```

See the following page for a listing of this program. This program must be stored as COLON.STB in the user's DEC 10 disk area for successful execution.

COLON Program

```
      OUTPUT(.TTOUT,'TTY:','T')
      IDENTIF = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
+               '0123456789'

ST    TTOUT = 'Enter input file name; '
            X = TTY
TTY = /FILE(X) "INPUT FILE COULDN'T BE FOUND"    :S(ST)

      INPUT(.IN,X)
      TTOUT = 'Makeup an output file name: '

NX    L = IN   :F(END)
      L LEN(1) . C
      IDENTIF C          :F(PRT)
      L 'EQU'            :s(PRT)
      L SPAN(IDENTIF) . X REM . Y = X ':' Y
PRT   OUT = L   :(NX)
END
```

APPENDIX C. PROBLEMS WITH TENCOM, DUMBT AND PASCAL

The initial problem encountered was to find a copy of the program and the thesis documentation on the program. Of the three committee members associated with the thesis, only Dr. David Straight was able to provide a copy, and he kindly loaned it to this project. No one knew where a copy of the program was stored. A letter was then sent to Chuck MacDowell, the author of the thesis, describing the problem. He promptly telephoned to say that 2 copies of the program were stored on a shelf in the office of the Mini Lab where the PDP-11 resides. These were easily found and testing was begun.

TENCOM documentation stated that successful execution of the program was dependent on the monitor in use by the DEC 10. The TENCOM logon procedure expected the string "7.01" to be transmitted from the DEC 10 7.01 monitor and any change in monitors would require a change in the TENCOM program. Even though the thesis project (TENCOM) had been just completed in the summer of 1980, by Fall, 1980, the monitor had changed. The procedure LINEINIT in the TENCOM1 data set was then altered to look for the constant characters "UIVE" (taken from "THE UNIVERSITY OF TENNESSEE") instead of "7.01", and the entire program was recompiled. The modified program was then tested with no success.

A second correspondence was made with Chuck MacDowell for possible suggestions on potential problem areas. He responded with a letter that included several possibilities.

The first possibility was that there might be a logon queue when attempting to logon. If this were the case, TENCOM would not work. The second and third possibilities were related to the logon procedure discussed above and already corrected. The fourth stated that the BIOS (Basic I/O System) used by Pascal on the IMSAI had been modified so that the remote input and output (REMIN and REMOUT) pointed to the list port instead of the console controller, and that perhaps the operating system being currently used for testing TENCOM did not have these changes. He was right about the fourth possibility. At this point, REMIN and REMOUT were altered to point to the List port and TENCOM was retested. The retest was still unsuccessful. The program would begin successfully, but fail when the logon procedure was encountered. It appeared to get no response from the DEC 10.

DUMBT

Dissection was begun on the TENCOM program to find the precise location of the problem. It was first discovered that not only was the logon procedure dependent on recognizing specific characters in the DEC 10 logon but also on receiving a certain number of lines of data and a specific communication interchange. So at this time a small program called DUMBT was written in assembler to try and make the IMSAI act like a dumb terminal and establish some kind of communication with the DEC 10. It was hoped that this little program could replace the elaborate logon procedure in TENCOM by allowing

the terminal operator to handle the logon and then transfer control over to TENCOM (by a special control character for example) once the logon was successful. The program DUMBT was tested with the DEC 10 with no success. It was then tested by interfacing the IMSAI with another CRT, a Hazeltine. The program would transfer data to the Hazeltine but would not receive data from the Hazeltine.

As dissection continued with the TENCOM program, the RECEIVE procedure was reviewed. At this point it was discovered that the TENCOM program would not transfer binary files. TENCOM was found to use DEC 10's SOS software to transfer the files and SOS expects an ASCII file, treats it as one, and modifies the top bit. This meant that TENCOM could not be used to transfer the TALKER program. TENCOM was not capable of transferring a binary file and although it could transfer the source file, a PASCAL 8080 assembler was not available to assemble the file once it was transferred.

Pascal Problems

During the period of testing TENCOM, problems with using the USCD Pascal operating system arose. The Pascal "Filer" mode did not operate as documented, making it very difficult to transfer or manipulate files. Additionally, the console I/O routine had a problem such that not all characters input through the keyboard were transmitted and received by the Pascal editor. Finally, whenever the CRT terminal attached to the IMSAI was exchanged or replaced by another brand of

CRT, the Pascal editor would no longer work. Pascal would then have to be regened, and screen handling information would have to be entered that corresponded directly with the hardware (screen and keyboard configuration) of that brand of CRT. See Appendix G for the specific problems. During the time period of this thesis project, CRT terminals were exchanged twice. The transfers occurred once early in the project when permanent disk problems on the IMSAI #1 caused the need to transfer the project to the IMSAI II and its CRT (different than the IMSAI #1's), and once very late in the project when the HEATH CRT on the IMSAI II was replaced with a Hazeltine model. When Pascal was regenned to operate on the Hazeltine model, the "Filer" mode of Pascal began to operate as it was documented. The assumption can be made that although the HEATH version of Pascal works, that the "Filer" problems occurred because the system was not regenned properly (or perhaps not completely) for use with the HEATH terminal. The I/O problems discussed earlier in this paper were not solved by the new Hazeltine version of Pascal.

APPENDIX D. TELNET - CHECKLIST FOR EXECUTION

This checklist should be followed to copy over a DEC file from the DEC 10 to an IMSAI disk file using TELNET. TELNET executes under CP/M and can only transfer about 30 DEC blocks of data safely at one time.

1. Set IMSAI LST baud rate to 300 baud.

2. Plug one of the null modem plugs of the Mini Lab Plug Panel into the IMSAI connector. Plug the other into the Modem connector.

3. Boot CP/M with a CP/M disk containing TELNET.COM.

4. Type "TELNET FILENAME". This will begin execution of TELNET. There is no response from the program to indicate that TELNET is executing.

5. Dial up the DEC 10.

6. Hit carriage return (CR) on the IMSAI keyboard and logon as if you were using a "dumb" terminal.

7. Type:

.TYPE DECFILE (no carriage return)

Do not hit the carriage return yet. DECFILE is the name of the file to be transferred from the DEC 10.

8. Type:

(Control) A (no carriage return)

a

The control A tells TELNET that you wish to send a special character. It will prompt for it, and you respond with the "a" above. This "a" tells TELNET to turn on text collection in an IMSAI memory buffer.

9. Now type a carriage return. This will start the file transfer.

10. When the file has been copied into the buffer, type:

(Control) A

b

This "b" turns off text collection.

11. Type:

(Control) A

e

This echos the file to the screen so that the user can check for transmission errors.

12. Type:

(Control) A

d

This writes the memory buffer to the IMSAI disk file. When the write is completed the program can be terminated.

APPENDIX E. TALKER - CHANGES AND INTERNAL PROCEDURES

Changes

The paragraph numbers correspond to an annotated copy of the TALKER program.

1) Delete:

```
FILLER  BSSZ    62H
        NOP
```

2) VSK Ports: After this line:

```
GETSAV  EQU     +3008H
```

Insert:

```
STATPT  EQU     0DFH            ; VSK STATUS PORT
DTAPT   EQU     0DFH            ; VSK DATA PORT
```

3) VSK I/O Routine: Delete "OUTTOV" and "OUTGO" paragraphs, and replace with:

```
OUTTOV: IN      STAPT           ; CHECK SPEAKER STATUS
        CALL    GETIN           ; CHECK FOR AND GET ANY INPUT
                                ;    FROM SERIAL PORT.
        ANI     80H             ; MAKE SURE SPEAKER NOT BUSY
        JZ      OUTTOV          ;
        LHLD    SL              ; SET STRING LENGTH
        XCHG                    ;    INTO DE
        LXI     H,STRING        ; START OF STRING
        INX     D               ; STARTS AT 0 NOT 1
OUTGO:  MVI     B,0             ; B WILL COUNT PHONEMES SAID
        MVI     A,03H           ; SHORT PAUSE TO SET UP
        CMA                     ;
        OUT     DTAPT           ; SEND TO VSK
        CALL WATILBZ            ; WAIT TIL VSK GETS PAUSE
```

4) VSK I/O Routine: Delete "OUTNOW" paragraph, and replace with:

```
OUTNOW: CALL    BUSYLP          ; WAIT TIL VSK NOT BUSY
        MOV     A,M             ; GET PHONEME AGAIN
        ANI     3FH             ; MASK OFF TOP 2 BITS
        CMA                     ; COMPLEMENT ACCUM.
        OUT     DTAPT           ; OUT TO VSK
```

```
        CALL    WATILBZ         ; LOOP UNTIL VSK GOES BUSY
        INR     B               ; ONE MORE PHONEME SAID
```

5) VSK I/O Routine: Delete "NLOOP" and "WAIT" paragraphs, and replace with:

```
NLOOP:  CALL    BUSYLP          ; WAIT TIL VSK NOT BUSY
        MVI     A,0FFH          ; LULL
        ANI     3FH             ; MASK TOP 2 BITS
        CMA                     ; COMPLEMENT
        OUT     DTAPT           ; SEND OUT LULL TO VSK.
        CALL    WATILBZ         ; LOOP UNTIL VSK GOES BUSY
        DCR     C
        CALL    GETIN
        JNZ     NLOOP
        MOV     A,D
        ORA     E
        RZ                      ; IF LAST ONE, WAIT LATER.
WAIT:   IN      STAPT           ; IN STATUS INFORMATION
        CALL    GETIN           ; KEEP CHECKING FOR SERIAL
                                ;    INPUT
        ANI     80H             ; LOOP UNTIL VSK NOT BUSY
        JZ      WAIT            ;
        RET
BUSYLP: IN      STAPT           ; LOOP UNTIL VSK NOT BUSY
        ANI     80H
        JZ      BUZYLP
        RET
WATILBZ:IN      STAPT           ; LOOP UNTIL VSK GOES BUSY
        ANI     80H
        JNZ     WATILBZ
        RET
```

6) Delète:

```
FILLER  BSSZ    70H             ; FILLER
        NOP
```

7) VSK, Console, and Serial ports: Delete all "EQU" statements after:

```
*  SOL SYSTEM VARIABLES
```

and before:

```
TERM    DI                      ; PROGRAM USES NO INTERUPTS
```

Insert:

```
STAPT   EQU     0DFH            ; STATUS PORT FOR VSK
DTAPT   EQU     0DFH            ; DATA PORT FOR VSK
```

```
DSTAT   EQU     70H             ; CONSOLE (KEYBOARD) STATUS PT.
CDATA   EQU     71H             ; CONSOLE (SCREEN) DATA PORT
SSTAT   EQU     72H             ; SERIAL (LST) STATUS PORT
SDATA   EQU     73H             ; SERIAL (LST) DATA PORT
```

8) Clear Screen Routine: Delete "TERMGO" and "CLERA" paragraphs. Delete ERAS1 up to but not including the following line:

```
CALL    SIGNON                  ; OPTIONAL SIGNON
```

Insert:

```
TERMGO: IN      DSTAT           ; CHECK KEYBOARD STATUS
        ANI     02H             ; LOOP UNTIL NOT BUSY
        JZ      TERMGO          ;
        MVI     A,1CH           ; SEND CLEAR SCREEN
        OUT     CDATA           ;    CHARACTER
```

9) Special Control Characters: Replace:

```
KEYWORK:CPI     81H             ; LOCAL CONTROL
```

With:

```
KEYWORK:CPI     02H             ; CONTROL B - BEGIN LOCAL
```

10) Special Control Characters: Replace:

```
        CPI     93H             ; RIGHT ERROR ENDS LOCAL
```

With:

```
        CPI     01H             ; CONTROL A ENDS LOCAL
```

11) Keyboard and Screen I/O, VSK quiet: Delete "ASKEY" and "QUIET" paragraphs. Insert:

```
ASKEY:  IN      DSTAT           ; KEYBOARD STATUS
        ANI     01H             ; CHECK STATUS FOR SOMETHING
                                ;     TO BE SENT
        RZ                      ; RETURN IF NOTHING (NO KEY)
        IN      CDATA           ; ELSE, GET THE KEY
        RET
QUIET:  RET                     : ROUTINE NOT APPLICABLE TO VSK
;*
;*      OUTPUT CHARACTER TO SCREEN
;*      EXPECTS CHARACTER TO BE IN REG B
;*
```

```
SOUT:   IN      DSTAT       ; IS CONSOLE NOT BUSY
        ANI     02H         ; LOOP, IF BUSY
        JZ      SOUT
        MOV     A,B         ; GET CHARACTER TO BE SENT
        OUT     CDATA       ; SEND IT TO THE SCREEN
        RET
```

12) Serial Ports: Replace "GETA", "GOTAC" and "AKLOP" paragraphs with:

```
GETA:   IN      SSTAT       ; SERIAL STATUS PORT
        ANI     01H         ; DATA READY BIT
        JNZ     GOTAC       ; GOT A CHARACTER
        STC                 ; REPORT A FAILURE
        RET                 ; FAILED
GOTAC   IN      SDATA       ; SERIAL DATA
        ANI     7FH         ; GET RID OF TOP BIT
        MOV     B,A         ; INTO B
        ANA     A           ; CLEAR CARRY
        RET                 ;
     (insert AKTOP)
AKLOP   IN      SSTAT       ; SERIAL STATUS PORT
        ANI     02H         ; BUFFER EMPTY BIT
        JZ      AKLOP       ; WAIT UNTIL READY
        POP     PSW         ; GET IT BACK
        OUT     SDATA       ; SEND CHARACTER
        RET                 ; DONE
```

13) Delete

```
FILLER  BSSZ    363H        ; FILLER
        NOP
```

14) Delete all statements from and including:

```
*  COMMON VARIABLE AREA USED BY ALL PROGRAMS
```

to and including:

```
BUFF    DS      3000        ; PASS 1 IPA, PASS 2 IPA
```

15) Hand Linkage Section: Hand link to AORULE". Contains pointer to letter rules. Paragraph "RULES".

16) Delete:

```
FILLER  BSSZ    2C6H        ; FILLER
        NOP
```

17) Delete:

```
FILLER  BSSZ    1FAH
        NOP
```

18) Handlinkage: Paragraph "RULEAL" - hand linking to A1RULE.

19) Delete:

```
A1RULE  EQU     RULEAL
N1RULE  EQU     RULENM
P1RULE  EQU     RULEC
FILLER  BSSZ    15H
*                               ; FILL UNUSED AREA UP TO NEXT
NOP                             ;      1K BOUNDARY WITH ZEROS
```

Internal Procedures

GETSAV - Gets a character from the serial port, sends the character to the screen, stores character in circular buffer, and sets cc (condition code) to Ø if end of buffer.

GETIN - Calls GETSAV.

ASKEY - Gets a character from keyboard if anything waiting to be transmitted.

KEYIN - Calls ASKEY.

OUTBUF - Checks to see if buffer empty (returns a Ø), gets the next character from the buffer and stores in B.

GOTONE - Moves character in buffer to string - increments string length.

EREG - Tries to match current string character pointed to by SW with current rule character pointed to by RW. If match found RT set = 1, else - 0 and ST set = 1. If match found cc set to 0.

MATCH - Finds bracket rule, matches string against rule until no match. Checks to see if at end of rule (]). If not, tries a new rule.

OUTPUT - Puts characters in buffer addressed by BP.

POVC - Gets second half of rule and calls OUTPUT.

INBUFF - Stores character in B into a circular buffer. Calls HLEQDE.

HLEQDE - Checks for end of circular buffer - if end, returns a Ø in condition code.

APPENDIX F. TALKER USER'S GUIDE

This guide is not applicable unless the TALKER program has been modified as detailed in this thesis, successfully tested, and stored on a bootable CP/M disk as TALKER.COM.

Setup for IMSAI II Communication with the DEC 10

Turn on the IMSAI II, the disk drive unit, and the CRT (3 separate switches). Turn the LST baud rate switch on the back of the IMSAI II to 300 baud. Go to the Mini Lab communications plug board and plug in one of the null modem connectors into the slot marked "Modem" and the other into the slot marked IMSAI #2. Return to the IMSAI and insert the TALKER disk into disk drive A (left drive) and close the driver door. With the switches located on the IMSAI, toggle "RESET" and then "RUN". This should boot CP/M. Type "1" to the question "How many drives?". To the response "A " type "TALKER":

```
A > TALKER
```

The program should respond:

```
READY
```

At this point you are ready to dial up the DEC 10 computer. Dial up the DEC 10 and then return to the IMSAI. Anything that you now enter through the keyboard will be transmitted to the DEC 10, and anything that goes to or comes out from the DEC 10 will be spoken.

Speech Control Features

The TALKER program has 12 speech control features which are accessed by entering a specific control key on the keyboard. Entering the control key will set the program in "Local" mode ready to accept a character to indicate what control feature is desired. Another control key must be entered to get out of "Local" mode and return to "Terminal" mode. The characters and the feature they control are:

B - repeat contents of buffer

S - stop continuous speech

G - to go ahead

L - toggles letter by letter

P - toggles pronounced punctuation

V - toggles line by line

E - toggles key echoing

J - jumps back

A - goes ahead

U - upper case

C - reinstate continuous speech

F - (finished) will clear any text stored in the terminal.

Pressing any character a second time will return you to the normal mode of the feature the key controls (10).

The control key used to set the program in Local mode is documented as being "SHIFT-downarrow-B" and to return to Terminal mode as being "SHIFT-downarrow-A" (9). These control keys are related to Radio Shack's TRS 80 terminals and correspond to a CONTROL-B (capital letter B, hex 02), and a CONTROL-A

(capital letter A, hex 01) respectively on the IMSAI (Hazeltine CRT). The special feature character "F" will also cause the program to return to Terminal mode. If "F" is used, any text stored in the terminal buffer will be cleared. If CONTROL-A is used, the text will not be cleared.

APPENDIX G. BLIND STUDENT QUESTIONNAIRE

BLIND STUDENT QUESTIONNAIRE

Name of your University: _____

How many blind Computer Science (or related) students do you have? _____

What facilities to you have to aid them (include brand name):

- Talking Terminal? _____
- Opticon?* _____
- Braille Printer? _____
- Micro or mini with Voice Synthesizer?

  _____
- Other? _____

If you have any automated system for the blind students use, what computers does it allow contact to?

_____

Who could I contact for additional information (please include phone number)? _____

THANK YOU !!

* A hand held camera with a tactile image.

(Backside of Questionnaire)

Kieninger
3909 Ormand Lane
Knoxville, Tenn. 37923

MS. JANE KIENINGER

c/o MRS. ETHEL WITTENBERG

THE UNIVERSITY OF TENNESSEE

DEPARTMENT OF COMPUTER SCIENCE

8 AYRES HALL

KNOXVILLE, TENNESSEE 37916

APPENDIX H. UNIVERSITY CONTACTS

University of Texas at Austin

Nell Dale, (512)471-4353
Austin, Texas 78712

University Of Minnesota

T. D. Hodge (Asst. Director of Computer Center)
Minneapolis, Minnesota 55455

Texas A & M University

Dan Drew, (713)845-5531
(Division Head, Computing Science Department)
College Station, Texas 77843

University of Colorado

Paul Zeiger, (303)492-7514
Boulder, Colorado 80302

University of Georgia

Dr. James Carmon, (404)542-2106
(Director of Computer Center)
Mr. David Fletcher, (404)542-3564
(Handicapped Student Services)
Athens, Georgia 30602

Pennsylvania State University

Dr. Charles Wissinger, Jr., (814)863-0422
(Computer Center)
University Park, Pennsylvania 16802

Georgia Institute of Technology

Walter L. Bloom, M.D., (704)894-5055
(Executive Asst. to the President, Associate Vice President for Academic Affairs)
Atlanta, Georgia 30332

Vanderbilt University (Peabody College)

Michael Young or Susan Williams, (615)327-8039
Box 328
Peabody College
Nashville, Tennessee 37203

University of Nebraska

Mark Dahmke
MCD Consulting
1515 Superior, Apartment 15
Lincoln, Nebraska

University of Illinois

Professor Peter Maggs, (217)333-6711
College of Law
Champaign, Illinois

APPENDIX I. MISCELLANEOUS USEFUL INFORMATION

Phoneme Coding for VOTRAX Synthesizers

Read (10) and the section on VOTRAX synthesizers in (10) and (12). Note that the larger the number in a phoneme code, the shorter the duration of the sound. To understand the phoneme notations in the TALKER program read (3).

Blind Faculty and Students at UT

Robert Shelton is a blind mathematics professor. He should be contacted if further work is done with the voice synthesizer. He is interested in using the system. His extension is 6733 (1981).

Mike Esling and Kenneth Flippen are two blind students who at one time were Computer Science majors. Call Martha Bryan, Director of Handicapped Students, for their telephone numbers and any other names.

VITA

Jane Kieninger was born in Bethesda, Maryland in 1949, and spent her childhood in Alexandria, Virginia. She received her Bachelor of Science degree with a major in Mathematics in June, 1971, from Mary Washington College of the University of Virginia.

She was employed by the Environmental Protection Agency as a computer specialist from 1971 to 1975, and was part of the design, programming and implementation of a nationwide grants information system. From 1975 to 1977 she was employed by the Medical University of South Carolina as a systems analyst in charge of the design and programming of an interactive personnel and payroll data base system.

In 1977 she enrolled in graduate school at The University of Tennessee in Knoxville, and received the Master of Science degree there with a major in Computer Science in June, 1981. In 1979, she began part-time computer consulting in Knoxville.

She is married and has two young sons.