

To the Graduate Council:

I am submitting herewith a thesis written by Yudhana Aimthikul entitled "Computer-Generated Speech." 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 Electrical Engineering.

Donald W. Bouldin

Donald W. Bouldin
Major Professor

We have read this thesis and
recommend its acceptance:

Robert W. Rochelle

Robert E. Bodensheimer

Accepted for the Council:

Levans G. G. G.

Vice Chancellor
Graduate Studies and Research

COMPUTER-GENERATED SPEECH

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Yudhana Aimthikul

December 1981

3055398

ACKNOWLEDGMENTS

The author is very grateful for the invaluable guidance, patience and helpful advice of his committee, Drs. D. W. Bouldin, R. E. Bodenheimer and R. W. Rochelle. Special thanks go to Dr. Bouldin who was constantly available to advise on the research and writing of the thesis. Appreciation is also extended to Mr. J. M. Jansen of Oak Ridge National Laboratory for his direction and advice.

Further appreciation is expressed to my parents for their support and encouragement throughout my education. Thanks are also due to Mr. Robert Meyer for his help that made interfacing the speech system to the PDP-11/40 computer possible.

The research for this project was sponsored by the United States Department of Energy under Contract No. W-7405-eng-26 with the Union Carbide Corporation.

ABSTRACT

The need for communication between man and machine is increasing as the use of computers pervades the industrial and information processing worlds. Since speech is perhaps the most natural means of communication, computer speech output is well-suited to applications in which an operator's eyes and hands are busy monitoring situations. The Instrumentation and Controls Division of the Oak Ridge National Laboratory (ORNL) sponsored this project because of the potential need to use computer speech output in numerous applications throughout ORNL. The specific objectives were (1) to establish the development tools necessary to readily implement a speech output capability in a computer system and (2) to evaluate commercially available speech modules for quality of speech and flexibility.

This thesis reviews the essential aspects of speech synthesis and distinguishes between the two prevailing techniques: compressed digital speech and phonemic synthesis. It then presents the hardware details of the five speech modules evaluated. FORTRAN programs were written to facilitate message creation and retrieval with four of the modules driven by a PDP-11 minicomputer. The fifth module was driven directly by a computer terminal. The compressed digital speech modules (T.I. 990/306, T.S.I. Series 3D and

N.S. Digitalker) each contain a limited vocabulary produced by the manufacturers while both the phonemic synthesizers made by Votrax permit an almost unlimited set of sounds and words. A text-to-phoneme rules program was adapted for the PDP-11 (running under the RSX-11M operating system) to drive the Votrax Speech Pac module. However, the Votrax Type 'N Talk unit has its own built-in translator.

Comparison of these modules revealed that the compressed digital speech modules were superior in pronouncing words on an individual basis but lacked the inflection capability that permitted the phonemic synthesizers to generate more coherent phrases. These findings were necessarily highly subjective and dependent on the specific words and phrases studied. In addition, the rapid introduction of new modules by manufacturers will necessitate new comparisons. However, the results of this research verified that all of the modules studied do possess reasonable quality of speech that is suitable for man-machine applications. Furthermore, the development tools are now in place to permit the addition of computer speech output in such applications.

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION	1
Background and Objectives	1
Chapter Content	2
2. VOICE RESPONSE TECHNIQUES AND SYSTEMS	3
Human Synthesis of Speech	3
Voice Response Techniques	8
Analog recording	8
Compressed digital speech	9
Phonemic synthesis	11
Commercial Speech Modules	13
Texas Instruments TM 990/306	13
Votrax Speech Pac	16
Telesensory Series 3D speech system	21
National Semiconductor Digitaltalker	22
Type'N Talk speech system	24
3. VOICE RESPONSE DEVELOPMENT SYSTEM	27
Introduction	27
Overview of the Application Programs	27
Message Creation	29
Message Retrieval	37
Programs for Separate Modules	38
I/O Driver	40
Hardware Development	46
4. COMPARISON OF SPEECH SYNTHESIS MODULES	51
Quality of Speech	51
Flexibility	53
Price	53
Comparison between Votrax Speech Pac and Type'N Talk System	54
Software Rules Evaluation and Improvement	55
5. SUMMARY AND CONCLUSIONS	60
Summary	60
Conclusions	62

	PAGE
LIST OF REFERENCES	64
APPENDICES	66
Appendix A	67
Appendix B	84
Appendix C	91
Appendix D	92
VITA	93

LIST OF TABLES

TABLE	PAGE
2.1. Votrax 64 Phonemes	19
2.2. I.P.A. Phonemes to Votrax Conversion	20
3.1. Special Characters Used in the Rules	33
3.2. Names and Functions of Programs	43
5.1. Comparison of Speech Modules	60

LIST OF FIGURES

FIGURE	PAGE
2.1. An Acoustic Signal	5
2.2. Frequency Response Representation of Speech Signal	5
2.3. Speech Synthesizer Block Diagram	7
2.4. Voice Response System	9
2.5. Phonemic Synthesizer	12
2.6. T.I. Vocabulary	14
2.7. Votrax Vocabulary	17
2.8. T.S.I. Vocabulary	23
2.9. N.S. DT1000 Vocabulary	25
3.1. Flowchart for Program DEMO	30
3.2. Flowchart of N.R.L.'s Algorithm	34
3.3. Message Retrieval Program	39
3.4. Flowchart of COMPO Program	41
3.5. Example of Terminal Output of Program COMPO . .	42
3.6. I/O Flow Control	44
3.7. Switching Circuit Block Diagram	49
3.8. Audio Circuit	50

CHAPTER 1

INTRODUCTION

Background and Objectives

Man-machine communication by voice is an extremely important and exciting new application area which is just developing and which shows tremendous potential for widespread use. Since speech is perhaps the most natural means of communication, computer speech output is advancing into the world of business and industry rapidly.

One of the major applications of man-machine communication by voice is the voice response system. It is designed to respond to a request for information using spoken messages. This system is initiated by another information processing system or directly by a human seeking information from it. Its response consists of messages which are produced in the form of speech utterances.

These speech output systems are very helpful in an application where visibility is limited for the operator. A good example of using speech output is as a warning system in a process control situation where the operator's eyes and hands are busy monitoring situations.

Oak Ridge National Laboratory (ORNL) supported the project described in this report for eventual use in a

control warning system. Both limited and unlimited vocabulary systems have been evaluated. The main objective was to establish the development tools necessary to implement voice response systems. These tools must provide for efficient interaction between the system designer (the one who designs the vocabulary and/or messages) and the end-user. Having this goal in mind, the use of outside vendors for generating custom vocabulary is therefore not desirable. The system was designed in such a way that the end-user or the development engineer can create the custom vocabulary for his or her own application, thereby reducing development time. These development aids were also devised to assist in the evaluation of commercial speech modules.

Chapter Content

Chapter 2 discusses types of voice synthesis techniques. The hardware details of the speech modules manufactured by Texas Instruments (T.I.), Telesensory (T.S.I.), National Semiconductor (N.S.) and Votrax are also covered in this chapter. Chapter 3 discusses the voice response development system in terms of both the hardware and software. The system level software including the I/O drivers are also included in this chapter. Chapter 4 compares the performance of the systems. The summary and conclusions of the project are presented in Chapter 5.

CHAPTER 2

VOICE RESPONSE TECHNIQUES AND SYSTEMS

Human Synthesis of Speech

The basic phonological element of speech is the phoneme. It represents a simple sound that by itself cannot distinguish different words. Phonemes, together with speaker inflection, volume, emphasis, etc., are the fundamental building blocks of speech. The common American English language consists of approximately 38 to 40 phonemes, 14-16 vowel sounds and 24 consonant sounds.

Three basic types of sounds form speech: voiced sounds, fricative sounds and plosive sounds. The latter two can be called unvoiced sounds. In producing voiced sound, first air pressure is raised in the lungs. The air is then forced out of the lungs causing the vocal cords to vibrate and generate quasi-periodic broad-spectrum pulses. Examples of voiced sound are all of the vowel sounds.

All the hissing voice sounds like s, sh, and th are fricative sounds. This type of sound is generated when the vocal track is partly closed and air is forced through the constriction at high enough velocity to produce turbulence. Plosive sound, on the other hand, are produced with lips and tongue while the vocal track is closed completely. Air

pressure is built behind closure, and then abruptly opens the closure. Plosive sound is typified by the consonant p, t and k.

All the vocal sounds, both voiced and unvoiced, have a broad spectrum of frequencies ranging from about 100 Hz to more than 3 KHz. The vocal system such as the mouth cavity and tongue act as a time-varying filter of the sound wave generated by the broad spectrum sources.

Time domain of an acoustic signal is shown in Figure 2.1. Figure 2.1a shows the periodic excitation of air passing through vocal cords. The period is typically about 10 ms for an adult male. The resulting air disturbance outside the lips is shown in Figure 2.1b.

Because of the complexity of its time domain representation, analysis of an acoustic signal is best performed in the frequency domain. The Fourier analysis of the periodic pulse excitation in Figure 2.1a is the frequency-domain representation shown in Figure 2.2a consisting of fundamental frequency and multiples of this fundamental. Figure 2.2b shows the frequency response of the mouth cavity when the tongue, lips, and jaw are positioned to make a particular vowel sound. This frequency response is approximated with several harmonic-oscillator resonances called "formants."

Figure 2.2c is the frequency response of the acoustic signal of Figure 2.1b. It is the result of feeding line

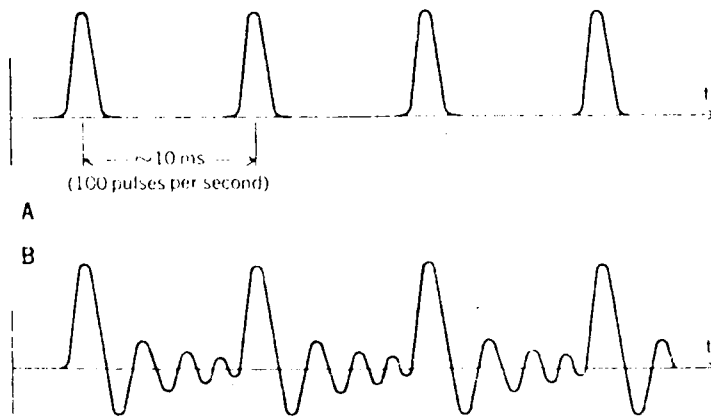


Figure 2.1. An Acoustic Signal [2].

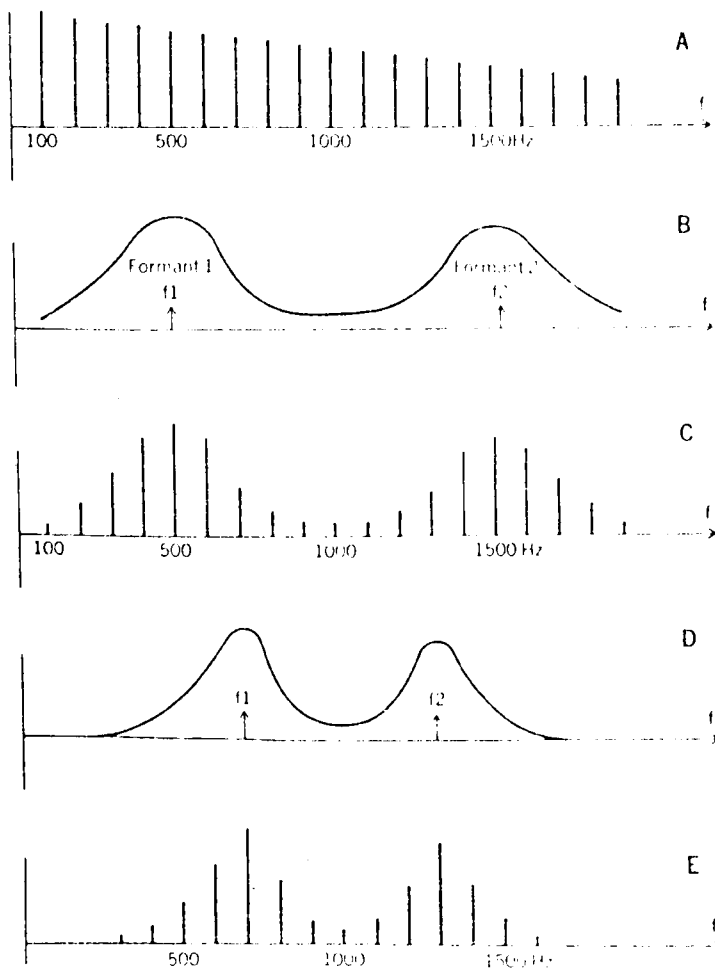


Figure 2.2. Frequency Response Representation of Speech Signal [2].

spectrum excitation of Figure 2.2a through the frequency response filter of Figure 2.2b. This signal represents a vowel. To make a different vowel, tongue, lips and jaw are moved to a different configuration, thus altering the frequency response of the cavity to that of Figure 2.2d. The frequency domain of the new vowel is in Figure 2.2e. To speak the same vowel at higher tone simply keep the mouth configuration fixed but increase the excitation rate. This results in a line spectrum of the same envelope but with the lines closer to each other.

The circuit to produce various sounds can be built consisting of a pulse generator that drives a filter whose formants can be varied to the formants of the vowel to be spoken. Other speech sounds such as the hissing voice require a noise excitation. A block diagram of a circuit capable of producing speech is shown in Figure 2.3. The frequency and amplitude of the periodic pulses are controlled by f_0 and a_0 respectively. F_1 , F_2 and F_3 are the filters driven by the periodic pulse excitation. Their center frequencies are set by parameters f_1 , f_2 and f_3 , corresponding to the formants used to approximate the frequency response of the mouth. The pulse excitation also passes through a "nasal" resonator, which introduces properties characteristic of sounds such as /m/ and /n/, produced as the mouth cavity opens into the nasal cavity. As all these parameters are updated continuously, synthesis of human speech is possible.

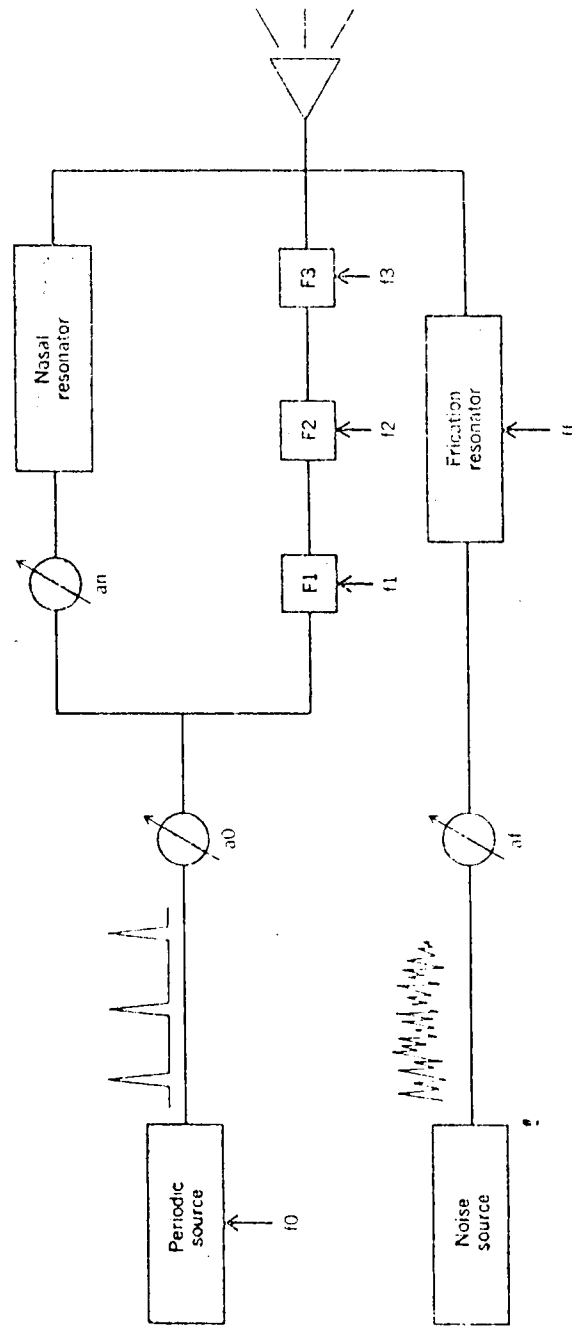


Figure 2.3. Speech Synthesizer Block Diagram [2].

Voice Response Techniques

Because of the dynamic properties of the human-speech mechanism that have to be simulated, synthetic speech of good quality is not easily achieved. Depending on the application method, speech output can be divided into three types of techniques: (1) analog recording, (2) compressed digital speech, and (3) speech synthesis from phonemes. Each type has its own advantages and disadvantages and will be discussed separately.

Analog Recording

Analog recording is a straightforward way to provide speech output under computer control. The most basic type is the one that uses recorded audio on one track of a magnetic tape and the digital address of an individual message on the other, permitting the computer to select a particular message to speak. Because of the slow serial access along the tape to the appropriate message, this approach is unsuitable for many situations. A random-access audio disk has been developed by the Plato computer-based education system [2]. The access time to any of the messages is a fraction of a second. This device has the advantage of high quality speech, true random access, a large number of messages and extremely low data rates. Since the playhead of this device is pneumatically controlled, a need for

compressed air is one of its disadvantages. Also a sentence constructed from words recorded separately has poor intonation and rhythm. Due to these disadvantages, the analog recording method was excluded from this project.

Compressed Digital Speech

Another computer voice response technique is the compressed digital speech method in which parameters of words or phrases to be spoken are stored in a storage medium, usually ROM or EPROM [1-2]. The output messages are created by concatenating isolated words or phrases whose parameters are prestored in the medium. Figure 2.4 shows an example of a complete voice response system.

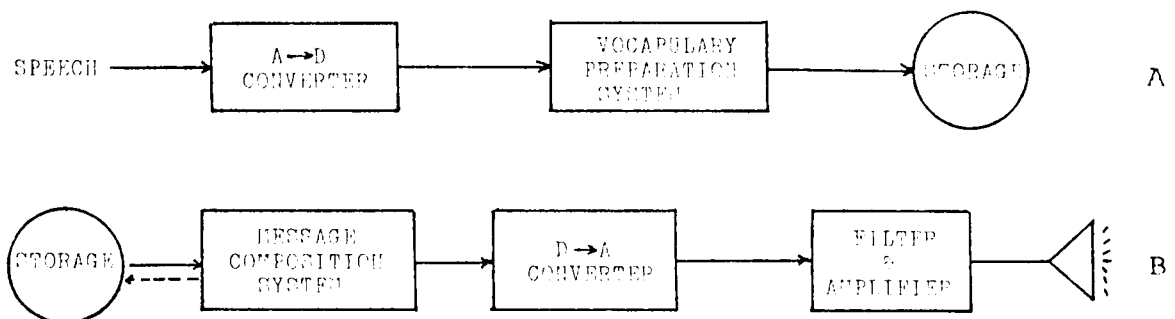


Figure 2.4. Voice Response System.

The system in Figure 2.4 can be divided into two parts. Part (a) digitizes the analog speech signal by an A→D converter, the output of which has a very high data rate of about 100 Kb/s. In the vocabulary preparation system,

Fourier analysis and correlation techniques are used to extract the frequency domain parameters that correspond to the input speech word. This change to frequency domain helps to reduce the data rate to 10 Kb/s. The outputs of the vocabulary preparation system consist of parameters of the individual word or phrase and are stored in memory. Later these are recalled by using the address corresponding to it. This process is usually done by the manufacturer of the system so the user does not have the capability to change the vocabulary.

Figure 2.4b shows the regeneration of the compressed speech. The main parts of the message composition system are the parametric synthesizer and the address decoder. First the decoder decodes the address of the word to be spoken, then the corresponding parameters prestored in the memory are fetched and fed into the parametric synthesizer. The encoded parameters are then reexpanded and the required word is spoken through a D→A converter, a filter, and an amplifier. The limited vocabulary speech synthesizers like the T.I. speech module or the Telesensory system which are used in this project are configured like Figure 2.4b. Some companies let the users mail in an audio tape of the messages required in their own application and use a system like Figure 2.4a to generate compressed speech in ROM. The user then is supplied with the system of Figure 2.4b with

the required words in ROM and the addresses of those individual words. To speak a sentence the user issues the correct sequence of addresses of words in the sentence.

Parameters of speech, e.g., pitch, formant and spectra are estimated using a technique called Linear Predictive Coding (L.P.C.). The basic idea behind this technique is that a speech sample can be approximated as a linear combination of the past samples. This coding method has the advantage of very low data rates resulting in a very low memory requirement. The mathematical details of the coding technique are beyond the scope of this project but the reader can consult reference [1] for further information.

The elimination of mechanical parts and the ability to retrieve data faster are the advantages of the compressed digital technique over the analog recording method. The disadvantage is, of course, that the quality of the speech is poorer than that of the analog recording. Also in the compressed digital system a vocabulary is limited and the assistance of the manufacturer is required for expansion.

Phonemic Synthesis

The third type of voice response technique is phonemic synthesis. First, text to be spoken is analyzed for phonemes (the building blocks of speech) using text-phonemes software rules as shown in the block diagram of Figure 2.5. The next stage involves converting the phonemes, durations, loudness,

and the pitch into appropriate frequency-domain parameters. The parametric synthesizer is driven by these parameters to produce speech.

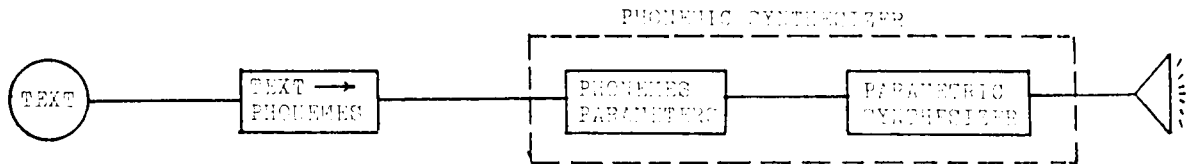


Figure 2.5. Phonemic Synthesizer.

In some systems, a phonemic synthesizer replaces the last two stages in Figure 2.5. It is obvious that a phonemic synthesizer is much easier to use than a "bare" parametric synthesizer because it is driven by a low data rate phoneme string instead of a high data rate set of frequency-domain speech parameters. The most prominent phonemic synthesizer is the SC-01 chip manufactured by Votrax. This synthesizer is used in this project and will be discussed in detail in the subsequent chapter.

The major advantage of this system over the previously mentioned ones is that it has unlimited vocabulary. In addition, transferring from one phoneme to another is done dynamically, thus eliminating the problem of pauses between words not normally contained in natural human speech. The main disadvantage is that the quality of speech using this

technique is still comparatively poor. Also the text-to-speech system requires some linguistic knowledge not normally found in a system designer. Fortunately, some rules to convert the text to phonemes have been written. The set of software rules requires at least 15k words of memory space in a PDP-11 system.

Commercial Speech Modules

In this project five speech modules were used and compared. They are Texas Instruments TM 990/306, Telesensory S3D, National Semiconductor Digitalker, Votrax Speech Pac and Votrax Type 'N Talk system. The hardware description of each speech module will be described separately in this chapter.

Texas Instruments TM 990/306

This module uses the L.P.C. technique for compressing and synthesizing words. It can be used to construct sentences having a somewhat flat inflection. This module has been specially designed to operate in noisy environments such as found in assembly lines, stock-rooms or automobiles. Other features of this module include:

- Self-contained vocabulary of 179 words (see Figure 2.6).
- External interface allows control by any computer system (in this project a PDP 11/40 with RSX-11M operating system).
- On-board amplifier provides up to two watts of power to an 8 ohm speaker.

A	East	Is	P	Speed
Abort	Eight	J	Pass	Start
Adjust	Electrician	Jog	Passed	Stop
Alert	Eleven	K	Percent	Switch
All	Enter	L	Pico	T
Amps	Equal	Left	Plus	Temperature
And	Exit	Light	Point	Ten
Automatic	F	Line	Position	Test
B	Fail	Low	Pound	The
Back	Farad	M	Power	Thousand
Button	Fast	Machine	Press	Three
C	Feet	Maker	Pressure	Time
Calibrate	Fire	Manual	Priority	Timer
Call	Five	Measure	Probe	Tool
Cancel	Flow	Mega	Processing	Turn
Carpenter	Foreman	Meter	Pull	Twelve
Check	Four	Micro	Push	Two
Clock	Frequency	Mill	Put	U
Close	From	Milli	Q	Under
Control	G	Minus	R	Unit
Crane	Gage	Minutes	Range	Up
Crease	Gate	Motor	Ready	V
Cycle	Get	Move	Red	Valve
D	Go	N	Repair	Volts
Danger	Green	Nine	Repeat	W
Days	H	North	Replace	Wait
Degrees	Henry	Number	Right	Ward
Delay 1	Hertz	O	S	Watts
Delay 2	High	Of	Safe	Welder
Delay 3	Hold	Off	Seconds	West
Delay 4	Hours	Ohms	Set	X
Device	Hundred	On	Seven	Y
Direction	I	One	Shut	Yellow
Display	Inch	Open	Six	Z
Door	Initialize	Operator	Slow	Zero
Down	Inspector	Out	Smoke	
E	Intruder	Over	South	

Figure 2.6. T.I. Vocabulary.

- An option to bypass the amplifier to use a custom amplifier.
- Interrupt or polled-status operation.
- Speech data is stored in a TMS 2716 EPROM.
- Uses +12V, -12V and +5V power supplies.

The system uses a 14-bit address, one talk/no-talk bit and one status bit. To speak a word the computer issues the address of the word and sets the talk bit to zero. The talk bit has to be set to one when the module finishes speaking that word or else the word will be spoken repeatedly. The status bit will tell whether the system is busy (speaking) or not. To speak a sentence (string of addresses) the computer initiates the first word, lets the system speak that word until it is ready for the next one (the status bit will go to zero), and then continues this procedure iteratively. Two techniques can be used to monitor the status bit, polled status and interrupt driven. The advantage of using the interrupt driven mode is that it frees the CPU to do other tasks while the speech module is still talking. This results in efficient use of computer time and will be used throughout this project. The status bit was inverted because the signal driving the PDP-11 interrupt line must make a zero to one transition instead of one to zero as it was.

Votrax Speech Pac

This is a low cost system with speech synthesis capability. The heart of the system is the SC-01 single chip speech synthesizer. This chip phonetically synthesizes continuous speech of unlimited vocabulary from low data rate inputs. Speech is synthesized by combining phonemes in the appropriate sequence. The SC-01 Speech Synthesizer contains 64 different phonemes which are accessed by a 6-bit code. It is the proper sequential combination of these phoneme codes that creates continuous speech.

Along with the SC-01 speech synthesizer chip, Speech Pac contains an audio amplifier, a parallel interface (TTL compatible) and a 16K EPROM-based vocabulary (expandable to 32K). Both the volume and the master clock frequency may be controlled externally. Once Speech Pac has been interfaced, the user can access vocabulary in one of two ways. First, any of the 255 preprogrammed words (as shown in Figure 2.7) stored in the EPROM may be selected for output. Second, the user may access the phonemes directly to create any desired vocabulary. Also, both approaches may be mixed.

A bit called SEL is used to select the mode of operation to be either word mode or phoneme mode. In word mode the system uses an 8-bit address to select the prestored word. In phoneme mode only six bits are needed and the system uses the lower six bits of the eight address bits as the phoneme

A	Farad	Move	Service
Abort	Fast	Much	Set
Accept	Feet	N	Seven
Add	Fifteen	Name	Seventy
Address	Fifty	Nine	Shut
Adjust	Fire	Ninty	Six
Alert	Five	No	Sixty
All	Flow	Noise	Slash
Amphere	Foreman	North	Slow
And	Forty	Number	Smoke
Ask	Four	O	South
Automatic	From	Object	Space
B	Full	Of	Speed
Back	Function	Off	Star
Basic	G	Ohm	Start
Board	Gauge	Omit	Stop
Break	Gate	On	Swap
Button	Get	One	Switch
By	Glitch	Open	T
Byte	Go	Operate	Teen
C	Great	Ounce	Ten
Call	Green	Out	Test
Cancel	H	Over	Than
Carpenter	Have	P	The
Case	Help	Paren	Thirteen
Cent	Henry	Part	Thirty
Centi	Hertz	Pass	Thousand
Change	High	Past	Three
Check	Hold	Pause	Time
Clear	Hour	Percent	Tone
Clock	How	Pico	Tool
Close	Hundred	Please	Total
Colon	I	Plus	Turn
Comma	Inch	Point	Twelve
Command	Initial	Position	Twenty
Connect	Insert	Pound	Two
Control	Inspect	Power	U
Crane	Intrude	Press	Under
Crease	Is	Pressure	Unit
Cycle	J	Price	Up
D	Jog	Probe	V
Danger	K	Process	Value
Day	L	Program	Valve
Degree	Left	Full	Volt
Device	Less	Push	W
Direction	Light	Put	Ward
Display	Line	Q	Weight
Dollar	Low	Quote	Welder
Door	M	R	West
Down	Machine	Range	What
E	Make	Read	X
East	Manual	Ready	Y
Eight	Mark	Red	Yellow
Eighty	Measure	Repair	Yes
Eleven	Mega	Repeat	Z
End	Meter	Replace	Zero
Enter	Micro	Request	
Equal	Mill	Return	
Er	Milli	Right	
Exit	Minus	S	"PHRASE 1"
F	Minute	"S"	
Facility	Mistake	Safe	
Fail	Module	Second	
"PAUSE"	Motor	Semi	"PHRASE 2"

"PHRASE 1" — 'THIS IS THE VOTRAX SPEECH-FAC'

"PHRASE 2" — 'I'VE A 250 WORD MEMORY'

Figure 2.7. Votrax Vocabulary.

selector. One difference between this system and the T.I. speech module is that it does not have the talk/no-talk bit. Instead, a strobe line is provided and after the address or phoneme is placed on the address/phoneme lines, this strobe line must be driven high and then low to strobe the data into the system and make the system speak. A zero to one transition will appear on the status bit and drive the interrupt line when the system is ready for the next data. Two inflection lines are also provided to control the inflection of the speech output. When a phoneme is strobed into the SC-01, the new inflection code will also be latched.

The 64 phonemes for the SC-01 synthesizer are listed in Table 2.1. To speak a word, find the string of votrax phonemes that correspond to that word and output them sequentially. This process can be made easier with the help of the International Phonetic Alphabet (I.P.A.) dictionary [5] which gives the I.P.A. phonemes of the word. The corresponding Votrax phonemes can be found by using IPA→Votrax phonemes translation [4] in Table 2.2.

The Naval Research Laboratory (NRL) has developed a set of software rules called "English text to Votrax parameters" Algorithm. These software rules translate the text into the Votrax phonemes corresponding to it. The original version was written for the DEC-10 computer but was not used in this project. The one used was obtained from DECUS [3] and was

Table 2.1
Votrax 64 Phonemes

Phoneme Code	Phoneme Symbol	Duration (ms)	Example Word
00	EH3	59	jacket
01	EH2	71	enlist
02	EH1	121	heavy
03	PA0	47	no sound
04	DT	47	butter
05	A2	71	made
06	A1	103	made
07	ZH	90	azure
08	AH2	71	honest
09	I3	55	inhibit
0A	I2	80	inhibit
0B	I1	121	inhibit
0C	M	103	mat
0D	N	80	sun
0E	B	71	bag
0F	V	71	van
10	CH ^a	71	chip
11	SH	121	shop
12	Z	71	zoo
13	AW1	146	lawful
14	NG	121	thing
15	AH1	146	father
16	OO1	103	looking
17	OO	185	book
18	L	103	land
19	K _t	80	trick
1A	J _t	47	judge
1B	H	71	hello
1C	G	71	get
1D	F	103	fast
1E	D	55	paid
1F	S	90	pass
20	A	185	day
21	AY	65	day
22	Y1	80	yard
23	UH3	47	mission
24	AH	250	mop
25	P	103	past
26	O	185	cold
27	I	185	pin
28	U	185	move
29	Y	103	any
2A	T	71	tan
2B	R	90	red
2C	E	185	meet
2D	W	80	win
2E	AE	185	dad
2F	AE1	103	after
30	AW2	90	salty
31	UH2	71	about
32	UH1	103	uncle
33	UH	185	cup
34	O2	80	for
35	O1	121	aboard
36	IU	59	you
37	U1	90	you
38	THV	80	the
39	TH	71	thin
3A	ER	146	bird
3B	EH	185	get
3C	E1	121	be
3D	AW	250	call
3E	PA1	185	no sound
3F	STOP	47	no sound

^a/T/ must precede /CH/ to produce CH sound.

^t/D/ must precede /J/ to produce J sound.

Table 2.2
I.P.A. Phonemes to Votrax Conversion

Standard I.P.A.	Votrax Hex	Example
i	2C	be <u>e</u> t
I	27	bi <u>t</u>
e	20,21	ga <u>t</u> e
ɛ	3B	ge <u>t</u>
æ	2E	fa <u>t</u>
a	15	fa <u>t</u> her
ɔ	3D	la <u>w</u> n
ɔ̃	35,37	lo <u>n</u> e
ʊ	17	fu <u>l</u> l
u	36,28	fo <u>o</u> l
ʊ̃	3A	mu <u>r</u> der
ə	31	ab <u>o</u> ut
ʌ	33	bu <u>t</u>
aɪ	15,00,3C	hi <u>d</u> e
aʊ	24,35	ho <u>w</u>
ɔɪ	35,3C	to <u>y</u>
p	25	pa <u>c</u> k
b	0E	ba <u>c</u> k
t	2A	ti <u>m</u> e
d	1E	di <u>m</u> e
k	19	co <u>a</u> t
g	1C	go <u>a</u> t
f	1D	fa <u>u</u> lt
v	0F	va <u>u</u> lve
θ	39	et <u>h</u> er
ð	38	ei <u>t</u> her
s	1F	su <u>e</u>
z	12	zo <u>o</u>
ʃ	11	lea <u>s</u> h
ʒ	07	lei <u>s</u> ure
h	1B	ho <u>w</u>
m	0C	su <u>m</u>
n	0D	su <u>n</u>
ŋ	14	su <u>n</u> g
l	18	la <u>u</u> gh
w	2D	wea <u>r</u>
j	29	yo <u>u</u> ng
r	2B	ra <u>t</u> e
tf	2A,10	cha <u>r</u>
d	1E,1A	ja <u>r</u>
hw	1B,2D	we <u>r</u> e

written for a PDP-11 computer under the RT-11 system. The rules needed some modification to accommodate the RSX-11M operating system and will be discussed in detail in the next chapter as a part of software development.

Telesensory Series 3D Speech System

The Series 3D Speech Module by Telesensory Inc. is a complete speech synthesizer circuit board designed for simple system integration. The module consists of Telesensory's proprietary speech synthesizer, a 119 word basic vocabulary, a speech filter and an amplifier, and an extra ROM socket. All of these are mounted on a small printed circuit board. Interfacing is easy because the I/O is TTL compatible and because the system is powered by a single +5 Volt supply. Both standard and custom vocabulary memories can be accommodated with capacity of up to 256 utterances for approximately 100 seconds of synthesized human speech.

Interfacing the system can be done by the same way as was described in the Votrax section. Synthesizing an utterance starts with placing eight bits that represent the desired utterance on the Word Pointer data lines. The Word Pointer selects one of up to 256 utterances, word or phrases, stored in one or two 64K bit (or smaller) memory devices. The pointer number is latched into the module on the rising edge of the strobe pulse. Immediately thereafter the speech

output begins. Two output signals, BUSY or XBUSY can be monitored to determine when the synthesis of current utterance is completed. The difference between the outputs is that the XBUSY has about 100 ms delay before giving the rising edge and was used in this project. The original 119 word Basic Vocabulary is shown in Figure 2.8.

National Semiconductor Digtalker

The DT100 Digtalker speech synthesis evaluation board is a complete speech synthesis board built around the MM54104 speech processor chip (S.P.C.). It contains all components required to output speech upon demand: two MAXI-ROMS (MM52164) contain 128 individual words, linear filter, audio amplifier, keyboard, and a COPS microcontroller complete with stored data programmed to provide the various functions on the board. The board contains a twelve pad keyboard to control the microcontroller. A COPS program is provided which permits the user to: (1) sequentially output each word automatically; (2) repeat any desired word; (3) build and store several short phrases for outputting when desired; (4) output a "canned" phrase which permits insertion and changing of a word in a phrase; (5) play a simple game which requires some interaction between the keyboard and the user; and (6) output the equivalent of decimal number inputs.

A	L	S
AMP	LEFT	SECOND
AND	LIMIT	SELECT
AT	LITER	SET
B	LOW	SEVEN
C	M	SIX
CALL	METER	SOUND
CENT	MILLION	START
CHECK	MINUS	STOP
CODE	MODE	SWITCH
CONTINUE	MOVE	T
CREASE	N	TEEN
D	NEXT	TEN
DATE	NINE	TEST
DEGREE	NUMBER	THANK
DOLLAR	O	THIR
E	OFF	THOUSAND
EIGHT	OHM	THREE
ELEVEN	ON	TIME
ENTER	ONE	TOTAL
EQUAL	OUNCES	TRY
ERROR	OUT	TURN
F	OVER	TWELVE
FAIL	P	TWENTY
FIF	PER	TWO
FIVE	PLEASE	U
FOUR	PLUS	V
G	POINT	VOLT
GRAM	POSITION	W
H	POUND	X
HERTZ	PUT	Y
HIGH	Q	Z
HOURL	R	ZERO
HUNDRED	RATE	ZOUND
I	RE	
IF	READY	
IN	RIGHT	
IS		
J		
K		

Figure 2.8. T.S.I. Vocabulary.

The DT1000 is designed so that it is possible to access only the Digitalker portion of the board through the edge connector for interfacing to an external processor system. The Digitalker portion is defined as the speech processor chip (S.P.C.) and the speech ROM(S) which contain the actual vocabulary (see Figure 2.9). It is necessary to remove the COP402 from the DT100 in this mode of operation. As the purpose of this project is to interface the module to the outside world, the microcontroller was removed and left the speech module with none of the options mentioned before. The Digitalker needs a strobe to start speaking a word like the S3D T.S.I. speech module and interfacing can be done the same way. The reader who is interested in the seven basic modes of operation of the COPS program should consult [9].

Type'N Talk Speech System

The Type'N Talk (T.N.T.) is a complete text-to-speech synthesizer produced by Votrax. T.N.T. is a small, light-weight unit that may be interfaced to a computer, modem or any RS-232 compatible serial device. It has a built-in amplifier system. The heart of the system is still the Votrax SC01 phoneme synthesizer chip like the Votrax Speech Pac. The thing that makes the T.N.T. special is that it includes a text-to-speech translation algorithm thus eliminating the need of the software translation program as in case of the

THIS IS DIGITALKER	IS	Q
ONE	IT	R
TWO	KILO	S
THREE	LEFT	T
FOUR	LESS	U
FIVE	LESSER	V
SIX	LIMIT	W
SEVEN	LOW	X
EIGHT	LOWER	Y
NINE	MARK	Z
TEN	METER	AGAIN
ELEVEN	MILE	AMPERE
TWELVE	MILLI	AND
THIRTEEN	MINUS	AT
FOURTEEN	MINUTE	CANCEL
FIFTEEN	NEAR	CASE
SIXTEEN	NUMBER	CENT
SEVENTEEN	OF	400HERTZ TONE
EIGHTEEN	OFF	80HERTZ TONE
NINETEEN	ON	20MS SILENCE
TWENTY	OUT	40MS SILENCE
THIRTY	OVER	80MS SILENCE
FORTY	PARENTHESIS	160MS SILENCE
FIFTY	PERCENT	320MS SILENCE
SIXTY	PLEASE	CENTI
SEVENTY	PLUS	CHECK
EIGHTY	POINT	COMMA
NINETY	POUND	CONTROL
HUNDRED	PULSES	DANGER
THOUSAND	RATE	DEGREE
MILLION	RE	DOLLAR
ZERO	READY	DOWN
A	RIGHT	EQUAL
B	SS (Note 1)	ERROR
C	SECOND	FEET
D	SET	FLOW
E	SPACE	FUEL
F	SPEED	GALLON
G	STAR	GO
H	START	GRAM
I	STOP	GREAT
J	THAN	GREATER
K	THE	HAVE
L	TIME	HIGH
M	TRY	HIGHER
N	UP	HOURL
O	VOLT	IN
P	WEIGHT (Note 2)	INCHES

Figure 2.9. N.S. DT1000 Vocabulary.

Votrax Speech Pac. For this reason, it will not slow down the throughput of the user's computer doing time-consuming text translations. With this responsibility off-loaded to a T.N.T., the user's computer is free to perform other tasks while speech is in progress.

The sentence to be spoken through T.N.T. can be inserted directly into the user's software programs similar to the way statements for activating a printer or CRT are written. Data in the form of ASCII codes can be sent to T.N.T. whenever the handshake line indicates that it is not busy. The T.N.T. system allows the user four modes of operation. PSEND mode allows the user's program to collect the phonemes of the words spoken from the T.N.T. If mode ECHO is selected, all data received by T.N.T. will be relayed to the next unit or peripheral in a daisy chain. This mode will be helpful only if more than one T.N.T. unit is linked together (daisy chained) to provide multiple voice outputs from a single computer interface. The mode CAPS ON forces all the words having two or more letters capitalized at the beginning to be spelled out. While the text is being sent to T.N.T., if more than approximately four seconds elapses after the last character was received, the text in the input buffer is translated and spoken. If this timer feature is not desired it can be disabled with the TIMER OFF command mode. The RESET mode resets all T.N.T. modes to their power-up settings, i.e., PSEND OFF, CAPS OFF and TIMER ON.

CHAPTER 3

VOICE RESPONSE DEVELOPMENT SYSTEM

Introduction

Once all the hardware components were received and tested separately, creation of the development tools began. The strategy was to utilize common hardware and software when possible. Thus, the four modules were put together in the same box and a computer-driven switch installed to select an individual system. This hardware was interfaced to the PDP-11 computer through a DR-11C parallel interface board. The PDP-11 computer runs under the RSX-11M time sharing operating system so an I/O driver was written to take care of all I/O processing (speech output).

In this chapter, all the software aspects, both user and system level, will be discussed. Listing of the programs described are given in Appendix A. At the end of the chapter the hardware development will be discussed.

Overview of the Application Programs

Four programs were written to drive the four speech modules. Programs SPEAK1, SPEAK2 and SPEAK3 speak using the T.I., T.S.I. and Digitalker respectively. The user types in the message he wants the module to speak. If any

of the words typed in is not contained in the vocabulary, the word will be typed out as an error message and the sentence will not be spoken. Program VOTRA7 contains the text to Votrax phonemes translation rules. The user just types in the sentence which will be translated by the rules and spoken through the Votrax system. For the user's convenience, a program called COMPO was written. This program lets the user select which modules he/she want to use, i.e., which of the four programs mentioned above.

Program DEMO works like all the previous programs except that it does not work with just an individual module but all of them at the same time. When the user types in the messages, it searches the vocabulary of the T.I., Digitalker, and T.S.I.'s modules respectively. If all fail, the rules will be used to translate the word into the corresponding Votrax phonemes.

All the above programs have the option to store the data spoken (phonemes or address) in a prespecified data file. These prestored data can later be retrieved by any programs that call subroutine TALK with the data file to be spoken as argument.

A program called DRTST was written during the system development phase. It allows the user to input data (addresses or phonemes) manually one by one. The purpose is to test the I/O driver and the hardware during development time.

Message Creation

In real-time applications such as in alarm signaling, external events will cause the system to speak prestored messages. The purpose of the message creation program is to make editing or storing the messages as convenient as possible and to enable the user to avoid learning the details of the speech systems.

A program DEMO was written for two purposes, to demonstrate the type-in speak-out system and to store data (message creation). In this program, the user types in a message he/she wants to store with spaces between words. When the Return key is hit, the message will immediately be spoken once and at the same time stored in a file to be retrieved later on. The program divides the text into words and searches the addresses of words individually. The T.I. module has the highest priority so each word is searched in its vocabulary first. If the search in the T.I. vocabulary fails, a similar search is done in the Telesensory's and the Digitalker's vocabularies. The phoneme rule is given the lowest priority and is used only if the word is not in the other three speech modules' vocabularies. With this approach all the words in the sentence will be spoken.

The flowchart of the whole program is in Figure 3.1.

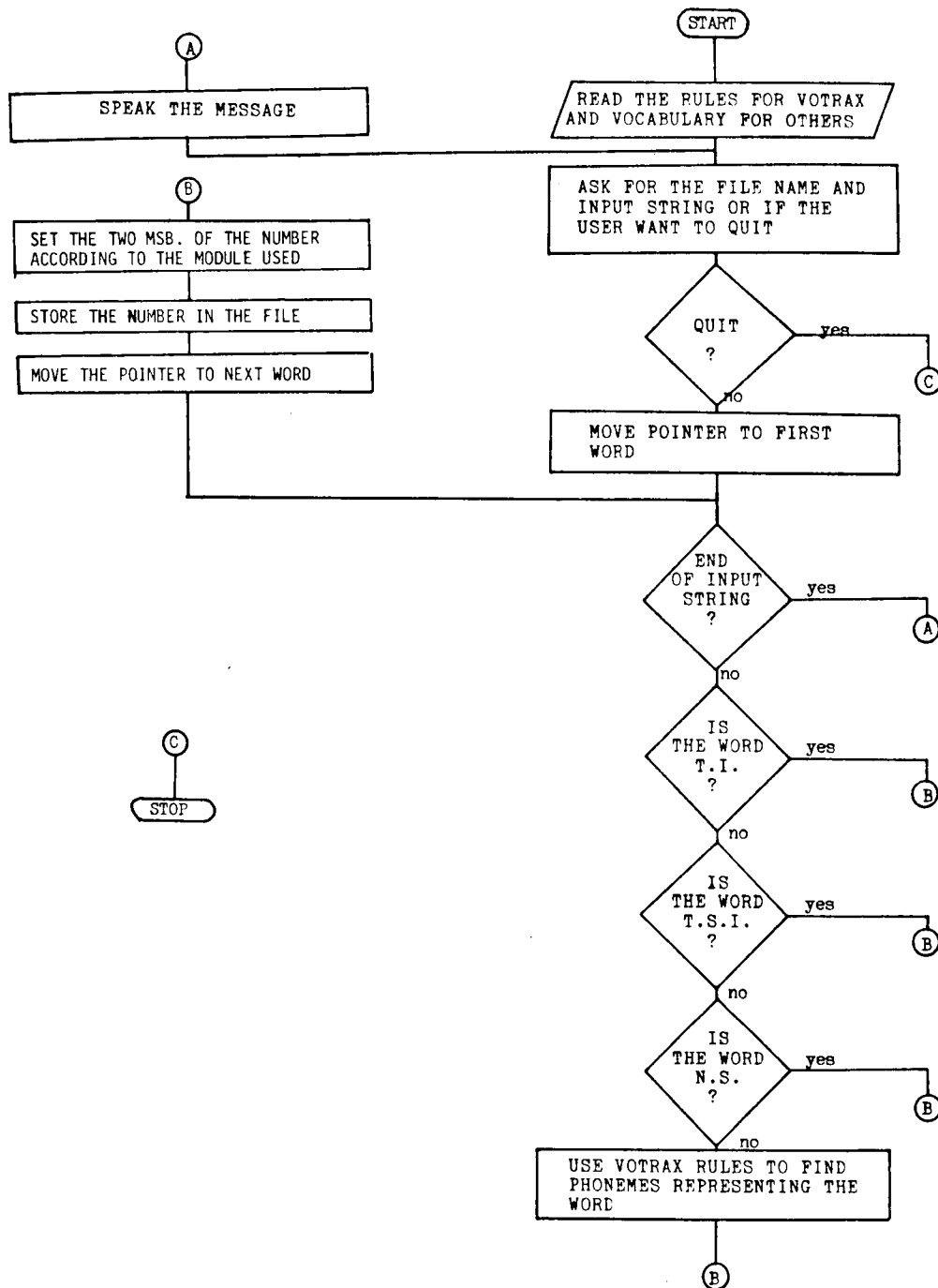


Figure 3.1. Flowchart for Program DEMO.

The program was written with the help of some subroutines. For example, "LEN" finds the length of the word from the pointer to the first space. "LOOK UP" searches the word at the current pointer position in the vocabulary list and comes out with the index of a word if the search is successful. If the search fails the index value is equal to zero and the program knows that the word is not in that particular vocabulary list.

The software rules for the Votrax system were modified from N.R.L.'s "English text to Votrax parameters" algorithm [3-4]. The modification was needed because the original program can run only on an RT-11 system and does not take into account the usages of T.I., Digitalker and T.S.I., all of which have higher priority than the rules. In addition, the RSX-11M system does not contain the string manipulation library like the RT-11, so a subroutine called "STRLIB" was written and linked with the main program during the task building process.

The N.R.L. team developed a set of 330 rules which statistically were shown to produce correct pronunciation for about 90% of the words in an average sample of English text [4]. Each rule is of the form:

$$A(B)C = D$$

That is, "the character string B, occurring with left content A and right content C, results in the pronunciation D." D, in this case, is one or more of the 64 Votrax input symbols. B is the character or character string to be translated. A and C are characters, character strings or special symbols representing a class of character strings which denote categories of sounds such as vowels, voiced consonants, etc. A list of the symbols is shown in Table 3.1. Blanks denote beginnings and ends of words. An unlimited number of rules can be added to improve the translation. However, when the user introduces more rules, he must ensure that more specific rules precede more general rules. Rule order is important because the algorithm ceases scanning when the first success is encountered. The flowchart of the algorithm is in Figure 3.2.

The algorithm to check prefixes and suffixes is as follows. The first prefix argument of the rules (first to the left of B(I)) is compared with the first set of the alphabets to the left of the pointer. The prefix argument may be a character, a vowel, two vowels, etc. If the comparison turns out successfully, the next argument of the rule is compared with the next set of alphabets more to the left of the pointer into AD. This is done repeatedly until either no match occurs (failure) or the prefix rule string is exhausted (success, since there must have been matches

Table 3.1
Special Characters Used in the Rules

Symbol	Meaning
#	1 or more vowels (A,E,I,O,U, or Y)
?	2 or more vowels
^	1 consonant (B,C,D,F,G,H,J,K,L,M,N,P,Q,R,S,T,V,W, X, or Z)
:	0 or more consonants
*	1 or more consonants
.	voiced consonant
\$	consonant plus E or I
%	suffix (E, ER, ES, ED, ING, or ELY)
&	sibilant (S,C,G,Z,X,J,CH, or SH)
@	nonpalatal (T,S,R,D,L,Z,N,J,TH,CH, or SH)
+	front vowel (E,I, or Y)

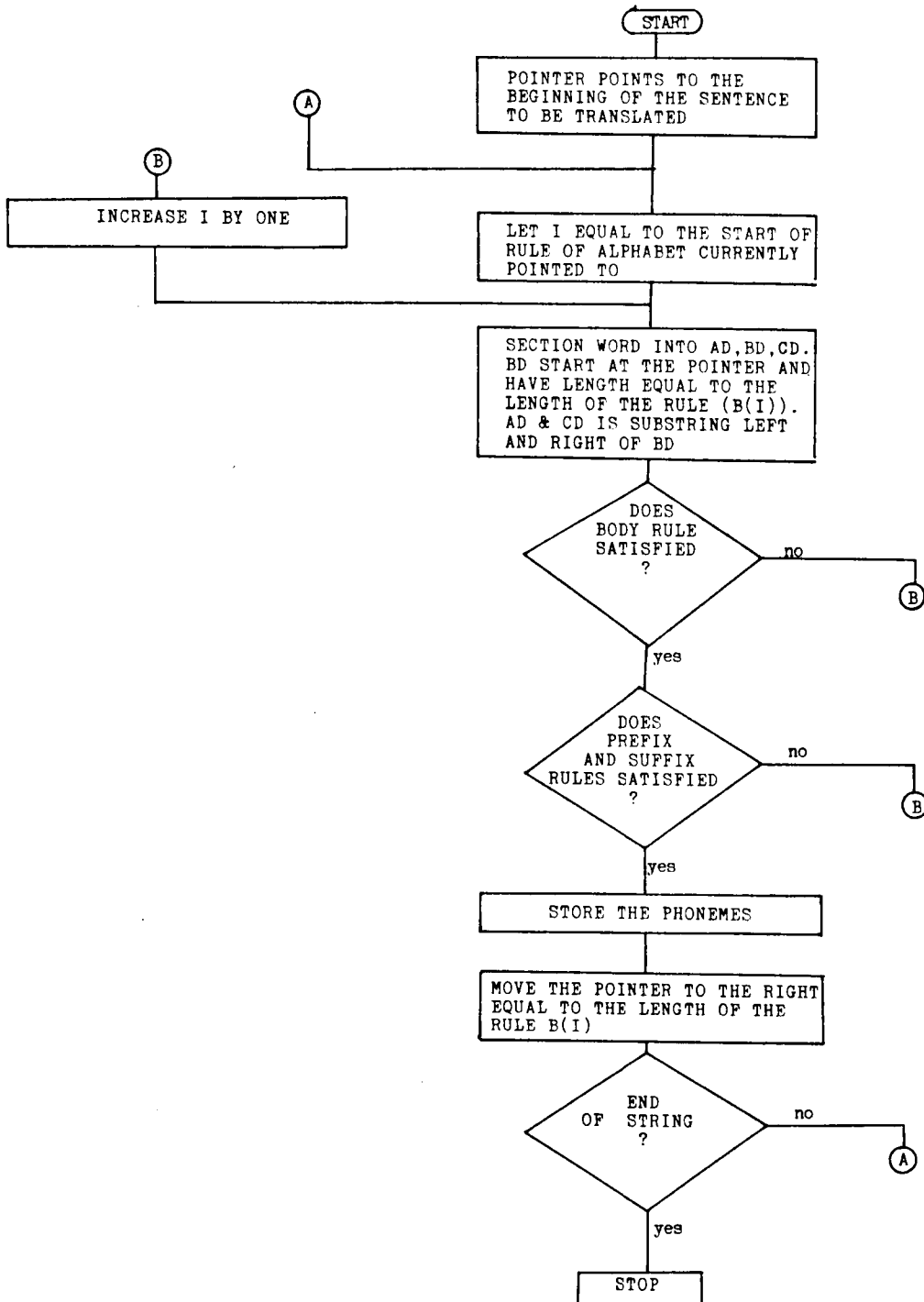


Figure 3.2. Flowchart of N.R.L.'s Algorithm.

up to that point) or AD is exhausted (failure, since there are still portions of the prefix rule string to be matched). The algorithm is carried out via two subroutines "FNPRES" and "FNSUF." For a given body, rule match "FNPRES" is called repeatedly with successively more left portions of the prefix rule, A(I), until the algorithm stops by one of the three reasons mentioned above. If "FNPRES" is successful, "FNSUF" is used to test the suffix rule, C(I), against CD in a similar manner. As an illustration of the operation of the algorithm, here is a worked example: the translation of RATIO using the English-to-IPA rules.

To the left of the first character, R, the program adds a blank to delimit the word, and the scan starts with the R, as indicated with a pointer: $\uparrow$ RATIO. The program searches the R-rules, the rules with R as first character between brackets. The first R-rule, "[RE]=/RIY/" forces the program to divide "RATIO" into the following part: "RA" as BD, "TIO" as CD. This rule fails to match for the obvious reason that "RE" is not equal to "RA." The next and last R-rules, "[R]=/R/," is the default and matches any R not matched by earlier rules. Consequently, /R/ goes to the output string and the scan moves past the R to A: R $\uparrow$ ATIO.

The search of A-rules turns up no match before "[A]^+# =/EY/," which applies when A is followed by a single consonant (^), a front vowel (E, I, or Y) and another vowel (#). At

this point, BD is "A," AD is "R" and CD is "TIO." As BD ("A") equals B(I) ("A") subroutine FNPRES is called with values of AD (blank) and A(I) (blank). The subroutine returns with a flag telling that the match is a success since the rule does not require anything for a prefix (indicated by blank). FNSUF is called with the first of CD(T) and first of C(I) (^). As the match succeeds the FNSUF is called twice more with "I," "+" and "O," "#" respectively. After the third call succeeds, C(I) is exhausted and makes the rules match succeed for "[A]^+##/EY/."

The program keeps scanning with the similar method described above. The rules that match are "[TI]O=/SH/" and "[O]=/OW/." The pointer is now past the O to the blank at the end of the word: RATIO_↑. The output string is /R/ /EY/ /SH/ /OW/ which represents the IPA /reʃo/, the correct transcription.

In some cases of message creation, the message to be stored contains one or more variable words. For example, one can store the message "THE MOTOR NUMBER 'n' IS OVER HEATED." The variable word "n" can be any number and need not be known during the editing time. In the message retrieval section, the variable messages will be discussed in detail.

A subroutine called VOICE was written to be used as a buffer between the user program and the I/O driver. The

user program calls VOICE with the buffer containing the data to be output and the number of data as parameters. The other parameter is the Logical Unit Number (LUN) of the DR-11C (1 in this project). Since the QIO directive of the RSX-11M operating system is issued in the VOICE subroutine, the user of the main program does not have to understand how to use the directive.

Message Retrieval

This program speaks the message which has already been stored by the message creation program. This program was written as a subroutine and may be called using the following format:

CALL TALK (FILE,VARIABLE,NUMBER)

where FILE represents the data file in which the required message was stored. VARIABLE is an array containing variable messages to fill the holes in the data file. The number of these variables is in NUMBER. These variables are in the form of addresses in the case of T.I., Digitalker or T.S.I. and phonemes in the case of Votrax. In the case of Votrax variables, many phonemes form a word. The subroutine will detect the end of a phoneme string for a word by seeing hex 8003 as one of the variables. The variable, address or phonemes, is used to fill the variable holes in respective

order. If there are not enough variable addresses or phonemes, error messages will be displayed on the screen. The flowchart of the message retrieval program is in Figure 3.3.

Programs for Separate Modules

In the DEMO program previously used to create messages in the message creation section, each word in the same sentence may come from T.I., T.S.I., Digitalker or the Votrax. There is a major drawback in this approach. The quality of voice of each module is different and to have a sentence spoken by different modules is unacceptable in many circumstances. For this reason, different programs were written to edit the speech of different modules. A program called SPEAK was written to take care of T.I. and T.S.I. and VOTRA7 for the Votrax system.

SPEAK is essentially the same as DEMO except that if it cannot find the word in the vocabulary list, it will not continue to the Votrax rules for help but will print out the error messages containing the illegal words. There are three versions of SPEAK, i.e., SPEAK1, SPEAK2, and SPEAK3. SPEAK1 uses the T.I. vocabulary while SPEAK2 and SPEAK3 use T.S.I.'s and Digitalker's respectively. The three programs are exactly the same except at the beginning of the programs they read in the vocabularies of different modules.

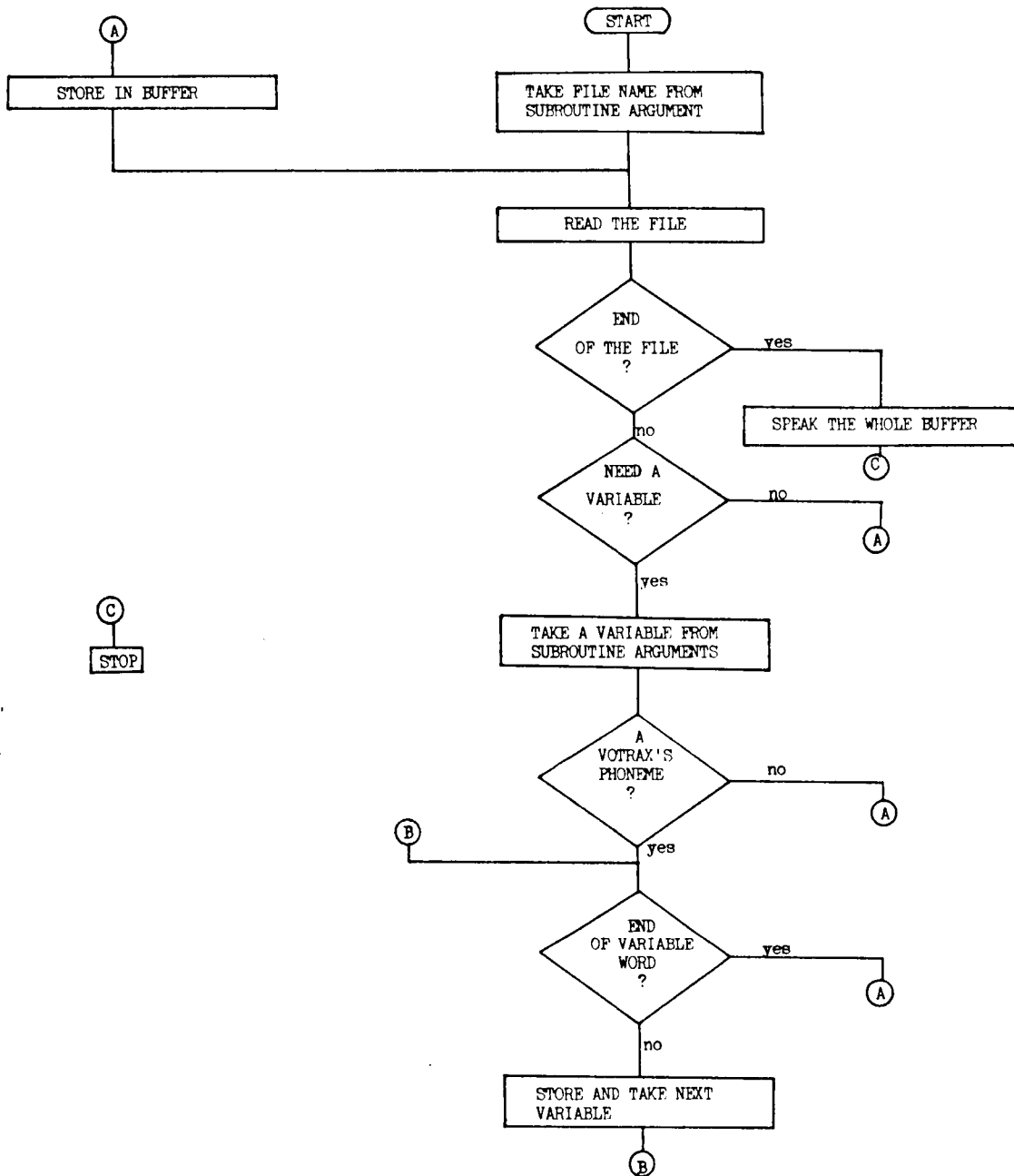


Figure 3.3. Message Retrieval Program.

In the program VOTRA7, first the Votrax rules are read into the program. The program then asks the user to type in the message and uses the same algorithm as the one used in DEMO to translate the text to the Votrax phonemes corresponding to it.

For more flexibility a program called COMPO was written. It consists simply of SPEAK and VOTRA7. First the program asks the user which module is to be used. If the answer is T.I., Digitaltalker or T.S.I. the program will go to the part resembling SPEAK and read T.I., Digitaltalker or T.S.I. vocabulary depending on which module was asked for. VOTRA7 part of the program will be reached if the user answers the question with "VOTRAX." While using one module the user can switch to another by typing 1 when the program asks for the message to be spoken. The program will again ask which module is to be used. The flowchart of the program is in Figure 3.4. Figure 3.5 shows an example of terminal display while running the program COMPO.

Many programs were written and each has different purposes. Table 3.2 helps the user to gain more information about each program and make selection easier.

I/O Driver

In the RSX-11M time sharing system, an I/O driver is needed for a peripheral device that performs an input or

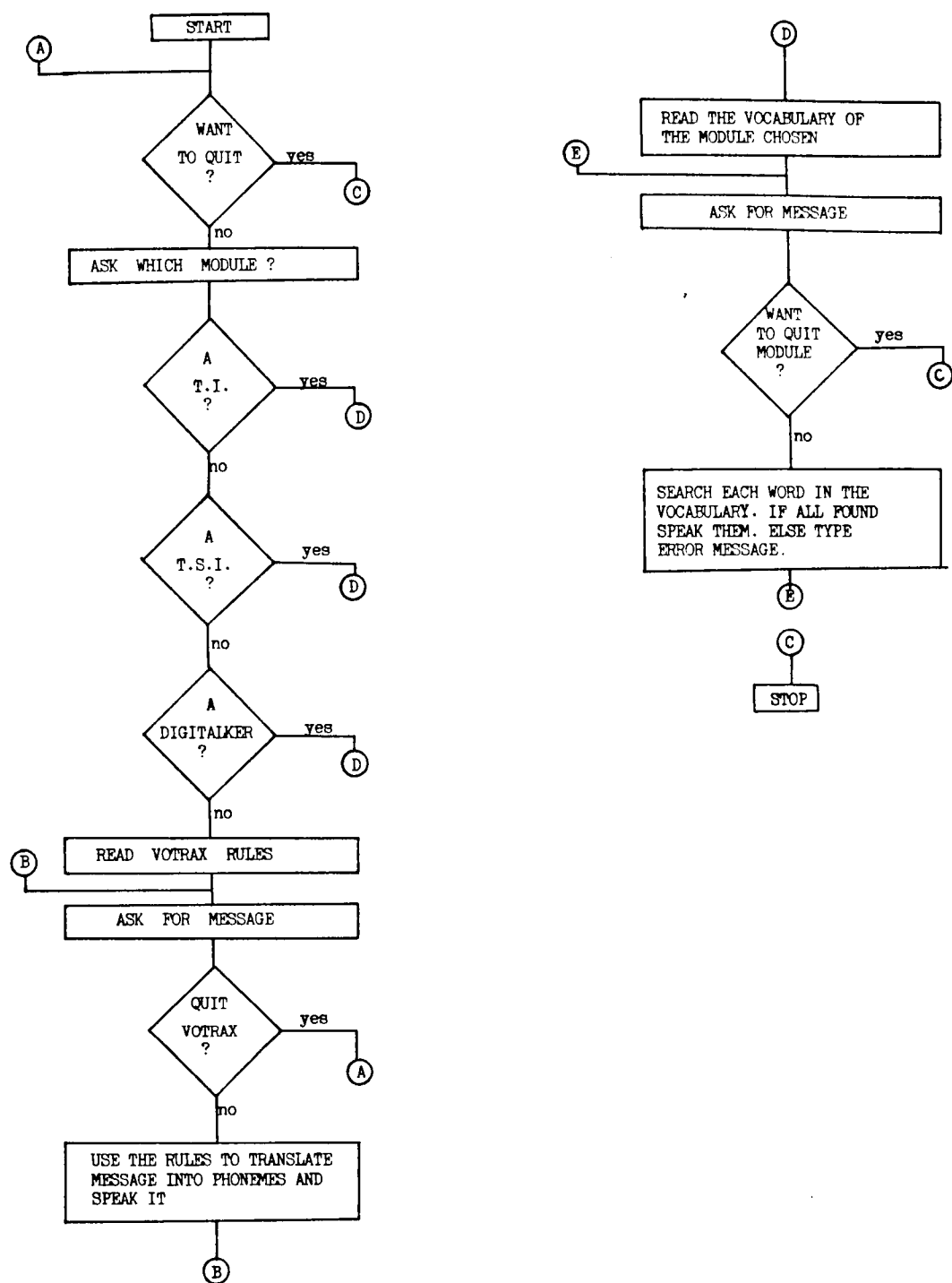


Figure 3.4. Flowchart of COMPO Program.

Which module do you want to use?
Type the following number to select:
1 For T.I.
2 For T.S.I.
3 For N.S.
4 For Votrax
5 To quit the program
?1

Input a sentence for me to speak:
To see my vocabulary hit return key.
To quit type a number 1
?Calibrate the machine
?1

Which module do you want to use?
Type the following number to select:
1 For T.I.
2 For T.S.I.
3 For N.S.
4 For Votrax
5 To quit the program
?5

TTY - STOP

Figure 3.5. Example of Terminal
Output of Program COMPO.

Table 3.2
Names and Functions of Programs

Program	Language	Function	Speed
SPEAK1	Fortran	Speak the T.I.	Real-time synthesis
SPEAK2	Fortran	Speak the T.S.I.	Real-time synthesis
VOTRA7	Fortran	Use rules to drive Votrax	Take one minute to read rules then real-time
COMPO	Fortran	Let the user select SPEAK1 SPEAK2 or VOTRA7	Takes one minute to read rules then real-time
DEMO	Fortran	Combine 3 modules with T.I. having first priority	Takes one minute to read rules then real-time
TALK	Fortran (Subroutine)	Message retrieval with variable message capability	Real-time synthesis
DRTST	Fortran	Input data manually	Real-time synthesis
VOICE	Fortran (Subroutine)	Called as buffer	Real-time
DRDRV	Macro Assembly	I/O driver called to output data through DR-11C	
DRTAB	Macro Assembly	Data base for DRDRV	

output function [6]. An I/O driver is an asynchronous process that calls and is called by the executive to service an external I/O device. A driver performs the following functions:

- Receives and services interrupts from its I/O device.
- Initiates I/O operation when requested to do so by the executive.
- Cancels I/O operation in progress.
- Performs other (device-specific) functions upon power failure and device timeout.

The block diagram of the I/O control flow is shown in Figure 3.6.

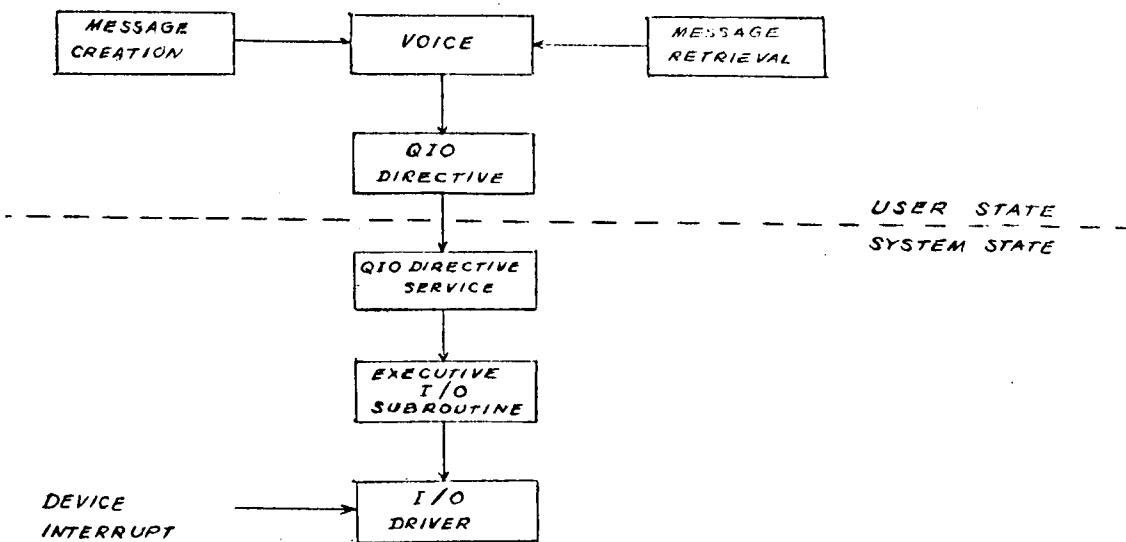


Figure 3.6. I/O Flow Control.

QIO directive is the lowest level of task I/O. Any task may issue a QIO directive. The QIO directive allows direct control over devices that are connected to a system and that have an I/O driver. The QIO directive forces all I/O requests from user tasks to go through the executive. The executive works to prevent tasks from destructively interfering with each other and with the executive itself. Throughout this project, the subroutine VOICE is the only one that issues QIO. When the program wants to speak the message, it calls VOICE with data to be spoken as parameters.

In this application, the peripheral used is the DR-11C which is a standard device so the I/O driver is also standard. The one used in this project was modified from a DECUS library program [7]. First, when the QIO directive is issued the driver is entered at the I/O initiator point. The I/O driver then waits for interrupt, thus allowing the CPU to do some other thing. When the interrupt bit is ready, the I/O driver will be entered again at the interrupt entry point to output the next word in the buffer. This process will be repeated until the whole buffer is emptied.

When the I/O driver is entered at the interrupt entry point, the data is loaded into the data buffer register (address of which is specified by the jumper on DR-11C) of the DR-11C. At this point bit 14 of the data buffer register is reset to zero, set to one, then reset to zero again. This

is the only modification from the standard I/O driver. Bit 14 serves as a strobe line to the T.S.I. and Votrax modules as mentioned before in Chapter 2. The control over the destination of this strobe is discussed in the hardware development section. Appendix B contains the listing of the I/O driver and the data structure.

Hardware Development

The speech system is interfaced to the PDP-11 through a DR-11C parallel interface board and the standard DEC cable. The interface board issues 16 bits of data to the system and contains a READY bit to monitor the status of the system. A zero to one transition at this bit causes the interrupt to the computer. Since there are four systems boxed together, special consideration must be taken to connect the READY bit to wait for the correct source of interrupt.

One difference between the T.I. and the other three modules is that it uses a 13-bit address instead of 8-bits like the others. In addition, it does not need a strobe to start speaking but uses a talk/no-talk bit for control. If this bit is zero, the T.I. module will keep speaking the word pointed to by the address until the talk bit goes to one. Every time the data is loaded into the system it goes to all four modules. Bit 14 of the DR-11C is the strobe line and the I/O driver will give the strobe every time the

data word is loaded. This strobe line is not connected to T.I. but is connected to Votrax, T.S.I. or Digitalker if any of the three is being used. The following method is used each time data is loaded:

1. If the T.I. module is to be used, clear talk bit (MSB of the DR-11C) to zero to enable it to talk. The strobe lines to T.S.I., Digitalker and Votrax are disabled thus preventing them from speaking. Connect the status bit to the interrupt line of T.I.

2. If the T.S.I. is to be used, set talk bit of T.I. to one to prevent it from speaking and direct the strobe into the T.S.I. strobe line. The T.S.I. interrupt line will be used as the source of interrupt.

3. To use the Votrax or Digitalker follow the method of T.S.I. except that the strobe will go to the selected module. Also the source of interrupt will come from the module interrupt line.

The selection of interrupt line is done with the help of a multiplexer. The four most significant bits are used as the control lines of the multiplexer. Inputs to the multiplexer are connected to the four interrupt sources while the outputs are connected to the status bit of the DR-11C. If the MSB is zero, T.I. is used as the source of interrupt. A 100, 101 and 110 combination of the two MSB's select Votrax, Digitalker and T.S.I. as the source of

interrupt respectively. The same combination is used to control a decoder which decodes the strobe. The only input of the decoder is connected to the strobe line (bit 14 of output). Outputs of the decoder go to T.S.I. and Votrax. If the MSB is zero (T.I. selected) the strobe will not go anywhere (because T.I. does not need a strobe). The block diagram of the above circuit is shown in Figure 3.7.

One purpose of the hardware development is to use the same speaker for the four modules. This can be done by using a summing operational amplifier circuit to mix the four signals together. The output of the summing circuit is amplified again to drive the 8 ohm speaker. This method can be used because the three modules do not speak at the same time. The block diagram of the voice mixing and amplification is shown in Figure 3.8. A pin diagram of this audio circuit is in Appendix D.

One reason that makes the above integration method flexible is that there are many bits of the data lines that are used by only T.I. but not by the others, i.e., bit 8 through 15. All these bits, except the strobe bit, can be used as control lines of a multiplexer and decoder. Adding more modules to the system can be done easily by replacing the multiplexer and decoder with a bigger one (more control lines). The pin diagram of the switching circuit is shown in Appendix C.

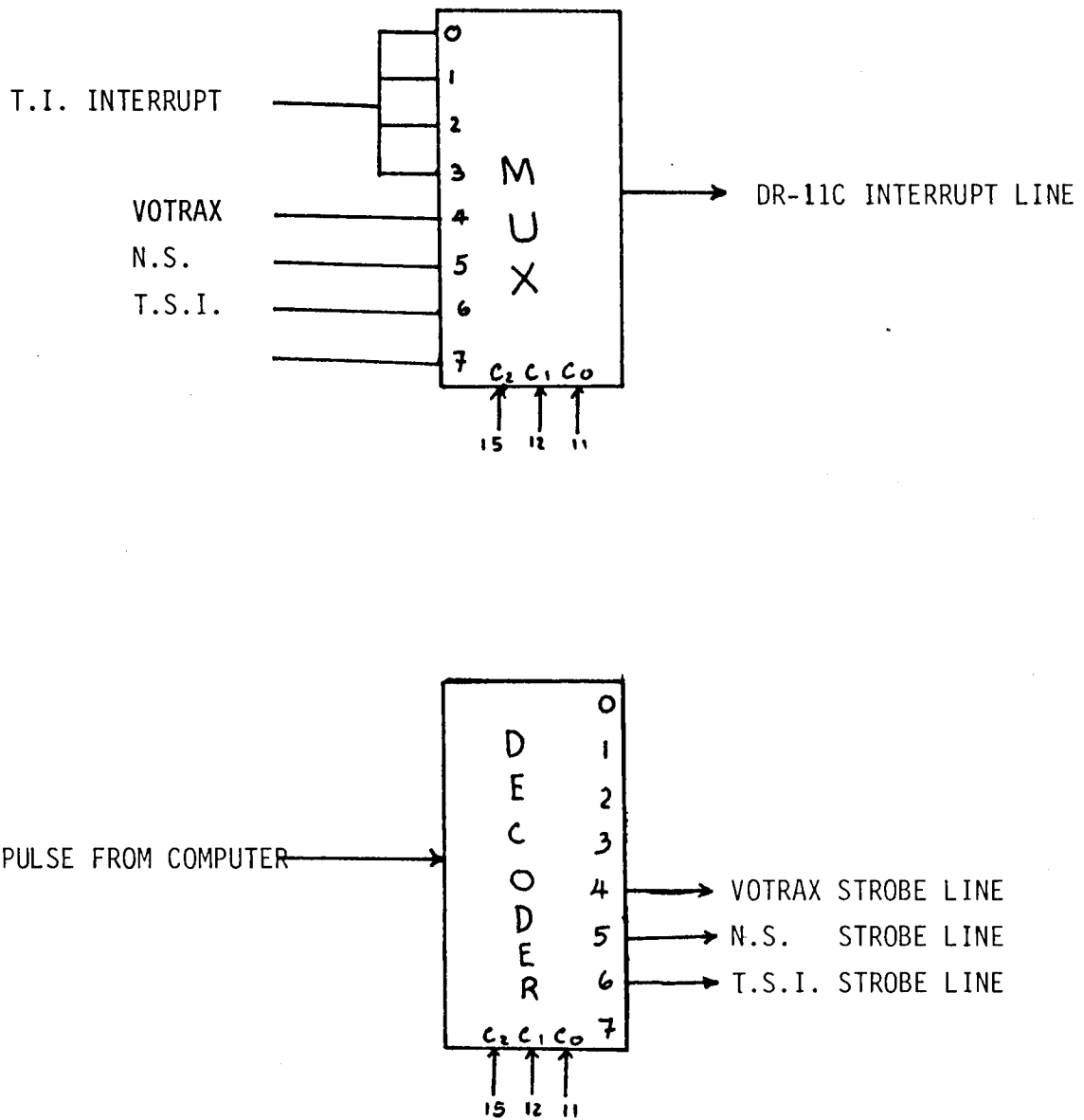


Figure 3.7. Switching Circuit Block Diagram.

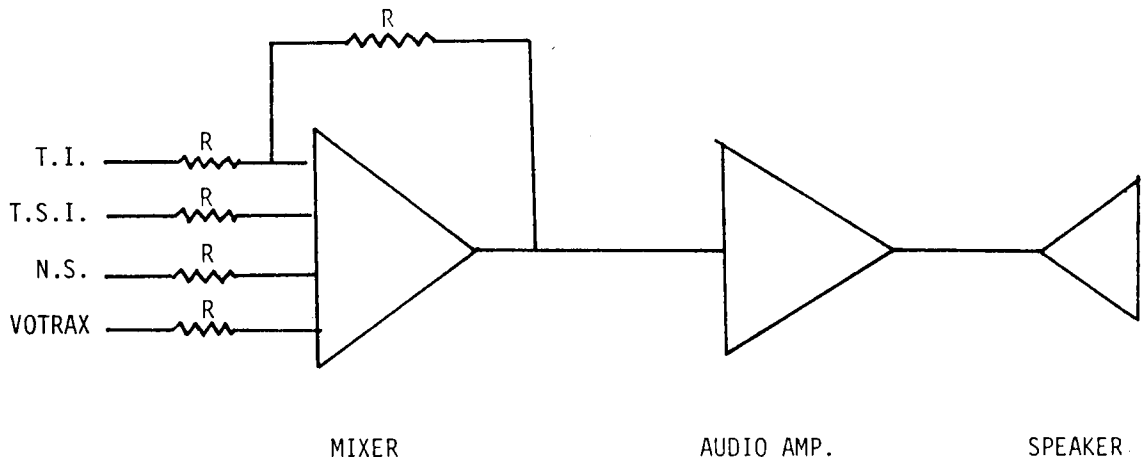


Figure 3.8. Audio Circuit.

CHAPTER 4

COMPARISON OF SPEECH SYNTHESIS MODULES

One of the main objectives of this project was to evaluate the five speech modules that were used. In this chapter, comparison is made among the speech modules in terms of quality of speech, flexibility and price. The four speech modules used to build the speech system are compared first. One by one comparison is then made between the two modules from Votrax, the Speech Pac and the Type 'N Talk. Also, the steps required to improve the text to phonemes algorithm are given. The N.R.L.'s software rules are also discussed with methods to improve them.

Quality of Speech

In most applications, the quality of speech of the module is the most important factor in deciding whether to use a system or not. Human ears are so sensitive to the quality of speech that a low quality module will not attract people to use it to build a voice response system. The best module is the module that can most nearly simulate natural human voice.

Quality comparison was done in this project with the help of a program called DRTST. This program allows the user to input a string of numbers, either addresses or

phonemes, and speak the numbers one by one until finishing the input sequence. To compare, select a word that is in T.I., T.S.I. and Digitalker and use the I.P.A. dictionary to find the phonemes corresponding to it. The addresses corresponding to the words in T.I., Digitalker, T.S.I. and phoneme string of Votrax are then entered one by one until finished. The number can be either decimal or octal or hexadecimal. Using this same method for other words, enough information was obtained to decide which module gives the best quality of speech.

It was clear to any listener that T.I. is by far the module that gives the best quality of speech. Its voice is closest to the natural human voice while the T.S.I. speaks in a robot-like manner. By joining together a series of fixed phonemes, the Votrax system gives a monotonic voice and the quality of speech was considered poorest among the four. Fortunately, the Votrax system contains two pitch control bits and the DRTST program can use the two most significant bits of the input phonemes to control these bits. With the help of these bits, the Votrax words were improved from monotonic to a little better quality speech signal. It was split among listeners between T.S.I. and Votrax (after the help of pitch control bit) as to which gives better speech. The author still believes that T.S.I. speech is better, at least more accurate than that of the Votrax system

and should be ranked third behind T.I. and Digitalker in terms of speech quality.

Flexibility

Flexibility in a voice response system makes it more attractive in most applications. The Votrax system has an advantage over the other two in that it has unlimited vocabulary. The end-user can design the custom vocabulary for his or her own application permitting significant reduction in development time. The T.S.I. vocabulary, although limited, is expandable but needs the help of the manufacturer to generate the desired custom vocabularies. This results in poor interaction between the system designer and the end-user. Still the T.S.I. gives more flexibility to the user than the T.I. whose vocabulary is limited and can be expanded only by replacing ROMS.

Price

Price is one of the most important factors in selecting an individual speech module. Its importance depends on the type of applications. The home computer hobbyist may consider price as the most important factor in buying a voice response system. In this project, however, price is considered a less important factor as compared to the quality of speech and flexibility.

The T.I. module which has the highest quality among the three costs \$1280. With a basic vocabulary of 119 words, the T.S.I. module costs \$395. The user can ask for more custom vocabulary at the price of \$125 per word. The Votrax Speech Pac module is the lowest priced system. The price with 255 words of stored vocabulary is \$275. Because of the low price and the high flexibility (unlimited vocabulary), this module is the most popular among the computer hobbyists.

Comparison between Votrax Speech Pac
and Type 'N Talk System

As both systems use the same speech processor (SC01), the quality of voice output is about the same if a similar amplifier system is used. Therefore, it is not the quality of voice but the quality of text translation algorithm which will be used to decide which module yields a better speech quality.

Ten identical sentences were typed in for both Speech Pac and Type 'N Talk. By listening to the output, both seem to have about the same accuracy in their translating algorithms. The N.R.L.'s algorithm is worse in some words but better in some others. For example, the T.N.T. speaks the sentence "I want to go to school" correctly. The Speech Pac pronounces the word "want" as "wan." On the other hand, the Speech Pac pronounces "The machine is overheated" correctly while the T.N.T. pronounces "machine" as "may-shine." To make

it pronounce the word correctly, "macheen" instead of "machine" should be typed in.

As far as the correctness of the rules is concerned, Speech Pac has the advantage over the T.N.T. because its rules are software rules and modification can always be made to give the rules a better performance. The T.N.T. text to phonemes translator is built into the hardware which makes modification by the user impossible.

Speed is a major concern in any speech synthesis system, especially the system that contains a text to speech translation capability. Since it is built into the hardware, the T.N.T. translation system gives the user a faster translation than the N.R.L.'s software rules used by the Speech Pac.

Interfacing to the T.N.T. can be done a lot easier than to the Speech Pac. It can be interfaced to any RS-232 compatible serial device. The elimination of the parallel interface board required by the other four speech modules gives the T.N.T. user an inexpensive and very convenient way to build a voice response system.

Software Rules Evaluation and Improvement

Speech synthesizers for computer voice output are most useful when not restricted to a prestored vocabulary. The simplest approach to unrestricted text-to-speech translation uses a small set of letter-to-sound rules, each specifying a

pronunciation for one or more letters in some context. The software used here is a set of 329-letter-to-sound rules translating English text into Votrax phonemes. The N.R.L. research group who developed the rules claim it can translate 90 percent of the words in an average text sample correctly [4]. In this project, the author found that the rules can hardly meet that specification. One reason is that the rules received and used in this project are not the latest version used by the N.R.L. research group to evaluate the performance. Also the group used the rules to translate from text to I.P.A. and performance was measured there. In reality, the Votrax's 64 phonemes do not cover all the I.P.A. phonemes. Even though the rules can translate text to I.P.A. with 90 percent correctness as claimed, this percentage is reduced when the software changes the I.P.A. phonemes to Votrax phonemes.

Rules were designed in such a way that they can be modified very easily. The following steps should be taken in rules development:

1. Find sampling group: Search a group of words that will probably be used in the application. The bigger the group is the better the result of rules development will be.
2. Translate: Use the software rules in program VOTRA7 to translate the word into the speech signal through the synthesizer.

3. Examine result: Listen to the translated output spoken by the synthesizer and mark mistakes. For a word that the rules do not give correct pronunciation, use the I.P.A. dictionary [5] to find the I.P.A. phonemes representing it. Input the Votrax phonemes of that word manually with the help of program DRTST. The output should be good, otherwise the developer should realize that the trouble is not in the software rules but in the synthesizer itself.

4. Classify error: Divide the mispronounced words into lists with headings like "TH problem," "AR instead of O problem," "Silent E problem." Then scan the lists to identify specific letter patterns being frequently mistranslated.

5. Modify: Modify the rules that constitute mistakes. Make sure the new rules cause less mistakes than the old ones for the total number of sampling words. For example, suppose "[EA]D=|IY|" is the original rule. This will make READY pronounced as REEDY. The rules may be replaced by the "[EA]D=|EH|" which will fix READY but will hurt a few previously correct words like BEAD. As the number of words hurt by this change are less than the number it fixed, the change is acceptable.

6. Iterate: Having produced a new version of rules, the rule developer should start the cycle over at step one with more sampling words until the overall performance of the software is considered acceptable.

For example, the word "hello" was pronounced "hella" by the original software rules. The algorithm used to translate this word is as follows:

The program put the pointer to the left of the first character "H." "H" rules are searched. The first "H" rules to match is the "[H]=[H]" which means that H followed by a vowel ("e" in this case) gives the output H. Consequently, /H/ goes into the output string, and the scan moves past H to E. The search of the E-rule turns up no match before "[E]=[EH]" which is located at the bottom of the table. As the pointer points to "L," the search in L-rule starts. "[L]=[L]" which is the default of the rule matches satisfactorily. The "L[L]=[PAO]" matches HELLO which sends "PAO" (silence, as represented by []) to the output string. Finally, as the pointer points to "O" the rules match is "[O]=[AH]".

As the scanning comes to an end, the string of phonemes output to the synthesizer is [H][EH][L][][AH] and pronounced as "hella." To fix this add the rule "[O]=[OW]". This will send [OW] instead of [AH] to the output string and the word will be pronounced "hello" correctly.

Another group of words that frequently constitute an error is the single consonant followed by "O" such as "go," "no," etc. The words are to be pronounced "gor" and "nor" respectively. By inserting the rule "[O]=[OW]" these words

will be fixed but will cause "to" to be pronounced as "tow."
Fortunately, in the T-rule there is a rule "[TO]=[T IU U]"
which will make "to" be pronounced correctly.

CHAPTER 5

SUMMARY AND CONCLUSIONS

Summary

From the comparison made in Chapter 4, it is impossible to decide which one is the best speech module. Each speech module has its own advantages and disadvantages and may be better than any other in some applications but worse in any other. The characteristics of these speech modules are summarized in general terms in Table 5.1.

Table 5.1

Comparison of Speech Modules

	T.I.	N.S.	T.S.I.	VOTRAX
Quality	Good	Good	O.K.	Poor
Flexibility	Poor	O.K.	O.K.	Excellent
Price	High	Medium	Medium	Low
Transfer rate	1000 b/s	1000 b/s	1000 b/s	100 b/s
Creation or revision of vocabulary	Can change the vocab. but cannot expand	Need help of the manufacturer	Need help of the manufacturer	Convenience

The T.I. 990/306 module is very suitable in the application area where high quality speech is required. It

was designed to operate in a noisy environment such as in a factory or assembly line. Because of its limited vocabulary, the module is not suitable in applications where a large vocabulary is required. Computer hobbyists and managers of small projects may find this module too expensive for their budget. T.I. has introduced some less expensive boards but to discuss them is beyond the scope of this project.

In applications where more flexibility and a lower price system is required, the T.S.I. S3D and N.S. Digitalker may be considered very attractive. Their vocabulary is expandable at the price of development time and money invested. These two modules may be found attractive among low budget consumers who require some flexibility and are not willing to sacrifice too much for the quality of speech.

The Votrax modules (Speech Pac and T.N.T.), with their low price and unlimited vocabulary should be found attractive in many areas. For example, in the application where operator's names have to be pronounced. As the operators may change frequently, the use of outside vendors to generate custom vocabulary will be too expensive and inconvenient. Even though the individual word quality of its speech is comparatively poor, the operator who has been trained to work with this system will find it acceptable or at least understandable. The ease in interfacing makes the T.N.T. more attractive than the Speech Pac in most areas, especially for the computer

hobbyist who cannot afford the parallel interface board and finds that the software rules used in the Speech Pac consume too much memory space of his small system.

Conclusions

A voice response system was developed by the author to be used as a tool to evaluate the performance of five speech modules, the T.I. 990/306, the T.S.I. S3D, the N.S. Digitalter, the Speech Pac and the T.N.T. The advantages and disadvantages of each of the modules in some particular application were mentioned. The development system, both the hardware and software, was designed for convenience in future expansion. The system was designed to interface to a PDP-11 computer running under RSX-11M operating system. Most of the programs were written in FORTRAN except the I/O driver which was written in assembly language (MACRO assembly). To interface to another computer, only the I/O driver has to be rewritten to accommodate the assembler of that particular computer.

Since the project was started, Votrax, T.I. and T.S.I. have introduced new, better or cheaper modules. For example, T.I. has introduced the TMS K01 whose price is less than the TM 990/306 used in this project and comparable to the T.S.I. S3D speech module. T.S.I. has introduced Speech 1000 and claims its quality of speech is as good as the T.I. 990/306

which was considered the best module in terms of speech quality. This project simply gives an example of how the modules can be compared and makes the addition of a new speech module to the system more convenient.

LIST OF REFERENCES

LIST OF REFERENCES

1. L. R. Rabiner, R. W. Schafer, Digital Processing of Speech Signals, Prentice-Hall, New Jersey, 1978.
2. The Computer Speaks, IEEE Spectrum, Vol. 16, pp. 18-25, August 1979.
3. L. R. Morris, English Text to Votrax Parameters, DECUS Programming Library, Colonel By Drive, Ottawa, Ontario, Canada, November 1978.
4. Letter-to-Sound Rules for Automatic Translation of English Text to Phonetics, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-24, pp. 446-459, December 1976.
5. J. S. Kenyon, T. A. Knott, I.P.A. Dictionary, G&C Merriam Courtsey, Springfield, Mass., 1953.
6. RSX-11M Guide to Writing I/O Driver, DEC Order No. AA-2600D-TC, Maynard, Mass. 01754, May 1979.
7. The Multi-Tasker, The Newsletter of the RSX-11/IAS Special Interest Group, Volume 12, Number 4, One Iron Way, MR2-3/E55, Marlboro, MA 01752, November 4, 1980.
8. Type 'N Talk User Manual, Votrax Division of Federal Screw Works, Troy, Michigan, 1981.
9. DT100 Digitaltalker User Manual, National Semiconductor, 2900 Semiconductor Drive, Santa Clara, Cal. 95051, December 1980.

APPENDICES

APPENDIX A

```

C                                     COMPO
C
C      MESSAGE CREATION PROGRAM
C
C      THIS PROGRAM IS USED TO STORE DATA, EITHER
C      ADDRESSES OR PHONEMES, IN A DATA FILE SO THAT IT
C      CAN BE RETRIEVED LATER.
C
C      THE PROGRAM FIRST ASK THE USER WHICH MODULE
C      HE WANT TO USE. IT THEN READ THE DATA FILE OF THE
C      MODULE. THE DATA FILES CAN BE THE PHONEME RULES
C      IN CASE OF VOTRAX OR VOCABULARY IN CASE OF THE
C      OTHER THREE MODULES. THE USER THEN ENTER THE NAME
C      OF DATA FILE HE/SHE WANTS TO EDIT. FROM THIS POINT
C      THE USER TYPE IN THE MESSAGE AND IT WILL BE SPOKEN
C      ONCE AND STORED IN THE DATA FILE.
C
C      IN THE TASK BUILDING PROCESS, THIS PROGRAM
C      SHOULD BE LINKED WITH 'VOICE','STRLIB','FN7'.
C
C      BY YUDHANA AIMTHIKUL
C      SEPTEMBER 1,1981
C
C      BYTE X(8),Y(7),Z(4),X2(7)
C      DATA X/'33','C','2','1',';','2','4','r'/
C      DATA X2/'33','C','1',';','2','4','r'/
C      DATA Y/'33','C','2','1',';','1','H'/
C      DATA Z/'33','C',6,'1'/
C      INTEGER RULDEX(30)
C      INTEGER IFLAG0,IFLAG1,IFLAG2
C      LOGICAL*1 FH(260),DR(64)
C      LOGICAL*1 STAR(2),VOWEL(8),CONSON(22),VOICED(12)
C      LOGICAL*1 FRONT(4),SUFF1(4),SUFF2(12),SUFF3(10)
C      LOGICAL*1 SIB1(8),SIB2(6),NONF1(10),NONF2(10),SPECL(12)
C      LOGICAL*1 Y1(4),Y2(2),Y3(2),Y4(3),YS(3)
C      LOGICAL*1 L(80),K(80),X1(10)
C      LOGICAL*1 A(6,400),B(8,400),C(5,400),E(10,400)
C      LOGICAL*1 AD(80),BD(80),CD(80)
C      LOGICAL*1 TEMP(10),L1(80),EXCLA(2)
C      LOGICAL*1 FILE(36),ANS(10)
C      DIMENSION INAM(6)
C      OCTAL EQUIVALENTS OF VOTRAX PHONEMES
C      DATA DR/'003','076','040','006','005','056','057','044',
C      *      '025','010','075','023','060','041','016','020',
C      *      '036','004','054','074','073','002','001','000',
C      *      '072','035','034','033','047','013','012','011',
C      *      '066','032','031','030','014','015','024','046',
C      *      '065','064','027','026','045','053','037','021',
C      *      '052','071','070','050','067','063','062','061',
C      *      '043','017','055','051','042','022','007','077'/
C
C      COMMON IX,IY,IX1,IY1
C      COMMON STAR,VOWEL,CONSON,VOICED
C      COMMON FRONT,SUFF1,SUFF2,SUFF3
C      COMMON SIB1,SIB2,NONF1,NONF2,SPECL
C      DATA INAM/2HDN,2HO:,2HPH,2HON,2H.D,2HAT/
C      LOGICAL*1 INBUF(81),WRDTBL(12,256),DLYCHR,ERROR,ALPH
C      INTEGER*2 OUTBUF(80),VALTBL(256),DELAY1,DELAY2,DELAY3,DELAY4
C      DATA DLYCHR/'-'/, MTBLSZ/256/, IWRDSZ/12/
C      DATA HOLE/'+'/
C
C      PUT DR11 ON LUN 1, READ VOCABULARY IN.

```

```

          CALL ASSIGN(1,'DR:')
999      CONTINUE
C
C  SET TOP AND BOTTOM REGION TO NORMAL.
C  THIS COMMAND WORK FOR ONLY DT100 TERMINAL.
C
          WRITE (5,513) X2
513      FORMAT ('+',7A1)
          DO 311 I=1,24
          WRITE (5,113)
113      FORMAT (' ')
311      CONTINUE
C
C  ASK THE USER WHICH MODULE TO BE USED
C
          WRITE (5,360)
360      FORMAT (1H0,'WHICH MODULE DO YOU WANT TO USE ?'/
1 1X,'1 FOR TI.'//
2 1X,'2 FOR TELESATORY.'//
3 1X,'3 FOR NATIONAL SEMICONDUCTOR.'//
4 1X,'4 FOR VOTRAX (UNLIMITED VOCABULARY).'//
5 1X,'5 TO QUIT THE PROGRAM.'//
6 1X,'?',$(
          READ (5,130) I
130      FORMAT(I1)
          IFAG=I
          IF(I.EQ.1) GO TO 125
          IF(I.EQ.2) GO TO 135
          IF(I.EQ.3) GO TO 145
          IF(I.EQ.4) GO TO 150
C
C  CHOOSE '5' SO QUIT THE PROGRAM.
C
          GO TO 9999
C
C  T.I. IS CHOSEN SO READ ITS DATA.
C
125      OPEN(UNIT=2,NAME='DATA.DAT',TYPE='OLD',READONLY)
          READ(2,100) DELAY1,DELAY2,DELAY3,DELAY4
100      FORMAT(12X,I7)
          I=1
          READ(2,200,END=70) (WRITBL(J,I),J=1,IWRDSZ),VALTBL(I)
200      FORMAT(12A1,I7)
          I=I+1
          IF (I.LE.MTBLSZ) GO TO 80
          TYPE *, 'WARNING -- VOCABULARY WILL NOT FIT IN ARRAY SPACE'
          I=MTBLSZ
          ITBLSZ=I
70      CLOSE(UNIT=2)
          GO TO 10
C
C  T.S.I. IS CHOSEN, SO READ ITS DATA.
C
135      OPEN(UNIT=2,NAME='DATA2.DAT',TYPE='OLD',READONLY)
          READ (2,101) DELAY1,DELAY2,DELAY3,DELAY4
101      FORMAT (12X,I7)
          I=1
          READ(2,2000,END=70) (WRITBL(J,I),J=1,IWRDSZ),VALTBL(I)
2000      FORMAT(12A1,I5)
C
C  SET 3 MSB'S BITS TO CONTROL MUX. AND DECODER
C
          VALTBL(I)=VALTBL(I).OR.'160000
          I=I+1
          IF (I.LE.MTBLSZ) GO TO 800

```

TYPE *, WARNING -- VOCABULARY WILL NOT FIT IN ARRAY SPACE

```

C
C N.S. IS CHOSEN, READ ITS VOCABULARY
C
145 OPEN(UNIT=2,NAME='DATA3.DAT',TYPE='OLD',READONLY)
    READ (2,102) DELAY1,DELAY2,DELAY3,DELAY4
102  FORMAT(12X,I3)
    I=1
8000 READ(2,20000,END=70)(WRDTBL(J,I),J=1,IWRDSZ),VALTBL(I)
20000 FORMAT(12A1,I3)
C
C SET 3 MSB'S TO CONTROL MUX. AND DECODER
C
    VALTBL(I)=VALTBL(I).OR.'150000
    I=I+1
    IF (I.LE.MTBLSZ) GO TO 8000
    TYPE *, 'WARNING -- VOCABULARY WILL NOT FIT IN ARRAY SPACE'
C
C GET SENTENCE. SHOW VOCABULARY IF DESIRED.
C
9    TYPE *, 'DO YOU WANT TO QUIT ?'
    READ (5,55) (ANS(I),I=1,3)
55   FORMAT (3A1)
    IF(ANS(1).EQ.'Y') GO TO 9999
C
C ASK FOR THE NAME OF FILE TO BE EDITED
C
10   CALL GETFNM('WHICH FILE DO YOU WANT TO EDIT ?','DAT',FILE,NCHR)
C
C THE NEXT 3 LINES WILL CLEAR THE FILE
C
    OPEN (UNIT=2,NAME=FILE,TYPE='OLD')
    CLOSE (UNIT=2,DISP='DELETE')
    OPEN (UNIT=2,NAME=FILE)
C
C ASK THE USER TO INPUT THE SENTENCE
C IT WILL BE SPOKEN ONCE AND THEN STORE THE
C ADDRESSES IN THE DATA FILE.
C
111  WRITE(5,300)
300  FORMAT(1H0,'INPUT A SENTENCE FOR ME TO SPEAK.//
    1 1X,'TO SEE MY VOCABULARY, TYPE A CARRIAGE RETURN ONLY.//
    2 1X,'TO QUIT TYPE A NUMBER 1.//
    3 1X,'? ',#)
    READ (5,400,END=9999) NCHAR,INBUF
400  FORMAT(Q,81A1)
    INBUF(81)=0
    IF (NCHAR.NE.0) GO TO 20
C
C THE USER JUST HIT RETURN KEY SO SHOW HIM THE VOCABULARY.
C
    TYPE *, 'MY VOCABULARY:'
    WRITE(5,500) ((WRDTBL(J,I),J=1,IWRDSZ),I=1,ITBLSZ)
500  FORMAT((1X,9(12A1,1X)))
C
C CONTROL THE SCROLLING REGION TO KEEP THE VOCABULARY ON
C THE SCREEN ALL THE TIME UNTIL QUIT THE MODULE. THE PURPOSE
C IS TO GIVE THE USER THE BEST INTERACTION. THIS MODE OF OPERATION
C IS POSSIBLE ONLY IF DT100 TERMINAL IS USED.
C
    DO 818 I=1,4
    WRITE (5,717)
717  FORMAT (' ')
818  CONTINUE
    WRITE (5,511) X

```

```

511     FORMAT ('+',3A1)
        WRITE (5,512) Y
512     FORMAT ('+',7A1)
        GO TO 111
20      CONTINUE
C
C CHECK IF HE WANTS TO QUIT (TYPED A '1')
C
        IF (NCHAR.EQ.1.AND.INBUF(1).EQ.'1') GO TO 998
C
C INITIALIZE FOR LINE SCAN.
C
        NOPTR=2
        IPTR=1
        ERROR=.FALSE.
C
C THE FIRST OUTPUT IS A DELAY AND WE OUTPUT THE DELAY OF
C THE MODULE BEING USED (SPECIFIED BY IFAG).
C
        IF(IFAG.EQ.1) OUTBUF(1)='034276
        IF(IFAG.EQ.2) OUTBUF(1)='160066
        IF(IFAG.EQ.3) OUTBUF(1)='150103
C
C CHECK FOR DELAY CHARACTER.
C
60      IF (INBUF(IPTR).NE.DLYCHR) GO TO 30
C
C A DELAY IS FOUND
C
        OUTBUF(NOPTR)=DELAY1
C
C WRITE THE DELAY CHARACTER IN DATA FILE
C
        WRITE(2,399) OUTBUF(NOPTR)
399     FORMAT(1X,'0',06)
        NOPTR=NOPTR+1
110     IPTR=IPTR+1
        GO TO 90
998     CLOSE (UNIT=2)
        GO TO 999
C
C CHECK FOR WORD AND IF ONE IS FOUND, LOOK IT UP. ELSE IGNORE THE
C CHARACTER.
C
30      IF (INBUF(IPTR).NE.'+') GO TO 31
C
C THAT '+' SPECIFIED A HOLE. WRITE IT DOWN IN THE DATA FILE.
C
        WRITE (2,299)
299     FORMAT (1X,'+')
        IPTR=IPTR+1
        GO TO 90
C
C CHECK IF THE WORD IS A NUMBER OR OTHER OPERATOR. IF IT
C IS, SKIP TO THE NEXT WORD.
C
31      IF (.NOT.ALPH(INBUF(IPTR))) GO TO 110
C
C CALL GTWORD TO FIND THE LENGTH OF THE WORD
C
        CALL GTWORD(INBUF(IPTR),LEN1)
C
C CALL LOOKUP TO SEARCH FOR THE ADDRESS OF THE WORD
C
        CALL LOOKUP(INBUF(IPTR),LEN1,WRDTBL,ITBLSZ,IWRDSZ,INDEX1)
C

```

```

C INDEX SPECIFIED THE ADDRESS OF THE WORD, IF IT IS ZERO
C THE SEARCH FAIL.
C
C     IF (INDEX1.EQ.0) GO TO 40
C
C STORE THE ADDRESS IN THE OUTPUT BUFFER
C
C     OUTBUF(NOPTR)=VALTBL(INDEX1)
C
C WRITE DOWN THE ADDRESS IN THE FILE
C
C     WRITE (2,499) OUTBUF(NOPTR)
499     FORMAT(1X,'0',06)
C     NOPTR=NOPTR+1
C     GO TO 50
C
C ILLEGAL WORD FOUND. TYPE OUT THE WORD AND SET A FLAG
C TO BE KNOWN THAT THE WHOLE SENTENCE WILL NOT BE SPOKEN
C EVEN THOUGH THE SEARCH WILL CARRY ON THROUGH OUT THE
C SENTENCE.
C
40     ERROR=.TRUE.
C     WRITE(5,600) (INBUF(I),I=IPTR,IPTR+LEN1)
600     FORMAT(1X,'MY VOCABULARY DOES NOT CONTAIN THE WORD ',80A1)
50     IPTR=IPTR+LEN1
C
C CHECK IF THE WHOLE LINE HAS BEEN SCANNED. IF IT HAS, SPEAK THE
C SENTENCE.
C
90     IF (IPTR.LE.NCHAR) GO TO 60
C
C CHECK IF AND ILLEGAL WORDS WERE FOUND. IF THE WERE,
C THE SENTENCE WILL NOT BE SPOKEN.
C
C     IF (ERROR) GO TO 1111
C
C SET THE LAST 2 WORDS IN THE BUFFER TO BE DELAY
C THEN WRITE THEM DOWN IN THE DATA FILE.
C
C     OUTBUF(NOPTR)='150103
C     WRITE (2,992) OUTBUF(NOPTR)
992     FORMAT (1X,'0',06)
C     NOPTR=NOPTR+1
C     OUTBUF(NOPTR)='100003
C     WRITE (2,993) OUTBUF(NOPTR)
993     FORMAT (1X,'0',06)
C
C THE LAST WORD IN DATA FILE IS 'S'
C
C     WRITE (2,994)
C
C
C SPEAK THE WHOLE OUTBUF BUFFER (NUMBER OF DATA IS NOPTR)
C
994     FORMAT (1X,'S')
C     CALL VOICE(OUTBUF,NOPTR,1)
1111    CLOSE (UNIT=2)
C     GO TO 9
C
C
C
C     VOTRAX *****
C
C
C THIS PART OF THE PROGRAM BASED ON THE N.R.L.'S SOFTWARE RULES
C THE SUBROUTINE SCOPY, CONCAT,SCOMP,INDEX,SUBSTR ARE PART OF

```

```

SUBROUTINE CALL SIRLIB WHICH MUST BE LINKED TO THIS PROGRAM
C DURING TASK BUILDING PROCESS.
C THE RULES MUST BE IN FILE PHON.DAT.
C
150    CONTINUE
C      VOTRAX PROGRAM: ENGLISH INPUT, VOTRAX OUTPUT
C      INITIALIZE THE VOTRAX UNIT
C      RULES FILE ON UNIT 2
C      IUNIT=2
C      OPEN (UNIT=2,NAME='PHON.DAT',TYPE='OLD',READONLY)
C      INITIALIZE THE CATEGORY COMPONENTS
C      CALL SCOPY('*',STAR)
C      CALL SCOPY('!',EXCLA)
C      CALL SCOPY('AEIOUY',VOWEL)
C      CALL SCOPY('BCDFGHJKLMNPQRSTVWXZ',CONSON)
C      CALL SCOPY('BDVGJLMNRWZ',VOICED)
C      CALL SCOPY('EIJ',FRONT)
C      CALL SCOPY('E',SUFF1)
C      CALL SCOPY('ER \ES \ED ',SUFF2)
C      CALL SCOPY('ING \ELY ',SUFF3)
C      CALL SCOPY('SCGZXJ',SIB1)
C      CALL SCOPY('CH\SH',SIB2)
C      CALL SCOPY('TSRDLZNJ',NONP1)
C      CALL SCOPY('TH\CH\SH',NONP2)
C      CALL SCOPY('!*%$%&@^+!',SPECL)
C      ASCII EQUIVALENT OF VOTRAX PHONEMES.  SAME ORDER AS ARRAY DR().
C      CALL SCOPY(' FA0 FA1 A  A1 A2 AE AE1 AH ',FH)
C      CALL CONCAT(PH,'AH1 AH2 AW  AW1 AW2 AY B  CH ',FH)
C      CALL CONCAT(PH,'D  DT E  E1 EH EH1 EH2 EH3 ',FH)
C      CALL CONCAT(PH,'ER F  G  H  I  I1 I2 I3 ',FH)
C      CALL CONCAT(PH,'IU J  K  L  M  N  NG O  ',FH)
C      CALL CONCAT(PH,'O1 O2 OO 001 F  R  S  SH ',FH)
C      CALL CONCAT(PH,'T  TH THV U  U1 UH UH1 UH2 ',FH)
C      CALL CONCAT(PH,'UH3 V  W  Y  Y1 Z  ZH ',FH)
C
C      I=1
C      DEFINE SYMBOLS USED AS DELIMITERS IN RULE TABLE
C      CALL SCOPY('[',Y2)
C      CALL SCOPY(']',Y3)
C      CALL SCOPY('=',Y4)
C      CALL SCOPY('[\ ',Y5)
C      CALL SCOPY('RULE.VS6',X1)
C      READ THE RULES *****
2140    CONTINUE
C      READ A LINE FROM PHON.DAT
C      READ(2,550) L
550    FORMAT(80A1)
C      L(80)=0
C      ' ' TERMINATES RULES
C      CALL SCOPY(L,TEMP,1)
C      CALL SCOMP(TEMP,EXCLA,ITEST)
C      IF(ITEST.EQ.0) GO TO 2151
C      '*' PRECEDES EACH SET OF RULES
C      CALL SCOMP(TEMP,STAR,ITEST)
C      IF(ITEST.EQ.0) IX=1
C      IF(ITEST.EQ.0) GO TO 15
C      CALL INDEX(L,X1,1,KI)
C      IX=3
C      IF(KI.EQ.0) IX=2
15    GO TO (10010,11000,11050), IX
2150    GO TO 2140
2151    GO TO 2160
C      SUBROUTINES FOR PARSING OF RULES
C      DUMMY SUBROUTINE
10010    GO TO 2150
C      PARSE

```

```

11000 CONTINUE
11010 CALL SCOPY(1H',Y1)
      CALL INDEX(L,'',1,ITEST)
C      IF * PRESENT, IT BECOMES THE DELIMITER INSTEAD OF '
      IF(ITEST.GT.0) CALL SCOPY('',Y1)
C      FIND POSITION OF DELIMITER SYMBOLS
      CALL INDEX(L,Y1,1,IY1)
      CALL INDEX(L,Y2,IY1,IY2)
      CALL INDEX(L,Y3,IY2,IY3)
      CALL INDEX(L,Y4,IY3,IY4)
      CALL INDEX(L,Y5,IY4,IY5)
C      READ LEFT RULE, BODY, RIGHT RULE, AND VOTRAX EQUIVALENT
11030 CALL SUBSTR(L,A(1,I),IY1+1,IY2-1-IY1)
      CALL SUBSTR(L,B(1,I),IY2+1,IY3-1-IY2)
      CALL SUBSTR(L,C(1,I),IY3+1,IY4-1-IY3)
      CALL SUBSTR(L,K,IY4+2,IY5-2-IY4)
C      STRING K CONTAINS VOTRAX INPUT FOR GIVEN RULE
      ISTRL=LEN(K)
      IS=1
      NCOUNT=1
C      FIND A BLANK
11031 CALL INDEX(K,' ',IS,ISTEST)
      IF(ISTEST.GT.0) GO TO 11032
C      NO MORE BLANKS
      ISTRL=ISTRL+1
C      COPY PHONEME INTO TEMP
11032 CALL SUBSTR(K,TEMP,IS,ISTEST-IS)
C      FIND START OF PHONEME IN MASTER ARRAY, 'PH'
C      FIRST PAD WITH LEADING, LAGGING BLANKS
      CALL CONCAT(' ',TEMP,L1)
      CALL CONCAT(L1,' ',TEMP)
      CALL INDEX(PH,TEMP,1,IT2)
      INT=IT2/4+1
C      GET OCTAL VOTRAX INPUT INTO 'E' ARRAY
      E(NCOUNT+1,I)=DR(INT)
      IS=ISTEST+1
      IF(ISTEST.GE.ISTRL) GO TO 11033
      NCOUNT=NCOUNT+1
      GO TO 11031
C      E(1,I)=# BYTES OUT TO VOTRAX FOR GIVEN RULE
11033 E(1,I)=NCOUNT
      IS=E(1,I)+1
      I=I+1
      GO TO 2150
C      PUNCTUATION AND NUMBER RULES
11050 CALL SUBSTR(L,TEMP,IX-1,1)
      IA=TEMP(1)-63
      CALL INDEX(L,'PUNCT',1,ITEST)
      IF(ITEST.GT.0) IA=1
      CALL INDEX(L,'NUMBER',1,ITEST)
      IF(ITEST.GT.0) IA=28
      RULDEX(IA)=I
      GO TO 2150
2160 RULDEX(29)=I
5000 CONTINUE
      CLOSE (UNIT=2)
5101 NO=1
C      ENGLISH TO VOTRAX*****
C
C      THIS SECTION ASK FOR THE SENTENCE. THEN USE THE RULES TO
C      TRANSLATE IT TO VOTRAX PHONEMES.
C      FIRST, THE FILE NAME TO BE EDITTED IS ASKED FOR.
C
5100 CALL GETFNM ('WHICH FILE DO YOU WANT TO EDIT?', 'DAT', FILE, NCHR)
C
C      THE NEXT 3 LINES CLEAR THE DATA FILES

```

```

C      OPEN (UNIT=2,NAME=FILE,TYPE='OLD')
      CLOSE (UNIT=2,DISP='DELETE')
      OPEN (UNIT=2,NAME=FILE)
C
C      ASK FOR THE SENTENCE
C
      WRITE(5,350)
350    FORMAT(1H0,'INPUT A SENTENCE FOR ME TO SPEAK:/'
      1 1X' TO QUIT TYPE A NUMBER 1'/
      2 1X '?'',%)
      NCHAR=0
C
C      GET A STRING FROM KEY BOARD; ALSO TRIM TRAILING BLANKS
C
      READ(5,450,END=9999) NCHAR,(L(I),I=1,NCHAR)
450    FORMAT(Q,80A1)
      IF (NCHAR.EQ.1.AND.L(1).EQ.'1') GO TO 999
      L(NCHAR+1)=0
C      ADD LEADING, LAGGING BLANK
      CALL CONCAT(' ',L,K,80)
      CALL CONCAT(K,' ',L,80)
C      POINTER POINTS TO FIRST (NON-BLANK) CHARACTER IN INPUT STRING
      IPOINT=2
C      LENGTH OF INPUT STRING
      ISTRLN=LEN(L)
C      DO FOLLOWING WHILE IPOINT < ISTRLN
5120   IF(IPOINT.GE.ISTRLN) GO TO 516
      IF (L(IPOINT).NE.'+') GO TO 5121
C
C      '+' REPRESENT A HOLE, IF FOUND WRITE IT DOWN.
C
      WRITE (2,5122)
5122   FORMAT (1X,'+')
      IPOINT=IPOINT+1
      GO TO 5120
5121   CALL SUBSTR(L,L1,IPOINT,1)
      IRULE=1
      ITEST=L1(1)
      IF(ITEST.GE.48.AND.ITEST.LE.57) IRULE=28
      IF(ITEST.GE.65.AND.ITEST.LE.90) IRULE=ITEST-63
5130   GO TO 12000
5131   GO TO 5120
C      END OF THIS STRING; GO GET ANOTHER
516    GO TO 11111
C      START OF MAIN PATTERN MATCH, TRANSLATION XXXXXXXXXXXXXXXXXXXXXXXXXXXX
12000  ISTR=RULDEX(IRULE)
      ISTOP=RULDEX(IRULE+1)-1
C      CHECK ALL RULES WHOSE 1'ST BODY CHARACTER MATCHES POINTER CHARACTER
      DO 12030 I= ISTR,ISTOP
C      SEGMENT INPUT STRING INTO LEFT STRING, MIDDLE STRING AND RIGHT STRING
C      LENGTH OF MIDDLE STRING = IB = LENGTH OF BODY OF RULE BEING TESTED
      IB=LEN(B(1,I))
      CALL SUBSTR(L,AD,1, IPOINT-1)
      CALL SUBSTR(L,BD,IPOINT, IB)
      CALL SUBSTR(L,CD,IPOINT+IB,LEN(L)+1-IPOINT-IB)
C
C      "BODY" OF STRING = BODY OF RULE? IF NOT, FORGET IT!
      CALL SCOMP(BD,B(1,I),IFLAGO)
      IF(IFLAGO.NE.0) GO TO 12030
C
C      CHECK PREFIX RULE FOR RELEVANT BODY
      CALL FNPREF(A(1,I),AD)
      IF(IX1.LE.0) GO TO 1110
      CALL SUBSTR(A(1,I),L1,1)
10001  CALL SUBSTR(L1,L1,1,IX-IX1)

```

```

10002 CALL SUBSTR(AD,1,17-1Y1)
105  CALL FNFRE(L1,AD)
    IF(IX1.GT.0) GO TO 10001
C    NO MATCH (OR NO MORE RULES) FOR PREFIX? FORGET IT!
1110  IF(IX1.NE.-1) GO TO 12030
C
C    CHECK SUFFIX RULE FOR RELEVANT BODY
    CALL FNSUF(C(1,I),CD)
    IF(IX1.LE.0) GO TO 2220
    CALL SUBSTR(C(1,I),L1,1)
20001  CALL SUBSTR(L1,L1,IX1+1,IX-IX1)
20002  CALL SUBSTR(CD,CD,IY1+1,IY-IY1)
205   CALL FNSUF(L1,CD)
    IF(IX1.GT.0) GO TO 20001
C    NO MATCH (OR NO MORE RULES) FOR SUFFIX? FORGET IT!
2220  IF(IX1.NE.-1) GO TO 12030
C
C    PREFIX, SUFFIX AND BODY ALL MATCH! SUCCESS!!
3330  LIM=E(1,I)
C    OUTPUT OCTAL EQUIVALENT OF PHONEMES TO VOTRAX
    DO 3333 JJ=1,LIM
C
C    STORE THE PHONEME STRING IN OUTPUT BUFFER 'OUTBUF'.
C
    OUTBUF(NO)=E(JJ+1,I)
C
C    SET THE MSB. TO 1 TO CONTROL MUX. AND DECODER.
C
    OUTBUF(NO)=OUTBUF(NO).OR.'100000
C
C    WRITE THE PHONEME IN DATA FILE.
C
    WRITE (2,3332) OUTBUF(NO)
3332  FORMAT (1X,'0',06)
    NO=NO+1
3333  CONTINUE
C
C    BUMP POINTER BY LENGTH OF RULE JUST FOUND, AND KEEP GOING
    IPOINT=IPOINT+IB
    GO TO 5131
12030 CONTINUE
C
C    MESSAGE FOR UNKNOWN TEXT
    WRITE(5,12031)IPOINT
12031  FORMAT(1X,'NO RULE FOUND AT CHAR' I7)
C    SKIP REST OF STRING WHEN UNKNOWN TEXT ENCOUNTERED
    IPOINT=ISTRLN
    GO TO 5131
C
C    THE LAST TWO OUTPUT ARE DELAY ADDRESS
C
11111  OUTBUF(NO)='150103
    WRITE (2,929) OUTBUF(NO)
929    FORMAT(1X,'0',06)
    OUTBUF(NO+1)='100003
    WRITE (2,939) OUTBUF(NO+1)
939    FORMAT(1X,'0',06)
    WRITE (2,339)
C
C    THE END OF FILE IS INDICATED BY 'S'
C
339    FORMAT (1X,'S')
C
C    SPEAK THE BUFFER OUTBUF ONCE
C
    CALL VOICE(OUTBUF,NO+1,1)

```

```

        CLOSE (UNIT=2)
        TYPE*, 'DO YOU WANT TO QUIT ?'
        READ (5,66) (ANS(I), I=1,3)
66      FORMAT (3A1)
        IF (ANS(1).NE.'Y') GO TO 5101
9999    CALL EXIT
        END
        LOGICAL*1 FUNCTION ALPH(CHAR)
C
C   THIS FUNCTION RETURN A TRUE IF 'CHAR' IS A LETTER OF THE ALPHABET
C
        LOGICAL*1 CHAR
        ALPH=.FALSE.
        IF (CHAR.LT.'A') RETURN
        IF (CHAR.GT.'Z') RETURN
        ALPH=.TRUE.
        RETURN
        END
        SUBROUTINE GTWORD(BUF,LEN)
C
C   THIS SUBROUTINE RETURNS THE NUMBER OF CONSECUTIVE ALPHA CHARACTERS THAT
C   ARE IN 'BUF'.
C
        LOGICAL*1 BUF(80),ALPH
        LEN=0
10      IF (.NOT.ALPH(BUF(LEN+1))) RETURN
        LEN=LEN+1
        GO TO 10
        END
        SUBROUTINE LOOKUP(WORD,LEN,TABLE,ITBSZ,IWRDSZ,INDEX)
C
C   THIS SUBROUTINE LOOKS UP A WORD OF LENGTH 'LEN' IN THE TABLE 'TABLE'.
C   'ITBSZ' IS THE NUMBER OF VALID ENTRIES IN THE TABLE AND 'IWRDSZ' IS
C   THE SIZE OF WORDS IN THE TABLE. IF A WORD IS FOUND IN THE TABLE, 'INDEX'
C   IS SET TO THE LOCATION IN THE TABLE OF THE WORD. IF THE SEARCH FAILS,
C   'INDEX' IS SET TO ZERO. THE SUBROUTINE USES A BINARY SEARCH, SO THE
C   TABLE MUST BE IN ALPHABETICAL ORDER.
C
        LOGICAL*1 WORD(LEN),TABLE(IWRDSZ,ITBSZ),WDGTR
        ISLEN=ITBSZ
        ISTBEG=1
10      ISINDX=(ISLEN+1)/2+ISTBEG-1
C
C   DETERMINE EQUALITY OF WORD TO TABLE VALUE. RETURN INDEX VALUE
C   IF EQUAL. ELSE SET WDGTR FLAG INDICATING WHETHER WORD IS GREATER
C   THAN TABLE ENTRY.
C
        WDGTR=.TRUE.
        DO 20 I=1,IWRDSZ
            IF (I.GT.LEN) GO TO 30
            IF (WORD(I).NE.TABLE(I,ISINDX)) GO TO 40
            GO TO 20
30      IF (TABLE(I,ISINDX).NE.' ') GO TO 50
20      CONTINUE
        INDEX=ISINDX
        RETURN
40      IF (WORD(I).GT.TABLE(I,ISINDX)) GO TO 60
50      WDGTR=.FALSE.
C
C   CHECK FOR OUR BASE CASES THAT INDICATE FAILURE. IF THE CASES
C   DO NOT APPLY, SEARCH PART OF TABLE THAT WORD SHOULD BE IN.
C
60      IF (ISLEN.NE.1) GO TO 80
70      INDEX=0
        RETURN
80      IF (WDGTR) GO TO 90

```

```
      IF (ISLEN.EQ.2) GO TO 70
C  SEARCH BOTTOM OF TABLE
      ISTBEG=ISTBEG
      ISLEN=ISINDX-ISTBEG
      GO TO 10
C  SEARCH TOP OF TABLE
90     ISLEN=ISTBEG+ISLEN-1-ISINDX
      ISTBEG=ISINDX+1
      GO TO 10
      END
```

```

C
C
C          TALK
C
C          MESSAGE RETRIEVAL PROGRAM
C
C  THIS SUBROUTINE, WHEN CALLED, SPEAK THE WHOLE
C  SPECIFIED DATA FILE. THE DATA FILE SHOULD BE CREATED
C  THROUGH PROGRAM 'COMPO'. IT HAS THE CAPABILITY TO
C  SPEAK THE VARIABLE MESSAGE (HOLE) AS SPECIFIED BY
C  '+' IN THE DATA FILE.
C
C  TO CALL USE THE FOLLOWING FORMAT
C
C          CALL TALK(FILE,IVAR,NUMBER)
C
C  WHERE 'FILE' INDICATE THE DATA FILE, 'IVAR' IS A LIST
C  OF VARIABLE MESSAGE TO FILL THE HOLE AND 'NUMBER' IS
C  THE NUMBER OF THOSE VARIABLES.
C
C
C          SUBROUTINE TALK(FILE,IVAR,NUMBER)
C          LOGICAL*1 ALPH,FILE(36)
C          DIMENSION ALPH(10),IARRAY(100),IVAR(10)
C
C  INITIALIZE THE VARIABLE COUNT AND OUTPUT COUNT
C
C          NVAR=1
C          NWRDS=1
C
C  ASSIGN LUN 1 TO DRDRV (THE I/O DRIVER)
C
C          CALL ASSIGN(1,'DR:')
C          NOWRDS=NWRDS
C          NWRDS=1
C
C  OPEN DATA FILE AND ASSIGN LUN 2 TO IT
C
C          OPEN(UNIT=2,NAME=FILE,TYPE='OLD')
C
C  READ THE DATA FILE
C
C          10      READ(2,100) NCHR,(ALPH(I),I=1,NCHR)
C          100     FORMAT(Q,1X,10A1)
C
C  THE NUMBER OF CHARACTER IN NCHR IS 1 ABOVE THE REAL
C  AMOUNT BECAUSE OF A SPACE AT THE BEGINNING (1X)
C
C          NCHR=NCHR-1
C
C  'S' INDICATE END OF FILE AND '+' INDICATE A HOLE
C
C          IF (ALPH(1).EQ.'S') GO TO 99
C          IF (ALPH(1).EQ.'+') GO TO 999
C
C  CHECK WHETHER IT IS AN OCTAL OR DECIMAL OR HEX
C
C          ITYPE=1
C          IF (ALPH(1).EQ.'D') ITYPE=2
C          IF (ALPH(1).EQ.'H') ITYPE=3
C          IOFFS=1
C          IF (ALPH(1).EQ.'O'.OR.TYPE.NE.1) IOFFS=2
C
C  CALL SUBROUTINE CONVERT TO CONVERT THE DATA TO INTEGER.

```

```

C THE RESULT WILL BE IN INUM.
C IF CONVERSION FAIL, IERR WILL BE NEGATIVE.
C
      CALL CNVRT(INUM,ALPH(IOFFS),NCHR-IOFFS+1,ITYPE,IERR)
      IF (IERR.LT.0) GO TO 20
C
C STORE THE NUMBER IN OUTPUT BUFFER.
      IARRAY(NWRDS)=INUM
      NWRDS=NWRDS+1
C
C GO AND READ NEXT DATA FROM THE FILE
C
      GO TO 21
C
C THE DATA IN THE FILE IS A HOLE. ONE OR MORE
C VARIABLE ARE NEEDED TO FILL THAT HOLE.
C A T.I., N.S., AND T.S.I., WILL NEED JUST
C ONE VARIABLE TO FILL THE HOLE. VOTRAX NEED
C MORE THAN ONE VARIABLE TO FILL A HOLE AND
C THE STRING OF VOTRAX VARIABLES FOR A HOLE
C IS INDICATED BY '100003.
C
999      IARRAY(NWRDS)=IVAR(NVAR)
C
C POINT TO NEXT OUTPUT AND NEXT VARIABLE
C
      NWRDS=NWRDS+1
      NVAR=NVAR+1
C
C CHECK IF THE VARIABLE IS VOTRAX.
C
      IF(IVAR(NVAR-1).GE.'150000) GO TO 10
      IF(IVAR(NVAR-1).LT.'100000) GO TO 10
C
C IT IS A VOTRAX VARIABLE. CHECK IF IT THE END
C OF THE STRING.
C
      IF(IVAR(NVAR-1).EQ.'100003)GO TO 10
C
C NOT END YET. GO GET NEXT VARIABLE
C
      GO TO 999
C
C ILLEGAL WORD IS FOUND
C
20      TYPE *, 'ILLEGAL FORMAT FOR NUMBER. IERR=',IERR
21      IF (NWRDS.LT.100) GO TO 10
      TYPE *, 'ARRAY IS FULL.'
99      IARRAY(NWRDS)='100003
C
C CALL VOICE TO SPEAK THE WHOLE BUFFER
C
      CALL VOICE(IARRAY,NWRDS,1)
      CLOSE (UNIT=2)
      STOP
      END
C
C
C
      SUBROUTINE CNVRT(INUM,ALPH,NCHR,ITYP,IERR)
C THIS SUBROUTINE CONVERT THE NUMBER IN FILE TO INTEGER
C
      LOGICAL*1 ALPH(NCHR)
      IERR=0
      INUM=0

```

```

      IF (NCHR.EQ.0) RETURN
      IERR=-2
      GO TO (10,20,30) ITYP
      IERR=-1
      RETURN
C
C   THE NUMBER IS OCTAL
C
10      MAXVAL=7
      IF(NCHR.GT.6) RETURN
      GO TO 40
C
C   THE NUMBER IS DECIMAL.
C
20      MAXVAL=9
      IF(NCHR.GT.6) RETURN
      GO TO 40
C
C   THE NUMBER IS HEX.
C
30      MAXVAL=15
      IF (NCHR.GT.4) RETURN
40      VAL =0
      IERR=-3
      DO 50 I=1,NCHR
        IF ((ALPH(I).LT.'0'.OR.ALPH(I).GT.'9').AND.
          1  (ALPH(I).LT.'A'.OR.ALPH(I).GT.'F')) RETURN
        DIGVAL=ALPH(I)-'0'
        IF (ALPH(I).GE.'A'.AND.ALPH(I).LE.'F') DIGVAL=10+(ALPH(I)-'A')
        IF (DIGVAL.GT.FLOAT(MAXVAL)) RETURN
        VAL=VAL*(MAXVAL+1)+DIGVAL
50      CONTINUE
      IF (VAL.GT.2.0**15-1.0) VAL = VAL-2.0**16
      INUM=IFIX(VAL)
      IERR=0
      RETURN
      END

```

```

C
C
C          DRTST
C
C  THIS PROGRAM LET THE USER INPUT NUMBER
C  MANUALLY THROUGH THE KEYBOARD. THE NUMBER
C  CAN BE OCTAL, DECIMAL OR HEX. THE DATA
C  MUST BE PRECEDED BY O,D OR H DEPENDING
C  ON WHAT UNIT THEY ARE. THE EXAMPLES ARE
C  0100003, D019228, H0A0A. WHEN FINISH ONE
C  DATA, TYPE RETURN KEY AND INPUT THE NEXT
C  ONE. THE WHOLE MESSAGE WILL BE SPOKEN
C  WHEN RETURN KEY IS HIT TWICE IN A ROW.
C
C          LOGICAL*1 ALPH
C          DIMENSION ALPH(10),IARRAY(100)
C          NWRDS=1
C
C  ASSIGN LUN 1 TO DRDRV (I/O DRIVER)
C
C          CALL ASSIGN(1,'DR:')
1  TYPE *, 'INPUT NUMBERS ONE BY ONE THEN INPUT <CR> ONLY.'
C          NOWRDS=NWRDS
C          NWRDS=1
C
C  READ THE KEY BOARD
C
10  ACCEPT 100,NCHR,(ALPH(I),I=1,NCHR)
100  FORMAT(Q,10A1)
C
C  CHECK IF THE USER INPUT A RETURN KEY (WHICH MEAN
C  HE WANT TO SPEAK THE WHOLE MESSAGE)
C
C          IF (NCHR.EQ.0) GO TO 99
C          ITYPE=1
C
C  CHECK THE TYPE OF INPUT (DECIMAL, OCTAL OR HEX)
C
C          IF (ALPH(1).EQ.'D') ITYPE=2
C          IF (ALPH(1).EQ.'H') ITYPE=3
C          IOFFS=1
C          IF (ALPH(1).EQ.'O'.OR.ITYPE.NE.1) IOFFS=2
C
C  CALL SUBROUTINE CONVERT TO CHANGE THE NUMBER TO INTEGER
C
C          CALL CNVRT(INUM,ALPH(IOFFS),NCHR-IOFFS+1,ITYPE,IERR)
C
C  CHECK IF CONVERSION SUCCEED
C
C          IF (IERR.LT.0) GO TO 20
C
C  STORE THE CONVERTED DATA IN OUTPUT BUFFER
C
C          IARRAY(NWRDS)=INUM
C          NWRDS=NWRDS+1
C          GO TO 21
19  NWRDS=NOWRDS
C          GO TO 21
C
C  ILLEGAL FORMAT NUMBER FOUND
C
20  TYPE *, 'ILLEGAL FORMAT FOR NUMBER.  IERR=',IERR

```

```

21      IF (NWRDS.LT.100) GO TO 10
        TYPE *, 'ARRAY IS FULL.'
C
C      CALL VOICE TO SPEAK THE WHOLE PROGRAM
C
99      CALL VOICE(IARRAY,NWRDS-1,1)
C
C      CHECK IF THE USER WANT TO OUTPUT ANOTHER ARRAY
C
        TYPE *, 'OUTPUT ANOTHER ARRAY?'
        ACCEPT 100,NCHR,ALPH(1)
C
C      IF ANSWER IS '0' SPEAK THE OLD BUFFER
C
        IF (ALPH(1).EQ.'0') GO TO 99
        IF (NCHR.NE.0.AND.ALPH(1).NE.'N') GO TO 1
        STOP
        END
C
C
C      THIS SUBROUTINE CONVERT THE OCTAL,DECIMAL HEX TO
C      INTEGER TO BE OUTPUT.
C
C
        SUBROUTINE CNVRT(INUM,ALPH,NCHR,ITYP,IERR)
        LOGICAL*1 ALPH(NCHR)
        IERR=0
        INUM=0
        IF (NCHR.EQ.0) RETURN
        IERR=-2
        GO TO (10,20,30) ITPF
        IERR=-1
        RETURN
C
C      OCTAL NUMBER
C
10      MAXVAL=7
        IF(NCHR.GT.6) RETURN
        GO TO 40
C
C      DECIMAL NUMBER IS FOUND
C
20      MAXVAL=9
        IF(NCHR.GT.6) RETURN
        GO TO 40
C
C      HEXADECEMAL
C
30      MAXVAL=15
        IF (NCHR.GT.4) RETURN
40      VAL =0
        IERR=-3
C
C      START CONVERSION ROUTINE
C
        DO 50 I=1,NCHR
            IF ((ALPH(I).LT.'0'.OR.ALPH(I).GT.'9').AND.
1          (ALPH(I).LT.'A'.OR.ALPH(I).GT.'F')) RETURN
            DIGVAL=ALPH(I)-'0'
            IF (ALPH(I).GE.'A'.AND.ALPH(I).LE.'F') DIGVAL=10+(ALPH(I)-'A')
            IF (DIGVAL.GT.FLOAT(MAXVAL)) RETURN
            VAL=VAL*(MAXVAL+1)+DIGVAL
50      CONTINUE
        IF (VAL.GT.2.0**15-1.0) VAL = VAL-2.0**16
        INUM=IFIX(VAL)
        IERR=0
        RETURN
        END

```

```

      SUBROUTINE VOICE(IBUF,INUM,LUN)
C     THIS VERSION OF VOICE WAS WRITTEN FOR RSX-11M.
C     IT ALWAYS WAITS FOR THE TRANSFER TO COMPLETE BEFORE RETURNING.
C     FREELY ADAPTED FROM VIDEO BY ROBERT MEYER.
      LOGICAL*1 ISB1(2)
      INTEGER*2 QIOBUF(6),IDSW,IOSTAT(2),IOWVB
      INTEGER*2 INUM,LUN,IBUF(INUM)
      EQUIVALENCE (IOSTAT(1),ISB1(1))
      DATA IOWVB/'11000/
      CALL GETADR(QIOBUF(1),IBUF(1))
      QIOBUF(2) = INUM * 2
      CALL WTRIO(IOWVB,LUN,LUN,,IOSTAT,QIOBUF,IDSW)
      IF(IDSW.LT.1) GOTO 9990
      IF(ISB1(1).GE.1) RETURN
      WRITE(5,1) IOSTAT
1     FORMAT('0***** DR IO ERROR *****',/,
1 ' IOSTAT(1)=' ,06,5X,'IOSTAT(2)=' ,06)
      RETURN
9990  WRITE(5,2) IDSW
2     FORMAT('0***** DR DSW ERROR *****',/,
1 ' IDSW=' ,06)
      RETURN
      END

```

APPENDIX B

```

D$$R11=1
LD$DR=0
    .TITLE DRDRV
    .IDENT /01/

; VERSION 01
;
; TERRY P. MEDLIN 7-NOV-78
;
; MODIFIED BY: Y. AIMTHIKUL
;
; DR11C-DRIVER FOR THE 8085 INTERFACE WITH REMOTE LAB
;
; MODIFIED TO BE DR-11C DRIVER FOR SPEECH MODULES INTERFACE
; MACRO LIBRARYS CALLS
;

    .MCALL ABODF$,HWDDF$,PKTDF$,UCBDF$,SCBDF$
    ABODF$      ;DEFINE TASK ABORT CODES
    HWDDF$      ;DEFINE HARDWARE REGISTER SYMBOLS
    PKTDF$      ;DEFINE I/O PACKET OFFSET
    UCBDF$      ;UCB OFFSETS
    SCBDF$      ;SCB OFFSETS

;
; EQUATED SYMBOLS
;
    .IIF      NDF      D$$R11,D$$R11=1

;
; DR11-C STATUS WORD BIT DEFINITIONS
;

READ=1          ;CURRENTLY READING
WRITE=2         ;CURRENTLY WRITING

REQB=100000     ;REQUEST B RUPT BIT
REQA=200        ;REQUEST A RUPT BIT
ENBA=100        ;ENABLE A RUPT BIT
ENBB=40 ;ENABLE B RUPT BIT
CSR1=2          ;CSR 1 BIT
CSR0=1          ;CSR 0 BIT
INBUF=4         ;OFFSET OF INPUT BUFFER
OUTBUF=2        ;OFFSET OF OUTPUT BUFFER
STRB=40000      ;STROBE FOR VOTRAX AT BIT 14

;
; LOCAL DATA
;
; CONTROLLER IMPURE DATA TABLES (INDEXED BY CONTROLLER NUMBER)
;

CNTBL: .BLKW D$$R11          ;ADDRESS OF UNIT CONTROL BLOCK

    .IF GT D$$R11-1
TEMP: .BLKW 1                ;TEMPORALY STORAGE FOR CONTROLLER NUMBER
    .ENDC

;
; DRIVER DISPATCH TABLE
;

$DRtbl: .WORD DRINI          ;DEVICE INITIATOR ENTRY POINT

```

```

.WORD  DRCAN          ;CANCEL I/O OPERATION ENTRY POINT
.WORD  DROUT          ;DEVICE TIMEOUT ENTRY POINT
.WORD  DRPWF          ;POWER FAIL ENTRY POINT

```

```

;+
; **-DRINI-DR11-C CONTROLLER INITIATOR
;
; THIS ROUTINE IS ENTERED FROM QUEUE I/O DIRECTIVE WHEN
; AN I/O REQUEST IS QUEUED AND AT THE END OF THE PREVIOUS
; I/O OPERATION TO PROPAGATE THE EXECUTION OF THE DRIVER.
; IF THE SPECIFIED CONTROLLER IS NOT BUSY, THEN AN ATTEMPT
; IS MADE TO DEQUEUE THE NEXT I/O REQUEST. ELSE A RETURN
; TO THE CALLER IS EXECUTED. IF THE DEQUEUE ATTEMPT IS
; SUCCESSFUL, THEN THE NEXT I/O OPERATION IS INITIATED.
; A RETURN TO THE CALLER IS THEN EXECUTED.
;
; INPUTS:
;
;      R5=ADDRESS OF UCB OF THE CONTROLLER TO BE INITIATED.
;
; OUTPUTS:
;
;      IF THE SPECIFIED CONTROLLER IS NOT BUSY AND AN I/O REQUEST
;      IS WAITING TO BE PROCESSED, THEN THE REQUEST IS DEQUEUED AND
;      THE I/O OPERATION IS INITIATED.
;-

DRINI:  CALL    $GTPKT      ;GET AN I/O PACKET TO PROCESS
        BCS     DREXIT      ;IF CS CONTROLLER BUSY OR NO REQUEST
;
; THE FOLLOWING ARGUMENTS ARE RETURNED BY $GTPKT:
;
;      R1=ADDRESS OF THE I/O REQUEST PACKET.
;      R2=PHYSICAL UNIT NUMBER OF THE FREQUEST UCB
;      R3=CONTROLLER INDEX.
;      R4=ADDRESS OF THE STATUS CONTROLL BLOCK.
;      R5=ADDRESS OF THE UCB OF THE CONTROLLER TOBE INITIATED.
;
; DR11-C I/O REQUEST PACKET FORMAT:
;
;      WD. 00--I/O QUEUE THREAD WORD.
;      WD. 01--REQUEST PRIORITY,EVENT FLAG NUMBER
;      WD. 02--ADDRESS OF THE TCB OF THE REQUESTER TASK
;      WD. 03--POINTER TO SECOND RUN WORD INREQUESTER TASK HEADER.
;      WD. 04--CONTENTS OF THE FIRST LUN WORD IN REQUESTER TASK HEADER (UCB)
;      WD. 05--I/O FUNCTION CODE (IO.RLB, IO.WLB)
;      WD. 06--VIRTUAL ADDRESS OF I/O STATUS BLOCK
;      WD. 07--RELOCATION BIAS OF I/O STATUS BLOCK.
;      WD. 10--I/O STATUS BLOCK ADDRESS (REAL OR DISPLACEMENT + 140000).
;      WD. 11--VIRTUAL ADDRESS OF AST SERVICE ROUTINE.
;      WD. 12--RELOCATION BIAS OF I/O BUFFER.
;      WD. 13--BUFFER ADDRESS OF I/O TRANSFER.
;      WD. 14--NUMBER OF BYTE TO BE TRANSFER.
;      WD. 15--NOTUSED.
;      WD. 16--NOT USED.
;      WD. 17--NOT USED.
;      WD. 20--NOT USED.
;
;
;
;      U.CW2 WILL BE USED TO STORE DRIVER STATUS INFOTMATION
;      U.CW3 IN UCB WILL BE USED TO STORE CSR VALUE WHEN A RUPT OCCURS
;
MOV     R5,CNTBL(R3)      ;SAVE UCB POINTER FOR INT ROUTINE
CLR     @S.CSR(R4)        ;CLR CSR

```

```

MOV B   S.ITM(R4),S.CTM(R4) ;SET TIMEOUT COUNT
CALL    CHECK                ;CHECK VALID SUBFUNCTION
BCS     BADFCT               ; BR IF ILLEGAL
BIS     #ENBA,R3             ;ENABLE THE A RUPT BIT
MOVB    I.FCN+1(R1),R2       ;MOVE FUNCTION CODE TO R2
CMPB    R2,#IO.RLB/256.      ;READ LOGICAL BLOCK FUNCTION?
BEQ     15$                  ;IF EQ YES

; DO
    BIS     #WRITE,U.CW2(R5);INDICATE WRITE OPERATION
    BR      20$

;   UTING
15$:    BIS     #READ,U.CW2(R5) ;INDICATE READ FUNCTION
20$:    BIS     R3,@S.CSR(R4)   ;ENABLE RUPT
        RETURN

BADFCT: MOV     #IE.IFC&377,R0  ; ILLEGAL SUBFUNCTION
        JMP     ENDIT          ;TERMINATE I/O

;
; POWER FAIL IS HANDLE VIA THE DEVICE TIMEOUT FACILITY
; AND THEREFORE CAUSES NO IMMEDIATE ACTION ON THE DEVICE.
; THIS IS DONE TO AVOID THE RACE CONDITION THAT COULD EXIST
; IN RESTARTING THE I/O OPERATION.
;
;     R5=UCB ADDR
;     R4=SCB ADDR
;     R3=CONTROLLER INDEX
;
DRPWF:  BIS     #CSR1,@S.CSR(R4) ;SET FORCE STOP BIT
DREXIT: RETURN

;+
; **-$DRINT-DR11-C CONTROLLERS INTERRUPTS
;--

$DRINT::                ;;;REF LABEL
        INTSV$ DR,PR5,D#$R11 ;;;GENERATE INTERRUPT SAVE CODE
;
; R5 NOW HAS UCB ADDRESS
;
; ONLY R4 AND R5 AVAILABLE UNTIL FORK IS CALLED
;

MOV      U.SCB(R5),R4      ;;;GET ADDRESS OF STATUS CONTROL BLOCK
MOVB     S.ITM(R4),S.CTM(R4) ;;;RESET TIMEOUT COUNT
MOV      S.CSR(R4),R4      ;;;POINT R4 TO CONTROL STATUS REGISTER
MOV      (R4),U.CW3(R5)    ;;;SAVE CSR VALUE
BIC      #CSRO!CSR1,(R4)   ;;;CLEAR CSRO/1 FOR TESTING PURPOSE
BIT      #WRITE,U.CW2(R5)  ;;;SEE IF DRIVER IS READING
BEQ      10$               ;;; IF EQUAL THEN YES WE ARE

; WRITING

CALL     $GTWRD            ;;;PICK UP WORD FROM BUFFER
MOV      (SP)+,OUTBUF(R4)  ;;;WORD TO BUFFER REGISTER
BIC      #STRB,OUTBUF(R4)  ;;;CLEAR BIT 14 FOR VOTRAX STROBE
BIS      #STRB,OUTBUF(R4)  ;;;SET THE BIT TO HIGH (STROBE HIGH)
NOP
NOP
NOP
NOP
BIC      #STRB,OUTBUF(R4)  ;;; STROBE LOW AGAIN
BR       15$

```

; READING

```

10$: BIT    $READ,U.CW2(R5)  ;;;MAKE SURE WE ARE READING
    BEQ     30$              ;;;BRANCH IF IT IS A TRASH INTERRUPT
    MOV     INBUF(R4),-(SP)  ;;;PUSH WORD READ
    CALL    $PTWRD          ;;;PLACE IT IN USER BUFFER
15$: SUB     $2,U.CNT(R5)     ;;;DECREMENT BYTE COUNT
    BLE     40$              ;;;IF EQUAL READ COMPLETE

;20$: BIT    $REQB,U.CW3(R5)  ;;;SEE IF B INTERRUPT
; BNE     40$                ;;;IF EQUAL READ COMPLETE
30$: JMP     $INTXT           ;;;EXIT FROM INTERRUPT

```

```

40$: CLR     (R4)              ;;;DISABLE DR11-C INTERRUPTS
    CALL    $FORK              ;;;CREATE SYSTEM PROCESS
    MOV     $IS.SUC&377,R0     ;ASSUME SUCCESSFUL TRANSFER

```

```

; BIT    $REQB,U.CW3(R5)  ;WAS I/O STOPPED BY REQB
; BEQ     ENDIT            ;BR IF NO
; MOV     $IE.EOF&377,R0   ;RETURN END OF FILE IF REQB WAS SET

```

```

ENDIT: CLR     U.CW2(R5)       ;CLEAR FLAG WORD
    CLR     U.CW3(R5)       ;CLEAR CSR ADDRESS SAVE WORD
    MOV     U.SCB(R5),R4     ;POINT R4 TO SCB
    MOV     S.PKT(R4),R1     ;POINT R1 TO I/O PACKET
    MOV     I.PRM+4(R1),R1   ;AND PICK UP CHARACTER COUNT
    SUB     U.CNT(R5),R1     ;CALCULATE CHARACTER TRANSFER
    CLR     R2               ;ZERO ERROR RETRY COUNT
    CALL    $IODON           ;INITIATE IO COMPLETION
    BR      DRINI            ;BRANCH BACK FOR NEXT REQUEST

```

; DEVICE TIMEOUT RESULT IN THE RETURN OF A DEVICE NOT READY ERROR CODE.

```

;
; R5=UCB ADDR
; R4=SCB ADDR
; R3=CONTROLLER INDEX
; R2=CSR ADDR
; R0=IE.DNR CODE
;

```

```

DROUT: CLR     @R2            ;;;DISABLE DR11-C RUPTS
    MTPS     $0              ;;;ALLOWS INTERRUPTS
    BR       ENDIT           ;CALL $IODON WITH IE.DNR CODE

```

; CANVEL INPROGRESS I/O

```

;
;
; R5=UCB ADDR
;
; R4=SCB ADDR
; R3=CONTROLLER INDEX
; R1=ADDRESS OF TCB OF CURRENT TASK
; R0=ADDRESS OF ACTIVE I/O PACKET
;

```

```

DRCAN: CMP     I.TCB(R0),R1   ;;;SAME TASK ?
    BEQ     1$               ;;; BR IF YES
    RETURN    ;;;NOOP IF NOT

```

```

1$: CLR     @S.CSR(R4)        ;;;CLEAR CSR TO FORCE TIMEOUT
    BIS     $CSR1,@S.CSR(R4) ;;;SET CSR1 TO STOP INTEL
    RETURN    ;;;WE ARE DONE

```

; ASSUMES R1 POINTS AT CURRENT I/O PACKET

; NOTE THAT R3 IS DESTROYED

;

```
CHECK:      MOVB      I,FCN(R1),R3      ;SUB FCT TO R3
            BLT       5$                ;BR IF ILLEGAL
            CMP       R3,#3             ;TEST HIGH LIMIT
            BGT       5$                ;BR IF ERROR
            CLC                          ;OK CLEAR C
            RETURN
5$:          SEC                          ;ERROR SET C
            RETURN
            .END
```

```

.TITLE DRTAB
.IDENT /01/
; DR11C DRIVER DATA TABLES
; AUTHOR: ROBERT MEYER
; VERSION 1      26-JAN-81
; IMAGE AND PATTERN ANALYSIS LAB
; UNIVERSITY OF TENNESSEE
;
;
; LEGAL CODES:   IO.WLB - WRITE LOGICAL BLOCK
;                IO.WVB - WRITE VIRTUAL BLOCK (CHANGED TO IO.WLB)
;                IO.RLB - READ LOGICAL BLOCK
;                IO.RVB - READ VIRTUAL BLOCK (CHANGED TO IO.RLB)
;                IO.DET - DETACH
;                IO.ATT - ATTACH
;                IO.KIL - CANCEL I/O REQUESTS
; ONLY IO.WLB AND IO.WVB CAUSE DRIVER ACTIVITY. (IO.KIL MAY CAUSE
;          ENTRY AT 'DDCAN'). OTHER I/O CODES RETURN AS AN ERROR.
; IO.RLB AND IO.WLB ALSO CAUSE DRIVER ACTIVITY
;
; NOTE: THIS DATA BASE IS FOR A LOADABLE DRIVER ON A MULTIUSER
;       SYSTEM.  THUS SPACE IS ALLOCATED FOR 'U.OWN', 'D.FCB',
;       AND THE RELOCATION BASE (WHICH IS AT THE END OF THE SCB).
;       ALSO, THE SYMBOLS 'LD$DD' AND 'D$D11' NEED NOT BE
;       DEFINED WHEN ASSEMBLING THE DATA TABLES.
;
.MCALL DEVDF$,HWDDF$
DEVDF$
HWDDF$                ;DEFINE DEVICE CONTROL BLK OFFSET
;
; DR11C DATA BASE
;
; DEVICE CONTROL BLOCK: <<DCB>>
$DRDAT::
DRICB: .WORD 0          ;<<D.LNK>> LINK TO NEXT DCB
        .WORD .DRO      ;<<D.UCB>> POINTER TO 1ST UCB
        .ASCII /DR/     ;<<D.NAM>> DEVICE NAME
        .BYTE 0,0        ;<<D.UNIT>> HIGHEST AND LOWEST DEVICES
                        ;COVERED BY THIS DCB
        .WORD DRND-DRST  ;<<D.UCBL>> LENGTH OF EACH UCB IN BYTES
        .WORD 0          ;<<D.DSP>> POINTER TO DRIVER DISPATCH TABLE
                        ;NORMALLY THIS IS DDTBL BUT
                        ;BECAUSE THIS DRIVER IS LOADABLE
                        ;IT MUST BE 0
        .WORD 37         ;<<D.MSK>> LEGAL FUNCTION MASK BITS 0-15
        .WORD 30         ;CONTROL FUNCTION MASK BITS 0-15
        .WORD 0          ;NOOP FUNCTION CODES BITS 0-15
        .WORD 0          ;ACF FUNCTIONS BITS 0-15
        .WORD 6          ;LEGAL FUNCT.S MASK BITS 16-31
        .WORD 0          ;CONTROL FUNCT.S          16-31
        .WORD 0          ;NOOPS FOR BITS 16-31
        .WORD 6          ;ACF CODES FOR BITS 16-31
        .WORD 0          ;<<D.FCB>> LEAVE FOR FCB ADDRESS
;
;***** DR11C UNIT CONTROL BLOCK <<UCB>>
;
DRST=.
.DRO::
        .WORD 0          ;<<U.OWN>> REQUIRED FOR MULTIUSER SYSTEMS
        .WORD DRICB      ;<<U.DCB>> POINTR BACK TO DCB
        .WORD .DRO       ;<<U.RED>> POINTER TO REDIRECT UNIT UCB
        .BYTE UC.ALG!1,0 ;<<U.CTL,U.STS>>
        .BYTE 0,0        ;<<U.UNIT,U.ST2>> PHYSICAL UNIT NUMBER,
                        ; AND UNIT STATUS EXTENSION

```

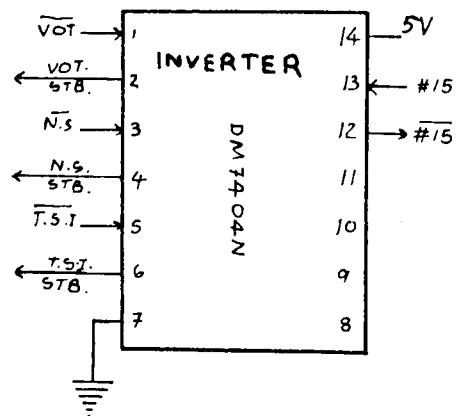
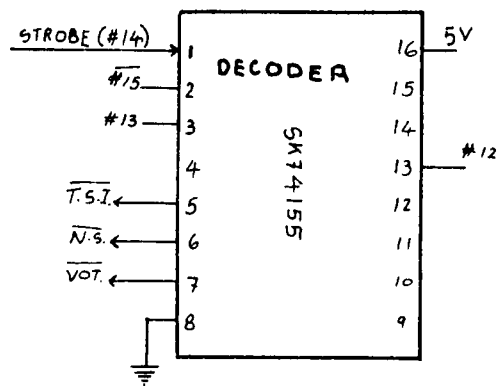
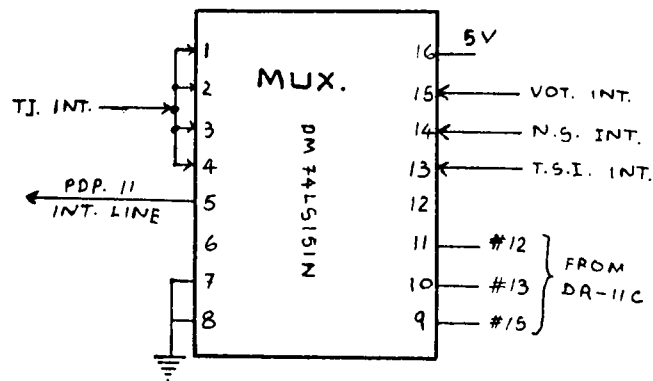
```

        .WORD    0          ;<<U.CW1>> FIRST DEVICE CHARACTERISTIC WRD
        .WORD    0          ;<<U.CW2>> SECOND DEV. CHARACTERISTIC WRD
        .WORD    0          ;<<U.CW3>> THIRD DEV. CHARACTERISTIC WRD
        .WORD    512.       ;<<U.CW4>> FOURTH DEV. CHAR. DEFAULT
                                ;BUFFER SIZE IN BYTES)
        .WORD    DRSCB      ;<<U.SCB>> POINTER TO SCB
        .WORD    0          ;<<U.ATT>> TCB ADDRESS OF ATTACHED TASK
        .BLKW    1          ;<<U.BUF>> RELOCATION BIAS OF BUFFER
                                ;
                                ; OF CURR. I/O REQUEST
        .BLKW    1          ;<<U.BUF+2>> ADDRESS OF BUFFER OF CURR.
                                ;
                                ; I/O REQUEST
        .BLKW    1          ;<<U.CNT>> BYTE COUNT OF CURRNT I/O REQST.
DRND=
;
;
;
DRSCB: .WORD    0          ;<<S.LHD>> CONTROLLER I/O QUE LSTHEAD.
                                ;
                                ; (PNTR TO 1ST ENTRY)
        .WORD    -2         ;
                                ; (PNTR TO LAST ENTRY)
        .BYTE    PR5,360/4  ;<<S.PRI,SVCT>> DEVICE PRI, INTRPT VEC ADD/4
        .BYTE    0,15.     ;<<S.CTM,S.ITM>> CURRNT AND INITIAL TIMEOUT STAT
        .BYTE    0,0       ;<<S.CON,S.STS>> CONTROLLER INDX AND STATUS
                                ;
                                ; (0=IDLE, 1=BUSY)
        .WORD    167770     ;<<S.CSR>> ADDRESS OF CONTROL STATUS REG
        .BLKW    1          ;<<S.PKT>> ADDRESS OF CURRENT I/O PAKET
        .BLKW    4          ;<<S.FRK>> FORK BLOCK ALLOCATION
        .BLKW    1          ;RELOCATION BASE OF DRIVER'S PAR
                                ;REQUIRED BECAUSE THIS IS LOAD-
                                ;BLE DRIVER - SEE I/O DRIVER MAN

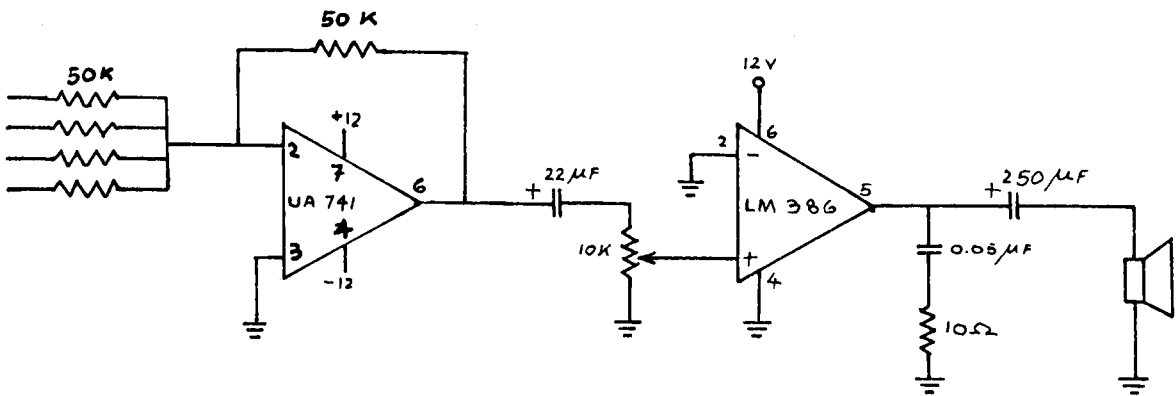
$DREND:: .END

```

APPENDIX C



APPENDIX D



VITA

Yudhana Aimthikul was born on May 11, 1956, in Bangkok, Thailand. After graduating from Assumption College High School in Bangkok in 1974, he attended Chulalongkorn University, from where he graduated in March 1978. He majored in Electrical Engineering with a specialization in power systems. He graduated from The University of Tennessee, Knoxville, in December 1981 with a Master of Science degree in Electrical Engineering. While at The University of Tennessee, he was both a teaching and research assistant. His major research area was computer speech output.