LIMITED CONTINUOUS SPEECH RECOGNITION BY PHONEME ANALYSIS
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING A HUSSAIN 08 DEC 83
UNCLASSIFIED AFIT/GE/EE/83D-31
LIMITED CONTINUOUS SPEECH RECOGNITION
BY PHONEME ANALYSIS

THESIS
AFIT/GE/83D-31
Ajmal Hussain
Captain  PAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY
Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
Approved for public release; Distribution is Unlimited
LIMITED CONTINUOUS SPEECH RECOGNITION
BY PHONEME ANALYSIS

THESIS

AFIT/GE/EE/83D-31
Ajmal Hussain
Captain PAF

Approved for public release; distribution unlimited.
LIMITED CONTINUOUS SPEECH RECOGNITION  
BY PHONEME ANALYSIS  

THESIS  

Presented to the Faculty of the School of Engineering  
of the Air Force Institute of Technology,  
Air University  
in Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science  

by  

Ajmal Hussain, BEE  
Captain PAF  
Graduate Electrical Engineering  
December 1983  

Approved for public release; distribution unlimited.
Preface

This work has been motivated by the research and enthusiasm of Dr. Matthew Kabrisky, Professor Electrical Engineering, Air Force Institute of Technology. This research effort produced a system capable of Limited Continuous Speech Recognition.

I would like to thank my advisor, Maj. Larry R. Kizer, and give special thanks to Dr. Matthew Kabrisky for his insight, and guidance during this project.

My greatest appreciation goes to my wife, Seemi Ajmal, who provided support in every way possible. Without her contributions, this research would never have been realized.

Ajmal Hussain
Table of Contents

Preface .......................................................... iii
List of Figures .................................................. v
List of Tables ................................................... vii
Abstract ........................................................... viii
I. Introduction ...................................................... 1
   Background ......................................................... 1
   Problem .......................................................... 2
   Scope ............................................................ 3
II. Theory and Techniques ........................................... 4
III. Hardware ......................................................... 8
   ASA-16 Spectrum Analyzer ....................................... 8
   Preprocessor ..................................................... 11
IV. Software .......................................................... 42
   Analog to Digital Conversion ..................................... 43
   Normalization ..................................................... 49
   Template Creation ................................................ 50
   Distance Matrix Creation ........................................ 53
   Phoneme Recognition ............................................. 55
   Word Library Creation ............................................ 61
   Speech Recognition ............................................... 63
V. Results and Conclusions ........................................ 67
   Phoneme Recognition ............................................. 67
   Discrete Word Recognition ....................................... 70
   Continuous Speech Recognition .................................. 70
Bibliography ......................................................... 74
Appendix A ........................................................... 75
Appendix B ........................................................... 76
Appendix C ........................................................... 79
Appendix D ........................................................... 81
Appendix E ........................................................... 83
Vita ................................................................. 140
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-1.</td>
<td>Hardware Block Diagram</td>
<td>9</td>
</tr>
<tr>
<td>3-2.</td>
<td>ASA-16 Functional Block Diagram</td>
<td>10</td>
</tr>
<tr>
<td>3-3.</td>
<td>Distribution of Analysis Band</td>
<td>10</td>
</tr>
<tr>
<td>3-4.</td>
<td>Schematic Diagram</td>
<td>13</td>
</tr>
<tr>
<td>3-5.</td>
<td>Frequency Response Preemphasis Filter</td>
<td>14</td>
</tr>
<tr>
<td>3-6.</td>
<td>3D-Plot Zero</td>
<td>15</td>
</tr>
<tr>
<td>3-7.</td>
<td>3D-Plot Zero with Preemphasis</td>
<td>16</td>
</tr>
<tr>
<td>3-8.</td>
<td>3D-Plot One</td>
<td>17</td>
</tr>
<tr>
<td>3-9.</td>
<td>3D-Plot One with Preemphasis</td>
<td>18</td>
</tr>
<tr>
<td>3-10.</td>
<td>3D-Plot Two</td>
<td>19</td>
</tr>
<tr>
<td>3-11.</td>
<td>3D-Plot Two with Preemphasis</td>
<td>20</td>
</tr>
<tr>
<td>3-12.</td>
<td>3D-Plot Three</td>
<td>21</td>
</tr>
<tr>
<td>3-13.</td>
<td>3D-Plot Three with Preemphasis</td>
<td>22</td>
</tr>
<tr>
<td>3-14.</td>
<td>3D-Plot Four</td>
<td>23</td>
</tr>
<tr>
<td>3-15.</td>
<td>3D-Plot Four with Preemphasis</td>
<td>24</td>
</tr>
<tr>
<td>3-16.</td>
<td>3D-Plot Five</td>
<td>25</td>
</tr>
<tr>
<td>3-17.</td>
<td>3D-Plot Five with Preemphasis</td>
<td>26</td>
</tr>
<tr>
<td>3-18.</td>
<td>3D-Plot Six</td>
<td>27</td>
</tr>
<tr>
<td>3-19.</td>
<td>3D-Plot Six with Preemphasis</td>
<td>28</td>
</tr>
<tr>
<td>3-20.</td>
<td>3D-Plot Seven</td>
<td>29</td>
</tr>
<tr>
<td>3-21.</td>
<td>3D-Plot Seven with Preemphasis</td>
<td>30</td>
</tr>
<tr>
<td>3-22.</td>
<td>3D-Plot Eight</td>
<td>31</td>
</tr>
<tr>
<td>3-23.</td>
<td>3D-Plot Eight with Preemphasis</td>
<td>32</td>
</tr>
<tr>
<td>3-24.</td>
<td>3D-Plot Nine</td>
<td>33</td>
</tr>
</tbody>
</table>
### List of Figures (con't)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-25.</td>
<td>3D-Plot Nine with Preemphasis</td>
<td>34</td>
</tr>
<tr>
<td>3-26.</td>
<td>3D-Plot Point</td>
<td>35</td>
</tr>
<tr>
<td>3-27.</td>
<td>3D-Plot Point with Preemphasis</td>
<td>36</td>
</tr>
<tr>
<td>3-28.</td>
<td>Frequency Response Low Pass Filter</td>
<td>38</td>
</tr>
<tr>
<td>3-29.</td>
<td>Low Pass and Preemphasis Filter</td>
<td>39</td>
</tr>
<tr>
<td>4-1.</td>
<td>Configuration File for CREATEMP</td>
<td>44</td>
</tr>
<tr>
<td>4-2.</td>
<td>Configuration File for SPEECH</td>
<td>45</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-1.</td>
<td>Filter Characteristics</td>
</tr>
<tr>
<td>4-1.</td>
<td>SAM Fortran Error Codes</td>
</tr>
<tr>
<td>4-2(a)</td>
<td>Distance Matrix</td>
</tr>
<tr>
<td>4-2(b)</td>
<td>Distance Matrix</td>
</tr>
<tr>
<td>4-2(c)</td>
<td>Distance Matrix</td>
</tr>
<tr>
<td>4-3.</td>
<td>Word and their Phoneme Representation</td>
</tr>
<tr>
<td>5-1.</td>
<td>Phoneme Recognition Results</td>
</tr>
<tr>
<td>5-2.</td>
<td>Library File</td>
</tr>
<tr>
<td>5-3.</td>
<td>Discrete Word Recognition Results</td>
</tr>
<tr>
<td>5-4.</td>
<td>Continuous Speech Recognition Results</td>
</tr>
</tbody>
</table>
Abstract

A Limited Continuous Speech Recognition system is developed based upon phoneme analysis. 16 bandpass filters are used to obtain the frequency components of the input speech. The input speech is broken into packets of 40 milliseconds each. These packets are compared with phonemes in a template file by a differencing of frequency magnitudes. The resulting phoneme string representation of the input speech is compressed and compared with strings in a library file for discrete word recognition. For continuous speech recognition the phoneme string is analyzed a phoneme at a time to construct word sequences. The word string which best matches the input phoneme string is recognized as the word sequence. The system has an accuracy of about 94% for discrete word recognition and about 80% for continuous speech recognition. The vocabulary used is the digits zero to nine and point.
LIMITED CONTINUOUS SPEECH RECOGNITION
BY PHONEME ANALYSIS

"THE AFTI F-16 WILL NOT BE SUCCESSFUL WITHOUT SPEECH RECOGNITION...CANNOT BE FLOWN DURING FULL COMBAT MANEUVERS."

---

JOHN C. RUTH, 1981
TECHNICAL DIRECTOR
F-16 ADVANCED FIGHTER DIV.
GENERAL DYNAMICS INC.

I. Introduction

This report documents the results and work accomplished during design of a Limited Continuous Speech Recognition (LCSR) system. The ultimate goal of this thesis is to obtain a system capable of recognizing a limited vocabulary with good accuracy.

Background

LCSR is generally understood in the speech research community to mean the problem of automatically recognizing natural human speech consisting of isolated utterances which are sequences of words chosen from a small (less than 30 word) vocabulary spoken continuously; i.e., without pauses or breaks between words. Speech is the most frequently used real time communications interface between
two human beings. A machine operator using voice control methods is free to use his hands and eyes in other ways. This can be a great advantage, for example in a fighter aircraft. A pilot can while undergoing a critical maneuver have access to his fire control or missile jamming systems through voice control. Continuous speech recognition, however has been an elusive goal, mainly due to variability that occurs in speech communication. The variability is a consequence of speaker-to-speaker variability, variations in the same speaker and effects of adjacent words with each other. In addition there is a background noise problem, especially in a cockpit environment.

Problem

The aim is to design and construct an Acoustic Analysis machine that will give a phoneme listing of a continuous speech input. The phoneme recognition should be fairly accurate in order to make the later word recognition step accurate. This machine will be part of an overall continuous speech recognition system. After the above problem of phoneme recognition was solved it was decided to do discrete word recognition based upon the output of the phoneme recognition system. Once this was achieved it was decided to do limited continuous speech recognition. Hence the overall problem became to design a system for limited continuous speech recognition.
Scope

This thesis is concerned with the recognition of phonemes uttered in continuous speech. The hardware consists of a preemphasis filter, an automatic gain control circuit and a bank of sixteen bandpass filters covering the frequency range of 200Hz to 7000Hz. The software consists of a routine to extract a set of phonemes then refine them in order to get an optimal prototype set. Another routine would use the results of a distance measurement computation for pattern matching in order to choose possible phoneme matches for each time period. Once the above was accomplished the scope of the thesis was increased to include isolated word recognition and limited continuous speech recognition. For this another routine is developed using shifting and the same distance measurement to construct a word or words from the phoneme sequence. The vocabulary used consisted of the digits zero to nine and point.
II. Theory and Techniques

In this chapter the approach used for speech recognition will be discussed. Initial approaches used will be discussed and reasons given, for choosing the final approach used.

Before going into the details it would be helpful to give an outline of a phoneme based speech recognition system. The first step is to take a speech input and break it up into a sequence of sounds. Each element of the sequence is compared against a set of unique sounds. These unique sounds are the phonemes. After comparison a sequence of phonemes is obtained which represents the input speech. This string is known as the phoneme representation of the input speech. This string is then processed to construct the word or words spoken.

The above outline was used in developing a phoneme based speech recognition system.

The input speech after preamplification is passed through a preemphasis filter. This preemphasis filter has a gain of 6db/octave above 500Hz. The reason for this is that the human voice has a roll off of 6db/octave above 500Hz. (Ref. 3 ). Next the input speech is passed through an automatic gain control circuit having a 60dB dynamic range. There were three reasons for using this AGC circuit. First the ASA-16 spectrum analyzer chip which follows, requires a certain minimum input level for proper operation.
Second the energy thresholding used (explained later), works on the basis of this AGC circuit. Third it reduces variations in the loudness of the input speech.

The ASA-16 spectrum analyzer chip was used to give sixteen band pass filter outputs of the input speech. The reasons for using a bank of bandpass filters in hardware (instead, say, of a fast fourier transform) are as follows. First it eliminates the inherent noise of a fast Fourier transform. Secondly the sampling can be done at a much lower rate. A typical sampling rate for a fast fourier transform method is 8KHz, whereas for a bandpass filter approach it is 400Hz.

The outputs of the sixteen bandpass filters of the ASA-16 are digitized using the Eclipse A/D/A device. A sampling frequency of 400Hz was chosen since this sampled each filter output at a frequency of 25Hz. The output of each bandpass filter is passed through a low pass filter having a cutoff frequency of 25Hz. This sampling rate was suitable since the variation in human speech does not go over 25 Hz. In this way one "slice" of the input speech, that is outputs of channels one through sixteen equals a time packet of 40 milliseconds.

Initially two slices of the input speech were taken as a phoneme representation. After repeated experimentation it was found that a single slice of sixteen channels was sufficient to represent a phoneme sound. Hence a phoneme sound is taken to be 40 milli seconds long and consists of a
sixteen dimensional vector whose elements are the outputs of the sixteen band pass filters.

After noise subtraction a 20 millivolt threshold level is used to get rid of background noise and D.C. offset errors. The phonemes or sixteen dimensional vectors are now individually energy normalized. This energy normalization is necessary for the phoneme recognition, comparison routine used.

These energy normalized vectors now represent the input speech. A set of unique phonemes known as the template is created. This method is explained in detail later. The phonemes in this template set are now compared with each of the vectors of the input speech. Initially a difference raised to the power of two approach was used. Finally a difference to the power of four approach was used for the comparison. This was done since it gave better results without overflowing the computer. The comparison method is explained in detail later.

This completes the phoneme recognition stage. The input speech is now in the form of a sequence of phonemes.

This sequence of phonemes is now compressed using techniques to be explained later. The reason for compression is to overcome the variations in the length of a word when spoken several times and the speed of speaking of a speaker.

This compressed phoneme representation of the input speech is fed to a word recognition algorithm. The details
of the discrete word recognition, and connected word recognition schemes are given later.

This then represents the outline of how speech recognition is done in this thesis.
III Hardware

A dynamic microphone placed close (1 inch) to the speaker's lips is used as the speech input device. After preamplification the audio signal is passed through a preemphasis filter. This filter has a 6db/octave gain from 500Hz upwards. An automatic gain control circuit with a dynamic range of 60db is used after the filter and is followed by a low pass filter having a cutoff frequency of 7000Hz. The output of the low pass filter is fed to the analog input of the ASA-16 spectrum analyzer chip. The sixteen band pass filter outputs of the ASA-16 are offset compensated and fed to the A/D converter of the Eclipse computer. A block diagram of the hardware is given as Fig 3-1.

ASA-16 Spectrum Analyzer

The ASA-16 is a monolithic audio spectrum analyzer with 16 channels of bandpass filters, half-wave rectifiers, and postfiltering. It is fabricated with double-poly NMOS technology and designed using switched-capacitor filter techniques.

A detailed functional block diagram of the chip is shown in Fig 3-2. A second-order bandpass filter serves to define the band of energy to be detected, followed by a half-wave rectifier and a low-pass filter. This individual function channel translates the analog waveform into a low-frequency signal that represents the corresponding energy
Figure 3-2 ASA-16 Functional block diagram

Figure 3-3 Distribution of analysis band
level within the band. There is a sampled-and-held multiplexer on the chip that sequentially outputs the sixteen outputs. The multiplex control timing is also incorporated on the chip. Direct access to outputs of all 16 channels is also available through 16 bonding pads. The distribution of the analysis band is shown in Fig 3-3, and the corresponding filter center frequencies and bandwidths are listed in Table 3-1. (Ref 2).

The chip has a dynamic range of better than 43dB, linearity of better than 1 percent, and center frequency accuracy of better than 1 percent. Technical data of the ASA-16 is given as Appendix B.

Preprocessor

A schematic diagram of the preprocessor is given in Fig 3-4. A preemphasis filter is used after the preamplifier since the human voice has an attenuation of about 6dB/octave from 500Hz upwards. So a preemphasis filter was designed which has a gain of 6dB/octave from 500Hz to 10KHz. The frequency response of the filter is given in Fig 3-5.

The effect of the preemphasis filter can be seen by comparing the three dimensional plots of the words zero to nine and point, with and without the filter. These three dimensional plots have the axis as frequency, time and magnitude. The plots are given in Fig 3-6 to Fig 3-27.
### TABLE 3-1. FILTER CHARACTERISTICS

<table>
<thead>
<tr>
<th>$f_0$(Hz)</th>
<th>Bandwidth(Hz)</th>
<th>Approximate Band Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$f_1$</td>
</tr>
<tr>
<td>260</td>
<td>130</td>
<td>203</td>
</tr>
<tr>
<td>390</td>
<td>130</td>
<td>330</td>
</tr>
<tr>
<td>520</td>
<td>130</td>
<td>459</td>
</tr>
<tr>
<td>650</td>
<td>130</td>
<td>588</td>
</tr>
<tr>
<td>780</td>
<td>130</td>
<td>718</td>
</tr>
<tr>
<td>910</td>
<td>140</td>
<td>843</td>
</tr>
<tr>
<td>1060</td>
<td>160</td>
<td>983</td>
</tr>
<tr>
<td>1220</td>
<td>180</td>
<td>1133</td>
</tr>
<tr>
<td>1400</td>
<td>200</td>
<td>1303</td>
</tr>
<tr>
<td>1600</td>
<td>220</td>
<td>1494</td>
</tr>
<tr>
<td>1820</td>
<td>250</td>
<td>1699</td>
</tr>
<tr>
<td>2070</td>
<td>300</td>
<td>1925</td>
</tr>
<tr>
<td>2370</td>
<td>340</td>
<td>2206</td>
</tr>
<tr>
<td>3035</td>
<td>1030</td>
<td>2563</td>
</tr>
<tr>
<td>4272</td>
<td>1445</td>
<td>3610</td>
</tr>
<tr>
<td>5997</td>
<td>2005</td>
<td>5077</td>
</tr>
</tbody>
</table>
Fig 3-11 3D-plot two with preemphasis
Fig 3-21  3D-plot seven with preemphasis
It is seen that without the preemphasis filter the higher frequency information is lost, and especially in the case of the words two and three they look almost similar without the preemphasis filter. With the filter they can be distinguished due to the amplification of the higher frequency components.

The transfer function of the preemphasis filter is

\[ F(S) = \frac{5 + 3333.33}{S + 31111.128} \]

It is calculated in Appendix C.

An active low pass filter follows the preemphasis filter. This low pass filter has a cutoff frequency of 7000Hz. It has a passband gain of 17dB and a peaking factor of 1. This low pass filter is used since the ASA-16 spectrum analyzer chip has a bandwidth of 200Hz to 7000Hz.

The transfer function of the low pass filter is:

\[ L(S) = \frac{-1.4945 \times 10^{10}}{s^2 + 47811.198s + 2.188 \times 10^9} \]

It is calculated in Appendix D.

The frequency response of the low pass filter is given in Fig 3-29.

A passband gain of 17dB was required to give the proper input level to the automatic gain control circuit that follows.

The overall frequency response of the preemphasis filter and low pass filter is given in Fig 3-30.
Fig 3-28  Frequency response low pass filter
Following the low pass filter is an automatic gain control circuit. This is used so that the ASA-16 spectrum analyzer chip gets a nearly constant average level input over a wide range of speaker voice levels. This automatic gain control circuit has a dynamic range of 60dB. The JFET acts as a voltage-controlled resistor in the peak-detecting control loop of the 741 operational amplifier. The circuit has an input range of 20mV to 20V. The output is about 1V-4V peak to peak over the entire 60dB range. The response time of the circuit is about 1 to 2 mSec., and has a delay of about 0.4 seconds. Use of the automatic gain control circuit eliminated the problems of clipping of the audio signal and also that of too low an input to the spectrum analyzer chip. The gain response of the AGC circuit is given in Fig 3-31.

A TTL crystal controlled 1MHz clock is used to clock the ASA-16 spectrum analyzer chip. The specifications of the ASA-16 spectrum analyzer chip, requires that the clock and power supply to the ASA-16 chip be applied simultaneously. The clock and power supply are hence set up in this way. The whole preprocessor board is powered by ±10V, and draws about 100mA of current. The ±10V supply is obtained by on board voltage regulators.

As explained earlier in the discussion of the ASA-16 spectrum analyzer chip, there is a D.C. offset on each of the sixteen band pass filter outputs of the chip. Also each of these sixteen outputs cannot be terminated with a
resistance less than 200Kohms. For this reason buffers with offset control are used between these outputs and the inputs to the analog to digital converter of the Eclipse computer. These inverting buffers are made of SN72L044 quad-op amp chips. These sixteen outputs give the frequency information of the input speech.

The Eclipse analog to digital converter is externally clocked with a 400Hz TTL signal. This samples each of the 16 channels at 25Hz.

The whole preprocessor hardware is designed on a single board.
IV Software

The software package works in five stages. The first stage is the creation of templates. In this stage a maximum of twenty seconds of speech is input to the system; this speech is selected to include a large number of different phonemes. This is done by using phonetically balanced sentences. After normalization (which will be explained in detail later) all the phonemes found are compared with each other. Those phonemes which are "close" to each other by the distance measurement used (explained later) are discarded. This gives us a smaller set of phonemes which are relatively "far" from each other, that is distantly different. This set of phonemes is stored in a file to be used for comparison in the phoneme recognition scheme.

The second stage is the formation of the distance matrix. Here the phonemes found in the first stage are compared with each other and their distances (explained later) calculated. These distances are stored in lower triangular form in a file called the distance matrix. This distance matrix will be used in the data compression and word recognition schemes.

The third stage is phoneme recognition. In this stage the speech input to be analyzed is compared with the template of phonemes created in stage one. This produces a string of phonemes which is compressed using the distance matrix. This string is the phoneme representation of the input speech.
The fourth stage is the creation of a word library. In this stage a file is created which contains the phoneme representation of the vocabulary. The phoneme strings for the vocabulary are created in the same way, as explained in stages one through three. This library file of the vocabulary will be used for comparison in the discrete word and continuous word recognition schemes.

The fifth stage is the word recognition algorithm. This is a recursive algorithm and uses the threshold, string lengths, distance matrix and error value information to come up with the best word or words which represent the input phoneme string. The exact method will be explained later.

The above gives a general outline of how the software package works. The details of the process will now be given.

**Analog to digital conversion**

The details of the Eclipse A/D/A device are given as Appendix A, to Gorden R. Allen's thesis "Expansion of the Eclipse digital signal processing system" (Ref 1). Two configuration files are required. One for program CREATEMP and the other for program SPEECH. Program SAMGEN is used to create these files and they are given in Fig 4-1 and Fig 4-2.

The Eclipse A/D/A device is set up to sample the sixteen band-pass filter outputs sequentially. The sampling is cyclic channel one to channel sixteen, and again from channel one, and so on. The A/D converter is clocked
Answers you gave in the SAMGEN dialog are shown in comment lines. Your inputs are immediately preceded by a colon (:) and appear in the same order as you gave them to SAMGEN.

Target operating system type: MRI
Number of DG/DAC 4300 chassis configured: 0
Fatal error handler name : -1
Fatal error handler mailbox: -1

DCB. X SAMCO '100 -1 -1

Number of Analog Subsystem: 1

A/D Con. #1 Device Code : 21 Mode : AD Fortran ID = IDS21
External interrupt handler specified : <NONE>
Number of pages in Data Channel area: 16
Specifying a starting address for Data Channel area: Y
Data Channel starting address : IBUFF

DCB. M DBS21 D. IDF + D. INF + D. DCH 21
DCB. I DTS21 SAINI 16. IBUFF
DCB?C -1 -1 DSS21
DCT. M DTS21 000377 INTSA DSS21

DCB'. N S21 D. FIF 21 00 AD
DCB'. S DBS21 0 AD. IS AD. IN SAIRT
DCB'. A

D/A Con. #1 Device Code : 23 Mode : BD Fortran ID = IDS23
External interrupt handler specified: <NONE>
Number of pages in Data Channel area: 16
Specifying a starting address for Data Channel area: Y
Data Channel starting address: IDUFO

DCB. M DBS23 D. IDF + D. INF + D. DCH 23
DCB. I DTS23 SAINI 16. IBUFO
DCB?C -1 -1 DSS23
DCT. M DTS23 000377 INTSA DSS23

DCB'. N S23 D. FIF 23 00 BD
DCB'. S DBS23 0 BD. IS BD. IN SAIRT
DCB'. A

DCB'. E

End of SAMGEN configuration file.

Figure 4-1 Configuration file for CREATEMP
Answers you gave in the SAMGEN dialog are shown in comment lines. Your inputs are immediately preceded by a colon (:) and appear in the same order as you gave them to SAMGEN.

- Target operating system type: MHD
- Number of DG/DAC 4200 chassis configured: 0
- Fatal error handler name: -1
- Fatal error handler mailbox: -1

```
DCB.X SAMCO 100 -1 -1
```

Number of Analog Subsystem: 1

- A/D Con. #1 Device Code: 21 Mode: AD Fortran ID = IDS21
  - External interrupt handler specified: <NONE>
  - Number of pages in Data Channel area: 2
  - Specifying a starting address for Data Channel area: Y
  - Data Channel starting address: IBUFF

```
DCB.M DBS21 D.IDF+D.INF+D.DCH 21
DCB.I DTS21 SAINI 2. IBUFF
DCB?C -1 -1 DSS21
DCT.M DTS21 000377 INTSA DSS21
DCB.N S21 D.FIF 21 00 AD
DCB.S DBS21 0 AD.IS AD.IN SAIRT
DCB.A
```

- D/A Con. #1 Device Code: 23 Mode: BD Fortran ID = IDS23
  - External interrupt handler specified: <NONE>
  - Number of pages in Data Channel area: 2
  - Specifying a starting address for Data Channel area: Y
  - Data Channel starting address: IBUFF

```
DCB.M DBS23 D.IDF+D.INF+D.DCH 23
DCB.I DTS23 SAINI 2. IBUFF
DCB?C -1 -1 DSS23
DCT.M DTS23 000377 INTSA DSS23
DCB.N S23 D.FIF 23 00 BD
DCB.S DBS23 0 BD.IS BD.IN SAIRT
DCB.A
```

End of SAMGEN configuration file.

Figure 4-2 Configuration file for SPEECH
externally with a 400Hz TTL signal. This means that each bandpass filter output is sampled at 25Hz. Since the ASA-16 spectrum analyzer chip has a 25Hz low pass filter at each band pass filter output, this sampling rate of 25Hz is sufficient. The voltage output of the sixteen channels of the ASA-16, each ranges between 0V to +4.5V. Similarly the outputs of the sixteen inverting buffers range between 0V to -4.5V. The Eclipse A/D converter is setup to accept an analog input voltage range of -5V to +5V.

The sampled voltage values are stored in integer form in a buffer. The maximum size of the buffer is decided by the configuration file used. A CALL DSTRT(IER) command is given to initialize the Eclipse A/D/A device. In case of an initialization error the error value will be displayed on the terminal. A CALL DOITW[ ] command is used to do the analog to digital conversion. Again if an error occurs the error number is displayed on the screen. A table of the error conditions is given in Table 4-1. If all went well with the A/D conversion a message "no errors reported" is displayed.

The variable space to hold the conversion values of a single conversion operation can be a maximum of 16KW of integer array space. At a sampling rate of 400Hz this gives us a maximum of 40 seconds of speech input, possible. So as to reduce processing time overlays and extended memory techniques were not used. This gave a 20 second speech input time for program CREATEMP and a 5sec speech input time
<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2179</td>
<td>No LNK routine in DCB, invalid DCB. Often results from an invalid device-id, so check the device-ids. The first two characters are ID, the third either S, A, or O, and the last two are numbers (e.g., IDS21).</td>
</tr>
<tr>
<td>2180</td>
<td>No DCB identifier in IORB, invalid DCB. Same cause as 2179.</td>
</tr>
<tr>
<td>2181</td>
<td>No used. This error should not occur.</td>
</tr>
<tr>
<td>2184</td>
<td>No initializing routine for a device that needs initialization. Same cause as 2179.</td>
</tr>
<tr>
<td>2185</td>
<td>Output requested to a channel for an illegal device (e.g., output to an A/D converter).</td>
</tr>
<tr>
<td>2186</td>
<td>Attempt to set up a locked IORB array. This can happen if a second DSAN/DSOR call uses the same IORB array argument before the original DSAN/DSOR completes.</td>
</tr>
<tr>
<td>2187</td>
<td>Unable to find free IORB block in IORB array. Can happen if the IORB array was DIMENSIONed too small. A multiple-operation call needs 8 elements + 8 elements per operation.</td>
</tr>
<tr>
<td>2188</td>
<td>No DCB exists with specified device-id. Same cause as 2179.</td>
</tr>
<tr>
<td>2189</td>
<td>Attempt to use unsupported feature (e.g., mapped call in unmapped system).</td>
</tr>
<tr>
<td>2190</td>
<td>Attempt to return bad buffer. Will never occur.</td>
</tr>
<tr>
<td>2191</td>
<td>An IDATAx argument gave an illegal clock setting for an A/D or D/A converter.</td>
</tr>
</tbody>
</table>

Table 4.1 SAM Fortran error codes
(SAM User's Manual, p. 6-9)
<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2192</td>
<td>Illegal conversion count -- more than 255 or less than 1 -- for an A/D converter mode in A2; DG/DAC only.</td>
</tr>
<tr>
<td>2193</td>
<td>Assembly language only. Attempt to move data channel map while IORB is locked. A task tried to change the map while a request was using the window.</td>
</tr>
<tr>
<td>2194</td>
<td>Attempt to move data channel map to an address outside the window.</td>
</tr>
<tr>
<td>2195</td>
<td>Illegal conversion count: less than 1 or more than the device allows.</td>
</tr>
<tr>
<td>2196</td>
<td>Interrupt occurred from 4222 without a strobe or latch change.</td>
</tr>
<tr>
<td>2197</td>
<td>Assembly language only. Attempt to use data channel map while it is being initialized or moved.</td>
</tr>
<tr>
<td>2198</td>
<td>Assembly language only. Data channel not initialized: use an RMAP call before issuing this mode A2 request.</td>
</tr>
<tr>
<td>2199</td>
<td>SAM panic code. SAM could not transmit (.IXMT) to the calling task on IORB array completion. SAM aborts the program unless you set up a fatal error handling RECEIVE task and gave its name to SAMGEN, as described in Chapter 5, “Initial Dialog”.</td>
</tr>
<tr>
<td>2200</td>
<td>External interrupt occurred on a stand-alone analog converter, aborting the request. This error returns from ISA calls only, not from DSAN/DSOR calls.</td>
</tr>
</tbody>
</table>

Table 4.1 continue
for program SPEECH. The actual voltage value of the sampled conversion value can be calculated using,

\[ \text{VOLTAGE} = \text{FLOAT} \left( \frac{\text{CONV.NUM}}{32768} \right) \cdot 5 \]

**Normalization**

As explained before each cycle i.e. sampling of channel one to channel sixteen takes 40 milliseconds. In this way the input speech is divided in slices of 40 milliseconds each. Now each 40 millisecond slice is represented by a 16 dimensional vector got from the 16 band pass filter outputs. This 16 dimensional vector is a unit of information and will be henceforth called a phoneme representation.

The normalization process consists of noise subtraction, thresholding, energy calculation and energy normalization. The noise subtraction is done by assuming that the very first 16 samples represent the average noise and D.C. offset in each of the channels. Hence the very first 16 dimensional vector is subtracted from all the remaining vectors in the data buffer. The thresholding is done by examining each component of every vector and putting it to zero if it has an integer value less than 200. An integer value of 200 represents a voltage of:

\[ V_{\text{threshold}} = \frac{200 \times 5}{32768} = 30.5 \text{ millivolts} \]

An energy calculation for each vector is done using the formula
This energy information is stored in an array for use later on in the data compression and connected word recognition schemes.

Now each of the vectors is energy normalized. This is done by dividing each component of a vector by its vector energy and multiplying it by 32000. This normalizes the vector to a value of 32000. Mathematically it is as follows:

\[
x(n, m) = \frac{x(n, m) = 32000}{\sum_{i=1}^{16} (x(i, m))^2}
\]

These normalized 16 dimensional vectors now represent the input speech divided into phonemes of 40 milliseconds each.

Template creation

Program CREATEMP is used for template creation. A maximum of 40 seconds of speech input is possible. This speech input is given in the form of phonetically balanced sentences. The aim is to have as many different phonemes as possible. The input buffer is then normalized using the method explained earlier. The normalized phonemes are then compared with each other, by the distance measurement.

\[E = \sum_{i=1}^{16} (x_i)^2\]
technique to be described later. Those phonemes which are close to each other distance wise are now discarded. The resultant subset of phonemes now make up the template file.

The main menu of program CREATEMP is as follows:

1. A/D conversion
2. data buffer display
3. data buffer print
4. normalize
5. compare phonemes
6. delete unwanted phonemes
7. compress template
8. template write to file
9. read template file
10. delete specified phonemes
11. exit

Each of the options will now be explained. The A/D conversion operation can take a speech input of 1 second to 20 seconds. The conversion values are stored in a 16KW integer data buffer. The data buffer display and data buffer print options are self explanatory. The normalize option, normalizes the data buffer as explained earlier.

The compare phonemes, compares each phoneme (or 16 dimensional vector) with every other phoneme in the data buffer. The comparison is done using the vector distance rule as follows.
\[ \text{Distance}(m,n) = \sum_{i=1}^{16} (x(m,i) - x(n,i))^4 \]

This gives a distance measurement between phoneme \( m \) and phoneme \( n \). Each phoneme with the phoneme closest to it and their distance is printed. The distance is normalized to a maximum of 100 for printing. A sample printout is given in Table 4-2. This gives an idea of which phonemes are too close to each other and must be deleted from the template.

The delete unwanted phonemes option, checks the energy value of each phoneme. From this energy value it is decided with a threshold level, whether this phoneme is noise or a speech input. Those phonemes which have an energy value below the threshold are deleted as being noise input.

The delete specified phoneme option asks for a phoneme number and deletes that particular phoneme. The aim is to delete those members of a phoneme string which are too close to a preceding phoneme. The closeness of the phonemes is decided using option compare phonemes as explained earlier.

The compress template option is used after the delete options to compress the whole template. That is those phonemes which were deleted are removed from the template and the rest compressed together. Note that this function now gives new numbers to all the phonemes, since the number of phonemes in the template are now reduced.
The read template file option, reads from the current directory a template file for editing purposes.

The template write to file option, creates a template file and stores the template in it. The number of phonemes in the template is stored in position (1121) of the integer file. This template file will be used in programs DISMAT and SPEECH. Details of these programs are given later.

Distance Matrix Creation

Program DISMAT is used for distance matrix creation. The main menu of the program is as follows:

1. read template file.
2. form distance matrix.
3. display distance matrix.
4. print distance matrix.
5. distance matrix write to file.
6. read distance matrix file.
7. display templates.
8. give distance between two phonemes.
9. exit.

The read template file option is first used to read the template file to be worked on into a 1130 word integer array. Before proceeding further it is necessary to first explain what a distance matrix is. The distance between each phoneme in a template and every other phoneme in the template is calculated. These distances make up the distance matrix for that template. The aim is to reduce the
speech recognition process time. This is done by avoiding
the calculation of these distances over and over again in
the speech recognition process. This is explained in detail
later. Distances between phonemes is calculated using the
same scheme as explained earlier for phoneme comparison.
The formula used is:

$$\text{Distance}(m,n) = \sum_{i=1}^{16} (x(m,i) - x(n,i))^4$$

This gives the vector distance between phoneme \(m\) and
phoneme \(n\). Since the distance is same between phoneme \(n\)
and phoneme \(m\). Hence only one distance is calculated and
stored. The distances are stored in a lower triangular
form, instead of in a two dimensional array. The reason for
doing this is to reduce storage space. In this scheme the
distance between phoneme \(m\) and phoneme \(n\) is given in
location:

$$\text{Location} (m,n) = \frac{m(m-1)}{2} + n$$

Provided \(m\) is greater than \(n\). If \(m\) is less than \(n\),
then values of \(m\) and \(n\) can be interchanged. The saving in
storage area can be made obvious with the help of an
example. Consider a template having 70 phonemes. Using a
two dimensional array for storing the distance matrix would
require \((70 \times 70) = 4900\) words of real space. On the other
hand using a lower triangular form would require
(70 x (70-1))/2 + 70 = 2485 words of real space. This is a saving of (4900 - 2485) = 2415 words of real space.

The form distance matrix option, calculates the distance matrix and stores it in a 2432 word real array, using the scheme explained above. The values of these distances are normalized to a maximum of 10000.

The print distance matrix option, prints the distance matrix in lower triangular array form. Since the printer can only handle 132 characters in a line. Hence the matrix is broken up into several pages, which can then be pasted together to display the whole distance matrix. The distance matrix is printed out in integer form. It is also normalized to a maximum value of 100 for printing purposes only. A sample printed output is shown in Table 4-2.

The give distance between two phonemes option, accepts two phoneme numbers and displays their distance. The rest of the options are quite self explanatory.

Phoneme recognition.

As explained earlier the speech recognition scheme used is based upon phoneme recognition and then construction of words from the phoneme string. The phoneme recognition scheme is a part of the speech recognition program called SPEECH. In program SPEECH a template file and its corresponding distance matrix file are read into buffers. These files are used in the phoneme recognition method.

Program SPEECH accepts as input a maximum of five seconds of speech. This speech input is normalized as
explained earlier. This normalized speech is the basic input to the phoneme recognition scheme. As explained earlier a phoneme is represented by a 16 dimensional vector, which are the outputs of the 16 bandpass filters. A scheme similar to the compare phoneme, explained earlier is used.

The phoneme recognition routine takes the first 16 dimensional vectors of the input speech and compares it with all the phonemes in the template. The comparison is done based upon the vector distances between the phonemes. The smaller the vector (i.e. less the distance), the closer is the match or comparison. The formula used for the distance measure is:

\[
Distance(m,n) = \sum_{i=1}^{16} (S(m,i) - T(n,i))^4
\]

where

\[
Distance(m,n) = \text{distance between speech input phoneme (m) and template phoneme (n)}. \\
S(m,i) = \text{i}^{th} \text{ component of speech input phoneme (m)}. \\
T(m,i) = \text{i}^{th} \text{ component of template phoneme (n)}. 
\]

The phoneme in the template which is closest to the phoneme in the speech input is taken as the phoneme representation of that phoneme in the speech input. In this way all the 16 dimensional vectors in the speech input are given phoneme representations from the template. This gives a phoneme string which represents the input speech. Those vectors in the input speech which have an energy level below
the threshold are given a phoneme number of zero. The threshold level was calculated from the background noise of the laboratory environment. This was done by operating the system with no speech input and taking an average energy value.

The phoneme string representation of the input speech is then compressed using a number of rules which will now be explained.

All adjacent similar phonemes are compressed. That is if there is repetition of the same phoneme, only one phoneme is kept. For example if the phoneme string is:

12, 21, 21, 21, 16, 19

it will be compressed to:

12, 21, 16, 19.

Leading edge zeros of the string are ignored. That is, the delay between start of speech input and start of the conversion process is removed. Since phoneme number zero represents noise or no speech input to the system.

Two or more adjacent zero number phonemes within the phoneme string are recognized as breaks in speech input and are represented by a single phoneme number zero. A single phoneme number zero within the phoneme string is recognized as a stop sound within a word and is ignored.

Distances between all adjacent phonemes in the string are obtained from the distance matrix. Those adjacent phonemes which have a distance less than a given threshold are compressed. The compression is done by discarding the higher numbered phoneme. This scheme tends to give the
lowest numbered phoneme for the sub string of adjacent, close phonemes.

This compression technique eliminates the problems caused due to variations in the length of a word. That is variations due to the speed of speaking of a speaker.

This final string of phonemes after compression will be used as input to the word recognition scheme.

Word library creation

The speech input to be analyzed is converted into a phoneme string as explained earlier. In order to be able to construct a word or words from this string of phonemes, it is necessary to know which phonemes actually make up the word. Hence each word in our vocabulary is represented by its phoneme string. The phoneme string which comprises a word is deduced by an averaging method.

The phoneme recognition scheme is run for a number of repetitions of the same word. From the phoneme strings produced the phoneme string which is repeated the maximum number of time is chosen.

This chosen string is the phoneme representation of that particular word. This process is repeated for all the words in the vocabulary. The words in the vocabulary and their respective phoneme representations are given in Table 4-3.

Program VOCAB is used to create a library file containing the phoneme representations of our vocabulary.
Table 4.3

Word and their Phoneme Representations

<table>
<thead>
<tr>
<th>Word</th>
<th>Phoneme Representations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>26, 27, 11, 29, 11</td>
</tr>
<tr>
<td>One</td>
<td>29, 36, 46</td>
</tr>
<tr>
<td>Two</td>
<td>3, 6, 18</td>
</tr>
<tr>
<td>Three</td>
<td>22, 53, 6</td>
</tr>
<tr>
<td>Four</td>
<td>9, 30, 9</td>
</tr>
<tr>
<td>Five</td>
<td>13, 38, 39, 11</td>
</tr>
<tr>
<td>Six</td>
<td>40, 42</td>
</tr>
<tr>
<td>Seven</td>
<td>2, 40, 38, 48</td>
</tr>
<tr>
<td>Eight</td>
<td>49, 6, 53</td>
</tr>
<tr>
<td>Nine</td>
<td>15, 56, 57, 58</td>
</tr>
<tr>
<td>Point</td>
<td>12, 13, 15, 39</td>
</tr>
</tbody>
</table>
This program is also used for editing and modification of a previously created library file.

**Speech recognition**

In order to explain the speech recognition process it is necessary to first explain the discrete word recognition scheme used. Subroutine FINDWORD does the discrete word recognition process. The input to this subroutine is the phoneme string representation of the input speech. This phoneme string is compared against all the phoneme strings in the library. The comparison is done on the basis of an error value. The error value is calculated based upon a number of rules, which will now be explained.

Each phoneme of the input string is compared with the corresponding phoneme in a string in the library. Every word in the library is in the form of a phoneme string. The first phoneme of the input string is compared against the first phoneme of a word string in the library. The second against the second and so on. The comparison is done using the distance matrix and adding that distance value to the error. This is done for all the words strings in the library. No form of transition rule for going to the next phoneme is used, since the compression stage (as explained earlier) eliminates multiple adjacent phonemes, and adjacent phonemes which are too close to each other. Hence a one to one comparison is done between the input string and all the word strings in the library.
Since the number of phonemes in a word differ from word to word a penalty value is added to the error value. This penalty value depends upon the difference in the number of phonemes in the input string, and the number of phonemes in a particular word string in the library. By repeated experimentation, an initial value of 120 was found for the penalty, which gave the best word recognition score. The penalty value is an accumulative function. For example if the difference between the number of phonemes in the input string and a particular word string in the library is given by \( N \), then the error is calculated as:

\[
E_{i+1} = E_i + 120 \quad i = N_1 \text{ to } N_2
\]

where \( N_2 - N_1 = N \)

The above calculation is repeated twice again. Once by shifting the input phoneme string one phoneme right and again by shifting the input phoneme string one phoneme left from its original position.

This means that it is assumed that the first phoneme in the input string is in error. This assumption was made because of the threshold technique used in finding the start of a word. It is possible to have noise as the first phoneme in the string or to miss the first phoneme in the word. The threshold technique used was explained earlier in the chapter.

In this an average value of error is calculated for each word in the library. The word string in the library
which gave the minimum error value is chosen as the best match to the input phoneme string. This word then is the output of the discrete word recognition subroutine.

The connected word recognition scheme makes use of the discrete word recognition scheme as follows. The input phoneme string is checked for number of phonemes between zero phoneme values. Two limits are put on the number of phonemes in a word. It is seen for the given vocabulary that the minimum number of phonemes in a word is two, and the maximum number of phonemes in a word is eight. So if the number of phonemes between any two zeroes is less than nine, it is assumed to be one word and the discrete word recognition scheme is used to find the word. If the number of phonemes is greater than or equal to nine between any two zeroes it is assumed that this string consists of two or more words. In this case the connected word recognition scheme is used. Each phoneme from the input string is added to the buffer one at a time. After each addition the temporary buffer is assumed to be a word string and the discrete word recognition algorithm applied to it. At the end of ten additions, a deck is made for the point where, in the addition of phonemes, the best match to a word in the library occurred. This word is chosen as the first word in the connected word string. The above process is repeated starting from the last phoneme of the previous word. The last phoneme of the previous word is used again since in connected speech the last phoneme of the previous word and
the first phoneme in the next word can be same. If they are not in a particular case, the error is removed by the shifting used in the discrete word recognition scheme, explained earlier.

The above process is repeated till the end of the input phoneme string is reached. The total error value for the whole input phoneme string is calculated. Next it is assumed that the recognition of the first word in the connected word string was in error. The next best match for the first word is chosen and the whole above process repeated. A total error value for this recognition of the input phoneme string is calculated. Next it is assumed that the second word then the third and so on are in error and all their total error values calculated. In the end the process which gave the minimum total error value is chosen as the best recognition of words for the input phoneme string. This sequence of words is displayed and the whole process repeated starting with the next string of phonemes, between two zero numbered phonemes. In this way discrete word recognition and connected word recognition is done till the end of the input phoneme string is reached.

The output is displayed on the H-19 terminal on a line in reverse video using subroutine WTYPE.
V. Results and Conclusions

The system was initially designed to recognize phonemes uttered in continuous speech. Once fairly consistent phoneme recognition was achieved, the problem of discrete word recognition was tackled. Once an accuracy of about 90% was achieved with an eleven word vocabulary, the problem of continuous speech recognition was approached. In the end an accuracy of about 80% was achieved for continuous speech recognition.

Phoneme Recognition

Using program CREATEMP a template file was created. The speech input was given by a tape containing the following sentences.

"It's time to round up the herd of Asain cattle," "Few theives are ever sent to the jug," "May we all hear the yellow lion roar," "We were away a year ago," "Zero one two three four five six seven eight nine point."

The template file is given as Appendix A.

Program DISMAT was used to create a distance matrix file for this template file. The distance matrix is given in Table 4-2.

Program SPEECH was used in the phoneme recognition made to give the phoneme strings recognized for the words in our vocabulary. The words in the vocabulary contain the digits
Table 5-1

Phoneme Recognition Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>25, 27, 18, 11, 34, 11, 29</td>
</tr>
<tr>
<td>Zero</td>
<td>26, 27, 11, 29, 11</td>
</tr>
<tr>
<td>One</td>
<td>29, 60, 36, 48</td>
</tr>
<tr>
<td>One</td>
<td>29, 60, 36, 44</td>
</tr>
<tr>
<td>Two</td>
<td>3, 6, 18</td>
</tr>
<tr>
<td>Two</td>
<td>3, 6, 18, 21</td>
</tr>
<tr>
<td>Three</td>
<td>6, 27</td>
</tr>
<tr>
<td>Three</td>
<td>6, 27</td>
</tr>
<tr>
<td>Four</td>
<td>9, 30, 9</td>
</tr>
<tr>
<td>Four</td>
<td>9, 30, 9</td>
</tr>
<tr>
<td>Five</td>
<td>35, 36, 39, 40</td>
</tr>
<tr>
<td>Five</td>
<td>35, 36, 39, 40, 43</td>
</tr>
<tr>
<td>Six</td>
<td>40, 42</td>
</tr>
<tr>
<td>Six</td>
<td>40, 42</td>
</tr>
<tr>
<td>Seven</td>
<td>40, 45, 46, 48</td>
</tr>
<tr>
<td>Seven</td>
<td>40, 46, 48</td>
</tr>
<tr>
<td>Eight</td>
<td>53, 23, 53, 11</td>
</tr>
<tr>
<td>Eight</td>
<td>53, 23, 53, 11</td>
</tr>
<tr>
<td>Nine</td>
<td>48, 46, 57, 58</td>
</tr>
<tr>
<td>Nine</td>
<td>15, 56, 57, 58</td>
</tr>
<tr>
<td>Point</td>
<td>12, 13, 15, 54</td>
</tr>
<tr>
<td>Point</td>
<td>12, 13, 15, 54</td>
</tr>
</tbody>
</table>
zero to nine and point. The result of the phoneme recognition is given in Table 5-1. It can be seen that phoneme strings for the same word are either quite similar or exactly the same.

**Discrete Word Recognition**

Program CLIB was used to create a library file. The phoneme strings produced by the phoneme recognition process are used to create this library file which represents our systems vocabulary. The library file is given in Table 5-2.

Program SPEECH is used in the speech recognition made to recognize words spokken one at a time i.e. discrete word recognition. Each word of the vocabulary is repeated ten times. The discrete word recognition results are given in Table 5-3.

An overall accuracy of about 94% was achieved.

**Continuous Speech Recognition**

As done previously program SPEECH is used in the speech recognition mode. A maximum of 5 seconds of speech input is possible. Various sequences of words in the library were tried. A sample of the results achieved is shown in Table 5-4.

The complete speech recognition system is very flexible. For example to change the vocabulary of the system it is required to change the library file and the output file WTYPE only.
Table 5-3

Discrete Word Recognition Results

<table>
<thead>
<tr>
<th>Word</th>
<th>Correctly identified out of 10 tries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>10</td>
</tr>
<tr>
<td>One</td>
<td>10</td>
</tr>
<tr>
<td>Two</td>
<td>10</td>
</tr>
<tr>
<td>Three</td>
<td>9</td>
</tr>
<tr>
<td>Four</td>
<td>10</td>
</tr>
<tr>
<td>Five</td>
<td>7</td>
</tr>
<tr>
<td>Seven</td>
<td>9</td>
</tr>
<tr>
<td>Eight</td>
<td>10</td>
</tr>
<tr>
<td>Nine</td>
<td>8</td>
</tr>
<tr>
<td>Point</td>
<td>10</td>
</tr>
</tbody>
</table>

accuracy = 93.6%
Table 5-4

Continuous Speech Recognition Results

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Correctly Recognized out of 5 tries</th>
<th>Incorrectly Recognized as:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.67</td>
<td>4</td>
<td>1.671</td>
</tr>
<tr>
<td>123456789.</td>
<td>3</td>
<td>123456185.</td>
</tr>
<tr>
<td>123456749.</td>
<td>4</td>
<td>123456749.</td>
</tr>
<tr>
<td>235.9</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>0.26</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>987654321</td>
<td>3</td>
<td>187654321</td>
</tr>
<tr>
<td>987654331</td>
<td>4</td>
<td>987654331</td>
</tr>
<tr>
<td>4.27</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
The system at this moment is speaker dependent. An attempt was made to make the system speaker independent by adding phoneme strings for different people in the library file. This worked but was abandoned as not being a good solution to the problem. It is possible to make the system speaker independent without any change in the basic system itself. This can be done by a recursive use of program CREATEMP to obtain a final template file which contains all or most of the phoneme sounds in the English language. This can be done over a period of time since program CREATEMP is able to edit previously created template files. This process would have taken a number of months for which there was no time during this thesis effort. It is recommended therefore that an effort be made in the future to use this system for obtaining a good if not ideal template file. This would greatly increase the accuracy and vocabulary of the system.
Bibliography


The ASA-16 is a 28-pin integrated circuit with 4800 equivalent transistors. It provides audio spectrum analysis over the range of intelligibility for speech that is 200 to 7000 Hz.

The analog input to the ASA-16 is 7 volts rms maximum, from a low-output impedance source of 600 ohms or less. The ASA-16 consists of 16 bandpass filters each followed by a halfwave rectifier and a second order low-pass filter with 25-Hz cutoff. The monolithic ASA-16 utilizes NMOS switched-capacitor technology with 100 operational amplifiers to achieve the required audio spectrum analysis. Additionally, this chip contains a 16-channel analog multiplexer and decoder and provides all the necessary timing signals from a single TTL 1-MHz clock. Each bandpass filter center frequency is linearly related to the clock frequency. Clock translation results in spectral translation. The analog multiplexer is addressed via four TTL lines. The analog output of the chip is from a buffer amplifier. This output is suitable for input to a 0 to 5-volts user-supplied analog-to-digital converter (National part number ADC0804).

The input and output signals for the ASA-16 speech preprocessor are shown in figure 1 and listed in table 1.

![Figure 1. ASA-16 Speech Preprocessor Pin Assignments](image-url)
Table 1. ASA-16 Input and Output Signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Pin No.</th>
<th>Signal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel 1 through 16</td>
<td>1 through 4, 11 through 18, 25 through 28</td>
<td>These pins provide output connection for each of the 16 spectrum analyzer channels prior to multiplexing. The high impedance outputs should be loaded with more than 200 kilohms. These low pass filter outputs permit using an external multiplexer and A/D converter such as the National ADC0816.</td>
</tr>
<tr>
<td>Analog Input</td>
<td>6</td>
<td>The speech input is applied to this pin following microphone amplification. The input signal level should not exceed 7 volts rms.</td>
</tr>
<tr>
<td>V_{DD} and V_{SS}</td>
<td>7, 19</td>
<td>Power is supplied to the ASA-16 using these pins. V_{DD} is +10 volts and V_{SS} is -10 volts, ±5 percent.</td>
</tr>
<tr>
<td>Master Clock</td>
<td>8</td>
<td>The master clock is a 1-MHz input signal that synchronizes the ASA-16 logic.</td>
</tr>
</tbody>
</table>
Table 1. ASA-16 Input and Output Signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Pin No.</th>
<th>Signal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplexer Output</td>
<td>24</td>
<td>The ASA-16 on-board analog multiplexer output is available at this pin. The output voltage is 4.5 volts dc, ±10 percent for a 5-volt rms signal at the analog input pin at the center frequency of the corresponding selected multiplexer channel.</td>
</tr>
<tr>
<td>Address (A₀ through A₃)</td>
<td>20 through 23</td>
<td>The control signals applied select the multiplexer channel for output from the ASA-16 on-board analog multiplexer. Pin 21 is the MSB and Pin 20 is the LSB.</td>
</tr>
<tr>
<td>Digital Ground</td>
<td>9</td>
<td>This line should be connected to a low TTL logic level. With this input low, the analog multiplexer output corresponds to the channel specified by the active 4-bit multiplexer address.</td>
</tr>
<tr>
<td>Offset Adjust</td>
<td>5</td>
<td>This pin provides a method for compensation of the ASA-16 offset characteristics.</td>
</tr>
</tbody>
</table>
Appendix C

The transfer function of the preemphasis filter is calculated as follows:

\[
P(s) = \frac{s + 1/T}{s + 1/\alpha T}
\]

since a cutoff frequency of 500Hz is desired. Hence

\[
T = \frac{1}{2 \times 500} = 3.18309 \times 10^{-4} \text{ sec.}
\]

For a gain of 6db/octave an \( \alpha = 0.1 \) is chosen. The component values \( R_1 \), \( R_2 \) and \( C \) are calculated as below.

\[
R_1 C = T = 3.18309 \times 10^{-4}
\]

Assuming \( C = 0.02 \mu F \)

\[
R_1 = \frac{3.18309 \times 10^{-4}}{0.02 \times 10^{-6}} = 15915.494 \Omega
\]

\[
\alpha = \frac{R_2}{R_1 + R_2}
\]
\[ R_2 = \alpha R_1 + \alpha R_2 \]

\[ R_2 = \frac{\alpha R_1}{1 - \alpha} = \frac{0.1 \times 15915.494 \, \Omega}{1 - 0.1} \]

\[ R_2 = 1.768 \times 10^3 \, \Omega \]

Hence the component values are chosen as:

\[ R_1 = 15K \, \Omega \]

\[ R_2 = 1.8K \, \Omega \]

\[ C = 0.02 \mu F \]

with these component values

\[ \alpha = \frac{1.8 \times 10^3}{(1.8 \times 10^3) + (15 \times 10^3)} = 0.1071428 \]

\[ T = 15 \times 10^3 \times 0.02 \times 10^{-6} = 3 \times 10^{-4} \, \text{sec.} \]

Hence the transfer function is:

\[ Ps = \frac{s + 3333.33}{s + 31111.128} \]
Appendix D

The transfer function of the low pass filter is calculated as follows:

\[ H(s) = \frac{-H_0 \omega_0^2}{s^2 + \omega_0 s + \omega_0^2} \]

where:

\[ H_0 = 10^{A/20} \quad , \quad A = \text{passband gain in dB} \]

\[ \omega_0 = 2F \quad , \quad F = \text{cutoff frequency in Hertz} \]

\[ \alpha = \text{peaking factor} \]
For our purpose:

\[ A = 17 \text{dB} \]
\[ F = 7000 \text{Hz} \]
\[ a = 1 \]

which gives

\[ H_0 = 7.0794 \]
\[ \omega_0 = 43982.297 \text{ rad/sec.} \]

Assuming \( C_1 = 0.02 \text{ F} \), the component values of the low pass filter are calculated as:

\[ C = \frac{4(1 + H_0)C_1}{a^2} = 0.6464 \mu\text{F} \approx 0.6 \mu\text{F} \]

\[ R_2 = \frac{a}{4\mu FC_1} = 568.41 \approx 560 \Omega \]

\[ R_1 = \frac{R_2}{H_0} = 80.29 \approx 82 \Omega \]

\[ R_3 = \frac{R_2}{H_0 + 1} = 70.35 \approx 68 \Omega \]

These values of components gives a voltage transfer function for the low pass filter as:

\[ \frac{E_o(s)}{E_1(s)} = \frac{-1.4945 \times 10^{10}}{s^2 + 47811.198s + 2.18837 \times 10^9} \]
Appendix E

Title: SPEECH.FR
Author: Capt. Ajmal Hussain
Date: Aug 83

Function:
This program does continuous speech recognition. It needs as input a template file, the corresponding distance matrix file, and library file. A maximum of 5 seconds of speech input is taken and typed out on the video console (H19 terminal) in reverse video.

Environment:
This is a Fortran V program that has been designed to run on a mapped-RDOS Eclipse S/250 minicomputer equipped with a model 4331 single board converter.

Compile command:
FORTRAN SPEECH

Load command:
RLDR/P 2/K SPEECH REDBUF SREDM FINDWORD WTYPE SAMCONFIG7 @SAMLIB8

Comments:
The input files required by the program are:-
Template file ---------- TEMP20.DA
Distance matrix file ---- MAT20.DA
Library file ----------- LIB20.DA

The hardware should be connected to the Eclipse A/D/A converter.

User's guide:
The hardware is connected to the Eclipse A/D/A converter as shown in the Thesis "Limited Continuous Speech Recognition by Phoneme Analysis". The D/A converter is clocked externally with a 400 Hz TTL signal.
Program SPEECH is run. It comes up with the main menu on the CRT as follows:-

Program SPEECH. SV executing

Please select which operation will be performed
0: change variables
1: speech conversions
2: read templates from file
3: print phonemes found
4: read distance matrix from file
5: read library
6: find word
7: exit
selection:*

Select operation " 0 ". For TEMP20.DA the variable values are:-
distance limit: 0.01
penalty: 8
min. word length: 1
max. word length: 9
select " 1 " to find words
From the main menu next select operation "2" and read in template file "TEMP20.DA".

From the main menu next select operation "4" and read in distance matrix file "MAT20.DA".

From the main menu next select operation "5" and read in library file "LIB2001.DA", or your own library file.

From the main menu next select operation "1". The following message will be displayed on the screen:

Speech input time in seconds (max. 5 secs.) or '0' for main menu =

Type the desired amount of time and press "carraige return" to start the conversion. Speak into the microphone for that amount of time. The recognized speech will be displayed on the CRT in reverse video and the system will ask you for the next amount of time.

INITIALIZATION

DEVICE=21

CALL DSTRT(IER) ; always initialize device
IF (IER.NE.1) CALL ERROR("DSTRT error") ; if 'error' display error ; number

EXTERNAL IDS21 ; A/D device
EXTERNAL IDS23 ; D/A device required by SAM
COMMON / IBUFF / IDATA3(2010) ; A/D data buffer
COMMON / IBUFF / ITEMP(1280) ; D/A data buffer required by SAM

INTEGER IORBA(16), IPHON(125), DEVICE, LIB(256), J, K, I, M, N, COUNT
INTEGER P, Q, CTHRD(10), WORD(125), S, T, U, V, W, REJ(10, 10), X, Y, Z, FLAG
REAL HAG(125), MAT(2432), LDISTOT(5), PEN
DOUBLE PRECISION REAL TEMP, TEMPI, TEMP3

84
MAIN MENU

10 TYPE "<CR>
   *Program SPEECH. SV executing"
   ACCEPT "<CR>
   *Please select which operation will be performed,<CR>
   * 0: change variables<CR>
   * 1: speech conversions<CR>
   * 2: read templates from file<CR>
   * 3: print phonemes found<CR>
   * 4: read distance matrix from file<CR>
   * 5: read library<CR>
   * 6: find word<CR>
   * 7: exit<CR>
   *selection: ", IOP
   go to code for selection made
   IF (IOP.EQ.0) GO TO 11
   IF (IOP.EQ.1) GO TO 20
   IF (IOP.EQ.2) GO TO 60
   IF (IOP.EQ.3) GO TO 110
   IF (IOP.EQ.4) GO TO 140
   IF (IOP.EQ.5) GO TO 50
   IF (IOP.EQ.6) GO TO 70
   IF (IOP.EQ.7) GO TO 160
   WRITE (10,1) ;display message if selection made
   ;from other than given options
   GO TO 10
   I FORMAT ("<CR><CR><CR><33><160>
   *Please make selections only from the given options
   <*33><161>")
C ACCEPT VALUES OF VARIABLES

11 TYPE LDIS, PEN, LEN1, LEN2 ; display current variable values

ACCEPT"<CR>" distance limit: ", LDIS ; distance between adjacent phonemes, less than which they are considered same

ACCEPT"<CR>" penalty: ", PEN ; penalty to be added to error due to differences in number of phonemes in input word and library word

ACCEPT"<CR>" min. word length: ", LEN1 ; number of phonemes in the smallest word in library

ACCEPT"<CR>" max. word length: ", LEN2 ; number of phonemes in the longest word in library

ACCEPT"<CR>" ; '1' for speech recognition
* 1: speech recognition<CR> ; '2' just phoneme analysis
* 2: just phoneme analysis<CR>
* selection: ", FWORD
GO TO 10
C******************************************************************************
C A/D CONVERSION
C******************************************************************************

20 IDATA1 = 61700K ; a/d 16 channels starting with channel 1 on to
; channel 16 cyclicly, using external clock

22 ACCEPT "<CR>
  *Speech input time in seconds (max. 5 secs) = " , IDATA2
  IF (IDATA2 .EQ. 0) GO TO 10
  IF ((IDATA2 .LT. 6) AND (IDATA2 .NE. 0)) GO TO 25
  TYPE "<CR><CR><33><160>
  *Time input should be less than 5 secs and greater than zero
  <33><161>"
  GO TO 22

25 IDATA2 = IDATA2 * 400
C start conversion

27 CALL DOITW(IORBA, IDS21, B, IDATA1, IDATA2, IDATA3, IER)
C display error message and number if error occurred in A/D conversion
C else display no errors reported

TYPE "<7><7><7><CR>
*<33><160>
*Conversion operation completed"

IF (IER .NE. 1) TYPE "DOIT error ", IER
IF (IORBA(14) .NE. 40000K) TYPE "IORBA(14) return ", IORBA(14)
IF (IER .EQ. 1 AND. IORBA(14).EQ.40000K) TYPE "No errors reported"
TYPE "<33><161>"
GO TO 9
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A
READ TEMPLATE FILE

CALL REDBUF(ITEMP,1280,5)
TYPE "<7><7><7><CR><33><160>
*Templates read into buffer
<33><161>
GO TO 10

NORMALIZE INPUT BUFFER

TEMP = 0
J = 1
K = 1
L = 1
DO 95 I = 1,(IDATA2/16)
TEMP = 0

DO 92 J = K, (K+15)
IDATA3(J) = IDATA3(J) - IDATA3(J-K+1)
IF (ABS(IDATA3(J)).LT.200) IDATA3(J)=0
TEMP = TEMP + FLOAT(IDATA3(J))**2
92 CONTINUE

TEMP = (SQRT(TEMP)/32000)
MAG(I) = TEMP

DO 94 J = K, (K+15)
IDATA3(J) = IDATA3(J)/TEMP
94 CONTINUE

K = K+16
95 CONTINUE
C=================================================================================
C PHONEME EXTRACTION AND COMPRESSION
C=================================================================================

80 DO 88 I = 1, 125 ; clear phoneme string
IPHON(I) = 0
88 CONTINUE

C initialize variables
TEMP = 0
I = 1
M = 0
TEMP1 = 9.0E 60

DO 87 L = 1, IDATA2, 16 ; analyze input buffer a phoneme at a time
IF (MAG(I).LT.0.25) GO TO 801 ; if energy value of phoneme is less than 0.25, consider it as noise and assign it phoneme number zero

DO 86 K = 1, (ITEMP(1121)*16), 16 ; compare with all phonemes in template file
C DO 89 N = -1, 1
N = 0

C compare each phoneme of input buffer with all phonemes in template file element by element. TEMP accumulates the error for each phoneme. The phoneme in the template with the smallest error value is chosen as the recognized phoneme and it's phoneme number added to the phoneme string IPHON.

DO 85 J = L, (L+15)
TEMP = TEMP + (FLOAT(IDATA3(J+N)) - FLOAT(ITEMP(K+M)))*8
M = M + 1
85 CONTINUE

IF (TEMP.GT.TEMP1) GO TO 82
TEMP1 = TEMP
IPHON(I) = (K+15)/16
82 TEMP = 0
M = 0
89 CONTINUE
86 CONTINUE
801 TEMP1 = 9.0E 60
I = I + 1
87 CONTINUE
compress phoneme string by combining adjacent phonemes which are same or are closer than variable LDIS from each other. Two or more adjacent zero's are represented by a single zero. One zero alone is ignored as an error. The length of the compressed phoneme string is stored in variable 'J'.

DO 806 K = 1,5
DO 809 I = 1,((IDATA2/16)-1)
IF (IPHON(I).EQ.0) GO TO 807
IF (IPHON(I+1).EQ.0) GO TO 807
IF (IPHON(I).EQ.IPHON(I+1)) GO TO 807
N = IPHON(I)
P = IPHON(I+1)
IF (N.GT.P) GO TO 810
G = N
N = P
P = G
810 IF (MAT(((N+(N-1))/2)+P).GE.LDIS) GO TO 807
IF (IPHON(I).LT.IPHON(I+1)) IPHON(I) = IPHON(I+1)
IF (IPHON(I).GT.IPHON(I+1)) IPHON(I) = IPHON(I+1)
807 CONTINUE
809 CONTINUE
806 CONTINUE

J = 1
DO 805 I = 1,(IDATA2/16)
IF (((IPHON(I).EQ.0).AND.((J.EQ.1)) GO TO 805
IF (((IPHON(I).EQ.0).AND.((IPHON(I-1).NE.0)) GO TO 805
IF (IPHON(I).EQ.IPHON(I+1)) GO TO 805
IPHON(J) = IPHON(I)
J = J + 1
805 CONTINUE
DO 808 L = J,(IDATA2/16)
IPHON(L) = 0
808 CONTINUE
PRINT PHONEME STRING

110 WRITE(12,114) ; print compressed phoneme string
L = 1
DO 112 I = 1,J
WRITE(12,111) IPHON(I)
IF (IPHON(I).EQ.0) GO TO 122
IF (L.NE.10) GO TO 116
122 WRITE(12,114)
L = L + 1
116 CONTINUE
WRITE(12,114)
WRITE(12,121)
121 FORMAT(20X)
L = 1
M = MAT(2416)
DO 113 I = 1,(J-1)
IF(IPHON(I).GE.IPHON(I+1)) GO TO 119
WRITE(12,117) MAT(((IPHON(I)+1)*(IPHON(I+1)-1))/2)+IPHON(I))
GO TO 120
119 WRITE(12,117) MAT(((IPHON(I)+1)*(IPHON(I+1)-1))/2)+IPHON(I+1))
120 IF(L.NE.10) GO TO 118
WRITE(12,114)
L = 0
118 L = L + 1
115 CONTINUE
WRITE(12,114)
117 FORMAT(";","Q10.1,I")
WRITE(12,114)
114 FORMAT(1X)
111 FORMAT(";","7X,I3.I")
IF(FWORD.EQ.1) GO TO 70
GO TO 20

READ DISTANCE MATRIX FILE

140 CALL SREDM(MAT,2432)
GO TO 10

READ LIBRARY FILE

50 CALL REDBUF(LIB,256,1)
GO TO 10
CONTINUOUS SPEECH RECOGNITION

70  DO 188 I = 1, 10  ! clear reject matrix
    DO 189 K = 1, 10
    REJ(I, K) = 0
    189  CONTINUE
188  CONTINUE

71  DO 187 I = 1, 5  ! clear total error matrix
    TOT(I) = 0
187  CONTINUE

72  R = 1  ! initialize string length variables
73  S = 1

74  DO 175 I = 1, 125  ! clear temporary word register
    WORD(I) = 0
175  CONTINUE

75  IF (R.GE.(J+1)) GO TO 20
    IF (IPHON(R)EQ.0) GO TO 170
    WORD(S) = IPHON(R)
    R = R + 1
    S = S + 1
    GO TO 79
170  IF (S.GT.LEN1) GO TO 178
    R = R + 1
    GO TO 79

C  start word recognition

C  discrete word recognition
    CALL FINDWORD(WORD, LIB, TEMP, MAT, PEN, L)
C  display recognized word
    CALL WTYPE(L, "001223445567778899.. 0012234455667777899..")
178  IF (S.GT.LEN2) GO TO 177
    S = 1
    GO TO 172

C  get next string
C continuous word recognition

177 V = 1
W = 1
X = 1
Y = 1
FLAG = 0

171 DO 176 I = 1,10
TWORD(I) = 0
176 CONTINUE

TEMP3 = 9.0E 60

DO 173 I = 1,10
TWORD(I) = WORD(V)
V = V + 1
IF (I.LE.2) GO TO 173

C get error value assuming word to be correct

CALL FINDWORD(TWORD,LIB,TEMP, MAT, PEN, L)
IF (Y.LE.1) GO TO 180

C reject word if same as previous try

IF ((L.EQ.REJ((Y-1),(Y-1))).AND.(FLAG.EQ.1)) GO TO 186

180 IF (TEMP.GE TEMP3) GO TO 173

TOT(Y) = TOT(Y) + TEMP3
X = X + 1
IF ((W+U).LT.S) GO TO 174
IF (Y.LE.4) GO TO 179

TOT(Y) = TOT(Y)/X
S = 1
TEMP3 = 9.0E60

DO 181 I = 1,Y
IF (TOT(I).GE. TEMP3) GO TO 181
TEMP3 = TOT(I)
Z = I
181 CONTINUE

REJ(Y,X) = T
TOT(Y) = TOT(Y) + TEMP3
X = X + 1
IF ((W+U).LT.S) GO TO 174
IF (Y.LE.4) GO TO 179

TOT(Y) = TOT(Y)/X
S = 1
TEMP3 = 9.0E60

DO 181 I = 1,Y
IF (TOT(I).GE. TEMP3) GO TO 181
TEMP3 = TOT(I)
Z = I
181 CONTINUE

93
display word string recognized

DO 182 I = 1, 10
   CALL WTYPE(REJ(I, I), '00112233445566778899..00112233445566778899..')
   CONTINUE
182   FORMAT('+', I3, I)
       GO TO 79
174   W = W + U
       V = W - 1
       GO TO 171
179   TOT(Y) = TOT(Y)/X
       Y = Y + 1
       X = 1
       FLAG = 1
       V = 1
       W = 1
       GO TO 171

C-----------------------------------------------------------------------------------

160   CALL EXIT
       :stop program

C-----------------------------------------------------------------------------------
Title: CREATEMP.FR
Author: Capt. Ajmal Hussain
Date: Aug 83

Function:
This program takes a speech input, normalizes it, deletes unwanted phonemes, compresses the buffer and stores it in a file to be used as a template file.

Environment:
This is a Fortran V program that has been designed to run on a mapped-RDOS Eclipse S/250 minicomputer equipped with a model 4331 single board converter.

Compile command:
FORTRAN CREATEMP

Load command:
RLDR/P 2/X CREATEMP SREDT NEWSCR SSEEIT PAPER^ SETUP HEADWR WRTTEMP SAMCFGIC3 SAMLIB@

Comments:
The hardware should be connected to the Eclipse A/D/A converter.
The program can be used to create a new template file or to edit an existing template file.

User's guide:
The hardware is connected to the Eclipse A/D/A converter as shown in the Thesis "Limited Continuous Speech Recognition by Phoneme Analysis". The A/D converter is clocked externally with a 400 Hz TTL signal.
Program CREATEMP is run. It displays the main menu on the CRT as follows:

Program CREATEMP.SV executing
Please select which operation will be performed.
1: A/D conversions
2: data buffer display
3: data buffer print
4: normalize
5: compare phonemes
6: delete unwanted phonemes
7: compress templates
8: template write to file
9: read template from file
10: delete specified phonemes
11: exit selection:

Select operation " 1 ". The program will ask for the speech input time (a maximum of 20 seconds of speech is possible). After pressing carriage return input the required speech via the microphone. Any errors occurring during the A/D conversion will be displayed, else " no errors reported " message is displayed and the program returns to the main menu.
Select operation "4" to normalize the input buffer and return to the main menu.

Select operation "5". This compares all the phonemes in the input buffer with each other and prints all the phoneme numbers and their closest match and the distance between them.

Select operation "6". This deletes all phonemes with energy value below a given limit.

Select operation "10" to delete those phonemes which are too close to each other.

Select operation "7" to compress the data buffer. This must be done before storing the data buffer in a template file.

Select operation "8" to write the buffer into a template file.

The rest of the operations are self explanatory and are used to analyze that everything is working well.

EXTERNAL ID821 ; A/D device
EXTERNAL ID823 ; D/A device required by SAM
COMMON / IBUFF / IDATA3(16384) ; A/D data buffer
COMMON / IBUF0 / IWAST ; D/A data buffer required by SAM

INTEGER IORBA(16), IDATA5(500), DEVICE, J, K, I, L, M
DOUBLE PRECISION REAL TEMP, TEMPI
REAL DIFF(500), IDATA6(500)

DEVICE=21
IDATA2=16000
CALL DSTRT(IER) ; always initialize device
IF (IER.NE.1) CALL ERROR("DSTRT error")

clear the screen

CALL NEWSCR
C MAIN MENU

C

10 TYPE "<CR>
  *Program CREATEMP.SV executing"

ACCEPT"<CR>
  *Please select which operation will be performed.<CR>
  * 1: A/D conversions<CR>
  * 2: data buffer display<CR>
  * 3: data buffer print<CR>
  * 4: normalize<CR>
  * 5: compare phonemes<CR>
  * 6: delete unwanted phonemes<CR>
  * 7: compress templates<CR>
  * 8: template write to file<CR>
  * 9: read template from file<CR>
  * 10: delete specified phonemes<CR>
  * 11: exit<CR>
  * selection: ", IOP

IF (IOP.EQ.1) GO TO 20
IF (IOP.EQ.2) GO TO 50
IF (IOP.EQ.3) GO TO 50
IF (IOP.EQ.4) GO TO 90
IF (IOP.EQ.5) GO TO 100
IF (IOP.EQ.6) GO TO 110
IF (IOP.EQ.7) GO TO 120
IF (IOP.EQ.8) GO TO 60
IF (IOP.EQ.9) GO TO 130
IF (IOP.EQ.10) GO TO 210
IF (IOP.EQ.11) GO TO 80
WRITE (10,1)
GO TO 1

1 FORMAT ("<CR><CR><CR><33><160>
  *Please make selections only from the given options
   <33><161>")
A/D CONVERSION

20 IDATA1 = 61700K

; A/D 16 channels starting with channel 1 on to
; channel 16 cyclicly, using external clock

22 ACCEPT"<CR>

; Speech input time in seconds (max. 20 secs) = ", IDATA2
IF ((IDATA2.LT.21).AND.(IDATA2.NE.0)) GO TO 25
TYPE "<CR><33><160>

; Time input should be less than 20 secs and greater than zero
"<33><161>"
GO TO 22

25 IDATA2 = IDATA2*400
CALL DOITH(IORBA, IDS21, B, IDATA1, IDATA2, IDATA3, IER)

TYPE "<7><7><7><CR>

; Conversion operation completed"

IF (IER.NE.1) TYPE "DOIT error ", IER
IF (IORBA(14).NE.4000K) TYPE "IORBA(14) return ", IORBA(14)
IF (IER.EQ.1 .AND. IORBA(14).EQ.4000K) TYPE "No errors reported"
GO TO 10

DATA BUFFER DISPLAY/PRINT

50 CALL SETUP(IFOR, IOP, ISTART, ISTOP)

; get the parameters specifying
; the section of data buffer to be
; worked with.

Display the user requested section of data buffer.

IF (IOP.EQ.2) CALL SEEIT(IFOR, ISTART, ISTOP, IDATA3, 16384)

Print the header and the user requested section of data buffer.

IF (IOP.EQ.3) CALL HEADER(DEVICE, FIRST, LAST, IDATA2, IER, IORBA, CLOCK)
IF (IOP.EQ.3) CALL PAPER(IFOR, ISTART, ISTOP, IDATA3, IDATA2)
GO TO 10
WRITE TEMPLATE TO FILE

CALL WRTTEMP(IDATA3, 16384); let the user write specified sections of data buffer to file
GO TO 10

NORMALIZE DATA BUFFER

TEMP = 0
J = 1
K = 1
L = 1

DO 95 I = 1, (IDATA2/16)
   TEMP = 0

DO 92 J = K, (K+15)
   IDATA3(J) = IDATA3(J) - IDATA3(J-K+1)
   IF (ABS(IDATA3(J)) .LT. 200) IDATA3(J) = 0
   TEMP = TEMP + (FLOAT(IDATA3(J))**2)
92 CONTINUE

TEMP = (SGRT(TEMP)/32000)
IDATA6(I) = ABS(TEMP)
DO 94 J = K, (K+15)
   IDATA3(J) = FLOAT(IDATA3(J))/TEMP
94 CONTINUE

K = K + 16
95 CONTINUE

TYPE "<7><7><7><CR><33><160>
*Normalization operation completed
<33><161>"
GO TO 10
COMPARE PHONEMES

100 TEMP = 0
TEMP1 = 9.0E60
DO 104 J = 1, IDATA2, 16
DO 102 K = 1, IDATA2, 16
IF (J .EQ. K) GO TO 103
DO 101 L = 0, 15
TEMP = TEMP + (FLOAT(IDATA3(J+L)) - FLOAT(IDATA3(K+L)))**8
101 CONTINUE
IF (TEMP .GE. TEMP1) GO TO 103
TEMP1 = TEMP
IDATA5(INT((J+15)/16)) = INT((K+15)/16)
DIFF(INT((J+15)/16)) = TEMP
103 TEMP = 0
102 CONTINUE
TEMP1 = 9.0E60
104 CONTINUE
TEMP = 0
DO 107 I = 1, (IDATA2/16)
IF (DIFF(I) .GT. TEMP) TEMP = DIFF(I)
107 CONTINUE
DO 108 I = 1, (IDATA2/16)
DIFF(I) = (DIFF(I)/TEMP)*100
108 CONTINUE
DO 105 I = 1, (IDATA2/16)
WRITE(12,106) I, IDATA5(I), DIFF(I), IDATA6(I)
105 CONTINUE
106 FORMAT(1X,F5.5,F15.5,F5.5,F11.5)
CLOSE 12

TYPE "<7><7><7><CR><33><160>
Comparison completed
<33><161>""
GO TO 10
C************DELETE UNWANTED PHONEMES************

110  K = 0
     DO 116 L=1,(IDATA2/16)
     IF (IDATA6(L).GT.0.25) GO TO 116
     I = ((L+15)-15)
     DO 111 J = I, (I+15)
     IDATA3(J) = 32000
     111  CONTINUE
     K = K + 1
     116  CONTINUE
     TYPE "<7><7><CR><33><160>
     *number of templates:* ,((IDATA2/16)-K)
     TYPE "<33><161>"
     IDATA3(IDATA2+1) = ((IDATA2/16)-K)
     GO TO 10

C******************************************************************************
CCOMPRESS DATA BUFFER
C******************************************************************************

120  J = 1
     DO 122 I = 1, IDATA2, 16
     IF ((IDATA3(I).EQ.32000).AND.(IDATA3(I+1).EQ.32000)) GO TO 122
     DO 123 K = 0, 15
     IDATA3(J) = IDATA3(I+K)
     J = J + 1
     123  CONTINUE
     122  CONTINUE
     DO 122 I = (IDATA3(IDATA2+1)*16), IDATA2
     IDATA3(I) = 0
     122  CONTINUE
     TYPE "<7><7><7><CR><33><160>
     *Templates compressed*
     <33><161>"
     GO TO 10
C**********************************************************************************************
C
C READ TEMPLATE FILE
C
C**********************************************************************************************

130 CALL SREDT(IDATA3,3400) ;let the user write specified sections
  TYPE "<7><7><7><CR><33><160>
  *Templates read into buffer
  <33><161>"
  IDATA3(IDATA2+1) = IDATA3(1121)
  GO TO 10

C**********************************************************************************************
C
C DELETE SPECIFIED PHONEMES
C
C**********************************************************************************************

210 K = 0
214 ACCEPT"<CR>
  *Template number to delete or zero to end : ",IOP
  IF ((IOP.LT.501).AND.(IOP.GT.0)) GO TO 212
  IF (IOP.EQ.0) GO TO 215
  TYPE "<CR><33><160>
  *Template number should be between 1 and 500
  <33><161>"
  GO TO 214

212 I = ((IOP*16)-15)
  DO 211 J = I, (I+15)
  IDATA3(J) = 32000
211 CONTINUE
  K = K + 1
  TYPE "<7><7><7><CR><33><160>
  *Template deleted
  <33><161>"
  GO TO 214

215 TYPE "Number of templates: ",IDATA3(IDATA2+1)-K
  IDATA3(IDATA2 + 1) = IDATA3(IDATA2+1)-K
  GO TO 10

C**********************************************************************************************
B0 CALL EXIT
END
C**********************************************************************************************
Title: DISMAT1.FR
Author: Capt. Ajmal Hussain
Date: Aug 83

Function:
This program takes a template file, calculates the distances between each template and stores it in another file.

Environment:
This is a Fortran V program that has been designed to run on a mapped-RDOS Eclipse S/250 minicomputer.

Compile command:
FORTRAN DISMAT1

Load command:
RLDR DISMAT1 SETUP NEWSCR SEEIT SREDM SWRM SREDT- SEEMAT @FLIB@

Comments:
A specified template file is read from the disk. It's distance matrix is calculated and can be stored in another disk file or printed out.

INTEGER J, K, I, L, PHON(1130)
DOUBLE PRECISION REAL TEMP, TEMPI
REAL MAT(2432)
CALL NEWSCR
**MAIN MENU**

10 TYPE "<CR>
*Program DISMATS.BV executing*

ACCEPT "<CR>
*Please select which operation will be performed.<CR>

1: read templates from file<CR>
2: form distance matrix<CR>
3: display distance matrix<CR>
4: print distance matrix<CR>
5: distance matrix write to file<CR>
6: read distance matrix from file<CR>
7: display templates<CR>
8: give distance between two phonemes<CR>
9: exit<CR>

*/selection: "", IOP

IF (IOP.EQ. 1) GO TO 20
IF (IOP.EQ. 2) GO TO 30
IF (IOP.EQ. 3) GO TO 40
IF (IOP.EQ. 4) GO TO 50
IF (IOP.EQ. 5) GO TO 60
IF (IOP.EQ. 6) GO TO 70
IF (IOP.EQ. 7) GO TO 80
IF (IOP.EQ. 8) GO TO 90
IF (IOP.EQ. 9) GO TO 90

WRITE (10,1)
GO TO 10

1 FORMAT ("<CR><CR><CR><<33><160>
*Please make selections only from the given options
<33><161>")

**READ TEMPLATE FILE FROM DISK**

20 CALL SREDT(PHON,1130)
TYPE "<7><7><7><<33><160>
*Templates read into buffer
<33><161>
GO TO 10
C******************************************************************************
C
C FORM DISTANCE MATRIX
C
C******************************************************************************

30 TEMP = 0
TEMP1 = 0
I = 1
DO 31 J = 1, (PHON(1121)*16), 16
DO 32 K = 1, (PHON(1121)*16), 16
IF (K.GT.J) GO TO 35
DO 33 L = 0, 15
TEMP = TEMP + (FLOAT(PHON(J+L)-PHON(K+L)))**8
33 CONTINUE
IF(TEMP.GT.TEMPI) TEMPI = TEMP
I = I + 1
35 TEMP = 0
32 CONTINUE
31 CONTINUE
TYPE TEMPI
TEMPI = TEMPI/10000
DO 34 I = 1, ((PHON(1121)*(PHON(1121)-1))/2)+PHON(1121)
MAT(I) = MAT(I)/TEMPI
34 CONTINUE
MAT(2416) = PHON(1121)

TYPE "<7><7><7><CR><33><160>
Distance matrix formed
"<33><161>"
GO TO 10

C******************************************************************************
C
C DISPLAY DISTANCE MATRIX
C
C******************************************************************************

40 CALL SEEMAT(MAT,4910)
GO TO 10
PRINT DISTANCE MATRIX

50 ACCEPT"<CR>Matrix number is : ", IOP
   L = PHON(1121)
   WRITE(12,51) IOP
51 FORMAT(50X,"MAT",12)
   WRITE(12,58)
   WRITE(12,58)
   IF (PHON(1121).LE.40) GO TO 52
   L = 40
   WRITE(12,55)
55 FORMAT("=",3X,Z)
52 DO 53 I = 1,L
   WRITE(12,54) I
53 CONTINUE
54 FORMAT("=",13,Z)
56 CONTINUE
58 FORMAT(IX)
IF (PHON(1121).LE.40) GO TO 500
   WRITE(12,59)
59 FORMAT("1")
   WRITE(12,51) IOP
   WRITE(12,58)
   WRITE(12,58)
   WRITE(12,55)
   DO 501 I = 1,40
   WRITE(12,54) I
501 CONTINUE
   K = 820
   DO 502 I = 41, PHON(1121)
   WRITE(12,58)
   WRITE(12,54) I
   DO 503 J = 1,40
   WRITE(12,54) (INT((MAT((I*(I-1)/2) + J)/100))
503 CONTINUE
502 CONTINUE
WRITE(12, 59)
WRITE(12, 51) IOP
WRITE(12, 58)
WRITE(12, 59)
DO 505 I = 41, PHON(1121)
WRITE(12, 54) I
505 CONTINUE
DO 506 I = 41, PHON(1121)
WRITE(12, 58)
WRITE(12, 59) I
DO 507 J = 41, PHON(1121)
IF(J.GT.1) GO TO 507
WRITE(12, 54) (INT(MAT((I*(I-1)/2) + J))/100)
507 CONTINUE
506 CONTINUE
500 TYPE "<7><7><7><CR><33><160>
*Distance matrix printed
<*33><161>"
GO TO 10

C*************************************************************************
C DISTANCE MATRIX WRITE TO FILE
C*************************************************************************
60 CALL SWRITE(MAT, 2432) 'let the user write specified sections
TYPE "<7><7><7><CR><33><160>
*Distance matrix written to file
 <*33><161>"
GO TO 10

C*************************************************************************
C READ DISTANCE MATRIX FROM FILE
C*************************************************************************
70 CALL SREDM(MAT, 2432)
PHON(1121) = MAT(2416)
TYPE "<7><7><7><CR><33><160>
*Distance matrix read into buffer
<*33><161>"
GO TO 10
C-----------------------------------------------
C           DISPLAY TEMPLATE FILE
C-----------------------------------------------

80 CALL SETUP(IFOR,2,ISTART,ISTOP) ; get the parameters specifying
    the section of data buffer to be worked with.

C          Display the user requested section of data buffer.
C          CALL SEEIT(IFOR,ISTART,ISTOP,PHON,1130)
          GO TO 10

C-----------------------------------------------
C           DISPLAY DISTANCE BETWEEN SPECIFIED PHONEMES
C-----------------------------------------------

100 jce "<CR>
   *first phoneme: " , I
   ACCEPT "  
   *second phoneme: " , J
   L = MAT(2416)
   IF((I.EQ.0).OR.(J.EQ.0)) GO TO 10
   IF((I.GT.L).OR.(J.GT.L)) GO TO 100
   IF(I.GE.J) GO TO 102
   K = I
   I = J
   J = K
102 WRITE(10,101) MAT(((I*(I-1))/2) + J)
101 FORMAT(1X,"<CR>distance is : ",012.5)
           GO TO 100

C-----------------------------------------------

90 CALL EXIT
END
C-----------------------------------------------
Title: CLIB.FR
Author: Capt. Ajmal Hussain
Date: Aug 83

Function:
This program creates a new library or edits an existing library

Environment:
This is a Fortran V program that has been designed to run
on a mapped-RDOS Eclipse S/250 minicomputer.

Compile command:
FORTRAN CLIB

Load command:
RLDR CLIB NEWSCR SETUP SEEIT REDBUF WRBUF $FLIB$

Comments:
The phoneme string representations of words to be recognized are
stored in a library file. Each word can have a maximum of 10
phonemes.

INTEGER J,K,I,L,LIB(256)

CALL NEWSCR

CALL NEWSCR

DO 11 I = 1,256
LIB(I) = 0
11 CONTINUE
C
C
C

10 TYPE "<CR>
  *Program CLIB.SV executing"

ACCEPT "<CR>
*Please select which operation will be performed.<CR>
  1: read library from file<CR>
  2: form library<CR>
  3: display library<CR>
  4: library write to file<CR>
  5: change library value<CR>
  6: print library<CR>
  7: exit<CR>
*selection: ", IOP

IF (IOP.EQ.1) GO TO 20
IF (IOP.EQ.2) GO TO 30
IF (IOP.EQ.3) GO TO 40
IF (IOP.EQ.4) GO TO 50
IF (IOP.EQ.5) GO TO 60
IF (IOP.EQ.6) GO TO 80
IF (IOP.EQ.7) GO TO 70
WRITE (10,1)
GO TO 10
1 FORMAT ("<CR><CR><CR><33><160>
*Please make selections only from the given options
*<33><161>")

C
C
C

20 CALL REDBUF(LIB,256,1)
GO TO 10
30 ACCEPT "starting position: ", I
   TYPE "to return to main menu give a phoneme value greater than 99"

31 TYPE "position: ", I
   ACCEPT "value: ", K
   IF (K .GE. 100) GO TO 10
   LIB(I) = K
   I = I + 1
   GO TO 31

40 CALL SETUP(IFOR, 2, ISTART, ISTOP)
   CALL SEEIT(IFOR, ISTART, ISTOP, LIB, 256)
   GO TO 10

50 CALL WRBUF(LIB, 256, 1)
   GO TO 10

60 TYPE "to return to main menu give a position of 0"

61 ACCEPT "<CR>position: ", I
   IF (I .EQ. 0) GO TO 10
   TYPE "old value: ", LIB(I)
   ACCEPT "<CR>new value: ", K
   LIB(I) = K
   GO TO 61
C
C PRINT LIBRARY
C

80  L = 1
   DO 81 I = 1, 220
   WRITE(12, 111) LIB(I)
   IF (L.NE.10) GO TO 82
   WRITE(12, 114)
   L = 0
82  L = L + 1
81  CONTINUE

   WRITE(12, 114)
   WRITE(12, 121)

111  FORMAT("*", 7X, 13.2)
114  FORMAT(1X)
121  FORMAT(20X)
   GO TO 10

C

70  CALL EXIT
END
Function:
This routine compares a phoneme string with word strings in a library based upon a distance matrix to give the word in the library which is the best match.

Compile command:
FORTRAN FINDWORD.FR

Comments:
The variables IPHON is the phoneme string array, LIB is the library array, TEMP3 returns the error value for the word matched, MAT is the distance matrix array, PEN is the penalty to be added for differences in the number of phonemes in the string and in a word in the library, L returns the number of the word matched.

SUBROUTINE FINDWORD(IPHON, LIB, TEMP3, MAT, PEN, L)
INTEGER I, K, L, M, IPHON(125), LIB(256)
DOUBLE PRECISION REAL TEMP, TEMPl, TEMP3
REAL MAT(2432), PEN
70 TEMP3 = 9.0E 60
TEMP = 0
COUNT = 0
C start comparison
DO 71 M = 1, 220, 10
    :library has 22 words, each a maximum of 10 phonemes long
DO 72 K = -1, 1
    :shift phoneme string one phoneme left, none and one phoneme right to account for error in first phoneme string
DO 73 I = 1, 10
    :compare phoneme at a time for each word in library
IF ((I+K).EQ.0) GO TO 73
    :skip first phoneme when string shifted left one phoneme
C if both phonemes zero error value unchanged
IF ((LIB(M+I-1).EQ.0).AND. (IPHON(I+K) EQ. 0)) GO TO 73
C if both phonemes not zero add distance between phonemes to error value
IF ((LIB(M+I-1).NE.0).AND. (IPHON(I+K).NE. 0)) GO TO 74

113
C if one phoneme zero only add penalty to error value

```
C TEMP1 = TEMP+PEN
GO TO 75

74 N = IPHON(I+K)
P = LIB(M+I-1)
IF(N .GE. P) GO TO 76
Q = N
N = P
P = 0

76 TEMP1 = MAT(((N*N-1))/2)+P)

75 TEMP = TEMP + TEMP1
COUNT = COUNT + 1

73 CONTINUE
TEMP = TEMP/COUNT
IF(TEMP. GT. TEMP3) GO TO 77
TEMP3 = TEMP
L = M

77 TEMP = 0
COUNT = 0
72 CONTINUE
71 CONTINUE
RETURN
END
```

C---------------------------------------------------------------
Title: Header
Author: Lt. Allen
Date: Dec 82

Function:
This routine prints on the printer a header specifying an Eclipse
A/D/A conversion operation. The conversion results specified can
then be printed beneath the header.

Compile command:
FORTRAN HEADER

Comments:
The variables that are passed to this routine have the following
meaning,

DEVICE 21 for A/D or 23 for D/A
SPEC1 starting channel for A/D or D/A
SPEC2 ending channel for A/D or mode set for D/A
IDATA2 conversion count
IER DOIW error return
IORBA the operation's IORBA array
CLOCK conversion count

SUBROUTINE HEADER (DEVICE, SPEC1, SPEC2, IDATA2, IER, IORBA, CLOCK)

INTEGER DEVICE, SPEC1, SPEC2, IDATA2, IER, IORBA(16), CLOCK

IF (DEVICE.EQ.21).OR. (DEVICE.EQ.23) GO TO 605
CALL ERROR("improper device number")

605 CALL FGDAY (IMON, IDAY, IYR)
CALL FTIME (IHOUR, IMIN, ISEC)

WRITE (12,10)
10 FORMAT (1X,"Eclipse A/D/A operation")
WRITE (12,115)
115 WRITE (12,11) IMON, IDAY, IYR
WRITE (12,12) IHOUR, IMIN
12 FORMAT (1X,"time: ",12:" ",12)
WRITE (12,115)
WRITE (12,1)
1 IF (CLOCK.EQ.1) WRITE (12,21)
IF (CLOCK.EQ.2) WRITE (12,24)
IF (CLOCK.EQ.3) WRITE (12,23)
IF (CLOCK.EQ.4) WRITE (12,22)
WRITE (12, 3) SPEC1
IF (DEVICE.EQ.21) WRITE(12, 4) SPEC2
IF (DEVICE.EQ.23) WRITE(12, 8) SPEC2
WRITE (12, 5) IDATA2
WRITE (12, 6) IER
WRITE (12, 7)
WRITE (12, 9) (IORBA(I), I=1,16)
1 FORMAT (1X,"analog-to-digital conversion")
20 FORMAT (1X,"digital-to-analog conversion")
2 FORMAT (1X,"Clock: ",I2)
3 FORMAT (1X,"First channel: ",I2)
4 FORMAT (1X,"Last channel: ",I2)
5 FORMAT (1X,"Conversion count: ",I3)
8 FORMAT (1X,"Mode: ",I2)
6 FORMAT (1X,"DOIT error: ",I4)
7 FORMAT (1X,"Iorb(1-16) (Octal format): ")
9 FORMAT (1X,16(I,06))
21 FORMAT (1X,"pulse clock")
22 FORMAT (1X,"DCH clock")
23 FORMAT (1X,"internal clock")
24 FORMAT (1X,"external clock")
WRITE (12,115)
115 FORMAT (1X)
RETURN
END
This routine erases the screen by typing 24 blank lines.

SUBROUTINE NEWSCR

DO 10 I=1,24
  TYPE
10  CONTINUE

RETURN
END
Function:
This routine prints sections of an integer data array on the
printer in 512-word pages. The calling program specifies all
of the parameters required.

This routine was designed for printing data collected with the
Eclipse A/D/A device. When executing the real number print
option, the integer word is converted to the real number
equivalent that this device uses to store data samples.

Compile command:
FORTRAN PAPER

Comments:
The variables that are passed to this routine have the following
meaning:

IFOR display format: 1 for integer, 2 for real number
and 3 for octal
ISTART the starting page
ISTOP the ending page
ARRAY the data array to be shown
LEN the length of the data array

SUBROUTINE PAPER(IFOR, ISTART, ISTOP, ARRAY, LEN)
INTEGER IFOR, ISTART, ISTOP, LEN, ARRAY(LEN), IPRT, IPAGE
REAL TOPVOLT, REALNUM

TOPVOLT=5.0 magnitude of Eclipse device bi-polar setting
IPRT=32

IPAGE=ISTART-1
I1=(ISTART-1)*512
610 I2=0
IPAGE=IPAGE+1
WRITE (12,8) IPAGE, IPRT
WRITE (12,115)
115 FORMAT (IX)
8 FORMAT (IX,"page"," of",I3)
615 I3=0
620 I4=0
625 I1=I1+1
14=I4+1
REALNUM=FLOAT(ARRAY(I1))/32768.0*TOPVOLT  ; convert to real number
IF (IFOR.EQ.1) WRITE (12,9) ARRAY(I1)
IF (IFOR.EQ.2) WRITE (12,14) REALNUM
IF (IFOR.EQ.3) WRITE (12,13) ARRAY(I1)
14 FORMAT ('+',1X,F7.4,Z)
13 FORMAT ('+',1X,16,Z)
9 FORMAT ('+',1X,06,Z)
WRITE (12,119)
I3=I3+1
IF (I3.NE.16) GO TO 625
WRITE (12,119)
12=12+1
IF (I2.NE.2) GO TO 615
RETURN
END
Title: RedBuf

Author: Capt. Ajeal Hussain

Date: Nov '83

Function:
This routine reads a section of disk file into an integer data array. The file is specified interactively by the user.

Compile command:
FORTRAN REDBUF

Comments:
The variables ARRAY and LEN that are passed to this routine are the data array and its length, respectively. INUM specifies the number of blocks of data to be transferred. On return the integer array contains the user data.

SUBROUTINE REDBUF(ARRAY, LEN, INUM)

INTEGER LEN, ARRAY(LEN), FILENAM(7), INUM, IDEC

500 TYPE
ACCEPT "Enter the filename for reading:"
READ (11,2) FILENAM(1)
2 FORMAT (B13)

CALL OPEN (1, FILENAM, 2, IER)
IF (IER.EQ.13) GO TO 510
IF (IER.NE.1) TYPE "OPEN error", IER
CALL RD_BLK(1, 0, ARRAY, INUM, IER)
IF (IER.NE.1) TYPE "RD_BLK error", IER
IF (IER.NE.1) GO TO 520
CALL RESET
GO TO 100

510 TYPE "<CR>
"This file does not exist."
GO TO 520

520 CALL RESET
ACCEPT "<CR>
"Do you want to, <CR>
"1: try another file<CR>
"2: return to the main menu<CR>
selection: ", IDEC

IF (IDEC.EQ.1) GO TO 500
IF (IDEC.EQ.2) GO TO 100
WRITE (10,1)
1 FORMAT (<CR><CR><CR>
"Please make selections only from the given options.")
GO TO 520

100 RETURN
END

FUNCTION READBU
Title: SeeIt
Author: Lt Allen
Date: Dec 82

Function:
This routine displays sections of an integer data array on the screen in 128-word pages. The calling program specifies all the parameters required.

This routine was designed for displaying data collected with the Eclipse A/D/A device. When executing the real number display option, the integer word is converted to the real number equivalent that this device uses to store data samples.

Compile command:
FORTRAN SEEIT

Comments:
The variables that are passed to this routine have the following meaning.

IFOR display format: 1 for integer, 2 for real number and 3 for octal

ISTART the starting page

ISTOP the ending page

ARRAY the data array to be shown

LEN the length of the data array

SUBROUTINE SEEIT(IFOR, ISTART, ISTOP, ARRAY, LEN)

INTEGER IFOR, ISTART, ISTOP, LEN, ARRAY(LEN), ITOT, IPAGE

REAL REALNUM, TOPVOLT

ITOT=128
TOPVOLT=5.0; magnitude of Eclipse device bi-polar setting

505 TYPE "<CR><CR>
*Press carriage return to begin and<CR>*
*to continue with the next page.<CR>"
ACCEPT

IPAGE=ISTART-1
11=(ISTART-1)=128

510 12=0
IPAGE=IPAGE+1
TYPE "<CR> page", IPAGE, " of", ITOT, "<CR>"

515 13=0
520 14=0
525 I1=I1+1
  I4=I4+1
  REALNUM=FLOAT(ARRAY(I1))/32768.0*TOPVOLT  ; convert to real number
IF (IFOR.EQ.1) WRITE (10,110) ARRAY(I1)
IF (IFOR.EQ.2) WRITE (10,111) REALNUM
IF (IFOR.EQ.3) WRITE (10,112) ARRAY(I1)
110 FORMAT (1X,Q6.1)
111 FORMAT (1X,F7.4,Z)
112 FORMAT (1X,F6.2,Z)
113 FORMAT (1X)
  I3=I3+1
  IF (I3.NE.8) GO TO 520
  WRITE (10,115)
  WRITE (10,115)
  I2=I2+1
  IF (I2.NE.2) GO TO 515
  ACCEPT
  IF (IPAGE.NE.ISTOP) GO TO 510
RETURN
END
Title: SetUp
Author: Lt Allen
Date: Dec 82

Function:
This is a special purpose routine used by program INDIGI and OUTDIGI. It allows the user to select the type of format and section of data buffer for printing/displaying.

Compile command:
FORTRAN SETUP

Comments:
The variable IOP that is passed to this routine has the value 2, for data buffer display, or 3, for data buffer print.
The other variable values are returned to the calling program as set by the user.

SUBROUTINE SETUP(IFOR, IOP, ISTART, ISTOP)

230 ACCEPT "<CR>
*What type of format?<CR>
* 1: two's complement<CR>
* 2: real number<CR>
* 3: integer number<CR>
*selection:"," IFOR

IF (IFOR.LT.1) GO TO 230
IF (IFOR.GT.3) GO TO 230
231 IF (IOP.EQ.2) GO TO 225
IF (IOP.EQ.3) GO TO 235

225 TYPE "<CR>
*There are 128 pages of data, numbered 1 through 128.<CR>
*with each page containing 128 samples."
GO TO 250

235 TYPE "<CR>
*There are 32 pages of data, numbered 1 through 32.<CR>
*with each page containing 512 samples."

250 ACCEPT "<CR>
*What page will be first? ", ISTART
ACCEPT " 
*What page will be last? ", ISTOP

IF (ISTART.LT.1) GO TO 231
ITEST=((-96*IOP)+320)
IF (ISTOP.GT.ITEST) GO TO 231
IF (ISTART.GT.ISTOP) GO TO 231
RETURN
END
Title: SREDM.FR
Author: Capt. Ajmal Hussain
Date: Aug 813

Function:
This routine reads the distance matrix file into a real data array. The file is specified interactively by the user.

Compile command:
FORTRAN SREDM

Comments:
The variables ARRAY and LEN that are passed to this routine are the data array and its length, respectively. On return, the array contains the user data.

SUBROUTINE SREDM(ARRAY, LEN)

INTEGER LEN, FILENAM(7), IDEC
REAL ARRAY(LEN)

500 TYPE
ACCEPt "
*Enter the filename for reading:"
READ (11,2) FILENAM(1)
2 FORMAT (813)

CALL OPEN (1, FILENAM, 2, IER)
IF (IER.EQ.13) GO TO 510
IF (IER.NE.1) TYPE "OPEN error", IER
CALL RDBLM (1,0, ARRAY, 19, IER)
IF (IER.NE.1) TYPE "RDBLM error", IER
IF (IER.NE.1) GO TO 520
CALL RESET
GO TO 100

510 TYPE "<CR>
*This file does not exist."
GO TO 520

520 CALL RESET
ACCEPt "<CR>
*Do you want to, <CR>
* 1: try another file<CR>
* 2: return to the main menu<CR>
*selection:": IDEC

IF (IDEC.EQ.1) GO TO 500
IF (IDEC.EQ.2) GO TO 100
WRITE (10, 1)
1 FORMAT("<CR><CR><CR>
*Please make selections only from the given options."
)GO TO 520

100 RETURN
END
SUBROUTINE SREDT(ARRAY, LEN)

INTEGER LEN, ARRAY(LEN), FILENAM(7), IFIRST, INUM, IDEC

500 TYPE
  ACCEPT "Enter the filename for reading:"
  READ (11, 2) FILENAM(1)

2 FORMAT (S13)

CALL OPEN (1, FILENAM, 2, IER)
IF (IER.EQ.13) GO TO 510
IF (IER.NE.1) TYPE "OPEN error", IER

IFIRST=0
INUM=5

CALL RDBLK(1, IFIRST, ARRAY, INUM, IER)
IF (IER.NE.1) TYPE "RDBLK error", IER
IF (IER.NE.1) GO TO 520

CALL RESET
GO TO 100

510 TYPE "<CR>
  *This file does not exist."
  GO TO 520

520 CALL RESET
  ACCEPT "<CR>
  *Do you want to, <CR>
  1: try another file<CR>
  2: return to the main menu<CR>
  *selection:", IDEC
IF (IDEC.EQ.1) GO TO 500
IF (IDEC.EQ.2) GO TO 100

WRITE (10, 1)
1 FORMAT("<CR><CR><CR>
  *Please make selections only from the given options."")
  GO TO 520

100 RETURN
END

C*****************************************************************
Title: WrtBuf
Author: Capt. Ajmal Hussain
Date: Oct 83

Function:
This is a special purpose routine used by program CREATEMP and
SPEECH. It allows the user to write specified sections of the
data buffer to a disk file.

Compile command:
FORTRAN WRTBUF

Comments:
The variables ARRAY and LEN that are passed to this routine are
the data buffer and its length, respectively. ISTOP is the number
of blocks of integer data to be written to disk file.

SUBROUTINE WRTBUF(ARRAY, LEN, ISTOP)
INTEGER LEN, ARRAY(LEN), FILENAM(7), ISTOP
ISTART = 0

255 ACCEPT "Enter the filename for writing:"
READ (11,15) FILENAM(1)
15 FORMAT (A13)

260 CALL CFILW(FILENAM,2,IER)
   IF (IER.EQ.12) GO TO 265
   IF (IER.NE.1) TYPE "CFILW error ",IER," with your file"

   CALL OPEN (1, FILENAM,2,IEN)
   IF (IER.NE.1) TYPE "OPEN error ",IER," with your file"

   CALL WRBLK(1, ISTART, ARRAY, ISTOP, IER)
   IF (IER.NE.1) TYPE "WRBLK error ",IER," with your file"

   CALL CLOSE (1, IER)
   IF (IER.NE.1) TYPE "CLOSE error ",IER," with your file"
   GO TO 280

265 ACCEPT "<CR>
*This file already exists.<CR><CR>
*Do you want to.<CR>
*  1: delete the current file<CR>
*  2: try another file<CR>
*selection: ": IDEL"
IF (IDEL.EQ.1) GO TO 270
IF (IDEL.EQ.2) GO TO 255
WRITE (10,1)
1 FORMAT ("<CR><CR><CR>
*Please make selections only from the given options."
GO TO 265

270 CALL DFILW (FILENAME, IER)
IF (IER.NE.1) TYPE "DFILW error ",IER," with your file"
GO TO 260

280 RETURN
END

C********************************************************************
Title: WRTTEMP.FR

Author: Capt. Ajmal Hussain

Date: Aug 83

Function:
This is a special purpose routine used by program CREATEMP.
It allows the user to write the Template buffer on to a disk file.

Compile command:
FORTRAN WRTTEMP

Comments:
The variables ARRAY and LEN that are passed to this routine are
the data buffer and it's length, respectively.

SUBROUTINE WRTTEMP(ARRAY, LEN)
INTEGER LEN, ARRAY(LEN), FILENAM(7)

ISTART=0
ISTOP=13

255 ACCEPT "Enter the filename for writing:"
READ (11,15) FILENAM(1)
15 FORMAT (S13)

260 CALL CFILW(FILENAM,2,IER)
IF (IER.EQ.12) GO TO 265
IF (IER.NE.1) TYPE "CFILW error ",IER," with your file"

CALL OPEN (1,FILENAM,2,IER)
IF (IER.NE.1) TYPE "OPEN error ",IER," with your file"

CALL WRBLK(1,ISTART,ARRAY,ISTOP,IER)
IF (IER.NE.1) TYPE "WRBLK error ",IER," with your file"

CALL CLOSE (1,IER)
IF (IER.NE.1) TYPE "CLOSE error ",IER," with your file"
GO TO 280

265 ACCEPT "<CR>
This file already exists.<CR><CR>
Do you want to;<CR>
1: delete the current file<CR>
2: try another file<CR>
selection: "IDEL"
IF (IDEL. EQ. 1) GO TO 270
IF (IDEL. EQ. 2) GO TO 255
WRITE (10,1)
1 FORMAT ('<CR><CR><CR>
#Please make selections only from the given options.
')
GO TO 260
270 CALL DFILW (FILENAME, IER)
IF (IER.NE.1) TYPE "DFILW error ",IER, " with your file"
GO TO 260
260 RETURN
END

C=====================================================================
SUBROUTINE WRTMAT(ARRAY, LEN)

INTEGER LEN, FILENAM(7)
REAL ARRAY(LEN)

ISTART=0
ISTOP=78

ACCEPT "Enter the filename for writing:"
READ (11, 15) FILENAM(1)
15 FORMAT (81S)

CALL CFILW(FILENAM, 2, IER)
IF (IER.EQ.12) GO TO 265
IF (IER.NE.1) TYPE "CFILW error ",IER," with your file"

CALL OPEN(1, FILENAM, 2, IER)
IF (IER.NE.1) TYPE "OPEN error ",IER," with your file"

CALL WRBLK(1, ISTART, ARRAY, ISTOP, IER)
IF (IER.NE.1) TYPE "WRBLK error ",IER," with your file"

CALL CLOSE(1, IER)
IF (IER.NE.1) TYPE "CLOSE error ",IER," with your file"
GO TO 280

265 ACCEPT "<CR>
*This file already exists <CR><CR>
*Do you want to, <CR>
* 1: delete the current file<CR>
* 2: try another file<CR>
*selection: ", IDEL

131
IF (IDEL.EQ.1) GO TO 270
IF (IDEL.EQ.2) GO TO 255

WRITE (10,1)
1 FORMAT (<CR><CR><CR> 
*Please make selections only from the given options.* )
GO TO 265

270 CALL DFILW (FILENAME,IER)
IF (IER.NE.1) TYPE "DFILW error ",IER," with your file"
GO TO 260

280 RETURN
END

C====================================================================
SUBROUTINE WTYPE(L, W)
INTEGER L, W(50)
IF(L.EQ.0) GO TO 10 ;skip zero numbered words
WRITE(12, 15) L ;print word
WRITE(12, 14) W(((L-1)/10)+1)
WRITE(10, 11) W(((L-1)/10)+1) ;display word
14 FORMAT("=", S1, Z)
11 FORMAT("<33><160>"., S1, "<33><161>"., Z)
19 FORMAT(2X, I4)
10 CONTINUE
RETURN
END
SUBROUTINE REDMAT(ARRAY, LEN)

INTEGER LEN, FILENAM(7), IFIRST, INUM, IDEC

REAL ARRAY(LEN)

500 TYPE
     ACCEPT "Enter the filename for reading:"
     READ (1,2) FILENAM(1)
2 FORMAT (8(1X))

CALL OPEN (1, FILENAM, 2, IER)
IF (IER EQ 13) GO TO 510
IF (IER .NE. 1) TYPE "OPEN error", IER

IFIRST=0
INUM=78

CALL RDBLK(1, IFIRST, ARRAY, INUM, IER)
IF (IER NE 1) TYPE "RDBLK error", IER
IF (IER .NE. 1) GO TO 520
CALL RESET
GO TO 100

510 TYPE "<CR>
     *This file does not exist."
     GO TO 520
CALL RESET
ACCEPT "<CR>">
  *Do you want to.<CR>
  * 1: try another <file><CR>
  * 2: return to the main menu<CR>
  *selection: "." IDEC

   IF (IDEC.EQ.1) GO TO 500
   IF (IDEC.EQ.2) GO TO 100
   WRITE (10,1)
   1 FORMAT("<CR><CR><CR>
     *Please make selections only from the given options.")
   GO TO 520

100 RETURN
END

C*******************************************************************************
SUBROUTINE REDTEMP(ARRAY, LEN)

INTEGER LEN, ARRAY(LEN), FILENAM(7), IFIRST, INUM, IDEC

500 TYPE
   ACCEPT "Enter the filename for reading:"
   READ (11,2) FILENAM(1)
   2 FORMAT (S13)

   CALL OPEN (1, FILENAM, 2, IER)
   IF (IER.EQ.13) GO TO 510
   IF (IER.NE.1) TYPE "OPEN error", IER
   IFIRST=0
   INUM=13
   CALL RDBLK(1, IFIRST, ARRAY, INUM, IER)
   IF (IER.NE.1) TYPE "RDBLK error", IER
   IF (IER.NE.1) GO TO 520
   CALL RESET
   GO TO 100

510 TYPE "<CR>
   *This file does not exist."
   GO TO 520

520 CALL RESET
   ACCEPT "<CR>
   *Do you want to: <CR>
   1: try another file <CR>
   2: return to the main menu <CR>
   *selection: ", IDEC
IF (IDEC.EQ.1) GO TO 500
IF (IDEC.EQ.2) GO TO 100
WRITE (10,1)
1 FORMAT("
*Please make selections only from the given options.
"
GO TO 520
100 RETURN
END

C========================================================================
SUBROUTINE SEEMAT(ARRAY, LEN)
 INTEGER IFOR, ISTART, ISTOP, LEN, ITOT, IPAGE
 REAL REALNUM, TOPVOLT, ARRAY(LEN)

500 ACCEPT "<CR>
 *There are 79 pages of data. numbered 1 through 79,<CR>
 *with each page containing 128 values.<CR><CR>
 *What page will be first? ", ISTART
 ACCEPT "
 *What page will be last? ", ISTOP

 IF (ISTART.LT.1) GO TO 500
 IF (ISTOP.GT.79) GO TO 500
 ITOT=79

505 TYPE "<CR><CR>
 *Press carriage return to begin and<CR>
 *to continue with the next page.<CR>"
 ACCEPT

 IPAGE=ISTART-1
 I=1(1,ISTART-1)128

910 I2=0
 IPAGE=IPAGE+1
 TYPE "<CR>page", IPAGE,"\n of", ITOT,"<CR>"
515  I3=0
520  I4=0
525  I1=I1+1
     I4=I4+1
     WRITE (10,111) ARRAY(I1)
111   FORMAT (1X,F7.4,Z)
     IF (I4.NE.8) GO TO 525
     WRITE (10,115)
115   FORMAT (IX)
     I3=I3+1
     IF (I3.NE.8) GO TO 520
     WRITE (10,115)
     I2=I2+1
     IF (I2.NE.2) GO TO 515
ACCEPT
     IF (IPAGE.NE.ISTOP) GO TO 510
RETURN
END
Vita

Ajmal Hussain, was born on 27 November 1955 in Pakistan. He graduated from Aitchison College in Lahore, Pakistan, 1974. In 1978, he graduated from the College of Aeronautical Engineering with the degree of Bachelor of Electrical Engineering with Honor. He entered the School of Engineering, Air Force Institute of Technology in June 1982.

Permanent Address: 664 Shadman Colony
Lahore
Pakistan
A Limited Continuous Speech Recognition system is developed based upon phoneme analysis. 16 bandpass filters are used to obtain the frequency components of the input speech. The input speech is broken into packets of 40 milliseconds each. These packets are compared with phonemes in a template file by a differencing of frequency magnitudes. The resulting phoneme string representation of the input speech is compressed and compared with strings in a library file for discrete word recognition. For continuous speech recognition the phoneme string is analyzed a phoneme at a time to construct word sequences. The word string which best matches the input phoneme string is recognized as the word.
sequence. The system has an accuracy of about 94% for discrete word recognition and about 80% for continuous speech recognition. The vocabulary used is the digits zero to nine and point.