Summer 1985

# Color Display of Vowel Spectra as a Training Aid for the Deaf

Amir Jalali Jagharghi
*Old Dominion University*

# COLOR DISPLAY OF VOWEL SPECTRA AS A TRAINING AID FOR THE DEAF

by

Amir Jalali Jagharghi
B.S.E.E. December 1982, West Virginia Institute of Technology

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF ENGINEERING

ELECTRICAL ENGINEERING

OLD DOMINION UNIVERSITY
June 1985

Approved by:

Stephen A. Zahorian (Director)

Jack B. Stoughton

Sharad V. Kanetkar

# VISUAL DISPLAY OF VOWEL SPECTRA AS A TRAINING AID FOR THE DEAF

Amir Jalali Jagharghi
Old Dominion University
Director: Stephen A. Zahorian

## ABSTRACT

The objective of this research was to develop a transformation for mapping speech parameters to color parameters. This transformation is done in real-time and the resulting color parameters are continuously displayed on a color monitor. This visual speech display is to be used as a speech articulation training aid for the deaf. The conversion of speech acoustic signals into speech parameters was accomplished using special-purpose electronics. The real-time conversion of speech parameters to display parameters was controlled by an 8086/8088 microprocessor operating in an S-100 bus structure. The coefficients of the Karhunen-Loeve series expansion of speech power spectra were used to encode speech into a set of parameters called principal-components. Each principal component is obtained as a linear combination of 16 spectral band energies. The focus of this research was to optimize the method for computing principal components for use with the visual speech display and to determine an optimal transformation from principal components to color parameters.

A series of experiments was completed to determine the principal-components basis vectors for both non-normalized and amplitude-normalized speech spectra. These basis vectors, determined from the statistical properties of the continuous speech of both male and female speakers, were found to be relatively speaker independent. In order to restrict the scope of the research to a specific objective, the transformation of speech parameters to color parameters was optimized for vowels. Clustering experiments of vowels in principal-components spaces showed that vowels are more clustered when level-normalized spectral band energies are used to compute principal-components parameters. However, implementation of a set of level-normalized spectral band energies was not feasible with the available hardware, because of the requirements for real-time operation. Therefore, the transformation from vowels to colors was based on the principal-components parameters obtained from non-normalized spectral band energies, although better results are expected if level-normalized spectral band energies are used to calculate the principal components.

A linear transformation was determined such that the three widely separated vowels /a/, as in hod, /i/, as in heed, and /u/, as in who'd, result in the three widely separated colors red, green, and blue respectively. A real-time flow-mode display of color patterns derived from speech sounds was implemented. A preliminary evaluation of the display indicates that many vowel sounds can be reliably identified by their visual display. Although separate transformations can be used for different speakers, a single fixed transformation appears adequate for males, females, and children.

## ACKNOWLEDGMENT

I would like to thank my advisor, Dr. Stephen A. Zahorian, for his guidance throughout the course of this research. This thesis would not have been possible without his wisdom and support. I feel that I have learned a great deal of knowledge from our association.

This research was made possible by the financial support provided by The Whitaker Foundation. This support was greatly appreciated.

I would also like to thank the other members of my thesis defense committee, Dr. Jack B. Stoughton, and Dr. Sharad V. Kanetkar for their help and criticism offered.

I would also like to thank Phil Berlinsky for preparing some of the figures presented in this thesis. David Whitmore and Gang Chen conducted two experiments to test the performance of the overall system and I thank them for their time.

Special thanks go to my dear family for their continuous love and support throughout my life. I would like to thank my brother Parviz for his love, guidance, caring, and support. I wish for the best of successes in his life and career.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

## LIST OF SYMBOLS

| | |
|---|---|
| PC | Principal Components |
| F1 | Formant one |
| F2 | Formant two |
| Pn | Value of the nth principal component |
| $Pn_0$ | Value of the nth principal component of vowel origin |
| R | Level of Red color |
| G | Level of Green color |
| B | Level of Blue color |
| $R_0$ | Level of the red color of color origin |
| $G_0$ | Level of the green color of color origin |
| $B_0$ | Level of the blue color of color origin |
| MSE | Mean Square Error |
| $\lambda_i$ | ith eigenvalue of statistical matrix |
| [T] | Transformation matrix |
| $[T]^T$ | Transformation matrix transposed |
| Hz | Hertz |
| KHz | Kilo Hertz |
| K | Kilo |

# CHAPTER 1

# INTRODUCTION

The general objective of this research was to develop and test a real-time visual display of vowel information as an articulation training aid for the deaf. The method developed utilizes both hardware and software to continuously transform speech into a color pattern. The display has been optimized for vowels so that perceptually similar vowels result in similar colors, whereas vowels which are perceptually far apart result in dissimilar colors. These colors are displayed on a color monitor such that the color obtained from the most recent sample of speech appears on one side of the display and flows to the opposite side. This display can be used as a highly effective aid for vowel articulation training. This vowel-to-color conversion is one aspect of a larger objective, i.e., a speech-to-color display which can be used as a complete articulation training aid for the deaf.

## 1.1 Survey of Aids for the Deaf

The process of teaching a deaf person to speak is complex, long, and not yet well understood. The potential impact of feedback provided by speech-training aids seems substantial. Thus there is a long history of research devoted to the development of aids for the deaf. These aids can be divided into two categories. The first

category includes aids used to teach suprasegmental skills and the second category includes aids used for articulation training of particular speech sounds.

Suprasegmental aspects of speech are considered to be level, nasality, and pitch. These are slowly-varying speech characteristics and can easily be displayed and are relatively easy to measure using a microphone or a vibration transducer attached to the nose or throat. Voice activated toys, such as clown dolls with a nose that lights up in response to sound (Harper, 1970), have been used to encourage young children to produce sound. Holbrook, Baily, and Rolnick (1974) used a wearable device to train normal-hearing persons with vocal nodules to control the level of their voice. The level of the signal from a vibration transducer on the nose or from a nasal air-flow representing the amount of nasality was indicated by a meter or oscilloscope display (Provonost, 1947; Martony, 1970; Boothroyd, 1977). Visual and tactile displays of voice pitch have been used to lower average pitch range. The simplest displays include one or more lights as a feedback used to inform the talker when pitch is above or below a given frequency range (Martony, 1968; Risberg, 1968).

A more complete speech-training aid was developed as part of a research program at Bolt, Beranek, and Newman, Inc. in Cambridge, Massachusetts. A detailed description of this system is given in Nickerson and Stevens (1973) and Nickerson, Kalikow, and Stevens (1976). It consisted of an analog processing section and a minicomputer. The analog section included a bank of 19 filters ranging from 80-6500 Hz, followed by level detectors, processing to extract pitch information from an accelerometer on the throat, and processing to extract average nasality. The computer was used to

sample the speech information and produce a real-time refreshed display. The display could be frozen and used in a split screen mode with independent upper-lower teacher-student display regions.

Speech-training aids have been used to teach articulation of vowels, consonants, diphthongs, and consonant clusters produced in isolation, in nonsense syllables, and in words. There are aids such as S-indicators, instantaneous spectral displays, formant displays, Lissajous figure displays, vocal tract shape displays, and spectrographic displays. S-indicators are simple and inexpensive devices used to indicate the occurrence of an S sound. A light may be used to indicate the presence of sound (Boothroyd & Decker, 1972). Instantaneous spectral displays provide a real-time display of the short-term spectrum of speech. The LUCIA display (Risberg) and the KAMPLEX display (Borrild) are examples of such displays which used arrays of small lights to indicate the amount of speech energy in each frequency region at any time. Two aids were developed to provide a color display based on formants. One aid (Shigenaga & Sekiguchi, 1978) provided an F1 versus F2 display which uses colors to mark the regions on the display that correspond to different vowels. The other aid (Watanabe, Kisu, Isayama, & Masuno, 1978) displayed colored patterns related to pitch and the first three formants. These patterns could either flow from the bottom to the top of the screen in real-time or they could be frozen.

The first real-time spectrographic display was produced by a device developed at Bell Telephone Laboratories called the "Visible Speech Translator" (Potter, Kopp, & Green, 1947). Today's spectrographic displays depict a two dimensional pattern of fixed length whose appearance depends on three variables. The X-axis

represents time, the Y-axis represents the frequency components of the signal at the corresponding time, and the intensity of the black color, at the corresponding time and frequency, represents the loudness of the speech signal. Figure 1-1 depicts a sample of such display. As can be seen, this display is complex and difficult to comprehend.

All of the aids discussed above use one or more parameters of speech spectral features (pitch, short-time spectrum, loudness) which are indirectly related to articulation features rather than direct use of articulation features themselves. A recent computer-based visual aid has been developed at the University of Alabama (Fletcher, 1982). The computer, in conjunction with special sensors placed in the mouth, detects and displays tongue and vocal tract movements. Thus, in contrast to the displays discussed above, this aid directly displays articulation features. Although the articulation approach may have some advantages over the spectral approach, it is more complex and expensive than some of the aids that use the spectral approach.

More complete summaries of aids are given in Lass (1982) and Levitt, Pickett, and Houde (1980).

## 1.2 Fundamental Speech Parameters

Three of the most important information-bearing parameters for speech are loudness, pitch, and short-time spectral envelope. The pitch signal represents the rate at which the vocal folds vibrate and can be obtained by extracting the fundamental frequency of a signal which is directly derived from vocal fold movement. Loudness, of course, represents the perceptual amplitude of the speech signal and

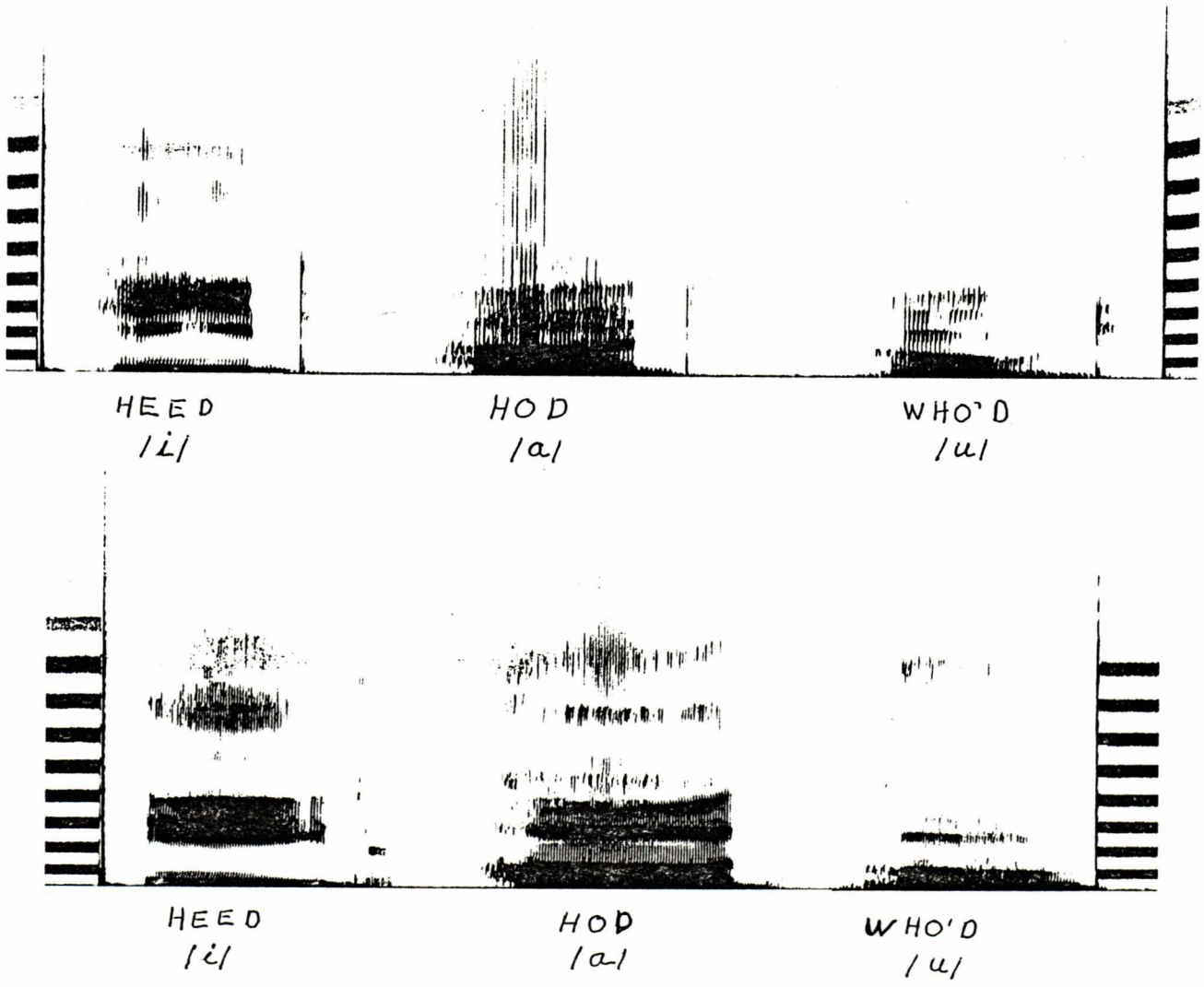



Figure 1-1. Sample spectrographic display for vowels /a/, /i/, and /u/.

is the easiest to obtain. The time-varying spectral envelope, which represents the changing vocal tract shape, can be obtained from the outputs of a bank of bandpass filters.

Since the perception of vowels depends primarily on their short-time spectrum, a parametric model for vowels should be based on the spectrum of vowels. Traditionally vowels have been characterized in terms of the peaks in the envelope of the spectrum (Peterson and Barney, 1952). However, reliable real-time extraction of formants, as required for the speech training aid is very difficult. Linear Predictive (LP) log-area ratios, which determine the shape and size of an acoustic tube model of speech production, also do not seem to be a suitable approach to the problem of real-time vowel-to-color conversion because they are not directly related to the short-time spectral envelope of speech. Thus we chose to use principal components (PC) as a way of representing the speech spectrum. The principal components are primarily related to the overall shape of the spectrum rather than to peaks in the spectrum. They are easy to compute in real-time, contain much information with small number of components, and are relatively speaker independent.

The principal-components analysis method can be used to obtain an efficient representation of a correlated data set, such that the redundancy is removed and a set of uncorrelated data is obtained. The results of statistical experiments, obtained by Zahorian (1978), showed that speech spectral band energies are highly correlated. In the principal-components analysis procedure, the spectral band energies are represented as a function of frequency in terms of the coefficients of an orthogonal series expansion. The basis vectors for this series expansion depend on the statistical properties of the

original data and are referred to as principal-components basis vectors. The uncorrelated coefficients of the basis vectors are referred to as principal components. The principal-components analysis procedure, also called a Karhunen-Loeve (K-L) series expansion, is a well known method in statistical theory.

The principal components, i.e. the coefficients of the basis vectors described above, have the following desirable properties:

1. The principal components are uncorrelated.
2. They are ordered such that the first one accounts for as much as possible of the variance of the original data set, the second one accounts for as much as possible of the remaining variance of the original data set, and so on.
3. The original data set can be approximated by a linear combination of principal components plus a constant additive term which depends on the mean value of the data set.
4. For a given number of components, the expected value of the mean-square error between the original and reconstituted data sets is minimized.

## 1.3 Objectives and Overview

Despite the wide variety of research, most aids developed to date have been relatively unsuccessful. These speech training aids can be divided into three categories depending on their complexity, completeness, and cost. There are aids such as spectrographic displays that contain much information but are complex and difficult to comprehend. The system itself is expensive. On the other hand

there are aids that are inexpensive and easy to interpret but only display a single speech parameter. An S-indicator and a pitch meter are examples of such aids. The vowel-to-color conversion aid, however, belongs to a third category which displays an intermediate amount of information with an easily understandable display format.

This system is different from all other articulation training aids in the sense that it makes use of principal components which no one has previously attempted to use. It is informationally complete, yet easy to interpret because it makes extensive use of color. Use of the latest low-cost technology, such as color CRT's, GDC's, and microprocessors, makes it affordable and therefore potentially available to a large number of individuals. In conjunction with a personal computer and special software, this system can be an effective articulation training aid for the deaf.

The first step toward meeting the objectives of this research was taken by developing various low-level procedures which were necessary to make an interactive and flexible system. Also many intermediate experiments were conducted to optimize the principal-components analysis procedure for use with the vowel-to-color training aid. Clustering of vowels in various PC spaces was investigated in order to determine whether or not it is possible to distinguish vowels based on PC's. A method was devised to determine a linear transformation for mapping vowels, as represented by the first three principal components, to colors. The details of this transformation were optimized experimentally through the course of the research.

Chapter 2 gives an overview of the system as a whole. It also discusses the hardware and software used in the development of the

vowel-to-color convertor. Chapter 3 includes a discussion of PC analysis of speech spectra. It also explains the experiments conducted to obtain an optimum set of basis vectors for the system and the experiments conducted to determine the clustering of vowels in various PC spaces. Chapter 4 explains the derivation of the method used to convert vowels into colors. Also included in Chapter 4 is a discussion of the implementation of this method based on a linear transformation which maps vowels into colors. Two different types of display formats are discussed. Results of two sets of experiments conducted to determine the effectiveness of the color display of vowel spectrum as a training aid are presented in Chapter 5. The conclusions of this research and suggestions for further research are also presented in this chapter.

# CHAPTER 2

## SYSTEM OVERVIEW

The transformation of vowels to a visual display for a speech training aid for the deaf requires both special purpose hardware and software. Software is utilized to make the processing of data as flexible as possible and to automate experiments for optimizing the system. The basic hardware for the system, depicted in Figure 2-1, consists of a speech parameter extractor, an S-100 based microprocessor system controlled by an Intel 8088 microprocessor, a color graphics display controller, and a color monitor. The speech parameter extractor is used to represent important features of speech with a set of uncorrelated parameters. These parameters are sampled by analog to digital (A-D) converters attached to the microprocessor. The microprocessor system converts the speech parameters to display parameters. The graphics display controller is used to control the detailed aspects of the display on the color monitor.

An overview of both the hardware and software for this system is presented in this chapter. Most of the hardware was either purchased or had been "custom" built prior to the start of the research reported in this thesis. Much of the software was developed during the course of the research. Since this particular project is one aspect of a broader research objective, i.e. the development of a general purpose speech training aid, some of the hardware and most of the software was designed to accommodate these broader objectives. The intent of this

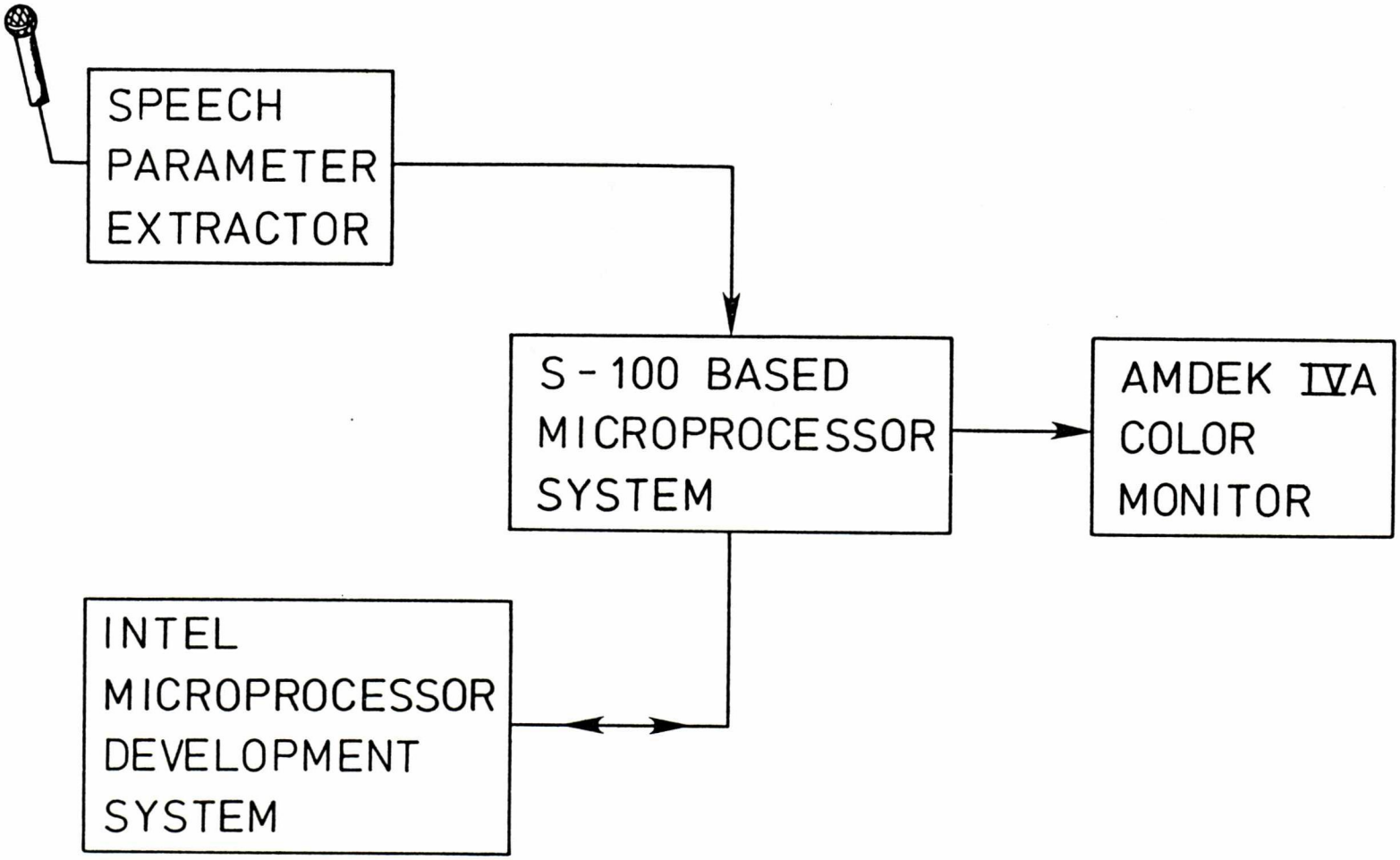



Figure 2-1. System block diagram.

chapter is to summarize the major components of the overall visual speech display system, with particular emphasis on those components which are most important to the specific research objectives of this thesis.

## 2.1 Speech Parameter Extractor

The speech parameter extractor, which extracts pitch, energy, and spectral principal components from the acoustic speech signals, is shown in the block diagram of Figure 2-2. The analog electronics for this real-time system were constructed with nine 4 1/2 " by 9" circuit boards which are placed in a 19" rack-mountable cage and interconnected with a backplane. In addition to various controls and inputs and outputs on the face panel, a bank of bargraph LED's are mounted on the faceplate so that the most important parameters extracted by the electronics can be monitored directly.

### 2.1a Principal-Components Extraction

Specially-designed electronics were constructed to extract the principal components in the manner depicted in Figure 2-3. The microphone signal is first amplified and high-frequency preemphasized at 6 dB/octave up to 3KHz (for spectral "flattening") with a preamplifier board. The preamplifier board also has the capability for expanding or compressing the dynamic range of speech signals, in order to compensate for deaf speakers who have difficulty with amplitude control. (Unfortunately, the compression scheme did not

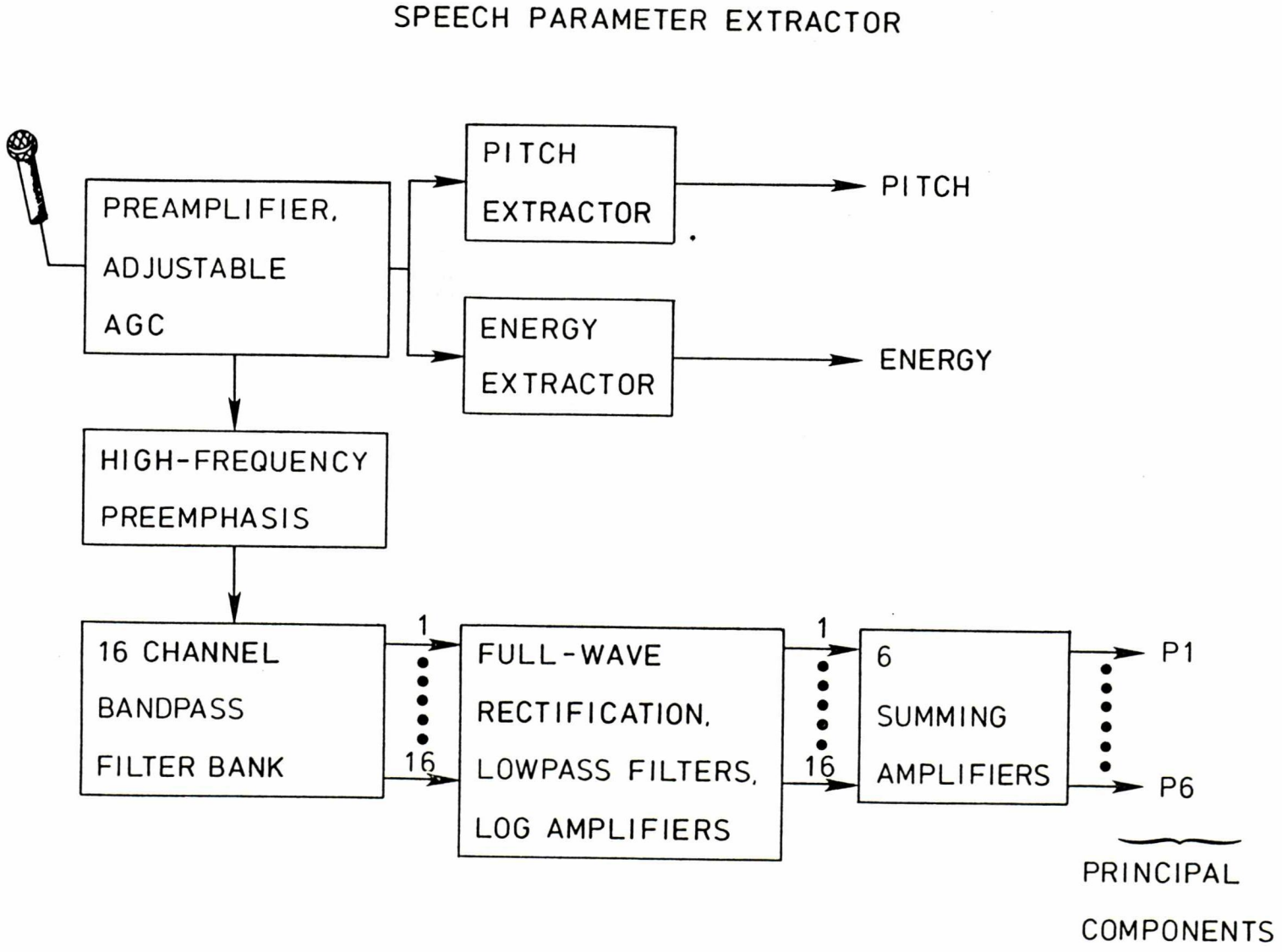



Figure 2-2. Block diagram of speech parameter extractor.

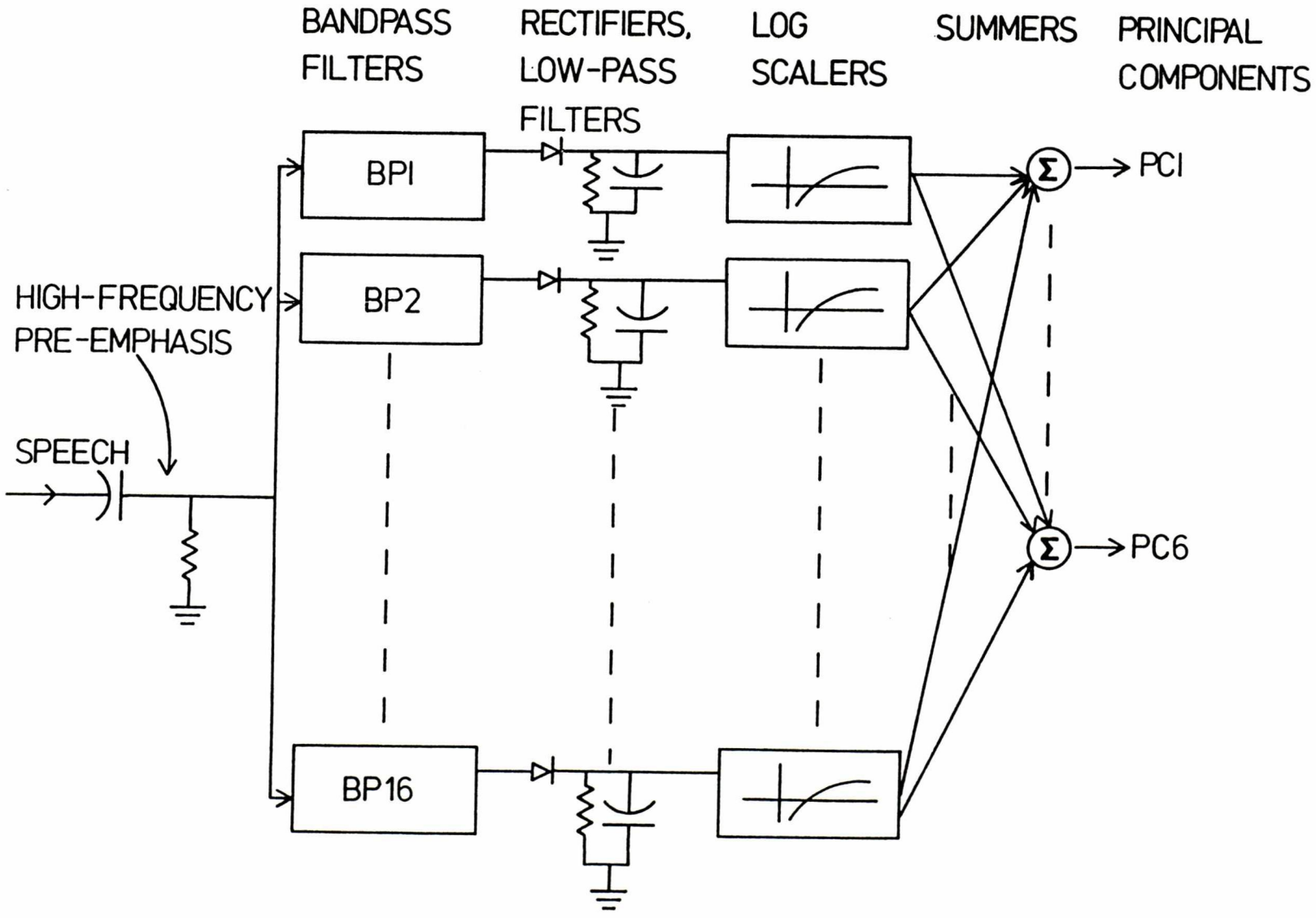



Figure 2-3. System for extracting principal components.

appear to be accurate enough to level normalize the signal to the extent desired for some of the experiments reported in this thesis).

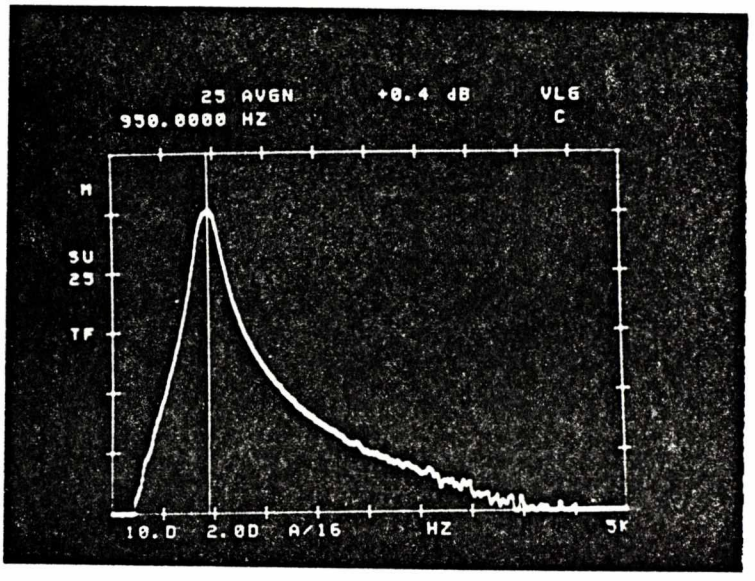
The amplified speech signal is then analyzed by a bank of 16 fourth-order bandpass Butterworth filters. These filters are equally spaced on the perceptual frequency scale of mels and span the frequency range from 240Hz to about 5.6KHz. Table 2-1 lists the measured center frequencies, bandwidths, and Q's. As can be seen, the filter Q's range from about 4 to 10. The frequency responses of the 5th and 13th filters are shown in Figure 2-4. Each filter was built with two operational amplifiers as active components and precision (1%) resistors and capacitors. Each filter output is full-wave rectified and lowpass filtered with a second-order Butterworth lowpass filter at 30Hz. The bandpass filters, rectifiers, and lowpass filters perform a spectral analysis very similar to the analysis of a typical channel vocoder (Gold, et al., 1981). The 16 analysis channels occupy three printed circuit boards.

The 16 low-frequency spectral signals are logarithmically amplitude scaled, using a precision integrated circuit logarithmic amplifier (Intersil 8048) in order to approximate a perceptual amplitude scale. The 16 spectral signals are time multiplexed (at about 1400 Hz rate) and passed through a single logarithmic amplifier. The log amplifier outputs are demultiplexed and input to a 16-channel sample and hold bank to obtain the 16 logarithmically scaled signals. Use of a single log amplifier insures that all signals are scaled in precisely the same way.

Measurements were taken to determine the approximate dynamic range and signal-to-noise ratio of the filter bank, logarithmic amplifier combination. The maximum signal level at the input to the

Table 2-1. Center frequencies and bandwiths of filter bank.

| BAND # | CENTER FREQUENCY (Hz) | BANDWIDTH (Hz) | Q |
|---|---|---|---|
| 1 | 300 | 120 | 2.5 |
| 2 | 430 | 139 | 3.1 |
| 3 | 568 | 145 | 3.9 |
| 4 | 716 | 168 | 4.3 |
| 5 | 900 | 178 | 5.1 |
| 6 | 1089 | 202 | 5.4 |
| 7 | 1390 | 250 | 5.6 |
| 8 | 1636 | 301 | 5.4 |
| 9 | 1935 | 340 | 5.7 |
| 10 | 2258 | 305 | 7.4 |
| 11 | 2609 | 339 | 7.7 |
| 12 | 3059 | 404 | 7.6 |
| 13 | 3483 | 443 | 7.9 |
| 14 | 4049 | 504 | 8.0 |
| 15 | 4651 | 699 | 6.7 |
| 16 | 5248 | 617 | 8.5 |



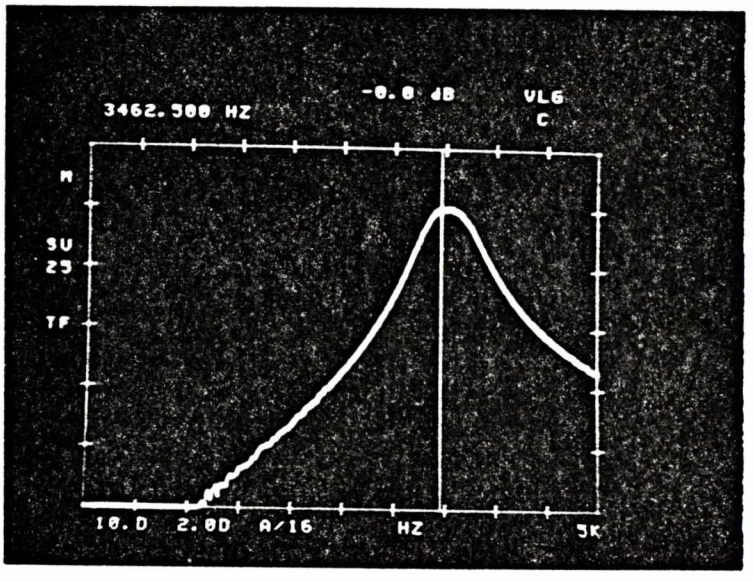




Figure 2-4. Frequency response of 5th and 13th filters.
Vertical: 10 dB/division
Horizontal: 500 Hz/division

filter bank, in order that there be no overload, is about 12 VPP which results in 9.6 VDC at the lowpass filter outputs. The average DC level at the lowpass filter outputs, with no input, is approximately 18 millivolts. Thus the noise at the lowpass filter outputs is about 54 dB below the maximum signal level. The gain of the log amplifier was determined to be approximately 0.2 volt/dB, with a maximum output of 9.2 volts DC for a DC input of 10.0 volts. The minimum output for no input is approximately 1.2 volts. Thus the dynamic range at the log amplifier output is about 8 volts or approximately 40 dB between the largest signal output and the output corresponding to no signal. However, the effective signal-to-noise ratio for speech signals is less than 40 dB since the average signal level at each bandpass filter output would be much less than the peak signal for sinusoidal inputs.

The 16 logarithmically scaled bandpass filter outputs are scaled with six operational amplifier summing circuits to compute the first six principal components. The gain of each summer input was determined by the appropriate coefficient of a principal-components basis vector. The experiments conducted to determine the principal-components basis vectors are reported in Chapter 3 of this thesis. Since some of the principal-components basis vector coefficients are negative, inverting amplifiers were also required for the bandpass filter signals.

The six signals representing the principal components are also amplitude scaled and DC shifted in order to match the 0 to +5VDC range of the A-D converters. In initial pilot experiments, the gains of the individual principal-components signals were independently adjusted so that the maximum signal level for each principal component would be approximately the same (i.e. 5VPP). However, in all the later

experiments the gain for each PC, except for PC1, was adjusted according to the corresponding PC basis vector. The gain for PC1, which would normally be the largest signal since PC1 has the largest variance, was set at 1/3 of the value indicated by the basis vector for PC1. With these gain settings, PC1 and PC2 had the largest and approximately equal variances. The variances for the remaining PC's were less, as indicated by the eigenvalues of the covariance matrix (Chapter 3). Finally each PC signal is lowpass filtered by a second order Butterworth filter, with -3dB frequency at 30Hz, in order to remove any high-frequency noise due to the multiplexed log amplifier.

2.1b Pitch and Energy Extractor

The short-term energy of the speech signal is detected as shown in Figure 2-5. The speech signal is first full-wave rectified, then lowpass filtered at 30Hz with a second-order Butterworth filter. The output of this lowpass filter is thus a low-frequency signal which represents the approximate amplitude or short-term energy of the speech signal. Figure 2-5 also depicts the signal processing used to determine if the speech signal is voiced or unvoiced. The speech signal is considered to be unvoiced if the energy above 4KHz exceeds the energy of the original signal below 2KHz.

As shown in Figure 2-6, a signal proportional to pitch (fundamental frequency of voicing) was also extracted with analog electronics. Although the pitch information is not used as part of the vowel display, the description of the pitch extraction circuitry is included merely to complete the description of the speech parameter extractor. The speech signal is first lowpass filtered at 1000Hz with

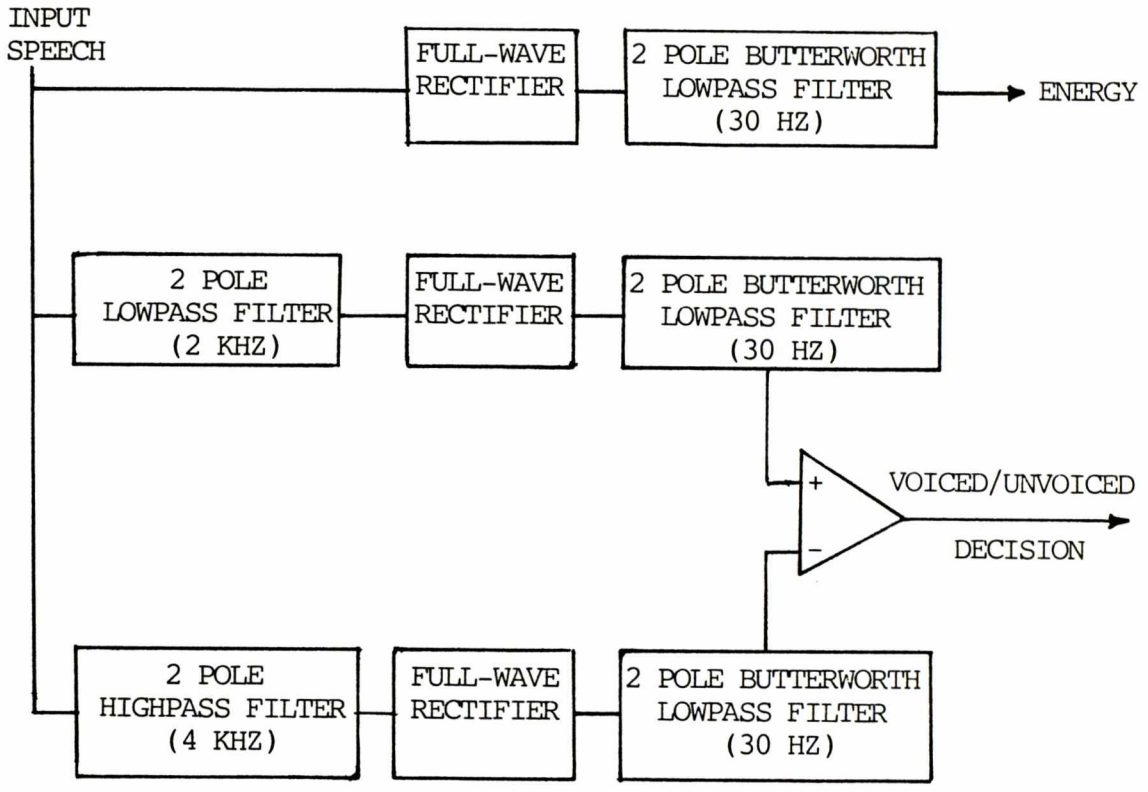



Figure 2-5. Block diagram of energy detector and voiced/unvoiced detector.

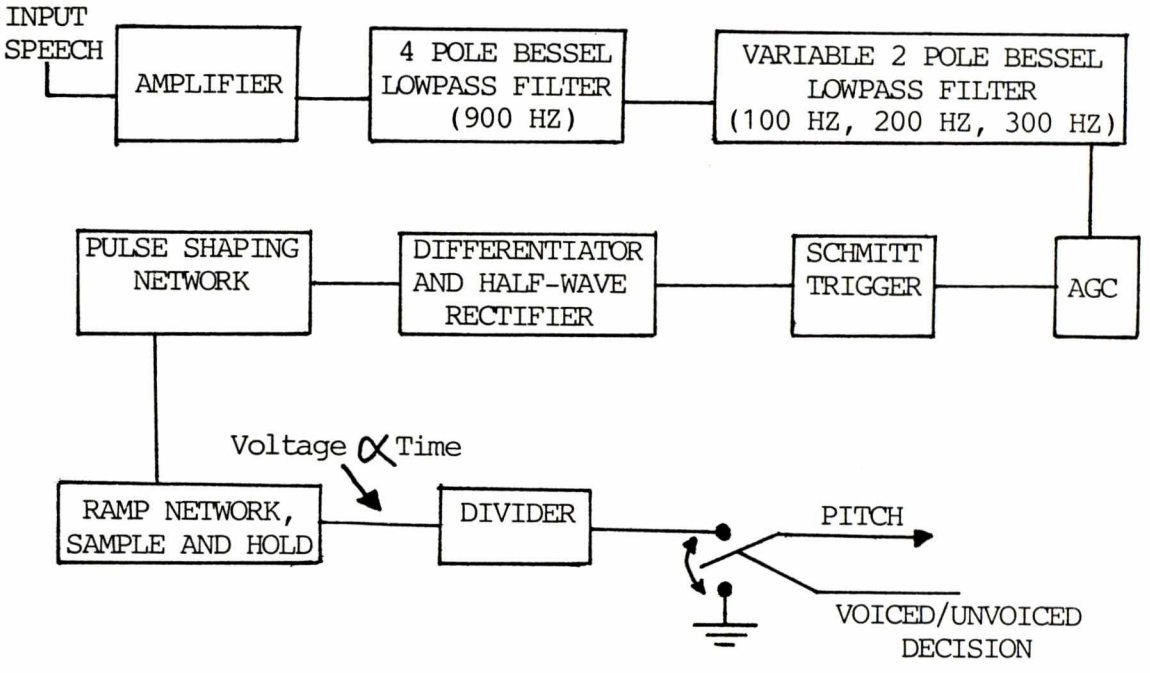



Figure 2-6. Block diagram of the pitch extractor.

a fourth-order Bessel filter. The signal is then lowpass filtered at either 100 Hz, 200 Hz, or 300 Hz, depending on speaker type (male, female, or child), with a second-order Bessel filter. This second filter is switch selectable on the front panel. This signal is then passed through an automatic gain control circuit (AGC) to remove amplitude variations in the signal. The output of the AGC is thus a nearly sinusoidal, almost constant amplitude, signal which is at the fundamental frequency of voicing. The zero crossings of this signal are determined and a voltage proportional to the period is produced. A divider circuit is used to convert the voltage to a signal which is proportional to frequency.

## 2.2 Microprocessor System

A block diagram of the microprocessor system is given in Figure 2-7. As can be seen from the figure, the microprocessor system is centered around an S-100 bus and controlled by an Intel 8088 microprocessor. The main functional components of this system, which will be briefly described in the following several paragraphs, are: 1. an 8088 CPU card; 2. RAM and EPROM memory; 3. an 8-channel 8-bit analog to digital (A-D) converter; 4. a 4-channel 8-bit digital to analog (D-A) converter; and 5. a graphics display controller (GDC). All these components were used in the real-time display. In addition a 16-channel A-D converter card and an EPROM burner card, both custom built for the S-100 bus, were used in the course of the research, although not for the real-time display.

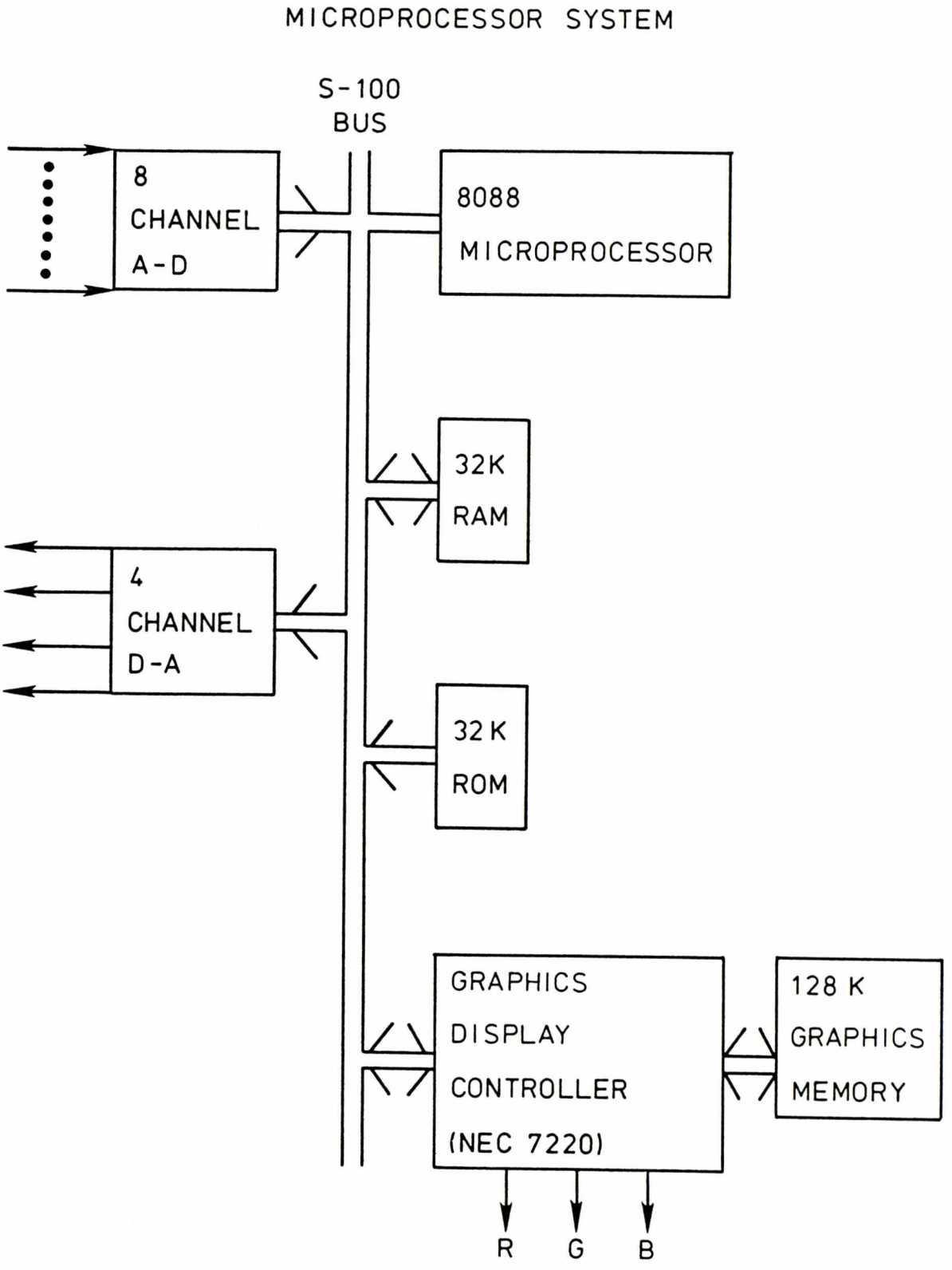



Figure 2-7. Block diagram of microprocessor system.

## 2.2a Microprocessor CPU card

The LDP88, built and marketed by Lomas Data Products Inc., was used as the CPU card. This card implements most of the IEEE S-100 bus signals, with an 8-bit data bus and 24-bit address bus. The primary functional components on the card are an Intel 8/16 bit 8088 microprocessor operating at 5MHz, a single serial RS232 port based on an Intel 8251 Universal Synchronous Asynchronous Receiver/Transmitter (USART), an Intel 8259A Programmable Interrupt Controller (PIC) for controlling eight vectored interrupts, and an on board monitor in a PROM. As discussed in a later section of this thesis, a special program was written for downloading code over the serial link to the LDP88. This program interacted with the monitor on the LDP88.

## 2.2b RAM and EPROM Memory

Commercial RAM and EPROM cards, marketed by Digital Research Computers, were used for memory. The 32 K static RAM card was functional with no wait states. Standard 450 nanosecond 2732 (4 K bytes) EPROM memory integrated circuits were used in the EPROM memory card for no wait state operation. Both cards were configured to operate with an 8-bit data bus.

## 2.2c Analog to Digital Converters

Both an 8-channel and a 16-channel 8-bit analog to digital (A-D) cards were built and interfaced to the S-100 bus. The two cards are quite similar in design. For the 8-channel case the design is based

on the National Semiconductor ADC0808; for the 16-channel case, the design is based on the ADC0816. Each card contains enough on board memory to store the results of one conversion for each channel. The cards can be programmed for the total number of channels to be sampled for each start of conversion command. The start of conversion is controlled either by a programmable timer, by software, or by an external trigger. The conversion time is about 70 microseconds for each channel, thus implying a maximum sampling rate of about 14KHz for single channel operation, and correspondingly lower rates if more than one channel is used. The end of conversion signal is connected to an interrupt on the microprocessor.

### 2.2d Digital to analog converter

A 4-channel 8-bit digital to analog converter (D-A) was also built and interfaced to the S-100 bus. A separate National Semiconductor DAC0830 integrated circuit was used for each channel of the D-A converter. The D-A converter was used in experiments for optimizing the display, and also initially as a simple graphics display controller for the three channels of the color monitor. The D-A card also contains an Intel 8254 programable timer, which was used for setting the desired sampling rates for most experiments.

### 2.2e EPROM card

In order to make the microprocessor system independent of the Intel Series II microcomputer, a special purpose EPROM burner card was designed and built for the system so that program codes could be

burned into EPROMs. These EPROMs would later be located in a designated available memory space. Burning EPROMs was highly advantageous because the downloading of program codes from the Intel Series II microcomputer to the target system was extremely slow. This EPROM burner card has the capability to burn hex formatted program codes into 2716 and 2732 EPROMs or any other compatible EPROM. It was also designed to be capable of copying program codes from a master EPROM into a slave EPROM, with both EPROMs residing on the EPROM burner card.

### 2.2f Graphics display controller

A commercial graphics display controller (GDC), built by I/O Technology, was used to control the color display for most experiments reported in this thesis. The GDC is centered around the NEC7220 GDC integrated circuit and contains a total of 64 K words of memory. This memory is arranged in 4 planes of 16 K words each. Each plane of memory controls 1 of the input bits to a programmable color mapper. The board is capable of supporting a 512 by 512 pixel display, with each pixel 1 of 16 colors, selected from a palette of 4096 possible colors. Since the palette updating process is too slow for real-time updates, the GDC supports up to 16 colors on the screen at any one time. Because of the particular color monitor available (Amdek color IVa), the monitor was arranged to be 1024 pixels horizontally and 240 pixels vertically (non-interlaced).

## 2.3 Software Development Process

As mentioned earlier, software is a tool utilized to make the mapping from speech parameters to color parameters as flexible as possible. This is desired because the details of the transformation must be determined experimentally. Software can also automate the measurement experiments needed to optimize the performance of the system.

An Intel Series II 8080 based microcomputer development system with a disk operating system, was used to develop program codes for the target 8088 microprocessor system. The 8088 microprocessor code is the same as 8086 code. In order to use code developed on the Intel development system, this code has to be serially downloaded from the Intel to the target system. Therefore a program, capable of correctly downloading the code, had to be developed. Also since the target system does not have a terminal of its own, another program had to be developed to make the Intel Series II microcomputer look like a "dummy" terminal to the target system. Both of these programs execute on the Intel development system. Since the Intel is an 8080 based system, these programs were developed using PLM80.

PLM86, a high level programming language which also enables good control over hardware, is used to develop software for the target system. PLM86 programs must be compiled by Intel's PLM86 compiler in order to generate 8086 object code. The object files can be linked and located to a specific block of memory.

## 2.3a LDP88 downloading program

The LDP88 program was written, in PLM80 language, to download the hexadecimal formatted object file of any program written in PLM86 language, through a serial RS232 communication port. Hexadecimal object file format is a way of representing an object file in ASCII. That is, an eight bit binary value is coded into two eight bit ASCII characters, each representing the ASCII code for a single hex (4 bit) digit. This type of representation, which requires twice as many bytes as a binary representation, is called ASCII hexadecimal.

There are four different types of records associated with an 8086 hexadecimal object file. They are:

(1). Extended Address Record;

(2). Start Address Record;

(3). Data Record;

(4). End of File Record.

Each record begins with a RECORD MARK field containing the ASCII code for colon(:), followed by a REC LEN field specifying the number of bytes of information or data which follows the REC TYP field of each record. Each record ends with CHECK SUM field.

The LDP88 program reads one record of the hexadecimal formatted object file at a time. If the record is of type Extended Address, it contains the information which provides bits 4-19 of the Segment Base Address. Bits 4-19 are referred to as the Upper Segment Base Address. If the record is of type DATA, it contains an offset in its Load Address field which can be added to the Segment Base Address to produce the absolute base address of the first data byte in that record. It also contains a block of data which will be stored in a

block of memory whose starting address is the absolute base address calculated for the first byte of data in that block. If the record is of type Start Address, it uses the information in that record to set the IP and CS registers of the 8088 microprocessor. If the record is of type End of File, it signifies the end of information and therefore the end of downloading.

Therefore, briefly speaking, the LDP88 program converts the ASCII formatted address of each record, if it has any, to its hexadecimal representation in order to compute the absolute starting address of the block of data in that record. Then it converts the absolute starting address to its ASCII representation and places it, along with all the data bytes in that record, in the format for the monitor's "E" (Enter) command. Note that ASCII coded data bytes need not be converted to hexadecimal numbers and can directly be included in the output record.

The LDP88 program uses the "E" command of the monitor routine of the LDP88 microprocessor system to download a block of data into a block of memory of the LDP88 target system whose starting address is the absolute base address calculated for the first byte of data in that block.

### 2.3b SDK88 program

A program was written in PLM80 language to make the Intel Series II microcomputer look like the LDP88's terminal. This program makes use of the Intel's system routines to display the received data on the screen and also to transmit the ASCII code of a pressed key to the LDP88 microprocessor system and display its echoed value on the

screen. The maximum communication baud rate to properly receive and display data was found to be 2400 bits/second.

## 2.3c Interrupt handling For the 8088

When a signal is placed on the interrupt pin of the 8088 microprocessor, it looks for the cause of this interrupt. The source of the interrupt is conveyed to the processor by an external device called the PIC (Programmable Interrupt Controller), through an eight bit code. The PIC must be initialized so that the 8-bit codes will correspond to desired pointers for particular devices. The processor multiplies this code by four in order to locate the starting address of a four byte pointer, which points to the start of the service routine associated with that interrupt. Thus the first 200H locations of memory is a table of interrupt vectors.

When an interrupt occurs, the current contents of the IP and CS registers are pushed on the stack. Then the first two bytes of the four byte pointer is loaded into the IP register and the next two bytes are loaded into the CS register. Thus the interrupt service routine must be located in memory at the location corresponding to these IP and CS values. Then the processor services that interrupt. After the interrupt has been serviced, the contents of the stack are popped back into the IP and CS registers, causing a program branch to the next instruction following the last instruction executed prior to the interrupt.

In PLM86 language, interrupts are declared as local procedures at the outer level of the program. An interrupt procedure is declared with the statement "label: PROCEDURE INTERRUPT n" where "label" can

be any desired string of characters and n is any constant integer from 0 to 255. Whenever the 8088 interrupt corresponding to n occurs, this interrupt procedure is activated to service that interrupt. The last line of the interrupt procedure must be the statement "END label;" which denotes the end of the service routine. Therefore the PIC must be programmed to generate an interrupt corresponding to the value of the highest five significant bits of the number n. The lowest three significant bits of n correspond to the priority level (0-8) of the device causing the interrupt.

A list of devices, along with the level of interrupt assigned to each one of them, is given in Table 2-2. For completeness of this report, a list of devices, along with the Input/Output addresses assigned to them, is given in Table 2-3.

### 2.3d Low-level procedures

In order to make the programs interactive, that is to have programs capable of displaying messages and data in several different formats, and also capable of decoding messages and data entered in different formats from the keyboard, various low-level procedures had to be developed. There were also many other "arithmetic" procedures developed, such as double precision integer addition, multiplication, and division. Also developed was a procedure to determine the square root of any positive word sized integer. These procedures along with a short description of their function are listed in Table A-1 in appendix A.

Table 2-2. Device interrupt assignments

| DEVICE | INTERRUPTS ASSIGNED |
|---|---|
| 8-channel A-D | 0 and 2 |
| 16-channel A-D | 3 |

Table 2-3. Input/Output addresses for peripherals

| DEVICE | ADDRESS | I/O | DESCRIPTION |
|---|---|---|---|
| 8-channel A-D | XX20-XX27 | I | Data buffers 0 through 7 |
| | XX28 | O | Port A of 8255. Contains the number of last analog channel to be sampled. |
| | XX29 | O | Port B of 8255. Contains the start signal select. |
| | XX2A | O | Port C of 8255. Used for software start. |
| | XX2B | O | Control register of 8255 |
| 16-channel A-D | XX30-XX3F | I | Data buffers 0 through 15 |
| | XX40 | O | Port A of 8255. Contains the number of last analog channel to be sampled. |
| | XX41 | O | Port B of 8255. Contains the start signal select. |
| | XX42 | O | Port C of 8255. Used for software start. |
| | XX43 | O | Control register of 8255 |
| EPROM burner | XX40 | O | Port A of 8255. Its least four significant bits contains the highest four significant bits of the EPROM address. |
| | XX41 | O | Port B of 8255. Contains the least significant byte of the address for EPROM. |
| | XX42 | I & O | Port C of 8255. Contains data to be written to or read from the EPROM. |
| | XX43 | O | Control register of 8255 |

Table 2-3. (Continued)

| DEVICE | ADDRESS | I/O | DESCRIPTION |
|---|---|---|---|
| 4-channel D-A | XX10 | O | Port A of 8254. Counter 0 |
| | XX11 | O | Port B of 8254. Counter 1 |
| | XX12 | O | Port C of 8254. Counter 2 |
| | XX14 | O | Control register of 8254 |
| | XX15 | O | D-A channel #1 |
| | XX16 | O | D-A channel #2 |
| | XX17 | O | D-A channel #3 |
| | XX18 | O | D-A channel #4 |
| GDC | XX80 | I & O | Status register read and parameters into FIFO |
| | XX81 | I & O | FIFO read and command into FIFO |
| | XX82 | O | Color mapper address and data |
| | XX83 | O | Mapper control logic |
| Serial port | XX00 | I & O | Receive and transmit data |
| | XX01 | O | Mode or command byte for 8251 |
| Peripheral Interface Control (PIC) | XX02 | I & O | Write: ICS1, OCW2, and OCW3<br>Read: Status and Poll |
| | XX03 | I & O | Write: ICW2, ICW3, ICW4, OCW1 (mask)<br>Read: OCW1 (mask) |

## 2.4 Graphics Software

In addition to programming the 8088 microprocessor, the NEC7220 GDC had to be programmed to implement the desired display format. From the viewpoint of the 8088, the GDC consisted of four ports through which either commands and data could be either written or read. The initialization consisted of programming the GDC to generate the synchronization signals for the monitor, and assigning the colors to the color table for the color mapper. Since the details of the initialization, although quite complex, are described in manuals published by NEC, this process will not be described. However, the objectives of this research required the use of far more than the 16 available colors from "standard" use of the GDC. Therefore this important "nonstandard" procedure for obtaining more colors, in a real-time display, will be described.

A technique called dithering (Foley and Van Dam, 1982) was used to obtain additional colors by trading spatial resolution for increased color resolution. In this particular implementation, pixels were grouped in horizontal groups of four, thus reducing the effective horizontal resolution from 1024 pixels to 256 pixels. Based on limits imposed by the color mapper, the brightness of red, green, and blue at each original pixel location was considered to be either 0 (off), 0.5 (half intensity), or 1.0 (full intensity). With this assumption, the 9 pattern dither table shown in Table 2-4 is obtained. Since this same dither pattern is used for each of red, green, and blue, a total of 9*9*9=729 colors is obtained. These dither patterns were used in conjunction with the color mapper assignment shown in Table 2-5.

Table 2-4. Nine pattern dither table.

| COLOR | RELATIVE INTENSITY FOR EACH PIXEL | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0.5 | 0 | 0 | 0 |
| 2 | 0.5 | 0 | 0.5 | 0 |
| 3 | 0.5 | 0.5 | 0.5 | 0 |
| 4 | 0.5 | 0.5 | 0.5 | 0.5 |
| 5 | 1 | 0.5 | 0.5 | 0.5 |
| 6 | 1 | 0.5 | 1 | 0.5 |
| 7 | 1 | 1 | 1 | 0.5 |
| 8 | 1 | 1 | 1 | 1 |

Table 2-5. Color mapper color assignments.

| COLOR | RED | BLUE | GREEN |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 0.5 | 0.5 |
| 3 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 |
| 5 | 0.5 | 0 | 0.5 |
| 6 | 0.5 | 0.5 | 0 |
| 7 | 0.5 | 0.5 | 0.5 |
| 8 | 0.5 | 0.5 | 1 |
| 9 | 0.5 | 1 | 0.5 |
| 10 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 |
| 12 | 1 | 0.5 | 0.5 |
| 13 | 1 | 1 | 0 |
| 14 | 1 | 1 | 0.5 |
| 15 | 1 | 1 | 1 |

An example is depicted in Table 2-6. Assume a value of red=2, blue=5, and green=8 is desired at some location (consisting of 4 pixel memory locations, but one effective pixel location). These values correspond to the three dither patterns shown in Table 2-4. In turn, these dither patterns require color 4, color 8, color 8, and color 1 from the color table in the four pixel locations. In order to specify these four colors, the bit codes which define the four colors must be written into the appropriate bit planes. In some cases the required colors are not available from the color mapper---a 27 entry color table rather than a 16 entry color table would have been required. For these cases, the colors were "rounded" to the nearest available color. This technique was tested and found to work well for displaying a large range of colors on the monitor.

## 2.5 EPROM Burning Procedure

There are four types of memory segments associated with any program written in PLM86 language. These four segments are the code segment, data segment, stack segment, and interrupt table segment. The code segment contains program instructions and its contents remain unchanged. Therefore the code segment can be in ROM space. The data segment contains the values of the variables of the program and these values are allowed to change. Therefore the data segment must be in RAM space. The program code writes variable information in the stack segment and therefore the stack segment must also reside in RAM space. The interrupt table segment, because of the LDP88's interrupt handling procedure, must reside in the lowest block of memory. The length of this block is 200H. Since this block of the memory is

Table 2-6. Color selection example.

| DESIRED COLOR | DITHER PATTERN | | | |
|---|---|---|---|---|
| RED = 2 | 0.5 | 0 | 0.5 | 0 |
| BLUE = 5 | 1 | 0.5 | 0.5 | 0.5 |
| GREEN = 8 | 1 | 1 | 1 | 1 |

| | COLOR NUMBERS FROM COLOR MAP |
|---|---|
| PIXEL 1 | 4 = (0,1,1)* |
| PIXEL 2 | 8 = (0.5,0.5,1) |
| PIXEL 3 | 8 = (0.5,0.5,1) |
| PIXEL 4 | 1 = (0,0,1)** |

* Since the color corresponding to (0.5,1,1) is not available in the assigned color map, the next lower color, (0,1,1), is selected. However the "missing" 0.5 from red is added to the red component for the next pixel so that the next color chosen is (0.5,0.5,1) rather than (0,0.5,1) as specified by the original dither pattern.

** Since (0,0.5,1) is not available, the next lower color, (0,0,1) is selected. Within each group of four pixels, the algorithim compensates for rounding down by adding to the next pixel, if possible. However the rounding down is not carried over to the next group of four pixels.

restricted to be RAM by the LDP88's monitor routines, the interrupt table segment has to reside in RAM space.

The bottom 32 K bytes of memory (0-32 K) is allocated for RAM space and the next block which has a length of 24 K bytes is allocated for EPROM space (32-56 K). When burning EPROMs, this latter block of memory is temporarily filled with 24 K bytes of RAM which will contain downloaded program code to be burned into the EPROMs.

Note that the code segment of the program, which is to be burned into the EPROMs, must be located such that it occupies a block of the memory within the 24 K bytes of the EPROM space. The assigned address must be the actual address in which the program will finally reside after it has been burned into the EPROMs, since the code contains absolute addresses.

All the fixed messages which will be displayed by the program, have to become a part of the code segment. The compiler will put these messages in the code segment if they are initialized with the DATA initialization statement.

When a program is downloaded, the interrupt table associated with that program is also downloaded to the lowest 200H locations of memory. However, when using a program on the EPROM card, the interrupt table associated with that program must be first "saved" in the EPROM and then transferred to the lowest 200H locations prior to using interrupts. This can be achieved by using the following procedure:

(1). In the program, a dummy array of length 200H must be dimensioned such that it occupies a block of the EPROM space not conflicting with the program code. Also there must be a code segment which copies the contents of that

dummy array into the lowest 200H locations of the RAM space prior to using interrupts.

(2). The program should be downloaded after being linked and located.

(3). The contents of the lowest 200H locations of the RAM space should be burned into the part of the EPROM which is designated to hold the contents of the dummy array.

(4). The program code should be burned into the EPROMs.

Whenever the program is executed, it copies the interrupt table into the lowest 200H locations of the RAM space before it starts to service any interrupts. Programs which do not use interrupts can be burned without following steps 1 and 3 of the above procedure.

# CHAPTER 3

## MEASUREMENT OF VOWEL SPECTRA

The varying spectral characteristics of vowels makes it possible for vowels to be distinguished from one another. Some vowels have fairly similar spectral characteristics and thus sound similar and are relatively difficult to distinguish. The following vowels in the hVd context are examples: heed and hid, hod and haw'd, hood and who'd, etc. There are also vowels with widely different spectral characteristics which do not sound similar and therefore are easy to distinguish. The following vowels in the hVd context can be used as examples: heed, hod, who'd, and had.

As discussed in Chapter 1, the principal components are related to the overall shape of the short-time spectrum rather than the peaks in the speech spectrum. They are easy to compute in real-time and contain much information with a small number of components. Therefore the principal components were used to represent speech spectra.

However, it is important to determine if vowels are "well" distributed in this principal-components space. That is, vowels which are perceptually similar should have similar parameter values and vowels which are perceptually far apart should be spaced far apart in the parameter space. If vowels are appropriately distributed in a principal-components space, then it should be possible to distinguish each vowel based on a principal-components representation.

This chapter will discuss the reasons for choosing the principal-components space, and will explain the experimental procedures for computing the principal-components basis vectors. An investigation of the distribution of vowels in the principal-components space will be reported.

The process of obtaining an optimum set of basis vectors for the system included three sets of similar experimental procedures. Since the first two sets of experiments were preliminary, they will not be discussed in full detail and only the last sets of experiments will be discussed in full detail.

## 3.1 Principal-Components Analysis of Speech Spectra

To compute the K-L basis vectors the covariance matrix associated with the spectral band energies must be determined and its eigenvalues and eigenvectors computed. These eigenvectors are the K-L basis vectors. The mean square error depends on the eigenvalues of the covariance matrix and is defined as:

$$MSE = \frac{\sum_{i=m+1}^{n} \lambda_i}{\sum_{i=1}^{n} \lambda_i}$$

In the equation above, the denominator represents the total energy and the numerator represents the eigenvalues not considered. A more detailed description is given in Zahorian (1978).

Figure 3-1 depicts a block diagram of the principal-components analysis procedure. In this figure N original parameters correspond to an N-dimensional data vector whose components are N amplitude-coded speech spectral band energies. M (M < N) K-L basis vectors are used as a linear transformation of the N-data points to obtain an M-dimensional data vector whose components are uncorrelated. The original data set can be obtained from this new data set by using the inverse transformation shown.

Since, for a given number of terms, the principal-components analysis "explains" a maximum amount of data variance, it is generally assumed that the principal components also retain a maximum amount of information for the data. The principal components are also straightforward to compute in real-time (once every 10-20 msec needed to represent changing speech events), fairly robust in the presence of noise, and relatively speaker independent. For all these reasons, a principal-components representation of filter bank data was chosen as a set of parameters for representing vowel data.

However, even if this method does produce maximum information about vowel spectra, it is not necessarily true that vowel spectra will be maximally clustered. Nevertheless, even if vowels are not well clustered in a principal-components parameter space, it might be possible to determine a rotated subspace such that the data does cluster according to vowel categories. If this is the case, the principal-components analysis is useful as a data reduction step. That is, instead of using 16 band energies, only three to five principal components can be utilized in assigning different colors to different vowels.

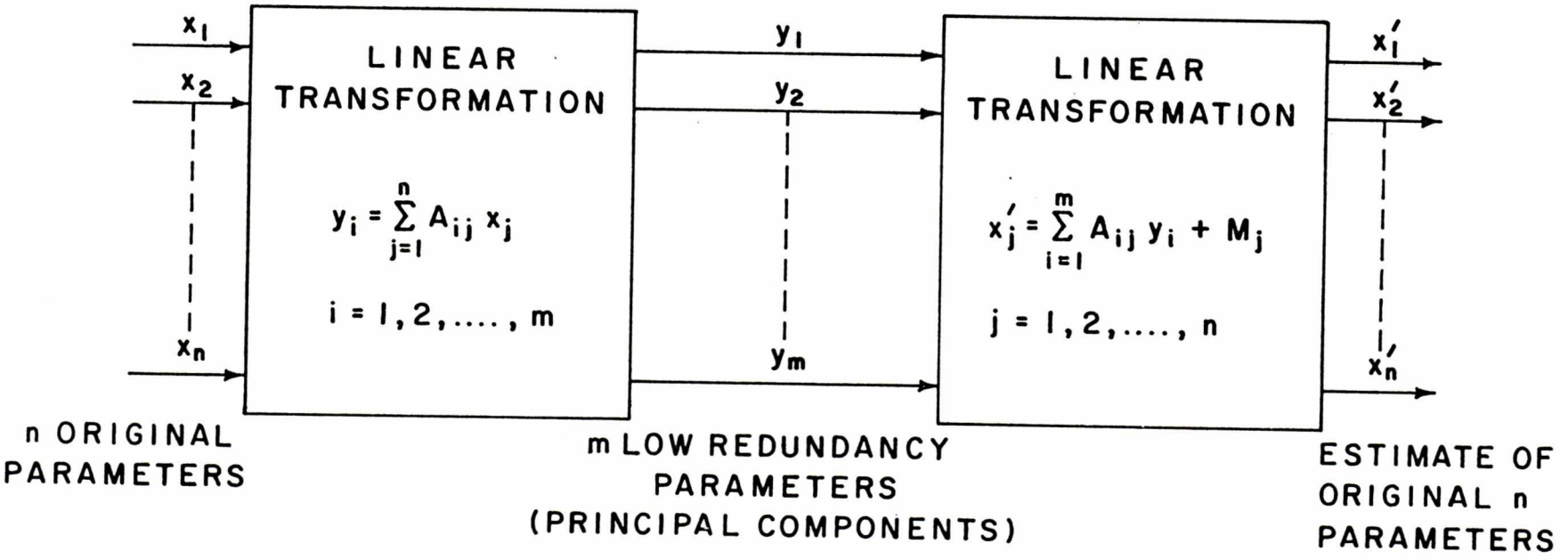


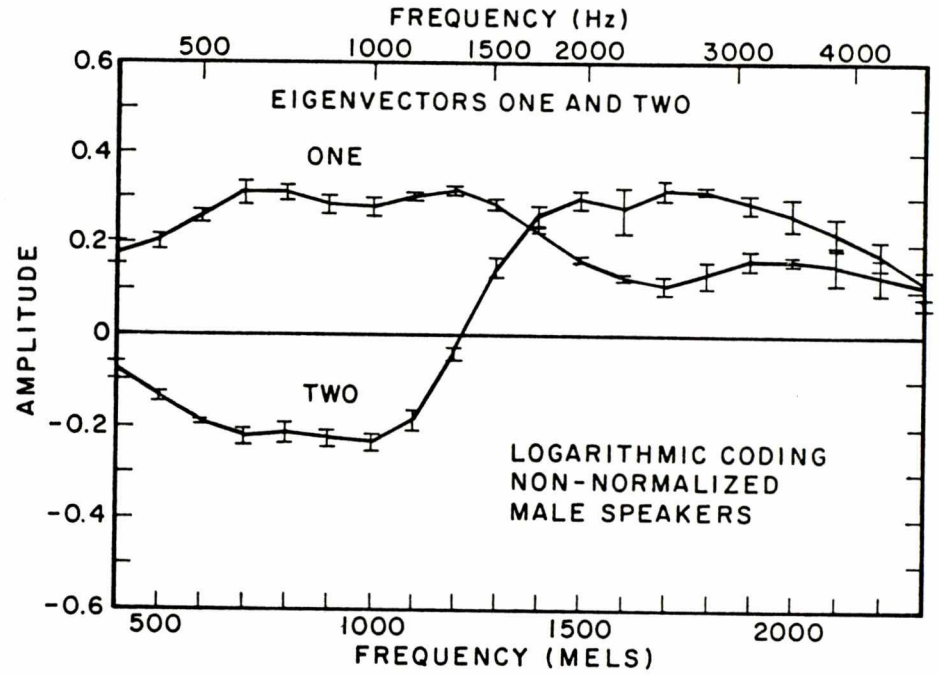

Figure 3-1. Block diagram of PC analysis procedure. The transformation coefficients are determined by PC basis vectors (Zahorian, 1978).

## 3.2 Computation of PC Basis Vectors

The first step in a PC analysis is to determine the PC basis vectors based on the statistical analysis of a large amount of data. These basis vectors are the eigenvectors of the covariance matrix obtained from the original multidimensional data. For this study, the data are the signals from the 16-channel filter bank.

### 3.2a Preliminary set of basis vectors

Initially, due to equipment limitations, it was impossible to obtain any data. However, Zahorian (1978) had already computed a set of principal-components basis vectors by using an FFT simulated filter bank. Figure 3-2 shows the first four principal components basis vectors as computed by Zahorian. These were computed from non-normalized spectral band energies with logarithmically coded amplitudes. Since details of Zahorian's filter bank (simulated by an FFT) were quite different from the filter bank of the present research, these basis vectors are not likely to be optimum for the present work. It can be seen that these basis vectors resemble a set of cosines. The work done by Gordy (1982) showed that a good spectral model can be obtained by using a set of discrete cosine basis vectors. The principal components are referred to as discrete cosine series coefficients when this type of modeling is used. Through speech synthesis experiments, the discrete cosine series coefficients were found to be nearly identical to principal components. That is, for a given number of terms, speech synthesized using cosine basis vectors was almost identical to the speech synthesized using



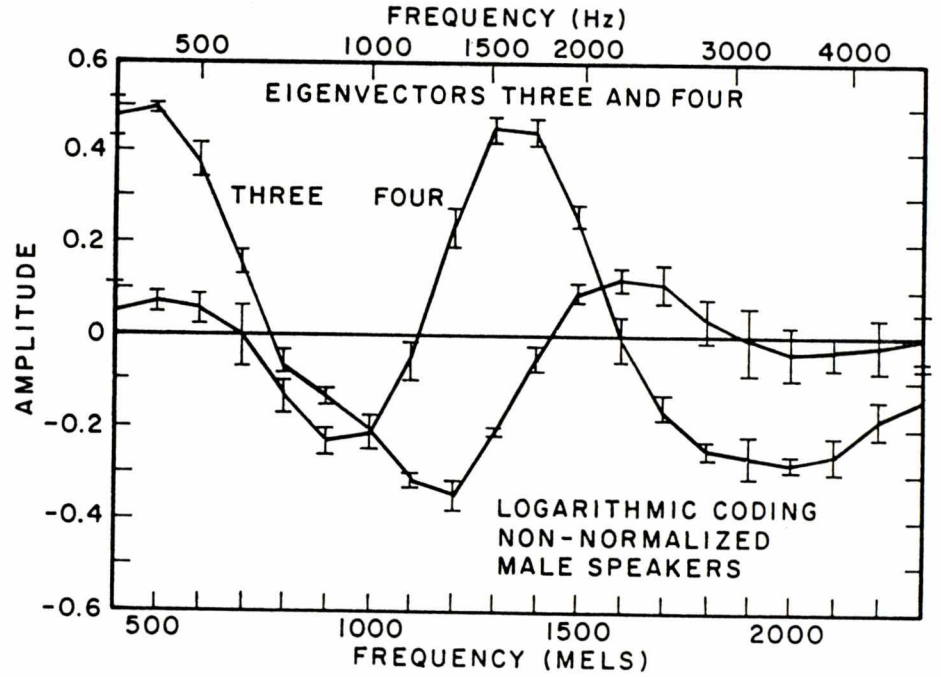




Figure 3-2. First four PC basis vectors as computed by Zahorian. The vertical bars represent two standard deviations.

principal-components basis vectors. Since neither cosine basis vectors nor the basis vectors obtained by Zahorian were likely to be optimum, and the cosine basis vectors were easier to deal with (well defined mathematical functions), cosine basis vectors were used in our initial experiments.

The cosine basis vectors were tested by examining the correlation characteristics of parameters based on these basis vectors, and through preliminary vowel-to-color experiments. At the time, a reel-to-reel tape with recorded speech was available from some previous experiments in the speech laboratory. Six individuals (three males and three females) were recorded on this tape. Prior to recording, their speech was lowpass filtered with a cut-off frequency of 5000 Hz. Each individual had read two different passages. The first was called the "Rainbow Passage" which, on the average, lasted approximately 90 seconds. The other passage was called the "Orwell Passage" and lasted around 270 seconds on the average. The 8-channel A-D converter, described in Chapter 2, was used to sample the parameters.

The tape described above was used to obtain statistical data for these parameters, including the covariance and correlation matrices. The tape was played and various parameters including discrete cosine coefficients were computed in real time by the speech parameter extractor. A sampling time of 20.0 milliseconds for the 8-channel A-D converter was sufficient time to allow the processor to read in seven of the eight channels and compute statistics during the remaining time. Six of these inputs were discrete cosine coefficients and the seventh input was the energy level of the speech signal. This channel was used for detection of speech on the tape. Since this was

an interrupt driven procedure, at the beginning of the service routine the energy signal was compared with a threshold value of 32 (on a scale of 0 to 255). If the energy signal had a higher level than 32 then it was assumed that speech was present and all six parameters were sampled and processed. Otherwise the service routine ended.

Through experimentation, it was found that sampling of the "Rainbow Passage" would produce enough data points (1500 to 1900 points) to determine an accurate measure of the covariance and correlation matrices. Therefore only the "Rainbow Passage" read by each individual was used to obtain the correlation matrices. Table 3-1 shows the correlation matrix computed for a male speaker. It can be seen that, even though double precision arithmetic was used to compute this matrix, there are still some round off errors.

If these parameters were optimum they should have been uncorrelated, but a typical result of the experiment, as given in Table 3-1 showed high correlation between some parameters. Also the initial vowel-to-color experiments were unsatisfactory. That is, different vowels as spoken by a single person would not necessarily produce different colors. Additionally, the same vowel spoken by different individuals would not always produce the same color.

This led to the conclusion that a set of cosine basis vectors was not suitable for this system and a new set of basis vectors was needed.

### 3.2b Second set of basis vectors

Since only an 8-channel A-D converter was available initially, a covariance matrix could only be computed for 8-dimensional data rather

Table 3-1. Correlation matrix of PC parameters computed based on the statistical data of a single male speaker. A set of cosine basis vectors was used in the system.

| | | | | |
|---|---|---|---|---|
| +0.995 | +0.646 | -0.421 | -0.174 | -0.201 |
| +0.646 | +0.977 | +0.155 | -0.469 | -0.339 |
| -0.421 | +0.155 | +0.937 | +0.077 | -0.062 |
| -0.174 | -0.469 | +0.077 | +1.041 | +0.633 |
| -0.201 | -0.339 | -0.062 | +0.633 | +1.011 |

Table 3-2. Correlation matrix of PC's computed based on the statistical data of all speakers. The set of basis vectors, computed using statistical data of only 8 band enrgies, was used in the system.

| | | | | |
|---|---|---|---|---|
| +0.994 | +0.246 | +0.082 | -0.017 | +0.190 |
| +0.246 | +0.998 | +0.070 | -0.259 | -0.197 |
| +0.082 | +0.070 | +0.988 | +0.043 | -0.069 |
| -0.017 | -0.259 | +0.043 | +0.999 | +0.045 |
| +0.190 | -0.197 | -0.069 | +0.045 | +0.982 |

Table 3-3. Correlation matrix of PC's computed based on the statistical data of all speakers. The set of basis vectors, computed using statistical data of all 16 band energies, was used in the system.

| | | | | |
|---|---|---|---|---|
| +0.979 | +0.046 | +0.027 | +0.118 | +0.111 |
| +0.046 | +1.017 | -0.043 | -0.105 | -0.017 |
| +0.027 | -0.043 | +0.959 | -0.045 | -0.097 |
| +0.118 | -0.105 | -0.045 | +1.027 | -0.005 |
| +0.111 | -0.017 | -0.097 | -0.005 | +0.933 |

than 16. Therefore it was decided to sample every other band energy since adjacent band energies are highly correlated. Bands 1, 3, 5, 7, 9, 11, 13, and 15 were connected to channels 1, 2, 3, 4, 5, 6, 7, and 8 of the A-D converter respectively.

Also because all eight channels of the A-D converter were used for sampling band energies, both the third channel and the fifth channel of the A-D converter were compared with a threshold value of 32 to determine the presence of speech.

The same tape used for the previous experiments was used again to obtain statistical data for the spectral band energies. A covariance matrix, based on eight spectral band energies, was calculated and new 8-dimensional basis vectors were computed. Linear interpolation between adjacent coefficients for each basis vector was used to determine a set of 16-dimensional basis vectors. In order to test the speaker independency of these basis vectors, four categories were chosen and the basis vectors were computed for each category. These categories were:

(1) Individual speakers;

(2) All male speakers;

(3) All female speakers;

(4) All speakers.

In the first category, six covariance matrices were obtained (each based on the data of a single speaker) and six sets of basis vectors were computed (one from each covariance matrix). In the second category, only a single covariance matrix based on the data of all male speech was obtained and from that a single set of basis vectors was computed. The third category was the same as the second category except that only female speech was used. In the fourth

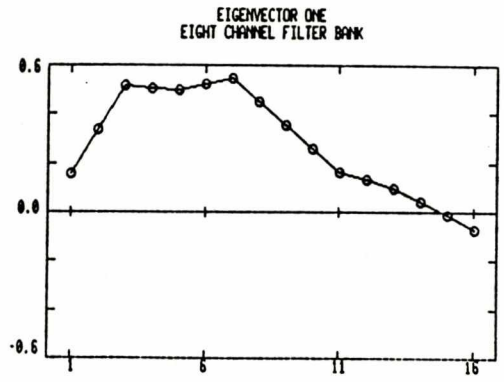
category the number of male speakers was equal to the number of female speakers. A single covariance matrix was obtained based on the data of all speakers. This matrix was then used to calculate a single set of basis vectors. A logical choice for a general set of basis vectors would be the basis vectors computed for all speakers because the basis vectors appeared to be relatively speaker independent. Five of the basis vectors, computed using data from all speakers, are shown in Figure 3-3. This set of basis vectors was incorporated into the system for further experiments.

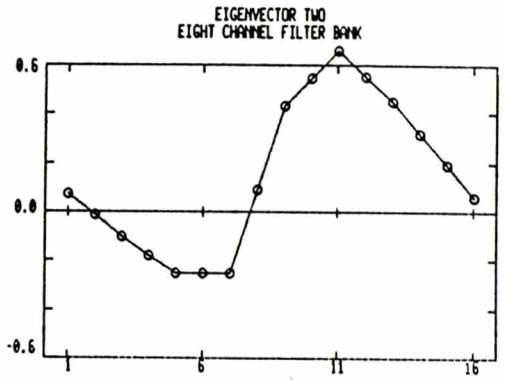
The tape recorded speech was used again to determine the correlation matrices for the new parameters. The correlation matrices showed that these parameters had lower correlations than the previous case where cosine basis vectors were used. The correlation matrix calculated using data of the all-speaker group is given in Table 3-2. The vowel-to-color experiment was promising with better results than for the previous case.

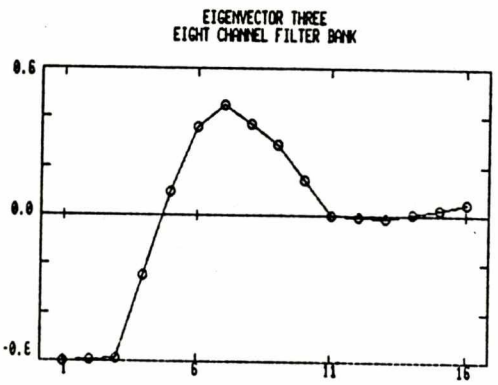
The improvement in results of the latter set of experiments over the previous one, suggested that a more accurate set of basis vectors, obtained by using all 16 spectral band energies, could further improve the results. Therefore it was necessary to have a 16-channel A-D converter. The design of the 8-channel A-D converter was modified to a 16-channel A-D converter and a new 16-channel A-D converter was built for this special purpose.

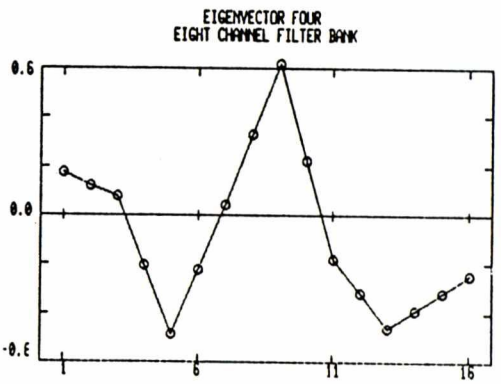
### 3.2c Final set of basis vectors

This final set of basis vectors was to be the optimum set of 16-dimensional basis vectors for the present system. Therefore the



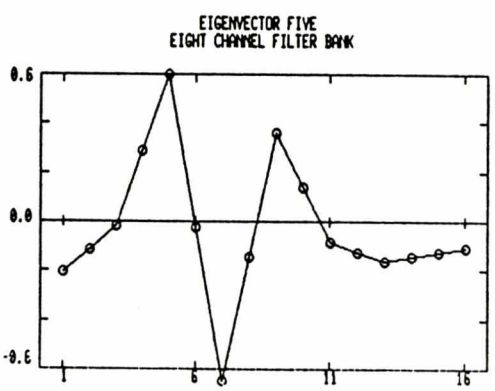




Figure 3-3. First five PC basis vectors computed using only eight non-normalized band energies.

details of the experimental procedure were carefully controlled and will be discussed here.

Since the speech used in previous experiments was lowpass filtered, prior to recording, with a cut-off frequency of 5000 Hz, the recorded speech signal had lost its high frequency content. Therefore some new speech materials were recorded on a cassette without passing the signal through any kind of filter. The same passages (Rainbow passage and Orwell passage) were read by nine individuals (five males and four females) and their voices were recorded. A condenser microphone (Realistic low impedance back electret cardioid microphone) was used. The speech was recorded with a Harman/Kardon cassette deck (an ultrawideband linear phase cassette deck, model hk 300xm), with Dolby noise reduction, in a quiet room. In addition, some of these individuals were asked to read a list of vowels in an hVd context which would be used later on to determine the distribution of vowels in the principal-components space. The recording level was adjusted so that the recorded voice of each individual would be at approximately the same level. Dolby noise reduction was also used during playback of the speech.

This cassette was used to obtain data from all 16 spectral band energies, from which the covariance matrix of spectral band energies was calculated. Using this covariance matrix, a set of 16-dimensional basis vectors was computed for the system. To do this, the 16-channel A-D converter was used to sample all 16 spectral band energies. Also one of the channels of the 8-channel A-D converter was used to sample the energy of the input signal for the detection of speech. It was experimentally verified that a threshold of 80 mv (i.e. a 4 on a scale of 0 to 255) would assure that the input signal was sampled only when

there was speech on the cassette. To maximize the range of the 8-bit, 0-5 volt A-D converter channels, the gain of the speech signal was adjusted so that the peak levels of the filter bank outputs were approximately 5 volts. It was also necessary to add a bank of zener diodes to the inputs of the A-D converter channels in order to prevent interactions among the A-D converter channels when overloads did occasionally occur.

Since the number of channels to be sampled was doubled, the sampling time had to be increased to 53.0 milliseconds to allow the processor to read in 17 channels and compute some partial statistics whenever there was speech on the cassette. Because of the increase in sampling time, the length of the playback had to be increased in order to obtain a sufficient number of data points. Therefore it was decided to sample the spectral band energies during the playback of both passages read by each individual. The program STAT20, contained in appendix B, was used to compute the covariance matrices.

To compute a new general set of basis vectors, and to test the speaker independency of the results, a set of experimental procedures and categories, similar to the one used for computation of the second set of basis vectors, was used. That is four categories were chosen:

(1) Individual speakers;

(2) All male speakers;

(3) All female speakers;

(4) All speakers.

The basis vectors corresponding to each category were calculated through a procedure identical to the one used for computation of the second set of basis vectors as previously discussed. Although the basis vectors computed for each individual were somewhat different

from each other, they might still span the same space. In order to determine if they were spanning the same space, a previously developed Fortran program, called EIGROT, was used to orthogonally rotate the individual speaker basis vectors toward three different group-averaged basis vectors. The three group-averaged data sets were: all-female data, all-male data, and all-speaker data. The basis vectors computed for each individual male were rotated toward the basis vectors computed from the all-male speaker data. The result of this rotation is shown in Figure 3-4. This figure shows six of the basis vectors computed from the data of all male speakers. The vertical lines crossing each basis vector represent the standard deviation as computed from the individual speaker data after rotation toward the group-averaged basis vectors, from the group-averaged basis vector. Each vertical line extends one standard deviation above and below the point where it crosses the basis vector.

Similarly the basis vectors computed for each individual female were rotated toward the basis vectors computed for the grouped female data and the result of this rotation is shown in Figure 3-5. Also the basis vectors computed for each individual speaker were rotated toward the basis vectors computed for the all-speaker data. The result of this rotation is shown in Figure 3-6. As can be seen from these figures, the first few basis vectors are relatively speaker independent and it would be logical to use the set of basis vectors computed for the all-speakers data for the system.

The computation of the previously shown basis vectors was affected, at least to some extent, by the input amplitude of the signal. Since the effect of signal amplitude may not be important perceptually, another set of 16-dimensional basis vectors was

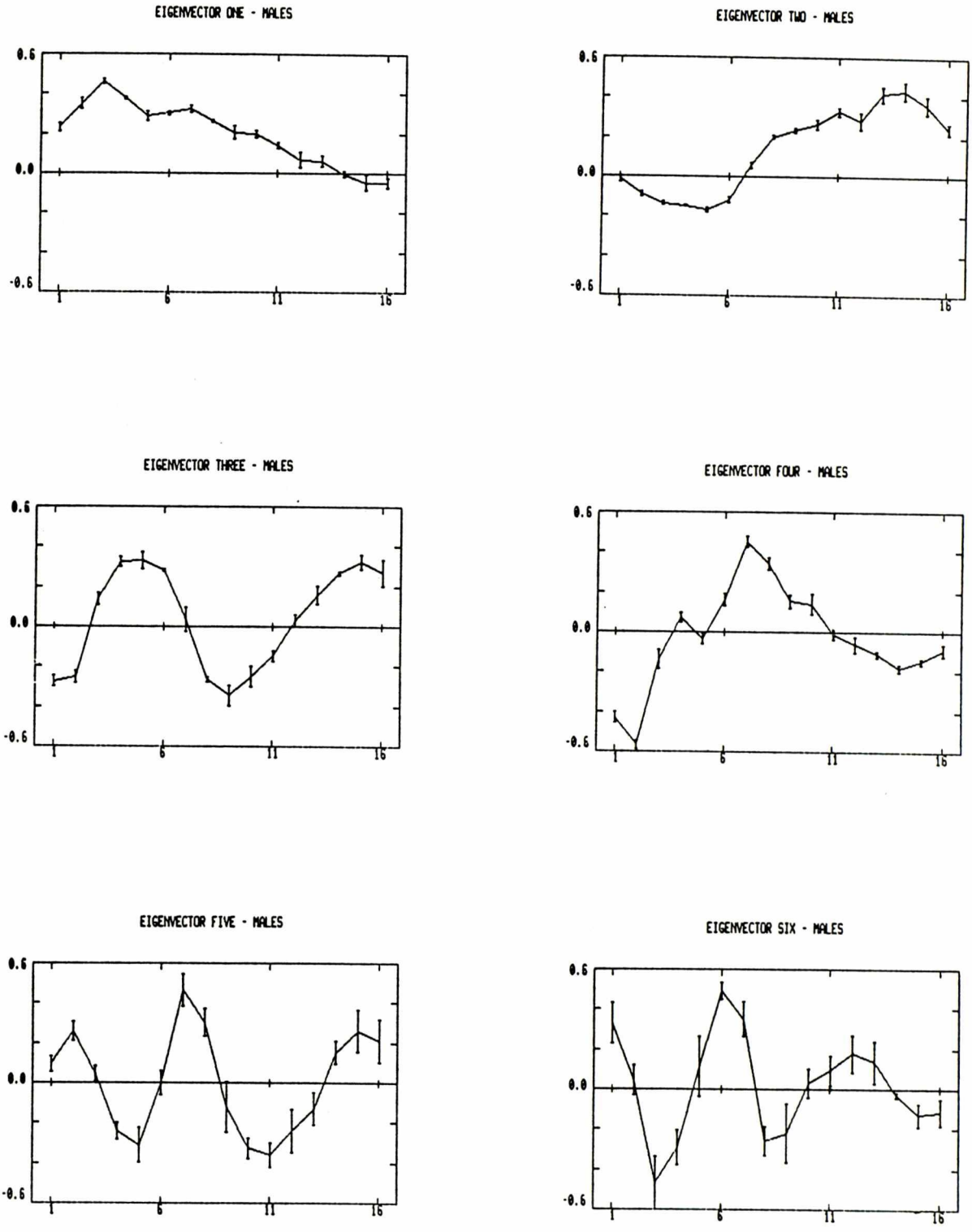



Figure 3-4. Average basis vectors for male speaker group (four speakers) using 16 non-normalized band energies. The vertical bars represent two standard deviations.

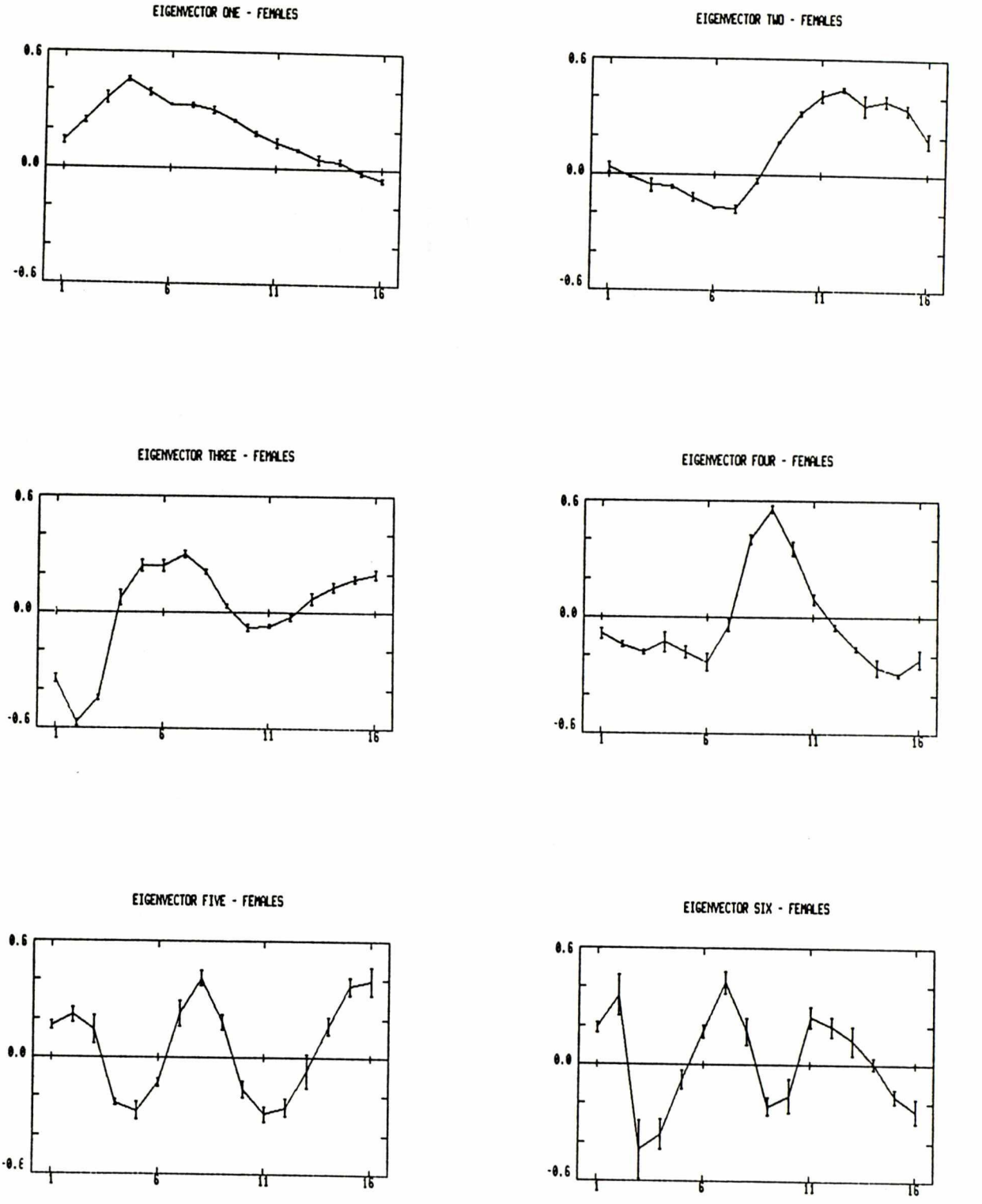



Figure 3-5. Average basis vectors for female speaker group (four speakers) using 16 non-normalized band energies. The vertical bars represent two standard deviations.

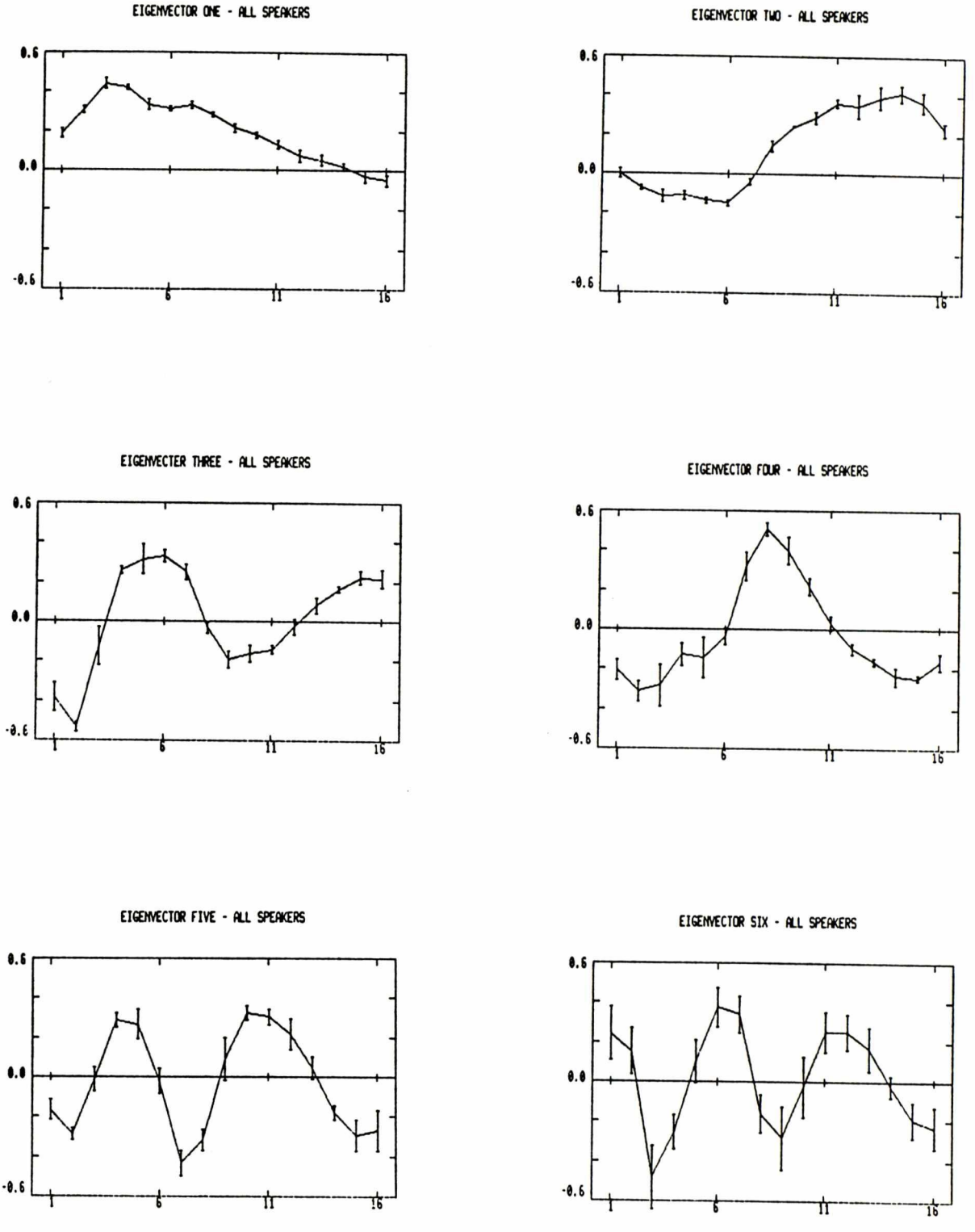



Figure 3-6. Average basis vectors for all speaker group (eight speakers) using 16 non-normalized band energies. The vertical bars represent two standard deviations.

calculated by using samples of level-normalized spectral band energies. Since it would have been difficult to level normalize the spectral band energies by hardware, the STAT20 program was modified to STAT21 to subtract the mean value of the 16-channel frequency spectrum from all 16 sampled spectral band energies and therefore obtain 16 level-normalized band energies. The mean in this case refers to the average of each group of 16 frequency samples, and was thus recomputed for each sampling of the filter bank data.

The previously described experimental procedures were used to compute new sets of basis vectors, based on level-normalized spectral band energies, for all the above categories. Then the EIGROT program was used again to rotate the new individual data sets toward the new group-averaged data sets. Results of these rotations are shown in Figures 3-7 through 3-9. Again it is seen that these new basis vectors appear to be relatively speaker independent.

Figures 3-10 through 3-12 depict the mean and variance of both non-normalized and level-normalized spectral band energies, based on statistical data, for all male speakers, all female speakers, and all speakers respectively. Unexpectedly, the data for all female speakers did not contain more high frequency content than the data for all male speakers.

Figure 3-13 depicts the cumulative variance based on the number of PC's used for both the non-normalized and the level-normalized cases. As can be seen, the relative amount of variance included in PC's for the non-normalized case is a little higher than for the level-normalized case. This difference is accounted for by the exclusion of amplitude variations in the spectrum. As can be seen from these figures, the amount of variance included in the first three

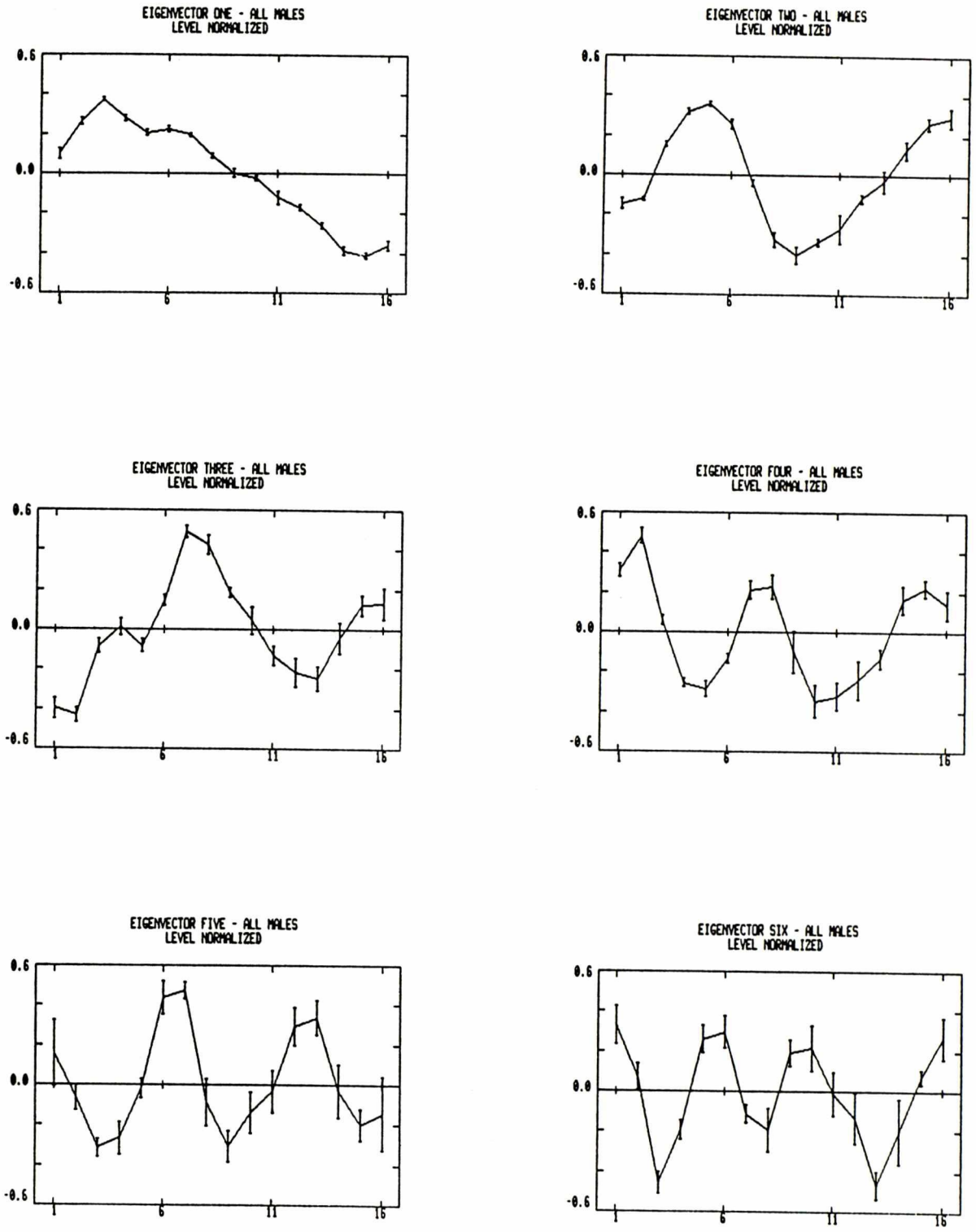



Figure 3-7. Average basis vectors for male speaker group (four speakers) using 16 level-normalized band energies. The vertical lines represent two standard deviations.

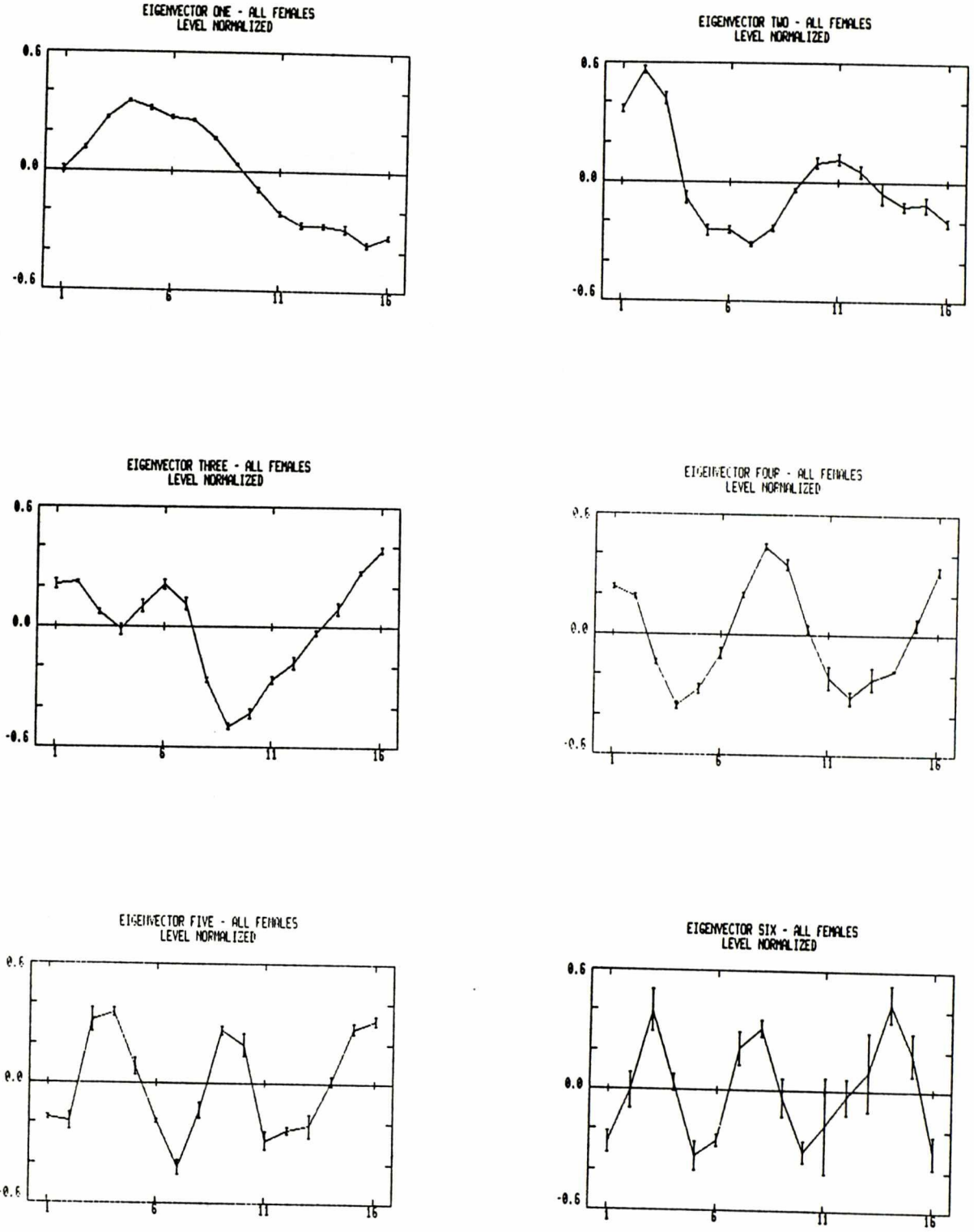



Figure 3-8. Average basis vectors for female speaker group (four speakers) using 16 level-normalized band energies. The vertical lines represent two standard deviations.

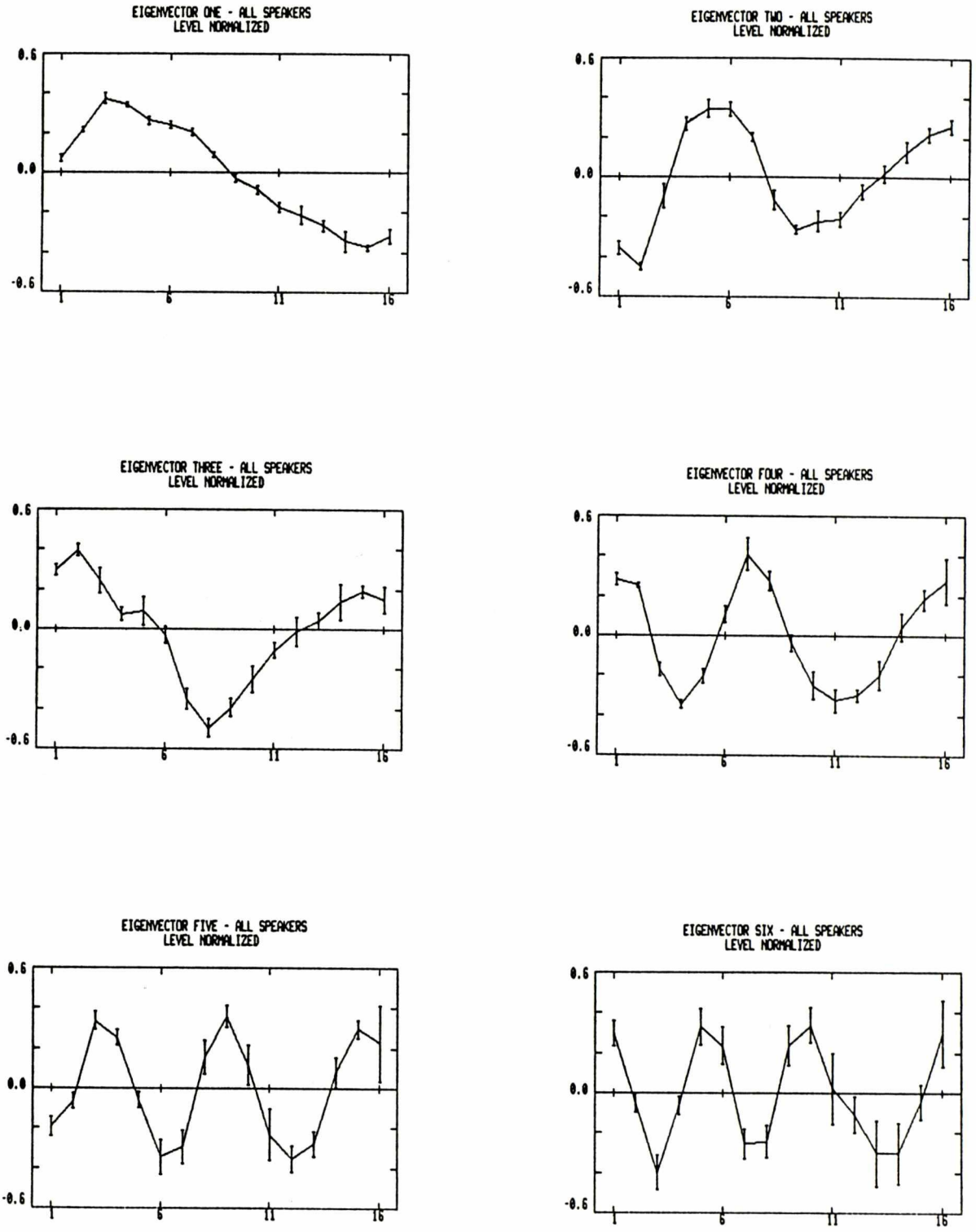



Figure 3-9. Average basis vectors for all speaker group (eight speakers) using 16 level-normalized band energies. The vertical lines represent two standard deviations.

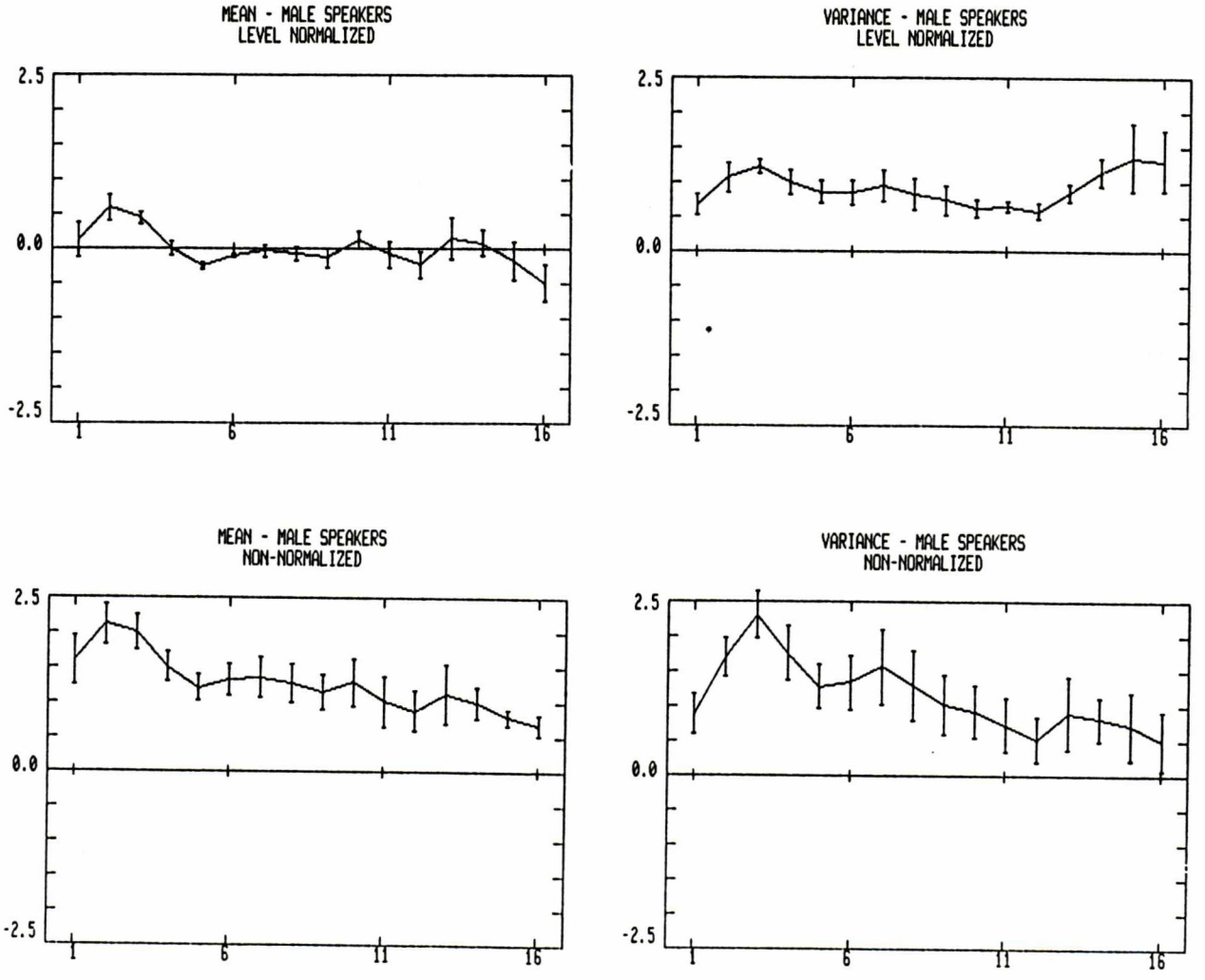



Figure 3-10. Mean and variance of 16 band energies (non-normalized and level-normalized) based on the data of male speaker group. Vertical bars represent two standard deviations.

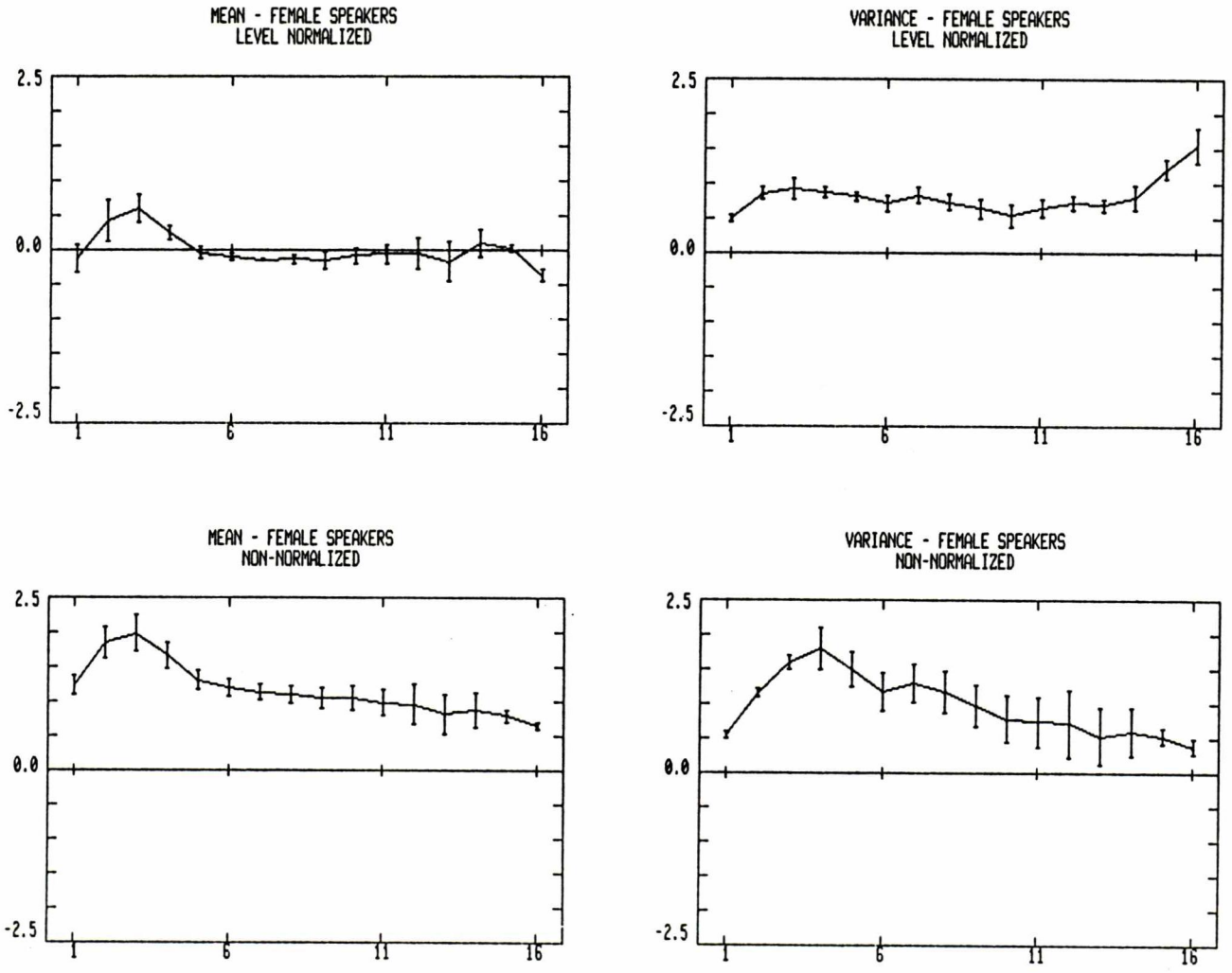



Figure 3-11. Mean and variance of 16 band energies (non-normalized and level-normalized) based on the data of female speaker group. Vertical lines represent two standard deviations.

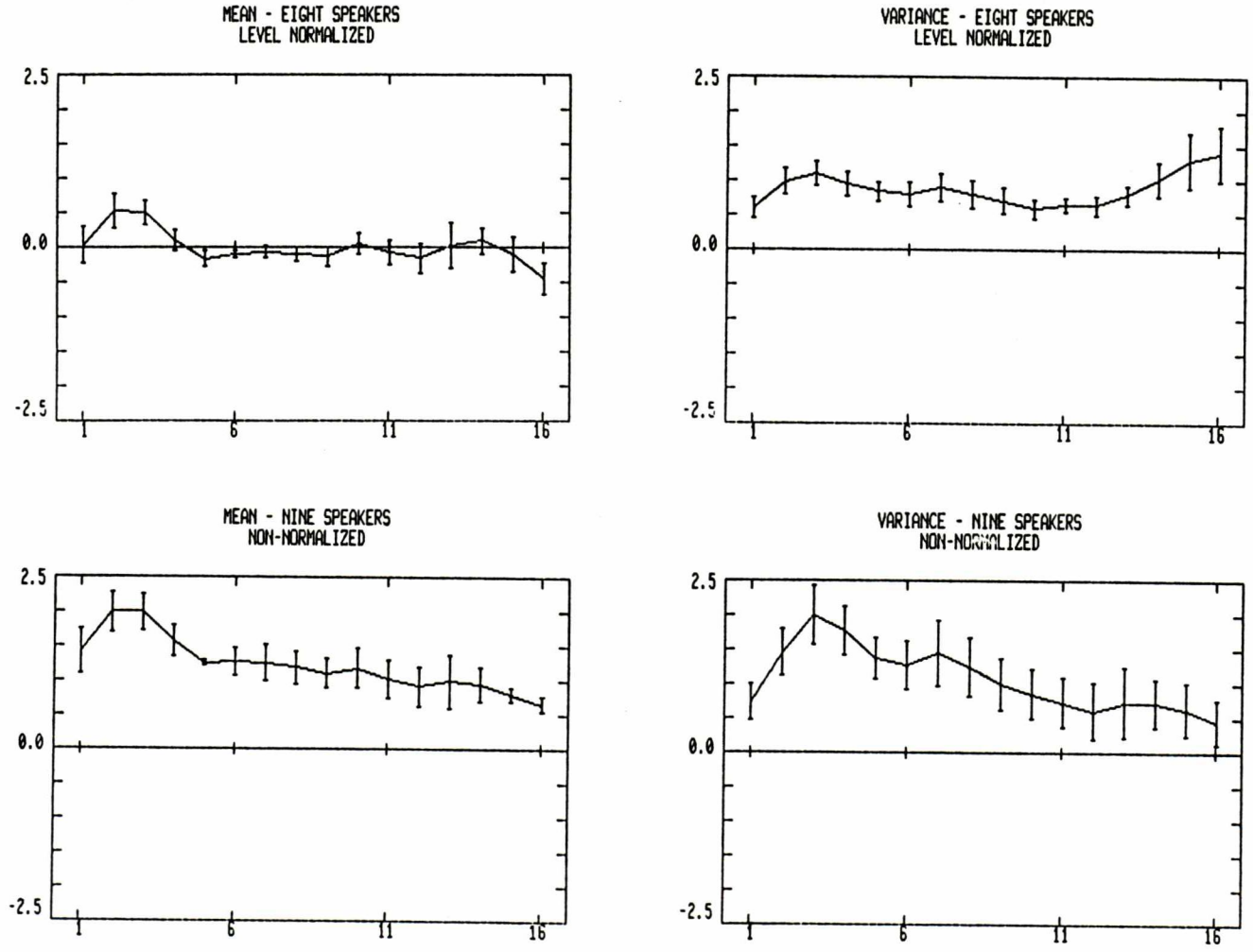



Figure 3-12. Mean and variance of 16 band energies (non-normalized and level-normalized) based on the data of all speaker group. Vertical lines represent two standard deviations.

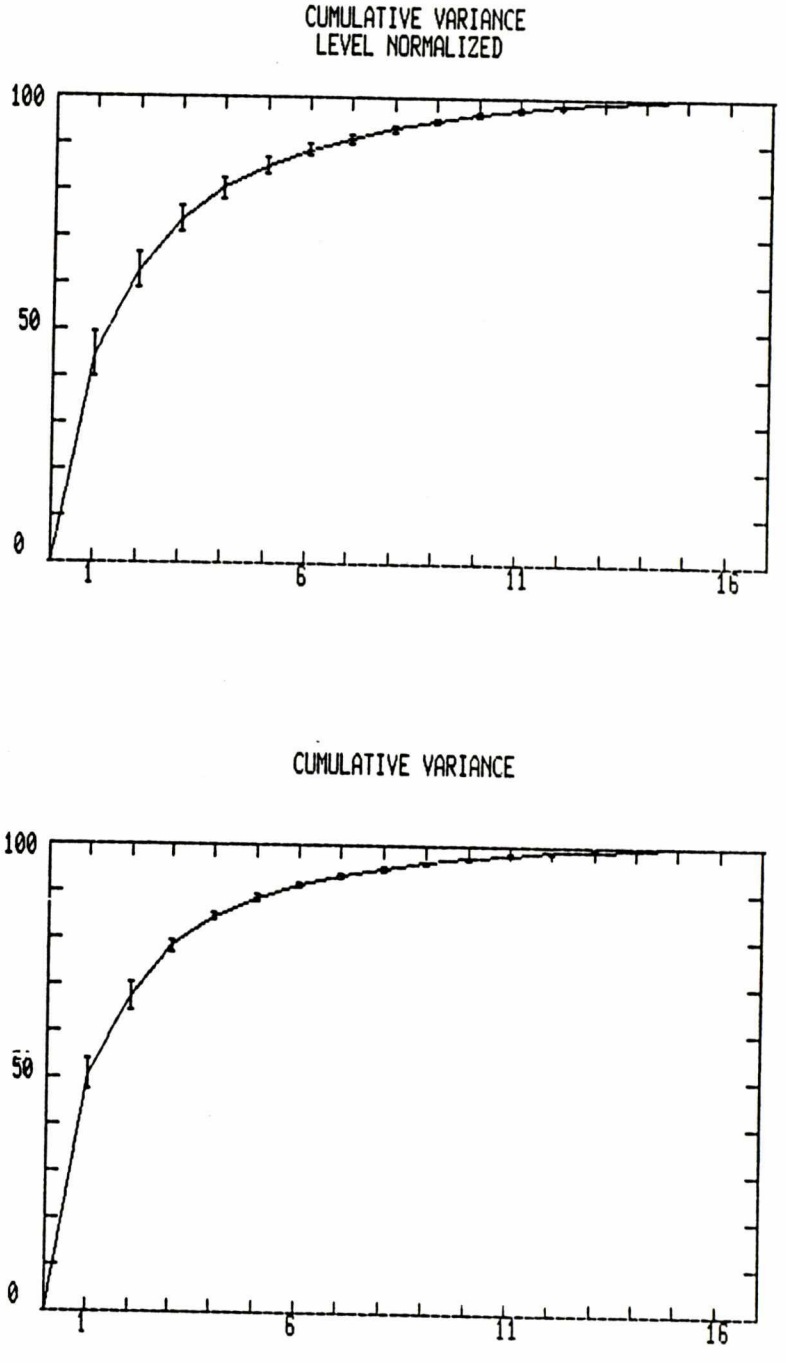



Figure 3-13. Cumulative variance of PC's for both non-normalized and level-normalized cases.

parameters for the non-normalized case is about 80% whereas for the level-normalized case it is about 74%.

The set of basis vectors, computed from the non-normalized spectral band energies of the all-speaker group-averaged data set, was incorporated into the system for further experiments.

The tape recorded speech was used to determine the correlation matrices for the new parameters. The results showed that these new parameters had, on the average, lower correlations than either of the previous cases. Table 3-3 shows the correlation matrix of PC's based on the data of the all-speaker group. The vowel-to-color experiments produced reasonable results. However some of the perceptually similar vowels resulted in similar colors. One reason for this might be that these vowels overlap in the principal-components space and therefore it would be impossible to distinguish one from the other. Therefore we had to determine if vowels are well clustered in the principal-components space.

## 3.3 Vowel Experiments

Since principal-components basis vectors computed from non-normalized spectral band energies are different from principal-components basis vectors computed from level-normalized spectral band energies, the distribution of vowels in both principal-components spaces was investigated in order to compare the two approaches. Note that not only are the two sets of basis vectors different, but also the relative signal level will affect the results for the non-normalized case whereas the results for the level-normalized case should be independent of signal level. Since

differences in amplitude would probably not be useful in determining vowel identity, the level normalization would seem to be a better approach.

The program VOWEL was written to efficiently measure the principal components values for vowels, beginning with the non-normalized spectral band energies. This program has two modes of operation which can be selected via keyboard. In the first mode, 25000 samples of the speech signal are acquired with a sampling frequency of 10000 Hz (2.5 seconds). Prior to sampling, the speech signal is lowpass filtered with a cut-off frequency of 5000 Hz, and then adjusted to vary between zero to four volts. It was also possible to repeatedly output any desired block of data of length 1000 samples (100 ms), on one D-A channel, and a sync pulse on another D-A channel. Thus the desired segment of data could be "frozen" and examined in detail using an oscilloscope. In the second mode, while the desired block of data was played back once, five of the principal components were measured at the end of this single play back. We assumed that the total sampling time for all parameters (around 350 microseconds) was short compared to the decay time of the system lowpass filters. This assumption and the program operation were verified by checking the results with an oscilloscope. (See appendix B for a listing of this program).

Since the level normalization of spectral band energies through the use of hardware was more difficult than with software, the VOWEL program was modified to BAND16 to first level normalize the spectral band energies and then compute all necessary principal components. The set of basis vectors computed from level-normalized spectral band energies were properly scaled for maximum accuracy and used in the

program as constants. The operation of the BAND16 program is identical to that of the VOWEL program except that in its second mode, it measures the spectral band energies, rather than the principal components, and then computes all necessary principal components by using the scaled constants. (See appendix B for a listing of the VOWEL program).

The recorded vowels on the cassette were used to find the distribution of vowels in both principal-components spaces. The VOWEL program, written for non-normalized spectral band energies, was used to sample one to two vowels. Through experimentation, we found that the vowel part of each word usually started about 100 milliseconds after the word started and lasted between 250 to 300 milliseconds. The studies by Pols (1973) suggested that the parameters should be sampled around 100 milliseconds after the start of the vowel itself (with a sampling frequency of 10000 Hz, it would be around the $1000^{th}$ sample of the vowel). The program was directed to focus on the block of data beginning with 100th sample of the vowel. That is, the principal components were measured 110 milliseconds after the start of the vowel itself. The PC's were sampled at 20 milliseconds intervals over a total of 100 to 140 milliseconds. The final values were obtained by averaging the section of sampled values that had minimum variations. The output level of the cassette deck was set to the same level for all the speakers. The result of this experiment is shown in Figure 3-14. The data points were obtained from four male speakers and four female speakers. Depicted in this figure are the clustering of the three vowels /a/, /i/, and /u/ in P1-P2, P1-P3, and P2-P3 planes. Also depicted in the bottom right corner of this figure are the projection of those vowels on an orthogonally rotated plane.

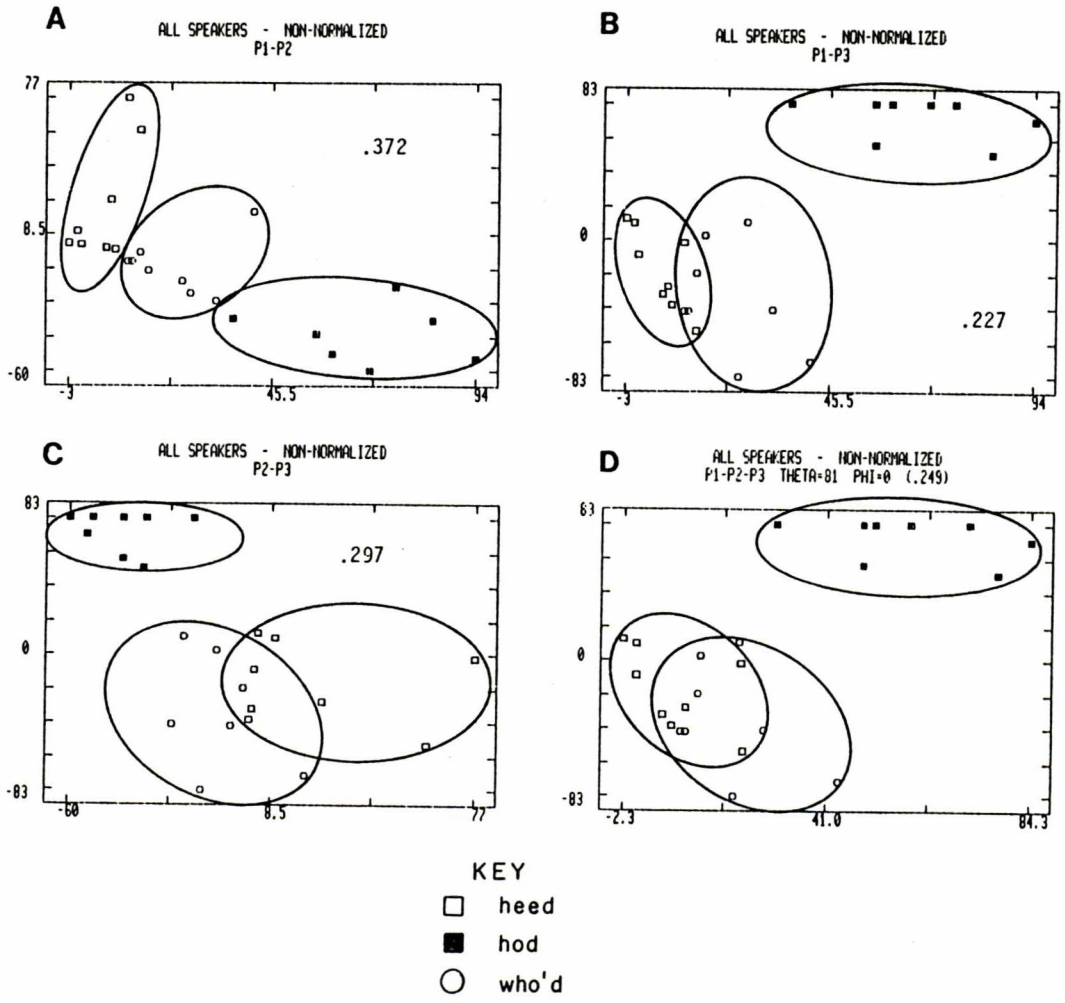



Figure 3-14. Scatter plots of vowel data in various PC planes. The PC's were computed from non-normalized filter bank data. For each plane, the ratio of the within class vowel variance to between class vowel variance is given. The plane in the lower right of the figure is a P2-P3 plane after orthogonal rotation of the P2 axis by 81 degrees away from the P1 axis.

This plane was determined such that the ratio of the within vowel class variance to between vowel class variance was minimized by orthogonally rotating two of the three axes. Note that a measure of clustering is also specified for each of the four planes in this figure. For the data shown, the P1-P3 plane had the minimum value for that measurement. However the data appears to be somewhat better clustered in the P1-P2 plane.

The BAND16 program is identical to the VOWEL program except that it measures the spectral band energies instead of the principal components, and it computes the principal components by using the constants that represent the set of basis vectors which were computed from the level-normalized band energies. The vowel distribution in this principal-components space was also found and the results are shown in Figure 3-15. The same data points as for the previous experiments were used in this experiment. The clustering of vowels in P1-P2, P1-P3, and P2-P3 planes are depicted in this figure as well as the clustering of vowels in an orthogonally rotated plane. The measure of clustering is also specified for each of these planes. It can be seen these three vowels are clustered better when level-normalized spectral band energies are used to compute the principal components. This conclusion also holds when more than three vowels are considered. Therefore the vowel to color experiments are expected to produce a better result if level-normalized band energies are used.

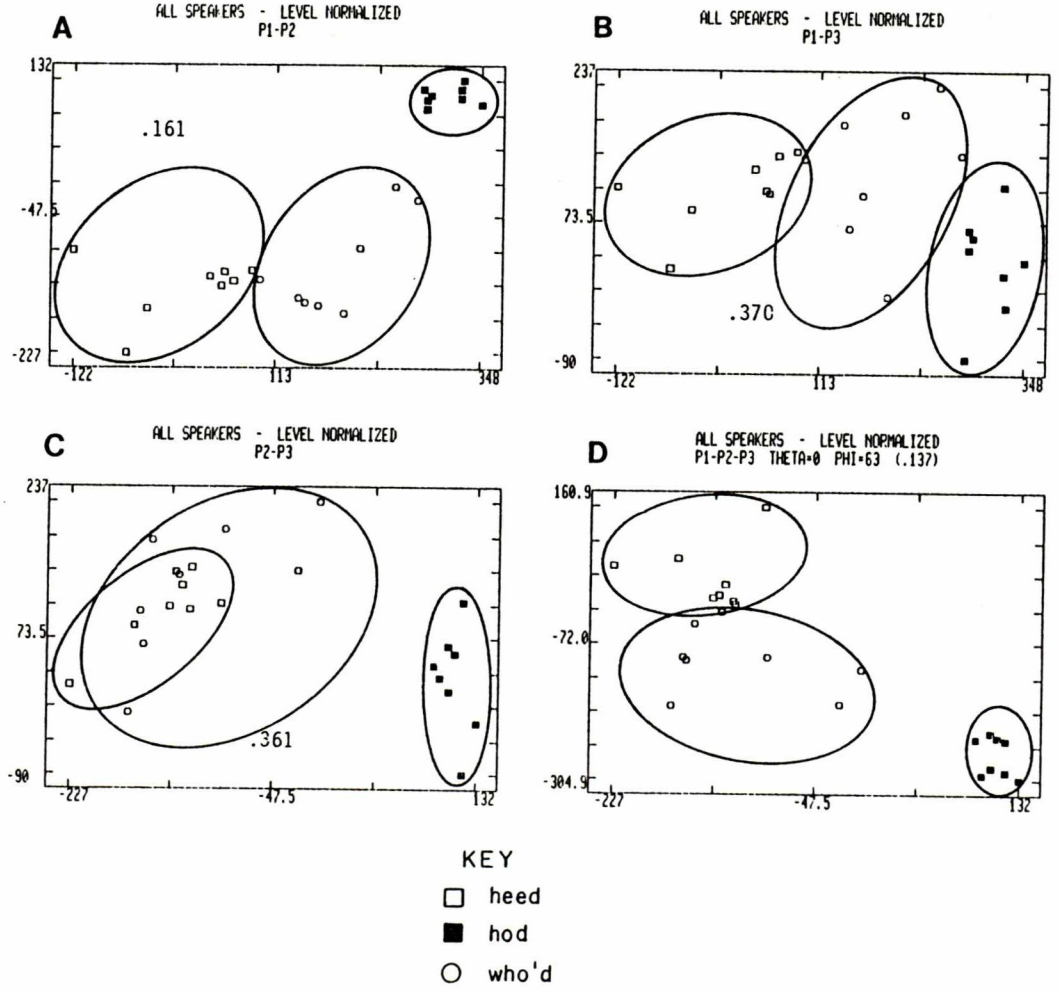



Figure 3-15. Scatter plots of vowel data in various PC planes. The PC's were computed from level-normalized filter bank data. For each plane, the ratio of the within class vowel variance to between class vowel variance is given. The plane in the lower right of the figure is a P2-P3 plane after orthogonal rotation of the P2 axis by 63 degrees toward the P1 axis.

# CHAPTER 4

## VISUAL DISPLAY OF VOWEL SPECTRA

In this chapter, the details of the procedure used to convert vowel spectra, as measured by principal components, to visual display parameters is presented. As discussed in the INTRODUCTION of the thesis, information in the display will primarily be communicated through color. The color space is considered to be a three dimensional space, represented by either hue, luminance, and saturation or the levels of the three primary colors red, green, and blue. In addition the experimental results reported in CHAPTER 3 indicate that vowel data is well represented in a three dimensional principal-components space. Thus a linear transformation was developed from the 3-dimensional speech parameter space to the 3-dimensional color space such that widely separated vowels produce widely separated colors.

It should also be noted that speech spectra in normal speech, and thus principal components vary slowly with time. The approximate bandwidth of the principal components is 30 Hz, implying that the parameters should be sampled at a 60Hz or greater sampling rate. Thus the transformation must also be performed at this rate in order to update the visual display. Since the sense of vision is more suited to spatial distinctions rather than rapidly varying temporal variations, a "flow-mode" display was implemented. That is, new information continuously enters on one side of the screen and flows

across the screen to the opposite side. Therefore time is represented by spatial position in the display.

Both the mathematical procedure for determining the transformation and the implementation of this transformation with a real-time display format are presented in this chapter. Two somewhat different display formats were developed. In one of these formats the energy of the input speech is used to determine the height of a bar of color which flows across the screen. Hence, for this particular display, four rather than three speech parameters control the display.

## 4.1 Linear Transformation From Speech Parameters to Color

The mapping of vowels into colors can be described by matrix equation 4.1. In this equation V is referred to as the vowel matrix which is represented by principal components; C is the color matrix represented by levels of the three primary colors red, green, and blue, and the T matrix is a linear transformation which maps the vowel space into the color space.

$$[\,C\,] = [\,T\,]\,[\,V\,] \qquad (4.1)$$

Therefore if T is known, equation 4.2 can be used to map a particular vowel into its corresponding color in the color space. The T matrix must be chosen so that perceptually similar vowels result in perceptually similar colors, while widely separated vowels result in widely separated colors.

The choice of the linearity of the transformation is arbitrary. The transformation was chosen to be linear since it is straightforward to compute and is much easier to work with. Although the transformation could have been chosen to be nonlinear, it probably would not have produced better results.

COLOR VOWEL

$$\begin{bmatrix} \text{Red} \\ \text{Green} \\ \text{Blue} \end{bmatrix} = \begin{bmatrix} \quad T \quad \end{bmatrix} \begin{bmatrix} P1 \\ P2 \\ P3 \\ \cdot \\ \cdot \\ \cdot \\ P16 \end{bmatrix} \qquad (4.2)$$

In the above equation, all of the principal components are used to represent speech. However, as discussed in Chapter 3, the first three principal components contain about 80% of the variance of the speech spectral data and therefore it seems reasonable to represent speech by the first three principal components. That is, only the first three principal components are used to map vowels into colors. In this particular case, the dimensions of the C and V matrices are compatible, thus implying that the T matrix will be a three by three square matrix. The T matrix is chosen so that the origin of the vowel space maps to the origin of the color space. In addition three arbitrarily specified points in the vowel space map to three arbitrarily specified points in the color space. This transformation

is depicted in Figure 4-1. From a geometric point of view, the transformation is a translation, rotation (possibly not orthogonal), and scaling of the coordinate system.

Although the origin of the vowel space is somewhat arbitrary, a logical choice would appear to be silence. Similarly black, which represents no color, is chosen to be the origin of the color space. The selection of the origins is expected to play a significant role in the final performance of the mapping of vowels into colors.

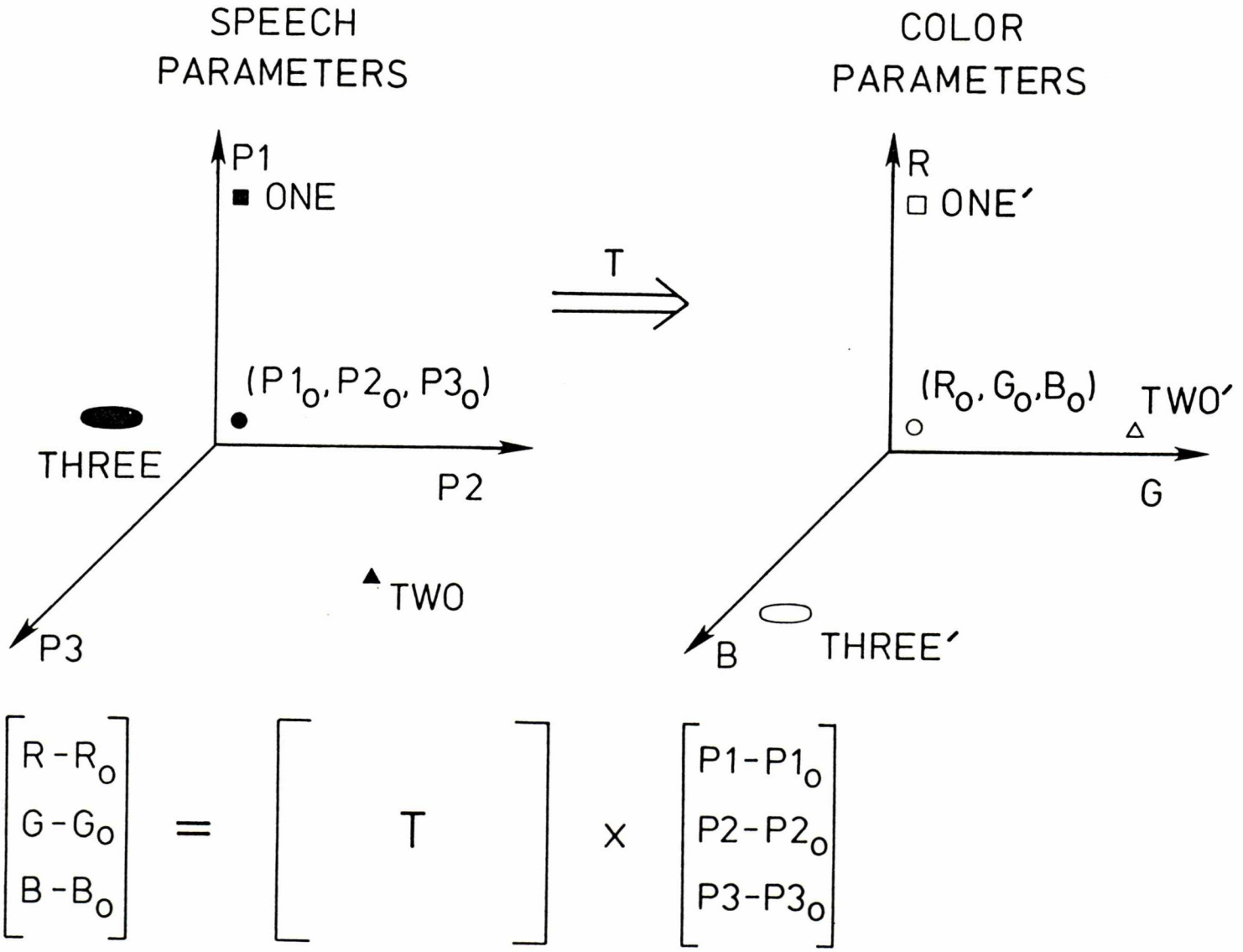
Since the transformation maps the vowel origin into the color origin, the deviation of vowels from the vowel origin, will be mapped into color deviations from the color origin. The red, green, and blue levels of the color origin can then be added to obtain the color which represents that particular vowel. Equations 4.3 and 4.4 describe this process.

$$\underline{\text{COLOR DEVIATION}} \qquad\qquad \underline{\text{VOWEL DEVIATION}}$$

$$\begin{bmatrix} R-R_0 \\ G-G_0 \\ B-B_0 \end{bmatrix} = \begin{bmatrix} \quad T \quad \end{bmatrix} \begin{bmatrix} P1-P1_0 \\ P2-P2_0 \\ P3-P3_0 \end{bmatrix} \qquad (4.3)$$

$$\begin{matrix} \text{COLOR} \\ \underline{\text{DEVIATION}} \end{matrix} \qquad \begin{matrix} \text{COLOR} \\ \underline{\text{ORIGIN}} \end{matrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} R-R_0 \\ G-G_0 \\ B-B_0 \end{bmatrix} + \begin{bmatrix} R_0 \\ G_0 \\ B_0 \end{bmatrix} \qquad (4.4)$$




$$\begin{bmatrix} R-R_o \\ G-G_o \\ B-B_o \end{bmatrix} = \begin{bmatrix} \quad T \quad \end{bmatrix} \times \begin{bmatrix} P1-P1_o \\ P2-P2_o \\ P3-P3_o \end{bmatrix}$$

Figure 4-1. Linear transformation from PC's to color parameters.

Matrix equation 4.5 represents the general form of matrix equation 4.3.

$$[\ C'\ ] = [\ T\ ][\ V'\ ] \qquad (4.5)$$

In order to determine the T matrix, the C' and V' matrices were first determined as follows. Since the T matrix has to be a three by three square matrix, matrix equation 4.5 can be solved for T if the V' matrix is a square matrix (in this case a three by three square matrix) and if the C' matrix is also three by three. Equation 4.6 describes the solution for T in terms of the C' and V' matrices.

$$[\ T\ ] = [\ C'\ ][\ V'\ ]^{-1} \qquad (4.6)$$

Each column of V' is comprised of parameter deviations from the origin for a particular vowel. Similarly each column of C' is comprised of color deviations from the origin for a particular color. To determine the T matrix, three arbitrary vowels are selected. Also three arbitrary colors are selected which will correspond to vowels selected. Equation 4.7 is the matrix equation describing these assignments. The T matrix, which can be determined by equation 4.6, forces the three vowels to map into the three color deviations specified on the left hand side of the equation.

$$\begin{bmatrix} R_1-R_0 & R_2-R_0 & R_3-R_0 \\ G_1-G_0 & G_2-G_0 & G_3-G_0 \\ B_1-B_0 & B_2-B_0 & B_3-B_0 \end{bmatrix} = \begin{bmatrix} \quad T \quad \end{bmatrix} \begin{bmatrix} P1_1-P1_0 & P1_2-P1_0 & P1_3-P1_0 \\ P2_1-P2_0 & P2_2-P2_0 & P2_3-P2_0 \\ P3_1-P3_0 & P3_2-P3_0 & P3_3-P3_0 \end{bmatrix} \qquad (4.7)$$

## 4.2 Procedure to Determine Transformation

A general computer program named "MINA" was written using PLM86 to both determine the transformation and to implement the vowel-to-color display based on the transformation in real-time. The program allows any specified color to be observed on the color monitor before it is assigned as one of the target colors. To determine the vowel origin (the no sound values of the principal-components parameters), the program samples the principal components when there is silence and assigns the sampled values as the origin. These are the DC levels of the principal components. The gain and offset board of the speech parameter extractor allows the DC levels of the principal components to be adjusted so that maximum dynamic range can be attained for the parameters.

Since the output level of the log amplifier may drift with time, and thus affect the DC level of the principal components, the vowel origin is determined every time the program is started. The program then takes a specified number of samples (up to 255) of the principal components for three separate vowels produced by a single individual, one vowel at a time. The data from each vowel over the specified interval is averaged in order to compute estimates of the parameter deviation values for each vowel.

Matrix inversion is easiest to implement in floating point arithmetic. Since floating point computations could not readily be accomplished with the microprocessor system, the matrix was inverted on the PDP11 minicomputer. A FORTRAN program called "INVERSE" was written to calculate the transformation matrix. This program uses the matrix inversion and the matrix multiplication programs already

available on the PDP11. The program allows both target vowels and target colors, which define the transformation, to be specified as input parameters. Thus the transformation can easily be "tuned" for individual speakers. The parameter deviation values, displayed by the "MINA" program, are entered into the "INVERSE" program in order to calculate the transformation matrix.

Even though this transformation is found by using only values of the first three principal components, more than three principal components can also be used to find a transformation which maps these parameter values into the color space. That is, if N PC's are used, N target vowels and N target colors are selected. Then T is determined so that target vowels map to target colors. However experiments indicated that the mapping was more speaker dependent if more than three PC's were used.

Table 4-1 shows typical experimental data, including parameter values for target vowels. Three speakers were used to obtain the data in this table (a 4 year old child, an adult male and a female). The transformations computed based on the parameter values of Table 4-1 are listed in Table 4-2. Also included in the table is a measure of the orthogonality of the transformations. The orthogonality of these transformations was investigated by multiplying each transformation matrix by its transpose. The degree of orthogonality can be determined by considering the closeness of the resulting matrices to the identity matrix. If the diagonal elements of the resulting matrix are much larger than the rest of its elements, then it can be said that this transformation is almost orthogonal. By using the above procedure, these transformations were found to be non-orthogonal.

Table 4-1. Typical values of PC's for target vowels /a/, /i/, and /u/. These values are the deviations from the vowel origin(silence). Three speakers were used. Also included in this table is a list of target colors assigned to target vowels.

(P1,P2,P3) for silence=(24,111,173)

| VOWEL | P1 | P2 | P3 | |
|---|---|---|---|---|
| /a/ | 138 | 18 | 66 | |
| /i/ | 141 | 73 | -57 | (4 year old child) |
| /u/ | 82 | -14 | -27 | |

| VOWEL | P1 | P2 | P3 | |
|---|---|---|---|---|
| /a/ | 110 | 21 | 40 | |
| /i/ | 19 | 109 | -39 | (male speaker) |
| /u/ | 48 | -62 | -57 | |

| VOWEL | P1 | P2 | P3 | |
|---|---|---|---|---|
| /a/ | 90 | 21 | 33 | |
| /i/ | -17 | 92 | -26 | (female speaker) |
| /u/ | 58 | -61 | -70 | |

| TARGET VOWEL | TARGET COLOR |
|---|---|
| /a/ | Red |
| /i/ | Green |
| /u/ | Blue |

Table 4-2. Typical transformations computed based on the parameter values listed in Table 4-1. Coefficients of three transformations are listed. All the coefficients are scaled up by a factor of 20.

$$T = \begin{bmatrix} 7 & -2 & 46 \\ -4 & 46 & -81 \\ 35 & -23 & -28 \end{bmatrix}$$ (4 year old child)

$$[T][T]^T = \begin{bmatrix} 2169 & -3846 & -997 \\ -3846 & 8693 & 1070 \\ -997 & 1070 & 2538 \end{bmatrix}$$

$$T = \begin{bmatrix} 27 & 4 & 28 \\ -6 & 33 & -15 \\ 18 & -34 & -42 \end{bmatrix}$$ (male speaker)

$$[T][T]^T = \begin{bmatrix} 1529 & -450 & -826 \\ -450 & 1350 & -600 \\ -826 & -600 & 3244 \end{bmatrix}$$

$$T = \begin{bmatrix} 33 & 1 & 25 \\ 4 & 44 & -5 \\ 13 & -39 & -43 \end{bmatrix}$$ (female speaker)

$$[T][T]^T = \begin{bmatrix} 1715 & 51 & -685 \\ 51 & 1977 & -1449 \\ -685 & -1449 & 3539 \end{bmatrix}$$

## 4.3 Display Implementation

After the transformation matrix is calculated by the "INVERSE" program, it is entered into the "MINA" program. The program then operates in a real-time mode and maps vowels to colors using the specified transformation.

Since the "MINA" program uses fixed point arithmetic, an overflow at any stage of the calculation could produce unpredictable results. Therefore both the data and the coefficient matrix had to be scaled to give good resolution without danger of overflow.

Initially the graphics display controller card was not available and the "MINA" program was written so that it would not only process the data but also display the data and generate the vertical sync pulses. This program has a flow mode display, such that new data appears at the top of the display and flows down the display. The display can be frozen to be studied by flipping a switch which sets the input to one of the A-D channels high. The display starts to flow when the same switch is flipped back to its original position. With this implementation, only one display area is presented and the displayed data has no information but color. Also the sampling frequency cannot be changed because it must remain at 60 Hz in order for the interrupt pulse to drive the vertical sync of the color monitor. The real-time processing is accomplished with an interrupt routine. However a practical display should have two separate and individually scrollable display areas so that the color pattern obtained from a teacher's voice can be frozen in one display area while the student tries to best match the frozen color pattern. The

student's color patterns appear in the other display area and can be frozen to be studied by the student or the teacher.

The NEC7220 graphics display controller card appeared to be the solution to this problem. It generates all necessary sync pulses and has two separate display areas. Each display area can be scrolled or panned independently. The display of data is relatively independent of the system microprocessor and memory. It has 128 Kbytes of bit-mapped display memory which provides a very high resolution display. Its only disadvantage is that it only allows a maximum of 16 colors to appear on the screen at any one time. As discussed in Chapter 2, a method was devised to take advantage of the high spatial resolution of the GDC and produce a large number of colors by assigning proper colors to horizontally adjacent pixels. With the method used, it is possible to specify a value from zero to eight to each of the three primary colors, thus providing (9*9*9=729) total possible colors on the screen at any point in time. (Actually, the number of usable colors is less than 729, since some of the combinations are not possible).

The program "MINA" was modified to include use of the GDC and write information into the GDC memory for display. In this new version, called "HAMID", all the calculations are exactly the same as for the "MINA" program. The only differences are that the information is represented as vertical bars having a color that is mapped from the sampled values of the principal components at the sampling instant with a height proportional to the energy of the signal at that sampling instant. These vertical bars first appear on the right side of the screen and flow to the left with a speed proportional to the sampling frequency. This new program also allows the sampling time to

be specified. The minimum sampling time was experimentally determined to be about 16 milliseconds. For this program the flow time is 128*(sampling time). Typically a sampling time of 25 milliseconds was used, corresponding to a flow time of 3.2 seconds.

## 4.4 Distribution of Vowels in the Color Space

In order to implement the transformation described in this chapter, a decision had to be made as to what colors should be assigned to which vowels. At first these assignments were arbitrary. However, limited experiments indicated that the assignment of the three primary colors red, green, and blue to the three widely separated vowels /a/, /i/, and /u/ respectively worked the best. That is, vowels became more separated in the color space. Figure 4-2 depicts four color bars, each with a different color and each representing a different vowel, as produced by the "MINA" program. As can be seen in this figure, color is the only variable of the display.

Figure 4-3 depicts color patterns representing different vowels as produced and displayed by the "HAMID" program. One can immediately observe the advantage of this display which uses the GDC over the previous display which did not use the GDC. However in both cases the assignment of the three primary colors red, green, and blue to the three vowels /a/, /i/, and /u/ produced the best result. With these assignments other vowels generally produced other colors. For example, the vowel in the word "heard" produced a violet color, the vowel in word "had" produced an orangish yellow color, and the vowel in word "hid" produced a cayan color.

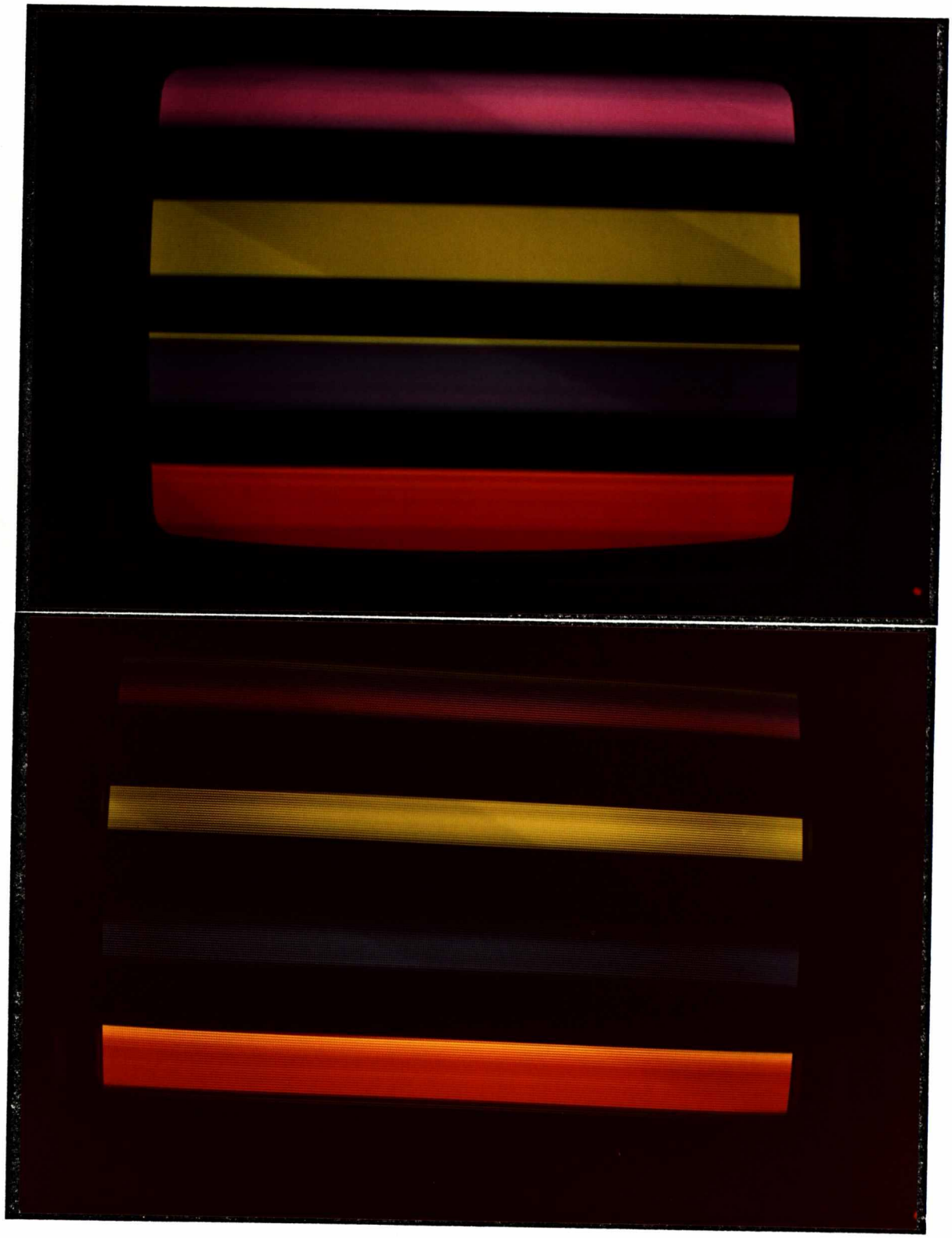


Figure 4-2. Typical display formats as produced by "MINA" program. Each picture depicts four horizontal color bars (bottom to top) representing the vowel parts of the words hod, heed, who'd, and heard respectively. Note that the program used a single transformation for both the male speaker (top picture) and the female speaker (bottom picture).

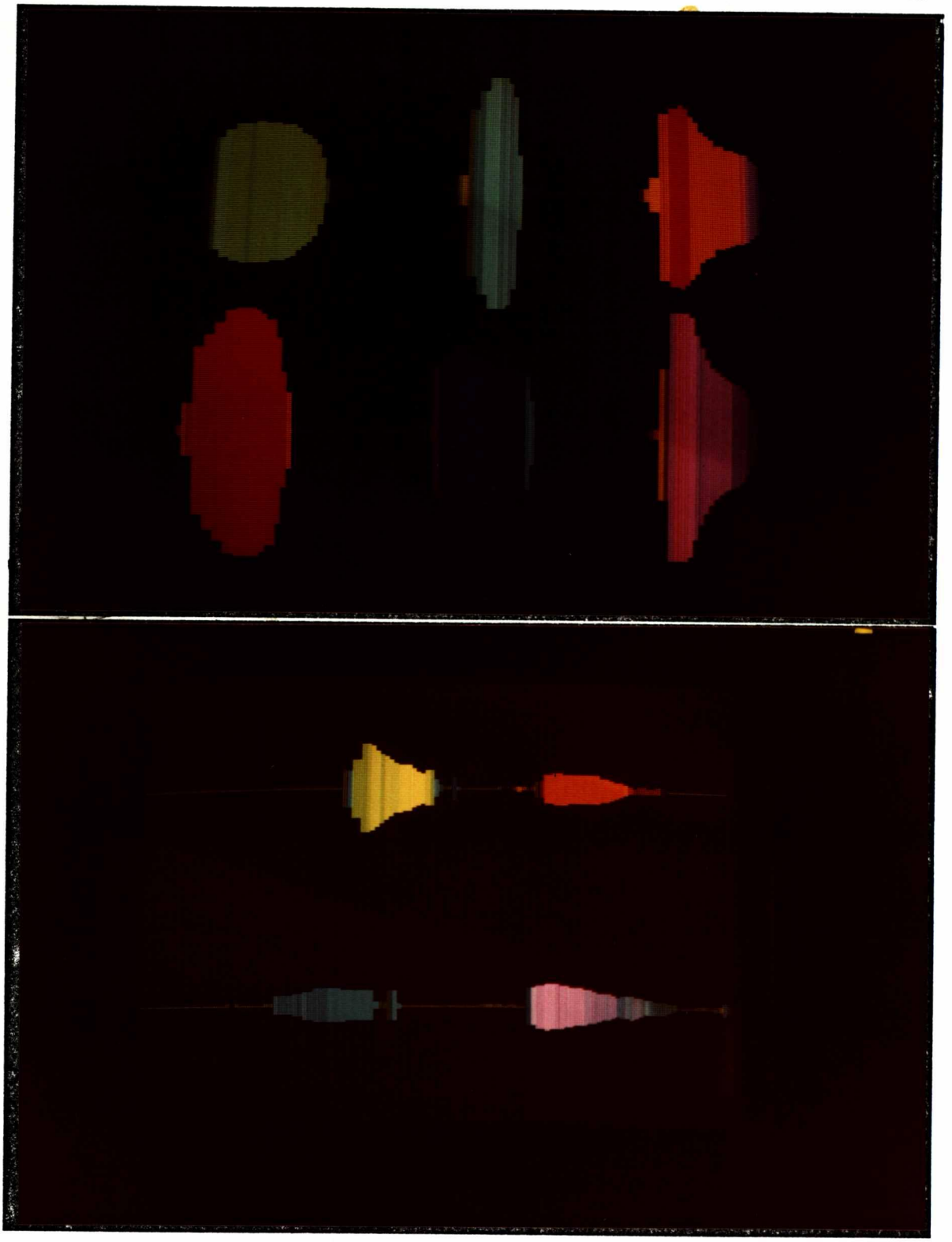


Figure 4-3. Typical display formats as produced by "HAMID" program. The top picture depicts six color patterns, three in each display area. The top display area represents the vowel part of the words heed, hid, and had (left to right) respectively. The bottom three color patterns represents the vowel part of the words hod, who'd, and heard respectively. A male speaker was used for this picture. A 4 year old child was used in the bottom picture. Top pattern: vowels in words heed and hod. bottom pattern: vowels in words who'd and heard. Note that a single transformation was used for both speakers.

# CHAPTER 5

# EXPERIMENTAL TESTING OF OVERALL SYSTEM AND RESULTS

Informal testing of the most recent display, described in Chapter 4, indicated that with a single fixed vowel-to-color transformation, the same vowel as spoken by a variety of speakers, appears as a very similar pattern, and different vowels generally appear as different patterns. The most recent display format appears to communicate information more effectively than the previous display format, which was tested during the development stages. It also has the capability of a split screen which is of absolute necessity for a teacher/student mode of operation. Therefore it was decided to utilize this recent display format in order to test the effectiveness of a color display as a training aid. In this chapter, the experimental procedures for this test of the display are presented. Because of time constraints, only two pilot experiments were conducted and their results are presented and analyzed.

## 5.1 Experimental Procedures and Results

Ten ODU students from a variety of ethnic backgrounds were selected as experimental subjects. They were each given about ten minutes of familiarization and practice with the visual speech display. In these experiments the seven words heed, hid, had, head, hod, heard, and who'd were used. Note that these words are all of the

form hVd, and thus vary only in the vowel part of the word. In the first experiment the ability of subjects to discriminate vowels based on the color pattern of the display was measured using an ABX paradigm. The patterns corresponding to vowels A and B were displayed on the display as well as the pattern corresponding to vowel X, which was either vowel A or vowel B. The speaker for vowel X was always different than the speaker for vowels A and B. If a male speaker was used for vowels A and B, a female speaker was used for vowel X. Similarly, if a female speaker was used for vowels A and B, a male speaker was used for vowel X. Subjects had to identify, in real-time as color patterns flowed across the screen, the third pattern (X) as being either more similar to pattern A or pattern B. Results of this experiment are shown in Table 5-1. The results indicate that about 95% of the discriminations were correct.

In the second experiment, called the word identification test, the same seven words as used in the first experiment, again varying only in the vowel part of the word, were recorded on a cassette tape by each of two speakers (one male and one female). The cassette tape was then played back through the visual display system and the pattern corresponding to one of the words was frozen on one display area. A hearing subject was asked to identify the word based on its color pattern. Subjects were given a list of the seven possible words. The subjects used a microphone and attempted to duplicate the frozen color pattern with their own voice on the other half of the screen. They were given as much time as needed to choose the word that best matched the color pattern and that word was recorded. The results of this experiments are shown in Table 5-2. These results indicated that in 50% of the cases the words were matched correctly.

ABX Word Comparison

Table 5-1. Confusion matrix resulting from an ABX experiment for identifying the visual representation of seven vowel stimuli. Ten "viewers" were used as subjects.

| | HAD | HEAD | HEARD | HEED | HID | HOD | WHO'D |
|---|---|---|---|---|---|---|---|
| HAD | 95% | | 20% | | | | |
| HEAD | | 90% | 30% | | 10% | | |
| HEARD | 10% | 35% | 88.6% | | | | 10% |
| HEED | | 10% | | 98.3% | | | |
| HID | | 10% | | 10% | 96% | | |
| HOD | 30% | | | | | 95.7% | |
| WHO'D | | 10% | 10% | | | | 96% |

Word Identification

Table 5-2. Confusion matrix resulting from an open-ended identification and matching experiment of seven vowel stimuli based on the visual speech display. Two speakers (one male, one female) were used to generate the stimuli. Ten subjects were used.

| | HAD | HEAD | HEARD | HEED | HID | HOD | WHO'D |
|---|---|---|---|---|---|---|---|
| HAD | 25% | | 5% | | | 70% | |
| HEAD | 15% | 30% | 35% | | 5% | 5% | 10% |
| HEARD | | | 60% | | 5% | 5% | 30% |
| HEED | | | | 60% | 40% | | |
| HID | | 10% | | 15% | 45% | | 30% |
| HOD | 10% | | 20% | | | 70% | |
| WHO'D | | | 5% | | | | 95% |

The pattern of errors indicates that "hod" was most frequently confused with "had." "Head" was relatively difficult to identify, and was most commonly confused with "heard." The most common error for "heed" was "hid." "Who'd" was generally identified correctly, but other words including "hid" and "heard" were sometimes mistaken for "who'd." As expected, on the average "heed," "hod," and "who'd" were identified most accurately.

The overwhelming differences which exist between results of these two tests are simply because of the nature of these two tests. That is the word identification test is a much harder test than the ABX test. In addition, for the word identification test, one stimulus was always prerecorded on a tape recorder while the second stimulus was obtained directly from a microphone output. Although not thoroughly verified, it appeared that there were some systematic differences between the microphone and tape recorder signals which lowered the word identification accuracy.

## 5.2 Conclusions

Results of the sets of experiments conducted to determine an optimum set of basis vectors for the system indicate that cosine basis vectors are not best suited for this particular system. However, the set of PC basis vectors computed using all 16 spectral band energies were found to be optimum as demonstrated by the correlation matrices of the PC's computed using these basis vectors versus the correlation matrices of the PC's computed using two other sets of basis vectors. This final set of basis vectors appeared to be relatively speaker independent.

Results of the word identification test indicate that the color display of vowel spectra is highly effective in identifying the correct word, out of a list of seven. That is if guessing were used, the results should have converged to 1 out of 7 or 14.3%---much less than the 50% obtained form the word identification test. Presumably better word identification accuracy will be possible if the vowel-to-color transformation process is further optimized.

Results of the ABX test, which correspond to the ability of individuals to recognize vowels by just looking at a color pattern, indicate that this display is a highly accurate way of distinguishing vowels from each other. Note that vowels were identified 95% of the times.

All of the above results indicate that a linear transformation is capable of mapping vowels, as represented by the first three PC's, to colors such that widely separated vowels map into widely separated colors, and perceptually similar vowels map into perceptually similar colors.

## 5.3 Suggestions for Further Research

This research has just opened the door to what might be a revolution in the history of articulation training aids for the deaf. This work was just the beginning. If this display is regarded as an effective training aid by speech therapists for the deaf, as it has by results of our experiments, there will be much research needed to further optimize the transformation from vowels to color. For example, this transformation is expected to produce improved results if the spectral band energies are first level-normalized in order to

remove the effect of the amplitude variation on PC's. As seen from the results of the vowel clustering experiments in Chapter 3, vowels appear to be better clustered when level normalization of the band energies is used.

Some experiments with deaf speakers will also be needed to determine the effectiveness of this display for them. Word identification and ABX test are examples of such experiments. Also anticipated for further research is optimizing the system for a color display of consonants.

## LIST OF REFERENCES

Boothroyd, A. Development of small speech training aids---third progress report. Clarke School for the Deaf Report, S.A.R.P. No. 27, 1977.

Boothroyd, A. & Decker, M. Control of voice pitch by the deaf. Audiology, 1972, 11, 343-353.

Foley, J. D. and Van Dam, A., Fundamentals of Interactive Computer Graphics, Addison-Wesely Publishing Company, Inc., Philippines 1982.

Gold, B., Blankenship, P. E., & McDulay, R. J., "New Applications of Channel Vocoders," IEEE Trans. Acoust., Speech, and Signal Process., 29, 13-23, (1981).

Gordy, P. E., "Speech Spectral Modeling and Speech Synthesis Using Finite Impulse Response Digital Filters," Masters Thesis, Dept. of Elec. Eng., Old Dominion University, Norfolk, Va, 1982.

Harper, P. A visible speech aid. Volta Review, 1970, 72, 349-352.

Holbrook, A., Rolnick, M. I., & Bailey, C. W. Treatment of vocal abuse disorders using a vocal intensity controller. Journal of Speech and Hearing Disorder, 39, 298-303.

Lass, N. J., Speech and Language: Advances in Basic Research and practice, Academic Press, New York, 1982, Vol. 7, pp. 106-133.

Levitt, H., Pickett, J. M., and Houde, R. A., Sensory Aids for the Hearing Impaired, IEEE Press, Inc. New York, 1980.

Martony, J. On the correction of voice pitch level for severely hard of hearing subjects. American Annals of the Deaf, 1968, 113, 195-202.

Martony, J. Visual aids for speech correction: Summary of three years' experience. In G. Fant(Ed.), Washington, D.C.: A. G. Bell Association for the Deaf, 1970, 345-349.

Nickerson, R. S., Kalikow, D. N., & Stevens, K. N. Computer-aided speech training for the deaf. Journalech and Hearing Disorders, 1976, 61, 120-132.

Nickerson, R. S. & Stevens, K. N. Teaching speech to the deaf: Can a computer help? IEEE Transactions on Audio and Electroacoustics, 1973, AU-21, 445-455.

LIST OF REFERENCES (Continued)

Pols, L. C., Tromp, H. R. C., and Plomp, R., "Frequency Analysis of Dutch Vowels from 50 Male Speakers," J. Acoust. Soc. Amer. 53, No. 4, 1093-1101, (1973).

Potter, R. K., Kopp, G. D. & Green, C. Visible speech. Princeton, N.J.: Van Nostrand-Reinhold, 1947.

Provonost, W. Visual aids to speech improvement. Journal of Speech and Hearing Disorders, 1947, 12, 387-391.

Risberg, A. Visual aids for speech correction. American Annals of the Deaf, 1968, 113, 178-194.

Shigenaga, M. & Sekiguchi, Y. Speech training systems using the lateral shape of the vocal tract and an F1-F2 diagram of hearing-impaired children. J. Acoust. Soc. Amer., 1978, 64, S53(A).

Watanabe, A., Kisu, S., Isayama, M. & Matsuno, O. A color display system of connected speech for the deaf. J. Acoust. Soc. Amer., 1978, 64, S53(A).

Zahorian, S. A., "Principal-Components Analysis for Low Redundancy Encoding of Speech Spectra," Ph.D. Dissertation, Dept. Elect. and Comp. Eng., Syracuse University, Syracuse, NY, Technical Report TR-78-10,

# APPENDIX A

The following table is a list of various low-level subroutines that had to be developed in order to make the system interactive and flexible. Also included in this table is a list of subroutines developed to enable the user to do double precision arithmetic.

Table A-1. Various low-level procedures

| Procedure | Description |
|---|---|
| READ | Stores ASCII codes of characters transmitted from a keyboard in a buffer and returns this buffer along with the number of characters when the return key is pressed. |
| DISPLY | Displays the first N bytes of a buffer on the terminal. N is a variable passed to the procedure. |
| PRINT | Similar to the DISPLY routine except that this version has more flexibility. |
| ASC2HEX | Converts the contents of a buffer whose length can be a maximum of five and contains ASCII code of a hexadecimal number, into its corresponding hex representation. |
| HEX2ASC | Converts a two digit hex number into two ASCII codes that represent that number. |
| FORMAT | Converts the hex representation of a signed integer into ASCII codes of that signed integer in decimal format. |
| TODECIMAL | Converts the hex representation of a word variable into its decimal formated ASCII codes. |
| ADD | Double precision integer addition. |
| MULT | Double precision integer multiplication. |
| DIVIDE | Double precision integer division. |
| SQRT | Caculates the square root of any positive word variable. |

# APPENDIX B

The following pages list the main PLM86 routines used to develop and implement a vowel-to-color transformation. These programs are listed in alphabetical order. A brief description of each program and the corresponding page numbers follow.

```
BAND16:
        DO;      /*   JAN 8, 1985
-------------------------------------------------------------------
     THIS PROGRAM CAN SAMPLE THE INPUT UP TO A SAMPLING FREQUENCY
     OF 10526 Hz.  THEN IT CAN PLAYBACK THE DATA REPEATEDLY IN
     ANY SPECIFIED LENGTH (MAXIMUM LENGTH IS 25000.).  YOU CAN ADVANCE
     THROUGH THE DATA OR GO BACK IN LENGTHS OF 10., 100., OR 1000. BY
     PRESSING THE REQUIRED KEY ON THE KEYBOARD.
     YOU CAN ALSO GO INTO A MODE IN WHICH THE FOCUSED BLOCK OF DATA IS
     PLAYED BACK ONCE AND IF THE 'P' KEY WAS HIT PRIOR TO THE PLAYBACK,
     THEN AT THE END OF THE PLAYBACK ALL 16 BAND ENERGIES ARE SAMPLED AND
     THEN LEVEL NORMALIZED.  THE SPEECH PARAMETERS ARE THEN CALCULATED
     USING THE LEVEL NORMALIZED BAND ENERGIES AND DISPLAYED ON
     THE TERMINAL.  YOU CAN ALSO ADVANCE THROUGH THE DATA OR GO BACK
     AS MENTIONED ABOVE.
-------------------------------------------------------------------
            TIME= SAMPLING TIME IN MICROSECONDS.
          LENGTH= NUMBER OF SAMPLES TO BE SENT OUT.

    THIS PROGRAM ASSUMES THAT THE INPUT IS AT A/D PORT 20H AND THAT THE
    OUTPUT IS AT D/A PORT 18H.
    THIS PROGRAM ALSO OUTPUTS A SYNC. SIGNAL TO D/A PORT 16H.

        THE FOLLOWING EXPLAINS THE FUNCTION OF EACH KEY WHEN THE
        PROGRAM IS RUNNING.

             F=  ADVANCE BY 10 POINTS.
             G=  ADVANCE BY 100 POINTS.
             H=  ADVANCE BY 1000 POINTS.
             R=  GO BACK BY 10 POINTS.
             T=  GO BACK BY 100 POINTS.
             Y=  GO BACK BY 1000 POINTS.
             L=  CHANGE THE LENGTH OF THE PLAYBACK.
             Q=  START THE PROGRAM OVER AGAIN.
             M=  CHANGE BETWEEN THE TWO MODES.
             C=  TO CHANGE THE FREQUENCY OF THE PLAYBACK.
             P=  TO SAMPLE THE BAND ENERGIES, LEVEL NORMALIZE THEM, COMPUTE
                 THE PC'S USING THE LEVEL NORMALIZED BAND ENERGIES AND THE
                 CONSTANTS WHICH REPRESENT THE BASIS VECTORS, AND DISPLAY
                 THE RESULT ON THE TERMINAL.
-------------------------------------------------------------------
*/
        INITAD: PROCEDURE(A,B,C,D) EXTERNAL;
                DECLARE A WORD;
                DECLARE(B,C,D) BYTE;
        END INITAD;

        IN16AD: PROCEDURE(A,B,C,D) EXTERNAL;
                DECLARE A WORD;
                DECLARE(B,C,D) BYTE;
        END IN16AD;

        PRINT:  PROCEDURE(A,B,C,D,E,F,H) EXTERNAL;
                DECLARE A POINTER;
                DECLARE (B,C,D,E,F,H) BYTE;
        END PRINT;

        FORMAT: PROCEDURE(X,MAX,SCALE,NUMDEC,WIDTH,LOC) EXTERNAL;
                DECLARE LOC POINTER,X INTEGER,(MAX,SCALE) WORD;
                DECLARE (NUMDEC,WIDTH) BYTE;
        END FORMAT;

        GET: PROCEDURE BYTE EXTERNAL;
        END GET;
```

```
        ENABLED$INKEY: PROCEDURE BYTE EXTERNAL;
        END ENABLED$INKEY;

    DECLARE (I,NUMBER,B1,C1,D1,O1,LENGTH,START,SUM,AVERAGE,FINISH) INTEGER;
    DECLARE (J,RATE,B2,C2,D2,O2,LEN,SAMPLING$RATE) WORD;
             DECLARE (C,K,CHAR,CR,UMAX,HHB,HB,LB) BYTE;
             DECLARE DUMMY(7) BYTE;
             DECLARE P(5) INTEGER;
             DECLARE E(5)STRUCTURE(F(16) INTEGER);
             DECLARE X(16) INTEGER;
  DECLARE MESS1(34) BYTE DATA ('HIT ANY KEY TO START THE SAMPLING ');
  DECLARE MESS2(34) BYTE DATA ('HIT ANY KEY TO START THE PLAYBACK ');
  DECLARE MESS3(34) BYTE DATA ('ENTER THE TIME IN MICROSECONDS    ');
  DECLARE MESS4(34) BYTE DATA ('ENTER THE LENGTH OF THE PLAYBACK  ');
        DECLARE BUFFER1(16) INTEGER DATA(18,56,95,87,67,62,53,23,-7,-22,
                                         -45,-55,-68,-88,-96,-81);
        DECLARE BUFFER2(16) INTEGER DATA(-91,-116,-25,69,88,88,52,-29,-67,
                                         -56,-54,-18,5,33,56,67);
        DECLARE BUFFER3(16) INTEGER DATA(76,102,63,19,23,-7,-90,-127,-102,
                                         -65,-27,-3,11,36,50,38);
        DECLARE BUFFER4(16) INTEGER DATA(72,64,-43,-89,-52,28,105,71,-9,
                                         -64,-83,-77,-50,11,47,71);
        DECLARE BUFFER5(16) INTEGER DATA(-49,-16,87,65,-15,-88,-76,41,92,
                                         30,-59,-90,-71,21,77,58);
             DECLARE DATA$BUFFER(25001) BYTE;

    READ: PROCEDURE(MESS,LENGTH) EXTERNAL;
             DECLARE (MESS,LENGTH) POINTER;
    END READ;

    ASC2HEX: PROCEDURE(BUFF,LEN,HHB,HB,LB)  EXTERNAL;
             DECLARE (BUFF,HHB,HB,LB) POINTER;
             DECLARE LEN BYTE;
     END ASC2HEX;

     ADD: PROCEDURE(A,B,C,D,E,F) EXTERNAL;
            DECLARE(A,C) INTEGER;
            DECLARE(B,D) WORD,(E,F) POINTER;
     END ADD;

     DIVIDE: PROCEDURE(A,B,C,D,E) EXTERNAL;
             DECLARE (A,C) INTEGER,B WORD,(D,E) POINTER;
     END DIVIDE;

           DISABLE;
/* ------------------------------------------------------------*/
/* THIS SECTION COPIES THE BASIS VECTORS INTO A 2 DIMENSIONAL
   ARRAY E( ).F( )                                             */

             DO K=0 TO 15;
                 E(0).F(K)=BUFFER1(K);
                 E(1).F(K)=BUFFER2(K);
                 E(2).F(K)=BUFFER3(K);
                 E(3).F(K)=BUFFER4(K);
                 E(4).F(K)=BUFFER5(K);
             END;

            CALL IN16AD(20,16,0,3);
             OUTPUT(42H)=0;

  BEGIN:    I=0;
            C=0;
            J=0;
 LOOP2:     CALL PRINT(@MESS3,34,34,1,10,0,0);
            CALL READ(@DUMMY,@UMAX);
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
```

```
            RATE=256*HB+LB;

            IF C=1 THEN GO TO LOOP3;
            CALL PRINT(@MESS1,34,34,1,10,0,2);
            CHAR=GET;
LOOP3:      CALL INITAD(RATE,1,2,0);
            IF C=1 THEN GO TO LOOP;

        DO I= 0 TO 25000;
OVER:       OUTPUT(2)=00001100B;   /*  POLL MODE   */
             IF INPUT(2)<128 THEN GO TO OVER;
            DATA$BUFFER(I)=INPUT(20H);
            OUTPUT(2)=01100000B;  /* SPECIFIED  EOI  0  */
        END;

                START=0;
   LOOP:    CALL PRINT(@MESS4,34,34,1,10,0,0);
            CALL READ(@DUMMY,@UMAX);
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
            LEN=256*DOUBLE(HB)+LB;
            LENGTH=INT(LEN);
            CALL PRINT(@MESS2,34,34,1,10,0,2);
            CHAR=GET;
            CALL INITAD(RATE,1,2,0);
            FINISH=LENGTH+START;

   OVER2:          DO I= START TO FINISH;
     AGAIN: OUTPUT(2)=00001100B; /* POLL MODE  */
            IF INPUT(2)<128 THEN GO TO AGAIN;
             OUTPUT(18H)=DATA$BUFFER(I);
             OUTPUT(2)=01100000B;  /*  SPECIFIED   EOI 0  */
                     END;

            CHAR=ENABLED$INKEY;

    DECIDE: IF CHAR<41H THEN GO TO NEXT$FRAME;

                 IF CHAR='F' THEN START=START+10;
                 IF CHAR='G' THEN START=START+100;
                 IF CHAR='H' THEN START=START+1000;
                 IF CHAR='R' THEN START=START-10;
                 IF CHAR='T' THEN START=START-100;
                 IF CHAR='Y' THEN START=START-1000;
                 IF CHAR='L' THEN GO TO LOOP;
                 IF CHAR='M' THEN GO TO LOOP5;
                 IF CHAR='Q' THEN GO TO BEGIN;
                 IF CHAR='C' THEN DO;
                             C=1;
                             GO TO LOOP2;
                                END;
                 IF START<0 THEN START=0;
                 IF START>(25000-LENGTH) THEN START=25000-LENGTH;

                 CALL FORMAT(START,1,1,0,7,@DUMMY);
                 CALL PRINT(@DUMMY,7,7,1,5,0,1);

    NEXT$FRAME:
             FINISH=START+LENGTH;
             OUTPUT(16H)=255;
             CALL TIME(1);
             OUTPUT(16H)=0;
             GO TO OVER2;

    LOOP5:   DO I=START TO FINISH;
    WAIT:        OUTPUT(2)=00001100B;   /*  POLL MODE  */
                 IF INPUT(2)<128 THEN GO TO WAIT;
```

```
                OUTPUT(18H)=DATA$BUFFER(I);
                OUTPUT(2)=01100000B;  /*   SPECIFIED EOI 0  */
            END;

              OUTPUT(2)=13H;
              OUTPUT(3)=24;
              OUTPUT(3)=15;
              OUTPUT(3)=0F7H;  /*    ENABLE INTERRUPT 3  */

                OUTPUT(42H)=255;
               OUTPUT(42H)=255;    /*  START OF THE CONVERSION  */
               OUTPUT(42H)=255;
               OUTPUT(42H)=0;
   WAIT2:      OUTPUT(2)=00001100B;  /*  POLL MODE  */
            IF INPUT(2)<128 THEN GO TO WAIT2;
              SUM=0;
            DO K=0 TO 15;
                X(K)=INT(INPUT(30H+K));
                SUM=SUM+X(K);
            END;
            OUTPUT(2)=01100011B;  /* SPECIFIED EOI 3  */

               CALL INITAD(RATE,1,2,0);

            CHAR=ENABLED$INKEY;

 DECIDE2:  IF CHAR<41H THEN GO TO NEXT$FRAME2;
           IF CHAR='F' THEN START=START+10;
           IF CHAR='G' THEN START=START+100;
           IF CHAR='H' THEN START=START+1000;
           IF CHAR='R' THEN START=START-10;
           IF CHAR='T' THEN START=START-100;
           IF CHAR='Y' THEN START=START-1000;
           IF CHAR='M' THEN GO TO OVER2;
           IF CHAR='Q' THEN GO TO BEGIN;
           IF START<0 THEN START=0;
           IF START>(25000-LENGTH) THEN START=25000-LENGTH;
           CALL FORMAT(START,1,1,0,7,@DUMMY);
           CALL PRINT(@DUMMY,7,7,1,5,0,1);
           IF CHAR='P' THEN GO TO PRINT$OUT;

NEXT$FRAME2:
            FINISH=START+LENGTH;
              OUTPUT(16H)=255;
              CALL TIME(1);
              OUTPUT(16H)=0;
              GO TO LOOP5;

 PRINT$OUT:
             AVERAGE=SUM/16;
                 CR=0;
              DO K=0 TO 15;
                  CALL FORMAT(X(K),1,1,0,5,@DUMMY);
                   IF K=15 THEN CR=2;
                  CALL PRINT(@DUMMY,5,5,1,2,0,CR);
                 X(K)=X(K)-AVERAGE;
              END;
                     CR=0;

              DO K=0 TO 15;
                 IF K=15 THEN CR=3;
                 CALL FORMAT(X(K),1,1,0,5,@DUMMY);
                 CALL PRINT(@DUMMY,5,5,1,2,0,CR);
              END;
```

```
      DO K= 0 TO 4;
       B1=0;
       B2=0;
          DO J=0 TO 15;
               NUMBER=E(K).F(J)*X(J);
               IF NUMBER<0 THEN C1=-1;
                     ELSE C1=0;
               C2=UNSIGN(NUMBER);
               CALL ADD(B1,B2,C1,C2,@D1,@D2);
               B1=D1;
               B2=D2;
           END;
        CALL DIVIDE(D1,D2,256,@Q1,@Q2);
        P(K)=SIGNED(Q2);
      END;

     DO J=0 TO 3;
          CALL FORMAT(P(J),1,1,0,5,@DUMMY);
          CALL PRINT(@DUMMY,5,5,1,3,0,0);
     END;

       CALL FORMAT(P(4),1,1,0,5,@DUMMY);
       CALL PRINT(@DUMMY,5,5,1,3,0,2);
       GO TO NEXT$FRAME2;
END BAND16;
```

```
HAMID:   DO;
               /*      MAR 12, 1985       */

/*     THIS PROGRAM IS SIMILAR TO THE 'MINA' PROGRAM EXCEPT THAT IT USES THE
       GDC AND WRITES INFORMATION INTO ITS MEMORY.  EACH DATA IS DISPLAYED AS
       A VERTICAL BAR WITH A HEIGHT PROPORTIONAL TO THE ENERGY OF THE SIGNAL
       AT THE SAMPLING INSTANT.  THE DISPLAY HAS TWO INDIVIDUALLY CONTROLABLE
       DISPLAY AREAS.  THE GDC GENERATES THE REQUIRED HORIZONTAL AND VERTICAL
       SYNCS.

            INITG1:PROCEDURE EXTERNAL;
            END INITG1;

            COLMP2:PROCEDURE  EXTERNAL;
            END COLMP2;

            CHANG:PROCEDURE(R,G,B,LOC1) EXTERNAL;
                  DECLARE (R,G,B) BYTE;
                  DECLARE LOC1 POINTER;
            END CHANG;

            RECT5:PROCEDURE(X,Y,HEIGHT,WIDTH,COLOR1) EXTERNAL;
                     DECLARE (X,Y,HEIGHT,WIDTH) WORD;
                     DECLARE COLOR1 POINTER;
            END RECT5;

            SMART:PROCEDURE(X,Y,HEIGHT,WIDTH,COLOR1) EXTERNAL;
                     DECLARE (X,Y,HEIGHT,WIDTH) WORD;
                     DECLARE COLOR1 POINTER;
             END SMART;

      /*     START OF THE OLD PROGRAM       */

    PRINT: PROCEDURE(MESS,MESS$LEN,GRP$LEN,NGRP,LSPC,SPC,CR) EXTERNAL;
              DECLARE MESS POINTER;
              DECLARE (MESS$LEN,GRP$LEN,NGRP,LSPC,SPC,CR) BYTE;
    END PRINT;

    ENABLED$INKEY: PROCEDURE BYTE EXTERNAL;
    END ENABLED$INKEY;

    ASC2HEX: PROCEDURE(LOC,N,HHB,HB,LB) EXTERNAL;
               DECLARE N BYTE,(LOC,HHB,HB,LB) POINTER;
    END ASC2HEX;

    INITAD: PROCEDURE(CNT,NUM,START,INTR) EXTERNAL;
              DECLARE CNT WORD,(NUM,START,INTR) BYTE;
    END INITAD;

    READ:  PROCEDURE(MESSAGE,LENGTH) EXTERNAL;
            DECLARE (MESSAGE,LENGTH) POINTER;
    END READ;

    FORMAT: PROCEDURE(X,MAX,SCALE,NUMDEC,WIDTH,LOC) EXTERNAL;
             DECLARE LOC POINTER;
             DECLARE X INTEGER;
             DECLARE (MAX,SCALE) WORD;
             DECLARE (NUMDEC,WIDTH) BYTE;
    END FORMAT;

DECLARE R(4) BYTE;
DECLARE CHANNEL(6) BYTE;
DECLARE BUFFER(13) BYTE DATA ('ENTER VOWEL #');
DECLARE MESS0(18) BYTE DATA ('ENERGY THRESHOLD? ');
DECLARE MESS1(28) BYTE DATA ('HIT THE "ENTER" KEY AND THEN');
DECLARE MESS2(28) BYTE DATA ('TOTAL # OF DESIRED SAMPLES? ');
DECLARE MESS3(29) BYTE DATA ('# OF SAMPLES TO BE AVERAGED? ');
DECLARE MESS4(23) BYTE DATA ('THE STARTING SAMPLE #? ');
DECLARE MESS5(43) BYTE DATA ('ENTER THE TRANSFORMATION MATRIX ROW BY ROW:');
DECLARE MESS6(34) BYTE DATA ('WHICH CHANNELS ARE TO BE SAMPLED? ');
DECLARE MESS7(9) BYTE DATA ('CHANNEL #');
```

```
DECLARE MESS8(29) BYTE DATA ('THE TRANSFORMATION MATRIX IS:');
DECLARE MESS9(23) BYTE DATA ('SAME FOR ALL  VOWELS?  ');
DECLARE MESSA(26) BYTE DATA ('WHAT IS THE SCALE FACTOR? ');
DECLARE MESSB(47) BYTE DATA ('TURN ON THE SPEECH PARAMETER EXTRACTOR AND THEN');
DECLARE MESSC(27) BYTE DATA ('THE PARAMETER ORIGIN IS AT:');
DECLARE MESSD(19) BYTE DATA ('NUMBER OF VOWELS?  ');
DECLARE MESSE(39) BYTE DATA ('WOULD YOU LIKE TO START FROM BEGINNING,');
DECLARE MESSI(30) BYTE DATA ('ENTER A NEW TRANSFORMATION, OR');
DECLARE MESSJ(43) BYTE DATA ('USE THE PREDEFINED TRANSFORMATION (B,T,P)? ');
DECLARE MESSF(30) BYTE DATA ('LENGTH OF THE OUTPUT BUFFER?  ');
DECLARE REFRED(14) BYTE DATA ('REFRENCE RED? ');
DECLARE REFBLUE(15) BYTE DATA ('REFRENCE BLUE? ');
DECLARE REFGREEN(16) BYTE DATA ('REFRENCE GREEN? ');
DECLARE TARRED(3) BYTE DATA ('R? ');
DECLARE TARGRN(3) BYTE DATA ('G? ');
DECLARE TARBLU(3) BYTE DATA ('B? ');
DECLARE MESSG(29) BYTE DATA ('LEVEL OF R THEN G AND THEN B:');
DECLARE MESSH(32) BYTE DATA ('ENTER N FOR R TO EXIT THIS MODE.');
DECLARE MESSL(36) BYTE DATA ('ENTER SAMPLING TIME IN MILISECONDS  ');
DECLARE MESSM(31) BYTE DATA ('HIT THE "F" KEY TO CONTINUE OR,');
DECLARE MESSN(48) BYTE DATA ('HIT THE "RETURN" KEY TO CHANGE THE DISPLAY AREA.');
DECLARE INVERSE(30) BYTE;
DECLARE DUMMY(5) BYTE;
DECLARE (UMAX,HHB,HB,LB,KEY,I,M,CR,J,K,PORT,THRESH) BYTE;
DECLARE (PTR,BEGIN,T,N,SGN,NEW,FINISH) BYTE;
DECLARE CH(5) STRUCTURE(SAMPLE(256) INTEGER);
DECLARE ROW(4) STRUCTURE(COLUMN(6) INTEGER);
DECLARE SOUND(6) STRUCTURE(PAR(6) INTEGER);
DECLARE C(5) STRUCTURE(S(256) BYTE);
DECLARE FLAG(3) BYTE;
DECLARE (SUM,AVERAGE,REF) (8) INTEGER;
DECLARE RESULT(7) BYTE;
DECLARE FACTOR INTEGER;
DECLARE (TOTAL$SAMPLES,NEEDED$SAMPLES,START,NP,OUTPUT$LENGTH) BYTE;
DECLARE (DATA1,DATA2,DATA3) (256) BYTE;
DECLARE P(6) INTEGER;
DECLARE COLOR(4) BYTE;
DECLARE (FLAG2,STATUS,COMMAND,EMAX,ENERGY,EMIN,EMAX0,EMAX1,EMIN1,EMIN0) BYTE;
DECLARE (RR,GG,BB) BYTE;
DECLARE (TIME,COUNTER,X,M1,START$ADDR,XCOMP) WORD;
DECLARE (ADD$COMP,CURRENT$X,CURRENT$START) WORD;
DECLARE (HEIGHT,Y) WORD;
DECLARE ERASE(5) BYTE DATA (0,0,0,0,0);
DECLARE (Y$ORIGIN,Y$ERASE) WORD;
DECLARE SRC(200H) BYTE AT (0B000H);
DECLARE DES(200H) BYTE AT (0H);


    COLOR$REFRENCE:  PROCEDURE;
            CALL PRINT(@REFRED,14,14,1,15,0,0);
            CALL READ(@DUMMY,@UMAX);
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
            R(1)=LB;    /*  R(1)=RED COLOR    */
            CALL PRINT(@REFGREEN,16,16,1,15,0,0);
            CALL READ(@DUMMY,@UMAX);
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
            R(2)=LB;    /*  R(2)=GREEN COLOR    */
            CALL PRINT(@REFBLUE,15,15,1,15,0,0);
            CALL READ(@DUMMY,@UMAX);
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
            R(3)=LB;    /*  R(3)=BLUE COLOR     */
     END COLOR$REFRENCE;

    PREDEFINED$TRANSFORMATION:      PROCEDURE;

            ROW(1).COLUMN(1)=31;
```

```
            ROW(1).COLUMN(2)=14;
            ROW(1).COLUMN(3)=41;

            ROW(2).COLUMN(1)=-27;
            ROW(2).COLUMN(2)=37;
            ROW(2).COLUMN(3)=-9;

            ROW(3).COLUMN(1)=43;
            ROW(3).COLUMN(2)=-51;
            ROW(3).COLUMN(3)=-58;

            R(1)=34;
            R(2)=34;
            R(3)=34;

            CHANNEL(1)=20H;
            CHANNEL(2)=21H;
            CHANNEL(3)=22H;
            CHANNEL(4)=23H;
            CHANNEL(5)=24H;

            FACTOR=20;
            NP=3;

     END PREDEFINED$TRANSFORMATION;

    TIMING:  PROCEDURE;
               CALL PRINT (@MESSL,36,36,1,10,0,0);
               CALL READ(@DUMMY,@UMAX);
               CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
               TIME=1000*LB;
    END TIMING;

    WAIT:       PROCEDURE;
                  CYCLE:STATUS=INPUT(80H) AND 4;
                  IF STATUS=0 THEN GO TO CYCLE;
     END WAIT;


     INT24:
             PROCEDURE INTERRUPT 24;
                   PTR=PTR+1;
              IF INPUT(26H)>THRESH THEN I=I+1;
                   DO J=32 TO 36;
                     C(J-32).S(PTR)=INPUT(J);
                   END;
     END INT24;

INT26:   PROCEDURE INTERRUPT 26;

ENERGY = INPUT(26H);
HEIGHT= SHR((60*ENERGY),7) +1;/* THIS ASSUMES MAXIMUM 8-BIT RANGE FOR ENERGY */
   Y=Y$ORIGIN-HEIGHT/2;

               DO I=1 TO NP;
                   P(I)=INT(INPUT(CHANNEL(I)))-REF(CHANNEL(I)-20H);
               END;

                  DO J=1 TO 3;
                      SUM(J)=0;
                         DO I=1 TO NP;
                              SUM(J)=SUM(J)+ROW(J).COLUMN(I)*P(I);
                         END;
                        SUM(J)=SUM(J)/FACTOR+INT(R(J));
                        IF SUM(J)>287 THEN SUM(J)=287;
                        IF SUM(J)<0 THEN SUM(J)=0;
```

```
                END;
                    RR=LOW(UNSIGN(SAR(SUM(1),5)));
                    GG=LOW(UNSIGN(SAR(SUM(2),5)));
                    BB=LOW(UNSIGN(SAR(SUM(3),5)));
              CALL CHANG(RR,GG,BB,@COLOR);
              CALL SMART(X,Y,HEIGHT,8,@COLOR);
    END INT26;

          DISABLE;
              CALL MOVB(@SRC,@DES,200H);

              CALL COLMP2;
              CALL WAIT;
              CALL INITG1;

              COMMAND=70H;

AGAIN:    DISABLE;
              FLAG2=0;
            CALL RECT5(0,0,256,1024,@ERASE);
              PTR=0;
            CALL PRINT(@MESSB,47,47,1,10,0,1);
            CALL PRINT(@MESS1,19,19,1,10,0,2);
            CALL INITAD(5000,8,2,0);
            CALL READ(@DUMMY,@UMAX);
          ENABLE;
LOOP5:      IF PTR<20 THEN GO TO LOOP5;
          DISABLE;

          DO J=0 TO 4;
                 REF(J)=0;
               DO I=1 TO 20;
                  REF(J)=INT(C(J).S(I))+REF(J);
               END;
            REF(J)=REF(J)/20;
          END;

          CALL PRINT(@MESSC,27,27,1,13,0,2);
          CR=0;
          DO I=0 TO 4;
               IF I=4 THEN CR=2;
             CALL FORMAT(REF(I),1,10,1,6,@RESULT);
             CALL PRINT(@RESULT,6,6,1,3,0,CR);
          END;
             OUTPUT$LENGTH=150;
WRONG:       CALL PRINT(@MESSE,39,39,1,10,0,1);
             CALL PRINT(@MESSI,30,30,1,10,0,1);
             CALL PRINT(@MESSJ,43,43,1,10,0,0);

             CALL READ(@DUMMY,@UMAX);
             IF DUMMY(0)='T' THEN GO TO GET$NP;
             IF DUMMY(0)='P' THEN DO;
                                  CALL PREDEFINED$TRANSFORMATION;
                                  GO TO CONTINUE;
                                     END;
             IF DUMMY(0)<>'B' THEN GO TO WRONG;
LOOP8:  /*  THIS SECTION IS FOR DISPLAYING TARGET COLORS  */
            CALL PRINT(@MESSG,29,29,1,10,0,1);
            CALL PRINT(@MESSH,32,32,1,10,0,2);
            CALL PRINT(@TARRED,3,3,1,10,0,0);
            CALL READ(@DUMMY,@UMAX);
            IF DUMMY(0)='N' THEN GO TO MODE2;
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
            RR=LB/32;
            CALL PRINT(@TARGRN,3,3,1,10,0,0);
            CALL READ(@DUMMY,@UMAX);
```

```
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
            GG=LB/32;
            CALL PRINT(@TARBLU,3,3,1,10,0,0);
            CALL READ(@DUMMY,@UMAX);
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
            BB=LB/32;
            CALL CHANG(RR,GG,BB,@COLOR);
            CALL RECT5(0,0,256,1024,@COLOR);
            GO TO LOOP8;

MODE2:      CALL RECT5(0,0,256,1024,@ERASE);
            CALL PRINT(@MESS0,19,19,1,5,1,0);
            CALL READ(@DUMMY,@UMAX);
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@NP);

                CALL PRINT(@MESS0,18,18,1,15,0,0);
                CALL READ(@DUMMY,@UMAX);
                CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@THRESH);
                FLAG(0)=0;
                K=1;
NEXT$SOUND:     I=0;
                M=K+30H;
               CALL INITAD(5000,8,2,0);
               IF FLAG(0)='Y' THEN GO TO SAME;
               CALL PRINT(@MESS2,28,28,1,15,0,0);
               CALL READ(@DUMMY,@UMAX);
               CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@TOTAL$SAMPLES);
               CALL PRINT(@MESS3,29,29,1,15,0,0);
               CALL READ(@DUMMY,@UMAX);
               CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@NEEDED$SAMPLES);
               CALL PRINT(@MESS4,22,22,1,15,0,0);
               CALL READ(@DUMMY,@UMAX);
               CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@START);
               CALL PRINT(@MESS9,23,23,1,15,0,0);
               CALL READ(@FLAG,@UMAX);
SAME:          CALL PRINT(@MESS1,28,28,1,10,0,1);
               CALL PRINT(@BUFFER,13,13,1,15,0,0);
               CALL PRINT(@M,1,1,1,1,0,2);
               CALL READ(@DUMMY,@UMAX);
        ENABLE;
  LOOP:
            IF I>=TOTAL$SAMPLES THEN GO TO SAVE;
            GO TO LOOP;
 SAVE:
           DISABLE;

      BEGIN=PTR-TOTAL$SAMPLES+START-1;
      FINISH=BEGIN+NEEDED$SAMPLES-1;

             DO J=0 TO 4;
               DO I=0 TO 255;
                   CH(J).SAMPLE(I)=INT(C(J).S(I));
               END;
             END;

             DO J=0 TO 4;
                  SUM(J)=0;
                I=BEGIN;
      LOOP2:    SUM(J)=SUM(J)+CH(J).SAMPLE(I);
                I=I+1;
                IF I=FINISH+1 THEN GO TO NEXT$SET;
                GO TO LOOP2;
NEXT$SET:     AVERAGE(J)=(SUM(J)/INT(NEEDED$SAMPLES))-REF(J);
             END;

             DO J=1 TO 5;
```

```
                    SOUND(K).PAR(J)=AVERAGE(J-1);
                END;

                K=K+1;
            IF K<NP+1 THEN GO TO NEXT$SOUND;

                DO J=1 TO 5;
                     CR=0;
                    DO K=1 TO NP;
                          IF K=NP THEN CR=1;
                          IF J=5 AND K=NP THEN CR=3;
                        CALL FORMAT(SOUND(K).PAR(J),1,10,1,6,@RESULT);
                        CALL PRINT(@RESULT,6,6,1,4,0,CR);
                    END;
                END;

TRANS:          CALL PRINT(@MESS5,43,43,1,15,0,2);

                DO J=1 TO 3;
                    N=0;
                       CALL READ(@INVERSE,@UMAX);
                     DO I=1 TO NP;
                         T=0;
                          IF INVERSE(N)='-' THEN DO;
                                SGN=1;
                                N=N+1;
                                GO TO LOOP;
                                                    END;
                    ELSE IF INVERSE(N)='+' THEN DO;
                                SGN=0;
                                N=N+1;
                                GO TO LOOP;
                                                END;
                    ELSE SGN=0;
LOOP:
                   IF INVERSE(N)=',' OR N=UMAX THEN GO TO NEXT;
                          DUMMY(T)=INVERSE(N);
                           T=T+1;
                           N=N+1;
                       GO TO LOOP;
NEXT:
                    CALL ASC2HEX(@DUMMY,T,@HHB,@HB,@LB);
                    ROW(J).COLUMN(I)=INT(HB)*256+INT(LB);
                        IF SGN=1 THEN ROW(J).COLUMN(I)=-ROW(J).COLUMN(I);
                      N=N+1;
                    END;
                END;
                CALL PRINT(@MESS8,29,29,1,15,0,2);
                DO K=1 TO 3;
                     CR=0;
                      DO J=1 TO NP;
                         IF J=NP THEN CR=1;
                         IF K=3 AND J=NP THEN CR=2;
                         CALL FORMAT(ROW(K).COLUMN(J),1,10,1,6,@RESULT);
                         CALL PRINT(@RESULT,6,6,1,4,0,CR);
                      END;
                END;

            CALL PRINT(@MESSA,26,26,1,15,0,0);
            CALL READ(@DUMMY,@UMAX);
            CALL PRINT(@DUMMY,UMAX,UMAX,1,1,0,2);
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
            FACTOR=INT(LB);
            CALL PRINT(@MESS6,34,34,1,10,0,2);
                 DO I=1 TO NP;
                     CALL PRINT(@MESS7,9,9,1,10,0,0);
```

```
                    CALL READ(@DUMMY,@UMAX);
                    CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
                    CHANNEL(I)=LB+20H;
                END;
REFRENCE:   DISABLE;
            CALL COLOR$REFRENCE;
CONTINUE:   X,START$ADDR=0;     CURRENT$START=0;
            Y$ORIGIN=60;    Y$ERASE=0;   ADD$COMP=512;

            CALL TIMING;

 C5:        COUNTER=1;

   THERE:   CALL INITAD(TIME,8,2,2);
   RUN:     ENABLE;

   OVER:  HALT;
              IF COUNTER=2 THEN DO;
                              COUNTER=1;
                              X=X+8;
                              GO TO DISPLAY$IT;
                                END;
               COUNTER=2;
                X=X+8;
               GO TO OVER;
DISPLAY$IT:
        IF (X<1024) THEN GO TO CHECK;

                /*MOVE THE COLOR BAR */

            XCOMP=X-1024;
           CALL SMART(XCOMP,Y$ERASE,120,16,@ERASE);  /* ERASE OLDEST BAR */
           START$ADDR=START$ADDR+1;
           CALL WAIT;

         OUTPUT(81H)=COMMAND;   /* UPDATE STARTING ADDRESS FOR DISPLAY AREA */
         OUTPUT(80H)=LOW(START$ADDR);  /* ONE USING PRAM GDC COMMAND */
         OUTPUT(80H)=HIGH(START$ADDR);

        IF (START$ADDR <ADD$COMP) THEN GO TO CHECK;

        DISABLE;
           X= 0;   /* REINITIALIZE X, START$ADDR */
           START$ADDR=CURRENT$START;

          /* RESET POSITION TO LEFT SIDE OF SCREEN*/

     OUTPUT(81H)=COMMAND;  /*  INITIALIZE STARTING ADDRESS FOR CURRENT AREA */
     OUTPUT(80H)=LOW(START$ADDR);
     OUTPUT(80H)=HIGH(START$ADDR);
     CALL RECT5(0,Y$ERASE,128,1024,@ERASE);  /* CLEAR CURRENT AREA */
          GO TO RUN;
CHECK:     KEY=ENABLED$INKEY;
             IF KEY='F' THEN GO TO FREEZE;
             IF KEY='Q' THEN GO TO AGAIN;
             IF KEY='C' THEN GO TO CHANGE;
             IF KEY='T' THEN GO TO GET$NP;
             GO TO OVER;

     GET$NP:  DISABLE;
              CALL PRINT(@MESSD,19,19,1,10,1,0);
              CALL READ(@DUMMY,@UMAX);
              CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@NP);
              GO TO TRANS;

     CHANGE:  DISABLE;
```

```
            CALL TIMING;
            GO TO THERE;

 FREEZE:  DISABLE;
           CALL PRINT(@MESSM,31,31,1,10,0,1);
           CALL PRINT(@MESSN,48,48,1,10,0,2);
STAY$HERE:    KEY=ENABLED$INKEY;
              IF KEY='F' THEN GO TO RUN;
              IF KEY<>0DH THEN GO TO STAY$HERE;

    IF FLAG2=1 THEN DO;
                       COMMAND=70H;
                       Y$ORIGIN=60;   Y$ERASE=0;
                       ADD$COMP=512;
                    X,START$ADDR,CURRENT$START=0;
                       FLAG2=0;
                        END;
                ELSE DO;
                        X=0;
                        COMMAND=74H;
                       Y$ORIGIN=188;  Y$ERASE=128;
                       ADD$COMP=8704;
                       START$ADDR,CURRENT$START=8192;
                       FLAG2=1;
                      END;
          OUTPUT(81H)=COMMAND;
          OUTPUT(80H)=LOW(START$ADDR);
          OUTPUT(80H)=HIGH(START$ADDR);
          CALL RECT5(0,Y$ERASE,128,1024,@ERASE);
           GO TO C5;

END HAMID;
```

```
      MINA:
/*     THIS PROGRAM IS WRITTEN TO IMPLEMENT THE VOWEL-TO-COLOR CONVERSION.
       ANY SPECIFIED COLOR CAN BE SEEN BEFORE IT IS ASSIGNED AS ONE OF THE
       TARGET COLORS.  THEN THE SPEECH PARAMETERS CAN BE SAMPLED FOR A
       SPECIFIED NUMBER OF VOWELS (ONE VOWEL AT A TIME).  THE TRANSFORMATION
       MUST BE FOUND USING FLOATING POINT ARITHMETIC BY USING THE SAMPLED
       VALUES.  THE TRANSFORMATION IS THEN ENTERED INTO THE PROGRAM ALONG
       WITH SOME OTHER VARIABLES AND THE PROGRAM CONTINUES ITS OPERATION IN
       REAL-TIME AND CONVERTS THE SPEECH PARAMETERS INTO COLORS.  THE SYSTEM
       GENERATES THE VERTICAL SYNC AS WELL AS PROCESSING OF DATA.  THREE D-A
       CHANNELS ARE USED TO OUTPUT THE R,G, AND B VALUES.  THE INTERRUPT PULSE
       IS USED TO DRIVE THE VERTICAL SYNC CIRCUITRY OF THE TV MONITOR.   */

        DO;
    PRINT: PROCEDURE(MESS,MESS$LEN,GRP$LEN,NGRP,LSPC,SPC,CR) EXTERNAL;
              DECLARE MESS POINTER;
              DECLARE (MESS$LEN,GRP$LEN,NGRP,LSPC,SPC,CR) BYTE;
    END PRINT;
        ENABLED$INKEY: PROCEDURE BYTE EXTERNAL;
        END ENABLED$INKEY;
    ASC2HEX: PROCEDURE(LOC,N,HHB,HB,LB) EXTERNAL;
               DECLARE N BYTE,(LOC,HHB,HB,LB) POINTER;
    END ASC2HEX;
          INITAD: PROCEDURE(CNT,NUM,START,INTR) EXTERNAL;
                     DECLARE CNT WORD,(NUM,START,INTR) BYTE;
          END INITAD;
    READ:  PROCEDURE(MESSAGE,LENGTH) EXTERNAL;
             DECLARE (MESSAGE,LENGTH) POINTER;
    END READ;
          FORMAT: PROCEDURE(X,MAX,SCALE,NUMDEC,WIDTH,LOC) EXTERNAL;
                    DECLARE LOC POINTER;
                    DECLARE X INTEGER;
                    DECLARE (MAX,SCALE) WORD;
                    DECLARE (NUMDEC,WIDTH) BYTE;
          END FORMAT;

DECLARE R(4) BYTE;
DECLARE CHANNEL(6) BYTE;
DECLARE BUFFER(13) BYTE DATA ('ENTER VOWEL #');
DECLARE MESS0(18) BYTE DATA ('ENERGY THRESHOLD? ');
DECLARE MESS1(28) BYTE DATA ('HIT THE "ENTER" KEY AND THEN');
DECLARE MESS2(28) BYTE DATA ('TOTAL # OF DESIRED SAMPLES? ');
DECLARE MESS3(29) BYTE DATA ('# OF SAMPLES TO BE AVERAGED? ');
DECLARE MESS4(23) BYTE DATA ('THE STARTING SAMPLE #? ');
DECLARE MESS5(43) BYTE DATA ('ENTER THE TRANSFORMATION MATRIX ROW BY ROW:');
DECLARE MESS6(34) BYTE DATA ('WHICH CHANNELS ARE TO BE SAMPLED? ');
DECLARE MESS7(9) BYTE DATA ('CHANNEL #');
DECLARE MESS8(29) BYTE DATA ('THE TRANSFORMATION MATRIX IS:');
DECLARE MESS9(23) BYTE DATA ('SAME FOR ALL  VOWELS?  ');
DECLARE MESSA(26) BYTE DATA ('WHAT IS THE SCALE FACTOR? ');
DECLARE MESSB(47) BYTE DATA ('TURN ON THE SPEECH PARAMETER EXTRACTOR AND THEN');
DECLARE MESSC(27) BYTE DATA ('THE PARAMETER ORIGIN IS AT:');
DECLARE MESSD(19) BYTE DATA ('NUMBER OF VOWELS?  ');
DECLARE MESSE(39) BYTE DATA ('WOULD YOU LIKE TO START FROM BEGINNING,');
DECLARE MESSI(30) BYTE DATA ('ENTER A NEW TRANSFORMATION, OR');
DECLARE MESSJ(43) BYTE DATA ('USE THE PREDEFINED TRANSFORMATION (B,T,P)? ');
DECLARE MESSF(30) BYTE DATA ('LENGTH OF THE OUTPUT BUFFER?  ');
DECLARE REFRED(14) BYTE DATA ('REFRENCE RED? ');
DECLARE REFBLUE(15) BYTE DATA ('REFRENCE BLUE? ');
DECLARE REFGREEN(16) BYTE DATA ('REFRENCE GREEN? ');
DECLARE TARRED(3) BYTE DATA ('R? ');
DECLARE TARGRN(3) BYTE DATA ('G? ');
DECLARE TARBLU(3) BYTE DATA ('B? ');
DECLARE MESSG(29) BYTE DATA ('LEVEL OF R THEN G AND THEN B:');
DECLARE MESSH(32) BYTE DATA ('ENTER N FOR R TO EXIT THIS MODE.');
DECLARE INVERSE(30) BYTE;
DECLARE DUMMY(5) BYTE;
DECLARE (UMAX,HHB,HB,LB,KEY,I,M,CR,J,K,PORT,THRESH) BYTE;
DECLARE (PTR,BEGIN,T,N,SGN,NEW,FINISH) BYTE;
DECLARE CH(5) STRUCTURE(SAMPLE(256) INTEGER);
DECLARE ROW(4) STRUCTURE(COLUMN(6) INTEGER);
DECLARE SOUND(6) STRUCTURE(PAR(6) INTEGER);
```

```
DECLARE C(3) STRUCTURE(S(256) BYTE);
DECLARE FLAG(3) BYTE;
DECLARE (SUM,AVERAGE,REF) (8) INTEGER;
DECLARE RESULT(7) BYTE;
DECLARE FACTOR INTEGER;
DECLARE (TOTAL$SAMPLES,NEEDED$SAMPLES,START,NP,OUTPUT$LENGTH) BYTE;
DECLARE (DATA1,DATA2,DATA3) (256) BYTE;
DECLARE P(6) INTEGER;
DECLARE COLOR(4) BYTE;

    COLOR$REFRENCE:  PROCEDURE;
            CALL PRINT(@REFRED,14,14,1,15,0,0);
            CALL READ(@DUMMY,@UMAX);
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
            R(1)=LB;    /*  R(1)=RED COLOR    */
            CALL PRINT(@REFGREEN,16,16,1,15,0,0);
            CALL READ(@DUMMY,@UMAX);
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
            R(2)=LB;    /*  R(2)=GREEN COLOR    */
            CALL PRINT(@REFBLUE,15,15,1,15,0,0);
            CALL READ(@DUMMY,@UMAX);
            CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
            R(3)=LB;    /*  R(3)=BLUE COLOR     */
     END COLOR$REFRENCE;

    PREDEFINED$TRANSFORMATION:      PROCEDURE;

            ROW(1).COLUMN(1)=21;
            ROW(1).COLUMN(2)=2;
            ROW(1).COLUMN(3)=10;
            ROW(1).COLUMN(4)=0;
            ROW(1).COLUMN(5)=0;

            ROW(2).COLUMN(1)=-8;
            ROW(2).COLUMN(2)=25;
            ROW(2).COLUMN(3)=-8;
            ROW(2).COLUMN(4)=0;
            ROW(2).COLUMN(5)=0;

            ROW(3).COLUMN(1)=15;
            ROW(3).COLUMN(2)=-28;
            ROW(3).COLUMN(3)=-25;
            ROW(3).COLUMN(4)=0;
            ROW(3).COLUMN(5)=0;

            R(1)=130;
            R(2)=130;
            R(3)=130;

            CHANNEL(1)=20H;
            CHANNEL(2)=21H;
            CHANNEL(3)=22H;
            CHANNEL(4)=23H;
            CHANNEL(5)=24H;

            FACTOR=20;
            NP=3;

    END PREDEFINED$TRANSFORMATION;

    INT24:
            PROCEDURE INTERRUPT 24;
                  PTR=PTR+1;
             IF INPUT(26H)>THRESH THEN I=I+1;
                 DO J=32 TO 36;
```

```
                C(J-32).S(PTR)=INPUT(J);
              END;
    END INT24;

  INT26:  PROCEDURE INTERRUPT 26;

           DO I=0 TO OUTPUT$LENGTH;
               J=NEW+I;
               OUTPUT(15H)=DATA1(J);
               OUTPUT(16H)=DATA2(J);
               OUTPUT(17H)=DATA3(J);
               OUTPUT(15H)=DATA1(J);
               OUTPUT(16H)=DATA2(J);
               OUTPUT(17H)=DATA3(J);
           END;

           IF INPUT(27H)<200 THEN NEW=NEW-1;

             DO I=1 TO NP;
                P(I)=INT(INPUT(CHANNEL(I)))-REF(CHANNEL(I)-20H);
             END;

                DO J=1 TO 3;
                    SUM(J)=0;
                       DO I=1 TO NP;
                           SUM(J)=SUM(J)+ROW(J).COLUMN(I)*P(I);
                       END;
                      SUM(J)=SUM(J)/FACTOR+INT(R(J));
                      IF SUM(J)>255 THEN SUM(J)=255;
                      IF SUM(J)<120 THEN SUM(J)=120;
                    COLOR(J)=LOW(UNSIGN(SUM(J)));
                END;
                 DATA1(NEW)=COLOR(1);
                 DATA2(NEW)=COLOR(2);
                 DATA3(NEW)=COLOR(3);

     END INT26;

AGAIN:    DISABLE;
              PTR=0;
            CALL PRINT(@MESSB,47,47,1,10,0,1);
            CALL PRINT(@MESS1,19,19,1,10,0,2);
            CALL INITAD(5000,8,2,0);
            CALL READ(@DUMMY,@UMAX);
          ENABLE;
  LOOP5:    IF PTR<20 THEN GO TO LOOP5;
          DISABLE;

          DO J=0 TO 4;
                 REF(J)=0;
                DO I=1 TO 20;
                   REF(J)=INT(C(J).S(I))+REF(J);
                END;
             REF(J)=REF(J)/20;
           END;

           CALL PRINT(@MESSC,27,27,1,13,0,2);
           CR=0;
           DO I=0 TO 4;
                IF I=4 THEN CR=2;
              CALL FORMAT(REF(I),1,10,1,6,@RESULT);
              CALL PRINT(@RESULT,6,6,1,3,0,CR);
           END;
              OUTPUT$LENGTH=150;
 WRONG:       CALL PRINT(@MESSE,39,39,1,10,0,1);
              CALL PRINT(@MESSI,30,30,1,10,0,1);
```

```
                CALL PRINT(@MESSJ,43,43,1,10,0,0);

                CALL READ(@DUMMY,@UMAX);
                IF DUMMY(0)='T' THEN GO TO GET$NP;
                IF DUMMY(0)='P' THEN DO;
                                CALL PREDEFINED$TRANSFORMATION;
                                GO TO THERE;
                                    END;
                IF DUMMY(0)<>'B' THEN GO TO WRONG;
LOOP8:
               CALL PRINT(@MESSG,29,29,1,10,0,1);
               CALL PRINT(@MESSH,32,32,1,10,0,2);
               CALL PRINT(@TARRED,3,3,1,10,0,0);
               CALL READ(@DUMMY,@UMAX);
               IF DUMMY(0)='N' THEN GO TO MODE2;
               CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
               OUTPUT(15H)=LB;
               CALL PRINT(@TARGRN,3,3,1,10,0,0);
               CALL READ(@DUMMY,@UMAX);
               CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
               OUTPUT(16H)=LB;
               CALL PRINT(@TARBLU,3,3,1,10,0,0);
               CALL READ(@DUMMY,@UMAX);
               CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
               OUTPUT(17H)=LB;
               GO TO LOOP8;

   MODE2:     CALL PRINT(@MESSD,19,19,1,5,1,0);
              CALL READ(@DUMMY,@UMAX);
              CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@NP);

                  CALL PRINT(@MESS0,18,18,1,15,0,0);
                  CALL READ(@DUMMY,@UMAX);
                  CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@THRESH);
                  FLAG(0)=0;
                  K=1;
    NEXT$SOUND:   I=0;
                  M=K+30H;
                CALL INITAD(5000,8,2,0);
                IF FLAG(0)='Y' THEN GO TO SAME;
                CALL PRINT(@MESS2,28,28,1,15,0,0);
                CALL READ(@DUMMY,@UMAX);
                CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@TOTAL$SAMPLES);
                CALL PRINT(@MESS3,29,29,1,15,0,0);
                CALL READ(@DUMMY,@UMAX);
                CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@NEEDED$SAMPLES);
                CALL PRINT(@MESS4,22,22,1,15,0,0);
                CALL READ(@DUMMY,@UMAX);
                CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@START);
                CALL PRINT(@MESS9,23,23,1,15,0,0);
                CALL READ(@FLAG,@UMAX);
    SAME:       CALL PRINT(@MESS1,28,28,1,10,0,1);
                CALL PRINT(@BUFFER,13,13,1,15,0,0);
                CALL PRINT(@M,1,1,1,1,0,2);
                CALL READ(@DUMMY,@UMAX);
          ENABLE;
     LOOP:
              IF I>=TOTAL$SAMPLES THEN GO TO SAVE;
              GO TO LOOP;
     SAVE:
             DISABLE;

        BEGIN=PTR-TOTAL$SAMPLES+START-1;
        FINISH=BEGIN+NEEDED$SAMPLES-1;

               DO J=0 TO 4;
```

```
                DO I=0 TO 255;
                    CH(J).SAMPLE(I)=INT(C(J).S(I));
                END;
              END;

              DO J=0 TO 4;
                   SUM(J)=0;
                  I=BEGIN;
    LOOP2:        SUM(J)=SUM(J)+CH(J).SAMPLE(I);
                  I=I+1;
                  IF I=FINISH+1 THEN GO TO NEXT$SET;
                  GO TO LOOP2;
NEXT$SET:     AVERAGE(J)=(SUM(J)/INT(NEEDED$SAMPLES))-REF(J);
              END;

              DO J=1 TO 5;
                  SOUND(K).PAR(J)=AVERAGE(J-1);
              END;

              K=K+1;
      IF K<NP+1 THEN GO TO NEXT$SOUND;

          DO J=1 TO 5;
               CR=0;
              DO K=1 TO NP;
                    IF K=NP THEN CR=1;
                    IF J=5 AND K=NP THEN CR=3;
                  CALL FORMAT(SOUND(K).PAR(J),1,10,1,6,@RESULT);
                  CALL PRINT(@RESULT,6,6,1,4,0,CR);
              END;
          END;

TRANS:        CALL PRINT(@MESS5,43,43,1,15,0,2);

           DO J=1 TO 3;
               N=0;
                  CALL READ(@INVERSE,@UMAX);
                DO I=1 TO NP;
                    T=0;
                     IF INVERSE(N)='-' THEN DO;
                            SGN=1;
                            N=N+1;
                            GO TO LOOP;
                                              END;
                ELSE IF INVERSE(N)='+' THEN DO;
                            SGN=0;
                            N=N+1;
                            GO TO LOOP;
                                            END;
                ELSE SGN=0;
    LOOP:
              IF INVERSE(N)=',' OR N=UMAX THEN GO TO NEXT;
                     DUMMY(T)=INVERSE(N);
                      T=T+1;
                      N=N+1;
                    GO TO LOOP;
    NEXT:
                CALL ASC2HEX(@DUMMY,T,@HHB,@HB,@LB);
                ROW(J).COLUMN(I)=INT(HB)*256+INT(LB);
                    IF SGN=1 THEN ROW(J).COLUMN(I)=-ROW(J).COLUMN(I);
                  N=N+1;
                END;
            END;
            CALL PRINT(@MESS8,29,29,1,15,0,2);
            DO K=1 TO 3;
                 CR=0;
```

```
                    DO J=1 TO NP;
                       IF J=NP THEN CR=1;
                       IF K=3 AND J=NP THEN CR=2;
                       CALL FORMAT(ROW(K).COLUMN(J),1,10,1,6,@RESULT);
                       CALL PRINT(@RESULT,6,6,1,4,0,CR);
                    END;
                END;

             CALL PRINT(@MESSA,26,26,1,15,0,0);
             CALL READ(@DUMMY,@UMAX);
             CALL PRINT(@DUMMY,UMAX,UMAX,1,1,0,2);
             CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
             FACTOR=INT(LB);
             CALL PRINT(@MESS6,34,34,1,10,0,2);
                  DO I=1 TO NP;
                      CALL PRINT(@MESS7,9,9,1,10,0,0);
                      CALL READ(@DUMMY,@UMAX);
                      CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
                      CHANNEL(I)=LB+20H;
                  END;
REFRENCE:    DISABLE;
             CALL COLOR$REFRENCE;

   THERE:    CALL INITAD(16666,8,2,2);
             ENABLE;
     OVER:     KEY=ENABLED$INKEY;
             IF KEY='Q' THEN GO TO AGAIN;
             IF KEY='L' THEN GO TO CHANGE;
             IF KEY='T' THEN GO TO GET$NP;
             GO TO OVER;

    GET$NP:  DISABLE;
             CALL PRINT(@MESSD,19,19,1,10,1,0);
             CALL READ(@DUMMY,@UMAX);
             CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@NP);
             GO TO TRANS;

    CHANGE:  DISABLE;
             CALL PRINT(@MESSF,30,30,1,10,3,0);
             CALL READ(@DUMMY,@UMAX);
             CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@OUTPUT$LENGTH);
             GO TO THERE;
END MINA;
```

```
STAT20:
     DO;
/*                  OCT 4, 1984                    */
/*                  NOV 28, 1984                   */


DECLARE MEAN(17) INTEGER;
DECLARE SD(17) INTEGER;
DECLARE MINIMA(17) INTEGER;
DECLARE MAXIMA(17) INTEGER;
DECLARE XX(17) STRUCTURE (YY(17) INTEGER);
DECLARE MEAN1(17) WORD;
DECLARE XX1(17) STRUCTURE (YY1(17) WORD);
DECLARE (NP,TEMP) INTEGER;
DECLARE (Y,N,TOTAL,B,D,D1,F) WORD;
DECLARE (A,C,C1,E) INTEGER;
DECLARE PAR(17) INTEGER;
DECLARE MESS1(27) BYTE DATA ('  ENTER NUM OF PARAMETERS  ');
DECLARE MESS2(18) BYTE DATA ('  ENTER THRESHOLD ');
DECLARE (I,J,NUM,X,FLAG,CHECK,THRESH,SIGN,NN) BYTE;
DECLARE (II,JJ) BYTE;
DECLARE (HHB,HB,LB) BYTE;
DECLARE BUFF1(130) BYTE;
DECLARE BUFF2(17) INTEGER;
DECLARE BUFF3(17) BYTE;
DECLARE LIST(17) BYTE;
DECLARE NMAX BYTE;
DECLARE MESS3(18) BYTE DATA ('  THE RESULTS ARE ');
DECLARE MESS4(66) BYTE DATA ('  MEAN    MEAN SQU. VARIANCE
       S.D.    MAXIMA   MINIMA      N    ');
DECLARE MESS5(31) BYTE DATA ('  ENTER NUMBER OF DATA POINTS  ');
DECLARE MESS6(31) BYTE DATA ('  THE COVARIANCE MATRIX IS     ');
DECLARE MESS7(46) BYTE DATA ('  ENTER A-D CHANNEL NUMBER FOR
       PARAMETER      ');
DECLARE MESS9(37) BYTE DATA (' HIT ANY KEY TO START, HIT S
       TO STOP.');
DECLARE (POS,NEG) BYTE;
DECLARE DUMB(4) BYTE;
DECLARE LEN WORD;
DECLARE (FIRST,SECOND) BYTE;


 PRINT: PROCEDURE (MES$LOC,MES$LEN,GRP$LEN,NGRP,LD$SPC,SPC,NCRLF) EXTERNAL;
          DECLARE MES$LOC POINTER;
        DECLARE (MES$LEN,GRP$LEN,NGRP,LDSPC,SPC,NCRLF) BYTE;
  END PRINT;

   READ: PROCEDURE (MESSAGE,LENGTH) EXTERNAL;
           DECLARE (MESSAGE,LENGTH) POINTER;
    END READ;

    ASC2HEX:  PROCEDURE (ASCII,N,HIGHEST,HI,LO)EXTERNAL;
          DECLARE (ASCII,HIGHEST,HI,LO) POINTER;
          DECLARE N BYTE;
     END ASC2HEX;

    HEX2ASC:  PROCEDURE (NUMBER,CHR1,CHR2) EXTERNAL;
              DECLARE NUMBER BYTE,(CHR1,CHR2) POINTER;
    END HEX2ASC;

FORMAT: PROCEDURE (X,MAX,SCALE,NUMDEC,WIDTH,LOC) EXTERNAL;
          DECLARE X INTEGER;
          DECLARE (MAX,SCALE) WORD;
          DECLARE (NUMDEC,WIDTH) BYTE;
          DECLARE LOC POINTER;
```

```
END FORMAT;

    SQRT: PROCEDURE(NUMBER) INTEGER EXTERNAL;
          DECLARE NUMBER INTEGER;
    END SQRT;

       ADD: PROCEDURE (A,B,C,D,E,F) EXTERNAL;
            DECLARE (A,C) INTEGER;
            DECLARE (B,D) WORD;
            DECLARE (E,F) POINTER;
       END ADD;

       DIVIDE: PROCEDURE (A,B,C,D,E) EXTERNAL;
            DECLARE (A,C) INTEGER;
            DECLARE B WORD;
            DECLARE (D,E) POINTER;
       END DIVIDE;

       MULTIPLY: PROCEDURE (A,B,C,D,E,F) EXTERNAL;
            DECLARE (A,C) INTEGER;
            DECLARE (B,D) WORD;
            DECLARE (E,F) POINTER;
       END MULTIPLY;

       ENABLED$INKEY: PROCEDURE BYTE EXTERNAL;
       END ENABLED$INKEY;

INITAD:
 PROCEDURE(COUNT,CHANNEL$NUMBER,START$SELECT,INTR$ENABLE)EXTERNAL;
     DECLARE(CHANNELNUMBER,STARTSELECT,INTR$ENABLE)BYTE;
     DECLARE COUNT WORD;
END INITAD;

  IN16AD:
      PROCEDURE(CNT,CH$NO,STRT$SEL,INT$ENABLE) EXTERNAL;
        DECLARE (CH$NO,STRT$SEL,INT$ENABLE) BYTE;
        DECLARE CNT WORD;
  END IN16AD;
/*-------------------------------------------------------------------------*/

/* THIS PROGRAM IS USED TO COMPUTE VARIOUS STATISTICS OF A SET OF
INPUT PARAMETERS. THE STATISTICS ARE MEANS,STANDARD DEVIATIONS,
MINIMA,MAXIMA, AND CORRELATIONS.  THE NUMBER OF PARAMETERS IS NUM.
THE PARAMETERS ARE ASSUMED TO BE INPUT ON A-D CHANNELS 0 TO NUM-1.
THE MAXIMUM NUMBER OF PARAMETERS IS 16.
    THE KEYBOARD IS USED TO START AND STOP THE DATA COLLECTION
PROCESS.  A RESPONSE OF 0 FOR THE TOTAL NUMBER OF SAMPLES WILL
MAKE THE PROGRAM TO CONTINUE SAMPLING UNTILL THE 'S' KEY IS PRESSED.


INTRPT:
       PROCEDURE INTERRUPT 27;
       IF (INPUT(26H) < THRESH) THEN GOTO NEXT;    /*CHECK ENERGY*/
       FLAG=1;   /*AT LEAST ONE DATA POINT*/
       N=N+1;

      DO I =0 TO (NUM-1) BY 1;
           X=INPUT(LIST(I)+30H);
           PAR(I)=(INT(X))-80H; /*CODE CONVERSION*/
      END;

     DO I=0 TO (NUM-1);
          IF (PAR(I) <0) THEN A=-1;
             ELSE A=00H;
          B=UNSIGN(PAR(I));
          C=MEAN(I);
```

```
          D=MEAN1(I);
          CALL ADD(A,B,C,D,@E,@F);
          MEAN(I)=E;
          MEAN1(I)=F;
           IF(PAR(I) < MINIMA(I)) THEN
                MINIMA(I)=PAR(I);
           IF(PAR(I) > MAXIMA(I)) THEN
                MAXIMA(I)=PAR(I);

     DO J=0 TO I;
          TEMP=PAR(I)*PAR(J);
          IF (TEMP <0) THEN A=-1;
             ELSE A=00H;
          B=UNSIGN(TEMP);
          C1=XX(I).YY(J);
          D1=XX1(I).YY1(J);
          CALL ADD(A,B,C1,D1,@E,@F);
          XX(I).YY(J)=E;
          XX1(I).YY1(J)=F;
     END;
   END;

   NEXT:
   FINISH:
END INTRPT;

    /* INITIALIZE ALL PARAMETERS BEFORE SAMPLING BEGINS */

  /* NEXT SECTION TO INPUT PARAMETERS*/
START:
   OUTPUT(02H)=13H;
   OUTPUT(03H)=24;
   OUTPUT(03H)=0FH;
   OUTPUT(03H)=0FFH;/*INITIALIZE THE 8259 TO MASK OFF ALL INTERRUPTS*/

       CALL PRINT(@MESS1,26,26,1,10,0,0);
       CALL READ(@BUFF1,@NMAX);
       CALL ASC2HEX(@BUFF1,NMAX,@HHB,@HB,@NUM);
       IF (NUM <1) THEN NUM=1;
       IF (NUM>16) THEN NUM=16;
       CALL HEX2ASC(NUM,@FIRST,@SECOND);
       BUFF1(0)=FIRST;
       BUFF1(1)=SECOND;
       CALL PRINT(@BUFF1,2,2,1,2,0,2);
       DUMB(2)='?';
       DUMB(3)=' ';

    DO I=1 TO NUM;
          CALL HEX2ASC(I,@FIRST,@SECOND);
          DUMB(0)=FIRST;
          DUMB(1)=SECOND;
       CALL PRINT(@MESS7,43,43,1,10,0,0);
       CALL PRINT(@DUMB,4,4,1,2,0,0);
       CALL READ(@BUFF1,@NMAX);
IF BUFF1(0)=0DH THEN GOTO CONT;  /*KEEP OLD VALUE IF CARRIAGE RETURN*/
          CALL ASC2HEX(@BUFF1,NMAX,@HHB,@HB,@LB);
          LIST(I-1)=LB;
    CONT:  CALL HEX2ASC(LIST(I-1),@FIRST,@SECOND);
            BUFF1(0)=FIRST;
            BUFF1(1)=SECOND;
          CALL PRINT(@BUFF1,2,2,1,2,0,2);
       END;

       CALL PRINT(@MESS2,18,18,1,10,0,0);
       CALL READ(@BUFF1,@NMAX);
       I=NMAX;
```

```
        CALL ASC2HEX(@BUFF1,I,@HHB,@HB,@THRESH);
        CALL PRINT(@MESS5,31,31,1,10,0,0);
        CALL READ(@BUFF1,@NMAX);
        CALL ASC2HEX(@BUFF1,NMAX,@HHB,@HB,@LB);
        TOTAL=256*HB+LB;

       TEMP=INT(TOTAL);
       CALL FORMAT(TEMP,1,1,0,7,@BUFF3);
       CALL PRINT(@BUFF3,7,7,1,3,0,2);

    DO I=0 TO (NUM-1);
         MEAN(I)=0;
         MEAN1(I)=0;
         SD(I)=0;
         MINIMA(I)=255;
         MAXIMA(I)=-255;
         FLAG=0;
         N=0;
         NP=0;
         POS='+';
         NEG='-';
              DO J=0 TO (NUM-1);
         XX(I).YY(J)=0;
         XX1(I).YY1(J)=0;
    END;
   END;

  CALL INITAD(52000,8,2,3);/* INITIALIZE THE 8 CH. A/D BOARD.  */
  CALL IN16AD(52000,16,2,3);/* INITIALIZE THE 16 CH. A/D BOARD.  */

        CALL PRINT(@MESS9,37,37,1,10,0,2);
WAIT:
  IF ENABLED$INKEY=0 THEN GO TO WAIT; /* WAIT UNTIL A KEY IS HIT */
     ENABLE;
LOOP:
       IF TOTAL=0 THEN GO TO OVER;
       IF (N>=TOTAL) THEN GO TO OUT;
  OVER: IF ENABLED$INKEY='S' THEN GO TO OUT;
       GO TO LOOP;

OUT:
      OUTPUT(13H)=10110110B;
      OUTPUT(12H)=00H;  /*SHUT OFF THE 8253 TIMER*/
OUT1:
       CALL PRINT(@MESS3,18,18,1,10,0,2);
       CALL PRINT(@MESS4,66,66,1,0,0,1);

       DO I=0 TO (NUM-1);       /*INSERT SCALE FACTORS*/
          CALL MULTIPLY(MEAN(I),MEAN1(I),0,100,@E,@F);
          MEAN(I)=E;
          MEAN1(I)=F;
          TEMP=SIGNED(N);
          CALL DIVIDE(MEAN(I),MEAN1(I),TEMP,@E,@F);
          MEAN(I)=SIGNED(F);    /*RESULT IS 100*MEAN */
        DO J=0 TO I;
           C=XX(I).YY(J);
           D=XX1(I).YY1(J);
           CALL DIVIDE(C,D,TEMP,@E,@F);
           XX(I).YY(J)=SIGNED(F);
           IF (MEAN(I) <0) THEN A=-1;
              ELSE A=0;
           IF (MEAN(J) <0) THEN C=-1;
              ELSE C=0;
           B=UNSIGN(MEAN(I));
           D=UNSIGN(MEAN(J));
           CALL MULTIPLY(A,B,C,D,@E,@F);
```

```
        CALL DIVIDE(E,F,10000,@A,@B);
     XX1(I).YY1(J)=B;    /* RESULT IS THE SQUARE OF THE MEAN */
    END;
END;

    DO I=0 TO 130;
       BUFF1(I)=' ';
    END;

    DO I=0 TO (NUM-1) BY 1;
        TEMP=MEAN(I)/100;
        CALL FORMAT(TEMP,128,250,2,5,@BUFF3);
       DO II=0 TO 4;
         BUFF1(II)=BUFF3(II);
       END;

        BUFF2(1)=XX(I).YY(I);
        TEMP=BUFF2(1)/32;
        TEMP=(49*TEMP)/4;
        CALL FORMAT(TEMP,6250,6250,3,6,@BUFF3);
       DO II=0 TO 5;
          BUFF1(10+II)=BUFF3(II);
       END;

        BUFF2(2)=BUFF2(1)-SIGNED(XX1(I).YY1(I));
        TEMP=BUFF2(2)/32;
        TEMP=(49*TEMP)/4;
        CALL FORMAT(TEMP,6250,6250,3,6,@BUFF3);
       DO II=0 TO 5;
          BUFF1(20+II)=BUFF3(II);
       END;
        BUFF2(3)=SQRT(BUFF2(2));
        TEMP=BUFF2(3);
        CALL FORMAT(TEMP,128,250,2,5,@BUFF3);
       DO II=0 TO 4;
          BUFF1(30+II)=BUFF3(II);
       END;
        TEMP=MAXIMA(I);
        CALL FORMAT(TEMP,128,250,2,5,@BUFF3);
       DO II=0 TO 4;
          BUFF1(40+II)=BUFF3(II);
       END;

        BUFF2(5)=MINIMA(I);
        TEMP=BUFF2(5);
        CALL FORMAT(TEMP,128,250,2,5,@BUFF3);
       DO II=0 TO 4;
          BUFF1(50+II)=BUFF3(II);
       END;

        TEMP=SIGNED(N);
        CALL FORMAT(TEMP,1000,1000,0,7,@BUFF3);
       DO II=0 TO 6;
           BUFF1(60+II)=BUFF3(II);
       END;

     CALL PRINT(@BUFF1,70,70,1,0,0,1);
END;
     CALL PRINT(@MESS6,31,31,1,10,0,1);
DO I=0 TO (NUM-1);
  DO J=0 TO (NUM-1);
     IF (J>I) THEN JJ=I;
      ELSE JJ=J;
     IF (J>I) THEN II=J;
      ELSE II=I;
   BUFF2(J)=(XX(II).YY(JJ))-(SIGNED(XX1(II).YY1(JJ)));
```

```
          TEMP=BUFF2(J)/32;
          TEMP=(49*TEMP)/4;
          CALL FORMAT(TEMP,6250,6250,3,6,@BUFF3);
          DO II=0 TO 5;
             BUFF1(6*J+II)=BUFF3(II);
          END;
       END;
       J=6*NUM;
       CALL PRINT(@BUFF1,J,6,NUM,0,1,2);
  END;
       GOTO START;
END STAT20;
```

```
STAT21:
    DO;
/*                  OCT 4, 1984                  */
/*                  NOV 28, 1984                 */

DECLARE MEAN(17) INTEGER;
DECLARE SD(17) INTEGER;
DECLARE MINIMA(17) INTEGER;
DECLARE MAXIMA(17) INTEGER;
DECLARE XX(17) STRUCTURE (YY(17) INTEGER);
DECLARE MEAN1(17) WORD;
DECLARE XX1(17) STRUCTURE (YY1(17) WORD);
DECLARE (NP,TEMP) INTEGER;
DECLARE (Y,N,TOTAL,B,D,D1,F) WORD;
DECLARE (A,C,C1,E) INTEGER;
DECLARE PAR(17) INTEGER;
DECLARE MESS1(27) BYTE DATA ('  ENTER NUM OF PARAMETERS  ');
DECLARE MESS2(18) BYTE DATA ('  ENTER THRESHOLD ');
DECLARE (I,J,NUM,X,FLAG,CHECK,THRESH,SIGN,NN) BYTE;
DECLARE (II,JJ) BYTE;
DECLARE (HHB,HB,LB) BYTE;
DECLARE BUFF1(130) BYTE;
DECLARE BUFF2(17) INTEGER;
DECLARE BUFF3(17) BYTE;
DECLARE LIST(17) BYTE;
DECLARE NMAX BYTE;
DECLARE MESS3(18) BYTE DATA ('  THE RESULTS ARE ');
DECLARE MESS4(66) BYTE DATA ('  MEAN     MEAN SQU. VARIANCE
       S.D.     MAXIMA   MINIMA      N     ');
DECLARE MESS5(31) BYTE DATA ('  ENTER NUMBER OF DATA POINTS  ');
DECLARE MESS6(31) BYTE DATA ('  THE COVARIANCE MATRIX IS     ');
DECLARE MESS7(46) BYTE DATA ('  ENTER A-D CHANNEL NUMBER FOR
       PARAMETER       ');
DECLARE MESS9(37) BYTE DATA (' HIT ANY KEY TO START, HIT S
       TO STOP.');
DECLARE (POS,NEG) BYTE;
DECLARE DUMB(4) BYTE;
DECLARE LEN WORD;
DECLARE (FIRST,SECOND) BYTE;


 PRINT: PROCEDURE (MES$LOC,MES$LEN,GRP$LEN,NGRP,LD$SPC,SPC,NCRLF) EXTERNAL;
          DECLARE MES$LOC POINTER;
        DECLARE (MES$LEN,GRP$LEN,NGRP,LDSPC,SPC,NCRLF) BYTE;
  END PRINT;

   READ: PROCEDURE (MESSAGE,LENGTH) EXTERNAL;
           DECLARE (MESSAGE,LENGTH) POINTER;
    END READ;

    ASC2HEX:  PROCEDURE (ASCII,N,HIGHEST,HI,LO)EXTERNAL;
          DECLARE (ASCII,HIGHEST,HI,LO) POINTER;
          DECLARE N BYTE;
     END ASC2HEX;

    HEX2ASC:  PROCEDURE (NUMBER,CHR1,CHR2) EXTERNAL;
              DECLARE NUMBER BYTE,(CHR1,CHR2) POINTER;
    END HEX2ASC;

FORMAT: PROCEDURE (X,MAX,SCALE,NUMDEC,WIDTH,LOC) EXTERNAL;
          DECLARE X INTEGER;
          DECLARE (MAX,SCALE) WORD;
          DECLARE (NUMDEC,WIDTH) BYTE;
          DECLARE LOC POINTER;
```

```
END FORMAT;

    SQRT: PROCEDURE(NUMBER) INTEGER EXTERNAL;
          DECLARE NUMBER INTEGER;
    END SQRT;

       ADD: PROCEDURE (A,B,C,D,E,F) EXTERNAL;
            DECLARE (A,C) INTEGER;
            DECLARE (B,D) WORD;
            DECLARE (E,F) POINTER;
       END ADD;

       DIVIDE: PROCEDURE (A,B,C,D,E) EXTERNAL;
            DECLARE (A,C) INTEGER;
            DECLARE B WORD;
            DECLARE (D,E) POINTER;
       END DIVIDE;

       MULTIPLY: PROCEDURE (A,B,C,D,E,F) EXTERNAL;
            DECLARE (A,C) INTEGER;
            DECLARE (B,D) WORD;
            DECLARE (E,F) POINTER;
       END MULTIPLY;

       ENABLED$INKEY: PROCEDURE BYTE EXTERNAL;
       END ENABLED$INKEY;

INITAD:
 PROCEDURE(COUNT,CHANNEL$NUMBER,START$SELECT,INTR$ENABLE)EXTERNAL;
     DECLARE(CHANNELNUMBER,STARTSELECT,INTR$ENABLE)BYTE;
     DECLARE COUNT WORD;
END INITAD;

  IN16AD:
      PROCEDURE(CNT,CH$NO,STRT$SEL,INT$ENABLE) EXTERNAL;
        DECLARE (CH$NO,STRT$SEL,INT$ENABLE) BYTE;
        DECLARE CNT WORD;
  END IN16AD;
/*-------------------------------------------------------------------------*/

/* THIS PROGRAM IS USED TO COMPUTE VARIOUS STATISTICS OF A SET OF
INPUT PARAMETERS. THE STATISTICS ARE MEANS,STANDARD DEVIATIONS,
MINIMA,MAXIMA, AND CORRELATIONS.  THE NUMBER OF PARAMETERS IS NUM.
THE PARAMETERS ARE ASSUMED TO BE INPUT ON A-D CHANNELS 0 TO NUM-1.
THE MAXIMUM NUMBER OF PARAMETERS IS 16.
    THE KEYBOARD IS USED TO START AND STOP THE DATA COLLECTION
PROCESS.  A RESPONSE OF 0 FOR THE TOTAL NUMBER OF SAMPLES WILL
MAKE THE PROGRAM TO CONTINUE SAMPLING UNTILL THE 'S' KEY IS PRESSED.


INTRPT:
       PROCEDURE INTERRUPT 27;
       IF (INPUT(26H) < THRESH) THEN GOTO NEXT;    /*CHECK ENERGY*/
       FLAG=1;    /*AT LEAST ONE DATA POINT*/
       N=N+1;

         SUM=0;
      DO I =0 TO (NUM-1) BY 1;
           X=INPUT(LIST(I)+30H);
           PAR(I)=(INT(X))-80H; /*CODE CONVERSION*/
           SUM=SUM+PAR(I);
      END;

            SUM=SUM/INT(NUM);

      DO I=0 TO (NUM-1);
```

```
            PAR(I)=(PAR(I)-SUM)/2;
        END;

        DO I=0 TO (NUM-1);
            IF (PAR(I) <0) THEN A=-1;
               ELSE A=00H;
            B=UNSIGN(PAR(I));
            C=MEAN(I);
            D=MEAN1(I);
            CALL ADD(A,B,C,D,@E,@F);
            MEAN(I)=E;
            MEAN1(I)=F;
             IF(PAR(I) < MINIMA(I)) THEN
                  MINIMA(I)=PAR(I);
             IF(PAR(I) > MAXIMA(I)) THEN
                  MAXIMA(I)=PAR(I);

         DO I=0 TO (NUM-1);
            PAR(I)=SAR(PAR(I),1);
         END;

        DO J=0 TO I;
              TEMP=PAR(I)*PAR(J);
              IF (TEMP <0) THEN A=-1;
                 ELSE A=00H;
              B=UNSIGN(TEMP);
              C1=XX(I).YY(J);
              D1=XX1(I).YY1(J);
              CALL ADD(A,B,C1,D1,@E,@F);
              XX(I).YY(J)=E;
              XX1(I).YY1(J)=F;
        END;
      END;

      NEXT:
      FINISH:
   END INTRPT;

       /* INITIALIZE ALL PARAMETERS BEFORE SAMPLING BEGINS */

     /* NEXT SECTION TO INPUT PARAMETERS*/
   START:
      OUTPUT(02H)=13H;
      OUTPUT(03H)=24;
      OUTPUT(03H)=0FH;
      OUTPUT(03H)=0FFH;/*INITIALIZE THE 8259 TO MASK OFF ALL INTERRUPTS*/

          CALL PRINT(@MESS1,26,26,1,10,0,0);
          CALL READ(@BUFF1,@NMAX);
          CALL ASC2HEX(@BUFF1,NMAX,@HHB,@HB,@NUM);
          IF (NUM <1) THEN NUM=1;
          IF (NUM>16) THEN NUM=16;
          CALL HEX2ASC(NUM,@FIRST,@SECOND);
          BUFF1(0)=FIRST;
          BUFF1(1)=SECOND;
          CALL PRINT(@BUFF1,2,2,1,2,0,2);
          DUMB(2)='?';
          DUMB(3)=' ';

       DO I=1 TO NUM;
             CALL HEX2ASC(I,@FIRST,@SECOND);
             DUMB(0)=FIRST;
             DUMB(1)=SECOND;
          CALL PRINT(@MESS7,43,43,1,10,0,0);
          CALL PRINT(@DUMB,4,4,1,2,0,0);
          CALL READ(@BUFF1,@NMAX);
```

```
IF BUFF1(0)=0DH THEN GOTO CONT;  /*KEEP OLD VALUE IF CARRIAGE RETURN*/
          CALL ASC2HEX(@BUFF1,NMAX,@HHB,@HB,@LB);
          LIST(I-1)=LB;
    CONT:  CALL HEX2ASC(LIST(I-1),@FIRST,@SECOND);
            BUFF1(0)=FIRST;
            BUFF1(1)=SECOND;
          CALL PRINT(@BUFF1,2,2,1,2,0,2);
       END;

       CALL PRINT(@MESS2,18,18,1,10,0,0);
       CALL READ(@BUFF1,@NMAX);
       I=NMAX;
       CALL ASC2HEX(@BUFF1,I,@HHB,@HB,@THRESH);
       CALL PRINT(@MESS5,31,31,1,10,0,0);
       CALL READ(@BUFF1,@NMAX);
       CALL ASC2HEX(@BUFF1,NMAX,@HHB,@HB,@LB);
       TOTAL=256*HB+LB;

      TEMP=INT(TOTAL);
      CALL FORMAT(TEMP,1,1,0,7,@BUFF3);
      CALL PRINT(@BUFF3,7,7,1,3,0,2);

    DO I=0 TO (NUM-1);
         MEAN(I)=0;
         MEAN1(I)=0;
         SD(I)=0;
         MINIMA(I)=255;
         MAXIMA(I)=-255;
         FLAG=0;
         N=0;
         NP=0;
         POS='+';
         NEG='-';
              DO J=0 TO (NUM-1);
         XX(I),YY(J)=0;
         XX1(I),YY1(J)=0;
   END;
  END;

  CALL INITAD(55000,8,2,3);/* INITIALIZE THE 8 CH. A/D BOARD.  */
  CALL IN16AD(55000,16,2,3);/* INITIALIZE THE 16 CH. A/D BOARD.  */

        CALL PRINT(@MESS9,37,37,1,10,0,2);
WAIT:
  IF ENABLED$INKEY=0 THEN GO TO WAIT; /* WAIT UNTIL A KEY IS HIT */
     ENABLE;
LOOP:
       IF TOTAL=0 THEN GO TO OVER;
       IF (N>=TOTAL) THEN GO TO OUT;
  OVER: IF ENABLED$INKEY='S' THEN GO TO OUT;
       GO TO LOOP;

OUT:
      OUTPUT(13H)=10110110B;
      OUTPUT(12H)=00H;  /*SHUT OFF THE 8253 TIMER*/
OUT1:
       CALL PRINT(@MESS3,18,18,1,10,0,2);
       CALL PRINT(@MESS4,66,66,1,0,0,1);

       DO I=0 TO (NUM-1);       /*INSERT SCALE FACTORS*/
          CALL MULTIPLY(MEAN(I),MEAN1(I),0,200,@E,@F);
          MEAN(I)=E*2;
          MEAN1(I)=F*2;
          TEMP=SIGNED(N);
          CALL DIVIDE(MEAN(I),MEAN1(I),TEMP,@F,@F);
          MEAN(I)=SIGNED(F);    /*RESULT IS 100*MEAN */
```

```
     DO J=0 TO I;
        C=XX(I).YY(J);
        D=XX1(I).YY1(J);
           CALL MULTIPLY(C,D,0,4,@E,@F);
           C=E;
           D=F;
        CALL DIVIDE(C,D,TEMP,@E,@F);
        XX(I).YY(J)=SIGNED(F)*4;
        IF (MEAN(I) <0) THEN A=-1;
           ELSE A=0;
        IF (MEAN(J) <0) THEN C=-1;
           ELSE C=0;
        B=UNSIGN(MEAN(I));
        D=UNSIGN(MEAN(J));
        CALL MULTIPLY(A,B,C,D,@E,@F);
        CALL DIVIDE(E,F,10000,@A,@B);
     XX1(I).YY1(J)=B;   /* RESULT IS THE SQUARE OF THE MEAN */
    END;
END;

    DO I=0 TO 130;
       BUFF1(I)=' ';
    END;

    DO I=0 TO (NUM-1) BY 1;
        TEMP=MEAN(I)/100;
        CALL FORMAT(TEMP,128,250,2,5,@BUFF3);
       DO II=0 TO 4;
         BUFF1(II)=BUFF3(II);
       END;

        BUFF2(1)=XX(I).YY(I);
        TEMP=BUFF2(1)/32;
        TEMP=(49*TEMP)/4;
        CALL FORMAT(TEMP,6250,6250,3,6,@BUFF3);
       DO II=0 TO 5;
          BUFF1(10+II)=BUFF3(II);
       END;

        BUFF2(2)=BUFF2(1)-SIGNED(XX1(I).YY1(I));
        TEMP=BUFF2(2)/32;
        TEMP=(49*TEMP)/4;
        CALL FORMAT(TEMP,6250,6250,3,6,@BUFF3);
       DO II=0 TO 5;
          BUFF1(20+II)=BUFF3(II);
       END;
        BUFF2(3)=SQRT(BUFF2(2));
        TEMP=BUFF2(3);
        CALL FORMAT(TEMP,128,250,2,5,@BUFF3);
       DO II=0 TO 4;
          BUFF1(30+II)=BUFF3(II);
       END;
        TEMP=MAXIMA(I)*2;
        CALL FORMAT(TEMP,128,250,2,5,@BUFF3);
       DO II=0 TO 4;
          BUFF1(40+II)=BUFF3(II);
       END;

        BUFF2(5)=MINIMA(I)*2;
        TEMP=BUFF2(5);
        CALL FORMAT(TEMP,128,250,2,5,@BUFF3);
       DO II=0 TO 4;
          BUFF1(50+II)=BUFF3(II);
       END;

        TEMP=SIGNED(N);
```

```
            CALL FORMAT(TEMP,1000,1000,0,7,@BUFF3);
           DO II=0 TO 6;
               BUFF1(60+II)=BUFF3(II);
           END;

          CALL PRINT(@BUFF1,70,70,1,0,0,1);
      END;
          CALL PRINT(@MESS6,31,31,1,10,0,1);
     DO I=0 TO (NUM-1);
       DO J=0 TO (NUM-1);
          IF (J>I) THEN JJ=I;
           ELSE JJ=J;
          IF (J>I) THEN II=J;
           ELSE II=I;
        BUFF2(J)=(XX(II),YY(JJ))-(SIGNED(XX1(II),YY1(JJ)));
          TEMP=BUFF2(J)/32;
          TEMP=(49*TEMP)/4;
          CALL FORMAT(TEMP,6250,6250,3,6,@BUFF3);
         DO II=0 TO 5;
            BUFF1(6*J+II)=BUFF3(II);
         END;
       END;
       J=6*NUM;
       CALL PRINT(@BUFF1,J,6,NUM,0,1,2);
  END;
      GOTO START;
END STAT21;
```

```
VOWEL:
        DO;      /*   JAN 8, 1985
-------------------------------------------------------------------
     THIS PROGRAM CAN SAMPLE THE INPUT UP TO A SAMPLING FREQUENCY
     OF 10526 Hz.  THEN IT CAN PLAYBACK THE DATA REPEATEDLY IN
     ANY SPECIFIED LENGTH (MAXIMUM LENGTH IS 25000.).  THE SPECIFIED
     FRAME OF PLAYBACK CAN BE ADVANCED THROUGH THE DATA IN BOTH DIRECTIONS
     IN LENGTHS OF 10., 100., OR 1000. WHEN THE CORRECT KEY IS HIT.
     YOU CAN ALSO GO INTO A MODE IN WHICH WHEN THE 'P' KEY IS HIT, THE
     FOCUSED BLOCK OF DATA IS PLAYED BACK AND AT THE END OF THIS PLAYBACK
     THE FIRST FIVE PC'S ARE SAMPLED AND THEN DISPLAYED ON THE SCREEN.
     YOU CAN ALSO ADVANCE THROUGH THE DATA OR GO BACK AS MENTIONED ABOVE.
-------------------------------------------------------------------
            TIME= SAMPLING TIME IN MICROSECONDS.
          LENGTH= NUMBER OF SAMPLES TO BE SENT OUT.

    THIS PROGRAM ASSUMES THAT THE INPUT IS AT A/D PORT 20H AND THAT THE
    OUTPUT IS AT D/A PORT 18H.
    THIS PROGRAM ALSO OUTPUTS A SYNC. SIGNAL TO D/A PORT 16H.

        THE FOLLOWING EXPLAINS THE FUNCTION OF EACH KEY WHEN THE
        PROGRAM IS RUNNING.

             F=  ADVANCE BY 10 POINTS.
             G=  ADVANCE BY 100 POINTS.
             H=  ADVANCE BY 1000 POINTS.
             R=  GO BACK BY 10 POINTS.
             T=  GO BACK BY 100 POINTS.
             Y=  GO BACK BY 1000 POINTS.
             L=  CHANGE THE LENGTH OF THE PLAYBACK.
             Q=  START THE PROGRAM OVER AGAIN.
             M=  CHANGE BETWEEN THE TWO MODES.
             C=  TO CHANGE THE FREQUENCY OF THE PLAYBACK.
             P= TO SAMPLE THE PARAMETERS AND PRINT THEM ON THE TERMINAL
-------------------------------------------------------------------
*/
        INITAD: PROCEDURE(A,B,C,D) EXTERNAL;
                DECLARE A WORD;
                DECLARE(B,C,D) BYTE;
        END INITAD;

        PRINT:  PROCEDURE(A,B,C,D,E,F,H) EXTERNAL;
                DECLARE A POINTER;
                DECLARE (B,C,D,E,F,H) BYTE;
        END PRINT;

        FORMAT: PROCEDURE(X,MAX,SCALE,NUMDEC,WIDTH,LOC) EXTERNAL;
                DECLARE LOC POINTER,X INTEGER,(MAX,SCALE) WORD;
                DECLARE (NUMDEC,WIDTH) BYTE;
        END FORMAT;

        GET: PROCEDURE BYTE EXTERNAL;
        END GET;

        ENABLED$INKEY: PROCEDURE BYTE EXTERNAL;
        END ENABLED$INKEY;

             DECLARE (I,NUMBER,LENGTH,START,FINISH) INTEGER;
             DECLARE (J,RATE,LEN,SAMPLING$RATE) WORD;
             DECLARE (C,CHAR,UMAX,HHB,HB,LB) BYTE;
             DECLARE DUMMY(7) BYTE;
             DECLARE P(5) BYTE;
   DECLARE MESS1(34) BYTE DATA ('HIT ANY KEY TO START THE SAMPLING ');
   DECLARE MESS2(34) BYTE DATA ('HIT ANY KEY TO START THE PLAYBACK ');
   DECLARE MESS3(34) BYTE DATA ('ENTER THE TIME IN MICROSECONDS    ');
```

```
   DECLARE MESS4(34) BYTE DATA ('ENTER THE LENGTH OF THE PLAYBACK  ');
DECLARE MESS5(51) BYTE DATA ('ENTER THE PARAMETER SAMPLING TIME IN MICROSECONDS ')
             DECLARE DATA$BUFFER(25001) BYTE;

     READ: PROCEDURE(MESS,LENGTH) EXTERNAL;
             DECLARE (MESS,LENGTH) POINTER;
     END READ;

     ASC2HEX: PROCEDURE(BUFF,LEN,HHB,HB,LB)  EXTERNAL;
             DECLARE (BUFF,HHB,HB,LB) POINTER;
             DECLARE LEN BYTE;
      END ASC2HEX;

            DISABLE;
    BEGIN:   I=0;
             C=0;
             J=0;
             CALL PRINT(@MESS5,51,51,1,10,0,0);
             CALL READ(@DUMMY,@UMAX);
             CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
             SAMPLING$RATE=256*HB+LB;

  LOOP2:     CALL PRINT(@MESS3,34,34,1,10,0,0);
             CALL READ(@DUMMY,@UMAX);
             CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
             RATE=256*HB+LB;

             IF C=1 THEN GO TO LOOP3;
             CALL PRINT(@MESS1,34,34,1,10,0,2);
             CHAR=GET;
LOOP3:       CALL INITAD(RATE,1,2,0);
             IF C=1 THEN GO TO LOOP;

        DO I= 0 TO 25000;
OVER:       OUTPUT(2)=00001100B;   /*  POLL MODE   */
             IF INPUT(2)<128 THEN GO TO OVER;
            DATA$BUFFER(I)=INPUT(20H);
            OUTPUT(2)=01100000B;  /* SPECIFIED  EOI   */
        END;

                START=0;
   LOOP:     CALL PRINT(@MESS4,34,34,1,10,0,0);
             CALL READ(@DUMMY,@UMAX);
             CALL ASC2HEX(@DUMMY,UMAX,@HHB,@HB,@LB);
             LEN=256*DOUBLE(HB)+LB;
             LENGTH=INT(LEN);
             CALL PRINT(@MESS2,34,34,1,10,0,2);
             CHAR=GET;
             CALL INITAD(RATE,1,2,0);
             FINISH=LENGTH+START;

   OVER2:          DO I= START TO FINISH;
     AGAIN:  OUTPUT(2)=00001100B; /* POLL MODE  */
             IF INPUT(2)<128 THEN GO TO AGAIN;
              OUTPUT(18H)=DATA$BUFFER(I);
              OUTPUT(2)=01100000B;  /*  SPECIFIED   EOI  */
                    END;

             CHAR=ENABLED$INKEY;

    DECIDE: IF CHAR<41H THEN GO TO NEXT$FRAME;

                 IF CHAR='F' THEN START=START+10;
                 IF CHAR='G' THEN START=START+100;
                 IF CHAR='H' THEN START=START+1000;
                 IF CHAR='R' THEN START=START-10;
```

```
                IF CHAR='T' THEN START=START-100;
                IF CHAR='Y' THEN START=START-1000;
                IF CHAR='L' THEN GO TO LOOP;
                IF CHAR='M' THEN GO TO LOOP5;
                IF CHAR='Q' THEN GO TO BEGIN;
                IF CHAR='C' THEN DO;
                            C=1;
                            GO TO LOOP2;
                                END;
                IF START<0 THEN START=0;
                IF START>(25000-LENGTH) THEN START=25000-LENGTH;

                CALL FORMAT(START,1,1,0,7,@DUMMY);
                CALL PRINT(@DUMMY,7,7,1,5,0,1);

NEXT$FRAME:
          FINISH=START+LENGTH;
          OUTPUT(16H)=255;
          CALL TIME(1);
          OUTPUT(16H)=0;
          GO TO OVER2;

LOOP5:   DO I=START TO FINISH;
WAIT:         OUTPUT(2)=00001100B;   /*  POLL MODE  */
              IF INPUT(2)<128 THEN GO TO WAIT;
              OUTPUT(18H)=DATA$BUFFER(I);
              OUTPUT(2)=01100000B;  /*   SPECIFIED EOI  */
         END;

             CALL INITAD(SAMPLING$RATE,5,2,0);

WAIT2:    OUTPUT(2)=00001100B;  /*  POLL MODE  */
          IF INPUT(2)<128 THEN GO TO WAIT2;
          P(0)=INPUT(20H);
          P(1)=INPUT(21H);
          P(2)=INPUT(22H);
          P(3)=INPUT(23H);
          P(4)=INPUT(24H);
          OUTPUT(2)=01100000B;  /* SPECIFIED EOI  */

             CALL INITAD(RATE,1,2,0);

          CHAR=ENABLED$INKEY;

DECIDE2:  IF CHAR<41H THEN GO TO NEXT$FRAME2;
          IF CHAR='F' THEN START=START+10;
          IF CHAR='G' THEN START=START+100;
          IF CHAR='H' THEN START=START+1000;
          IF CHAR='R' THEN START=START-10;
          IF CHAR='T' THEN START=START-100;
          IF CHAR='Y' THEN START=START-1000;
          IF CHAR='M' THEN GO TO OVER2;
          IF CHAR='Q' THEN GO TO BEGIN;
          IF START<0 THEN START=0;
          IF START>(25000-LENGTH) THEN START=25000-LENGTH;
          CALL FORMAT(START,1,1,0,7,@DUMMY);
          CALL PRINT(@DUMMY,7,7,1,5,0,1);
          IF CHAR='P' THEN GO TO PRINT$OUT;

NEXT$FRAME2:
           FINISH=START+LENGTH;
             OUTPUT(16H)=255;
             CALL TIME(1);
             OUTPUT(16H)=0;
             GO TO LOOP5;
```

```
    PRINT$OUT:
                    DO J=0 TO 3;
                         NUMBER=INT(P(J));
                         CALL FORMAT(NUMBER,1,1,0,5,@DUMMY);
                         CALL PRINT(@DUMMY,5,5,1,3,0,0);
                    END;

                      NUMBER=INT(P(4));
                      CALL FORMAT(NUMBER,1,1,0,5,@DUMMY);
                      CALL PRINT(@DUMMY,5,5,1,3,0,2);
                      GO TO NEXT$FRAME2;
END VOWEL;
```