Title

A high accuracy computer recognizer of speech for medical data input

Author

Parfitt, Richard Alan

Publication Date

1980

A HIGH ACCURACY COMPUTER RECOGNIZER OF SPEECH FOR MEDICAL DATA INPUT

by

Richard Alan Parfitt

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Medical Information Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA

San Francisco

[redacted]

ABSTRACT

The design, construction, and evaluation of a low-cost, isolated-phrase recognition system for use in Medical Information Systems is described in this dissertation. Low cost and high accuracy for an isolated-phrase recognition system are two prerequisites for making such systems useful for the types of medical data input problems described in Chapter VII. Previous efforts in this field are briefly summarized. A detailed examination of these previous efforts brought to light an important unanswered question. Of the two common approaches to speech recognition, it was not clear which method is best. The two approaches are the use of Linear Predictive Coding, LPC, and the use of Bandpass Filters.

Two theoretical models were built and tested to determine the merits of each approach. From the results of the modeling, the best method was chosen and designed into a system for low-cost implementation. From the design, a proto-type development system was built and tested. The results of these efforts have been at least partially successful, and open up new promising avenues for further research.

ACKNOWLEDGEMENTS

I would like to thank Dr. Ronald Henley, Dr. Scott Blois, and Dr. John Starkweather for their comments, critiques and review of this work.

I would like to express my special appreciation to Dr. George White of Auricle, Inc. for his patience and help with the difficult problems I had to solve in this work.

The assistance of Professor Bob Broderson from the Electrical Engineering Department of UC Berkeley helped make the proto-type construction possible for me. Professor John Ohala of the Linguistics Department at UC Berkeley is also to be thanked for allowing me the use of the Linguistics Lab.

Finally, my thanks are owed to Professor Raj Reddy of the Carnegie Mellon University for allowing me the use of the Speech Research facilities, while doing my theoretical modeling.

Financial support was provided by a traineeship from the National Library of Medicine, Grant number 2T15 LM07000.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

A. General

The last fifteen years have seen a gradual evolution in the sophistication and success of computers in medicine. A large number of useful and important medical information systems have been developed. This period has also led to a clear demonstration of the problems medical professionals have interacting with many of these information systems. The principal output device of the newer generations of medical information systems has been the Cathode-Ray-Tube*, CRT. Together with the CRT, a number of input devices have evolved. These include the touch screen, the light pen and the keyboard. While in many instances these devices work well, in others, especially those related to direct physician input, they have been at best poorly accepted. A more natural means for data entry is needed. The effort described here is the work that has been completed to develop a more natural means of data entry. Human speech is the means by which data is entered, but with some restric-

*See Glossary for definition of unfamiliar terms.

tions. The system that has been built accepts only isolated phrases, but it has a high accuracy of recognition, and is designed for low-cost implementation.

Since many groups with great experience and resources have worked for many years on the subject of continuous speech recognition with only partial success, it was decided not to attempt this approach even though it is very appealing from the viewpoint of an ideal interface. The recognition of isolated phrases by a user, known to the system, was felt to be a more productive area.

The ability of the computer to recognize isolated utterances must be consistent and accurate for the system to be acceptable. Throughout the work described here, the rate of recognition is used as a criterion for comparing different methods to solve the recognition problem. The final goal, to build a system with a high enough recognition rate to be acceptable by the user, has been achieved for a potentially useful vocabulary.

B. Significance

The experimental system described represents an original solution to several important problems. Two methods for doing speech recognition, Linear Predictive Coding, LPC, and Bandpass Filters, are compared. For the models used and the data tested, bandpass filters appear to be superior to

LPC for representing speech. A large number of very special bandpass filters are combined together with support electronics in a unique way to accurately capture speech information. The speech information thus provided makes the recognition task solvable with a low cost micro-processor. The system is highly accurate and with the use of a special bandpass filter the whole system is designed to be low-cost. The importance and use of such a system to handle medical data input is described in chapter VII.

C. Organization of Dissertation

The reader is referred to the Glossary for definitions of terms that are not familiar.

This chapter briefly describes the important aspects of this work.

The second chapter reviews related work done by others. A complete history of work in this field would be monumental. References to more comprehensive reviews of this field are included in the references of Chapter II.

The third chapter introduces an idealized model of a speech recognition system. The purpose of the model is to form a basis for understanding the complex parts that make up a real recognition system.

The fourth chapter gives a detailed description of two recognition systems that were modeled in order to determine the best approach for achieving the objectives of

this work, a low-cost, high-accuracy, speech recognition system. The modeling is used in helping to select between LPC and bandpass filters as two means for representing speech.

The fifth chapter describes the experimental system that was actually built. The components of the system are described. How the components interrelate is explained, and the importance of each component is discussed.

The sixth chapter gives the results of the data that was processed through the two models and the experimental system.

The seventh chapter describes one of several medical applications that could benefit from speech recognition.

The final chapter discusses the findings and conclusions that can be drawn from this work.

CHAPTER II

PRIOR RESEARCH IN THE FIELD

A. Overview of Speech Research Using Computers

Since the early 1950's word recognition by electronic devices has been an attractive area of research. Academic institutions, several corporations, and the Federal Government are currently spending several hundred thousand dollars a year on developing speech recognition systems. Some useful systems have been developed and some practical applications have been made. Nevertheless, speech recognition is still in its infancy.

Many potential applications for voice recognition by computer remain undeveloped or untested. New designs and new approaches are needed to take advantage of microelectronic advancements that are the supporting medium for automatic speech recognition.

This work is concerned with automatic speech recognition (ASR), but the books on speech synthesis and acoustic phonetics, such as those of Flanagan [1], Fant [2], and Lehiste [3] present an extensive coverage of speech, acoustics, and phonetics which provide a necessary basis for the work done in ASR.

ASR can be broken down into three distinct areas of

increasing complexity: isolated-phrase recognition, continuous speech recognition, and understanding systems. Each area will be discussed below. However, the main emphasis will be on isolated-phrase recognition. To date, only isolated phrase systems have been used in practice. The application of ASR being proposed will use an isolated phrase recognition system. Two excellent review articles exist which put more emphasis on continuous speech recognition [10], [11].

A large range of choices are available for man/machine communications: card decks, keyboards, CRT's, light pens, RAND tablets, joy-sticks, microphones, etc. A comprehensive study of different modes of communication tested the effect of ten communication modes on cooperative problem solving [17]. The teams consisted of two members, one a source of information and the other the seeker of information. Since computers are heavily used as sources of information, it is useful to know which mode of communication was most efficient for the seeker in order to obtain information from the source. The problems that were presented to the human teams were not unlike those that might be solved with the aid of a computer. The ten modes studied were: typewriting only, handwriting only, handwriting and typewriting, typewriting and video, handwriting and video, voice only, voice and typewriting, voice and handwriting, voice and video, and a combination of typewriting-video-voice-handwriting. All teams that were

allowed the use of voice with, or without the use of another means of communication, did better than those that were not allowed to use voice; no differences were found between teams that had access to voice and some other mode of communication. When a team had voice communication to use, they inevitably did better.

Other investigators have attempted to determine the advantages of voice input to the computer [4, 19, 20]. Physical mobility, naturalness of speech, and man's relatively extensive speech training are the most frequently mentioned advantages from the human standpoint. Martin [4] cites several applications where voice input has proven more effective than traditional means of data entry. Another important argument used for voice input is its greater bandwidth, or greater rate of entry, than other modes of input such as typing [20]. Even for a good typist, voice has been shown to be a faster means of inputing data. Time is saved by using voice and one can take advantage of the fact that most people are already proficient at using their voice for communication.

B. History of Isolated Phrase Recognition Systems

Isolated phrase recognition (IPR) systems are defined as those systems that require a pause between each spoken utterance. Martin [4] in his comprehensive review of IPR systems sets this pause at about 100-ms in duration.

The first publicly demonstrated attempt at IPR was

done by Davis, Biddulph and Balashek at Bell Laboratories [21] in 1952. Their system was capable of recognizing digits spoken over the telephone with a 97% accuracy. No attempt was made to recognize actual phrases. No rejection mode existed so that all utterances were best fitted against stored patterns and the best match was selected. The device had to be recalibrated, or new reference patterns generated for different speakers. The whole system was built out of a series of analog electronics that employed 28 condenser-resistor circuits to break up the incoming signal into segments of a changing voltage proportional to the signal frequency spectra. The electronics were unwieldy and recalibrating the system for different users was a formidable task.

Other similar systems were built in the 1950's [22]. Their important characteristics were: the use of cumbersome analog electronics, the need to hand tune for each speaker, and the high accuracy rates that were achieved. Most of this work was done at Bell Labs. The advent of computers in the early 1960's was to provide a solution to the problem of adapting to new speakers that was previously so difficult.

In 1960, Denes and Mathews [23], at Bell Labs used an IBM 704 for digit recognition. They first captured the acoustic signal on tape recorder. The signal was then digitized by using 17 band-pass filters sampled at 70 times per second. The filtered amplitudes were converted into a ten bit binary number. Using magnetic tape for input, the

collected data was then analyzed by computer using the greatest cross-correlation coefficient, when the data was compared to ten stored patterns. The system did not use analog electronics that needed hand tuning but, reference patterns which were stored internally. Several reference patterns for the same word were averaged together to improve accuracy. The use of digital computers made other innovations possible such as time normalization. The time normalization process involved determining the beginning and end of the utterance. The samples for the normalized word would then be either compressed or expanded to some standard length before cross-correlations with the stored patterns. The system represents a significant advancement, since the use of a digital computer allows a much greater choice in methods on how to analyze the input. A rejection of highly uncorrelated utterances was now possible, but the system no longer ran in real time. A new problem arose. The cumbersome analog electronics had now been replaced by a very expensive general purpose computer that took large amounts of time to recognize one word.

In 1962 Bakis and Sholtz at IBM Research Center, Yorktown Heights, presented their work on the computer recognition of spoken digits using vowel-consonant segmentation [24]. Their system used a tape recorder to collect the acoustical signal. The tape was then sampled at a rate of 16000 samples/second. the amplitude of the signal was converted into a 6 digit binary number. Using a

computer, the waveform was transformed into its power spectrum by means of a simulated set of 40 filters, seperated by 200Hz. The recognition process was then started by segmenting the input phrase into vowels and consonants using such properties as: pauses, variation in amplitude, amplitude thresholds and duration. Once the phrase was converted into a sequence of segments it was compared against each entry in a dictionary until an exact match was found, or the phrase was rejected. The dictionary was constructed from the utterances of ten speakers and then optimized by adding the utterances of 40 more speakers. Half the speakers were male and the other half were female. The system was then tested against the same utterances that were used to construct the dictionary. A 97% success rate was reported. This paper begins to demonstrate the power of using a digital computer to develop sophisticated methods for segmenting a phrase, and to simulate analog band-pass filters. However, it was a simulation that required large amounts of computation.

One of the more recent efforts at recognizing spoken digits was done by Keller at Bell Labs [25]. An on-line computer is used to gain the flexibility of a computer program and a near real-time response previously associated with hardware implementations. The first two formants were tracked and the input utterance was segmented by method of articulation: voiced fricative, unvoiced fricative, plosive, silence, nasal, and vowel like. The formants are frequency

bands of higher energy that are shaped and determined by the human vocal tract. As the tongue and the lips move around, the formants change too. The segmentation technique by method of articulation was intended to eliminate some of the difficulty associated with the segmentation of phonemes which have a less well defined structure. Bayes procedure is used to select a best match. Error rates were reported to be between 95% and 98.8% depending on speaker. The reference patterns were generated from a group of four male speakers and the system was able to perform well for any one of the four speakers.

This brings us to the use of mini-computers in real-time systems. These latest systems will be discussed in section E under Current Isolated Phrase Recognition Systems.

C. Continuous Speech Recognition

The principal focus of this research is in the area of IPR systems. But, a discussion of both continuous speech recognition and understanding systems is included in order to view IPR in perspective to the entire field and to give the interested reader some view of the relative progress, accuracy, and cost of the respective approaches.

Beginning in the early seventies, several continuous speech recognition systems, CSR, were demonstrated by ARPA contractors and IBM. Important aspects of several of these systems will be discussed.

Word segmentation and internal word representation are much more of a problem in CSR than in Isolated Word

Recognition. The acoustic characteristics of sounds and words exhibit much greater variability in connected speech. The isolated word technique of storing a complete template for all possible utterances is no longer useable. Even a ten word vocabulary of the digits requires the storage of ten million reference patterns to recognize all possible seven digit numbers. Analysis of sub-parts of utterances is a necessity. The internal representation, of words as elementary units, also becomes insufficient because of the large variation in pronounciation of words in CSR. This has led to the use of phones, phonemes or syllables to represent words. Thus the speech signal needs to be segmented into discrete, acoustically invariant parts. This causes many difficult problems. For example, in the phrase "some milk" the nasal /m/ might be matched with the end of "some."

Before human speech can be recognized, it must be converted into a form that can be manipulated. The acoustic signal is transformed into a symbol string by a front end processor. This is an active area of research, so there are many approaches to this problem. The Harpy System, which evolved out ot the Dragon system [27], at Carnegie Mellon takes speech samples at 10ms time intervals. The samples consist of a vector of twelve amplitude and zero crossing parameters. A probability is computed for each of 50 to 100 possible phonemic symbols. Several vectors may be combined to make up a phonemic symbol because of allophonic variation. The Hearsay-II System, [28], also at Carnegie

Mellon uses a similar technique for phonemic label assignment. However, the front end processor goes on to deduce syllable-like segments from the phonemes by using local maxima and minima from the amplitude function of the utterance. These larger segments are given gross feature labels such as silence, fricative and voiced. The IBM System, [30], at the Watson Research Center uses a complex pattern recognition technique developed by Dixon and Silverman, [31, 32]. Complex decision rules and dynamic segmentaion are used. The energy, spectra, spectral change and an ordered list of five "most similar" classes and their similarity values are used for the segmentation process. Sixty-two label classes are used for a prototype matching of the segments.

Another problem for CSR systems is how to represent knowledge. General types of knowledge are important in CSR systems. Three of them are phonological rules, lexicon, and syntax.

In the Harpy system the various sources of knowledge are organized into a hierarchy of probabilistic functions of Markov processes. These sources of knowledge are the templates representing utterances and the information about how these utterances can be combined to form meaningful phrases. A network is constructed to provide an integrated representation of the hierarchy. Recognition of an utterance corresponds to finding an optimum path through the network.

The Hearsay-II system organizes knowledge into

independent and cooperating knowledge processes, for easy addition or removal of a knowledge source. Representation of knowledge within each process is arbitrary. Syntax is represented as a set of productions (generative rewriting rules), and antiproductions (analytic prediction rules). The lexicon contains only the phonemic base forms. The phonological information is embedded in various acoustic analysis procedures.

The IBM system uses a finite-state grammar and a directed graph representation of each lexical element. Phonological rules are compiled into the lexicon. Extensive use is made of statistical information to provide transition probabilities within the finite-state networks, representing task-dependent information.

A third problem for CSR systems is matching the input string against stored templates and controlling the progress of the search.

The Harpy system searches for all possible paths in parallel. The Markovian assumption permits it to collapse many alternative sequences into a single state, thus avoiding exponential growth. Alternate paths are recombined at exactly the same rate that new branches are formed.

The other two systems use a stack (or a set) containing a list of alternative word sequences (or state sequences), arranged in descending order of their likelihoods to represent the partial sentence analysis so far. Given a word sequence with the highest likelihood, the task-specific knowledge generates all the words that can follow that sequence. Each possible new word is matched against the unmatched symbol (phonemic) string to estimate conditional likelihoods of occurence. A new list of acceptable word sequences and their likelihoods is generated. The process is repeated until the whole utterance is analyzed.

The Hearsay-II system uses a heuristic technique for generating likelihood estimates while Harpy and IBM use the principles of stochastic modeling.

The performance of CSR varies from system to system. The IBM system achieved an 81% sentence recognition accuracy, but, tests were run with a single speaker in a noise free environment. Hearsay-II performed the poorest with only 31% accuracy, but tests were made with multiple speakers. The Harpy system obtained an 88% accuracy when run against the same data set as the Hearsay system. The Harpy is more accurate, but runs four to five times slower.

Certain areas of CSR could be improved. Better searching, better matching, and better segmentation and labeling are all essential. A parallel search strategy is being tried at Carnegie Mellon University, where several promising alternative paths are considered in parallel. Word matching and verification accuracies need to approach those of word recognition systems, i.e. greater than 99%.

D. Understanding Systems

Speech Understanding Systems, SUS, are still in their

infancy. Most of the work in this area has been supported by grants from ARPA. The distinguishing feature of SUS's is the attempt to use semantic information about the topic of discourse in order to improve the recognition algorithms. The use of semantic information is important in overcoming the problems of recognizing grammatically incorrect utterances. Speech like noise can also be removed if it makes no semantic sense. The recognition process can now work in noisy environments with poorly pronounced words.

The left to right analysis of speech used by CSR systems is more difficult, since an unknown error or babble may be contained in the middle of an utterance. SU requires semantic knowledge sources to disambiguate the acoustic signal. Obviously, the ideal system organization for an SUS to arrange knowledge processes, is not clear.

The VDMS system, [37,38], developed jointly by the System Development Corporation and Stanford Research Institute is organized around a parser that directs the search strategy. A best-first approach is taken, but a strict left to right organization is not necessary. The SPEECHLIS project, [36], developed at Bolt Beranek and Newman used humans to solve the recognition task first. Different parts of the task were assigned to different people communicating by teletypes from different rooms. Through incremental simulation the human problem solvers were replaced one by one. The Hearsay-II system, [34,35], developed at Carnegie Mellon uses a common data link or

blackboard model to share information from different knowledge sources, (KS's). A KS is a small piece of represented knowledge such as a digitally encoded speech waveform. When a KS generates a hypothesis it is shared by means of the blackboard for testing. The intent was a direct attempt to use the artificial intelligence paradigm of hypothesis-and-test. The KS's as in Hearsay-I are independent from one another and treated uniformally. Therefore easy updating of KS's is possible. The KS's operate in an asynchronous manner while sharing hypotheses and results with other KS's. A focus-of-attention KS directs the choice of efforts and terminates after a sufficient amount of time has transpired, or highly successful results have been obtained and no promising hypotheses are left to test.

E. Current Isolated Pharase Recognition Systems

Five isolated phrase recognition systems are discussed. Each system is currently commercially available. However, one or more problems make the system unuseable for the type of medical data input problems described in chapter VII. The systems are described here to bring the work presented in this dissertation into perspective.

Heuristics Technology

The Heuristic Speech System, [39], is currently the least expensive system available, and it has the fewest number of capabilities. The system consists of a low cost audio microphone and a single plated contact board that contains the analog to digital conversion electronics. A host micro computer is required to perform the template matching process.

The microphone preamp has the following distortion properties: gain increases at a rate of 6dB/octave up to 5kHz, from 5kHz to 16kHz the gain of the amplifier is constant, and the gain drops off beyond 16kHz at 6dB/octave. A second amplifier is therefore used to amplify up to 5kHz at 6dB/octave. The output of the 2nd amplifier is available to the host computer for template matching. The amplifier output is also fed into a zero-crossing detector, the output of which is also available for use by the host computer. The microphone preamp signal is also fed into three bandpass filters which roughly cover the first three formants:

150Hz to 900Hz

900Hz to 2200Hz

2200Hz to 5000Hz

Before going to the host computer, all signals are fed into a 8-to-1 analog multiplexer where they are selected individually by the host computer. The output of the 8-to-1 multiplexer leads to a second 2-to-1 multiplexer where a compression amplifier boosts low amplitude signals, and compresses high amplitude signals. Both the compressed and the uncompressed outputs of the 8-to-1 multiplexer are available to the host computer after going through a 6-bit A/D converter. The A/D output is available to the host

computer by way of the S-100 bus and an eight bit I/O port.

Some 8080 assembly language software is supplied with the Heurisitcs system for running the host computer. Commands to the speech board may also be given through a BASIC interpreter. Limited experiments by the author indicate that assembly language routines supplied will recognize the spoken digits with an 80% accuracy in a noise free environment. The vocabulary is easily changeable. Sixty-four bytes of storage are required to save each word template for the matching algorithm. A sixteen to thirty-two word vocabulary is possible. The software supplied is rudimentary and improvement in performance can be achieved by the use of more complex software.

Phonics Inc.

The Phonics system is another extremely low cost system that takes advantage of low cost LSI. The system may run in one of two modes, stand-alone or with a host computer. In the stand-alone mode the system can recognize up to sixteen words with a claimed accuracy of 95% in a noise free environment. In the host mode, the system communicates through an eight bit parallel I/O port with the host. The system may be down-line loaded from the host computer to extend its vocabulary size or it may be loaded with previously generated training samples. 32 bytes of storage are required to save each reference pattern.

The recognition algorithm is implemented on a Motorola M6800 microprocessor. The recognition process uses

amplitude and time normalization. Training samples are generated by parametrically saving the whole phrase instead of using a phonetic approach. The system uses eight bandpass filters and eight bits of amplitude information. The exact details of the recognition algorithm are proprietary. The system retails for about $600. No field support is provided and the user must develop all the application software. Limited experiments by the author indicate that in the case of the spoken digits, an 80% accuracy is obtained.

Threshold Technology

Threshold is one of the oldest companies currently producing commercial systems. The company was started by Tom Martin after he completed his thesis work on isolated word recognition. The system uses a hardware preprocessor to do the feature extraction. Thirty-two speech parameters are used including vowels, consonants and phonetic features. All features are time normalized. A great deal of effort has been put into removing extraneous noise from the input. Many of the techniques are considered proprietary and have not been published. The system compensates for speakers with head colds, and hoarseness by making use of permanent features of the speaker's voice.

The system runs on a DEC LSI-11 and can handle 100 words or more. Every user must train the system by repeating each word in the vocabulary ten times. Typically, a host computer is used to process the captured data. A 99% recognition rate in noisy environments is claimed. Several

major applications for use have been developed by Threshold [4]. The major drawback for the system is the cost which is in excess of ten thousand dollars for each system installed.

Interstate Electronics

The original system was developed by Scope Electronics. Interstate later purchased Scope, after work on their own micro-processor based system utilizing fourier analysis, progressed more slowly than desired. The system is somewhat similar to Threshold Technology's. Sixteen bandpass filters are used to extract the energy-frequency distribution of each utterance. The utterance is time normalized and a 240-bit pattern is extracted for comparison against reference templates.

The system runs on a Data General Nova 3 and supports up to four user's. A 250-word to 900-word vocabulary can be handled depending on the number of users. The manufacturer claims a 98-99% recognition rate for a fifty word vocabulary. Each user is provided with a 40-character alpha-numeric display for utterance verification. The basic system cost $125,000., but includes two CRTs, a printer, a disk drive, and the host computer. The availability of the host computer for other tasks besides speech recognition, is not given in the published descriptions. The current system is still under development and is only a recent acquisition for Interstate.

Nippon Electric Voice Data-Input Terminal

The system has just recently been anounced by Nippon

Electric in the April issue of Electronics. The system is based around the 2901 4-bit-slice bipolar micro processor in a 16-bit configuration. Another bipolar processor coordinates system data flow, while an 8080 micro processor is dedicated to handling I/O. Details of the system operation are not available and marketing will intially be limited to Japan. The system appears to have similar capabilities to Interstate and Threshold, but represents a significant departure in strategy by using micro processors. The system cost $60,000 to $80,000.

CHAPTER III

MODEL OF A SPEECH RECOGNITION SYSTEM

A. Integrated System

For convenience and conceptual clarity the model described here has been broken into several parts. Figure III-1 is a block diagram of the parts that make-up this model. At the simplest level, a speech recognition system monitors the environment for waves consisting of mechanical disturbances having frequencies between about 120 and 6,000 Hertz, Hz. A more comprehensive system might monitor visual and other sensory cues. This model is limited to acoustic monitoring only. The recognition process occurs by labeling classes of wave patterns with a particular classifier, or a set of possible classifiers. For the research described here, classifiers consist of names for wave patterns. Each name may have many wave patterns associated with it, while each wave pattern may be matched to several names with different probabilities. However, each wave pattern has only one correct name associated with it. The measure of accuracy for a speech recognition system is how often it associates the right name with each wave pattern. One important class of wave patterns is called noise. A useful speech recognizer must be able to discriminate between noise and the wave

MAJOR COMPONENTS OF A SPEECH RECOGNITION SYSTEM

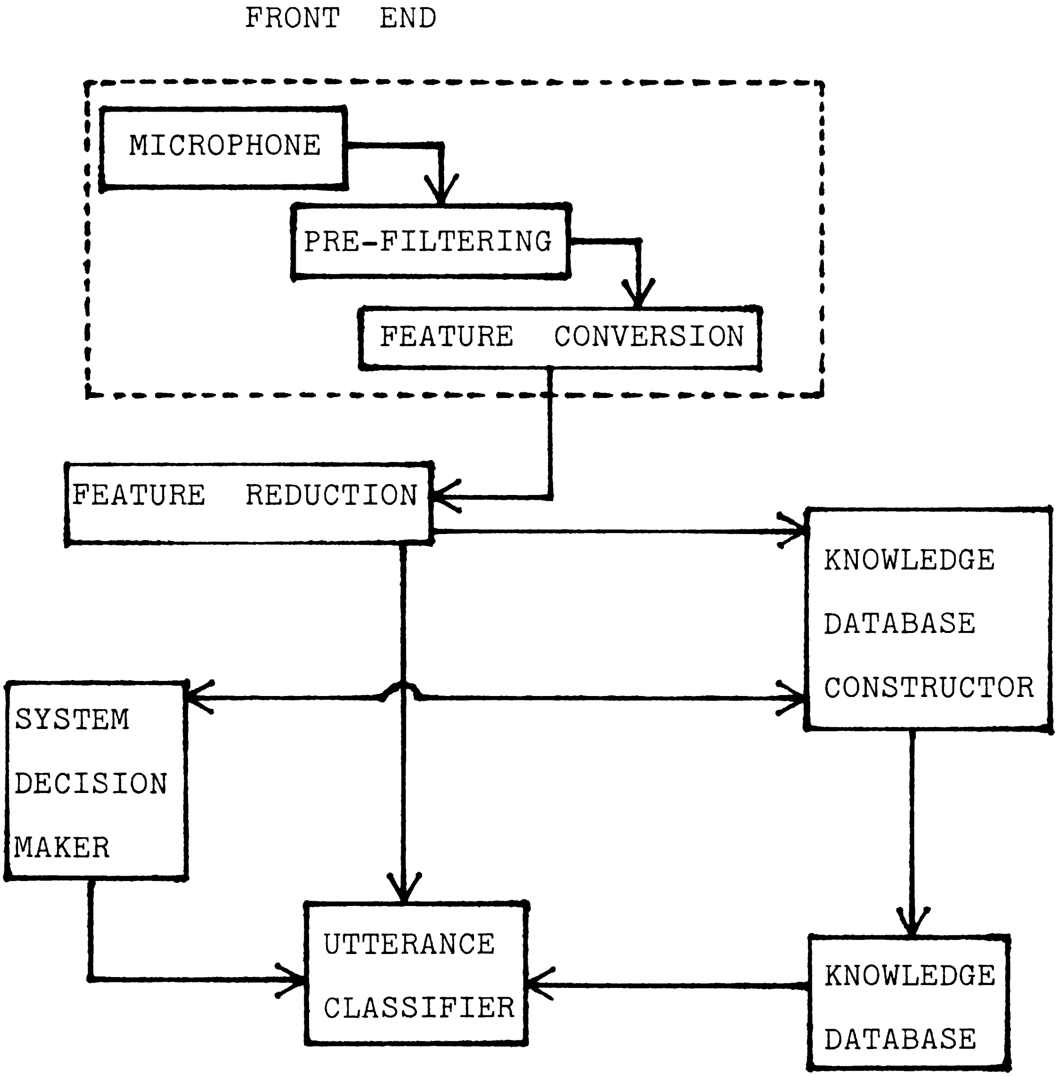
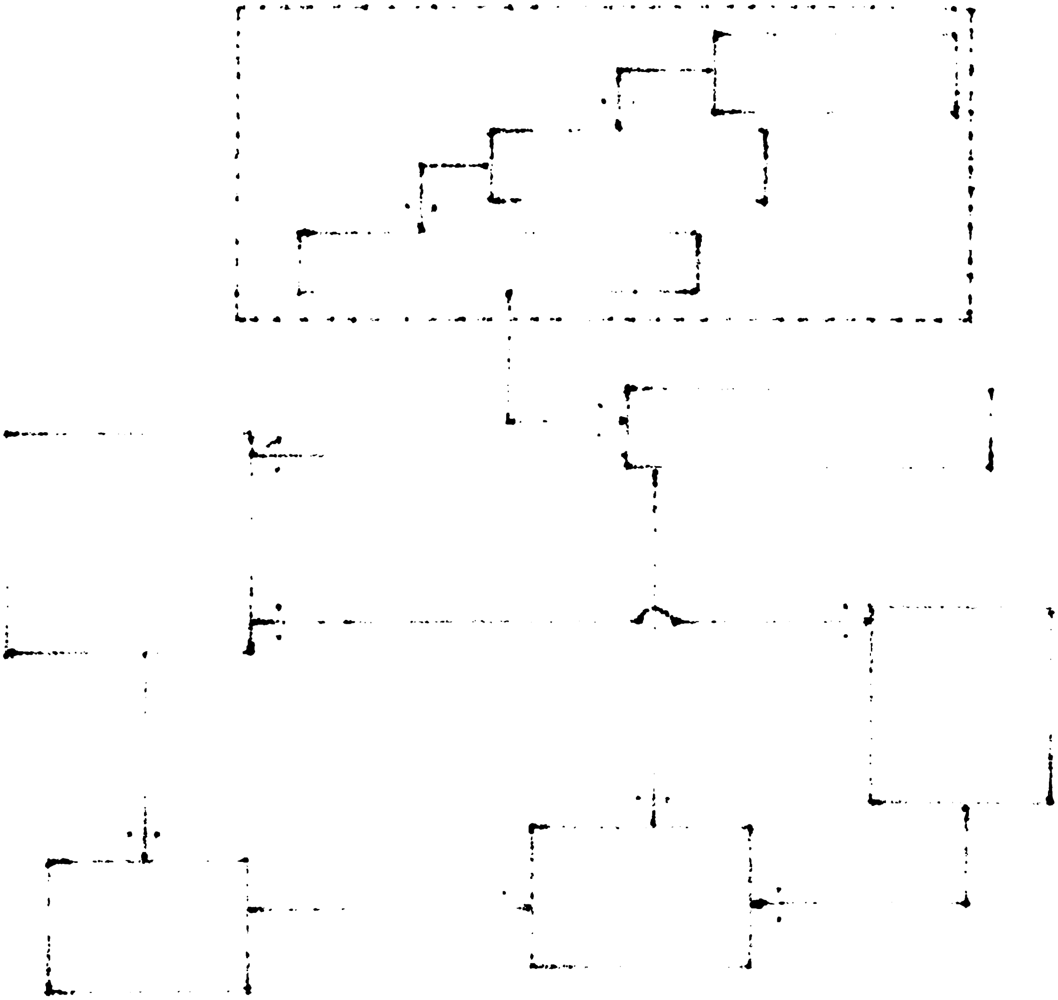


FIGURE III-1



patterns which have a useful purpose.

The ultimate performance of any recognition system is dependent upon the limitations of each part in the system. An important problem in modeling such a system is the correct specification of the limits for each component. Only then can different techniques for performing the same sub-function be compared. The basis for comparison is the ability of each technique to approach the theoretical limit for perfect preservation of information. The preservation of information is analogous to fidelity, the exactness of reproductive detail. Because perfect fidelity is so difficult to achieve, the actual basis for choosing the optimum technique for each sub-function is dependent on its cost/performance relationship. The system with the lowest cost for the required level of performance is chosen.

As an example of the difficulty in building a sub-function that approaches its theoretical limit for accuracy, the knowledge data base for the classifier can be considered. The classifier tries to place the unknown utterance into one of many sets of utterances. Each set of utterances corresponds to a unique name, i.e. the word "one" is a unique name for a large set of utterances. But, it is very unlikely that a new unknown utterance that has been preserved with perfect fidelity will match exactly with any of the utterances in the appropriate set. The exact definition of the classifier and knowledge data base used in this model will be discussed in another section.

B. System Front End

The purpose of the system front end is to convert the mechanical air disturbances into its characteristic features. The ideal front end would convert each different sound into a unique set of features. One possible way to characterize a sound is to convert the mechanical disturbances into an energy vs. frequency continuum over the 120 Hz to 6,000 Hz domain. Each time the features that represent the sound wave change significantly, a new set of features must be saved. The sound wave must be sampled at discrete time intervals. The time intervals must be short enough so that any change in the sound wave that represents useful information is captured. The smallest sound units are called phonemes. Each phoneme is a useful piece of information. If the sound wave is sampled every ten milliseconds, even the shortest phonemes can be captured and identified [1]. For a given time interval, the features should be compact, yet still retain all the information orginally in the speech signal. Each of the important steps taken in the feature conversion process are described below.

1. Microphone

The first element of a recognizer is the microphone. This element can be thought of as a function that converts sound waves into an analog form, a time-varying voltage. The best solution to this function is a device that converts

only the desired sound source and has a response over the appropriate energy-frequency ranges. A zero drop-off outside the desired frequency range would be ideal. The development of such a microphone is not the subject of concern here. A description of the actual microphone used with the experimental system can be found in Appendix F. For modeling purposes, a microphone with ideal characteristics is assumed. The data used in testing different solutions to sub-functions was generated in a low noise environment with a high-quality, audio recording system.

2. Pre-Filtering

Two important tasks are accomplished by this sub-function: rejection of those parts of the signal that are introduced by a non-perfect microphone, and enhancement of those parts of the signal that will later be difficult to distinguish. Pre-filtering usually includes some kind of high frequency and low frequency cut-off. The microphone signal is amplified and the high frequencies are increased in relative energy. The pre-emphasis of high frequencies is necessary because of the lower energy at high frequency in spoken language. This makes differences in energy concentration at two high frequencies easier to distinguish. This problem is well understood [1], and no new techniques are presented for this part of the system.

3. Feature Conversion

A large number of papers on speech recognition have appeared in the literature. Most of these papers tend to treat two distinct processes as one, feature extraction. In this model feature extraction is broken into two parts: feature conversion and feature reduction. The purpose of feature conversion is to extract the identifying features out of the incoming signal that are necessary to prevent losing any information. The purpose of feature reduction is to remove redundant information. Feature reduction is discussed further in section C.

The goal of feature conversion is to translate the incoming signal into the minimum number of features necessary to preserve all information contained in the signal. The signal will ultimately be represented as a set of discrete parameters. The features generated by the feature conversion process are taken at discrete time intervals. It is the information in that discrete time interval that must be preserved. The difficulty of this task is attested to by the use of approximation techniques. The three most common classes of techniques for solving this problem are bandpass filtering, fourier analysis and linear predictive coding. Variations on each of these techniques exist, but all are essentially reducible to one of the three methods.

Each method is inherently limited. Bandpass filters are used to partition the incoming signal into frequency bands. The energy in each band then gives a rough

approximation to the energy vs. frequency distribution. The larger the number of bands, the more accurate the approximation is. The solution to how many bands to use has typically been empirical. The number of bands is increased until an acceptable recognition rate is achieved for a specific vocabulary. The limitations of LPC, and Fourier analysis is analogous to bandpass filters. These limitations are modeled mathematically in the next section for bandpass filters and LPC.

C. Feature Reduction

The purpose of this function is to reduce the features to those sufficient for making the utterance classification. The importance of this function is the way it minimizes the complexity of the utterance classifier. The benefits of reducing the number of features are: a decrease in system run time, a savings in storage space, and an increase in possible vocabulary size. Reducing the number of features can result in lower system identification accuracy. The important criteria used for measuring ideal performance for this function is the ability to provide adequate features for placing utterances in their appropriate classes. This is dependent on the types of classes chosen. If the classes are very similar, then more features must be saved in order to distinguish the classes.

Many methods for feature reduction exist. One of the simplest is to compress the speech in time. The assumption

is that human speech is relatively stable over short periods of time, so only one sample is needed from each period of stability. A more sophisticated method to reduce the number of features is to look for redundancy. Again, the speech is compressed in time, but the compression is based on a test for redundancy instead of assumed redundancy. Two adjacent time periods are combined if they are similar, otherwise they are kept separate.

D. Database Constructor

This function is potentially much more complex than any of the other functions described so far. The complexity of the Constructor determines the complexity of the database which in turn defines the complexity of the utterance classifier.

The simplest knowledge database would contain only a simple list of acoustic information from the feature conversion process. Each item in the list would represent one utterance. From here, the possibilities are enormous. Each item in the list might represent the average of several utterances. Each item in the list might be sub-divided, where each sub-section has an associated weight. The list might be organized into a hierarchal tree. The knowledge database might include information about previous utterances for use in conjunction with syntactical information. Prosodics, semantics and pragmatics are just a few other types of information that might be built into a knowledge

database. How to represent complex types of information such as prosodics is an active area of research that falls into the category of Speech Understanding Systems reviewed in Chapter II. For isolated-phrase recognition systems, the context and useage of the vocabulary is so limited that information about prosodics, semantics and pragmatics is not necessary.

The acoustic signal being analyzed has a well defined range of information that it can contain. The information can be reduced to managable representation due to the well understood characteristics of the human speech production. The physical production of human speech limits the types of information the acoustic signal can contain. It is not possible to make all the sounds of a violin with the human vocal tract. The classification of one sound into only one of many possible choices is a problem that can require many types of information. The information needed for making the correct choice is not limited by physical systems. The approach that is used here is to impose artificial restrictions. The number of utterances is limited and the difference between utterances is well defined.

E. Utterance Classification

Operation of the utterance classifier is dependent on information in the knowledge database and the information

provided by the feature converter. The performance of this function can be greatly enhanced by information embedded in the definition of the classification function. A variety of classification functions exist. The simplest might be the computation of a Euclidian distance measure between the unknown utterance and a list of reference utterances in the knowledge data base. A more complex approach is to allow a dynamic time alignment of the unknown utterance against the reference utterances before calculating the distance measure. Dynamic time alignment is a type of embedded information. It is a type of information that says the unknown utterance may be distorted in time relative to the stored utterance that it is being matched against.

For the models described in Chapter IV and for the experimental system described in Chapter V, a very simple utterance classification is used. The unknown utterance is compared against a list of stored templates. The type of distance measure used in the comparison and the type of time alignment used is described in the respective chapters. The exact method for building the templates is also described.

CHAPTER IV

DESCRIPTION OF RECOGNITION SYSTEMS MODELED

A. Data Collection

Data was collected from eight native speakers of the English language. Four of the speakers, two males and two females, were from the California area. The other four speakers, two males and two females, were from the Pittsburgh, Pennsylvania area and spoke with a slight regional accent.

To critically compare the two models decribed below, a difficult, but useful vocabulary was chosen. The vocabulary consists of thirty-six words, the phonetic pronunciation of the alphabet and the digits zero through nine. This is a difficult vocabulary for even human listeners because of such similar sounding utterances as "b" - "v" and "m" - "n", etc.

Each speaker repeated the complete vocabulary ten times. The utterances were arranged in random order to avoid the effect the next utterance on the list, or the previous utterance on the list might have on pronunciation. The effect is still there, but in a random fashion. The four Californians repeated the 36 utterances ten times in one sitting. The other four speakers repeated the 36 utterances two times each day over a period of five days. The purpose

of spreading the data collection over five days was to allow for variation in speaking habits over time. The data was captured by a Shure SM10 close fitting microphone. See Appendix F for more details on this microphone. The data was saved on Ferri-Chrome cassette tapes using Dolby noise reduction. A Sonny model TC-D5 cassette tape recorder and a Sanyo model RD 5340 cassette tape recorder were used. The two tape recorders gave similar results.

To get the data into a computer, the analog tapes were played into a sixteen bit analog to digital converter at Carnegie Mellon University. The data was sampled at 10 KHz, thus preserving frequncy information up to 5 KHz. Almost all of the information in human speech is in the frequencies below 5 KHz [2].

B. Feature Extraction

Two methods for extracting the features out of the raw speech wave form are linear predictive coding, LPC, and bandpass filters. Each method is described separately.

1. Linear Predictive Coding

The Durbin recursive method for computing the LPC coefficients is used [33]. Fourteen LPC coefficients are computed at each sample point. Later, when two sets of LPC coefficients are compared, the Itakura distance metric is used [8]. The Itakura coefficients are computed from the LPC coefficients and saved in a file at the same time.

The coefficients are computed over a twenty millisecond window. The window is shaped by fitting a Hamming function over it. This minimizes the effects of end-point distortion. A Hamming function is approximately equivalent to SIN(x) when x varies from zero to pi. The net results are to force the end points of the window toward zero. This helps compensate for the fact that LPC theory works best on an infinite periodic wave form. Small numbers at the beginning of the window help to stabilize the LPC calculation. Before the LPC coefficients are actually calculated, the raw wave form is digitally weighted to increase the effective amplitude of the high frequency components of speech. Such selective amplification of the high frequency components of speech is called pre-emphasis. The higher frequencies of speech contain useful information, but the energy of the high frequency components is too low to make this information useful until pre-emphasis is performed. The window is advanced by ten milliseconds before computing the next set of coefficients. Most of the English speech sounds, especially vowels, last longer than ten milliseconds. However, some phonemes, such as voiced stops, last only ten milliseconds. Thus, a ten milliseconds sample rate was chosen.

2. Bandpass Filters

The same windowing and pre-emphais used in the the LPC calculations were performed before the bandpass filter model was used to do the feature extraction.

To transform the raw wave form into the energy frequency domain a discrete fast fourier transform was computed. A 256 point FFT algorithm was used. Since there are only 200 points in each window, the last 56 points were filled with zeros. For each filter band a function was computed; each function conforms to the shape of one filter band in the experimental system. This function was then fit over the 128 discrete energies from the FFT calculation to compute the approximate energy for the filter band. The center frequency of each filter band is given in Table IV-1. Seventeen of the bands are 1/3 octave apart with a 55 dB per octave roll off. The eighteenth filter is 1-octave wide with 55 dB per octave roll-off. The model has only eighteen filters. The experimental system has one additional 1/3-octave filter with a center frequency of 5.64 KHz. To get the final parametric representation for each time slice another transformation is computed. The eighteen filter energies are converted into seventeen log ratios. The absolute value of each log ratio becomes one parameter. Where P(1) is the first parameter, F(1) is the filter energy with the lowest center frequency and F(2) is the filter energy with the next lowest center frequency, P(1) is computed by taking the absolute value of the log of F(1) minus the log of F(2). The reason for taking the log ratios of the energies from each filter is to make the frequency information partially immune to changes in amplitude. An eighteenth parameter is computed by taking the log of the

MODELED FILTER BANK

18 - BANDPASS FILTERS

17 - 1/3 OCTAVE FILTERS

F1	114 Hz
F2	143 Hz
F3	179 Hz
F4	230 Hz
F5	287 Hz
F6	358 Hz
F7	458 Hz
F8	573 Hz
F9	717 Hz
F10	917 Hz
F11	1.15 KHz
F12	1.44 KHz
F13	1.83 KHz
F14	2.29 KHz
F15	2.87 KHz
F16	3.67 KHz
F17	4.59 KHz

ONE OCTAVE FILTER

4.0 KHz

TABLE IV - 1

C. Database Construction

For each model, a database of thirty-six items was constructed. For each speaker, the first set of thirty-six utterances was chosen to make-up the database. The database for each speaker was then used for utterance classification of the other nine sets of thirty-six words.

Each item in the database represents one utterance. No form of feature reduction was performed on the LPC data, or on the filter bank data. A crude end-point analysis was performed to separate the speech out from the surrounding silence. When the total energy for a given window exceeded a set threshold it was assumed that speech was present. The end of the utterance was determined in a similar manner. A period of fifteen windows, or 150 milliseconds of silence was required to indicate the end of an utterance. The total energy in each window had to be less than the threshold. The threshold was set at four times the total energy during a window of silence. The lengths of items in the database varied between .3 seconds of speech and 1 second of speech, depending on the utterance and the rate of speaking for each subject.

D. Utterance Classification

To classify an unknown utterance, the utterance is compared against each item in the data base. An exact match is never made unless the item in the database and the unknown utterance are the same. For each item in the database, a similarity score is computed. The item with the lowest score is chosen as the best match.

The length of an item in the database and the length of an unknown utterance are rarely the same. That is because even the same speaker does not repeat a given word at the same rate. The problem when matching an unknown utterance to an item in the database is to try and match them up in time. The simplest method is to depend on the crude end-point analysis already mentioned, and to start comparing the two items time slice by time slice. When the end of the shortest item is reached, the comparison is completed. Another method is to do a linear time alignment. The shortest utterance is stretched until it is the same length as the longer utterance.

The method used in this work uses a dynamic programming algorithm to perform stretching as needed to minimize the final score. A more comprehensive discussion of dynamic programming can be found in the paper by Sako & Chiba [39].

Both models used the same dynamic programming algorithm for utterance classification. However, in the LPC model when two time slices are compared the Itakura distance metric is used. In the filter bank model a simple Euclidian distance measure is used, the sum is taken of the absolute values of the differences of each parameter.

How the two models compare when tested with speech

CHAPTER V

DESIGN & IMPLEMENTATION OF EXPERIMENTAL SYSTEM

A. System Overview

To test the ideas being proposed here, a prototype system was designed and tested. A series of experimental data was run through the system to determine its performance capabilities. Since the system is intended primarily for evaluation, several added features are included.

A dual drive floppy disk is used for data storage and to aid in program development. The CPM operating system and a compiled FORTRAN are the main software development tools. CPM is a registered trademark of the Digital Research Company. The CPM operating system is a single-user, single-job operating system. Several utilities and commands make-up the system. A text editor, an assembler, and a file manipulation program are the main utilities. The system is essentially a small subset of the Digital Equipment Corporation's Top's Ten operating system for the PDP-10. An external amplifier is used to control the microphone/tape-recorder input levels. A speaker and headphones are used to monitor the input data.

The major hardware components are shown in the

BLOCK DIAGRAM OF RECOGNITION SYSTEM

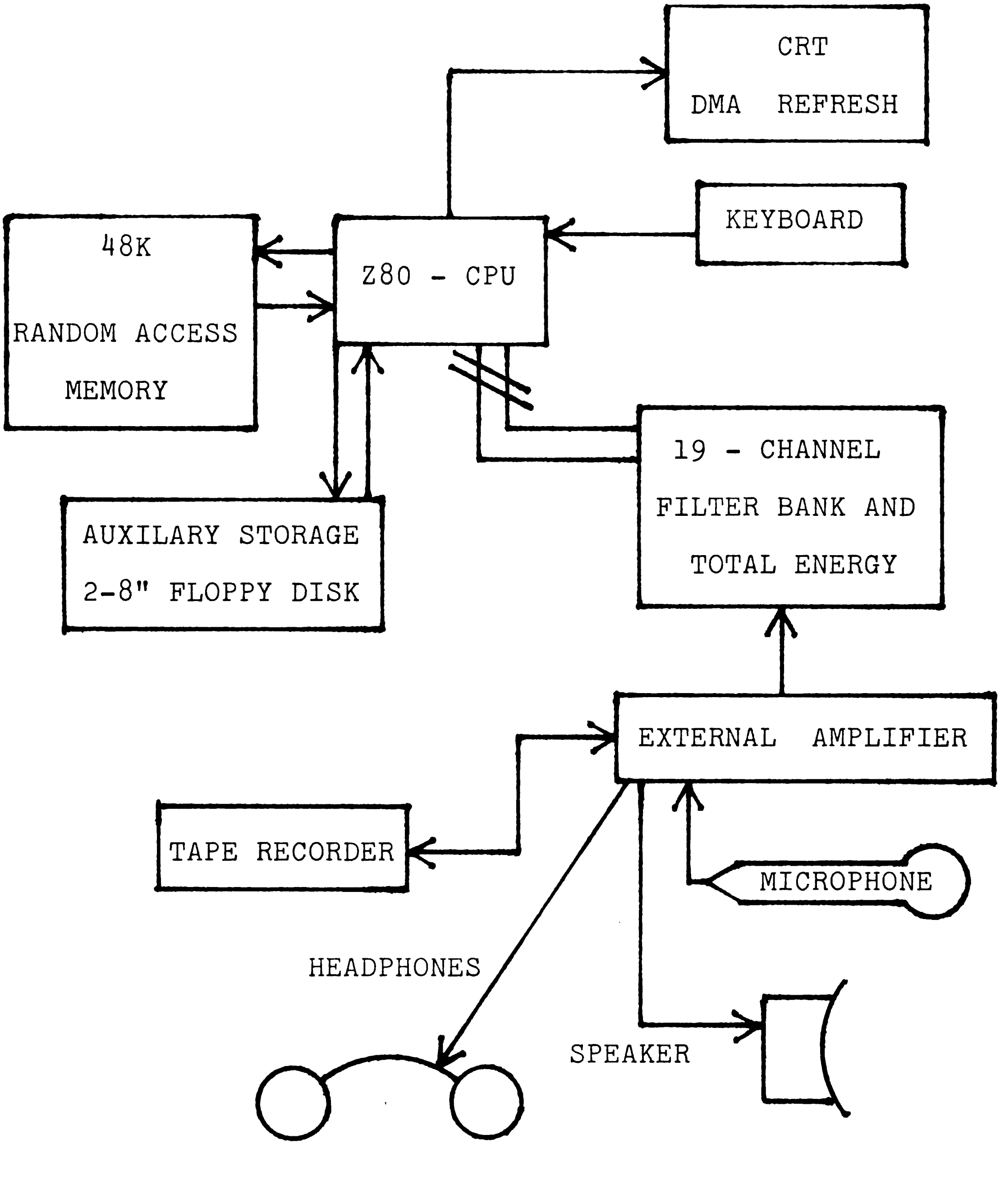


DIAGRAM V-1

A Z80 micro-processor with 48K of memory is used to perform the recognition task. The basic system is built around the S-100 bus. Standards for the S-100 bus have been defined by the Institute of Electrical and Electronic Engineers. The CRT is refreshed from 1K of RAM that is writable by the Z80 computer. The display has a capacity for 16 lines of text with 64 characters per line.

For auxilary storage, two 8" floppy disk are interfaced to the Z80 by a DMA disk controller. The 19-channel filter bank is wire-wrapped on two S100 proto-type boards. For direct data input, a Shure SM10 microphone is used. Most of the experimental data is pre-recorded onto Ferri-Chrome cassette tapes and then played back on a Sanyo model RD 5340 cassette tape recorder.

B. Filter Bank configuration

A complete schematic of the filter bank hardware is provided in Appendix A. To start with, the speech is first captured by a Shure SM10 microphone. The analog representation, a time varying voltage proportional to amplitude, is then either recorded onto cassette tape or routed through an external amplifier, before going to the filter bank hardware.

The output of the external amplifier which is adjustable, is fed into an active filter for pre-emphases. The signal is pre-emphasized at 6db/octave starting at 500

Hz. The output of this filter is then broken into two parts to be fed through anti-aliasing filters, before entering the monolithic bandpass filters. The term "aliasing" refers to the fact that high-frequency components of a time function can impersonate low frequencies. The first eighteen filters are Reticon R5604 1/3 octave bandpass filters. They use the new switched capacitor technology [40] which allows high levels of integration for analog electronics. The current part being used, the R5604, comes with three bandpass filters per package. This is not a limitation of the technology. This approach has been chosen, because the technology makes it possible to put all nineteen bandpass filters into a single package. The nineteenth bandpass filter spans one-octave with a center frequency of approximately 4.5 KHz. A single digital clock pulse is fed into each bandpass filter chip to determine the center frequency for each of the 1/3-octave filter bands.

The output of the bandpass filters are first rectified and then fed into a 30 Hz smoothing filter to determine the average energy in that band as a relative voltage level. The center frequencies of each bandpass filter are given in Diagram V-2.

ANALOG FILTER BANK

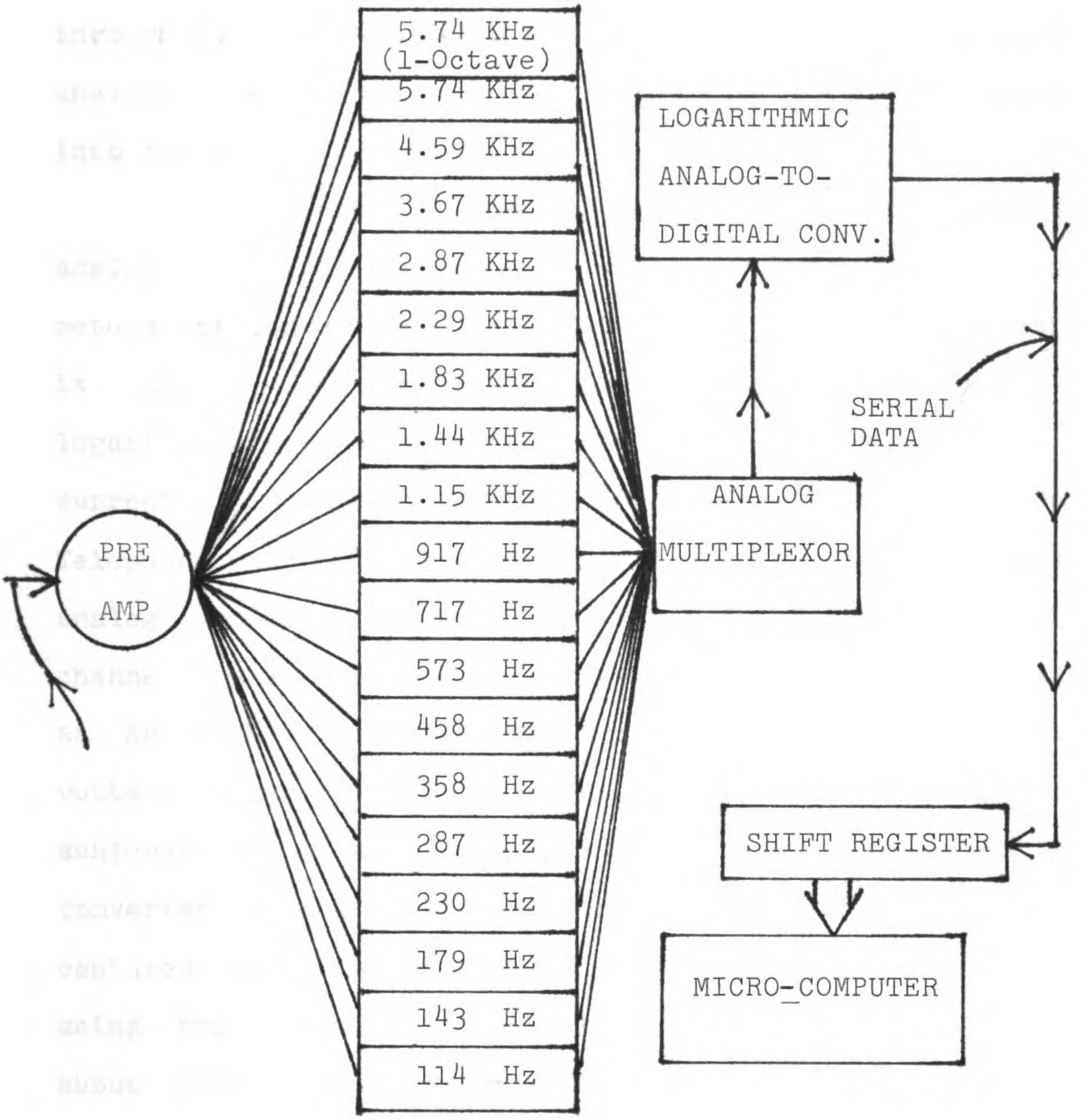
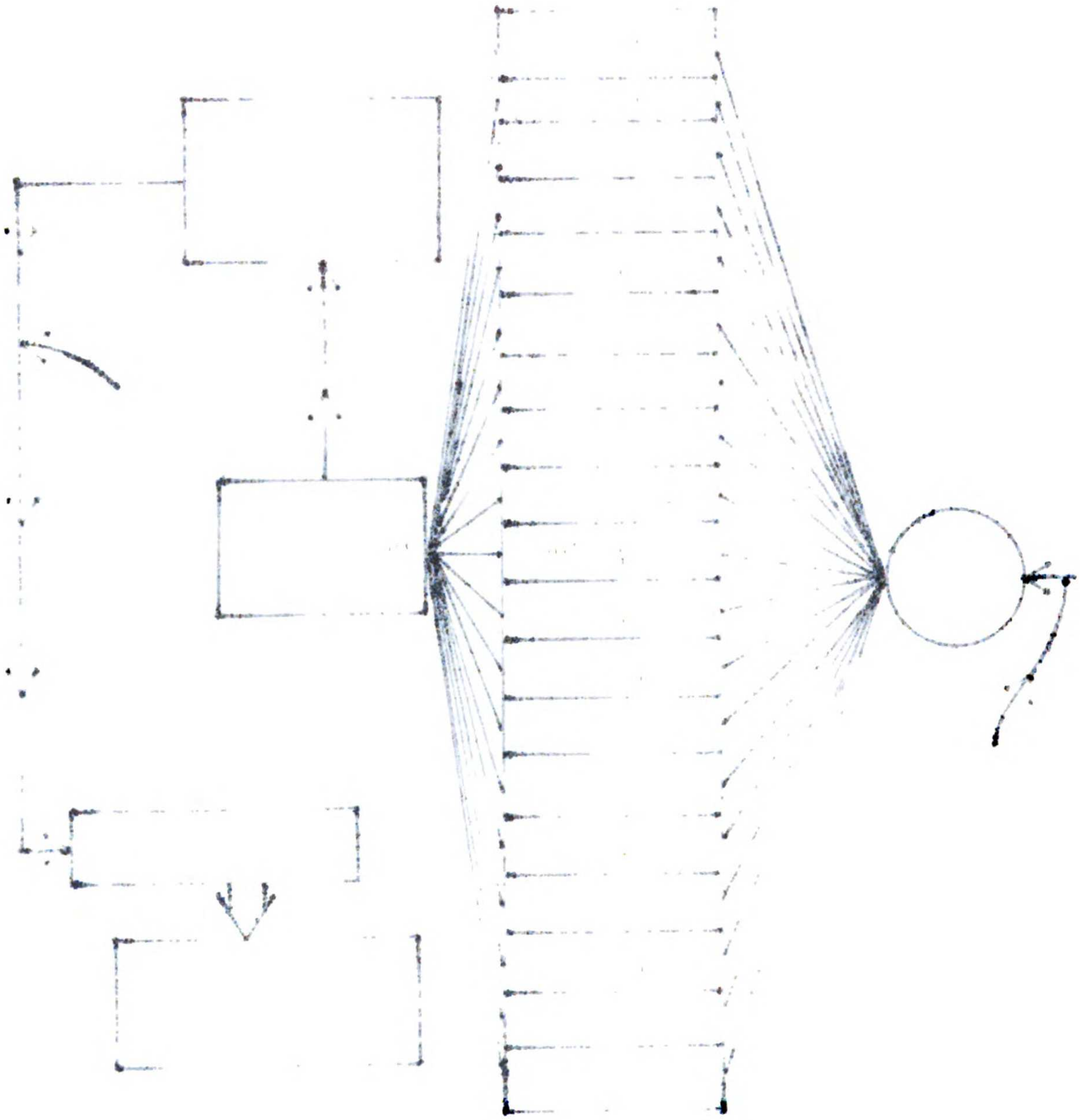


DIAGRAM V-2



From the 30 Hz filters, the 19 signals are fed through two 16-to-1 analog multiplexors for selection by the analog- to-digital converter. The total energy is also fed into the analog multiplexor.

A Siliconix CMOS codex, DF331, is used for analog-to-digital conversion. This part was chosen for its method of conversion. A micro-255 law companding conversion is used. The result is a close approximation to a logarithmic conversion. The micro-255 law, which is currently in use in the United States, is defined by Bell Telephone as the transfer function to be used in doing the analog (voice) to digital conversion in a telecommunication channel bank. The eight bit digital word is nearly the same as an eight bit logarithmic representation of the analog voltage. This gives the same dynamic range that would be achieved with a standard twelve bit analog-to-digital converter. The full dynamic range of speech can normally be captured by a twelve bit analog-to-digital converter or by using the technique described here. Twelve bit resolution about zero and 72 dB dynamic range is achieved. As a third benefit, this technique tends to automatically normalize variations in speaker amplitude.

The analog-to-digital converter can run up to 16 KHz. Since the speech signal is only sampled once every ten milliseconds, or at a rate 100 Hz, the analog to digital converter is turned on for 20 conversions once every ten milliseconds. Conversion is done at a rate of 10 KHz. The Z80 computer inputs the data through a parallel port when ready by polling a status bit. When the data is ready, the status bit goes high, it becomes a digital one. Thus, speech data is provided to the Z80 computer at the rate of 2,000 bytes per second, or twenty bytes per ten milliseconds - nineteen filter energies and one total energy.

C. Software Recognition Algorithms

The initial recognition algorithms used in the experimental system are designed to be nearly identical to the algorithms used in the filter bank model. The only differences are slight and due mostly to the real-time data requirements of the experimental system.

There are some important differences between the model and the experimental system, because the experimental system uses an analog filter bank instead of a modeled one. These differences were mentioned in Chapter IV.

There are six important software modules that make up the recognition system: LISTEN, FETCH, TRAIN, RECOG, ENDPT, and DYNAM.

FETCH is an assembly language program that returns nineteen parameters, eighteen filter energy differences and one total energy. The module is written in assembly language to keep up with the rate at which parameters are generated. Each filter energy is only available for 100 microseconds for input. FETCH performs one more important function. The parameters input from the filter bank hardware are digital numbers representing the log energy for each filter band. To make the recognition system immune to variations in speaker amplitude, the log ratio of energy is computed. The nineteen raw filter band energies get translated into eighteen parameters of the form LOG(F(i+1)/F(i)) where F(i) is the filter band energy and has a center frequency 1/3 octave lower than F(i+1). Since the analog to digital converter performs the log function, FETCH only needs to do a simple subtraction, LOG(F(i+1)/F(i)) = LOG(F(i+1)) - LOG(F(i)). For a micro-processor, divisions are computationally expensive, thus the use of a logarithmic analog-to-digital converter helps make the complete system run in real-time.

LISTEN is used to determine when speech has occured. The module calls FETCH until forty milliseconds of data are collected where the raw energy in each ten millisecond frame exceeds the background noise by a factor of four. LISTEN uses a circular buffer, so that the previous sixty milliseconds of speech are also captured. Since many fricatives have low total energy, but high energy in the high frequency, the first sixty milliseconds may contain useful information.

TRAIN is used for database construction. The user is allowed to label utterances as they are collected and made into templates. The beginning of speech is detected by the LISTEN module and then the FETCH module is called by TRAIN until one second of data is collected. The speech parameters are made into a template after the ENDPT module determines the end-point of the utterance.

ENDPT takes one second of parameters and searches for an end-point based on the total energy. The end-point of the speech is found by looking for fifteen consecutive windows, or 150 milliseconds of data where the total energy is less than four times the background noise. The length of each utterance is also saved.

RECOG performs the actual recognition task by listening for a speech sound, and then trying to find a best match in the template database built-up by the TRAIN module. Once an utterance has been collected in the same way that TRAIN builds a template, the DYNAM module is used to compare each template in the database to the unknown utterance. A score is computed for each template and the template with the lowest score is the most probable match for the unknown utterance.

DYNAM uses a version of the Sakoe-Chiba [39] dynamic programming algorithm to dynamically warp a template and an unknown utterance in time. This is the same algorithm used in the two models already described. As in the filter model, the distance measure used to compare two time slices is a simple Euclidian distance measure. The difference is taken between each of the corresponding nineteen parameters. The sum of the absolute value of each of the differences is computed as the final distance measure.

The performance of the experimental system is

CHAPTER VI

RESULTS AND EVALUATION OF DATA

A. Results From Modeled Systems

The modeled systems were tested on the digits zero through nine and on the letters of the alphabet. Not all of the available data was processed through the models. After the data from three speakers had been tested, an obvious trend became apparent. The recognition rates by speaker for each model, the filter model and the LPC model, are summarized in Table VI-1. The recognition rate averaged over three speakers for each model is also presented in Table VI-1.

SUMMARY OF RECOGNITION RATES FOR

LPC AND FILTER MODELS

SPEAKER	FILTER RECOGNITION RATE	LPC RECOGNITION RATE	NUMBER OF UTTERANCES	NUMBER OF REPETIONS
MALE #1	83%	76%	180	5
MALE #2	74%	64%	72	2
FEMALE # 3	73%	68%	180	5

AVERAGE RECOGNITION RATE FOR EACH MODEL   FILTERS - 77% LPC - 70%

TABLE VI - 1

An immediate conclusion can be drawn. This vocabulary with only a single training has an unacceptably low recognition rate. This, however is not the most important conclusion. The vocabulary was chosen to test the discriminating ability of LPC and filter banks as two possible speech front-ends.

For the recognition rates of each speaker, the filter model always outperforms the LPC model. The average recognition rate for the filter model over all speakers is 77% , while the overall recognition rate for the LPC model is 70% . For the data set tested, and for the models used, the filter bank method for representing speech is superior to the LPC approach.

The next question that can be asked about the data is: where do the two models fail, and what are the similarities or differences of these failures? In Table VI-2 and Table VI-3, a list is given of each error made by the two models, along with the number of times each error is made. For the filter model seven utterances constitute 53% of the errors. Those seven utterances are: THREE (7), D (12), E (6), K (8), P (5), T(5), and Z (8). The number in parenthesis is the number of times the utterance was misclassified. The letters D, K, and Z, are most frequently missed with the respective errors of twelve, eight, and eight. For the LPC model seven letters constitute 45% of the misrecognized utterances. Those seven letters are: B (7), D

(7), E (11), K (6), P (9), T (8), and X (6). The letters E, P, and T are most frequently missed with respective errors of eleven, nine and eight. The difficulty in recognizing some of the utterances can be better understood by looking at spectrograms made for each of the utterances. The similarity between "P" and "T" can be seen visually by looking at the respective spectrograms in Appendix B. These spectrograms were made from the audio tapes of the data.

FILTER MODEL DATA RESULTS

ERRORS MADE

THREE SPEAKERS - 13 REPETITIONS

terance	Utterance Mistaken For
	NO MISSES
	C,M,Q
	B,EIGHT(2x),V(4x)
	Y
	I(4x),R,Y(3x)
	X(2x)
	NO MISSES
	THREE
	ONE
	NO MISSES
	K
	E,D,T(2x)
	T
	B(3x),C,E(3x),P,T(2x),V,Z
	B,D(2x),P(2x),Z
	S(2x)
	C(2x)
	EIGHT
	Y
	SIX
	A(2x),J(6x)
	NO MISSES
	N(2x)
	M(3x)
	NO MISSES
	E(2x),V(3x)
	C
	FIVE
	F(2x)
	G,P(4x)
	Q,W
	B,D,P,T
	NO MISSES
	NO MISSES
	FIVE
	C(3x),V(5x)

indicates that EIGHT was mistakenly chosen two times

TABLE VI - 2

LPC MODEL DATA RESULTS

ERRORS MADE

THREE SPEAKERS - 12 REPETITIONS

Correct Utterance	Utterance Mistaken For
ONE	NINE
TWO	Q(2x), U(2x)
THREE	NINE, J, V, Z(2x)
FOUR	R
FIVE	NINE, I(3x)
SIX	X
SEVEN	NO MISSES
EIGHT	A(2x), B, P, Z
NINE	NO MISSES
ZERO	NO MISSES
A	K(3x), N
B	D, E(2x), P, T, V
C	T(4x), V
D	B(3x), G, T(2x), Z
E	B(4x), D, G, P(3x), T, V
F	NINE, S(2x)
G	J(2x), T
H	B
I	FIVE, NINE
J	A
K	A, J(4x), T
L	NO MISSES
M	N(3x)
N	M
O	NO MISSES
P	B(2x), D, T, V(4x), Z
Q	ZERO,TWO, THREE, U
R	FOUR
S	Y
T	D(3x), G, P(3x), V
U	Q, W
V	B, P(2x), V
W	NO MISSES
X	S(6x)
Y	ONE, FIVE, I
Z	THREE, G, T, V

Q(2x)-indicates that Q was mistakenly chosen two times

TABLE VI - 3

The two models seem to differ. The LPC model ognizes "S" for "X" six times. The filter model never ognizes "X". The filter model misrecognizes "Z" eight but the LPC model also misrecognizes "Z" four times. odels miss "B" and "THREE" several times.

There are several utterances that are never ognized by either model. The utterances SEVEN, ZERO, and W are never missed by either model. LPC never NINE while the filter model misses it one time. The model never misses ONE or X. The LPC model misses ONE out misses X six times.

The main difference in the two models is the ulty that the LPC model has in discriminating "X" from

ults From Experimental System

To test the experimental speech recognizer, several of data were run through the system. The first set of s taken from one male and one female speaker repeating igits and the letters of the alphabet, the same data in the modeling described above. The rest of the data ents an effort to identify a vocabulary made-up of -symbols that can be recognized with a high degree of cy. A single-symbol vocabulary is important for making selections on a CRT screen. The following vocabularies were tested: the military word equivalent of the alphabet, the Greek alphabet, and the digits zero through nine. The military pronunciation of the alphabet is also used in aviation and is sometimes referred to as the phonetic alphabet.

Data from five repetitions of the digits and the letters of the alphabet for male speaker number one and for female speaker number one were tested. These were the same five repetitions of the vocabulary used to test the two models. Both speakers scored a recognition rate of 68%. The missed utterances and the utterances they were mistaken for are listed in Table VI-4. Seven most frequently missed utterances cause 47% of the errors. Those seven are: FIVE (7), D (9), E (6), G (7), P (7), V (9), and Z (8). Six utterances are never missed; they are SIX, SEVEN, ZERO, N, O, and S.

EXPERIMENTAL SYSTEM

ERRORS MADE

TWO SPEAKERS - 10 REPETITIONS

Correct Utterance	Utterance Mistaken For
ONE	M, N
TWO	P(2x)
THREE	P, V(2x), Z
FOUR	O
FIVE	I(4x), N, R(2x)
SIX	NO MISSES
SEVEN	NO MISSES
EIGHT	E,H(4x)
NINE	I, N
ZERO	NO MISSES
A	D, J
B	D(2x), E, V
C	T
D	B(2x), E(4x), P, V(2x)
E	D, B, P, V(3x)
F	FIVE, S
G	E, P, T(2x), V(3x)
H	EIGHT
I	FIVE(3x), R, Y(2x)
J	E, K
K	A(2x), E, J(2x)
L	ONE
M	ONE(2x), N(2x), X
N	NO MISSES
O	NO MISSES
P	B, E, T(4x), V
Q	THREE
R	Y
S	NO MISSES
T	G, P(3x)
U	E(2x)
V	B(2x), D, E(3x), P(2x), T
W	T
X	F(2x), S(3x)
Y	R
Z	C, E(2x), P, T(4x)

P(2x)-indicates that P was mistakenly chosen two times

TABLE VI - 4

SUMMARY OF RECOGNITION RATES OF

EXPERIMENTAL SYSTEM - DIGITS & ALPHABET

SPEAKER	NUMBER OF UTTERANCES	NUMBER OF REPETITIONS	RECOGNITION RATE
MALE #1	180	5	68%
FEMALE #1	180	5	68%

AVERAGE RECOGNITION RATE OVER BOTH SPEAKERS	68%

TABLE VI - 5

In search of a single symbol vocabulary that could be recognized with a high rate of accuracy the military pronunciation of the alphabet was tested. The military pronunciation is given in Table VI-6.

MILITARY PRONUNCIATION OF THE ALPHABET*

LETTER	SPOKEN AS
A	ALFA
B	BRAVO
C	CHARLIE
D	DELTA
E	ECHO
F	FOXTROT
G	GOLF
H	HOTEL
I	INDIA
J	JULIETT
K	KILO
L	LIMA
M	MIKE
N	NOVEMBER
O	OSCAR
P	PAPA
Q	K-BEK
R	ROMEO
S	SIERRA
T	TANGO
U	UNIFORM
V	VICTOR
W	WHISKEY
X	XRAY
Y	YANKEE
Z	ZULU

*Taken from the U.S. NAVY - MARINE CORPS MILITARY AFFILIATE RADIO SYSTEM, pg 6-14

TABLE VI - 6

Two speakers were tested, male number three and female number two. For the male speaker, three repetitions of the vocabulary were run through the experimental system from audio tape. A 96% recognition rate was achieved. Three errors were made. LIMA was mistaken for PAPA, ALPHA was mistaken for GOLF, and BRAVO was mistaken for ROMEO.

For the female speaker, four repetitions of the vocabulary produced a 95% recognition rate. Five errors were made. They were: NOVEMBER mistaken for SIERRA, ECHO mistaken for VICTOR, OSCAR mistaken for ALFA, VICTOR mistaken for ECHO and KILO mistaken for ECHO.

SUMMARY OF RECOGNITION RATES FOR

EXPERIMENTAL SYSTEM - MILITARY WORDS

SPEAKER	NUMBER OF UTTERANCES	NUMBER OF REPETITIONS	RECOGNITION RATE
MALE #3	78	3	96%
FEMALE #2	104	4	95%

AVERAGE RECOGNITION RATE
OVER BOTH SPEAKERS - 95.4%

TABLE VI - 7

EXPERIMENTAL SYSTEM

ERRORS MADE

MILITARY WORD EQUIVALENT OF THE ALPHABET

TWO SPEAKERS - 7 REPETITIONS

Correct Utterance	Utterance Mistaken For
ALFA	OSCAR
BRAVO	NO MISSES
CHARLIE	NO MISSES
DELTA	NO MISSES
ECHO	KILO, VICTOR
FOXTROT	NO MISSES
GOLF	ALPHA
HOTEL	NO MISSES
INDIA	NO MISSES
JULIETT	NO MISSES
KILO	NO MISSES
LIMA	NO MISSES
MIKE	NO MISSES
NOVEMBER	NO MISSES
OSCAR	NO MISSES
PAPA	LIMA
K-BEK	NO MISSES
ROMEO	BRAVO
SIERRA	NOVEMBER
TANGO	NO MISSES
UNIFORM	NO MISSES
VICTOR	OSCAR
WHISKEY	NO MISSES
X-RAY	NO MISSES
YANKEE	NO MISSES
ZULU	NO MISSES

TABLE VI - 8

A second alphabet, the Greek alphabet, was tested for its ability to be recognized by the experimental system. See Table VI-9 for a list of the Greek alphabet.

GREEK ALPHABET

ALPHA
BETA
GAMMA
DELTA
EPSILON
ZETA
ETA
THETA
IOTA
KAPPA
LAMBDA
MU
NU
XI
OMICRON
PI
RHO
SIGMA
TAU
UPSILON
PHI
CHI
PSI
OMEGA

TABLE VI - 9

The vocabulary was tested from five repetitions from audio tape generated by male speaker number three. From 120 utterances, 32 errors were made to give a recognition rate of 73%. The errors made are summarized in Table VI-11. Only one male speaker was tested since the recognition rate was so poor.

SUMMARY OF RECOGNITION RATES OF

EXPERIMENTAL SYSTEM - GREEK ALPHABET

SPEAKER	NUMBER OF UTTERANCES	NUMBER OF REPETITIONS	RECOGNITION RATE
MALE #3	120	5	73%

TABLE VI - 10

EXPERIMENTAL SYSTEM

ERRORS MADE

GREEK ALPHABET

ONE SPEAKER - 5 REPETITIONS

Correct Utterance	Utterance Mistaken For
ALPHA	LAMBDA
BETA	ETA, THETA(2x)
GAMMA	THETA
DELTA	NO MISSES
EPSILON	UPSILON(3x)
ZETA	NO MISSES
ETA	THETA(4x)
THETA	BETA(2x)
IOTA	LAMBDA
KAPPA	NO MISSES
LAMBDA	NO MISSES
MU	NU
NU	MU
XI	PHI(2x), PSI(2x)
OMICRON	NO MISSES
PI	PHI(3x), CHI
RHO	NO MISSES
SIGMA	NO MISSES
TAU	NO MISSES
UPSILON	NO MISSES
PHI	PSI
CHI	PI(2x), PHI
PSI	PHI(2x)
OMEGA	PHI

THETA(2x)-indicates that THETA was mistakenly chosen two times

TABLE VI - 11

The final vocabulary tested was the digits zero through nine. Three speakers, male number three, male number four and female number two were tested. Each speaker's utterances were tested on a single training and 150 utterances. A second test was performed on three trainings and fifty utterances for each speaker. The purpose of the second test was to determine the effect of multiple trainings on the recognition rate. From each training a set of templates was constructed. Thus, from three trainings and a ten word vocabulary thirty templates were constructed. The results of these tests are given in Tables VI-12 through VI-14.

SUMMARY OF RECOGNITION RATES FOR

EXPERIMENTAL SYSTEM - DIGITS

SPEAKER	NUMBER OF TRAININGS	NUMBER OF UTTERANCES	NUMBER OF REPETITIONS	RECOGNITION RATE
MALE #3	1	150	15	96%
	3	50	5	100%
MALE #4	1	150	15	94%
	3	50	5	100%
FEMALE #2	1	150	15	93%
	3	50	5	98%

Average Recognition Rate Over all Speakers	Single Training - 94% Three Trainings - 99.3%

TABLE VI - 12

EXPERIMENTAL SYSTEM

ERRORS MADE

DIGITS 0-9

SINGLE TRAINING

THREE SPEAKERS -45 REPETITIONS

Correct Utterance	Utterance Mistaken For
ZERO	SIX (5x), SEVEN
ONE	NINE (3x)
TWO	NO MISSES
THREE	NINE (2x)
FOUR	ONE
FIVE	ONE (4x), SEVEN, NINE
SIX	NO MISSES
SEVEN	ZERO (2x), NINE
EIGHT	THREE (2x), SIX, FOUR
NINE	NO MISSES

TABLE VI - 13

EXPERIMENTAL SYSTEM

ERRORS MADE

DIGITS 0-9

THREE TRAININGS

THREE SPEAKERS -15 REPETITIONS

Correct Utterance	Utterance Mistaken For
ZERO	SIX
ONE	NO MISSES
TWO	NO MISSES
THREE	NO MISSES
FOUR	NO MISSES
FIVE	NO MISSES
SIX	NO MISSES
SEVEN	NO MISSES
EIGHT	NO MISSES
NINE	NO MISSES

TABLE VI - 14

C. Comparison of Results From the Three Systems

The three systems, the filter model, the LPC model and the experimental system can be compared on their performance with similar data sets. By rank of their recognition scores, the filter model did best, followed by the LPC model.

There are several possible reasons why the experimental system did not perform as well as the filter model. The most likely source of error is due to lack of precision on the experimental system. In the filter model, a sixteen bit analog-to-digital converter was used. In the experimental system, an eight bit logarithmic analog-to-digital converter was used giving only an equivalent of twelve bits of accuracy.

Another possible source of error is the greater precision carried throughout the filter model. In critical areas of computation floating point was used in the filter model. In contrast, the experimental model, because of processing time constraints, used sixteen bit integer representation throughout, once the eight bit speech data was collected.

The filter model and the experimental system may also be compared on the basis of the errors made. The seven most frequent errors made by the filter model were THREE (7), D (12), E (6), K (8), P (5), T (5), and Z (8). The seven most frequent errors made by the experimental system were FIVE (8), D (9), E (6), G (7), P (7), V (9), and Z (8). Both systems miss FIVE and THREE several times. The only significant difference seems to be the inability of the experimental system to correctly recognize the letter X; it was missed five times. The system mistakenly classifies X as an S. The filter model never missed the letter X. The difference might be due to the addition of one extra filter on the experimental system. The filter is centered around 5.74 KHz. The energy coming from this filter may obscure the difference between the letters X and S.

D. Comparison of Results for Different Vocabularies

The experimental system was tested on a number of different vocabularies. These vocabularies can be ranked by order of recognition rate. The results are summarized below:

68%	Digits & Alphabet
73%	Greek Alphabet
94%	Digits on a Single Training
95.5%	Military Pronunciation of the Alphabet
99.3%	Digits on Three Trainings

From these results it can be seen that the choice of vocabulary clearly influences the recognition rate. Multiple repetitions of the vocabulary for increasing the number of templates used in the matching process also appears to increase the recognition rate. This has only been verified for the digits.

E. Conclusion

Several important conclusions can be drawn. For the vocabulary tested the filter model outperformed the LPC model. The experimental model did not do as well as the filter model, although the differences are not great. The digits and the letters of the alphabet are not easily recognizable by any of the systems tested.

The performance of the experimental system is greatly improved by certain vocabularies. The Greek alphabet is difficult to recognize, but the military pronunciation of the English alphabet scored very high. However, the highest recognition rate, 99.3%, was scored when three trainings of the digits were used.

The experimental system can perform as a high accuracy speech recognition system with at least one vocabulary, the digits, when three trainings are used.

CHAPTER VII

PROPOSED USE OF SPEECH RECOGNITION FOR MEDICAL DATA INPUT

A. A Class of Medical Data Input Systems

The research described in this dissertation has investigated and developed new methods for low cost speech recognition for medical applications. It is hypothesized that a class of cost effective applications in the medical environment may now be implemented. It is proposed that one or more of the following projects be implemented to substantiate the hypothesis: a radiology reporting system, a clinical laboratory data input system, a surgical reporting system, or a cardiology reporting system. The most promising project to date, and the example that has involved the largest effort to date, is the cardiology reporting system. However, any of the above set of applications would be both interesting and revealing, since each is representative of a much more general class of data input problems. In the paragraphs below, the current cardiology system is described first and then a proposed intelligent, voice-aided system is described.

B. A Medical System with Speech Recognition

Present System

Currently, Kaiser hospital of Oakland takes routine cardiograms for older patients, as part of a multi-phasic physical in conjunction with their attempt at early detection of heart disease. Large numbers of cardiograms must be individually interpreted by a qualified cardiologist. The physician's findings are intially captured by dictation machines. The dictation is then transcribed by a medical transcriptionist. An attempt is currently being made to have the information transcribed into a computer database, so that this part of the patient's medical record is available for machine manipulation and retrieval. A group at Kaiser has started to explore the use of voice input to capture the cardiology reports. Members of the Kaiser group approached the author after hearing of the MIS program's interest in voice recognition.

In order for the author to learn how cardiology reports are handled, a smaller EKG facility at French hospital in San Francisco was studied. The bulk of their work consists of the standard 12-lead EKGs. Three cardiologists divide-up the work load. The reports are handwritten on a standard form and then transcribed onto a typed form. For each hour of the cardiologist's time in interpreting the findings, it takes about a third of an hour of a transcriber's time to type the results.

The handwritten form used by the physician is imprinted with the patient's name and attending physician using an embossed card. Several questions about the patient must be answered on the top half of the form: drugs being taken, current illness, bed number and date of the EKG. The cardiologist must also intialize the form. All questions on this part of the form are answered by a number or by checking a yes-no alternative.

The bottom two-thirds of the form are filled in with the EKG data. Three completed, handwritten forms are included in the appendix. The first six items are completely numeric: heart rate-atrial, heart rate-ventricular, PR, QRS, QT, and electrical axis. The next five items are non-numeric. The responses appear to be limited to one of several standard conditions. The five items are: Rhythm, P waves, QRS complexes, ST segments, and T waves. The last two items, "Remarks" and "Conclusion" allow for some form of free text. The text is severely limited by the context to a finite number of heart conditions that can exist.

After examining forty-four forms, it appears that an additional data item could be added, "progress of disease." The types of response would be, condition deteriorating, condition the same, condition improving, etc.

The type-written report has a much different format. The top two-thirds of the report is taken up by the twelve cardiograms and a long narrow cardiogram from lead II. The bottom third of the form contains the typed text. The transcriber must also type in the patients name, hospital number, date, and time. The final report must be verified and signed by the cardiologist. If there are problems in transcribing the handwritten report, the report is sent back to the physician for clarification. Such problems are usually caused by illegible handwritting or undecipherable abbreviations.

The above description is the result of a preliminary study. Before a computer aided system is implemented more forms should be studied and more knowledge about the cardiology facility should be acquired. Several sample forms follow as an example of the data collected at French Hospital.

EKG FRENCH HOSPITAL

Date to be Done____________________

Pre-op? _________yes _________no

STAT (Reason)____________________

Name____________________Date of Birth__________Sex______Bed________

Hosp. No.______________Pvt/Soc________Attending Physician____________________

Clinical Diagnosis____

Digitalis?________Quinidine?________Other?____________________

Date Record Taken______________Time__________By_________Comments______________

Rate - atrial: PR: QRS: QT:

- ventricular: Electrical Axis:

Rhythm:

P waves:

QRS complexes:

ST segments:

T waves:

Remarks:

CONCLUSION:

Pt. will pick-up Ekg

EKG **FRENCH HOSPITAL**

Date to be Done____________

Pre-op? ______yes ______no

STAT (Reason)____________

Name____________ Date of Birth______ Sex______ Bed______

Hosp. No.________ Pvt/Soc______ Attending Physician______

Clinical Diagnosis____________

Digitalis?______ Quinidine?______ Other? Halotestin / adriamycin

Date Record Taken 4-17 Time 9:00 AM By ms Comments______

Rate - atrial: 95 PR: .14 QRS: .08 QT: .32

- ventricular: 95 Electrical Axis: +30

Rhythm: nsr

P waves: nl

QRS complexes: low voltage

ST segments: nl

T waves: nl

Remarks:

CONCLUSION:

AR

QRS
~~low voltage~~ ste

QRS voltage is low and slightly reduced SRO 3-27-78

FRENCH HOSPITAL

ELECTROCARDIOGRAPH DEPARTMENT

Admission No. ____________ Date or Requisition ____________

Name ____________ Date of Birth ____________ Sex ____________

Address ____________ Book No. ____________ Bed No. ____________

____________ Attending Physician ____________

Clinical Diagnoses Severe Degenerative Arthritis Left Hip

Digitalis ? ____________ Quinidine ? ____________ Other ____________

Do NOT write below this line ••• For Departmental Use Only

Date Record Taken 4-17 355pm

Rate - atrial: 73 PR: .18 QRS: .08 QT: .36

- ventricular: 73 Electrical axis: -30

Rhythm:

NSR

P waves:

↓ 2 3 F

QRS complexes:

nl

ST segments:

↓ I, L

T waves:

nl

Remarks:

CONCLUSION:

Prob.

AR

LAD

Minor STS Δ

Low atrial rhythm

SRW 11-12-74 RSC

EKG **FRENCH HOSPITAL** [redacted]

Date to be Done

Pre-op? ✓ yes ________ no

STAT (Reason)

Name ________ Date of Birth 01-11-96 Sex ________ Bed

Hosp. No. ________ Pvt/Soc ________ Attending Physician

Clinical Diagnosis Cat. Lt. eye

Digitalis? ________ Quinidine? ________ Other?

Date Record Taken 4/[illegible] Time 2.[illegible] By [illegible] Comments

Rate - atrial: 72 PR: 15 QRS: 07 QT: 36

- ventricular: 72 Electrical Axis: +30

Rhythm: NSR c̄ freq VPCs in bigeminal patter

P waves: nl

QRS complexes: nl

ST segments: ↓ I, L, V4-6

T waves: ↓ V1-3 & diph V4-6, I, L

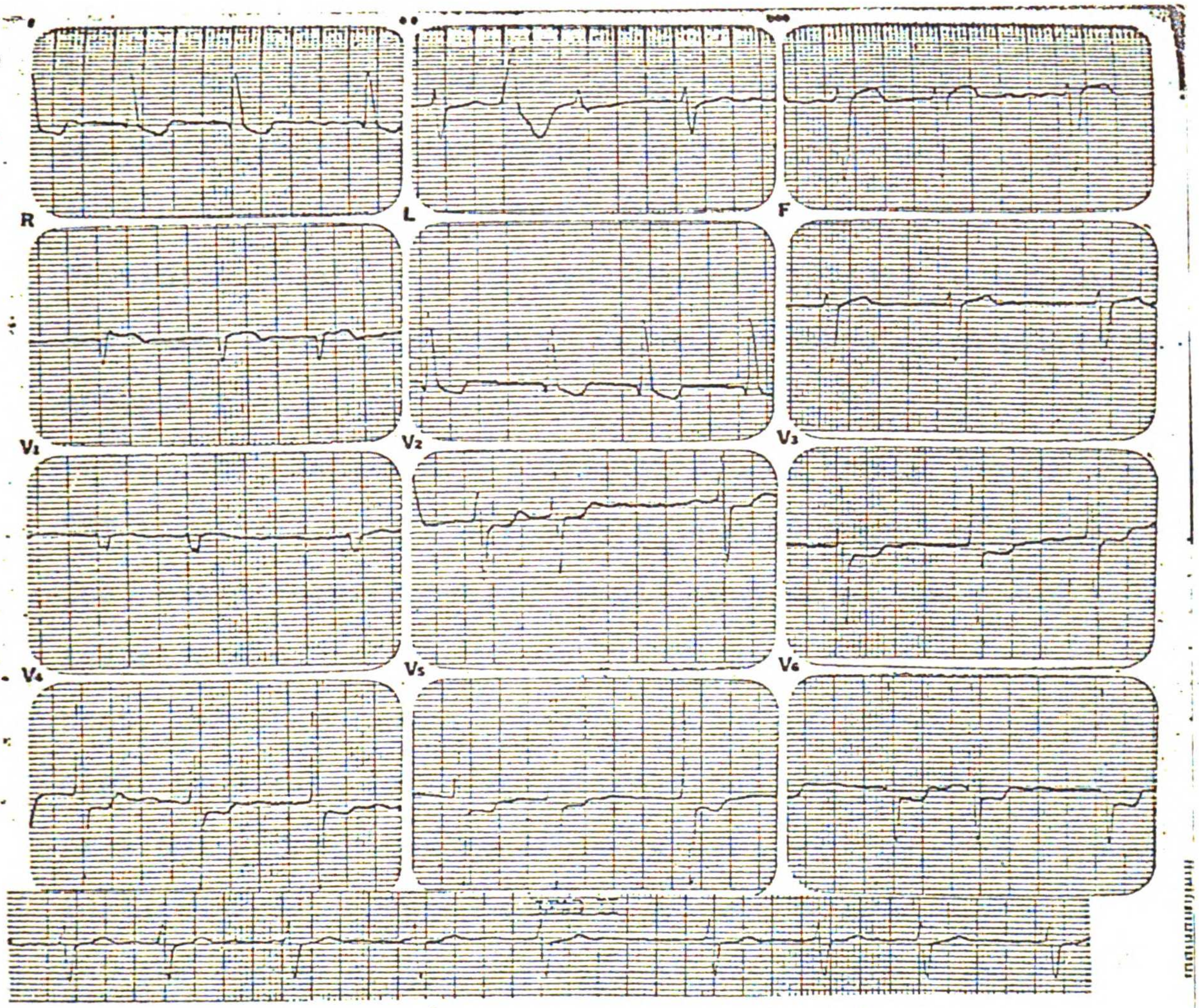
Remarks:

CONCLUSION:

AR-

freq VPC's & STT-WA's [illegible]

SRO 1-2-73



Rate: Ventricular 70 QRS: 0.10 QT: 0.36 Axis: -45°
Rhythm: atrial fibrillation with rare ventricular premature contraction
QRS complexes: Tall R in AVL
ST segments: Depressed in I, AVL, V2-V6
T waves: Diphasic

CONCLUSION; Abnormal record.
Atrial fibrillation.
Left anterior fascicular block.
Left ventricular hypertrophy with associated IV conduction defect and ST-T wave abnormality.
Since 1-30-78 T waves are less inverted.

EKG Date 2-9-78 Time 11:00pm Hosp. No. Patient

1.3 0.4 80. 17. 13.

Intelligent Voice Aided System

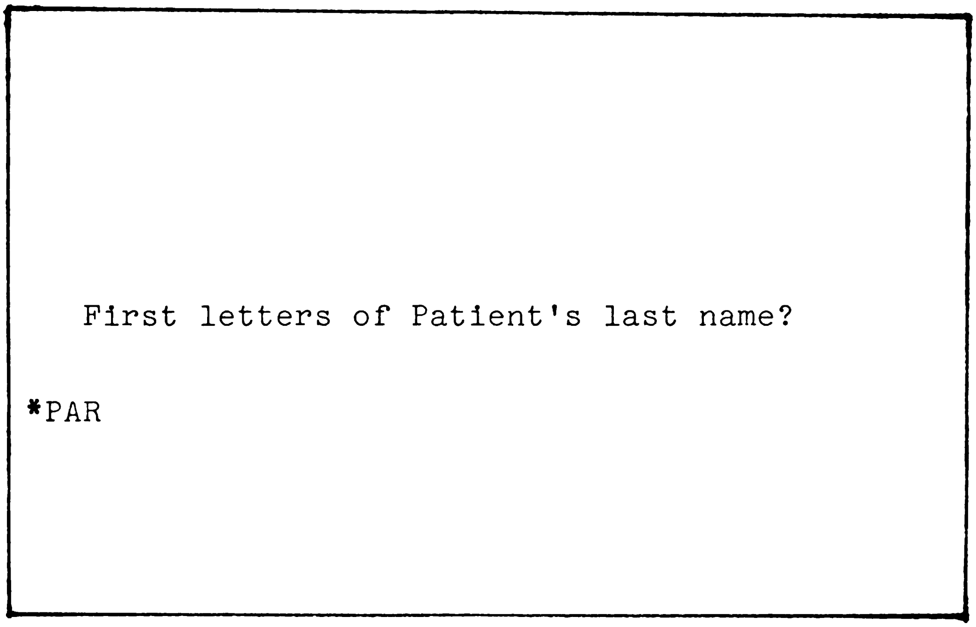
The basic premise is that the physician's input can be satisfactorily captured by a cost effective, intelligent speech recognizer. The basic outcome would be the elimination of the handwritten or dictated report and the automatic generation of typewritten reports by computer when necessary.

The hardware could consist of a speech recognizer, a CRT terminal, a micro-processor, and an inexpensive printer. The exact configuration would depend on the needs of the installation. A low cost floppy disk or an interface to a host computer could provide long term storage for data. A high baud rate interface between the CRT and micro-computer or a DMA refresh memory would be used to provide a virtually instantaneous output device. A considerable effort remains in the design of all the appropriate hardware components to do the particular application eventually selected.

The speech recognizer would be constrained to a limited vocabulary of short phrases. To help overcome this limitation, the use of list-selection and context switching could be used.

As an example of list-selection, the problem of patient identification is used. Assume the micro-computer is linked to a host computer with a patient ID data base. The problems of entering and keeping patient ID information in a computer database is a solved problem for some hospitals and will not be dealt with here. The cardiologist could select the appropriate patient information by speaking the first few letters of the patient's name. This would cause all patient's names that match on those letters to appear on the CRT screen. Each patient's name would be numbered and the physician would pick his patient by speaking the appropriate number. See the following example of possible CRT displays that show the selection of a patient's name from a list of names. For a stand alone system a clerk could "register" the patient into the system at the time the request for the procedure is made, or when the procedure is performed.

CRT Display for Retrieving Patient Name

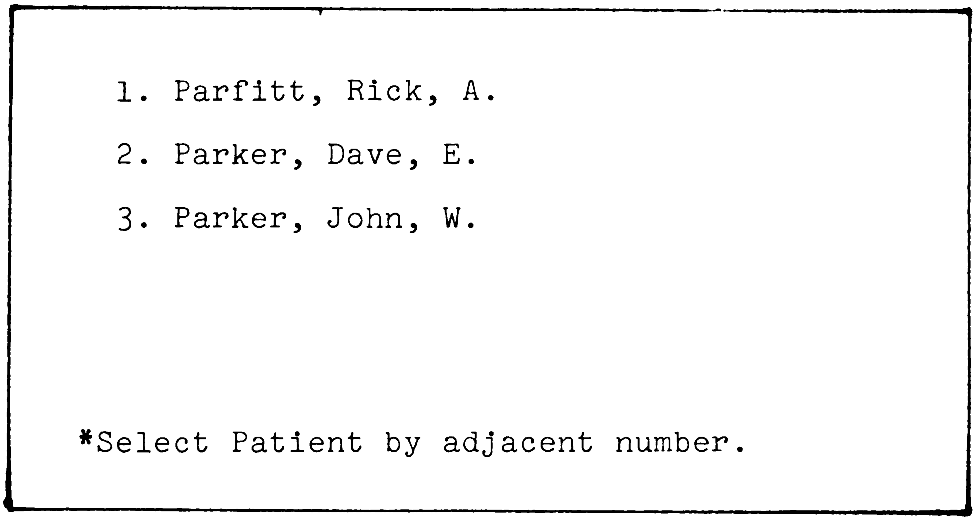


First letters of Patient's last name?

*PAR

The system responds to voice input by printing letters spoken next to asterick. The input is terminated by speaking the command "done."

DISPLAY 1

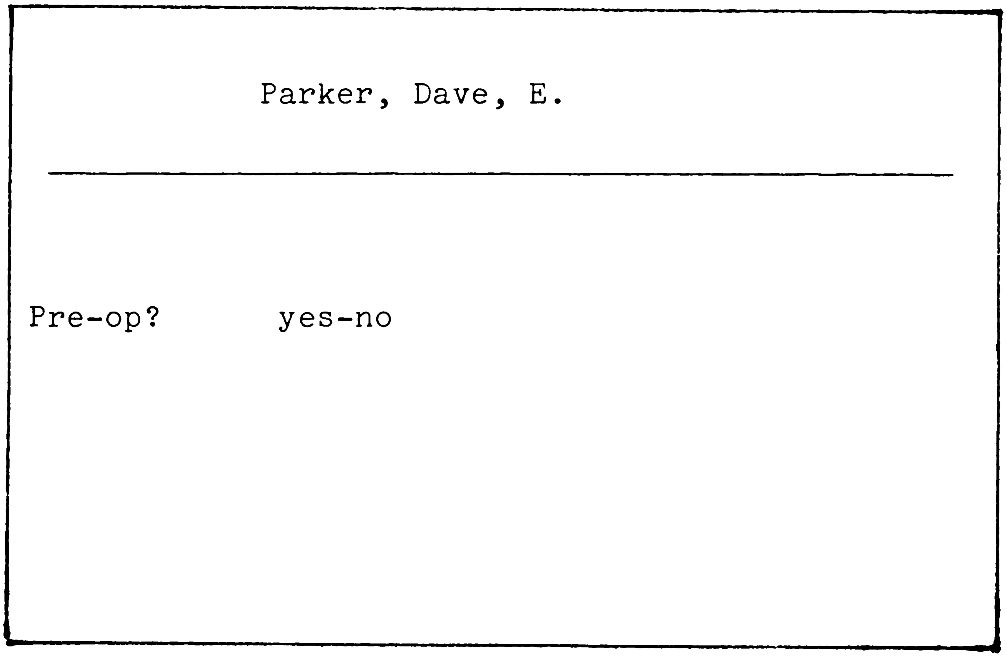


1. Parfitt, Rick, A.
2. Parker, Dave, E.
3. Parker, John, W.

*Select Patient by adjacent number.

The desired patient is now selected by speaking the appropriate number next to the patient name.

DISPLAY 2



Parker, Dave, E.

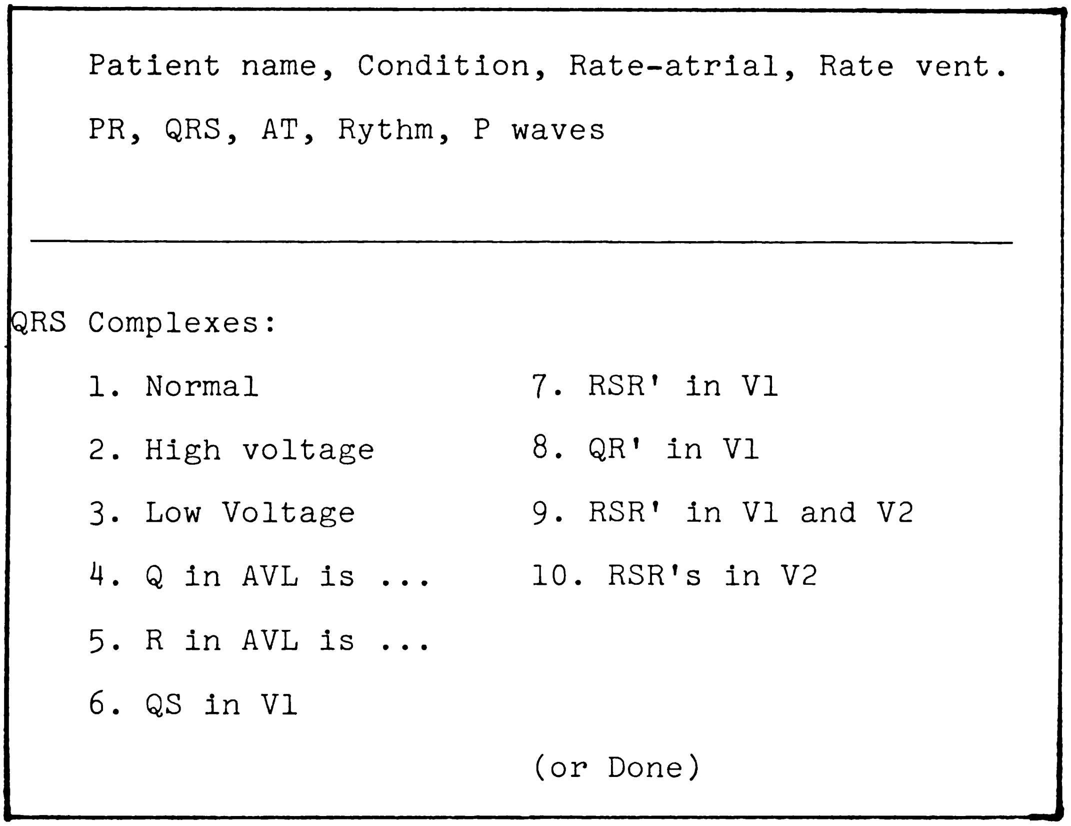
Pre-op? yes-no

The patient's name is now placed at the top of the screen and the system is now ready for the next command. The physician answers yes or no to whether the patient is Pre-op.

DISPLAY 3

The use of context switching could be tried to expand the allowable vocabulary size. The vocabulary would be divided into sets. Each set would be recognized only in the appropriate context. When describing the axis of the heart, the digits 0-9, "minus", and "plus" would constitute the recognizable vocabulary. And, when it comes time to describe the QRS complexes, a different vocabulary set may be more appropriate. The following example of a CRT display for selecting the appropriate QRS complex uses numerical selections. Another alternative is to use a small vocabulary that is only recognizable in the context of a QRS selection. The vocabulary might represent the actual values being selected. Instead of voicing the number "1" for "normal", the user might utter "normal." The user would utter "High Voltage" instead of "2", etc. (See the following CRT display.).

CRT SCREEN FOR DESCRIBING THE QRS COMPLEXES



Patient name, Condition, Rate-atrial, Rate vent.

PR, QRS, AT, Rythm, P waves

QRS Complexes:

1. Normal
2. High voltage
3. Low Voltage
4. Q in AVL is ...
5. R in AVL is ...
6. QS in V1
7. RSR' in V1
8. QR' in V1
9. RSR' in V1 and V2
10. RSR's in V2

(or Done)

Out of thirty-four reports examined, fourteen had a normal QRS complex reported. Other responses were short and comprised a small vocabulary. In the above frame, the physician selects the adjacent number to his choice. The query continues from there if numeric data is required. Several choices may be made until the physician gives the "Done" command.

DISPLAY 4

The choice of actual words in the vocabulary recognized by the computer must be made very carefully. Words that sound similar or confuse the recognition algorithm should be avoided.

It is believed that a 2% error rate or less for recognizing phrases may be necessary to make the system tolerable. The exact acceptable error rate is difficult to determine at this point and should be one of the subjects of the investigation.

A simple error correction feed-back loop is essential. When a word is spoken by the user, a convenient means must exist for the user to know what the system recognized. If an error occurs, a simple means for correcting the error must be provided.

All the physician's responses are through spoken phrases. The exact content of the final frames should be determined after a much larger number of cardiology reports are studied and the results are reviewed by an experienced cardiologist.

The final report could be printed by the computer in a format similar to the forms currently being used. The facilities for capturing the electrocardio wave forms with a computer and then reproducing them with a printer should be utilized if available. Otherwise, the old cut-and-paste methods could be used to put the cardiograms on the final report.

CHAPTER VIII

SUMMARY & CONCLUSION

A. Summary

This dissertation represents an effort to accomplish several interrelated tasks. The importance of this work must be judged in relationship to previous efforts and the current state of knowledge about speech recognition.

Chapter two described previous work that has been done to develop speech recognition. The description was not complete, but it covers most of the major developments in the field. A short history of isolated-phrase recognition systems was given and several important current systems were described. It is the description of the currently available systems that lends perspective to the magnitude of the research described here.

The work described herein solves several important problems that have not been dealt with until now. Chapter III presents an idealized model of what a speech recognition system is. The purpose of Chapter III was to make the description of efforts in the rest of the dissertation more understandable.

Chapter IV described the modeling of one part of the recognition process, the "Front-End." The speech

"Front-End" is where the analog representation of speech, a time varying voltage, is converted into some form of feature representation. It is here that a major effort has been made. Of the several methods that have been used for representing speech, this work has been concentrated on two, filter banks and LPC. They are the two methods that have enjoyed widespread usage by other researchers. To begin the process of building a high-accuracy, low-cost speech recognition sytem, one of the two methods had to be selected. Chapter four described the efforts made to model the two methods. The purpose of the two models was to test the performance of each technique on real speech data. The technique with the best characteristics, bandpass filters, was selected.

The next step in this research effort was to reduce the cost by simplifying the design of the speech "Front-End." Although the final goal of this approach is the design of a single integrated circuit for feature extraction, only the first major step in this process has been taken. Chapter five describes the design and proto-typing of a speech recognition system based around an inexpensive micro-processor. The choice of components, the design of the circuits and the choice of technology have all been done carefully. This care has been taken so that the final system design could be used by an integrated circuit designer as a guide for laying out a semi-conductor mask.

Prior to Chapter VI, only ideas, models and designs have been presented. In Chapter VI, the data results used in the test to evaluate the models and designs are given. The results from the modeling indicate that a filter oriented design should be used. When the data is processed through the two models, the filter bank representation of speech outperforms the LPC representation of speech. The experimental system was also critically tested against a series of data. The results of these tests confirm the hypothesis that a low-cost, high-accuracy speech recognition system can be built using the design described in Chapter V. Such a system can perform with a high degree of accuracy, 99.3%, with a small vocabulary and three training samples for each utterance.

Chapter VII describes the importance of the recognition system for certain medical data input problems. An example of an actual medical data input problem is identified and a potential solution to that problem using speech recognition is outlined. The solution needs to be tested, but is not the task defined for this research effort. The possible usage of speech recognition in the medical environment is presented here because it has been the incentive for conducting this research.

B. Conclusion

Several important problems have been solved by the work described in this dissertation. Two methods for representing speech in a digital computer have been compared. The method with the best performance characteristics was chosen as the basis for a speech recognition system. An actual system was built using an innovative design and a new technology, the switched capacitor, so that the final system could be implemented in a low-cost integrated circuit chip set. The combination of the system design and algorithms for speech recognition was tested against real speech data.

The problem of which method for representing speech was solved. The problem of how to design the system for low-cost implementation has been solved. And the question of whether the system can perform with a high degree of accuracy has been answered in the affirmative when a vocabulary of three repetitions of the digits is used.

C. Future Work

There are at least three major areas where efforts could be profitably made. These three areas are feature transformation, feature reduction, and integrated circuit fabrication.

A function for feature transformation could be developed to transform the nineteen filter energies plus the total energy into a more useful form. It seems highly probable that the importance of each filter is not equivalent and some form of weighting function needs to be developed. The affect of the weighting function must be determined by recognition scores.

The current experimental system does no feature reduction. Data is collected at the rate of 2,000 bytes per second. But, due to the highly redundant nature of speech, most of the data collected is unnecessary. A more compact representation would save both space and processing time.

A third effort should be made to fabricate a single integrated circuit to perform the 19-bandpass filter function which is currently implemented on three dozen integrated circuits of medium scale integration. The possibility of putting the whole design into a single Large Scale Integrated circuit should be seriously pursued.

References

[1] J. L. Flanagan, Speech Analysis, Synthesis and Perception. New York: Springer, 1972

[2] C. G. M. Fant, Speech Sounds and Features. Cambridge, MA: MIT. Press, 1973

[3] I. Lehiste, Suprasegmentals. Cambridge, MA: MIT Press, 1970.

[4] T. B. Martin, "Practical Applications of Voice Input to Machine," IEEE Proc., pp. 487-501, 1976

[5] S. Itahashi, S. Makino, and K. Kido, "Discrete-word recognition utilizing a word dictionary and phonological rules," IEEE Trans. Audio Electroacoust., vol. AU-21, pp. 239-249, June 1973.

[6] P. Vicens, "Aspects of speech recognition by computer," PhD. dissertation, Comput. Scie. Dep., Stanford Univ., Stanford, Ca, 1969.

[7] H. G. Goldberg, "Segmentation and labeling of speech: A comparative performance evaluation," PhD dissertation, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pa, 1975.

[8] F. Itakura, "Minimum prediction residual principle applied to speech recognition," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-23, pp. 67-72, Feb. 1975

[9] G. M. White and R. B. Neely, "Speech recognition experiments with linear prediction, bandpass filtering, and dynamic programming," in Proc, 2nd USA-Japan Computer Conf., Tokyo Japan, Aug. 1975.

[10] D. R. Reddy, "Speech recognition by machine: a review", IEEE Proc., vol 64, pp. 501-531, April 1976

[11] B. M. White, "Speech recognition: a tutorial overview," Computer, pp. 40-53, May 1976

[12] M. F. Plute and R. B. Cox, "Voice programming - A new dimension to NC," in NCS Proc. Tech Conf. 1975 -NC/CAM EXPO '75 (Wash., DC), pp. 89-97, May 1975.

[13] T. B. Martin and E. F. Grunza, "Voice recognition applied to problems of quality control," in 1975 ASQC Tech. Conf. Trans.(San Diego, CA) pp. 8-15, May 1975.

[14] "Spoken words drive a computer," Business Week, New York: McGraw-Hill, Dec. 2, 1972, pp. 36H-36L.

[15] "Tell your cartons where to get off!" in Traffic Management, Feb. 1975, pp. 24-27.

[16] R. Turn, A. Hoffman, and T. Lippiatt, "Military applications of speech understanding systems," Rep. R-1434-ARPA, June 1974 (available from ARPA under Order 189-1).

[17] R. B. Ochsman and A. Chapanis, "The effects of 10 communication modes on the behavior of teams during co-operative problem- solving," Int. J. Man-Machine Studies, vol 6, pp. 579-619, 1974

[18] J. R. Pierce, "Wither speech recognition?" J. Acoust. Soc. Amer., vol. 46, no. 4, pp. 1049-1051, 1969

[19] N. Lindgren, "Speech-Man's natural communication," IEEE Spectrum, vol. 4, pp. 75-86, June 1967

[20] W. A. Lea, "Establishing the value of voice communication with computers," IEEE Trans Audio Electroacoust., vol AU-16 pp. 184-197, June 1968

[21] K. H. Davis, R. Biddulh, and S. Balashek, "Automatic recognition of spoken digits," J. Acoust. Soc. Amer., vol 24, p. 637, 1952

[22] H. Dudley and S. Balashek, "Automatic recognition of phonetic patterns in speech," J. Acoust. Soc. Amer., vol 30, pp. 721 - 732, 1958

[23] P. Denes and M. V. Mathews, "Spoken digit recognition using time-frequency pattern matching," J. Acoust. Soc. Amer., vol 32, pp. 1450-1455, 1960

[24] P. N. Sholtz and R. Bakis, "Spoken digit recognition using vowel-consonant segmentation," J. Acoust. Soc. Amer., vol 34, pp. 1-5, 1962

[25] J. G. von Keller, "An on-line recognition system for spoken digits," J. Acoust. Soc. Amer., vol 49, no. 4, pp. 1288-1296, 1971

[26] F. Itakura, "minimum prediction residual applied to speech recognition," IEEE Trans. Acoust. Speech Siegnal Proc., vol. ASSP-23, pp. 67-72, Feb. 1975

[27] J. K. Baker, "The DRAGON system-An overview," IEEE Trans. Acoust., Speech, Signal Proc., vo. ASSP-23, pp. 24-29, Feb. 1975

[28] D. R. Reddy, L.D. Erman, and R. B. Neely, "A model and a system for machine recognition of speech," IEEE Trans. Audio Electroacoust., vol. AU-21, pp. 229-238, June 1973

[29] J.W. Forgie, D.E. Hall, and R.W. Wiesen, "An overview of the Lincoln Laboratory speech recognition system," J. Acoust. Soc. Amer., vol. 56, S27 (A), 1974

[30] F. Jelinek, "Continuous speech recognition by statistical methods," Proc. IEEE, pp. 532-556, April 1976

[31] N.R. Dixon and H.F. Silverman, "A general language-operated decision implementation system (GLODIS): Its application to continuous speech segmentation," IEEE Trans Acoust. Speech Signal Processing, vol. ASSP-24, 1976

[32] ___, "Some encouraging results for general purpose continuous speech recognition," Proc. 1975 Inct. Conf. Cybernetics and Society, San Francisco, Ca., pp. 293-295, 1975

[33] J. Makhoul, "Linear prediction: A tutorial review," Proc. of the IEEE, vol. 63, pp.561-580, Apr. 1975

[34] L.D. Erman, "Overview of the Hearsay speech understanding research," Comput. Sci. Res. Rev., Comput. Sci. Dep., Carnegie-Mellon Univ., Pittsburgh, PA, 1975.

[35] V.R.Lesser, R.D. Fennell, L.d. Erman, and D.R. Reddy, "Organization of the Hearsay-II speech understanding

system," IEEE Trans. Acoust. Speech, Signal Processing, vo. ASSP-23, pp. 11-23, 1975.

[36] W.A. Woods, "Motivation and overview of SPEECHLISS: An experimental prototype for speech understanding research," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-23, pp. 2-10, 1975.

[37] B. Ritea, "Automatic speech understanding systems," in Proc. 11th Annu. IEEE Computer Society Conf., Washington, DC, Sept. 1975.

[38] D.E. Walker, W.H. Paxton, J.J. Robinson, G.G. Hendrix, B. G. Deutsch, and A.E. Robinson, "Speech understanding research annual report," Artificial Intelligence Center, Stanford Res. Inst., Menlo Park, Ca. Project 3804, 1975.

[39] H. Sakoe and S. Chiba, "IEEE Transactions on Acoustics, Speech and Signal Processing," February 1978

[40] R. W. Brodersen, and D. D. Buss, "Applications of CCD and Switched Capacitor Filter Technology," Proceeding of the IEEE, vol. 67, no. 10, Oct. 1979

GLOSSARY

Aliasing-

in signal processing a term used to describe the fact that high-frequency components of a time function can impersonate low frequencies if the sampling rate is to low

Allophones-

a slight variation in the pronunciation of a phoneme

Anti-aliasing filter-

a low pass filter used to prevent aliasing

Bit-

in computer science a single memory cell with one of two states - zero or one

Byte-

a word used in computer science to describe eight bits of binary information

Chip-

another name for an integrated circuit - the name is derived from the thin slice of silicon that makes up the integrated circuit

Continuous Speech Recognizers-

systems that deal with normal speech containing co-articulation and no word boundraries

CRT-

A cathode-ray-tube used to display computer output

Dynamic Programming-

used to achieve nonlinear time adjustment (warping) to align multisyllabic utterances

Floppy Disk-

a storage device used for computer data, the storage media is a thin 8 inch mylar disk which rotates at 60 RPM

Formants-

natural resonances which appear as peaks in amplitude of the vocal cords.

Fricative-

a class of phonemes that are characterized by the turbulent flow of air through the vocal tract such as the "ZZ" sound in Zoo

Isolated Word Recognizers-

systems that recognize a small number of words (less than 1000) and that require at least 0.1 seconds of silence between words

Lexicon-

the sound patterns of words

Linear Predictive Coding-

a form of Wiener filtering used to represent speech in a highly compressed form and also to produce spectral smoothing that clearly reveals formants

Linear Predictive Residual-

the error that remains when a linear predictive filter is applied to a time series representation of speech. Also known as "matched filtering" in pattern recognition.

Markov Models-

used to model time alignments as well as syntactic, semantic, acoustic, and other knowledge domains

Monolithic-

in electronics a complex circuit implemented on a single integrated circuit

Phonation-

sounds produced by the vocal cords

Phonemes-basic sounds which are the building blocks of speech. The English language has been broken down into about 35 phonemes

Phonological Rules-

a set of syntactic rules for phonemes which specify what sequences of phonemes are legal

Plosives-

a sound created by completely closing the vocal tract at some point, allowing the air pressure to build up and then abruptly re-opening the passage

Pragmatics-

sometimes used to label the set of knowledge that is intrinsic to the context of discourse

Pre-emphasis-

in signal processing, the selective amplification of the high frequency component of the signal

Semantics-

the meaning of words and sentences

Semiconductor Mask-

a photographic negative used to make a semiconductor wafer - the negative is typically made of glass and the wafer has several dozen duplicate circuits on it that are later cut apart and mounted in separate packages

Semiotic-

information arising from non-acoustic sources, such as syntactical and semantical information

Syntax-

the grammatical structure of language

Understanding Systems-

systems that attempt to utilize semantic information, information arising from a model of the domain of discourse

Unvoiced Sounds-

sounds produced by a turbulent flow of air caused by a stricture of the vocal tract

Voiced Sounds-

voiced sounds are produced by the vibratory motion of the vocal cords. A clear distinction should be made between voiced sounds and speech.

APPENDIX A

SCHEMATICS OF FILTER BANK HARDWARE

Diagram number one shows the pre-emphasis filters. The output of the pre-emphasis goes to each of the monolithic bandpass filter chips.

The support circuitry for the monolithic filter chips is shown in diagram two. The retification of the signal output from the bandpass filter and the 30 Hz smoothing filter are also shown.

The third diagram shows the clocking used to control the multiplexor and the analog-to-digital converter. The clocking to drive the monolithic filters is indicated by the notation "F1", "F2", etc. "F1" refers to the clocking for monolithic filter number one.

Diagram four shows the analog multiplexor and the analog-to-digital converter.

Diagram five shows the parallel port circuitry used to make the speech information available to the Z80 micro-processor. The S-100 bus is used to transfer the data to the host processor.

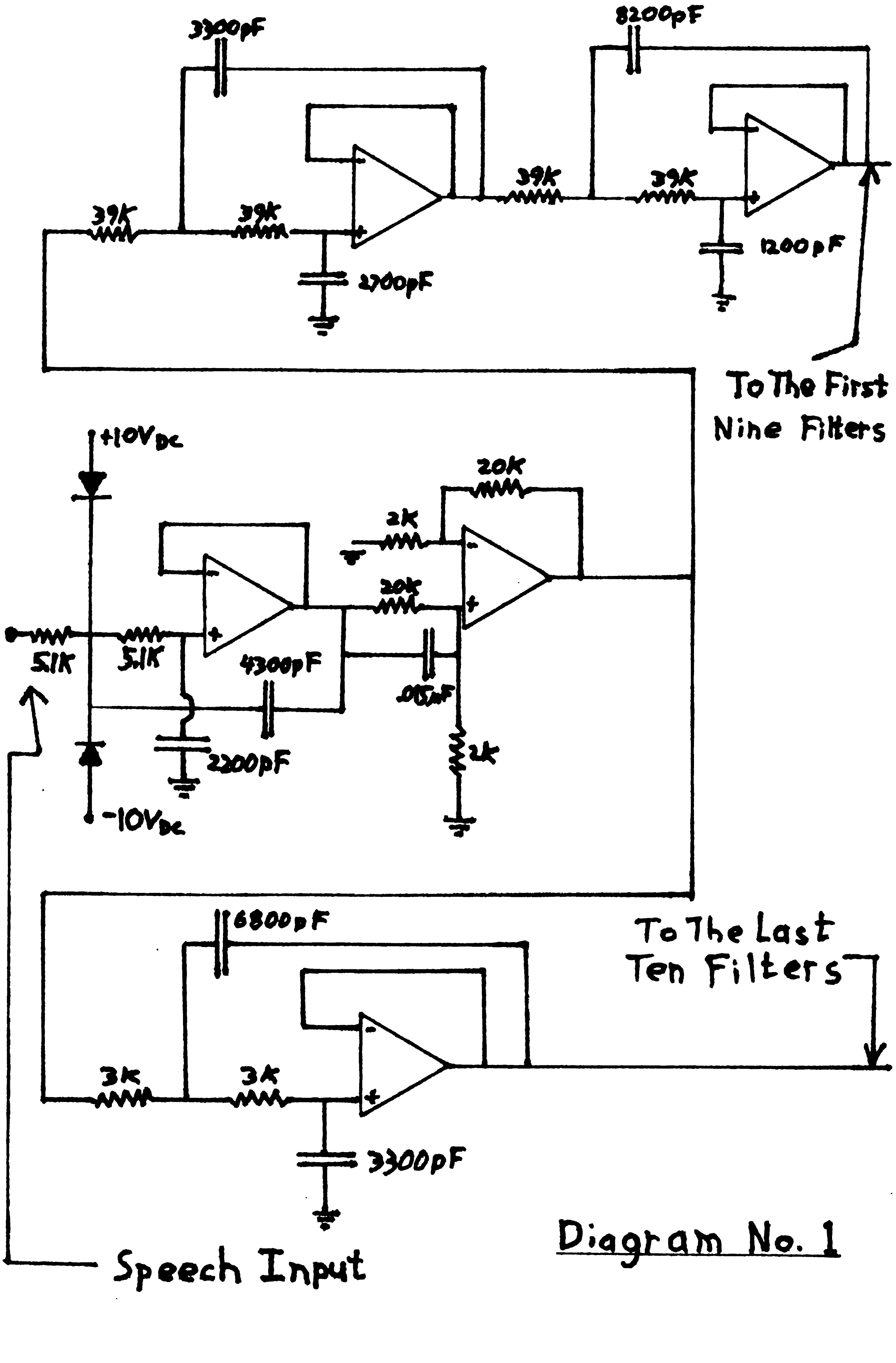

3300pF
8200pF
39K
39K
39K
39K
2700pF
1200pF
To The First
Nine Filters
+10VDC
20K
2K
20K
5.1K
5.1K
4300pF
.015μF
2200pF
2K
-10VDC
6800pF
To The Last
Ten Filters
3K
3K
3300pF
Speech Input

Diagram No. 1

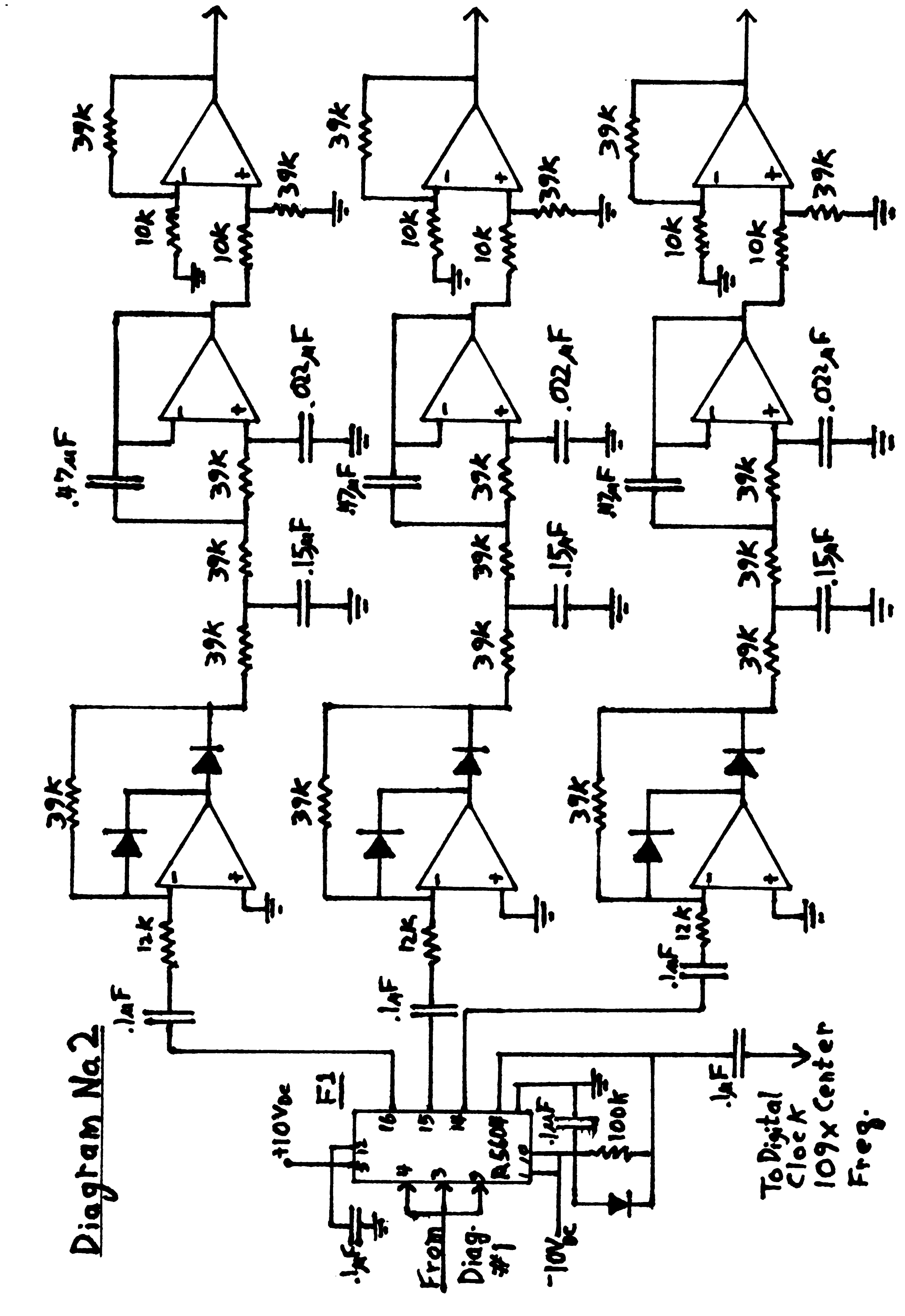


Diagram No. 2
F1
+10Vdc
.1µF
From Diag. #1
-10Vdc
.1µF
100k
.1µF
To Digital Clock 109x Center Freq.
16
15
14
.1µF
12K
39K
39K
39K
.15µF
39K
.47µF
.022µF
10K
10K
39K
39K

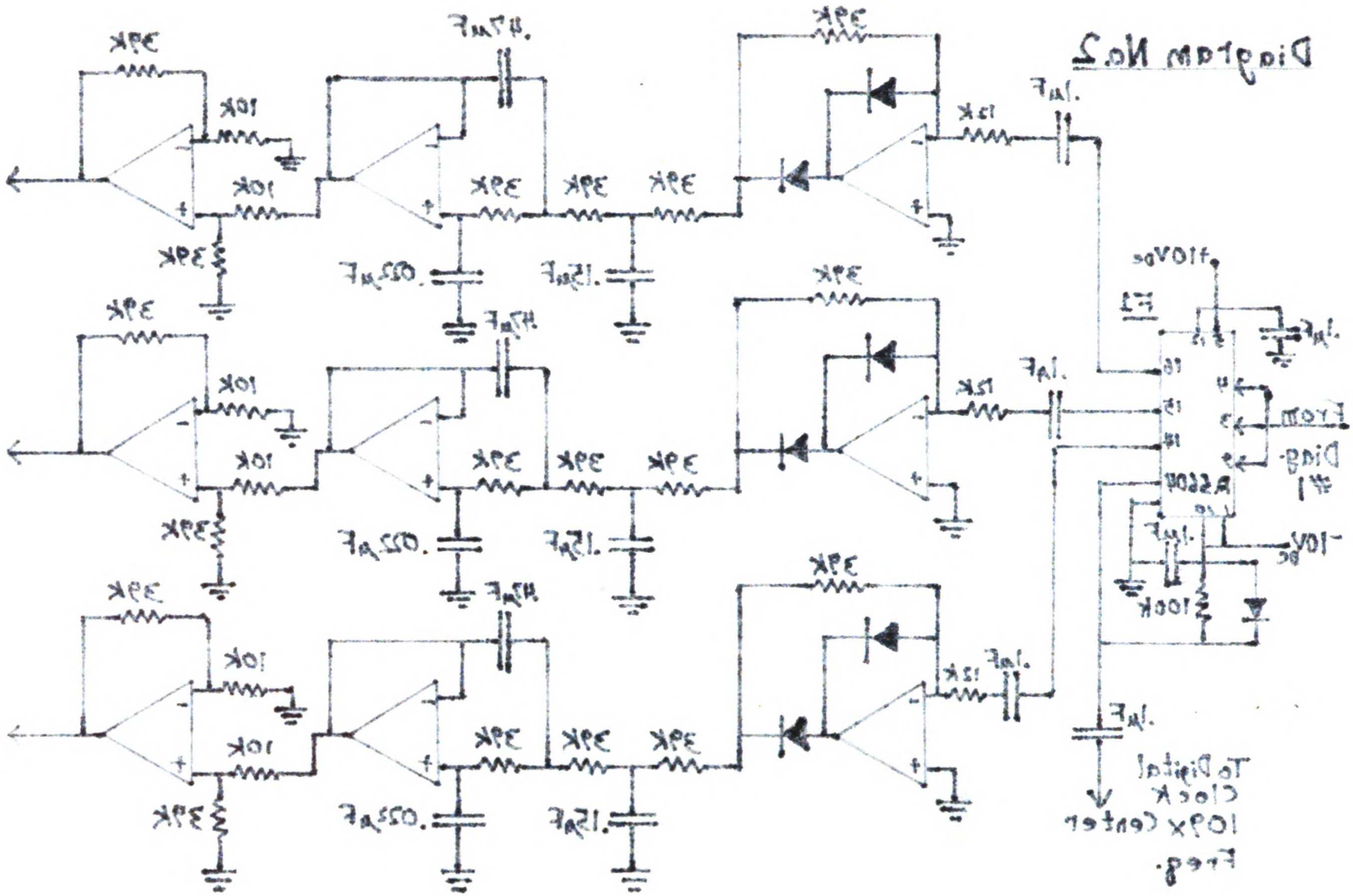


Diagram No2
39K
10K
10K
39K
.47μF
39K
39K
39K
.022μF
.15μF
39K
12K
.1μF
39K
10K
10K
39K
.47μF
39K
39K
39K
.022μF
.15μF
39K
12K
.1μF
39K
10K
10K
39K
.47μF
39K
39K
39K
.022μF
.15μF
39K
12K
.1μF
+10Vdc
E1
.1μF
16
15
14
4
3
2
RA560N
From
Diag.
#1
.1μF
-10Vdc
100K
.1μF
To Digital
Clock
109x Center
Freq.

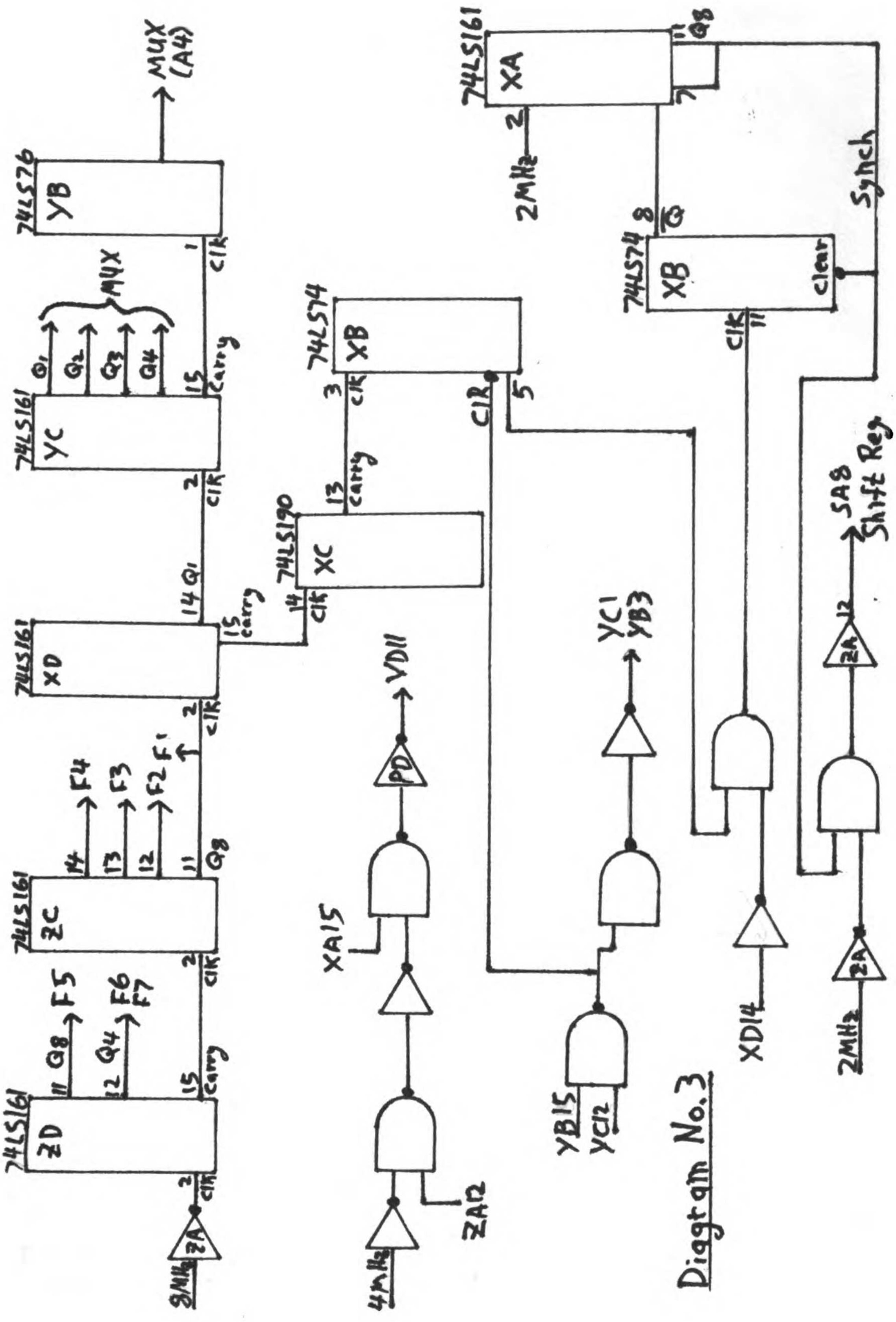

74LS161
ZD
74LS161
ZC
74LS161
XD
74LS161
YC
74LS76
YB
74LS190
XC
74LS74
XB
74LS161
XA
74LS74
XB
8MHz
ZA
2
Clk
11 Q8
F5
12 Q4
F6
F7
15
Carry
14
13
12
11
F4
F3
F2
F1
Q8
14 Q1
15
carry
Clk
Q1
Q2
Q3
Q4
MUX
15
Carry
1
MUX
(A4)
13
carry
3
Clk
5
2MHz
Q8
Q̄
clear
Synch
4MHz
ZA12
XA15
PD
VD11
YB15
YC12
YC1
YB3
XD14
2MHz
ZA
SA8
Shift Reg

Diagram No. 3

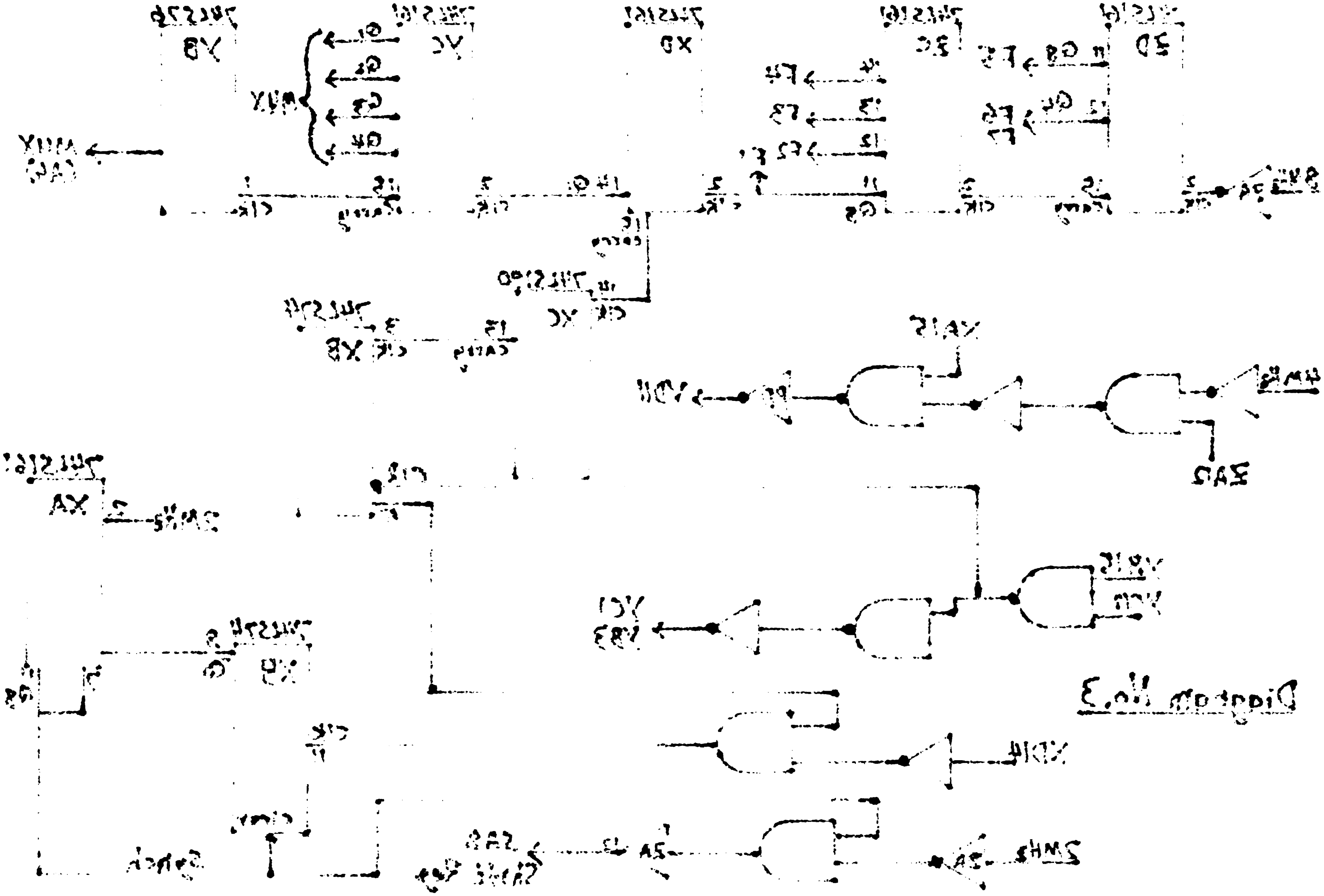


Diagram No. 3

74LS76
YB
74LS161
YC
MUX
XD
74LS161
ZC
74LS161
ZD
F4
F3
F2
F5
F6
F7
G5
G3
G4
MUX
(A4)
carry
clk
74LS160
clk XC
74LS76
clk XB
carry
XA15
4MHz
ZA2
VD11
74LS161
XA
2MHz
74LS76
XB
XB15
XC11
XC1
XB3
XD14
2MHz
2A
ZA2
SAB
Synch

Diagram No. 4

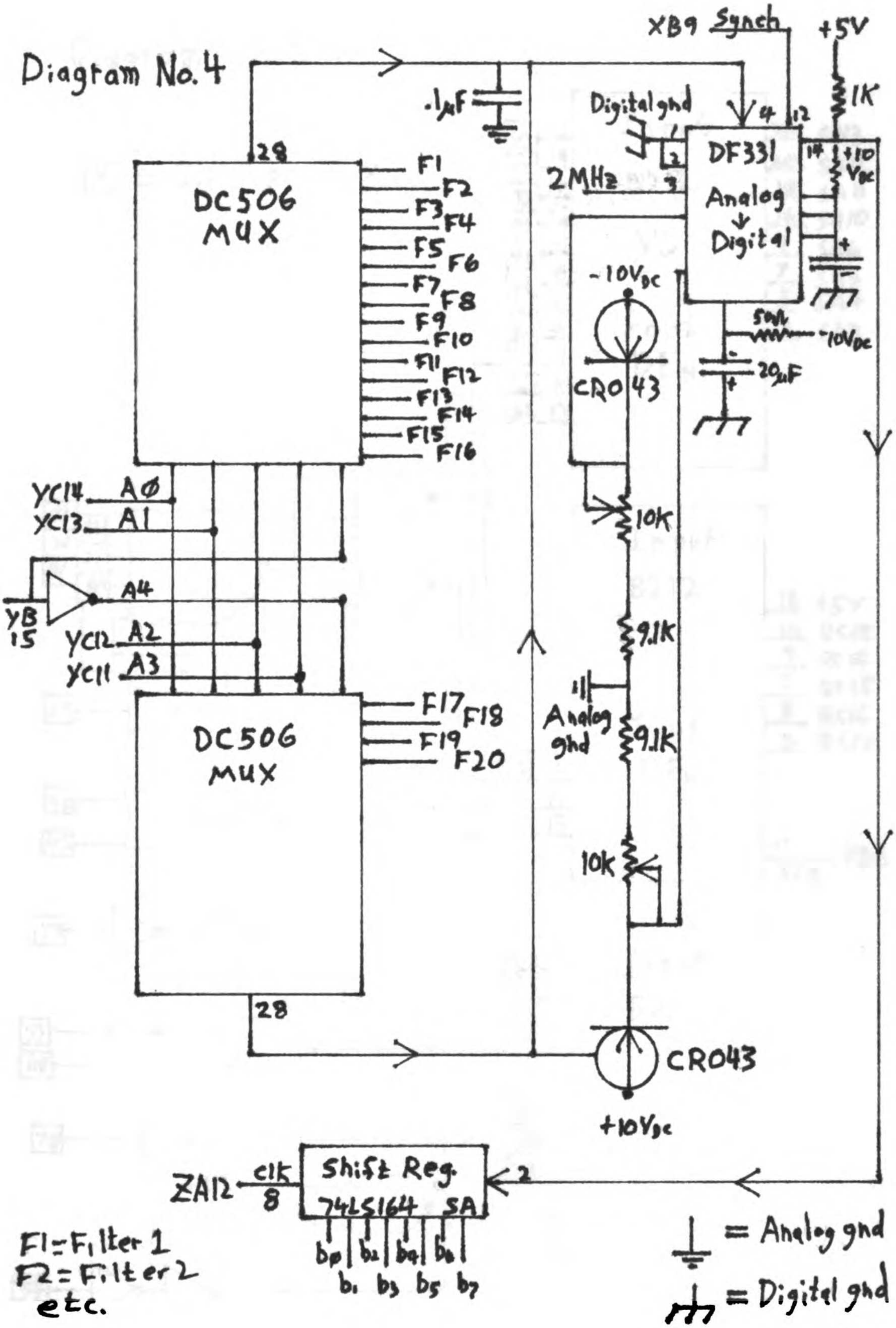
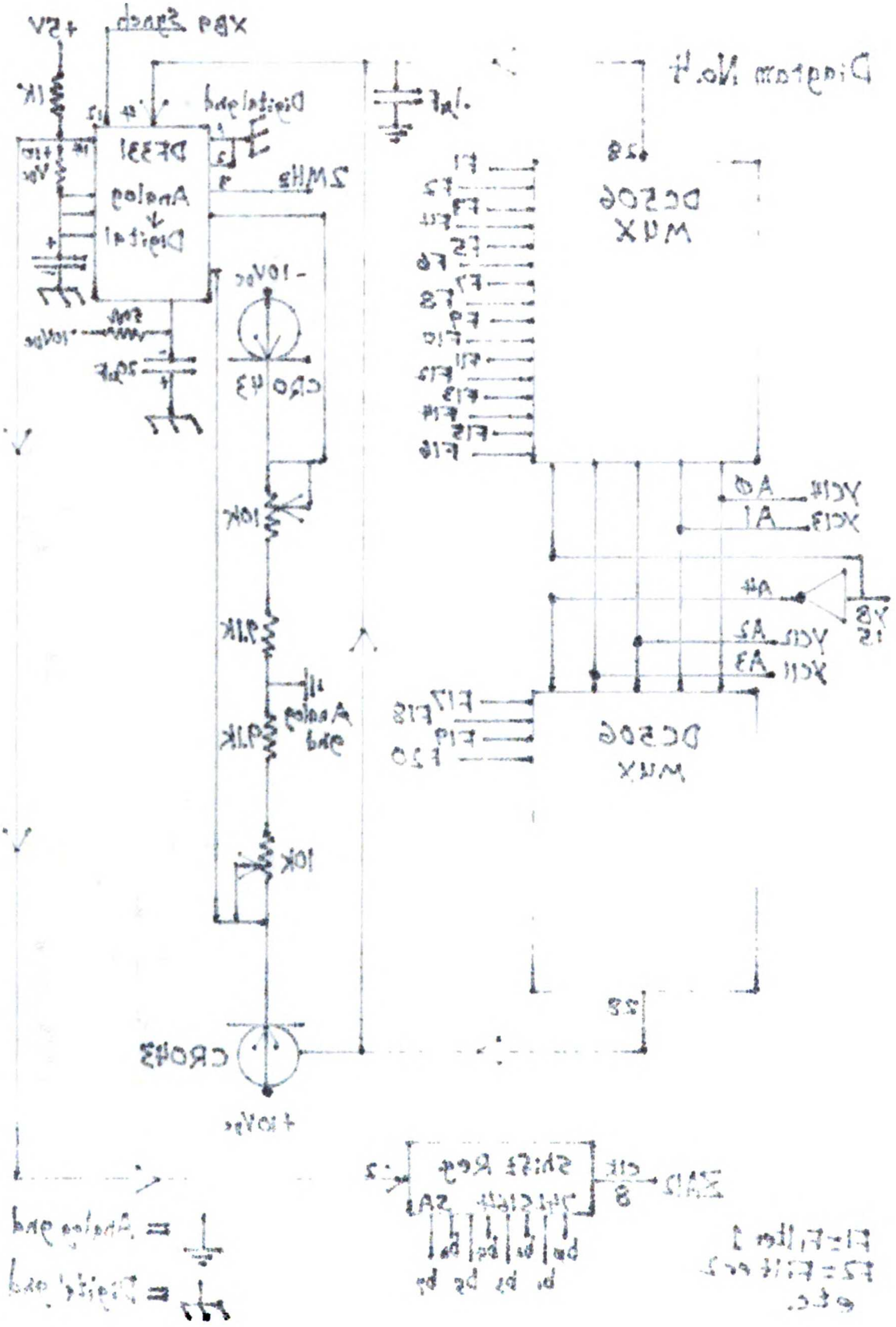

XB9 Synch
+5V
1K
.1µF
Digital gnd
4
12
1
2
3
DF331
Analog
↓
Digital
14
+10 V_DC
+
2MHz
-10V_DC
50K
-10V_DC
-
20µF
+
CRO43
28
DC506
MUX
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
F13
F14
F15
F16
YC14
AØ
YC13
A1
YB 15
A4
YC12
A2
YC11
A3
10K
9.1K
Analog gnd
9.1K
10K
DC506
MUX
F17
F18
F19
F20
28
CRO43
+10V_DC
ZA12
clk
8
Shift Reg.
74LS164
SA
2
b0
b2
b4
b6
b1
b3
b5
b7
F1 = Filter 1
F2 = Filter 2
etc.
= Analog gnd
= Digital gnd

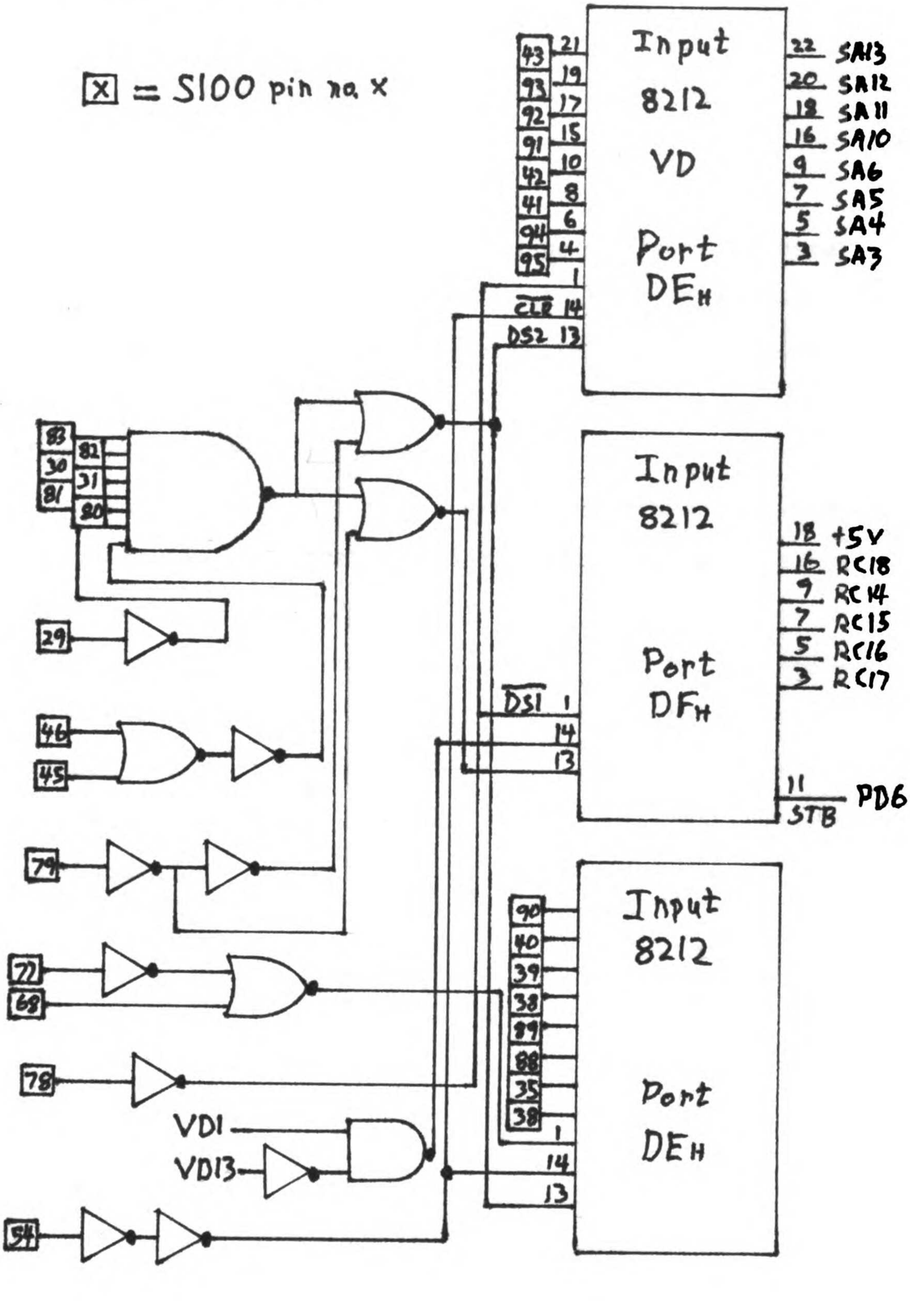
Diagram No.4



+5V
XB4 Synch
1K
DF331
Analog
Digital
Digital gnd
2MHz
.1μf
-10Vdc
CRO43
10K
9.1K
Analog gnd
+10Vdc
DC506
MUX
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
F13
F14
F15
F16
F17
F18
F19
F20
A0
A1
A4
A2
A3
YC14
XC13
YC12
XC11
Y8
15
28
Shift Reg
74LS164
Clk
8
2MHz
SA
b0 b1 b2 b3 b4 b5 b6 b7
= Analog gnd
= Digital gnd
F1=Filter 1
F2=Filter 2
etc.

Diagram No 5

[X] = S100 pin no. x

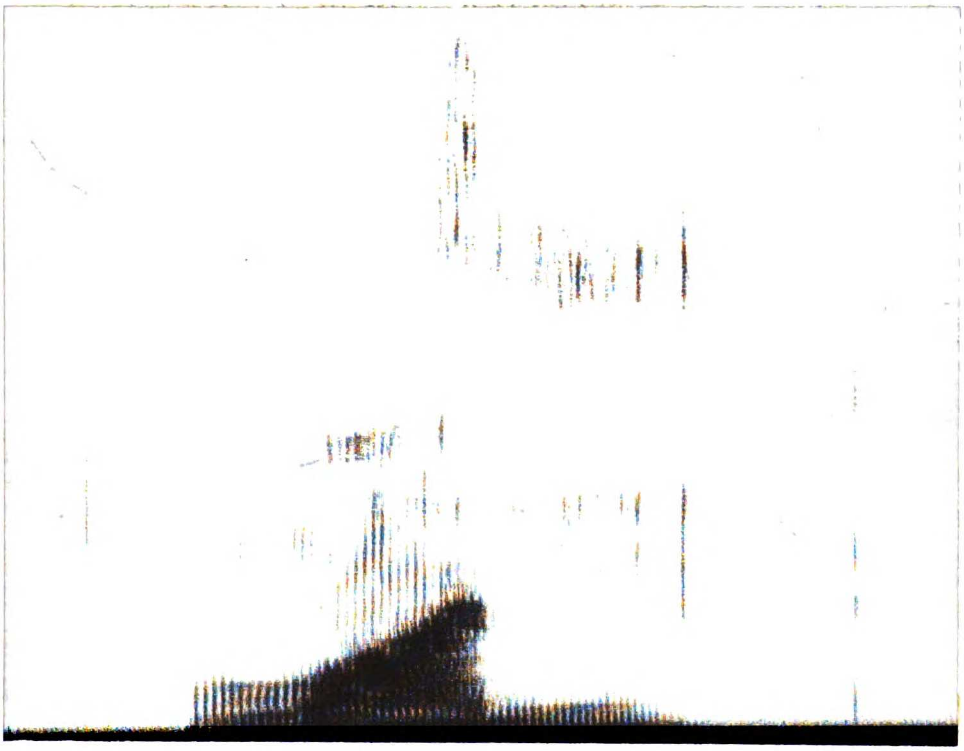


APPENDIX B

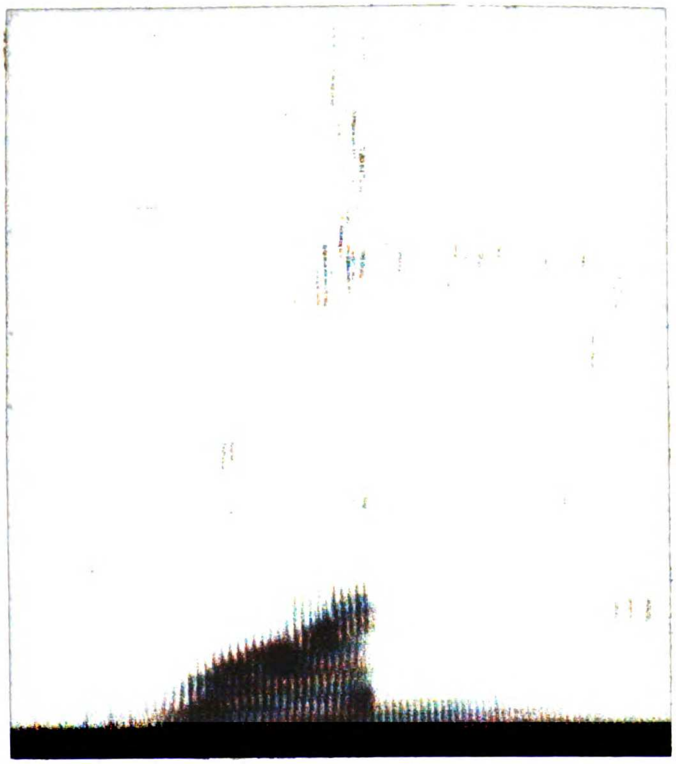
SPECTROGRAMS OF THE ALPHABET AND DIGITS

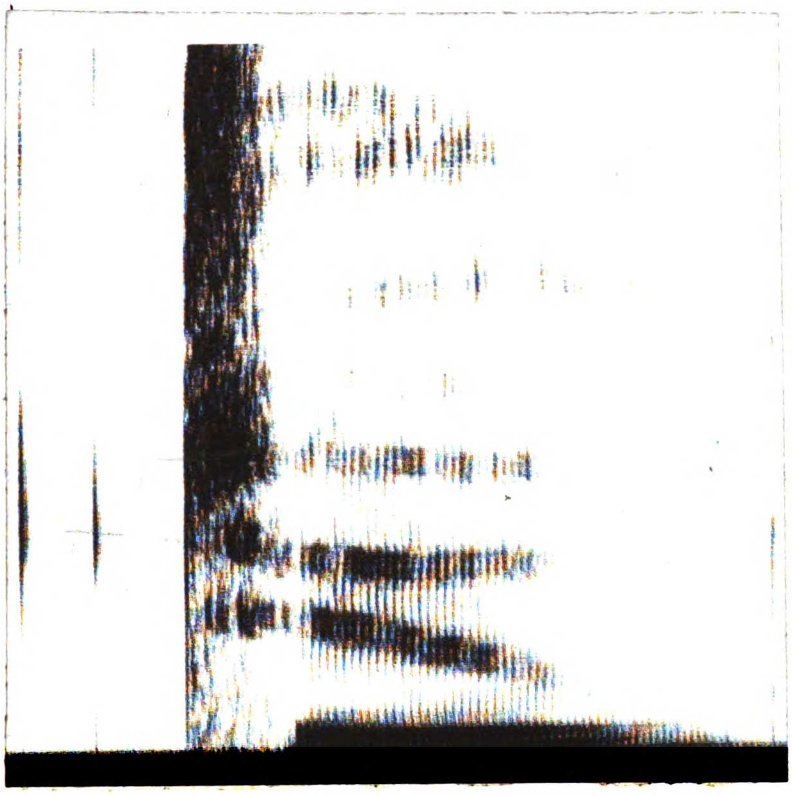
The following seventy-two spectrograms represent two repetitions of the digits and the alphabet. The spectrograms were made from the audio tapes of male speaker number one. Each page contains two spectrograms representing two utterances of the same phrase.

The spectrograms are a three dimensional representation of speech. The horizontal axis represents time. One inch is equivalent to .196 seconds of speech. The vertical axis represents frequency. One inch is aproximately equivalent to 1200 Hz. The third dimension is represented by intensity and corresponds to energy. The darker a location on the spectrogram, the more energy in that frequency and time location.

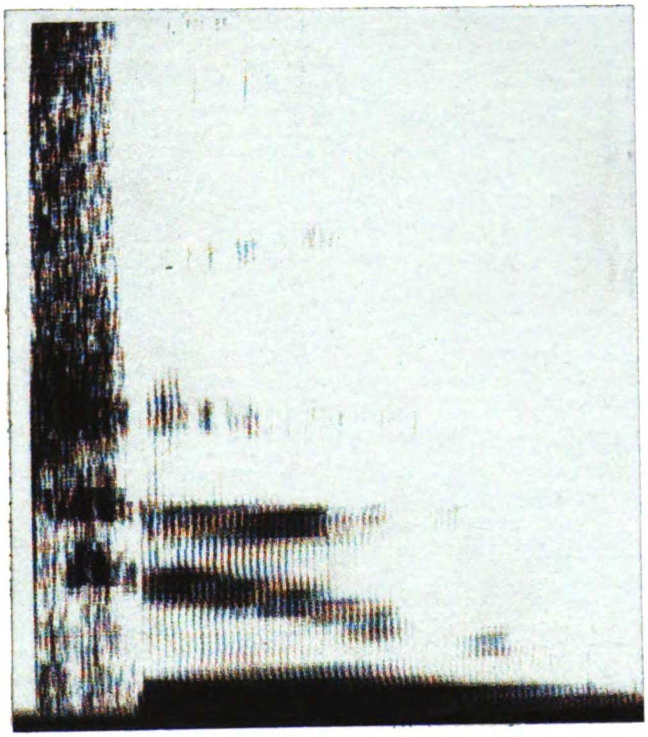


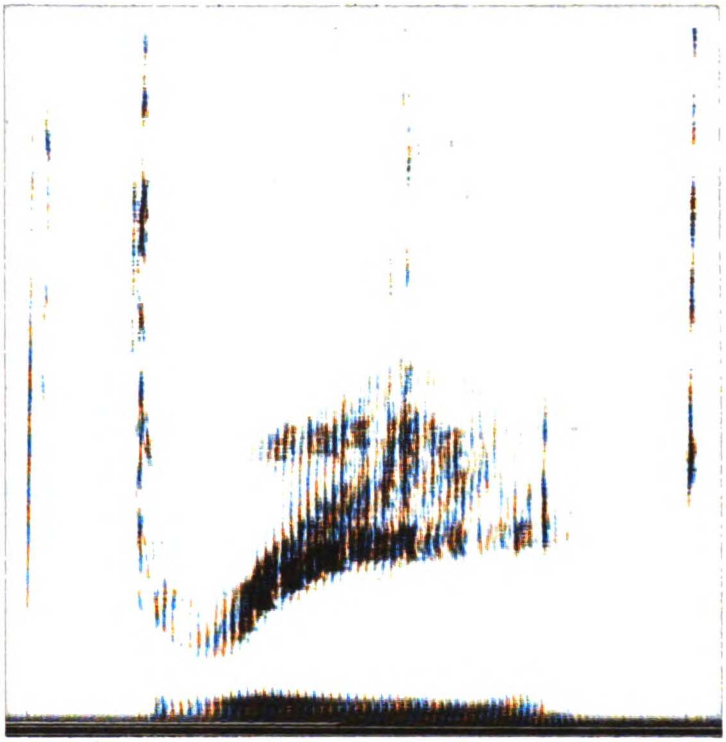
"ONE"



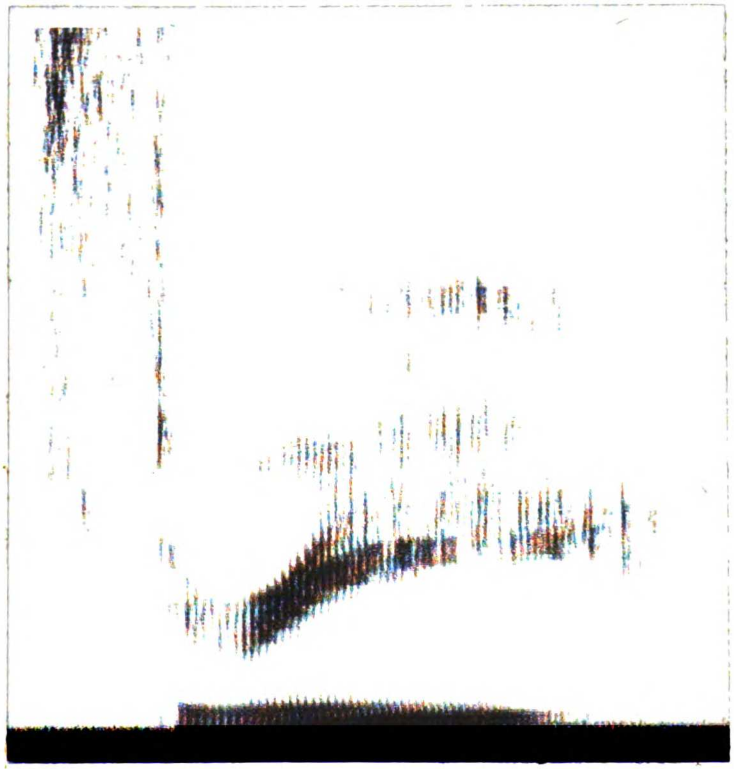


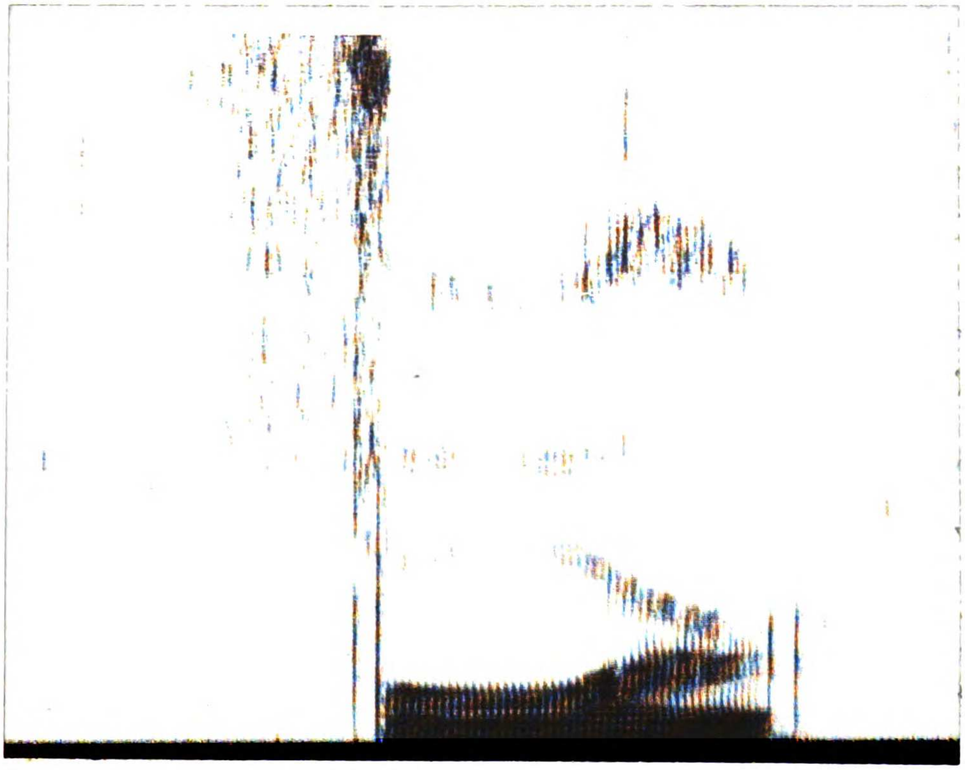
"TWO"



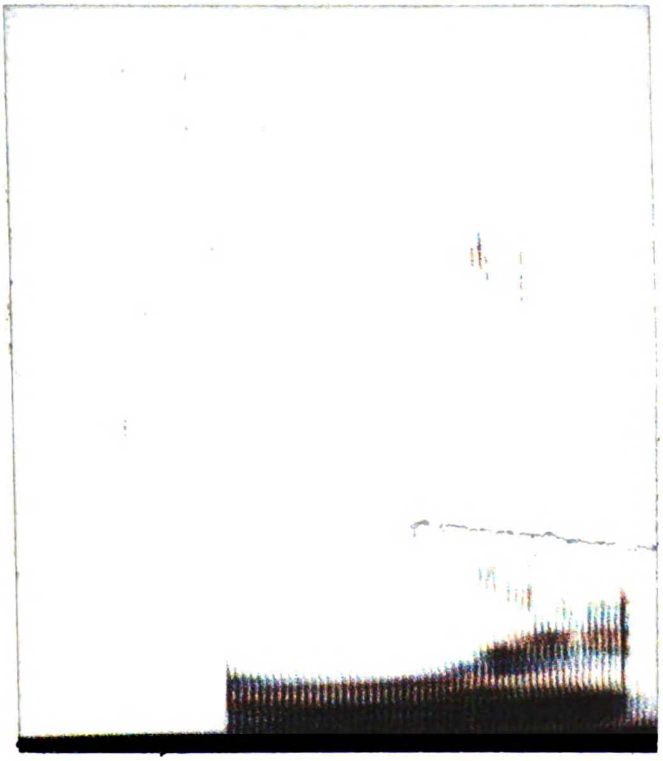


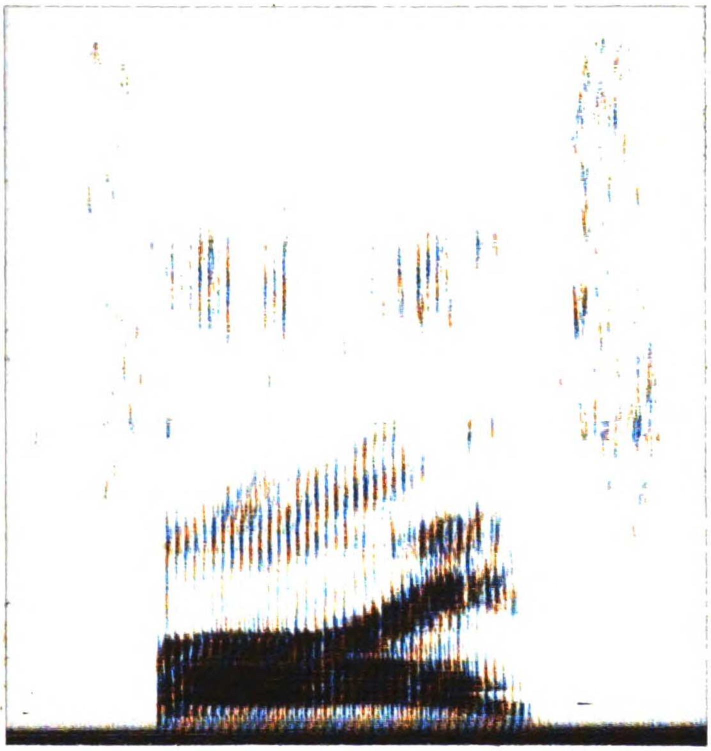
"THREE"



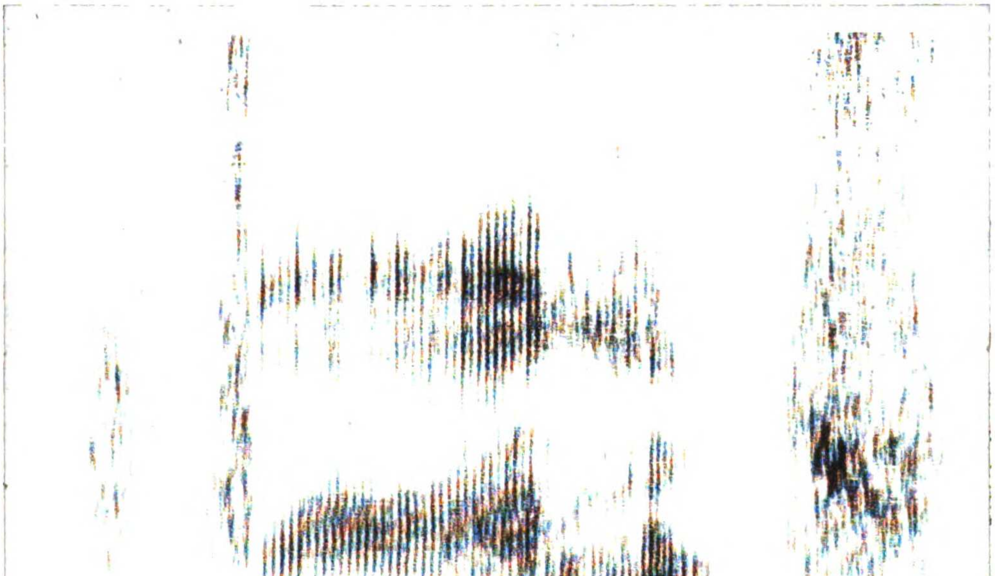


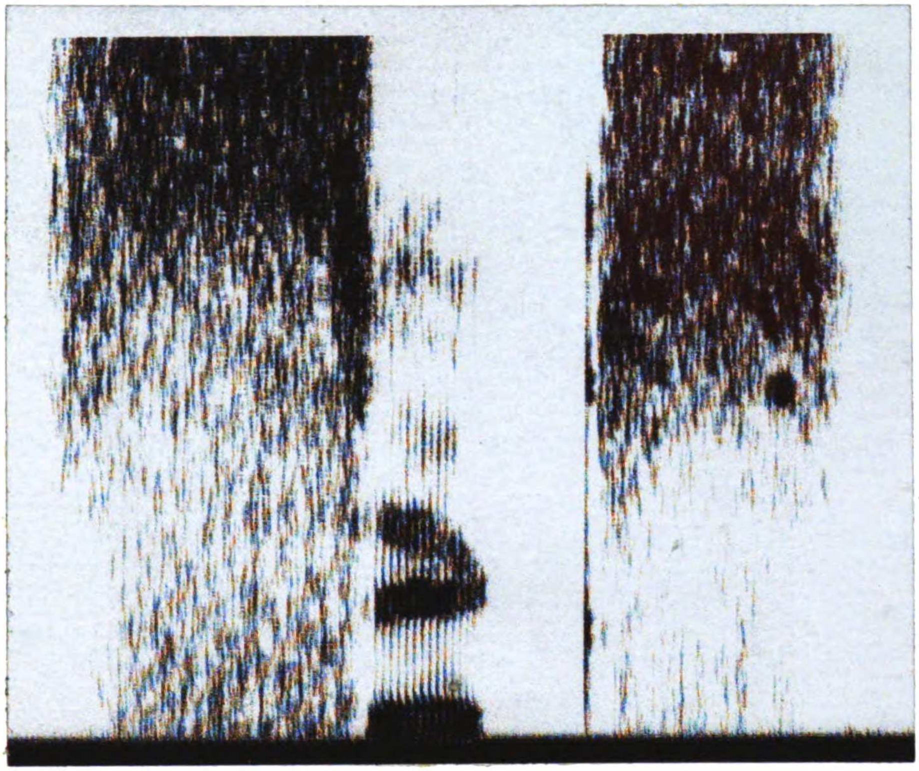
"FOUR"



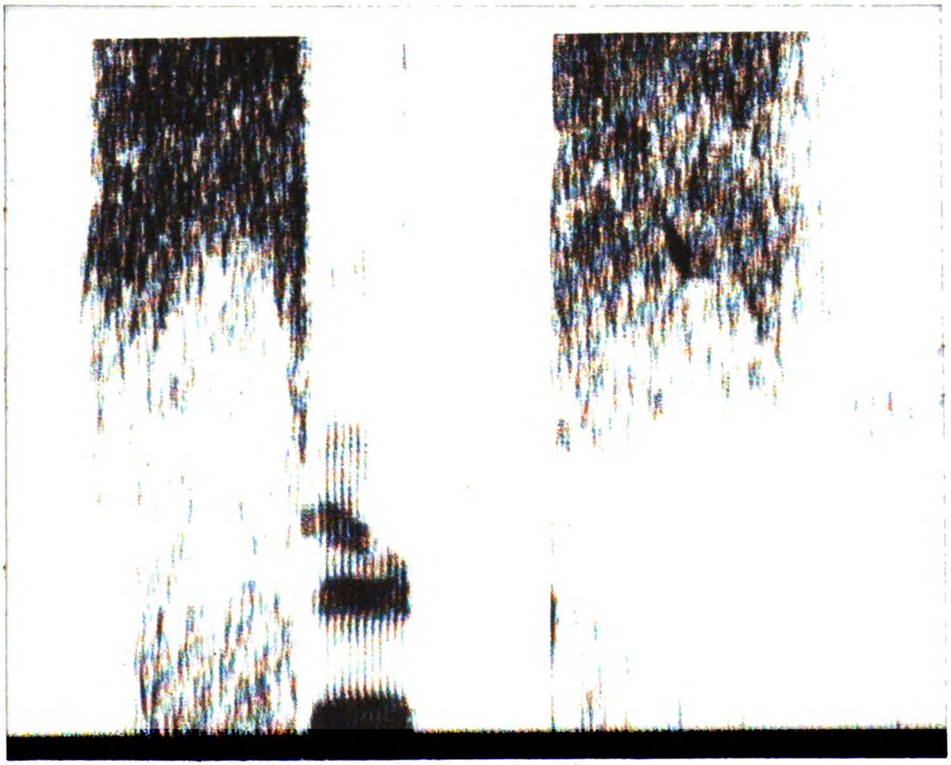


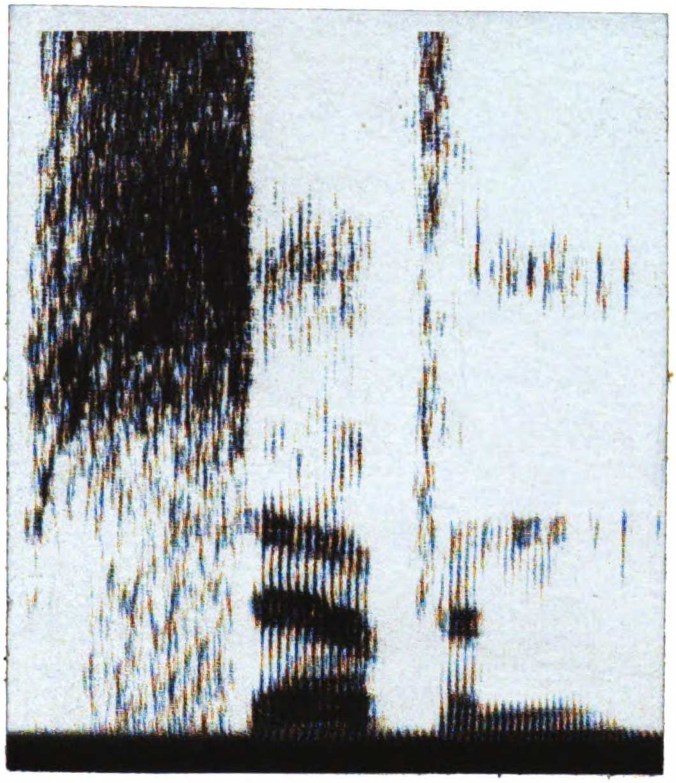
"FIVE"



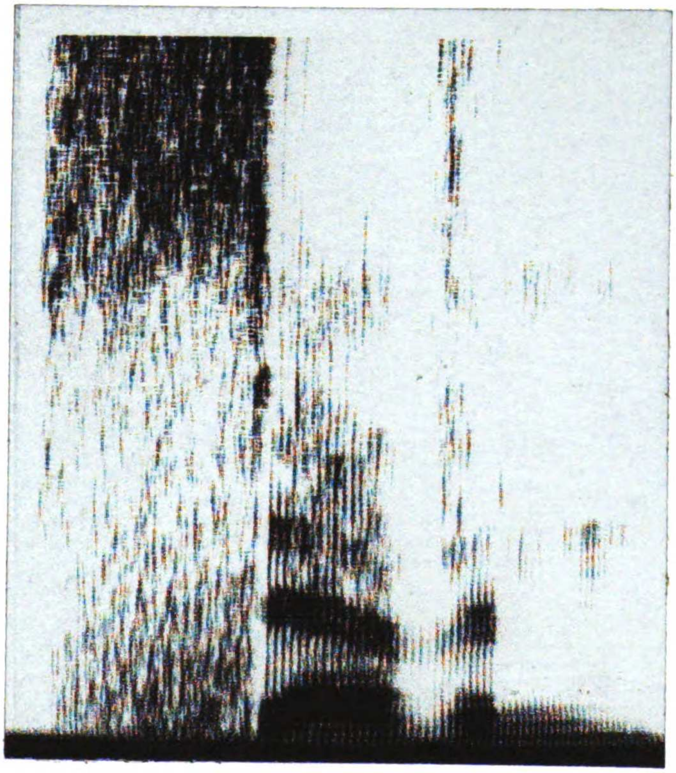


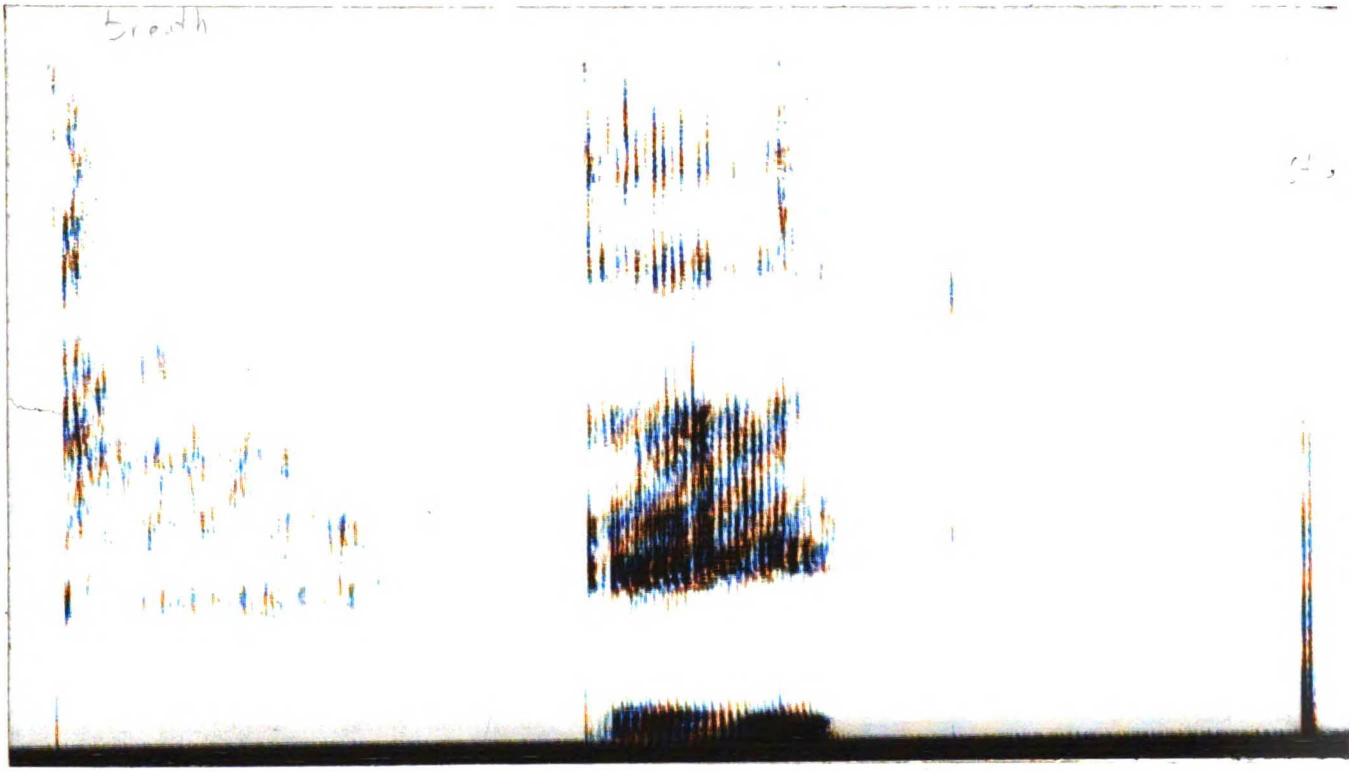
"SIX"



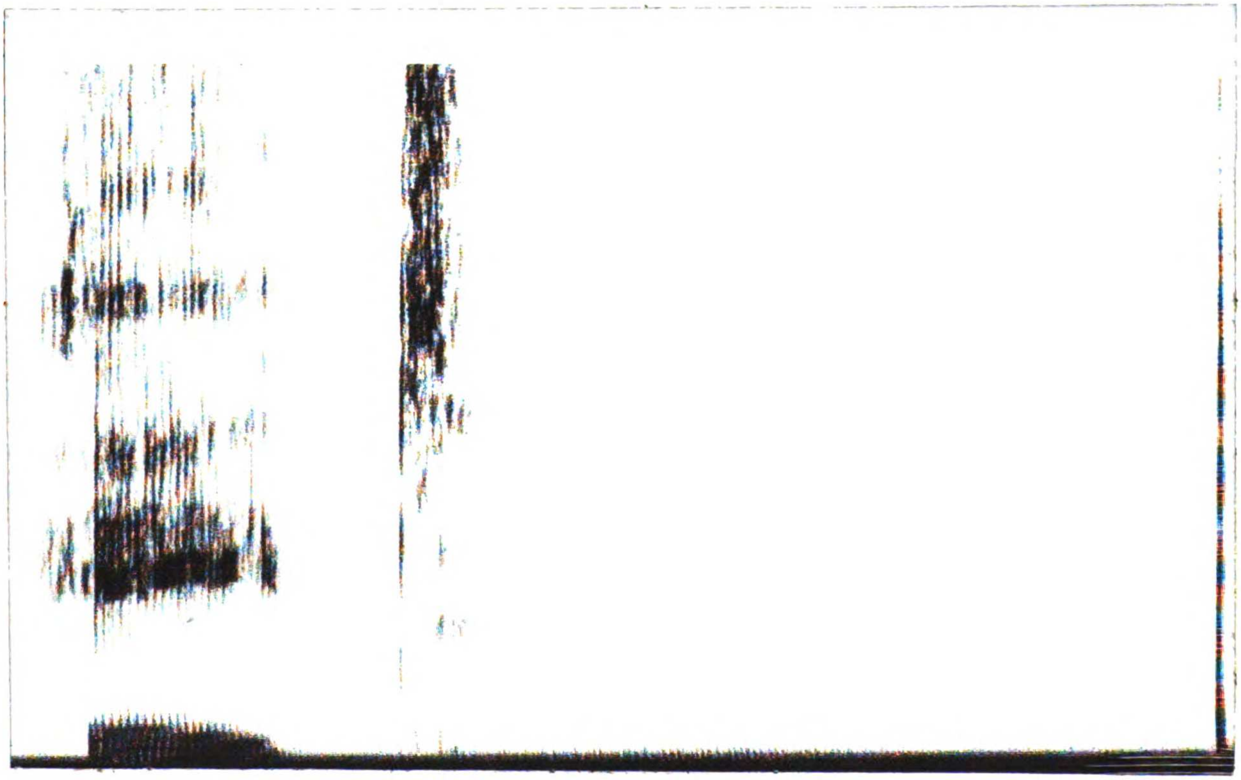


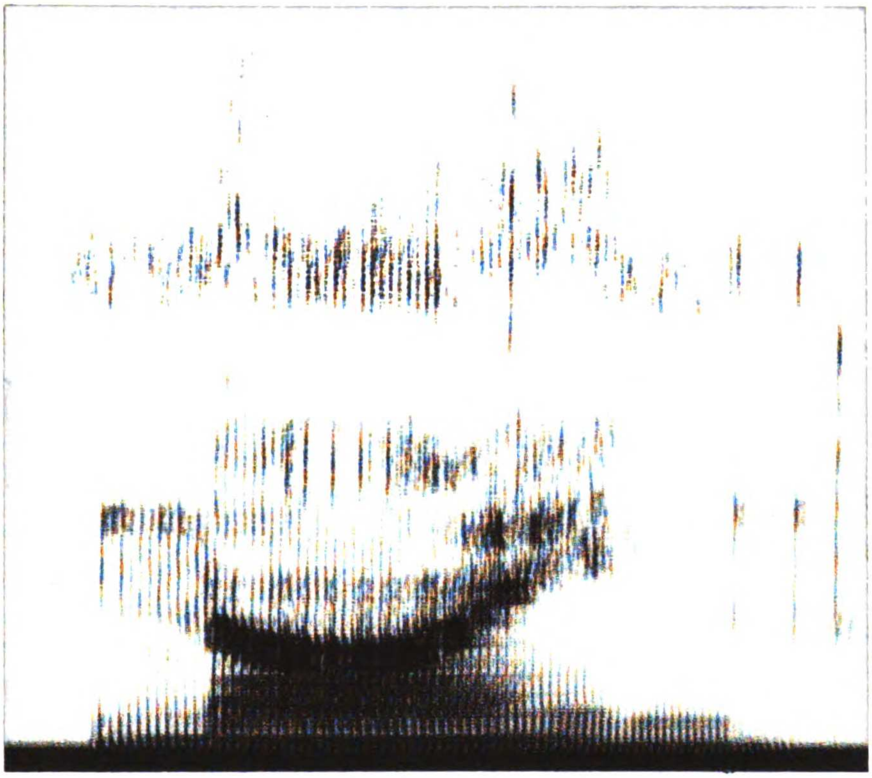
"SEVEN"



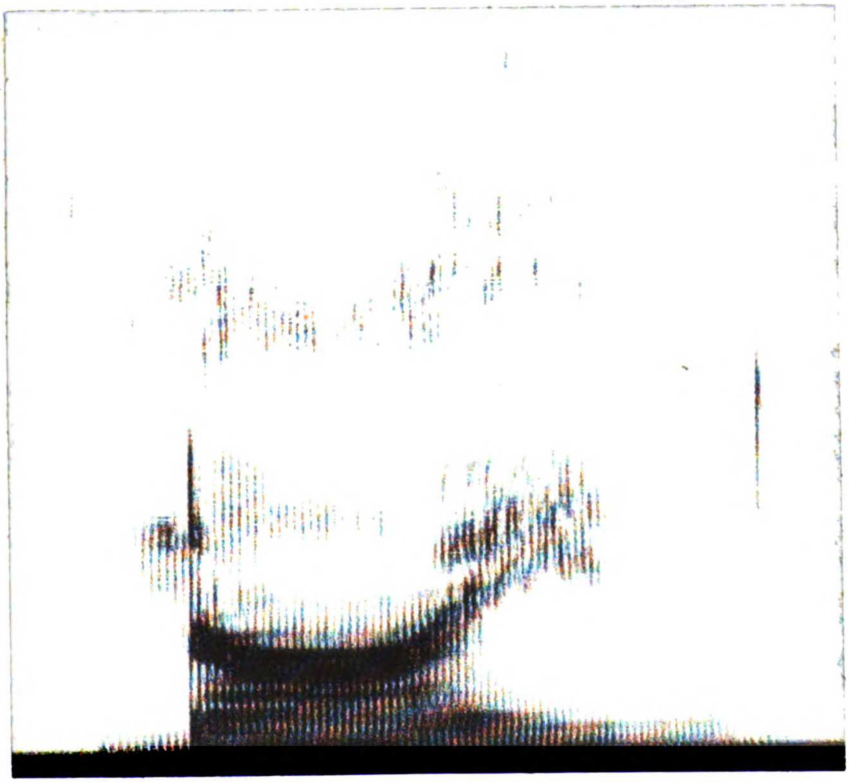


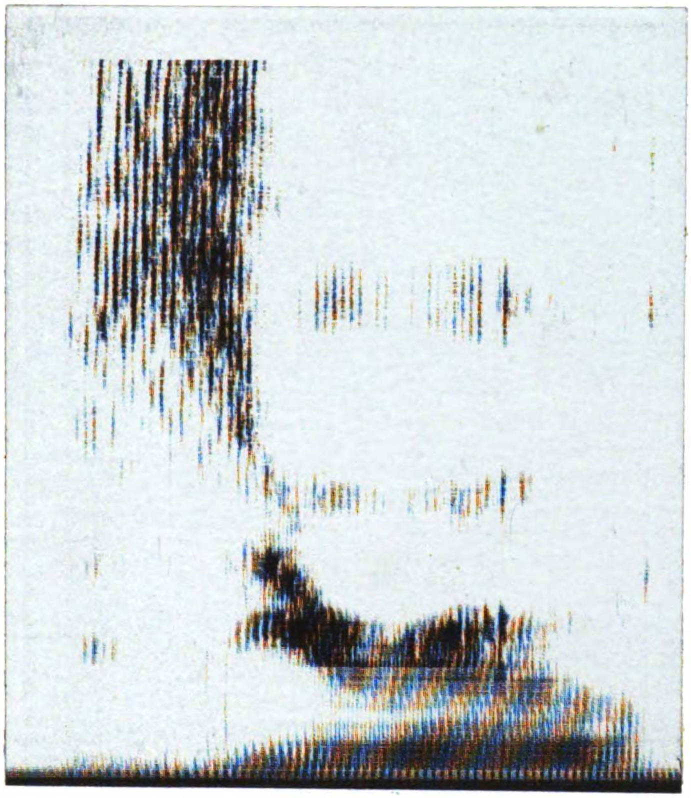
"EIGHT"



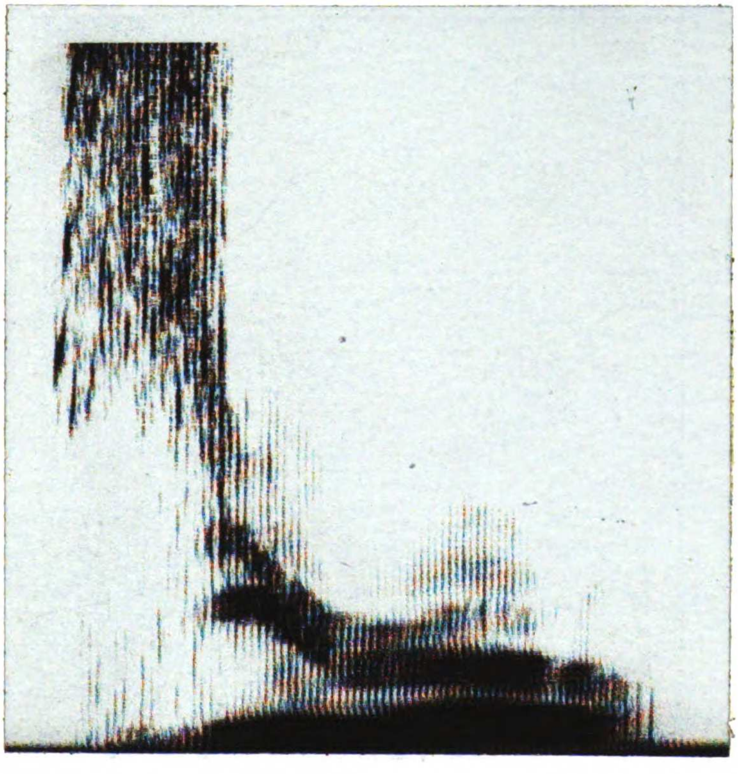


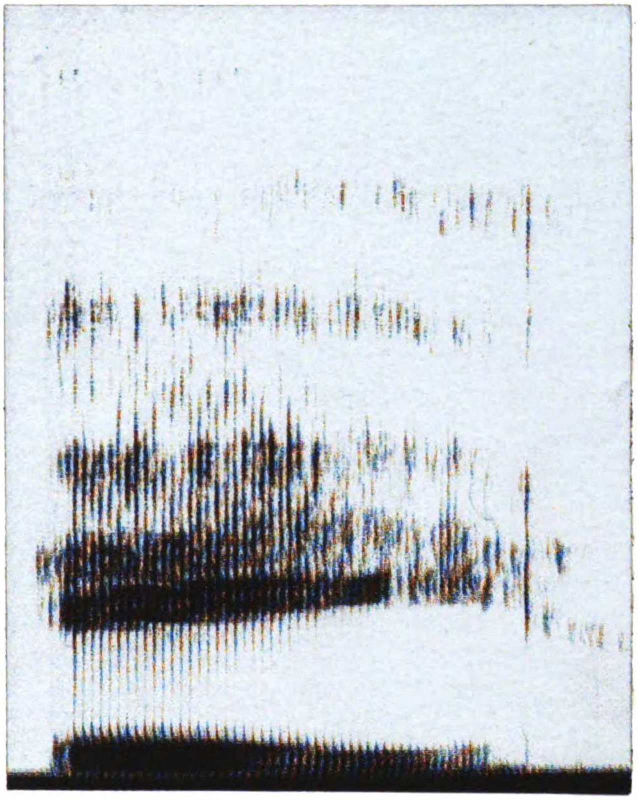
"NINE"



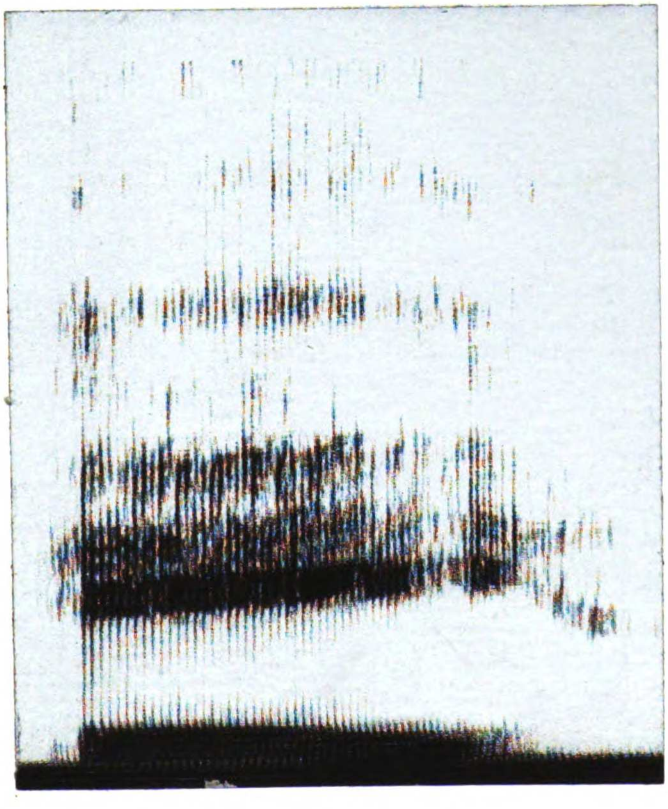


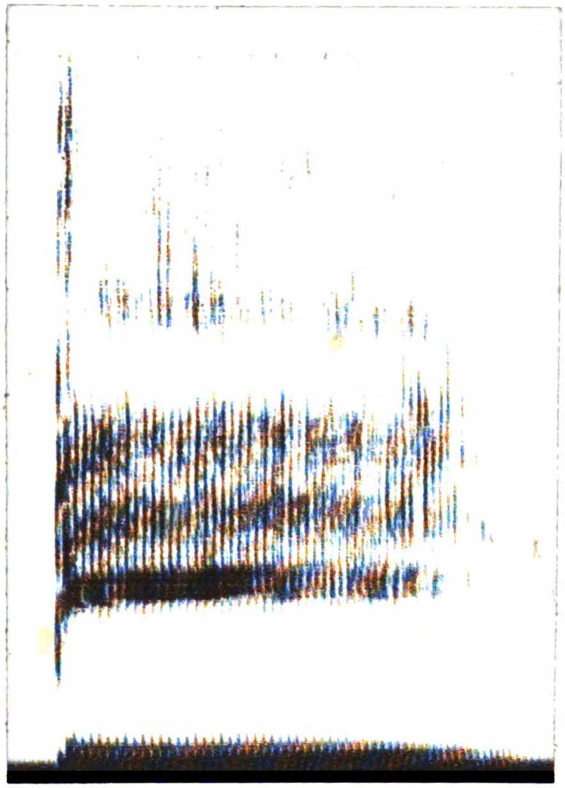
"ZERO"



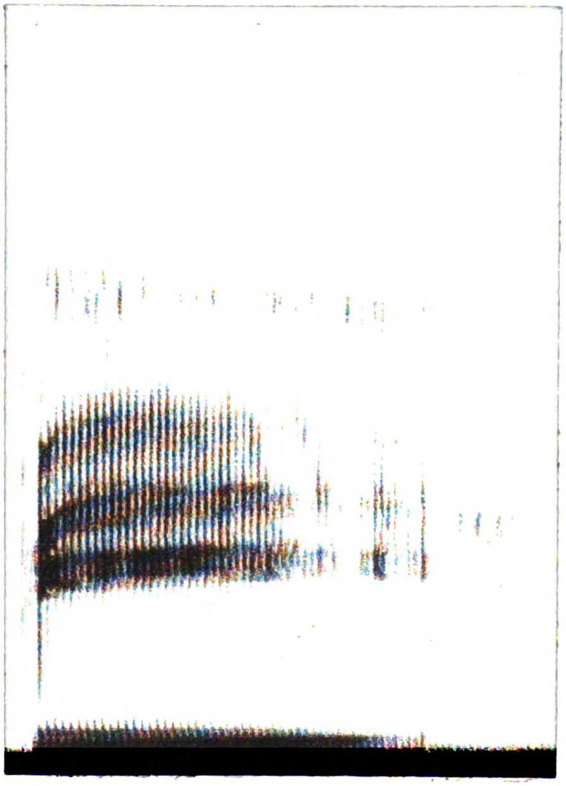


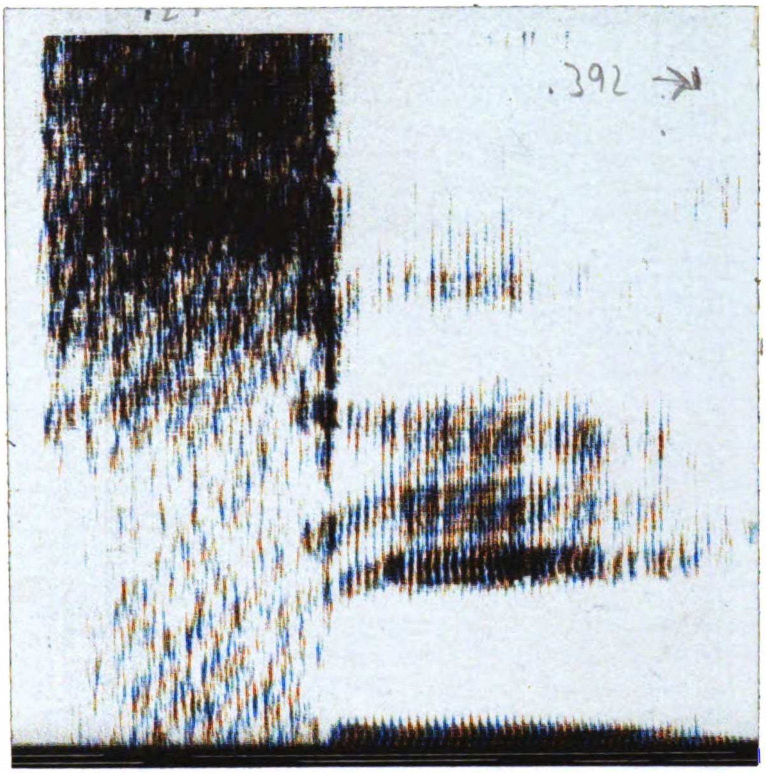
"A"



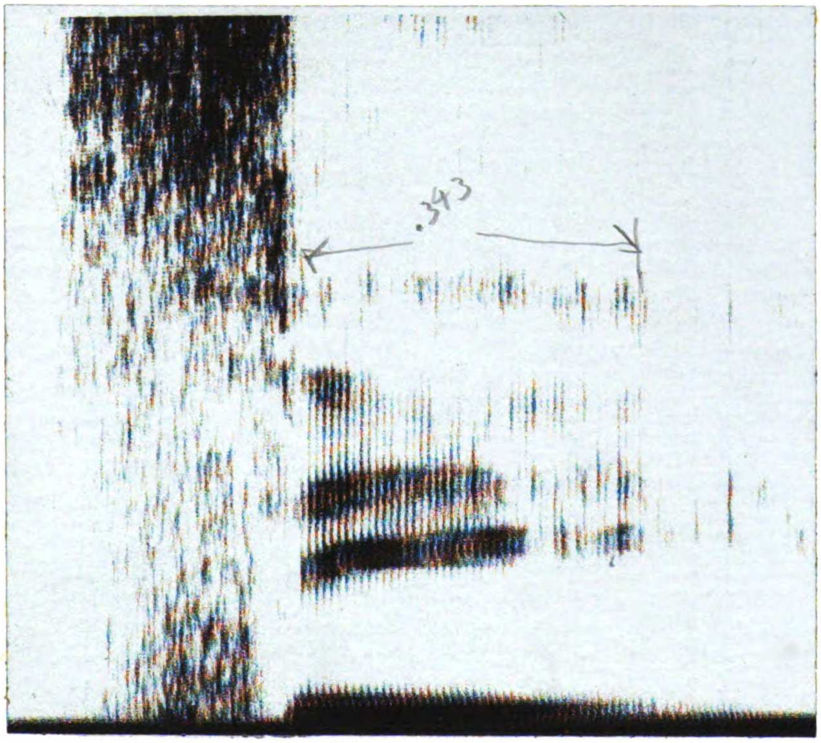


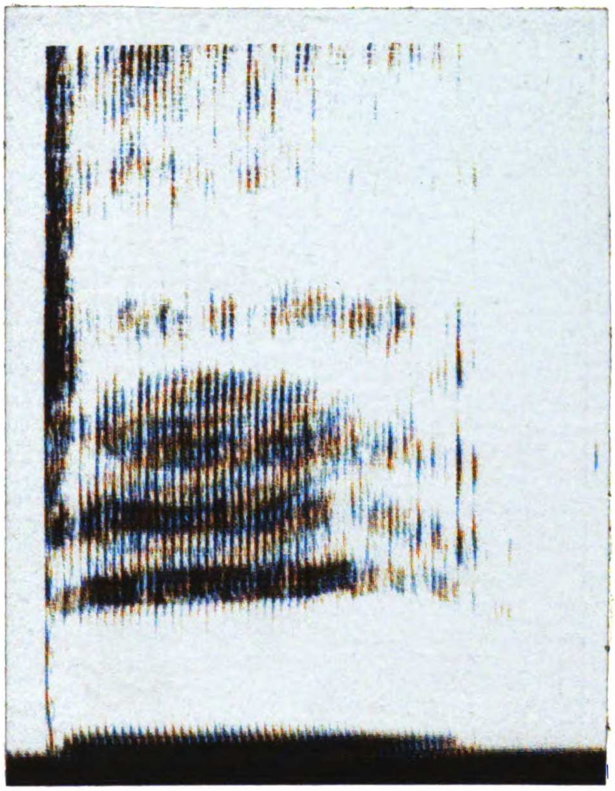
"B"



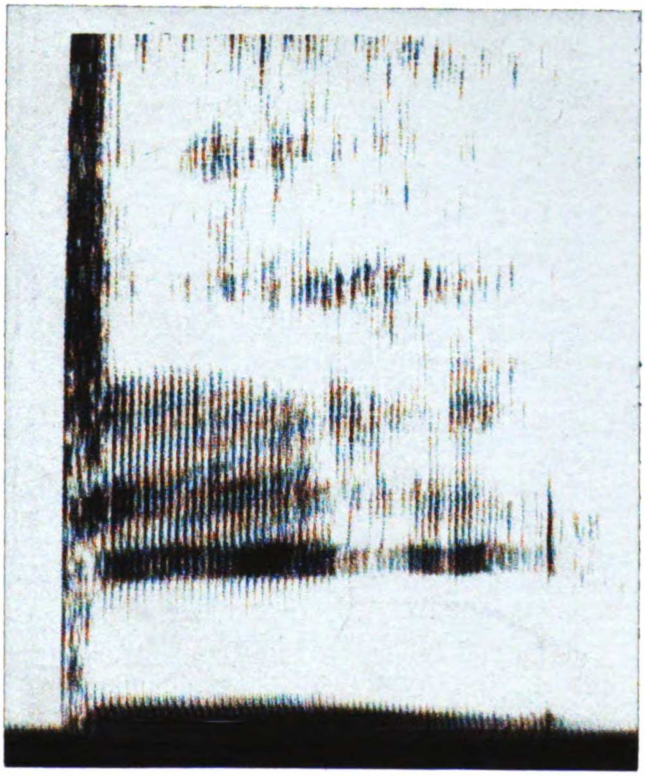


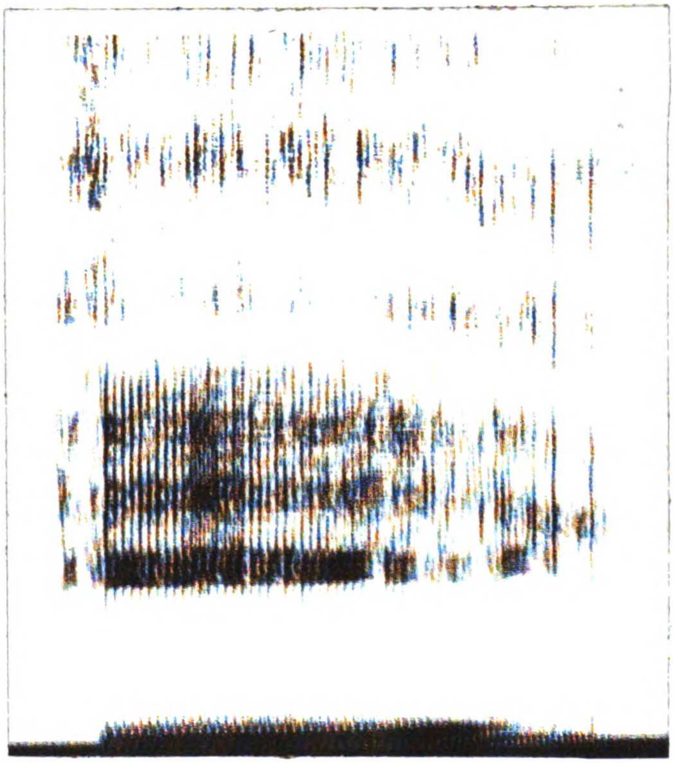
"C"



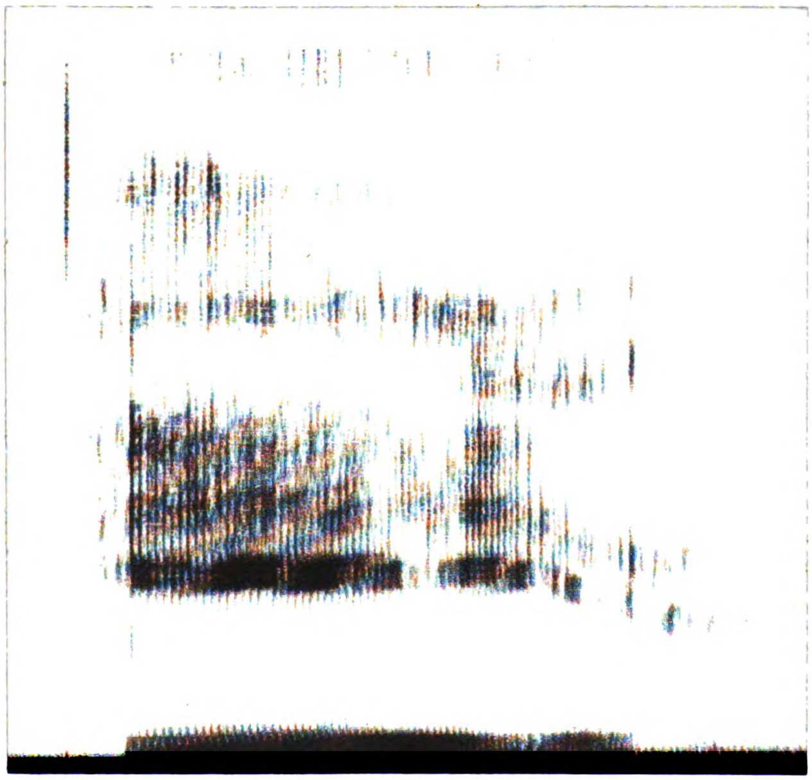


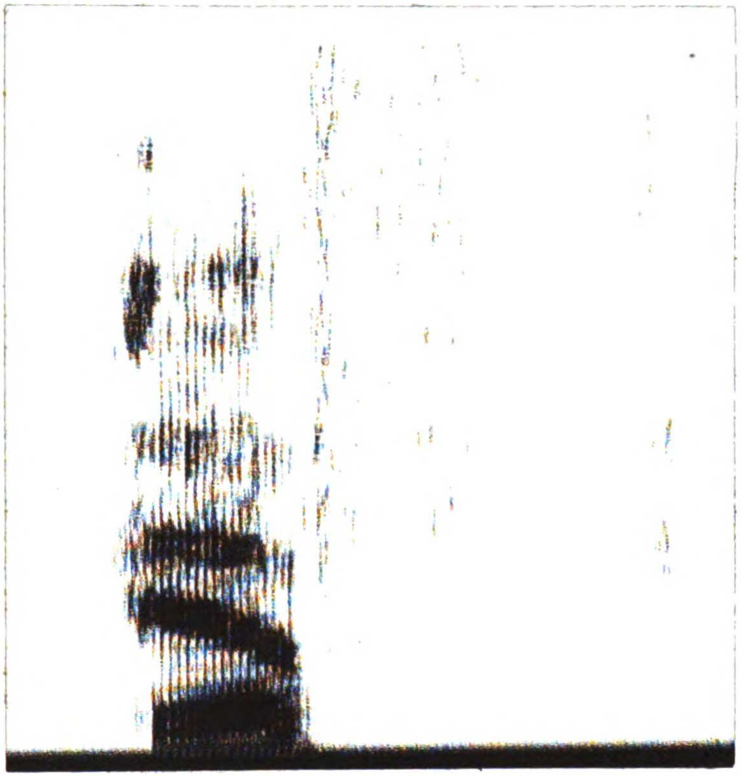
"D"



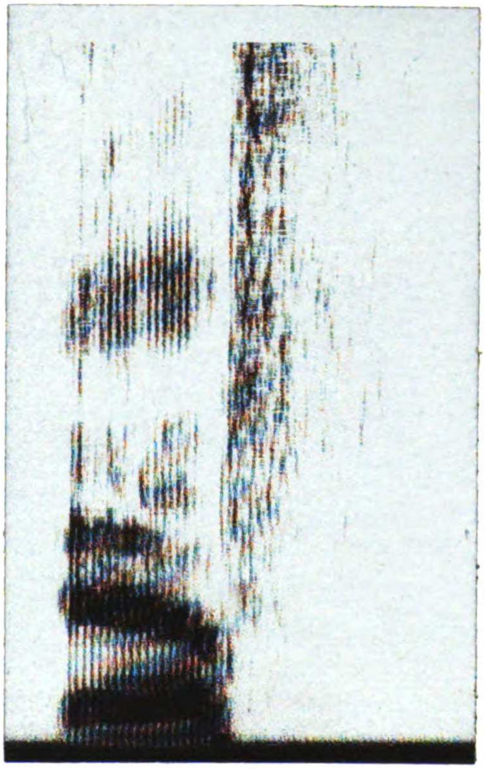


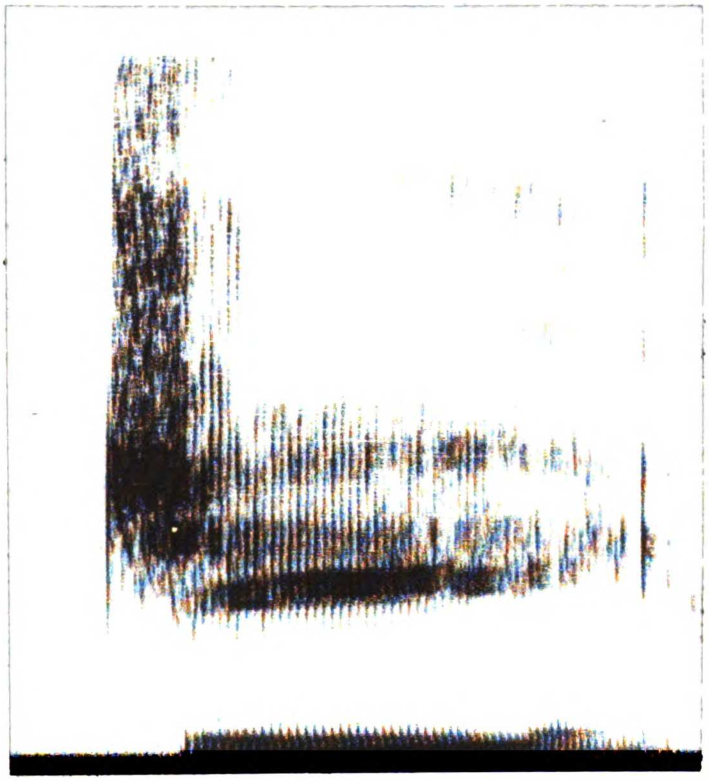
"E"



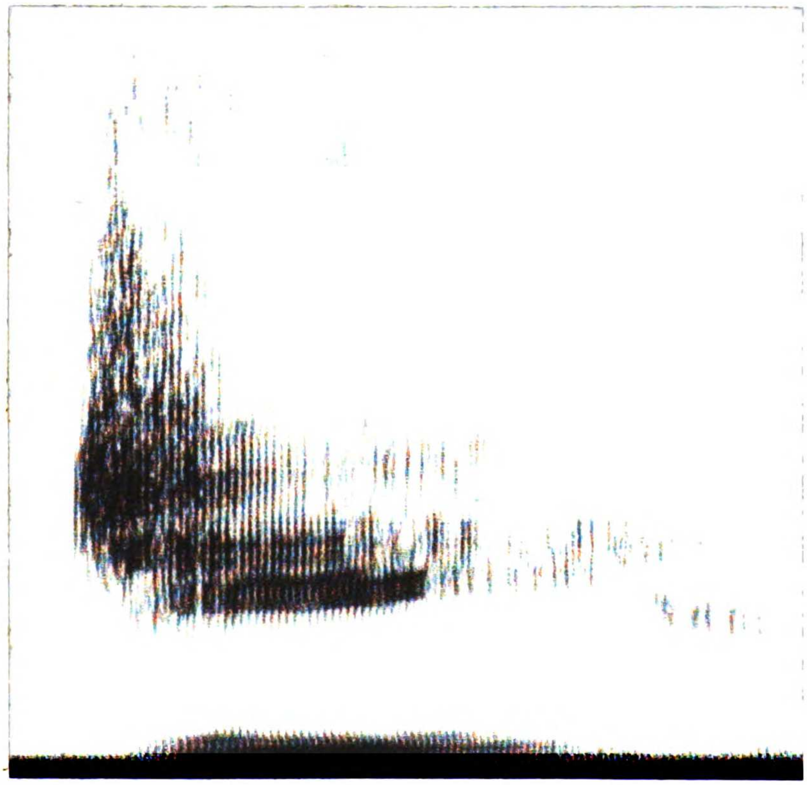


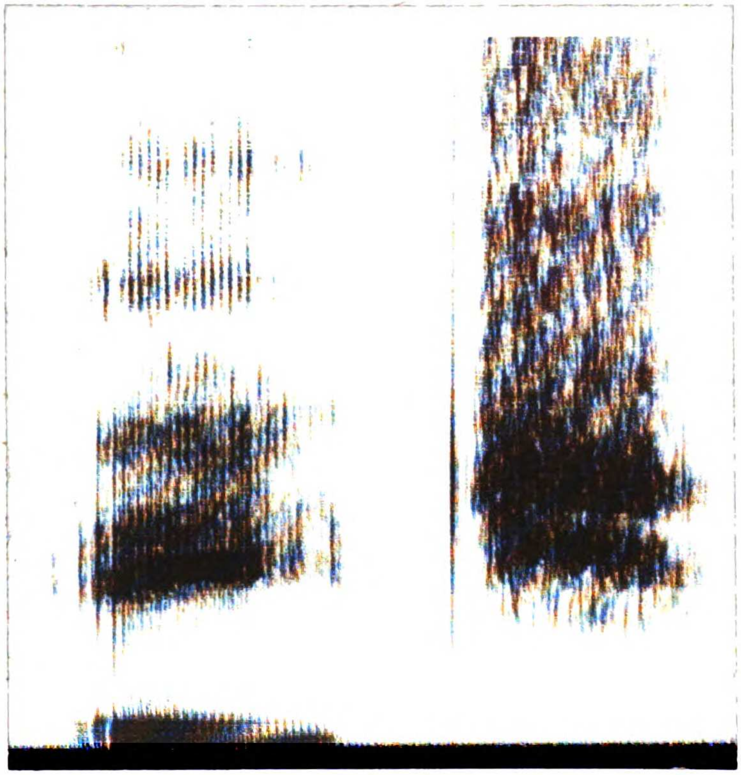
"F"



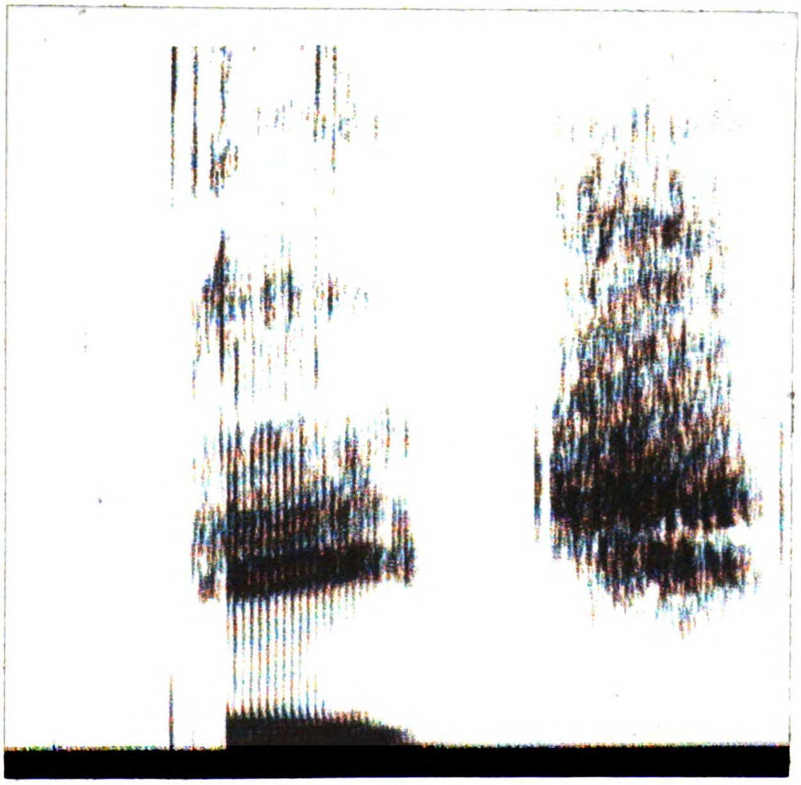


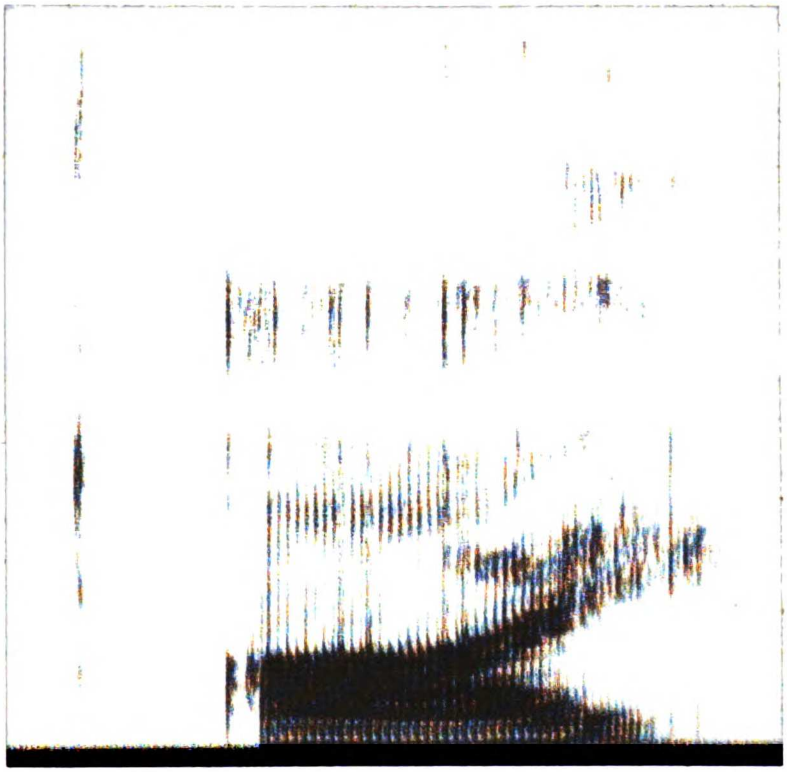
"G"



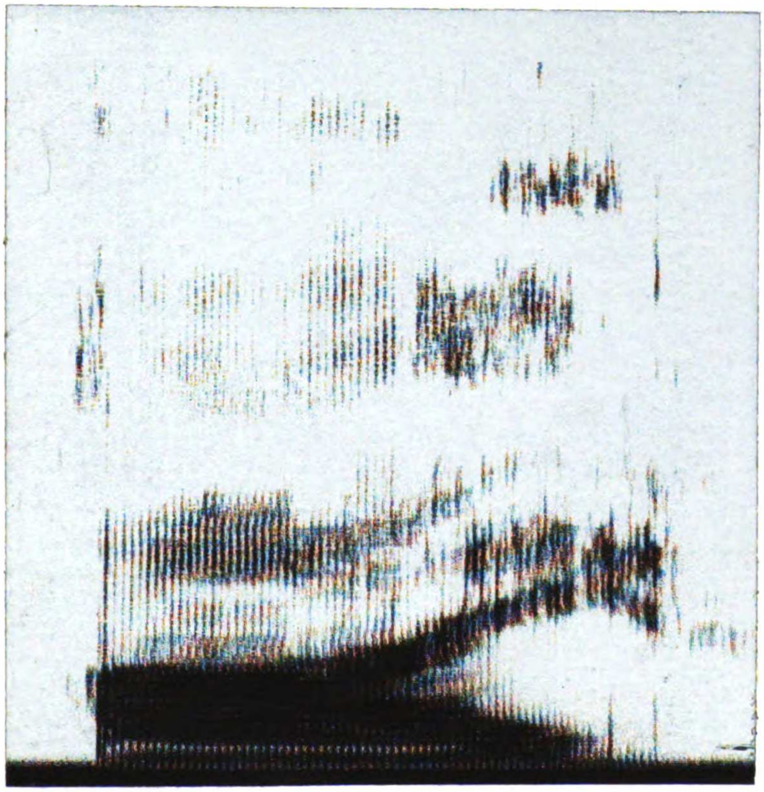


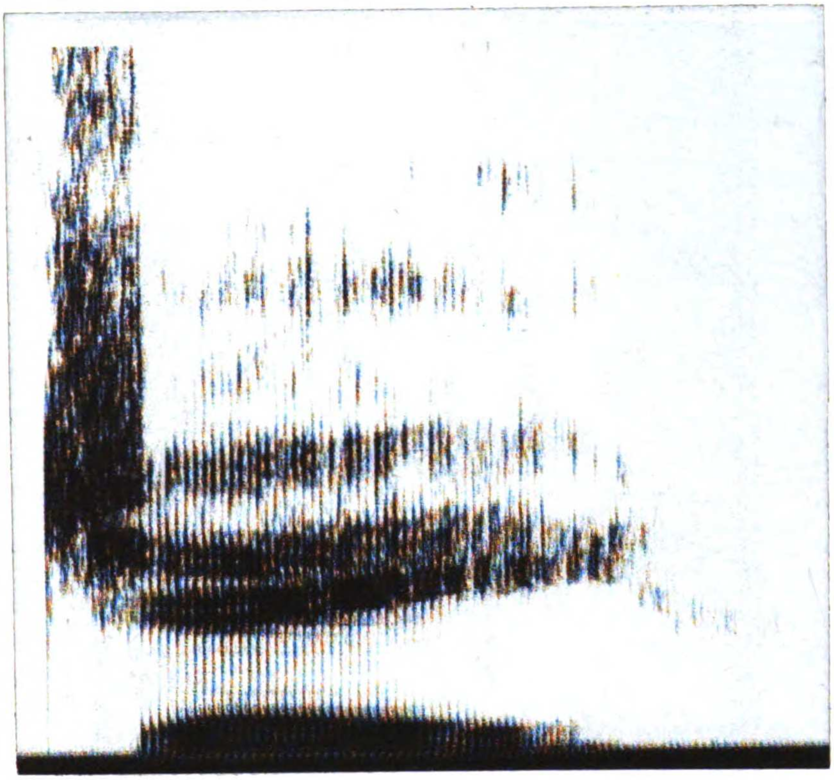
"H"



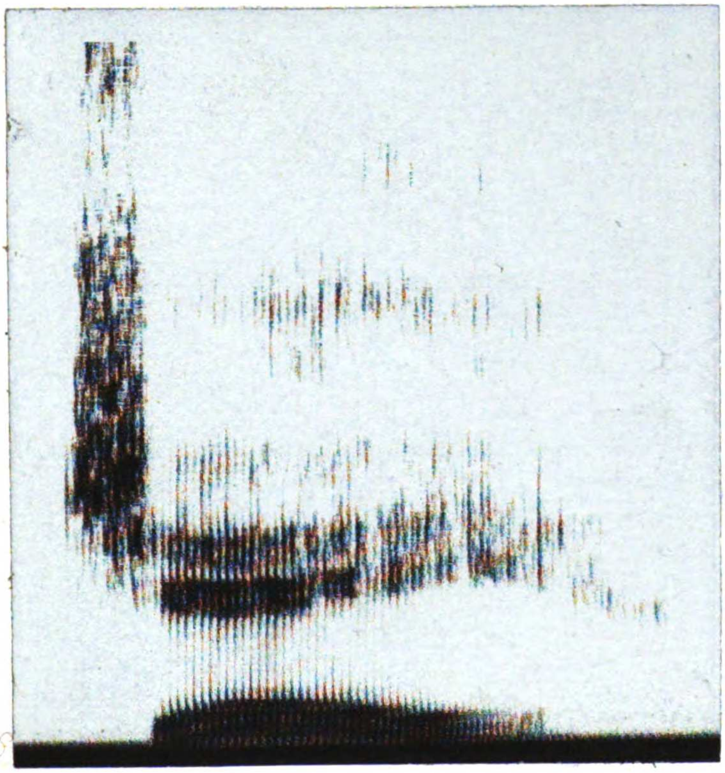


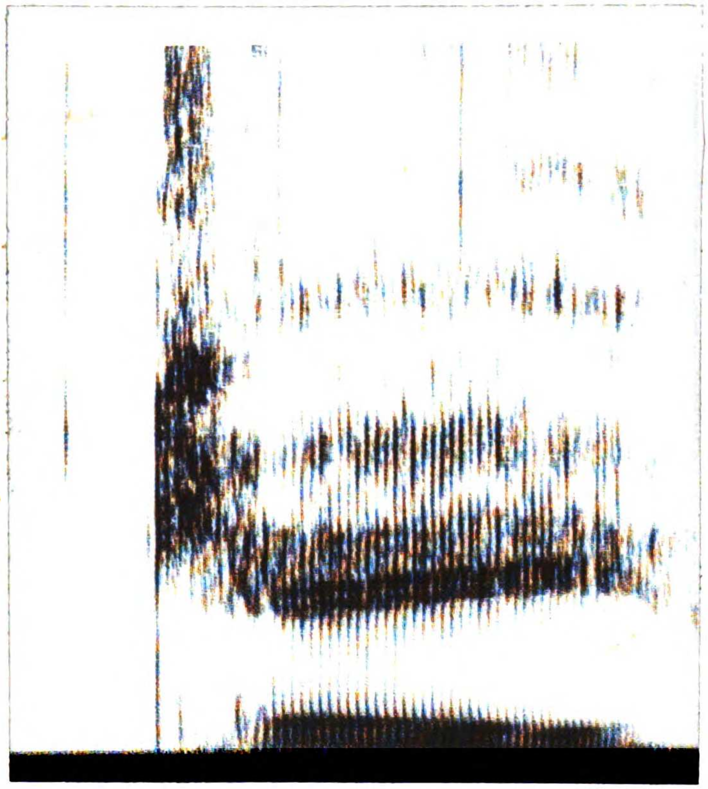
"I"



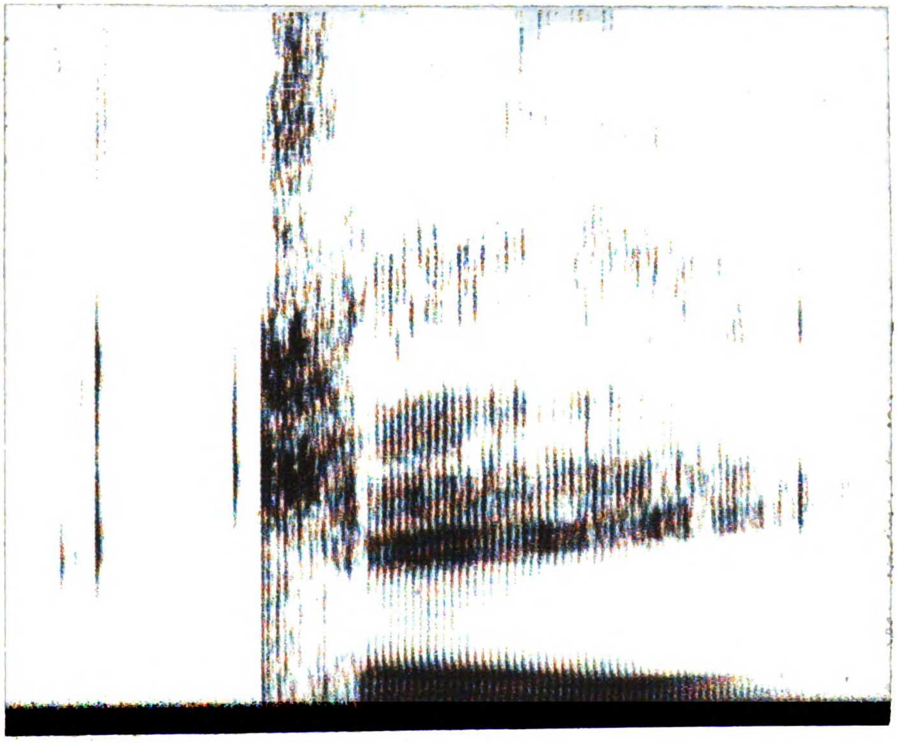


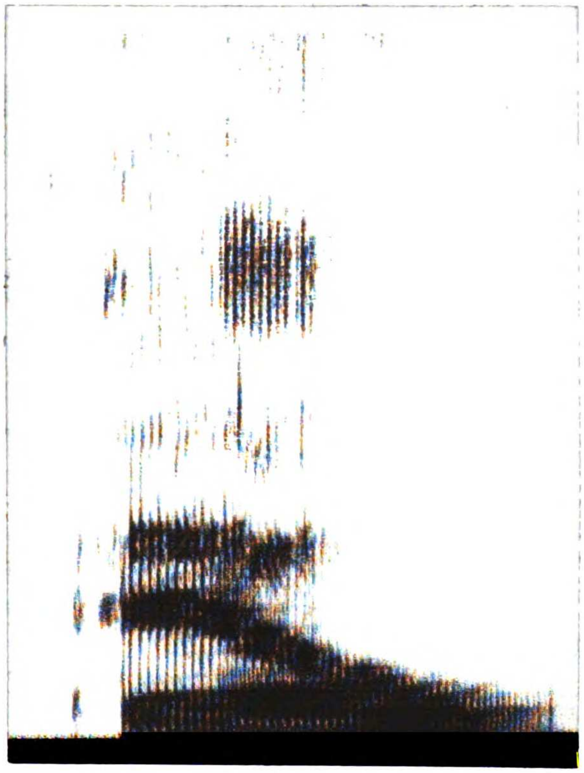
"J"



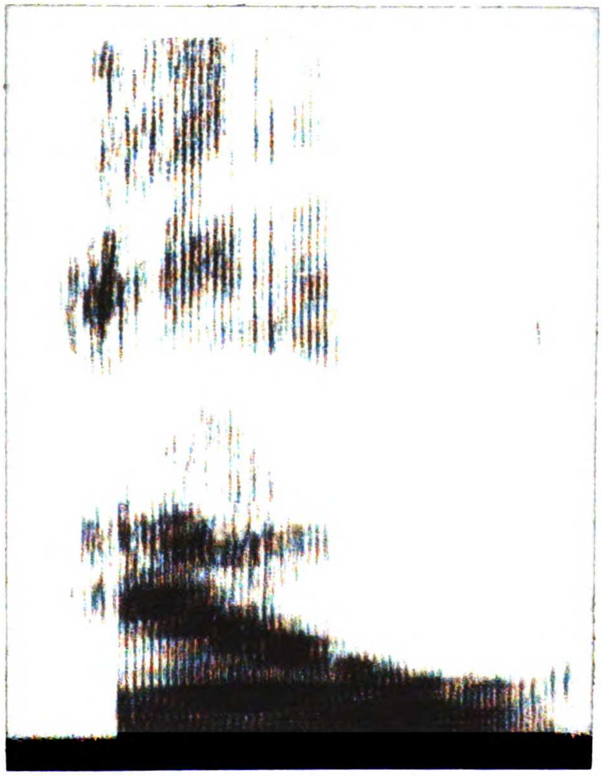


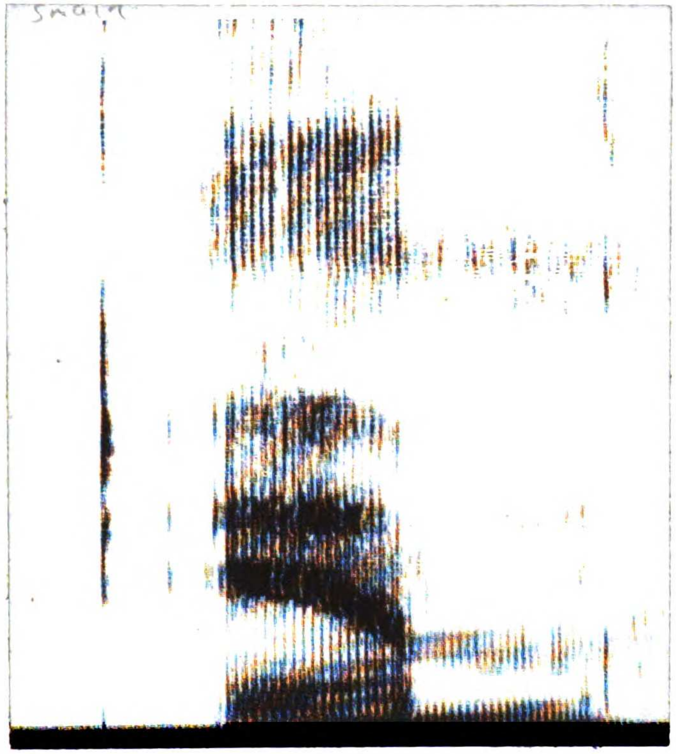
"K"



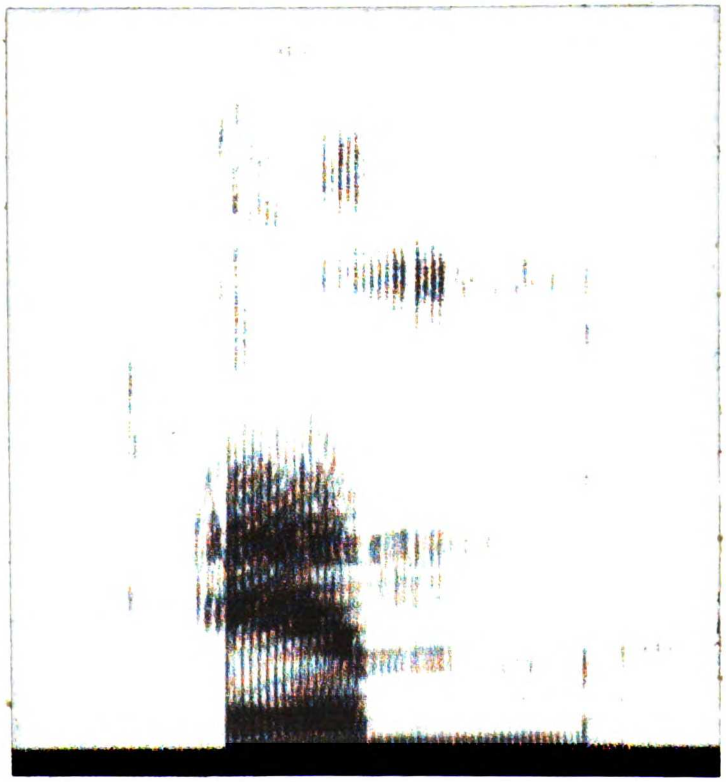


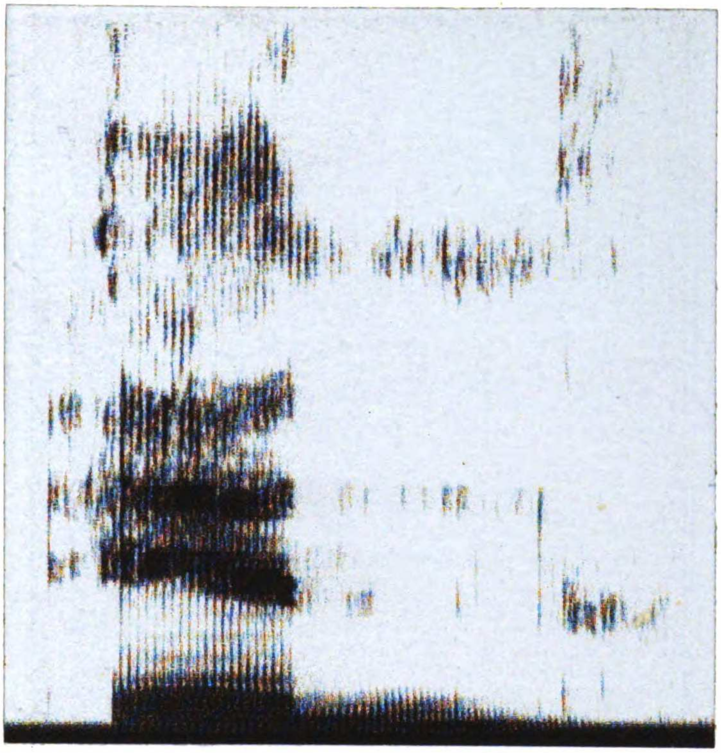
"L"



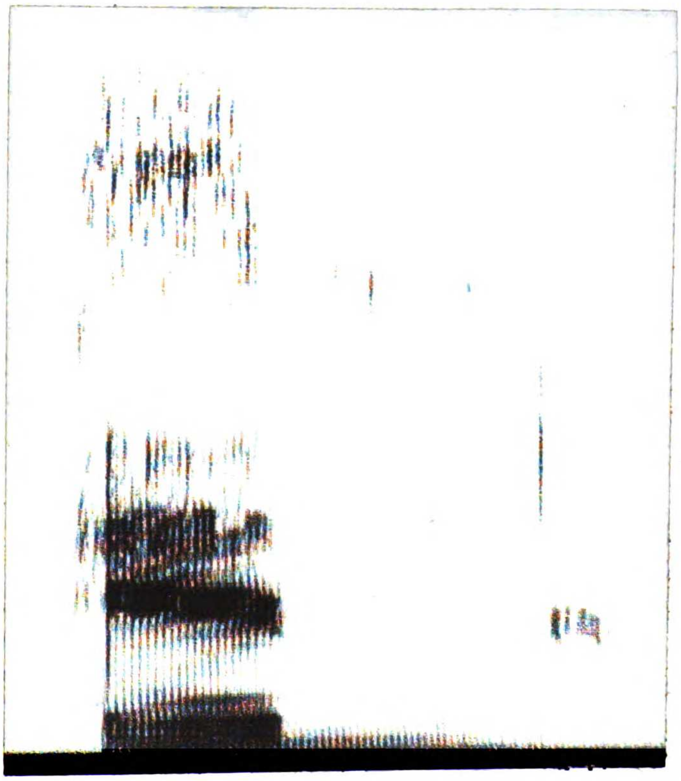


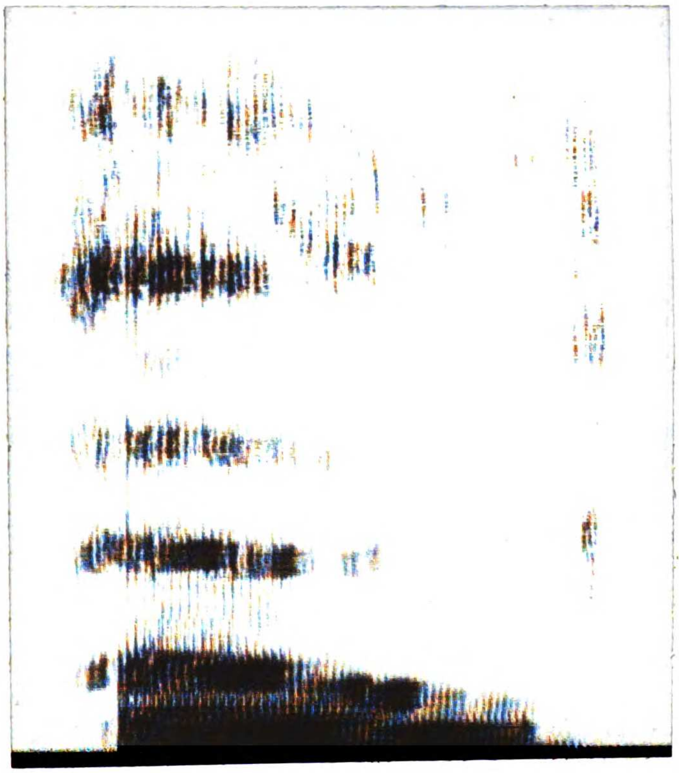
"M"



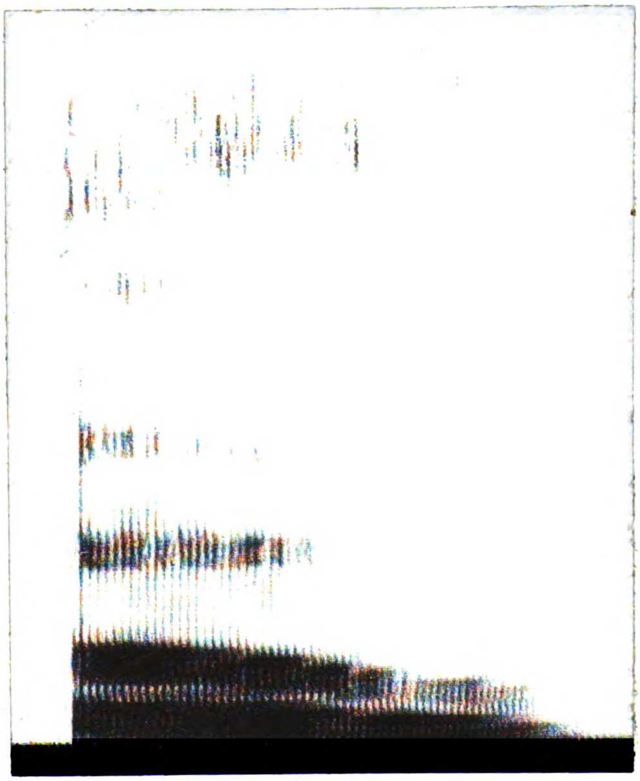


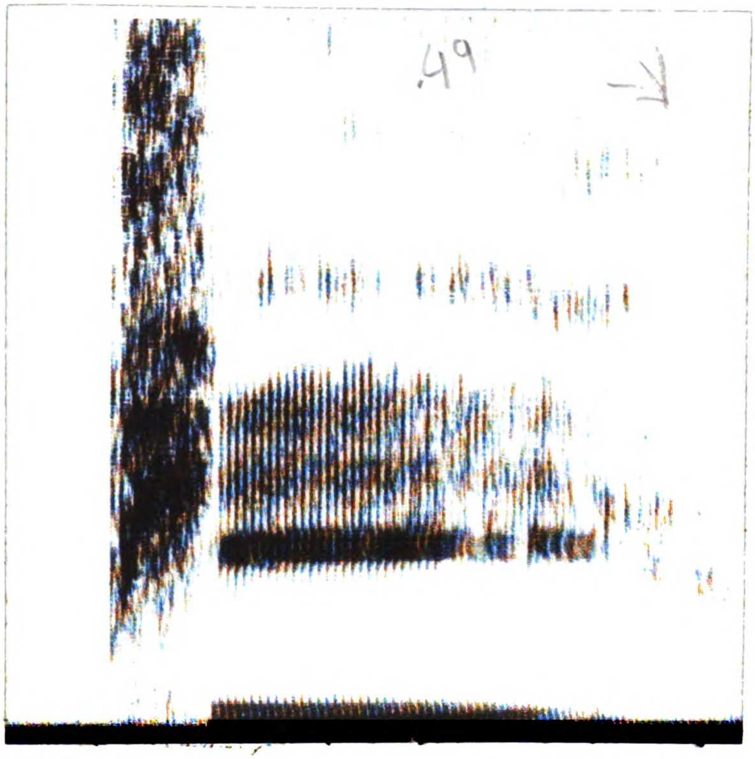
"N"



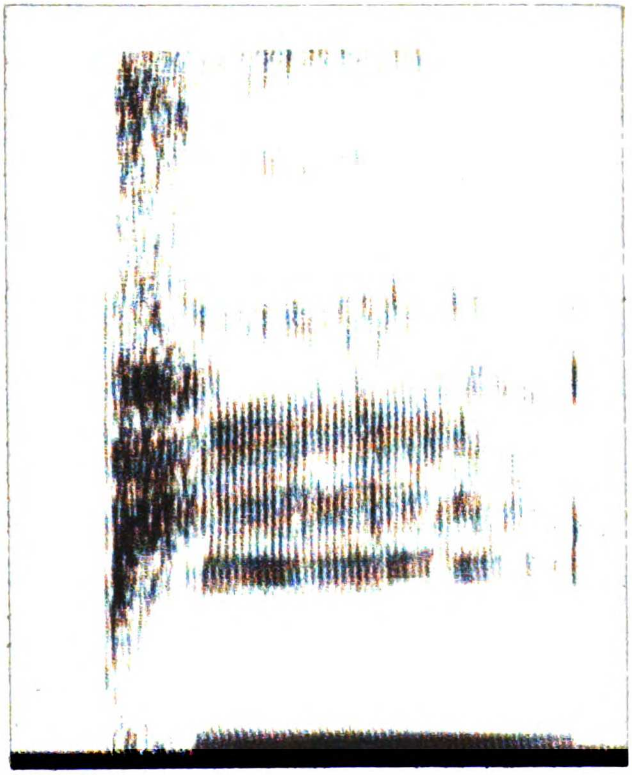


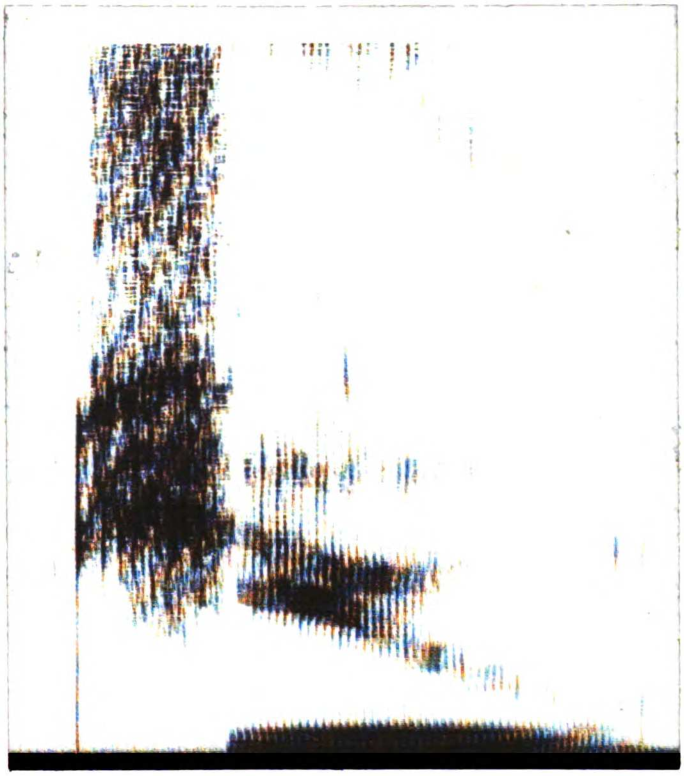
"O"



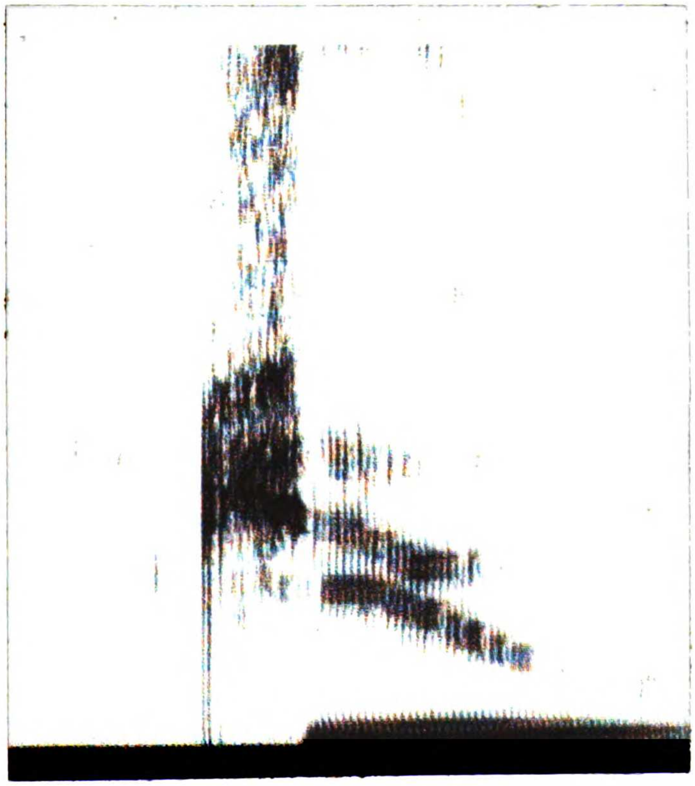


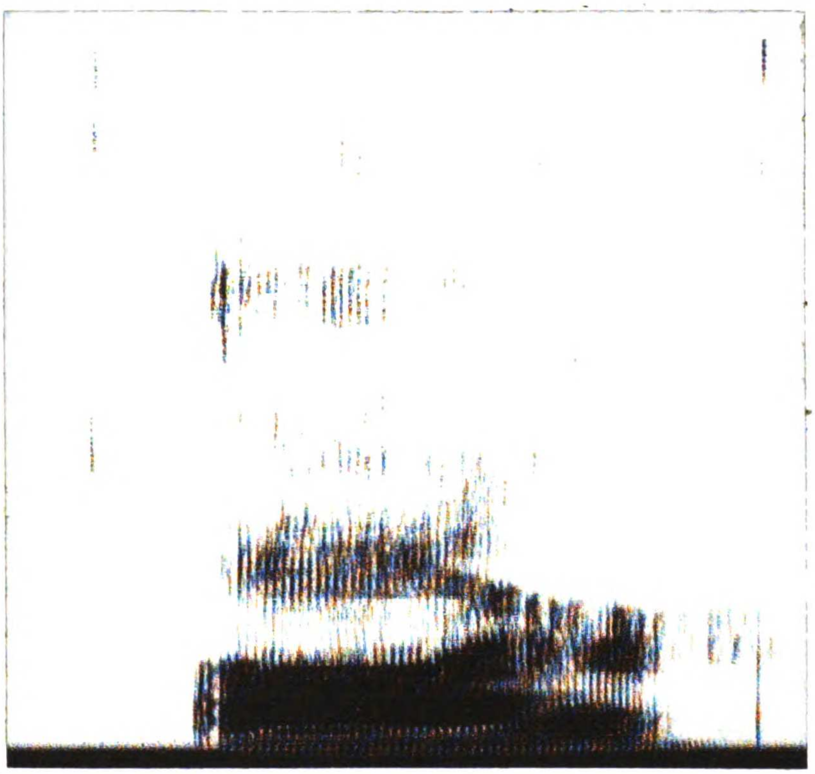
"P"



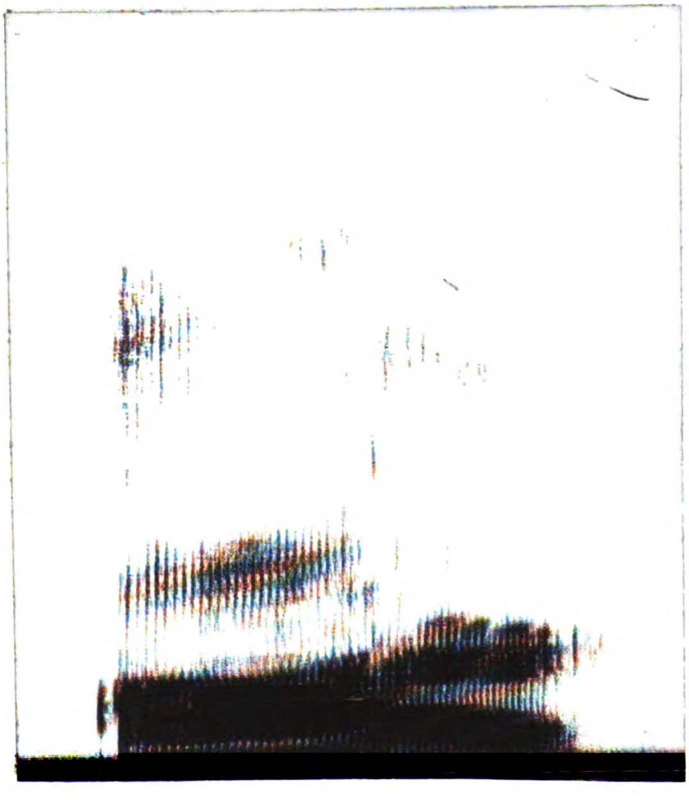


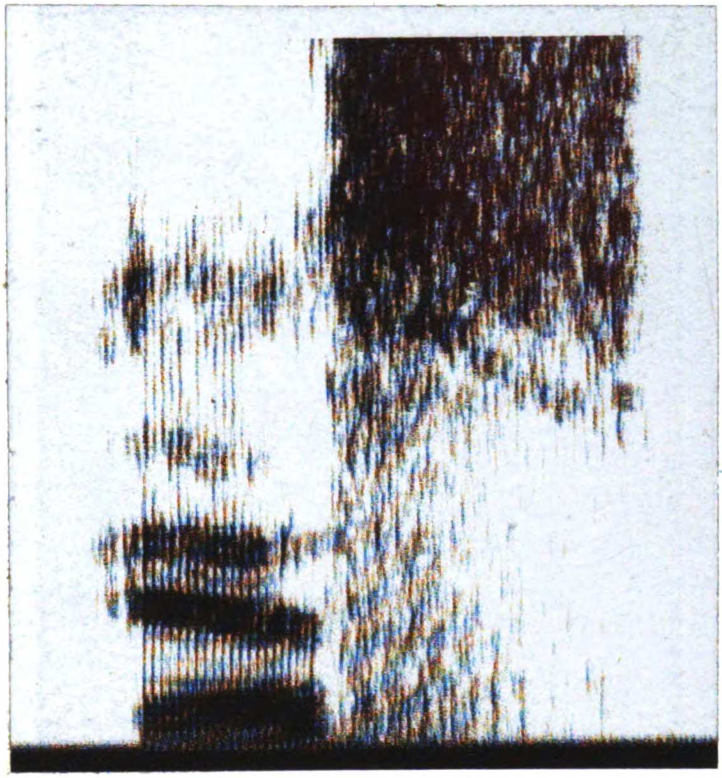
"Q"



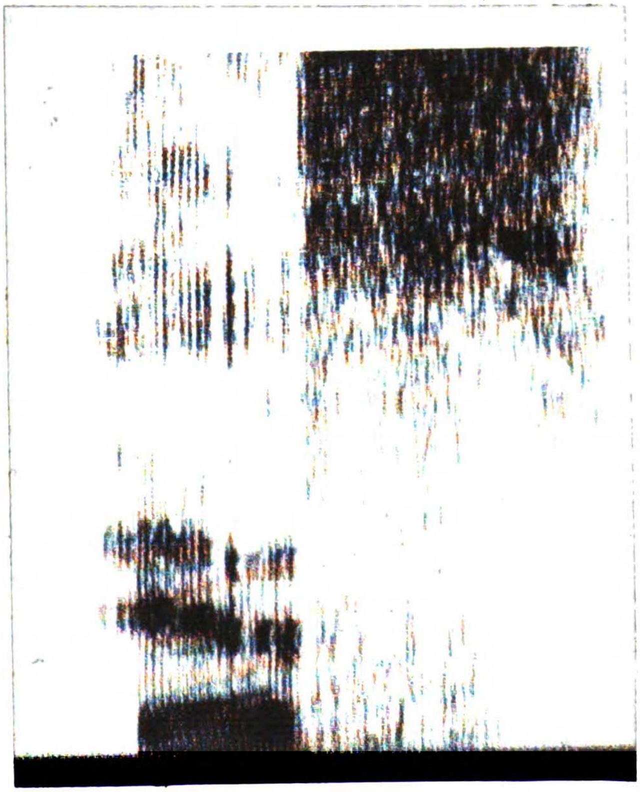


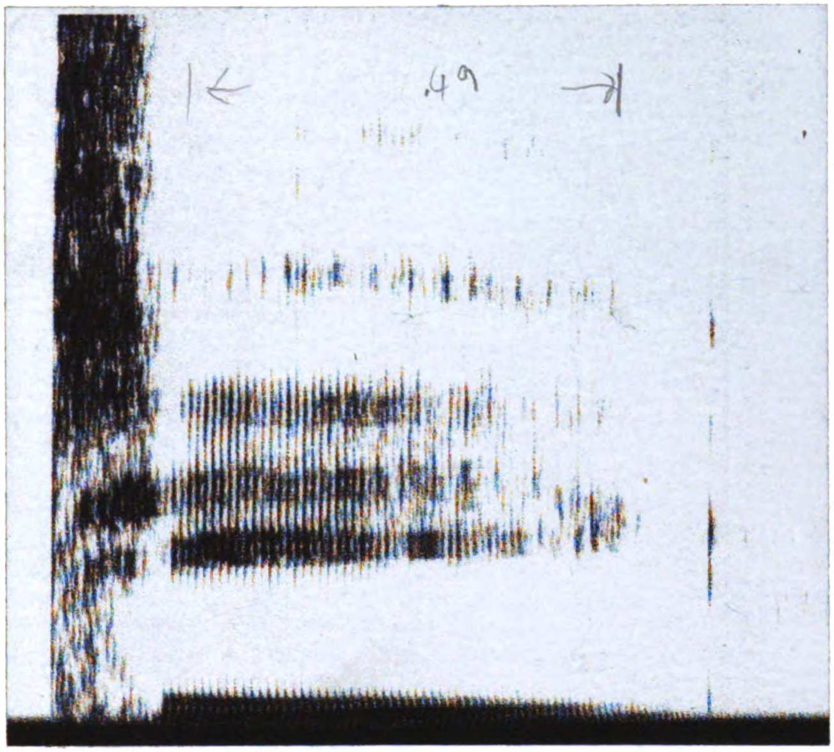
"R"



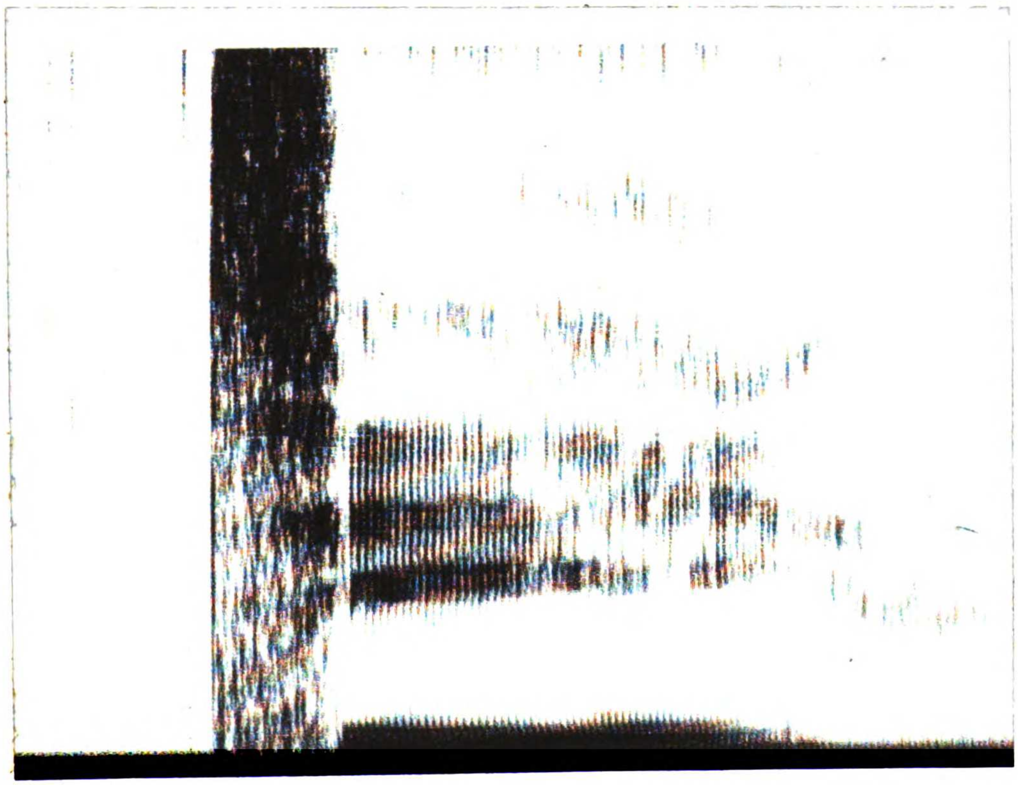


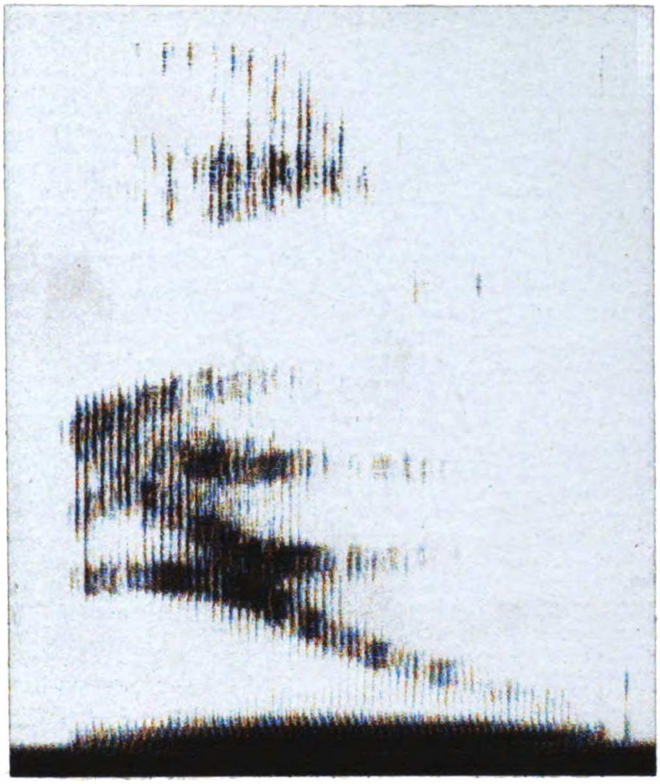
"S"



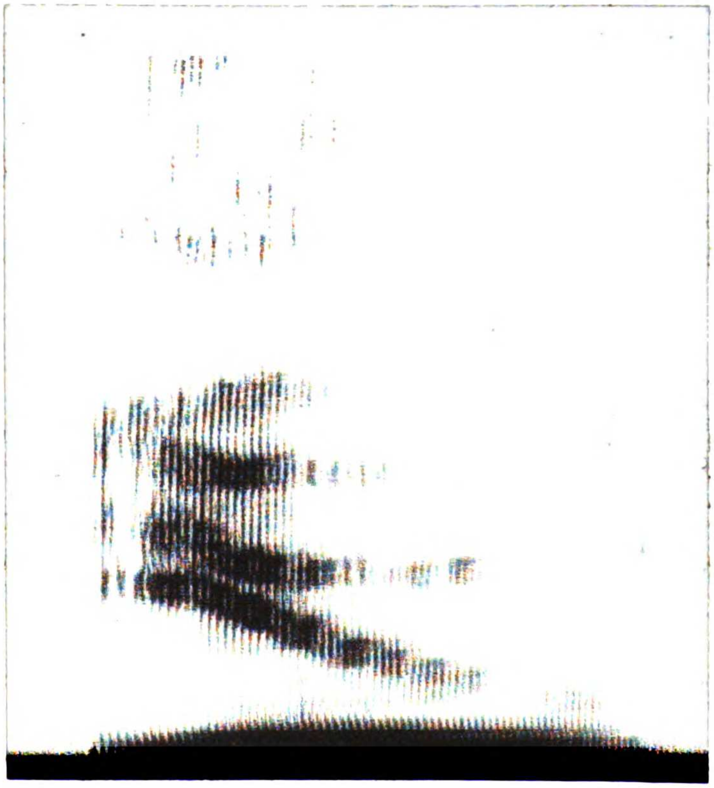


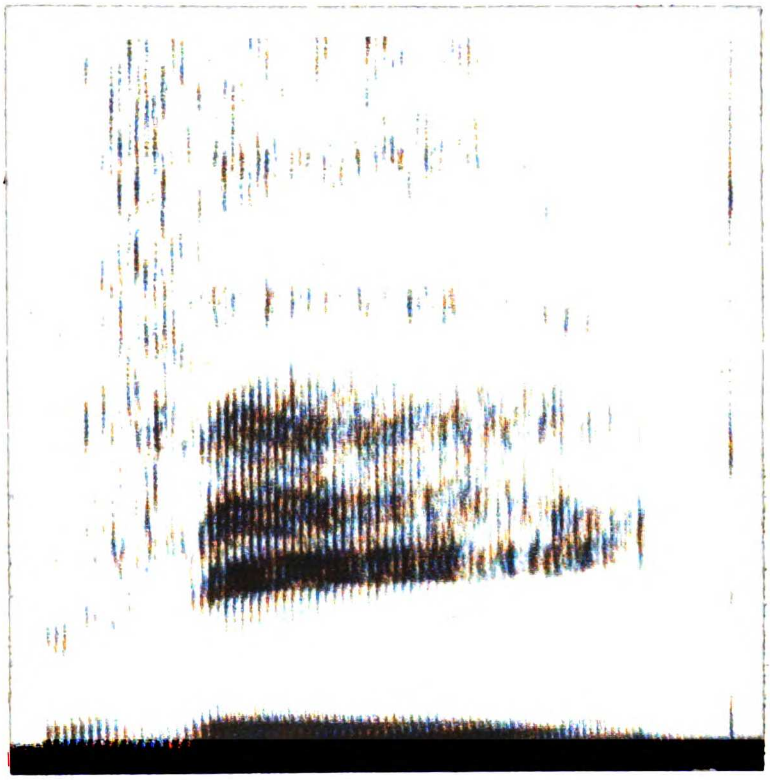
"T"



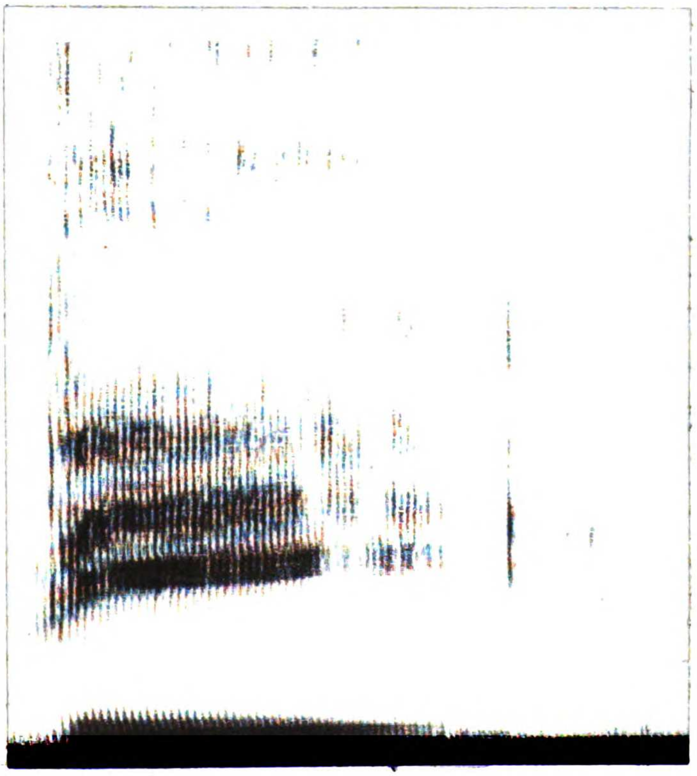


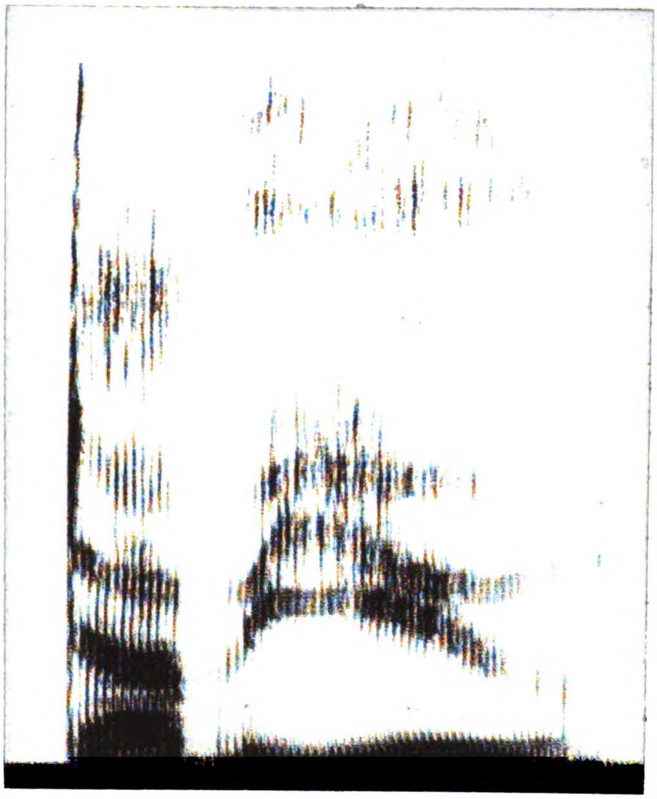
"U"



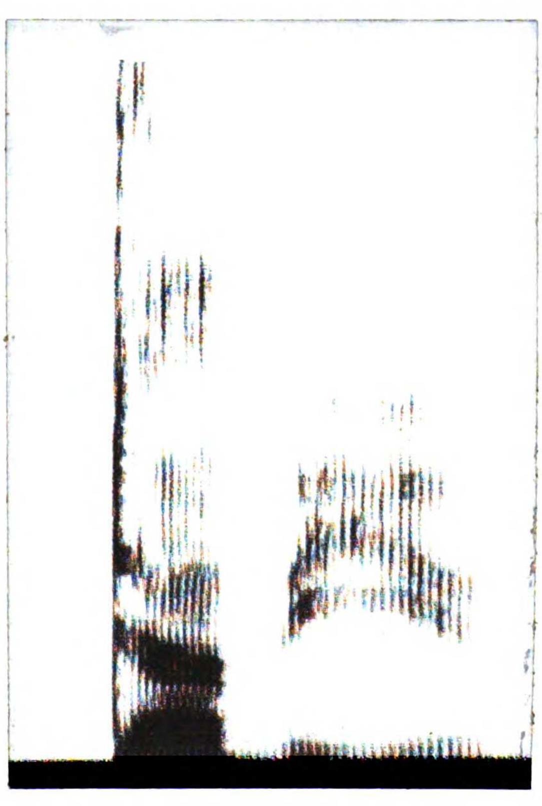


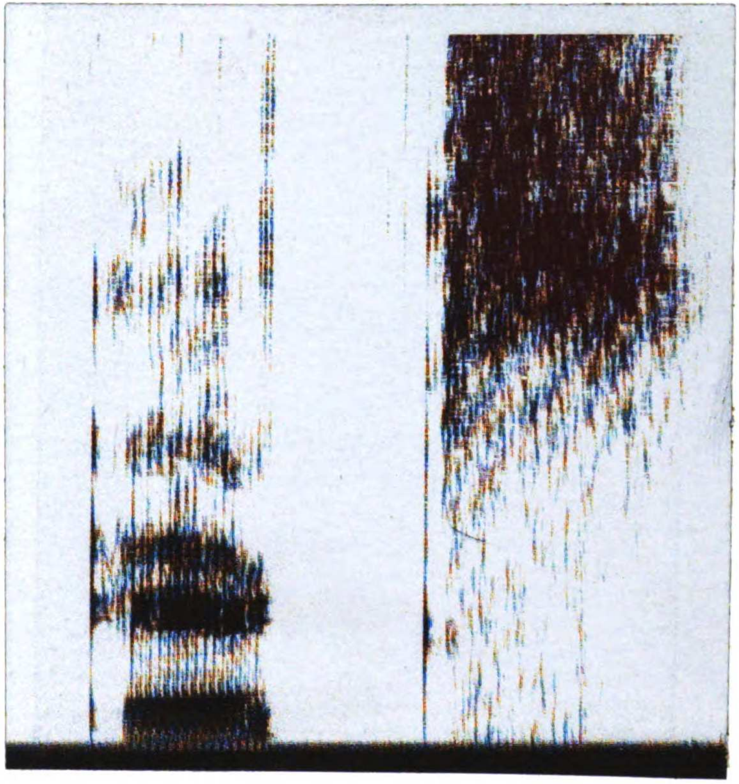
"V"



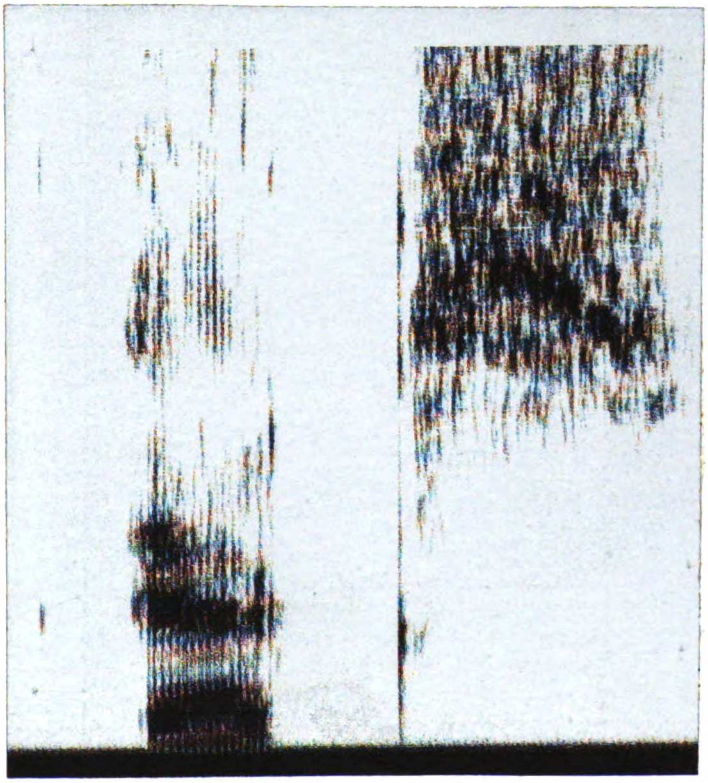


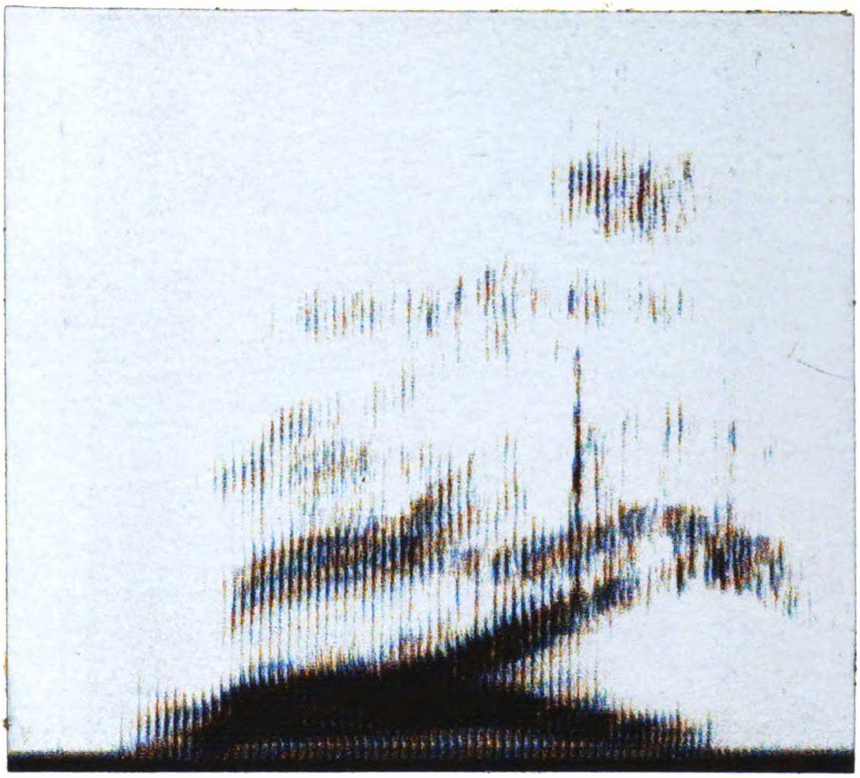
"W"



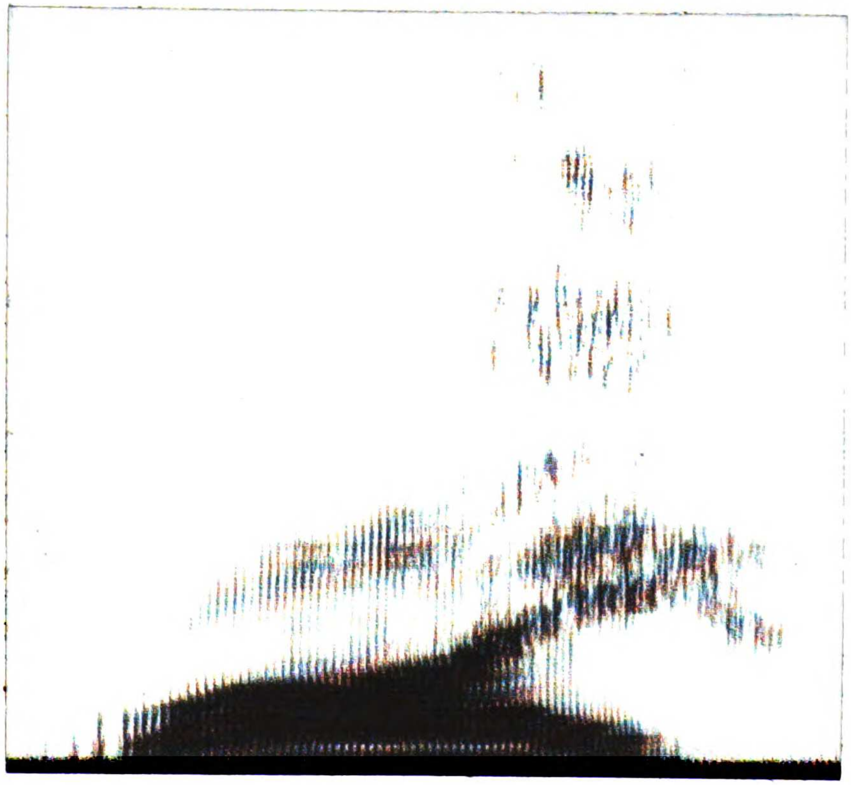


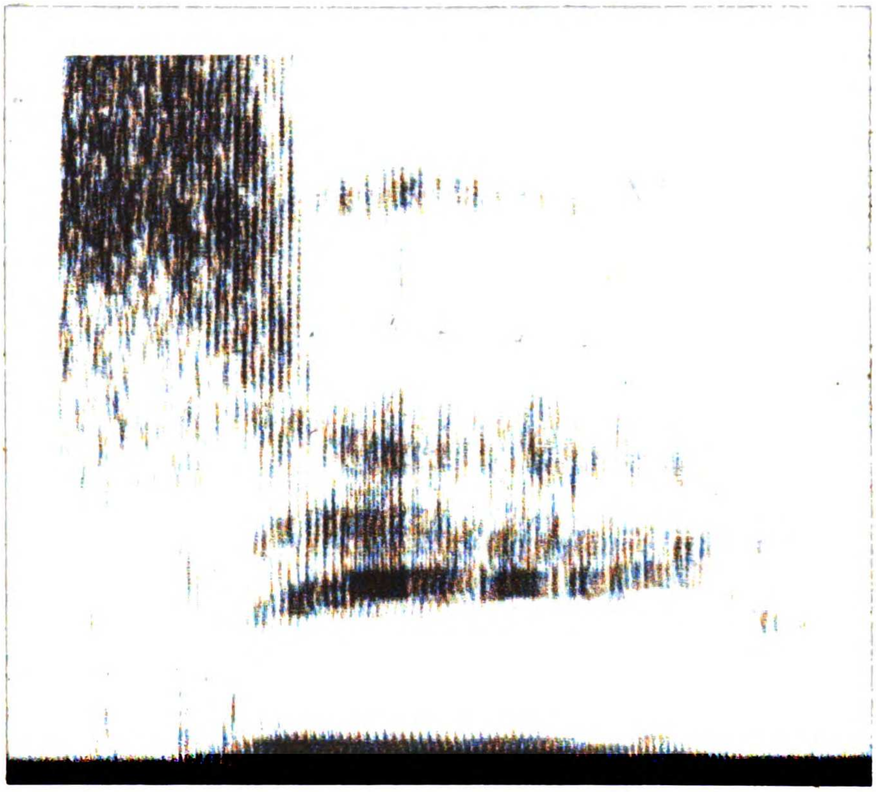
"X"



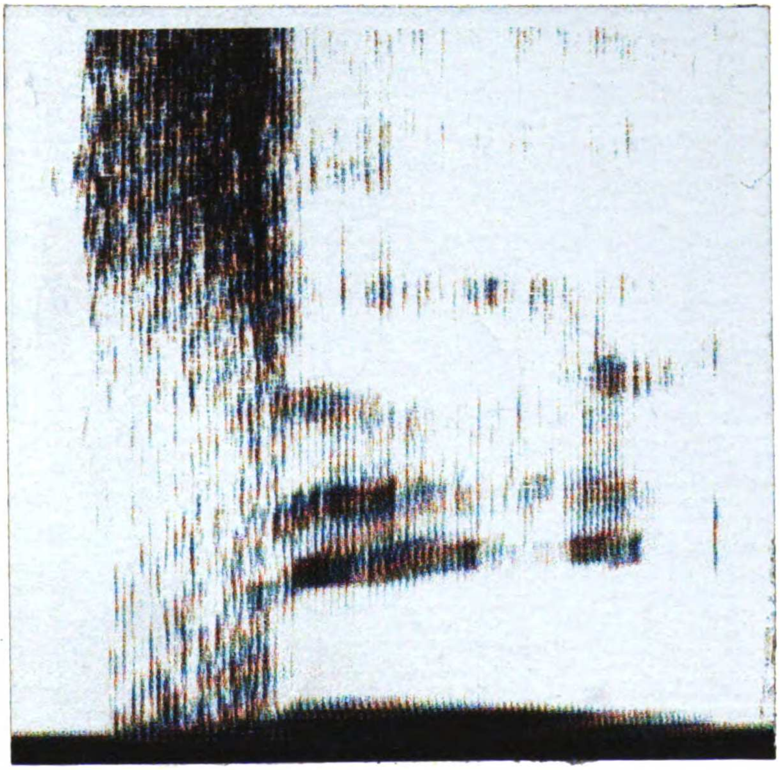


"Y"





"Z"



APPENDIX C

PROGRAM LISTINGS FOR LPC MODEL

```
#
/*
 *
*  This program calculates lpc coefficients, auto.cor. and ita distance metrics.

 *
 *
   Parts of this software are taken from the CMU Harpy Speech Understanding
 system as implemented on the PDP 11/40 UNIX system. All of the algorithms
 for calculating the LPC parameters are taken from Harpy.

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

/* #define DEBUG 1  /* un-comment to turn debugging on */

#ifdef DEBUG
#endif

#include "label.h"
#include "iobuf.h"
#include "types.h"
#include <stdio.h>

extern  long hmul ();

/*
 *********** All this stuff is in labpac.h ***************
 */
int
        numtpl,              /* number of templates */
        lbls[MAXTPL + 1],    /* label names in radix 50 */
        itcs[MAXTPL + 1],    /* itakura c constants for each template */
        tpls[MAXTPL + 1][NPAR + 1];  /* itakura b constants */

int     numtac,
        taclbls[MAXTPL + 1],
        instances[MAXTPL],
        ck[NPAR + 1],   /* k parameters from lpc calculation */
        tacs[MAXTPL +1][NPAR +1];


long    noise,
        engl;
char uttfile[64],
     adcfile[64],
     tplfile[64],
     tacfile[64],
     phnfile[64],
     hslfile[64],
     segfile[64];
/*
 **********************************************************
 */


int     headersize;          /* adc header size in words */

 /* table of 11 bit mantissas for log routine */

int     mantis[256]
{
    0, 12, 23, 34, 46, 57, 68, 80, 91, 102,
    113, 124, 135, 146, 157, 168, 179, 190, 201, 212,
    222, 233, 244, 254, 265, 275, 286, 296, 307, 317,
    327, 338, 348, 358, 368, 379, 389, 399, 409, 419,
    429, 439, 449, 459, 469, 478, 488, 498, 508, 517,
    527, 537, 546, 556, 566, 575, 585, 594, 603, 613,
    622, 631, 641, 650, 659, 669, 678, 687, 696, 705,
    714, 723, 732, 741, 750, 759, 768, 777, 786, 795,
    803, 812, 821, 830, 838, 847, 856, 864, 873, 882,
    890, 899, 907, 916, 924, 933, 941, 949, 958, 966,
    974, 983, 991, 999, 1007, 1016, 1024, 1032, 1040, 1048,
    1056, 1064, 1072, 1080, 1088, 1096, 1104, 1112, 1120, 1128,
    1136, 1144, 1151, 1159, 1167, 1175, 1183, 1190, 1198, 1206,
    1213, 1221, 1229, 1236, 1244, 1251, 1259, 1266, 1274, 1281,
    1289, 1296, 1304, 1311, 1319, 1326, 1333, 1341, 1348, 1355,
    1363, 1370, 1377, 1384, 1392, 1399, 1406, 1413, 1420, 1427,
    1435, 1442, 1449, 1456, 1463, 1470, 1477, 1484, 1491, 1498,
    1505, 1512, 1519, 1525, 1532, 1539, 1546, 1553, 1560, 1566,
    1573, 1580, 1587, 1594, 1600, 1607, 1614, 1620, 1627, 1634,
    1640, 1647, 1653, 1660, 1667, 1673, 1680, 1686, 1693, 1699,
    1706, 1712, 1719, 1725, 1732, 1738, 1744, 1751, 1757, 1764,
    1770, 1776, 1783, 1789, 1795, 1801, 1808, 1814, 1820, 1826,
    1833, 1839, 1845, 1851, 1857, 1863, 1870, 1876, 1882, 1888,
    1894, 1900, 1906, 1912, 1918, 1924, 1930, 1936, 1942, 1948,
    1954, 1960, 1966, 1972, 1978, 1984, 1990, 1996, 2001, 2007,
    2013, 2019, 2025, 2031, 2036, 2042
};


 /* 200 point hamming window.  (1.0==077777)  */

int     hw[201]
{
    0,
    2621, 2628, 2651, 2688, 2740, 2806, 2888, 2984, 3094, 3219,
    3359, 3512, 3679, 3861, 4055, 4264, 4485, 4720, 4967, 5227,
    5500, 5784, 6080, 6387, 6706, 7036, 7376, 7726, 8086, 8455,
    8834, 9221, 9617, 10021, 10432, 10851, 11276, 11708, 12145, 12588,
    13036, 13488, 13945, 14406, 14869, 15336, 15805, 16275, 16747, 17220,
    17694, 18167, 18640, 19112, 19583, 20052, 20518, 20982, 21442, 21899,
    22351, 22799, 23242, 23680, 24111, 24537, 24955, 25366, 25770, 26166,
    26553, 26932, 27301, 27661, 28012, 28352, 28681, 29000, 29307, 29604,
    29888, 30160, 30420, 30667, 30902, 31124, 31332, 31527, 31708, 31875,
    32029, 32168, 32293, 32403, 32500, 32581, 32648, 32700, 32737, 32759,
    32767, 32759, 32737, 32700, 32648, 32581, 32500, 32404, 32293, 32168,
    32029, 31875, 31708, 31527, 31332, 31124, 30902, 30668, 30420, 30160,
    29888, 29604, 29308, 29000, 28681, 28352, 28012, 27662, 27301, 26932,
    26553, 26166, 25770, 25366, 24955, 24537, 24111, 23680, 23242, 22799,
    22351, 21899, 21442, 20982, 20518, 20052, 19583, 19112, 18640, 18167,
    17694, 17220, 16747, 16275, 15805, 15336, 14869, 14406, 13945, 13489,
    13036, 12588, 12145, 11708, 11276, 10851, 10432, 10021, 9617, 9222,
    8834, 8455, 8086, 7726, 7376, 7036, 6706, 6387, 6080, 5784,
    5500, 5227, 4967, 4720, 4485, 4264, 4055, 3861, 3679, 3512,
    3359, 3219, 3094, 2984, 2888, 2806, 2740, 2688, 2651, 2628
};


/*
 * logabs returns the base 2 log of the absolute value of
 * its argument.  the argument should have the binary point
 * to the right of bit 8.  the result has the point to the
 * right of bit 11.
 */

logabs (n)
int     n;
{
    register int    ch,      /* characteristic */
                    num;

    if ((num = n) == 0)
        return (NEGINFINITY);
    if (num < 0)
        num = -num;
    ch = 0;
    if ((num & 0177400) == 0)
        while ((num & 0177400) != 0400)
        {
            ch =- 04000;
            num =* 2;
        }
    else
        while ((num & 0177400) != 0400)
        {
            ch =+ 04000;
            num =/ 2;
        }
    return (ch | mantis[num & 0377]);
};
```

```
/*
 * dot does a dot product of two n word vectors starting
 * at locations a and b.  the high order 16 bits of the
 * dot product are returned.
 */

dot (a, b, n)
int     a[],
        b[],
        n;
{
    long dotprod;
    register int    i,
                   *ap,
                   *bp;

    ap = a;
    bp = b;
    i = n;
    dotprod = 0;
    do
    {
        dotprod =+ hmul (*ap++, *bp++);
    } while (--i);
    return (dotprod.hiword);
};


/* prepro preprocesses the input signal to make it suitable
 * for parameter calculation.  the preprocessing involves
 * scaling the input frame to 14 bits, differencing the
 * scaled signal with a preemphasis factor of .92, scaling
 * the differenced signal to 13 bits, and finally applying
 * a hamming window.  the resulting preprocessed signal
 * contains 12 bits of significance.
 */

prepro (d, p)
int     d[],
        p[];
{
    int     maxv,
            snum,
            last,
            abs_d,
            new;
    register int    i,
                   *iptr;
    long l;
    /* find max value in frame */
    maxv = 0;                                /* initial maximum */
    iptr = &d[1];
    for (i = 1; i <= WINDOWSIZE; i++)
    {
        abs_d = abs (*iptr++);
        if (abs_d > maxv)
            maxv = abs_d;
    }
#ifdef DEBUG
    printf ("MAX AMP = %d#", maxv);
#endif

    /*
     * from max value, determine how much to shift
     * the input so that it has 14 bits of significance.
     */
    for (snum = 0; maxv < 040000; snum++)
        maxv =<< 1;
    snum--;

    /*
     * scale and difference the input frame.
     * a preempasis factor of .92 is used in the differences
     */
    maxv = 0;
    last = (d[1] << snum);
    for (i = 1; i < WINDOWSIZE; i++)
    {
        new = (d[i + 1] << snum);
        l = hmul (last, 072703);
        l =<< 1;
        maxv = maxi (maxv, abs (p[i] = new - l.hiword));
        last = new;
    };

    /*
     * determine how much to shift the differenced signal
     * so that it contains 13 bits of significance.
     */
    for (snum = 0; maxv < 040000; snum++)
        maxv =<< 1;
    snum =- 2;

    /*
     * scale up the differenced signal and apply a hamming window.
     * the results will have 12 bits of significance.
     */
    for (i = 1; i < WINDOWSIZE; i++)
    {
        l = hmul (p[i] << snum, hw[i]);
        p[i] = l.hiword;
    }
    p[WINDOWSIZE] = 0;

};


/* calculate and normalize the autocorrelation coefficients
 * of array d.  return the coefficients with the binary point
 * to the right of bit 14 in the array r.
 */

autoc (d, r)
int     d[],
        r[];
{
    int     i,
            lag,
            scl,
            snum;
    long ac;
    register int   *iptr1,
                   *iptr2,
                    itmp;

    /* calculate first coefficient for normalization */
    ac = 0;
    iptr1 = &d[1];
    itmp = WINDOWSIZE;
    do
    {
        ac =+ hmul (*iptr1, *iptr1);
        *iptr1++;                           /* increment */
    } while (--itmp);

    /* shift until bit 14 of ac is significant */
    for (itmp = 0; (ac.hiword & 040000) == 0; itmp++)
        ac =<< 1;
    snum = itmp - 2;
    scl = ac.hiword;
    r[0] = 040000;                          /* make 1st coefficient 1.0 */

    /* calculate coefficients for lags */
    for (lag = 1; lag <= NPAR; lag++)
    {
        ac = 0;
        itmp = WINDOWSIZE - lag;
        iptr1 = &d[1];
        iptr2 = &d[1 + lag];
        do
            ac =+ hmul (*iptr1++, *iptr2++);
        while (--itmp);
        ac =<< snum;
        r[lag] = ac / scl;
    }
};


/* Solve the autocorrelation equations for LPCs.  Called with
 * the autocorrelation coefficients in acr.  Returns LPCs
 * with the binary point to the right of bit 12 in lpc
 */

solve (acr, lpc)
int     acr[],
        lpc[];
{
    register int    i,
                    n,
                    c;
    int     alpha,
            t[NPAR + 1];
    long beta, l;

    /* set up initial solution */
    lpc[0] = 010000;
    alpha = acr[0];
    smear (&lpc[1], NPAR, 0);

    for (n = 1; n <= NPAR; n++)
    {
        beta = 0;
        for (i = 0; i < n; i++)
            beta =+ hmul (lpc[i], acr[n - i]);
        beta =<< 3;
        c = -beta / alpha;
        ck[n] = c;
        for (i = 1; i <= n; i++)
        {
            l = hmul (c, lpc[n - i]);
            l =<< 1;
            t[i] = lpc[i] + l.hiword;
        }
        movcor (&t[1], &lpc[1], n);
        l = hmul (c, beta.hiword);
        l =<< 2;
        alpha =+ l.hiword;
    }
}
```

```
/* adcgetframe takes a time in 100's of micro seconds */
/* a buffer for the data of windowsize and the adc    */
/* channel                                            */

adcgetframe (tim, fr, adcchn)
int     fr[];
{
    register    offset,
                err,
                i;

long
        loff;
    offset = tim;
    if (offset < 0)
        offset = 0;
    offset =* 2;        /* make byte address */
    offset =+ 2*headersize;
     loff = offset;
    lseek (adcchn, loff, 0);
   if ( read (adcchn, &fr[1], WINDOWSIZE * 2) == 0)
    return(0);
#ifdef DEBUG
    printf ("#Input frame at time = %d#", tim);
    printf ("header size = % words#", headersize/2);
    printf ("word address = %d#", offset/2);
#endif
    /*
* shift the data right two bits to get 14 bits of significance.
     */
for (i =1; i <= WINDOWSIZE; i++)
fr[i] = fr[i] >> 2;
return(1);
}


readtpls ()
{
    int     chn,
            tmp;
    chn = wantread (tplfile, 0, "TPL file: ");
    read (chn, &tmp, 2);                    /* version */
    if (tmp != 1)
        quit (-1, "TPL file has wrong version number (%d)#", tmp);
    read (chn, &numtpl, 2);                 /* number of templates */
    if (numtpl <= 5 || numtpl > MAXTPL)
        quit (-1, "TPL file has %d templates; quite absurd.#", numtpl);

    if (read (chn, &lbls[1], numtpl * 2) != numtpl * 2)
        quit (-1, "read error while reading templates#");
    read (chn, &tmp, 2);
    if (tmp != 012345)
        quit (-1, "marker in tpl file missing#");
    tmp = numtpl * 2;
    if (read (chn, &itcs[1], tmp) != tmp)
        quit (-1, "read error while reading ITCS#");
    read (chn, &tmp, 2);
    if (tmp != 012345)
        quit (-1, "marker missing in tpl file#");
    tmp = numtpl * NUMCOEFF * 2;
    if (read (chn, &tpls[1][0], tmp) != tmp)
        quit (-1, "read error while reading TPLS#");

    printf ("tplfile: %s: %d templates#", tplfile, numtpl);
    close (chn);
}


getseg (segb)
int     segb[];
{
    register    chn;
    int nsegs,precision;
    chn = wantread (segfile, 0, "SEG file: ");
    read (chn, &nsegs, 2);
    read (chn, &precision, 2);
    if (precision != 100)
        printf ("Precision in segfile not = 100#");
    read (chn, &segb[1], 512);
    close (chn);
    printf ("segfile: %s: %d segments#", segfile, nsegs);
    return (nsegs);                         /* number of segments */
}



/* itagen generates templates from the lpc */
/* coeffecients. itagen also generates the */
/* itakura c constanst                     */

itagen (lpc, tpl, itakurcs)
int lpc[],tpl[];
int   *itakurcs;

{
register j,i,jmax;
long tmp, l;

*itakurcs = dot (lpc , lpc, NPAR+1);         /* the itakurcsa c constant */
tpl[0] = 010000;                             /* make first coeff 1     */
for ( i = 0; i <= NPAR-1; i++)
        {
        jmax = NPAR - (i +1);
        tmp = 0;
        for ( j = 0; j <= jmax; j++)
                tmp =+ hmul (lpc[j], lpc[j+i+1]);
        tmp =>> 3;
        tmp = tmp/(*itakurcs);
        tpl[i + 1] = tmp.loword;
        }
*itakurcs = logabs( *itakurcs);
}
/*
 * Read the Autocorrelation coeff. If ignore is true
 * the dont complain about not finding the file and
 * say that you're intializing the arc's
 */

readtacs (ignore)
int ignore;
{
    int     chn,
            tmp;
    chn = open (tacfile, 0);
    if ((chn <= 0) && ignore) { printf("Tac file not found-creating new);
    read (chn, &tmp, 2);                    /* version */
    if (tmp != 1)
        quit (-1, "TAC file has wrong version number (%d)#", tmp);
    read (chn, &numtac, 2);                 /* number of tacplates */
    if (numtac < 0 || numtac > MAXTPL)
        quit (-1, "TAC file has %d tacplates; quite absurd.#", numtac);

    if (read (chn, &taclbls[1], numtac * 2) != numtac * 2)
        quit (-1, "read error while reading tacplates#");
    read (chn, &tmp, 2);
    if (tmp != 012345)
        quit (-1, "marker in tac file missing#");
    tmp = numtac * 2;
    if (read (chn, &instances[1], tmp) != tmp)
        quit (-1, "read error while reading INSTANCES#");
    read (chn, &tmp, 2);
    if (tmp != 012345)
        quit (-1, "marker missing in tac file#");
    tmp = numtac * NUMCOEFF * 2;
    if (read (chn, &tacs[1][0], tmp) != tmp)
        quit (-1, "read error while reading TACS#");

    printf ("tacfile: %s: %d tacplates#", tacfile, numtac);
    close (chn);
}

main ()
{
    register adcfd, time, i;
    int bufl[WINDOWSIZE + 2];
    int buf2[WINDOWSIZE + 2];
    int arcdata[NPAR + 1];
    int lpcdata[NPAR + 1];
    int n1;
int
        k,windno,h,snum,temp,
       icnt,cnt,
        fd4,fd5,fd6,itac,
        j,l,m,t,fd1,fd2,fd3,fam,utt;
float
        filf[19], maxv,
       energy[128],
        b[768],outval;
char
        file3[64],file4[64],file5[64],file6[64],
        file1[64],file2[64],ch;


/*
        Set up a loop for going through all the files for a single
  speaker.

*/
file1[0] = 0;
getstr("family file prefix... i.e. ds01.01",file1,file1);
fam = (file1[2] - '0');
fam = fam*10 + (file1[3] - '0');
utt = (file1[5] - '0');
utt = utt*10 + (file1[6] - '0');
for (i=0; i<7; i++)
{
file3[i] = file4[i] = file5[i] = file1[i];
file2[i] = file1[i];
}
file3[10] = file4[10] = file5[10] = '0';
file1[10] = file2[10] = '0';
while(fam < 11 )
 {
        while ( utt < 37 )
          {
           sprintf(&file1[2], "%02d.%02d.ad",fam,utt);
           sprintf(&file2[2], "%02d.%02d.rd",fam,utt);
           sprintf(&file3[2], "%02d.%02d.kd",fam,utt);
           sprintf(&file4[2], "%02d.%02d.ld",fam,utt);
           sprintf(&file5[2], "%02d.%02d.id",fam,utt);
           printf("%s#",file1);
           printf("%s#",file2);
           printf("%s#",file3);
           printf("%s#",file4);
           printf("%s#",file5);
           fd2 = wantread(file1,0, "can't find adc file.");
           fd3 = wantwrite(file2,0, "can't create r file.");
           fd4 = wantwrite(file3,0, "can't create k file.");
           fd5 = wantwrite(file4,0, "can't create lpc file.");
           fd6 = wantwrite(file5,0, "can't create ita file.");
           windno = 0;
           time = findspeech(fd2,bufl);
/*     process the utterance - for each 20 mil second frame,
call auto-coorelation procedure, save aut-cor. call solve for
lpc and save k-constants and lpc, call itakura and save ita
metrics. put log energy in location zero of each set of parameters
saved...   r  k  1  i   .....

  and advance time by 10 mil seconds. Stop when 150 mil seconds
  of silence or noise.

*/
           cnt = 0;
           icnt = 0;
           time = time + 100;

           while (((cnt = nextframe(bufl,buf2,time,fd2)) != -1) && (icnt < 16))
            {
              if (cnt == 0)
               icnt = 0;
              else
               icnt = icnt + 1;
             if (windno < 34) icnt = 0;
              temp = engl >> 7;
              temp = logabs(temp);

/*   start calculation of parameters for this window ....

*/

        autoc(buf2, &bufl[1]);
        bufl[0] = temp;
        write(fd3,bufl,32);      /* save 16 parameters. */

        solve( &bufl[1], &buf2[1]);
        ck[0] = temp;            /* save k-constants from lpc */
        write(fd4,ck,30);

        buf2[0] = temp;          /* save energy + 15 LPCs */
        write(fd5,buf2,32);

        itagen( &buf2[1], &bufl[1], &itac);
        bufl[0] = temp;
        bufl[NPAR + 2] = itac;
        write (fd6,bufl,34);

               time = time + 100;
               windno++;
             }
           printf("no. of speech windows is...%d#",windno);
           close(fd2);
           close(fd3);
           close(fd4);
           close(fd5);
           close(fd6);
           utt++;
        };
 fam++;
 utt = 1;
 }
}
/*      find the beginning of speech and return the time in 100's of
microseconds.
*/
findspeech(fd,databuf)
int
        fd,databuf[];
{
int
        thres,
        eng,time,i,j;

time = 0;
noise = 0;
/*      get first 40 milseconds total engergy for threshold of speech,
   assume this sample represents background noise.
*/
adcgetframe(time,databuf,fd);
for (i=1; i < WINDOWSIZE + 1; i++)
noise = noise + abs(databuf[i]);
time = time + WINDOWSIZE;
adcgetframe(time,databuf,fd);
for (i=1; i < WINDOWSIZE + 1; i++)
noise = noise + abs(databuf[i]);
time = time + WINDOWSIZE;
engl = 0;
while ((adcgetframe(time,databuf,fd) != 0) && (engl < noise))
 {
engl = 0;
   for (i = 1; i< 201; i++)
  {
   engl = engl + abs(databuf[i]);
   }
   time = time + WINDOWSIZE;
 }
time = time -200;
return (time);
}
nextframe(buf1,buf2,time,fd2)
int
        buf1[],buf2[],time,fd2;
```

```
{
int
        i;

/*      get a frame of speech, return minus one if at end of file.
  otherwise find total energy and return through global variable
  engl. If engl is over the noise level, then we still have speech,
  otherwise return a one to increment count of silence windows.

*/

if(adcgetframe(time,buf1,fd2) == 0)
return(-1);
engl = 0;
for ( i=1; i < WINDOWSIZE + 1; i++)
 engl = engl + abs(buf1[i]);
prepro(buf1,buf2);
if (engl > noise)
 return(0);
else
 return(1);
}
```

```
#
/*
        This program calculates the difference measure between
sets of utterances. There are 36 words per set. and up to ten
sets. One set must be used for templates. Only nine of the
36 templates can fit in memory at once.

      The itakura distance metrix is used to calculate the distance
between two sets of LPC coefficients.

     The Sakoe-Chiba dynamic programming algorithm is used.

*/


#define WINDOWNO 100
#define NPAR 14
double
    x1,x2,x3;
int
   alast,blast,j1,i1,
   utt,utt2,glflg;


/*   procedure to find minimum value of three floating point numbers.
*/
   float fmin(f1,f2,f3)
    float f1,f2,f3;
   {
     float ans;
      ans = f1;
      if (f2 < ans)
      ans = f2;
      if (f3 < ans)
      ans = f3;
   return(ans);
   if (ans < 0)
    {
     printf("fmin %f#",ans);
     printf("temp %d#",utt);
      printf("unk %d#",utt2);
      printf("j1 %d#",j1);
      printf("i1 %d#",i1);
     glflg = 1;
    }
    return(ans);
   }


/*
   Calculate the delta differences between the filter energies.
*/
```

```
long delta(a,b)
int
   a[],b[];
{
long
  ans;

   ans = 0;
   ans = b[16] + logabs(dot(&a[1],&b[1],NPAR + 1)) - a[16];
return(ans);
}


/*
   Dynamic Programming Algorith - Version 1.0
     Path constraint defines window.
       Variable Definitions:

   a = unknown vector
   b = template vector
*/

float score(a,b)
int
        a[],b[];
{
  float fmin();
long delta();
int
    midlen,
    n,i,j,k,unk;
float
     g[19][100],inf,temp;

/*
   Let midlen be the nxn dimension of dynamic matrix - take
shortest length utteracne to determine midlen...
*/

   midlen = j1-10;
   if (i1 < j1)
   midlen = i1-10;
   if (midlen < 10)
   return(inf);
  inf=9999999;
  unk=0;


/*
    Initialize row zero and column zero, first ten elements
*/
```

```
   g[0][0] = delta(&a[unk],&b[unk]);
   for (i = 1; i < 10; i++)
    {
      unk =+ 17;
     g[i][0] = g[i-1][0] + 3*delta(&a[unk],&b[0]);
     g[0][i] = g[0][i-1] + 3*delta(&a[0],&b[unk]);
    }

/*
   Compute columne one which has length 11.
*/


g[1][1] = g[0][0] + 2*delta(&a[17],&b[17]);
 n = 11;
  for (i = 2; i < n; i++)
   {
    temp = delta(&a[17*i],&b[17]);
    g[i][1] = fmin(g[i-2][0] + 2*delta(&a[17*(i-1)],&b[17]) +
      temp, g[i-1][0] + 2*temp, inf);
   }

/*
   Compute the next eight columns which increase by one for
exh new column.
*/

n = 11;
  for (j=2; j< 10; j++)
   {
    temp = delta(&a[17],&b[j*17]);
    g[1][j] = fmin(inf,g[0][j-1] + 2*temp, g[0][j-2] + 2*delta(
        &a[17],&b[(j-1)*17]) + temp);
     for (i=2; i < n; i++)
      {
        temp = delta(&a[i*17],&b[j*17]);
        g[i][j] = fmin(g[i-2][j-1] + 2*delta(&a[(i-1)*17], &b[j*17])
          + temp, g[i-1][j-1] + 2*temp, g[i-1][j-2] +
           2*delta(&a[i*17],&b[(j-1)*17]) + temp);
      }
  temp = delta(&a[n*17],&b[j*17]);
   g[n][j] = fmin(g[n-2][j-1] + 2*delta(&a[(n-1)*17],&b[j*17]) + temp,
         g[n-1][j-1] + 2*temp, inf);
  n++;
  }

/*
   Compute the next x columns.
*/
   g[18][8] = g[17][8];
  j= 10;
  unk = (j-9) * 17;
```

```
   while(j < midlen)
  {
    temp = delta(&a[unk],&b[j*17]);
    g[0][j] = fmin(inf, g[0][j-1] + 2*temp, g[1][j-2] +
              2*delta(&a[unk],&b[(j-1)*17]) + temp);
    i = 1;
    unk =+ 17;
  while (i < 18)
    {
     temp = delta(&a[unk],&b[j*17]);
     g[i][j]=fmin(g[i-1][j-1] + 2*delta(&a[unk-17],&b[j*17]) + temp,
             g[i][j-1] + 2*temp, g[i+1][j-2] +
             2*delta(&a[unk],&b[(j-1)*17]) + temp);
     i++;
     unk =+ 17;
    }
     temp = delta(&a[unk],&b[j*17]);
     g[18][j]=fmin(g[17][j-1] + 2*delta(&a[unk-17], &b[j*17]) + temp,
              g[18][j-1] + 2*temp, inf);
    j++;
    unk = (j-9)*17;
  }

/*
   Compute the last ten columns. Each successive columns decreases in
 length by one.
*/

  n = 18;
  j = midlen;
  unk = (j-9)*17;
  while (j < midlen + 10)
  {
   temp = delta(&a[unk],&b[j*17]);
   g[0][j] = fmin(inf,g[0][j-1] + 2*temp, g[1][j-2]+2*delta(
             &a[unk],&b[(j-1)*17]) + temp);
   i = 1;
   unk =+ 17;
  while (i < n)
   {
    temp = delta(&a[unk],&b[j*17]);
    g[i][j] = fmin(g[i-1][j-1] + 2*delta(&a[unk-17],&b[j*17]) + temp,
              g[i][j-1] + 2*temp, g[i+1][j-2] + 2*delta(&a[unk],
              &b[(j-1)*17]) + temp);
      i++;
    unk =+ 17;
   }
    temp = delta(&a[unk],&b[j*17]);
    if (n != 18)
    g[n][j] = fmin(g[n-1][j-1]+2*delta(&a[unk-17],&b[j*17]) + temp,
   g[n][j-1] + 2*temp,g[n+1][j-2] + 2*delta(&a[unk],&b[(j-1)*17]) + temp);
   if (n==18)
   g[18][j] = fmin(g[n-1][j-1] + 2*delta(&a[unk-17],&b[j*17]) + temp,
     g[n][j-1] + 2*temp, inf);
  j++;
  unk = (j-7)*17;
    n--;
  }

  midlen = midlen + 9;
   temp = g[0][midlen];
   for (i=1; i < 10; i++)
    {
      if(g[i][midlen] < temp)
      temp = g[i][midlen];
      if (g[9+i][midlen-i] < temp)
      temp = g[9+i][midlen-i];
    }
     temp = g[9][midlen];
    temp = temp/midlen;
  return(temp);
}


/*  calculate the delta differences between the fiter energies..

*/




main()
{
int
    ish,fd3,
   lpc[17],
   ie,te,je,be,se,
  min2,min3,
   tplen[9],
      nwg,
        fd1,fd2,j,k,
      temp[11420],cnt,ind,ms,jj,
        unk[17*WINDOWNO + 19],in,i,bs;
float
      max,min,
    finx,fpt,
        sc[36][36];
char
   file3[64],
        file1[64],file2[64];
   g1flg = 0;
file1[0] = 0;
getstr("template file prefix, i.e. ds01",file1,file1);
utt = 0;
```

```
getstr("unknown family name.. i.e. ds02", file2,file2);

for (i=0; i<4; i++)
   file3[i] = file2[i];
printf("found all input ok, up and going.");
for (ms=0; ms < 6; ms++)
 {


    for(i=0; i < 11400; i++)
     {
      temp[i] = 0;
     }
for (i=0; i < 6; i++)
 {
   ind = i*1700;
   utt++;
   sprintf(&file1[4],".%02d.id",utt);
   fd1 = wantread(file1,0,"can't find template file");
   k = 0;
     tplen[i] = 0;
      nwg = 0;
   while (( read(fd1,&temp[ind],34) != 0) && (k < 100 )  )
    {
   tplen[i] =+ 1;
      ind =+ 17;
k++;
    }
   close(fd1);
 }

utt2 = 0;
for (i=0; i < 36; i++)
  {
    utt2++;
    sprintf(&file2[4], ".%02d.rd",utt2);
    sprintf(&file3[4], ".%02d.ld",utt2);
    fd2 = wantread(file2,0,"can't find unknown template file");
    fd3 = wantread(file3,0, "can:t find unknown lpc file");
    in = 0;
    k = 0;
   nwg = 0;
    while ( (read(fd2,&unk[in],32) != 0) && (k < 100) )
     {
   read(fd3,&lpc[0],32);
        k++;
   unk[in+16] = dot(&lpc[1],&unk[in+1],NPAR + 1);
   unk[in+16] = logabs(unk[in+16]);
   for (ish = 1; ish <= NPAR + 1; ish++)
   unk[ish+in] =>> 2;
        in =+ 17;
   }
jl = in/17;
 if (ms == 5)
  {
  }
  close(fd2);
 close(fd3);
  while (in < 17*WINDOWNO)
   {
     unk[in] = 0;
     in++;
   }
  for (j=0; j<6; j++)
    {
 il = tplen[j];
    ind = j*1700;
 ie = 1;
 if (ie == 0)
 {
 if (il < jl)
 {
  max = jl;
  min = il;
  finx = min/max;
  fpt = min -1;
  for (ie = jl-1; ie > -1; ie--)
  {
   te = ie * 17;
   je = fpt;
   be = je * 17;
   for (se = 0; se < 17; se++)
   temp[te+se+ind] = temp[be+se+ind];
   fpt = fpt -finx;
   if (fpt < 0)
   fpt = 0;
  }
  il = jl;
 }
 if (jl < il)
 {
  max = il;
  min = jl;
  finx = min/max;
  fpt = min - 1;
  for (ie = il-1; ie > -1; ie--)
  {
   te = ie * 17;
   je = fpt;
   be = je * 17;
   for (se=0; se < 17; se++)
   unk[te+se] = unk[be +se];
   fpt = fpt - finx;
```

```
     if (fpt < 0)
     fpt = 0;
    }
    jl = il;
   }
   }
       jj = j + ms*6;
        sc[jj][i] = score(&unk[0],&temp[ind]);
      }
  }

}

    max = 0;
    nwg = 0;
for (i=0; i < 36; i++)
  {
    printf("#");
   bs = 0;
   max = 0;
   for (j=0; j < 36; j++)
   {
  if (sc[j][i] > max)
   max = sc[j][i];
  }
   min = max;
   finx = max;
   fpt = max;
   for (j = 0; j< 36; j++)
   {
     if (sc[j][i] < min)
     {
      min = sc[j][i];
      bs = j;
     }
   }
   for (j=0; j < 36; j++)
   {
     if ((sc[j][i] < finx) && (j != bs))
      {
       finx = sc[j][i];
       min2 = j;
       }
     }
    for (j=0; j < 36; j++)
    {
     if (( sc[j][i] < fpt) && (j != bs) && ( j != min2))
     {
       fpt = sc[j][i];
       min3 = j;
     }
```

```
      }
  printf("the range is...%f#", max-min);
  printf("the best is.. %d#",bs+1);
    printf("best score is.. %f#", min);
  printf("second best.. %d#", min2+1);
   printf("second best score.. %f#", finx);
  printf("third best.. %d#", min3+1);
    printf("the third best score is.. %f#", fpt);
   printf ("the best match should be %d#", i + 1);
    if ( bs != i)
   nwg++;
  }
    printf("the number wrong is ... %d#",nwg);
for (i=0; i<100; i++)
{
 printf("fill-up the buffer");
}
}
```

APPENDIX D

PROGRAM LISTINGS FOR FILTER MODEL

```
#
/*
 *
*  This program calculates 1/3 octave energies using FFT
 *
*     Parts of this software are taken from the CMU Harpy Speech
*  understanding system running on the 11/40.
*
 *
 **************
 */

/* #define DEBUG 1  /* un-comment to turn debugging on */

#ifdef DEBUG
#endif

#include <stdio.h>
#include <math.h>
#include "label.h"
#include "iobuf.h"
#include "types.h"

extern  long hmul ();

/*
 *********** All this stuff is in labpac.h ***************
 */
int
    start2,     /* starting location in speech file - by frame no. */
        numtpl,             /* number of templates */
        lbls[MAXTPL + 1],    /* label names in radix 50 */
        itcs[MAXTPL + 1],    /* itakura c constants for each template */
        tpls[MAXTPL + 1][NPAR + 1];  /* itakura b constants */

int     numtac,
        taclbls[MAXTPL + 1],
        instances[MAXTPL],
        tacs[MAXTPL +1][NPAR +1];


float
    noise,
  engl3,
   engl2,
        engl;
char uttfile[64],
     adcfile[64],
     tplfile[64],
     tacfile[64],
     phnfile[64],
     hslfile[64],
     segfile[64];
/*
 **********************************************************
 */


int     headersize;          /* adc header size in words */

 /* table of 11 bit mantissas for log routine */

int     mantis[256]
{
    0, 12, 23, 34, 46, 57, 68, 80, 91, 102,
    113, 124, 135, 146, 157, 168, 179, 190, 201, 212,
    222, 233, 244, 254, 265, 275, 286, 296, 307, 317,
    327, 338, 348, 358, 368, 379, 389, 399, 409, 419,
    429, 439, 449, 459, 469, 478, 488, 498, 508, 517,
    527, 537, 546, 556, 566, 575, 585, 594, 603, 613,
    622, 631, 641, 650, 659, 669, 678, 687, 696, 705,
    714, 723, 732, 741, 750, 759, 768, 777, 786, 795,
    803, 812, 821, 830, 838, 847, 856, 864, 873, 882,
    890, 899, 907, 916, 924, 933, 941, 949, 958, 966,
    974, 983, 991, 999, 1007, 1016, 1024, 1032, 1040, 1048,
    1056, 1064, 1072, 1080, 1088, 1096, 1104, 1112, 1120, 1128,
    1136, 1144, 1151, 1159, 1167, 1175, 1183, 1190, 1198, 1206,
    1213, 1221, 1229, 1236, 1244, 1251, 1259, 1266, 1274, 1281,
    1289, 1296, 1304, 1311, 1319, 1326, 1333, 1341, 1348, 1355,
    1363, 1370, 1377, 1384, 1392, 1399, 1406, 1413, 1420, 1427,
    1435, 1442, 1449, 1456, 1463, 1470, 1477, 1484, 1491, 1498,
    1505, 1512, 1519, 1525, 1532, 1539, 1546, 1553, 1560, 1566,
    1573, 1580, 1587, 1594, 1600, 1607, 1614, 1620, 1627, 1634,
    1640, 1647, 1653, 1660, 1667, 1673, 1680, 1686, 1693, 1699,
    1706, 1712, 1719, 1725, 1732, 1738, 1744, 1751, 1757, 1764,
    1770, 1776, 1783, 1789, 1795, 1801, 1808, 1814, 1820, 1826,
    1833, 1839, 1845, 1851, 1857, 1863, 1870, 1876, 1882, 1888,
    1894, 1900, 1906, 1912, 1918, 1924, 1930, 1936, 1942, 1948,
    1954, 1960, 1966, 1972, 1978, 1984, 1990, 1996, 2001, 2007,
    2013, 2019, 2025, 2031, 2036, 2042
};


 /* 200 point hamming window.  (1.0==077777)  */

int     hw[201]
{
    0,
    2621, 2628, 2651, 2688, 2740, 2806, 2888, 2984, 3094, 3219,
    3359, 3512, 3679, 3861, 4055, 4264, 4485, 4720, 4967, 5227,
    5500, 5784, 6080, 6387, 6706, 7036, 7376, 7726, 8086, 8455,
    8834, 9221, 9617, 10021, 10432, 10851, 11276, 11708, 12145, 12588,
    13036, 13488, 13945, 14406, 14869, 15336, 15805, 16275, 16747, 17220,
    17694, 18167, 18640, 19112, 19583, 20052, 20518, 20982, 21442, 21899,
    22351, 22799, 23242, 23680, 24111, 24537, 24955, 25366, 25770, 26166,
    26553, 26932, 27301, 27661, 28012, 28352, 28681, 29000, 29307, 29604,
    29888, 30160, 30420, 30667, 30902, 31124, 31332, 31527, 31708, 31875,
    32029, 32168, 32293, 32403, 32500, 32581, 32648, 32700, 32737, 32759,
    32767, 32759, 32737, 32700, 32648, 32581, 32500, 32404, 32293, 32168,
    32029, 31875, 31708, 31527, 31332, 31124, 30902, 30668, 30420, 30160,
    29888, 29604, 29308, 29000, 28681, 28352, 28012, 27662, 27301, 26932,
    26553, 26166, 25770, 25366, 24955, 24537, 24111, 23680, 23242, 22799,
    22351, 21899, 21442, 20982, 20518, 20052, 19583, 19112, 18640, 18167,
    17694, 17220, 16747, 16275, 15805, 15336, 14869, 14406, 13945, 13489,
    13036, 12588, 12145, 11708, 11276, 10851, 10432, 10021, 9617, 9222,
    8834, 8455, 8086, 7726, 7376, 7036, 6706, 6387, 6080, 5784,
    5500, 5227, 4967, 4720, 4485, 4264, 4055, 3861, 3679, 3512,
    3359, 3219, 3094, 2984, 2888, 2806, 2740, 2688, 2651, 2628
};


/*
 * logabs returns the base 2 log of the absolute value of
 * its argument.  the argument should have the binary point
 * to the right of bit 8.  the result has the point to the
 * right of bit 11.
 */

logabs (n)
int     n;
{
    register int    ch,       /* characteristic */
                    num;

    if ((num = n) == 0)
        return (NEGINFINITY);
    if (num < 0)
        num = -num;
    ch = 0;
    if ((num & 0177400) == 0)
        while ((num & 0177400) != 0400)
        {
            ch =- 04000;
            num =* 2;
        }
    else
        while ((num & 0177400) != 0400)
        {
            ch =+ 04000;
            num =/ 2;
        }
    return (ch | mantis[num & 0377]);
};
```

```
/* prepro preprocesses the input signal to make it suitable
 * for parameter calculation.  the preprocessing involves
 * scaling the input frame to 14 bits, differencing the
 * scaled signal with a preemphasis factor of .92, scaling
 * the differenced signal to 13 bits, and finally applying
 * a hamming window.  the resulting preprocessed signal
 * contains 12 bits of significance.
 */

prepro (d, p)
int     d[],
        p[];
{
    int     maxv,
            snum,
            last,
            abs_d,
            new;
    register int    i,
                   *iptr;
    long 1;


    /*
     * scale and difference the input frame.
     * a preempasis factor of .92 is used in the differences
     */
     last = d[1];
    for (i = 1; i < WINDOWSIZE; i++)
    {
      new = d[i+1];
        1 = hmul (last, 072703);
        1 =<< 1;
        p[i] = new - 1.hiword;
        last = new;
    };

    /*
     * determine how much to shift the differenced signal
     * so that it contains 13 bits of significance.
     */

    /*
     * scale up the differenced signal and apply a hamming window.
     * the results will have 12 bits of significance.
     */
    for (i = 1; i < WINDOWSIZE; i++)
    {
       1 = hmul (p[i], hw[i]);
        p[i] = 1.hiword;
    }
    p[WINDOWSIZE] = 0;

};



/* adcgetframe takes a time in 100's of micro seconds */
/* a buffer for the data of windowsize and the adc    */
/* channel                                            */

adcgetframe (tim, fr, adcchn)
int     fr[];
{
    register    offset,
                err,
                i;
     long
                 loff;

    offset = tim;
    if (offset < 0)
        offset = 0;
    offset =* 2;        /* make byte address */
    offset =+ 2*headersize;
    loff = offset;
    lseek (adcchn, loff, 0);
   if ( read (adcchn, &fr[1], WINDOWSIZE * 2) == 0)
    return(0);
#ifdef DEBUG
    printf ("#Input frame at time = %d#", tim);
    printf ("header size = % words#", headersize/2);
    printf ("word address = %d#", offset/2);
#endif
    /*
*  shift data right two bits to fit following procedure formats..
     */
for (i = 1; i <= WINDOWSIZE; i++)
  fr[i] = fr[i] >> 2;
return(1);
}

main ()
{
    register adcfd, time, i;
    int buf1[WINDOWSIZE + 2];
    int buf2[WINDOWSIZE + 2];
    int arcdata[NPAR + 1];
    int lpcdata[NPAR + 1];
    int nl;
int
   fd4,
        k,windno,h,snum,temp,
       filter[19],icnt,cnt,
        j,l,m,t,fd1,fd2,fd3,fam,utt;
float
        filf[19], maxv,
       energy[128],
        b[768],outval;
char
    file3[64],
        file1[64],file2[64],ch;


/*      initialize the array for shaping fourier energies into 1/3
octave filters, remeber, there are eighteen filters.
*/

sprintf(file1,"skirt.flt");
fd1 = wantread(file1,0, "skirt.flt");
t = 0;
for (i = 1; i < 7; i++)
  {
   t = (i-1) * 128;
   read(fd1,&b[t],512);
  }
/*
        Set up a loop for going through all the files for a single
  speaker.

*/
file1[0] = 0;
getstr("family file prefix... i.e. ds01.01",file1,file1);
fam = (file1[2] - '0');
fam = fam*10 + (file1[3] - '0');
utt = (file1[5] - '0');
utt = utt*10 + (file1[6] - '0');
for (i=0; i<7; i++)
file2[i] = file1[i];
file1[10] = file2[10] = '0';
while(fam < 11 )
 {
        while ( utt < 37 )
          {
         sprintf(&file1[2], "%02d.%02d.ad",fam,utt);
           sprintf(&file2[2], "%02d.%02d.fd",fam,utt);
           printf("%s#",file1);
           printf("%s#",file2);
           fd2 = wantread(file1,0, "can't find adc file.");
           fd3 = wantwrite(file2,0, "can't create output file.");
           windno = 0;
           time = findspeech(fd2,buf1);
/*     process the utterance - for each 20 mil second frame,
  call fft, shape into 1/3 octave filters, save in .f.dat file
  and advance time by 10 mil seconds. Stop when 150 mil seconds
  of silence or noise.

*/
           cnt = 0;
           icnt = 0;

           while (((cnt = nextframe(buf1,buf2,time,fd2)) != -1) && (icnt < 16))
            {
              if (cnt == 0)
               icnt = 0;
              else
               icnt = icnt + 1;
             if (windno < 40) icnt = 0;
              fft(&buf2[1],energy);
              m = 0;
   temp = (engl2/3200)/20;
              filter[0] = logabs(temp);
              maxv = 0;
              for (i = 1; i< 19; i++)
               {
                 filf[i] = 0;
                 l = b[m];
                 m++;
                 t = b[m];
                 for(k =l; k < l+t; k++)
                  {
                   m++;
                   filf[i] = energy[k] * b[m] + filf[i];
                  }
                m++;
               if( maxv < filf[i]) maxv = filf[i];
              }

  snum = 0;
  while(maxv < 16384)
  {
   maxv = maxv * 2;
   snum++;
  }
  maxv = 1;
  while (snum != 0)
  {
   maxv = maxv * 2;
   snum--;
  }
  for (i = 1; i< 19; i++)
   {
    temp = filf[i] *maxv;
  filter[i] = logabs(temp);
   }
              time = time + 100;
              write(fd3,filter,38);
```

```
               windno++;
             }
   printf("start is..%d#",start2);
   printf("no. of wind.. %d#",windno);
           close(fd2);
           close(fd3);
           utt++;
        };
 fam++;
 utt = 1;
 }
}
/*      find the beginning of speech and return the time in 100's of
microseconds.
*/
findspeech(fd,databuf)
int
        fd,databuf[];
{
int
        tcnt,
        thres,
        eng,time,i,j;

time = 0;
noise = 0;
/*      get first 40 milseconds total engergy for threshold of speech,
   assume this sample represents background noise.
*/
adcgetframe(time,databuf,fd);
for (i=1; i < WINDOWSIZE + 1; i++)
{
  engl2 = databuf[i];
  noise = noise + engl2 *engl2;
    }
time = time + WINDOWSIZE;
adcgetframe(time,databuf,fd);
for (i=1; i < WINDOWSIZE + 1; i++)
{
   engl2 = databuf[i];
   noise = noise + engl2 * engl2;
  }
time = time + WINDOWSIZE;
engl = 0;
   start2 = 1;
  noise = noise*2;
tcnt = 0;

while ((adcgetframe(time,databuf,fd) != 0) && (engl < noise))
 {
engl = 0;
   start2++;
   for (i = 1; i< 201; i++)
  {
   engl2 = databuf[i];
   engl = engl + engl2*engl2;
   }
   time = time + WINDOWSIZE;
     if (engl > noise)
     tcnt = tcnt + 1;
     if (tcnt < 1)
     engl = 0;
 }
time = time -200;
return (time);
}
nextframe(bufl,buf2,time,fd2)
int
        bufl[],buf2[],time,fd2;
{
int
        i;

/*      get a frame of speech, return minus one if at end of file.
  otherwise find total energy and return through global variable
  engl. If engl is over the noise level, then we still have speech,
  otherwise return a one to increment count of silence windows.

*/

if(adcgetframe(time,bufl,fd2) == 0)
return(-1);
engl = 0;
   engl2 = 0;
for ( i=1; i < WINDOWSIZE + 1; i++)
{
 engl = engl + abs(bufl[i]);
  engl3 = bufl[i];
   engl2 = engl2 + engl3 * engl3;
}
   engl = engl2;
prepro(bufl,buf2);
if (engl > noise)
 return(0);
else
 return(1);
}
```

```
#
/*
        This program calculates the difference measure between
sets of utterances. There are 36 words per set. and up to ten
sets. One set must be used for templates. Only half of the
36 templates can fit in memory at once.
      The program opperates on 1/3 octave filter energies and uses
 a Euclidian distance measure.
*/


#define WINDOWNO 100
double
    x1,x2,x3;
int
   alast,blast,j1,i1,
   utt,utt2,glflg;


/*   procedure to find minimum value of three floating point numbers.
*/
   float fmin(f1,f2,f3)
    float f1,f2,f3;
   {
     float ans;
      ans = f1;
      if (f2 < ans)
      ans = f2;
      if (f3 < ans)
      ans = f3;
   if (ans < 0)
    {
     printf("fmin %f#",ans);
     printf("temp %d#",utt);
      printf("unk %d#",utt2);
      printf("j1 %d#",j1);
      printf("i1 %d#",i1);
     glflg = 1;
    }
    return(ans);
   }


/*
   Calculate the delta differences between the filter energies.
*/

long delta(a,b)
int
   a[],b[];
{
   double log();
```

```
int
   tp1,tp2;
float
    emph;
long
  ans;
register int
   i;

   ans = 0;
emph = 1;
for(i=1; i<18; i++)
 {
   tp1 = a[i+1] - a[i];
   tp2 = b[i+1] - b[i];
   tp2 = abs(tp1-tp2);
      if (i>12)
     tp2 = tp2 * 2;
    ans = ans + tp2;
 }
return(ans);
}


/*
   Dynamic Programming Algorith - Version 1.0

       Use the Sakoe-Chiba Dyanamic programming algoirthm - see
  IEEE ASSP Feb 1978




   Definition of variables

   a = unknown vector
   b = template vector
*/

float score(a,b)
int
        a[],b[];
{
  float fmin();
long delta();
int
    midlen,
    n,i,j,k,unk;
float
     g[19][100],inf,temp;

/*
```

```
   Let midlen be the nxn dimension of dynamic matrix - take
shortest length utteracne to determine midlen...
*/

   midlen = j1-10;
   if (i1 < j1)
   midlen = i1-10;
   if (midlen < 10)
   return(inf);
  inf=9999999;
  unk=0;


/*
    Initialize row zero and column zero, first ten elements
*/
   g[0][0] = delta(&a[unk],&b[unk]);
   for (i = 1; i < 10; i++)
    {
      unk =+ 19;
     g[i][0] = g[i-1][0] + 3*delta(&a[unk],&b[0]);
     g[0][i] = g[0][i-1] + 3*delta(&a[0],&b[unk]);
    }

/*
   Compute columne one which has length 11.
*/


g[1][1] = g[0][0] + 2*delta(&a[19],&b[19]);
 n = 11;
  for (i = 2; i < n; i++)
   {
    temp = delta(&a[19*i],&b[19]);
    g[i][1] = fmin(g[i-2][0] + 2*delta(&a[19*(i-1)],&b[19]) +
      temp, g[i-1][0] + 2*temp, inf);
   }

/*
   Compute the next eight columns which increase by one for
exh new column.
*/

n = 11;
  for (j=2; j< 10; j++)
   {
    temp = delta(&a[19],&b[j*19]);
    g[1][j] = fmin(inf,g[0][j-1] + 2*temp, g[0][j-2] + 2*delta(
        &a[19],&b[(j-1)*19]) + temp);
     for (i=2; i < n; i++)
      {
        temp = delta(&a[i*19],&b[j*19]);
        g[i][j] = fmin(g[i-2][j-1] + 2*delta(&a[(i-1)*19], &b[j*19])
          + temp, g[i-1][j-1] + 2*temp, g[i-1][j-2] +
           2*delta(&a[i*19],&b[(j-1)*19]) + temp);
      }
  temp = delta(&a[n*19],&b[j*19]);
   g[n][j] = fmin(g[n-2][j-1] + 2*delta(&a[(n-1)*19],&b[j*19]) + temp,
         g[n-1][j-1] + 2*temp, inf);
  n++;
  }

/*
   Compute the next x columns.
*/
   g[18][8] = g[17][8];
  j= 10;
  unk = (j-9) * 19;
   while(j < midlen)
  {
    temp = delta(&a[unk],&b[j*19]);
    g[0][j] = fmin(inf, g[0][j-1] + 2*temp, g[1][j-2] +
              2*delta(&a[unk],&b[(j-1)*19]) + temp);
    i = 1;
    unk =+ 19;
  while (i < 18)
    {
     temp = delta(&a[unk],&b[j*19]);
     g[i][j]=fmin(g[i-1][j-1] + 2*delta(&a[unk-19],&b[j*19]) + temp,
             g[i][j-1] + 2*temp, g[i+1][j-2] +
             2*delta(&a[unk],&b[(j-1)*19]) + temp);
     i++;
     unk =+ 19;
    }
     temp = delta(&a[unk],&b[j*19]);
     g[18][j]=fmin(g[17][j-1] + 2*delta(&a[unk-19], &b[j*19]) + temp,
              g[18][j-1] + 2*temp, inf);
    j++;
    unk = (j-9)*19;
  }

/*
   Compute the last ten columns. Each successive columns decreases in
 length by one.
*/

  n = 18;
  j = midlen;
  unk = (j-9)*19;
  while (j < midlen + 10)
  {
   temp = delta(&a[unk],&b[j*19]);
   g[0][j] = fmin(inf,g[0][j-1] + 2*temp, g[1][j-2]+2*delta(
             &a[unk],&b[(j-1)*19]) + temp);
   i = 1;
   unk =+ 19;
  while (i < n)
   {
    temp = delta(&a[unk],&b[j*19]);
    g[i][j] = fmin(g[i-1][j-1] + 2*delta(&a[unk-19],&b[j*19]) + temp,
              g[i][j-1] + 2*temp, g[i+1][j-2] + 2*delta(&a[unk],
              &b[(j-1)*19]) + temp);
      i++;
    unk =+ 19;
   }
    temp = delta(&a[unk],&b[j*19]);
    if (n != 18)
    g[n][j] = fmin(g[n-1][j-1]+2*delta(&a[unk-19],&b[j*19]) + temp,
   g[n][j-1] + 2*temp,g[n+1][j-2] + 2*delta(&a[unk],&b[(j-1)*19]) + temp);
   if (n==18)
   g[18][j] = fmin(g[n-1][j-1] + 2*delta(&a[unk-19],&b[j*19]) + temp,
     g[n][j-1] + 2*temp, inf);
  j++;
  unk = (j-9)*19;
    n--;
  }

  midlen = midlen + 9;
   temp = g[0][midlen];
   for (i=1; i < 10; i++)
    {
      if(g[i][midlen] < temp)
      temp = g[i][midlen];
      if (g[9+i][midlen-i] < temp)
      temp = g[9+i][midlen-i];
    }
     temp = g[9][midlen];
    temp = temp/midlen;
  return(temp);
}


/*  calculate the delta differences between the fiter energies..

*/




main()
{
int
   ie,te,je,be,se,
  min2,min3,
   tplen[9],
      nwg,
        fd1,fd2,j,k,
      temp[11420],cnt,ind,ms,jj,
        unk[19*WINDOWNO + 19],in,i,bs;
float
      max,min,
    finx,fpt,
        sc[36][36];
char
        file1[64],file2[64];
   glflg = 0;
file1[0] = 0;
getstr("template file prefix, i.e. ds01",file1,file1);
utt = 0;

getstr("unknown family name.. i.e. ds02", file2,file2);

printf("found all input ok, up and going.");
for (ms=0; ms < 6; ms++)
 {


    for(i=0; i < 11400; i++)
     {
      temp[i] = 0;
     }
for (i=0; i < 6; i++)
 {
   ind = i*1900;
   utt++;
   sprintf(&file1[4],".%02d.fd",utt);
   fd1 = wantread(file1,0,"can't find template file");
   k = 0;
     tplen[i] = 0;
      nwg = 0;
   while (( read(fd1,&temp[ind],38) != 0) && (k < 80 )  )
    {
   tplen[i] =+ 1;
      ind =+ 19;
k++;
    }
   close(fd1);
 }

utt2 = 0;
for (i=0; i < 36; i++)
  {
    utt2++;
    sprintf(&file2[4], ".%02d.fd",utt2);
    fd2 = wantread(file2,0,"can't find unknown template file");
    in = 0;
    k = 0;
   nwg = 0;
  while ( (read(fd2,&unk[in],38) != 0) && (k < 100) )
   {
      k++;
      in =+ 19;
   }
jl = in/19;
 if (ms == 5)
  {
  }
  close(fd2);
  while (in < 19*WINDOWNO)
   {
     unk[in] = 0;
     in++;
   }
  for (j=0; j<6; j++)
    {
 il = tplen[j];
    ind = j*1900;
 ie = 1;
 if (ie == 0)
 {
 if (il < jl)
 {
  max = jl;
  min = il;
  finx = min/max;
  fpt = min -1;
  for (ie = jl-1; ie > -1; ie--)
  {
   te = ie * 19;
   je = fpt;
   be = je * 19;
   for (se = 0; se < 19; se++)
   temp[te+se+ind] = temp[be+se+ind];
   fpt = fpt -finx;
   if (fpt < 0)
   fpt = 0;
  }
  il = jl;
 }
 if (jl < il)
 {
  max = il;
  min = jl;
  finx = min/max;
  fpt = min - 1;
  for (ie = il-1; ie > -1; ie--)
  {
   te = ie * 19;
   je = fpt;
   be = je * 19;
```

```
     for (se=0; se < 19; se++)
     unk[te+se] = unk[be +se];
     fpt = fpt - finx;
     if (fpt < 0)
     fpt = 0;
    }
    jl = il;
   }
   }
       jj = j + ms*6;
        sc[jj][i] = score(&unk[0],&temp[ind]);
      }
  }

}

    max = 0;
    nwg = 0;
for (i=0; i < 36; i++)
  {
    printf("#");
   bs = 0;
   max = 0;
   for (j=0; j < 36; j++)
   {
    if (sc[j][i] > max)
    max = sc[j][i];
       }
    min = max;
    finx = max;
    fpt = max;
   for (j=0; j< 36; j++)
   {
     if (sc[j][i] < min)
    {
      min = sc[j][i];
    bs = j;
    }
   }
   for (j=0; j < 36; j++)
   {
    if((sc[j][i] < finx) && (j != bs))
    {
      finx = sc[j][i];
      min2 = j;
    }
   }
   for (j=0; j < 36; j++)
   {
    if ((sc[j][i] < fpt) && (j != bs) && (j != min2))
     {
      fpt = sc[j][i];
      min3 = j;
     }
   }
  printf("the range is...%f#", max-min);
  printf("the best is.. %d#",bs+1);
   printf(" the best score is.. %f#", min);
  printf("second best.. %d#", min2+1);
  printf("second best score .. %f#", finx);
  printf("third best.. %d#", min3+1);
   printf("the third best score is ... %f#", fpt);
   printf ("the best match should be %d#", i + 1);
    if ( bs != i)
   nwg++;
  }
    printf("the number wrong is ... %d#",nwg);
for (i=0; i<100; i++)
{
 printf("fill-up the buffer");
}
}
```

```
#
#include "sin.tab"
#include "cos.tab"

/*
 * Fast Fouier Transform done mostly with integers
     written by Fil Alevea at CMU for Rick Parfitt
 */

#define PTS     256
#define LNPTS   8
#define NB2     128

extern hmul();
extern double sqrt();
long (*hlmul)() &hmul;

fft (x, e)
int  *x;
float *e;
{
    register k1,k2,k3;                  /* these are accesed the most */
    int     j,
            i2j,
            n2j,
            n2,
            n1,
            i,
            in2j,
            k,
            k4,
            c,
            s,
            xx,
            xy,
            y[PTS],
            x1[PTS],
            y1[PTS];
    int *xptr,
        *yptr,
        *x1ptr,
        *y1ptr,
        *tmp;

    float *eptr;
    double hack;

    xptr = x;
    yptr = y;
    x1ptr = x1;
    y1ptr = y1;
```

```
smear (y, PTS, 0);

for (j = 0; j < LNPTS; j++)
{
    i2j = (2 << j);
    n2j = (PTS >> (j + 1));
    n2 = (i2j >> 1);
    n1 = n2j;

    for (i = 0; i < n2; i++)
    {
        in2j = i * n2j;
        c = cos [in2j];
        s = sin [in2j];

        k1 = in2j;
        k2 = in2j << 1;
        k3 = k2 + n2j;
        k4 = k1 + NB2;

        for (k = 0; k < n1; k++)
        {
            xx = hmul( xptr[k3] << 1, c) - hmul( yptr[k3] << 1, s);
            xy = hmul( yptr[k3] << 1, c) + hmul( xptr[k3] << 1, s);

            x1ptr[k1] = (xptr[k2] + xx) >> 1;
            y1ptr[k1] = (yptr[k2] + xy) >> 1;

            x1ptr[k4] = (xptr[k2] - xx) >> 1;
            y1ptr[k4] = (yptr[k2] - xy) >> 1;

            k1++;  k2++;  k3++;  k4++;
        }
    }

    tmp = xptr;
    xptr = x1ptr;
    x1ptr = tmp;

    tmp = yptr;
    yptr = y1ptr;
    y1ptr = tmp;
}

eptr = e;
for (i = 0; i < NB2; i++)
{
    hack  = (*hlmul)( *xptr,*xptr) + (*hlmul)( *yptr,*yptr);
    *eptr++ = sqrt(hack);
    yptr++;
    xptr++;
}
```

```
/*
    Program to generate data for use by mskirt.c, a program to generate a
 data file of filter shapes.
*/
main()   {
float
        c[128], freq[18],f1;
int
        j, i, fd1, fd2;
char
        filename[64];

filename[0]=0;
fd1= wantwrite(filename,0,"skirt data file?");

freq[0] = 114;
freq[1] = 143;
freq[2] = 179;
freq[3] = 230;
freq[4]=287;
freq[5]=358;
freq[6]=458;
freq[7]=573;
freq[8]=717;
freq[9]=917;
freq[10]=1150;
freq[11]=1440;
freq[12] = 1830;
freq[13]=2290;
freq[14]=2870;
freq[15]=3670;
freq[16]=4590;
freq[17]=4000;
for (j=0; j<17; j++)
{
        for (i=0;  i<128; i++)
        c[i]=0;
        i=1;
        f1=59;
        while(f1/freq[j] < .552)
        {
          i++;
          f1 =+ 39;
        }
        while (f1/freq[j] < .9)
        {
          c[i] = f1/freq[j] * 2.87 -1.59;
          i++;
          f1 =+ 39;
        }
        while ((f1/freq[j] <= 1.11) && (i < 128))
        {
          c[i] = 1;
          i++;
          fl =+ 39;
        }
        while ((fl/freq[j] < 1.81) && (i<128))
        {
          c[i] =  -1.43*fl/freq[j] + 2.58;
          i++;
          fl =+ 39;
        }
        write(fdl,c,512);
}
for ( i = 0; i < 128; i++)
        c[i] = 0;
i = 5;
fl = 215;
while ( fl/freq[17] < .24)
{
        i =+ 1;
        fl =+ 39;
}
while ( fl/ freq[17]  < .75)
{
        c[i] = fl/freq[17] * 1.96 - .47;
        i++;
        fl =+ 39;
}
while (( fl/freq[17] < 1.33) && ( i < 128))
{
  c[i] = 1;
  i++;
  fl =+ 39;
}
write (fdl, c,512);
filename[0] = 0;
fd2 = wantread(filename,0,"skirt file for read?");
i= 0;
while ( read(fd2,c[0],512) != 0)
{
        i++;
        printf("filter no. is... %d#", i);
        for (j = 0; j < 128; j++)
        printf( "%f   ",c[j]);
}
}
```

```
/*   Program for generating a data file to be used for shaping
 1/3 octave filters from FFT output.
*/
main ()
{
 float
        c[128],b[768];
int
        fd1,fd2,
        i,j,k,m,t;
char
        filename[64];

 filename[0] = "skirt.dat";
fd1=wantread(filename,0,"?");
filename[0]=0;
fd2=wantwrite(filename,0, "skirt.flt");
m=0;
for (i=1; i<19; i++)
  {
        j=0;
        k=0;
        t=0;
        read(fd1,c,512);
        while(c[j] <= 0)
        {
          j++;
        }
        b[m] = j;
        m++;
        t=m;
        while ((c[j] >0) && (j < 128))
        {
          k++;
          m++;
          b[m] = c[j];
          j++;
        }
        b[t] = k;
        m++;
 }
for (i=1; i < 7; i++)
 {
        t= (i-1) * 128;
        write(fd2, &b[t],512);
        for (j=0; j < 128; j++)
          printf( "%f ", b[t+j]);
 }
}
```

APPENDIX E

PROGRAM LISTINGS FOR EXPERIMENTAL SYSTEM

```
C            MODIFIED LAST ON 12/4/79
C
C      MAIN MODULE OF SPEECH RESEARCH SOFTWARE TOOL
C            BY RICK PARFITT
C
      BYTE ANS,SP(17600),LAB(200)
      INTEGER THRES
      COMMON /S1/ SP,THRES,LAB
C
C      ERASE THE SCREEN, PUT UP MAIN MENU AND BRANCH TO
C APPROPRIATE MODULE
C
      DO 5 I=1,17600
      SP(I)=0
5      CONTINUE
10      CONTINUE
      CALL CLRSCN
      CALL F1
      CALL RESP(ANS)
      IF (ANS .EQ. 41) CALL GETTHR
      IF (ANS .EQ. 51) CALL TRAIN
      IF (ANS .EQ. 61) CALL RECOG
      GO TO 10
      END
C
C      CLEAR THE VIDEO SCREEN
C
      SUBROUTINE CLRSCN
      BYTE B
      B=-96
      J=-4097
      DO 10 I=1,1024
      CALL POKE(I+J,B)
10      CONTINUE
      RETURN
      END
C
C      SUBROUTINE TO PUT UP MASTER FRAME
C
      SUBROUTINE F1
      BYTE MESG1(20),MESG2(31),MESG3(30),MESG4(17)
      BYTE MESG5(22),MESG6(17),MESG7(19),MESG8(25)
      DATA MESG1/1H1,1H1,1H ,1HS,1HP,1HE,1HC,1HT,1HR,1HU,
     X  1HM,1H ,1HA,1HN,1HA,1HL,1HY,1HZ,1HE,1HR/
      DATA MESG2/1H2,1H1,1H ,1HA,1HV,1HE,1HR,1HA,1HG,1HE,
     X  1H ,1HE,1HN,1HE,1HR,1HG,1HY,1H ,1H1,1H/,1H1,1H0,
     X  1H ,1HS,1HE,1HC,1HO,1HN,1HD,1H ,1H /
      DATA MESG3/1H3,1H1,1H ,1HA,1HV,1HE,1HR,1HA,1HG,1HE,
     X  1H ,1HE,1HN,1HE,1HR,1HG,1HI,1HE,1HS,1H-,1H1,1H/,1H2,
     X  1H ,1HS,1HE,1HC,1HO,1HN,1HD/
      DATA MESG4/1H4,1H1,1H ,1HG,1HE,1HT,1H ,1HT,1HH,
     X  1HR,1HE,1HS,1HH,1HO,1HL,1HD/
      DATA MESG5/1HS,1HP,1HE,1HE,1HC,1HH,1H ,1HR,1HE,1HS,
     X 1HE,1HA,1HR,1HC,1HH,1H ,1HS,1HY,1HS,1HT,1HE,1HM/
      DATA MESG6/1H5,1H1,1H ,1HM,1HA,1HK,1HE,1H ,1HT,1HE,
     X  1HM,1HP,1HL,1HA,1HT,1HE,1HS/
      DATA MESG7/1H6,1H1,1H ,1HR,1HE,1HC,1HO,1HG,1HN,
     X  1HI,1HZ,1HE,1H ,1HS,1HP,1HE,1HE,1HC,1HH/
      DATA MESG8/1H1,1H2,1H ,1HL,1HE,1HN,1HG,1HT,1HH,1H ,
     X  1HO,1HF,1H ,1HW,1HO,1HR,1HD,1H ,1HM,1HI,1HL,1H ,
     X  1HS,1HE,1HC/
      CALL WRSCN(MESG1(1),3,2,20)
      CALL WRSCN(MESG2(1),5,2,31)
      CALL WRSCN(MESG3(1),7,2,30)
      CALL WRSCN(MESG4(1),9,2,16)
      CALL WRSCN(MESG5(1),1,20,22)
      CALL WRSCN(MESG6(1),11,2,17)
      CALL WRSCN(MESG7(1),13,2,19)
      CALL WRSCN(MESG8(1),3,34,25)
      RETURN
      END
C
C      SUBROUTINE TO WRITE MESSAGES ON THE VIDEO SCREEN.
C      MESG-MESAGE IN ASCII TO SENT TO SCREEN
C      LN  -LINE TO BEGIN WITH
C      ST  -STARTING CHARACTER ON SCREEN
C      N   -NUMBER OF CHARACTER TO SEND TO SCREEN
C
      SUBROUTINE WRSCN(MESG,LN,ST,N)
      BYTE MESG(1)
      INTEGER LN,ST,N
      INTEGER ADD
      ADD=-4097 +(LN-1)*64 + ST-1
      DO 10 I=1,N
      CALL OUTCH(ADD,MESG(I))
      ADD=ADD+1
10      CONTINUE
      RETURN
      END
C
C      INITIALIZE THE VIDEO DISPLAY - ERASE SCREEN AND
C  WRITE LOG-ON MESSAGE.
C
      SUBROUTINE INIT
      BYTE MESG1(25),MESG2(64)
      LOGICAL      B,D
      DATA MESG1 /1HS,1HP,1HE,1HC,1HT,1HR,1HU,1HM,
     X 1H ,1H ,1HA,1HN,1HA,1HL,1HY,1HZ,1HE,1HR,
     X 1H ,1H ,1H ,1H ,1H ,1H ,1H /
      DATA MESG2 /1H ,1H ,1H1,1H ,1H ,1H2,1H ,1H ,1H3,
     X  1H ,1H ,1H4,1H ,1H ,1H5,1H ,1H ,1H6,1H ,1H ,1H7,
     X  1H ,1H ,1H8,1H ,1H ,1H9,1H ,1H ,1H1,1H0,
     X  1H ,1H1,1H1,1H ,1H1,1H2,1H ,1H1,1H3,1H ,1H1,1H4,1H ,
     X  1H1,1H5,1H ,1H1,1H6,1H ,1H1,1H7,1H ,1H1,1H8,1H ,
     X  1H1,1H9,1H ,1H2,1H0,1H ,1H ,1H /
      D=0
      B=-96
      J=-4097
      DO 10 I=1,1024
      CALL POKE(I+J,B)
10      CONTINUE
      J=-4096 + 8*64 + 2
      DO 11 I=1,19
      CALL POKE(J,D)
      J=J+3
11      CONTINUE
      J=-4096 + 20
      DO 33 I=1,25
      CALL OUTCH(J+I,MESG1(I))
33      CONTINUE
      J=-4097+15*64
      DO 44 I=1,64
      CALL OUTCH(J+I,MESG2(I))
44      CONTINUE
      RETURN
      END
C
C      SET BIT SEVEN FOR VIDEO CHARACTER WRITE AND SEND
C THE ASCII CHARACTER SPECIFIED IN D TO THE ADDRESS
C J.
C
      SUBROUTINE OUTCH(J,D)
      INTEGER J
      LOGICAL D,TP
      TP=D .OR. -128
      CALL POKE(J,TP)
      RETURN
      END
C
C
C      RESP - RESPONSE TO SCREEN QUERRY
C
      SUBROUTINE RESP(I)
      BYTE I
      BYTE M1,M2
      I=0
      CALL PROMPT
1      CONTINUE
      CALL GETCHR(M1)
      M1=M1-48
      IF ((M1 .LT.1) .OR. (M1 .GT. 7)) GO TO 1
      CALL WRSCN(M1+48,16,50,1)
2      CONTINUE
      CALL GETCHR(M2)
      M2=M2-48
      IF ((M2 .LT. 1) .OR. (M2 .GT. 2)) GO TO 2
      CALL WRSCN(M2+48,16,51,1)
      I=M1*10 + M2
      RETURN
      END
C
C       SUBROUTINE TO PROMPT USER FOR INPUT
C
      SUBROUTINE PROMPT
      BYTE MESG1(20)
      DATA MESG1/1HS,1HE,1HL,1HE,1HC,1HT,1HI,1HO,1HN,1H?,1H ,1H ,1H ,
     X  1H ,1H ,1H ,1H ,1H ,1H ,1H /
      MESG1(50)=0
      CALL WRSCN(MESG1(1),16,39,20)
      RETURN
      END
C
C       SUBROUTINE TO CHECK STATUS OF THE KEYBOARD.
C
C             RETURN A 0 IF NO CHARACTER READY
C             REUTRN A 1 IF A CHARACTER IS READY
C
      SUBROUTINE STATUS(ANS)
      BYTE ANS
      ANS=INP(0)
      ANS=ANS .AND. 1
      ANS = .NOT. ANS
      ANS = ANS .AND. 1
      RETURN
      END
C
C
C       SUBROUTINE TO FETCH A SINGLE CHARACTER FROM KEY-
C    BOARD.
C
      SUBROUTINE GETCHR(CHR)
      INTEGER BOOT
      BYTE CHR,CTRLC,HRDCPY
      BOOT =0
      CTRLC=3
      HRDCPY=29
10      CONTINUE
      CHR=INP(0)
      CHR= CHR .AND. 1
      IF (CHR .NE. 0) GO TO 10
      CHR=INP(2)
      IF (CHR .EQ. CTRLC) GO TO BOOT
      IF (CHR .EQ. HRDCPY) CALL PRTSCN
      IF (CHR .EQ. HRDCPY) GO TO 10
      RETURN
      END
C
C       SUBROUTINE TO FETCH SPEECH UTTERANCE
C             SP(N)-CONTAINS RAW SPEECH
C             SPEECH IS IN GROUPS OF 20
C             SUM=SUM OF FIRST 100 MILSECONDS OF SPEECH
C             BANDS 1, 19 AND 4 ARE USED IN SUM
```

```
C             THRES-THRESHOLD FOR SPEECH
C
      SUBROUTINE GETTHR
      INTEGER ST,THRES
      BYTE DONE,T1
      BYTE MESG1(28)
      BYTE SP(17600)
      COMMON /S1/ SP,THRES
      DATA MESG1/1HN,1HU,1HM,1HE,1HR,1HI,1HC,1H ,1HV,
     X  1HA,1HL,1HU,1HE,1H ,1HF,1HO,1HR,1H ,1HT,1HH,1HR,
     X  1HE,1HS.1HH.1HO.1HL,1HD,1H?/
C
C INITIALIZATION
C
      CALL CLRSCN
      CALL WRSCN(MESG1(1),1,15,28)
1      CONTINUE
      CALL GETNUM(THRES.DONE)
      IF (DONE.EQ.1) RETURN
      CALL LISTEN(ST)
      CALL STATUS(DONE)
      IF (DONE.NE.0) RETURN
      GO TO 1
      END
C
C      SUBROUTINE TO GET FIRST PART OF SPEECH. USES THRESHOLD
C 'THRES' STORED IN COMMON BLOCK TO DETERMINE ACTUAL SPEECH
C
      SUBROUTINE LISTEN(ST)
      INTEGER ST,SUM,BASE,THRES,PT,LSPT,LASE
      BYTE T1,SP(17600)
      COMMON /S1/SP,THRES
      DO 5 I=1,200
      SP(I)=0
5      CONTINUE
      PT = -19
10      CONTINUE
      PT = PT + 20
      IF (PT .EQ. 221) PT = 1
      CALL FETCH(SP(PT))
      BASE = SP(PT)
      IF (BASE .GT. THRES) SUM = SUM + 1
      IF (BASE .LT. THRES) SUM = 0
      IF (SUM .NE. 5) GO TO 10
      ST = PT - 100
      IF (ST .LT. 0) ST = ST + 202
      RETURN
      END
C
C      SUBROUTINE TO GET A NUMERIC VALUE FROM KEYBD.
C
      SUBROUTINE GETNUM(VAL,DN)
      INTEGER VAL,I,TP2
      BYTE TP1,CR,DN,ESC
      CR=13
      ESC=27
      DN=0
9      CONTINUE
      I=51
      CALL PROMPT
      CALL GETCHR(TP1)
      IF (TP1 .EQ. ESC) DN=1
      IF (TP1.EQ.ESC) RETURN
      VAL=0
1      CONTINUE
      IF (TP1.EQ. CR) RETURN
      IF (TP1.EQ.ESC) DN=1
      IF (TP1.EQ.ESC) RETURN
      TP2=TP1-48
      TP1=TP1-48
      IF ((TP1 .LT. 0) .OR. (TP1 .GT. 9)) GO TO 9
      CALL WRSCN(TP1+48,16,I,1)
      I=I+1
      IF (I .EQ. 61) GO TO 9
      VAL=VAL*10 + TP2
      CALL GETCHR(TP1)
      GO TO 1
      END
C
C      SUBROUTINE TO PRINT THE SCREEN OUT TO IBM SELECTRIC
C
      SUBROUTINE PRTSCN
      I = 0
      RETURN
      END
C
C      SUBROUTINE FOR BUILDING TEMPLATES
C
      SUBROUTINE TRAIN
      BYTE SP(17600).LAB(200).FLG
      INTEGER THRES,ST,J,M,L,I,TP,TP2,AV1,AV2
      COMMON /S1/ SP,THRES,LAB
      CALL CLRSCN
      DO 100 I=1,10
      TP=I*10 - 9
      CALL GETNAM (LAB(TP).FLG)
      IF (FLG .EQ. 1) RETURN
      CALL LISTEN(ST)
C
C      LISTEN FOR 1 SECONDS
C
      DO 200 J=201,1600,20
      CALL FETCH(SP(J))
200      CONTINUE
C
C      CALL THE ENDPT ROUTINE WHICH WILL NORMALIZE THE
```

```
C UTTERANCE TO A STANDARD LENGHT AND DO A CRUDE END POINT
C ANALYSIS
C
      CALL ENDPT(ST)
C
C      START SAVING TEMPLATES AT LOCATION 3001
C
      M = I*1600 + 1
      TP = SP(1)
      TP = TP*20 + M -1
      K = 1
      DO 300 J=M,TP
      SP(J) = SP(K)
      K=K+1
300      CONTINUE
100      CONTINUE
      I = SP(M)
      WRITE(4,404) I
404      FORMAT('  SP(M)', I8)
      RETURN
      END
C
C      SUBROUTINE TO GET LABEL FOR SPEECH TEMPLATE.
C TEMPLATE NAMES ARE LIMITED TO TEN CHARACTERS
C
      SUBROUTINE GETNAM(LAB,FLG)
      BYTE LAB(10),CR,TP1,DEL,ESC,FLG
      FLG=0
      DEL=127
      CR=13
      ESC=27
1      CONTINUE
      CALL PROMPT
      DC 2 I=1,10
      LAB(I)=' '
2      CONTINUE
      CALL PROMPT
      DO 10 I=1,10
      CALL GETCHR(TP1)
      IF (TP1 .EQ. CR) RETURN
      IF (TP1 .EQ. ESC) FLG = 1
      IF (TP1 .EQ. ESC) RETURN
      IF (TP1 .EQ. DEL) GO TO 1
      LAB(I) = TP1
      CALL WRSCN(TP1,16,50+I,1)
10      CONTINUE
      RETURN
      END
C
C      SUBROUTINE TO RECOGNIZE SPEECH
C USES NOT END POINT DETECTION AND DOES NOT TIME ALIGNMENT.
C
      SUBROUTINE RECOG
      INTEGER TP1,ST,THRES,TP2,TP3
      BYTE SP(17600),LAB(200),FLG,MESG1(10)
      INTEGER SCORE(15),TPF
      COMMON /S1/ SP,THRES,LAB
      DATA MESG1/1HF,1HO,1HU,1HN,1HD,1H ,1HW,1HO,1HR.1HD/
      CALL CLRSCN
      CALL PROMPT
      CALL LISTEN(ST)
      DO 100 J=201,1600,20
      CALL FETCH(SP(J))
100      CONTINUE
      CALL WRSCN(MESG1(1),5,20,10)
      CALL ENDPT(ST)
      DO 300      I =1,10
      TP1 = 0
      L = SP(1)
      K = I*1600 + 1
      IF (SP(K) .LT. L) L = SP(K)
      L = L*20
      DO 350 J =1,L
      TP2 = SP(J)
      TP3 = SP(K)
      TP1 = TP1 + IABS(TP2-TP3)
      K = K + 1
350      CONTINUE
      L = L/2
      IF(TP1 .LT. 0) TP1 = 32000
      SCORE(I) = TP1/L
300      CONTINUE
      TP1 = SCORE(1)
      ST = 1
      DO 310 I = 2,10
      IF (SCORE(I) .GT. TP1) GO TO 310
      TP1 = SCORE(I)
      ST = I
310      CONTINUE
      TP2 = 32000
      DO 309 I =1,10
      IF (SCORE(I) .EQ. TP1) GO TO 309
      IF (SCORE(I) .GT. TP2) GO TO 309
      TP2 = SCORE(I)
      TP3 = I
309      CONTINUE
      L = 32000
      DO 308 I = 1,10
      IF (SCORE(I) .EQ. TP1 ) GO TO 308
      IF (SCORE(I) .EQ. TP2) GO TO 308
      IF (SCORE(I) .GT. L) GO TO 308
      L = SCORE(I)
      M = I
308      CONTINUE
      WRITE(4,311) TP1,ST,TP2,TP3
311      FORMAT('  SCORE',I8,' TEMPLATE',I8,' SCORE2 ',I8,I8)
```

```
      I1 = ST
      I2 = TP3
      I3 = M
      ST = I1 * 1600 + 1
      CALL DYNAM(ST,TP1)
      ST = I2 * 1600 + 1
      CALL DYNAM(ST,TP2)
      ST = I3 * 1600 + 1
      CALL DYNAM(ST,TP3)
      ST = I1
      TPF = TP1
      IF (TP2 .GT. TP1) GO TO 410
      TPF = TP2
      ST = I2
      IF (TP3 .GT. TP2) GO TO 420
      TPF = TP3
      ST = I3
      GO TO 420
410      CONTINUE
      IF (TP3 .GT. TP1) GO TO 420
      ST = I3
      TPF = TP3
420      CONTINUE
      ST = ST-1
      ST=ST*10 + 1
      CALL WRSCN(LAB(ST),16,51,10)
      WRITE(4,43) TP1,TP2,TP3
43      FORMAT(' THREE BEST SCORES.. ',I8,I8,I8)
      WRITE(4,44) TPF
44      FORMAT('  BEST SCORE..',I8)
      CALL GETCHR(FLG)
      RETURN
      END
      SUBROUTINE ENDPT(ST)
      BYTE SP(17600), LAB(200),TEMP(200)
      INTEGER ST,I,J,BEGP,LN,LO,K,FLG,M,LTHRES,LUP,THRES
      COMMON /S1/ SP,THRES,LAB
C MODIFIED END POINT ANALYSIS. JUST DO BEGINNIN POINT, AND
C THEN TAKE FIRST .4 SECONDS OF SPEECH.
C
      K = 1
      I = ST
      DO 4 M=1,10
      LUP = I + 19
      DO 2 J=I,LUP
      TEMP(K) = SP(J)
      K = K+1
2       CONTINUE
      I = I + 20
      IF (I .EQ. 201) I = 1
4       CONTINUE
      DO 6 I= 1,200
      SP(I) = TEMP(I)
```

```
6        CONTINUE
        LN = 0
        K = 181
        BEGP = 10
100        CONTINUE
        K = K+20
        IF (K .EQ. 1601) GO TO 300
        BEGP = BEGP + 1
        LTHRES = SP(K)
        IF (LTHRES .LT. THRES) LN = LN + 1
        IF (LTHRES .GT. THRES) LN = 0
        IF(LN .NE. 15) GO TO 100
        IF (BEGP .LT. 22) GO TO 100
300        CONTINUE
        IF (LN .GT. 5) BEGP = BEGP -LN + 5
        SP(1) = BEGP
        WRITE(4,33) BEGP
33        FORMAT('  NO. OF WINDOWS IS..', I8)
        RETURN
        END
```

```
C
C      LAST MODIFIED    12/6/1979
C
C      DYNAMIC PROGRAMMING ALGORITHM VERS. 1.2
C            DECEMBER 5, 1979
C
C
C      G(19,100) - DYNAMIC PROGRAMMING ARRAY
C      UNK - INDEX INTO UNKOWN UTTERANCE ARRAY
C      TPK - INDEX INTO TEMPLATE ARRAY
C      LU - UPPER LIMIT OF DO LOOP
C      LD - LOWER LIMIT OF DO-LOOP
C      INF - A VERY LARGE NUMBER
C      TPII - INDEX BASE INTO TEMPLATE
C      1 - BEGINNING OF UNKNOWN ARRAY
C      MDL - LENGTH OF SHORTEST SPEECH UTTERANCE
C
C
      SUBROUTINE DYNAM(TPII,SCR)
      INTEGER INF,UNK,TPK,THRES,SCR,TPII,LD,LU
      INTEGER I,J,N
      INTEGER MDL
      INTEGER IDD
      INTEGER G(19,100)
      BYTE SP(17600),LAB(200)
      COMMON /S1/SP,THRES,LAB/DARG/UNK,TPK
C
C
      MDL = SP(1)
      I = SP(TPII)
      IF ( I.LT. MDL) MDL = I
      WRITE(4,3) MDL
3      FORMAT('  MDL IS..',I8)
      IF (MDL .GT. 21) GO TO 4
      SCR = 32000
      RETURN
4      CONTINUE
      MDL = MDL -10
C
C      INITIALIZE ROW ONE AND COLUMN ONE
C
      TPK = TPII
      UNK = 1
      G(1,1) = -IDEL(IDD)
      DO 20 I=2,10
      UNK = UNK + 20
      G(I,1) = G(I-1,1) - IDEL(IDD)
20      CONTINUE
      UNK = 1
      DO 22 I=2,10
      TPK = TPK + 20
      G(1,I) = G(1,I-1) - IDEL(IDD)
22      CONTINUE
```

```
C
C      COMPUTE COLUMN ONE WHICH HAS LENGTH 11.
C
      TPK = TPII + 20
      UNK = UNK + 20
      G(2,2) = MINN(G(1,1),INF,INF,IDEL(IDD))
      N = 10
      DO 30 I=3,N
      UNK = UNK + 20
      G(I,2) = MINN(G(I-1,1),INF,G(I-1,2),IDEL(IDD))
30      CONTINUE
      UNK = UNK + 20
      G(11,2) = MINN(G(10,1),INF,G(10,2),IDEL(IDD))
C
C      COMPUTE THE NEXT EIGHT COLUMNS WHICH INCREASE BY ONE FOR
C EACH NEW COLUMN.
C
      N = 11
      DO 100 J = 3,10
      TPK = TPII + 20*J - 20
      UNK = 21
      DO 110 I = 2,N
      G(I,J) = MINN(G(I-1,J-1),G(I,J-1),G(I-1,J),IDEL(IDD))
      UNK = UNK + 20
110      CONTINUE
      N = N+1
      G(N,J) = MINN(G(I-1,J-1),INF,G(I-1,J),IDEL(IDD))
100      CONTINUE
C
C      COMPUTE THE NEXT X COLUMNS
C
      UNK = 21
      DO 200 J = 11, MDL
      TPK = TPII + J*20 - 20
      G(1,J) = MINN(G(1,J-1),G(2,J-1),INF,IDEL(IDD))
      UNK = UNK + 20
      DO 210      I = 2,18
      G(I,J) = MINN(G(I,J-1),G(I+1,J-1),G(I-1,J),IDEL(IDD))
      UNK = UNK + 20
210      CONTINUE
      G(19,J) = MINN(G(I,J-1),INF,G(I-1,J),IDEL(IDD))
      UNK = (J-10)*20 + 1
200      CONTINUE
C
C      COMPUTE THE LAST TEN COLUMNS.
C
      N = 18
      J = MDL + 1
      UNK = (J-10) * 20 + 1
      LD = MDL + 1
      LU = MDL + 10
      DO 300 J = LD,LU
      TPK = TPII + J*20 - 20
      G(1,J) = MINN(G(1,J-1),G(2,J-1),INF,IDEL(IDD))
      UNK = UNK + 20
      DO 310 I = 2, N
      G(I,J) = MINN(G(I,J-1),G(I+1,J-1),G(I-1,J),IDEL(IDD))
      UNK = UNK + 20
310      CONTINUE
      IF (N.EQ.18) GOTO 311
      G(N+1,J)=MINN(G(N+1,J-1),G(N+2,J-1),G(N,J),IDEL(IDD))
311      CONTINUE
      IF(N.EQ.18) G(19,J)=MINN(G(N+1,J-1),INF,G(N,J),IDEL(IDD))
      UNK = (J-10) *20 + 1
      N = N-1
300      CONTINUE
C
C      NORMALIZE ANSWER BY LENGTH OF SHORTEST UTTERANCE - MDL
C
      MDL = MDL + 10
      SCR = G(10,MDL)
      SCR = SCR/MDL
      IF (SCR .LT. 0) SCR = -SCR
      RETURN
      END
C
C        RETURN A NEGATIVE ANSWER IF VERTICAL OR HORIZONTAL
C PATH TAKEN. FORCE A DIAGNOL PATH OF LAST PATH HORIZONTAL
C OR VERTICAL. I4 IS THE CURRENT DISTANCE
C BETWEEN UNKNOWN AND TEMPLATE.
C
      INTEGER FUNCTION MINN(I1,I2,I3,I4)
      INTEGER I1,I2,I3,I4,II1,II2,II3,II4
      INTEGER INF
      INF = 32000
      II1 = I1
      II2 = I2
      II3 = I3
      II4 = I4
      IF(I1 .LT. 0) II1 = - II1
      IF (I2 .LT. 0) II2 = INF
      IF (I3 .LT. 0) II3 = INF
      MINN = II1 + II4
      IF (II2 .LT. II1) MINN = -(II2 + II4)
      IF (II3 .GT. II2) RETURN
      IF (II3 .LT. II1) MINN = -(II3 + II4)
      RETURN
      END
C
C      FUNCTION TO CALCULATE DISTANCE BETWEEN TWO TIME
C SLICES.
C
      INTEGER FUNCTION IDEL(IDD)
      INTEGER THRES,UNK,TPK,TP1,TP2,TP3,TP4,TP5
      INTEGER IDD
      BYTE SP(17600),LAB(200)
      COMMON /S1/SP,THRES,LAB/DARG/UNK,TPK
      IDEL = 0
      TP4 = SP(UNK)
      TP5 = SP(TPK)
      IDEL = IABS(TP4 - TP5)
      TP4 = SP(UNK + 2)
      TP5 = SP(TPK + 2)
      TP4 = TP4 + SP(UNK+3) + SP(UNK + 4)
      TP5 = TP5 + SP(TPK + 3) + SP(TPK + 4)
      IDEL = IDEL + IABS(TP4-TP5)
      TP1 = UNK + 6
      TP2 = UNK + 19
      TP3 = TPK + 6
      DO 10 I=TP1,TP2
      TP4 = SP(I)
      TP5 = SP(TP3)
      IDEL = IDEL + IABS(TP4-TP5)
      TP3 = TP3 + 1
10      CONTINUE
      RETURN
      END
      ENTRY      FETCH
      ENTRY IBMIO
;      RELOCATABLE Z80 CODE FOR FETCHING PARAMETERS
; FROM THE 19 CHANNEL FILTER BOARD.
;
;      REGISTER ASSIGNMENTS:
;            B - BAND NUMBER
;            C - LAST VALUE READ FROM BOARD
;            D - TEMPOARY STORAGE LOCATION.
;
FETCH:
      LD      (SADD),HL
      LD      A,(DE)
      CP      1
      JP      NZ,INIT
      LD      A,0
      LD      (OK2),A
INIT:
      LD      B,0
      LD      C,0
START:
      IN      A,0DFH
      AND      20H
      JR      Z,START
      IN      A,0DFH
      AND      01FH
      CP      B
      JR      NZ,START
      IN      A,0DEH
      RLA
      JR      NC,NEG
      RRA
      CPL
OK:
      LD      D,A
OK2:
      SUB      C
      LD      C,D
      LD      (HL),A             ;RETURN ENERGY IN HL LOC.
      INC      HL
      INC      B
      LD      A,B
      CP      14H
      JR      NZ,START
;
;  MUST ADD BAND ZERO TO BAND ONE, SUBTRACTION WAS MISTAKE
;
      LD      HL,(SADD)
      LD      A,(HL)
      INC      HL
      ADD      A,(HL)
      LD      (HL),A
      RET
```

```
NEG:
      LD        A,0
      JP        OK
;
;      SAVE TWO BYTES FOR STORING BEGINNING OF DATA ARRAY
;
SADD:
      DS 2
IBMIO:
      LD        A,(HL)                 ;PUT CHARACTER IN A
      CALL        TRAN
      CALL        OUTIT
      RET
TRAN:
      LD        B,A
      LD        HL,TABLE
ALLTAB:
      LD        A,(HL)
      CP        B
      JP        Z,FOUND
      CP        05H
      JP        Z,UCTB
      INC        HL
      INC        HL
      JP        ALLTAB
FOUND:
      INC        HL
      LD        B,(HL)
      LD        A,10H
      RET
UCTB:
      INC        HL
      INC        HL
      LD        A,(HL)
      CP        B
      JP        Z,HAVE
      CP        05H
      JP        Z,NOHAVE
      JP        UCTB
HAVE:
      INC        HL
      LD        B,(HL)
      LD        A,08H
      RET
NOHAVE:
      LD        A,B
      CP        0DH
      JP        NZ,SPAC
      LD        B,0
      LD        A,02H
      RET
SPAC:
      CP        20H
```

```
        JP      NZ,NOMOR
        LD      B,0
        LD      A,04H
        RET
NOMOR:
        CP      0AH
        JP      NZ,STCH
        LD      B,0FFH
        RET
STCH:
        LD      B,3BH               ;"0" ATRANGE CHARACTER.
        LD      A,08H
        RET
OUTIT:
        LD      C,A
        LD      A,B
        CP      0FFH
        RET      Z
        CALL      EOLCH
        LD      A,C
GO:
        AND      10H
        JP      Z,UCAS
        CALL      MLC
        JP      CSET
UCAS:
        LD      A,C
        AND      8
        JP      Z,CSET
        CALL      MUC
CSET:
        LD      A,B
        AND      A
        JP      Z,NEXT
        CALL      READY
        LD      A,B
        CPL
        OUT      2,A
        CALL      NOTB
NEXT:
        CPL
        OUT      0FFH,A
        CPL
        CALL      SPCH
        CALL      CRCH
        RET
CRCH:
        LD      A,C
        AND      2
        RET      Z
        CALL      READY
        LD      A,2
        CPL
```

```
      OUT      3,A
      CALL      NOTB
      RET
NOTB:
      IN      A,3
      AND      1
      JP      NZ,NOTB
      LD      A,0FFH
      OUT      3,A
      OUT      2,A
      RET
READY:
      IN      A,3
      AND      1
      JP      Z,READY
CAWT:
      IN      A,3
      AND      4
      JP      Z,CAWT
      RET
MUC:
      IN      A,3
      AND      2
      RET      Z
      CALL      READY
      LD      A,8
      CPL
      OUT      3,A
      CALL      NOTB
      RET
MLC:
      IN      A,3
      AND      2
      RET      NZ
      CALL      READY
      LD      A,80H
      CPL
      OUT      3,A
      CALL      NOTB
      RET
SPCH:
      LD      A,C
      AND      4
      RET      Z
      CALL      READY
      LD      A,4
      CPL
      OUT      3,A
      CALL      NOTB
      RET
EOLCH:
      IN      A,3
      AND      8
```

```
          RET        NZ
          CALL        READY
          LD        A,2
          CPL
          OUT        3,A
          CALL        NOTB
          RET
TABLE:
          DEFW        4A61H,0462H,4C63H,5D64H
          DEFW        1C65H,6866H,7967H,1568H
          DEFW        0B69H,386AH,0D6BH,546CH
          DEFW        5431H,7A6DH,2C6EH,526FH
          DEFW        1970H,0871H,5B72H,1373H
          DEFW        3D74H,6D75H,6B76H,0277H
          DEFW        7C78H,1079H,3E7AH,7F5DH
          DEFW        2F32H,6E33H,5734H,1F35H
          DEFW        0E36H,5E37H,4F38H,0739H
          DEFW        1630H,012DH,293DH,3B21H,583BH
          DEFW        1A27H,492CH,2A2EH,512FH,0005H
          DEFW        4A41H,0442H,4C43H,5D44H,1C45H,6846H
          DEFW        7947H,1548H,0B49H,384AH,0D4BH,544CH
          DEFW        7A4DH,2C4EH,524FH,1950H,0851H
          DEFW        5B52H,1353H,3D54H,6D55H,6B56H
          DEFW        0257H,7C58H,1059H,3E5AH,7F5BH
          DEFW        2F40H,6E23H,5724H,1F25H
          DEFW        5E26H,4F2AH,0728H,1629H
          DEFW        015FH,292BH,583AH,1A22H
          DEFW        492CH,2A2EH,513FH,0005H
          END
```

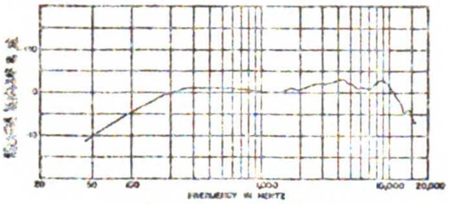
APPENDIX F

SHURE SM10 MICROPHONE

SPECIFICATIONS

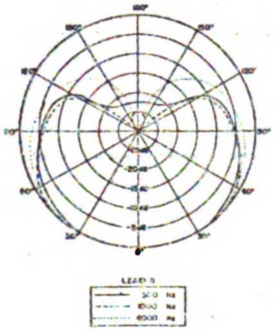
Type
Dynamic, Close-Talking

Frequency Response (at 8 mm [5/16 in.])
50 to 15,000 Hz (see Figure 2)



TYPICAL FREQUENCY RESPONSE
FIGURE 2

Polar Pattern
Cardioid (unidirectional) response—uniform with frequency, symmetrical about axis (see Figure 3)



POLAR PATTERN
FIGURE 3

Impedance
Microphone rating impedance is 150 ohms (200 ohms actual) for connection to microphone inputs rated at 19 to 300 ohms

Output Level (close-talked at 1,000 Hz)
Open Circuit Voltage .... −47.0 dB (4.5 mV)
(0 dB = 1 volt per 100 microbars)
Power Level .... −66.0 dB
(0 dB = 1 milliwatt per 10 microbars)

Hum Sensitivity (typical)
36.4 dB equivalent SPL in a 1 millioersted field

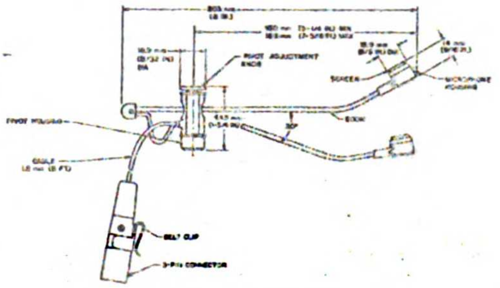
Phasing
Positive pressure on diaphragm produces positive voltage on pin 2 of microphone connector

Connector
Professional three-pin male audio connector designed to mate with Cannon XL series, Switchcraft A3 (Q.G.) series or equivalent connectors.

Cable
Non-detachable, 1.5m (5 ft), two-conductor, shielded, plastic-jacketed

Case
Black thermoplastic microphone and pivot housing, anodized aluminum end caps, stainless steel grille and boom

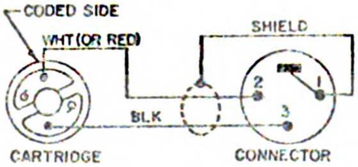
Dimensions
See Figure 4



OVERALL DIMENSIONS
FIGURE 4

Net Weight
78 grams (2.7 ounces) less cable and connector

Packaged Weight
745 grams (1 lb, 10 oz)



INTERNAL CONNECTIONS
FIGURE 5

STEREO HEADPHONE USE

The SM10 microphone may be used as a boom microphone with stereophonic headphones having provisions for mounting boom microphones. These headphones are identified by a removable knurled knob or screw on the left headphone cup.

To mount the SM10 on a stereo headphone, remove the retaining clip from the SM10 headband arm by removing two retaining clip screws (No. 3-48 x 3/16") and hex nuts. Mount the retaining clip on the headphone adapter plate supplied using the retaining clip screws and hex nuts. Unscrew the removable knob or screw from the stereo headphone cup. Position the headphone adapter plate with the retaining clip downward, and replace the knob or screw on the headphone cup (see Figure F). Snap the SM10 pivot housing into the retaining clip, pivot adjustment knob upward, and place the headphone-microphone assembly on the user's head. Adjust the microphone position as described in step 4 of the *Assembly* procedure.