REPORT R-524 NOVEMBER, 1971 UILU-ENG 71-2227

COORDINATED SCIENCE LABORATORY

SPEECH DISPLAY SIMULATION SYSTEM FOR A COMPARATIVE STUDY OF SOME VISUAL SPEECH DISPLAYS

BERNARD J. NORDMANN, JR.

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

UILU-ENG 71-2227

SPEECH DISPLAY SIMULATION SYSTEM FOR A COMPARATIVE STUDY OF SOME VISUAL SPEECH DISPLAYS

by

Bernard J. Nordmann, Jr.

This work was supported in part by the Joint Services Electronics Program (U. S. Army, U. S. Navy and U. S. Air Force) under Contract DAAB-07-67-C-0199, Coordinated Science Laboratory, and in part by Contract AT(11-1)-2118 (Atomic Energy Commission), Digital Computer Laboratory.

ACKNOWLEDGMENT

The author would like to express his sincere gratitude to Professor Sylvian R. Ray and Professor H. Gene Slottow for their advice and encouragement during the time in which the work described in this report was conducted.

Thanks are also gratefully extended to Mrs. Roberta Andre' and Mrs. Judy Arter for typing and correcting the manuscript and to Mr. Stan Zundo for preparing the drawings.

TABLE OF CONTENTS

LIST OF FIGURES

Section 1

INTRODUCTION

The purpose of this report is to give a detailed description of a speech display simulation system which has been programmed for the CDC 1604 computer system located at the Coordinated Science Laboratory at the University of Illinois. This simulator was written as part of an investigation into the relative effectiveness of various types of speech displays (Nordmann [1971]). As such, its primary goal was to be as flexible as possible as far as the generation of various types of speech displays is concerned.

In describing the details of the Speech Display Simulator, the philosophy has been to describe why each program was written the way it was as well as what it is actually trying to accomplish. In addition, any test programs which may have been written for a particular program are also explained.

The Speech Display Simulation can be divided into five main areas: the common data base, the command processor, the data handling routines, the speech display routines, and the various subprocessing routines.

The common data base consists of the input speech data buffer, BUFF, the output display data buffer, FINT, the CRT display command buffers, ISCOPE and ISCOP1, and all of the constants and variables used to control these buffers. These buffers and variables are all kept in COMMON storage. The problem of keeping the COMMON declaration in each subroutine identical is handled by means of the CSL FORTRAN title feature. This extension of the FORTRAN language allows the programmer to specify FORTRAN statements which will then appear in every program in which the statement TITLE* appears. Any type of valid FORTRAN statement can be put in the title and thus the whole common data base need only be written down once.

The command processor is used to accept fixed format commands from the operator at the typewriter or from the paper tape reader. After determining the command it performs the indicated operations by setting up parameters and calling the various subroutines indicated. The operator has the ability, via the command processor, to change the values of the system constants and variables and to call the various display routines. In addition, he can dump out the contents of the various arrays and variables.

The data handling routines are used to obtain external data from the A to D convertor, edit this data, copy it on to data tapes and then manipulate this data for use by other routines. The data gathering and editing is done off-line. This eliminates the possibility of a real time simulation, but the slowness of the CDC 1604 compared to the amount of processing to be done had already effectively prohibited this.

The speech display routines consist of the programs used to simulate the various speech displays. These programs manipulate the common data base using the various subprocessing routines to produce the displays desired.

The subprocessing routines consist of the programs which are used to perform various operations on and transformations of data. Each routine performs a single type of operation and might be used in the construction of several different displays.

As the Speech Display Simulation System developed, certain key principles were developed as follows:

1) The common data base, command processor and speech display routines should be basically machine independent. This means that they should be written in standard FORTRAN as much as possible and any use of

CSL FORTRAN extensions should be fully documented by means of comment statement in the code itself.

2) The subprocessing routines may be written in machine language or in a combination of FORTRAN and machine language as is allowed in the CSL FORTRAN system. However, this should only be done if a significant speedup in time or savings in space results or if it is necessary to perform some special function, such as communicating with the CRT display unit. In either case all occurrences of machine code should be explained both in the overall sense and at the detailed instruction level by comments within the program.

3) The subprocessing routines should not use the COMMON area buffers or transmit parameters through COMMON. Any external data needed should in general be obtained by means of parameters. This makes the routines easier to understand and reduces the number of mysterious side effects. The few exceptions allowed to this rule involve short subroutines which are used very often and which have several parameters. In such cases the overhead involved in handling explicit parameters becomes excessive.

4) Test programs used to check out the various subprocessing routines are not normally to be loaded with the rest of the system. They are kept on the library tape, however, so that when needed, they may be easily loaded by making a call request to the CSL Operating System. These programs should be well commented with exact instructions on their use since it is easy to forget their operation within a matter of weeks if they are not used regularly.

In actual use the speech display simulation system operates under typewriter control. The operator can manipulate the data tape to locate the speech to be processed. Then he can select various processing and display subroutines to operate on this data. In the case of a long production operation, such as the generation of a series of photographs of the displays of various speech words, a paper tape can be produced containing the various commands which need to be given. Then by placing the system in the paper tape input mode, the operator can avoid the need for a long period of continual interaction with the system. This also allows the operations to be reproducible at later times.

The remaining sections of this report are devoted to describing the system itself in terms of its data structure and individual subroutines. The subroutines have been broken down into groups of related programs so as to facilitate their explanation. Following each description is a fully commented source code listing giving, in addition to a short description of the program, a variable list complete with definitions.

Section 2

THE COMMON DATA BASE

The Common Data Base is the collection of arrays and variables used to contain the data being processed by the Speech Display System as well as the status of the system itself. Its structure is determined by the COMMON statements found in the title block, which is included in every CSL FORTRAN program containing the statement:

TITLE *

at the beginning of its source listing. This title feature causes the title block to be compiled along with the rest of the source code for these programs.

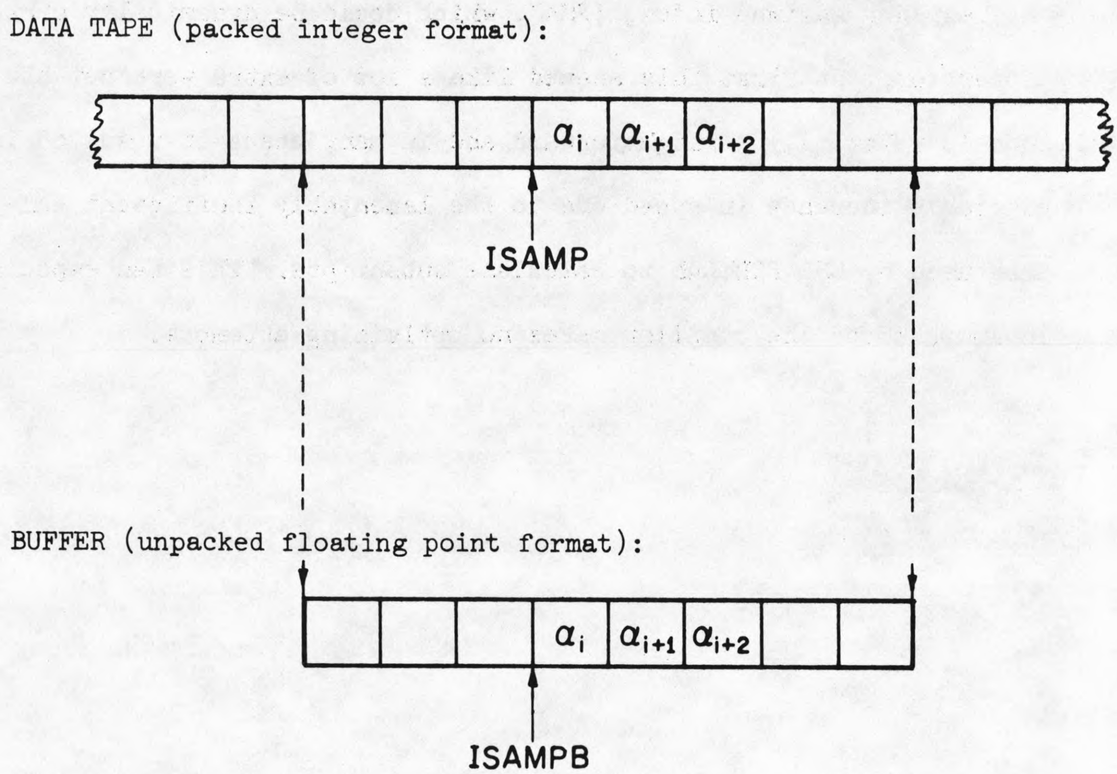
In order to provide some semblance of order in the COMMON area, the complete data area has been established by declaring the COMMON array, A. All other data arrays are then established using EQUIVALENCE statements to various subparts of this main array. This allows the relative position of each array with respect to the others to be explicitly stated in the EQUIVALENCE statement. It also allows other COMMON arrays to be declared for particular subroutines only, by EQUIVALENCE'ing them to A within the particular subroutines needing them. This latter technique should only be used for scratch areas since otherwise the side effects may become very complicated.

The common data base has several key features. Since digital tape was used for storing the input speech data, it was unnecessary to provide a full-sized buffer to contain a complete speech utterance. Instead, the floating point buffer, BUFF, is used to contain only that portion of the data which is of current interest.

As can be seen in figure 2.1, there are two corresponding pointers for the data tape and the buffer, BUFF. ISAMP is the main data pointer and selects the initial sample of a set of data points from the complete set of data (consisting of many speech utterances) on the data tape. Its value may range up to around 900,000 since this is the approximate number of packed sample points which can be written on a single tape. ISAMPB corresponds to ISAMP in that it points to the same data as ISAMP but it refers to the data as it happens to be currently loaded in BUFF. Thus ISAMPB only varies from 0 to the maximum length of BUFF (currently 3000 words).

The display generating routines are free to move ISAMP up and down the data tape whenever they wish. Before they utilize this new data position, however, they must call the subroutine ADJUS2. This subroutine checks the status of BUFF. If the data corresponding to the new position of ISAMP is not currently in BUFF, it moves the tape forward or backward until it can load BUFF with the proper data and converts the data to floating point. Once BUFF is made to contain the desired data, ADJUS2 sets ISAMPB so that it can be used as an index for BUFF to obtain the desired data. It is this pointer that the speech processing programs use to obtain the speech data.

The second feature involves the FINT array. This array is basically a two-dimensional array containing the intensity values with its dimensions corresponding to frequency vs. time. However, it was felt that it would be much more convenient to be able to vary the relative maximum sizes of these two dimensions even while the total length of the array remains fixed. This is especially nice for short speech samples in which it is desired to have a spectrographic analysis with a very small increment between frequencies, since in this case the maximum index for



Example: ISAMP = 54627

ISAMPB = 1627

Figure 2.1: Relationship Between ISAMP and ISAMPB

the frequency dimension must be increased. Unfortunately FORTRAN has no provision for dynamically assigning array dimensions. Therefore it was decided to require each program using FINT to calculate its own subscripts using a frequency maximum index, IFMAX, which could be dynamically chosen by the operator. At first this seemed like a lot of extra work but the technique is relatively straightforward and in many cases it resulted in a considerable increase in speed due to the lamentably inefficient calculations used by CSL FORTRAN to calculate subscripts. This was especially true in loops since the compiler makes no optimizing attempts.

```
OSL FORTRAN OF SEPT 1968, DATE  9/9/71
      TITLE COM
C
C          THIS BLOCK OF CODE IS A SPECIAL FEATURE OF OSL FORTRAN.  IT
C        IS A TITLE BLOCK WHICH IS AUTOMATICALLY COMPILED INTO EVERY OTHER
C        SUBSEQUENT PROGRAM AND SUBROUTINE IN WHICH THE STATEMENT,
C                                TITLE*
C        APPEARS.  IT ALLOWS THE PROGRAMMER TO SPECIFY THE STRUCTURE OF A
C        DATA BASE COMMON TO MANY PROGRAMS BY ONLY TYPING IT OUT ONCE.  THIS
C        ELIMINATES MANY ERRORS DUE TO MISMATCHED COMMON STATEMENTS IN SUB-
C        ROUTINES.
C
         COMMON A(20003)
C
C          DECLARE ALL SYSTEM VARIABLES WHICH CAN BE CHANGED BY
C        SIMPLE INTEGER ASSIGNMENT STATEMENTS ON COMMAND OF THE
C        OPERATOR.
C
         COMMON IDEL,  IDELT, IDELX, IDELY, IENDB, IFMAX, IFREQ,
     1          ITIME, ITLIM, ITMAX, IXMAR, LGNSM, LOCOP, NSMT2
C
C          DECLARE ALL SYSTEM VARIABLES WHICH CAN BE CHANGED BY
C        SIMPLE FLOATING POINT ASSIGNMENT STATEMENTS ON COMMAND OF
C        THE OPERATOR.
C
         COMMON EMPFC, FINTM, OVFAC
C
C          DECLARE ALL REMAINING SYSTEM VARIABLES.
C
         COMMON ICBLK, IFQPS, ISAMP, ISAMPB, IUNIT, IWID, NSAMT
C
C          DECLARE ALL SYSTEM CONSTANTS.  THESE VALUES WILL NOT
C        CHANGE FOR ANY PARTICULAR VERSION OF THE SYSTEM.
C
         COMMON DELF,  DTIME, ISAMF, LBLK,  LBLK4, LGNBF, NBLK,
     1          SAMF
C
C          DECLARE ALL COMMON VARIABLES USED FOR COMMUNICATION
C        BETWEEN SUBROUTINES IN THE SYSTEM.
C
         COMMON FIN, ICDC, IPTR, IX, OVFL, PERIOD, PTLST
C
         DIMENSION BUFF(3000), ISCOPE(5000), ISCOP1(2000),
     1             FINT(100,150), X(512), Y(512)
         EQUIVALENCE (BUFF(1), A(1)), (ISCOPE(1), A(1)),
     1            (ISCOP1(1),A(3002)), (X(1),A(3002)),
     2            (Y(1),A(4002)), (FINT(1),A(5003))
      END
--FORTRAN
```

Section 3

COMMAND PROCESSOR AND AUXILIARY ROUTINES

The Command Processor and its auxiliary routines constitute the main control mechanism of the Speech Display System. In order to take advantage of the presence of a dedicated computer, it was felt that a reasonably flexible interactive system must be developed. To achieve this, the command processor has been completely modularized into various subcomponents which perform the necessary command manipulations. The main program essentially consists of a series of calls to various subroutines which accept and/or process the commands sent in by the operator.

With the progress of time, the system became too big to fit into the core of the 1604 and thus it was split in two, with all the data gathering and preliminary processing routines going into one version and all of the speech processing and display generating routines going into the other version. Both versions contain the common diagnostic and utility programs.

The main program in the system is either TESTP (the processing version of the system) or TESTD (the data gathering version) depending on which system is being used. These programs call INPTCM, which reads in commands from either the typewriter or the paper tape reader, VARSET, PROSCL, DATGCL, TAPCOM, and DIAGNG which detect and execute the various commands, INITI which is used to load initial values into all the various system constants and variables, FINI which calculates the dependent variables and prints out the values of the system constants and variables on the printer, and DISSY which is used to produce a display of the speech data.

The command detection and execution has been spread over several subroutines in order to simplify their programming and to add flexibility to the system. Each of these subroutines operates in a similar manner. There is a character array containing a list of the commands which are processed by the subroutine and a DO loop which compares the input command to each of these names. If a match is found, control is transferred, by means of an indexed GO TO statement to the appropriate code to carry out the command.

The principle of modularity allows the command structure to be expanded easily since new subroutines to process new commands can be easily added. The main convention which must be followed is that if a command processing subroutine completes the processing of a command, it must set the command name to zero so that the calling program will know that the command was recognized and executed. Note that it would be possible to return other values as well which would perhaps signify alternative messages.

In addition to being used to run various display programs, the command processor can be used to change system variables. This allows the operator to easily modify the various displays. It also causes a certain number of problems due to the manner in which some of the system variables and constants interact. An example of this problem occurs in the spectrographic display, where the number of samples to be processed per time slice fixes the interval between frequency coefficients and vice versa.

The solution to this problem was to allow the user to set certain parameters independently and then have the system calculate the effect of these choices on the other dependent parameters and print them out (this operation is performed by the FINI subroutine). Thus for example, the operator can choose the desired number of data samples he wants processed per time slice in the spectrographic display and the system will respond by indicating the frequency increment between coefficients and the total frequency range which will be displayed given the current value of IFREQ.

An alphabetical list of all the commands executed by the system is given in Figure 3.1. A detailed description of each command is given in the section describing its respective detection and execution subroutine.

Command	Subroutine Which Executes Command	Command Operation
BEGN	TAPCOM	Rewind data tape & initialize system
BUFF	DIAGNG	Print out buffer contents
C	DIAGNG	Next input will be a comment
COPY	DATGCL	Copy data tape
DISP	DIAGNG	Display buffer contents on CRT
F	TAPCOM	Short form of FOWD = 1000
FINIS	PROSCL & DATGCL	Calculate dependent variables & turn off CRT
FIND	TAPCOM	Search data tape for specified speech words
FORME	PROSCL	Call FORMEX display routine
FOWD	TAPCOM	Move data tape forward NVAL samples
HEADT	TAPCOM	Process header block
HIEMP	PROSCL	Add high frequency emphasis to display data
INITI	PROSCL & DATGCL	Initialize system variables
INTAP	DIAGNG	Assign input command medium
IWIDE	TAPCOM	Assign window size for data tape display
LOCA	TAPCOM	Print out value of data pointer
MOVE	TAPCOM	Move data pointer to NVAL
NORMF	PROSCL	Normalize display data
OBTAI	DATGCL	Use A to D convertor to obtain speech data
PHOTO	PROSCL & DATGCL	Take picture of last display
PYRON	PROSCL	Call PYRON display routine
READF	DIAGNG	Read out display data stored on tape unit 3
REWIN	DIAGNG	Rewind tape unit NVAL
SAVEF	DIAGNG	Write display data on to tape unit 3
SPDIS	PROSCL	Display the display data array on the CRT
SPECT	PROSCL	Call SPECTO display routine
STAND	PROSCL	Produce a standard Spectrograph display
THRSP	DATGCL	Call THRSPIC data processing routine
WHATN	PROSCL & DATGCL	Call WHATNOW subroutine
ZEROC	PROSCL	Call ZEROC display routine.

Figure 3.1 Commands Executed by Speech System

3.1 TESTD and TESTP

These two programs are alternate forms of the main system program. They are actually almost identical with the exception of one subroutine call in each program. TESTD calls DATGCL while TESTP calls PROSCL. These two subroutines call the respective sets of routines used by the two versions of the system. This arrangement allows only those subroutines actually needed by a particular system version to be loaded into core memory.

The general operation of the command processor centers around the typewriter, specifically the commands typed on the typewriter by the operator. Every time control is given to the operator (this is done when INPTCM is called), a "+" is typed out and then the subroutine waits for a reply. Once this is received, the command and its parameters are read into the respective arguments of INPTCM and the subroutine returns.

The next step is to determine which command has been received and to execute it. This is done by calling the various command processing subroutines in turn. When the last subroutine returns, a check is made to see if the command was recognized by any of the subroutines. If so, the program which recognized the command will have executed it, so control will be returned to INPTCM. If not, a check is made for the system return command, i.e. ".", and if that is not the command, a message is typed out indicating that the command was not recognized before returning control to INPTCM.

There is one additional feature of the main program. Occasionally when using paper tape as the input medium, the paper tape command stream may become mixed up in such a way that it is not clear where you are on the paper tape. In a case like this, sense switch 1 can be set and the system will read in and type out successive lines from the paper tape input.

The operator must type any character and a carriage return in order to have another line read. This method allows him to adjust the paper tape and to find his place on it. Switch 1 can then be reset and the operation continued by typing a final character and carriage return.

This feature is implemented by checking for the state of sense switch 1 before calling INPTCM. If it is on, the call to INPTCM is skipped and instead NVAR is set to 1HC. This will cause the repeated execution of the comment command, until sense switch 1 is turned off.

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
          PROGRAM TESTD
      C
      C          THIS IS THE DATA PROCESSING VERSION OF TEST.
      C
      C
      C          THE COMMAND PROCESSING PROGRAM PERFORMS SEVERAL ACTIONS.
      C     FIRST, IT INTERACTS WITH THE OPERATOR TO DETERMINE ANY VARI-
      C     ABLE VALUES WHICH NEED TO BE MODIFIED.  IN THIS CASE THE OP-
      C     ERATOR TYPES THE NAME OF THE VARIABLE AND ITS NEW VALUE IN
      C     THE FORMAT REQUIRED BY INPTCM.  SECONDLY, THE COMMAND PROCESSOR
      C     ALLOWS THE OPERATOR TO CALL SUBROUTINES BY TYPING THE FIRST
      C     5 CHARACTERS IN THEIR NAMES.  FINALLY, IT WILL ALSO EXECUTE
      C     VARIOUS COMMANDS TO THE SYSTEM USING THE FORMAT,
      C                NVAR =  NVAL, NVAL2, FNAME
      C
      C     VARIABLE LIST.
      C
      C     NVAL  = FIXED POINT VALUE READ IN FROM TYPEWRITTER.
      C     NVAR  = 5 CHARACTER HOLLERITH VARIABLE USED TO CONTAIN THE COMMAND
      C             READ IN FROM TYPEWRITTER.
      C
            TITLE*
      C
      C          THIS STATEMENT CAUSES NVAR TO BE DEFINED AS A REAL NUM-
      C     BER AS IS REQUIRED BY THE CSL FORTRAN SYSTEM WHEN HANDLING
      C     TYPEWRITTER I/O.
      C
            TYPE R NVAR
            PRINT 1
00005     1 FORMAT (16H1 BEGIN NEW RUN.)
00005       IUNIT = 19
      C
      C          BEGIN COMMAND PROCESSING LOOP.  FIRST CHECK SENSE SWITCH
      C     1 TO SEE IF THE NEXT INPUT LINE SHOULD BE TREATED AS A COMMENT.
      C
00006    15 IF (SENSE SWITCH 1)    16, 18
00010    16 NVAR = 1HC
00012       GO TO 25
      C
      C          IF THE NEXT INPUT LINE IS GOING TO BE A COMMAND, CALL INPTCM
      C     TO READ IT INTO THE COMPUTER.
      C
00013    18 CALL INPTCM(NVAR,NVAL,NVAL2,FNAME,IUNIT)
      C
      C          DETERMINE THE COMMAND AND EXECUTE IT.
      C
00021    25 CALL VARSET(NVAR, NVAL)
00023       CALL DATCOL(NVAR, NVAL, NVAL2, FNAME)
00030       CALL TAPCOM(NVAR, NVAL)
00032       CALL DIAGNG(NVAR, NVAL, NVAL2)
00036    42 IF (NVAR - 1H.)       100, 300, 100
      C
      C          IF COMMAND WAS NOT RECOGNIZED, TYPE MESSAGE AND REPEAT
```

```
       CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
        C          REQUEST FOR COMMAND.
        C
00041   100        IF (NVAR)      101, 15, 101
00043   101        WRITE OUTPUT TAPE 19, 102
00046   102        FORMAT (15H WHAT WAS THAT.)
00046              GO TO 15
00047   300        CONTINUE
00047           END
--FORTRAN
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
          PROGRAM TESTP
      C
      C           THE COMMAND PROCESSING PROGRAM PERFORMS SEVERAL ACTIONS.
      C     FIRST, IT INTERACTS WITH THE OPERATOR TO DETERMINE ANY VARI-
      C     ABLE VALUES WHICH NEED TO BE MODIFIED.  IN THIS CASE THE OP-
      C     ERATOR TYPES THE NAME OF THE VARIABLE AND ITS NEW VALUE IN
      C     THE FORMAT REQUIRED BY INPTCM.  SECONDLY, THE COMMAND PROCESSOR
      C     ALLOWS THE OPERATOR TO CALL SUBROUTINES BY TYPING THE FIRST
      C     5 CHARACTERS IN THEIR NAMES.  FINALLY, IT WILL ALSO EXECUTE
      C     VARIOUS COMMANDS TO THE SYSTEM USING THE FORMAT,
      C                 NVAR =  NVAL, NVAL2, FNAME
      C
      C     VARIABLE LIST.
      C
      C     NVAL  = FIXED POINT VALUE READ IN FROM TYPEWRITTER.
      C     NVAR  = 5 CHARACTER HOLLERITH VARIABLE USED TO CONTAIN THE COMMAND
      C                 READ IN FROM TYPEWRITTER.
      C
            TITLE*
      C
      C           THIS STATEMENT CAUSES NVAR TO BE DEFINED AS A REAL NUM-
      C     BER AS IS REQUIRED BY THE CSL FORTRAN SYSTEM WHEN HANDLING
      C     TYPEWRITTER I/O.
      C
            TYPE R NVAR
            PRINT 1
00005     1 FORMAT (16H1 BEGIN NEW RUN.)
00005       IUNIT = 19
      C
      C           BEGIN COMMAND PROCESSING LOOP.  FIRST CHECK SENSE SWITCH
      C     1 TO SEE IF THE NEXT INPUT LINE SHOULD BE TREATED AS A COMMENT.
      C
00006    15 IF (SENSE SWITCH 1)     16, 18
00010    16 NVAR = 1HC
00012       GO TO 25
      C
      C           IF THE NEXT INPUT LINE IS GOING TO BE A COMMAND, CALL INPTCM
      C     TO READ IT INTO THE COMPUTER.
      C
00013    18 CALL INPTCM(NVAR,NVAL,NVAL2,FNAME,IUNIT)
      C
      C           DETERMINE THE COMMAND AND EXECUTE IT.
      C
00021    25 CALL VARSET(NVAR, NVAL)
00023       CALL PROSCL(NVAR, NVAL)
00025       CALL TAPCCM(NVAR, NVAL)
00027       CALL DIAGNG(NVAR, NVAL, NVAL2)
00033    42 IF (NVAR - 1H.)          76, 300, 76
00036    76 IF (NVAR - 5HPITCH)   100, 176, 100
      C
      C           IF COMMAND WAS NOT RECOGNIZED, TYPE MESSAGE AND REPEAT
      C     REQUEST FOR COMMAND.
      C
```

```
            CSL FORTRAN OF SEPT 1968,  DATE   8/12/71
  00041     100     IF (NVAR)      101, 15, 101
  00043     101     WRITE OUTPUT TAPE 19, 102
  00046     102     FORMAT (15H WHAT WAS THAT.)
  00046             GO TO 15
  00047     176     CALL FINI
  00050             DO 1761, I=1, ITIME
  00054             CALL ADJUS2(NSAMT, -1)
  00056             PRINT 1762, I
  00063     1762    FORMAT (//2X, I4)
  00063             CALL PITCH(BUFF(ISAMPB), NSAMT)
  00067             ISAMP = ISAMP + IDELT
  00070     1761    CONTINUE
  00072             GO TO 15
  00073     300     CONTINUE
  00073          END
 --FORTRAN
```

3.2 INPTCM (VARNAM, N1, N2, F1, IUNIT)

This program is used to read in commands for the command processor from the unit specified by IUNIT. At the present time, the only units allowed are the typewriter (IUNIT = 19) and the paper tape reader (IUNIT = 54). Any other value of IUNIT will produce an error comment on the typewriter and IUNIT = 19 will be assumed.

If the input medium is the typewriter, the subroutine will print out a '+' on the typewriter and wait for a reply from the operator. Otherwise it will simply read the next line of input. The main purpose for having the paper tape input facility is to allow the operator to produce pre-punched input tapes which he can then use to perform "sterotyped" operations without having to do a large amount of typing.

Once a command is read in by the read statement, the various parameters are automatically loaded with their respective values and the subroutine returns.

As the subroutine is presently written, it only accepts fixed format commands. However, since the system is modular, it would not cause undue perturbations if a free format input subroutine were used. Lack of time has been the main reason for neglecting this improvement.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
      SUBROUTINE INPTCM(VARNAM, N1, N2, F1, IUNIT)
   C
   C        THIS SUBROUTINE IS USED TO READ INPUT COMMANDS FROM THE OPERATOR
   C     AT THE TYPEWRITTER (IF IUNIT = 19) OR FROM THE PAPER TAPE READER
   C     (IF IUNIT = 54).  OTHER UNITS MAY BE FEASIBLE BUT HAVE NOT BEEN TRIED
   C     THE COMMAND, ALONG WITH ITS THREE POSSIBLE PARAMETERS, IS READ IN
   C     FIXED FORMAT AS GIVEN IN FORMAT STATEMENT 61.  THEN IT IS PRINTED ON
   C     THE PRINTER TO KEEP A RECORD OF WHAT IS HAPPENING, BEFORE THE SUB-
   C     ROUTINE RETURNS.
   C
         PLUS = 1H+
00003    IF (IUNIT - 19)      15, 20, 15
00006  15   IF (IUNIT - 54)      18, 60, 18
00011  18   WRITE OUTPUT TAPE 19, 19, IUNIT
00015  19   FORMAT (2X,I6,22H IS NOT VALID UNIT NO.)
00015  20   WRITE OUTPUT TAPE 19, 21, PLUS
00021  21   FORMAT (A1)
   C
   C           WAIT FOR DATA RETURN FROM OPERATOR OR TAPE READER.
   C
00021  60   READ INPUT TAPE IUNIT, 61, VARNAM, N1, N2, F1
00030  61   FORMAT (A5,1X,I6,1X,I6,1X,A6)
00030       PRINT 65, VARNAM, N1, N2, F1
00037  65   FORMAT (12H COMMAND IS A5,2H= I6,1H,I6,1H,A6)
00037       RETURN
00040    END
--FORTRAN
```

3.3 VARSET (COM, NVAL)

This subroutine checks for and executes commands involving the setting of system variables. It handles integer and floating point variables separately. In the case of integers the variable is set equal to NVAL. In the case of floating point numbers, the value of NVAL is converted to floating point and divided by 100.0 before being loaded into the selected variable. Thus the operator must multiply all floating point numbers by 100 when setting them under control of the command processor.

The method used to set the system variables relys on a technique of EQUIVALENCE'ing an array to the same memory space used to store the COMMON system variables. This allows the variables to be addressed by means of an indexed array. However, care must be taken to see that the order of the variables in core specified by the COMMON statement corresponds to the order of the names in the character arrays. In the CSL FORTRAN system, COMMON variables are assigned core space in the reverse order in which they are listed. This means that the last named variable should be EQUIVALENCE'ed to the zeroth position of the corresponding array and that the character arrays should list the variables in the reverse order of the COMMON statements.

The integer system variables which can be set by VARSET are the following:

IDEL	-	Distance between points being generated for the display measured in coordinate positions. Typical values = 4 to 8.
IDELT	-	Spacing between analyses of speech data, measured in number of time samples. Typical values = 100 to 1000.
IDELX	-	Number of display points to be interpolated between each x data value (usually time samples). Typical values - 2 to 16.
IDELY	-	Number of display points to be interpolated between each y data value (usually frequency slices). Typical values = 2 to 16.
IENDB	-	Number of locations to be used in the input data buffer by the OBTAIN subroutine. Must be a multiple of 1000. Maximum value = 20,000.
IFMAX	-	Maximum number of entries along frequency dimension of FINT array. May be adjusted along with ITMAX to vary the relative dimensions.
IFREQ	-	Number of frequency slices in FINT array.
ITIME	-	Number of time slices in FINT array.

ITLIM - Threshold value used by THRSPIC for determining whether or not to count a given sample when enumerating the number of significant samples in a data block. Typical value = 10 to 50.

ITMAX - Maximum number of entries along time dimension of FINT array. May be adjusted along with IFMAX to vary the relative dimensions of FINT.

IXMAR - Position of left edge of the three dimensional display generated by SPDISP. Measured in number of time slices from the left hand edge of the display. Typical value = 50.

LGNSM - Log to the base 2 of NSAMT. This tells the FFTB subroutine how many iterations it will have to perform. It must be set by the operator if he changes the value of NSAMT and intends to use FFTB.

LOCOP - Total number of samples which have been written out on to tape unit 3 by COPYT. The operator can preset this value in those cases where COPYT is used to add data to a tape already containing valid data.

NSAMT - Number of time samples to be analyzed per time slice. Whenever NSAMT is set, NSMT2 is automatically set to NSAMT/2. The reverse is not true however.

NSMT2 - Number of output coefficents to be calculated by FFTB. Must be a power of two. Usually equal to NSAMT/2 but may be set independently of NSAMT if desired by first setting NSAMT (NSMT2 is then set to NSAMT/2) and then setting NSMT2 to whatever is desired.

The floating point variables which can be set by VARSET are the following:

EMPFC - Factor used by HIEMP to control the non-linear emphasis of the high-frequency display components in those displays utilizing frequency as the vertical dimension. Typical values = .15 to .40.

FINTM - Minimum intensity value which will be displayed by SPDISP. Any values less than this will be set to the background intensity. This eliminates a great deal of superfluous intensity changing at the low value range. Typical value = 20.0.

OVFAC - Factor used by the NORMF subroutine when normalizing the display data. The highest intensity point to be displayed is set to 255*OVFAC and the other values are linearly normalized according to this value.

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
      SUBROUTINE VARSET(COM, NVAL)
C
C           THIS SUBROUTINE CHECKS FOR AND EXECUTES COMMANDS INVOLVING
C       THE SETTING OF SYSTEM VARIABLES.  IT TAKES CARE OF BOTH INTEGER
C       AND FLOATING POINT VARIABLES.
C
C       CHRF   - CHARACTER ARRAY CONTAINING THE NAMES OF THE FLOATING
C                POINT VARIABLES WHICH CAN BE SET.
C       CHRI   - CHARACTER ARRAY CONTAINING THE NAMES OF THE INTEGER
C                VARIABLES WHICH CAN BE SET.
C       COM    - COMMAND TO BE EXECUTED.
C       FVAR   - FLOATING POINT ARRAY WHICH HAS BEEN EQUIVALENCED TO
C                THE AREA CONTAINING THE FLOATING POINT SYSTEM VARIABLES.
C                THIS ALLOWS THESE VARIABLES TO BE ADDRESSED AS ELEMENTS
C                OF AN ARRAY.
C       IVAR   - INTEGER ARRAY WHICH HAS BEEN EQUIVALENCED TO THE AREA
C                CONTAINING THE INTEGER SYSTEM VARIABLES.  THIS ALLOWS
C                THESE VARIABLES TO BE ADDRESSED AS ELEMENTS OF AN ARRAY.
C       NVAL   - INTEGER PARAMETER FOR COM.
C
        TITLE*
        DIMENSION CHRI(14), CHRF(2), IVAR(14), FVAR(2)
C
C           NOTE THAT CHRI AND CHRF ARE SET EQUAL TO THE NAMES OF THE
C       VARIABLES IN THE COMMON AREA.  IT IS MANDITORY THAT THE ORDER OF
C       THE VARIABLES IN THESE ARRAYS BE THE REVERSE OF THE ORDER OF THE
C       VARIABLES IN THE COMMON STATEMENT, SINCE CSL FORTRAN ASSIGNS STOR-
C       AGE IN COMMON IN THE REVERSE ORDER IN WHICH THE VARIABLES ARE DE-
C       CLARED.
C
        DATA(CHRI(0) = 5HNSMT2, 5HLOCOP, 5HLGNSM, 5HIXMAR, 5HITMAX,
     1                 5HITLIM, 5HITIME, 5HIFREQ, 5HIFMAX, 5HIENDB,
     2                 5HIDELY, 5HIDELX, 5HIDELT, 4HIDEL)
        DATA(CHRF(0) = 5HOVFAC, 5HFINTM, 5HEMPFC)
C
C           NOTE THAT BY SETTING IVAR(0) AND FVAR(0) EQUIVALENT TO NSMT2
C       AND OVFAC, RESPECTIVELY, WE FORCE THE ITH ENTRY IN EACH OF THESE
C       ARRAYS TO BE EQUIVALENT TO THE VARIABLE WHOSE NAME IS STORED IN
C       THE ITH ENTRY OF THE CORRESPONDING CHARACTER ARRAY.  (SEE ABOVE
C       NOTE).
C
        EQUIVALENCE (IVAR(0), NSMT2), (FVAR(0), OVFAC)
C
C           COMPARE COM AGAINST THE NAMES IN CHRI TO SEE IF IT IS A
C       COMMAND TO SET THE VALUE OF AN INTEGER VARIABLE.
C
        DO 20, I=0, 14
00006   IF (COM - CHRI(I))      20, 50, 20
00011   20  CONTINUE
C
C           IF COM DID NOT MATCH A NAME IN CHRI, CHECK IT AGAINST CHRF
C       TO SEE IF IT IS A COMMAND TO SET THE VALUE OF A FLOATING POINT
C       VARIABLE.
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
       C
00012            DO 40, I=0, 2
00016            IF (COM - CHRF(I))      40, 60, 40
00021      40    CONTINUE
00022            IF (COM - 5HNSAMT)      41, 51, 41
00025      41    RETURN
       C
       C              IF COM MATCHED A VARIABLE IN CHRI OR CHRF, CHANGE THE
       C         APPROPRIATE VARIABLE.
       C
00026      50    IVAR(I) = NVAL
00027            IF (COM - 5HNSAMT)      500, 51, 500
00032      51    NSAMT = NVAL
00033            NSMT2 = NSAMT/2
00035            GO TO 500
00037      60    I = I + 1
00041            VAL = FLOATF(NVAL)/100.0
00044            GO TO (61, 62, 63) I
00055      61    OVFAC = VAL
00056            GO TO 500
00057      62    FINTM = VAL
00060            GO TO 500
00061      63    EMPFC = VAL
00062            GO TO 500
       C
       C              AFTER SETTING THE DESIGNATED VARIABLE, SET COM TO 0.0
       C         TO INDICATE THAT THE COMMAND WAS EXECUTED AND RETURN.
       C
00063     500    COM = 0.0
00064            RETURN
00065          END
--FORTRAN
```

3.4 PROSCL (COM, NVAL)

This subroutine checks for and executes commands which involve calls to those subroutines which are loaded with the processing version of the system. It also checks for commands which are used exclusively by the data gathering version of the system but in this case, it simply types a message reminding the operator that that command does not exist in the currently loaded system.

The execution of the commands is relatively straightforward and simply involves a CALL statement. In some cases, however, particularly in the case where the subroutine is a speech processing routine working directly on the speech data, it was found desirable to first call FINI so that a printout of the status of the system variables and constants right before the execution of the subroutine would be available.

During the course of the development of the speech system, certain operations came to be performed very often. Eventually it was decided to have a single command which would perform a whole series of conventional commands. STAND is an example of this. There are actually two varients. If no numeric value is given (in which case the I/O programs set NVAL = -0), it will not move the data tape while if a value is given, it will move to the position specified, look for a header block and load in the proper value for ITIME from the header. If no header is found, an error comment will be made.

Once the tape is positioned, if required, the command causes a series of calls to be made which result in a spectrographic display of the data. The last action is to set COM to 4HLOCA. This will cause the LOCA command to be executed thus indicating the value of ISAMP, provided that the subroutine which detects and executes the LOCA command comes after PROSCL in the list of calls in TESTP.

The commands which are executed by PROSCL are as follows:

FINIS	-	call	FINI
FORME	-	call	FORMEX
HIEMP	-	call	HIEMP
INITI	-	call	INITI
NORMF	-	call	NORMF
PHOTO	-	call	PHOTO (NVAL)
PYRON	-	call	PYRON
SPDIS	-	call	SPDISP
SPECT	-	call	SPECTO
TFFBT	-	call	TFFBT
WHATN	-	call	WHATNOW
ZEROC	-	call	ZEROC

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
        SUBROUTINE PRCSCL(COM, NVAL)
  C
  C           THIS SUBROUTINE CHECKS FOR COMMANDS INVOLVING CALLS TO
  C       SUBROUTINES WHICH ARE USED BY THE SPEECH PROCESSING VERSION
  C       OF THE SYSTEM.  IT PRINTS A MESSAGE FOR ANY COMMAND REQUEST-
  C       ING A SUBROUTINE WHICH IS NOT LOADED.
  C
  C       BUFF   - COMMON ARRAY.  INPUT DATA BUFFER.
  C       CHR1   - CHARACTER ARRAY CONTAINING THE NAMES OF THE SPEECH SYSTEM
  C                SUBROUTINES WHICH ARE LOADED IN THE PROCESSING VERSION OF
  C                THE SYSTEM.
  C       CHR2   - CHARACTER ARRAY CONTAINING THE NAMES OF THE SPEECH SYSTEM
  C                SUBROUTINES WHICH ARE NOT LOADED IN THE PROCESSING VERSION
  C                OF THE SYSTEM.
  C       COM    - COMMAND TO BE EXECUTED.
  C       IDELT  - COMMON VARIABLE.  SPACING (IN NO. OF TIME SAMPLES) BETWEEN
  C                ANALYSES OF DATA.
  C       ISAMP  - COMMON VARIABLE.  POSITION OF SAMPLE POINTER RELATIVE TO
  C                THE DATA TAPE.
  C       ISAMPB - COMMON VARIABLE.  POSITION OF SAMPLE POINTER RELATIVE TO
  C                BUFF.
  C       ITIME  - COMMON VARIABLE.  NUMBER OF TIME POSITIONS IN DISPLAY.
  C       IWID   - COMMON VARIABLE.  WIDTH OF DATA DISPLAY WINDOW (IN NO. OF
  C                SAMPLES).
  C       LBLK   - COMMON VARIABLE.  NUMBER OF ENTRIES IN EACH BLOCK IN BUFF.
  C       NVAL   - INTEGER PARAMETER FOR COM.
  C
  C
          TITLE*
          DIMENSION CHR1(12), CHR2(2)
          DATA (CHR1(0)=5HFORME, 5HSPECT, 5HSPDIS, 5HPYRON, 5HNORMF,
       1                 5HZEROC, 5HHIEMP, 5HSTAND, 5HTFFTB, 5HPHCTO,
       2                 5HWHATN, 5HINIT1, 5HFINIS)
          DATA (CHR2(0)=4HCOPY,  5HOBTAI, 5HTHRSP)
  C
  C           COMPARE COM AGAINST THE NAMES IN CHR1 TO SEE IF IT IS
  C       A COMMAND TO CALL A SUBROUTINE IN THE PROCESSING VERSION OF
  C       THE SYSTEM.
  C
          DO 20, I=0, 12
00006     IF (COM - CHR1(I))     20, 50, 20
00011  20 CONTINUE
  C
  C           IF COM DID NOT MATCH A NAME IN CHR1, CHECK IT AGAINST
  C       CHR2 TO SEE IF IT IS A COMMAND TO CALL A SUBROUTINE NOT IN
  C       THE PROCESSING VERSION OF THE SYSTEM.
  C
00012     DO 40, I=0, 2
00016     IF (COM - CHR2(I))     40, 60, 40
00021  40 CONTINUE
00022     RETURN
  C
  C           IF COM MATCHED A NAME IN CHR1, TRANSFER CONTROL TO THE
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
       C        APPROPRIATE CALL.
       C
00023    50     I = I + 1
       C              FORME-SPECT-SPDIS-PYRON-NORMF-ZEROC-HIEMP-STAND-
00025           GO TO ( 100,  110,  120,  130,  140,  150,  160,  170,
       C
       C              IFFTE-PHOTO-WHATN-INITI-FINIS
            1         180,  190,  200,  210,  220) I
       C
       C             IF COM MATCHED A NAME IN CHR2, PRINT OUT A MESSAGE
       C        TELLING THE OPERATOR IT IS NOT LOADED.
       C
00050    60     WRITE OUTPUT TAPE 19, 61
00053           PRINT 61
00056    61     FORMAT (20H PROGRAM NOT LOADED.)
00056           GO TO 500
       C
       C             CALL REQUESTED SUBROUTINE.
       C
00057   100     CALL FORMEX
00060           GO TO 500
00061   110     CALL FINI
00062           CALL SPECTO
00063           GO TO 500
00064   120     CALL SPDISP
00065           GO TO 500
00066   130     CALL FINI
00067           CALL PYRON
00070           GO TO 500
00071   140     CALL NORMF
00072           GO TO 500
00073   150     CALL FINI
00074           CALL ZEROC
00075           GO TO 500
00076   160     CALL HIEMP
00077           GO TO 500
00100   170     IF (NVAL)      172, 172, 171
00102   171     ISAMP = NVAL
00103           CALL ADJUS2(IWID, -1)
       C
       C             LOAD ITIME WITH THE NUMBER OF TIME SAMPLES IN THE CURRENT
       C        SPEECH WORD.  BUFF(ISAMPB + 4) WILL CONTAIN THE NUMBER OF BLOCKS
       C        IN THE SPEECH WORD (FLOATING POINT) AS A RESULT OF THE UNPACKING
       C        OPERATION IN ADJUS2.  (NOTE THAT UNPACK STARTS AT THE RIGHT
       C        QUARTER OF THE PACKED WORD).
       C
00105           ITIME = (BUFF(ISAMPB+4)*LBLK)/IDELT
00116           ISAMP = ISAMP + LBLK
00120   172     CALL FINI
00122           CALL SPECTO
00123           CALL NORMF
00124           CALL HIEMP
00125           CALL SPDISP
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
              COM = 4HLOCA
              RETURN
      C
      C            NOTE THAT SETTING COM = 4HLOCA WILL CAUSE THAT COMMAND TO
      C       BE EXECUTED AFTER PROSCL RETURNS, PROVIDED THAT PROSCL IS CALLED
      C       BEFORE THE SUBROUTINE WHICH DETECTS AND PROCESSES THE LOCA COMMAND
      C
        180   CALL FINI
              CALL TFFTB
              GO TO 500
        190   CALL PHOTO(NVAL)
              GO TO 500
        200   CALL WHATNOW
              GO TO 500
        210   CALL INITI
              GO TO 500
        220   CALL FINI
              GO TO 500
      C
      C            AFTER CALLING THE APPROPRIATE SUBROUTINES, SET COM TO 0.0
      C       TO INDICATE THAT THE COMMAND WAS EXECUTED AND NO OTHER COMMAND
      C       NEEDS TO BE PERFORMED.
      C
        500   COM = 0.0
              RETURN
            END
--FORTRAN
```

3.5 DATGCL (COM, NVAL, NVAL2, FNAME)

This subroutine checks for and executes commands which involve calls to those subroutines which are loaded with the data gathering version of the system. It also checks for commands which are used exclusively by the processing version of the system but in this case, it simply types a message reminding the operator that that command does not exist in the currently loaded system.

The execution of the subroutines is entirely straightforward and simply involves a CALL statement. The commands which are executed by DATGCL are as follows:

COPY	-	call	COPY (NVAL, NVAL2, FNAME)
FINIS	-	call	FINI
INITI	-	call	INITI
OBTAI	-	call	OBTAIN
PHOTO	-	call	PHOTO (NVAL)
THRSP	-	call	THRSPIC (NVAL, NVAL2)
WHATN	-	call	WHATNOW

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
           SUBRCUTINE DATCCL(COM, NVAL, NVAL2, FNAME)
      C
      C          THIS SUBROUTINE CHECKS FOR CCMMANDS INVCLVING CALLS TO
      C       SUBROUTINES WHICH ARE CNLY USED BY ONE OF THE TWO SYSTEM
      C       VERSIONS.  IT EXECUTES ALL CALLS TO SUBROUTINES USED BY THE
      C       DATA GATHERING VERSION AND PRINTS OUT A MESSAGE FOR ANY COM-
      C       MAND REQUESTING A SUBRCUTINE WHICH IS NOT LOADED.
      C
      C       CHR1  - CHARACTER ARRAY CONTAINING THE NAMES OF THE SPEECH SYSTEM
      C               SUBRCUTINES WHICH ARE NOT LOADED IN THE DATA INPUT VERSION
      C               CF THE SYSTEM.
      C       CHR2  - CHARACTER ARRAY CONTAINING THE NAMES OF THE SPEECH SYSTEM
      C               SUBRCUTINES WHICH ARE LOADED IN THE DATA INPUT VERSION OF
      C               THE SYSTEM.
      C       COM   - CCMMAND TO BE EXECUTED.
      C       FNAME - CHARACTER VARIABLE PARAMETER FOR COM.
      C       NVAL  - FIRST INTEGER PARAMETER FCR COM.
      C       NVAL2 - SECCND INTEGER PARAMETER FOR COM.
      C
              TITLE*
              DIMENSION CHR1(8), CHR2(6)
              DATA (CHR1(0)=5HFORME, 5HSPECT, 5HSPDIS, 5HPYRON, 5HNORMF,
            1                5HZERCC, 5HHIEMP, 5HSTAND, 5HTFFTB)
              DATA (CHR2(0)=4HCOPY,  5HOBTAI, 5HTHRSP, 5HPHOTO, 5HWHATN,
            1                5HINITI, 5HFINIS)
      C
      C          COMPARE COM AGAINST THE NAMES IN CHR1 TC SEE IF IT IS A
      C       COMMAND TC CALL A SUBRCUTINE IN THE PROCESSING VERSION OF THE
      C       SYSTEM.
      C
              DO 20, I=0, 8
00006         IF (COM - CHR1(I))      20, 50, 20
00011    20   CONTINUE
      C
      C          IF CCM DID NOT MATCH A NAME IN CHR1, CHECK IT AGAINST
      C       CHR2 TO SEE IF IT IS A COMMAND TO CALL A SUBRCUTINE IN THE
      C       DATA GATHERING VERSIIN OF THE SYSTEM.
      C
00012         DO 40, I=0, 6
00016        IF (COM - CHR2(I))      40, 60, 40
00021    40   CONTINUE
00022         RETURN
      C
      C          IF CCM MATCHED A NAME IN CHR1, PRINT OUT A MESSAGE
      C       TELLING THE CPERATCR IT IS NOT LOADED.
      C
00023    50   WRITE OUTPUT TAPE 19, 51
00026         PRINT 51
00031    51   FORMAT (20H PROGRAM NOT LOADED.)
00031         GO TO 500
      C
      C          IF CCM MATCHED A NAME IN CHR2, TRANSFER CONTROL TO
      C       THE APPRCPRIATE CALL.
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
        C
00032    60     I = I + 1
        C              COPY -OBTAI-THRSP-PHOTO-WHATN-INITI-FINIS
00034           GO TO ( 100,  110,  120,  130,  140,  150,  160) I
        C
        C       CALL REQUESTED SUBROUTINE.
        C
        C
        C            COPY.
        C
00051   100     CALL COPYT(NVAL, NVAL2, FNAME)
00055           GO TO 500
        C
        C            OBTAI.  IN THE CASE OF OBTAIN, THE SUBROUTINE WILL
        C       LOOP UNTIL THE MACHINE IS STOPPED MANUALLY.  THERE IS
        C       NO RETURN.
        C
00056   110     CALL OBTAIN
        C
        C            THRSP.
        C
00057   120     IF (NVAL2)     122, 121, 122
00061   121     NVAL2 = LCCCF/LBLK - 1
00064   122     CALL THRSPIC(NVAL, NVAL2)
00066           GO TO 500
        C
        C            PHOTO.
        C
00067   130     CALL PHOTO(NVAL)
00070           GO TO 500
        C
        C            WHATN.
        C
00071   140     CALL WHATNOW
00072           GO TO 500
        C
        C            INITI.
        C
00073   150     CALL INITI
00074           GO TO 500
        C
        C            FINIS.
        C
00075   160     CALL FINI
00076           GO TO 500
        C
        C            AFTER CALLING THE APPROPRIATE SUBROUTINE OR PRINTIMG THE
        C       MESSAGE, SET COM TO 0.0 TO INDICATE THAT THE COMMAND HAS BEEN
        C       TAKEN CARE OF AND THEN RETURN.
        C
00077   500     COM = 0.0
00100           RETURN
00101         END
--FORTRAN
```

3.6 TAPCOM (COM, NVAL)

This subroutine checks for and executes commands involving the data tape position and the data display. After performing the indicated operations it readjusts the data tape and/or generates an updated display if necessary. The commands are as follows:

BEGN - Set up the system to begin reading a data tape, i.e. rewind tape unit 4 and load up BUFF from it. Also set all constants and variables to the initial values determined by INITI and print them out.

F - Short version of FOWD=001000.

FIND - This command is used to find specific speech words on a data tape by looking at the header blocks along the tape. If NVAL is positive and non-zero, the system will accept the next NVAL 5-character entries as names to be searched for and will save them in an array. If NVAL is greater than 16 (the number of entries in the array) an error comment will be given. If NVAL is zero or negative, the system will search the data tape, beginning with the block in which ISAMP is currently located, for a header containing the name of any word previously saved in the storage array. Care must be taken since if no match is found, the system will eventually run off the end of the data tape.

FOWD - Move ISAMP forward NVAL samples and recompute the display beginning at that point.

HEADT - Causes the block currently being pointed to by ISAMP to be interpreted as a header block. ITIME is calculated from the length of the data word as indicated by the header block and then ISAMP is advanced by one block length.

IWIDE - Change the width of the display to NVAL samples and recompute the display.

LOCA - Print out the current value of ISAMP on the printer and on the typewriter.

MOVE - Move ISAMP to the position on the data tape indicated by NVAL and recompute the display.

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
        SUBROUTINE TAPCOM(COM, NVAL)
C
C             THIS SUBROUTINE CHECKS FOR AND EXECUTES COMMANDS INVOLVING
C        THE DATA DISPLAY AND DATA TAPE POSITION.
C
C        CHAR   - CHARACTER ARRAY CONTAINING THE NAMES OF THE COMMANDS WHICH
C                 ARE EXECUTED BY TAPCOM.
C        CHRS   - CHARACTER ARRAY USED TO HOLD THE NAMES OF THE WORDS BEING
C                 LOOKED FOR IN THE EXECUTION OF A FIND COMMAND.
C        COM    - COMMAND TO BE EXECUTED.
C        FTMP   - TEMPORARY FLT. PT. VARIABLE USED TO CONTAIN THE NO. OF
C                 SAMPLES TO BE PLOTTED BY DISSY.
C        IBUF   - TEMPORARY BUFFER USED BY BLOCKRD TO HOLD DATA WHEN THE
C                 DATA TAPE IS BEING SEARCHED.
C        ICBLK  - COMMON VARIABLE.  BLOCK NO. OF THE CURRENT BLOCK WHICH IS
C                 STORED IN THE FIRST BLOCK OF BUFF.
C        IDELT  - COMMON VARIABLE.  SPACING (IN NO. OF SAMPLES) BETWEEN
C                 ANALYSES OF DATA.
C        ISAMP  - COMMON VARIABLE.  POSITION OF SAMPLE POINTER RELATIVE TO
C                 THE DATA TAPE.
C        ISAMPB- COMMON VARIABLE.  POSITION OF SAMPLE POINTER RELATIVE TO
C                 BUFF.
C        ISCOP1- COMMON ARRAY.  DISPLAY BUFFER.
C        IUNIT  - COMMON VARIABLE.  UNIT TO BE USED AS THE COMMAND INPUT
C                 DEVICE.
C        IWID   - COMMON VARIABLE.  WIDTH OF DATA DISPLAY WINDOW (IN NO. OF
C                 SAMPLES).
C        LBLK   - COMMON VARIABLE.  NUMBER OF ENTRIES IN EACH BLOCK IN BUFF.
C        MASK   - CODE USED TO IDENTFY A HEADER BLOCK.
C        NBLK   - COMMON VARIABLE.  NUMBER OF DATA BLOCKS IN BUFF.
C        NUM    - HIGHEST INDEX USED IN CHRS.
C        NVAL   - INTEGER PARAMETER USED BY COM.
C        SYM    - LOCATION CONTAINING SYMBOLIC NAME OF THE DATA ASSOCIATED
C                 WITH THE HEADER BLOCK CURRENTLY BEING LOOKED AT.
C        XB     - DUMMY VARIABLE USED IN INPTCM AND DISSY CALLS.
C
         TITLE*
         DIMENSION CHAR(7), CHRS(15), IBUF(250)
         EQUIVALENCE (IBUF(1), A(3002)), (SYM, A(3004))
         DATA (MASK=77777777777777778)
         DATA (CHAR(0)=4HBEGN,  1HF,      4HFOWD,  5HIWIDE, 4HLOCA,
      1                 4HMOVE,  5HHEADT, 4HFIND)
C
C             COMPARE COM AGAINST THE NAMES IN CHAR TO SEE IF IT IS A
C        COMMAND INVOLVING THE STATE OF THE DATA DISPLAY OR THE POSI-
C        TION OF THE DATA TAPE.
C
         DO 20, I=0, 7
00006    IF (COM - CHAR(I))    20, 50, 20
00011  20   CONTINUE
00012    RETURN
C
C             IF COM MATCHED A NAME IN CHAR, TRANSFER CONTROL TO THE
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
        C          APPROPRIATE CODE.
        C
00013    50        I = I + 1
        C                BEGN -   F   -FOWD -IWIDE-LOCA -MOVE -HEADT-FIND
00015              GO TO ( 100,  110,  120,  130,  140,  150,  160,  170) I
        C
        C                EXECUTE THE APPROPRIATE COMMAND.
        C
00033   100        REWIND 4
00035              CALL INITI
00036              ICBLK = -NBLK
00037              ISAMP =  NVAL
        C
        C                FILL UP SAMPLE BUFFER BY CALLING ADJUS2.  SINCE ICBLK = -NBLK,
        C          ADJUS2 WILL KNOW THAT THE TAPE IS RESTING AT BLOCK NUMBER ZERO.
        C          ALSO, SINCE ISAMP IS SOME POSITIVE NUMBER, THERE WILL ALWAYS BE AN
        C          OVERFLOW CONDITION, AND WITH NFLAG = -1, THE BUFFER WILL BE ADJUSTED
        C          WITH ISAMP IN THE LEFTMOST BLOCK.
        C
00041              CALL ADJUS2(IWID, -1)
00043              GO TO 510
00044   110        NVAL = 1000
00045   120        ISAMP = ISAMP + NVAL
00046              GO TO 500
00050   130        IWID = NVAL
00051              GO TO 500
00052   140        WRITE OUTPUT TAPE 19, 141, ISAMP
00056              PRINT 141, ISAMP
00062   141        FORMAT (7H ISAMP=I6)
00062              GO TO 520
00063   150        ISAMP = NVAL
00064              GO TO 500
00065   160        ITIME = (BUFF(ISAMP6+4)*LBLK)/IDELT
00076              ISAMP = ISAMP + LBLK
00100              GO TO 500
        C
        C                FIND.
        C
00102   170        IF (NVAL)    173, 173, 171
        C                CHECK TO SEE THAT THE NUMBER OF NAMES WILL NOT
        C          EXCEED THE BOUNDS OF THE CHRS ARRAY AND THEN READ THEM
        C          IN.
00104   171        IF (NVAL - 16)    1713, 1713, 1711
00107   1711       WRITE OUTPUT TAPE 19, 1712
00112   1712       FORMAT (24H NO. OF WORDS TOO LARGE.)
00112              GO TO 500
00113   1713       NUM = NVAL - 1
00114              DO 172, I=0, NUM
00121              CALL INPTCM(COM, XB, XB, XB, IUNIT)
00127              CHRS(I) = COM
00130   172        CONTINUE
00131              GO TO 520
        C                SEARCH DATA TAPE FOR A NAME CONTAINED IN CHRS.
```

```
      CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
00132   173     ISAMP = (ISAMP/LBLK)*LBLK
00136           CALL ADJUS2(0, 0)
00140   174     CALL BLOCKRD(0, 0, 0)
00144           ICBLK = ICBLK + 1
00145           IF (IBUF(1) - MASK)   1761, 175, 1761
00153   175     DO 176, I=0, NUM
00157           IF (CHRS(I) - SYM)     176, 177, 176
00162   176     CONTINUE
00163   1761    ISAMP = ISAMP + LBLK
00164           GO TO 174
      C               WHEN A NAME IS FOUND, TYPE MESSAGE, EXECUTE A HEADT COMMAND,
      C         AND RETURN.
00166   177     WRITE OUTPUT TAPE 19, 178, SYM, ISAMP
00173           PRINT 178, SYM, ISAMP
00200   178     FORMAT (7H FOUND A6,12H AT ISAMP = I6)
00200           CALL ADJUS2(IWID, -1)
00202           ITIME = (FLOATF(IBUF(2))*LBLK)/IDELT
00215           ISAMP = ISAMP + LBLK
00217           GO TO 500
      C
      C               ADJUST BUFFER IF NECESSARY.
      C
00221   500     CALL ADJUS2(IWID, 1)
      C
      C               GENERATE NEW DISPLAY AND THEN RETURN.
      C
00223   510     FTMP = IWID
00224           XB   = 0.0
00226           CALL DISSY(XB,BUFF(ISAMPB),IWID,ISCOP1,2000,FTMP,1024.,0)
00242   520     COM = 0.0
00243           RETURN
00244         END
--FORTRAN
```

3.7 DIAGNG (COM, NVAL, NVAL2)

This subroutine checks for and executes commands which are used for diagnostic or utility purposes, i.e., those commands which can be used to look at, save, or read back various types of data. The execution of the commands, whose descriptions are given below, is very straightforward.

BUFF - Causes the contents of array A between the limits NVAL and NVAL2 to be printed out. Note that since all major data arrays have been defined equivalent to some portion of A, this command allows the operator to print out any data array provided he knows its location within A.

C - Causes the next line of data from the input device to be treated as a comment. It will be printed out on the printer and typed out on the typewriter.

DISP - Causes the contents of array A between the limits NVAL and NVAL2 to be displayed on the CRT.

INTAP - Sets the input medium. If NVAL = 0, the typewriter is selected. Otherwise, the paper tape reader is used. Note that IUNIT is not loaded directly by the operator which could be done if it were included with the other operator-controlled system variables.

This was done to ensure that it would not be inadvertently set to some value other than those allowed by the I/O programs.

READF - Read in the FINT array and the values of IFREQ and ITIME from magnetic tape unit #3. If NVAL is specified and not zero, read the tape NVAL times. This allows the operator to space past save areas on the tape without giving successive READF commands.

REWIN - Rewind magnetic tape unit NVAL.

SAVEF - Write out FINT array on to magnetic tape unit 3. Also save the values of IFREQ and ITIME to indicate how much of the data is valid.

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
        SUBROUTINE DIAGNG(COM, NVAL, NVAL2)
 C
 C           THIS SUBROUTINE CHECKS FOR AND EXECUTES COMMANDS WHICH ARE
 C      USED FOR DIAGNOSTIC AND UTILITY PURPOSES, IE. THOSE COMMANDS WHICH
 C      CAN BE USED TO LOOK AT, SAVE, OR READ BACK VARIOUS TYPES OF DATA.
 C
 C      A       - COMMON ARRAY.  NAME FOR SYSTEMS COMMON STORAGE ARRAY.
 C      CHAR    - CHARACTER ARRAY CONTAINING THE NAMES OF THE COMMANDS WHICH
 C                ARE EXECUTED BY DIAGNG.
 C      COM     - COMMAND TO BE EXECUTED.
 C      FINT    - COMMON ARRAY.  OUTPUT INTENSITY ARRAY.
 C      FTMP    - TEMPORARY FLT. PT. VARIABLE USED TO CONTAIN THE NUMBER OF
 C                SAMPLES TO BE PLOTTED BY DISSY.
 C      IFMAX - COMMON VARIABLE.  MAXIMUM FREQUENCY INDEX FOR FINT.
 C      IFREQ - COMMON VARIABLE.  NUMBER OF FREQUENCY POSITIONS IN DISPLAY.
 C      IMAX    - MAXIMUM LENGTH OF FINT ARRAY.
 C      ISCOP1- COMMON ARRAY.  DISPLAY BUFFER FOR DISSY.
 C      ITIME - COMMON VARIABLE.  NUMBER OF TIME POSITIONS IN DISPLAY.
 C      ITMAX - COMMON VARIABLE.  MAXIMUM TIME INDEX FOR FINT.
 C      ITMP    - TEMPORARY INTEGER VARIABLE.
 C      IUNIT - COMMON VARIABLE.  UNIT TO BE USED AS THE COMMAND INPUT
 C                DEVICE.
 C      MESS    - TEMPORARY BUFFER USED TO HOLD INPUT COMMENT MESSAGE
 C                PRIOR TO ITS BEING PRINTED AND TYPED.
 C      NVAL    - FIRST INTEGER PARAMETER FOR COM.
 C      NVAL2 - SECOND INTEGER PARAMETER FOR COM.
 C      XB      - DUMMY VARIABLE USED IN DISSY.
 C
        TITLE*
        DIMENSION CHAR(6), MESS(10)
        DATA (CHAR(0)=4HBUFF,   4HDISP,   5HREADF, 5HSAVEF, 5HREWIN,
      1                  1HC,      5HINTAP)
 C
 C           COMPARE COM AGAINST THE NAMES IN CHAR TO SEE IF IT IS A
 C      COMMAND INVOLVING DIAGNOSTICS.
 C
        DO 20, I=0, 6
00006   IF (COM - CHAR(I))      20, 50, 20
00011  20  CONTINUE
00012   RETURN
 C
 C           IF COM MATCHED A NAME IN CHAR, TRANSFER CONTROL TO THE
 C      APPROPRIATE CODE.
 C
00013  50  I = I + 1
 C              BUFF -DISP -READF-SAVEF-REWIN-  C  -INTAP
00015   GO TO ( 100,  110,  120,  130,  140,  150,  160) I
 C
 C           PERFORM THE REQUESTED DIAGNOSTICS.
 C
00032 100  PRINT 601, (A(I), I=NVAL, NVAL2)
00044   GO TO 500
00045 110  ITMP = NVAL2 - NVAL + 1
```

```
   CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
00047          FTMP = ITMP
00050          XB   = 0.0
00052          CALL DISSY(XE,A(NVAL),ITMP,ISCOP1,2000,FTMP,1024.,0)
00066          GO TO 500
00067    120   IMAX = IFMAX*ITMAX
00070          IF (NVAL)      122, 121, 122
00073    121   NVAL = 1
00074    122   DO 123, J=1, NVAL
00100    123   READ TAPE 3, (FINT(I), I=0, IMAX)
       C
       C       RESTORE THE VALUES OF ITIME AND IFREQ.
       C
00112          ITIME = FINT(IMAX)
00116          IFREQ = FINT(IMAX-1)
00123          GO TO 500
       C
       C
00125    130   IMAX = IFMAX*ITMAX
00126          FINT(IMAX)    = ITIME
00133          FINT(IMAX-1) = IFREQ
00140          WRITE TAPE 3, (FINT(I), I=0, IMAX)
00152          GO TO 500
00153    140   REWIND NVAL
00155          GO TO 500
       C
       C               READ IN A COMMENT FROM THE CURRENT INPUT MEDIUM AND WRITE
       C       IT OUT ON THE TYPEWRITTER AND THE PRINTER.  THEN WAIT FOR ANY
       C       TYPE OF INPUT FROM THE TYPEWRITER BEFORE CONTINUING.
       C
00156    150   READ INPUT TAPE IUNIT, 151, MESS
00170    151   FORMAT (10(A8))
00170          WRITE OUTPUT TAPE 19, 151, MESS
00202          PRINT 151, MESS
00214          READ INPUT TAPE 19, 151, MESS
00226          GO TO 500
00227    160   IUNIT = 19
00230          IF (NVAL)      161, 500, 161
00232    161   IUNIT = 54
00233          GO TO 500
       C
       C               AFTER PERFORMING THE REQUESTED DIAGNOSTICS, SET COM TO
       C       0.0 TO INDICATE THAT THE COMMAND WAS EXECUTED AND THEN RETURN.
       C
00234    500   COM = 0.0
00235          RETURN
00236    601   FORMAT (2X,10(F8.2,2X))
00236    501   FORMAT (2X,10(F8.2,2X))
00236        END
--FORTRAN
```

3.8 INITI

This subroutine is used to initialize the values of the system constants and variables. Its structure is extremely simple since it consists almost entirely of assignment statements. After the values have been assigned, INITI calls FINI to calculate the remaining dependent variables and to print the values of the variables out on the printer.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71  .
      SUBROUTINE INITI
C
C           THIS SUBROUTINE IS USED TO INITIALIZE THE VALUES OF THE
C        VARIOUS COMMCN AREA VARIABLES USED BY THE SYSTEM.  AFTER THIS
C        HAS BEEN DONE, THE SUBROUTINE PRINTS OUT A MESSAGE, WHICH
C        INDICATES THAT ALL VARIABLES HAVE THEIR STANDARD VALUES AND
C        THAT ANY VARIATIONS DESIRED MUST BE TYPED IN.
C
C        EMPFC = FACTCR USED BY HIEMP TO CONTROL THE NON-LINEAR EMPHASIS
C                CF THE HIGH FREQUENCY DISPLAY COMPONENTS IN THOSE DIS-
C                PLAYS UTILIZING FREQUENCY AS THE VERTICAL DIMENSION.
C        FINI  = SUBRCUTINE.  CALCULATES THE DEPENDENT PARAMETERS BASED
C                ON THE VALUES JUST GIVEN TO THE INDEPENDENT PARAMETERS,
C                TURNS OFF THE CRT, AND PRINTS OUT THE CURRENT VALUES
C                OF THE SYSTEM VARIABLES AND CONSTANTS ON THE PRINTER.
C        FINTM = MINIMUM INTENSITY VALUE WHICH WILL BE DISPLAYED BY SPDISP.
C        IDEL  = DISTANCE BETWEEN POINTS BEING GENERATED FOR DISPLAY
C                BY SPDISP.
C        IDELX = NUMBER OF DISPLAY POINTS INTERPOLATED BETWEEN EACH HORI-
C                ZONTAL DATA POINT IN THE ARRAY TO BE DISPLAYED BY SPDISP.
C        IDELY = NUMBER OF DISPLAY POINTS INTERPOLATED BETWEEN EACH VERTI-
C                CAL DATA POINT IN THE ARRAY TO BE DISPLAYED BY SPDISP.
C        IDELT = SPACING(IN NO. OF TIME SAMPLES) BETWEEN ANALYSES OF DATA.
C        IENDB = NUMBER OF LOCATIONS IN INPUT DATA BUFFER WHICH WILL BE
C                USED BY THE OBTAIN SUBROUTINE.
C        IFMAX = MAXIMUM NUMBER OF ENTRIES ALONG THE FREQUENCY DIMENSION
C                CF FINT.
C        IFREQ = NUMBER OF FREQUENCY POSITIONS IN DISPLAY.
C        ISAMF = SAMPLING FREQUENCY OF A TC D CONVERTCR.
C        ITIME = NUMBER OF TIME POSITIONS IN DISPLAY.
C        ITLIM = USED BY THRSPIC AS THE THRESHOLD VALUE FOR DETERMINING
C                WHETHER OF NOT TO COUNT A GIVEN SAMPLE WHEN ENUMERATING
C                THE NUMBER CF SIGNIFICANT SAMPLES IN A DATA BLOCK.
C        ITMAX = MAXIMUM NUMBER OF ENTRIES ALONG THE TIME DIMENSION OF
C                FINT.
C        IWID  = WIDTH OF DATA DISPLAY WINDOW(IN NO. OF SAMPLES).
C        IXMAR = USED BY SPDISP TO DETERMINE THE LEFT HAND MARGIN OF THE
C                DISPLAY RELATIVE TO THE X=0 LOCATION.  EXPRESSED IN NO.
C                OF EQUIVALENT TIME SAMPLES.
C        LBLK  = NUMBER CF ENTRIES IN EACH BLOCK IN BUFF(MUST BE MULTIPLE
C                CF 4).
C        LBLK4 = 1/4 THE LENGTH OF ONE BLOCK IN BUFF.  THIS DETERMINES THE
C                 NUMBER OF ENTRIES IN A PACKED BLOCK ON MAGNETIC TAPE.
C        LGNBF = LENGTH CF BUFF (IN NUMBER OF ENTRIES).
C        LGNSM = LOG TO THE BASE 2 OF NSAMT.  THIS TELLS THE FFTB SUBROU-
C                TINE HOW MANY ITERATIONS IT WILL HAVE TO PERFORM.
C        NBLK  = NUMBER OF BLOCKS IN BUFF.
C        NSAMT = NUMBER CF TIME SAMPLES TO BE ANALYSED IN EACH TIME SLICE.
C        OVFAC = FACTCR USED BY THE NORMF SUBROUTINE WHEN NORMALIZING THE
C                DISPLAY DATA.  THE HIGHEST INTENSITY POINT TO BE DIS-
C                PLAYED IS SET TO 255*OVFAC AND THE OTHER POINTS ARE NOR-
C                MALIZED LINEARLY ACCORDING TO THIS VALUE.
C
```

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
                TITLE*
         5      ITLIM =     10
 00003          IWID  =    900
 00004          FINTM =    20.0
 00005          IDEL  =      7
 00006          IDELX =      4
 00007          IDELY =      6
 00010          IDELT =    256
 00011          IENDB =  20000
 00012   7      IXMAR =    40
 00013          IFREQ =     90
 00014          IFMAX =    100
 00015          ITMAX =    150
 00016          ISAMF =  20000
 00017          ITIME =    146
 00020          LBLK  =   1000
 00021          LBLK4 = LBLK/4
 00023          LGNSM =      7
 00025          NBLK  =      3
 00026          LGNBF = LBLK*NBLK
 00027          NSAMT =    256
 00031          NSMT2 = NSAMT/2
 00033          OVFAC =     1.7
 00035          EMPFC =       .15
       C
       C        PRINT OUT ASSIGNED VALUES.
       C
 00036    10    CALL FINI
 00037          RETURN
 00040         END
--FORTRAN
```

3.9 FINI

This subroutine is used to calculate the system's dependent parameters on the basis of the current values of the independent parameters. It is generally called when the system is initialized or whenever the value of an independent parameter is changed.

Once the dependent values have been calculated, the subroutine turns off the CRT display, prints out the values of the pertinent variables, and returns.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
         SUBROUTINE FINI
  C
  C            THIS SUBROUTINE IS USED TO CALCULATE DEPENDENT VARIABLES
  C        AFTER THE OPERATOR HAS FINISHED CHOOSING PARAMETERS;  IT THEN
  C        PRINTS OUT ALL THE VALUES.
  C
  C        DELF   = COMMON VARIABLE.  DIFFERENCE FREQUENCY BETWEEN FREQUENCY
  C                 SAMPLES OF THE DATA ARRAY TO BE DISPLAYED.  THIS VARIABLE
  C                 IS ALSO SIMPLY THE FUNDAMENTAL FREQUENCY FOR THE TIME
  C                 PERIOD REPRESENTED BY THE LENGTH OF THE PRESENT TIME SLICE.
  C        DTIME  = COMMON VARIABLE.  LENGTH OF TIME(IN SEC.) BETWEEN DATA
  C                 SAMPLES.
  C        IOVLAP= OVERLAP BETWEEN SUCCESSIVE TIME SLICES(IN NO. OF SAMPLES).
  C        ISAMF  = COMMON VARIABLE.  INTEGER REPRESENTATION OF THE SAMPLING
  C                 FREQUENCY OF THE A TO D CONVERTOR(IN SAMPLES/SEC.).
  C        SAMF   = COMMON VARIABLE.  FLOATING POINT SAMPLING FREQUENCY OF
  C                 A TO D CONVERTOR (IN SAMPLES/SEC.).
  C        TOTFQ  = TOTAL FREQUENCY SPREAD OF DISPLAY.
  C        TOTIM  = TOTAL TIME SHOWN ON DISPLAY (IN SEC.).
  C
           TITLE*
  C
  C            CALCULATE THE DEPENDENT PARAMETERS ON THE BASIS OF THE VALUES
  C        PREVIOUSLY CHOSEN FOR THE INDEPENDENT PARAMETERS.
  C
   200     SAMF   = FLOATF(ISAMF)
00004      DELF   = SAMF/FLOATF(NSAMT)
00010      DTIME  = 1.0/SAMF
00011      TOTFQ  = IFREQ*DELF
00014      TOTIM  = ITIME*IDELT
00016      TOTIM  = TOTIM/SAMF
00020      IOVLAP= NSAMT - IDELT
  C
  C            COMMAND IS FINIS.  PRINT THE VALUES WHICH HAVE BEEN
  C        CHOSEN AND THE EFFECT OF THESE CHOICES ON THE DEPENDENT
  C        VARIABLES.
  C
00022      CALL STOPSCOP
00024      PRINT 250, FINTM, ICBLK, IDEL,  IDELT, IDELX, IDELY,
       1              IENDB, IFREQ, ISAMF, ISAMP, ISAMPB,ITIME,
       2              IFMAX, ITMAX, LGNBF, ITLIM, IXMAR, EMPFC
00051      PRINT 251, LBLK,  LBLK4, LGNSM, NBLK,    NSAMT, NSMT2,
       1              OVFAC, DELF,  TOTIM, TOTFQ, IOVLAP
00067  250 FORMAT      (8H1FINTM =F6.0,8H ICBLK =I6,8H IDEL  =I6,
       1   /8H IDELT =I6,8H IDELX =I6,8H IDELY =I6,8H IENDB =I6,
       2   /8H IFREQ =I6,8H ISAMF =I6,8H ISAMP =I6,8H ISAMPB=I6,
       3   /8H ITIME =I6,8H IFMAX =I6,8H ITMAX =I6,8H LGNBF =I6,
       4   /8H ITLIM =I6,8H IXMAR =I6,8H EMPFC =F6.2)
00067  251 FORMAT      (8H LBLK  =I6,8H LBLK4 =I6,8H LGNSM =I6,
       1   /8H NBLK  =I6,8H NSAMT =I6,8H NSMT2 =I6,8H OVFAC =F6.1,
       2   /8H DELF  =F6.0,8H TOTIM =F6.3,8H TOTFQ =F6.0,
       3    8H OVLAP =I6,9H SAMPLES.)
00067      RETURN
00070    END
--FORTRAN
```

3.10 DISSY (XBUF, YBUF, NUM, ISCOPE, ISCLGN, XMAX, YMAX, IF1)

This is the general data displaying subroutine. It obtains the x and y coordinates of the points to be displayed from successive entries in XBUF and YBUF respectively. If IF1 = 0, successive integers are used as the x coordinates instead of the contents of XBUF. In either case a continuous line is drawn through all the points being displayed.

XMAX and YMAX specify the maximum values allowed for the points to be plotted. The CSL display routines will use these values to determine the size of the display and to calculate the position of each point. Any coordinate which may exceed its corresponding maximum will be truncated to the maximum value.

ISCOPE is the buffer to be used by the CSL display routines in constructing the display commands. ISCLGN specifies the length of this buffer. If a command is every given which causes the display routine to completely fill the buffer, a comment will be typed out on the typewriter by the CSL display routines.

If IF1 = 0, an additional option can be used, namely the production of cursor lines vertically across the display. If IF1 = 0, then the zeroth entry in XBUF will be checked to find out how many cursors are to be plotted, and the successive entries in XBUF will be used as the locations of these cursors along the X axis. Note that if no cursors are to be drawn, a single variable (not an array) containing the value "0.0" may be used for XBUF.

```
CSL FORTRAN OF SEPT 1968,  DATE  9/9/71
            SUBROUTINE DISSY(XBUF,YBUF,NUM,ISCOPE,ISCLGN,XMAX,YMAX,IF1)
       C
       C           THIS SUBROUTINE DISPLAYS THE DATA IN XBUF AND YBUF
       C        AS THE X AND Y COORDINATES, RESPECTIVELY , OF A SERIES OF
       C        NUM POINTS.  THE BUFFER, ISCOPE, IS USED TO CONSTRUCT THE
       C        DISPLAY BUFFER.  PROCESSING BEGINS WITH THE ZEROTH LOCATION
       C        IN XBUF AND YBUF.
       C         IBFRPT= DISPLAY BUFFER POINTER - USED BY DISPLAY ROUTINES AS POINTS
       C                 ARE ADDED TO THE DISPLAY.
       C         IF1    = -1, USE XBUF AS A FLOATING POINT NUMBER ARRAY.
       C                =  0, USE SUCCESSIVE INTEGERS BEGINNING WITH 1, INSTEAD
       C                      OF XBUF.  XBUF CONTAINS A LIST OF POSITION MARKERS
       C                      (IN FLT. PT.) TO BE DISPLAYED ALONG WITH THE DATA.
       C                      XBUF(0) CONTAINS THE NUMBER OF MARKERS.
       C                = +1, USE XBUF AS AN INTEGER NUMBER ARRAY.
       C         ISCLGN= LENGTH OF DISPLAY BUFFER.
       C         ISCOPE= DISPLAY BUFFER TO BE USED BY DISPLAY ROUTINES.
       C         NUM    = NUMBER OF POINTS TO BE DISPLAYED.
       C         XBUF   = X COORDINATE BUFFER.
       C         XMAX   = MAXIMUM X COORDINATE VALUE.
       C         YBUF   = Y COORDINATE BUFFER.
       C         YMAX   = MAXIMUM Y COORDINATE VALUE.
       C
                DIMENSION XBUF(1), YBUF(1), ISCOPE(1)
       C
       C              STOP SCOPE DISPLAY WHILE BUFFER IS BEING LOADED.
       C
                CALL STOPSCOP
00003           IBFRPT = 1
00004           DISPLAY 200 (ISCOPE,ISCLGN,IBFRPT)/XMAX,0.0,YMAX,0.0
00015           DO 100, I=0, (NUM-1)
00022           IF (IF1)      40, 50, 60
00024     40    FTMP = XBUF(I)
00025           GO TO 90
00026     50    FTMP = I
00027           GO TO 90
       --ILLAR
       XX60       LDA      1  XBUF
                  STA         ITMP
       --FORTRAN
00034           FTMP = ITMP
00035     90    DISPLAY/FTMP, YBUF(I)
00040    100    CONTINUE
00041           IF (IF1)     150, 110, 150
       C
       C              IF IF1 = 0, PLOT THE VERTICAL POSITION MARKERS.
       C
00043    110    DISPLAY 201/XMAX, 0.0, YMAX, 0.0
00050           ITMP = XBUF(0)
00054           PRINT 130, ITMP
00061    130    FORMAT (2X,7HITMP = I5)
00061           DO 140, I=1, ITMP
00065           DISPLAY/XBUF(I), 0.0, XBUF(I), YMAX
```

```
      CSL FORTRAN OF SEPT 1968,  DATE  9/9/71
  140    CONTINUE
  150    DISPLAY
  200    FORMAT(*XMAX, XMIN, YMAX, YMIN, AXES, CLINEMODE)
  201    FORMAT (*XMAX, XMIN, YMAX, YMIN, LINEMODE)
  250    RETURN
       END
--FORTRAN
```

Section 4
INPUT DATA PROCESSING ROUTINES

The Input Data Processing package consists of those routines used by the system to obtain data from the A to D converter and to process it to the point where it can be used by the display routines. The process used is somewhat tedious and much improvement remains to be done in the area of automatization of this process.

The subroutine OBTAIN is used to take data as it comes in on the A to D converter and to store it on tape. WAITSIG is used by OBTAIN to wait until the A to D converter input exceeds a certain threshold, at which time it begins recording the data. THR is used to actually check the data for the threshold value.

Since OBTAIN can only gather a maximum of 4 seconds of speech before its buffer fills, the data collection procedure involves playing the analog data tape for short segments of time. In between time, the data is written on to a digital tape. Once several tapes have been filled, THRSPIC is used to determine the number of samples in each block which are above some threshold value and then to print this number out for each block on the tape. On the basis of this printout and a record of what words were recorded on the analog tape when it was being fed into the A to D converter, the desired digital data can be located and marked off.

Once the specific words have been located on the data tapes, the COPYT routine can be used to extract the data from the original tape and record it on a new tape complete with a header block containing the word recorded and the number of blocks used. The editing process is performed under control of a command tape which is read by the command processor.

The COPY commands, which cause the command processor to call COPYT, must contain the block numbers of the beginning and end of the segment to be saved and a six character representation of the data word contained in that segment.

4.1 OBTAIN

This program obtains data from the A to D converter and writes it out on magnetic tape in blocks of 250 words, each word containing four 12-bit samples. The data is loaded into data buffer A before being written on to tape.

The program uses the subroutine WAITSIG to monitor the data coming into the A to D converter. WAITSIG uses the first two 250 word blocks in A to alternately store and test the data coming in. When this data exceeds a certain threshold value, indicated by the common variable ITLIM, WAITSIG returns, thereby signalling to OBTAIN that it should begin loading the data buffer beginning at the third 250 word block.

When the data buffer has been loaded, OBTAIN writes the data out on to magnetic tape unit 4 in blocks of 250 words each. Due to the operation of WAITSIG, the first two blocks may be in reverse temporal order depending on which block was being loaded when it detected the point above threshold. If the first two blocks are in proper temporal order, WAITSIG returns with its flag parameter set to 1. Otherwise the flag is set to 0. Then OBTAIN can use this parameter to determine the order in which to write out the first two blocks of A.

When the program has written out all of the data in A, it returns.

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
          SUBROUTINE OBTAIN
 C
 C
 C             THIS PROGRAM IS USED TO OBTAIN DATA FROM THE A-D
 C          CONVERTOR AND TO WRITE IT ONTO TAPE UNIT 4 IN 250 WORD
 C          BLOCKS OF PACKED DATA CONSISTING OF 4 SAMPLES PER WORD.
 C          THE BUFFER CAN HOLD 20000 WORDS OR 80000 SAMPLES OF DATA
 C          THIS IS 4.0 SECONDS OF DATA AT A 20 KC SAMPLING RATE.
 C               THE PROGRAM USES THE THRESHOLD DETECTING SUBROUTINE,
 C          WAITSIG, TO DETECT THE OCCURANCE OF A SIGNAL GREATER
 C          THAN ITLIM BEFORE THE RECORDING IS BEGUN.
 C
 C          A     = DATA BUFFER.
 C          IENDB = COMMON VARIABLE.  INDICATES LENGTH OF DATA BUFFER
 C                  TO BE USED.
 C          IENDD = INITIAL ADDRESS OF LAST BLOCK IN DATA BUFFER, RELA-
 C                  TIVE TO THE BEGINNING OF A.
 C          IFL   = FLAG USED BY WAITSIG TO INDICATE THE TEMPORAL ORDER
 C                  OF THE FIRST 2 BLOCKS IN A.
 C                = 1 IF IN PROPER ORDER
 C                = 0 IF IN REVERSE ORDER.
 C          ITLIM = COMMON VARIABLE.  SETS THE AMPLITUDE THRESHHOLD
 C                    FOR WAITSIG.
 C          IX    = VARIABLE USED TO INDEX FINAL WRITE TAPE LOOP.  CAN-
 C                    NOT USE INDEX REG. SINCE NUMBERS BECOME TOO LARGE.
 C          IXI   = INITIAL VALUE OF THE INDEX IN THE FINAL DATA WRIT-
 C                  ING LOOP.  ITS ACTUAL VALUE DEPENDS ON THE ORDER OF
 C                  THE FIRST TWO BLOCKS, AS INDICATED BY IFL.
 C          LN    = END ADDRESS OF DATA BUFFER AS DETERMINED BY IENDB.
 C
            TITLE*
        5   CALL FINI
00003       IENDD = IENDB - LBLK4
00004       LN = 0
    --ILLAR
    *
    *       CALCULATE END OF BUFFER AND SAVE IN LN.
    *
             ENA        A             LOAD A REG. WITH ADD. OF BUFFER A.
             ADD        IENDB         ADD CONTENTS OF IENDB.
             STA        LN            STORE IN LN.
    *
    *             CALCULATE THE ADDRESS OF THE THIRD BLOCK IN A AND STORE
    *       IT IN THE ACTIVATE COMMAND FOR THE A TO D CONVERTOR.
    *
             ENA        A             LOAD A REG. WITH ADD. OF BUFFER A.
             ADD        LBLK4         ADD LENGTH OF A PACKED BLOCK.
             ADD        LBLK4         ADD LENGTH AGAIN.
             SAL        ACT           STORE RESULTING ADD. IN ADDRESS FIELD OF
    *                                   LOWER HALF OF LOCATION ACT.
    --FORTRAN
00010   9   CALL WAITSIG(A, ITLIM, IFL)
    --ILLAR
    *
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
   *       WAIT FOR CHANNEL 5 INACTIVE. THEN LOAD CHANNEL CONTROL WORD AND
   *        START LCADING BUFFER A.
   *
   -        EXF      7  51B
            LDA         LN
            SAL         00005B
   -ACT     EXF      5  **          ADD. FIELD HAS BEEN LOADED WITH ADD. OF 3RD BLOCK
   *
   *       WAIT FOR CHANNEL 5 INACTIVE.
   *
   -        EXF      7  51B
   --FORTRAN
   C
   C       WRITE OUT DATA CNTC TAPE IN BLOCKS CONTAINING LBLK/4 WORDS.
   C
00016      IXI = 0
00017      IF (IFL)     30, 30, 35
   C
   C            IF IFL IS ZERC, THE FIRST TWC BLCCKS IN A ARE IN REVERSE
   C       TEMPORAL CRDER.  THEREFORE WRITE THEM ON TO TAPE UNIT 4 SEP-
   C       ARATELY AND BEGIN LCOP WITH THIRD BLOCK.
   C
00021   30 WRITE TAPE 4, (A(J), J=250, 499)
00032      WRITE TAPE 4, (A(J), J=  0, 249)
00043      IXI = 500
   C
   C            TAPE WRITING LCOP.
   C
00044   35 DO 40, IX=IXI, IENDD, LBLK4
00047      JEND = IX + LBLK4 - 1
00051      WRITE TAPE 4, (A(J), J=IX, JEND)
00062   40 CONTINUE
00065      PRINT 100
00070  100 FORMAT (6H CETAI)
00070      GO TO 5
00071     END
--FORTRAN
```

4.2 WAITSIG (BUFF, ITHRES, IFLAG)

The purpose of this subroutine is to read in data from the A to D converter and to return to the calling sequence as soon as it detects a point outside of a certain threshold region above or below the "zero level". The foremost aim of this subroutine is to read all the data coming in from the converter for any sampling rate equal to 20KC or less.

The range of values which are read in from the A to D converter is 0 to 1023, i.e. 10 bits. The value 512 is supposed to represent the zero level while the magnitude of the extremes (0 to 1023) is determined by an external switch on the converter itself (1.5 v., 3 v., 6 v., or 12 v.). Unfortunately, as a practical matter, the converter is usually miscalibrated to some other zero value (when most of the data used in this study was recorded on digital tape the average value was around 493). This does not generally cause too much trouble as long as the error is not too great. It may, however, tend to make WAITSIG too conservative on small threshold values since the actual zero level will get very close to one of the threshold regions if it is too far from the assumed zero value of 512.

A solution to this problem could have been to calculate the average value and to give this value to the subroutine which does the actual threshold search, especially since the routine has the ability to change its "zero level". However, this would have taken more time than was available if every sample is to be tested. It was also unnecessary since the problem can be more easily solved by either extending the value of ITHRES (in which case you effectively only test the threshold in one direction, i.e. on the positive or the negative side but not both) or recalibrating the converter.

The WAITSIG subroutine uses the INTHR and THRES entry points in the THR subroutine to do the actual threshold detecting. INTHR is called with the Q register containing the zero level, 512, and the A register containing the magnitude of the threshold, i.e. ITHRS, in order to get the threshold constants initialized. Then as each block has been read into core, WAITSIG executes a loop to process each word in the block using the THRES entry. Simultaneously WAITSIG loads up the other block with the next batch of data from the converter. The Q register is initially set to zero so that if THRES ever returns with a non-zero value in Q, WAITSIG knows that at least one of the four samples in the word being tested exceeded the threshold regions.

If no sample exceeding the threshold is detected, WAITSIG waits for the second block to be filled. Then it activates the converter so that it will fill the block which was just checked and while this is being done, it checks the block which was just filled.

Eventually THRES will return with the Q register not equal to zero, indicating that a sample exceeding the threshold has been found. In this case a transfer out of the checking loop for that particular block will be made. The transfer is such that IFL (which was originally set to zero) will be set to 1 if the sample detected occurred in the first block and remain unchanged if it occurred in the second block. Thus IFL will indicate the temporal order of the blocks (remembering that while WAITSIG is checking one block, it is loading the other block with data coming after it in time).

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
      SUBROUTINE WAITSIG(BUFF, ITHRES, IFLAG)
C
C           THIS SUBROUTINE READS IN DATA FROM THE A TO D CONVERTOR
C     AND STORES IT ALTERNATELY IN ONE OF TWO 250 WORD BLOCKS.
C     SIMULTANEOUSLY, IT CHECKS THE DATA IN THE INACTIVE BLOCK TO
C     SEE IF ANY OF ITS POINTS ARE ABOVE OR BELOW 512 BY THE AMOUNT
C     SPECIFIED BY ITHRES.  IF THIS OCCURS THE SUBROUTINE RETURNS
C     WITH IFLAG SET TO INDICATE THE LAST BUFFER FILLED.
C
C     BUFF   = BUFFER USED TO HOLD A TO D CONVERTOR DATA.  IT IS
C              BROKEN UP INTO 2 BLOCKS, EACH 250 WORDS LONG.
C     IFLAG  = OUTPUT PARAMETER FLAG WHICH INDICATES IF THE 2 BUFFER
C              BLOCKS USED IN BUFF ARE IN PROPER TEMPORAL ORDER
C              (IFLAG=1) OR REVERSE TEMPORAL ORDER(IFLAG=0).
C     ITHRES = INTEGER THRESHOLD.  RANGES FROM 0 TO 512 AND REP-
C              RESENTS THE MAGNITUDE OF THE THRESHOLD ABOVE AND
C              BELOW THE A TO D CONVERTORS ZERO VALUE OF 512.
C     LAST1, = CONTAINS THE LAST ADDRESS PLUS 1 OF THE FIRST
C      LAST2   AND SECOND BLOCKS IN BUFF RESPECTIVELY.  THESE
C              NUMBERS MUST BE LOADED INTO THE LOWER HALF OF THE
C              CHANNEL CONTROL WORD FOR CHANNEL 5(AT LOCATION 5)
C              WHENEVER THE A TO D CONVERTOR IS ACTIVATED TO LOAD
C              THE CORRESPONDING BLOCK.  THIS LETS THE CHANNEL
C              KNOW WHERE TO STOP.
C
      DIMENSION BUFF(500)
      IFLAG = 0
--ILLAR
       EXT          INTHR,THRES
*
*     ADJUST  THRESHOLD VALUE IN THRES SUBROUTINE.
*
       LDQ          =512
       LDA          ITHRES
-      SLJ    4     INTHR
*
*     INITIALIZE Q REGISTER AND INDEX REGISTERS 1, 2, AND 3.
*
       ENI    1  0B
       ENI    2  0B
       ENI    3  0B
       ENQ       0B
*
*     CALCULATE END OF BUFFER BLOCKS.
*
       ENA          BUFF
       INA          372B        ADD LENGTH OF ONE BLOCK (250 DECIMAL).
       STA          LAST1       SAVE ADDRESS OF END OF FIRST BLOCK + 1 IN LAST1
       SAL          BGPB1       LOAD STARTING ADDRESS OF 2ND BLOCK
       SAU          LCOP2          INTO EXF AND LOAD INSTRUCTIONS.
       INA          372B        ADD LENGTH OF ONE BLOCK (250 DECIMAL).
       STA          LAST2       SAVE ADDRESS OF END OF 2ND BLOCK + 1 IN LAST2.
*
```

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
 *        WAIT FOR CHANNEL 5 INACTIVE.  THIS INSTRUCTION WILL LOOP UNTIL
 *         CHANNEL 5, THE CHANNEL CONTAINING THE A TO D CONVERTOR, IS
 *         INACTIVE AND READY TO ACCEPT A NEW COMMAND.
 *
 *
 -         EXF      7  51B
 *
 *        SELECT CONVERTOR.  THIS INSTRUCTION SELECTS THE A TO D
 *         CONVERTOR FOR INPUT CHANNEL 5 AND ESTABLISHES INITIAL
 *         OPERATING CONDITIONS WITHIN THE CONVERTOR.
 *
           EXF         50000B
 *
 *        SENSE CONVERTOR READY.  THIS INSTRUCTION WILL LOOP UNTIL THE
 *         CONVERTOR IS READY, IE. NO ACTIVITY IS TAKING PLACE.
 *
 -         EXF      7  50000B
 *
 *        LOAD END ADDRESS OF 1ST BUFFER INTO CONTROL WORD AND
 *         ACTIVATE 1ST BUFFER.
 *
           LDA         LAST1
           SAL         00005B
 -         EXF      5  BUFF
 *
 *        WAIT FOR CHANNEL 5 INACTIVE.  THIS INSTRUCTION WILL LOOP UNTIL
 *         CHANNEL 5, THE CHANNEL CONTAINING THE A TO D CONVERTOR, IS
 *         INACTIVE AND READY TO ACCEPT A NEW COMMAND.
 *
 -WAIT1    EXF      7  51B
 *
 *        LOAD END ADDRESS OF THE SECOND BLOCK INTO THE CONTROL WORD.
 *
           LDA         LAST2
           SAL         00005B
 *
 *        BEGN PROCESSING FOR THRESHOLD IN FIRST BLOCK AND
 *         ACTIVATE SECOND BLOCK.
 *
 -BGPB1    EXF      5  **          ADD. FIELD IS LOADED WITH BEGIN. ADDR. OF 2ND BLOCK
 LOOP1     LDA      3  BUFF
           INI      3  1B
 -         SLJ      4  THRES
 *         ISK      3  371B        IF LOOP IS FINISHED, SKIP NEXT INSTRUCTION.
           QJP      0  LOOP1       IF Q=0, THRES DID NOT FIND A POINT, THUS CONTINUE.
 *
 *        IF CONTROL ARRIVES AT THIS POINT, THE LOOP IS EITHER FINISHED,
 *         OR A POINT ABOVE THRESHOLD HAS BEEN FOUND BY THRES:  THEREFORE
 *         CHECK THE Q REGISTER AND IF IT IS NOT EQUAL TO ZERO, THEN GO
 *         TO THE END AT 100.  OTHERWISE GO TO WAIT2 AND WAIT FOR THE
 *         SECOND BLOCK TO BE FINISHED FILLING.
 *
           QJP      1  XX100
```

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
 *
 *         WAIT FOR CHANNEL 5 INACTIVE.
 *
 -WAIT2    EXF     7  51B
 *
 *         LOAD END ADDRESS OF FIRST BLOCK INTO CONTROL WORD.
 *
           LDA        LAST1
           SAL        00005B
 *
 *         BEGIN PROCESSING FOR THRESHOLD IN SECOND BLOCK AND
 *          ACTIVATE FIRST BLOCK.
 *
 -BGPB2    EXF     5  BUFF
  LOOP2    LDA     3  **           ADD. FIELD IS LOADED WITH BEGIN. ADDR. OF 2ND BL(
           INI     3  1B
 -         SLJ     4  THRES
 +         ISK     3  371B         IF LOOP IS FINISHED, SKIP NEXT INSTRUCTION.
           QJP     0  LOOP2        IF Q=0, THRES DID NOT FIND A POINT; THUS CONTINUE
 *
 *         IF CONTROL ARRIVES AT THIS POINT, THE LOOP IS EITHER FINISHED
 *          OR A POINT ABOVE THRESHOLD HAS BEEN FOUND BY THRES.  THEREFORE
 *          CHECK THE Q REGISTER AND IF IT IS NOT EQUAL TO ZERO, THEN GO
 *          TO THE END AT 101.  OTHERWISE GO TO WAIT1 AND WAIT FOR THE
 *          FIRST BLOCK TO BE FINISHED FILLING.
 *
           QJP     1  XX101
           SLJ     0  WAIT1
 *
 *         DEFINE VARIABLES
 *
  LAST1    BSS        1
  LAST2    BSS        1
  --FORTRAN
00005   100    IFLAG = 1
00006   101    RETURN
00007          END
--ILLAR
```

4.3 THR

This is a threshold detecting program which was written in ILLAR assembly language in order to make it as fast as possible. When called, it checks the word in the A register, which contains four 10-bit samples, to see if any of them are outside the threshold region. If so, it increments index register #2 by 1 for each such sample and sets the Q register to 1. The THRES entry point uses index register #1 to count the number of samples which have been processed, and the calling sequence should be sure that it is set to zero before THRES is entered. It is automatically reset to zero before returning.

In order to set up the threshold limits it is necessary to call INTHR with the average value of the data stored in the Q register and the threshold magnitude stored in the A register. INTHR will then calculate the threshold region as the average ± the threshold.

The routine works by first seeing if the point is greater or less than 512 (this is done by looking at the highest order bit).

Then it compares the number with either the upper or lower threshold boundary depending on which side of 512 the number is on. Note that this implies that for proper operation the upper and lower boundaries should be on opposite sides of the 512 level boundary. If this is <u>not</u> true, then any points between 512 and the nearest boundary will be considered within the threshold region and will be ignored (note Figure 4.3.1).

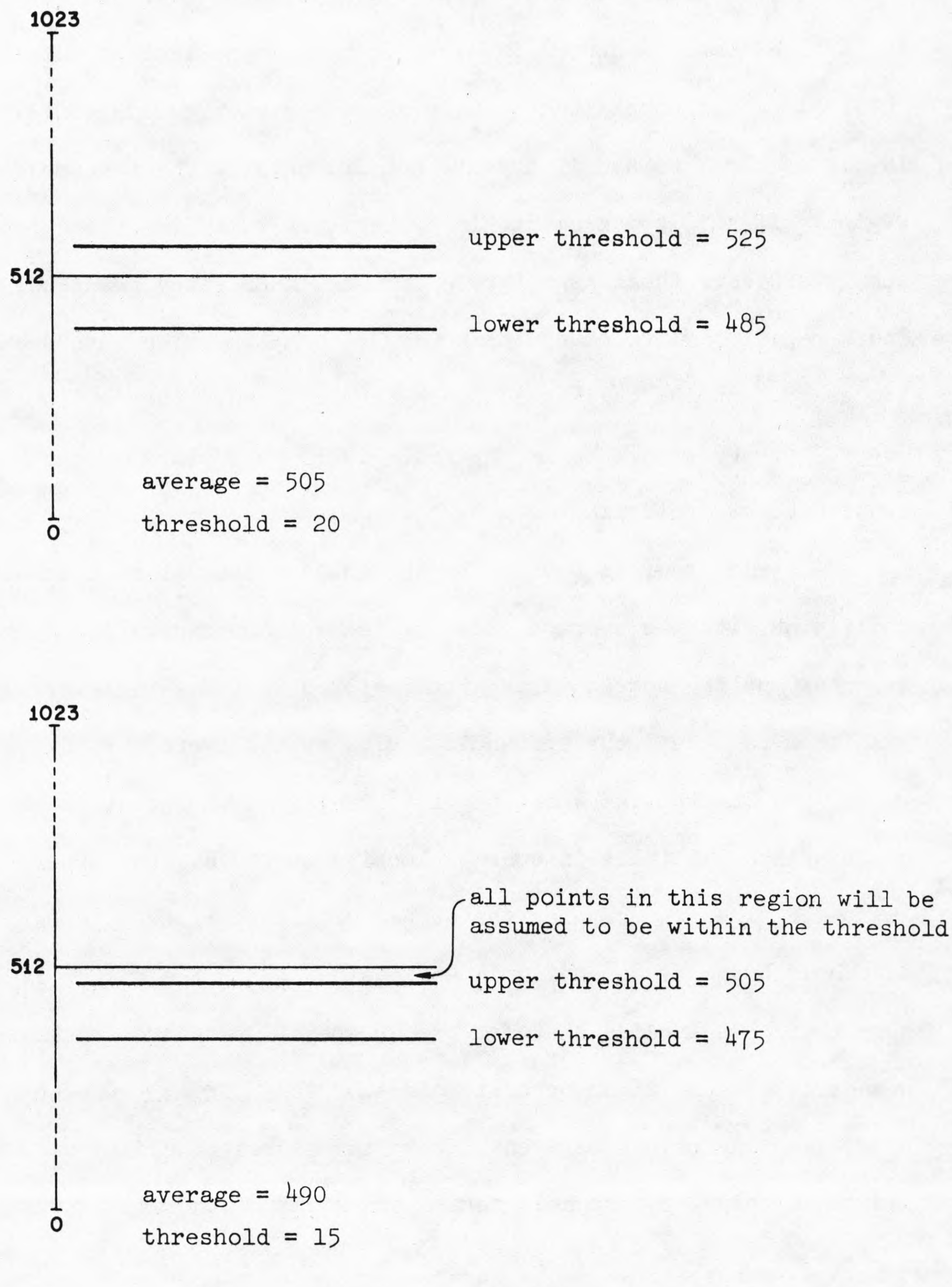


Figure 4.3.1 Interaction between average and threshold values in THR

```
                                                IDENT      THR
00000+         75  0  77777      INTHR          ENTRY
                                 *                   THE INITIALIZING PART OF TH
                                 *              SUBROUTINE IS CALLED WITH THE TH
                                 *              FROM 0 TO 512), LOADED IN THE A
                                 *              VALUE OF THE DATA (VARYING FROM
                                 *              Q REGISTER.
                                 *                   IT SHOULD BE NOTED THAT FOR
                                 *              ATION, THE QUANTIES IAV + ITH AN
                                 *              OPPOSITE SIDES OF THE VALUE 512.
                                 *              THEN POINTS BETWEEN 512 AND THE
                                 *              WILL NOT BE CONSIDERED TO BE OVE
                                 *              MOST CASES THIS WILL ONLY CAUSE
                                 *              SINCE THE PERCENTAGE OF POINTS I
                                 *              SMALL.
                                 *
               21  0  00011+                    STQ        IAV
00001+         20  0  00006+                    STA        ITH
               14  0  00011+                    ADD        IAV
00002+         05  0  00046                     ALS        46B
               20  0  00007+                    STA        ITHN          ITHN = LOW
00003+         12  0  00011+                    LDA        IAV
               15  0  00006+                    SUB        ITH
00004+         05  0  00046                     ALS        46B
               20  0  00010+                    STA        ITHP          ITHP = UPP
00005+         75  0  00000+                    SLJ     0  INTHR
00006+                00001      ITH            BSS        1B
00007+                00001      ITHN           BSS        1B
00010+                00001      ITHP           BSS        1B
00011+                00001      IAV            BSS        1B
00012+         75  0  77777      THRES          ENTRY
                                 *
                                 *                   THIS PART OF THE THRESHOLD
                                 *              EXAMINES EACH OF 4 SAMPLES CONTA
                                 *              WHEN THE SUBROUTINE IS CALLED.
                                 *              ABOVE THRESHOLD, THE SUBROUTINE
                                 *              A 1 AND INCREMENTS INDEX REGISTE
                                 *              OVER THE THRESHOLD.  WHEN THE SU
                                 *              REGISTER WILL CONTAIN THE ORIGIN
                                 *              PLACES TO THE RIGHT, AND INDEX R
                                 *              NONE OF THE 4 SAMPLES IS ABOVE T
                                 *              WILL RETURN WITH THE Q REGISTER
                                 *                   IT SHOULD BE NOTED THAT NOR
                                 *              REQUIRES THE CALLING ROUTINE TO
                                 *              1 IS SET TO ZERO BEFORE THE FIRS
                                 *
                                 *
                                 *                   SHIFT THE A REGISTER LEFT 2
                                 *              ORDER DATA BIT WILL BE IN THE SI
                                 *
               05  0  00002                     ALS        2B
00013+         22  3  00017+     TEST           AJP     3  NEG           IF THE HIG
               15  0  00010+                    SUB        ITHP          ELSE TEST
```

```
                              THR             6/28/71
                              *
                              *                     IF THE POINT IS BELOW THE U
                              *               CONTINUE BY SHIFTING LEFT 12 POS
                              *               REGISTER 1 TO SEE IF 4 SAMPLES H
                              *
00014+       22 3  00023+                     AJP     3  SUCC
             05 0  00014                      ALS        14B
00015+       54 1  00003      RETL            ISK     1  3B
             75 0  00013+                     SLJ     0  TEST
00016+       75 0  00012+                     SLJ     0  THRES
00017+       15 0  00007+     NEG             SUB        ITHN
             22 2  00023+                     AJP     2  SUCC
                              *
                              *                     IF THE POINT IS ABOVE THE L
                              *               CONTINUE BY SHIFTING LEFT 12 POS
                              *               REGISTER 1 TO SEE IF 4 SAMPLES H
                              *
00020+       05 0  00014                      ALS        14B
00021+       54 1  00003      *               ISK     1  3B
             75 0  00013+                     SLJ     0  TEST
00022+       75 0  00012+                     SLJ     0  THRES
                              *
                              *                     IF THE POINT EXCEEDS THE TH
                              *               Q REGISTER WITH 1 AND INCREMENT
                              *
00023+       04 0  00001      SUCC            ENQ        1B
             51 2  00001                      INI     2  1B
00024+       05 0  00014                      ALS        14B
             75 0  00015+                     SLJ     0  RETL
                                              END
```

4.4 THRSPIC (I1, I2)

This subroutine will scan a packed data tape written in blocked format from block I1 through block I2. During the scanning process it counts the number of data points in each block which lie outside the range of the average value ± ITLIM and prints out the result for each block. If the block is a header (indicated by an initial word containing all 1's) the subroutine prints out a header comment containing the word represented by the data and the number of blocks used to store the word.

If the block is not a header, the THRSPIC subroutine unpacks the packed data and calculates its average value which is then stored as an integer in ISUM. Next, using this value and the threshold value ITLIM, it calls the initializing entry point for the threshold detecting subroutine and sets up the necessary threshold parameters. Then it enters the threshold calculating loop in which repeated calls are made to THRES. Note that the operation of THRES is such that for each point above threshold, index register 2 is incremented by 1 and furthermore, that in CSL FORTRAN index register 2 is equivalent to the variable J. Thus when the whole block has been searched the subroutine prints out the contents of "J" and repeats the loop.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
          SUBROUTINE THRSPIC(I1, I2)
      C
      C          THIS SUBROUTINE SCANS A DATA TAPE WRITTEN IN PACKED
      C     BLOCKS AND PRINTS OUT, FOR EACH BLOCK, THE BLOCK NUMBER
      C     AND HOW MANY DATA POINTS IN IT EXCEED A SPECIFIED THRESH-
      C     HOLD.  IF THE BLOCK IS A HEADER BLOCK, THE PROGRAM PRINTS
      C     OUT THIS FACT AND ALSO THE WORD REPRESENTED AND THE NUMBER
      C     OF DATA BLOCKS USED FOR THAT WORD.
      C          THE ROUTINE BEGINS WITH BLOCK I1 ON THE DATA TAPE
      C     AND ENDS WITH BLOCK I2.  FOR EACH BLOCK IT USES UNPACK TO
      C     SEPARATE THE DATA POINTS AND THEN CALCULATES THE AVERAGE
      C     VALUE FOR THE DATA.  USING THIS AVERAGE VALUE AND THE
      C     SUBROUTINE THRES, IT THEN CALCULATES THE NUMBER OF POINTS
      C     OUTSIDE THE REGION = AVERAGE VALUE PLUS OR MINUS ITLIM.
      C
      C     I1    = BLOCK NUMBER OF FIRST BLOCK TO BE SCANNED.
      C     I2    = BLOCK NUMBER OF LAST BLOCK TO BE SCANNED.
      C     IBUF  = TEMPORARY BUFFER TO HOLD PACKED DATA.
      C     ICBLK = BLOCK NUMBER OF CURRENT BLOCK WHICH IS STORED IN THE FIRST
      C             BLOCK OF BUFF.
      C     INTHR = INITIALIZING ENTRY POINT OF THRESHOLD SUBROUTINE.
      C     ISAMP = DATA POINTER.  POINTS TO BEGINNING OF NEXT BLOCK TO
      C             BE SCANNED.
      C     ITLIM = THRESHOLD LIMIT TO CHECK FOR.
      C     LBLK  = LENGTH OF A DATA BLOCK IN NUMBER OF SAMPLES.
      C     LBLK4 = LBLK/4 = LENGTH OF A PACKED DATA BLOCK IN NUMBER OF WORDS.
      C     MASK  = CODE INDICATING THAT THE CURRENT BLOCK JUST SCANNED  IS
      C             A HEADER BLOCK.
      C     OUT   = TEMPORARY BUFFER USED TO HOLD UNPACKED FLOATING POINT
      C             DATA.
      C     SUM   = SUMMATION OF THE VALUES OF THE SAMPLE POINTS IN THE
      C             CURRENT BLOCK.  USED TO CALCULATE THE AVERAGE VALUE.
      C     THRES = PROCESSING ENTRY POINT OF THRESHOLD SUBROUTINE.
      C
            TITLE*
            DATA (MASK = 77777777777777777B)
            DIMENSION IBUF(250), OUT(1000)
            EQUIVALENCE (IBUF(1), A(3002)), (OUT(1), A(4002))
      C
      C          CALL ADJUS2 TO MOVE TAPE TO FIRST BLOCK TO BE READ.
      C
         10 ISAMP = I1*LBLK
00003       CALL ADJUS2(0,0)
00006       DO 200, K=I1, I2
00012       READ TAPE 4, (IBUF(L), L=1, LBLK4)
00023       ISAMP = ISAMP + LBLK
00024       ICBLK = ICBLK + 1
      C
      C          IF THE FIRST WORD OF THE BUFFER INDICATES THAT THE
      C     BLOCK IS A HEADER, PRINT OUT THE HEADER MESSAGE.
      C
00026       IF (IBUF(1) - MASK)    90, 50, 90
00034    50 PRINT 60, K, IBUF(3), IBUF(2)
```

```
        CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
00051      60      FORMAT (13H HEADER BLOCKI4,6H  FOR A6,3H.  I10,
              1    8H BLOCKS,)
00051              GO TO 200
          C
          C              IF THE BLOCK IS NOT A HEADER BLOCK, CALCULATE THE
          C        AVERAGE VALUE OF THE DATA IN THE BLOCK.
          C
00052      90      CALL UNPACK(IBUF, OUT, 1, 1, LBLK4)
00060              SUM = 0.0
00061              DO 95, L=1, LBLK
00065     95       SUM = SUM + OUT(L)
00070              ISUM = SUM
00071              ISUM = ISUM/LBLK
          C
          C              NEXT CALCULATE THE NUMBER OF POINTS ABOVE THRESHOLD
          C        USING THE PREVIOUSLY CALCULATED AVERAGE VALUE AND ITLIM AS
          C        THE THRESHOLD VALUE.
          C
          --ILLAR
                    EXT          INTHR,THRES
                    LDQ          ISUM
                    LDA          ITLIM
          -         SLJ     4    INTHR
                    ENI     1    0E
          --FORTRAN
00077              J = 0
00100              DO 100, L=1, LBLK4
          --ILLAR
                    LDA     4    IBUF
          -         SLJ     4    THRES
          --FORTRAN
00107     100      CONTINUE
          C
          C              PRINT OUT THE BLOCK NUMBER AND NUMBER OF POINTS ABOVE
          C        THRESHOLD.
          C
00110              PRINT 150, K, J, ISUM
00120     150      FORMAT (8X,5HBLOCKI4,5H HAS I4,8H POINTS.70X,I4)
00120     200      CONTINUE
00121                 END
--FORTRAN
```

4.5 COPYT (IFBLK, ILBLK, SYMBL)

This subroutine is used to edit the data on one tape by extracting only the useful information on it and copying this information on to a second tape. The subroutine reads data from tape unit 4 and writes on to tape unit 3.

In operation, the subroutine begins copying with block number IFBLK. However, before doing this it generates a header block containing a header code (a word containing all 1's), the 6 character name, SYMBL, and the number of blocks which will be copied. These items occupy words 1, 2, and 3 of the header while the rest of it contains garbage.

Once the header block has been written out, COPYT copies the designated blocks out one at a time on to unit 3. During this process ISAMP and LOCOP are continually updated, the latter quantity being a system variable which indicates the total number of samples written on to unit 3 so far. After the copying is complete, LOCOP is printed on the printer, ICBLK is updated as if the program had been reading data into the data buffer, and the subroutine returns.

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
      SUBROUTINE COPYT(IFBLK, ILBLK, SYMBL)
 C
 C         THIS SUBROUTINE IS USED TO COPY DATA FROM THE DATA
 C     TAPE ON UNIT 4 TO A NEW DATA TAPE ON UNIT 3.  THIS ALLOWS
 C     THE OPERATOR TO EXTRACT FROM A TAPE ONLY THOSE PORTIONS
 C     WHICH ARE ACTUALLY USEFUL.
 C         THE SUBROUTINE FIRST PRODUCES A HEADER BLOCK FOR THE
 C     GIVEN BATCH OF DATA.  THEN IT COPIES DATA BEGINNING WITH THE
 C     BLOCK NUMBER SPECIFIED BY IFBLK AND CONTINUING TO THE BLOCK
 C     NUMBER SPECIFIED BY ILBLK  THE VARIABLE LCCOP IS INCREASED
 C     BY THE NUMBER OF BLOCKS WRITTEN ON TO TAPE UNIT 3.  AFTER
 C     PRINTING OUT THE FINAL VALUE OF LCCOP, THE SUBROUTINE RETURNS.
 C         IT IS IMPORTANT TO NOTE THAT THE DATA ON BOTH TAPES IS
 C     WRITTEN IN THE PACKED FORMAT, 4 SAMPLES PER WORD AND THAT THE
 C     VALUES OF LCCOP AND ISAMP REFER TO THE NUMBER OF SAMPLES, NOT
 C     THE NUMBER OF WORDS.
 C
 C     IBUF  = TEMPORARY STORAGE BUFFER USED DURING COPYING.
 C     IFBLK = FIRST BLOCK WHICH IS TO BE COPIED.
 C     ILBLK = LAST BLOCK WHICH IS TO BE COPIED.
 C     ILOC  = POSITION OF FIRST SAMPLE TO BE COPIED.
 C     LBLK  = LENGTH OF A DATA BLOCK.
 C     LBLK4 = LENGTH OF A DATA BLOCK OF PACKED DATA.
 C           = LBLK/4.
 C     LGNBF = TOTAL LENGTH OF SAMPLE BUFFER, BUFF.
 C           = NBLK * LBLK.
 C     LOCOP = THE TOTAL NUMBER OF SAMPLES WHICH HAVE BEEN WRITTEN ON
 C             TAPE UNIT 3.
 C     NUMB  = NUMBER OF DATA BLOCKS WHICH WILL BE COPIED.
 C     SYMBL = CHARACTER REPRESENTATION (A6) OF THE SPEECH WORD
 C             BEING COPIED.
 C
       TITLE*
       DIMENSION IBUF(250)
       EQUIVALENCE (IBUF(1), A(3002)),(SYM, A(3004))
       DATA (MASK=77777777777777777B)
 C
   10  ISAMP = IFBLK*LBLK
00003  CALL ADJUS2(0,0)
 C
 C         SET UP AND WRITE OUT THE HEADER BLOCK CONTAINING THE
 C     HEADER BLOCK INDICATOR,IBUF(1), THE NUMBER OF BLOCKS OF DATA,
 C     IBUF(2), AND THE WORD REPRESENTED BY THE DATA, SYMB.
 C
00006  IBUF(1) = MASK
00011  IBUF(2) = ILBLK - IFBLK + 1
00016  SYM     = SYMBL
00020  WRITE TAPE 3, (IBUF(J), J=1, LBLK4)
00031  LOCOP = LOCOP + LBLK
 C
 C         NEXT COPY OUT THE REQUIRED NUMBER OF DATA BLOCKS,
 C     UPDATING ISAMP AND LOCOP AS YOU GO.
 C
```

```
      CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
  100     DO 199, I=IFBLK, ILBLK
          READ TAPE 4,   (IBUF(J), J=1, LBLK4)
          WRITE TAPE 3,  (IBUF(J), J=1, LBLK4)
          ISAMP = ISAMP + LBLK
          LOCOP = LOCOP + LBLK
  199     CONTINUE
C
C              FINALLY, PRINT OUT THE RESULTING NEW VALUE OF
C         LOCOP.
C
          PRINT 200, LOCOP
  200     FORMAT (9H LOCOP = I8)
C
C              AT THE END OF A COPY, RESET ICBLK TO A POSITION NBLK BLOCKS
C         IN FRONT OF THE CURRENT POSITION OF THE TAPE UNITS READ HEAD.
C         THIS KEEPS ADJUS2 FROM GETTING CONFUSED ABOUT THE STATUS OF THE
C         TAPE UNIT.
C
          ICBLK = ISAMP/LBLK - NBLK
C
C              IF THE PAPER TAPE READER IS BEING USED AS THE INPUT
C         DEVICE, CONTINUE TO READ COMMANDS IN THE FORM GIVEN IN FORMAT
C         STATEMENT 251 UNTIL A NEGATIVE VALUE OF IFBLK IS FOUND.
C
          IF (IUNIT - 54)      300, 250, 300
  250     READ INPUT TAPE 54. 251, IFBLK, ILBLK, FNAME
  251     FORMAT (I3,1X,I3,1X,A6)
          IF (IFBLK)      300, 260, 300
  260     PRINT 261, IFBLK, ILBLK, FNAME
  261     FORMAT (8H COPY = I3,1H,I3,1H,A6)
          GO TO 10
  300     RETURN
        END
--FORTRAN
```

Section 5

DATA MANIPULATION ROUTINES

This set of subroutines is used to perform the operations necessary to keep the data buffer, BUFF, full and ISAMPB adjusted properly. ADJUS2 checks to see if there are IWIDE samples in the buffer beyond the current ISAMPB position. If not, it reads along the data tape until it can load the proper data into BUFF. Then it recalculates the value of ISAMPB and returns.

BLOCKRD is used by ADJUS2 to read the data tape from tape unit 4. It was written so that all of the tape manipulations used in ADJUS2 would be centered in one module.

UNPACK is the subroutine used by BLOCKRD to unpack the packed data after it is read in from the tape and to convert the samples from integer to floating point.

The data tape itself is made up of a sequence of data blocks which are stored in a packed format. This essentially means that the data is stored as CDC 1604, 48-bit integer numbers where each number is made up of 4, 12-bit integer samples. Each block contains a total of LBLK packed data samples or LBLK/4 words. LBLK is a system constant which has always been 1000. However, the data manipulation programs have been written such that this number can be varied without reprogramming. A typical full length data tape can contain around 900 data blocks or about 900,000 data samples.

The data tapes used to produce the final speech displays have been divided up by previous editing into individual speech words, each beginning with a special header block. The first word of the header block contains all 1's as an identification. The second word contains the number of blocks in the speech word and the third word contains the symbolic representation of the recorded word. The header blocks are used for several purposes: to identify the speech words when the contents of the data tape are being printed out by THRSPIC, to indicate the length of the data so that ITIME can be calculated before the speech data is processed and to search the tape for particular speech words instead of an address.

5.1 ADJUS2 (IWIDE, NFLAG)

The purpose of this subroutine is to ensure that the data buffer, BUFF, contains the data which will be needed by the processing routines. It can also be used to position the data tape in preparation for some type of direct processing on it. IWIDE is used to indicate to the subroutine how many samples beyond the current position of ISAMP will be needed. NFLAG designates the type of processing option to be used by ADJUS2.

ADJUS2 first calculates new values for ISAMPB, the data pointer location within the data buffer; ITBLK, the block number of the next block to be read from the data tape; and ISBLK, the block number of the data block currently containing ISAMP, on the basis of the latest values of the data pointer, ISAMP; the block number of the initial block appearing in BUFF, and the various buffer constants pertaining to length, block size, etc.

If NFLAG = 0, ADJUS2 then simply moves the tape so that the next block which is read after the move will contain ISAMP. This option is especially useful during data tape editing. To accomplish this, the block number of the block containing ISAMP is checked for 0, since in this case a simple rewind can be performed. If it is not zero, the subroutine calculates the number of backspaces necessary to position the tape and then performs them[1]. If this number is negative, the subroutine skips forward. In either case, after the tape has been removed, new values for ICBLK and ISAMP are calculated before returning. If the number of spaces is zero, no action is performed and ADJUS2 returns immediately.

---

1 The use of backspaces instead of forward spaces came about from the fact that the backspacing subroutine existed before the forward spacing routine, i.e. BLOCKRD.

If NFLAG does not equal zero, the subroutine must check the buffer to see if the desired data is within it and if not, the buffer will be reloaded. If the buffer must be reloaded (due to either overflow or underflow) and NFLAG = +1, the buffer will have to be reloaded in such a way as to cause ISAMPB to be in the middle of the buffer. This option is useful in those cases where forward and backward movements along the data are equally likely. If NFLAG = -1, the buffer will be reloaded so that ISAMPB will be in the first block of the buffer. This is most useful in those cases where only forward motion along the data tape is contemplated. The only difference between the two as far as operations are concerned is the value assigned to NCBLK, the new initial block number which will replace the current value of ICBLK if the tape must be moved.

Once the value of NCBLK has been calculated, the subroutine, using the values of ISAMPB and IWIDE, checks to see if all of the data is within the length of BUFF. If there is an overflow or underflow, the tape will be spaced to a position where it can begin to read the proper data.

At this point, there is a slight non-uniformity in the handling of overflow and underflow. In an overflow condition, ADJUS2 will check to see if any of the data in the buffer is useful and if so, will shift it to the front of the buffer before reading in new data to fill up the end of the buffer. In an underflow condition, the tape is simply backed up to a position from which it can completely read in the whole buffer. This non-uniformity is due, in part at least, to the fact that our records cannot be read backwards by the tape units.

Once the tape unit has been properly positioned, the correct number of blocks are read from the tape and loaded into their proper positions in BUFF after having been unpacked and converted to floating point. Then the updated values of ICBLK, and ISAMP are calculated and the subroutine returns.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
      SUBROUTINE ADJUS2(IWIDE, NFLAG)
C
C         THIS SUBROUTINE CHECKS THE POSITION OF ISAMP TO SEE IF THE
C     DATA TO BE UTILIZED IS STILL COMPLETELY WITHIN THE SAMPLE BUFFER.  IF
C     AN OVERFLOW OR UNDERFLOW OCCURS, THE TAPE IS REPOSITIONED AND THEN IF
C     NFLAG IS NON-ZERO, THE SAMPLE BUFFER IS RELOADED.  WHEN NFLAG = +1,
C     BUFF WILL BE REPOSITIONED SO THAT ISAMP IS IN THE MIDDLE, AND WHEN
C     NFLAG = -1, IT WILL BE REPOSITIONED SO THAT ISAMP IS IN THE LEFTMOST
C     BLOCK.  IF NFLAG = 0, ADJUS2 POSITIONS THE TAPE SO THAT ISAMP IS IN
C     THE NEXT BLOCK TO BE READ FROM THE TAPE REGARDLESS OF WHETHER OR NOT
C     THERE IS AN OVERFLOW OR UNDERFLOW CONDITION.
C
C     BUFF  = THE SAMPLE BUFFER.  THIS CONSISTS OF SEVERAL EQUAL-LENGTH
C             BLOCKS.
C     ISAMP = POSITION OF THE SAMPLE POINTER WITH REFERENCE TO DATA TAPE.
C     ISAMPB= POSITION OF THE SAMPLE POINTER WITH REFERENCE TO BUFF.
C     ISBLK = BLOCK NUMBER OF BLOCK CURRENTLY CONTAINING ISAMP.
C     ITBLK = BLOCK NUMBER OF BLOCK WHICH IS CURRENTLY THE NEXT TO
C             BE READ FROM TAPE.
C     IWIDE = WIDTH OF THE WINDOW TO BE DISPLAYED.
C     ICBLK = BLOCK NUMBER OF THE CURRENT BLOCK WHICH IS STORED IN THE
C             FIRST BLOCK OF BUFF.
C     NBLK  = THE NUMBER OF BLOCKS IN BUFF
C     LBLK  = THE LENGTH OF EACH BLOCK IN BUFF
C     LGNBF = TOTAL LENGTH OF THE SAMPLE BUFFER, BUFF.
C           = NBLK * LBLK.
C     NFLAG =  0 POSITION TAPE SO READ HEAD WILL READ BLOCK CONTAINING
C                ISAMP NEXT.
C           = -1 IF OVERFLOW OR UNDERFLOW, POSITION ISAMP IN LEFTMOST BLOCK.
C           = +1 IF OVERFLOW OR UNDERFLOW, POSITION ISAMP IN MIDDLE BLOCK.
C
      TITLE*
   10 ISAMPB = ISAMP - ICBLK*LBLK
      ITBLK = ICBLK + NBLK
      ISBLK = ISAMP/LBLK
C
C         CHECK NFLAG FOR ADJUS2 OPTION TO BE USED.
C
      IF (NFLAG)    95, 20, 90
C
C         IF NFLAG = 0, MOVE THE TAPE SO THAT ISAMP APPEARS IN THE FIRST
C     BLOCK ON TAPE AFTER READ HEAD AND ADJUST ICBLK AND ISAMPB
C     ACCORDINGLY.  NOTE THAT IF NCBLK IS NEGATIVE, THE TAPE WILL
C     UNLOAD.
C
   20 ICBLK = ISBLK - NBLK
      ISAMPB = ISAMP - ISBLK*LBLK + LGNBF
C
C         CHECK ISBLK FOR ZERO.
C
      IF (ISBLK)    25, 21, 25
C
C         IF ISBLK = 0, REWIND TAPE INSTEAD OF BACKSPACING.
```

```
        CSL FORTRAN OF SEPT 1968,   DATE   6/28/71
         C
00024     21    REWIND 4
00026           GO TO 150
         C
         C            IF ISBLK IS NOT ZERO, CALCULATE THE NUMBER OF BACKSPACES
         C      NEEDED.
         C
00027     25    NBKSP = ITBLK - ISBLK
00030           IF (NBKSP)    40, 150, 30
00033     30    CALL BKSPFILE(4, NBKSP)
00035           GO TO 150
00036     40    CALL BLOCKRD(1, -NBKSP, 0)
00044           GO TO 150
         C
         C            IF NFLAG = +1, CALCULATE THE NEW VALUE OF NCBLK TO BE
         C      USED IF THE DATA TAPE MUST BE REPOSITIONED.  IN THIS CASE
         C      ISAMP SHOULD APPEAR IN THE MIDDLE OF THE BUFFER.
         C
00045     90    NCBLK = ISBLK - (NBLK/2)
00051           GO TO 100
         C
         C            IF NFLAG = -1, CALCULATE THE NEW VALUE OF NCBLK TO BE
         C      USED IF THE DATA TAPE MUST BE REPOSITIONED.  IN THIS CASE
         C      ISAMP WILL APPEAR IN THE INITIAL BLOCK OF THE BUFFER.
         C
00052     95    NCBLK = ISBLK
         C
         C      CHECK FOR OVERFLOW AND UNDERFLOW OF SAMPLE BUFFER.
         C
00053    100    IF (ISAMPB+IWIDE-LGNBF)    101, 101, 126
00056    101    IF (ISAMPB)    110, 150, 150
         C
         C            ON UNDERFLOW, BACKSPACE TAPE UNTIL THE READ HEAD PASSES
         C      THE FIRST BLOCK WHICH IS TO BE IN THE ADJUSTED BUFFER.  THEN
         C      TRANSFER TO THE LOOP WHICH LOADS UP BUFF FROM THE TAPE UNIT.
         C
00060    110    IF (NCBLK)    115, 115, 118
00062    115    NCBLK = 0
00063           REWIND 4
00065           GO TO 145
00066    118    NBKSP = ITBLK - NCBLK
00067           CALL BKSPFILE(4, NBKSP)
00072           GO TO 145
         C
         C            ON OVERFLOW, CHECK THE DISTANCE BETWEEN THE PRESENT TAPE
         C      HEAD POSITION AND THE NEW FIRST BLOCK TO BE READ INTO BUFF.
         C
00073    126    IF (NCBLK - ITBLK)    125, 145, 140
         C
         C            IF THE DISTANCE IS NEGATIVE, SOME OF THE DATA IN BUFF IS
         C      STILL USEFUL.  RELOAD ALL OF THESE BLOCKS INTO THE FRONT PART
         C      OF BUFF.  THIS LOADING WILL BEGIN WITH THE DATA BLOCK WHICH IS
         C      TO BE THE NEW FIRST BLOCK OF BUFF AND WILL END WITH THE FINAL
```

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
 C         DATA WORD IN THE OLD SAMPLE BUFFER.
 C
   125     IBEG = (NCBLK - ICBLK)*LBLK
           IEND = LGNBF - 1
           DO 130, I=IBEG, IEND
           BUFF(I-IBEG) = BUFF(I)
   130     CONTINUE
 C
 C              NEXT LOAD THE REMAINING BLOCKS IN THE NEW BUFFER FROM TAPE.
 C              THE TAPE LOADING WILL START WITH THE FIRST BLOCK NOT LOADED
 C         BY THE MEMORY TRANSFER, AND WILL END WITH THE FINAL BLOCK IN THE
 C         SAMPLE BUFFER,
 C
           IBEG = ITBLK - NCBLK
           GO TO 146
 C
 C              IF THE DISTANCE BETWEEN THE PRESENT TAPE HEAD POSITION AND
 C         THE NEW FIRST BLOCK TO BE READ INTO BUFF IS GREATER THAN ZERO,
 C         FIRST POSITION THE TAPE UNIT TO THE CORRECT POINT.
 C
   140     CALL BLOCKRD(ITBLK, (NCBLK-1), 0)
   145     IBEG = 0
 C
 C              FINALLY, LOAD BUFF WITH THE REQUIRED NUMBER OF BLOCKS FROM
 C         TAPE.
 C
   146     IEND = NBLK - 1
           CALL BLOCKRD(IBEG, IEND, 1)
 C
 C         CHANGE ICBLK TO REFLECT NEW STATE OF BUFFER.
   149     ICBLK = NCBLK
           ISAMPB = ISAMP - ICBLK*LBLK
   150     RETURN
         END
--FORTRAN
```

5.2 BLOCKRD (IBEG, IEND, MODE)

This subroutine is the tape reading module for ADJUS2. It reads blocks of data in packed format from tape unit 4 and loads them into a temporary buffer, IBUF. If the MODE variable equals zero, the data is not used and the subroutine simply reads the number of blocks which would be between IBEG and IEND, inclusive. If the MODE variable is non-zero, BLOCKRD unpacks each block after it has been read into IBUF and loads the resulting floating point data into BUFF starting with the block specified by IBEG and continuing until the block specified IEND has been filled.

It should be noted that IBEG and IEND refer to block numbers within BUFF, not as they occur on the data tape. The tape is read beginning with wherever it is positioned on the tape unit. Thus in order to avoid overwriting other data, IBEG and IEND should both be less than the number of blocks allocated to BUFF (unless of course MODE equals 0).

The main reason for writing this operation as a subroutine (other than the fact that it took less core space as a subroutine than if the code had had to be written several times inline) was to isolate all of the tape reading operations into one program so that it would be easier to modify if that ever became necessary.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
          SUBROUTINE BLOCKRD(IBEG, IEND, MODE)
       C
       C         THIS SUBROUTINE IS USED TO READ BLOCKS OF DATA FROM THE
       C      DATA TAPE.  IF MODE IS NON-ZERO, THE BLOCKS ARE UNPACKED AND
       C      LOADED INTO BUFF.  IF MODE IS ZERO, THE BLOCKS ARE DISCARDED
       C      AS THEY ARE READ.  IBEG IS THE STARTING BLOCK AS REFERENCED
       C      IN BUFF AND IEND IS THE ENDING BLOCK.  READING BEGINS FROM
       C      WHEREVER THE TAPE HAPPENS TO BE POSITIONED.
       C
              TITLE*
              DIMENSION IBUF(250)
              EQUIVALENCE (IBUF(1), A(3002))
       C
       C         CALL STOPSCOP IN ORDER TO BE ABLE TO USE IBUF, WHICH IS
       C      LOCATED IN THE SAME AREA AS THE DISPLAY BUFFER, ISCOP1.
       C
              CALL STOPSCOP
 00003        DO 100, I=IBEG, IEND
 00007        JSTRT = I*LBLK
 00010        READ TAPE 4, (IBUF(J), J=1, LBLK4)
 00022        IF (MODE)     50, 100, 50
 00024    50  CALL UNPACK(IBUF, BUFF, 1, JSTRT, LBLK4)
 00032   100  CONTINUE
 00033   200  RETURN
 00034      END
--FORTRAN
```

5.3 UNPACK (B1, B2, I1, I2, N)

The UNPACK subroutine is used to convert data from the format used by the A to D converter and the data tapes to the floating point format used by the processing routines. It takes N successive packed format words from Buffer B1 starting at position I1 and converts each of them to 4 floating point numbers which are then stored consecutively in buffer B2 starting at I2. The packed samples are stored right to left in each word in B1.

The subroutine works on each B1 word by shifting one sample at a time from the A register to the Q register and then converting the sample to floating point. Note that the contents of B1 remain unchanged.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
          SUBROUTINE UNPACK(B1,B2,I1,I2,N)
      C
      C           THIS SUBROUTINE TAKES N 4-SAMPLE WORDS FROM B1 STARTING
      C        AT I1, AND CONVERTS EACH OF THEM TO 4 FLT. PT. WORDS STORED
      C        CONSECUTIVELY IN B2, STARTING AT I2.  THE SAMPLES ARE STORED
      C        RIGHT TO LEFT IN EACH B1 WORD.
      C
      C        B1     = INPUT BUFFER CONTAINING FOUR PACKED INTEGER SAMPLES
      C                 IN EACH ENTRY.
      C        B2     = OUTPUT BUFFER CONTAINING FLOATING POINT ENTRIES.
      C        I1     = STARTING POINT IN B1.
      C        I2     = STARTING POINT IN B2.
      C        ITEM2 = USED TO SAVE UNPACKED INTEGERS.
      C        L      = INDEX REGISTER 4.  USED TO KEEP TRACK OF POSITION
      C                 IN B2.
      C        TEM    = TEMPORARY STORAGE USED TO HOLD PACKED DATA WHILE IT
      C                 IS BEING PROCESSED.
      C
               DIMENSION B1(10000), B2(10000)
               L = I2
00003          NU= I1 + N - 1
00006          DO 5, I=I1, NU
00012          TEM = B1(I)
      --ILLAR
                ENI     2  0B           SET INDEX REG. 2 TO ZERO.
      LOOP      LDA        TEM          LOAD REMAINING PACKED SAMPLES, RT. JUST.
                LRS        14B          SHIFT ONE 12 BIT SAMPLE INTO Q REG.
                QRS        44B          RT JUST. SAMPLE IN Q REGISTER
                STQ        ITEM2
      *
      *           RESTORE THE REST OF THE WORD (CONTAINING UNPROCESSED
      *        SAMPLES) BACK INTO TEM, SINCE THE CONTENTS OF THE A REGISTER
      *        WILL BE DESTROYED DURING THE FLOATING POINT CONVERSION.
      *
                STA        TEM
      --FORTRAN
      C
      C           CONVERT UNPACKED INTEGER SAMPLE TO FLOATING POINT
      C        WHILE LOADING IT INTO 2ND. BUFFER.
      C
00015          B2(L) = ITEM2
      --ILLAR
                INI     4  1B           INCREMENT INDEX REG. 4 BY 1.
      *         ISK     2  3B           IF IR2=3, WORD IS FINISHED, SKIP TO STATEMENT 5
                SLJ     0  LOOP            OTHERWISE INCREMENT IR2 AND JUMP TO LOOP.
      --FORTRAN
00021      5   CONTINUE
00022          RETURN
00023       END
--FORTRAN
```

Section 6

SPEECH DISPLAY ROUTINES

The Speech Display routines consist of those programs which are actually used to generate display data for the display routine. They make use of the various types of speech processing programs to obtain the necessary display.

6.1 SPECTO

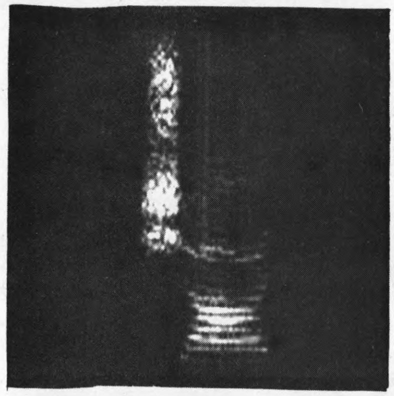
This routine is the display routine for generating spectrograms. It obtains data from the COMMON data buffer, BUFF, using the ADJUS2 subroutine to read data in from the data tape and update the buffer as necessary. It uses the fast Fourier transform subroutine, FFTB, to analyze the input data and takes the resultant complex coefficients (stored in the X and Y arrays) and converts them to frequency component magnitudes which are loaded into the appropriate time slice in FINT.

The Characteristics of the spectrographic intensities loaded into FINT depend on the various COMMON area system variables and constants. ISAMPB points to the beginning of the data to be processed in BUFF. It is continually updated as data is processed and whenever the buffer is refilled by ADJUS2.

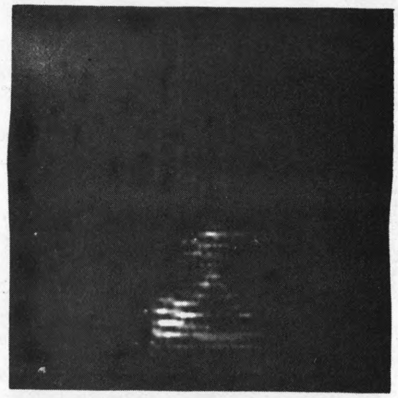
NSAMT indicates the number of data samples to be processed per time slice, i.e. the time slice's width, while IDELT specifies the number of data samples between successive time slices. Normally these two quantities are equal, but it is possible for them to have different values, in which case there will be either a spacing or an overlap between successive time slices.

IFREQ and ITIME specify the maximum number of frequency and time slice entries, respectively, which are loaded into FINT by the SPECTO subroutine. Note that the standard system practice of accessing FINT as a one-dimensional array is used. The two-dimensional subscripts are calculated explicitly using the IFMAX system variable. This allows both an increase in speed and the ability to change the relative size of the two dimensions in FINT without recompiling.

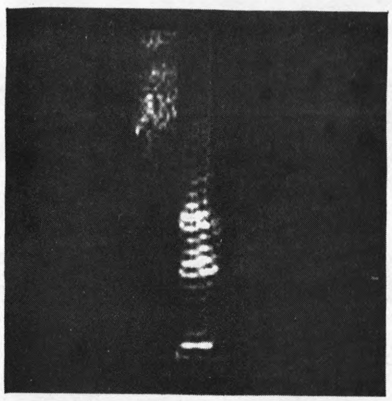
Figure 6.1.1 shows several examples of spectrographic displays calculated by SPECTO and produced using the system CRT display routines.



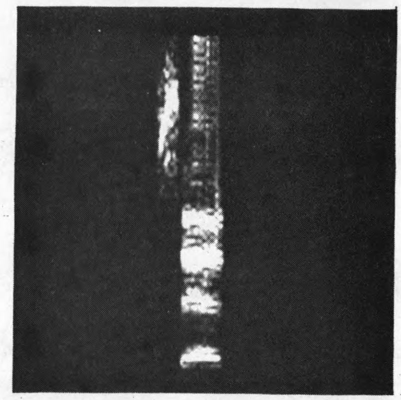
Speaker a
"shod"



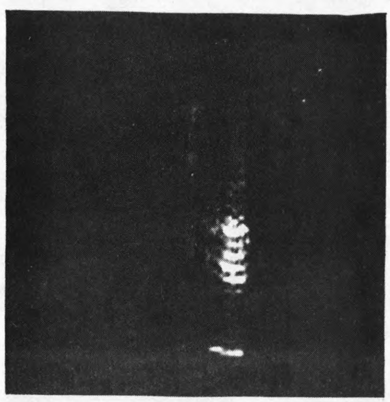
Speaker a
"vile"



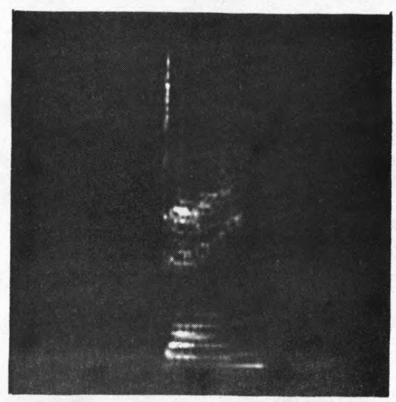
Speaker b
"said"



Speaker c
"said"



Speaker b
"ted"



Speaker d
"dame"

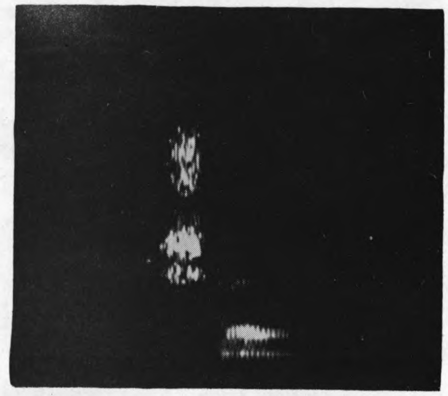
Figure 6.1.1 Examples of Displays Produced Using SPECTO

The commands needed to actually produce the display after having moved the data tape pointer to the position of the header block for the particular speech word are as follows:

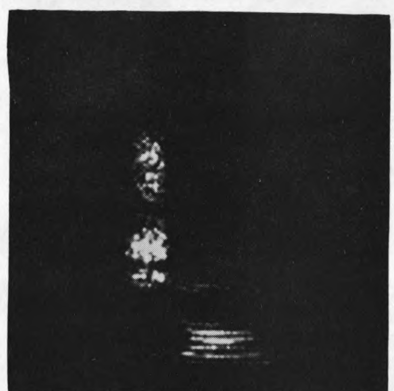
HEADT	Calculates ITIME based on time slice size and the length of data word.
FINIS	Prints out the variable values and turns off the display.
SPECTO	
NORMF	Normalizes the output of SPECTO.
HIEMP	Adds high frequency emphasis.
SPDISP	Produces the picture of the display.

These displays all involve the standard variable values specified in INITI. Figure 6.1.2 shows the effect of varying the size of the time slice. The other system variables have been adjusted to give the resulting displays an equivalent position size and parameter range. There have also been minor deviations from the standard display parameters so as to allow smooth transitions by factors of 2 in all of the relevent parameters (i.e. IDELX and IDELY were changed from their "standard" values in the displays in figure 6.1.1).

nsamt = 64



nsamt = 128



nsamt = 256

nsamt = 512

nsamt = 1024

Figure 6.1.2 Effect of Variations in Time Slice Size on the word "shod"

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
            SUBROUTINE SPECTO
      C          THIS SUBROUTINE GENERATES A SPECTROGRAM FROM DATA GIVEN IN BUFF,
      C     STARTING AT ISAMP.  THE AMPLITUDES OF THE OUTPUT FREQUENCIES AS A
      C     FUNCTION OF TIME ARE LOADED INTO THE FLOATING POINT ARRAY,
      C     FINT(FREQUENCY, TIME).
      C
      C     BUFF   = BUFFER CONTAINING TIME SAMPLES TO BE ANALYZED.
      C     IDELT  = SPACING (IN NO. OF TIME SAMPLES)BETWEEN ANALYSES OF DATA.
      C     IFREQ  = NO. OF FREQUENCY POSITIONS IN DISPLAY.
      C     IFMAX  = COMMON VARIABLE - MAXIMUM NUMBER OF ENTRIES ALONG
      C              FREQUENCY DIMENSION OF FINT.  USED TO CALCULATE
      C              DOUBLY SUBSCRIPTED ADDRESSES IN FINT.
      C     FINT   = OUTPUT INTENSITY ARRAY.  DIMENSION MUST BE SET TO
      C              FINT(IFREQ,ITIME) BEFORE COMPILING.
      C     ISAMPB = POSITION OF SAMPLE POINTER WITH RESPECT TO BUFF.  THIS
      C              IS THE FIRST SAMPLE IN BUFF TO BE PROCESSED WHEN FFTB
      C              IS CALLED.
      C     ISAMP  = POSITION OF THE SAMPLE POINTER WITH RESPECT TO
      C              THE COMPLETE DATA TAPE.
      C     ITIME  = NUMBER OF TIME SLICES.
      C     L      = INDEX REGISTER 4
      C     LGNSM  = LOG TO BASE 2 OF NSMT2
      C     M      = INDEX REGISTER 5
      C     NSAMT  = NO. OF TIME SAMPLES TO BE ANALYZED PER TIME SLICE.
      C     NSMT2  = NUMBER OF OUTPUT FREQUENCIES IN X.
      C            = NSAMT/2.
      C     X      = COMMON VARIABLE - REAL COMPONENT OUTPUT BUFFER FOR
      C              FFTB SUBROUTINE.
      C     Y      = COMMON VARIABLE - IMAGINARY COMPONENT OUTPUT BUFFER
      C              FOR FFTB SUBROUTINE.
      C
            TITLE*
            DO 600, I=1, ITIME
      C
      C          BEGIN LOOP TO PROCESS TIME SAMPLES.  START BY CHECKING
      C     THE BUFFER TO MAKE SURE IT CONTAINS THE COMPLETE TIME SLICE.
      C
00006       CALL ADJUS2(NSAMT, -1)
00010  361  CALL FFTB(NSAMT, NSMT2, LGNSM, BUFF(ISAMPB), 0)
      C
      C          LOAD OUTPUT OF FFT INTO FINT.  NOTE THAT ONLY A SINGLE
      C     SUBSCRIPT IS USED IN ACCESSING FINT EVEN THOUGH IT IS A DOUBLY
      C     SUBSCRIPTED ARRAY.  THIS GREATLY INCREASES THE SPEED OF THE
      C     CALCULATION IN CSL FORTRAN AND ALSO ALLOWS THE RELATIVE SIZES
      C     OF THE FINT DIMENSIONS TO BE CHANGED WITHOUT RECOMPILING.
      C
00021       L = (I-1)*IFMAX + 1
00025       DO 500, M=2, (IFREQ+1)
00032       FINT(L) = SQRTF(X(M)*X(M) + Y(M)*Y(M))
00041       L = L + 1
00044  500  CONTINUE
      C
      C          UPDATE ISAMP TO POINT TO THE NEXT TIME SLICE.
```

```
       CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
        C
 00045            ISAMP = ISAMP + IDELT
 00046    600     CONTINUE
 00050          END
--FORTRAN
```

6.2 ZEROC

This routine produces data for a zero-crossing display in which the frequency of the zero crossing rates is plotted as a function of time. It utilizes four digital filters to separate the speech input into four different frequency regions. Then it calculates the zero-crossing rate for each of these regions and makes entries in the FINT array at the frequencies corresponding to these rates. The magnitude of these entries is proportional to the average magnitude of the data within the respective filter region.

The program begins by initializing the digital filters and setting up various arrays and constants. It should be noted that the parameters which are to be used in the calculations of each particular filter are calculated by INILER and then stored in the arrays which have been named as parameters. In order to cut down on the size of the programs, the processing of the frequency bands is performed by means of a DO loop. Thus in order to avoid complications, the four filters utilize the same arrays to hold their "characterizing parameters" and simply use different regions of the array. The format is shown schematically in figure 6.2.1. The arrays can be thought of as doubly subscripted arrays in which the subscripts are being calculated explicitly "by hand".

The limits of the four frequency bands are contained in two constant arrays, IF1 and IF2, which are given as parameters to INILER. As can be seen in the program printout, the first filter has a negative lower edge value. This turns out empirically to give the flatest low frequency response. The reason for this can be seen intuitively from the fact that we are using a bandpass filter in an application where we would really prefer a lowpass filter.

filter variables		filter constants		delayed input	
Y1(0)	Y2(0)	CONST(0)	EAT(0)	XIN(0)	filter 1
Y1(1)	Y2(1)	CONST(1)			
Y1(2)	Y2(2)	CONST(2)			
Y1(3)	Y2(3)	CONST(3)			
Y1(4)	Y2(4)	CONST(4)	EAT(1)	XIN(1)	filter 2
Y1(5)	Y2(5)	CONST(5)			
Y1(6)	Y2(6)	CONST(6)			
Y1(7)	Y2(7)	CONST(7)			
Y1(8)	Y2(8)	CONST(8)	EAT(2)	XIN(2)	filter 3
Y1(9)	Y2(9)	CONST(9)			
Y1(10)	Y2(10)	CONST(10)			
Y1(11)	Y2(11)	CONST(11)			
Y1(12)	Y2(12)	CONST(12)	EAT(3)	XIN(3)	filter 4
Y1(13)	Y2(13)	CONST(13)			
Y1(14)	Y2(14)	CONST(14)			
Y1(15)	Y2(15)	CONST(15)			

Figure 6.2.1 Schematic Showing Use of Filter Variables and Constants in ZEROC

Once ZEROC has calculated the constants for the four filters by iteratively calling INILER, it begins the processing loop by zeroing that portion of the FINT array which corresponds to the current time slice. Note that as is the standard system practice, FINT is accessed as a single subscripted array even though it is conceptually doubly subscripted. IFQPS is used to keep track of the initial frequency entry (i.e. the lowest frequency) for the present time slice.

In the processing loop, the input data is divided into time slices each containing NSAMT samples. These samples are then processed four times, once for each frequency region, by entering the filter iteration loop. The iteration loop itself begins by digitally filtering the data, the output of the digital filter being loaded into the temporary array, BUF. (It should be noted that although this output is "discarded" at the end of each iteration, the filter does not get lost since it retains the necessary end state condition information in its filter variable arrays.) Once the data has been filtered, ZEROC calculates its average value. This is necessary since each filter passes a different amount of the data's DC component. Then the zero-crossing rate is calculated using the average value as the zero level.

Finally the average magnitude of the output of the filter is loaded into the appropriate entry of FINT based on the frequency of the zero-crossing rate. Note that an empirically determined fudge factor from the FMG array is multiplied times the average value before it is loaded into FINT. This is necessary because the four versions of the digital filter have varying magnitude factors. The filter program could have been written to correct for this, but it was more efficient to make one fudge on the average value rather than one fudge on each data point as it went through the filter.

At the end of each of the four iterations the corresponding entry of the REM array is loaded with the contents of PERIOD, the amount of time since the last zero-crossing in the current time slice. Note that this value will be restored at the appropriate iteration on the next time slice before calling ZECPIC to calculate a new zero-crossing rate. Thus no information is lost between successive zero-crossing calculations.

At the end of the iteration, note that if sense switch 2 is on, a line of data will be printed out for debugging purposes. After four loop iterations, ZEROC goes on to the next time slice and when ITIME time slices have been processed, the subroutine returns.

Figure 6.2.2 gives several examples of displays calculated by ZEROC and produced by the system CRT display routines. The commands needed to actually produce the display after having moved the data tape pointer to the position of the header block for a particular speech word are as follows:

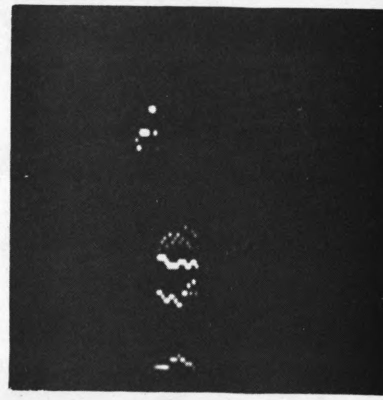
HEADT	Calculates ITIME based on the time slice size and the length of the data word.
FINIS	Prints out variable values and turns off display.
ZEROC	
NORMF	Normalizes the output of ZEROC
HIEMP	Adds high frequency emphasis
SPDISP	Produces the picture of the display

These displays all use the standard values specified in INITI.

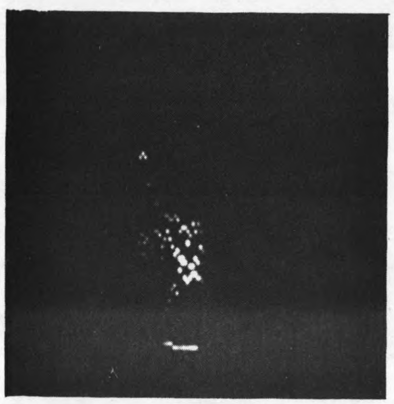
Speaker a
"shod"

Speaker a
"vile"

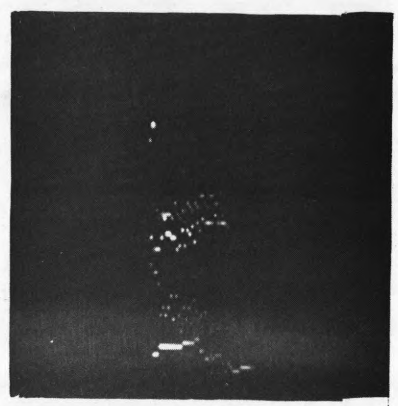
Speaker b
"said"



Speaker c
"said"



Speaker b
"ted"



Speaker d
"dame"

Figure 6.2.2 Examples of Displays Produced Using ZEROC

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
      SUBROUTINE ZEROC
C
C        THIS ROUTINE PRODUCES DATA FOR A ZERO-CROSSING DISPLAY BY
C     FIRST DIVIDING A TIME SLICE INTO 4 FREQUENCY BANDS BY MEANS
C     OF A DIGITAL FILTERING PROGRAM.  NEXT IT ANALYZES THE NUM-
C     BER OF ZERO CROSSINGS IN EACH FREQUENCY BAND AND CONVERTS
C     THESE TO FREQUENCIES.  THE CORRESPONDING FREQUENCIES IN FINT
C     ARE THEN LOADED WITH THE AVERAGE OF THE ABSOLUTE MAGNITUDE OF
C     THE OUTPUT OF THE CORRESPONDING FILTER BAND.
C
C     AVEMAG = AVERAGE MAGNITUDE OF FILTER OUTPUT.
C     BUFF   = COMMON VARIABLE.  INPUT DATA BUFFER.
C     CONST  = CONSTANT ARRAY USED BY DIGITAL FILTER.  THE ARRAY
C              IS DIVIDED INTO 4 BLOCKS (4 WORDS EACH), ONE BLOCK
C              CORRESPONDING TO EACH FREQUENCY BAND.
C     DTIME  = TIME PERIOD BETWEEN SAMPLES.
C     EAT,   =  CONSTANT ARRAYS USED BY DIGITAL FILTER.  EACH ARRAY
C      XIN     CONTAINS 4 ENTRIES (BEGINNING AT ENTRY 0), ONE FOR
C              EACH FILTER.
C     FILT   = FILTER OUTPUT BUFFER.
C     FMG    = CONSTANT ARRAY USED TO CONTAIN EMPIRICALLY DETERMINED
C              FUDGE FACTORS TO EQUALIZE THE MAGNITUDE OF THE OUTPUTS OF
C              THE FOUR FILTERS USED IN THE SUBROUTINE.
C     FREQ   = FREQUENCY CORRESPONDING TO THE AVERAGE RATE OF ZERO
C              CROSSINGS.
C     INILER = INITIALIZING SUBROUTINE FOR DIGITAL FILTER.  SETS
C              CONSTANT ARRAYS ACCORDING TO DESIRED EDGE FREQUENCIES
C              FOR OUTPUT OF FREQUENCY BAND.
C     ISAMF  = COMMON VARIABLE.  SAMPLING FREQUENCY.
C     ISAMP  = COMMON VARIABLE.  POSITION OF SAMPLE POINTER RELATIVE
C              TO DATA TAPE.
C     ISAMPB = COMMON VARIABLE.  POSITION OF SAMPLE POINTER RELATIVE
C              TO BUFF.
C     IF1,IF2= CONSTANT ARRAYS USED TO CONTAIN THE INITIAL AND FINAL
C              FREQUENCY VALUES, RESPECTIVELY, FOR THE FOUR BANDPASS
C              FILTERS USED BY THE PROGRAM (THE FILTERS CORRESPONDING
C              TO THE 0TH THROUGH 3RD ENTRIES IN EACH ARRAY).  THE
C              NEGATIVE VALUE FOR IF1(0) WAS CHOSEN SO THAT THE INITIAL
C              FILTER WOULD HAVE A FLATTER FREQUENCY RESPONSE, SINCE IT
C              IS ACTUALLY A LOW PASS RATHER THAN A BAND PASS FILTER.
C     IFQPS  = ADDRESS WITHIN FINT OF THE FIRST FREQUENCY ENTRY IN THE
C              CURRENT TIME SLICE.
C     K      = INDEX FOR FILTER LOOPS.  IT IS USED TO KEEP TRACK OF THE
C              FILTER BEING USED AND TO ADDRESS FILTER-SPECIFIC CONSTANTS
C              AND VARIABLES SUCH AS IF1,IF2,FMG,EAT, ETC.
C     L      = INDEX FOR FILTER DATA ARRAYS.  IT IS USED TO KEEP TRACK
C              OF THE INITIAL POSITION OF THE DATA ARRAYS USED BY THE
C              FILTER PROGRAM, SUCH AS Y1, Y2, AND CONST.  L IS INCRE-
C              MENTED BY 4, IE. THE LENGTH OF THE DATA ARRAYS, EACH TIME
C              A NEW FILTER IS TO BE USED.
C     LERNFIL= DIGITAL FILTERING SUBROUTINE.
C     M      = ADDRESS IN FINT CORRESPONDING TO THE ZERO CROSSING
C              FREQUENCY.
```

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
      C        NZCRS  = NUMBER OF ZERO CROSSINGS IN CURRENT TIME SLICE AT
      C                 CURRENT FREQUENCY BAND.
      C        PERIOD = COMMON VARIABLE.  PERIOD OF TIME IN CURRENT TIME SLICE
      C                 AFTER LAST ZERO CROSSING IN CURRENT FREQUENCY BAND.
      C        TIMPD  = TIME PER TIME SLICE WHICH CONTAINS THE ZERO CROSSINGS
      C                 ACTUALLY COUNTED.
      C        Y1,Y2, = INTERMEDIATE DATA FOR DIGITAL FILTER.  EACH ARRAY IS
      C                 DIVIDED INTO 4 BLOCKS (4 WORDS EACH) ONE BLOCK COR-
      C                 RESPONDING TO EACH FREQUENCY BAND.
               TITLE*
               DIMENSION BUF(256), CONST(15), EAT(3), FILT(256), IF1(3),
           1             IF2(3), PER(256), REM(3), XIN(3), Y1(15),
           2             Y2(15), FMG(3)
               DATA(IF1(0)= -97, 900,2200,3500)
               DATA(IF2(0)=1000,2800,4000,7000)
               DATA(FMG(0)=.909,1.64,1.56,3.03)
               EQUIVALENCE (BUF(1),A(3002)),(PER(1),A(4002)),(FILT(1),A(4302))
      C
      C             INITIALIZE THE FOUR FILTERS USED BY ZEROC, CALCULATE THE
      C        NEEDED CONSTANTS, AND ZERO OUT THE REM ARRAY.
      C
               L = 0
00003          DO 5, K=0, 3
00010          CALL INILER(IF1(K), IF2(K), BUF, FILT, Y1(L), Y2(L),
           1                  CONST(L), EAT(K), XIN(K))
00022          L = L + 4
00024     5    CONTINUE
00025          TIMPD = NSAMT*DTIME
      C
00027          DO 10, K=0, 3
00033    10    REM(K) = 0.0
      C
      C             BEGIN LOOP TO PROCESS ITIME TIME SLICES.
      C
00035          DO 500, J=1, ITIME
00041          IFQPS = (J-1)*IFMAX + 1
      C
      C             ZERO OUT THE FREQUENCY ENTRIES IN FINT CORRESPONDING TO THE
      C        CURRENT TIME SLICE.
      C
00044          DO 100, K=IFQPS, (IFQPS+IFREQ-1)
00053   100    FINT(K) = 0.0
      C
      C             DETERMINE THE AVERAGE VALUE OF THE CURRENT TIME SLICE
      C        AND LOAD BUF WITH THE INPUT DATA (WITH ITS DC COMPONENT
      C        REMOVED).
      C
00055          CALL ADJUS2(NSAMT, -1)
00057          CALL AVEMAG(BUFF(ISAMPB), 0.0, NSAMT, AVE)
00067          DO 150, K=0, (NSAMT-1)
00074   150    BUF(K) = BUFF(ISAMPB+K) - AVE
      C
      C             BEGIN FILTER LOOP TO PROCESS DATA THROUGH EACH OF THE
```

```
CSL FORTRAN OF SEPT 1968,  DATE   6/28/71
 C        FOUR FILTERS.
 C
00102          L = 0
00103          DO 400, K=0, 3
00110          CALL LERNFIL(X, NSAMT, BUF, FILT, Y1(L),
          1                 Y2(L), CONST(L), EAT(K), XIN(K))
00122          L = L + 4
00124          PERIOD = REM(K)
 C
 C              NEXT CALCULATE THE NUMBER OF TIMES THE FILTERED SIGNAL
 C        CROSSES ITS AVERAGE VALUE LINE.  NOTE THAT IN THE CALL TO
 C        ZECPIC, PER IS USED AS A DUMMY ARRAY TO CONTAIN THE OUTPUT
 C        PRODUCED BY THE CALL.  THIS DATA IS NOT USED HOWEVER, SINCE
 C        WE ONLY NEED THE NUMBER OF ZERO CROSSINGS AND NOT THE POSI-
 C        TIONS WHERE THEY OCCURRED.
 C              NOTE ALSO THAT SINCE THE DATA FROM FILT IS BOTH + AND -,
 C        -1000 IS USED AS THE ZERO LEVEL IN AVEMAG, SO THAT AND AVERAGE
 C        VALUE CAN BE CALCULATED INSTEAD OF AN AVERAGE MAGNITUDE.
 C
00125          CALL AVEMAG(FILT, -1000., NSAMT, AVE)
00132          AVE = AVE - 1000.
00133          CALL ZECPIC(FILT, AVE, NSAMT, PER, PER, NZCRS)
 C
 C              CALCULATE THE FREQUENCY CORRESPONDING TO THE NUMBER OF
 C        ZERO CROSSINGS DETECTED, AND CALCULATE THE FINT ENTRY CORRESPOND-
 C        ING TO THIS FREQUENCY AND TIME.
 C
00143    350   FREQ = 0.5*NZCRS/(REM(K) + TIMPD - PERIOD)
00150          M    = FREQ/DELF + IFQPS
 C
 C              CALCULATE THE AVERAGE MAGNITUDE OF THE OUTPUT OF THE DIGITAL
 C        FILTER AND ENTER THIS NUMBER(CORRECTED FOR THE RELATIVE GAIN OF THE
 C        FILTER) IN THE FINT ENTRY CORRESPONDING TO THE FREQUENCY OF THE
 C        NUMBER OF DETECTED ZERO CROSSINGS.
 C
00157          CALL AVEMAG(FILT, AVE, NSAMT, AVMAG)
00164          FINT(M) = FINT(M) + AVMAG*FMG(K)
00167          REM(K) = PERIOD
00170    371   IF (SENSE SWITCH 2)    373, 400
00172    373   ITMP = M - IFQPS + 1
00175          FTMP = AVMAG*FMG(K)
00176          PRINT 375, J, FREQ, ITMP, FTMP, AVE
00210    375   FORMAT (3H J=,I6, 7H  FREQ=,F12.1, 2X,2HM=,I6,
          1    8H  AVMAG=F8.1,2X,F8.3)
00210    400   CONTINUE
00211          ISAMP = ISAMP + IDFLT
00212    500   CONTINUE
00214          RETURN
00215         END
--FORTRAN
```

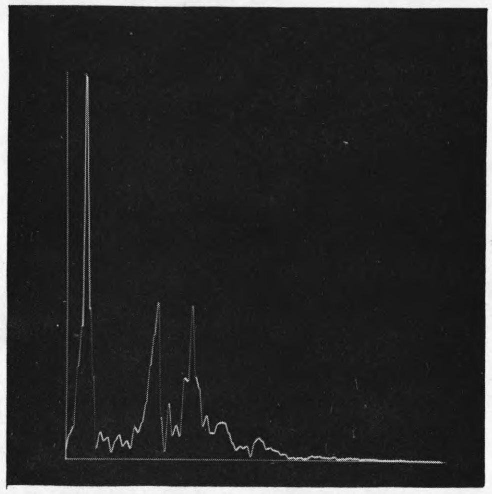
6.3 FORMEX

This subroutine is used to process the output of the SPECTO subroutine once it has been normalized and to extract the formant structure from it. After determining the formants, it clears the FINT array so as to leave only the formant peaks to be displayed.

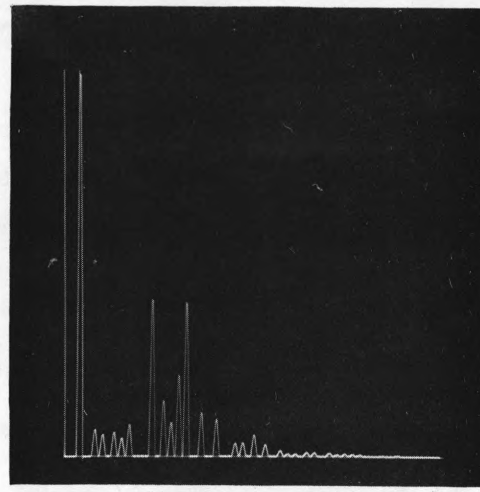
The subroutine iteratively processes each time slice individually. Thus the subroutine consists of one huge loop which is repeated until ITIME time slices have been processed. The loop begins by searching the time slice for its frequency component peak magnitudes. This search must be performed a second time on the results of the first search to get the actual peaks, since the first search only produces a frequency magnitude envelope. The processing of a typical time slice is shown in figure 6.3.1. Note that the initial peak picking operation will always produce an envelope consisting of all the minor peaks in the display. These minor peaks are the various harmonics of the voice pitch frequency and are characteristic of voiced speech.

As is the general characteristic of the system, FINT is treated as a singly subscripted array throughout the loop. IFQPS, the position of the first frequency component(i.e. lowest frequency) of the particular time slice, is calculated at the beginning of each pass through the loop and this value is then used throughout the remainder of the loop to determine positions in FINT.

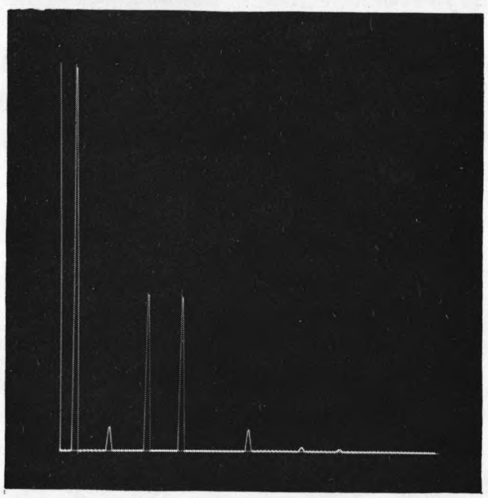
After calling the peak picking programs twice, FORMEX has to untangle the problem of indexes, since PEAPIC returns the index value relative to the array specified as the input and on the second call the input array is the output of the first call. This means that the values in the INDX2 array refer to positions in PBUF1 instead of in FINT. Thus FORMEX must do an



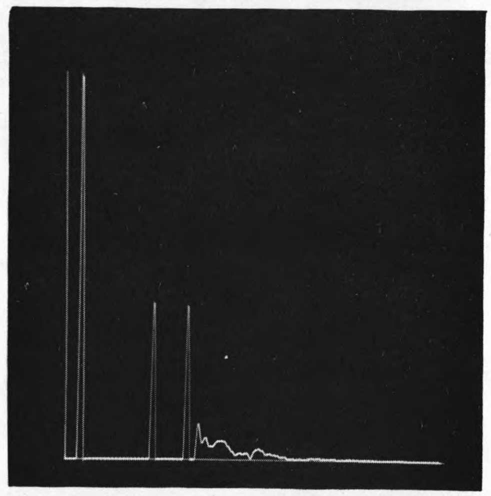
Initial Spectrum Analysis
of Time Slice



First Pass Selects All
Peaks Due to Pitch Harmonics



Second Pass Selects
Potential Formants



Final Formant Selection

Figure 6.3.1 Effect of the Peak-picking Process
on the Spectrum Analysis of a Single Time Slice

"indirect address" access to the INDX1 array to get the peak positions relative to FINT.

Once the proper peak positions have been loaded into INDX2, the data in FRMAG and INDX2 are sorted on the basis of the magnitude values in FRMAG. If the largest frequency component magnitude is too small to display, there is no further need to process the time slice and FORMEX goes to the next one. If it is large enough to be displayed, and it is above 3500 cps (this is determined by its corresponding position in FINT and the frequency spacing between frequency magnitude samples in FINT), the time slice is classified as a pure friction time slice and no further extraction is performed.

If the largest magnitude frequency component is over the minimum magnitude displayed and under 3500 cps, the time slice is classified as a formant time slice. It should be kept in mind however, that there may still be a significant amount of friction in the higher frequencies.

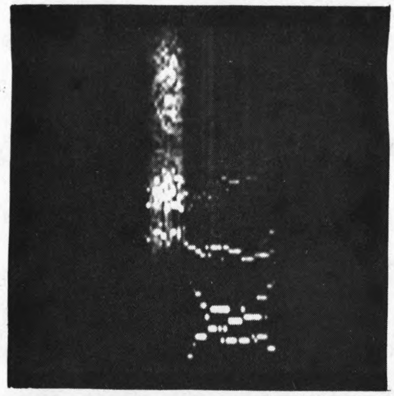
The formant processing assumes that only the four largest frequency magnitude components are formant candidates. In addition, if it turns out that the fourth largest magnitude is less than half the size of the third largest, the fourth largest is also discarded. This comparison of the two smallest formant candidates is repeated until the smallest candidate is greater than half the size of the second largest. Finally, any remaining candidates over 4000 cps. are eliminated since it is unlikely that a formant would appear above that frequency.

Once the formants have been picked, the FINT time slice is zeroed out beginning at the lowest frequency component and continuing up to the second entry higher than the highest frequency formant. Then only those magnitudes corresponding to the selected formants are reloaded into FINT. This technique retains the highest frequency components of the time slice and thus does not destroy any high frequency frictction which may be present. Note that the magnitudes of the formants are doubled when they are replaced in FINT. This allows them to stand out more forcibly in the display.

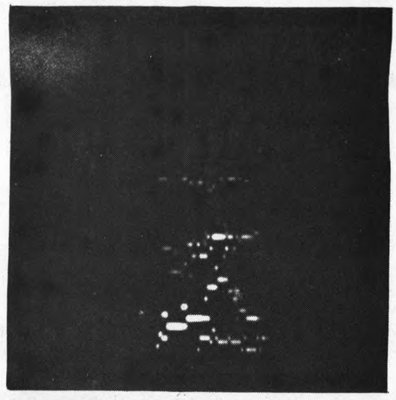
Once the time slice has been reloaded, FORMEX proceeds to the next time slice and when ITIME time slices have been processed, it returns.

Figure 6.3.2 shows several examples of spectrographic displays which have been processed by FORMEX to extract their formants. They have been produced using the system CRT display routines. The commands needed to actually produce the display after having moved the data tape pointer to the position of the header block for the particular speech word are as follows:

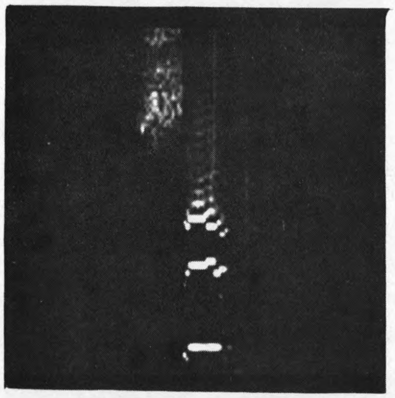
HEADT	Calculate ITIME based on time slice size and the length of the data word.
FINIS	Print out the variable values and turn off the display.
SPECTO	
NORMF	Normalize the output of SPECTO
HIEMP	Add high frequency emphasis
FORMEX	Extract formants
SPDISP	Produce the picture of the display.



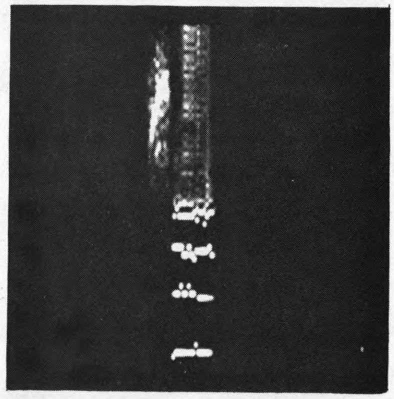
Speaker a
"shod"



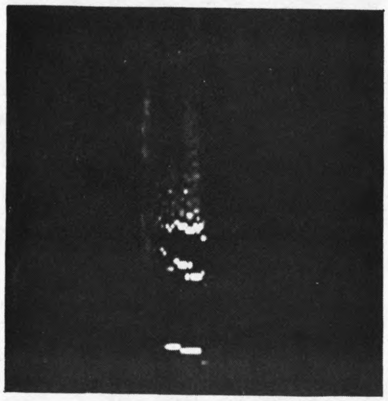
Speaker a
"vile"



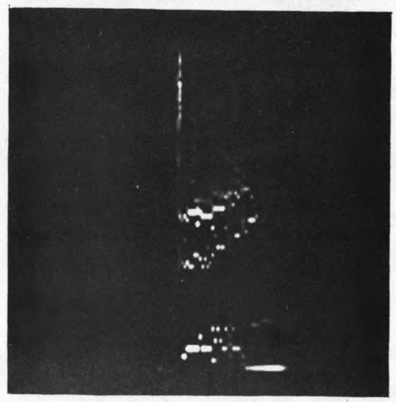
Speaker b
"said"



Speaker c
"said"



Speaker b
"ted"



Speaker d
"dame"

Figure 6.3.2 Examples of Displays Produced Using FORMEX

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
          SUBROUTINE FORMEX
     C
     C          THIS SUBROUTINE TAKES THE OUTPUT OF SPECTO, IE, A TIME VS.
     C      FREQUENCY INTENSITY ARRAY, AND PROCESSES IT TO EXTRACT THE FORMANT
     C      PEAKS.  THE EXTRACTION IS PERFORMED BY FINDING THE MAJOR FREQUENCY
     C      PEAKS IN EACH TIME SLICE AND THEN ELIMINATING THOSE WHICH ARE TOO
     C      SMALL OR WHOSE FREQUENCIES ARE TOO HIGH TO BE FORMANTS.
     C
     C      DELF  = COMMON VARIABLE.  FREQUENCY DIFFERENCE BETWEEN FREQUENCY
     C              MAGNITUDE ENTRIES IN FINT.
     C      FINT  = COMMON ARRAY.  CONTAINS FREQUENCY COMPONENT MAGNITUDES.
     C      FINTM = COMMON VARIABLE.  MINIMUM INTENSITY WHICH WILL BE DIS-
     C              PLAYED BY THE DISPLAY ROUTINE.
     C      FQMAX = FREQUENCY OF THE LARGEST MAGNITUDE FORMANT CANDIDATE.
     C      FRMAG = TEMPORARY BUFFER USED TO HOLD PEAK MAGNITUDES OF FORMANT
     C              CANDIDATES.
     C      FRMPT = SUBROUTINE.  USED TO PRINT OUT FORMANTS OR FORMANT
     C              CANDIDATES.
     C      IFMAX = COMMON VARIABLE.  MAXIMUM NUMBER OF ENTRIES ALONG THE
     C              FREQUENCY DIMENSION IN FINT.  USED TO CALCULATE SUB-
     C              SCRIPTS IN FINT.
     C      IFQPS = COMMON VARIABLE.  INITIAL FREQUENCY POSITION WITHIN FINT
     C              FOR THE CURRENT TIME SLICE.  USED TO CALCULATE SUBSCRIPTS
     C              IN FINT.
     C      IFREQ = COMMON VARIABLE.  NUMBER OF FREQUENCY ENTRIES IN DISPLAY.
     C      INDX1 = TEMPORARY BUFFER USED TO HOLD INDEX POSITIONS OF PEAKS IN
     C              FINT.
     C      INDX2 = TEMPORARY BUFFER USED TO HOLD INDEX POSITIONS OF FORMANT
     C              CANDIDATES.
     C      LIM1  = LAST ENTRY USED IN PBUF1 AND INDX1.
     C      LIM2  = LAST ENTRY USED IN FRMAG AND INDX2.
     C      NFORM = NUMBER OF FORMANTS.
     C      ORDER = SUBROUTINE.  USED TO SORT MAGNITUDE AND INDEX ARRAYS.
     C      PBUF1 = TEMPORARY BUFFER USED TO HOLD PEAK MAGNITUDES.
     C
            TITLE*
            DIMENSION PBUF1(50), FRMAG(25),
         1            INDX1(50), INDX2(25)
            EQUIVALENCE (PBUF1(1), A(101)),
         1              (FRMAG(1), A(201)), (INDX1(1), A(301)),
         2              (INDX2(1), A(401))
            DO 300, I=1, ITIME
00006       NFORM = 4
00007       IFQPS = (I-1)*IFMAX + 1
     C
     C          USE PEAPIC TWICE TO EXTRACT THE PEAK FREQUENCY VALUES AND
     C      POSITIONS.  THESE PEAK FREQUENCIES ARE THE INITIAL FORMANT CANDIDATES
     C
00012       CALL PEAPIC(FINT(IFQPS),IFREQ,PBUF1,INDX1,LIM1)
00024       CALL PEAPIC(PBUF1, LIM1, FRMAG, INDX2, LIM2)
00032   40  PRINT 50, I, LIM1, LIM2
00041   50  FORMAT (///11H TIME SLICE,3I5)
00041       DO 10, J=1, LIM2
```

```
      CSL FORTRAN OF SEPT 1968,  DATE   6/28/71
00045    10    INDX2(J) = INDX1(INDX2(J))
       C
       C            SORT PEAK MAGNITUDES IN ORDER OF HIGHEST TO LOWEST
       C       VALUES.
       C
00052          CALL ORDER(FRMAG, INDX2, LIM2, 0, 0)
       C
       C            NEXT CLASSIFY THE TIME SLICE ACCORDING TO THE MAGNITUDE AND
       C       FREQUENCY VALUES OF THE PEAKS.  IF THE LARGEST MAGNITUDE IS TOO
       C       SMALL TO BE DISPLAYED, GO TO THE NEXT TIME SLICE,
       C
00060          IF (FRMAG(1) - FINTM)      300, 300, 100
       C
       C            IF THE LARGEST FREQUENCY COMPONENT IS ABOVE 3500 CPS., THE
       C       TIME SLICE IS CLASSIFIED AS FRICTION, IE. S, F, T, ETC.
       C            IF THE LARGEST FREQUENCY COMPONENT IS LESS THAN 3500 CPS.,
       C       THE TIME SLICE MAY STILL CONTAIN FRICTION.  IN THIS CASE, PRO-
       C       CESS THE TIME SLICE FOR FORMANTS FIRST.
       C
00065   100    FQMAX = INDX2(1)*DELF
00072          IF (FQMAX - 3500.)      200, 200, 140
       C
       C            PROCESSING OF A FRICTION TIME SLICE.
       C            IN THIS CASE PRINT OUT ALL OF THE PEAKS IN ORDER OF THEIR
       C       FREQUENCIES AND GO TO THE NEXT TIME SLICE.
       C
00075   140    CALL ORDER(FRMAG, INDX2, LIM2, 1, 1)
00103          CALL FRMPT(FRMAG, INDX2, LIM2)
00107          GO TO 300
       C
       C            PROCESSING OF A FORMANT TIME SLICE.
       C       BEGIN WITH THE FOUR LARGEST FORMANT CANDIDATES.  CHECK THE SIZE
       C       OF THE 4TH. LARGEST AS COMPARED TO THE 3RD. LARGEST.
       C
00110   200    IF (FRMAG(3) - 2.0*FRMAG(4)) 250, 250, 240
       C
       C            IF THE SMALLEST OF THE POTENTIAL FORMANTS IS LESS THAN HALF
       C       THE SIZE OF THE NEXT LARGEST FORMANT, ELIMINATE IT.
       C
00121   240    FRMAG(4) = 0.0
00124          NFORM = NFORM - 1
       C
       C            ORDER FORMANTS BY FREQUENCY.
       C
00126   250    CALL ORDER(FRMAG, INDX2, NFORM, 1, 1)
       C
       C            ELIMINATE ANY FORMANTS ABOVE 4000 CPS SINCE FORMANTS
       C       SHOULD NOT BE THAT HIGH.
       C
00135   251    FQMAX = INDX2(NFORM)*DELF
00142          IF (FQMAX - 4000.)      254, 254, 252
00145   252    NFORM = NFORM - 1
00146          GO TO 251
```

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
          C
          C               PRINT OUT THE FORMANTS.
          C
00150      254     CALL FRMPT(FRMAG, INDX2, NFORM)
          C
          C               THEN ZERO OUT THE TIME SLICE UP TO THE SECOND FREQUENCY
          C        ENTRY ABOVE THE HIGHEST FORMANT AND ONLY FILL IN THOSE LOWER
          C        FREQUENCY ENTRIES CORRESPONDING TO FORMANT FREQUENCIES.
          C
00154              JFINI = IFQPS + INDX2(NFORM) + 1
00160              DO 255, J=IFQPS, JFINI
00165      255     FINT(J) = 0.0
00167              DO 260, J=1, NFORM
00173              M = IFQPS - 1 + INDX2(J)
00175              FINT(M) = 2.0*FRMAG(J)
00177      260     CONTINUE
00201      300     CONTINUE
00202              RETURN
00203           END
--FORTRAN
```

6.4 FRMPT (FRMAG, INDX2, NFORM)

This is a short subroutine used exclusively by FORMEX to print out lists of formants or formant candidates. FRMAG is an array containing magnitudes, INDX2 is an array containing index positions, and NFORM specifies the number of formants to be printed out.

```
      CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
                SUBROUTINE FRMPT(FRMAG, INDX2, NFORM)
          C
          C          THIS SUBROUTINE IS USED BY FORMEX TO PRINT OUT THE FORMANT
          C     FREQUENCIES AND THEIR VALUES.  EACH FORMANT IS PRINTED OUT GIV-
          C     ING ITS NUMBER, FREQUENCY INDEX, FREQUENCY VALUE, AND MAGNITUDE.
          C     AFTER NFORM FORMANTS HAVE BEEN PRINTED, THE SUBROUTINE RETURNS.
          C
                TITLE*
                DIMENSION FRMAG(1), INDX2(1)
                IF (SENSE SWITCH 2)    50, 300
 00004     50   DO 200, J=1, NFORM
 00011          FRFRQ = INDX2(J)*DELF
 00013          PRINT 100, J, INDX2(J), FRFRQ, FRMAG(J)
 00023    100   FORMAT (9H FORMANT ,I3,2X,14HFREQ. SLICE = ,I3,2X,
             1         7HFREQ. =,F7.0,2X,8HMAGNIT.=,F10.3)
 00023    200   CONTINUE
 00024    300   RETURN
 00025         END
--FORTRAN
```

6.5 PYRON

This subroutine is used to generate x-y speech displays of the type described by Pyron and Williamson [1965]. Basically these displays plot a zero-crossing rate vs. the amplitude envelope of the speech signal. Sense switch 3 is used to select one of two options. If sense switch 3 is up, the routine will produce a plot using the zero-crossing rate of the original speech signal, Z_1, while if sense switch 3 is down it will produce a plot using the zero-crossing rate of the derivative of the speech signal Z_2.

The subroutine takes its data from BUFF and processes it in "time slices" although this term does not have the same significance that it does in the case of programs such as SPECTO. In effect, the time slice (which contains NSAMT data samples) is merely a convenient processing unit for the various routines used by PYRON. At any rate after setting up the various constants and filters, PYRON enters its main loop which processes ITIME of these slices.

The main loop is divided into two sections, one to process the y input for the display, i.e. the Z signal, and one to produce the x input, i.e. the amplitude envelope. The y input processing begins by placing the data to be processed in the ZBUF array. In the Z_1 case (i.e. Sense switch 3 is up) this simply involves loading the time slice data from BUFF to ZBUF. In the Z_2 case, the subroutine DIFFER is called to differentiate the data in BUFF and load it into ZBUF.

Next a threshold check is made on the data in ZBUF. If its average magnitude deviation about its average value is less than 5.0 the time slice most probably is simply noise with no speech signal present and the program skips the processing and proceeds to the next time slice.

This threshold detecting allows the program to eliminate the "blank spaces" before and after words, since these regions would still have a definite zero-crossing rate even though their amplitudes were small and would thus produce obscuring garbage on the display. It should be noted that in this particular case, the size of each time slice would make a difference in how much of the non-signal data is eliminated, since a large time slice with only a small amount of signal at the end would be processed, whereas if it were broken up into smaller time slices only the last one would be processed.

If the input magnitude is large enough to be processed, the data is first used to obtain the Z signal. To do this the data is sent through the ZECPIC subroutine whose output ABUF then contains the length in seconds of the successive zero-crossing intervals. PYRON then zeros out ZBUF and inserts impulses in it at intervals approximating the positions of the zero-crossings. Finally the pulses are sent through a smoothing filter, i.e. the subroutine ENVLP, to obtain a buffer containing the successive samples of the smoothed Z signal output.

In order to have a full scale deflection on the display represent about 6000 zero-crossings per second while using a smoothing filter with a response time of 10 ms. (these are the values used by Pyron and Williamson), we need to adjust the amplitude of the impulses generated by the zero-crossings. If:

A = amplitude of the impulse produced by a zero-crossing

n = the number of zero value samples between crossings

A' = full scale deflection at 6000 crossings/sec.

then at equilibrium with a 3000 cps input, the value of the decayed output after n zero valued samples, A', will be:

$$A' = (\text{Amplitude after last pulse}) \times (\text{decay factor for 1 sample})^n$$

$$= (\ (1-e^{-.001})\ A + e^{-.001}\ A'\ ) \times (e^{-.001})^n$$

$$= (.005A + .995\ A') \times .995^n$$

But at 6000 zero crossings per second (using a 20 KC sampling rate) there will be 3.3 samples per crossing or 2.3 zero-value samples per crossing. If we let $A' = 1000$ (full scale deflection is actually 1024) then we get:

$$1000 = .989\ (.005A + 995.)$$

$$1000 = .00499A + 984$$

$$16 = .00499A$$

or $$A \approx 3200$$

and empirically this turns out to give a reasonable display. Figure 6.5.1 shows an example of the output of the zero-crossing rate detector.

Obviously, the Z signal could have been produced much more simply directly from the output of ZECPIC. However, the present method was chosen because it was directly analogous to the method used by Pyron and Williamson in their hardware implementation of the display and would therefore hopefully contain the same quirks and peculiarities as theirs.

In order to produce the amplitude envelope the data is first half-wave rectified and then sent through a smoothing filter. In this case the smoothing filter was adjusted to have a rise time of 1 ms. and a fall time of 25 ms. Figure 6.5.2 gives an example of the filter's operation. Pyron and Williamson [1965] specify a 5 ms. rise time but this was an empirically determined value and in the present case 1 ms. seemed to work better. The successive output samples of the filter routine are loaded into the ABUF array.

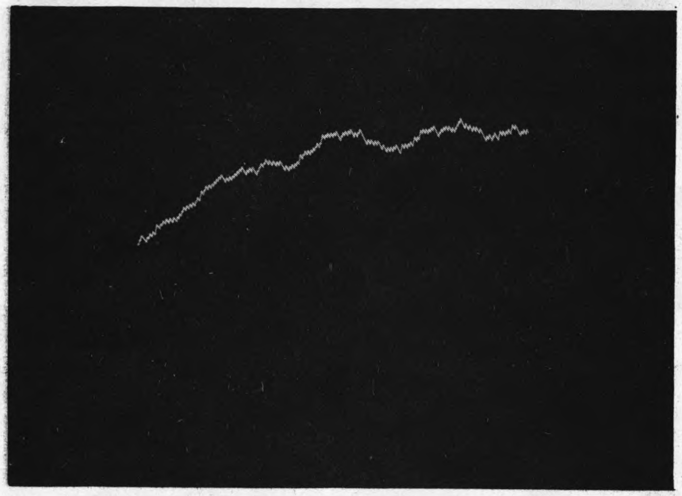


Figure 6.5.1 Zero Rate Output Used as X Input to PYRON Display

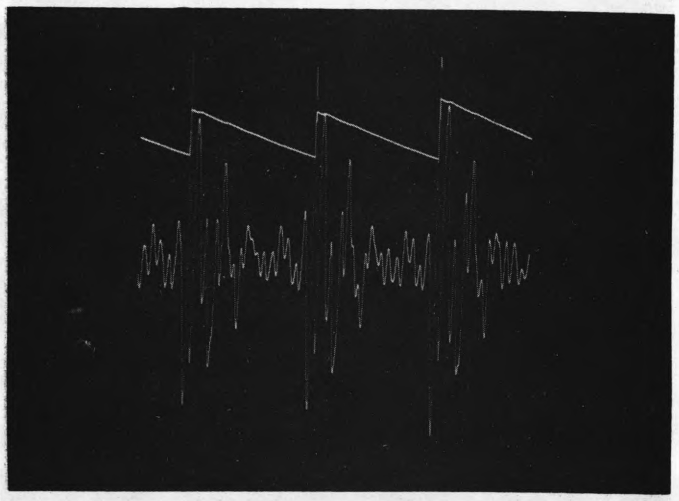


Figure 6.5.2 Amplitude Envelope Output Used as Y Input to PYRON Display. Superimposed on Speech Signal Input.

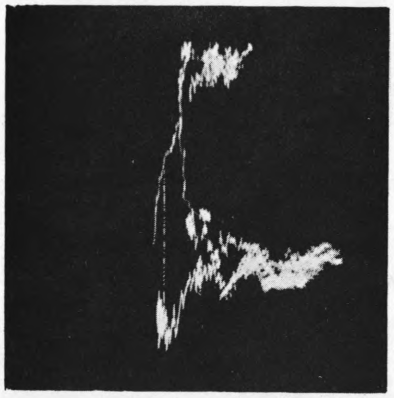
The final operation in the time slice is to plot the Z signal vs. the amplitude envelope. This is done using the DISSY subroutine. Then a photograph is taken of that section of the display, the data pointer is updated, and the subroutine moves on to process the next time slice.

It should be noted that throughout the subroutine, sense switch 2 is used to indicate the debugging mode. If it is on, then at various stages in the subroutine the intermediate data will be displayed and/or printed out.

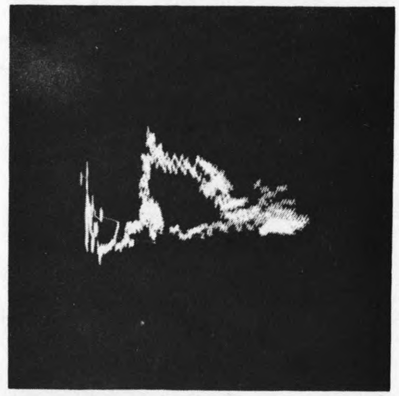
Figure 6.5.3 shows several examples of Z_1 vs. amplitude displays while figure 6.5.4 shows examples of displays of Z_2 vs. amplitude. The commands used to produce the display after having moved the data tape pointer to the position of the header block for the particular speech word are:

HEADT	Calculate ITIME based on time slice size and the length of the data word.
FINIS	Print out the variable values and turns off DISPLAY.
PYRON	

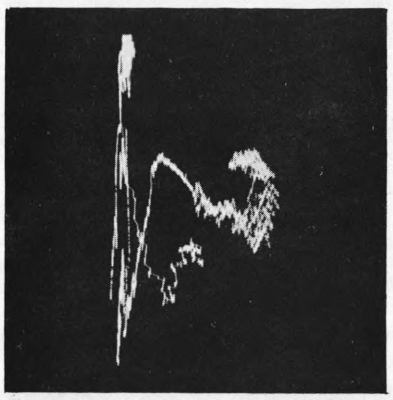
The "time slize size" is not especially critical in these displays but was kept at 1000 for the pictures in figure 6.5.3 and 6.5.4.



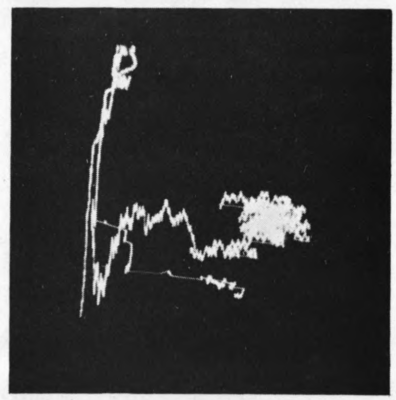
Speaker a
"shod"



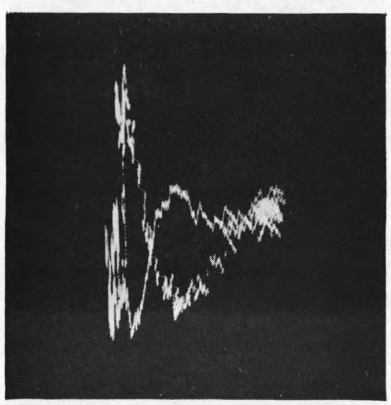
Speaker a
"vile"



Speaker b
"said"



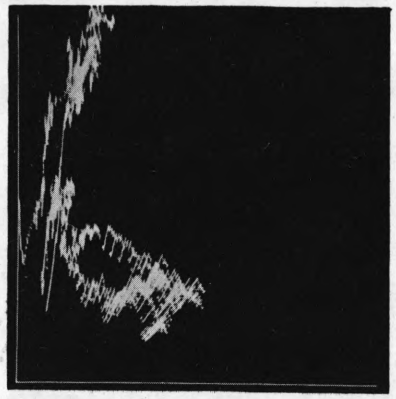
Speaker c
"said"



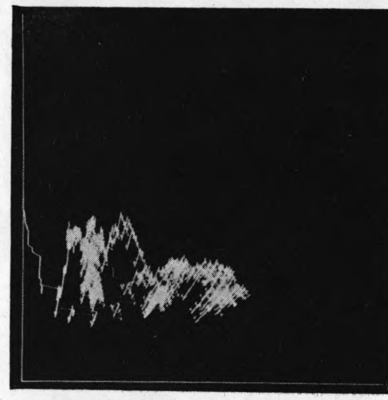
Speaker b
"ted"

Speaker d
"dame"

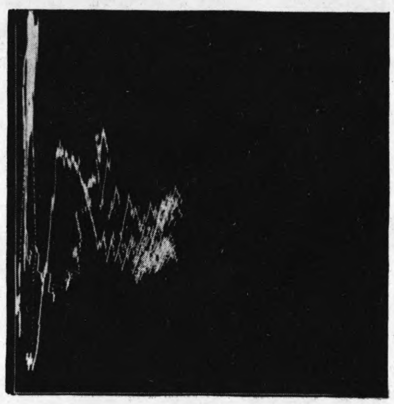
Figure 6.5.3 Examples of Z_1 vs. Amplitude Envelope Displays Produced by PYRON



Speaker a
"shod"

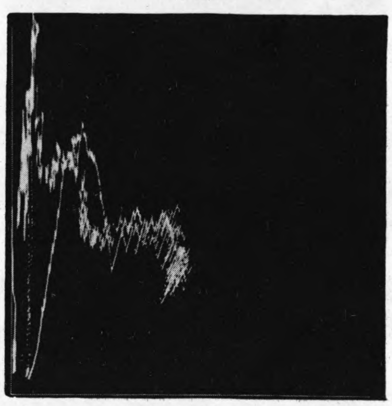


Speaker a
"vile"

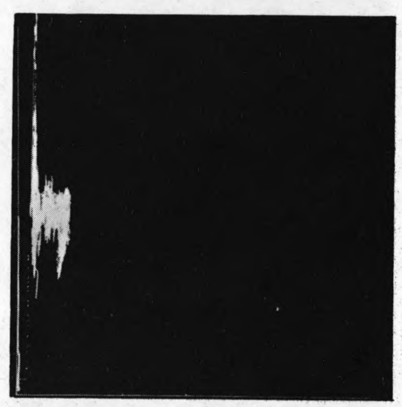


Speaker b
"said"

Speaker c
"said"



Speaker b
"ted"



Speaker d
"dame"

Figure 6.5.4 Examples of Z_2 vs. Amplitude Envelope Displays Produced by PYRON

```
CSL FORTRAN OF SEPT 1968, DATE  8/12/71
      SUBROUTINE BYRON
C
C           THIS SUBROUTINE IS USED TO GENERATE SPEECH DISPLAYS OF THE
C       TYPE DESCRIBED BY BYRON AND WILLIAMSON.  IF SENSE SWITCH 3 IS UP,
C       THE SUBROUTINE WILL PRODUCE A Z1 VS. AMPLITUDE ENVELOPE DISPLAY
C       OF THE SPEECH INPUT, WHILE IF SENSE SWITCH 3 IS DOWN, IT WILL
C       PRODUCE A Z2 VS. AMPLITUDE ENVELOPE DISPLAY.  Z1 IS PROPORTIONAL
C       TO THE NUMBER OF ZERO CROSSINGS IN THE SPEECH SIGNAL ITSELF, WHILE
C       Z2 IS PROPORTIONAL TO THE NUMBER OF ZERO CROSSINGS IN THE DERIVA-
C       TIVE OF THE SPEECH SIGNAL.
C           THE SUBROUTINE UTILIZES TWO DIFFERENT SMOOTHING FILTERS, ONE
C       FOR THE Z SIGNALS WHICH SMOOTHS OUT THE SPIKES PRODUCED AT EACH
C       ZERO CROSSING, AND THE OTHER TO FOLLOW THE RECTIFIED SPEECH IN-
C       PUT AND PRODUCE THE AMPLITUDE ENVELOPE.
C           THE SUBROUTINE CONTAINS A DEBUGGING FEATURE UTILIZING SENSE
C       SWITCH 2.  WHEN THIS SWITCH IS UP, THE PROGRAM WILL DISPLAY ON
C       THE CRT, THE DATA TO BE PROCESSED DURING EACH PROCESSING CYCLE.
C       IF THE DIFFERENTIATION OPTION IS USED, IT WILL ALSO DISPLAY THE
C       INTERMEDIATE RESULTS AFTER DIFFERENTIATION.  FINALLY, IT PRINTS
C       OUT THE FINAL CONTENTS OF ZBUF, (IE. THE AVERAGE ZERO CROSSING
C       RATE) AT THE END OF EACH CYCLE.
C
C       ABUF    = BUFFER USED TO CONTAIN THE SAMPLES OF THE AMPLITUDE
C                 ENVELOPE.
C       AVE     = AVERAGE VALUE CALCULATED BY AVEMAG.
C       AVEM    = AVERAGE MAGNITUDE CALCULATED BY AVEMAG.
C       BF1,BF2= FALLING SIGNAL COEFFICIENTS USED BY ENVLP.
C       BR1,BR2= RISING SIGNAL COEFFICENTS USED BY ENVLP.
C       BUFF    = DATA BUFFER.
C       DTIME   = TIME BETWEEN DATA SAMPLES.
C       EAFT,   = FALL TIME CONSTANT USED BY ENVLP
C        EAF2
C       EART,   = RISE TIME CONSTANT USED BY ENVLP.
C        EAR2
C       FTMP    = TEMPORARY FLOATING POINT VARIABLE USED TO CONTAIN
C                 THE MAXIMUM X VALUE FOR DISSY.
C       ISAMP   = POSITION OF THE DATA POINTER WITH RESPECT TO THE
C                 DATA TAPE.
C       ISAMPB  = POSITION OF THE DATA POINTER WITH RESPECT TO BUFF.
C       ISCOP1  = DISPLAY BUFFER FOR DISSY.
C       ITIME   = NUMBER OF TIME SLICES TO BE PROCESSED.
C       NSAMT   = NUMBER OF SAMPLES TO BE PROCESSED PER TIME SLICE.
C       NZCR    = NUMBER OF ZERO CROSSINGS FOUND BY ZCRPIC.
C       XB      = DUMMY VARIABLE USED BY DISSY.
C       Y1,Y2  = VARIABLES USED BY ENVLP TO SAVE LAST DATA POINT
C                PROCESSED BY FILTER.
C       ZBUF    = BUFFER USED TO CONTAIN THE SAMPLES OF THE ZERO CROSSING
C                 RATE.
C
        TITLE*
        DIMENSION ABUF(4000), ZBUF(4000)
        EQUIVALENCE (ABUF(0), A(11002)), (ZBUF(0), A(15003))
C
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
       C          INITIALIZE CONSTANTS AND SET UP FILTER PARAMETERS.
       C
                  CALL IFILT(10.0,NSAMT,10.0,BR1,EAR1,BF1,EAFT,Y1)
00013             CALL IFILT( 1.0,NSAMT,25.0,BR2,EAR2,BF2,FAF2,Y2)
00024             PERIOD = 0.0
00025             XB = 0.0
00026             Y1 = 500.
00027             Y2 = 0.0
00030             FTMP = NSAMT
       C
       C               PROCESS ITIME TIME SLICES.
       C
00031             DO 299, I=1, ITIME
00036             CALL ADJUS2(NSAMT, -1)
       C
       C               CHECK SENSE SWITCH 3 TO DETERMINE THE TYPE OF PLOT TO BE
       C          PRODUCED.
       C
00040      25     IF (SENSE SWITCH 3)     26, 30
       C
       C               FOR A Z1 VS. AMPLITUDE ENVELOPE PLOT, TAKE THE DATA DIRECTLY
       C          FROM BUFF WITHOUT PROCESSING.
       C
00042      26     DO 27, J=0, (NSAMT-1)
00050      27     ZBUF(J) = BUFF(ISAMPB + J)
00055             GO TO 35
       C
       C               FOR A Z2 VS. AMPLITUDE ENVELOPE PLOT, TAKE THE DERI-
       C          VATIVE OF THE INPUT.
       C               IF THE DEBUGGING MODE IS BEING USED, DISPLAY THE INPUT
       C          TO THE DIFFERENTIATOR ON THE CRT.
       C
00056      30     IF (SENSE SWITCH 2)     31, 34
00060      31     CALL DISSY(XB,BUFF(ISAMPB),NSAMT,ISCOP1,8000,FTMP,1024.,0)
00075             CALL WHATNOW
00076      34     CALL DIFFER(BUFF(ISAMPB), NSAMT, ZBUF)
00105      35     IF (SENSE SWITCH 2)     40, 45
00107      40     CALL DISSY(XB,ZBUF,NSAMT,ISCOP1,8000,FTMP,1024.,0)
00121             CALL WHATNOW
00122             PRINT 43, I
00127      43     FORMAT (9H ITIME = I4,3X,27HINTERMEDIATE ZBUF CONTENTS.)
00127             PRINT 300, (ZBUF(J), J=0, NSAMT)
00141      45     CONTINUE
       C
       C               NEXT CHECK THE RESULTING DATA TO SEE IF IT IS ABOVE
       C          THE MINIMUM AMPLITUDE THRESHOLD.  IF IT IS NOT, SKIP TO THE
       C          NEXT TIME SLICE.
       C
00141             CALL AVEMAG(ZBUF, 0.0, NSAMT, AVE)
00146             CALL AVEMAG(ZBUF, AVE, NSAMT, AVEM)
00153             IF (AVEM - 5.0)     290, 290, 46
       C
       C               IF THE AVERAGE MAGNITUDE IS ABOVE THE THRESHOLD VALUE,
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
      C        CONTINUE PROCESSING BY FINDING THE ZERO CROSSINGS IN THE
      C        DATA IN ZBUF.
      C
00156    46    CALL ZECRIC(ZBUF, AVE, NSAMT, ZBUF, ABUF, NZCR)
00165          DO 48, J=0, NSAMT
00171    48    ZBUF(J) = 0.0
00173          DO 50, J=0, (NZCR - 1)
00200          M = ABUF(J)/DTIME
00203          ZBUF(M) = 3200.
00204    50    CONTINUE
      C
      C              FINALLY SMOOTH OUT THE PULSES IN ZBUF USING A SMOOTHING
      C        FILTER TO PRODUCE A SMOOTHED ZERO RATE SIGNAL IN ZBUF.
      C
00205          CALL ENVLP(ZBUF,NSAMT,ZBUF,BR1,EAR1,BF1,EAF1,Y1)
      C
      C              IN ORDER TO PRODUCE THE AMPLITUDE ENVELOPE, FIRST CALCU-
      C        LATE THE AVERAGE VALUE OF THE INPUT DATA AND THEN HALF-WAVE
      C        RECTIFY THIS DATA ABOUT THE AVERAGE VALUE.
      C
00216          CALL AVEMAG(BUFF(ISAMPB), -1023., NSAMT, AVE)
00226          AVE = AVE - 1023.
00227          CALL RECTIF(BUFF(ISAMPB), AVE, NSAMT, ABUF)
      C
      C              FINALLY, REMOVE THE DC COMPONENT TO PRODUCE A RANGE OF
      C        VALUES BETWEEN 0 AND 512.
      C
00240          DO 75, J=0, (NSAMT - 1)
00245    75    ABUF(J) = ABUF(J) - AVE
      C
      C              NEXT, FILTER THE RESULTING SIGNAL USING THE SUBROUTINES
      C        SECOND SMOOTHING FILTER AND STORE THE RESULT IN ABUF.  ABUF
      C        NOW CONTAINS THE SMOOTHED AMPLITUDE ENVELOPE OF THE SPEECH
      C        INPUT SIGNAL.
      C
00250          CALL ENVLP(ABUF,NSAMT,ABUF,BR2,EAR2,BF2,EAF2,Y2)
      C
      C              FINALLY PLOT THE CONTENTS OF ABUF VS. ZBUF TO PRODUCE
      C        AN NSAMT LONG SEGMENT OF THE DESIRED DISPLAY.  THEN PHOTO-
      C        GRAPH THE DISPLAY (ONE EXPOSURE) AND REPEAT THE PROCESS ON
      C        THE NEXT DATA SLICE.
      C
00261          CALL DISSY(ABUF,ZBUF,NSAMT,ISCOP1,8000,512.,1024.,-1)
00272          CALL PHOTO(1)
00273   290    ISAMP = ISAMP + IDELT
      C
      C              IF THE DEBUGGING MODE IS BEING USED, PRINT OUT THE TIME
      C        SLICE INDEX AND THE CONTENTS OF ZBUF.
      C
00274          IF (SENSE SWITCH 2)     295, 299
00277   295    PRINT 296, I
00305   296    FORMAT (9H ITIME = ,I4,3X,7HOUTPUT.)
00305          PRINT 300, (ZBUF(J), J=0, NSAMT)
```

```
        CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
00317     299      CONTINUE
00320              RETURN
00321     300      FORMAT (2X,10(F9.2,1X))
00321     305      FORMAT (2X,10(F9.7,1X))
00321             END
--FORTRAN
```

Section 7

THREE DIMENSIONAL DISPLAY ROUTINES

The Three Dimensional Display package consists of a set of programs used to produce two dimensional displays with the third dimension represented by varying the displays intensity. The input for the display package is the data in FINT, which has previously been generated by one or more of the Speech Display routines. The output is a finished Polaroid photograph.

The package can be divided into three parts: NORMF, FINTPT and HIEMP which perform preparatory processing on the data before a display is actually generated; SPDISP, ENTER, and HEDBUF which produce the actual display and photograph it; and finally, the CSL system programs which are used as the direct interface to the CRT display unit. In order to provide a more complete description of the total Speech Display system, these latter programs will be described, even though they are technically part of the CSL CDC 1604 Operating System, since without them many of the operations performed in the other programs in the display package would not make sense. However, the listings will not be given since they are highly dependent on the particular hardware itself.

7.1 NORMF

This subroutine normalizes the data in the FINT array before it is used by SPDISP. Its basic operation begins by finding the maximum value in FINT within the limits of IFREQ, the number of frequency components being used, and ITIME, the number of time slices being used in the display. The subroutine then calculates a multiplicative factor which when multiplied by the maximum value will give a value close to the range of intensity values which can be displayed. Finally all of the values in FINT are multiplied by this value, thus giving a linear normalization of the data and a known maximum value.

There are two complications which are added to this basic algorithm. First of all, if sense switch 2 is up, the subroutine will print out the FINT array. This operation is actually performed by the FINTPT subroutine.

Secondly, the normalized maximum value can be varied by the operator. The maximum intensity value which can be used by the display is 255. If any value higher than this is used, ENTER, the program which generates the display buffer, will truncate it to 255. The normalized maximum value is controlled by making it equal to the product of 255 times the system variable OVFAC, whose value can be controlled by the operator.

The main reason for choosing a value other than 1.0 for OVFAC is to change the effective contrast of the display. In many of the displays under consideration, a very small percentage of the data points will have values much higher than the normal range. If the maximum value were made equal to 255, and the rest of the data were normalized linearly, almost all of the points would be considerably less than 255 and a significant proportion of the intensity range would contain only a few points. By letting the maximum normalized value be greater than 255 and then truncating any intensities over 255 to that value, we in effect spread out the lower and middle ranges at the cost of distorting a very few number of points in the display.

An alternative method would be to use a type of logrithmic normalization but this would be more complicated and take more time.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
             FINT(J) = FINT(J)*FACTOR
       59    CONTINUE
       60    CONTINUE
      C
      C           AFTER NORMALIZATION, PRINT OUT THE FINT ARRAY IF SENSE SWITCH
      C      2 IS UP.
      C
             CALL FINTPT
             RETURN
             END
--FORTRAN
```

7.2 FINTPT

This is a very short program used by NORMF to print out the contents of FINT. The printing is performed only if sense switch 2 is up. Note that sense switch 2 is rechecked after each frequency slice has been printed. This allows the operator to halt the printing even after it has started if he so desires.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
           SUBROUTINE FINTPT
      C
      C           THIS SUBROUTINE IS USED TO PRINT OUT THE CONTENTS OF THE
      C       FINT ARRAY PROVIDED THAT SENSE SWITCH 2 IS UP.  THE ARRAY IS
      C       PRINTED ONE FREQUENCY SLICE AT A TIME IN ORDER OF INCREASING
      C       FREQUENCY AND TIME WITH A SKIPPED LINE BETWEEN EACH FREQUENCY
      C       COMPONENT SLICE.
      C
      C       FINT  = FREQUENCY INTENSITY ARRAY.
      C       IFMAX = MAXIMUM FREQUENCY INDEX IN FINT.
      C       IFREQ = NUMBER OF FREQUENCY ENTRIES USED IN FINT.
      C       JEND  = ADDRESS OF ZEROTH ENTRY IN THE LAST TIME SLICE IN FINT.
      C       JSTOP = LAST ENTRY TO BE PRINTED IN A GIVEN FREQUENCY SLICE.
      C
              TITLE*
              JEND = (ITIME - 1)*IFMAX
 00005        DO 15, I=1, IFREQ
 00011        IF (SENSE SWITCH 2)    10, 20
 00013    10  JSTOP = I + JEND
 00015        PRINT 11, (FINT(J), J=I, JSTOP, IFMAX)
 00033        PRINT 12
 00036    11  FORMAT (2X,10(F8.2,2X))
 00036    12  FORMAT (/)
 00036    15  CONTINUE
 00037    20  RETURN
 00040       END
--FORTRAN
```

7.3 HIEMP

This subroutine is used to give high frequency emphasis to the normalized data in the FINT array, if desired. Its operation is very straightforward since it simply goes through the array and multiplies each entry by a factor whose magnitude depends on the frequency of the data. The emphasis begins with the entry in the array nearest the 2000 cps. level. The multiplicative factor begins with a value of 1 and increases uniformly at a rate of 12.5 x EMPFC per 1000 cps. The usual value of EMPFC is .15.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
          SUBROUTINE HIEMP
  C
  C         THIS SUBROUTINE TAKES THE CONTENTS OF FINT AND ADDS
  C      EMPHASIS TO THE HIGH FREQUENCY COMPONENTS.  THE HIGH FREQUENCY
  C      EMPHASIS BEGINS AT 2000 CPS. THE MULTIPLICATIVE FACTOR STARTS
  C      AT 1 AND IS INCREMENTED UNIFORMLY AT A RATE OF EMPFC*12.5 PER
  C      1000 CPS.
  C
  C      DELF  = COMMON VARIABLE.  DIFFERENCE (IN CPS.) BETWEEN FREQUENCY
  C              VALUES CORRESPONDING TO ADJACENT ENTRIES IN FINT.
  C      DLFAC = DELTA FACTOR BY WHICH FAC INCREASES FOR EACH FREQUENCY
  C              ENTRY IN FINT.
  C      EMPFC = COMMON VARIABLE.  INDICATES RATE OF INCREASE IN FAC.
  C      FAC   = MULTIPLICATIVE FACTOR FOR CURRENT FREQUENCY ENTRY.
  C      FINT  = COMMON VARIABLE.  TWO DIMENSIONAL FREQUENCY INTENSITY
  C              ARRAY.  IN ORDER TO BE MORE EFFICENT AND ALSO TO ALLOW
  C              THE DIMENSIONS TO BE CHANGED DYNAMICALLY, THE ADDRESS
  C              OF EACH ENTRY IN FINT IS CALCULATED DIRECTLY INSTEAD OF
  C              USING THE FORTRAN INDEXING.
  C      IFMAX = COMMON VARIABLE.  MAXIMUM NUMBER OF ENTRIES ALONG FREQUENCY
  C              DIMENSION OF FINT.  USED TO CALCULATE ADDRESSES IN FINT.
  C      INFRQ = RELATIVE LOCATION OF THE 2000 CPS. ENTRY IN A GIVEN TIME
  C              SLICE.
  C      ITIME = COMMON VARIABLE.  NUMBER OF TIME SLICES.
  C      JEND  = ADDRESS OF ZEROTH ENTRY IN LAST TIME SLICE OF FINT.
  C
         TITLE*
         INFRQ = 2000./DELF + .5
00004    DLFAC = EMPFC*25./(2000./DELF)
00010    FAC   = 1.0
00012    JEND  = (ITIME - 1) * IFMAX
00015    DO 200, I = INFRQ, IFREQ
00021    FAC = FAC + DLFAC
00022    DO 100, J = I, (JEND + I), IFMAX
00031    FINT(J) = FINT(J)*FAC
00032  100    CONTINUE
00036  200    CONTINUE
00037    RETURN
00040       END
--FORTRAN
```

7.4 SPDISP

This routine is used to produce a photograph of a variable intensity CRT display generated by using the common array, FINT, as a two dimensional array of intensities to be displayed. The SPDISP routine interpolates between the points to produce a smoothly varying display which it then photographs. Depending on the characteristics of the data in FINT, it may be necessary to break the picture into strips and take a multiple exposure photograph.

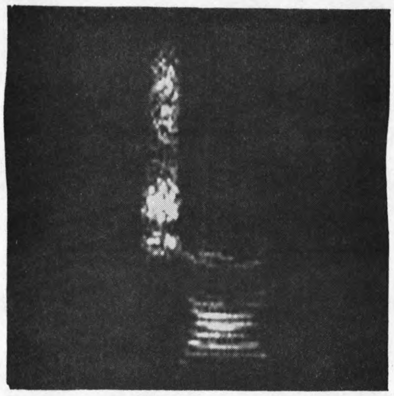
The floating point contents of FINT have been previously produced by one of the various speech display routines. The intensity values should be normalized to somewhere around 256.0 since that is the maximum display intensity and any point above this value will be truncated. In order to get a better spread of display values, it is sometimes desirable to have a few points over 256.0 (which then become truncated) and thus allow the smaller values to spread out more. This is especially true if, as is usually the case, the data contains a few samples relatively larger than the rest.

The characteristics of the display itself are determined by common variables whose values may be chosen by the operator. IDEL is a display variable and specifies the spacing of the points which will be displayed, i.e. how many positions apart each point will be (there are a total of 4096 x 4096 positions). IDELX specifies the number of display points which will be interpolated between each FINT entry in the X direction while IDELY specifies the same information for the Y direction. IXMAR indicates where the display of the data will begin relative to the left-most border of the CRT face. It is given in units of time slices, i.e. it tells how many time slices on the left hand side of the display will contain no information whatsoever. Finally, the operator can determine the minimum intensity value which will be displayed. By setting this value greater than zero, useless clutter can be eliminated with a consequent speed up in processing time. Figure 7.4.1 shows the effect of varying the size of this minimum intensity variable.

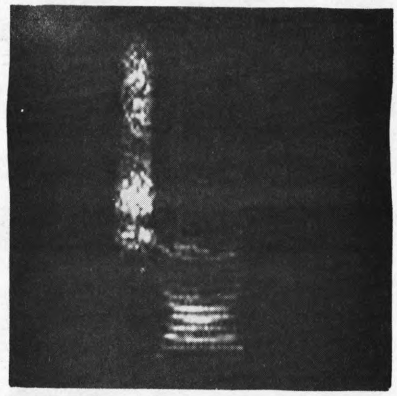
FINT is a two dimensional array with maximum dimensions of IFMAX x ITMAX. Because of the slowness of the CSL FORTRAN addressing algorithms and because time is very critical in SPDISP, most of the addressing in the inner loops of SPDISP is done in assembly language using as many shortcuts as possible. The number of entries in FINT which actually contain data is given by IFREQ for the frequency dimension and ITIME for the time dimension.

The subroutine begins by loading all of the constants which will be used by the program and checking the value of the various display parameters to be sure that they are within acceptable bounds. Note that the FI array is filled with the floating point representations of the integers from 0 to 32. This allows the low integer indexes of the internal loops to be converted to floating point by table lookup instead of calling the conversion routines. This puts a maximum limit on IDELX and IDELY of 33. If any parameter limits are exceeded a message will be typed out and the program will return.

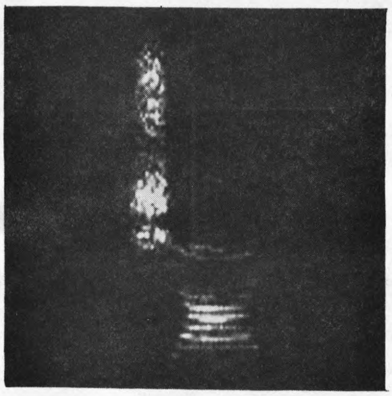
Next the display file is set up by calling the system subroutine ACTSCOPE. This call causes the display buffer and its pointer to be recorded in the appropriate system subroutines for future display purposes. Then SPDISP calls HEDBUF which initializes the display constants which will later be used by ENTER and also loads the initial two entries in the display buffer, ISCOPE, with the proper control words to begin a variable intensity TV-type display scan.



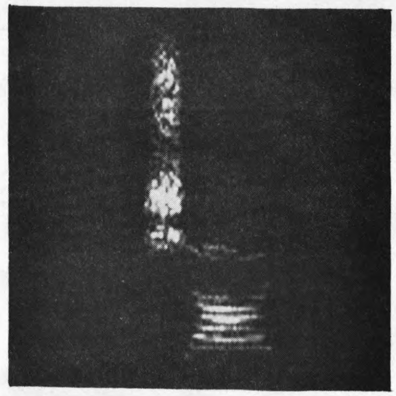
min. inten. = 1



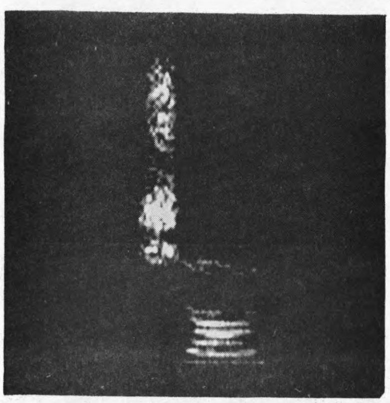
min. inten. = 20



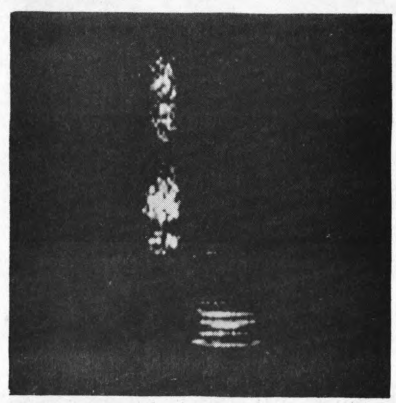
min. inten. = 40



min. inten. = 60



min. inten. = 100



min. inten. = 140

full intensity scale = 255

Figure 7.4.1 Effect of Increasing the Minimum Intensity Threshold on a Spectrogram of the Word "shod".

The interpolation itself basically works by first interpolating between frequency slices to determine the intensity value at each time slice and then interpolating along this new interpolated frequency line to get interpolated points between time slices. The interpolating begins from the lower left-hand corner of the display which represents low frequency and time values.

When this program was first written, this method seemed the most natural since the CRT display produces pictures in a left-to-right, bottom-to-top direction. However, since that time it has become evident that a considerable savings in storage could have been made by turning the display on its side, i.e. have the frequency displayed on the horizontal axis and time on the vertical. Since the actual display surface is square and since we are using photographs anyway, it would make no difference in the final display. In addition it would mean that the display could be produced as the data was generated rather than having to wait until all the data was generated before making the display. Unfortunately time has not been available for making the rather extensive changes which would be necessary in the programs involved.

In order to speed up the interpolation process, the differences between adjacent frequency entries all along the given frequency slice are calculated once before going into the actual interpolation process. These differences are divided by the number of interpolations which must be made in the Y direction between each frequency slice and are then stored in the DELTAY array for use in the interpolation of each line between the two frequency slices. The dimension of DELTAY must therefore be great enough to handle the maximum number of time slices which might occur. Currently this array is 400 words long and is stored at the end of the ISCOPE array. If ITIME is ever set greater than 400 an error comment will be given on the typewriter and the subroutine will return.

If the intensity of each interpolated point is determined to be greater than the selected minimum intensity which will be displayed, it is compared with the intensity of the last point which was entered into the display buffer. If the difference is greater than 1, the new point is entered into the display buffer by calling the subroutine ENTER. When the last data point has been interpolated, an intensity entry of 1 is produced so that the remaining part of the display has a uniform intensity. (A background intensity of 1 has been found to give more consistant contrast results in photographs than a zero intensity level.) Finally, a unit intensity entry is made with an X position of 0. This insures that the scan will complete the old line and begin the next line with a unit intensity background.

At the end of each line, a check is made to see how full the display buffer is. If it is not within less than 200 locations of the end of the buffer and if the program has not interpolated the last line of data, a new line will be started. If the buffer is within 200 locations of being full, or if the interpolation has been completed, the buffer will be photographed. In order to do this if the buffer has its last word only half full, the remaining half must be filled with all 1's. This code in effect keeps the CRT from interpreting the garbage in the last half of the word as a new X position. Then SPDISP takes a picture of the display, prints out the current line number, reinitializes the display buffer, and continues on to the next line. If the data has been exhausted,the subroutine returns.

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
      SUBROUTINE SPDISP
C         THIS SUBROUTINE TAKES AN IFREQ BY ITIME ARRAY OF INTENSITIES
C     AND CONVERTS IT TO A CONTINUOUS IMAGE CRT DISPLAY BY INTERPOLATING
C     BETWEEN THE POINTS IN 2 DIMENSIONS.  THE SUBROUTINE TAKES A PICTURE
C     OF THIS DISPLAY.  THE CORRECT IFREQ AND ITIME VALUES MUST BE ENTERED
C     INTO THE DIMENSION STATEMENT FOR FINT(IFREQ,ITIME), DELTAY(ITIME),
C     AND POINT(ITIME*IDELX) BEFORE THE PROGRAM IS COMPILED.
C
C         BPTNX = INTENSITY OF NEXT BASIC TIME POSITION ON CURRENT
C                 FREQUENCY LINE BEING PROCESSED.
C         BPTPV = INTENSITY OF PREVIOUS BASIC TIME POSITION ON CURRENT
C                 FREQUENCY LINE BEING PROCESSED.
C         DELTAX= INTENSITY INCREMENT BETWEEN 2 SPECIFIC TIME
C                 POSITIONS ON A GIVEN FREQUENCY LINE.
C         DELTAY= INTENSITY INCREMENTS BETWEEN FREQUENCY SLICES.  USED
C                 IN INTERPOLATING ACROSS THE FREQUENCY SLICE.
C                 ONE ENTRY FOR EACH TIME POSITION.
C         FI    = FLOATING POINT ARRAY USED TO HOLD CONSTANTS FROM
C                 1.0 TO THE MAXIMUM INDEX OF FI, IE. FI(I)=I.  THIS
C                 ALLOWS LOW INTEGERS TO BE CONVERTED TO FLOATING POINT
C                 BY TABLE LOOKUP INSTEAD OF CALLING FLOATF.
C         FIN   = COMMON VARIABLE USED TO TRANSMIT THE INTENSITY
C                 OF THE DISPLAY POINT TO THE ENTER SUBROUTINE.
C         FINT  = TWO DIMENSIONAL INPUT INTENSITY ARRAY.
C         I     = INDEX REGISTER 1.
C         IDEL  = SPACING BETWEEN DISPLAY POINTS ON CRT.
C         IDELX = NUMBER OF DISPLAY POINTS BETWEEN EACH TIME SAMPLE.
C         IDELY = NUMBER OF DISPLAY POINTS BETWEEN EACH FREQ. POSITION.
C         IFREQ = NUMBER OF FREQUENCY POSITIONS IN DISPLAY.
C         IFMAX = COMMON VARIABLE - MAXIMUM NUMBER OF ENTRIES ALONG
C                 FREQUENCY DIMENSION OF FINT.  USED TO CALCULATE
C                 DOUBLY SUBSCRIPTED ADDRESSES IN FINT.
C         IPTR  = POINTER TO NEXT UNUSED CELL IN DISPLAY BUFFER, ISCOPE.
C         ISCOPE= BUFFER USED BY ACTSCOPE TO CONTAIN INFORMATION FOR
C                 THE DISPLAY.
C         ITIME = NUMBER OF TIME SLICES IN DISPLAY.
C         IX    = COMMON VARIABLE USED TO TRANSMIT X POSITION OF
C                 THE DISPLAY POINT TO THE ENTER SUBROUTINE.
C         IXMAR = POSITION OF LEFT HAND MARGIN OF DATA RELATIVE TO
C                 X = 0 POSITION ON DISPLAY.
C         J     = INDEX REGISTER 2.
C         K     = INDEX REGISTER 3.
C         KEND  = MAXIMUM VALUE OF THE FREQUENCY INTERPOLATION INDEX.
C                 (MAY NOT BE GREATER THAN MAXIMUM SIZE OF FI ARRAY.)
C         LINE  = CURRENT LINE NUMBER BEING PRODUCED BY DISPLAY
C                 GENERATION PORTION OF PROGRAM.
C         L     = INDEX REGISTER 4
C         LEND  = MAXIMUM VALUE OF THE TIME INTERPOLATION INDEX.
C                 (MAY NOT BE GREATER THAN MAXIMUM SIZE OF FI ARRAY.)
C         M     = INDEX REGISTER 5.
C         N     = INDEX REGISTER 6.
C         OVFL  = COMMON VARIABLE USED TO KEEP TRACK OF THE NUM-
C                 BER OF INTENSITY POINTS EXCEEDING 256.
```

```
         CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
          C             POINT = INTENSITY OF POINT CURRENTLY BEING INTERPOLATED
          C                     ALONG A GIVEN FREQUENCY LINE BETWEEN THE INTENSITIES
          C                     AT THE TWO BASIC TIME POINTS ON THAT FREQUENCY LINE.
          C             PTLST = INTENSITY OF LAST POINT ENTERED INTO DISPLAY BUFFER.
          C
          C        SUBROUTINES NEEDED = DISPLAY ROUTINES, ENTER, REDBUF.
          C
                   TITLE*
                   DIMENSION DELTAY(400), FI(32)
                   EQUIVALENCE (DELTAY(0), A(4603))
                   DATA (MASK=7777777700000000B)
                   DATA (COMM=0000000077777777B)
          C
          C             SET UP CONSTANTS USED BY PROGRAM.
          C
              10   DELX = IDELX
         00003     DELY = IDELY
         00005     IEND = IFREQ - 1
         00007     KEND = IDELY - 1
         00011     LEND = IDELX - 1
         00013     OVFL = 0.0
          C
          C             CHECK THE VALUES OF THE SYSTEM VARIABLES TO SEE IF THEY
          C        ARE WITHIN THE CONSTRAINTS SET BY SPDISP.  IF NOT, TYPE A
          C        MESSAGE AND RETURN SO THE OPERATOR CAN MODIFY THEIR VALUES.
          C
         00015     IF ((IXMAR+ITIME)*IDELX*IDEL - 4095)      30, 20, 20
         00022  20 WRITE OUTPUT TAPE 19, 21
         00025  21 FORMAT (6H IXMAR)
         00025     GO TO 40
         00026  30 IF (IFREQ*IDELY*IDEL - 4095)     33, 33, 31
         00032  31 WRITE OUTPUT TAPE 19, 32
         00035  32 FORMAT (6H IFREQ)
         00035     GO TO 40
         00036  33 IF (ITIME - 400)    36, 36, 34
         00041  34 WRITE OUTPUT TAPE 19, 35
         00044  35 FORMAT (6H ITIME)
         00044     GO TO 40
         00045  36 IF (KEND - 32)    37, 37, 38
         00050  37 IF (LEND - 32)    45, 45, 38
         00053  38 WRITE OUTPUT TAPE 19, 39
         00056  39 FORMAT (6H INDEX)
         00056  40 WRITE OUTPUT TAPE 19, 41
         00061  41 FORMAT (1H+7X,13HIS TOO LARGE.)
         00061     GO TO 600
         00062  45 CONTINUE
         --ILLAR
          *        LOAD IFMAX INTO INSTRUCTIONS USED IN CALCULATING FINT INDEXES.
          *
                    LDA        IFMAX
                    SAU        LCC1
                    SAU        LCC2
         --FORTRAN
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
        C
        C        CALCULATE VALUES FOR THE INTERGER-TO-FLOATING POINT CONVERSION ARRAY.
        C
00064            DO 9, I=0, 32
00070        9   FI(I) = I
        C
        C            INITIALIZE THE DISPLAY FILE AND THE DISPLAY FILE CONSTANTS.
        C        THE ZEROTH DISPLAY LINE IS NOT USED, SO THE INITIAL DISPLAY
        C        LINE WILL START AT Y = 1*IDEL
        C
00073            IPTR = 0
00074            CALL ACTSCOPE(ISCOPE, IPTR, -1)
00100            CALL HEDELF(1.0,0,IDEL,ISCOPE)
00105            BPTPV      = 0.0
        C
        C            BEGIN INTERPOLATION, TAKING ONE FREQUENCY AT A TIME.
        C        I = FREQUENCY POSITION INDEX.  DISPLAY FILE IS BUILT UP ONE
        C        HORIZONTAL LINE AT A TIME WHILE INTERPOLATION IS GOING ON.
        C            THE 500 LOOP PROCESSES ONE FREQUENCY SLICE AT A TIME.
        C
00106            DO 500, I=1, IEND
        C
        C            CALCULATE Y INTENSITY INCREMENTS FOR NEW FREQUENCY
        C        SLICE.  J = TEMPORARY TIME SLICE INDEX.
        C
00112            M = I
00113            DO 100, J=1, ITIME
00120            N = M + 1
        --ILLAR
                  LDA    6  FINT               DELTAY(J)=(FINT(N)-FINT(M))/DELY
                  FSB    5  FINT
                  FDV       DELY
                  STA    2  DELTAY
        LOC1      INI    5  **                 INCREMENT M BY IFMAX.
        --FORTRAN
00124     100    CONTINUE
        C
        C            USING THESE Y INTENSITY INCREMENTS, INTERPOLATE A-
        C        CROSS THE FREQUENCY SLICE TO OBTAIN THE INTENSITY VALUES
        C        FOR THE BASIC TIME POINTS AT THE PARTICULAR FREQUENCY
        C        VALUE LINE BEING DISPLAYED.  THESE POINTS ARE CALCULATED
        C        ONE AT A TIME BEGINNING AT TIME SLICE 1.  TIME SLICE 0 IS
        C        ASSUMED TO BE 0 INTENSITY.  X DIRECTION INTERPOLATION BEGINS
        C        FROM TIME SLICE 0 TO TIME SLICE 1.  K=FREQUENCY INTERPOLATION
        C        INDEX.  J = TIME SLICE POSITION INDEX.
        C
        C
00125            DO 400, K=0, KEND
        C
        C            BPTPV IS SET TO ZERO TO INITIALIZE THE INTENSITY INTER-
        C        POLATION FOR THE NEW FREQUENCY VALUE LINE.
        C
00131            BPTPV = 0.0
```

```
00132          M = I
        C
        C         THE 300 LOOP PRODUCES ONE LINE FOR THE DISPLAY.
        C
00133          DO 300, J=1, ITIME
        C
        C              CALCULATE THE NEXT BASIC TIME SLICE INTENSITY VALUE, BPTNX,
        C         INTERPOLATING ACROSS THE FREQUENCY DIRECTION.
        C
        --ILLAR
                LDA     3  FI                  BPTNX=FI(K)*DELTAY(J) + FINT(M)
                FMU     2  DELTAY
                FAD     5  FINT
                STA        BPTNX
        *
        *          NOW INTERPOLATE BETWEEN THIS POINT AND THE PREVIOUS BASIC
        *            TIME SLICE INTENSITY VALUE, BPTPV.  L = TIME SLICE INTERPOLATION
        *            INDEX.  MAKE A NEW ENTRY IN ISCOPE EACH TIME THE NEWLY INTER-
        *            POLATED INTENSITY VALUE IS GREATER THAN THE MINIMUM INTENSITY
        *            ALLOWED, FINTM, PROVIDED THAT IT DIFFERS FROM THE PREVIOUS
        *            ENTRY IN ISCOPE BY MORE THAN 1 INTENSITY UNIT.
        *
        *          CALCULATE THE INTENSITY INCREMENT TO BE USED IN THE
        *         TIME SLICE INTERPOLATION.
        *
                FSB        BPTPV               DELTAX=(BPTNX-BPTPV)/DELX
                FDV        DELX
        LOC2    STA        DELTAX
                INI     5  **                  M = M + IFMAX
                ENI     4  0B                  DO 200, L=0, LEND
                LDA        LEND
                SAU        XX200
                AJP     3  XX210               IF LEND IS NEG, SKIP LOOP
        *
        *          BEGINNING OF THE INTERPOLATION LOOP.
        *
        START   BSS        0
                LDA     4  FI                  FIN=FI(L)*DELTAX+BPTPV
                FMU        DELTAX
                FAD        BPTPV
                STA        FIN
                FSB        FINTM               IF(FIN-FINTM) 130,140,140
                AJP     2  XX140
        *
        *          IF THE CURRENT INTENSITY VALUE IS BELOW FINTM, CHECK TO
        *         SEE IF THE PREVIOUS INTENSITY VALUE WAS 1.0, IE. THE BACKGROUND
        *         VALUE.  IF NOT, INSERT AN ENTRY OF THIS VALUE IN THE DISPLAY
        *         BUFFER IN ORDER TO DROP THE INTENSITY VALUE TO THE BACKGROUND
        *         LEVEL.
        *
        XX130   LDA        PTLST               IF(PTLST-1.0) 135,200,135
                FSB        =20014000000000000B
                AJP     0  XX200
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
         SLJ     0  XX135
 XX140   LDA        FIN                 DIF = FIN - PTLST
         FSB        PTLST
         AJP     0  XX200               IF(DIF) 142,200,141
         AJP     3  XX142
         AJP     2  XX141
 DELTAX  BSS        1E
 *
 *          IF THE INTENSITY INCREMENT SINCE THE LAST ENTRY IS GREATER
 *       THAN 1.0, MAKE A NEW ENTRY.
 *
 XX141   FSB        =20014000000000000B
         AJP     2  XX150               IF(DIF-1.0) 200,200,150
         SLJ     0  XX200
 XX142   FAD        =20014000000000000B
         AJP     3  XX150               IF(DIF+1.0) 150,200,200
         SLJ     0  XX200
 XX135   LDA        =20014000000000000B
         STA        FIN                 FIN = 1.0
 XX150   ENA     2  0E                  IX = ((J-1+IXMAR)*IDELX + L)*IDEL
         INA        77776B
         ADD        IXMAR
         MUI        IDELX
         INA     4  0E
         MUI        IDEL
         STA        IX
 +       ENI        0E                  NO OPERATION
         RTJ        ENTER               CALL ENTER
 XX200   ISK     4  **                  ADD. FIELD IS LOADED WITH CONTENTS OF LEND
         SLJ     0  START
 --FORTRAN
 C
 C          FINALLY LOAD THE PREVIOUS BASIC TIME SLICE INTENSITY
 C       VALUE VARIABLE, BPTPV. WITH THE CURRENT TIME SLICE INTENSITY
 C       AND THEN REPEAT THE LOOP.
 C
00151    210     BPTPV = BPTNX
00152    300     CONTINUE
 C
 C          AT THE END OF THE DATA GENERATE A ONE INTENSITY
 C       ENTRY SO THAT THE REMAINING PART OF THE LINE WILL CONTAIN A
 C       UNIFORM BACKGROUND INTENSITY.
 C
00153            IX  = (ITIME + IXMAR)*IDELX*IDEL
00156            FIN = 1.0
00160            CALL ENTER
 C
 C          TEST FOR END OF DISPLAY GROUP BY SEEING IF IPTR IS
 C       WITHIN 200 LOCATIONS OF THE END OF ISCOPE, OR IF K=KEND AND
 C       I=IEND (IE. THE END OF THE DISPLAY HAS BEEN REACHED).  IF
 C       NOT DONE, THEN CONTINUE.
 C
 --ILLAR
```

```
            LDA        IPTR                   IF(IPTR-4400)  315,320,320
            INA        67317B                 INCREMENT A REG. BY -4400
            AJP    2   XX320
    XX315   ENA    3   0E                     IF(K-KEND) 390,317,317
            SUB        KEND
            AJP    1   XX390
    XX317   ENA    1   0E                     IF(I-IEND) 390,320,320
            SUB        IEND
            AJP    1   XX390
    --FORTRAN
    C
    C             AT THE END OF A DISPLAY GROUP, INSERT A ZERO ENTRY TO ALLOW THE
    C          SCAN TO FINISH THE LINE.
    C
00165   320     IX = 0
00166           FIN = 0.0
00167           CALL ENTER
    --ILLAR
    *
    *                IF THE END OF THE DISPLAY GROUP HAS BEEN REACHED, FILL
    *            IN THE LAST BUFFER WORD, IF NECESSARY, WITH ALL ONES,
    *            DISPLAY THE BUFFER, AND TAKE A PICTURE OF IT.
    *
            LDA        IODD                   IF (IODD)   325, 340, 325
            AJP    0   XX340
            AJP    0   XX340
    XX325   LDQ        MASK                   MASK = 7777777700000000B
            LDA        COMM                   COMM = 0000000077777777B
            LIL    4   IPTR                   LOAD INDEX REG. 4 WITH IPTR
            ADL    4   ISCOPE                 ADD LOGICAL, IE. ISCOPE(IPTR) =
            STA    4   ISCOPE                  A REG .OR. (MASK .AND. ISCOPE(IPTR))
    --FORTRAN
00173           IPTR = IPTR + 1
00174   340     IF (SENSE SWITCH 2)      341, 343
00177   341     PRINT 342 (ISCOPE(I), I=0, IPTR)
00212   342     FORMAT (2X, 5(O16,4X))
00212   343     CALL PHOTO(1)
00213           CALL STOPSCOP
    C
    C           FINALLY PRINT OUT THE IPTR POSITION IN ISCOPE.
    C
00214           LINE = (I-1)*IDELY + K + 1
00220           PRINT 310, LINE, IPTR
00225   310     FORMAT (20H AT THE END OF LINE ,I4,1X,7HIPTR = ,I6,1H.)
    C
    C           RECALCULATE INITIAL CONSTANTS IN DISPLAY BUFFER AND CONTINUE.
    C
00225           ITMP = (LINE + 1)*IDEL
00230           CALL HEDBUF(1.0,0,ITMP,ISCOPE)
00235           GO TO 400
    C
    C             IF THE END OF THE DISPLAY BUFFER HAS NOT BEEN REACHED, CON-
    C           TINUE BY MAKING AN ENTRY WITH X=0 AND AN INTENSITY OF 1.  THIS
```

```
      CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
       C         LETS THE CRT SCAN KNOW THAT SUCCEEDING X POSITIONS REFER TO
       C         THE NEXT LINE.  IT ALSO STARTS THE NEW LINE OFF WITH AN IN-
       C         TENSITY OF 1.
       C
 00236  390      IX  = 0
 00237           FIN = 1.0
 00240           CALL ENTER
 00241  400      CONTINUE
 00242  500      CONTINUE
 00243           PRINT 505, OVFL
 00247  505      FORMAT (8H OVFL = , F10.0)
 00247  600      RETURN
 00250          END
--FORTRAN
```

7.5 ENTER

This subroutine is used by SPDISP to generate the buffer which controls the CRT display. It is one of the exceptions to the rule about not having parameters in the common area since it is called in a very tight loop in SPDISP and is used very often. Thus in order to save time the parameter values are saved in COMMON storage so that the tedious address loading operations are not necessary every time ENTER is called. This cuts down the number of operations both in the call itself and in the prologue of the subroutine.

In order to understand the operation of ENTER, it is necessary first to describe in a general way how the CRT display operates in the variable intensity, TV scan mode. From an initial position and intensity (which is generated by the subroutine HEDBUF) the display beam will scan to the right (increasing x addresses). The data in the display buffer is broken up into two 24-bit entries per 48-bit CDC 1604 word. Each entry specifies an x position in the 12 high order bits and an intensity in the 12 low order bits. The display continually compares its current x position as it scans with the x position in the next display buffer entry. When it makes a match, it changes the intensity to the value in the current display buffer entry and continues, using the next entry for comparison purposes.

Thus the purpose of ENTER is to generate an entry for the display buffer given an x position, IX, and an intensity value, FIN. The subroutine is only called for x positions where SPDISP has determined that a change in intensity will be necessary. This is determined by comparing the interpolated intensity values at each x position with the last value entered into the display buffer (PTLST) and only calling ENTER when the difference is 1.0 or greater.

When the subroutine is called, it first checks the value of FIN to see if it is greater than 255.0 (the maximum intensity which can be produced by the display). If it is, then ENTER will use an intensity of 255 provided that the previous entry was not also 255. If it was in fact 255, the subroutine returns. For each value over 255, the ENTER subroutine increments the common variable OVFL by 1.0. This variable is used by SPDISP to keep track of the number of points which have been truncated.

If an entry is added to the display, its floating point value is converted to fixed point and PTLST is updated to reflect the new intensity value. Then a combination of shifts and masked transfers are performed to produce a 24-bit display buffer entry in the A register.

At this point it is necessary to know into which half of the current display word to load the entry. For this purpose the variable IODD is used. If it is 0, the current entry is loaded into the left half of the current buffer entry. If it is 1, it is loaded into the right half. In the latter case the pointer to the current word, IPTR, is incremented. In either case the value of IODD is changed to its alternate value.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
          SUBROUTINE ENTER
     C
     C          THIS MIXED FORTRAN-ILLAR SUBROUTINE GENERATES A SINGLE
     C      ENTRY TO THE SCAN MODE DISPLAY FILE, BUF, AND UPDATES THE
     C      DISPLAY FILE CONSTANTS.  IF THE INTENSITY IS GREATER THAN
     C      255, IT IS PRINTED OUT AND THEN TRUNCATED TO 255 BEFORE BEING
     C      ENTERED IN THE FILE.  AFTER GENERATING, IN THE RIGHT HALF
     C      OF THE A REGISTER, A 24 BIT ENTRY CONSISTING OF A 12 BIT X
     C      POSITION ON THE LEFT AND AN 8 BIT INTENSITY RIGHT JUSTIFIED
     C      ON THE RIGHT, THE SUBROUTINE CHECKS THE IODD PARAMETER TO
     C      DETERMINE WHERE TO STORE THE ENTRY.  IF IODD=0, THE ENTRY IS
     C      LOADED INTO THE LEFT HALF OF BUF(IPTR).  IF IODD=1, THE
     C      ENTRY IS LOADED INTO THE RIGHT HALF OF BUF(IPTR), WHILE PRE-
     C      SERVING THE CONTENTS OF THE LEFT HALF OF THAT WORD, AND IPTR
     C      IS INCREMENTED BY 1.  IN BOTH CASES THE VALUE OF IODD IS RE-
     C      VERSED.
     C
     C      FIN    = INTENSITY OF NEW ENTRY IN DISP. BUFF.(FLT. PT.).
     C      PTLST  = INTENSITY OF PREVIOUS ENTRY IN DISPLAY BUFFER.
     C      INT    = INTENSITY OF NEW ENTRY IN DISPLAY BUFF.(FIXED).
     C      IODD   = 0 IF LEFT HALF OF CURRENT BUFF. WORD IS TO CONTAIN
     C               NEW ENTRY.  =1 IF RIGHT HALF OF CURRENT BUFFER WORD
     C               IS TO CONTAIN ENTRY.
     C      IPTR   = CURRENT WORD IN BUF TO BE LOADED WITH A NEW ENTRY.
     C      ISCOPE= BUFFER USED TO HOLD SCAN MODE DISPLAY BUFFER.  EACH
     C               WORD IN ISCOPE CONTAINS TWO 24-BIT DISPLAY ENTRIES.
     C      IX     = X POSITION OF NEW ENTRY IN DISP. BUFF.(FIXED).
     C      MASK1 = MASK USED TO LOAD RIGHT-MOST 12 BITS OF THE A REG.
     C               WITH THE INTENSITY VALUE OF THE CURRENT ENTRY.
     C      MASK2 = MASK USED TO LOAD LEFT-MOST 24 BITS OF THE A REG.
     C               WITH THE LEFT HALF OF BUF(IPTR) WHEN IODD=1.
     C
            TITLE*
     C
     C          CONVERT CURRENT INTENSITY TO FIXED POINT AND UPDATE FLIN.
     C      IF FIN IS GREATER THAN 255(THE MAXIMUM ALLOWED INTENSITY),
     C      TRUNCATE THE FIXED POINT INTENSITY VALUE TO 255.  THEN MAKE AN
     C      ENTRY ONLY IF THE PREVIOUS ENTRY WAS NOT 255.
     C
            IF (FIN - 255.)     70, 70, 60
00005    60 INT = 255
00006       OVFL = OVFL + 1
00011       IF (PTLST - 255.)     80, 200, 200
00014    70 INT = FIN
00015    80 PTLST = FIN
   --ILLAR
             LDA        IX                  LOAD 12 BIT X POSITION VALUE:
             ALS        12                  SHIFT X VALUE TO POSITIONS 24 THRU 12.
             LDQ        MASK1
             ADL        INT                 MASK OFF ALL BUT LOW ORDER 12 BITS OF
    *                                          INTENSITY AND ADD IT TO SHIFTED X VALUE
    *                                          TO GET A 24 BIT X VALUE-INTENSITY PAIR.
             LIL    1   IPTR                LOAD IR1 FOR USE IN ACCESSING BUF(IPTR).
```

```
CSL FORTRAN OF SEPT 1968,  DATE   6/28/71
          LDQ         IODD                  IF IODD = 0,
          QJP    0    LEFT                     LOAD PAIR INTO LEFT HALF OF WORD.
  RIGHT   LDQ         MASK2
          ADL    1    ISCOPE                ADD LEFT HALF OF ISCOPE(IPTR) TO A REG.
          STA    1    ISCOPE                RESTORE 2 ENTRIES IN A REG. TO ISCOPE(IPRT)
          RAO         IPTR                  INCREMENT IPTR BY 1.
          RSO         IODD                  DECREMENT IODD BY 1, IE. SET IT TO 0.
          SLJ    0    OUT                   UNCONDITIONAL JUMP TO OUT.
  LEFT    ALS         24                    SHIFT ENTRY TO LEFT HALF OF A REG.
          STA    1    ISCOPE                STORE ENTRY IN LEFT HALF OF ISCOPE(IPTR).
          RAO         IODD                  INCREMENT IODD BY 1, IE. SET IT TO 1.
          SLJ    0    OUT                   UNCONDITIONAL JUMP TO OUT.
  MASK1   OCT         0000000000001777
  MASK2   OCT         7777777700000000
  OUT     INI    0    0                     NO OPERATION.
  --FORTRAN
00021   200    RETURN
00022       END
--FORTRAN
```

7.6 HEDBUF (FI, IXX,IY,BUF)

This subroutine is used to produce the "heading" for a display buffer containing commands to the CRT for a variable intensity TV-type scan display. The subroutine has as parameters, the initial intensity, FI, the initial x position, IX, the initial y position, IY, and the buffer to be used by the CRT, BUF. These data items are then arranged in the order as shown in figure 7.6.1.

This subroutine work is extremely simple and consists mainly of shifting and masking in data to generate the first word of the buffer. The zeroth word is simply loaded with all 1's. This is essentially a command to the CRT which lets it know that a mode change command will be coming next.

Once the zeroth and first words have been loaded, HEDBUF sets IPIR = 2 and IODD = 0 to indicate to the ENTRY subroutine that the next entry is to be loaded in the left half of the 2nd word in the display buffer. PTLST, the last intensity loaded into the buffer, is set to the initial intensity specified in the mode change command word. Then the routine returns.

48		0
111	. . . .	111

zeroth word

48 45	44 36	35 24	23 12	11 0
1000	I_0	Δ	X_0	Y_0

lst. word

where I_0 = initial intensity

Δ = step size between display points

X_0 = initial X position

Y_0 = initial Y position

Figure 7.6.1 Format for the first 2 words of the display buffer when used in the variable intensity scanning mode

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
      SUBROUTINE HEDBUF(FI, IXX, IY, BUF)
C
C           THIS SUBROUTINE GENERATES THE ZEROTH AND FIRST WORDS IN
C        THE CRT DISPLAY BUFFER WHICH CONTAIN THE NECESSARY COMMANDS TO
C        START A SCAN MODE DISPLAY.  THE INITIAL INTENSITY, AND THE
C        INITIAL X AND Y POSITIONS, IX AND IY, ARE GIVEN AS PARAMETERS.
C        THE SUBROUTINE LOADS THE ZEROTH WORD OF BUF WITH ALL ONES, WHICH
C        PREPARES THE CRT DISPLAY TO ACCEPT A MODE CONTROL WORD.  THEN
C        IT USES MASKING AND SHIFTING TO CONSTRUCT IN THE A REGISTER, THE
C        PROPER MODE CONTROL WORD AND INITIAL SCAN POSITION.  THIS IS THEN
C        STORED IN THE FIRST ENTRY IN BUF.  THE SUBROUTINE ALSO INITIALIZES
C        PTLST, IODD, AND IPTR.
C
C        BUF   = DISPLAY BUFFER
C        FI    = INITIAL INTENSITY OF SCAN(FLOATING POINT).
C        IDEL  = SPACING BETWEEN POINTS IN DISPLAY.
C        INT   = INITIAL INTENSITY OF SCAN(FIXED POINT).
C        IODD  = INDICATOR FOR THE ENTER SUBROUTINE.
C              = 0 IF LEFT HALF OF CURRENT BUFFER WORD IS TO CONTAIN
C                NEW ENTRY,
C              = 1 IF RIGHT HALF OF CURRENT BUFFER WORD IS TO CONTAIN
C                NEW ENTRY.
C        IPTR  = CURRENT WORD IN BUFFER TO BE LOADED WITH A NEW ENTRY.
C        IXX   = INITIAL X POSITION OF SCAN.
C        IY    = INITIAL Y POSITION OF SCAN.
C        MASK1 = MASK USED TO LOAD THE 8 BIT INITIAL INTENSITY INTO THE A
C                REGISTER.
C        MASK2 = MASK USED TO LOAD 12 BIT DATA FIELDS INTO THE A
C                REGISTER.
C        PTLST = INTENSITY OF THE LAST PREVIOUS POINT TO BE ENTERED
C                INTO THE DISPLAY BUFFER.
C
         TITLE*
         DATA(ICON0=7777777777777777B)
         DATA(ICON1=0000000000004000B)
         DATA(MASK1=0000000000000377B)
         DATA(MASK2=0000000000007777B)
         DIMENSION BUF(10000)
         INT   = FI
0003     BUF(0) = ICON0
  --ILLAR
          LDA       ICON1          LOAD CODE FOR SCAN MODE DISP. IN LOW ORDER
  *                                   12 BITS OF A REGISTER.
          LDQ       MASK1
          ADL       INT            MASK IN INITIAL INTENSITY IN LOW ORDER 8 BITS
          ALS       12             SHIFT SCAN MODE CODE AND INTEN. LEFT 12 BITS.
          LDQ       MASK2
          ADL       IDEL           MASK IN IDEL IN LOW ORDER 12 BITS OF A REG.
          ALS       12
          ADL       IXX            MASK IN INITIAL X POSITION
          ALS       12
          ADL       IY             MASK IN INITIAL Y POSITION
          STA       TEMP
```

```
CSL FORTRAN OF SEPT 1968, DATE 6/28/71
 --FORTRAN
00013   100     BUF(1) = TEMP
00016           PTLST  = FI
00020           IODD   = 0
00021           IPTR   = 2
00022           RETURN
00023         END
--FORTRAN
```

7.7 DISPLAY

DISPLAY is a general purpose oscilloscope output routine for alphanumeric and/or graph plotting. It is a CSL ILLAR system program which was written by Donald Lee and Charles Arnold. Graphing features include automatic normalization of data points; automatic axis, hash, or grid plotting; point, line segment, or connected line modes; special matrix plotting; polar coordinates and features allowing multiple graphs.

The use of the DISPLAY subroutine has been directly incorporated into the CSL FORTRAN system by adding several new statements to the FORTRAN compiler to facilitate its use. For alphanumeric plotting, DISPLAY is controlled by a FORTRAN FORMAT statement and follows the conventions of the FORTRAN FORMAT. To control graphing options and modes, the FORMAT statement is also used in an extended form. The FORMAT statment number, the address of the buffer into which the display buffer is to be packed (which is an array in the user's program), the maximum length of the buffer, and the buffer pointer are transmitted to DISPLAY by the following statement:

```
DISPLAY 100 (BUFFER,BUFLING,IPOINT)
```

where 100 is the format statement number, BUFFER is the name of the array in the user's program into which the display buffer is packed, BUFLING is the maximum length of the buffer (usually an integer constant), and IPOINT is the name of an integer variable which is updated by DISPLAY and is equal to the length of the buffer. IPOINT is not to be changed by the user; however, it may be referenced to obtain the buffer length.

In order to transmit data which may be needed by the FORMAT statement (e.g. origin coordinates, maximum and minimum coordinate values, etc.) this statement can be written as:

```
DISPLAY 100 (BUFFER, BUFLING, IPOINT)/LIST
```

where LIST may be any conventional FORTRAN list format. Also signed and unsigned constants are recognized.

To transmit data to the display once the subroutine has been set up by one of the above statements, the following statement can be used:

DISPLAY/LIST

where LIST is again defined as above. In this case LIST will contain the data needed to specify the successive display points. This statement may be repeated several times or it may be part of a DO loop generating display data.

If the user wishes to change formats in the construction of a single display buffer, the following statement is used:

DISPLAY 100

or DISPLAY 100/LIST

where 100 is the format statement number. All graph options are reset to standard values by this statement. This statement may be necessary if more than one graph is being displayed at once or if the graphing mode is switched. Note that the buffer array, buffer length and buffer pointer need not and <u>must</u> not be used if the format is changed in the construction of a single buffer. This information <u>is</u> transmitted if an entirely new graph is desired.

Once the complete set of display data has been generated, the CRT can be initiated by the statement:

DISPLAY

In addition to controlling the input of alphanumeric information, the FORMAT statement (as it is specially extended in CSL FORTRAN) may be used to specify display options, and the placement, size, and intensity of alphanumeric information.

A graph mode display option must be given if a graph is desired, but all other options may be omitted if the standard parameters are desired. The display options are introduced by the character '*' (54B) in the FORMAT statement. For example:

```
100 FORMAT(7HEXAMPLE*GRAPHMODE,AXES)
```

indicates that there is an alphanumeric heading "EXAMPLE" and that the graphing mode is GRAFMODE and that axes are desired. All data points received by DISPLAY will be interpreted according to the GRAFMODE mode, unless the format is changed. Thus the number of points in the graph need not be specified.

Alphanumeric statements and display options may be mixed where necessary. The '*' character indicates that the interpretation is switched, for example:

```
100 FORMAT(*YORG*7HEXAMPLE*GRAFMODE).
```

The initial '*' indicates that the following information, up to the next '*' is display option information. The second '*' indicates a switch to alphanumeric format, and the third '*' switches back to display options. (YORG controls the Y direction orientation of alphanumeric information.) A restriction on the use of switching the format interpretation is that <u>no switch</u> may be made after the graph mode is set. In other words, the graph mode must be given in the last set of display options. Display options may be given in any order. However, those options requiring a parameter to be supplied must be in the same order as the parameter information, similar to the standard FORTRAN format.

In the CSL display system there are a variety of graph options available. For the purposes of this report, however, only the most commonly used (in the Speech Display system, at least) will be described. These include: GRAFMODE, LINEMODE, CLINEMOD, and CHARMODE.

In the GRAFMODE option, the data entries are interpreted as alternate X and Y entries. The values may be either floating point or integer entries, as specified by the format statement. The entries are normalized, converted into scope coordinates and packed into a buffer. The coordinates are displayed as points.

LINEMODE is the line segment mode. Data is entered in four element groups: X1, Y1, X2, Y2. This group defines a single vector on the scope. Data may be either floating point or integer, as specified in the FORMAT statement. The entries are normalized, converted into scope coordinates, and packed into the buffer. CLINEMOD is the continuous line mode and is the same as LINEMODE except that data is entered in two element groups each group defining a line with the previous group as its origin.

In the CHARMODE option, each data entry is treated as a BCD word. The program must insert its own carriage return characters (32b). The options XORG, YORG, SIZE, XS, S, M, and L can be used to specify the characters' placement and size.

In addition to the graph mode options, a variety of options are provided by DISPLAY to control intensity, letter size, portion of scope face containing graph, automatic axis, hash mark and grid plotting, values to be treated as maxima and minima, interpretation of data coordinates, integer or floating point mode, and others. No option need be given unless it varies from the standard parameters. Some of the most commonly used options which do not require the user to supply parameter information are:

I — Specifies that data is treated as integers. If not set, floating point is assumed.

AXES — Axes will be drawn through the point specified by XØ and YØ. If XØ and YØ are not given, the point is assumed to be (Ø,Ø). An axis or hash mark that would be off the screen is discarded.

HASH — Hash marks will be drawn on the axes. It is not necessary to specify AXES with HASH; the axes will automatically be plotted. To locate where the marks are plotted, n is calculated such that $10^{n} <$ maxvalue-minvalue $\leq 10^{n+1}$. Marks are then plotted at units of SETXHASH X $10^{n\;1}$ (or SETYHASH). If SETXHASH or SETYHASH is not given, the value is assumed to be 1. This results in from 11 to 100 hash marks on each axis.

POLAR — Indicates that data is to be treated as polar coordinates (r, theta). Data must be floating point. Cannot be used in the increment modes.

TRUNCATE — When XMAX, MIN, YMAX, or YMIN are specified (see below) data entries falling outside the specified range are set to the given maximum or minimum.

I1,I2,I3,I4 — Scope intensity level, initialized to I1. The intensity setting will control all intensities up to the next intensity setting. For example,

```
100    FORMAT(*I4*7HEXAMPLE*I3,AXES,I2,HASH,I1,GRAFMODE)
```

will cause the alphabetic information to be displayed at the highest intensity, I4, the axis at I3, the hashmarks at I2, and the graph at I1.

XS,S,M,L — Extra small, small, medium, large. Sets size of letters, initialized to M. The following example shows how two letter sizes can be specified in one format statement:

100 FORMAT(*5HSMALL/*L*5HLARGE)

Note that the letter size must precede the alphanumeric information that it is to govern. Once set, the letter size remains so unless another letter size option is encountered, or a new format is set.

The following options require the user to supply parameter information in a parameter list.

XMAX — Used to specify the first value of the maximum X value. If a data item is found that is larger than this value, the larger data item is used as the new maximum X value, unless TRUNCATE is set, in which case the larger value is ignored.

YMAX,XMIN,YMIN — Used like XMAX.

XØ,YØ — Specifies the origin of the AXES. Initialized to (Ø,Ø). The parameter data is integer or floating point as specified in the FORMAT statement.

SETXHASH,SETYHASH — Specifies the unit used in plotting hashmarks. See HASH. Parameter data is integer or floating point as specified in the FORMAT statement.

SETXMAX,SETXMIN SETYMAX,SETYMIN	Sets the boundaries on the scope face on to which the graph will be drawn. Initial values are 7777b,Ø,7777b,Ø. The parameter must be actual scope coordinates. These options are useful for plotting graphs side by side or separating a graph from alphabetic information.
XORG,YORG	Scope coordinates of the first letter of alphanumeric data. Initialized to Ø,7777b. XORG or YORG must precede the alphanumeric format that it is to govern in the format statement. For instance, 100 FORMAT(*XORG,YORG*7HEXAMPLE)
LENGTH	The number of characters per line, initialized to 60. The following table gives maximum line length corresponding to each letter size: XS - 133 S - 96 M - 64 L - 32

7.8 ACTSCOPE(BUFFER, LENGTH, MODE)

This subroutine activates the display buffer, BUFFER, of length, LENGTH. If MODE is positive, the buffer will be treated as a circular buffer and will be displayed continuously. If it is negative, it will be displayed once.

ACTSCOPE is used by the DISPLAY subroutine and can also be used by any other programs which construct their own display buffers from scratch (e.g. SPDISP).

When using ACTSCOPE to produce a continuous display, all system routine output to the printer will automatically turn the scope off before printing and turn it back on after printing a single line. This is necessary since the printer and CRT are on the same output channel.

7.9 STOPSCOP

This subroutine is used to halt the operation of the CRT. It must be used whenever a new graph is to be constructed in a buffer which is currently being used by the CRT, or in any case where the buffer may be written upon. Otherwise all hell will break loose since the display will continue to interpret whatever is in the buffer as display commands.

The subroutine itself is extremely simple since its main component is a single machine instruction which inactivates the CRT output channel.

7.10 PHOTO(N)

This subroutine is used to take a photograph of the current buffer being displayed by an ACTSCOPE call or of the last such buffer to be displayed if the display is currently off. The subroutine opens the camera shutter and exposes the display N times before closing it again.

7.11 WHATNOW

WHATNOW is an entry point to the DISPLAY subroutine which allows the operator to modify the display on line, photograph it, exit to the main system, or simply to look at the display before continuing in the program. When it is entered for the very first time, it produces the following comment on the typewriter:

type p,c,a,d,r,s,af,rf, or period.

*

At this time the operator may type one of these options and a carriage return and the respective operations as described below will be performed. Alternatively, the operator can type:

noret

and a carriage return. This will eliminate the need for carriage returns in all future use of the WHATNOW subroutine. Note that this option is only valid on the very first use of WHATNOW when the message is printed out. No matter what option is chosen on this first call, successive calls will not print the message.

The actions resulting from the various WHATNOW options are as follows:

. exit to the main system.

c continue to the next statement in the user's program after the call to WHATNOW.

p take a picture of the scope display. The scope camera must be on and ready.

a add a line of heading to the scope display. WHATNOW will type:

xorg =

The user then types a <u>four</u> digit number less than 4096, which will become the x coordinate of the lower left corner of the first word to be displayed.

WHATNOW will then type:

yorg =

and the user supplies the y coordinate of the lower left corner of the first word to be displayed. Again, a four digit number if required. WHATNOW will then type

xs,s,m,l-size =

to determine the letter size, extra small, small, medium or large. Simply type one of the letters. WHATNOW will now type a minus sign indicating it is ready for the line of heading. A full line consists of 120 xs, 96 s, 64 m, or 32 l sized letters. If a typing mistake is made, a space followed by a carriage return will delete the whole line and it may be retyped. Otherwise the line will appear on the screenafter making a carriage return. This line may be replaced or deleted by other directives. After the line is displayed, an asterisk will be typed and WHATNOW will await a new directive.

r This directive works just as 'a' does, except the line typed is used to replace the last line added to the display with an 'a' directive.

af This works just as 'a' does except that XORG and YORG are given as unnormalized data coordinates. If the data is floating point, the number is read in E8.2 format. If the data is integer, an I10 format is used.

rf This works just as 'r' except that XORG and YORG are specified as in 'af'.

d The last line added by a directive is deleted.

s All lines added by directives are deleted.

Section 8

SPEECH PROCESSING ROUTINES

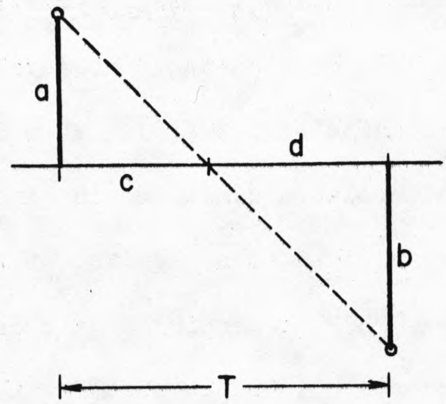
The Speech Processing Routines are a collection of subroutines used by the Speech Display routines to produce the various types of displays. They are usually somewhat general in nature and in many cases are used by several different speech displays. The idea is that they should be a type of universal speech processing building block.

8.1 ZCRPIC (BUF, ZERO, NUM, PBUF, FINDX, LIMIT, REM)

This subroutine finds the positions at which the data in the input buffer, BUF, crosses the zero level defined by ZERO. It scans the input data beginning with BUF(0). When it finds a zero crossing it calculates the interpolated distance (in time) from the last previous crossing and converts this to an equivalent frequency (in cps.) which is stored in the array PBUF, beginning with location 0. It also stores the position of the crossing in time (secs.) with respect to the beginning of the buffer, BUF, in the corresponding entry of the floating point array FINDX. After NUM intervals from BUF have been processed, the subroutine loads LIMIT with the number of zero crossings which have been detected and returns. Note that at this time REM will contain the distance (in time) from the last zero crossing to the end of the buffer, BUF.

The actual detection of the zero crossings is performed by calculating the distance of each point above the zero level and then taking the products of the pairs of endpoints for each interval. A zero crossing occurs whenever the signs of the pair are different and this will result in a product sign of "-". When it is determined that an interval contains a zero crossing, the actual interpolated crossing point is calculated using the equation derived from the calculations in figure 8.1.1.

A special situation occurs in the case of a point lying on the zero level line. In this case, the point will cause the two adjacent pairs to produce products of zero. The program has been written so that the initial pair will be considered to have a zero crossing at its second end point, but the second pair (and all subsequent pairs, if several



$$\frac{c}{d} = \frac{a}{b}$$

$$c = \frac{a}{b} \times d \ ; \quad c + d = T$$

$$c = \frac{a}{b} \times (T - c)$$

$$c \left(\frac{a}{b} + 1\right) = \frac{a}{b} \times T$$

$$c = \frac{\frac{T \times a}{b}}{1 + \frac{a}{b}} = \frac{T \times a}{b + a}$$

$$c = \frac{a \times T}{a + b}$$

Figure 8.1.1 Finding the Interpolated Position of a Zero Crossing Point zero points occur together) will be ignored. This implies two constraints:

1. Maxima or minima points occurring on the zero level line will be considered to be one zero crossing when in fact the signal did not really cross the line.
2. If the signal remains on the zero level for several successive points only one zero crossing will be counted.

Although these constraints are not the only possible ones, they are just as reasonable as any alternatives.

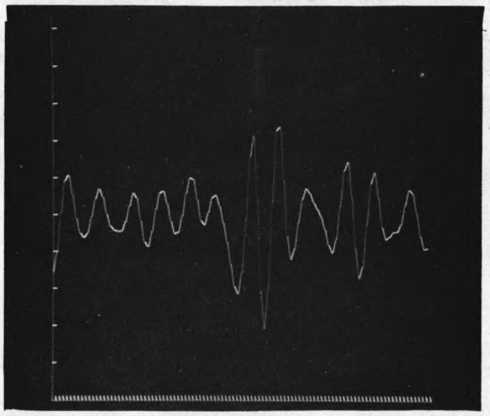
```
CSL FORTRAN OF SEPT 1968,  DATE  9/9/71
          SUBROUTINE ZCRFIC(BUF, ZERO, NUM, PBUF, FINDX, LIMIT, REM)
      C
      C         THIS SUBROUTINE IS USED TO EXTRACT THE ZERO CROSSING
      C      POINTS IN THE INPUT BUFFER, BUF, USING THE ZERO LEVEL DE-
      C      FINED BY THE INPUT PARAMETER, ZERO.  IT SCANS THE INPUT DATA
      C      BEGINNING WITH THE ZEROTH POSITION IN BUF, AND CALCULATES
      C      THE INTERPOLATED DISTANCE FROM THE LAST ZERO CROSSING.  THE
      C      EQUIVALENT INSTANTANEOUS FREQUENCY FROM THE PREVIOUS CROSSING
      C      TO THE PRESENT CROSSING IS STORED IN PBUF, WHILE THE FLOATING
      C      POINT POSITION OF THE CROSSING (IN NO. OF SAMPLES), WITH RESPECT
      C      TO THE TIME OF THE INITIAL POINT IN BUF IS STORED IN FINDX.
      C      PBUF AND FINDX ARE LOADED BEGINNING WITH LOCATION 0.  AFTER
      C      NUM INTERVALS HAVE BEEN PROCESSED, THE SUBROUTINE LOADS
      C      LIMIT WITH THE NUMBER OF ZERO CROSSINGS DETECTED AND RETURNS.
      C
             TITLE*
             DIMENSION BUF(1), PBUF(1), FINDX(1)
          10 J = 0
00003        FIR  = BUF(0) - ZERO
      C
      C         PROCESS NSAMT INTERVALS IN BUFF LOOKING FOR ZERO CROSSINGS.
      C
00010        DO 299, I=1, NUM
00014        SEC = BUF(I) - ZERO
      C
      C         CHECK THE SIGN OF THE PRODUCT OF FIR AND SEC TO SEE IF
      C      THE SIGNAL CROSSED THE ZERO LEVEL DURING THE CURRENT INTERVAL.
      C
00015        IF (FIR*SEC)    250, 240, 275
      C
      C         IF THE PRODUCT OF THE ENDPOINTS IS ZERO, ONE OF THEM MUST
      C      BE ZERO.  COUNT IT AS A ZERO CROSSING INTERVAL ONLY IF FIR IS
      C      NOT ZERO.  THIS ENSURES THAT A POINT TOUCHING THE ZERO LEVEL
      C      WILL ONLY BE COUNTED AS ONE ZERO CROSSING.
      C
00021    240 IF (FIR)        250, 275, 250
      C
      C         IF THE PRODUCT OF THE ENDPOINTS IS -, THE SIGNAL MUST HAVE
      C      CROSSED THE ZERO LEVEL.  THEREFORE CALCULATE THE DISTANCE OF THE
      C      CROSSING POINT FROM THE INITIAL ENDPOINT AND ADD THIS DISTANCE
      C      TO REM BEFORE CALCULATING THE FREQUENCY EQUIVALENT TO THE LENGTH
      C      OF TIME BETWEEN THIS CROSSING AND THE LAST PREVIOUS CROSSING.
      C
00023    250 RESIDUE = ABSF(SEC*DTIME/(FIR-SEC))
00030        PBUF(J) = REM + (DTIME - RESIDUE)
00034        FINDX(J) = I - RESIDUE/DTIME
00040        PBUF(J) = .5/PBUF(J)
00041        REM = RESIDUE
      C
      C         SET THE VALUE OF REM EQUAL TO THE DISTANCE OF THE ZERO CROS-
      C      SING FROM THE FINAL ENDPOINT OF THE INTERVAL.
      C
00043        J = J + 1
```

```
CSL FORTRAN OF SEPT 1968,  DATE  9/9/71
              GO TO 290
      C
      C            IF THE PRODUCT OF FIR AND SEC IS +, THERE IS NO ZERO CROSSING
      C       BETWEEN THE ENDPOINTS.  THEREFORE INCREMENT REM BY THE LENGTH
      C       OF ONE SAMPLING PERIOD.
      C
        275   REM = REM + DTIME
        290   FIR = SEC
        299   CONTINUE
              LIMIT = J
              RETURN
            END
--FORTRAN
```

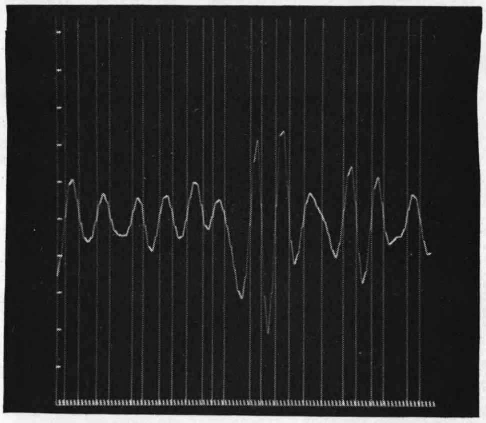
8.2 TZCRP

This program tests out the ZECPIC subroutine. It is used in conjunction with the TESTP program which gets the actual data and loads it into BUFF. Then TZCRP uses ZECPIC to process the data and prints out the results on the printer. It also displays the data on the CRT using cursor lines to indicate the zero crossing points.

The exact details concerning the use of TZCRP are given in the comment at the beginning of the source program. Figure 8.2.1 shows an example of its operation on a section of data.



Data before ZECPIC



Data with zero crossings picked by ZECPIC

Figure 8.2.1 Effect of ZECPIC on a Section of Speech Data

```
CSL FORTRAN OF SEPT 1968,  DATE  9/9/71
          PROGRAM TZCRP
      C
      C          THIS PROGRAM TESTS THE ZCRPIC SUBROUTINE.  IN ORDER TO LOAD
      C     THIS PROGRAM, TYPE OUT,
      C
      C          EXITCALL, PROGRAM TAPE, TESTP, TZCRP
      C
      C     THE SYSTEM WILL TYPE TAPE 5, IF THE COMPLETE LIBRARY IS NOT ON
      C     THE PROGRAM TAPE.  IN THIS CASE, TYPE A SPACE PROVIDED THE REST OF
      C     THE LIBRARY IS INDEED ON TAPE UNIT 5.
      C          ONCE THE PROGRAMS ARE LOADED, TYPE TESTP AND THEN USE THE
      C     MAIN PROGRAM TO SELECT A PORTION OF THE DATA TAPE TO BE RUN.  THEN
      C     EXIT FROM THE MAIN PROGRAM BY TYPING .  NEXT TYPE TZCRP TO BEGIN
      C     THE TEST.
      C          THE TEST CONSISTS OF DISPLAYING THE DATA WHICH WILL BE PRO-
      C     CESSED, PROCESSING IT, PRINTING OUT THE OUTPUT ARRAYS AND THEN
      C     DISPLAYING THE ORIGINAL DATA WITH CURSORS MARKING EACH DETECTED
      C     ZERO CROSSING.
      C
            TITLE*
            DIMENSION PBUF(1000), FINDX(1000)
            EQUIVALENCE (PBUF(1), A(7003)), (FINDX(1), A(9003))
      C
      C          BUFF CONTAINS THE DATA TO BE PROCESSED, WHILE NSAMT INDICATES
      C     HOW MANY POINTS ARE TO BE PROCESSED.
      C
            CALL ADJUS2(NSAMT, -1)
00004    10 XB   = 0.0
00005       FTMP = NSAMT
00006       CALL DISSY(XB,BUFF(ISAMPB),NSAMT,ISCOP1,4000,FTMP,1024.,0)
00023       CALL WHATNOW
      C
      C          AFTER DISPLAYING THE DATA, PROCESS IT BY CALLING ZCRPICT
      C
00024       CALL ZCRPIC(BUFF(ISAMPB),512.,NSAMT,PBUF,FINDX(1),LIM,REM)
00041       FINDX(0) = LIM
      C
      C          PRINT OUT THE OUTPUT ARRAYS.
      C
00045   100 PRINT 500, (PBUF(I), I=0, LIM)
00060       PRINT 501
00063       PRINT 500, (FINDX(I),I=0, LIM)
      C
      C          DISPLAY THE ZERO CROSSINGS ON THE CRT.
      C
00075       CALL DISSY(FINDX,BUFF(ISAMPB),NSAMT,ISCOP1,4000,FTMP,1024.,0)
00111       CALL WHATNOW
00112   500 FORMAT (2X,10(F9.2,1X))
00112   501 FORMAT (///)
00112       END
--FORTRAN
```

8.3 ORDER (PBUF, INDX, NUM, ISW1, ISW2)

This subroutine sorts the data in the input buffers PBUF and INDX. It is used primarily by FORMEX to sort formant candidates for processing, in which case PBUF contains floating point peak values and INDX contains the corresponding integer index positions of these peaks. The two buffers are always sorted together, i.e. the moving operations are performed on both corresponding entries in the two buffers. ISW1 determines which buffer will be used in determining the new order and ISW2 determines whether it will be sorted high-to-low or low-to-high. NUM indicates the number of entries in PBUF and INDX which are to be sorted.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
        SUBROUTINE ORDER(PBUF, INDX, NUM, ISW1, ISW2)
C
C           THIS SUBROUTINE SORTS THE DATA IN THE INPUT BUFFERS, PBUF
C       AND INDX.  PBUF IS USUALLY AN ARRAY OF FLOATING POINT AMPLITUDES,
C       WHILE INDX IS AN ARRAY OF FIXED POINT INDEXES.  THE SORTING IS
C       DONE ON EITHER PBUF OR INDX ACCORDING TO THE STATE OF ISW1, WITH
C       THE CORRESPONDING ENTRIES OF THE UNCHOSEN ARRAY BEING SORTED SO
C       THAT THEY MAINTAIN THEIR POSITIONS WITH RESPECT TO THE SORTED
C       ARRAY.  ISW2 DETERMINES WHETHER THE SORT WILL ARRANGE THE DATA
C       FROM HIGH TO LOW OR LOW TO HIGH.
C
C       ISW1  = 0 IF PBUF IS TO BE SORTED.
C             = 1 IF INDX IS TO BE SORTED.
C       ISW2  = 0 IF SORT IS HIGH-TO-LOW.
C             = 1 IF SORT IS LOW-TO-HIGH.
C
        DIMENSION PBUF(1), INDX(1)
        DO 100, I=1,(NUM - 1)
00007   DO 100, J=(I + 1), NUM
00014   IF (ISW1)       100, 40, 30
00016 30    IF (INDX(I) - INDX(J))    50, 100, 45
00021 40    IF (PBUF(I) - PBUF(J))    50, 100, 45
00024 45    IF (ISW2)       100, 100,  55
00026 50    IF (ISW2)       100,  55, 100
00030 55    TMP     = PBUF(J)
00031       PBUF(J) = PBUF(I)
00032       PBUF(I) = TMP
00033       ITMP    = INDX(J)
00034       INDX(J) = INDX(I)
00035       INDX(I) = ITMP
00036 100   CONTINUE
00040       RETURN
00041     END
--FORTRAN
```

8.4 PITPIC (BUF, ZERO, NUMP, PBUF1, INDX3, LIMIT)

The algorithm for this routine is based on a pitch extracting device developed at Northeastern University by L. O. Dolansky [1955, 1965, 1966]. It utilizes a filter having a fast rise time and a slow fall time. A block diagram of the process is given in figure 8.4.1.

The process begins by half-wave rectifying the speech input. The rectified signal is then used as the input to the filter which will follow the rapid initial "attack" of a pitch period and then slowly drop during the rest of the period. Figure 8.4.2 shows the filter output which results in this example. By applying a peak picking algorithm to the filter output, the initial starting time of the pitch period can be detected to within a few samples (there is a slight delay due to the filtering process, but this delay is virtually a constant).

Unfortunately there is a small problem which can occur if the secondary peaks of a pitch period are relatively large (as is quite often the case). If this happens, there may be several peaks in the filter output at the beginning of each pitch period as shown in figure 8.4.3. Dolansky's analog device removed this problem to a certain degree by critically adjusting the rise and fall times so that the secondary peaks would not affect the output. However, this is a tricky task and was not entirely effective. By making use of the digital computer's versatility, it was possible to eliminate these additional peaks in the present program simply by calculating the pitch period which would have resulted from using it as a primary peak. In the case of any unreasonable pitch frequency, the second peak could be eliminated with confidence.

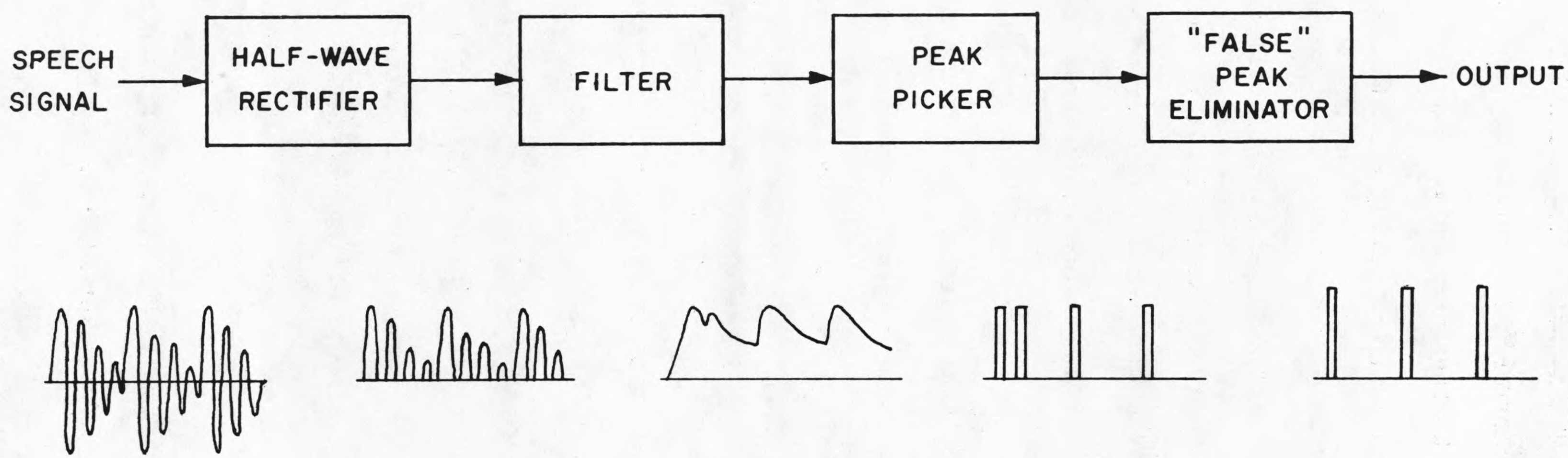


Figure 8.4.1 Block Diagram for PITPIC

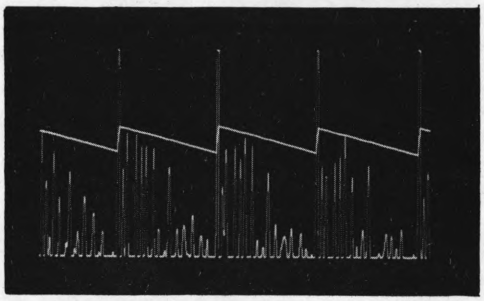


Figure 8.4.2 Peak Picking Filter Output

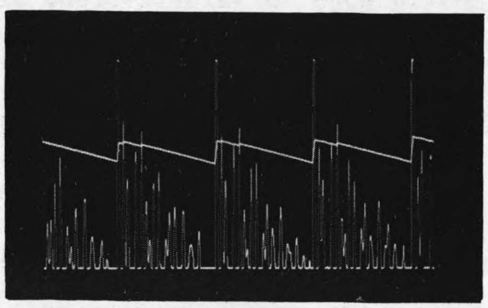


Figure 8.4.3 Peak Picking Filter Output
in the Case of Large Secondary Peaks

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
      SUBROUTINE PITPIC(BUF, ZERO, NUMP, PBUF1, INDX3, LIMIT)
C
C           THIS SUBROUTINE IS USED TO EXTRACT THE PITCH PEAK
C       POSITIONS IN THE INPUT BUFFER BUF.  IT SCANS THE INPUT
C       DATA BEGINNING WITH ENTRY 0, AND FINDS THE APPROXIMATE
C       POSITION OF THE INITIAL PEAK IN EACH PITCH PERIOD.  THE
C       POSITION IS APPROXIMATE(IT MAY BE OFF BY ONE OR TWO
C       INDEX POINTS), BECAUSE A FILTERING TECHNIQUE IS USED
C       WHICH PRODUCES A SHORT DELAY IN THE RESPONCE OF THE SUB-
C       ROUTINE.
C           FOR EACH PITCH PEAK DISCOVERED, THE EQUIVALENT IN-
C       STANTANEOUS PITCH FREQUENCY FROM THE PREVIOUS PITCH
C       PEAK TO THE CURRENT PEAK IS STORED IN THE CORRESPONDING
C       ENTRY IN PBUF WHILE THE INDEX POSITION OF THE CURRENT
C       PEAK IS STORED IN INDX.  PBUF AND INDX ARE BOTH LOADED
C       BEGINNING WITH 0.
C           AFTER PROCESSING NUM DATA POINTS, THE SUBROUTINE
C       LOADS THE NUMBER OF PITCH PEAKS FOUND INTO LIMIT AND
C       RETURNS.
C
C       BF     = FALL TIME CONSTANT CALCULATED AND USED BY FILTERING PROGRAM.
C       BR     = RISE TIME CONSTANT CALCULATED AND USED BY FILTERING PROGRAM.
C       BUF    = PARAMETER.  INPUT DATA BUFFER.
C       EAF    = FALL TIME CONSTANT CALCULATED AND USED BY FILTERING PROGRAM.
C       EAR    = RISE TIME CONSTANT CALCULATED AND USED BY FILTERING PROGRAM.
C       ENVLP  = SUBROUTINE.  FILTERING PROGRAM.
C       F1     = INSTANTANEOUS PITCH FREQUENCY PRODUCED BY TWO SUCCESSIVE PEAKS.
C       FQMAX  = MAXIMUM ALLOWABLE PITCH FREQUENCY.  IF TWO SUCCESSIVE PEAKS
C                ARE FOUND WHICH PRODUCE A HIGHER PITCH FREQUENCY, ONE OF
C                THEM MUST BE ELIMINATED.
C       IBEG   = LOCATION IN BUF OF INITIAL PITCH PEAK.
C       IEND   = LOCATION IN BUF OF FINAL PITCH PEAK.
C       IFILT  = SUBROUTINE.  INITIALIZATION ENTRY POINT FOR FILTERING PROG.
C       INDX3  = PARAMETER.  OUTPUT DATA BUFFER - CONTAINS LOCATION OF INITIAL
C                POINT OF EACH PITCH PERIOD.
C       INVL   = NUMBER OF INTERVALS BETWEEN PITCH PEAKS IN BUF.
C       LIM3   = INDEX WITHIN INDX3 ARRAY OF THE LAST PEAK FOUND IN THE
C                OUTPUT OF THE FILTERING PROGRAM. (ONE LESS THAN THE ACTUAL
C                NUMBER OF PEAKS FOUND).
C       LIMIT  = PARAMETER(OUTPUT).
C       NUM    = NUMBER OF DATA POINTS IN BUF - SET EQUAL TO NUMP FOR INCREASED
C                EFFICENCY OF OPERATION.
C       NUMP   = PARAMETER.  NUMBER OF DATA POINTS IN BUF.
C       PBUF1  = PARAMETER. OUTPUT DATA BUFFER - CONTAINS  INSTANTANEOUS
C                PITCH FREQUENCY FOR EACH PITCH PERIOD.
C       PFREQ  = AVERAGE PITCH FREQUENCY OVER BUF.
C       SAMF   = COMMON VARIABLE.  SAMPLING FREQUENCY IN FLOATING POINT.
C       Y      =
C       ZERO   = PARAMETER. ZERO LEVEL OF DATA - TO BE USED BY RECTIF.
C
      DIMENSION BUF(1), PBUF1(1), INDX3(1)
      TITLE*
      NUM = NUMP
```

```
      CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
        C
        C             INITIALIZE FILTER CONSTANTS.
        C
00003    21    CALL IFILT(.1, NUM, 20., BR, EAR, BF, EAF, Y)
00014          FTMP = NUM
00015          CALL RECTIF(BUF, ZERO, NUM, PBUF1)
00023          CALL ENVLF(PBUF1, NUM, PBUF1, BR, EAR, BF, EAF, Y)
00034          CALL PEAPIC(PBUF1, NUM, PBUF1, INDX3, LIM3)
00042          PRINT 30, (PBUF1(I), I=0, LIM3)
00054          PRINT 31, (INDX3(I), I=0, LIM3)
00066    30    FORMAT (2X,10(F9.2,1X))
00066    31    FORMAT (2X,10(I9,1X))
00066    42    IBEG = INDX3(1)
        C
        C             NEXT TRY TO ELIMINATE FALSE PEAKS DUE TO SECONDARY PEAKS.
        C      ASSUME THAT THE PEAK AT INDX3(1) IS IN FACT A VALID PEAK.
        C      THEN PROCEED BY CALCULATING THE EFFECTIVE PITCH FREQUENCY BE-
        C      TWEEN EACH REMAINING PEAK AND ITS NEAREST NEIGHBOR EARLIER IN
        C      TIME.  IF ANY PITCH FREQUENCY THUS CALCULATED IS TOO HIGH, E-
        C      LIMINATE THE ASSOCIATED PEAK.
        C
00071          FQMAX = 500.
00073          DO 50, J=2, LIM3
00077          F1 = SAMF/(INDX3(J)-INDX3(J-1))
00107          IF (FQMAX - F1)      45, 45, 50
00112    45    PBUF1(J) = 0.0
00113    50    CONTINUE
        C
        C             NOW ELIMINATE ALL ZERO-VALUE PEAKS, INCLUDING THE ZERO
        C      VALUE PEAK LOADED INTO ENTRY 0 OF PBUF1 BY THE PEAPIC SUB-
        C      ROUTINE.
        C
00114          K = 0
00115          DO 60, J=0, LIM3
00122          IF (PBUF1(J))        55, 60, 55
00124    55    PBUF1(K) = PBUF1(J)
00125          INDX3(K) = INDX3(J)
00126          K = K + 1
00130    60    CONTINUE
00131          LIM3 = K - 1
00132          IEND = INDX3(LIM3)
00136          INVL = LIM3 - 1
00140          IF (INVL)      75, 75, 78
00143    75    PFREQ = 0.0
00144          GO TO 19
00145    78    PFREQ = FLOATF(ISAMF*INVL)/(IEND - IBEG)
00155    19    PRINT 20, PFREQ, IBEG, IEND, LIM3, INVL
00165    20    FORMAT (2X,F12.2,7(I10,2X))
00165          PRINT 31, (INDX3(I), I=0, LIM3)
00177          LIMIT = LIM3
00200          RETURN
00201       END
--FORTRAN
```

8.5 FFTB(NP, N, N2POWT, T, IFLAG)

This program is the descendant of a fast Fourier transform program originally written by Gary Horlick at the Coordinated Science Laboratory. It is a version of the original Cooley-Tukey algorithm which has been much discussed in the literature (see for example, Cooley and Tukey[1965], Gentleman and Sande[1966], Cochran, et al.[1967], or Brigham and Morrow[1967]). Since the purpose of the present program has generally been to analyze real data, FFTB has a provision for using the process described in Cooley[1966] which allows the program to analyze twice as much real data with the same time and storage requirements.

The subroutine utilizes two COMMON arrays, X and Y, as the real and imaginary components, respectively, of the intermediate processing array and also as the final output. The parameter N specifies the number of complex coefficients which will be used in X and Y in the calculation. This number controls the number of output frequency samples and the spacing between them, and must be a power of two.

The IFLAG parameter indicates which of the input options is being used. If IFLAG = 0, the data is assumed to be purely real with NP data samples stored in T. In this case NP cannot be more than 2 times N. If it is, the additional samples in T are ignored. In this option the odd samples are loaded into the real processing array X and the even samples are loaded into the imaginary processing array Y. If T contains less than 2N points, the unfilled points in X and Y are loaded with zeros. Then normal processing is performed as if X and Y contained N complex input samples.

At the end of the processing, in order to get the coefficients for the real data, as opposed to the artificial data which was constructed in the X and Y arrays, another transformation must be performed. The derivation for this transformation and its FORTRAN implementation are given in Appendix A.

If IFLAG = 1, the data is assumed to be complex and already in the X and Y processing arrays. In this case there is no loading from T and no reconversion afterwards. This option is used when it is desired to perform a transform on complex data and at a later time perform the inverse (perhaps after some modification of the coefficients) transform. An inverse FFT can be performed by dividing the magnitude of the complex coefficients by N and reversing the sign of the imaginary components before calling FFTB.

The FFTB subroutine calculates the complex coefficients by means of two separate loops. The first consists of the radix 4 passes while the second consists of a radix 2 pass (if the number of points is not evenly divisible by 4). In each loop note that the sine and cosine values are determined for each iteration by means of the trigonometric identities for the sum of two angles (the previous angle plus the iteration increment). This is much faster than calling the sine and/or cosine routine on each iteration. A further speedup is achieved by using the simplified forms of the loop equations on the first iteration when the cosine terms are +1 and the sine terms are 0.

In calculating the output coefficients, the Cooley-Tukey algorithm scrambles the order of the coefficients. Therefore the next step after the calculation of the complex coefficients is to unscramble them on the basis of a binary sort. The limits for the various sorting loops are calculated on the basis of the number of output coefficients which were calculated. The FFTB subroutine contains 13 loops which allows for output data fields containing up to 2^{13} coefficients. If less than this number are used, the outermost loops will only be performed once. Note that the fact that J1, J2, J3, and J4 (the first four loop indices in the sorting section) are index registers is of no significance in this part of the loop and occurs only because these same variables were previously used in the radix 4 calculation of the complex coefficients.

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
      SUBROUTINE FFTB(NP, N, N2POWT, T, IFLAG)
C
C         THIS SUBROUTINE CALCULATES THE COMPLEX FOURIER COEFFICENTS
C      FOR THE INPUT ARRAY, T.  IT USES ARRAYS X AND Y AS PROCESSING
C      ARRAYS AND LEAVES THE RESULTS IN THESE ARRAYS WHEN DONE.  THE
C      ZEROTH COMPLEX COEFFICENT(CORRESPONDING TO THE DC COMPONENT) IS
C      STORED IN X(1) AND Y(1), THE FIRST FUNDAMENTAL FREQUENCY COEF-
C      FICENT, IN X(2) AND Y(2), ETC.  THE PROGRAM RETURNS THE COEF-
C      FICENTS AS REAL (X) AND IMAGINARY (Y) COMPONENTS.
C         THE METHOD USED IS A VARIENT OF THE COOLEY-TUKEY FAST
C      FOURIER TRANSFORM WHICH USES BOTH RADIX FOUR AND RADIX TWO
C      CALCULATIONS.
C
C      C1        = COSINE FUNCTION USED FOR COMPLEX MULTIPLICATIONS IN THE
C                  RADIX 4 LOOP AND IN THE FINAL STEP FOR RECONSTRUCTION
C                  OF THE FOURIER COEFFICENTS.
C      C2        = COSINE OF TWICE THE ARGUMENT OF C1.
C      C3        = COSINE OF THREE TIMES THE ARGUMENT OF C1.
C      FI1,FI2,FI3,FI4 = IMAGINARY COMPONENTS OF INTERMEDIATE VARI-
C                  ABLES USED IN THE RADIX 4 AND RADIX 2 CALCULATIONS
C                  AND ALSO IN THE FINAL STEP FOR RECONSTRUCTING THE
C                  FOURIER COEFFICENTS.
C      FR1,FR2,FR3,FR4 = REAL COMPONENTS CORRESPONDING TO FI1, ETC.
C      IPASS     = ITERATION VARIABLE FOR RADIX 4 LOOP.
C      ISQLOC = ITERATION VARIABLE USED WITHIN RADIX 4 LOOP TO OBTAIN
C                  SUBSETS OF 4 COMPLEX VARIABLES EACH, WHICH ARE THEN
C                  PROCESSED TO OBTAIN THE NEXT ITERATION VALUES FOR THESE
C                  SAME COMPLEX VARIABLES.
C      J1,J2,J3,J4 = NAMES FOR INDEX REGISTERS 3,4,5,AND 6 RESPECTIVELY.
C                  THESE VARIABLES ARE USED TO ALLOW THE COMPILER TO USE
C                  INDEXED INSTRUCTIONS WHEN ACCESSING DATA IN THE RADIX
C                  4 AND RADIX 2 LOOPS INSTEAD OF GOING THRU THE LABORI-
C                  OUS PROCESS OF CALCULATING AN ADDRESS AND THEN USING
C                  INDIRECT ADDRESSING.
C      LENGTH =
C      N         = NUMBER OF COMPLEX COEFFICENTS(MUST BE A POWER OF 2).
C      NP        = NUMBER OF DATA POINTS IN T TO BE ANALYZED(MUST NOT
C                  BE GREATER THAN 2*N.  IF SO, THE ADDITIONAL SAMPLES
C                  WILL BE IGNORED).
C      NXTLTH
C      N2POW     = LOG TO THE BASE 2 OF N.
C      N4POW     = N2POW/2. IE. THE NUMBER OF TIMES 4 IS A FACTOR OF N.
C      S1        = SINE OF SAME ARGUMENT AS C1.
C      S2        = SINE OF TWICE THE ARGUMENT OF C1.
C      S3        = SINE OF THREE TIMES THE ARGUMENT OF C1.
C      T         = ARRAY CONTAINING INPUT SAMPLES.
C      X         = REAL COMPONENTS OF COMPLEX PROCESSING ARRAY (FOR MAX.
C                  EFFICENCY SHOULD BE AN ARRAY INTERNAL TO FFTB OR IN
C                  COMMON AREA).
C      Y         = IMAGINARY COMPONENTS OF COMPLEX PROCESSING ARRAY(FOR
C                  MAXIMUM EFFICENCY SHOULD BE INTERNAL TO FFTB OR IN
C                  COMMON AREA).
C      DC1,DS1= COSINE AND SINE, RESPECTIVELY, OF THE ANGLE INCREMENT
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
 C                    IN THE LOOP USED IN THE RECONSTRUCTION OF THE FOURIER
 C                    COEFFICENTS.  THEY ARE USED TO CALCULATE A NEW SINE
 C                    AND COSINE FOR EACH ITERATION WITHOUT HAVING TO CALL
 C                    COSF AND SINF EACH TIME.
 C
        TITLE*
        DIMENSION T(1),LL(13)
 C
 C              THIS EQUIVALENCE STATEMENT IS USED BY THE BINARY SORT LOOP
 C         BEGINNING AT STATEMENT 90 AND ENDING ON STATEMENT 600.  IT
 C         ENABLES THE SORT PARAMETERS TO BE REFERENCED AS AN ARRAY WHEN
 C         BEING CALCULATED BY THE 99 LOOP INSTEAD OF HAVING TO BE CALCU-
 C         LATED INDIVIDUALLY.
 C
        EQUIVALENCE (L13,LL( 1)),(L12,LL( 2)),(L11,LL( 3)),
      1             (L10,LL( 4)),(L9, LL( 5)),(L8, LL( 6)),
      2             (L7, LL( 7)),(L6, LL( 8)),(L5, LL( 9)),
      3             (L4, LL(10)),(L3, LL(11)),(L2, LL(12)),
      4             (L1, LL(13))
 C
 C              THIS STATEMENT CAUSES THE VARIABLES J1, J2, J3, AND J4
 C       TO BE ASSIGNED TO INDEX REGISTERS.
 C
        INDEX J1(3), J2(4), J3(5), J4(6)
 C
 C              IF IFLAG = 0, THE DATA IS ALL REAL AND IS IN T.  THERE-
 C         FORE, LOAD THE ODD POINTS OF THE INPUT ARRAY, T,
 C         INTO THE REAL COMPONENTS OF THE PROCESSING ARRAY AND THE EVEN
 C         POINTS INTO THE IMAGINARY COMPONENTS OF THE PROCESSING ARRAY.
 C         THIS ALLOWS THE ANALYSIS OF 2N REAL POINTS USING ONLY N COM-
 C         PLEX STORAGE LOCATIONS.  IF THE NUMBER OF VALID DATA SAMPLES
 C         IS LESS THAN 2N, THE PROCESSING ARRAY ENTRIES CORRESPONDING TO
 C         THE UNUSED SAMPLES ARE INITIALLY SET TO ZERO.
 C
        N2POW = N2POWT
00003   IF (IFLAG)    45, 10, 45
00005   10  JTMP = (NP/2) + 1
00011       DO 20 J=JTMP, N
00015       X(J) = 0.0
00016       Y(J) = 0.0
00017   20  CONTINUE
00020       JTMP = (NP + 1)/2
00024       DO 30 J=1, JTMP
00030       X(J) = T(2*J - 1)
00035   30  CONTINUE
00037       JTMP = NP/2
00041       DO 40 J=1, JTMP
00046       Y(J) = T(2*J)
00053   40  CONTINUE
 C
 C              CHECK TO SEE IF THE NUMBER OF POINTS TO BE ANALYZED
 C         HAS ANY MULTIPLES OF 4.
 C
```

```
CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
00054     45  NTHPOW = 2**N2POW
00056         N4POW  = N2POW/2
00061            IF (N4POW)     50, 63, 50
          C
          C         THE 80 LOOP CONSISTS OF THE RADIX 4 PASSES, IF ANY.
          C
00064     50  DO 80 IPASS=1, N4POW
00067         NXTLTH = 2**(N2POW - 2*IPASS)
00074         LENGTH = 4*NXTLTH
00076         SCALE  = 6.2831853/FLOATF(LENGTH)
00102         DC1 = COSF(SCALE)
00104         DS1 = SINF(SCALE)
00107         C1   = 1.0
00111         S1   = 0.0
          C
00112         DO 79  J=1, NXTLTH
          C
00116         DO 61  ISOLOC=LENGTH, NTHPOW, LENGTH
          C
          C         FIRST CALCULATE INDEXES TO BE USED FOR THIS ITERATION.
          C
00121         J1 = ISOLOC - LENGTH + J
00123         J2 = J1 + NXTLTH
00126         J3 = J2 + NXTLTH
00130         J4 = J3 + NXTLTH
          C
          C         NEXT CALCULATE THE REAL AND IMAGINARY COMPONENTS OF THE TEM-
          C         PORARY VARIABLES NEEDED IN THE COMPLEX MULTIPLICATIONS FOR
          C         THIS ITERATION.
          C
00132         FR1 = X(J1) + X(J3)
00133         FR2 = X(J1) - X(J3)
00135         FR3 = X(J2) + X(J4)
00137         FR4 = X(J2) - X(J4)
00141         FI1 = Y(J1) + Y(J3)
00143         FI2 = Y(J1) - Y(J3)
00145         FI3 = Y(J2) + Y(J4)
00147         FI4 = Y(J2) - Y(J4)
          C
          C         NOW CALCULATE THE NEW X AND Y VALUES.  IF J=1, NOTE THAT
          C         SIN(ARG)=0 AND COS(ARG)=1 AND THUS THE SIMPLIFIED EQUATIONS
          C         CAN BE USED.
          C
00151         X(J1) = FR1 + FR3
00153         Y(J1) = FI1 + FI3
00155         IF (J-1)     64, 62, 64
00161     64  TEM1   = FI2 - FR4
00162         TEM2   = FR2 + FI4
00164         X(J3) = TEM1*S1 + TEM2*C1
00171         Y(J3) = TEM1*C1 - TEM2*S1
00176         TEM1   = FI1 - FI3
00200         TEM2   = FR1 - FR3
00202         X(J2) = TEM1*S2 + TEM2*C2
```

```
        CSL FORTRAN OF SEPT 1968,  DATE   8/12/71
00207          Y(J2) = TEM1*C2 - TEM2*S2
00214          TEM1  = FI2 + FR4
00216          TEM2  = FR2 - FI4
00220          X(J4) = TEM1*S3 + TEM2*C3
00225          Y(J4) = TEM1*C3 - TEM2*S3
00232          GO TO 61
        C
00234   62     X(J3) = FR2 + FI4
00235          Y(J3) = FI2 - FR4
00237          X(J2) = FR1 - FR3
00241          Y(J2) = FI1 - FI3
00243          X(J4) = FR2 - FI4
00245          Y(J4) = FR4 + FI2
00247   61     CONTINUE
        C
        C        CALCULATE THE SINE AND COSINE FUNCTIONS NEEDED FOR THE NEXT
        C        ITERATION OF THE LOOP.
        C
00253          TC1 = C1
00254          C1  = TC1*DC1 -  S1*DS1
00260          S1  =  S1*DC1 + TC1*DS1
00265          C2  = C1*C1 - S1*S1
00272          S2  = C1*S1 + C1*S1
00277          C3  = C1*C2 - S1*S2
00304          S3  = C2*S1 + S2*C1
00311     79   CONTINUE
00313     80   CONTINUE
        C
        C              NEXT CHECK TO SEE IF N4POW WAS TRUNCATED WHEN IT WAS
        C        CALCULATED FROM N2POW/2.  IF SO, ONE ADDITIONAL RADIX 2
        C        PASS MUST BE PERFORMED.
        C
00315     83   IF (N2POW - 2*N4POW)    85, 90, 85
        C
00321     85   DO 89 J=1, NTHPOW, 2
00324          J1 = J + 1
00326          FR1 = X(J) + X(J1)
00327          FR2 = X(J) - X(J1)
00331          FI1 = Y(J) + Y(J1)
00333          FI2 = Y(J) - Y(J1)
00335          X(J)   = FR1
00337          Y(J)   = FI1
00340          X(J1)  = FR2
00341          Y(J1)  = FI2
00342     89   CONTINUE
        C
        C        AT THIS POINT WE HAVE OBTAINED THE FOURIER COEFFICENTS FOR
        C        THE ORIGINAL CONTENTS OF THE COMPLEX PROCESSING ARRAY.  HOWEVER,
        C        THEY HAVE BECOME SCRAMBLED DURING THE CALCULATIONS AND MUST
        C        THEREFORE BE SORTED OUT.
        C
        C      SET UP PARAMETERS FOR SORT
        C
```

```
      CSL FORTRAN OF SEPT 1968,  DATE  8/12/71
00345     90  DO 99 J=1, 13
00351         LL(J) = 1
00352         IF (J - N2POW)     95, 95, 99
00355     95  LL(J) = 2**(N2POW + 1 - J)
00362     99  CONTINUE
      C
      C       NOTE EQUIVALENCE OF LI AND L(14-I)
      C       BINARY SORT
      C
00364         I=1
00365         DO 600 J1 =  1, L1
00372         DO 600 J2 = J1, L2, I1
00376         DO 600 J3 = J2, L3, I2
00402         DO 600 J4 = J3, L4, I3
00406         DO 600 J5 = J4, L5, I4
00411         DO 600 J6 = J5, L6, I5
00414         DO 600 J7 = J6, L7, I6
00417         DO 600 J8 = J7, L8, I7
00422         DO 600 J9 = J8, L9, I8
00425         DO 600 J10= J9,L10, I9
00430         DO 601 J11=J10,L11,L10
00433         DO 601 J12=J11,L12,L11
00436         DO 601 J  =J12,L13,L12
      C
00441         IF (I-J)      610, 610, 601
00445    610  FR1  = X(I)
00446         X(I) = X(J)
00447         X(J) = FR1
00450         FI1  = Y(I)
00451         Y(I) = Y(J)
00452         Y(J) = FI1
00453    601  I = I+1
00466    600  CONTINUE
      C
      C         NOW WE HAVE THE COEFFICENTS IN THEIR PROPER ORDER.  HOWEVER,
      C         IF WE WERE PROCESSING PURE REAL DATA TO BEGIN WITH, WE SAVED
      C         TIME AND SPACE BY LOADING ALTERNATE SAMPLES INTO THE REAL AND
      C         IMAGINARY PARTS OF THE COMPLEX PROCESSING ARRAY.  IF THIS IS
      C         ACTUALLY THE CASE, THEN IN ORDER TO GET THE COEFFICENTS FOR
      C         THE REAL DATA, AS OPPOSED TO THE
      C         ARTIFICIALLY CONSTRUCTED COMPLEX DATA WITH WHICH WE STARTED,
      C         IT IS NECESSARY TO PERFORM ONE LAST TRANSFORMATION ON THE CO-
      C         EFFICENTS.
      C
00522         IF (IFLAG)     120, 110, 120
00524    110  ARG = 3.1415927/ FLOATF(N)
00527         DC1 =  COSF(ARG)
00531         DS1 = -SINF(ARG)
00535         C1  = 1.0
00536         S1  = 0.0
00537         N2 = N/2
00541         N2P1 = N2 + 1
      C
```

CSL FORTRAN OF SEPT 1968, DATE 8/12/71

```
00543         DO 115 J=2, N2F1
00550         J1 = N + 2 - J
00553         FR1 = X(J) + X(J1)
00555         FI1 = Y(J) - Y(J1)
00557         TC1 = C1
00561         C1  =  C1*DC1 - S1*DS1
00565         S1  = TC1*DS1 + S1*DC1
00572         FR2 = X(J) - X(J1)
00574         FI2 = Y(J) + Y(J1)
00576         FR3 = C1*FR2 - S1*FI2
00603         FI3 = C1*FI2 + S1*FR2
00610         Y(J) = 0.5*(FI1-FR3)
00614         X(J) = 0.5*(FR1+FI3)
00617         IF (J - N2F1)    113, 115, 113
00622   113   Y(J1) = 0.5*( FI1+FR3)
00625         X(J1) = 0.5*( FR1-FI3)
00630   115   CONTINUE
00631   120   CONTINUE
00631         END
--FORTRAN
```

8.6 IFILT(BUF, NUM, OBUF, BR, EART, BF, EAFT, Y1)

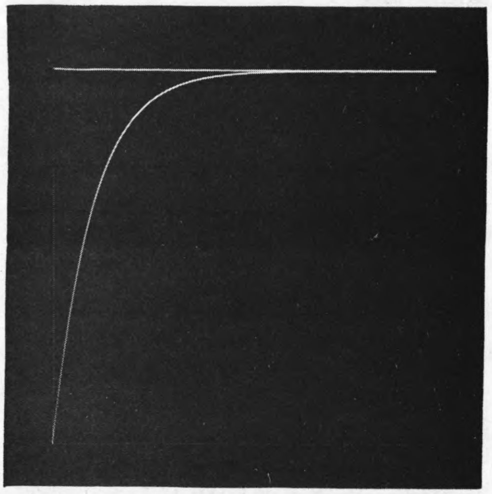
This subroutine simulates a digital filter with independently specifiable rise and fall times. The data from the input buffer, BUF, is filtered and loaded into the output buffer, OBUF (which may be the same physical buffer if desired). There are two entry points, IFILT, which sets up the filter constants and ENVLP, which does the actual filtering. When IFILT is called, BUF(0) will contain the desired rise time in msec. and OBUF(0) will contain the desired fall time. IFILT then calculates the values of BR, EART, BF and EAFT which are the coefficients for the difference equations to be used in the case of rising and falling signals, respectively. When ENVLP is called, NUM samples from BUF are filtered and loaded into OBUF.

The difference equations for the filter were chosen to be of the following form:

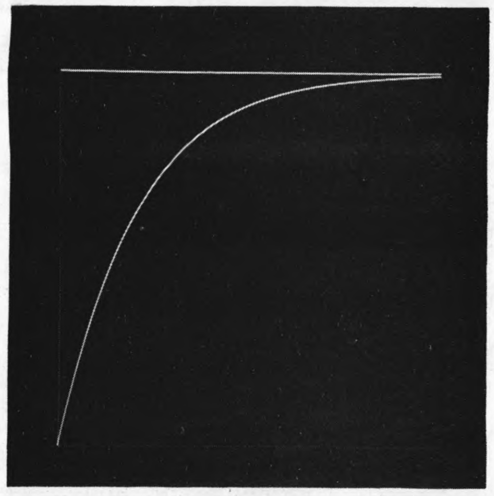
$$y(nT) = [1 - e^{-aT}]\, x(nT) + e^{-aT} y(nT - T)$$

with the time constant, a, being independently specified for a rising and a falling signal. The subroutine compares the input value with the last previous output value and then chooses the "rising signal" equation or the "falling signal" equation on the basis of this comparison.

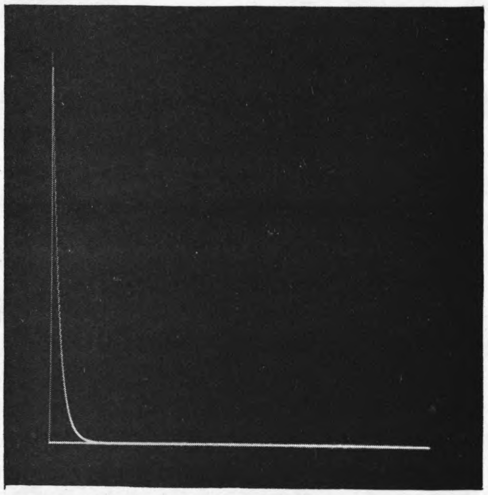
Figure 8.6.1 shows the step response of the filter for various rise and fall time constants.



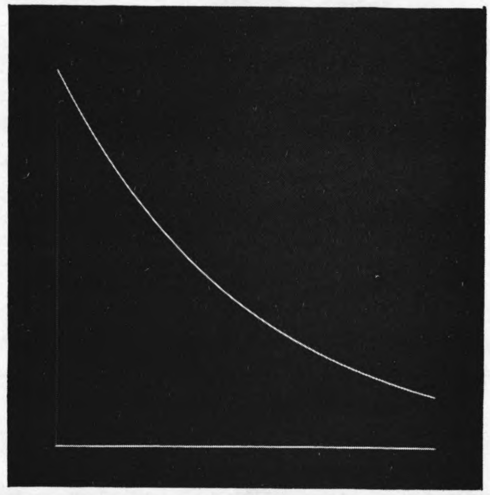
rise time = 5 ms



rise time = 10 ms



fall time = 1 ms



fall time = 25 ms

full time scale = 50 ms

Figure 8.6.1 Step Response of ENVLP
for Various Rise and Fall Time Constants

```
CSL FORTRAN OF SEPT 1968,  DATE   6/28/71
      SUBROUTINE IFILT(BUF,NUM,OBUF,BR,EART,BF,EAFT,Y1)
C
C           THIS SUBROUTINE IS A DIGITAL FILTER WHICH IS USED TO
C       SMOOTH OUT THE DATA IN THE INPUT BUFFER, BUF.  IT CAN BE
C       UTILIZED AS AN ENVELOPE TRACER AND CAN BE ADJUSTED TO HAVE
C       DIFFERENT RISE AND FALL TIMES.
C           IN ORDER TO CALCULATE THE FILTER VALUES TO BE USED,
C       IFILT MUST BE CALLED WITH BUF = RISE TIME AND OBUF = FALL
C       TIME IN MILLESECONDS OF THE DESIRED FILTER.  THEN DATA MAY
C       BE PROCESSED BY CALLING ENVLP WITH BUF = INPUT BUFFER AND
C       OBUF = OUTPUT BUFFER.
C           NOTE. THIS ROUTINE WILL STILL WORK PROPERLY WITH
C       THE BUF AND OBUF PARAMETERS BOTH INDICATING THE SAME BUFFER.
C
C       AR   = RISE TIME CONSTANT
C       AF   = FALL TIME CONSTANT
C       BF   = INPUT SAMPLE COEFFICENT FOR FALLING SIGNAL
C       BR   = INPUT SAMPLE COEFFICENT FOR RISING SIGNAL
C       BUF  = INPUT DATA BUFFER
C       DTIME= COMMON VARIABLE.  TIME PERIOD BETWEEN INPUT DATA SAMPLES
C       NUM  = NUMBER OF POINTS IN BUF
C       OBUF = OUTPUT DATA BUFFER
C       Y1   = PREVIOUS SAMPLE VALUE.
C
        TITLE*
        DIMENSION BUF(1), OBUF(1)
       10   AR   = 1000./BUF
00003       AF   = 1000./OBUF
00005       EART = 1.0/EXPF(AR*DTIME)
00013       EAFT = 1.0/EXPF(AF*DTIME)
00020       BR   = 1 - EART
00022       BF   = 1 - EAFT
      C
00024       RETURN
      C
      C
00025  40   ENTRY ENVLP
      C
00027       DO 300, I=0, (NUM-1)
00034       IF (BUF(I) - Y1)    50, 100, 100
      C
      C          FALLING SIGNAL
      C
00037  50   OBUF(I) = BF*BUF(I) + EAFT*Y1
00043       GO TO 200
      C
      C          RISING SIGNAL
      C
00045 100   OBUF(I) = BR*BUF(I) + EART*Y1
      C
00051 200   Y1 = OBUF(I)
00053 300   CONTINUE
00054       RETURN
00055     END
--FORTRAN
```

8.7 TENVL

This is a test program for IFILT. It allows the operator to select various values for the rise and fall times of the filter and then produces positive and negative steps to test out the program. The directions for the use of the program are given in the comment at the beginning of the program listing. Examples of TENVL's output are given in figure 8.6.1 in the previous section.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
          PROGRAM TENVL
   C
   C          THIS ROUTINE IS USED TO TEST THE ENVLP DIGITAL FILTERING
   C      SUBROUTINE.  IT SETS THE RISE AND FALL TIMES FOR THE FILTER
   C      ACCORDING TO NUMBERS TYPED IN ON THE TYPEWRITTER AND THEN TESTS
   C      IT OUT WITH A + AND - STEP INPUT OF MAGNITUDE 1000.
   C          IN ORDER TO LOAD THIS PROGRAM, TYPE OUT,
   C
   C                EXITCALL, PROGRAM TAPE, TENVL
   C
   C      THE SYSTEM WILL TYPE TAPE 5, IF THE COMPLETE LIBRARY IS NOT ON
   C      THE PROGRAM TAPE.  IN THIS CASE TYPE A SPACE PROVIDED THE REST
   C      OF THE LIBRARY IS INDEED ON TAPE UNIT 5.
   C          ONCE THE PROGRAM IS LOADED, TYPE TENVL TO START THE PROGRAM.
   C      THE PROGRAM WILL TYPE A + AFTER WHICH THE OPERATOR MUST TYPE,
   C
   C                TENVL=RISE TIME(IN MSEC.),FALL TIME(IN MSEC.)
   C
   C      THE PROGRAM WILL THEN CONTINUE, PRINT OUT THE RESULTS, AND DIS-
   C      PLAY THEM ON THE CRT.
   C
   C      BF     = FALL TIME COEFFICENT CALCULATED BY IFILT AND USED BY ENVLP.
   C      BR     = RISE TIME COEFFICENT CALCULATED BY IFILT AND USED BY ENVLP.
   C      EAFT   = FALL TIME COEFFICENT CALCULATED BY IFILT AND USED BY ENVLP.
   C      EART   = RISE TIME COEFFICENT CALCULATED BY IFILT AND USED BY ENVLP.
   C      F1     = ALPHABETIC PARAMETER READ IN BY INPTCM.  NOT USED BY TENVL.
   C      FTMP1 = TEMPORARY VARIABLE USED TO CONVERT N1 TO FLT. PT.
   C      FTMP2 = TEMPORARY VARIABLE USED TO CONVERT N2 TO FLT. PT.
   C      N1     = FIRST NUMERIC PARAMETER READ IN BY INPTCM.  RISE TIME IN
   C               MSEC. FOR FILTER (INTEGER).
   C      N2     = SECOND NUMERIC PARAMETER READ IN BY INPTCM.  FALL TIME IN
   C               MSEC. FOR FILTER (INTEGER).
   C      NVAL   = STEP VALUE.
   C      OBUF   = OUTPUT DATA BUFFER.
   C      TBUF   = INPUT DATA BUFFER.
   C      VAR    = INPUT COMMAND READ IN BY INPTCM (5 CHARACTER NAME).
   C      Y      = LAST PREVIOUS OUTPUT VALUE.
   C
          TITLE*
          DIMENSION TBUF(1000), OBUF(1000)
          EQUIVALENCE (TBUF(1), A(6003)), (OBUF(1), A(7004))
   C
   C          INITIALIZE VARIABLES AND CONSTANTS.
   C
          CALL INITI
 00003    X = 0.0
   C
   C          READ IN AND CHECK THE COMMAND FROM THE TYPEWRITTER.
   C
 00004  2 CALL INPTCM(VAR, N1, N2, F1, 19)
 00012    IF (VAR - 5HTENVL)     2, 5, 2
   C
   C          INITIALIZE THE FILTER ACCORDING TO THE FREQUENCY RANGE
```

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
C         GIVEN BY THE INPUT PARAMETERS.
C
      5   FTMP1 = N1
          FTMP2 = N2
          CALL IFILT(FTMP1,1000,FTMP2,BR,EART,BF,EAFT,Y)
          NVAL   = 1000
          Y      =     0.0
          TBUF(0)=     0.0
     10   DO 15, I=1, 1000
     15   TBUF(I) = NVAL
          CALL ENVLP(TBUF, 1000, OBUF,BR,EART,BF,EAFT,Y)
C
C              PRINT OUT AND DISPLAY INPUT TO FILTER.
C
          PRINT 100, (TBUF(I), I=0, 1000)
          CALL DISSY(X,TBUF,1000,ISCOP1,3000,1000..1024.,0)
     40   CALL WHATNOW
C
C              DISPLAY AND PRINT OUT OUTPUT FROM FILTER.
C
          CALL DISSY(X,OBUF,1000,ISCOP1,3000,1000..1024.,0)
          CALL WHATNOW
          PRINT 100, (OBUF(I), I=0, 1000)
C
C              IF NVAL IS NON-ZERO, THEN REPEAT PROCESS FOR NEGATIVE STEP
C         INPUT.  OTHERWISE, IF NVAL IS 0, RETURN.
C
          IF (NVAL)     200, 200, 90
C
C              SET UP THE PARAMETERS FOR A NEGATIVE STEP GOING FROM 1000
C         TO 0.  THEN REPEAT THE TEST.
C
     90   NVAL    =     0
          Y       = 1000.0
          TBUF(0)= 1000.0
          GO TO 10
    100   FORMAT (2X, 10(F9.2,1X))
    200   CONTINUE
          RETURN
        END
--FORTRAN
```

8.8 INILER, LERNFIL (IAP, IBP, BUF, PBUF, Y1, Y2, CONST, EAT, XIN)

This program simulates the action of a 4th order Lerner-type band-pass filter. The INILER entry point sets up the various constants needed while the LERNFIL entry point does the actual processing of data. In the setup phase, IAP and IBP specify the frequency bounds of the filter. In the processing phase IBP specifies the number of points in BUF which are to be processed while IAP is not used. PBUF is the output buffer, while Y1, Y2, CONST, EAT, and XIN are variables and constants used to store the information which characterizes a particular filter.

As originally described by Learner[1964], the continuous Lerner filter is defined by the system function[1]:

$$Y(s) = \sum_{i=1}^{m} \frac{B_i \times (s+a)}{(s+a)^2 + b_i^2}$$

where:

$$B_1 = \frac{1}{2} \qquad B_i = (-1)^{i+1} \qquad i = 2, \ldots, m-1$$

$$B_m = \frac{(-1)^{m+1}}{2}$$

The primary advantage of this type of filter is its flat pass band transmission combined with a linear phase change over most of the pass band. In addition (and what is more important in the present case) it is easily implemented on a computer since each term in the system function is quite similar. In particular, following Rader and Gold[1967],

---

[1]The notation used follows Rader and Gold[1967]. As is to be expected, it does not agree with Learner's original notation.

we can develop the following Z Transform equations for the same Lerner filter:

$$Y(Z) = \sum_{i=1}^{m} \frac{B_i (1 - e^{-at}(\cos b_i T)Z^{-1})}{1 - 2e^{-aT}(\cos b_i T)Z^{-1} + e^{-2aT}Z^{-2}}$$

An m-order bandpass digital Lerner filter can then be realized using the following difference equations:

$$y_i(nT) = e^{-aT}(\cos b_i T)[2y_i(nT - T) - x(nT - T)]$$

$$- e^{-2aT} y_i(nT - 2T) + x(nT)$$

$$i = 1, 2, \ldots, m$$

and

$$y(nT) = \sum_{i=1}^{m} B_i y_i$$

Note that with the equations in this form, we can increase the order of the filter (and thus improve its quality) by simply increasing the number of iterations per sample and the storage used. The algorithm remains unchanged.

For this system a four-pole filter was chosen. To get the Learner filter equations into a form more compatible with the FORTRAN programming language, we can do the following:

$$Y1(i) = y_i(nT)$$

$$Y2(i) = y_i(nT - T)$$

$$Y3(i) = y_i(nT - 2T)$$

and

$$CONST(i) = e^{-aT} \cos(b_i T)$$

$$BUF(N) = x(nT)$$

$$XIN = x(nT - T)$$

$$EAT = e^{-2aT}$$

where the subscripts in the various arrays signify the variables for the equations relating to the ith pole.

As a result the equations for the outputs of the four pole filter for any input sample n become:

```
Y1(I) = CONST(I) * (2.* Y2(I) - XIN2) - EAT*Y3(I) + BUF(N)

PBUF(N) = .5 * Y1(1) - Y1(2) + Y1(3) - .5 * Y1(4)
```

Of course, in the process of time and space optimization a few changes were made. In particular Y3 does not need to be stored in an array, since its value is not saved from one iteration to the next. Also the equations were rearranged slightly to speed up the arithmetic.

The program has been written in such a manner that given the upper and lower frequency limits of the desired bandpass filter, it will calculate the proper pole positions, (i.e. evaluate a, b_1, b_2, b_3, and b_4). In order to do this certain assumptions were made about the desirable characteristics of the filter. In particular a decision had to be made on how to pick the ratio of $a/\Delta b$ where a is the distance of the filter poles from the $j\omega$ axis and $2\Delta b$ is the distance between the interior poles of the Lerner filter. After studying Lerner's design criteria and running a few test programs, it was decided to make the ratio equal to one. Also as a result of the test runs, it was decided to arbitrarily increase whatever bandwidth is specified by the calling sequence by a factor of 1.1. This allows the sharpest part of the magnitude curve to occur right on the cutoff frequencies specified instead of further inside the pass band. The theoretical effect of this decision is simply to arbitrarily redefine the originally arbitrary definition of "bandwidth".

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
      SUBROUTINE INILER(IAP,IBP,BUF,PBUF,Y1,Y2,CONST,EAT,XIN)
C
C          THIS SUBROUTINE IS A DIGITAL FILTERING PROGRAM.  IT ACCEPTS
C     A BUFFER OF FLOATING POINT NUMBERS, BUF, AND SIMULATES THE AC-
C     TION OF A 4TH. ORDER BAND-PASS LERNER FILTER TO PRODUCE THE DATA
C     IN THE OUTPUT BUFFER, PBUF.
C          THE CONSTANTS FOR THE FILTER ARE SET UP BY CALLING INILER
C     WITH IAP = LOW FREQUENCY EDGE AND IBP = HIGH FREQUENCY EDGE.
C     AFTER CALCULATING THE CONSTANTS AND STORING THEM IN THE ARRAYS
C     GIVEN AS PARAMETERS, INILER PRINTS OUT THE CALCULATED CONSTANTS
C     AND RETURNS.  WHEN AN INPUT BUFFER NEEDS TO BE FILTERED,
C     LERNFIL IS CALLED WITH IBP = LENGTH OF THE BUFFER AND GIVING AS
C     PARAMETERS, THAT SET OF CONSTANT ARRAYS CORRESPONDING TO THE
C     DESIRED FILTER.
C
C     A        = DISTANCE FROM POLES TO THE IMAGINARY AXIS.
C     B        = ONE HALF THE FREQUENCY DIFFERENCE BETWEEN THE CENTER
C                POLES, = BW/4.
C     BUF      = INPUT DATA BUFFER.
C     BW       = BANDWIDTH IN RADIANS AFTER FUDGE IS MADE.
C     CONST    = CONSTANT ARRAY USED IN FILTER CALCULATIONS.  IT CONTAINS
C                4 ENTRIES BEGINNING WITH ENTRY 0.
C     DTIME    = COMMON VARIABLE.  SAMPLING PERIOD OF INPUT DATA.
C     EAT      = CONSTANT ARRAY USED IN FILTER CALCULATIONS.  IT CONTAINS
C                2 ENTRIES BEGINNING WITH ENTRY 0.
C     FUDGE    = FUDGE FACTOR TO STEEPEN SLOPE AT EDGE OF PASS BAND. ITS
C                EFFECT IS TO WIDEN THE BANDWIDTH SLIGHTLY.
C     I1,I2,   = NAMES FOR INDEX REGISTERS 3 THROUGH 6.  THESE REGISTERS
C      I3,I4     ARE LOADED WITH THE INTEGER CONSTANTS 0 THROUGH 3 SO THAT
C                ELEMENTS IN THE CONSTANT ARRAYS CAN BE ACCESSED BY INDEX-
C                ING INSTEAD OF BY SUBSCRIPT CALCULATION WHICH IS MUCH
C                LESS EFFICENT IN CSL FORTRAN.
C     IAP      = INPUT PARAM. = LOW FREQ. EDGE FOR INFIL CALLS(IN CPS.).
C     IBP      = INPUT PARAM. = HIGH FREQ. EDGE FOR INFIL CALLS(IN CPS.).
C                             = NUMBER OF INPUT SAMPLES FOR LERNFIL CALLS.
C     ISAMF    = SAMPLING FREQUENCY.  COMMON VARIABLE PRESET BEFORE CALL.
C     P        = POLE POSITIONS IN RADIANS.  THIS ARRAY CONTAINS 4
C                ENTRIES BEGINNING WITH ENTRY 0.
C     PBUF     = OUTPUT DATA BUFFER.
C     XIN      = DELAYED INPUT SAMPLE (FROM BUF) PROCESSED JUST PRIOR
C                TO SAMPLE CURRENTLY BEING PROCESSED.
C     Y1,Y2    = INTERMEDIATE PARAMETERS USED BY LERNFIL.  NUMBER REP-
C                RESENTS NUMBER OF TIME DELAYS.  EACH PARAMETER REPRE-
C                SENTS A FOUR ENTRY ARRAY BEGINNING AT ENTRY 0.
C     Y3       = INTERMEDIATE VARIABLE USED BY LERNFIL.  REPRESENTS THE
C                OUTPUT OF THE ITH COMPONENT OF THE LERNER FILTER 3 TIME
C                DELAYS AGO.  THIS VARIABLE IS USED ONLY ONCE AND NEED
C                NOT BE SAVED.  THUS IT DOES NOT NEED TO BE A PARAMETER
C                OR AN ARRAY.
C
      TITLE*
      DIMENSION Y1(3), Y2(3), CONST(3), P(3),
     1          BUF(1), PBUF(1)
```

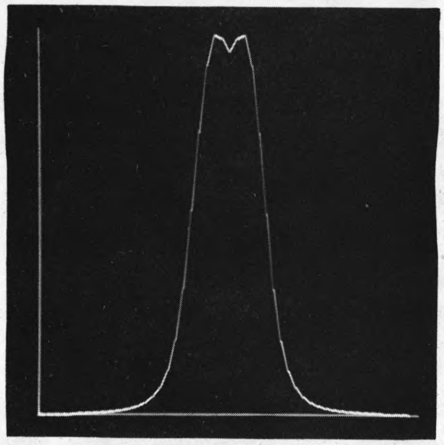
```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
              INDEX I1(3), I2(4), I3(5), I4(6)
      C
      C       SET UP PCLES OF FILTER, IE. CALCULATE B AND CTHER CONSTANTS,
      C
      --ILLAR
      *
      *              LOAD REGISTERS 3 THROUGH 6 WITH THE CONSTANTS 0 THROUGH
      *       3.  THESE WILL BE USED TO ACCESS THE CONSTANT ARRAYS.
      *
               ENI     3   0E
               ENI     4   1E
               ENI     5   2E
               ENI     6   3E
      --FORTRAN
00004     1   APR = FLCATF(IAP)*6.2831853
00007         BPR = FLCATF(IBP)*6.2831853
00012         FUDGE = 1.1
00013         BW = (BPR - APR)*FUDGE
00016         B = BW/4.0
00017         A = B
00021         P(I1) = APR - (FUDGE - 1.)*BW/2.
00026         P(I2) = P(I1) + B
00027         P(I3) = P(I2) + B + B
00032         P(I4) = P(I3) + B
00033         AT = A*DTIME
00035         TAT = AT + AT
00037         ET  = 1./EXPF(AT)
00043         EAT = 1./EXPF(TAT)
00046         DO 5, I=0, 3
00052         PT = P(I)*DTIME
00053         CONST (I) = COSF(PT)*ET
00057     5   CONTINUE
00060         DO 55, I=0, 3
00064         PRINT 50, I, P(I), I, CONST(I)
00075    50   FORMAT (3H P(I1,4H) = F20.8,4X,6HCONST(I1,4H) = F20.8)
00075    55   CONTINUE
00076         RETURN
      C
00077   100   ENTRY LERNFIL
      --ILLAR
      *
      *              LOAD REGISTERS 3 THROUGH 6 WITH THE CONSTANTS 0 THROUGH
      *       3.  THESE WILL BE USED TO ACCESS THE CONSTANT ARRAYS.
      *
               ENI     3   0E
               ENI     4   1E
               ENI     5   2E
               ENI     6   3E
      --FORTRAN
00103         DO 199, J=0, (IBP - 1)
      C
      C       CALCULATE NEW VALUES FOR Y1, Y2, AND Y3.
      C
```

```
   CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
    --ILLAR
               ENI     1  0E
    --FORTRAN
     110      Y3    = Y2(I)
              Y2(I) = Y1(I)
              Y1(I) = CCNST(I)*(Y2(I) + Y2(I) - XIN) - EAT*Y3 + BUF(J)
    --ILLAR
    XX120      ISK     1  3E
               SLJ     0  XX110
    --FORTRAN
    C
    C         CALCULATE OUTPUT
              PBUF(J) = .5*Y1(I1) - Y1(I2) + Y1(I3) - .5*Y1(I4)
    C
    C         CALCULATE DELAYED VALUE OF THE INPUT FOR THE NEXT ITERATION.
    C
              XIN = BUF(J)
     199      CONTINUE
              RETURN
            END
--FORTRAN
```

8.9 LTEST

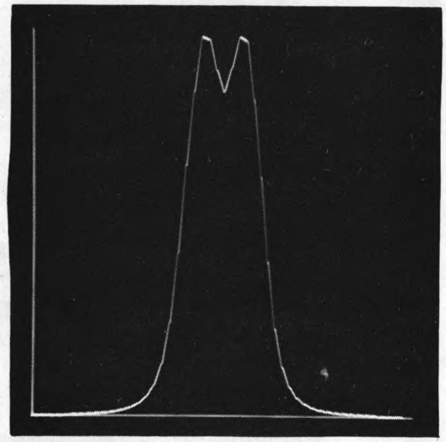
This program tests the Lerner filter subroutine by generating an input of cosine functions at specific frequencies (in 100 cps. steps) and then plotting the output of the filter as a function of frequency. The operator can set the band limits for the frequency by typing in their values. The exact operating procedure is described in the comment at the beginning of the program printout.

The main purpose of this routine is to allow the operator to find out the frequency characteristics of a particular filter before he uses it in his system. It was also helpful during the initial programming of the Lerner filter when the various design parameters were being decided. Figure 8.9.1 shows the effect of changes in the various filter design parameters for a 2000-3000 cps. filter.



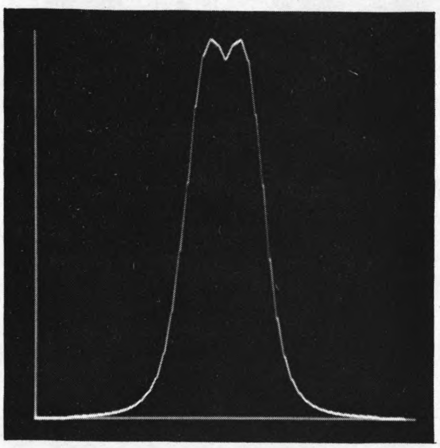
bandwidth factor = 1.0

A = B



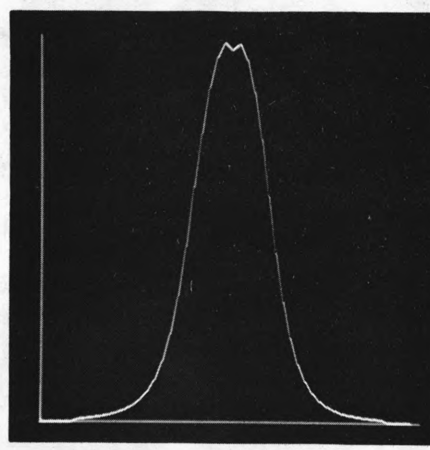
bandwidth factor = 1.1

A = .75(B)



bandwidth factor = 1.1

A = B



bandwidth factor = 1.1

A = 1.5(B)

Figure 8.9.1 Effect of Variations in Filter Parameters for a 2000-3000 cps. Filter

```
CSL FORTRAN OF SEPT 1968,  DATE  9/9/71
      PROGRAM LTEST
C
C          THIS PROGRAM TESTS LERNFIL BY GENERATING COSINE
C      FUNCTIONS AT SPECIFIC FREQUENCIES (IN 100 CPS STEPS) AND THEN
C      RUNNING 4000 SUCCESSIVE SAMPLES (1/4 SEC. AT 20 KC) THROUGH
C      THE FILTER.  THE PROGRAM DISPLAYS THE INPUT AND OUTPUT OF THE
C      FILTER AS A FUNCTION OF FREQUENCY.
C          IN ORDER TO LOAD THIS PROGRAM, TYPE OUT,
C
C              EXITCALL,PROGRAM TAPE,LTEST
C
C     THE SYSTEM WILL TYPE TAPE 5, IF THE COMPLETE LIBRARY IS NOT ON
C     THE PROGRAM TAPE.  IN THIS CASE TYPE A SPACE, PROVIDED THE REST
C     OF THE LIBRARY IS INDEED ON TAPE UNIT 5.
C          ONCE THE PROGRAM IS LOADED, TYPE LTEST TO START THE PROGRAM.
C     THE PROGRAM WILL TYPE A + AFTER WHICH THE OPERATOR MUST TYPE,
C
C              LTEST=LOWER FILTER EDGE(IN CPS.),UPPER FILTER EDGE(IN CPS.)
C
C     THE PROGRAM WILL THEN CONTINUE, PRINT OUT THE RESULTS, AND DIS-
C     PLAY THEM ON THE CRT.
C
C     CONST, = ARRAYS USED BY DIGITAL FILTER DURING PROCESSING.
C      Y1,Y2
C     DIS1    = ARRAY CONTAINING AVERAGE MAGNITUDE OF INPUT AT EACH
C               FREQUENCY.
C     DIS2    = ARRAY CONTAINING AVERAGE MAGNITUDE OF OUTPUT AT EACH
C               FREQUENCY.  AT THE END OF THE PROGRAM, THESE VALUES
C               ARE CHANGED TO THE GAIN AT EACH FREQUENCY BY DIVIDING
C               BY THE CORRESPONDING INPUT AVERAGE.
C     EAT,    = VARIABLES USED BY THE DIGITAL FILTER DURING PROCES-
C      XIN      SING.
C     I1      = INITIAL FREQUENCY EDGE OF FILTER TO BE TESTED (IN
C               CPS.) - TYPED IN BY OPERATOR.
C     I2      = FINAL FREQUENCY EDGE OF FILTER TO BE TESTED (IN CPS.)
C               TYPED IN BY OPERATOR.
C     F1      = DUMMY VARIABLE FOR INPTCM. NOT USED BY LTEST.
C     VAR     = CHARACTER VARIABLE CONTAINING COMMAND TYPED IN BY OPERATOR.
C     XSUM    = TEMPORARY VARIABLE USED IN CALCULATING AVERAGE MAGNI-
C               TUDE OF THE INPUT SIGNAL.
C
      TITLE*
      DIMENSION DIS1(100), DIS2(100), Y1(3), Y2(3), CONST(3)
      EQUIVALENCE (DIS1(1), A(15003)), (DIS2(1), A(15103))
C
C          OBTAIN INPUT COMMAND AND PARAMETERS.
C
    2 CALL INPTCM(VAR, I1, I2, F1, 19)
      IF (VAR - 5HLTEST)    2, 4, 2
C
C          INITIALIZE VARIABLES, CONSTANTS AND FILTER.
C
    4 CALL INITI
```

```
CSL FORTRAN OF SEPT 1968,  DATE  9/9/71
            CALL INILER(I1,I2,FINT,FINT,Y1,Y2,CONST,EAT,XIN)
            X = 0.0
            DO 5, I=0, 100
            DIS1(I) = 0.
        5   DIS2(I) = 0.
      C
      C           TEST FREQUENCIES WILL INCLUDE ALL MULTIPLES OF 100 CPS.
      C     WITHIN THE FILTER BAND AS WELL AS + OR - 1500 CPS ON EITHER SIDE
      C     OF IT.  HOWEVER, IF THIS CAUSES NEGATIVE FREQUENCIES, THE TEST
      C     WILL BEGIN AT ZERO CPS.
      C
            ISTRT = 0
            IF (I1 - 1500)     9, 9, 7
        7   ISTRT = (I1/100) - 15
        9   ISTOP = (I2/100) + 15
            DO 80, I=ISTRT, ISTOP
            ARG = 6.2831853*FLOATF(I)*100./20000.
            XSUM = 0.
            PRINT 15, I
       15   FORMAT (22H ENTER INNER LOOP I = I3)
      C
      C           GENERATE THE INPUT DATA BY CALCULATING THE COSINE FUNCTION
      C     FOR THE FIRST 200 POINTS.  THEN SET THE REMAINING POINTS TO THIS
      C     SAME REPEATING PATTERN. THIS METHOD WILL WORK AS LONG AS THE
      C     TEST FREQUENCIES HAVE AN INTEGRAL NUMBER OF CYCLES IN 200 SAMPLES.
      C
            TEMP = 0.1
            DO 20, J=0, 199
            TEMP = TEMP + ARG
            FINT(J) = 512.*COSF(TEMP)
            XSUM = XSUM + ABSF(FINT(J))
            DO 19, K=(J+200), 4000, 200
            FINT(K) = FINT(J)
       19   CONTINUE
       20   CONTINUE
            DO 25, J=0, 3
            Y1(J) = 0.0
       25   Y2(J) = 0.0
            XIN = 0.0
            CALL LERNFIL(X,4000,FINT,FINT(5000),Y1,Y2,CONST,EAT,XIN)
      C
      C           CALCULATE AVERAGE MAGNITUDE OF THE INPUT AND OUTPUT AND
      C     STORE IN DIS1 AND DIS2 RESPECTIVELY.
      C
            DIS1(I) = XSUM/200.
            CALL AVEMAG(FINT(5000), 0.0, 4000, DIS2(I))
       80   CONTINUE
            PRINT 100, (DIS1(I), I=0. 100)
            PRINT 100, (DIS2(I), I=0, 100)
            DIS2(0) = DIS2(0)*DIS1(ISTOP)/512.
            PRINT 100, (DIS2(I), I=0, 100)
      100   FORMAT (10(F8.1,2X))
            CALL DISSY(X,DIS2,50,ISCOP1,2000,50.,100.,0)
            CALL WHATNOW
            END
--FORTRAN
```

8.10 DIFFER(BUF, NUM, PBUF)

This subroutine differentiates the NUM + 1 data samples in BUF by taking the difference between each successive pair of samples and storing the NUM resulting differences into the successive locations in PBUF. The indexes within the DO loop have been arranged so that BUF and PBUF may be the same physical array if desired.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
          SUBROUTINE DIFFER(BUF, NUM, PBUF)
 C
 C              THIS SUBROUTINE DIFFERENTIATES THE NUM + 1 DATA SAMPLES
 C         IN BUF, BY TAKING THE DIFFERENCE BETWEEN EACH SUCCESSIVE PAIR
 C         OF SAMPLES AND STORING THE NUM RESULTING NUMBERS IN THE SUC-
 C         CESSIVE LOCATIONS IN PBUF.  IN NORMAL USE, THE INPUT DATA
 C         CONSISTS OF NUM SAMPLES FROM SOME LARGER BUFFER SO THAT THE
 C         FINAL DIFFERENCE, WHICH UTILIZES THE NUM + 1 SAMPLE, IS STILL
 C         VALID.
 C              THIS SUBROUTINE WILL STILL OPERATE CORRECTLY IF BUF AND
 C         PBUF ARE THE SAME PHYSICAL ARRAY.
 C
 C         BUF  = INPUT DATA ARRAY.
 C         NUM  = NUMBER OF DIFFERENCES TO BE CALCULATED.
 C         PBUF = OUTPUT DATA ARRAY.
 C
           DIMENSION BUF(1), PBUF(1)
           DO 100, I=0, (NUM - 1)
00007      PBUF(I) = BUF(I+1) - BUF(I)
00013  100 CONTINUE
00015      RETURN
00016     END
--FORTRAN
```

8.11 AVEMAG(BUF, ZERO, NUM, AVE)

This subroutine is used to calculate the average magnitude, AVE, of the NUM data samples in the data buffer, BUF, about the "zero value" specified by ZERO. It was specified in the above manner so as to make it applicable to the widest variety of cases.

The zero level parameter is handy since the input data generally has a significant DC component by virtue of the way the A to D convertor operates. If an average value is needed, ZERO can be set to 0.0 if all the data is positive or to some large negative value if it is not. In the latter case, this value must be subtracted from the resulting average to get the average value of the original input.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
          SUBROUTINE AVEMAG(BUF, ZERO, NUM, AVE)
      C
      C           THIS SUBROUTINE IS USED TO CALCULATE THE AVERAGE MAGNITUDE OF
      C        THE NUM DATA SAMPLES IN THE INPUT BUFFER, BUF, BEGINNING AT LOCATION
      C        0.  SPECIFICALLY, THE VALUE OF THE PARAMETER, ZERO, IS USED TO DE-
      C        TERMINE THE ZERO LEVEL OF THE INPUT DATA ABOUT WHICH THE AVERAGE
      C        MAGNITUDE IS TO BE CALCULATED.  THE RESULT, AVE, IS THE AVERAGE
      C        MAGNITUDE OF THE DEVIATION OF THE DATA FROM THIS ZERO LEVEL.
      C
      C        AV    = TEMPORARY VARIABLE USED TO CALCULATE AVERAGE.
      C
               DIMENSION BUF(1)
               AV = 0.0
  00003        DO 10, L=0, (NUM - 1)
  00010    10  AV = AV + ABSF(BUF(L)-ZERO)
  00015        AVE = AV/FLOATF(NUM)
  00020        RETURN
  00021        END
--FORTRAN
```

8.12 PEAPIC(BUF, NUM, PBUF, INDX, LIMIT)

This subroutine is used to pick out the maxima peaks in the input buffer, BUF. The subroutine scans NUM samples in BUF and for each maximum peak it finds, it creates entries in PBUF and INDX containing the value of the maximum peak and its location within BUF respectively. Upon completion the subroutine enters the number of peaks found into the LIMIT parameter and returns.

PEAPIC scans the contents of BUF beginning with entry 0 and continuing for NUM samples. Thus it effectively inspects NUM - 1 intervals. PBUF and INDX are loaded beginning with entry 1. Entry 0 in both buffers is set to zero. This is helpful in those cases where PEAPIC is used to operate on its own output.

The operation of PEAPIC is perfectly straightforward. It looks at the slopes between samples and watches for changes in the sign of the slope. IFLAG is used to store the sign of the previous slope, index 1 is used to keep track of the position within BUF, and index register 2 is used to keep track of the positions within PBUF and INDX and to count the number of peaks found by PEAPIC.

Note that the portion of the subroutine starting at RECORD is used to store the peak magnitudes which have been found. Index register 1 is used to make an indexed memory access to BUF in order to obtain this magnitude. However, since at this point, IR1 has already been incremented to the next sample in order to calculate the "latest" slope, the address field of the indexed instruction is set to the address of BUF minus one.

```
CSL FORTRAN OF SEPT 1968,  CATE  6/28/71
          SUBROUTINE PEAPIC(BUF, NUM, PBUF, INDX, LIMIT)
  C
  C             THIS SUBROUTINE IS USED TO EXTRACT THE MAXIMA POINTS FROM
  C         THE INPUT BUFFER, BUF.  IT SCANS THE INPUT DATA, BEGINNING WITH
  C         THE ZEROTH POSITION IN BUF, AND STORES THE VALUE AND INDEX POSI-
  C         TIONS OF THE SUCCESSIVE MAXIMA INTO THE PBUF AND INDX ARRAYS,
  C         RESPECTIVELY, BEGINNING WITH LOCATION 1.  THE ZEROTH ENTRIES
  C         IN PBUF AND INDX ARE SET TO ZERO.  AFTER PROCESSING NUM
  C         INTERVALS FROM BUF, THE SUBROUTINE LOADS THE NUMBER OF MAX-
  C         IMA FOUND INTO LIMIT AND RETURNS.
  C             THIS SUBROUTINE WILL WORK PROPERLY IF BUF AND PBUF ARE
  C         THE SAME ARRAY IN STORAGE.
  C
  C         BUF    = INPUT DATA ARRAY.
  C         FIR    = TEMPORARY LOCATION USED TO STORE PREVIOUS SAMPLE WHEN
  C                  CALCULATING SLOPES.
  C         I      = INDEX REGISTER 1 - USED TO KEEP TRACK OF POSITION IN
  C                  BUF AS DATA IS PROCESSED.
  C         IFLAG  = INDICATOR FOR PREVIOUS SLOPE.
  C                = +, IF SLOPE IS +, - IF SLOPE IS -.
  C         INDX   = OUTPUT DATA ARRAY FOR LOCATION OF PEAKS FOUND BY PEAPIC.
  C         J      = INDEX REGISTER 2 - USED TO KEEP TRACK OF NUMBER OF PEAKS
  C                  STORED IN INDX AND PBUF.
  C         LIMIT  = NUMBER OF PEAKS FOUND BY PEAPIC.
  C         NUM    = NUMBER OF INTERVALS TO BE PROCESSED BY PEAPIC.
  C         PBUF   = OUTPUT DATA ARRAY FOR MAGNITUDE OF PEAKS FOUND BY PEAPIC.
  C
            DIMENSION BUF(1), PBUF(1), INDX(1)
  C
  C             INITIALIZE I AND J (INDEX REGISTERS 1 AND 2 RESPECTIVELY) AND
  C         ALSO FIR AND IFLAG.
  C
     20     I = 1
00003       J = 1
00005       FIR = BUF
00007       IFLAG = 1
  --ILLAR
  *             CALCULATE END ADDRESS OF DATA BUFFER.
  *
             LDA        NUM
             SAU        ITEST       STORE NUMBER OF SAMPLES IN THE ADDRESS PORTION
  *                                  OF THE LOOP TEST INSTR. IN THE UPPER HALF OF
  *                                  LOCATION ITEST.
             ENA        BUF         STORE ADDRESS OF THE BUFFER, BUF
             INA        77776B       MINUS 1, IN THE ADDRESS PORTION OF THE INSTR.
             SAU        RECORD       WHICH ACCESSES A DISCOVERED PEAK, WHICH IS
  *                                  LOCATED IN THE UPPER HALF OF LOCATION RECORD.
  *
  *             CHECK THE DIFFERENCE BETWEEN THE FIRST TWO BUFFER ENTRIES TO
  *         DETERMINE THE INITIAL SLOPE OF THE DATA AND SET THE SIGN OF THE Q
  *         REGISTER ACCORDINGLY.
  *
             LDA    1  BUF
```

CSL FORTRAN OF SEPT 1968, DATE 6/28/71

```
         FSB         FIR
         AJP    2    LOOP             IF SLOPE IS +, GO DIRECTLY TO LOOP
         LDQ         =-1                 OTHERWISE SET Q AND IFLAG TO -
         STQ         IFLAG             BEFORE GOING TO LOOP
         SLJ    0    LOOP
*
*           IF PROCESSING IS TO CONTINUE AFTER THE END TEST, CALCULATE
*        THE SIGN OF THE SLOPE BETWEEN THE NEW PAIR OF POINTS AND LOAD IT
*        IN THE A REGISTER,  IF THE SLOPE IS ZERO ASSUME THERE IS NO PEAK AND
*        CONTINUE.  IF THE SLOPE IS NOT ZERO, LOAD THE Q REGISTER WITH THE
*        SIGN OF THE SLOPE OF THE PREVIOUS POINT PAIR AND TEST THE SIGN OF THE
*        A REGISTER.
*
CONT     LDA    1    BUF          LOAD CURRENT SAMPLE AND SUBTRACT PREVIOUS SAMPLE
         FSB         FIR            TO GIVE THE RESULTING SLOPE IN THE A REG.
         AJP    0    LOOP                 IF SLOPE IS 0, GO DIRECTLY TO LOOP.
         LDQ         IFLAG        LOAD Q REG. WITH IFLAG FOR TEST LATER ON.
         AJP    2    POS          IF SLOPE IS +, GO TO POS.
*
*           IF THE CURRENT 2 POINTS HAVE A NEGATIVE SLOPE, CHECK THE
*        SIGN OF THE PREVIOUS SLOPE(CURRENTLY STORED IN THE Q REGISTER)
*        AND IF IT WAS POSITIVE, A PEAK HAS BEEN FOUND.  IF THE PREVIOUS
*        SLOPE WAS NEGATIVE, RETURN TO THE LOOP.
*
NEG      QJP    3    LOOP         IF OLD SLOPE WAS ALSO -, CONTINUE BY GOING TO LOOP.
         SLJ    0    RECORD         OTHERWISE RECORD THE DISCOVERED PEAK.
*
*           IF THE CURRENT TWO POINTS HAVE A POSITIVE SLOPE, CHECK THE
*        SIGN OF THE PREVIOUS SLOPE, AND IF IT WAS NEGATIVE A MINIMUM
*        POINT HAS BEEN FOUND.  IN THIS CASE THE SIGN OF THE PREVIOUS
*        SLOPE MUST BE CHANGED TO POSITIVE BEFORE RETURNING TO THE LOOP.
*        IF THE PREVIOUS SLOPE WAS POSITIVE, RETURN DIRECTLY TO THE LOOP.
*
POS      QJP    2    LOOP         IF OLD SLOPE WAS ALSO +, CONTINUE BY GOING TO LOOP.
         LDQ         =1             OTHERWISE CHANGE THE SIGN OF IFLAG
         STQ         IFLAG
         SLJ    0    LOOP           BEFORE GOING TO LOOP.
*
*           WHEN A PEAK IS FOUND, RECORD ITS MAGNITUDE IN PBUF(J) AND
*        ITS LOCATION IN INDX(J).  THEN INCREMENT J(INDEX REGISTER 2)
*        BY 1.
*
RECORD   LDA    1    **                   ADDRESS FIELD IS LOADED WITH BUF - 1.
         STA    2    PBUF
         ENA    1    0E           LOAD A WITH INDEX REGISTER 1.
         INA         77776B       INCREMENT A REG. BY -1.
         STA    2    INDX         STORE THIS VALUE IN THE CURRENT INDX ENTRY.
         INI    2    1E           INCREMENT CURRENT ENTRY ADDRESS (IE. XR2) FOR
*                                   INDX AND PBUF BUFFERS BY 1.
*
*           RESET THE SLOPE OF THE PREVIOUS POINT PAIR (IE, IFLAG) TO
*        NEGATIVE.  THEN CONTINUE WITH THE LOOP.
*
```

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
 SLNEG      LDQ        =-1
            STQ        IFLAG
 *
 *                LOAD THE SECOND MEMBER OF THE CURRENT PAIR OF POINTS INTO
 *          FIR, THE STORAGE LOCATION FOR POINT NUMBER ONE, IN PREPARATION
 *          FOR THE NEXT ITERATION.
 *
 LOCP       LDA     1  BUF
            STA        FIR
 *
 *                TEST I (INDEX REGISTER 1) TO SEE IF THE LAST POINT HAS BEEN
 *          REACHED.  NOTE THAT THE SUBROUTINE TESTS NUM POINTS WHICH MEANS
 *          THAT IT PROCESSES NUM - 1 INTERVALS.
 *
 ITEST      ISK     1  **                  ADDRESS FIELD IS LOADED WITH NUM.
            SLJ     0  OCNT
 --FORTRAN
00012  100    LIMIT = J - 1
00013         PBUF = 0.0
00015         INDX = 0
00016         RETURN
00017       END
--FORTRAN
```

8.13 RECTIF(BUF, ZERO, NUM, PBUF)

This subroutine is a half-wave rectifier. Using the value in ZERO as a "zero level", it examines each value in BUF and transfers it to PBUF if it is greater than ZERO. If it is not, it loads the corresponding entry of PBUF with ZERO. After processing NUM samples (beginning with the zeroth sample in BUF), it returns. Note that the subroutine will operate correctly even if BUF and PBUF are both the same physical array.

```
CSL FORTRAN OF SEPT 1968,  DATE  6/28/71
          SUBROUTINE RECTIF(BUF, ZERO, NUM, PBUF)
 C
 C              THIS SUBROUTINE HALFWAVE RECTIFIES THE NUM DATA SAMPLES
 C          IN THE INPUT BUFFER, BUF, BEGINNING WITH LOCATION 0.  ALL
 C          POINTS WHICH ARE LESS THAN THE ZERO LEVEL VALUE STORED IN
 C          THE PARAMETER ZERO ARE TRUNCATED TO THAT VALUE BEFORE BEING
 C          TRANSFERRED TO THE OUTPUT BUFFER, PBUF.  POINTS GREATER THAN
 C          THIS VALUE ARE LOADED INTO THEIR RESPECTIVE ENTRIES IN PBUF
 C          UNCHANGED.
 C              BUF AND PBUF MAY BE THE SAME BUFFER IF DESIRED. AND THE
 C          SUBROUTINE WILL STILL OPERATE.
 C
            DIMENSION BUF(1), PBUF(1)
            DO 100, I=0, (NUM - 1)
00007       IF (BUF(I) - ZERO)    10, 10, 20
00012   10  PBUF(I) = ZERO
00013       GO TO 100
00014   20  PBUF(I) = BUF(I)
00015  100  CONTINUE
00016       RETURN
00017      END
--ILLAR
```

SECTION 9

REFERENCES

Brigham, E.O. and Morrow, R.E., "The Fast Fourier Transform", IEEE Spectrum, Vol. 4, No. 12, December 1967, pp. 63-70.

Cochran, W.T., Cooley, J.W., Favin, D.L., Helms, H.D., Kaenel, R.A., Lang, W.W., Maling, G.C. Jr., Nelson, D.E., Rader, C.M. and Welch, P.D., "What is the Fast Fourier Transform?", IEEE Trans. on Audio and Electoacoustics, Vol. AU-15, No. 2, June 1967, pp. 45-55.

Cooley, J.W., "HARM - Harmonic Analysis Subroutine", IBM Share Library, HARM 3425, January 1966.

Cooley, J.W. and Tukey, J.W., "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computation, Vol. 19, No. 90, April 1965, pp. 297-301.

Dolansky, L.O., "An Instantaneous Pitch Period Indicator", Journal of Acoust. Soc. Am., Vol. 27, No. 1, January 1955, pp. 67-72.

Dolansky, L., Ferullo, R.J., O'Donnell, M.C. and Phillips, N.D., "Teaching Intonation and Inflections to the Deaf", Northeastern University, Cooperative Res. Proj. No. S-281, 1965.

Dolansky, L. and Phillips, N.D., "Teaching Vocal Pitch Patterns Using Visual Feedback From the Instantaneous Pitch-Period Indicator for Self-Monitoring", Northeastern University, VRA Proj. 1907-S, October 1966.

Gentleman, W.M. and Sande, G., "Fast Fourier Transforms - For Fun and Profit", Fall Joint Computer Conference Proceedings - 1966, Vol. 29, Sparten, Washington, D.C., 1966, pp. 563-578.

Nordmann, B.J. Jr., "A Comparative Study of Some Visual Speech Displays", Digital Computer Laboratory, University of Illinois, Tech. Report No. 479, September 1971. Also as Coordinated Science Laboratory, University of Illinois, Tech. Report R-523, September 1971.

Pyron, B.O. and Williamson, F.R. Jr., "Study and Analysis of Signal Display and Bandwidth Compression Techniques", Georgia Institute of Tech., Final Report Project A-791, Contract DA 49-092-ARO-52, AD 616 644, June 1965.

Appendix A

FORTRAN Implementation of Transformation Equations for the Analysis of 2N Real Sample Points With N Complex Coefficients

Following Cooley[1966] we first note that given

$$Z_j = X_j + iY_j = \sum_{k=0}^{n-1} A_k W^{jk} \qquad (1)$$

$$j = 0, 1, \ldots, n-1$$

where X_j and Y_j are real and $W^{jk} = e^{\frac{2\pi jki}{N}}$, then the complex amplitudes defined by:

$$X_j = \sum_{k=0}^{n-1} A'_k W^{jk} \qquad (2)$$

$$Y_j = \sum_{k=0}^{n-1} A''_k W^{jk} \qquad (3)$$

can be shown to be given by:

$$A'_k = \frac{(A_k + A^*_{(n-k)})}{2} \qquad (4)$$

$$A''_k = \frac{(A_k - A^*_{(n-k)})}{2i} = \frac{-i(A_k - A^*_{(n-k)})}{2} \qquad (5)$$

Thus A'_k and A''_k represent the complex amplitudes for the 2 real data sets, X and Y, respectively. If we now let these two sets be the odd and the even data samples, respectively, of the real data set T, we get:

$$T_{2i+1} = \sum_{k=0}^{n/2-1} A'_k (W^2)^{jk} \qquad (6)$$

(odd points)

$$T_{2j} = \sum_{k=0}^{n/2-1} A''_k (W^2)^{jk} \qquad (7)$$

(even points)

$$j = 0, 1, \ldots, n/2$$

By plugging in the inverse of equations (6) and (7), it can be shown that the coefficients of:

$$T_j = \sum_{k=0}^{n-1} A_k''' W^{jk} \qquad (8)$$

$$j = 0, 1, \ldots, n-1$$

are given by:

$$A_k''' = (A_k' + A_k'' W^{-k}) \qquad (9)$$

$$A_{n/2+k}''' = (A_k' - A_k'' W^{-k}) \qquad (10)$$

$$k = 0, 1, \ldots, n/2-1$$

Where the factor W^{-k} comes from the shifting of the zero time location in the even data samples.

Thus by plugging equations (4) and (5) into equations (9) and (10), we obtain the expressions for the complex Fourier coefficients for the original real data, namely:

$$A_k''' = \frac{A_k + A^*_{(n-k)} - iA_k W^{-k} + iA^*_{(n-k)} W^{-k}}{2} \qquad (11)$$

$$A_{n/2+k}''' = \frac{A_k + A^*_{(n-k)} + iA_k W^{-k} - iA^*_{(n-k)} W^{-k}}{2} \qquad (12)$$

However, since the CSL FORTRAN cannot make use of complex numbers or operations, we will express W^{-k} as:

$$W^{-k} = e^{\frac{-2\pi ik}{N}} = \cos\frac{2\pi k}{N} - i(\sin\frac{2\pi k}{N}),$$

and separate equations (11) and (12) into real and imaginary components.
Thus since:

$$A_k W^{-k} = A_{kr}\cos\alpha + iA_{ki}\cos\alpha - iA_{kr}\sin\alpha + A_{ki}\sin\alpha$$

and

$$A^*_{(n-k)} W^{-k} = A_{(n-k)r}\cos\alpha - iA_{(n-k)i}\cos\alpha - iA_{(n-k)r}\sin\alpha - A_{(n-k)i}\sin\alpha$$

where $\alpha=\frac{2\pi k}{N}$, we get:

$$A'''_{kr} = \frac{A_{kr} + A_{(n-k)r} + A_{ki}\cos\alpha - A_{kr}\sin\alpha + A_{(n-k)i}\cos\alpha + A_{(n-k)r}\sin\alpha}{2}$$

$$= \frac{A_{kr} - A_{(n-k)r} + \cos\alpha[A_{ki} + A_{(n-k)i}] - \sin\alpha[A_{kr} - A_{(n-k)r}]}{2},$$

$$A'''_{ki} = \frac{A_{ki} - A_{(n-k)i} - A_{kr}\cos\alpha - A_{ki}\sin\alpha + A_{(n-k)r}\cos\alpha - A_{(n-k)i}\sin\alpha}{2}$$

$$= \frac{A_{ki} - A_{(n-k)i} - \cos\alpha[A_{kr} - A_{(n-k)r}] - \sin\alpha[A_{ki} + A_{(n-k)i}]}{2},$$

$$A'''_{(n-k)r} = \frac{A_{kr} + A_{(n-k)r} - A_{ki}\cos\alpha + A_{kr}\sin\alpha - A_{(n-k)i}\cos\alpha - A_{(n-k)r}\sin\alpha}{2}$$

$$= \frac{A_{kr} + A_{(n-k)r} - \cos\alpha[A_{ki} + A_{(n-k)i}] + \sin\alpha\,[A_{kr} - A_{(n-k)r}]}{2},$$

$$A'''_{(n-k)i} = \frac{A_{ki} - A_{(n-k)i} + A_{kr}\cos\alpha + A_{ki}\sin\alpha - A_{(n-k)r}\cos\alpha + A_{(n-k)i}\sin\alpha}{2}$$

$$= \frac{A_{ki} - A_{(n-k)i} + \cos\alpha[A_{kr} - A_{(n-k)r}] + \sin\alpha[A_{ki} + A_{(n-k)i}]}{2} \cdot$$

In order to transform these equations into the form used in the FFTB subroutine, the following simplifications are used:

$$FR1 = A_{kr} + A_{(n-k)r}$$

$$FI1 = A_{ki} - A_{(n-k)i}$$

$$FR2 = A_{kr} - A_{(n-k)r}$$

$$FI2 = A_{ki} + A_{(n-k)i}$$

$$C1 = \cos\alpha$$

$$S1 = -\sin\alpha$$

Thus we have:

$$A'''_{kr} = .5*[FR1 + C1*FI2 + S1*FR2]$$

$$A'''_{ki} = .5*[FI1 - C1*FR2 + S1*FI2]$$

$$A'''_{(n-k)r} = .5*[FP1 - C1*FI2 - S1*FR2]$$

$$A'''_{(n-k)r} = .5*[FI1 + C1*FR2 - S1*FI2]$$

But if we use the following representations:

$$A''_{kr} \rightarrow X(k)$$

$$A'''_{ki} \rightarrow Y(k)$$

$$A'''_{(n-k)r} \rightarrow X(n-k)$$

$$A'''_{(n-k)i} \rightarrow Y(n-k) \ ,$$

replacing k by J and n-k by J1 and using the additional simplifications:

FR3 = C1*FR2 - S1*FI2

FI3 = C1*FI2 + S1*FR2 ,

we finally get:

X(J) = .5*[FR1 + FI3]

Y(J) = .5*[FI1 - FR3]

X(J1) = .5*[FR1 - FI3]

Y(J1) = .5*[FI1 + FR3] .

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Coordinating Science Laboratory University of Illinois Urbana, Illinois	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

SPEECH DISPLAY SIMULATION SYSTEM FOR A COMPARATIVE STUDY OF SOME VISUAL SPEECH DISPLAYS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

Bernard J. Nordmann, Jr.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
November, 1971	221	10

8a. CONTRACT OR GRANT NO. DAAB-07-67-C-0199; AT(11-1)-2118 b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) R-524 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* UILU-ENG 71-2227

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Joint Services Electronics Program through U. S. Army Electronics Command, Fort Monmouth, New Jersey

13. ABSTRACT

The purpose of this report is to give a detailed description of a speech display simulation system which has been programmed for the CDC 1604 computer system located at the Coordinated Science Laboratory at the University of Illinois. This simulator was written as part of an investigation into the relative effectiveness of various types of speech displays (Nordmann [1971]). As such, its primary goal was to be as flexible as possible as far as the generation of various types of speech displays is concerned.

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Speech processing						
Speech displays						
Simulation						
Computer graphics						

Security Classification