PROCEDURES FOR COMPUTER ANALYSIS OF POWER SPECTRA IN BEHAVING SUBJECTS

A Thesis

Presented to

the Faculty of the Department of Psychology

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

By

James H. Stramler, Jr.

May 1977

ACKNOWLEDGMENTS

There are many people who have aided me in writing these programs and in understanding the principles behind signal analysis. I would like to thank my advisor, Dr. Sheer, for his general support and Dr. Everett for making the suggestion that got me into this originally.

Dave Killough of the University of Houston ESSL staff was extremely helpful in devising the logic detection system and in de-bugging the ATOD program. This project would probably never have reached this stage without his assistance.

I would also like to thank Bill Drake for drawing the logic detection circuit shown in Figure 1.

Dr. Ktonas of the University of Houston Electrical Engineering Department was invaluable in explaining some of the theoretical and practical aspects of power spectra computation.

PROCEDURES FOR COMPUTER ANALYSIS OF POWER SPECTRA IN BEHAVING SUBJECTS

An Abstract of a Thesis
Presented to
the Faculty of the Department of Psychology
University of Houston

In Partial Fulfillment
of the Requirements for the Degree
Master of Arts

By
James H. Stramler, Jr.
May 1977

ABSTRACT

Two computer programs for analysis of EEG data are described. Both programs are coded in FORTRAN and were written to be compatible with the University of Houston ESSL hybrid facility. The first of these is an analog-to-digital conversion program (ATOD), and the second (PWRSPC) computes power spectra for the data converted. Theoretical and practical issues involved with such data processing are discussed, and examples of output spectra are presented.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Since Berger (1929) first reported the occurrence of electrical activity on the scalp of man, various techniques have been developed in attempts to obtain as much information as possible from the electroencephalogram (EEG).

The initial reports, and even many reports today, contain merely visual pattern descriptions of the EEG. The EEG contains at least four major features which may be used to describe it. These include:

1) amplitude
2) frequency/wavelength
3) waveform
4) phase

Automated analyses have been attempted on all of these features of brain electrical activity. Extensive reviews of these techniques are presented in Matousek (1973), Walter (1972), Knott (1953), etc.; but brief mention will be made here for completeness and definitional purposes.

Amplitude is usually defined (Brazier, et al, 1961) as the peak-to-peak voltage of the signal present. Amplitude measures have been used in several ways, but the general technique is to pass the full-wave rectified electrical signal to an integrator and then provide either an analog or,

more commonly, a digital output, depending on the experimenter's preference. The signal may be a filtered band of the EEG (e.g., Baldock and Walter, 1946), or it may be the entire EEG waveform (e.g., Runnals, 1963).

Frequency analysis in electrophysiology includes the measure of amplitude as well as frequency. There are many types of frequency analysis. Early attempts at frequency analysis either used many narrow bandpass electrical filters, each covering a different band of the EEG frequency range (e.g., Baldock and Walter, 1946) or a single strip of EEG record run repeatedly through a variable filter (e.g., Grass and Gibbs, 1938). More recently, with the introduction of high-speed digital computers, analog-to-digital (A-D) conversions, and the fast Fourier transform (FFT), generally more accurate and complete frequency spectra are obtained than possible with analog filters.

Frequency and wavelength (or period) are inversely related. Frequency is defined by Brazier, et al (1961) as the number of complete cycles in one second, and wavelength (period) is the time interval from the beginning to the end of a wave. Different types of data are obtained with period analysis. According to Saltzberg (1973), the basic unit of measurement is the time required from one baseline cross to the next. Also used are the times at which the signal has an inflection point. These correspond to the times when the signal and its first two derivatives are zero, respectively.

Automated waveform or pattern recognition usually involves some type of pattern detection by a computer. The pattern of interest may consist of a single cycle or it may be a burst of a certain type of activity. Methods used include sequential comparison of frequency analyses (e.g., Larsen and Walter, 1970) and integration of waveforms as in amplitude analysis (e.g., Frost, 1970) for sleep stage determination or for complex comparison as might occur in epilepsy (e.g., Frost, personal communication).

Phase relationships are used in autocorrelation to determine various periodicities present in a single channel. Becoming more common today, however, are cross-channel comparisons to detect phase relationships between two different EEG derivations (e.g., Barlow, 1973).

Probably the most common type of EEG analysis today is spectral analysis, and some believe it is the most important and useful type of analysis (e.g., Dumermuth, 1973; Larsen, 1971). As Dumermuth (1973) explains, there are several types of spectral analysis. Power spectra provides the average intensity as a function of frequency and are a measure of the second moment of a signal. Cross spectra give the average intensity shared by two signals, also as a function of frequency. Bispectra are a measure of the third moment of a signal and give the relations between frequencies within the same signal. Coherence is a measure of the strength of a linear relationship between activity in two different channels (Walter, 1972).

The most common form of spectral analysis is the power spectrum. This is a useful tool for estimating the intensity present at each frequency. It has proven useful in determining frequency changes under different conditions with the normal EEG, but has not proven very effective for clinical diagnoses (Walter, 1972). The latter is probably because the Fourier transform used to convert from the time domain to the frequency domain assumes a sinusoidal waveform, and pathological EEGs do not usually have such waveforms.

Since power spectra occupy such an important position in electrophysiological research today, every institution involved in such research should have the capability for these computations. Benignus (unpublished) had written a program for computing spectral analysis, but it was very slow and the output appears questionable in some cases. Osorio (1976) had written a series of programs as his dissertation for estimating power spectra during different stages of sleep in the human under the influence of drugs, I have discovered some flaws in Osorio's power programs (e.g., tapering over added zeroes and the use of a questionable smoothing procedure). In addition, his programs were written for processing only a single channel of data. With such a set of programs it is impossible to do computations involving relationships between EEG channels - i.e., coherence, phase relations, etc. These measures appear to be developing now as important tools in describing brain function.

Thus I have coded into FORTRAN the two programs described. The first is an A-D conversion (ATOD) for four channels of EEG data simultaneously. The second program (PWRSPC) computes and plots power spectra for each of these four channels in sequence.

CHAPTER II

ANALOG-TO-DIGITAL CONVERSION

Theory and General Considerations

There are several basic factors which must be considered in performing an analog-to-digital (A-D) conversion. These include:

1) sampling rate
2) quantizing levels
3) epoch length
4) identification of events

In sampling or digitizing the analog EEG data, one must obviously consider future processing of the data. This usually means that a sequence of programs, and not a single program, must be written to fully process the data. Another important fact is that once the data is digitized, any error included with the data or created by the A-D operation cannot be corrected by any form of later digital processing (e.g., Walter, 1963). There are numerous potential sources of such errors, and these are discussed.

The sampling rate must be at least twice that of the highest frequency present in the data (Dumermuth, 1976). If it is not, a phenomenon called aliasing occurs. Aliasing is the "folding back" of higher frequencies onto lower frequencies. For example, if sampling is performed at 128/sec

and frequencies above 64 Hz are present in the analog data, the ascending frequencies above 64 Hz will be aliased back onto the frequencies below 64 Hz in descending order such that 65 Hz will fold back onto 63 Hz, 66 Hz onto 62 Hz, etc. The frequency at which folding occurs is known as the Nyquist frequency (Blackman & Tukey, 1958), and it is defined as:

$$f_n = \frac{1}{2\Delta t}$$

where Δt is the time interval between samples (Bendat & Piersol, 1971).

Some sources of error must be considered even before recording the EEG. One of these is the output frequency response of the amplifier. Many amplifiers have a low-pass filter to attenuate the higher EEG frequencies. The Grass Instruments Model 78 used in this laboratory has six possible low-pass filter settings, ranging from 100 Hz to 30 KHz. According to the Model 78 manual for the 7P511 amplifier (Grass, 1967), signal frequencies are attenuated by one-half at the setting indicated. While the actual rolloff characteristics of these filters are not specified, they appear to be approximately 6 db per octave, judging from the characteristics of other curves which are plotted in the manual. And according to Macy (1965), this is a standard rolloff. The highest frequency present in the signal, therefore, depends directly on the setting of this low-pass filter, and the sampling rate must be selected

accordingly. A certain amount of error must be accepted here, however, because the amplitudes of these higher frequencies are theoretically never reduced to zero.

Another factor which must be considered prior to recording EEG's if an analog tape system is used is that of recorder noise. There are several factors which contribute to tape recorder noise - dirty heads, flutter, etc. - and these may affect decisions on sampling rate.

If only a low sampling rate is possible, and no cross-channel processing is expected, additional analog filtering of the signal(s) may be performed prior to the A-D conversion in order to remove most of the high-frequency components which would fold back into the range of interest. This procedure was used by Osorio (1976) for one channel and Gotman (1973) for multiple channels to compute power spectra.

Should it be desirable to process several channels of data simultaneously, however, it becomes impractical to pre-filter each channel for at least three reasons. First is the number of physical filters required; second is the possibility of introducing different noise components by the different filters; and third is the fact that filters introduce different phase shifts across the frequency spectrum (e.g., Saunders & Jell, 1959) which make the accuracy of cross-channel processing results questionable, at least for phase relationships. Probably the only ready solution to this difficulty is to select a sampling rate high enough such that the unwanted frequencies are not aliased to a significant degree onto the frequency range of interest.

In performing an A-D conversion at a given sampling rate, the signal is converted to a digital value corresponding to the digital voltage level nearest the actual voltage of the analog signal. These digital voltage levels are termed quantizing levels (e.g., Bendat & Piersol, 1971). Theoretically, the greater the number of quantizing levels, the greater the accuracy in amplitude representation, assuming utilization of the full voltage range. In practice, Dumermuth (1976) claims that about 2^{10} or 2^{11} quantizing levels is sufficient for EEG data.

To sample several channels of data simultaneously, sample and hold amplifiers are frequently used, and are the best way to represent data for later cross-channel processing. These amplifiers sample all channels at the same point in time and hold the quantized levels until they can be transferred to storage. The use of such amplifiers makes feasible cross-channel studies without the phase differences introduced by the time delay between successive samples if the sample-hold amplifiers are not used (e.g., Macy, 1965). Regardless of the sampling technique, when several channels are processed, the data is usually multiplexed in storage - that is, the first data point is the first sampled point from the first channel, the second data point is the first sampled point from the second channel, etc.

The epoch length selected determines the maximum resolution of the frequency spectrum (Blackman & Tukey, 1958).

This particular point is discussed in more detail in the next chapter, but it must be selected with consideration of the following: the mathematical product of the epoch length, sampling rate, and number of channels sampled gives the total number of data points obtained per epoch. This number obviously must be less than or equal to the computer memory available and the capacity of the computer to write to digital tape or some other form of permanent storage.

And, finally, the proper identification of events on analog tape is very important, especially if only selected portions of the tape are to be digitized. Simply noting meter settings is not sufficiently precise for computer work.

Dumermuth and Fluhler (1967) suggest marking one channel of the analog tape with a DC signal to indicate portions to be sampled. Other methods have been used also. One semi-standardized technique is a time code devised by the Inter-Range Instrumentation Group (IRIG). This is discussed in Systron-Donner (1970), and is a code to identify time with an accuracy of 0.01 second using pulses on one channel of the analog tape. Another technique is the inputting of sign waves of different frequencies (e.g., Hartley, 1972).

The data must also be identified once it is in digital form. This is usually accomplished by attaching some sort of coded label to the data block itself.

Program ATOD

Before elaborating on the A-D program, I believe it necessary to present a brief description of the computers involved. This program, and the PWRSPC to follow, were written specifically for these machines; it is unlikely, even though both written in FORTRAN, that these programs could be used on other computers without some modification.

These computers are located in the University of Houston Engineering Systems Simulation Laboratory (ESSL) in the Cullen College of Engineering. The analog computer is a Hybrid Systems SS100; the digital computer is an IBM 360/44; and these are interfaced by a Hybrid Systems 1044 Hybrid Linkage Unit. Together, these computers receive the analog input, perform the A-D conversions, and write the digitized data to digital tape for permanent storage.

The SS100 is a large analog computer. It has two patch panels, one for performing logical functions and the other for electronic signal input, processing, and output. The computer provides amplifying, switching, integrating, etc., capabilities.

The 1044 Linkage Unit performs the A-D conversions and multiplexing operations, and inputs data to the 360/44.

IBM's 360 digital computer is fully documented in the IBM literature, and the computer is widely used throughout the world. The unit in ESSL consists of the CPU, two tape drives, three disks, a card reader and punch, a high-speed printer, and an operator's console. Word length on this machine is 32 bits, broken down into 4 bytes of 8 bits each.

A series of library subroutines called the "X" subroutines (ESSL, no date) are available to assist in performing A-D and digital-to-analog (D-A) conversions with the hybrid computer arrangement. The "X" routines used in this A-D program are described briefly below.

XRDSNS is a logical function which reads and tests the specified sense line for its logical value - true or false. ATOD uses this subroutine for directing the program to the appropriate segment, based on the logical determination.

The XSTCTL and XRSCTL subroutines are used, respectively, to set (make true) and reset (make false) specified control lines. Control lines may thus be used to provide a variable logic at given points in a program simply by calling the appropriate subroutine.

XRDAD is a subroutine for performing A-D conversion. In calling this routine, a number of conditions must be specified. These include the number of data points to be sampled, the location in memory where the data are to be stored, the multiplexing positions to be sampled, and the type of control desired over A-D conversion.

The CCWCB subroutine is used to build channel commands to carry out functions for which specific subroutines do not exist. Channel commands are short sequences of steps which, after initiation, are capable of operating independent of hybrid control. The primary advantage in using these here is the conservation of time. For example, one use of such commands in ATOD is to write data to digital tape while the

hybrid is freed from this function to begin further processing and prepare for the next event to be sampled. Without channel commands, and if events were closely spaced in time, the hybrid itself might not have sufficient time to control the output to digital tape, perform necessary record-keeping functions, and prepare for the detection of the next event before the occurrence of that event. Other uses of CCWCB are to write a tape mark (end of file) on and rewind the digital tape.

FSTIO is a subroutine which calls and causes execution of the channel commands constructed by CCWCB.

ITIME is a library subroutine which is used to compute the time of the initial event and each succeeding event. Subtraction of these times gives an elapsed time which is useful for comparison with the meter on the analog tape recorder to verify that the desired events were, in fact, selected.

Two other subroutines, TYPE (0) and TYPE (1), are used to print information on the operator's console and to both print on and read from the console respectively. TYPE (0) is used primarily to keep the operator informed of the program's progress and to indicate steps to be taken, if necessary, at certain points. TYPE (1) is used to input data or information codes at the appropriate times.

An Ampex SP300 seven-channel analog tape recorder is used to input data from magnetic tape to the SS100. Data is both recorded and played back in the FM mode. Recordings

are made at 1 7/8 ips and playback is at 15 ips - accelerated eight times. ATOD multiplexes four channels of EEG data and requires a fifth channel as an event marker.

The tape marking system used with this program consists of DC signals as recommended by Dumermuth and Fluhler (1967). Instead of a single DC marker, however, different DC amplitudes are used for timing and detecting different types of events. This program uses a logic detection system capable of identifying up to six different types of events. The system uses comparators on the SS100 analog patch panel to establish voltage windows for the different marker amplitudes. Figure 1 shows the circuit for this detection system.

Since the program was initially written for determining power spectra in different states from sleep to wakefulness, these events are labelled WAK (awake), CTL (control, a marker for determining system noise), SWS (slow wave sleep), REM (rapid-eye-movement sleep), MIS (for miscellaneous segments of the EEG record of interest), and INV (the invalid regions, those amplitudes not covered by the other markers). The INV signal was attributed to noise, but no INV events have occurred when the analog tapes are properly marked. The stages, their marker amplitudes and windows are illustrated in Table 1.

For purposes of the sleep EEG, division is made only according to two stages - SWS and REM - because of the rapid alternation between what would normally be labelled stages 1-4 (Rechtschaffen & Kales, 1968). At least one other study

Figure 1. Circuit Diagram for A-D Logic

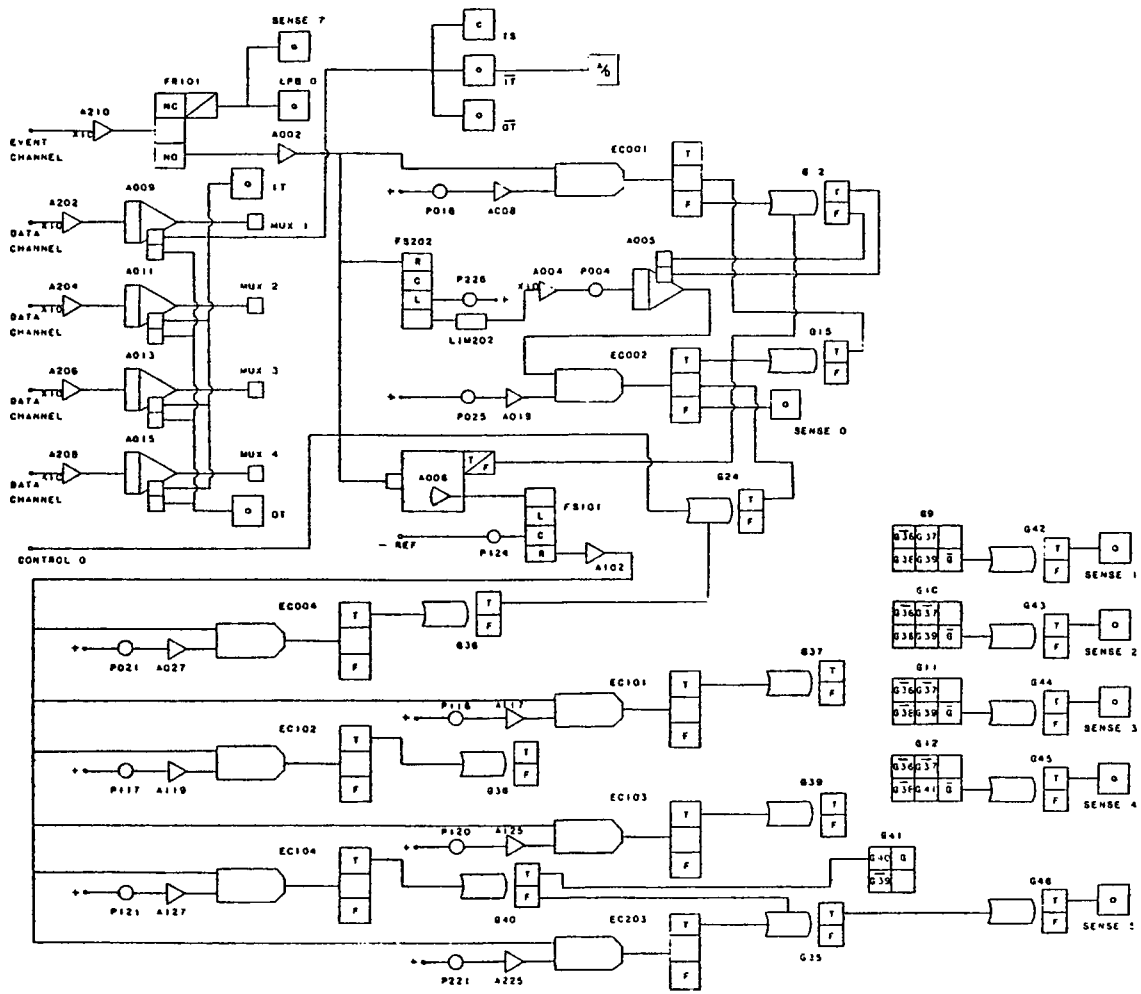
SENSE 7
FR101
LPB 0
A210
EVENT CHANNEL
A002
A202
DATA CHANNEL
A009
MUX 1
A204
A011
MUX 2
A206
A013
MUX 3
A208
A015
MUX 4
CONTROL 0
EC001
P018
AC08
FS202
P226
A004
P004
A005
LIM202
EC002
P025
A019
A006
FS101
REF
P124
A102
EC004
P021
A027
EC101
EC102
P117
A119
EC103
P120
A125
EC104
P121
A127
P221
A225
SENSE 0
SENSE 1
SENSE 2
SENSE 3
SENSE 4
SENSE 5

TABLE 1

EVENTS USED AS MARKERS FOR ATOD

Stage	Event	Amplitude(V)	Window(V)
1	CTL	0.3	0.2-0.45
2	WAK	0.6	0.45-0.75
3	SWS	0.9	0.75-1.05
4	REM	1.2	1.05-1.35
5	MIS	1.5	1.35-1.65
6	INV	-	0.0-0.2, >1.65

has observed the same effects in the chimp (Freemon, McNew, and Adey, 1971).

Since it is not feasible to detail every step of the program, I must rely on the program itself to clarify many points and limit myself to a general discussion of the rationale involved in the program and to brief explanations of why certain things were done the way they were. The program itself is presented in Figure 2. Many comment cards elaborate on steps taken in the program.

The overall approach in this program is to test sense lines until a specified logical value is returned, indicating that some action is necessary. The program then proceeds accordingly.

I preferred to write this program as one entity rather than as a main program and several subroutines because I would have had to COMMON virtually every label in use to maintain arrays, etc. The program requires extensive communication between sub-areas.

ATOD is much more efficient than either of the two previously existing A-D programs for EEG data at University of Houston (Osorio, 1976; Benignus, unpublished). It is an improvement over the Benignus program because of its accelerated playback capability; and it is an improvement over both programs because of its sampling of multiple channels and the capability of converting more than one analog

Figure 2. ATOD Program

```
C01   THE FOLLOWING PROGRAM IS INTENDED TO PERFORM A/D CONVERSION OF EEG
C     DATA AND WRITE RAW DATA TO TAPE.  IT USES UH ESSL IBM 360/44 AND
C     HYBRID SYSTEMS SS100.  ESSL "X" SUBROUTINES AND HYBRID SYSTEMS
C     PROGRAMS ARE USED WITHIN THIS PROGRAM.  A/D CONVERSION IS FOR 4
C     CHANNELS OF EEG DATA SIMULTANEOUSLY VIA MULTIPLEXING, WITH AN
C     ADDITIONAL CHANNEL SERVING AS AN EVENT MARK.  SAMPLING RATE IS
C     512/CHANNEL/SEC OVER A 16 SEC RT EPOCH.
C
C11   THIS PROGRAM HAS BEEN DESIGNED SUCH THAT MULTIPLE ANALOG TAPES MAY
C       BE RUN WITHOUT RE-INSERTING THE PROGRAM CARDS INTO THE CARD READER.
C
C12   AUTHOR'S ABBREVIATIONS USED IN THIS PROGRAM INCLUDE:
C     CTL:  EPOCH TO TEST FOR SYSTEM NOISE.  THIS EPOCH IS GENERATED BY
C       CUTTING ALL INPUT TO THE POLYGRAPH AND LETTING THE RECORDER RUN
C       WITH ALL NORMAL CONNECTIONS.  THIS WILL ALSO MEASURE ANY COMPUTER
C       NOISE INTRODUCED, BUT WILL NOT IDENTIFY THE SOURCE.
C     CTR:  COUNTER
C     ELTIM:  ELAPSED TIME SINCE INITIM
C     INITIM:  INITIAL TIME FOR EACH TAPE.  TO FOR EACH TAPE IS THE
C       TIME OF THE FIRST REGISTERED EVENT MARK
C     INV:  INVALID EVENT/EPOCH
C     MIS:  MISCELLANEOUS EVENT/EPOCH WHICH MUST BE SPECIFIED WITH THE
C       TAPE IN USE.
C     PB:  PLAYBACK RATE
C     RT:  REAL/RECORD TIME
C     REM:  RAPID EYE MOVEMENT EVENT/EPOCH
C     SWS:  SLOW WAVE SLEEP EVENT/EPOCH, ROUGHLY COMPARABLE TO STAGE 4.
C     WAK:  RESTING AWAKE EVENT/EPOCH
C     DELTAT:  ELAPSED TIME FROM ONE EVENT TO ANOTHER.
C
C07   THE EVENT SYSTEM USED TO MARK THE ANALOG SLEEP TAPES IS AS FOLLOWS:
C     A 1.5 VOLT POSITIVE PULSE IS DIRECTED TO A VOLTAGE DIVIDER WHICH
C     BREAKS THE SIGNAL INTO MULTIPLES OF 0.3 VOLTS.  LABELS FOR THESE ARE
C                                E = 0.3 V = CTL
C                              2*E = 0.9 V = WAK
C                              2*E = 0.6 V = WAK
C                              3*E = 0.9 V = SWS
C                              4*E = 1.2 V = REM
C                              5*E = 1.5 V = MIS
C     THE ANALOG AND LOGIC PATCH PANELS ARE WIRED AND SET ACCORDINGLY.
C
      LOGICAL XRDSNS
C04   LOGICAL XRDSNS CARD REQUIRED BY XRDSNS SUBROUTINE
C
      INTEGER*4 ELTIM(30,4), WAK(6,5), INV(6,5), SWS(6,5), REM(6,5),
     2  MIS(6,5), CTL(6,5), BYTCT, SA, EA, ICON, N
      INTEGER*2 CC, SLI, IHOLD, TAPMRK, TODGTP, LOC(32769,2), REWIND,
     2 DUH1, DUH2, CTLCTR, WAKCTR, SWSCTR, REMCTR,          STG, DELAY,
     3  CAT, TEST, DELTAT
      REAL*8 CWADR(3,5)
C
      N=32764
C05   N IS CALCULATED FROM (512 SAMPLES/ RT SEC / CHANNEL) ( 2 PB SEC )
C       ( 8 RT SEC/PB SEC) ( 4 CHANNELS)
C
      ICON=2
      SA=1
      EA=4
C       N, ICON, SA, EA, LOC(1,L) APPLY TO XRDAD SUBROUTINE
      ICODE=1
C       ICODE, IDATE APPLY TO ITIME SUBROUTINE
      ICHAN=4
      ICHK=0
      TAPMRK=31
      CC = 12
      SLI=4
      TODGTP=1
      REWIND=15
      BYTCT=21846
C       ICHAN, ICHK, TAPMRK, CC, SLI, TODGTP, REWIND. BYTCT APPLY TO
C       FSTIO SUBROUTINE AND COMMAND WORDS.
      DIMENSION IDATE(2)
C
C     THESE COMMAND-CHAINED INSTRUCTIONS (CWADR 1-3) WRITE DATA FROM 1
C     EVENT ONTO DIGITAL TAPE IN 3 EQUAL-LENGTH BLOCKS, EACH BYTCT BYTES
C     IN LENGTH.  THE FIRST 2 BYTES OF THE FIRST BLOCK CONTAIN THE LABEL
C     IDENTIFYING THE EVENT.
      CALL CCWCB (CWADR(1,1), TODGTP, CC, LOC(1,1), BYTCT)
      CALL CCWCB (CWADR(2,1), TODGTP, CC, LOC(10924  ,1), BYTCT)
      CALL CCWCB (CWADR(3,1), TODGTP,SLI, LOC( 21847   ,1), BYTCT)
      CALL CCWCB (CWADR(1,2), TODGTP, CC,  LOC(1,2), BYTCT)
      CALL CCWCB (CWADR(2,2), TODGTP, CC,  LOC(10924  ,2), BYTCT)
      CALL CCWCB (CWADR(3,2), TODGTP,SLI,  LOC( 21847   ,2), BYTCT)
      CALL CCWCB (CWADR(1,3), TAPMRK, SLI, DUH1, DUH2)
      CALL CCWCB (CWADR(1,4), REWIND, SLI, DUH1, DUH2)
C
      PAUSE  ' MOUNT SCRATCH TAPE ON 281--RING IN FOR WRITING'
C       TYPE R 00,'U' ON CONSOLE TO PROCEED AFTER PAUSE.
      CALL FSTIO (ICHAN, CWADR(1,4), ICHK)
C
C       THE FOLLOWING TEST IS DESIGNED TO PREVENT ENTRY INTO THE PROGRAM
C       IF ANALOG LOGIC PUSHBUTTON #7 HAS BEEN LEFT ON BY A PREVIOUS USER
C       OR FROM A PREVIOUS RUN.  THIS PUSHBUTTON IS USED TO END THIS PROGRAM.
C
      DO 92 I=1,2
      DO 55 J=32766,32769
```

```
      LOC(J,I) = 0
   55 CONTINUE
   92 CONTINUE
   54 IF (.NOT. XRDSNS (8)) GOTO 95
      WRITE (15,94)
   94 FORMAT (' LOGIC PB 7 IS ON.  IT MUST BE TURNED OFF TO PROCEED.')
      CALL TYPE (0)
   93 IF (.NOT. XRDSNS (8)) GOTO95
      GOTO 93
   95 M=0
      L=1
      CTLCTR=0
      WAKCTR=0
      SWSCTR=0
      REMCTR=0
      MISCTR=0
      INVCTR=0
      CAT=0
      TEST=0
      DELAY=0
      IHOLD=0
C        IHOLD APPLIES TO ESSL SUBROUTINES XSTCTL AND XRSCTL
      INITIM=0
      ICLOCK=0
      CALL XRSCTL(IHOLD,0)
C        XRSCTL IS NECESSARY TO SET INITIAL LOGIC ON ANALOG.
      DO 51 I=1,30
        DO 51 J=1,4
        ELTIM (I,J) = 0
   51 CONTINUE
      DO 52 I=1,6
        DO 53 J=1,5
        CTL (I,J) =0
        WAK(I,J)=0
        SWS (I,J) =0
        REM (I,J) =0
        MIS (I,J) =0
        INV(I,J)=0
   53   CONTINUE
   52 CONTINUE
C
C**   SENSE LINE 0 RETURNS TRUE AS LONG AS ONLY NOISE LEVELS ARE PRESENT
C        ON THE MARKER CHANNEL OF ANALOG TAPE.  ANY SIGNAL APPEARING ON
C        THIS CHANNEL WILL DIRECT SEQUENTIAL TESTING OF THE FOLLOWING
C        SENSE LINES AND DIRECTION TO THE APPROPRIATE SUB-AREA OF THE PROGRAM.
C        THIS TESTING ASSUMES A DELAY BUILT INTO THE LOGIC CIRCUITRY TO VALIDATE
C        A SIGNAL, AS WELL AS A PROGRAM DELAY BELOW.  THE PROGRAM DELAY IS
C        NECESSARY TO GIVE THE EVENT MARK TIME TO RISE TO ITS FULL AMPLITUDE

C        BEFORE TESTING SENSE LINES.  IF THIS DELAY IS NOT PRESENT, CTL WILL
C        RETURN EACH TIME, REGARDLESS OF THE EVENT MARK ACTUALLY PRESENT.
C
   30 WRITE (15,26)
   26 FORMAT (1H1, ' ENTER DESIRED DELAY IN 1X,I5 FORMAT.')
      CALL TYPE (1)
      READ (15,27) DELAY
   27 FORMAT (1X,I5)
      WRITE (15,90)
   90 FORMAT (' ENTER CAT ID AND TEST TAPE NOS. IN "1X,I2,1X,I2" FORMAT.
     2  IF A TEST TAPE, ENTER 00 FOR CAT ID AND VICE VERSA.')
      CALL TYPE (1)
      READ (15,91) CAT, TEST
   91 FORMAT (1X,I2,1X, I2)
   99 IF (XRDSNS(6). OR. XRDSNS(7)) GOTO 98
      GOTO 7
   98 WRITE (15,97)
   97 FORMAT (' LOGIC PBS #0 AND/OR #1 HAVE BEEN LEFT ON.  PROGRAM  CAN
     2NOT BE EXECUTED UNTIL THEY ARE BOTH OFF.')
      CALL TYPE (0)
   96 IF (.NOT.XRDSNS(6).AND..NOT. XRDSNS(7)) GOTO 7
      GOTO 96
    7 WRITE (15,45)
   45 FORMAT (' IF CONTROL LAMP #1 ON ANALOG PANEL IS FLASHING, HYBRID I
     2S READY TO RECEIVE DATA FROM ANALOG TAPE.')
      CALL TYPE(0)
   46 IF (XRDSNS(7)) GOTO 5
C
C**   THE TWO DO LOOPS INDICATED BELOW WILL CAUSE CONTROL LAMP 1 ON THE
C     ANALOG CONSOLE TO FLASH AT A RATE OF ABOUT 1/SEC.  THIS INDICATES
C     THAT THE HYBRID IS READY TO RECEIVE DATA FROM TAPE.
C
      CALL XSTCTL (IHOLD,1)
      DO 8 I=1,30000
    8 CONTINUE
      IF (XRDSNS(8)) GOTO 3
      CALL XRSCTL (IHOLD,1)
      DO 9 I=1,30000
    9 CONTINUE
      GOTO 46
    5 CONTINUE
C
C**   TO END THE PROGRAM MANUALLY FOR A GIVEN TAPE, PUSH LOGIC PUSH-
C     BUTTON 1.  THIS CAUSES SNS LINE 6 TO RETURN A TRUE AND WILL INITIATE
C     PRINTOUT OF CTR VALUES, ETC. FOR THAT TAPE.  TO END THE ENTIRE
C     PROGRAM SEE C12.
C
    1 IF (XRDSNS(6).OR..NOT. XRDSNS(0)) GOTO 2
```

```
      GOTO 1
    2 CALL XSTCTL (IHOLD,0)
      DO 25 I=1, DELAY
   25 CONTINUE
      IF (XRDSNS(1)) GOTO 100
      IF (XRDSNS(2)) GOTO 200
      IF (XRDSNS(3)) GOTO 300
      IF (XRDSNS(4)) GOTO 400
      IF (XRDSNS(5)) GOTO 500
      IF (XRDSNS(6)) GOTO 12
      GOTO 600
    4 CALL XRSCTL (IHOLD,0)
    6 IF (XRDSNS(0)) GOTO 5
      IF (.NOT.XRDSNS(0)) GOTO 6
      GOTO 5
   12 IF (CAT.EQ.0) GOTO 73
      WRITE (6,71) CAT, M
   71 FORMAT (/, ' THE TOTAL NUMBER OF EVENTS DETECTED FOR CAT NO.', I2,
     2  ' WAS ', I2,'.')
      GOTO 75
   73 WRITE (6,74) TEST, M
   74 FORMAT (/, ' THE TOTAL NUMBER OF EVENTS DETECTED FOR TEST TAPE NO.
     2',I2, ' WAS ', I2, '.')
   75 WRITE (6,72)
   72 FORMAT (/, ' THE EVENTS ARE LISTED IN ORDER OF OCCURRENCE.', /,
     2 2X, 'EVENT', 3X, 'ELAPSED TIME', 3X, 'STAGE', 2X, 'COUNTER', 6X,
     3  'TAPE LABEL', 4X, 'DELTA T',/)
      DO 13 I=1,M
      IF (I.GT.1) GOTO 40
      DELTAT = 0
      GOTO41
   40 DELTAT = ELTIM(I,1) - ELTIM(I-1,1)
   41 WRITE (6,14) I,(ELTIM(I,J), J=1,4), DELTAT
   14 FORMAT (4X, I2, 6X, I7, 8X, I1,  8X,  I1, 10X, I5, 8X, I5)
   13 CONTINUE
      WRITE (6,31)
   31 FORMAT (' THE NUMBER OF EVENTS DETECTED FOR EACH STG IS LISTED.')
      WRITE (6,11) CTLCTR, WAKCTR, SWSCTR, REMCTR, MISCTR, INVCTR
   11 FORMAT (/,' CTLCTR = ',I2,5X, 'WAKCTR = ',I2,5X, 'SWSCTR = ',I2,
     2  5X, 'REMCTR = ',I2,5X, 'MISCTR = ', I2,5X,'INVCTR = ',I2)
      WRITE (6,57)
   57 FORMAT (/, 2X, 'CUMULATIVE EVENT', 7X, 'CTL EVENT', 6X, 'ELAPSED T
     2IME', 6X, 'BUFFER', 4X, 'TAPE LABEL', /)
      WRITE (6,77) ((CTL (J,K), K=1,5), J=1,CTLCTR)
   77 FORMAT    (8X, I2, 19X, I1, 12X, I7, 12X, I1, 8X, I5)
      WRITE (6,56)
   56 FORMAT (/, 2X, 'CUMULATIVE EVENT', 7X, 'WAK EVENT', 6X, 'ELAPSED T
     2IME', 6X, 'BUFFER', 4X, 'TAPE LABEL', /)

      WRITE (6,77) ((WAK (J,K), K=1,5), J=1,WAKCTR)
      WRITE (6,58)
   58 FORMAT (/, 2X, 'CUMULATIVE EVENT', 7X, 'SWS EVENT', 6X, 'ELAPSED T
     2IME', 6X, 'BUFFER', 4X, 'TAPE LABEL', /)
      WRITE (6,77) ((SWS (J,K), K=1,5), J=1,SWSCTR)
      WRITE (6,59)
   59 FORMAT (/, 2X, 'CUMULATIVE EVENT', 7X, 'REM EVENT', 6X, 'ELAPSED T
     2IME', 6X, 'BUFFER', 4X, 'TAPE LABEL', /)
      WRITE (6,77) ((REM (J,K), K=1,5), J=1,REMCTR)
      WRITE (6,60)
   60 FORMAT (/, 2X, 'CUMULATIVE EVENT', 7X, 'MIS EVENT', 6X, 'ELAPSED T
     2IME', 6X, 'BUFFER', 4X, 'TAPE LABEL', /)
      WRITE (6,77) ((MIS (J,K), K=1,5), J=1,MISCTR)
      WRITE (6,61)
   61 FORMAT (/, 2X, 'CUMULATIVE EVENT', 7X, 'INV EVENT', 6X, 'ELAPSED T
     2IME', 6X, 'BUFFER', 4X, 'TAPE LABEL', /)
      WRITE (6,77) ((INV (J,K), K=1,5), J=1,INVCTR)
C
C12   IF NO FURTHER A/D PROCESSING IS ANTICIPATED IMMEDIATELY, PRESS
C       LOGIC PUSHBUTTON 7.  THIS WILL CAUSE SENSE LINE 8 TO BECOME TRUE
C       AND END THIS PROGRAM.
C
    3 IF (XRDSNS(8)) GOTO 10
      GOTO 95
   10 CALL FSTCHK (ICHAN)
      CALL FSTIO (ICHAN, CWADR(1,3), ICHK)
      CALL FSTIO (ICHAN, CWADR(1,4), ICHK)
      STOP
  100 IF (L.EQ.1) GOTO 101
      L=1
      GOTO 102
  101 L=2
  102 CALL XRDAD (N, ICON, SA, EA,LOC(1,L))
      CALL ITIME (ICODE, IDATE, ICLOCK)
C
C14   ITIME SUBROUTINE FOUND IN PROGRAM LIBRARY USERS GUIDE, P. 29.
C       IT IS USED TO MEASURE ELTIME AND GIVE RELATIVE LOCATIONS OF
C       EVENTS REGISTERED FROM ANALOG TAPE.
C
      IF (CTLCTR.EQ.0.AND.M.EQ.0) GOTO 103
  104 CTLCTR = CTLCTR +1
      M = M+1
      STG = 1
      LOC (1,L) = TEST*2048 + CAT*64 + STG*8 + CTLCTR
      CALL FSTCHK(ICHAN)
      CALL FSTIO (ICHAN, CWADR(1,L), ICHK)
      ELTIM (M,1) = ICLOCK - INITIM
      ELTIM (M,2) = STG
```

```
      ELTIM (M,3) = CTLCTR
      ELTIM (M,4) = LOC (1,L)
      CTL (CTLCTR,1) =M
      CTL (CTLCTR,2) = CTLCTR
      CTL (CTLCTR,3) = ELTIM (M,1)
      CTL(CTLCTR,4) = L
      CTL (CTLCTR,5) = LOC (1,L)
      IF(TEST.NE.1) GOTO 4
   80 FORMAT (/,' THE FOLLOWING OUTPUTS PRODUCE SOME SAMPLE DATA FROM TE
     2ST TAPE 1.  IT CONSISTS OF FIRST AND LAST 100 DATA POINTS AND LABE
     3L', /,' OF EACH EVENT IN SEQUENCE AS TAKEN FROM TEST TAPE.')
      WRITE (6,80)
      WRITE (6,78) (LOC(J,L), J=1,101)
   78 FORMAT (/,' LABEL AND FIRST 100 DATA POINTS IN LOC ARE ', I5, /,
     2    (20 (1X, I5)))
      WRITE (6,81) (LOC(J,L), J=10874, 10973)
   81 FORMAT (/,' DATA COVERING LAST 50 POINTS OF FIRST TAPE BLOCK AND
     2FIRST 50 POINTS OF SECOND BLOCK ARE', /, (20(1X,I5)))
      WRITE (6,82) (LOC(J,L), J=21797, 21896)
   82 FORMAT (/,' DATA COVERING LAST 50 POINTS OF SECOND BLOCK AND FIRS
     2T 50 POINTS OF THIRD BLOCK ARE', /, (20(1X,I5)))
      WRITE (6,79) (LOC(J,L), J=32670, 32769)
   79 FORMAT (/,' LAST 100 DATA POINTS FROM LOC ARE'./, (20(1X,I5)  ))
      GOTO 4
  103 INITIM=ICLOCK
      GOTO 104
  200 IF (L.EQ.1) GOTO 201
      L=1
      GOTO 202
  201 L=2
  202 CALL XRDAD (N, ICON, SA, EA,LOC(1,L))
C
C14   ITIME SUBROUTINE FOUND IN PROGRAM LIBRARY USERS GUIDE, P. 29.
C       IT IS USED TO MEASURE ELTIM  AND GIVE RELATIVE LOCATIONS OF
C        EVENTS REGISTERED FROM ANALOG TAPE.
C
      CALL ITIME (ICODE, IDATE, ICLOCK)
      IF (M.GT.0) GOTO 203
      INITIM =ICLOCK
  203 WAKCTR = WAKCTR+1
      M=M+1
      STG=2
      LOC (1,L) = TEST*2048 + CAT*64 + STG*8 + WAKCTR
      CALL FSTCHK(ICHAN)
      CALL FSTIO (ICHAN, CWADR(1,L), ICHK)
      ELTIM (M,1) = ICLOCK - INITIM
      ELTIM (M,2) = STG
      ELTIM (M,3) = WAKCTR

      ELTIM (M,4) = LOC (1,L)
      WAK (WAKCTR,1) =M
      WAK (WAKCTR,2) = WAKCTR
      WAK (WAKCTR,3) = ELTIM (M,1)
      WAK (WAKCTR,4) = L
      WAK (WAKCTR,5) = LOC (1,L)
      IF(TEST.NE.1) GOTO 4
      WRITE (6,78) (LOC(J,L), J=1,101)
      WRITE (6,81) (LOC(J,L), J=10874, 10973)
      WRITE (6,82) (LOC(J,L), J=21797, 21896)
      WRITE (6,79) (LOC(J,L), J=32670, 32769)
      GOTO 4
  300 IF (L.EQ.1) GOTO 301
      L=1
  302 CALL XRDAD (N, ICON, SA, EA,LOC(1,L))
      CALL ITIME (ICODE, IDATE, ICLOCK)
      IF (M.GT.0) GOTO 303
      INITIM =ICLOCK
  303 SWSCTR = SWSCTR+1
      M=M+1
      STG=3
      LOC (1,L) = TEST*2048 + CAT*64 + STG*8 + SWSCTR
      CALL FSTCHK(ICHAN)
      CALL FSTIO (ICHAN, CWADR(1,L), ICHK)
      ELTIM (M,1) = ICLOCK - INITIM
      ELTIM (M,2) = STG
      ELTIM (M,3) = SWSCTR
      ELTIM (M,4) = LOC (1,L)
      SWS (SWSCTR,1) =M
      SWS (SWSCTR,2) = SWSCTR
      SWS (SWSCTR,3) = ELTIM (M,1)
      SWS (SWSCTR,4) =L
      SWS (SWSCTR,5) = LOC (1,L)
      IF(TEST.NE.1) GOTO 4
      WRITE (6,78) (LOC(J,L), J=1,101)
      WRITE (6,81) (LOC(J,L), J=10874, 10973)
      WRITE (6,82) (LOC(J,L), J=21797, 21896)
      WRITE (6,79) (LOC(J,L), J=32670, 32769)
      GOTO 4
  301 L=2
      GOTO 302
  400 IF (L.EQ.1) GOTO 401
      L=1
  402 CALL XRDAD (N, ICON, SA, EA,LOC(1,L))
      CALL ITIME (ICODE, IDATE, ICLOCK)
      IF (M.GT.0) GOTO 403
      INITIM =ICLOCK
  403 REMCTR=REMCTR+1
```

```
      M=M+1
      STG =4
      LOC (1,L) = TEST*2048 + CAT*64 + STG*8 + REMCTR
      CALL FSTCHK(ICHAN)
      CALL FSTIO (ICHAN, CWADR(1,L), ICHK)
      ELTIM (M,1) = ICLOCK - INITIM
      ELTIM (M,2) = STG
      ELTIM (M,3) = REMCTR
      ELTIM (M,4) = LOC (1,L)
      REM (REMCTR,1) = M
      REM (REMCTR,2) = REMCTR
      REM (REMCTR,3) = ELTIM (M,1)
      REM (REMCTR,4) =L
      REM (REMCTR,5) = LOC (1,L)
      IF(TEST.NE.1) GOTO 4
      WRITE (6,78) (LOC(J,L), J=1,101)
      WRITE (6,81) (LOC(J,L), J=10874, 10973)
      WRITE (6,82) (LOC(J,L), J=21797, 21896)
      WRITE (6,79) (LOC(J,L), J=32670, 32769)
      GOTO 4
401   L=2
      GOTO 402
500   IF (L.EQ.1) GOTO 501
      L=1
502   CALL XRDAD (N, ICON, SA, EA,LOC(1,L))
      CALL ITIME (ICODE, IDATE, ICLOCK)
      IF (M.GT.0) GOTO 503
      INITIM =ICLOCK
503   MISCTR = MISCTR+1
      M=M+1
      STG =5
      LOC (1,L) = TEST*2048 + CAT*64 + STG*8 + MISCTR
      CALL FSTCHK(ICHAN)
      CALL FSTIO (ICHAN, CWADR(1,L), ICHK)
      ELTIM (M,1) = ICLOCK - INITIM
      ELTIM (M,2) = STG
      ELTIM (M,3) = MISCTR
      ELTIM (M,4) = LOC (1,L)
      MIS  (MISCTR,1) = M
      MIS (MISCTR,2) = MISCTR
      MIS (MISCTR,3) = ELTIM (M,1)
      MIS (MISCTR,4) = L
      MIS (MISCTR,5) = LOC (1,L)
      IF(TEST.NE.1) GOTO 4
      WRITE (6,78) (LOC(J,L), J=1,101)
      WRITE (6,81) (LOC(J,L), J=10874, 10973)
      WRITE (6,82) (LOC(J,L), J=21797, 21896)
      WRITE (6,79) (LOC(J,L), J=32670, 32769)

      GOTO 4
501   L=2
      GOTO 502
600   IF (L.EQ.1) GOTO 601
      L=1
602   CALL XRDAD (N, ICON, SA, EA, LOC(1,L))
      CALL ITIME (ICODE, IDATE, ICLOCK)
      IF (M.GT.0) GOTO 603
      INITIM =ICLOCK
603   INVCTR = INVCTR+1
      M=M+1
      STG=6
      LOC (1,L) = TEST*2048 + CAT*64 + STG*8 + INVCTR
      CALL FSTCHK(ICHAN)
      CALL FSTIO (ICHAN, CWADR(1,L), ICHK)
      ELTIM (M,1) = ICLOCK - INITIM
      ELTIM (M,2) = STG
      ELTIM (M,3) = INVCTR
      ELTIM (M,4) = LOC (1,L)
      INV (INVCTR,1) = M
      INV (INVCTR,2) = INVCTR
      INV (INVCTR,3) = ELTIM (M,1)
      INV (INVCTR,4) = L
      INV (INVCTR,5) = LOC (1,L)
      IF(TEST.NE.1) GOTO 4
      WRITE (6,78) (LOC(J,L), J=1,101)
      WRITE (6,81) (LOC(J,L), J=10874, 10973)
      WRITE (6,82) (LOC(J,L), J=21797, 21896)
      WRITE (6,79) (LOC(J,L), J=32670, 32769)
      GOTO 4
601   L=2
      GOTO 602
      END
```

tape by recycling to an initial point in the program. These features save a great amount of computer time and permit cross-channel EEG analyses not possible before at this institution on digital equipment.

The program assumes a 16-second real time epoch (2 seconds accelerated playback) for each event. Sampling is performed simultaneously across the four data channels and held until the data are multiplexed into the memory location provided. Sampling rate is controlled by the SS100 master timer and is set at 512.3 per second real time.

I selected the rate of 512 per second to avoid major aliasing problems. The highest frequencies present in the data above noise levels are about 240 Hz, and these seem to come most often from the tape recorder itself. Had I used a lower rate, such as 256 per second, this 240 Hz activity would have aliased back into the EEG frequency range of interest at about 16 Hz, and this was clearly an unsatisfactory condition.

The program has two major "safety checks" built in, and these involve the logic pushbuttons on the SS100 console and the sense lines used in conjunction with them. Three of these pushbuttons (0, 1, and 7) are used to set logic for ATOD. Since it is possible, and even highly likely, that these pushbuttons may be inadvertently left on after the function they control is completed, tests for this are written into the beginning of the program. If the sense lines are true, a message is printed on the operator's

console instructing the operator to turn them off. The program then enters a holding loop and cannot proceed until the required pushbuttons are turned off.

Another check is wired into the logic circuitry of the SS100. This is a "validation circuit" for the event mark on analog tape (discussed earlier). Since there are occasional "glitches" on the event channel, it was anticipated that these might cause selection of an unwanted epoch. The validation procedure consists of integrating a constant voltage signal upon detection of a pulse from the event channel. If the channel pulse remains for 0.8 seconds real time (0.1 second playback time), it is considered a valid pulse and processing for sampling begins. If the pulse is not constant for this interval, the integrator resets to zero and awaits the next pulse. This 0.1 second interval was selected arbitrarily, but observation of sampled events and the computer output clearly show that no unwanted epochs have been sampled.

The program begins by declaring and initializing certain labels, and by building certain channel commands. Due to the large amount of data acquired from each epoch, three commands are chained to write the data to tape in three separate blocks. (The maximum number of bytes that can be written to digital tape in one block is 32,760 (IBM, 1973); but each epoch produces 65,538 bytes. In order to simplify writing this program and the power spectrum program to be discussed later, these data are written to tape in three

equal-sized blocks of 21,846 bytes each.) Other commands are built to write the ENDFILE on and rewind the digital tape.

Next, the first of the safety checks provides an instruction if logic pushbutton (LPB) #7 has been left on.

Label 95 is the first point of actual processing for an analog tape. When multiple analog tapes are being digitized, this is the point to which the program recycles.

The DELAY read from the console in label 27 was inserted because I observed that the hybrid system cycled to begin sampling so quickly that all event marks were interpretted as being of the smallest valid amplitude for an event mark, regardless of the actual amplitude. The apparent reason for this was that the marker took milliseconds to rise to amplitude while the computer was cycling in microseconds. A DO loop was written to provide this delay. This delay and the validation integrator discussed earlier are redundant systems to some extent, but both are used.

ATOD then encounters the point for entering the CAT number. When recycling with different analog tapes, this step provides a place to enter the new subject number.

The second safety check for the LPB-sense lines occurs next. Then the program progresses to the point where it is ready to accept data from analog tape, a DO loop is used to flash a control line indicator on the analog panel. After starting the analog tape recorder, depressing LPB #0 begins the testing of sense lines for an event. Either depressing LPB #1 or detection of an event from tape will direct the program to a sequential testing of sense lines 1-6.

Line 6 occurs as a result of LPB #1 and is used to end processing for a given tape. This then causes printout of the tabulated data for the cat indicated, and recycles the program to label 95 discussed above. Sample data printouts are presented in Figure 3. The first of these tables lists the events detected in temporal order. It provides the elapsed time of each event relative to the first, the type of event, the number of the event within that type, the label used to identify that event, and the time between events. The second of the tables provides similar information, but it is arranged sequentially according to the type of event.

Lines 1-5 direct the program to the region indicated by the event mark detected from tape. Within each of these regions, the labels are numbered 100 through the 600's for purposes of easily distinguishing which region is being considered. The 600 series are reserved for when an event has been detected but does not fall within the windows provided for a valid event.

Each of these regions performs similar tasks; they are differentiated only by the different types of events.

Two data buffers are used in these regions. Both perform the same function, but they are used in an alternating sequence. The purpose in having two buffers is primarily precautionary. I was worried about the possibility of having two closely spaced events and encountering problems because I was attempting to store new information in a buffer

THE TOTAL NUMBER OF EVENTS DETECTED FOR CAT NO. 7 WAS 10.

THE EVENTS ARE LISTED IN ORDER OF OCCURRENCE.

EVENT	ELAPSED TIME	STAGE	COUNTER	TAPE LABEL	DELTA T
1	0	1	1	457	0
2	4526	2	1	465	4526
3	7487	2	2	466	2961
4	9065	2	3	467	1578
5	14944	3	1	473	5879
6	22619	3	2	474	7675
7	23920	3	3	475	1301
8	29949	4	1	481	6029
9	31162	4	2	482	1213
10	32383	4	3	483	1221

THE NUMBER OF EVENTS DETECTED FOR EACH STG IS LISTED.

CTLCTR = 1 WAKCTR = 3 SWSCTR = 3 REMCTR = 3 MISCTR = 0 INVCTR = 0

CUMULATIVE EVENT	CTL EVENT	ELAPSED TIME	BUFFER	TAPE LABEL
1	1	0	2	457

CUMULATIVE EVENT	WAK EVENT	ELAPSED TIME	BUFFER	TAPE LABEL
2	1	4526	1	465
3	2	7487	2	466
4	3	9065	1	467

CUMULATIVE EVENT	SWS EVENT	ELAPSED TIME	BUFFER	TAPE LABEL
5	1	14944	2	473
6	2	22619	1	474
7	3	23920	2	475

CUMULATIVE EVENT	REM EVENT	ELAPSED TIME	BUFFER	TAPE LABEL
8	1	29949	1	481
9	2	31162	2	482
10	3	32383	1	483

CUMULATIVE EVENT	MIS EVENT	ELAPSED TIME	BUFFER	TAPE LABEL
0	0	0	0	0

CUMULATIVE EVENT	INV EVENT	ELAPSED TIME	BUFFER	TAPE LABEL
0	0	0	0	0

Figure 3. ATOD Output Tables

while a WRITE command was still in the process of writing the old data to digital tape. The L in each of the sub-areas of the program is the label which indicates the buffer, and these are alternated at the beginning of each procedure.

Following buffer re-assignment, XRDAD is called to digitize the epoch. ITIME supplies the time, and counters are incremented to assist in identifying the event later.

A label is assigned to the multiplexed data, and both the label and data are written to digital tape using FSTIO.

Counter values and other information are then assigned to the arrays which are tabulated at the end of the analog tape and the program recycles to label 4 to begin searching for the next event.

When a digital tape has been filled, or an end to processing is desired, depressing LPB #7 causes an end file mark on and rewinds the digital tape before ending the program. If LPB #7 is pushed simultaneously with LPB #1, the program will end without recycling to label 95. If not, data such as CAT, etc. will have to be entered before the program ends.

CHAPTER III

POWER SPECTRA

Theory and General Considerations

Estimations of power spectra today are usually performed either on special-purpose computers or on large general-purpose digital computers. Since a general-purpose digital computer, an IBM 360/44, was used for estimations in this program, only the techniques used in digital computations will be discussed here.

The general technique is to perform a Fourier transform of the EEG data points, compute power, carry out a smoothing operation, and plot the power spectrum. The Fourier transform is a mathematical tool for converting (transforming) data from the time domain to the frequency domain, where an estimate of power can be made (e.g., Dumermuth, 1973).

The power spectrum may be defined mathematically as:

$$G_x(f) = 2 \lim_{T \to \infty} \frac{1}{T} E \left[|X(f,T)|^2 \right]$$

where $X(f,T) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi ft} dt$

which is the Fourier transform of the time domain signal x(t) (Bendat & Piersol, 1971). The continuous Fourier transform has integration limits from $-\infty$ to $+\infty$, but in estimations involving actual computations the discrete

Fourier transform is used (e.g., Bergland, 1969). The discrete transform can be written:

$$X(j) = \frac{1}{N} \sum_{k=0}^{N-1} x(k) e^{-i2\pi jk/N}$$

where N is the number of data points sampled from the time series x(k).

Dietsch (1932) was the first to use the Fourier transform to obtain a frequency spectrum. The calculations were extremely time-consuming, however, and I have found no further attempts to use the mathematical transformations until high-speed digital computers appeared in the late 1950's and early 1960's. Several studies (Grass & Gibbs, 1938; Knott, Gibbs, & Henry, 1942; etc.) claim to use the Fourier transform, but in reality have used analog measures instead of the mathematical computation.

Even the use of digital machines to compute the Fourier transform was a time-consuming, and therefore expensive, process. For this reason, the autocorrelation was used to compute power spectra indirectly until 1965. Dumermuth and Fluhler (1967) present data showing a comparison of computation times required on one digital machine for different methods of computing power spectra. For a sample containing 8192 data points, the Fourier transform required about 204 minutes to compute a power spectrum; the indirect method using autocorrelation required approximately 15 minutes. A new technique, known as the Fast Fourier Transform (FFT),

was introduced in 1965 (Cooley & Tukey, 1965), and this greatly reduced computation time. In Dumermuth and Fluhler's (1967) terms, the power spectrum now required only about 70 seconds.

The dramatic decrease in computing time is due to a special convention which uses a number of sample data points equal to an integer power of 2. As Cooley and Tukey (1965) illustrated, this reduces the number of operations necessary from N^2 to $2N\log_2 N$, where N is the number of data points in the sample to be transformed. A graphic comparison of the number of operations in the two methods is presented in Bergland (1969). The mathematical details involved in computing the transform via the FFT may be found in several publications (e.g., Cooley and Tukey, 1965; Bergland, 1969; Cochran, et al, 1967) and will not be discussed here. FFT algorithms are now frequently maintained in the libraries of large computing facilities, available on call.

When computing power spectra using the Fourier transform, a phenomenon called leakage occurs. At the frequency under consideration, a certain power value is obtained, but at adjacent points on either side of this frequency the computed power oscillates in a damped fashion between negative and positive values. Figures illustrating this may be found in Bendat and Piersol (1971), Dumermuth (1973), etc. The introduction of these oscillations (or "side lobes") is the leakage. Since these side lobes contribute to the power calculated at these other frequencies, leakage is obviously undesirable.

Two methods devised to reduce leakage are tapering and Hanning. Tapering is the application of a cosine function to both ends of the time series data. Bingham (1967) recommends the use of the cosine functions

$$\tfrac{1}{2}\left(1-\cos\pi\,\frac{t}{0.1T}\right)$$

and $$\tfrac{1}{2}\left(1-\cos\pi\,\frac{T-t}{0.1T}\right)$$

where t = data points from 0 to T-1

and T = total number of data points

for the beginning and the end of the data sequence, respectively.

Hanning is the weighting of Fourier coefficient, after transformation to the frequency domain, but prior to squaring and adding, of the form

$$A_k = -.25a_{k-1} + .5a_k - .25a_{k+1}$$

According to Bingham (1967), Hanning after squaring and adding as Osorio (1976) did is useless for reducing leakage.

The EEG must be considered random data, as opposed to deterministic data, because it cannot be represented by an explicit mathematical relation. According to Bendat and Piersol (1971), there are two types of random data - stationary and non-stationary. Stationarity of a signal is defined as a signal, all of whose moments and joint moments are time invariant. Thus, if a signal has a constant mean, variance,

etc. when sampled at different points in time, it is stationary. For practical purposes, however, a signal is often considered stationary if it has a constant mean and a joint moment (autocorrelation) which depends only on the lag, or time displacement, of the signal with respect to itself.

In addition to stationarity, the distribution of the signal about its mean amplitude value must be considered. If the signal is normally distributed, it is said to be Gaussian (Bendat and Piersol, 1971); otherwise it is non-Gaussian.

According to Dumermuth (1973), however, the EEG is usually neither stationary nor Gaussian. The EEG is usually assumed to be both stationary and Gaussian, though, for purposes of approximation. The computations involved in non-stationary data are much more complex (Bendat and Piersol, 1971).

Power Spectra Program

The procedures used to compute power spectra in the program described here generally follow those recommended by Bendat and Piersol (1971). The program itself, with the subroutines, is given in Figure 4. The sequence involves tapering both ends of the data, transforming via an FFT, computation of raw power, and averaging to reduce the resolution to that desired.

Figure 4. PWRSPC Program

```
C     THIS PROGRAM AND THE SUBROUTINES IT CALLS COMPUTE, PLOT, AND WRITE
C     TO TAPE THE POWER SPECTRUM FOR EACH CHANNEL OF EEG DATA, AS DIRECTED.
C     CERTAIN STATISTICAL PARAMETERS ARE ALSO COMPUTED.  THE PROGRAM INTERFACES
C     WITH THE ESSL CONSOLE FOR CHECKING AND INPUTTING DATA.  THE HYBRID
C     MODE IS NECESSARY BECAUSE OF THE TYPE SUBROUTINE WHICH WRITES AND READS
C     FROM THE CONSOLE, BECAUSE OF THE RDTAPE SUBROUTINE, AND BECAUSE
C     THE CHANNEL COMMANDS ARE REQUIRED TO MANIPULATE THE HYBRID TAPE.
      INTEGER*2 ODATA(32769), BLOCK1(16923), BLOCK2(16923), BLOCK3
     2 (16923), DARRAY(4,8192), EPOCTR, WHATDO, PRNTCK, CHANGE, CAT,
     3  TRCHNL(4), BRNLOC(4), STAGE, STGCTR, CATSTD, POINT, CHANL,
     4  FRSCOD, SMUCOD, EPOSKP, HANCOD,BKSPAC, SLI, BSPACE, PLTCOD,
     5  TPRCOD, REWIND, REWUNL, TSTPRT
      INTEGER TEST, TSTSTD, PLOTNO, SMPFRQ, TPRPTS, DUM1, DUM2
      DIMENSION DATA(8192), PRAMTR(15), SMUSPC(512), TPRCOS(820),
     2  RAWSPC(4096)
      REAL MINPWR, MAXPWR, MEAN, KURT
      REAL*8 CWADR1, CWADR2, CWADR3, CWADR4
      EQUIVALENCE (ODATA(2), DARRAY(1,1)),(ODATA(1), BLOCK1),(ODATA
     2  (16924), BLOCK2), (ODATA(21847), BLOCK3), (PRAMTR(1), SUM),
     3  (PRAMTR(2), SUMSQ), (PRAMTR(4), SUM4TH), (PRAMTR(5), MEAN),
     4  (PRAMTR(6), VAR), (PRAMTR(7),STDDEV), (PRAMTR(8), SKEW),
     5  (PRAMTR(9), KURT), (PRAMTR(10), DATPTS), (PRAMTR(11), MINPWR),
     6  (PRAMTR(12), MAXPWR), (PRAMTR(13), TOTPWR), (PRAMTR(3), SUMCUB)
      BSPACE=39
      SLI=4
      ICHAN=3
      ICHK=1
      REWIND=7
      CALL CCWCB (CWADR1, BSPACE, SLI, DUM1, DUM2)
      CALL CCWCB (CWADR2, REWIND, SLI, DUM1, DUM2)
      CALL TOPEN
      EPOCTR = 0
      NUMPLT = 0
      CATSTD = 0
      TSTSTD = 0
C     PAUSE 'MOUNT ODATA TAPE 280, SPCTRA TAPE 281.  FILE PROTECT 280.'
      REWIND 1
      CALL FSTIO (ICHAN, CWADR2, ICHK)
  801 WRITE (6,60)
   60 FORMAT ('1')
      WRITE (15,2)
    2 FORMAT (' ENTER LBLSTD IN 1X,I5 FORMAT')
      CALL TYPE (1)
      READ (15,3) LBLSTD
    3 FORMAT (1X,I5)
      EPOCTR = EPOCTR +1
      CALL RDTAPE (BLOCK1, &990)
      CALL RDTAPE (BLOCK2, &980)

      CALL RDTAPE (BLOCK3, &970)
      TEST = ODATA(1)/2048
      CAT = (ODATA(1) - TEST*2048)/64
      STAGE = (ODATA(1) - TEST*2048 - CAT*64)/8
      STGCTR = ODATA(1) - TEST*2048 - CAT*64 - STAGE*8
      IF(ODATA(1).EQ.LBLSTD) GOTO 4
      WRITE (15,5) ODATA(1), LBLSTD
    5 FORMAT (    ' IS THERE AN ERROR?  ODATA(1) ', I6, ' AND LBLSTD ',
     2  I6, ' DO NOT CORRESPOND.')
      CALL TYPE (0)
      WRITE (15,6) CAT, TEST, STAGE, STGCTR, EPOCTR
    6 FORMAT (' ODATA(1) SHOWS CAT', I3, ' TEST', I2, ' STAGE', I2,
     2  ' STGCTR', I2, ' FOR EPOCTR', I3, '.')
      CALL TYPE (0)
      GOTO 800
  807 CONTINUE
    4 IF (CAT.EQ.CATSTD.AND.IABS(TEST).EQ.IABS(TSTSTD)) GOTO 7
      IF (CAT.GT.0) GOTO 8
      IF (IABS(TEST).EQ.IABS(TSTSTD)) GOTO 9
      WRITE (15,10) ODATA(1), TEST, STAGE, STGCTR, EPOCTR
   10 FORMAT (    ' LABEL', I6, ' WITH TEST TAPE', I2, ' STAGE', I2,
     2  ' STGCTR', I2, ' AND EPOCTR', I4, ' READY.  ENTER TRCHNLS IN 4(1
     3X,I2) FORMAT.')
      CALL TYPE (1)
      READ (15,11) (TRCHNL (I), I=1,4)
   11 FORMAT (4(1X,I2))
      TSTSTD = TEST
      CATSTD = 0
      GOTO 12
    8 IF (CAT.EQ. CATSTD) GOTO 13
      WRITE (15,14) ODATA(1), CAT, STAGE, STGCTR, EPOCTR
   14 FORMAT (    ' LABEL', I6, ' WITH CAT', I3, ' STAGE', I2, ' STGCTR',
     2  I2, ' AND EPOCTR', I4, ' READY.  ENTER BRNLOCS IN 4(1X,I2) FORMA
     3T.')
      CALL TYPE (1)
      READ (15,11) (BRNLOC(I), I=1,4)
      CATSTD = CAT
      TSTSTD = 0
      GOTO 12
    9 WRITE (15,15)
   15 FORMAT (' ERROR. TEST NOT EQUAL TO TSTSTD AFTER INITIAL CHECK.')
      CALL TYPE (0)
      GOTO 800
   13 WRITE (15,16)
   16 FORMAT (' ERROR. CAT NOT EQUAL TO CATSTD AFTER INITIAL CHECK.')
      CALL TYPE (0)
      GOTO 800
    7 WRITE (15,17) ODATA(1), STAGE, STGCTR, EPOCTR
```

```
 17 FORMAT (' SAME CRNLOGS/TRCHANLS USED FOR LABEL', I6, ' STAGE', I2,
   2  ' STGCTR', I2, '.  EPOCTR IS', I4,'.')
      CALL TYPE (0)
 12 DO 80 I = 32766, 32769
      ODATA(I) = 0
 80 CONTINUE
803 DO 20 CHANL=1,4
      MAXAMP = 0
      DO 19 POINT=1, 8192
      IF (DARRAY(CHANL,POINT).GE.8191.OR.DARRAY(CHANL,POINT).LE.-8191)
     2  GOTO 21
      GOTO 19
 21   MAXAMP = MAXAMP + 1
      WRITE (6,22) MAXAMP, CHANL, POINT
 22   FORMAT (/, ' MAXAMP NUMBER', I3, ' AT', I2, ',', I4,'.')
 19   CONTINUE
      IF (MAXAMP.EQ.0) GOTO 20
      WRITE (6,81)
      WRITE (15,23) MAXAMP, CHANL
 23   FORMAT (1X, I3, ' POINTS AT MAXAMP IN CHANL', I2,'.')
      CALL TYPE (0)
 20   CONTINUE
      WRITE (15,899)
      CALL TYPE (1)
      READ (15,898) WHATDO
      GOTO (801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811
     2                                                        ), WHATDO
804   WRITE (15,24)
 24   FORMAT (    ' ENTER DESIRED FRQCOD IN 1X,I2 FORMAT.')
      CALL TYPE (1)
      READ (15,25) FRQCOD
 25   FORMAT (1X, I2)
      GOTO (26,27,28      ), FRQCOD
 26   INCRMT = 1
      SMPFRQ=512
      NODPTS = 8192
      GOTO 29
 27   INCRMT = 2
      SMPFRQ=256
      NODPTS = 4096
      GOTO 29
 28   INCRMT = 4
      SMPFRQ=128
      NODPTS=2048
 29   DATPTS=NODPTS
      PRAMTR(14) = SMPFRQ
      CALL TPRCMP (NODPTS, SMPFRQ, TPRCOS, TPRPTS)
      DO 30 CHANL=1,4

      SUM = 0.0
      SUMSQ = 0.0
      SUMCUB = 0.0
      SUM4TH = 0.0
      DO 31 POINT = 1, NODPTS, INCRMT
      DATA(POINT) = DARRAY(CHANL,POINT)/81.92
      SUM = SUM + DATA(POINT)
 31   CONTINUE
      MEAN = SUM/DATPTS
      DO 32 POINT = 1, NODPTS, INCRMT
      DATA(POINT) = DATA(POINT) - MEAN
      SUMSQ = SUMSQ + DATA(POINT)**2
      SUMCUB = SUMCUB + DATA(POINT)**3
      SUM4TH = SUM4TH + DATA(POINT)**4
 32   CONTINUE
      VAR = SUMSQ/DATPTS
      STDDEV = SQRT(VAR)
      SKEW = (SUMCUB/DATPTS        )/SQRT (VAR**3)
      KURT = ((SUM4TH/DATPTS       )/(VAR**2)) -3.0
      CALL TAPER (TPRCOS, DATA, NODPTS, TPRPTS)
 87   CALL PWRCMP (DATA, RAWSPC, NODPTS)
805   WRITE (15,33)
 33   FORMAT (' ENTER SMUCOD, LOGCOD, HANCOD, AND PLOTNO IN 3(1X, I1),
     2  1X,I3 FORMAT.')
      CALL TYPE (1)
      READ (15,34) SMUCOD, LOGCOD, HANCOD, PLOTNO
 34   FORMAT (3(1X,I1), 1X,I3)
      GOTO (35, 36, 37, 89, 39          ), SMUCOD
 39   GOTO (45, 47, 88,    44) , LOGCOD
 44   IF (CAT.GT.0) GOTO 48
 49   LABEL = FRQCOD*(2**29)+NOHANS*(2**27)+SMUCOD*(2**24)+LOGCOD*(2**21
     2  )+TEST*(2**15)+TRCHNL(CHANL)*64+STAGE*8+STGCTR
      LABEL = -LABEL
      WRITE (6,50) EPOCTR, TEST, TRCHNL(CHANL), STAGE, STGCTR, LABEL,
     2  SMUCOD, LOGCOD, NOHANS, SMPFRQ
 50   FORMAT (/, ' EPOCH ', I3, ' FROM ODATA TAPE.  THE FOLLOWING PLOT C
     2ONTAINS THE POWER SPECTRUM FOR TEST TAPE NO.', I2, ' TRCHNL', I2,
     3  ', STAGE', I2, ' NUMBER', /, I2, ',  PWRSPC LABEL IS', I12,
     4  ' WITH SMUCOD=', I2, ', LOGCOD=', I2, ', NOHANS=', I2, ' AND SMP
     5FRQ=', I4, '.')
 51   CALL OOPLOT (PRAMTR, PLOTNO, SMUSPC, NUMPLT, SMUCOD)
 82   WRITE (1) LABEL, (SMUSPC(I), I=1, PLOTNO), (PRAMTR(I), I=1,15)
      WRITE (6,53) NUMPLT, LABEL
 53   FORMAT ('1', ' PARAMETERS FOR PLOT NO. ', I4,' LABEL',I12,' WERE')
      DO 54 I = 1,15
      WRITE (6,55) I, PRAMTR(I)
 55   FORMAT (1X, I2, 3X, 1PE15.7)
 54   CONTINUE
```

```
      WRITE (6,81)
      WRITE (15,56) CHANL
   56 FORMAT (    ' CHANL', I2, ' HAS COMPLETED THIS RUN.')
      CALL TYPE (0)
      WRITE (15,899)
      CALL TYPE(1)
      READ (15,898) WHATDO
      GOTO (801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811
     2                                                        ), WHATDO
   30 CONTINUE
      WRITE (15,57) EPOCTR, ODATA(1)
   57 FORMAT (' PROCESSING FOR EPOCH NO.', I5, ' LABEL', I6,' COMPLETE')
      CALL TYPE (0)
      GOTO 801
   47 DO 43 I=1, PLOTNO
      SMUSPC(I) = ALOG10(SMUSPC(I))
   43 CONTINUE
      GOTO 44
   88 CALL DECIBL (SMUSPC, PLOTNO)
      GOTO 44
   45 DO 46 I=1, PLOTNO
      SMUSPC(I) = ALOG(SMUSPC(I))
   46 CONTINUE
      GOTO 44
   48 LABEL = FRQCOD*(2**29)+NOHANS*(2**27)+SMUCOD*(2**24)+LOGCOD*(2**21
     2  )+CAT*(2**15)+BRNLOC(CHANL)*64+STAGE*8+STGCTR
      WRITE (6,52) EPOCTR, CAT, BRNLOC(CHANL), STAGE, STGCTR, LABEL,
     2  SMUCOD, LOGCOD, NOHANS, SMPFRQ
   52 FORMAT (/, ' EPOCH ', I3, ' FROM ODATA TAPE.  THE FOLLOWING PLOT C
     2ONTAINS THE POWER SPECTRUM FOR CAT NO.', I3, ', BRNLOC', I3,', STA
     3GE', I2, ', NUMBER', /, I2, '.  PWRSPC LABEL IS', I12, ' WITH SMUC
     4OD=', I2, ', LOGCOD=', I2, ', NOHANS=', I2, ', AND SMPFRQ=',I4,'.'
     5  )
      GOTO 51
   99 WRITE (15,991)
  991 FORMAT (    ' NORMAL ENDFILE FOR DATA.')
      CALL TYPE (0)
      GOTO 800
  980 WRITE (15,981)
  981 FORMAT (    ' ERROR.  ENDFILE AT BLOCK2.')
      CALL TYPE (0)
      GOTO 800
  970 WRITE (15,971)
  971 FORMAT (    ' ERROR.  ENDFILE AT BLOCK3.')
      CALL TYPE (0)
      GOTO 800
  960 WRITE (6,961) (ODATA(I), I=1, 101)
  961 FORMAT (/, ' LABEL AND FIRST 100 POINTS OF ODATA ARE', I6, /,

     2  (20(1X,I5)))
      WRITE (6,962) (ODATA(I), I=10874, 10973)
  962 FORMAT (/, (20(1X,I5)))
      WRITE (6,962) (ODATA(I), I=21797, 21896)
      WRITE (6,962) (ODATA(I), I=32670, 32769)
      WRITE (6,81)
   81 FORMAT (/)
      GOTO 800
   35 CALL OVRLAP (FRQCOD, RAWSPC, SMUSPC)
      GOTO 38
   36 CALL TRIANG (FRQCOD, SMUSPC, RAWSPC)
      GOTO 38
   37 CALL RCTNGL (SMPFRQ, SMUSPC, RAWSPC)
      GOTO 38
   89 CALL DIRECT(RAWSPC, SMUSPC, PLOTNO)
   38 IF (HANCOD.EQ.1) GOTO 40
      NOHANS=0
      GOTO 39
   40 WRITE (6,81)
      WRITE (15,41)
   41 FORMAT (' ENTER NOHANS IN 1X,I2 FORMAT.')
      CALL TYPE (1)
      READ (15,25) NOHANS
      DO 42 I=1, NOHANS
      CALL HANN (PLOTNO, SMUSPC)
   42 CONTINUE
      GOTO 39
  800 WRITE (6,81)
      WRITE (15,895)
  895 FORMAT (' ENTER WHATDO IN 1X, I2 FORMAT.  NO PASS POSSIBLE.')
  899 FORMAT (' ENTER WHATDO IN 1X, I2 FORMAT.  ENTER 99 TO PASS.')
      CALL TYPE (1)
      READ (15,898) WHATDO
  898 FORMAT (1X,I2)
      GOTO (801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811
     2                                                        ), WHATDO
  808 WRITE (15,897)
  897 FORMAT (' ENTER CHANGE IN 1X, I2 FORMAT.')
      CALL TYPE (1)
      READ (15,25) CHANGE
      GOTO (850, 860, 870                    ), CHANGE
  850 WRITE (15,851)
  851 FORMAT (    ' ENTER CORRECT TEST AND TRCHNL(1-4) IN 1X, I2, 4(1X,I3
     2) FORMAT.')
      CALL TYPE (1)
      READ (15,852)  TEST, (TRCHNL(I) , I=1,4)
  852 FORMAT (1X, I2, 4(1X,I3))
      GOTO 800
```

```
      WRITE (15,861)
  861 FORMAT (' ENTER CAT AND BRNLOC(1-4) IN 1X, I2, 4(1X,I3) FORMAT.')
      CALL TYPE (1)
      READ (15,852) CAT, (BRNLOC(I), I=1,4)
      GOTO 800
  870 WRITE (15,871)
  871 FORMAT (' ENTER STAGE AND STGCTR IN 1X,I1,1X,I1 FORMAT.')
      CALL TYPE (1)
      READ (15,872) STAGE, STGCTR
  872 FORMAT (1X,I1, 1X, I1)
      GOTO 800
  806 WRITE (15,85)
   85 FORMAT (' TO SKIP EPOCHS, ENTER EPCSKP IN 1X, I2 FORMAT.')
      CALL TYPE (1)
      READ (15,25) EPOSKP
      DO 86 M=1, EPOSKP
      EPOCTR = EPOCTR +1
      CALL RDTAPE (BLOCK1, &990)
      CALL RDTAPE (BLOCK2, &980)
      CALL RDTAPE (BLOCK3, &970)
   86 CONTINUE
      GOTO 801
  809 WRITE (15,840)
  840 FORMAT (' ENTER TSTPRT IN 1X, I2 FORMAT.')
      CALL TYPE (1)
      READ (15,25) TSTPRT
  841 WRITE (6,845) NUMPLT
  845 FORMAT (/, ' TEST PRINT DATA FOR PLOT NO. ', I5,' FOLLOWS.')
      GOTO (842, 843, 844, 960          ), TSTPRT
  844 WRITE (6,846) (SMUSPC(I), I=1, PLOTNO)
  846 FORMAT (' COMPLETE SMUSPC ARRAY FOLLOWS', /, 10(1PE12.4))
      GOTO 800
  843 WRITE (6,847) (RAWSPC(I), I=1, 96)
  847 FORMAT (' SAMPLE FROM RAWSPC ARRAY FOLLOWS',/,    8(1PE12.4))
      WRITE  (6,846) (SMUSPC(I), I=1, PLOTNO)
      GOTO 800
  842 WRITE (6,848) (DATA(I), I=1,100)
  848 FORMAT (' SAMPLE DATA ARRAY FOLLOWS', /, 10(1PE12.4))
      WRITE (6,847) (RAWSPC (I), I=1,96)
      WRITE  (6,846) (SMUSPC(I), I=1, PLOTNO)
      GOTO 800
  810 WRITE (15,896)
  896 FORMAT (' ENTER BKSPAC IN 1X, I2 FORMAT.  TAPE IS BACKSPACED ONE
     2EPOCH FOR EACH COUNT.')
      CALL TYPE (1)
      READ (15,25) BKSPAC
      DO 894 I=1, BKSPAC
      DO 893 J=1, 3

      CALL FSTIO(ICHAN, CWADR1,ICHK)
  893 CONTINUE
      EPOCTR = EPOCTR - 1
  894 CONTINUE
      GOTO 800
  811 CALL GENRAT (DARRAY, NODPTS)
      PLTCOD=1
      GOTO 804
  812 ENDFILE 1
      REWIND 1
      STOP
      END


      SUBROUTINE DOPLOT (PRAMTR, PLOTNO, SMUSPC, NUMPLT, SMUCOD)
C     THIS SUBROUTINE COMPUTES MINPWR, MAXPWR, AND TOTPWR AS WELL AS PLOTTING
C     SUBROUTINE PLOT1.
      DIMENSION PRAMTR(15), SMUSPC(512), IDX(6), IDY(6), TITLE(8), X(512
     2  ), Y(512)
      REAL MINPWR, MAXPWR, MIN, MAX
      INTEGER TITLE, SHADE, PLOTNO, PLOTPT
      INTEGER*2 SMUCOD
      MAXPWR=0.0
      MINPWR=10.0**70
      TOTPWR=0.0
      DO 401 PLOTPT=1, PLOTNO
      MAXPWR = AMAX1(MAXPWR, SMUSPC(PLOTPT))
      MINPWR = AMIN1(MINPWR, SMUSPC(PLOTPT))
      TOTPWR = TOTPWR+SMUSPC(PLOTPT)
  401 CONTINUE
      PRAMTR(11) = MINPWR
      PRAMTR(12) = MAXPWR
      PRAMTR(13) = TOTPWR
      PRAMTR(15) = PLOTNO
      NUMPLT = NUMPLT + 1
      MIN = MINPWR - .001*MINPWR
      MAX = MAXPWR + .001*MAXPWR
      IF (SMUCOD.EQ.4) GOTO 403
      XINCR=0.5
  402 SHADE=1
      DATA IDX/'FREQ', 'UENC', 'Y IN', '    ', ' HZ ', '    '/
      DATA IDY/'POWE', 'R IN', ' VOL', 'TS S', 'QUAR', 'ED  '/
      DATA TITLE/'POWE', 'R SP', 'ECTR', 'UM  ', '    ', '    ',',    ',
     2  '    '/
      CALL PLOT1 (0.0, XINCR, SMUSPC, PLOTNO, MIN   , MAX   , SHADE,
     2  TITLE, 1.0, IDY, IDX)
      RETURN
  403 XINCR=0.0625
      GOTO 402
      END
```

```
      SUBROUTINE DECIBL (SMUSPC, PLOTNO)
      DIMENSION SMUSPC(512)
      INTEGER PLOTNO
      REAL MAXPWR
      MAXPWR=0.0
      DO 101 I=1, PLOTNO
      MAXPWR = AMAX1(MAXPWR, SMUSPC(I))
  101 CONTINUE
      DO 102 I=1, PLOTNO
      SMUSPC(I) = ALOG10 (SMUSPC(I)/MAXPWR)
  102 CONTINUE
      RETURN
      END

      SUBROUTINE TRIANG (FRQCOD, SMUSPC, RAWSPC)
C     THIS SUBROUTINE SMOOTHS RAWSPC BY A TRIANGULAR WINDOW OF SPAN SUFFICIENT
C     TO COVER JUST THE POINTS TO BE REDUCED TO ONE SMUSPC POINT.
      DIMENSION SMUSPC (512), RAWSPC(4096)
      INTEGER*2 PWRPTS, FRQCOD
      DO 110 I=1, 512
      SMUSPC(I) = 0.0
  110 CONTINUE
      GOTO(101, 102, 103), FRQCOD
      WRITE (15,104)
  104 FORMAT (' ERROR. FRQCOD INVALID FOR TRIANG SUBROUTINE.')
      CALL TYPE (0)
  105 RETURN
  101 PWRPTS = 4096
  109 INCRMT = 8
      K=1
      SMUSPC(1) = (4*RAWSPC(1) + 3*RAWSPC(2) + 2*RAWSPC(3) + RAWSPC(4))
     2  /10
      DO 106 N=5, PWRPTS, INCRMT
      K=K+1
      SMUSPC(K) = (RAWSPC (N)+2*RAWSPC(N+1)+3*RAWSPC(N+2)+4*RAWSPC(N+3)+
     2  4*RAWSPC(N+4)+3*RAWSPC(N+5)+2*RAWSPC(N+6)+RAWSPC(N+7))/20.
  106 CONTINUE
      GOTO 105
  102 PWRPTS=2048
      GOTO 109
  103 PWRPTS=1024
      GOTO 109
      END

      SUBROUTINE TAPER (TPRCOS, DATA, NODPTS, TPRPTS)
      DIMENSION DATA(1), TPRCOS(1)
      INTEGER*2          TPRPTS, HI
      HI=NODPTS+1
      DO 307 LO=1, TPRPTS
      HI=HI-1
      DATA(LO) = DATA(LO)*TPRCOS(LO)
      DATA(HI) = DATA(HI)*TPRCOS(LO)
  307 CONTINUE
  305 RETURN
      END

      SUBROUTINE PWRCMP (DATA, RAWSPC, NODPTS)
      DIMENSION DATA(8192), RAWSPC(4096), IWK(13)
      INTEGER   NODPTS, IWK
      COMPLEX GAMN
      CALL FFTR (DATA, GAMN, NODPTS, IWK)
      J=0
      DO 201 I=1, NODPTS,2
      J=J+1
      RAWSPC(J) = DATA(I)**2+DATA(I+1)**2
  201 CONTINUE
      RETURN
      END

      SUBROUTINE TPRCMP (NODPTS, SMPFRQ, TPRCOS, TPRPTS)
      DIMENSION TPRCOS(820)
      INTEGER TPRPTS, SMPFRQ
      PI=3.141593
      TPRPTS=NODPTS/10
      J=-1
      DO 308 I=1, TPRPTS
      J=J+1
      ARGUMT = (PI*J)/TPRPTS
      TPRCOS(I) = 0.5*(1.0-COS(ARGUMT))
  308 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE OVRLAP (FRQCOD, RAWSPC, SMUSPC)
      DIMENSION RAWSPC(4096), SMUSPC(512)
      INTEGER*2 PWRPTS, FRQCOD
      DO 71 I=1, 512
      SMUSPC(I) = 0.0
   71 CONTINUE
      GOTO (125, 150, 175), FRQCOD
      WRITE (15,101)
  101 FORMAT (' ERROR. FRQCOD INVALID FOR OVRLAP.')
      CALL TYPE(0)
  102 RETURN
  125 PWRPTS=4096
  128 INCRMT=8
      SMUSPC(1)=8*(RAWSPC(1)+7*RAWSPC(2)+6*RAWSPC(3)+5*RAWSPC(4)+4*
     2  RAWSPC(5)+3*RAWSPC(6)+2*RAWSPC(7)+RAWSPC(8))/36.0
      NMINUS=PWRPTS-16
      K=1
      DO 126 I=1, NMINUS, INCRMT
      K=K+1
      SMUSPC(K) = (RAWSPC(I)+2*RAWSPC(I+1)+3*RAWSPC(I+2)+4*RAWSPC(I+3)+
     2  5*RAWSPC(I+4)+6*RAWSPC(I+5)+7*RAWSPC(I+6)+8*RAWSPC(I+7)+9*RAWSPC
     3  (I+8)+7*RAWSPC(I+9)+6*RAWSPC(I+10)+5*RAWSPC(I+11)+4*RAWSPC(I+12)
     4  +3*RAWSPC(I+13)+2*RAWSPC(I+14)+RAWSPC(I+15))/72.0
  126 CONTINUE
      K=K+1
      SMUSPC(K) = (RAWSPC(PWRPTS-7)+2*RAWSPC(PWRPTS-6)+3*RAWSPC(PWRPTS-5
     2  )+4*RAWSPC(PWRPTS-4)+5*RAWSPC (PWRPTS-3) +6*RAWSPC(PWRPTS-2)+7*
     3  RAWSPC(PWRPTS-1)+8*RAWSPC(PWRPTS))/36.0
      GOTO 102
  150 PWRPTS=2048
      GOTO 128
  175 PWRPTS=1024
      GOTO 128
      END

      SUBROUTINE RCTNGL (SMPFRQ, SMUSPC, RAWSPC)
C     THIS SUBROUTINE SMOOTHS WITH EQUAL WEIGHTING ALL POINTS WITHIN THE
C     SPAN TO BE SMOOTHED.
      DIMENSION SMUSPC(512), RAWSPC(4096)
      INTEGER SMPFRQ
  106 NUMBER=4
      SMUSPC(1) = 0.0
      DO 111 I=1, 4
      SMUSPC(1) = RAWSPC(I) + SMUSPC(1)
  111 CONTINUE
      SMUSPC(1) = SMUSPC(1)/4
      INCRMT=8
      DO 107 I=2, SMPFRQ
      SMUSPC(I) = 0.0
      NUMBER = NUMBER + INCRMT
      K = NUMBER - INCRMT + 1
      DO 108 J=K, NUMBER
      SMUSPC(I) = SMUSPC(I)+RAWSPC(J)
  108 CONTINUE
      SMUSPC(I) = SMUSPC(I)/INCRMT
  107 CONTINUE
  105 RETURN
      END

      SUBROUTINE HANN (PLOTNO, SMUSPC)
C     THIS SUBROUTINE IS INTENDED TO HANN AN ALREADY SMOOTHED SPECTRUM.
      DIMENSION HANSPC(512), SMUSPC(512)
      INTEGER PLOTNO
      NOMIN1=PLOTNO-1
      HANSPC(1) = 0.5*(SMUSPC(1)+SMUSPC(2))
      DO 101 I=2, NOMIN1
      HANSPC(I) = 0.25*(SMUSPC(I-1)+SMUSPC(I+1)) + 0.5*SMUSPC(I)
  101 CONTINUE
      HANSPC(PLOTNO) = 0.5*(SMUSPC(NOMIN1)+SMUSPC(PLOTNO))
      DO 102 I=1, PLOTNO
      SMUSPC(I) = HANSPC(I)
  102 CONTINUE
      RETURN
      END

      SUBROUTINE DIRECT (RAWSPC, SMUSPC, PLOTNO)
      DIMENSION SMUSPC (512), RAWSPC(4096)
      INTEGER PLOTNO
      DO 101 I=1, PLOTNO
      SMUSPC(I) = RAWSPC(I)
  101 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GENRAT (DARRAY, NCOPTS)
      INTEGER NODPTS
      INTEGER*2 DARRAY(4,8192)
      CALL SINE10 (NCOPTS, DARRAY)
      CALL SINE40 (NODPTS, DARRAY)
      CALL SQWAV4 (DARRAY, NODPTS)
      CALL SQWAV2 (DARRAY, NODPTS)
      RETURN
      END


      SUBROUTINE SINE40 (NODPTS, DARRAY)
      INTEGER X
      INTEGER*2 DARRAY(4,8192)
      PI = 3.141593
      X=20.0
      DO 701 I=1, NCOPTS
      ARGUMT = (2*PI*40*I)/512
      DARRAY(4,I) = X*SIN(ARGUMT)
701   CONTINUE
      RETURN
      END

      SUBROUTINE SINE10 (NODPTS, DARRAY)
      INTEGER X
      INTEGER*2 DARRAY(4,8192)
      PI = 3.141593
      X=20.0
      DO 701 I=1, NCOPTS
      ARGUMT = (2*PI*10*I)/512
      DARRAY(3,I) = X*SIN(ARGUMT)
701   CONTINUE
      RETURN
      END


      SUBROUTINE SQWAV4 (DARRAY, NCOPTS)
      INTEGER*2 DARRAY(4,8192)
      INCRMT=64
      NUMBER=1
700   MAX = NUMBER+INCRMT-1
      DO 701 I= NUMBER, MAX
      DARRAY(2,I) = 2000
701   CONTINUE
      NUMBER = NUMBER + INCRMT
      MAX = NUMBER + INCRMT -1
      DO 702 J = NUMBER, MAX
      DARRAY(2,J)=-2000
702   CONTINUE
      NUMBER = NUMBER + INCRMT
      IF (NUMBER.GE.NODPTS) GOTO 703
      GOTO 700
703   RETURN
      END

      SUBROUTINE SQWAV2 (DARRAY, NCOPTS)
      INTEGER*2 DARRAY(4,8192)
      INCRMT=128
      NUMBER=1
700   MAX = NUMBER+INCRMT-1
      DO 701 I= NUMBER, MAX
      DARRAY(1,I) = 2000
701   CONTINUE
      NUMBER = NUMBER + INCRMT
      MAX = NUMBER + INCRMT -1
      DO 702 J = NUMBER, MAX
      DARRAY(1,J)=-2000
702   CONTINUE
      NUMBER = NUMBER + INCRMT
      IF (NUMBER.GE.NODPTS) GOTO 703
      GOTO 700
703   RETURN
      END
```

PWRSPC begins by initializing certain labels and making necessary declarations. The EQUIVALENCE statements are used extensively to save storage space and permit simpler WRITE statements. Due to the amount of data, each epoch must be read in the three blocks since it was so written in ATOD. These three blocks are equivalenced to an ODATA array, and to DARRAY which breaks the data up into individual channels.

The first execution step prints a statement instructing the operator to enter LBLSTD. LBLSTD is the label which is expected to occur with the next epoch, and can be read from the printout of ATOD. The label entered is compared with that of the epoch read from digital tape; if it doesn't correspond, an error message is written on the console.

After incrementing an epoch counter, the program calls the ESSL library subroutine RDTAPE which reads the epoch into memory. RDTAPE must be used at this point instead of a FORTRAN read statement because the hybrid system writes to tape with assembler language, and an assembler routine must also be used to read the tape. A channel command word could be used for this, but RDTAPE has steps to compensate for tape reading errors. Another ESSL library subroutine, TOPEN, must be called once at the beginning of the program to prepare for RDTAPE. Printouts of these subroutines were not made here, but they may be found in Osorio (1976) and the ESSL library.

The epoch label, which is located in the first position of the epoch, is decomposed to provide PWRSPC with the cat or test tape number, stage, and counter. Certain record-

keeping operations are then performed and the console prints a message instructing the operator to enter either the BRNLOC or TRCHNL array codes. Each of these consists of a single array of four positions. BRNLOC values are a code indicating which area of the brain is represented and in what derivation. These are illustrated in Table 2. One BRNLOC corresponds to the data from one channel of EEG. TRCHNL is analogous, but is used for the test tapes to represent the channel of the tape recorder from which the signal was taken.

Since the maximum integer value which can be represented from the 1044 hybrid linkage unit is 8191, a DO loop scans each channel array for values at this level. Should any points be found, it probably indicates clipping of a wave during A-D conversion. The suspect points are printed for later evaluation.

I have made extensive use of computed GOTO statements (see e.g., Cress, Dirksen, Graham, 1970) in this program. These statements are used for determining paths to be taken at option points in the program. One of these option points is the selection of the effective sampling frequency. Since the data were sampled and converted at 512 per second, this is obviously the highest rate possible. But by selecting every second or fourth point, rates of 256/second or 128/second, respectively, can be used to reduce the number of computations required. Because of the approximate 240 Hz noise component discussed previously, however, only the 512/second rate was used in final processing.

TABLE 2

LISTING OF BRNLOC CODES*

1-l.	hippocampus (bipolar)
2-r.	hippocampus (bipolar)
3-l.	hippocampus (monopolar)
4-r.	hippocampus (monopolar)
5-l.	basal forebrain (bipolar)
6-r.	basal forebrain (bipolar)
7-l.	basal forebrain (monopolar)
8-r.	basal forebrain (monopolar)
9-l.	mesencephalic reticular (bipolar)
10-r.	mesencephalic reticular (bipolar)
11-l.	mesencephalic reticular (monopolar)
12-r.	mesencephalic reticular (monopolar)
13-l.	center median (bipolar)
14-r.	center median (bipolar)
15-l.	center median (monopolar)
16-r.	center median (monopolar)
17-l.	caudate (bipolar)
18-r.	caudate (bipolar)
19-l.	caudate (monopolar)
20-r.	caudate (monopolar)
21-l.	visual (monopolar)
22-r.	visual (monopolar)
23-X.	visual
24-l.	visual - r. frontal
25-l.	visual - l. frontal
26-l.	frontal (monopolar)
27-r.	frontal (monopolar)
28-X.	frontal
29-l.	frontal - r. visual
30-r.	frontal - r. visual

*all monopolar leads referenced against midline sinus.

Once the sampling frequency (SMPFRQ) has been selected, subroutine TPRCMP is called to compute the array TPRCOS to be used in tapering both ends of the time series data. This array is computed similarly to what Bingham (1967) recommends, except that the formula

$$\tfrac{1}{2}\left(1-\cos \pi \frac{t}{0.1T}\right)$$

is used for both ends, for t from 0 to TPRPTS, which is a variable depending on the sampling frequency. Bingham's formulae are not quite symmetrical, and using exact symmetry for both ends of the data saves approximately half the computer time when the taper is applied.

Since the results of TPRCMP depends only on SMPFRQ, the array can be computed only once for all four channels. Tapering is used in this program, as opposed to Hanning, because it is simpler and consumes less time.

The program then enters a DO loop which sequentially computes power for each of the four channels. The first steps in this loop initialize certain statistical variables and transfer the data to an array which is full word length where floating point operations can be performed. Dividing by 81.92 converts the data from quantizing level representation to voltage. Statistical formulas were taken from McNemar (1969).

The SUM is calculated and then the mean determined. As Bendat and Piersol (1971) recommend, the mean is then subtracted from each value to give the data a zero mean. This

greatly simplifies later calculations of the variance, standard deviation, shewness, and kurtosis. These statistics are computed here for informational purposes only. Subroutine TAPER is then called to apply the cosine taper array computed in TPRCMP to both ends of the data.

My subroutine PWRCMP then calls an ESSL library subroutine FFTR (IMSL, 1975) to compute the Fourier transform of the data. The data are returned in complex notation and both coefficients are squared and summed to compute power. These power values are then stored in the RAWSPC array.

The console then prints instructions for the operator to enter the values for two computed GOTO statements (SMUCOD, LOGCOD), a code for Hanning (HANCOD), and the number of power spectrum points to be plotted (PLOTNO).

At RAWSPC, the power spectrum has 1/16 Hz resolution. Since this is greater than necessary for general spectral work, I decided to reduce this resolution to 0.5 Hz. This reduction is performed by one of three averaging subroutines, according to the number entered for SMUCOD. The value 1 directs the program to OVRLAP, 2 to TRIANG, and 3 to RCTNGL. RCTNGL is the smoothing procedure recommended by Bendat and Piersol (1971), and it simply provides an average of 8 points at 1/16 Hz resolution to be represented by one point at 0.5 Hz resolution. This is the smoothing procedure commonly used in this program.

TRIANG and OVRLAP were written for different emphases. TRIANG gives a triangular weighting of the eight points such

that the frequencies nearer the final resolution point have greater weighting than those more remote. OVRLAP is also a triangular weighting, but averages across points from adjacent final resolution frequencies. In doing this, it performs something on the order of averaging and Hanning simultaneously. All smoothing subroutines output the data in the SMUSPC array.

A fourth subroutine (DIRECT) performs a direct transfer from RAWSPC to SMUSPC, thus retaining the 1/16 Hz resolution. This could be useful if working with very low frequencies of data such as tremor. To utilize this, a SMUCOD value of 4 must be entered.

If a 1 had been entered for HANCOD, the program prints an instruction to enter the number of times Hanning is desired (NOHANS). One or two is commonly used. A DO loop governs the number of Hanns performed. Hanning is performed in this program with natural numbers in all cases, though the final spectrum may be plotted as a logarithm, as explained later.

I feel I must emphasize that Hanning is only used here for purposes of emphasizing power peaks which are not so readily visible in an unhanned spectrum.

The number entered for LOGCOD determines the ordinate of the spectral plot. If a number 4 or greater is entered, the plot is made with natural numbers. If 3, the plot is made in db . If 2, ordinate values are plotted as a function of $\log_{10}$. And if 1 is entered, the plot is a

function of $\log_e$. A comparison of these different plots is made in Figure 6.

LABEL is then coded according to cat number, stage, etc. as in ATOD. If the data is from a test tape, LABEL is converted to its negative for ready distinction between cat and test labels.

A paper plot of the spectrum is then printed on the high-speed printer. The maximum frequency represented in the plot is determined by the value entered for PLOTNO according to the formula:

$$\frac{\text{PLOTNO}-1}{2}$$

My subroutine DOPLOT computes the maximum power (MAXPWR) and minimum power (MINPWR) of the spectrum and uses these values to scale the printed spectrum. This subroutine also calls ESSL library plot PLOT 1 to do the actual plotting. Normally, frequencies above 58 Hz would not be considered because of the 60 Hz line artifact and higher frequencies have little relative power.

A list of parameters pertaining to the plot follows each plot. This array includes the results of statistical computations equivalenced at the beginning of the program, PLOTNO, SMPFRQ, etc.

The label, power spectrum, and parameter array are all written to a second digital tape for storage until later processing may be carried out. No further analysis is performed with this data in this program.

Finally, the console prints a statement notifying the operator that the channel has completed its processing and is ready to cycle back in the DO loop to begin with the next channel.

Now that the majority of the program has been explained, I feel I am ready to detail what I consider an important feature of this program. I mentioned earlier that I had used computed GOTO statements to direct the program into different subroutines, etc., according to the codes entered. There is one other computed GOTO used, and I have named this WHATDO. WHATDO can serve as a diagnostic tool, as a means of obtaining further information, or as a re-processing tool. The WHATDO statement appears in several places as the program proceeds, and these points are identified by an 800 label. For example, 801 indicates the step at which LBLSTD is entered, 802 the REWIND and END steps, etc. These points and their functions are listed in Table 3.

When a WHATDO statement is encountered in the program, the console prints a message instructing the operator to enter a code. If the code number is larger than the number of options possible, the computer bypasses this step and continues with the next step of the program. I commonly use 99 as this value for consistency. If a WHATDO option not in the main program sequence is selected, i.e. after exercising one WHATDO option, entering a code value too large for the possible options should not be done. This could lead to disastrous results. On these options a "NO

TABLE 3

LISTING OF WHATDO OPTIONS AND THEIR FUNCTIONS

Code	Label	Function and Brief Description
1	801	Enter LBLSTD; used to begin a new epoch
2	802	Write ENDFILE; used to conclude program
3	803	Enter at test for maximum amplitude
4	804	Enter at select frequency code
5	805	Enter at smoothing point; can only get to from inside loop; used to provide comparison of different smoothing techniques without re-transforming
6	806	Skip epochs; used to bypass epochs on digital tape without having to process
7	807	Enter after label comparison; used to re-enter program in event of a mistake entering LBLSTD or other
8	808	Provide a means of changing data, as specified by code: 1 = TEST, TRCHNL; 2 = CAT, BRNLOC; 3 = STAGE, COUNTER
9	809	Provide a printout of certain data arrays as specified by code: 1 = RAWSPC, SMUSPC, DATA; 2 = RAWSPC, SUMSPC; 3 = SMUSPC
10	810	Backspace; used to re-read any epoch previously passed.
11	811	Call GENRAT; used to synthesize waveform mathematically for program processing

PASS POSSIBLE" statement follows the instruction, and a code to re-enter the main program sequence must be used. The one 800 label inside the DO loop for computing power is the only such label which cannot be reached from any point in the program; this label can only be reached from inside the loop.

Should one decide in advance that all spectra are to be computed in the same manner, it is possible to save a great deal of time by replacing certain console TYPE(1) statements sequences by statements assigning particular values to labels such as FRQCOD, SMUCOD, PLOTNO, etc. Executing the WRITE and READ statements on the console requires as much time as computing the power spectrum itself.

CHAPTER IV

RESULTS

Many WRITE statements were used during the debugging and testing processes with these programs to check results of calculations. Results in these tests showed the program computations were correct when compared with sample results performed on a calculator.

Three methods were used to validate the accuracy of the FFT. Initially, sine waves of different frequencies were A-D converted from analog tape and the power spectra computed. Examples of these for frequencies of 5 Hz and 8 Hz are illustrated in Figures 5 A, B. Sine waves were also input for frequencies of 10 Hz, 20 Hz, 30 Hz, 40 Hz, and 50 Hz; all of these were correctly represented in their respective power spectra.

Sine waves of 10 Hz and 40 Hz were generated (see appropriate subroutines in Figure 4) in the computer and power spectra computed for these also. The spectrum for the 10 Hz wave is presented in Figure 5 C.

Square waves of 2 Hz and 4 Hz were also generated for testing with their characteristic power spectra. These results are illustrated in Figures 5 D, E. The square wave was considered a good test for the FFT and the program because of its characteristic power spectrum (e.g., Gilbert, 1973).

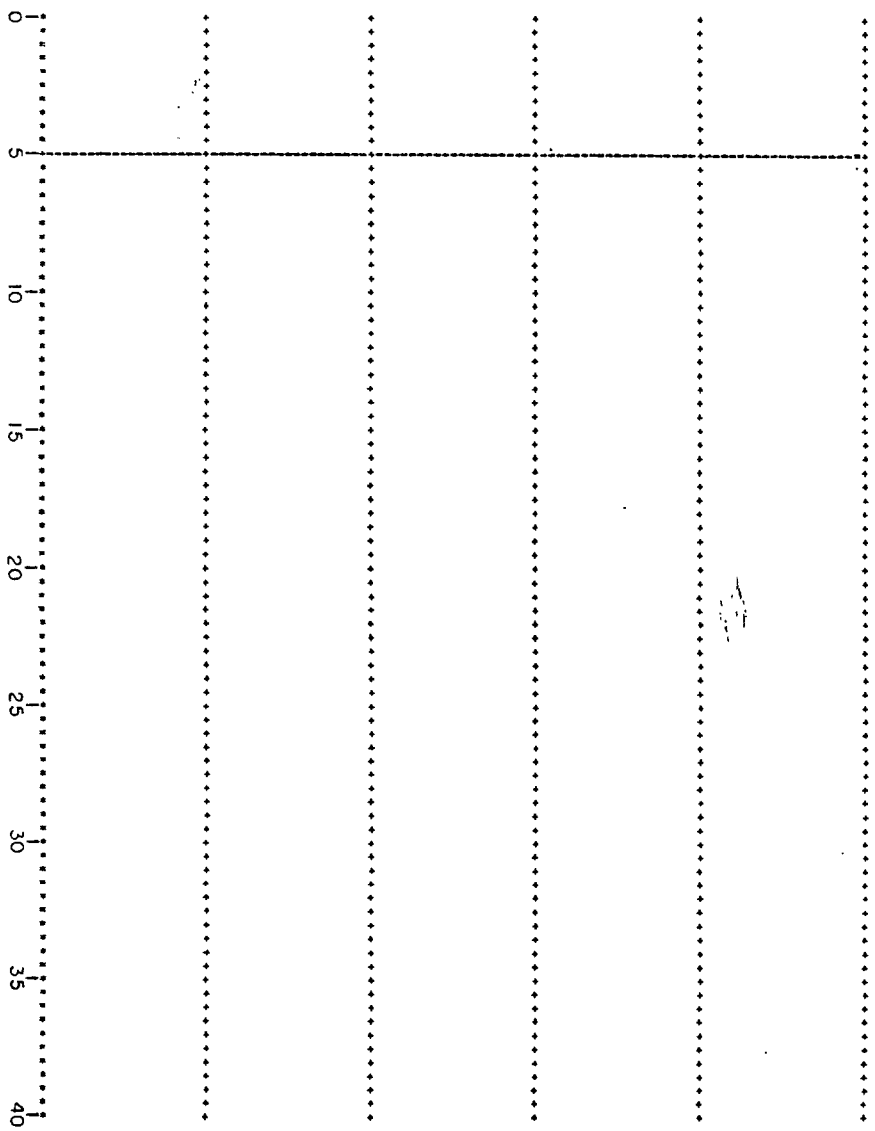

Figure 5A. 5 Hz A-D Sine Wave Spectrum

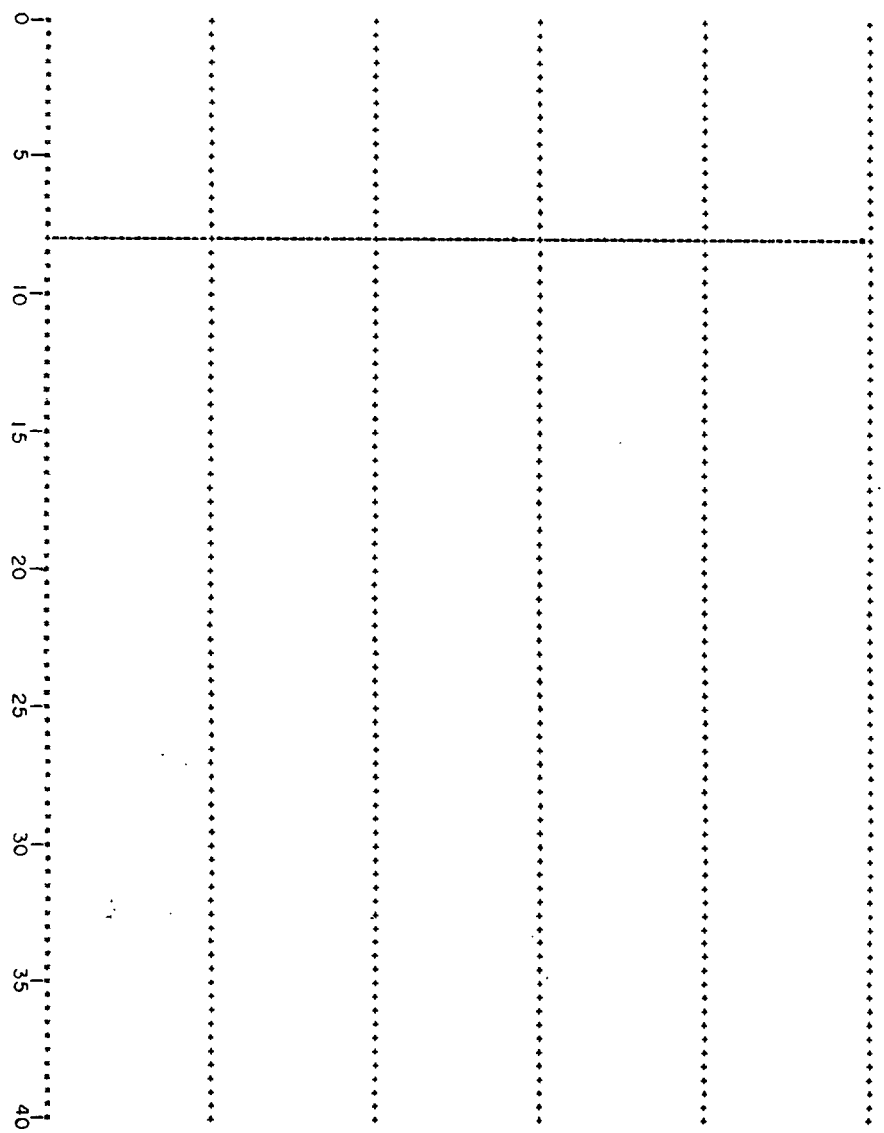

Figure 5B. 8 Hz A-D Sine Wave Spectrum

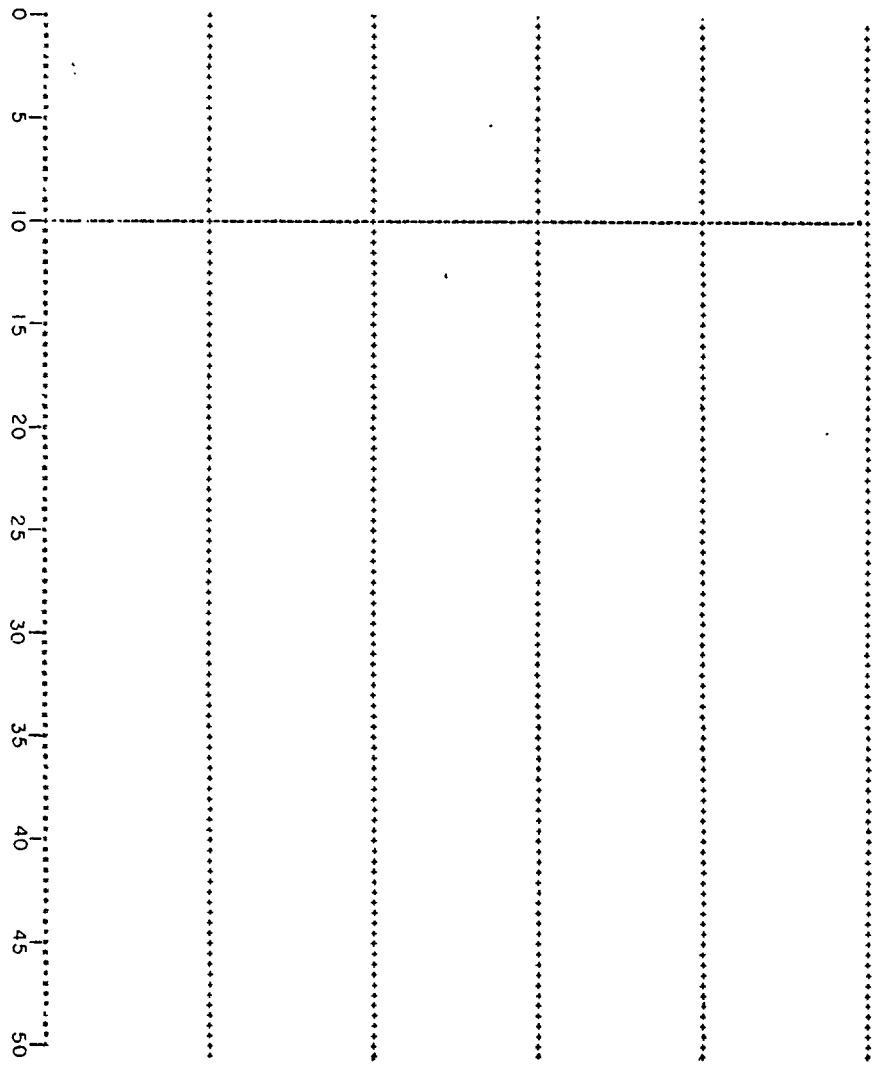

Figure 5C. 10 Hz Generated Sine Wave Spectrum

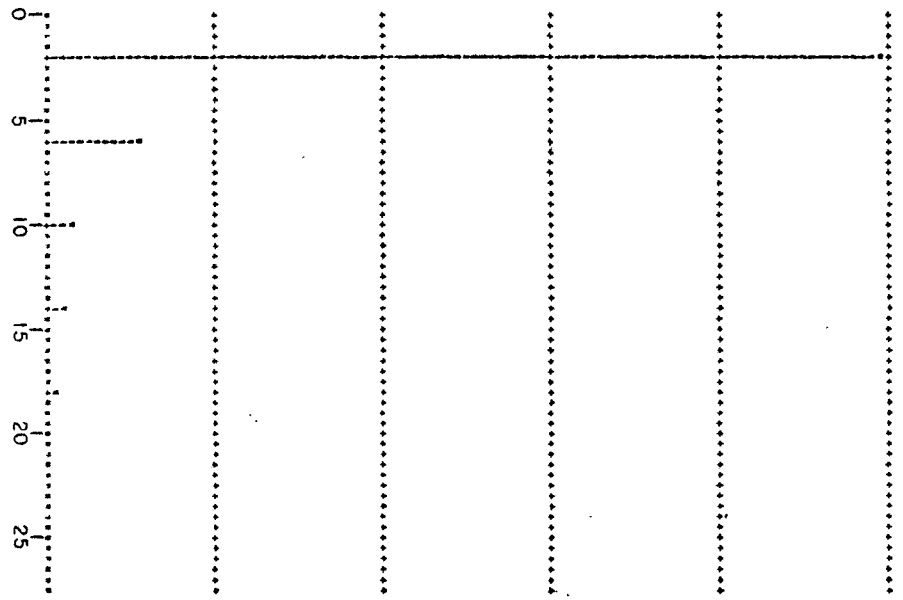

Figure 5D. 2 Hz Generated Square Wave Spectrum

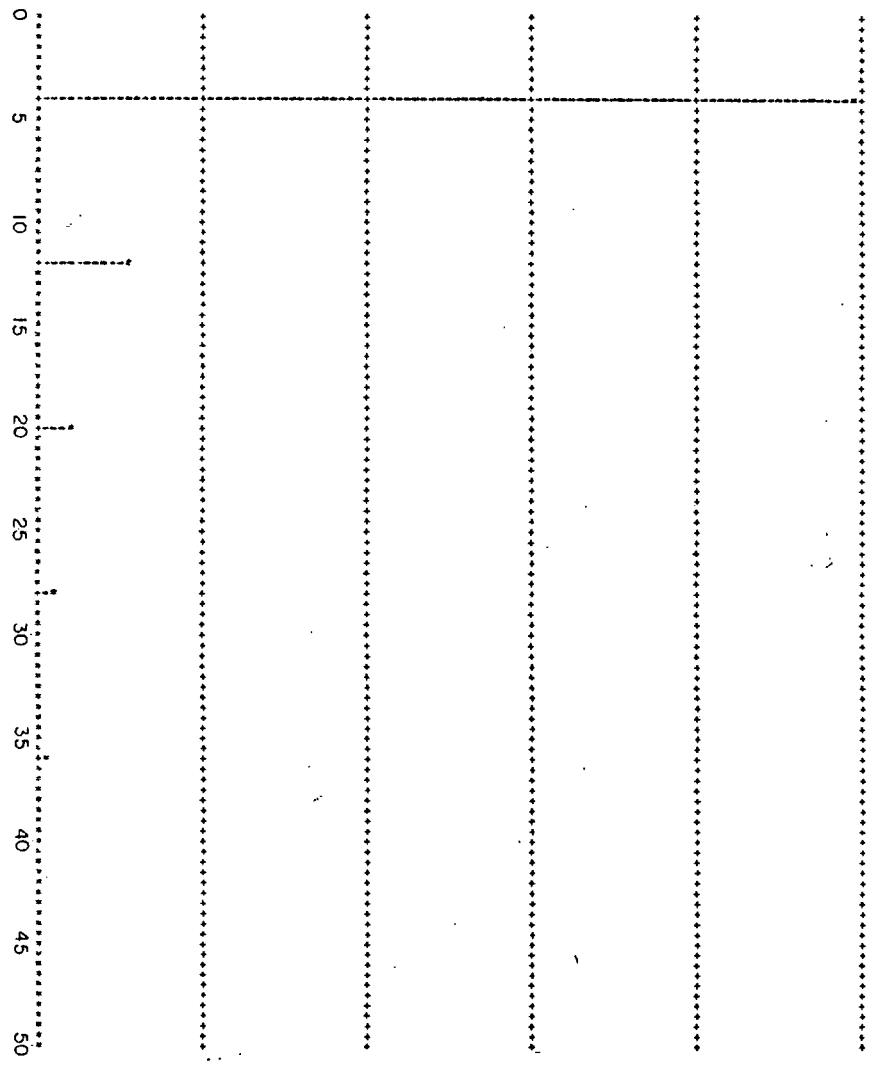

Figure 5E. 4 Hz Generated Square Wave Spectrum

Illustrations of logarithmic plots vs. natural number plots are given in Figures 6 A, B. Since the power decreases so rapidly with increasing frequency, a logarithmic plot is obviously preferred for the large frequency range used here. Figure 6 C shows the effect of Hanning after power computation. This serves to emphasize peaks for visual observation, but serves no other purpose when used under these conditions.

Figures 7 A-C illustrate the purpose for which these programs were written. They show power spectra of the mesencephalic reticular formation (mrf) while the cat is awake (WAK), in slow wave sleep (SWS), and in REM sleep, respectively. All spectra are Hanned twice to provide better visualization of the frequency peaks.

The recordings for these spectra were taken monopolarly, with the reference electrode in the bone over the frontal sinus. Exact coordinates will be described at a later date when the study is completed (Stramler, in progress).

Since these spectral plots are scaled individually on a relative power basis, it is difficult to see certain things readily. For instance, the peak at 1.50 Hz during SWS is two points larger than the maximum peak during WAK or REM on this $\log_e$ scale, demonstrating the greater power at low frequencies during SWS. The total power in these spectra is also greatest during SWS, and least during REM.

Some features are obvious, however. For example, during WAK there are large relative peaks in the theta and 40 Hz regions; these peaks are not present during SWS, and

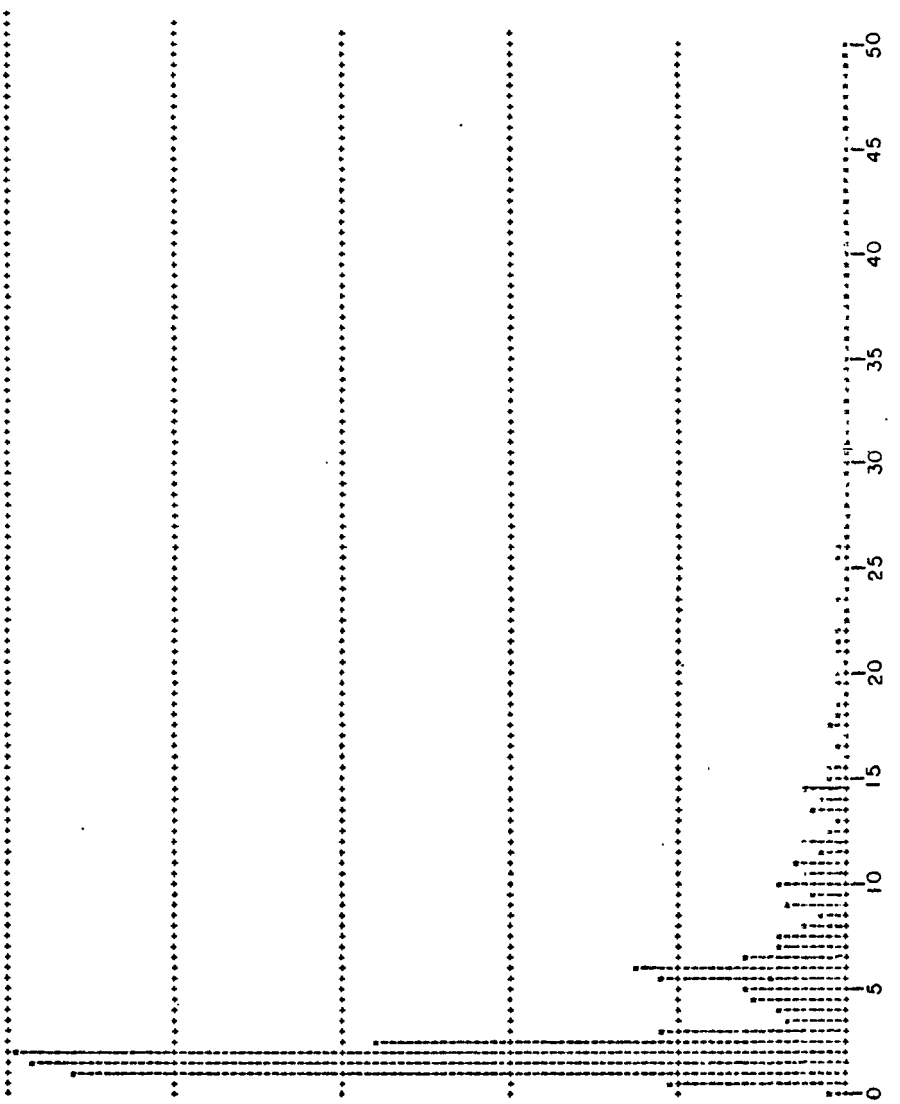

Figure 6A. Natural Number Plot of EEG Data

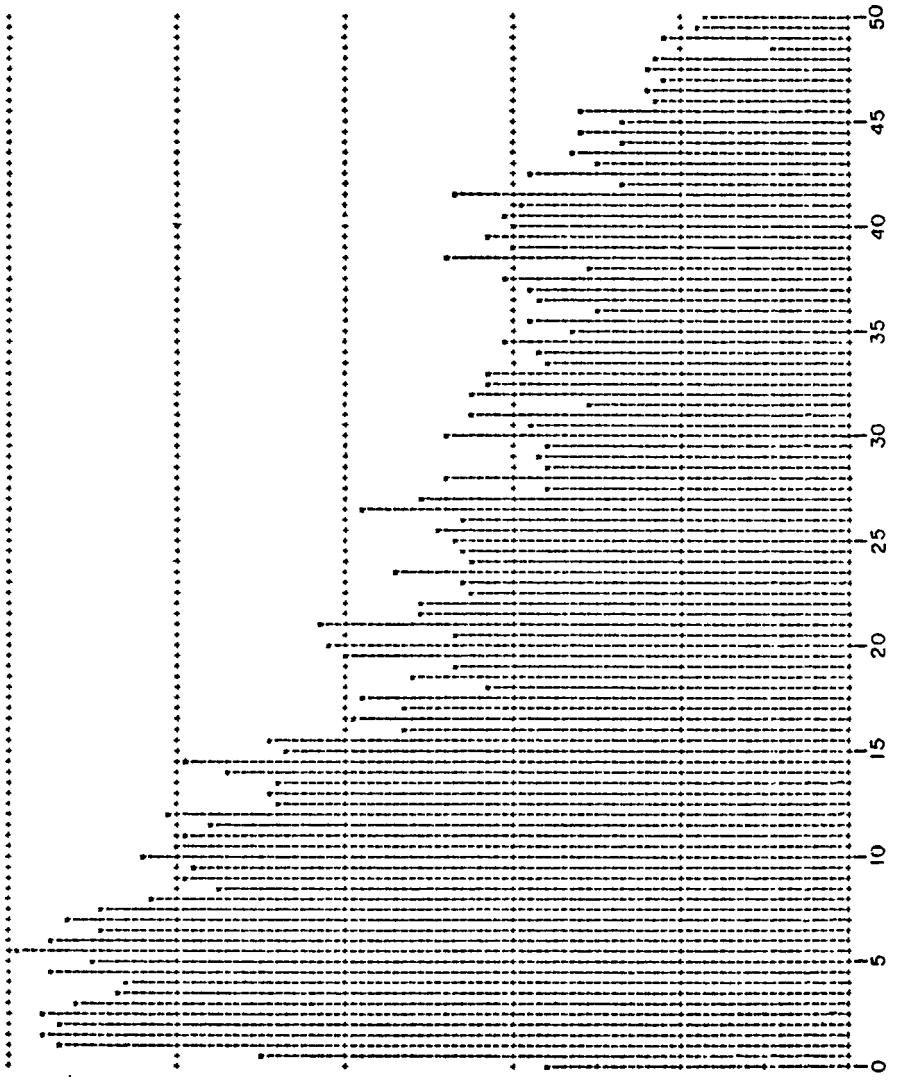

Figure 6B. Log_e Plot of Same EEG Data as Figure 6A

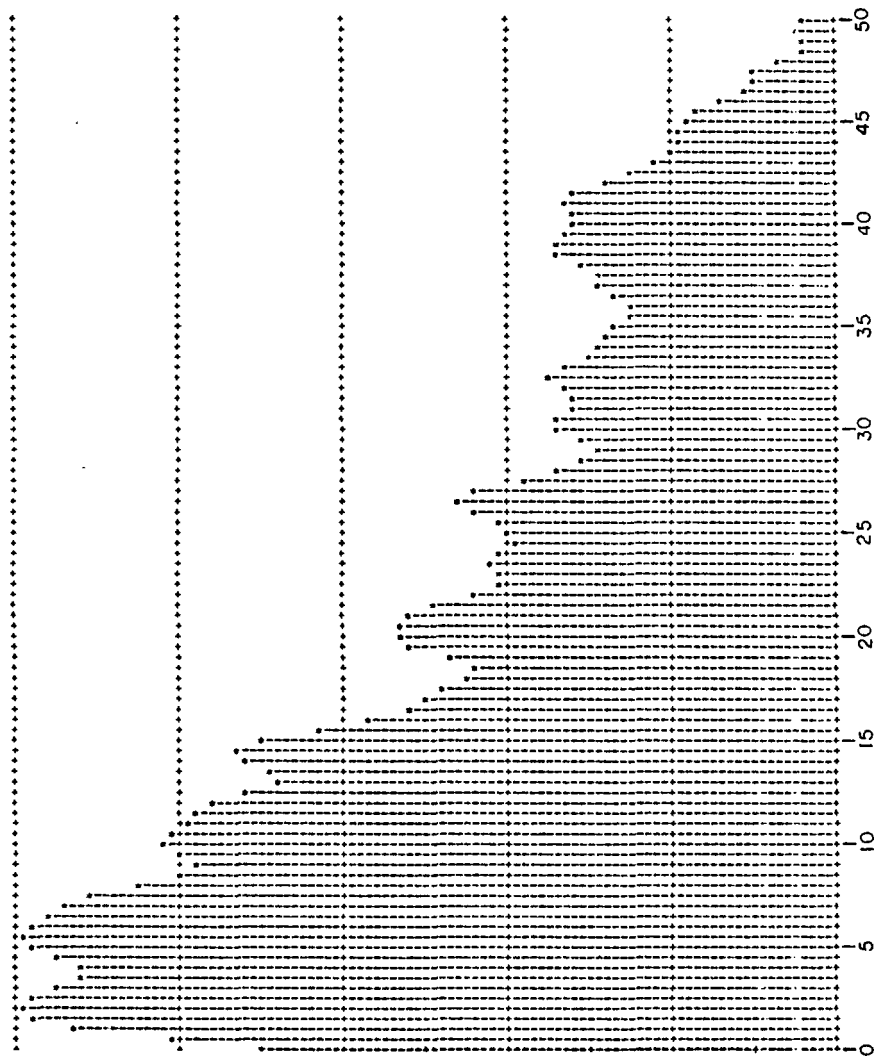

Figure 6C. Log_e Hanned Plot of Same EEG Data as Figure 6B

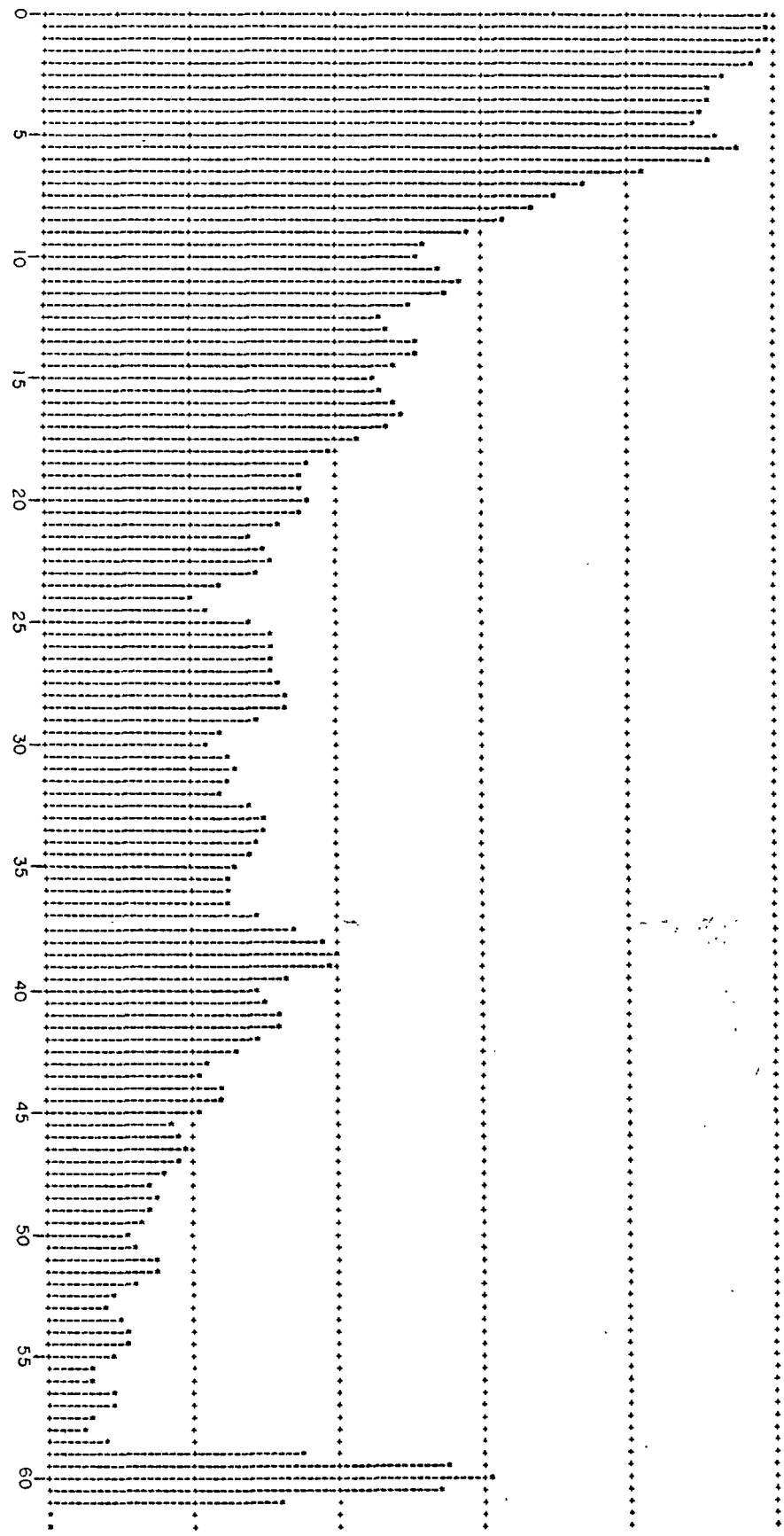

Figure 7A. Reticular Formation EEG When Awake

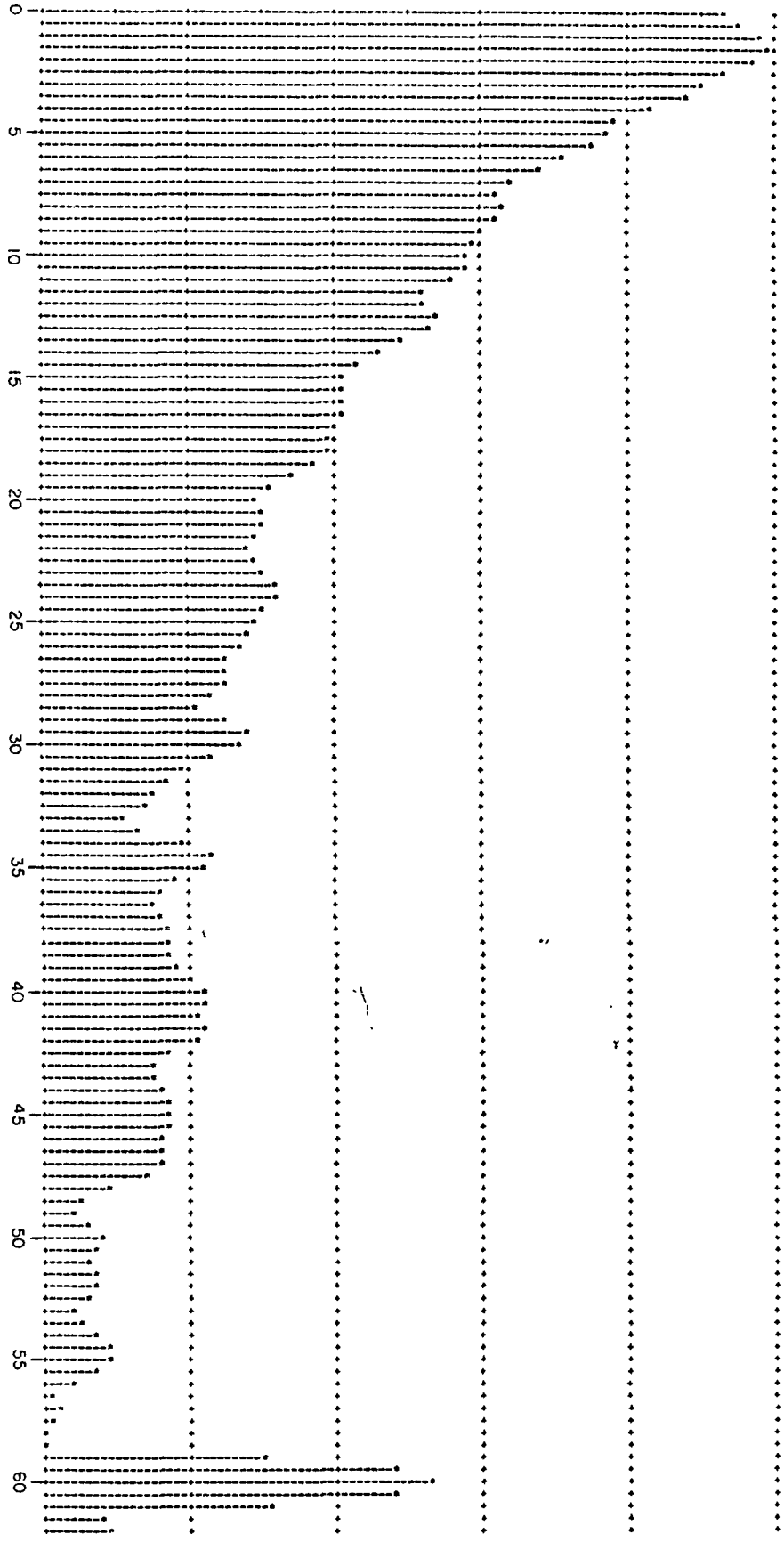

Figure 7B. Reticular Formation EEG When in Slow Wave Sleep

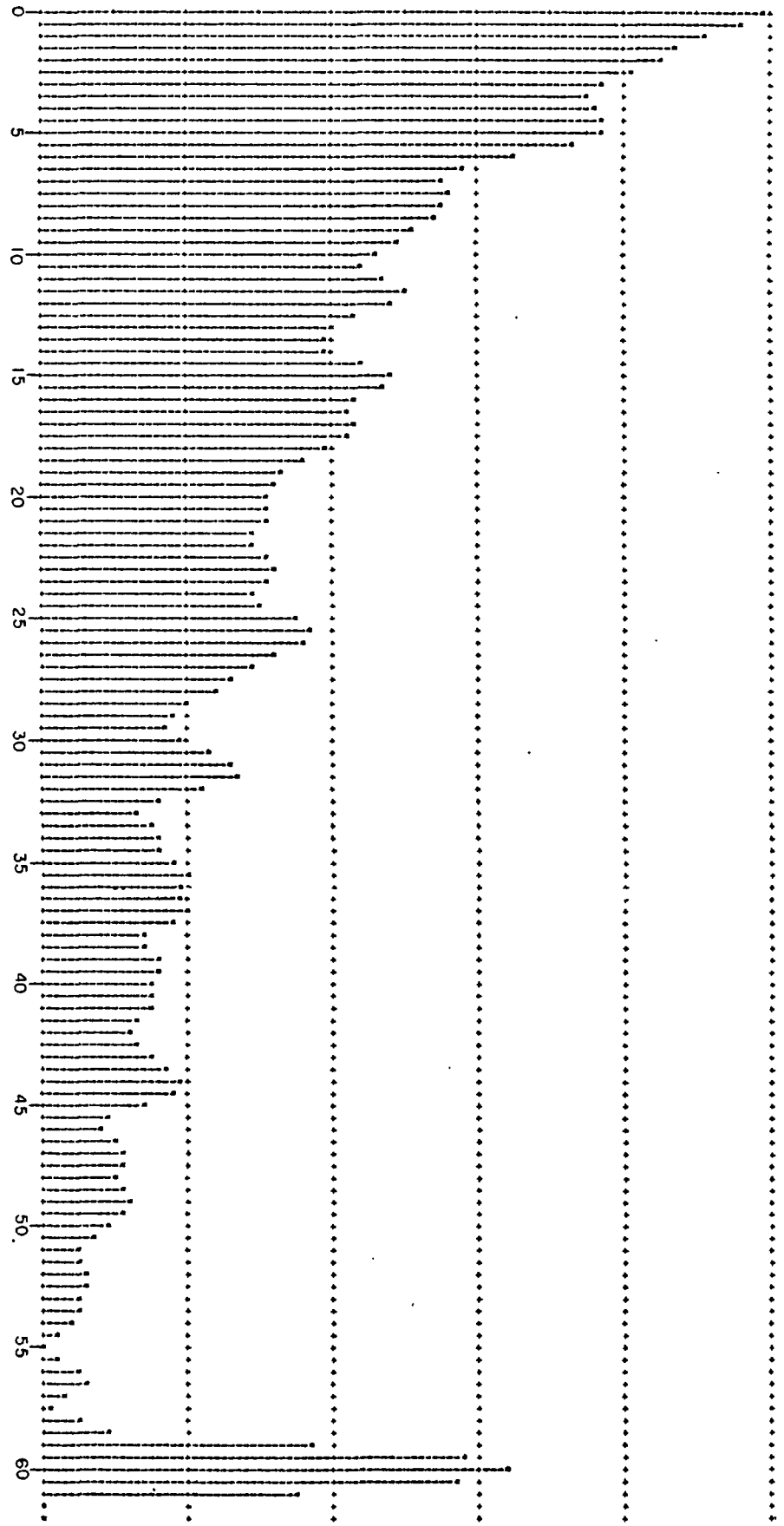

Figure 7C. Reticular Formation EEG When in REM Sleep

only a small theta peak is present in REM. With this derivation, the 40 Hz (actually 38.5 Hz) peak is probably obtained from olfactory regions, and not from mrf.

I discovered many things while debugging this program. In the hybrid system for A-D conversion, several words must be of specific length. Many word lengths are specified in the manuals, but some are not. Because of the length of these programs and the large amount of data to be processed, I tried to use one-half word lengths whenever possible. Some labels must be full-word length, however.

One of these labels is that indicating the number of bytes to be written to tape in ATOD (BYTCT). The maximum number of bytes which can be written to tape is 32,760, a number which will fit into a half-word space but which must be of full word length for proper processing.

Another label which must be full-word length but whose apparent maximum will fit into a half-word space is the number of points to be digitized (N in ATOD). Since the maximum number which can be represented in a half-word is 32,767 and since 32,768 data points make up a full 16-second epoch, it was necessary to sample only 32,764 data points per epoch. The remaining four data points (one point per channel) are filled with zeroes. This does not introduce any error because on cosine tapering the last data point of each channel becomes zero. The 32,767 data points is an artificial limit determined by the subroutine and is not a limit of storage capacity.

The command code used to chain the blocks of data written to digital tape (CC in ATOD) theoretically should have been equal to 8 (IBM, no date). Using this value did not cause chaining, however, and only one block was written to tape. Experimentation and suggestions by ESSL personnel resulted in the finding that another command code for suppressing a possible incorrect length indicator (SLI), equal to 4, had to be added to the 8 to get chaining of blocks. Thus CC is assigned the value 12 in ATOD.

Some investigation was done regarding the repeatibility of the event marking system on analog tape to determine how closely this system could come to the same relative sampling time on repeated runs. The ITIME subroutine was useful here, and it indicated a 0.03 second mean deviation from one pass to another. This indicates that repeated passes through the analog tape would introduce approximately this time shift into data sampled and supports the use of simultaneous sample and hold amplifiers for cross-channel processing since the amount and direction of deviation are not predictable.

PWRSPC takes advantage of the large core of the IBM 360 to compute spectra for the entire epoch at the same time. Some researchers (e.g., Gotman, 1973) who apparently don't have ready access to such large machines have broken one epoch into several smaller epochs for individual processing. They then average these smaller epochs later to obtain a spectrum of the large epoch. According to Bendat and Piersol (1971), treatment of smaller epochs is an acceptable method;

but Dumermuth (1973) claims that the latter technique introduces more leakage and does not give the same results as computation with a single large epoch.

The ESSL library subroutine PLOT1, used for plotting the power spectra on paper, apparently has a slight flaw. It is supposed to print a dotted line from top to bottom on the plot if an ordinate value is less than the minimum given for that plot. This it does, but it also prints this dotted line for some values which are larger than the minimum for the plot. I have not yet detected a consistent pattern to this error, but it usually occurs when the ordinate value is close to, but still larger than, the minimum value on natural number plots.

Since I believed initially that this might be due to rounding errors, I submitted a minimum value that was 0.001 less than the true minimum. This value should have more than compensated for any rounding errors, since the 360 has seven figure accuracy in floating point representation. The problem still occurred, however.

As mentioned in the description of the PWRSPC program, the maximum resolution is 1/16 Hz. Maximum resolution of a spectrum is determined by the formula:

$$R = \frac{1}{T}$$

where T is the epoch length (Blackman and Tukey, 1958). There seems to be no general agreement on what resolution is best for EEG representation. Hord, et al (1965) claims that

a great deal of information is lost if less resolution is used, but also acknowledges that the variance increases from one epoch to another when finer resolution is used. Many studies, on the other hand, use a final resolution of 1 Hz. Thus for the initial phases of my research, I selected a final resolution of 0.5 Hz for the majority of the work. The option of 1/16 Hz resolution, however, is still available using my subroutine DIRECT.

And finally, I believe it is of importance to consider the mechanism of EEG generation in any discussion of EEG analysis, since the mechanism obviously should determine what analytic techniques are used. Apparently the most commonly accepted theory of the EEG origin is that of generation by ipsp's and epsp's (Elul, 1972). Since synchrony in neurons is only intermittent, it seems more realistic to consider the EEG as non-stationary. Elul (1972) mentions only one study, and that unpublished, which attempts to deal with non-linear power spectra, and I have found no other studies treating the EEG as non-stationary data. Thus, for present purposes, virtually everyone seems to assume the EEG stationary as an approximation to true power spectra and other relationships.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

Programs for A-D conversion of analog EEG signals and for computation of power spectra from these EEG data have been described. Once the basic theoretical principles and machine operations are understood, modification of these programs or writing new programs becomes a relatively simple task. For example, an A-D program has also been written for Pat Johnson for multiple channel conversion of EEG data recorded previously with another tape marking system.

A great deal of work could be done to further analyze the EEG signals digitized in ATOD. With the simultaneous sampling, studies of coherence, cross-spectra, and phase relations become possible with suitable programs.

Programs in addition to these for further statistical processing are also possible. For example, a program to do analyses of the results obtained in PWRSPC, either for each cat or across all cats, could give an indication of stability of the spectra across different samples for each type of event. Matousek (1973) offers a review of the literature and discusses some of these possibilities.

Treatment of the EEG as non-stationary data should also be considered, since this would be a more nearly accurate representation.

I am currently investigating the possibility of digital filtering these data with Nicola Papp, an Electrical Engineer graduate studient at University of Houston. This process will involve a series of subroutines to low-pass filter the signals without changing phase relationships and thus preserve the capacity for cross-channel analysis. Filtering would also permit a lower effective sampling rate and thus require fewer calculations and less computer time.

Though the programs described here were written specifically for computations with cat sleep data, they could be readily converted to use in any other type of electrophysiological study using similar frequencies.

REFERENCES

REFERENCES

Baldock, G. R. and W. G. Walter. "A New Electronic Analyzer". Electronic Engineering, 18, (1946), 339-44.

Barlow, J. S. "Autocorrelation and Cross-Correlation". In Matousek, (1973), 79-99.

Bendat, J. S. and A. G. Piersol. Random Data: Analysis and Measurement Procedures. New York: Wiley-Interscience, 1971.

Berger, H. "On the Electroencephalogram of Man". Translation in Gloor, (1969), 37-73. Original in Archiv fur Psychiatrie und Newenhrankheiten, 87, (1929), 527-70.

Bergland, G. D. "A Guided Tour of the Fast Fourier Transform". IEEE Spectrum, 6, (1969), 41-52.

Bingham, C., M. D. Godfrey, and J. W. Tukey. "Modern Techniques of Power Spectrum Estimation". IEEE Transaction Audio and Electroacoustics, AU-15, (1967), 56-66.

Blackman, R. B. and J. W. Tukey. The Measurement of Power Spectra. New York: Dover, 1958.

Brazier, M. A. B., et al. "Preliminary Proposal for an EEG Terminology by the Terminology Committee of the International Federation for Electroencephalography and Clinical Neurophysiology". EEG and Clinical Neurophysiology, 13, (1961), 646-50.

Cochran, W. T., et al. "What is the Fast Fourier Transform". IEEE Transaction Audio and Electroacoustics, (June, 1967), 45-55.

Cooley, J. W. and J. W. Tukey. "An Algorithm for the Machine Calculation of Complex Fourier Series". Mathematical Computations, 19, (1965), 297-301.

Cress, P., P. Dirksen, J. W. Graham. Fortran IV with WATFOR and WATFIV. Englewood Cliffs: Prentice Hall, 1970.

Dietsch, G. "Fourier-Analyse von Elektrencephalogrammen des Menschen". Pflugers Archives of Physiology, 230, (1932), 106-12. Cited from Gotman, et al (1973).

Dumermuth, G. "Numerical Spectral Analysis of the Electroencephalogram". In Matousek, M., (1973a), 33-60.

Dumermuth, G. "Sampling and Data Reduction". In Lopesda Silva, F. H., (1976), 23-31.

Elul, R. "The Genesis of the EEG". International Review of Neurobiology, 15, (1972), 227-72.

ESSL. "'X' Series of Hybrid Executive Fortran Subroutines for Use with the Operating System 360". (no date)

Freeman, F. R., J. J. McNew, and W. R. Adey. "Chimpanzee Sleep Stages". EEG and Clinical Neurophysiology, 31, (1971), 485-89.

Frost, J. D., Jr. "An Automatic Sleep Analyzer". EEG and Clinical Neurophysiology, 29, 88-92.

Gilbert, D. C. Computer Analysis of Electroencephalic Output. Anapolis, Maryland: U.S. Naval Academy, 1973.

Gloor, P. Hans Berger on the Electroencephalogram of Man. Amsterdam: Elsevier, 1969.

Gotman, J., D. R. Skuce, C. J. Thompson, P. Gloor, J. R. Sves, and W. F. Ray. "Clinical Applications of Spectral Analysis and Extraction of Features from Electroencephalograms with Slow Waves in Adult Patients". EEG and Clinical Neurophysiology, 35, (1973), 225-35.

Grass, A. M. and F. A. Gibbs. "A Fourier Transform of the Electroencephalogram". Journal of Neurophysiology, 1, (1938), 521-26.

Hartley, L. R. "A Method for Storing a Program of Stimuli and Responses on Magnetic Tape Using Frequency Coding". Psychophysiology, 9, (1972), 380-82.

Hord, D. J., L. C. Johnson, A. Lubin, and M. T. Austin. "Resolution and Stability in the Autospectra of EEG". EEG and Clinical Neurophysiology, 19, (1965), 305-08.

IBM. OS Data Management Services Guide. (3rd edition), 1973.

IBM. "System 360 Reference Data Card". White Plains, New York, no date.

International Mathematical and Statistical Libraries, Inc. The IMSL Library. Houston: IMSL, 1975.

Kellaway, P. and I. Petersen. Automation of Clinical Electroencephalography. New York: Raven Press, 1973.

Knott, J. R. "Automatic Frequency Analysis". In Walker, Henry, and Merlis, (1953), 17-25.

Knott, J. R., F. A. Gibbs, and C. E. Henry. Fourier Transforms of the Electroencephalogram During Sleep". Journal of Experimental Psychology, 31, (1942), 465-77.

Larsen, L. E., E. H. Ruspini, J. R. McNew, D. O. Walter, and W. R. Adey. "Classification and Discrimination of the EEG During Sleep". In Kellaway and Petersen (1973), 243-68.

Larsen, L. E. and D. O. Walter. "On Automatic Methods of Sleep Staging by EEG Spectra". EEG and Clinical Neurophysiology, 28, (1970), 459-67.

Lopes da Silva, F. H. Handbook of Electroencephalography and Clinical Neurophysiology, (Vol. 4A). Amsterdam: Elsevier, 1976.

Lopes da Silva, F. H. and A. Kamp. "Time Tracking and Coding of Identifiers". In Lopes da Silva (1976), 32-41.

Macy, J. "Analog-Digital Conversion Systems". In Stacy and Waxman (1965), 3-34.

Matousek, M. Handbook of Electroencephalography and Clinical Neurophysiology, (Vol. 5A). Amsterdam: Elsevier, 1973a.

Matousek, M. "Frequency Analysis: Processing of Output Data". In Matousek (1973a), 61-66.

McNemar, Q. Psychological Statistics. (4th Edition). New York: Wiley and Sons, 1969.

Osorio, P. Automated Analysis of EEG Patterns in Subjects Under Abusive Levels of Sedative-Hypnotics. Ph.D. Dissertation, University of Houston, 1976.

Rechtschaffen, A. and A. Kales (editors). A Manual of Standardized Terminology, Techniques, and Scoring System for Sleep Stages of Human Subjects. Los Angeles: University of California, 1968.

Runnals, S. "Appraisal of Alertness and Attention: the Feedback Method". American Journal of EEG Technology, 3, (1963), 51-55. Cited from Matousek's (1973a).

Saltzberg, B. "Period Analysis". In Matousek (1973a), 67-78.

Saunders, and Jell. "Time Distortion in Electroencephalograph Amplifiers". EEG and Clinical Neurophysiology, 11, (1959), 814-16.

Stacy, R. W. and B. D. Waxman (editors). Computers in Biomedical Research, Vol 2. New York: Academic Press, 1965.

Systron-Donner. Precision Time Keeping and Tape Search. Concord, California: Systron-Donner Company, 1970. Cited from Lopes da Silva and Kamp (1976).

Walker, A. E., C. E. Henry, J. K. Merlis. International Congress of Electroencephalography and Clinical Neurophysiology, Supplement 4, Montreal, 1953.

Walter, D. O. "Spectral Analysis for Electroencephalograms: Mathematical Determination of Neurophysiology Relationships from Records of Limited Duration. Experimental Neurology, 8, (1963), 155-81.

Walter, D. O. Handbook of Electroencephalography and Clinical Neurophysiology. (Vol. 4B), Amsterdam: Elsevier, 1972.