A Vibration Analysis System Using Spectral Estimation Techniques

by

Brick Andrew Verser

B.S., Kansas State University, 1982

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

Kansas State University
Manhattan, Kansas

Approved by:

Donald H Lenhert
Major Professor

A Vibration Analysis System Using Spectral Estimation Techniques

TABLE OF CONTENTS

I. INTRODUCTION

In the last 20 years, thanks to the discovery of the Fast Fourier Transform (FFT) and the increasing use of digital computers, spectral estimation has taken on increasingly valuable roles in many different fields. While various techniques of spectral estimation had been used prior to the advent of the FFT, they were typically of limited scale due to the expense of the methods in use at the time [1].

One area where digital spectral estimation has been welcomed is in the analysis of structural dynamics. Understanding the way a structure behaves under dynamic loading is important, and spectral estimation techniques can be used to characterize the dynamic response of a structure. Techniques for analytically calculating the major vibration mode shapes and frequencies of simple structures exist, but using these techniques for complex structures often leads to inaccuracies from the inevitable approximations and idealizations that must be made [2]. Thus, actual measurement of the response of a structure is often necessary.

This paper describes a system which was created to assist in the measurement of the dynamic characteristics of structures through the use of real-time data acquisition and off-line spectral estimation. The system would also function well in acquiring and analyzing other signals, but it has been specifically tuned to serve as a structural dynamics signal analysis system.

II. BACKGROUND

A. History

While spectral estimation has very ancient roots, beginning with the creation of calendars and clocks, the modern era of spectral analysis is generally said to begin in 1949 with J. W. Tukey's presentation of a paper which described how to compute power spectra from empirical data [3]. Tukey's methods were soon used to compute spectra from hand-digitized data, though the work was slow. Even with the increasing use of digital computers, spectral estimation was an expensive process due to the large computation time required for calculating auto-correlation estimates. The introduction in 1965 of the fast Fourier transform by J. W. Tukey and J. W. Cooley changed this, making it possible to inexpensively analyze long time series and vastly increasing the number of applications which could make use of the techniques.

B. Definitions

For a wide sense stationary random process x(t), the auto-correlation is:

$$R_x(\tau) = E\{x(t)x(t+\tau)\}$$

where E represents the statistical expectation operator. From the Wiener-Khinchin theorem, the power spectral density (PSD) is the Fourier transform of the auto-correlation function:

$$S_x(f) = \int_{-\infty}^{\infty} R_x(\tau)\exp(-j2\pi f\tau)\, d\tau$$

In making PSD estimates of actual signals, the basic problem is that it is never possible to have access to all the sample functions of the random process, and typically only a portion of one sample function is available [4]. Certain assumptions about the process must be made. The process is assumed to be wide sense stationary such that it is independent of absolute time; the mean is constant and the auto-correlation function is a function of time differences only. It is also assumed that the process is ergodic so that time averages can be substituted for statistical averages.

Given these assumptions and the limited time window over which the process is sampled, the problem is then one of attempting to estimate the auto-correlation of the measured process such that the estimates do not vary significantly from estimate to estimate. A great many approaches to this problem have been tried and have met with reasonable success [1][3].

In addition to attempting to understand a single random process, it is often desirable to compare two random processes. For example, comparing the movement at various places along a beam which is being subject to a forced vibration can lead to useful insights about the transmission of the vibrations. For this, the auto-correlation estimate can be extended to a cross-correlation estimate from two wide sense stationary random processes, x(t) and y(t):

$$R_{xy}(\tau) = E\{x(t)y(t+\tau)\}$$

Similarly, a cross power spectral density can be defined:

$$S_{xy}(f) = \int_{-\infty}^{\infty} R_{xy}(\tau)\exp(-j2\pi f\tau)\, d\tau$$

The transfer function of the two processes is defined as:

$$T_{xy}(f) = \frac{S_x(f)}{S_y(f)}$$

The coherence function is defined as:

$$K_{xy}(f) = \frac{|S_{xy}(f)|^2}{S_x(f)S_y(f)}$$

The coherence function is similar to a correlation coefficient, but is a function of frequency. If the coherence function is nearly zero at a particular frequency, the two signals are incoherent at that frequency, and are thus probably unrelated. If the

coherence function is nearly one at a particular frequency, the signals are highly coherent at that frequency and are probably closely related.

C. Application to Vibration Analysis

Understanding the dynamics of a structure is a complex task. One approach to the problem is to determine various physical characteristics of the structure and to then attempt to calculate its expected response to dynamic loading. The desired result is a description of the basic vibration mode frequencies and shapes. An analytic method is necessary and is usually fairly accurate. It is a time consuming and error prone process, however, so for complex structures, experimental determination of the actual vibration modes is desirable.

One basic approach taken in experimental modal analysis is to place accelerometers at several locations on the structure and to record the ambient vibrations. The recorded data can then be analyzed to find the major vibration mode frequencies and shapes [5][6]. As such work has been done for many years, the equipment used has generally been traditional analog data recorders, strip chart plots, and for the more extensive investigations, hardware spectrum analyzers.

Better technology for modal analysis has become available recently. Hewlett Packard now markets several dynamic signal analyzers which work well for real-time analysis of data, but this approach still has several drawbacks. The analysis is typically in real time which requires that to analyze the results from several sensors the data must still be recorded on analog or digital tape so it can be replayed multiple times into the signal analyzers. The cost of the equipment is also relatively high.

A less expensive and less cumbersome approach to doing modal analysis was obviously needed. Due to ever decreasing costs and increasing power, the use of a personal computer for the acquisition and analysis of the data seemed ideal. Commercial hardware for converting analog data to digital form was readily available for IBM PC's, and the total cost of a system would certainly be much less than the cost of a single spectrum analyzer.

III. PROJECT REQUIREMENTS

Several requirements were initially identified.

1. System should be capable of handling up to 8 channels of data at up to 200 samples/second/channel for durations of 20 minutes/test. System should provide a real-time display of data as it is acquired.
2. System should compute single and dual channel Fourier spectra.
3. System should compute dual channel transfer functions and coherence functions.
4. System should compute auto and cross-correlations.
5. System should provide an interactive display of any two items.
6. System should provide pen plotter output.
7. System cost should be kept as low as possible.

Adding data acquisition hardware to a PC appeared to be a promising approach to the data acquisition problem. While minimizing the system cost was not the top priority, it was a concern. Zenith Z-150's are very common on campus, and the use of a PC for the acquisition system would allow the machine to be used for other purposes when it was not actually collecting data. Thus, expenses could

be kept to a minimum if a system using a Zenith PC could be put together.

The form the data analysis system would take was initially unclear. Some thought was given to doing the analysis on a PC, but it was quickly determined that a PC does not have adequate speed. The KSU College of Engineering has two Harris 800 minicomputers which would perhaps have worked well for the system, but the machines have only recently been installed and do not yet have the base of terminals, software, and personnel needed to support a major programming project. That left the central campus mainframe as the most likely computer on which to implement the analysis system.

A. Data Acquisition System

Real-time data analysis was not required for this project. Indeed, real-time spectrum analyzers have been used and suffer from the need to record the data samples on some media that can be replayed multiple times into the analyzer to obtain various calculations for each of the channels. As a typical collection of data might consist of 4 channels sampled for 20 minutes, to analyze each of the individual channels, and then to perform cross-channel

analyses, represents a long process of repeated playbacks of the data. Thus, real-time analysis by itself is not practical.

While real-time analysis was not necessary, real-time display of the data being sampled was very desirable so as to ensure that the system was functioning properly during the data collection. The structures at which data is collected are often somewhat remote, and setting up the sensors is occasionally difficult, so it is highly desirable that the experimenter be able to ascertain that he is indeed collecting reasonable data. A real-time display of the data, as it is being collected, would facilitate this.

The data acquisition system would have to be reasonably portable. It did not have to be battery powered, but did need to fit in a car. While a normal IBM PC with its attendant keyboard and monitor is not usually considered portable, for this application it was adequate.

The system had to be capable of handling 4 and preferably 8 data channels, each running at 200 samples per second for durations as long as 20 minutes continuously. This corresponds to approximately 2 million total samples in a single 20 minute period. Dealing with

this quantity of data turned out to be one of the more difficult aspects of the data acquisition system.

It was desirable to have available some simple on-site data analysis routines. The ability to replay a small segment of the data or to compute a simple spectrum estimate from a data channel would greatly aid the experimenter in determining whether the sensors were functioning properly and would aid in determining necessary parameters such as the sample rate and external lowpass filter settings needed.

B. Data Analysis System

Full-featured on-site data analysis was not considered necessary for this system. The IBM PC, or equivalent system, that was envisioned as the center of the data acquisition system does not have adequate processing power to speedily compute spectrum estimates from many thousands of data values. While a PC can quickly compute the Fourier transform of one thousand data points, computing a spectral estimate from many thousands of values would be impractical. Thus, the analysis system was expected to run on a large host computer. The host system chosen was the central campus mainframe, an IBM/370-compatible NAS

6630 running VM/SP CMS. The data collected during an experiment would be uploaded to CMS at a later time for complete analysis.

The mainframe analysis system needed to be able to provide estimates of the following: power spectral density and cross power spectral density, auto-correlation and cross-correlation, transfer function, and coherence function. Interactive and paper plots of the results were required.

IV. EVOLUTION OF THE DATA ACQUISITION SYSTEM

The current implementation of the data acquisition system was chosen after rejecting several alternate approaches. The basic requirements of the system were not absolutely steadfast, but it was desirable to achieve each of the stated objectives if at all practical.

Some thought was given to trying to implement the data acquisition system without using a personal computer. This would have the advantage of not requiring a somewhat expensive and delicate instrument be carried with the sensors. But no alternative could provide the desired simple on-site data analysis for a comparable price. A stand-alone analog or digital tape recorder could be used, but then the data is collected essentially blind since no simultaneous playback is available with such a system. After a set of data is collected, it can be played back to a strip chart recorder to verify that something reasonable was indeed collected; still, no analysis is available with this approach. Adding a spectrum analyzer solves this lack of processing power, but very significantly increases the cost of the needed equipment. It was fairly apparent that a PC based system would be less expensive.

Given that a PC with an analog-to-digital (A/D) converter was going to be used, the next problem was to find the hardware and software that would meet the project requirements. Three A/D systems were reviewed in some detail before the hardware was chosen.

Cyborg Corporation markets a relatively powerful and well known group of data acquisition and control products and seemed to have an appropriate system which attaches to the IBM-PC. The ISAAC 2000 is essentially a stand-alone system which contains its own processor, memory and A/D equipment. This allows it to acquire data into a buffer memory of up to 2MB which is transferred to the PC after the experiment has ended. 2MB of memory is about enough to satisfy the requirements for the amount of data that needed to be collected in each run, but the ISAAC 2000 does not have provisions for the real-time display of the collected data. In addition, the company had never tested the system on a Zenith Z-150 IBM-PC compatible, was not interested in doing so, and would not provide a system for a trial period. As the Z-150 was the most likely PC to be used for the data acquisition system, this was an important consideration.

MetraByte Corporation makes several data acquisition boards which can be plugged into the expansion slots of an

IBM-PC or compatible. While their DASH-8 board provided the necessary eight A/D channels and would support sample rates up to 30,000 samples per second, the interface to the PC requires that the software continuously poll the board to collect the data. This would make the continuous collection of large amounts of data very difficult due to the need for overlapping disk I/O with the data acquisition. MetraByte also offers the DASH-16 which interfaces to the IBM-PC via direct memory access (DMA). This allows the PC to perform other functions (such as disk I/O) concurrent with the data acquisition. While this board would very probably have performed well for our application, another board was chosen for a non-technical reason.

The Data Translation DT2801-A was actually chosen to provide the needed A/D functions. The board is functionally similar to the MetraByte DASH-16 and was, in fact, slightly more expensive. It provides 16 channels of A/D input with a maximum sample rate of 27,500 Hz with a DMA interface to the PC, 2 channels of D/A output, and 16 bits of digital I/O. The DT2801-A was made particularly attractive due to the availability of a board for test use. The KSU Department of Speech had previously purchased a DT2801-A and made the hardware available for

an extended period. This allowed software development to proceed considerably earlier than had the project had to wait for another board to be purchased.

One of the requirements of the data analysis system was that it be able to compare the phase of the input data channels. Normal multi-channel A/D boards contain a single A/D converter and simply gate the appropriate data channel to the converter for each sample, thus causing the channels to be converted at different times. For signals with frequency components much less than the sample rate, this will result in only minimal phase distortion, though for signal frequencies near the Nyquist frequency, the phase data will be very badly skewed.

This dictated that the data acquisition system be able to simultaneously sample each of the data channels, or that the sample rate be set much higher than the highest frequency component in the data. Simply increasing the sample rate causes a corresponding increase in the amount of data collected and is thus very undesirable. Hardware which simultaneously samples 4 channels of data was found, but was considerably more expensive than equivalent hardware which did not provide the simultaneous sampling feature. In addition, this hardware suffered from the requirement that the data be sampled at a rate of at least

200 samples per second which again would often create excessive amounts of data. An alternative approach was to use an external clocking device for the A/D hardware which would send a stream of clock signals at the maximum rate the A/D hardware can convert (one clock pulse for each data channel), and follow it by a relatively long pause. So long as the A/D conversion rate is considerably higher than the sample rate, very little phase distortion will occur.

Originally, it was hoped that commercial software meeting the requirements could be found. Many programs for driving the A/D hardware were found, and many programs for doing the necessary analysis were also found. Unfortunately, no software for acquiring the data could be found that met the requirement for collecting two million samples per experiment, and that could provide real-time data displays. Several of the programs are simply routines designed to be called from interpreted BASIC which fill BASIC data arrays. As this would limit the acquisition to less than 30,000 samples, these packages were completely unacceptable. Examples include Data Translation's PCLAB and Laboratory Technologies' NOTEBOOK. Other packages could meet the requirement for collecting two million samples but provided no means of real-time

display. Examples of these include Signal Technology's ILS-PC and Macmillan Software's ASYST. Were these routines inexpensive, it might make sense to use them and do without the real-time display, but ILS-PC and ASYST are not inexpensive and come without program source. They would not provide the desired function, would not be tailorable to this particular application, and would not be cheap. It thus made sense to write the data acquisition software.

V. THE DATA ACQUISITION SYSTEM

The data acquisition system consists of a Zenith Z-150 personal computer (note that a Z-158 will NOT work, as will be explained later) with a 20 MB Winchester disk drive and 8087 numeric coprocessor, a DT2801-A A/D board, a DT707 screw terminal panel, and a custom-built A/D clock. The DT2801-A and the A/D clock occupy two expansion slots in the Z-150 and are connected to the outside world through connectors accessible at the back of the PC. The DT2801-A connects to the DT707 screw terminal panel through a ribbon cable and the DT707 is placed in a custom built sheet metal box which has BNC connectors for each of the eight data channels. In addition the metal box has a small phone plug connector which is used to get the external clock from the A/D clock board in the PC to the DT707.

A. The External Clock Hardware

A clock board was built to drive the DT2801-A external clock input. The board plugs into an expansion slot in the Z-150 and creates a train of pulses, one pulse for

each data channel, where the pulses within a burst are separated by a small amount of time (the time is programmable and is set by software to 4.0E-05 seconds), and the time between bursts is set by software to the sample rate per channel chosen by the user. This has the effect of providing nearly simultaneous sampling of each of the data channels when the overall sample rate is considerably below 25 KHz.

The DT2801-A requires that, after it is programmed to do the data acquisition, a one millisecond delay be followed by a single synchronizing pulse on the external clock input. After this, each succeeding pulse causes the next data channel to be latched and converted. Thus, the external clock must first send a single pulse and then follow it with the pulse trains. The major difficulty encountered designing the external clock was to provide this careful synchronization. If a free-running pulse generator is used to provide the clocking, the DT2801-A will likely not be processing channel 1 at the time the first pulse of the train arrives, and the PC will be unable to determine which data channel is converted by the first pulse of the train.

The board consists of an Intel 8253-A timer chip, a TTL oscillator, the address decoding logic required to give

the board a unique hardware address (the addresses used are hexadecimal 2B0, 2B1, 2B2 and 2B3), and other necessary support devices (see figure 1).

The 8253-A consists of three independent 16-bit counters, each with several different modes of operation, with its own external clock, gate input, and output [7]. Careful use of the gates and output signals, and appropriate programming of the 8253, allows the board to generate the desired synchronizing transition and clock pulse train.

The board is built so that a single 500 KHz clock feeds each of the three counters. The gate for counter 2 is always enabled, thus allowing counter 2 to run at any time. Counter 2 sets the overall sample rate when generating the pulse trains. The output of this counter is used as the gate for counter 1; counter 1 can run only when the output of counter 2 is high. When generating pulse trains, counter 1 enables counter 0 for short periods to allow counter 0 to generate the correct number of pulses; that is, counter 1 sets the number of pulses generated. The output of counter 1 is then inverted and used to gate counter 0; counter 0 can run only when the output of counter 1 is low. Counter 0 feeds the DT2801-A; when generating pulse trains it pulses once for each of the channels to be sampled.

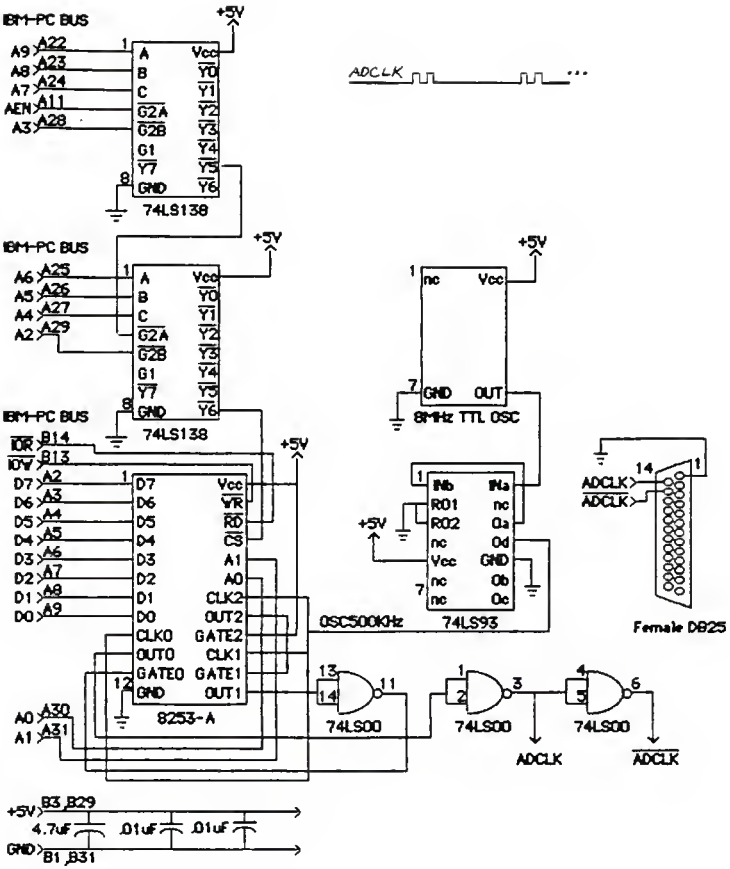


Figure 1. Schematic of external clock board.

The programming of the 8253-A to generate the single synchronizing pulse is fairly straightforward. Counter 2 is forced high and left that way to allow counter 1 to be programmed as a one-shot into counter 0 which is in turn programmed to pulse low one time. Counter 0 is attached to the external clock interface of the DT2801-A. The 8253-A programming steps are shown:

```
Set Counter0 mode to 1 (One-shot triggered by gate)
Set Counter1 mode to 4 (Strobe triggered by software)
Set Counter2 mode to 4
Set Counter0 value to 2 (To generate a very short
                strobe when Counter1 triggers our
                gate)
Set Counter1 value to hex 800 (Clocked at 500 KHz,
                this is 2 milliseconds)
```

As soon as a value is loaded into Counter1, it begins counting down since Counter2, which provides the gate signal, is set to allow Counter1 to run. When Counter1 counts down to zero, it sends a pulse which causes Counter0 to send a pulse which is interpreted by the DT2801-A as its synchronizing signal.

In a similar fashion, the counters can be programmed to send the pulse trains. The output of counter 0 is sent to the DT2801-A external clock interface. It is programmed to count at a rate near the 27.5 KHz maximum rate of the DT2801-A, and is only allowed to count when its gate signal is high. Counter 1 provides the gate signal for counter 0. It is programmed to rise and stay high for a

short period when its gate signal goes active. The duration of its high state is programmed so as to allow "n" pulses from counter 0 before counter 1 returns to a low state, where "n" is the number of channels of data being sampled. Finally, counter 2 provides the trigger signal for counter 1. Counter 2 is programmed to send a square wave at the desired sample rate.

For this application, counter 0, which provides the clock for the DT2801, is set to send one pulse every several clock cycles. Counter 1 is set as a one-shot which allows counter 0 to run while counter 1 is counting down. Counter 2 is set to send a square wave which is used to start the counter 1 one-shot. Thus, the value loaded into counter 0 is chosen to send one pulse every 20 clock cycles where the clock runs at 500 KHz. This provides the 4.0E-05 second spacing between adjacent channel conversions and is close to the 27.5 KHz maximum rate the DT2801-A can handle. The value loaded into counter 1 determines how many pulses will be sent and thus the number of channels that are converted during each pulse train. The value loaded into counter 2 sets the overall sample rate per channel. The programming steps are shown:

```
Set Counter0 mode to 2 (Divide by N rate generator)
Set Counter0 value to decimal 20
Set Counter2 mode to 3 (Square wave rate generator)
Set Counter1 mode to 1 (One-shot triggered by gate)
Set Counter1 value to 20*chancnt+2 (To let Counter0
                pulse once for each channel)
Set Counter2 value to sample rate (This will start
                the pulse trains)
```

It should be noted that the order in which the modes and values are set is not arbitrary; they are chosen so as to prevent stray pulses from being sent while the timer is being reprogrammed.

B. The Data Acquisition Software

The software that is used to drive the DT2801-A and external clock boards was written in C and compiled using the Computer Innovations C86 compiler. It takes advantage of several of the run-time routines that are provided with the CI compiler, but otherwise should be fairly easy to port to another C compiler. Appendix B contains the source listing of the DTDATA program. The DTDATA program is menu driven, allowing the user to set several parameters before beginning the data acquisition.

A real-time plot of one or two data channels on the Z-150's graphic display is provided. If more than two channels are being recorded, only two at a time can be displayed, though the choice of the two channels can be

changed during the data acquisition. Due to performance limitations, the real-time display can be run only up to a total of 250 samples per second. One channel running at 250 sps can be displayed, or two channels running at 125 sps can be displayed, but two channels running at 200 sps exceed the display speed. A faster processor, or the coding of the display routine in 8086 assembler would provide for faster real-time displays.

The acquired data is written to disk as it is obtained. A short header, written at the start of the data file, records several experiment parameters such as the current time and date, the number of data channels recorded, the sample rate, the external lowpass filter setting for each channel, the conversion multiplier needed to change the 12-bit integer data to units of centimeters per second per second, and a 15 character ASCII string for each channel which can be used to record other data such as a sensor location code. The header contains only ASCII characters and normal-ordered integers (rather than the byte swapped integers used by the 8086). To record the sample rate and conversion factor, a four byte integer is used as a base and is multiplied by a four byte integer field, divided by a four byte integer, multiplied by a two byte integer and divided by a two byte integer. While this scheme appears

slightly awkward, it actually is fairly straightforward to code and works well. It is highly desirable to avoid putting binary floating point values in the header since the internal floating point representations of machines differ. Integer representations are much more standard.

When the user begins the DTDATA program, a menu is presented to allow appropriate parameters to be set. The menu is shown below:

```
A.  Recall parameters
B.  Disk filename:  <no disk log>
C.  Number of channels to record: 1
D.  Sample rate per channel: 125.00 Hz
E.  Duration of sampling: <continuous>
F.  Number of display channels: 1
G.  Clocking:  EXTERNAL
H.  Data log
I.  Save parameters

Enter letter of item to change, RETURN to begin, or
ESC to exit:
```

The menu is ordered such that the user will likely select choice A first, followed by B, and so on. Selection A is used to retrieve previously recorded parameters from the disk file, "DTDATA.DEF." Selection I writes this file using the currently set parameters. For selections B, C, D, E, and F, the user is prompted for the appropriate value after selecting the menu item. Selection G, clocking, simply toggles between EXTERNAL and INTERNAL while selections A and I simply perform the

requested function. When selection H, data log, is chosen, the user is requested to enter data which is placed in the header record. For each of the data channels, the user is requested to enter a 15 character location code, an external amplifier attenuation setting, and a lowpass filter setting. The external amplifier attenuation and lowpass filter setting are controls on the Kinemetrics SC-1 signal conditioner which is normally used to amplify the signals from the accelerometers.

After the appropriate parameters are set, pressing the RETURN key will begin the data collection and display. During collection, the upper part of the PC screen is used to display the data, using left and right halves if two channels are displayed. The bottom part of the screen shows what channels are being displayed and has reminder lines which describe how to stop the data collection, how to change display channels, and how to change the gain on the display. The adjustable gain is useful when low level signals are being collected since only 128 graphics pixels are available for plotting the vertical axis and the input can take on 2048 values; unless the input is running at nearly full scale, the display gain can be increased to make better use of the display area available. Figure 2 shows an example of the display produced by DTDATA collecting two channels of data.

Figure 2. Sample display from DTDATA.

After the data collection has finished, either because it was aborted by pressing the ESC key or because the duration limit was reached, the menu is redisplayed. The user can exit DTDATA and return to MS-DOS by pressing the ESC key, or the parameters can be reset and another collection can be started.

The use of DMA on the DT2801-A board allowed a fairly simple scheme to collect data continuously for long durations; the actual amount collected is limited only by the size of the hard disk available to store the data. This is done by allocating a large DMA buffer, programming the DMA hardware on the Z-150 to repeat when it reaches the end of the DMA buffer, and then starting the DT2801-A collecting. The DT2801-A will simply send data to the DMA controller as it is available while the DMA controller will, while the PC is free to perform other tasks, place the data in the memory of the PC using the circular buffer. After the DTDATA program starts the data collection, it simply reads the status of the DMA controller to determine what address will be used next. This value is continuously updated by the DMA controller and therefore tracks the input. DTDATA can calculate how much data has been processed, write it to disk and optionally graphically display the values without

concerning itself with handling individual samples from the incoming data. Without DMA, the PC software would have to itself perform the multitasking which the DMA controller handles. The software would have to continuously poll the A/D device or handle interrupts from it and issue appropriate commands to transfer data from the hardware to memory, while at the same time display the data and record it to disk. Because of the need for the PC to be constantly polling or fielding interrupts, the maximum data rate of the system would be significantly slower than by using DMA, and the software would be considerably more complicated.

Unfortunately, the use of the DMA controller chip in its repetitive mode caused some problems. While this worked perfectly on a Zenith Z-150, it did not work on the newer Zenith Z-158. Some experimentation lead to the conclusion that the DMA controller in the Z-158 was not restarting its operation after it filled the input buffer one time. It seemed to ignore the command it had been given to reload itself when the buffer filled. A quick look at the schematics of the Z-158 showed the reason for this. While the Z-150 uses an Intel 8257 DMA controller chip, identical to what the IBM-PC uses, the Z-158 places all the DMA functions into a custom VLSI chip.

Apparently, the VLSI chip does not faithfully emulate the Intel chip. Thus, DTDATA does not work on Z-158's and it is unlikely that Zenith will re-engineer their VLSI device just to fix this problem.

C. The PC Signal Analysis Software

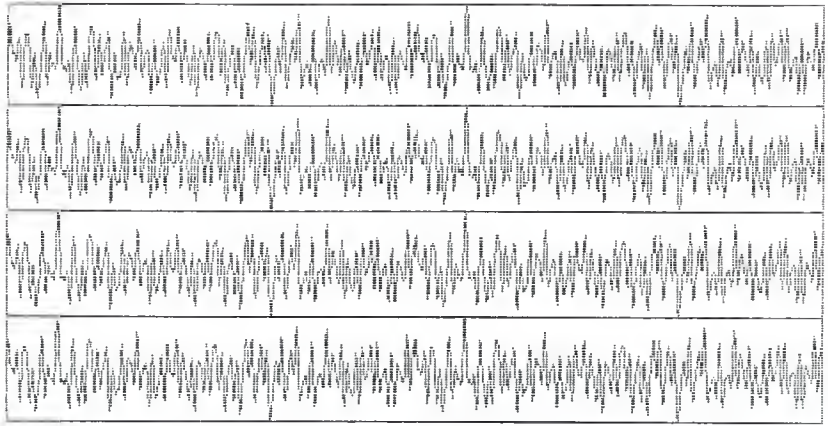
In order to assist the experimenter in setting appropriate parameters such as sample rate and lowpass filter frequency, the programs PLTDAT, PLTDAT4, PLTFAS, and PLTPSD were written. The programs provide very limited on-site analysis and data verification; they are not intended in any way to replace the mainframe data analysis routines. Appendix B contains the source listings of these programs, including source for several ancillary routines required by these programs.

PLTDAT and PLTDAT4 produce screen plots of the first 1024 values of a data file. PLTDAT plots only one channel of the data while PLTDAT4 plots four channels on the same screen. They enable the user to verify that the data actually did make it onto disk, and that the data visually appears correct.

PLTFAS and PLTPSD produce screen plots of the Fourier amplitude spectrum and power spectral density of the first

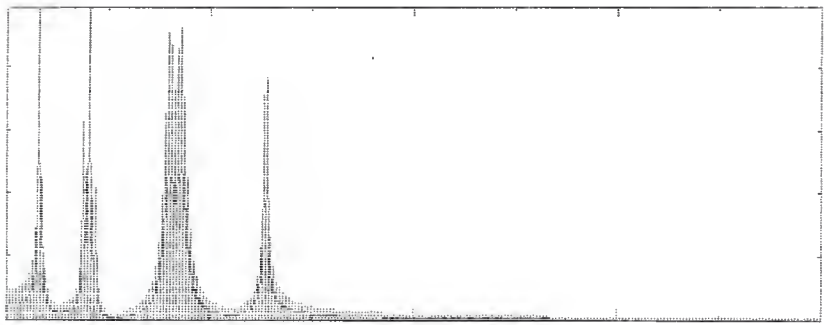
1024 points of a data file. No windowing or time averaging is done on these plots so they will tend to be be quite rough and inaccurate, but accurate enough in most cases to provide the needed information. Their primary use is in selecting appropriate settings for the sample rates used in an experiment. Typically, the user will not know the frequencies of the vibration modes of the structure being tested and so will first collect a short segment of data at a very high sample rate (>= 200 sps) with the lowpass filters set correspondingly high. By looking at the displays produced by PLTFAS or PLTPSD, the experimenter can determine the highest frequency of the significant modes and choose an appropriate sample rate and filter frequency to capture the desired modes without oversampling. Both PLTFAS and PLTPSD can plot on a linear or logarithmic vertical scale.

Figure 3 shows an example of the display produced by PLTDAT4 and figure 4 shows a display from PLTFAS. The data is the same in both cases and is simply a mixture of multiple sinusoids.



File 'blah4.raw', 1st Channel = 1, sps = 100.00
maximum = 0.022cm/sec/sec minimum = -0.018cm/sec/sec #pts = 1024

Figure 3. Sample display from PLTDAT4.



File 'blah4.raw' Channel 1, Max Frequency=50.0000
Max Amplitude=0.0016, #Points=1024

Figure 4. Sample display from PLTFAS.

VI. EVOLUTION OF THE DATA ANALYSIS SYSTEM

Once the data has been collected by the PC and A/D hardware, it must be analyzed. While it would certainly be possible to do the desired analysis on the PC, the time required to do so makes the prospect somewhat unpleasant. Given a PC approximately five times the speed of an IBM-PC, microcomputer based analysis would make good sense. Since such a machine was not available at a reasonable cost, it was decided to use the campus mainframe.

The basic requirements for the analysis system were that it be able to provide estimates of power spectral density and cross power spectral density, auto-correlation and cross-correlation, transfer function, and coherence function. Interactive displays of any two of these functions was desired in addition to high quality hardcopy plots. The means of getting the raw data from the PC to the mainframe is provided by the KERMIT file transfer program [8]. While 9600 baud file transfers do not appear particularly fast when moving a few megabytes of data, the transfers can be started and allowed to run unattended so the time demands on the user are relatively small.

Several tenets were developed very early in the design of the system. These provided the basic philosophy that shaped the system as design and implementation progressed.

It was decided early on that it would be inappropriate to attempt to use a single computer language to implement the system. While FORTRAN provides very good computational functions, it is extremely lacking in certain areas which were important. It has poor character handling, poor structured programming constructs, and limited I/O interfaces. Thus, while FORTRAN was used to implement the basic computational routines, another language was used for the higher level work. The C programming language would have worked fairly well for this, but the mainframe lacks a good C compiler. In addition, C programs tend to be difficult to debug, particularly for inexperienced programmers who might find themselves working on this system in the future. PL/I would have been a reasonable language for both the user interface and the computational work, but it is a complex, uncommon language which has implementations for only a few computers. About the only remaining mainframe language that would be appropriate for this function is REXX [9][10][11]. REXX is an interpreted language which has very powerful structured programming constructs, is easy

to debug, is not difficult to learn, and which interfaces to the CMS command environment very naturally.

During the implementation, attempts were made to keep the programs reasonably portable. In cases where there seemed to be no alternative, such as in choosing REXX for the user interface, portability was sacrificed. But the basic FORTRAN programs which do the computation should be relatively easy to port to another machine, including the IBM-PC. Since a version of REXX is available for the PC which provides an environment almost identical to that in CMS, the entire analysis system should be fairly easy to move to an IBM-PC compatible machine. To move the system to a Unix, VMS, or Vulcan environment, several hundred lines of REXX would have to be rewritten in another language such as C; this would not require undue effort since C uses many of the same language constructs as REXX. Even if the user interface had been coded in C on the mainframe, a fair amount of conversion effort would be required since a certain amount of operating system specific code is inevitable.

The analysis system is very modular. The basic computation is done by several separate FORTRAN routines. These routines are called by small REXX programs which know the basic requirements of the FORTRAN code and can

issue the CMS commands to point the FORTRAN I/O units at the data. These low level REXX routines are in turn called by a higher level REXX program which provides the command interface to the user. The user sees a single command interface while the programmer sees a highly modular collection of related programs without the types of code and data interdependancies that would result from trying to create a single large program to handle all functions. In addition, the memory requirements of the system are considerably reduced by this approach since only the array space needed for a particular application need be allocated while a combined program would have to have arrays for each of the functions pre-allocated (thanks to the lack of dynamic memory allocation in FORTRAN).

While the high degree of modularity somewhat dictated this, another design consideration was that all the working data be stored on disk rather than in CPU memory. This was also forced in part due to the large potential size of some of the input data expected (a single data channel might well contain 250,000 samples). While the mainframe and most large minicomputers could deal with multiple megabyte memory requirements, most smaller machines could not. Reading and writing the data to disk

tends to slow the speed of the system, but in general this should not be a major consideration as the processor time required to perform the calculations usually far exceeds the time required to read the data and write the results.

VII. THE DATA ANALYSIS SYSTEM

From the user's point of view, the entire data analysis system consists of a single CMS command called "VAS" (Vibration Analysis System). This one command accepts many parameters and options to provide all the required data analysis and display functions. In actual implementation, a large number of separate REXX and FORTRAN programs are used. Appendix C contains source listings for the programs.

The data and results are stored in CMS disk files. In CMS, files on a disk are identified by a filename and a filetype which can each contain up to eight alphanumeric characters. The filename is used by the VAS command to uniquely identify a single experiment and appears on the printed and plotted output as the test name. The filetype is used internally by VAS to identify the kind of data contained in the file. While CMS allows filetypes to be up to eight characters in length, for portability considerations only three character filetypes are used. The user can in general concern himself very little with the filetype. The following filetypes are reserved by the VAS command:

FILETYPE	Function
RAW	Unprocessed data in the form created by the DTDATA command
Dii	Data for channel "i"
IOO	Various information about the data in other files
Sij	Cross PSD estimates for channels "i" and "j" or auto PSD estimate if i=j
Cij	Auto or cross-correlation estimates for channels "i" and "j"
Kij	Coherence function estimates for channels "i" and "j"
Tij	Transfer function estimates for channels "i" and "j"

The user of the system seldom has to concern himself with filetypes. The filetype of "RAW" must be used when uploading the data, and filetypes are needed when using standard CMS commands to list and manipulate disk contents.

A. The VAS CONVERT Command

When a single file containing the data from a particular experiment is uploaded, it must first be converted to a format more amenable to processing on the mainframe. The raw data file begins with a header record which contains information about the experiment followed by the data in reversed-byte integer format with the data for multiple channels interleaved. The "VAS CONVERT" command is used to separate the raw format data into one

CMS file for each data channel plus an informational file which contains information from the header. The data is kept in files with a filetype of "Dnn" where "nn" is "11" for the first channel, "22" for the second channel, through "88" for the eighth channel. The data is written in internal REAL*4 format (in binary with four bytes per value). The informational data is written to a file with a filetype of "I00". A single record containing the sample rate and the date and time the data was collected is followed by eight records containing the lowpass filter frequency and location code for each channel. In all cases, the filename of the files is identical to the filename of the raw data. For instance, if the file "TEST1 RAW" contained the raw data for an experiment run with two channels, the command:

```
VAS CONVERT TEST1
```

would create three files: "TEST1 D11," "TEST1 D22," and "TEST1 I00."

The conversion function is performed by the FORTRAN program SPCONV which is driven by a REXX program of the same name ("SPCONV EXEC").

B. The VAS SPECTRUM Command

PSD estimates and cross PSD estimates are computed by the "VAS SPECTRUM" command. For instance, the following commands could be used to compute the auto and cross PSD estimates containing 1024 frequency components of an experiment called "TEST1":

```
VAS SPECTRUM TEST1 1 1 (LENGTH 1024
VAS SPECTRUM TEST1 2 2 (LENGTH 1024
VAS SPECTRUM TEST1 1 2 (LENGTH 1024
```

Options are available on the "VAS SPECTRUM" command to select from two spectral estimation methods, to select a smoothing window and to select the number of frequency components the estimation is to use. Refer to the Vibration Analysis System User's Guide in Appendix A for the complete format of the "VAS SPECTRUM" command.

The command uses files of filetype "Dii" and "Djj" as input and produces a file with a filetype of "Sij" as output where "i" and "j" represent channel numbers. For a cross spectral estimate, "i" and "j" are different while for an auto spectral estimate they are the same. The resulting output file is written in COMPLEX*8 format (binary with 8 bytes per value, 4 containing the real part of the result and 4 containing the imaginary part of the result). For an auto-spectrum, the imaginary part of the result is zero though the data is still written in complex format.

Two methods of spectral estimation were implemented. The default is the method of modified periodograms suggested by Peter Welch and often known simply as Welch's method, or as the method of weighted, overlapped, segment averaging (WOSA) [12]. A cross PSD is computed using this method by the following procedure:

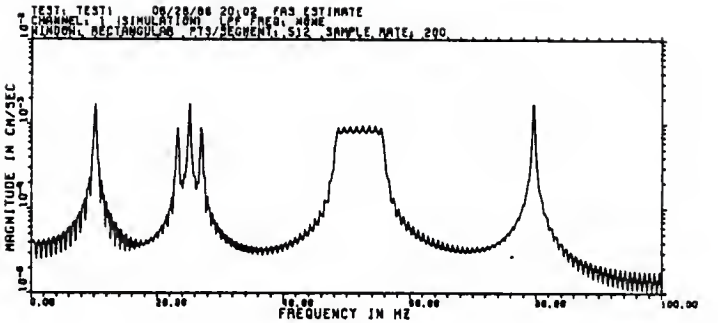
1. The two input sequences, x and y, are broken into subsequences of length, M, where the subsequences are overlapped by 50%. The two sequences must be of the same length.
2. Each subsequence is multiplied by the chosen window.
3. The subsequences are zero padded to length 2*M to improve the apparent frequency resolution.
4. The PSD of the subsequences of x and y are computed as CONJ(DFT(X))*DFT(Y).
5. The PSD of each of the subsequences is averaged to produce the desired result after adjusting the values to correct for the power reduction caused by the application of the data window.

The second method of spectral estimation implemented is an older method which is nicely outlined by Beauchamp [13]. In this method, the input data is also broken into subsequences (though the subsequences are not overlapped), but the windowing is done on the auto-correlation estimates of the subsequences rather than directly on the data. The procedure outlined by Beauchamp is as follows:

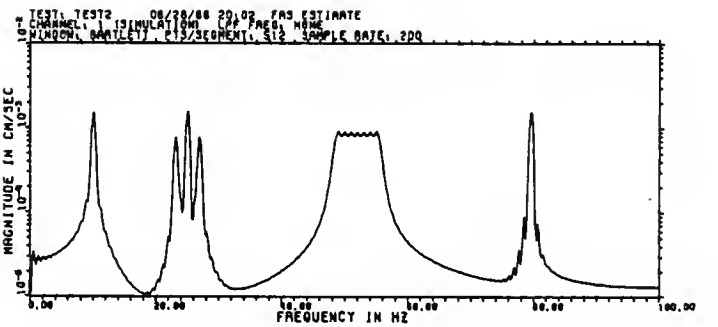
1. Divide the input x and y of length N into segments of M values each.
2. Attach M zeros to each segment and perform a 2M-point FFT on the segments of x and y. This produces X(f) and Y(f).
3. Average CONJ(X(f))*Y(f) over all the segments.
4. Produce the cross-correlation estimate of x and y by computing the inverse FFT of the averaged result.
5. Multiply the correlation estimate by the chosen window.
6. Produce the PSD estimate by computing the FFT of the windowed correlation estimate.

The major advantage of this method appears to be speed. For long sequences, Welch's method performs about twice the number of FFT computations as this speedier traditional approach. But the speed appears to come with some penalties as the spectral estimates generally do not have quite the same resolution and accuracy of those from Welch's method.

Figures 5 (a-g) and 6 (a-g) show the results of computing 512 point Fourier amplitude spectrum (FAS) estimates of a synthesized signal. (The FAS is simply the square-root of the PSD and is more applicable in many cases to the study of structural dynamics than is the PSD.) The signal, 8192 values sampled at 2000 Hz, consists of the sum of sinusoids at 101 Hz and 796 Hz plus an amplitude modulated sinusoid at 251 Hz with sidelobes at 213 and 289 Hz, plus a swept frequency sinusoid which begins at 480 Hz and ends at 520 Hz. Figure 5 shows the

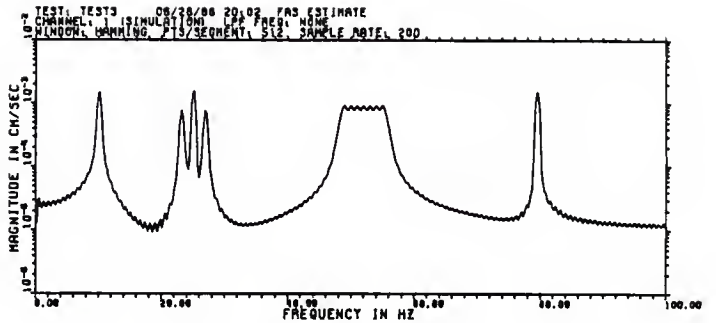


(a) Rectangular window.

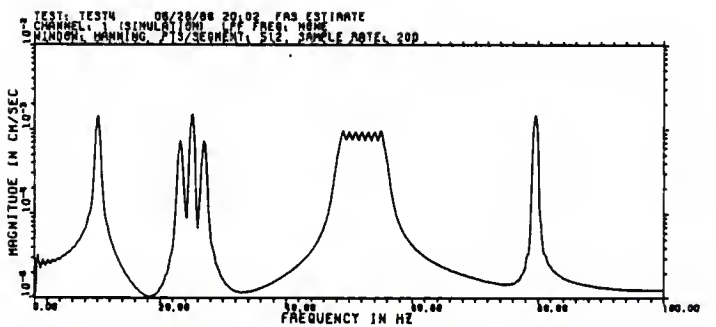


(b) Bartlett window.

Figure 5. Welch's method FAS estimate of summation of sines.

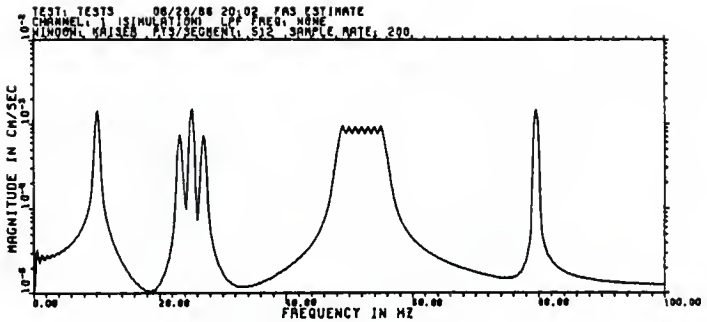


(c) Hamming window.

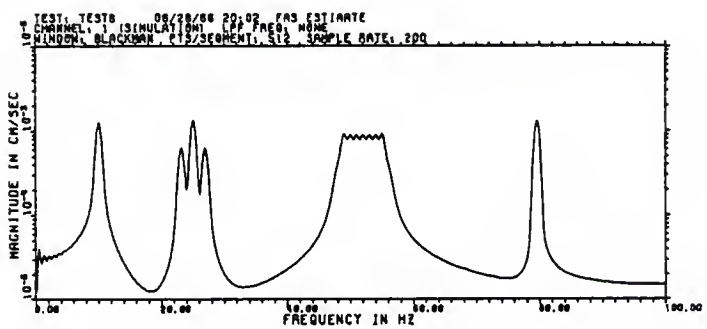


(d) Hanning window.

Figure 5. Welch's method FAS estimate of summation of sines (cont.)

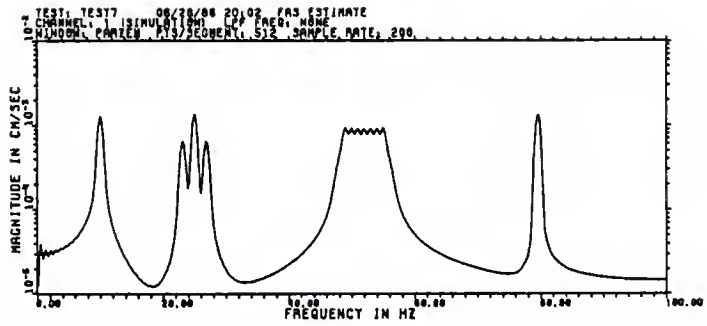


(e) Kaiser window.



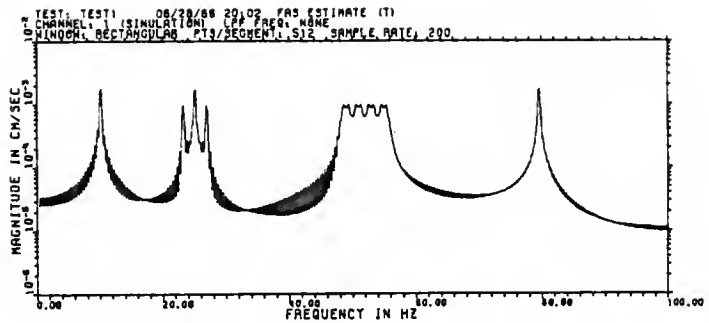
(f) Blackman window.

Figure 5. Welch's method FAS estimate of summation of sines (cont.)

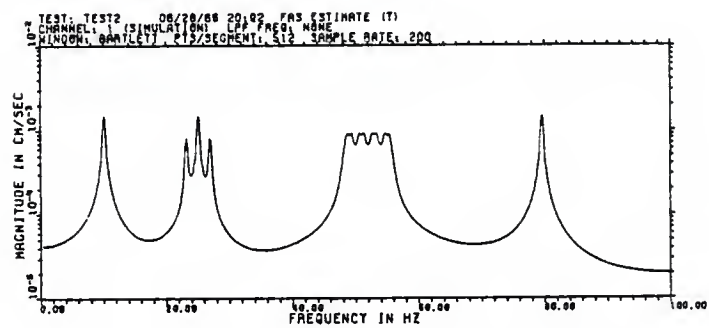


(g) Parzen window.

Figure 5. Welch's method FAS estimate of summation of sines (cont.)

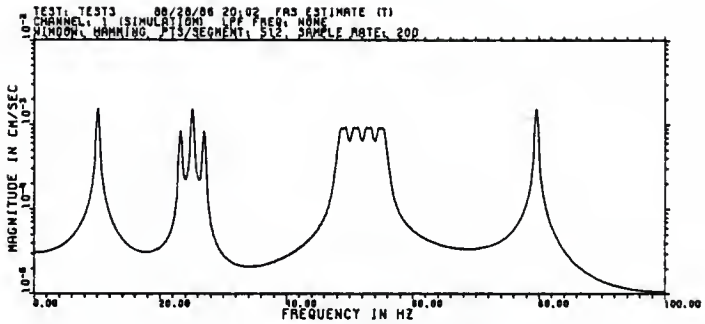


(a) Rectangular window.

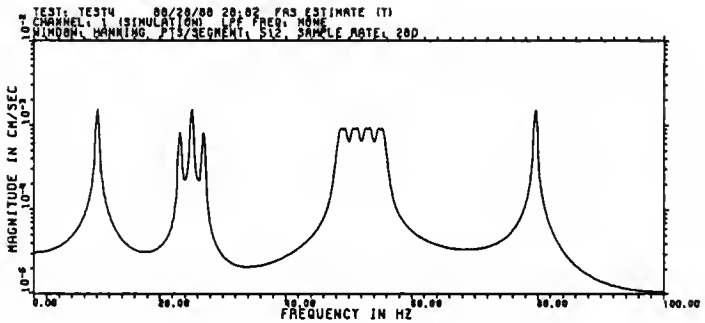


(b) Bartlett window.

Figure 6. Traditional FAS estimate of summation of sines.

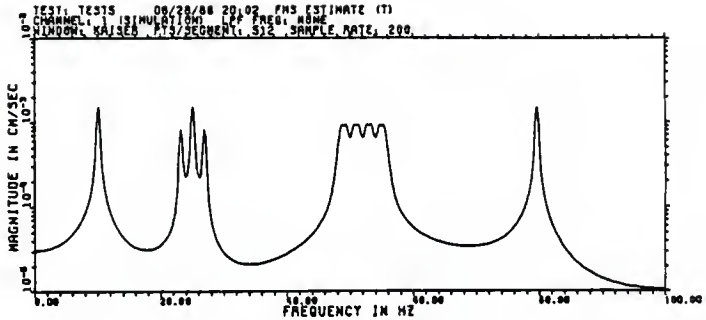


(c) Hamming window.

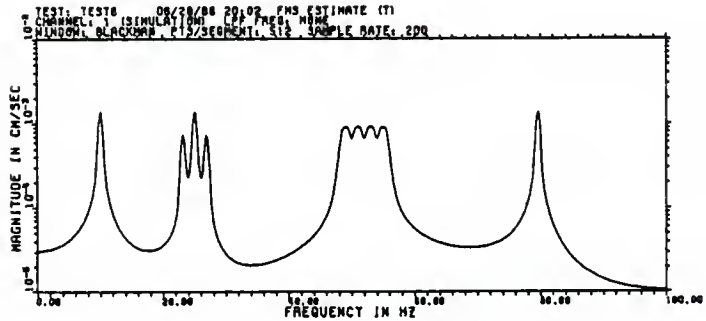


(d) Hanning window.

Figure 6. Traditional FAS estimate of summation of sines (cont.)

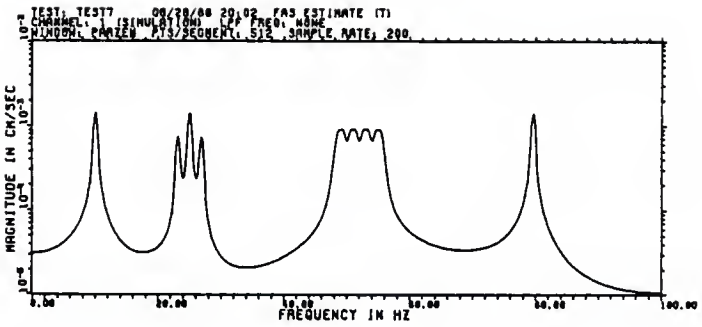


(e) Kaiser window.



(f) Blackman window.

Figure 6. Traditional FAS estimate of summation of sines (cont.)



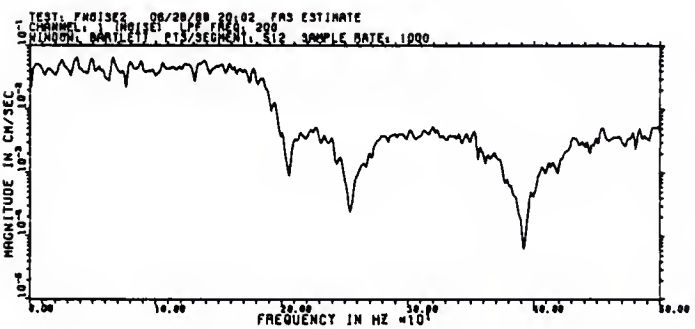
(g) Parzen window.

Figure 6. Traditional FAS estimate of summation of sines (cont.)

FAS estimate computed with Welch's method while figure 6 shows the estimates computed with Beauchamp's method, each using various windows. It can be seen that Welch's method generally reduces the width of the spectral peaks and reduces the magnitude of the valleys between peaks. These plots also show the effects of applying the various windows; the Hanning window in general seems to perform nicely and is the default for the "VAS SPECTRUM" command.

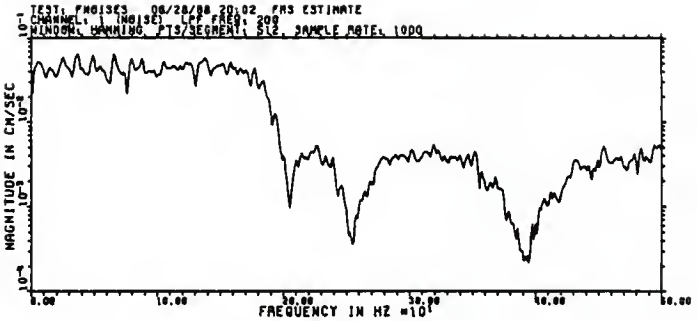
Figures 7 (a-g) and 8 (a-g) show examples of computing 512 point FAS estimates of another synthesized signal. The signal in this case is 2048 samples of a lowpass filtered Gaussian white noise generator. The FAS of unfiltered white noise should be flat over all frequencies. The filter used in this case was a 6 pole Chebyshev type II lowpass filter with a cutoff at 200 Hz and a stopband attenuation of 20 dB. The expected PSD of such a signal would be flat in the passband and have three ripples in the stopband which peak at 20 dB below the passband value. The ripples should peak at 222, 308 and 500 Hz while the PSD should be zero at 206, 252 and 390 Hz. Even more than in the previous example, the power of Welch's method shows through in this example.

(a) Rectangular window.

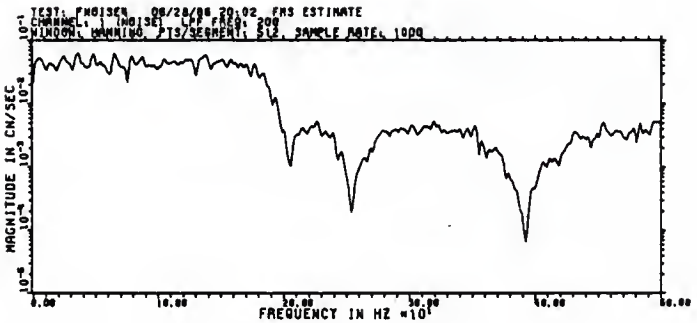


(b) Bartlett window.

Figure 7. Welch's method FAS estimate of filtered noise.

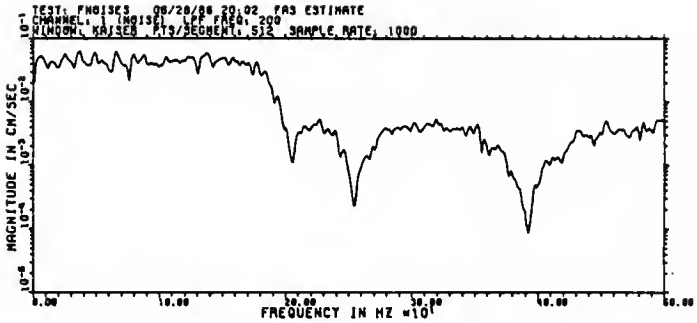


(c) Hamming window.

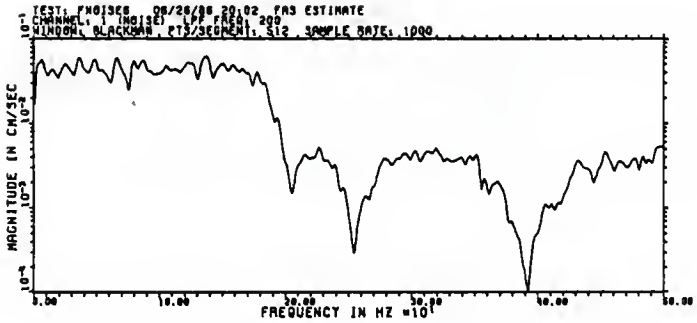


(d) Hanning window.

Figure 7. Welch's method FAS estimate of filtered noise (cont.)

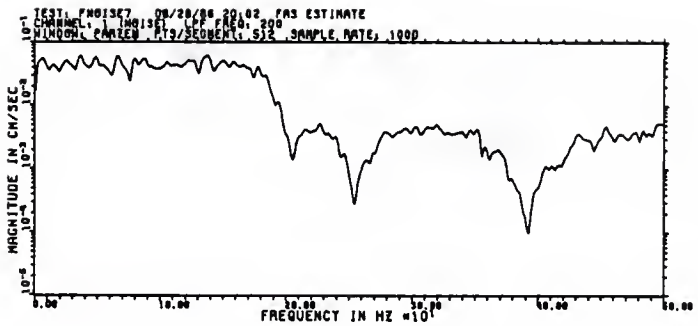


(e) Kaiser window.



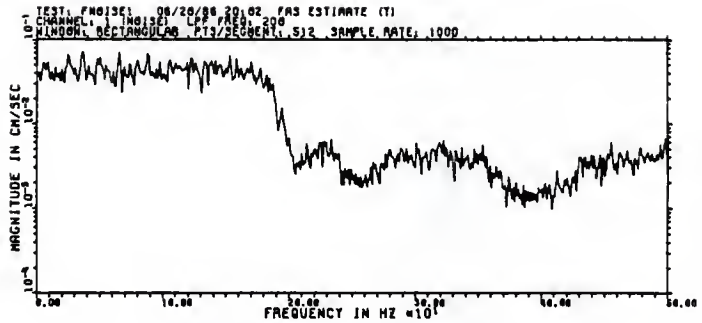
(f) Blackman window.

Figure 7. Welch's method FAS estimate of filtered noise (cont.)

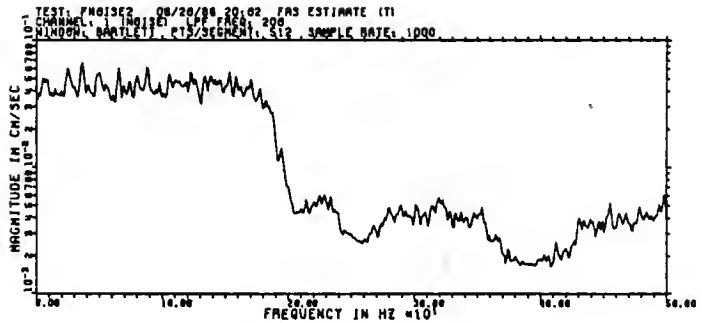


(g) Parzen window.

Figure 7. Welch's method FAS estimate of filtered noise (cont.)

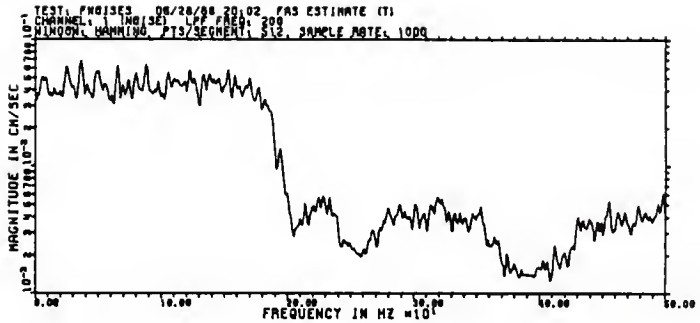


(a) Rectangular window.

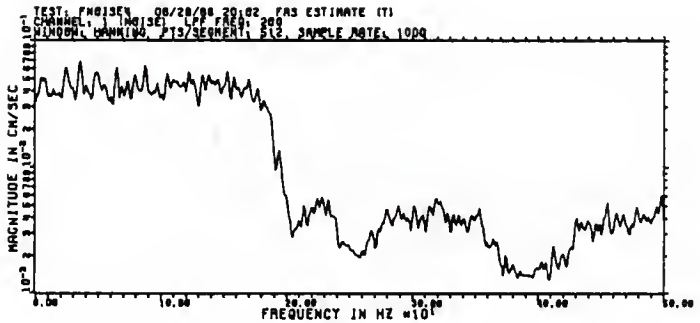


(b) Bartlett window.

Figure 8. Traditional FAS estimate of filtered noise.

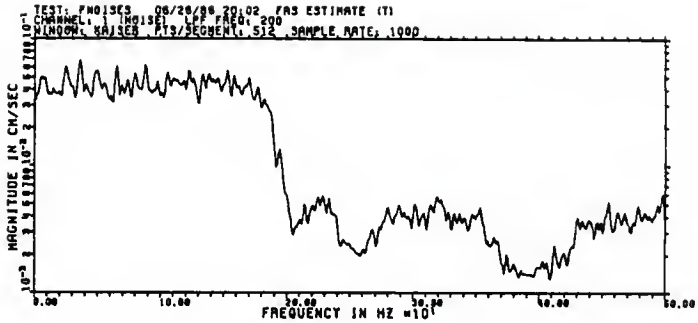


(c) Hamming window.

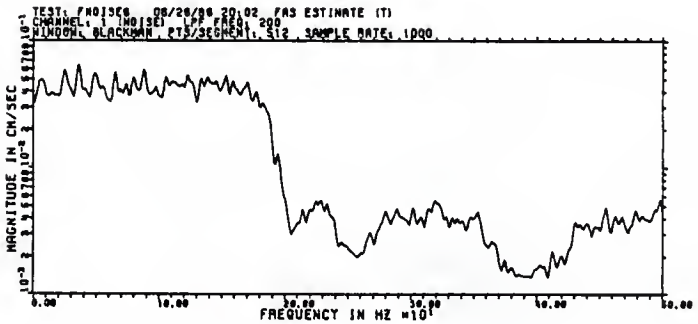


(d) Hanning window.

Figure 8. Traditional FAS estimate of filtered noise (cont.)



(e) Kaiser window.

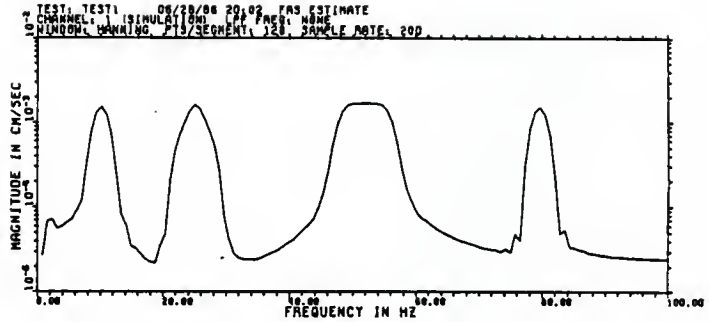


(f) Blackman window.

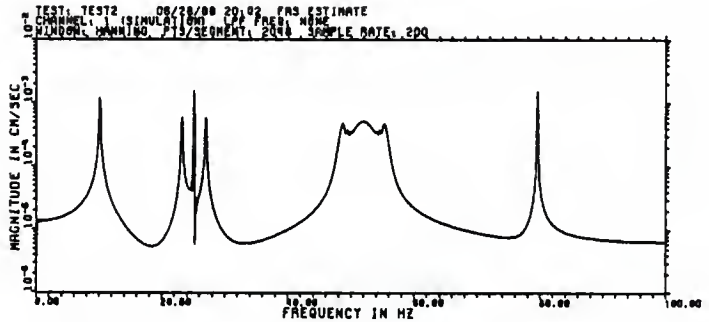
Figure 8. Traditional FAS estimate of filtered noise (cont.)

(g) Parzen window.

Figure 8. Traditional FAS estimate of filtered noise (cont.)



(a) 128 points per segment.



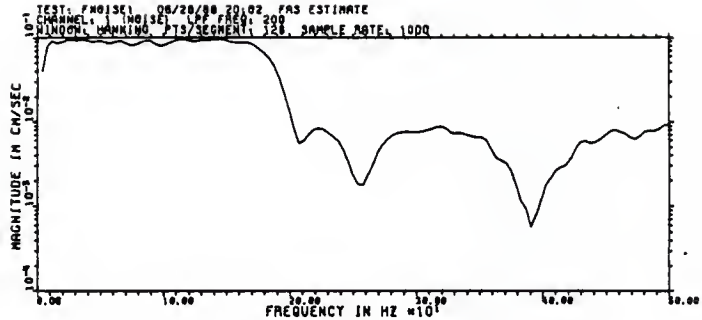
(b) 2048 points per segment.

Figure 9. Welch's method FAS estimate of summation of sines.

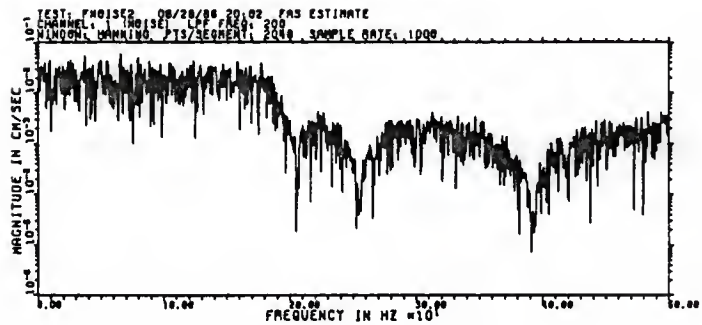
Figure 9a shows the result of computing a 128 point FAS estimate of the multiple sinusoid simulation while figure 9b shows the result of computing a 2048 point estimate. It can be seen that with too few points, inadequate frequency resolution results. Additional points in the estimation adds very little to the result.

Figure 10a shows the result of computing a 128 point FAS estimate of the filtered white noise simulation; figure 10b shows the result cf computing a 2048 point estimate. Again, too few points in the estimate can be seen to decrease the resolution, but at the same time it can be seen that the additional time averages that this allows smooths out the noisy estimate; given a 2048 point signal, six overlapped subsequence estimates are averaged to produce an estimate using Welch's method if 512 point subsequences are used, while 30 subsequence estimates are averaged if 128 point subsequences are used. Figure 10b, a 2048 point estimate, includes no averaging of subsequences and can be seen to be extremely noisy. In general, the length of the sequence being analyzed should be several times the length of the estimate to provide many averages so as to minimize errors in the estimate.

Figure 11 shows the difference between a PSD estimate and a FAS estimate using the filtered white noise. In the

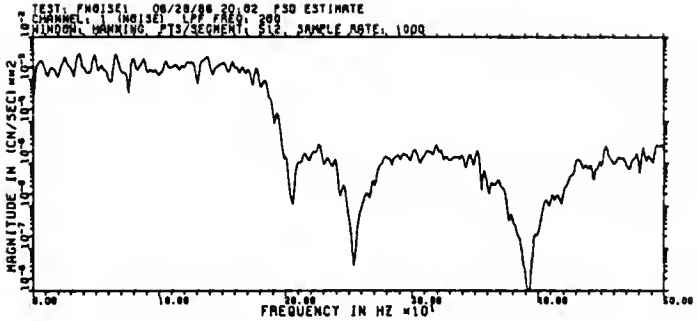


(a) 128 points per segment.

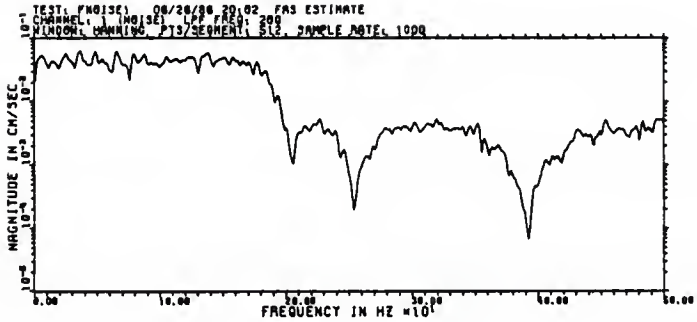


(b) 2048 points per segment.

Figure 10. Welch's method FAS estimate of filtered noise.



(a) PSD estimate.



(b) FAS estimate.

Figure 11. Comparison of FAS and PSD estimates.

FAS estimate, the stopband ripples peak at about 1/10 the passband (20 dB down). In the PSD estimate, the stopband ripple peak at about 1/100 the passband (40 dB). In some applications, a power estimate is more appropriate than an amplitude estimate. The "VAS SPECTRUM" command actually ccmputes a PSD estimate; the "VAS PLOT" command is used to convert to a FAS estimate if desired.

Additional examples of the use of the "VAS SPECTRUM" command appear in subsequent sections.

C. The VAS CORRELATION Command

Auto-correlations and cross-correlations are computed by the "VAS CORRELATION" command. The following commands could be used to compute the auto and cross-correlation estimates of the first two channels of an experiment called "TEST2":

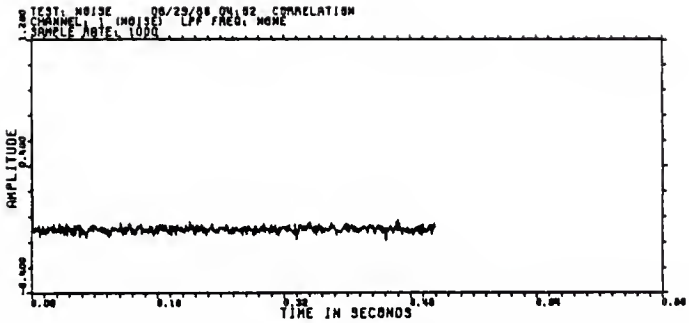
```
VAS CORRELATION TEST2 1 1
VAS CORRELATION TEST2 2 2
VAS CORRELATION TEST2 1 2
VAS CORRELATION TEST2 2 1
```

By default, 512 correlation lags are computed, though the actual number can be changed by specifying the LENGTH option on the command line.

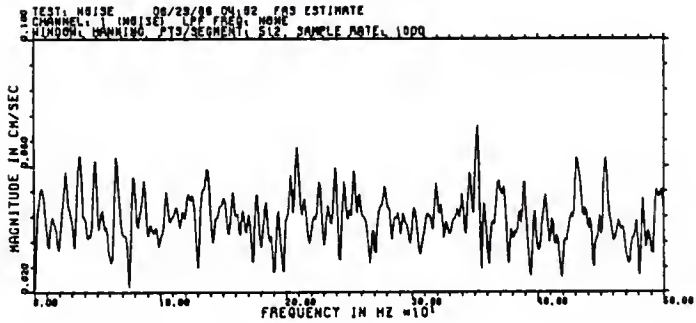
The command uses files of filetype "Dii" and "Djj" as input and produces a file with a filetype of "Cij" as output where "i" and "j" represent channel numbers. For a cross-correlation estimate, "i" and "j" are different while for an auto-correlation estimate they are the same. The resulting output file is written in REAL*4 format.

The brute force computation of correlation estimates is a process which requires time proportional to the square of the number of components in the data. This can be significantly improved by using a method described by Rabiner [14] in which the FFT is used to compute the correlation estimate.

Figure 12 shows the result of computing the auto-correlation and FAS estimate of a unit-variance Gaussian white noise process. As expected, the auto-correlation function is nearly zero except for a peak at a delay value of zero. The value of the auto-correlation estimate at zero is simply the variance of the signal; elsewhere the signal is nearly uncorrelated as would be expected for white noise. The FAS estimate is fairly flat as expected, though it is quite rough; a smoother estimate would be expected from a longer input sequence, which in this case was 2048 points, or from computing fewer points in the estimate, which in this case contains 512 points.



(a) Correlation estimate.



(b) FAS estimate.

Figure 12. Correlation and FAS estimates of white noise.

D. The VAS TRANSFER Command

Estimates of the frequency domain transfer function of two data channels can be computed with the "VAS TRANSFER" command. The following command could be used to compute the transfer function estimate of the first two channels of experiment "SAMPLE4":

```
VAS TRANSFER SAMPLE4 1 2
```

The command uses files of filetype "Sii" and "Sjj" as input and produces a file with a filetype of "Tij" as output where "i" and "j" represent channel numbers. That is, its input is the auto PSD estimate of the data channels and must previously have been computed by using the "VAS SPECTRUM" command. The output is written in COMPLEX*8 format and is simply the result of dividing the first channel PSD estimate by the second channel PSD estimate.

Examples of the use of the "VAS TRANSFER" command appear in following sections.

E. The VAS COHERENCE Command

An estimate of the coherence function of two data channels can be computed with the "VAS COHERENCE" command.

The following command could be used to compute the coherence function estimate of the first two channels of experiment "SAMPLE5":

```
VAS COHERENCE SAMPLE5 1 2
```

The command uses files of filetype "Sii", "Sjj", and "Sij" as input and produces a file with a filetype of "Kij" as output where "i" and "j" represent channel numbers. Its input is the auto PSD estimates of the individual data channels and their cross PSD estimate; these must previously have been computed using the "VAS SPECTRUM" command. The output is written in REAL*4 format.

Examples of the use of the "VAS COHERENCE" command appear in following sections.

F. The VAS PLOT Command

The results of computing PSD, correlation, transfer function, and coherence function estimates as well as data can be plotted by using the "VAS PLOT" command. Options on the command allow the plots to be made either interactively on a Tektronix 4010 compatible terminal or on the mainframe Calcomp drum plotter. Other options

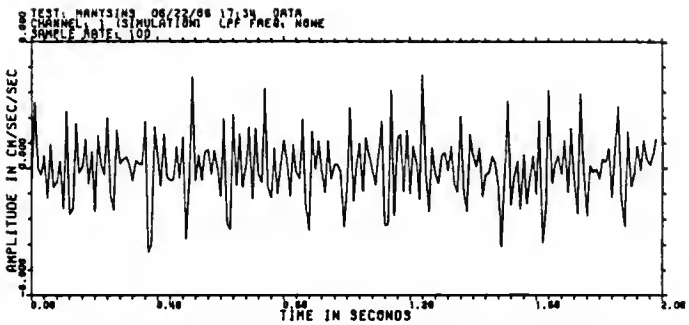
control such things as the number of points plotted, the starting point number, whether the axes are to be scaled linearly or logarithmically, etc. The Vibration Analysis System User's Guide in Appendix A contains a complete description of the options available.

G. The VAS PRINT Command

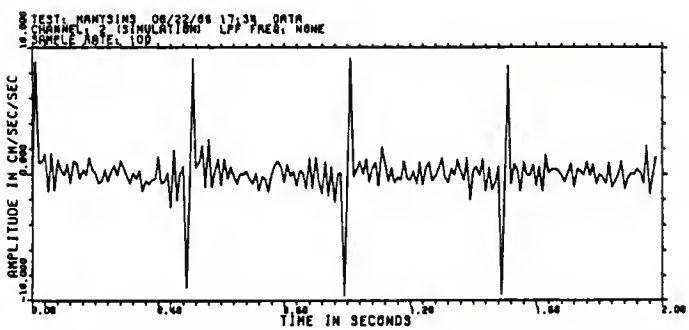
While the "VAS PLOT" command is used to graphically display the results of computations, the "VAS PRINT" command will print the results in tabular form. Again, refer to the Vibration Analysis System User's Guide for a complete description.

H. Example--Multiple Sinusoids in White Noise

The plots of figure 13 (a-j) show the results of computing several estimates from another simulated sequence of data. Channel 1 of MANYSINS contains twenty distinct sinusoids of varying amplitude, all with zero initial phase shift, added to unit-variance Gaussian white noise. Channel 2 contains twenty similar sinusoids with varying initial phase shifts and slightly greater amplitudes added to unit-variance Gaussian white noise. The frequency of the sinusoids and their initial phase shift is shown in the following table:

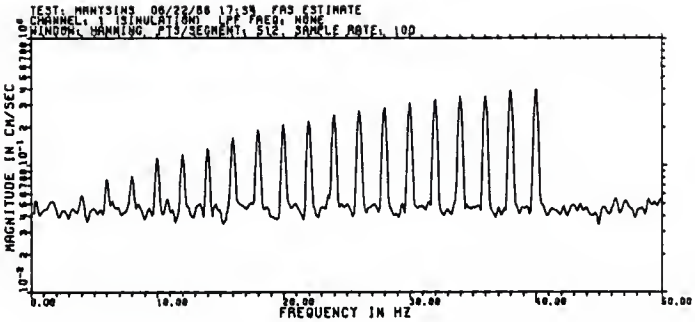


(a) Channel 1 data.

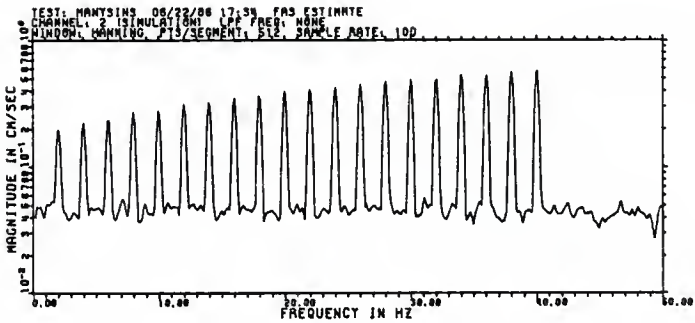


(b) Channel 2 data.

Figure 13. Analysis of summation of many sines on noise.

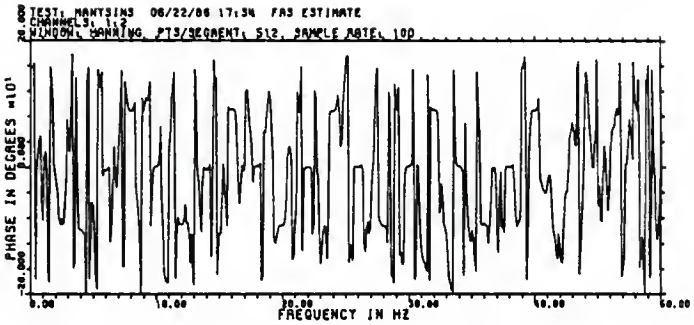


(c) Channel 1 FAS estimate.

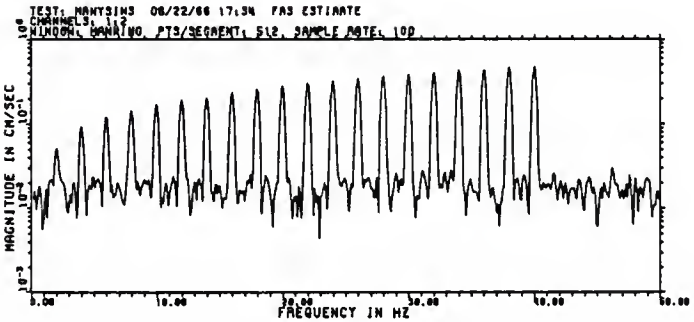


(d) Channel 2 FAS estimate.

Figure 13. Analysis of summation of many sines on noise (cont.)

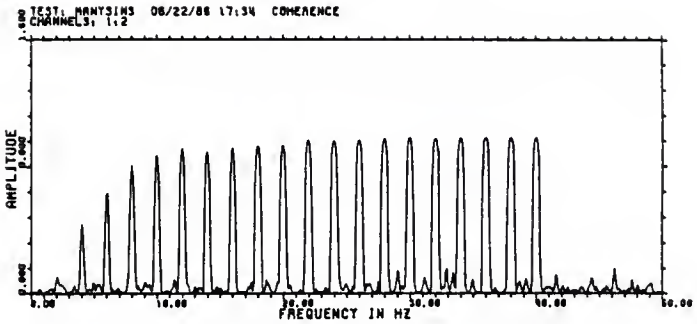


(e) Cross-FAS estimate phase.

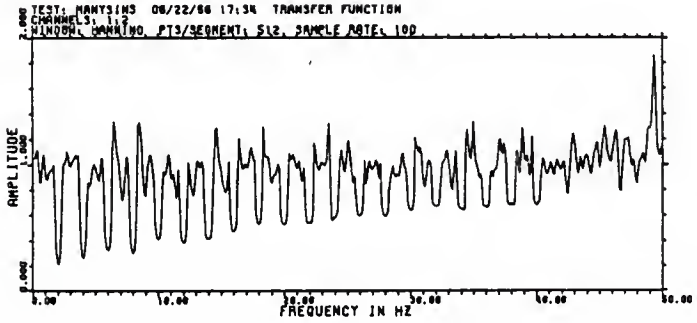


(f) Cross-FAS estimate magnitude.

Figure 13. Analysis of summation of many sines on noise (cont.)

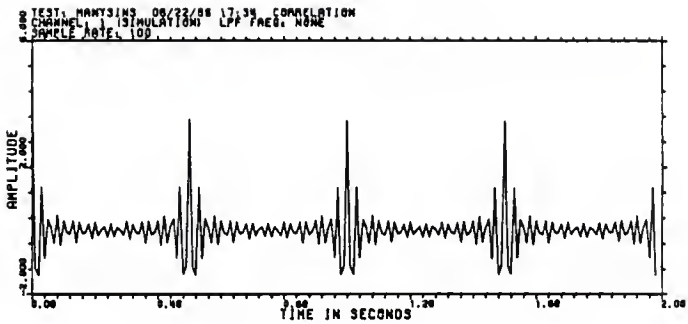


(g) Coherence function estimate.

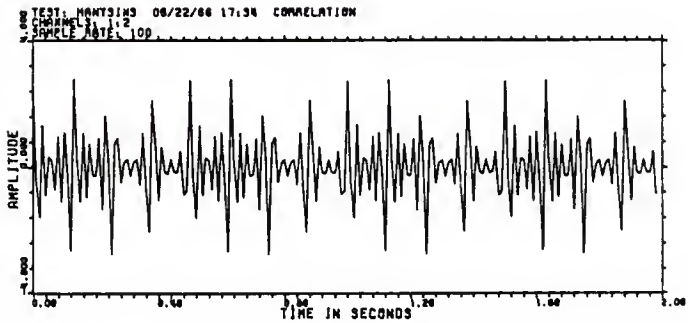


(h) Transfer function estimate.

Figure 13. Analysis of summation of many sines on noise (cont.)



(i) Channel 1 auto-correlation estimate.



(j) Cross-correlation estimate.

Figure 13. Analysis of summation of many sines on noise (cont.)

Channel 1			Channel 2		
Freq	Amp	Phase	Freq	Amp	Phase
2	.05	0	2	.50	0
4	.10	0	4	.55	90
6	.15	0	6	.60	0
8	.20	0	8	.65	-90
10	.25	0	10	.70	0
12	.30	0	12	.75	90
14	.35	0	14	.80	0
16	.40	0	16	.85	-90
18	.45	0	18	.90	0
20	.50	0	20	.95	90
22	.55	0	22	1.00	0
24	.60	0	24	1.05	-90
26	.65	0	26	1.10	0
28	.70	0	28	1.15	90
30	.75	0	30	1.20	0
32	.80	0	32	1.25	-90
34	.85	0	34	1.30	0
36	.90	0	36	1.35	90
38	.95	0	38	1.40	0
40	1.00	0	40	1.45	-90

The simulated signal was sampled at 100 Hz for 8192 samples. A small section of each of the data channels is shown in figure 13a and 13b. These plots were produced using the following VAS command:

```
VAS PLOT (CALCOMP SCALE .5 FOR 199) MANYSINS DATA 2 2
        MANYSINS DATA 1 1
```

The plots of figure 13c and 13d were produced from the following VAS commands:

```
VAS SPEC MANYSINS 1 1
VAS SPEC MANYSINS 2 2
VAS PLOT (CALCOMP SCALE .5 FROM 2 LOGY) MANYSINS SPEC
        2 2 MANYSINS SPEC 1 1
```

The first point of the FAS estimate was left off the plot since it represents the DC coefficient of the spectrum and is nearly zero, thus not fitting well on a logarithmic scale.

It can be seen from figure 13c that the small amplitude sinusoids are nearly lost in the white noise background. The amplitude of the first few sines are .05, .10, .15, and .20 and represent very unfavorable signal-to-noise ratios (SNR's).

Figures 13e and 13f were produced from the following commands:

```
VAS SPEC MANYSINS 1 2
VAS PLOT (CALCOMP SCALE .5 FROM 2) MANYSINS SPEC 1 2
         (MAGNITUDE LOGY) MANYSINS SPEC 1 2 (PHASE)
```

As the cross-spectrum of a signal contains phase information, the "VAS PLOT" command can be told to plot either the magnitude or the phase.

The phase plot shows essentially what was expected from the cross-spectrum. The phase of every other peak is zero while the phase of the remaining peaks alternates from 90 degrees to -90 degrees. Between the peaks, the phase is essentially random.

The plots of figures 13g and 13h were produced from the following commands:

```
VAS COHER MANYSINS 1 2
VAS TRAN MANYSINS 1 2
VAS PLOT (CALCOMP SCALE .5 FROM 2) MANYSINS TRAN 1 2
         MANYSINS COHER 1 2
```

The coherence at frequencies corresponding to the spectral peaks is seen to vary from .15 in the worst case

to 1.0 where the spectral peaks were well above the noise level. Between the peaks, the coherence function is nearly zero which is as expected since the noise background is uncorrelated.

The transfer function at frequencies corresponding to spectral peaks should be expected to range from 0.1 (.05/.50) for the first peak to .69 (1.00/1.45) at the last peak. At frequencies between the peaks, the transfer function should be about 1.0 corresponding to a constant amplitude background noise. Figure 13h shows that this is indeed about what was found. While the transfer function at the first spectral peak is somewhat inaccurate, this is to be expected given that the coherence function at that frequency indicates that the noise is dominating the estimate.

The final figures, 13i and 13j, show the results of computing auto-correlation and cross-correlation estimates with the following commands:

```
VAS CORR MANYSINS 1 1
VAS CORR MANYSINS 1 2
VAS PLOT (CALC SCA .5 FOR 199) MANYSINS CORR 1 2
         MANYSINS CORR 1 1
```

The auto-correlation shows the sinusoidal nature of channel 1 much better than the plot of the data itself. In addition, the value of the auto-correlation at zero is

4.55 while the value of subsequent peaks is about 3.5. This indicates as expected that there is unit-variance noise in the signal.

The value of the cross-correlation plot in this example is questionable. It appears to provide very little additional information.

I. Example--Analysis of a Structure

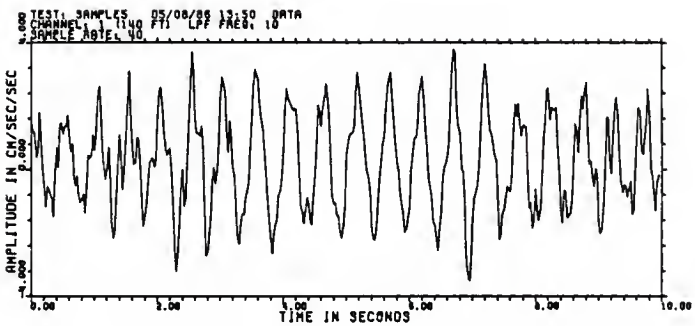
While the purpose of this paper is not to describe the analysis of structural dynamics, it would be incomplete without at least including a demonstration of the system in use examining a real structure.

For this study, the U.S. Grain Marketing Research storage facility was used. This structure is a tall, essentially rectangular building approximately 150 feet high. Accelerometers were placed at four different landings at heights of approximately 140 feet, 110 feet, 70 feet, and 30 feet and were feed into the Kinemetrics SC-1 signal conditioner as channels 1 through 4. Due to what appeared to be an intermittent failure in the channel 3 cable, only three useful data channels were collected. The SC-1 provides signal amplification, variable in 6 dB steps from about 100,000 to 200, and provides 60 dB per decade lowpass filtering variable from 1 Hz to 100 Hz.

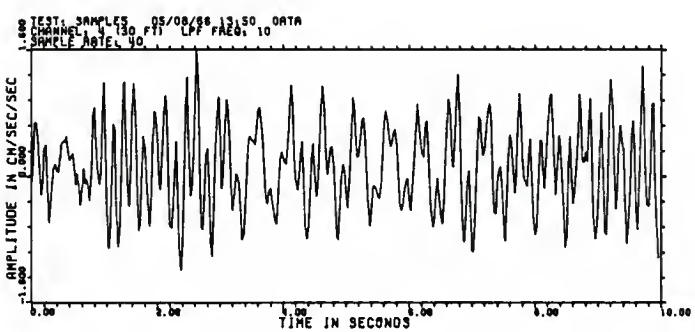
An initial on-site FAS analysis of a short sequence of data collected at 500 sps/channel with the lowpass filters removed indicated that the major spectral peaks were confined to below about 40 Hz. The sequence of data named SAMPLE4 was then collected with the lowpass filter set at 50 Hz and the sample rate set at 200 Hz; approximately 25,000 samples per channel were collected.

An on-site analysis of a small piece of SAMPLE4 indicated that the first vibration mode was at approximately 2 Hz and a second major mode was located at about 6 Hz. The third mode, much smaller than the first two, was just below 10 Hz. The sequence of data named SAMPLE5 was collected with the lowpass filter set at 10 Hz and the sample rate set at 40 Hz; approximately 14,500 samples per channel were collected. 512 point estimates of the spectrum, coherence, and transfer function where calculated and the first 255 points after the DC component are plotted; the values of the estimates beyond the lowpass filter setting are not of interest.

Figures 14a and 14b show the first 400 samples of the first and the fourth channels of SAMPLE5. As expected, channel 1 (from the sensor located near the top of the building) shows considerably greater acceleration than channel 4. The first 4000 samples of the same data is

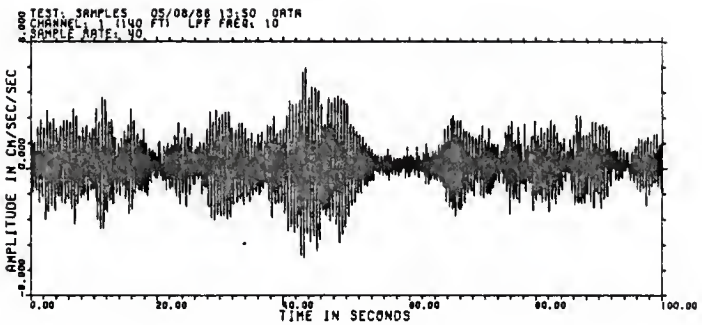


(a) 400 data points from channel 1.

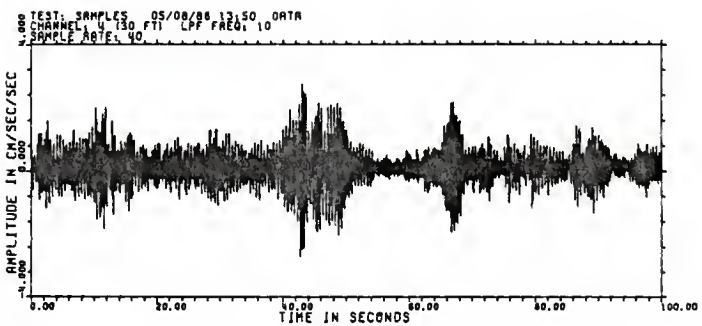


(b) 400 data points from channel 4.

Figure 14. Analysis of a structure at low frequencies.

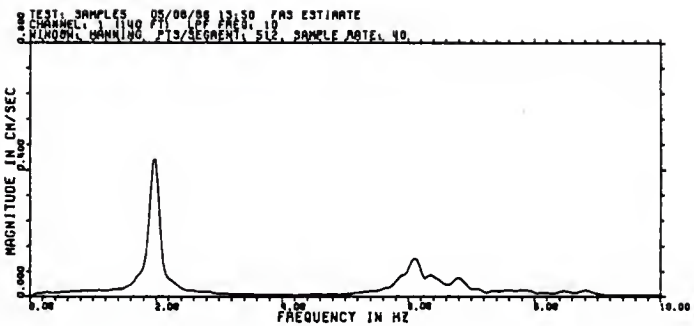


(c) 4000 data points from channel 1.

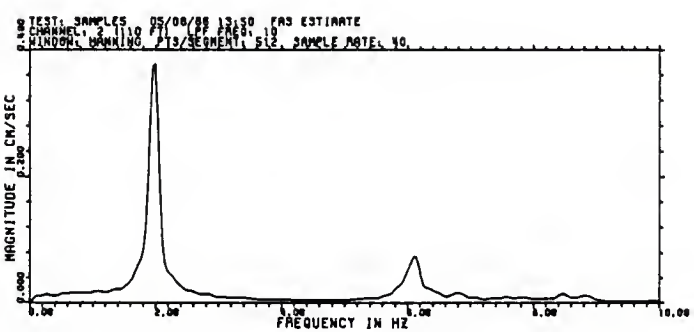


(d) 4000 data points from channel 4.

Figure 14. Analysis of a structure at low frequencies (cont.)

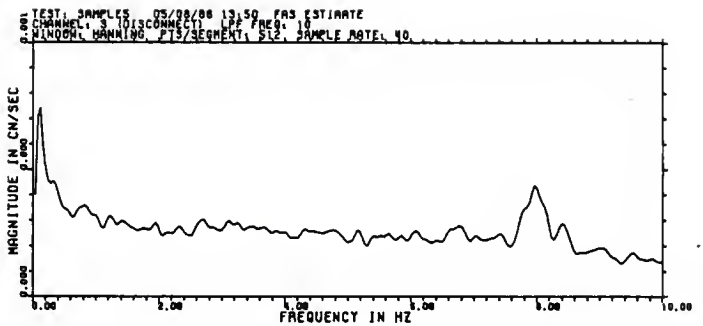


(e) Channel 1 FAS estimate.

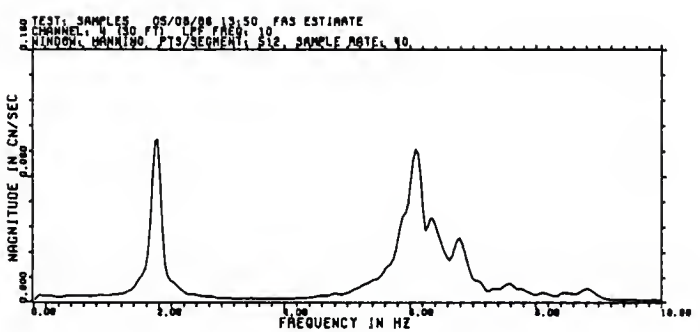


(f) Channel 2 FAS estimate.

Figure 14. Analysis of a structure at low frequencies (cont.)

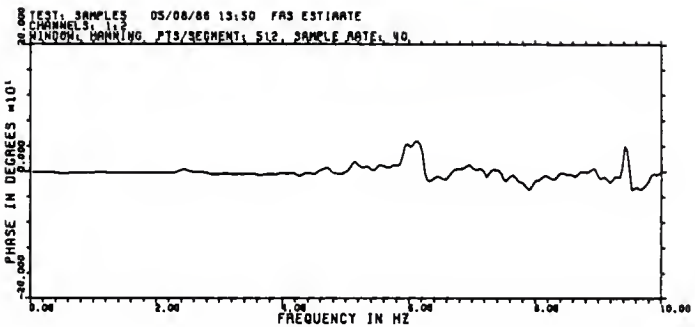


(g) Channel 3 FAS estimate.

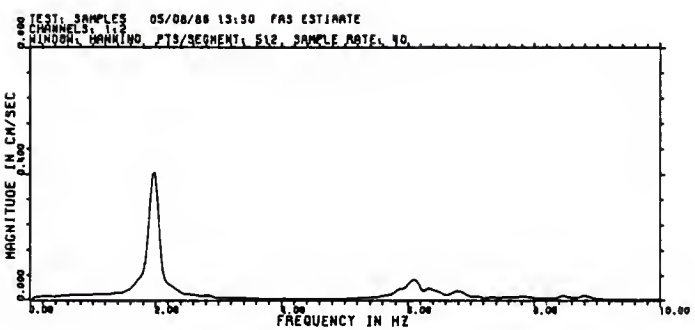


(h) Channel 4 FAS estimate.

Figure 14. Analysis of a structure at low frequencies (cont.)

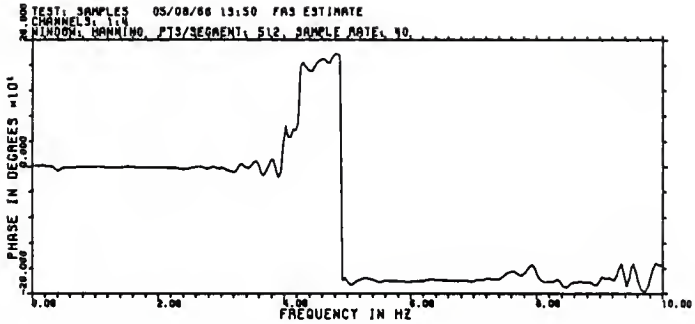


(i) Channel 1:2 FAS estimate phase.

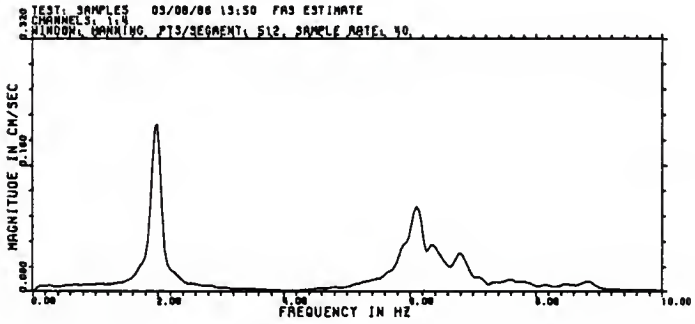


(j) Channel 1:2 FAS estimate magnitude.

Figure 14. Analysis of a structure at low frequencies (cont.)

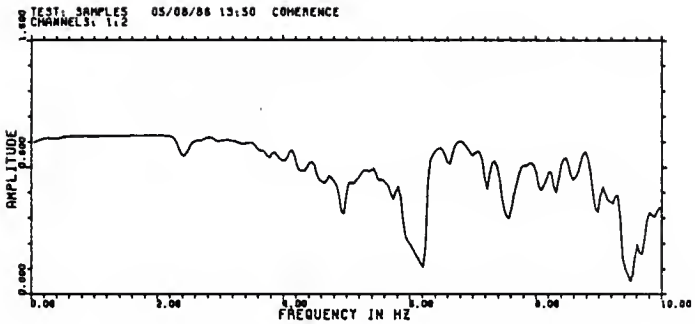


(k) Channel 1:4 FAS estimate phase.

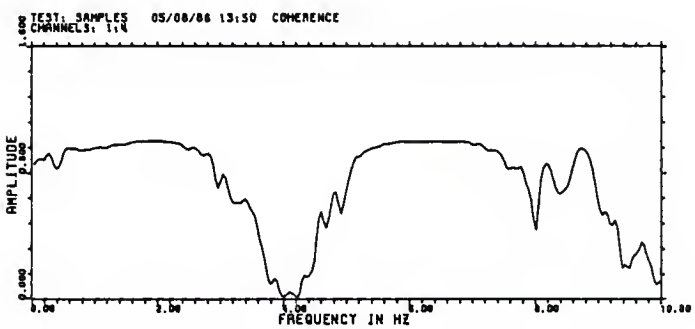


(l) Channel 1:4 FAS estimate magnitude.

Figure 14. Analysis of a structure at low frequencies (cont.)

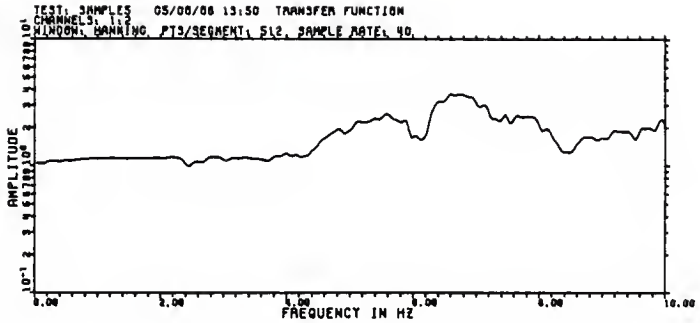


(m) Channel 1:2 coherence function estimate.

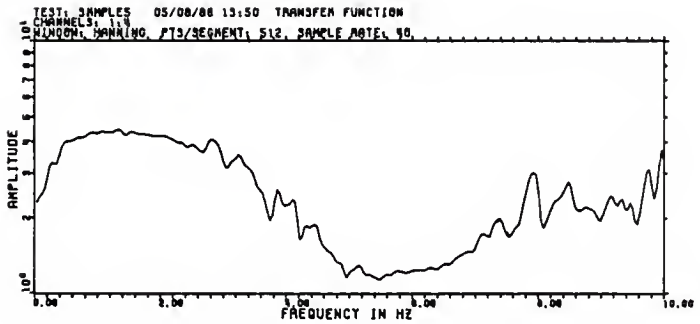


(n) Channel 1:4 coherence function estimate.

Figure 14. Analysis of a structure at low frequencies (cont.)

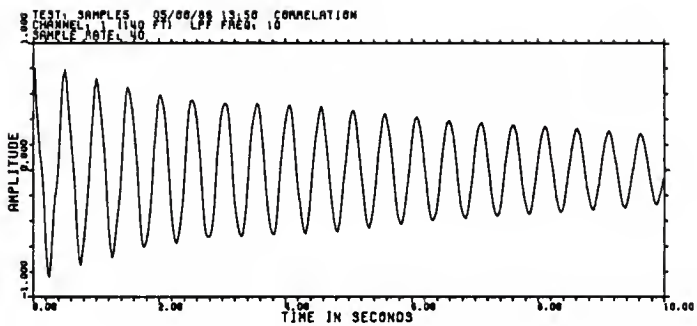


(o) Channel 1:2 transfer function estimate.

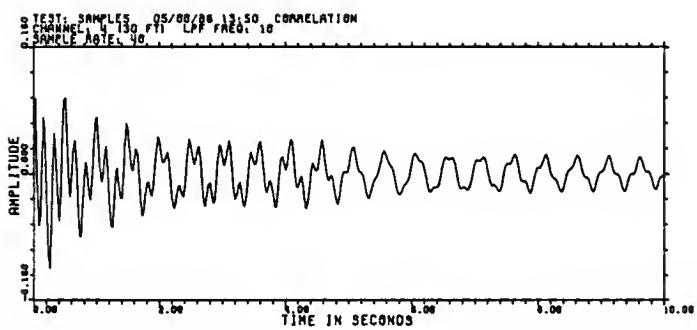


(p) Channel 1:4 transfer function estimate.

Figure 14. Analysis of a structure at low frequencies (cont.)



(q) Channel 1 auto-correlation estimate.



(r) Channel 4 auto-correlation estimate.

Figure 14. Analysis of a structure at low frequencies (cont.)

shown in figures 14c and 14d which gives a better indication of the variability of the vibrations. The areas where the vibrations are large correspond to periods of high wind while the areas with weak vibrations correspond to periods of little wind.

The auto FAS estimate of the four data channels of SAMPLE5 are shown in figures 14 (e-h). Remembering that channel 3 was not connected, they show the first two vibration modes quite clearly, and signs of the third can be seen near 9 Hz. It is interesting to note that the magnitude of the first mode is much larger near the top of the building than near the bottom, while the magnitude of the second mode is almost constant.

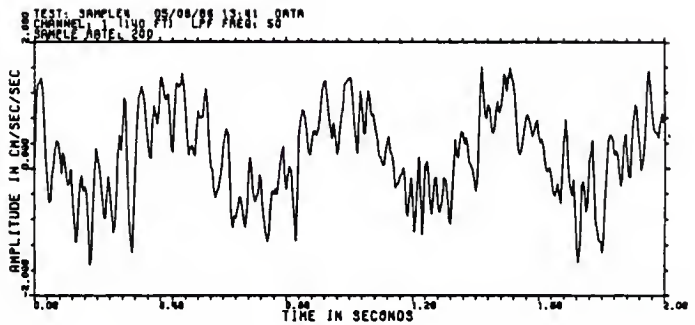
Cross FAS estimates of SAMPLE5 are shown in figures 14 (i-l). These show very clearly the 180 degree phase shift of the second mode which should occur about two-thirds of the way up the structure. In the first mode, the entire building sways back and forth with a single node at the bottom. For the second mode, in addition to the node at the bottom, a second node is found about two-thirds of the way toward the top. Channel 1 is apparently above the second node while channels 2 and 4 are below it.

Coherence plots are shown in figures 14m and 14n. As might be expected, the coherence is close to unity at the frequency of the first mode. Near 6 Hz, however, the coherence function between channels 1 and 2 actually dips; no explanation for this has been posited (and this same dip is seen in SAMPLE4 coherence plots). The coherence function between channels 1 and 4 is much more reasonable.

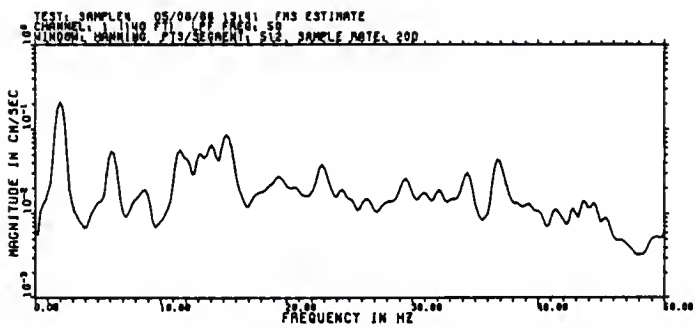
The transfer functions shown in figures 14o and 14p are self-explanatory.

Auto-correlation estimates for channels 1 and 4 are shown in figures 14q and 14r. Channel 1 is obviously dominated by the 2 Hz primary mode and contains relatively little noise. Channel 4 is noisier relative to the signal strengths, and obviously contains more than the single primary mode.

Figures 15 (a-l) contain plots of the results of analyzing SAMPLE4. Since SAMPLE4 covers frequencies up to 50 Hz, it is much busier and many other vibration modes can be seen. No attempt is made here to explain these plots in detail. Figures 16 (a-d) show in printed form the channel 1:4 cross FAS estimate; for particularly busy plots, the printed output is often useful for matching frequency and phase components.

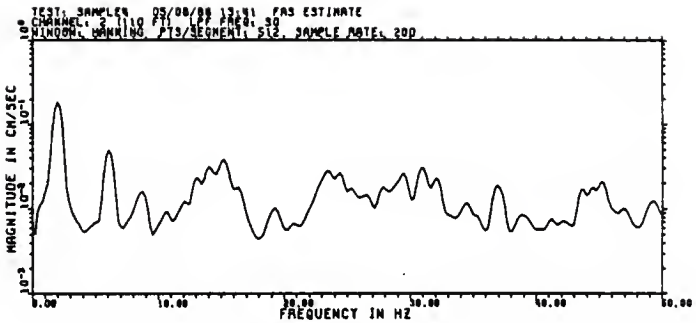


(a) 400 data points from channel 1.

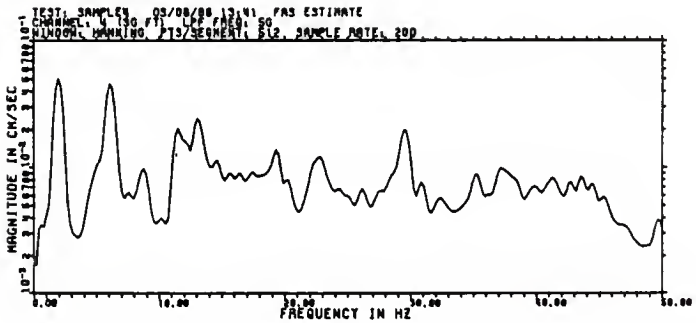


(b) Channel 1 FAS estimate.

Figure 15. Analysis of a structure at high frequencies.

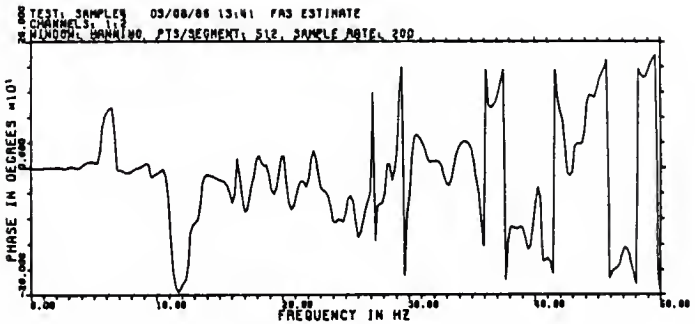


(c) Channel 2 FAS estimate.

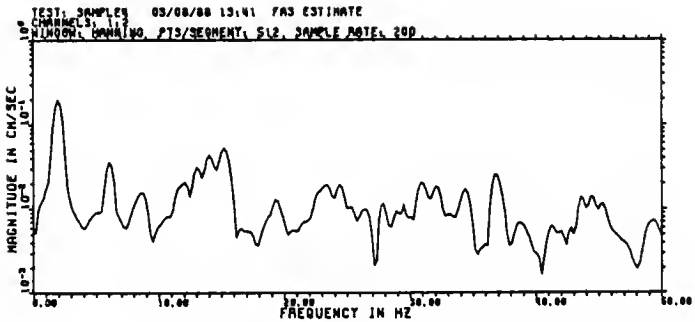


(d) Channel 4 FAS estimate.

Figure 15. Analysis of a structure at high frequencies (cont.)

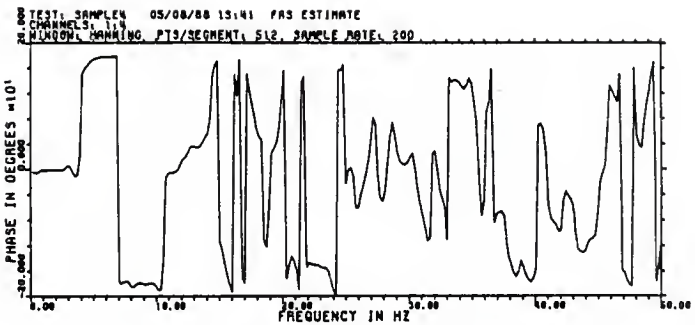


(e) Channel 1:2 FAS estimate phase.

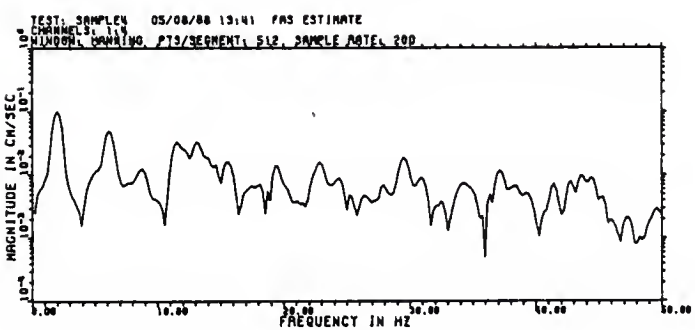


(f) Channel 1:2 FAS estimate magnitude.

Figure 15. Analysis of a structure at high frequencies (cont.)

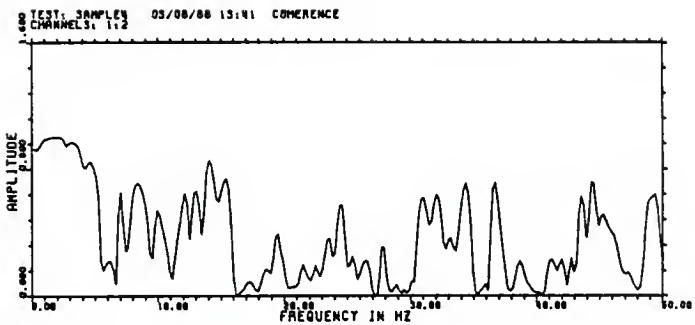


(g) Channel 1:4 FAS estimate phase.

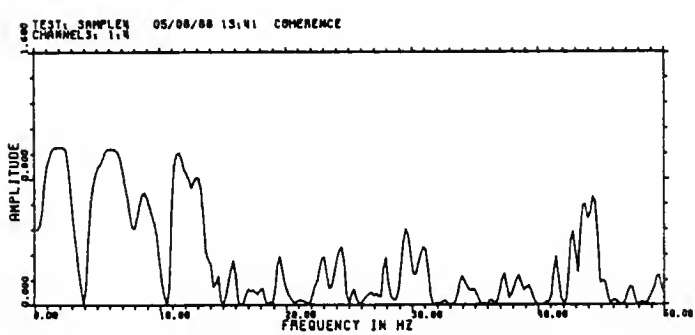


(h) Channel 1:4 FAS estimate magnitude.

Figure 15. Analysis of a structure at high frequencies (cont.)

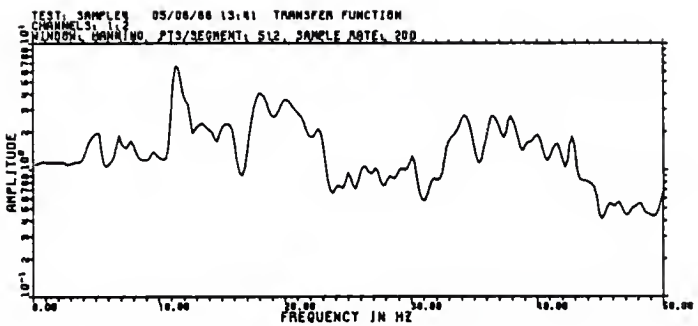


(i) Channel 1:2 coherence function estimate.

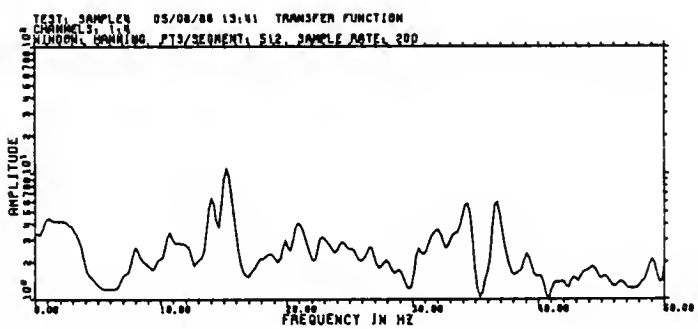


(j) Channel 1:4 coherence function estimate.

Figure 15. Analysis of a structure at high frequencies (cont.)



(k) Channel 1:2 transfer function estimate.



(l) Channel 1:4 transfer function estimate.

Figure 15. Analysis of a structure at high frequencies (cont.)

TEST: SAMPLE4 05/06/80 13:41 FAS ESTIMATE
CHANNELS: 1:4
WINDOW: HANNING PTS/SEGMENT: 512 SAMPLE RATE: 200

FREQUENCY	MAGNITUDE	FREQUENCY	MAGNITUDE	FREQUENCY	MAGNITUDE
0.19531	0.2536E-02	9.96094	0.3970E-02	19.72656	0.1063E-01
0.39063	0.4907E-02	10.15625	0.3601E-02	19.92188	0.7791E-02
0.58594	0.5808E-02	10.35156	0.2965E-02	20.11719	0.6781E-02
0.78125	0.6445E-02	10.54688	0.1628E-02	20.31250	0.6009E-02
0.97656	0.8340E-02	10.74219	0.4022E-02	20.50781	0.4668E-02
1.17188	0.1018E-01	10.93750	0.1048E-01	20.70313	0.3666E-02
1.36719	0.2043E-01	11.13281	0.2061E-01	20.89844	0.3769E-02
1.56250	0.5025E-01	11.32813	0.3012E-01	21.09375	0.3331E-02
1.75781	0.8507E-01	11.52344	0.3357E-01	21.28906	0.3403E-02
1.95313	0.1028E+00	11.71875	0.3025E-01	21.48438	0.3972E-02
2.14844	0.9113E-01	11.91406	0.2627E-01	21.67969	0.3143E-02
2.34375	0.5826E-01	12.10938	0.2520E-01	21.87500	0.3688E-02
2.53906	0.2551E-01	12.30469	0.2265E-01	22.07031	0.3505E-02
2.73438	0.9549E-02	12.50000	0.1832E-01	22.26563	0.3012E-02
2.92969	0.6318E-02	12.69531	0.2021E-01	22.46094	0.1128E-01
3.12500	0.4746E-02	12.89063	0.2829E-01	22.65625	0.1404E-01
3.32031	0.4044E-02	13.08594	0.3347E-01	22.85156	0.1391E-01
3.51563	0.3298E-02	13.28125	0.3166E-01	23.04688	0.1477E-01
3.71094	0.2694E-02	13.47656	0.2523E-01	23.24219	0.1165E-01
3.90625	0.1564E-02	13.67188	0.2006E-01	23.43750	0.7759E-02
4.10156	0.3013E-02	13.86719	0.1914E-01	23.63281	0.6917E-02
4.29688	0.5438E-02	14.06250	0.1631E-01	23.82813	0.6911E-02
4.49219	0.7324E-02	14.25781	0.1415E-01	24.02344	0.7361E-02
4.68750	0.8823E-02	14.45313	0.1357E-01	24.21875	0.8432E-02
4.88281	0.1037E-01	14.64844	0.1496E-01	24.41406	0.8902E-02
5.07813	0.1167E-01	14.84375	0.9804E-02	24.60938	0.7752E-02
5.27344	0.1223E-01	15.03906	0.7404E-02	24.80469	0.6878E-02
5.46875	0.1451E-01	15.23438	0.1262E-01	25.00000	0.2703E-02
5.66406	0.2506E-01	15.42969	0.1600E-01	25.19531	0.4773E-02
5.85938	0.4043E-01	15.62500	0.1627E-01	25.39063	0.4503E-02
6.05469	0.5014E-01	15.82031	0.1405E-01	25.58594	0.3214E-02
6.25000	0.4735E-01	16.01563	0.9916E-02	25.78125	0.2289E-02
6.44531	0.3419E-01	16.21094	0.5144E-02	25.97656	0.3117E-02
6.64063	0.1949E-01	16.40625	0.2421E-02	26.17188	0.4272E-02
6.83594	0.1080E-01	16.60156	0.3208E-02	26.36719	0.4720E-02
7.03125	0.7584E-02	16.79688	0.4879E-02	26.56250	0.4526E-02
7.22656	0.6684E-02	16.99219	0.5582E-02	26.75781	0.4171E-02
7.42188	0.7067E-02	17.18750	0.5956E-02	26.95313	0.3633E-02
7.61719	0.7419E-02	17.38281	0.6041E-02	27.14844	0.3383E-02
7.81250	0.7390E-02	17.57813	0.6017E-02	27.34375	0.3494E-02
8.00781	0.7576E-02	17.77344	0.6298E-02	27.53906	0.4243E-02
8.20313	0.8608E-02	17.96875	0.6794E-02	27.73438	0.6303E-02
8.39844	0.1031E-01	18.16406	0.7175E-02	27.92969	0.8946E-02
8.59375	0.1197E-01	18.35938	0.6837E-02	28.12500	0.9594E-02
8.78906	0.1251E-01	18.55469	0.5437E-02	28.32031	0.9087E-02
8.98438	0.1110E-01	18.75000	0.5403E-02	28.51563	0.8770E-02
9.17969	0.8130E-02	18.94531	0.5911E-02	28.71094	0.9137E-02
9.37500	0.5365E-02	19.14063	0.9807E-02	28.90625	0.9725E-02
9.57031	0.4325E-02	19.33594	0.1391E-01	29.10156	0.1145E-01
9.76563	0.4170E-02	19.53125	0.1390E-01	29.29688	0.1029E-01

(a) Channel 1:4 FAS estimate magnitude.

Figure 16. Output of VAS PRINT command.

```
TEST: SAMPLE4   05/08/80 13:41  FAS ESTIMATE
CHANNELS: 1:4
WINDOW: HANNING  PTS/SEGMENT: 512  SAMPLE RATE: 200
```

FREQUENCY	MAGNITUDE	FREQUENCY	MAGNITUDE	FREQUENCY	MAGNITUDE
29.49219	0.1885E-01	39.25781	0.5188E-02	49.02344	0.1837E-02
29.68750	0.1744E-01	39.45313	0.4850E-02	49.21875	0.2172E-02
29.88281	0.1314E-01	39.64844	0.3919E-02	49.41406	0.2744E-02
30.07813	0.8734E-02	39.84375	0.2823E-02	49.60938	0.2744E-02
30.27344	0.6599E-02	40.03906	0.1639E-02	49.80469	0.2578E-02
30.46875	0.6715E-02	40.23438	0.1071E-02	50.00000	0.2349E-02
30.66406	0.8012E-02	40.42969	0.1934E-02		
30.85938	0.8952E-02	40.62500	0.2545E-02		
31.05469	0.8553E-02	40.82031	0.2733E-02		
31.25000	0.6580E-02	41.01563	0.3995E-02		
31.44531	0.3998E-02	41.21094	0.6432E-02		
31.64063	0.1601E-02	41.40625	0.7107E-02		
31.83594	0.2675E-02	41.60156	0.5732E-02		
32.03125	0.3240E-02	41.79688	0.3556E-02		
32.22656	0.3197E-02	41.99219	0.2319E-02		
32.42188	0.3778E-02	42.18750	0.2657E-02		
32.61719	0.3660E-02	42.38281	0.4727E-02		
32.81250	0.2653E-02	42.57813	0.7007E-02		
33.00781	0.1313E-02	42.77344	0.7785E-02		
33.20313	0.1980E-02	42.96875	0.6410E-02		
33.39844	0.3004E-02	43.16406	0.5159E-02		
33.59375	0.4298E-02	43.35938	0.7025E-02		
33.78906	0.5661E-02	43.55469	0.9065E-02		
33.98438	0.6713E-02	43.75000	0.9408E-02		
34.17969	0.7295E-02	43.94531	0.7712E-02		
34.37500	0.7304E-02	44.14063	0.7661E-02		
34.57031	0.6784E-02	44.33594	0.8969E-02		
34.76563	0.6291E-02	44.53125	0.8716E-02		
34.96094	0.5707E-02	44.72656	0.8493E-02		
35.15625	0.4797E-02	44.92188	0.8031E-02		
35.35156	0.3345E-02	45.11719	0.8305E-02		
35.54688	0.2043E-02	45.31250	0.8495E-02		
35.74219	0.2245E-02	45.50781	0.8431E-02		
35.93750	0.4983E-03	45.70313	0.1722E-02		
36.13281	0.3606E-02	45.89844	0.1949E-02		
36.32813	0.4881E-02	46.09375	0.1821E-02		
36.52344	0.3636E-02	46.28906	0.1525E-02		
36.71875	0.6943E-02	46.48438	0.1155E-02		
36.91406	0.1050E-01	46.67969	0.8575E-03		
37.10938	0.1169E-01	46.87500	0.1553E-02		
37.30469	0.1068E-01	47.07031	0.2026E-02		
37.50000	0.8159E-02	47.26563	0.2107E-02		
37.69531	0.5820E-02	47.46094	0.1914E-02		
37.89063	0.5811E-02	47.65625	0.1458E-02		
38.08594	0.6174E-02	47.85156	0.1108E-03		
38.28125	0.6631E-02	48.04688	0.8269E-03		
38.47656	0.6641E-02	48.24219	0.1020E-02		
38.67188	0.5574E-02	48.43750	0.9394E-03		
38.86719	0.4692E-02	48.63281	0.1014E-02		
39.06250	0.4826E-02	48.82813	0.1512E-02		

(b) Channel 1:4 FAS estimate magnitude (cont.)

Figure 16. Output of VAS PRINT command (cont.)

TEST: SAMPLE4 05/08/86 13:41 FAS eSTIMATE
CHANNELS: 1:4
WINDOW: HANNING PTS/SEGMENT: 512 SAMPLE RATE: 200

FREQUENCY	PHASE(DEG)	FREQUENCY	PHASE(DEG)	FREQUENCY	PHASE(DEG)
0.19531	-0.4441E+01	9.9e094	0.1771E+03	19.72656	0.e221E+02
0.39063	-0.5849E+01	10.15625	0.1700E+03	19.92188	0.1094E+03
0.58594	-0.5510E+01	10.35156	0.1704E+03	20.11719	0.1500E+03
0.78125	-0.2187E+01	10.5468d	-0.1477E+03	20.31250	-0.1726E+03
0.97656	-0.1421E+01	10.74219	-0.1430E+02	20.5078d	-0.1532E+03
1.17188	-0.1602E+01	10.93750	-0.e498E+01	20.70313	-0.1377E+03
1.36719	-0.1137E+01	11.13281	-0.5e23E+01	20.e9044	-0.1438E+03
1.56250	-0.1204E+01	11.32d13	-0.51e0E+01	21.09375	-0.1d74E+03
1.75781	-0.1246E+01	11.52344	-0.41d1E+01	21.28906	0.1702E+03
1.95313	-0.1283E+01	11.71875	-0.317dE+00	21.4e438	0.1350E+03
2.14844	-0.1334E+01	11.9146e	0.94e4E+01	21.e7969	0.1e0eE+03
2.34375	-0.1431E+01	12.10938	0.1651E+02	21.e7500	-0.1549E+03
2.53906	-0.1532E+01	12.30469	0.1929E+02	22.07031	-0.1479E+03
2.73438	0.1261E+01	12.50000	0.2e19E+02	22.26563	-0.1496E+03
2.92969	0.5882E+01	12.69531	0.3524E+02	22.46094	-0.1305E+03
3.12500	0.4095E+01	12.89063	0.3559E+02	22.65625	-0.1510E+03
3.32031	-0.3420E+01	13.08594	0.3494E+02	22.85156	-0.1516E+03
3.51563	-0.1076E+02	13.28125	0.3402E+02	23.04688	-0.1536E+03
3.71094	-0.9657E+01	13.47656	0.3563E+02	23.24219	-0.1506E+03
3.90625	0.1932E+02	13.67188	0.4196E+02	23.43750	-0.1552E+03
4.10156	0.1515E+03	13.86719	0.5006E+02	23.63281	-0.1601E+03
4.29688	0.1591E+03	14.06250	0.5545E+02	23.82813	-0.1749E+03
4.49219	0.1634E+03	14.25781	0.7d54E+02	24.02344	0.1693E+03
4.68750	0.1688E+03	14.45313	0.1455E+03	24.21875	0.1605E+03
4.88281	0.1726E+03	14.64844	0.1659E+03	24.41406	0.1569E+03
5.07813	0.1747E+03	14.84375	0.1714E+03	24.60938	0.1574E+03
5.27344	0.1762E+03	15.03906	-0.1126E+03	24.d0469	0.1e37E+03
5.46875	0.1770E+03	15.23438	-0.1268E+03	25.00000	-0.2299E+02
5.66406	0.1773E+03	15.42969	-0.14d0E+03	25.19531	-0.2e6dE+01
5.85938	0.1775E+03	15.62500	-0.1664E+03	25.39063	0.e2d7E+01
6.05469	0.1777E+03	15.82031	0.1777E+03	25.5e594	-0.9d70E+01
6.25000	0.1779E+03	16.01563	0.1657E+03	25.78125	-0.e2d4E+02
6.44531	0.1783E+03	16.21094	0.1502E+03	25.97656	-0.6d93E+02
6.64063	0.1787E+03	16.40625	0.1156E+03	26.17188	-0.4d37E+02
6.83594	0.1786E+03	16.60156	0.1726E+03	26.36719	-0.2e17E+02
7.03125	-0.1799E+03	16.796dd	-0.1720E+03	26.56250	-0.99e0E+01
7.22656	-0.1788E+03	16.99219	-0.1799E+03	26.7578d	0.9747E+01
7.42188	-0.1778E+03	17.18750	0.1504E+03	26.95313	0.4129E+02
7.61719	-0.1761E+03	17.38281	0.1245E+03	27.14844	0.8117E+02
7.81250	-0.1784E+03	17.57813	0.1077E+03	27.34375	0.7191E+02
8.00781	0.1755E+03	17.77344	0.8869E+02	27.53906	-0.7d5dE+01
8.20313	0.1745E+03	17.96875	0.6361E+02	27.73438	-0.4d44E+02
8.39844	0.1787E+03	18.16406	0.514dE+02	27.92969	-0.5162E+02
8.59375	-0.17d8E+03	18.35938	0.4695E+02	28.12500	-0.3543E+02
8.78906	-0.1792E+03	18.55469	-0.1127E+03	28.32031	0.9e7dE+01
8.98438	0.1792E+03	18.75000	-0.1222E+03	28.51563	0.5253E+02
9.17969	0.1785E+03	18.94531	-0.6483E+02	28.71094	0.73e2E+02
9.37500	-0.1796E+03	19.14063	0.246dE+02	28.90625	0.5127E+02
9.57031	-0.178dE+03	19.33594	0.3200E+02	29.10156	0.2490E+02
9.76563	-0.1795E+03	19.53125	0.4039E+02	29.29688	0.1177E+02

(c) Channel 1:4 FAS estimate phase.

Figure 16. Output of VAS PRINT command (cont.)

TEST: SAMPLE4 05/08/80 13:41 FAS ESTIMATE
CHANNELS: 1:4
WINDOW: HANNING PTS/SEGMENT: 512 SAMPLE RATE: 200

FREQUENCY	PHASE(DEG)	FREQUENCY	PHASE(DEG)	FREQUENCY	PHASE(DEG)
29.49219	0.7280E+01	39.25781	-0.1668E+03	49.02344	0.1203E+03
29.68750	0.6864E+01	39.45313	-0.1734E+03	49.21875	0.1490E+03
29.88281	0.9703E+01	39.64844	-0.1772E+03	49.41406	0.1595E+03
30.07813	0.1878E+02	39.84375	-0.1732E+03	49.60938	-0.1767E+03
30.27344	0.2466E+02	40.03906	-0.1586E+03	49.80469	-0.1520E+03
30.46875	0.1723E+00	40.23438	0.6887E+02	50.00000	-0.1182E+03
30.66406	-0.3088E+02	40.42969	0.7176E+02		
30.85938	-0.5507E+02	40.62500	0.6265E+02		
31.05469	-0.7379E+02	40.82031	0.2988E+02		
31.25000	-0.9056E+02	41.01563	-0.5506E+02		
31.44531	-0.1131E+03	41.21094	-0.7721E+02		
31.64063	-0.1101E+03	41.40625	-0.8294E+02		
31.83594	0.2325E+02	41.60156	-0.8778E+02		
32.03125	0.2882E+02	41.79688	-0.9770E+02		
32.22656	-0.1845E+01	41.99219	-0.9707E+02		
32.42188	-0.3344E+02	42.18750	-0.4942E+02		
32.61719	-0.4433E+02	42.38281	-0.3520E+02		
32.81250	-0.5695E+02	42.57813	-0.4039E+02		
33.00781	-0.1120E+03	42.77344	-0.4058E+02		
33.20313	0.1445E+03	42.96875	-0.5562E+02		
33.39844	0.1376E+03	43.16406	-0.9741E+02		
33.59375	0.1403E+03	43.35938	-0.1254E+03		
33.78906	0.1406E+03	43.55469	-0.1296E+03		
33.98438	0.1377E+03	43.75000	-0.1318E+03		
34.17969	0.1313E+03	43.94531	-0.1292E+03		
34.37500	0.1264E+03	44.14063	-0.1100E+03		
34.57031	0.1323E+03	44.33594	-0.1105E+03		
34.76563	0.1437E+03	44.53125	-0.1090E+03		
34.96094	0.1364E+03	44.72656	-0.1053E+03		
35.15625	0.1094E+03	44.92188	-0.7168E+02		
35.35156	0.7517E+02	45.11719	-0.2005E+02		
35.54688	-0.1493E+02	45.31250	-0.6735E+01		
35.74219	-0.7336E+02	45.50781	0.3301E+01		
35.93750	-0.5068E+02	45.70313	0.6220E+02		
36.13281	0.8984E+02	45.89844	0.1322E+03		
36.32813	0.1053E+03	46.09375	0.1251E+03		
36.52344	0.1590E+03	46.28906	0.1147E+03		
36.71875	-0.8563E+02	46.48438	0.1060E+03		
36.91406	-0.7164E+02	46.67969	0.1498E+03		
37.10938	-0.6783E+02	46.87500	-0.1602E+03		
37.30469	-0.6706E+02	47.07031	-0.1629E+03		
37.50000	-0.7286E+02	47.26563	-0.1750E+03		
37.69531	-0.1053E+03	47.46094	0.1769E+03		
37.89063	-0.1380E+03	47.65625	0.1765E+03		
38.08594	-0.1487E+03	47.85156	0.1599E+03		
38.28125	-0.1627E+03	48.04688	0.5501E+02		
38.47656	-0.1698E+03	48.24219	0.3775E+02		
38.67188	-0.1616E+03	48.43750	0.3366E+02		
38.86719	-0.1443E+03	48.63281	0.7217E+02		
39.06250	-0.1529E+03	48.82813	0.1011E+03		

(d) Channel 1:4 FAS estimate phase (cont.)

Figure 16. Output of VAS PRINT command (cont.)

VIII. PROPOSED ENHANCEMENTS

No project is ever completely finished; there will always be areas where enhancements can be made. Continued use of the Vibration Analysis System will surely ferret out additional changes that should be made, but already a few are apparent.

The analysis of signals really needs a good interactive interface. Currently, careful analysis requires that data be collected, calculations be made, and plots be viewed. And since plots are often difficult to read accurately, printed results are often needed. Since there are multiple parameters which can be adjusted during the analysis, the whole process is often repeated with slight adjustments. Using a powerful interactive work station, the need for hardcopy output during analysis would be alleviated and the entire process would be much smoother. Given a pointing device and appropriate software, the user could view a plot and simply by pointing to appropriate areas of the screen be shown printed values or corresponding areas of other plots. With a very fast processor, the user could quickly view the effects changing various parameters would have on the

calculations. These are improvements to the user interface; they do not provide better estimates, but rather make the estimates easier to obtain.

The methods of spectral estimation implemented by this analysis system are traditional, speedy, reliable techniques that have been used for years. More recent innovations in the science suggest themselves as alternatives. The maximum entropy method (MEM) of spectral estimation and autoregressive-moving average (ARMA) models often provide much better spectral estimates than the traditional methods, particularly when presented with limited sequences of data [1][3]. It is unclear whether these methods would provide significant improvements when applied to the particular application of vibration analysis, but additional study is surely warranted.

IX. CONCLUSIONS

The Vibration Analysis System performs quite well. Both the data acquisition system and the data analysis system perform as expected and are not unduly difficult to use. The system appears to function correctly, both on empirical data and on synthesized data.

The data acquisition system meets the requirements which were initially developed. It is able to provide a real-time display of the data as it is being sampled, it is able to sample eight data channels at well over 200 samples per second, and it is able collect over two million samples in a continuous sampling period. Combined with the simple on-site analysis programs, the system is nearly ideal for its purpose. However, the equipment is not as portable or as rugged as could be hoped, though it is unclear whether this will become a problem; the system has not been field tested sufficiently to know.

The data analysis system also works very well. It provides the needed functions and runs with a relatively simple command interface. It does suffer from working within the complex CMS environment. This necessitates

that the user understand more about CMS than might be desired, though the procedures needed to tailor the CMS environment for the VAS command are not unduly long and can be describe with a set of simple instructions.

It should be noted that the VAS is an experimental system. It has not been subjected to thorough testing for accuracy of results under all conditions. While it appears to create accurate results, this cannot be guaranteed. The system can be used as an adjunct to other analysis methods, and is valuable in this role. However, it should not be used as a substitute for these other methods in any case where the results might impact the safety of a structure.

REFERENCES

1. S. M. Kay and S. L. Marple, Jr., "Spectrum Analysis--A Modern Perspective," Proceedings of the IEEE, Vol. 69, No. 11, November 1982, pp 1380-1419

2. R. W. Clough and J. Penzien, Dynamics of Structures, McGraw-Hill, Inc., New York, 1975, pg. 208

3. E. A. Robinson, "A Historical Perspective of Spectrum Estimation", Proceedings of the IEEE, Vol. 70, No. 9, Sept. 1982, pp 885-907

4. P. Z. Peebles, Jr., Probability, Random Variables, and Random Signal Principles, McGraw-Hill, New York, 1980, pg 136

5. A. M. Abdel-Ghaffar and R. H. Scanlan, "Ambient Vibration Studies of Golden Gate Bridge: I. Suspended Structure," Journal of Engineering Mechanics, Vol. 111, No. 4, April 1985, pp 463-482

6. A. M. Abdel-Ghaffer and R. H. Scanlan, "Ambient Vibration Studies of Golden Gate Bridge: II. Pier-Tower Structure," Journal of Engineering Mechanics, Vol. 111, No. 4, April 1985, pp 483-499

7. Intel Component Data Catalog, Intel Corporation, 1982

8. F. da Cruz, KERMIT: A File Transfer Protocol, Digital Press, Digital Equipment Corporation, 1986

9. M. F. Cowlishaw, The REXX Language, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1985

10. R. P. O'Hara and D. R. Gomberg, Modern Programming Using REXX, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1985

11. VM/SP System Product Interpreter User's Guide, SC24-5238, International Business Machines, 1983

12. P. D. Welch, "The Use of the FFT for Estimation of Power Spectra: A Method Based on Averaging Over Short, Modified, Periodograms," IEEE Trans. on Audio and Electroacoustics, Vol. 15, No. 2, 1967, pp 70-73

13. K. G. Beauchamp and C. K. Yuen, Digital Methods for Signal Analysis, University Press, Cambridge, 1979, pg 159

14. L. R. Rabiner and B. Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1975, pg 402

APPENDIX A--VAS USER'S GUIDE

This appendix contains the Vibration Analysis System User's Guide.

Vibration Analysis System User's Guide

VAS User's Guide

I. INTRODUCTION

The Vibration Analysis System (VAS) is a system of hardware and software which runs on an IBM-PC running MS-DOS and an IBM mainframe running VM/SP CMS to provide data acquisition and spectral analysis capabilities to aid in the analysis of structural dynamics. The IBM-PC is used to provide data acquisition and very simple data analysis while the IBM mainframe is used to provide extensive data analysis.

The data acquisition system run on the IBM-PC consists of a Data Translation DT2801-A analog-to-digital converter PC expansion-slot board, a DT707 screw terminal panel, a custom external clock expansion-slot board, and appropriate software. The software is specifically tailored to interface to a Kinemetrics SC-1 signal conditioner which provides sensor signal amplification and filtering.

The data analysis system consists of a set of programs which run under IBM's VM/SP CMS operating system. The programs provide powerful data analysis and display capabilities, including the ability to obtain high quality pen plots of the analysis results. Functions are provided to compute auto and cross power spectral density estimates, auto and cross-correlation estimates, and cross channel coherence and transfer function estimates.

The notation used to define the command syntax in this document follows the standard set forth in section 1 of the IBM VM/SP CMS Command and Macro Reference (publication number SC19-6209). Where an abbreviation of a command parameter is permitted, the shortest allowable form is shown in uppercase letters with the remainder in lowercase (e.g. "CONvert" indicates that any of CON, CONV, CONVE, CONVER, or CONVERT may be used). Uppercase letters and words should be entered exactly as they appear in the format box as should parentheses. An entire word printed in lowercase is used to represent a variable for which an appropriate value should be substituted (e.g. "nnnn" might represent a numeric value which should be entered in place of "nnnn" in the format box). Square brackets are used to surround command parameters which are optional so the brackets themselves should not be entered. Where one of several choices must be selected, the choices are

represented by stacking the mutually exclusive choices vertically in the format box. An underscore is used to indicate a default option or value.

The document assumes that the user is familiar with the basic systems on which VAS operates. No attempt is made to provide detailed instruction on the use of MS-DOS, CMS, KERMIT, Tektronix 4010 graphics terminals, etc. The assumption is also made that the system is being used with the Kinemetrics SC-1 signal conditioner so the reader should be somewhat familiar with this device. Other sources of input signals can be used with the system, and their use can be easily inferred from the descriptions of the use of the SC-1.

This document is intended to describe the use of the VAS. It will assume that the reader has some knowledge of spectral estimation terminology and techniques. While it is not intended to provide a tutorial on spectral estimation, it does attempt to provide some suggestions about use of the system to assure that reasonable estimates can be made.

II. THE DATA ACQUISITION SYSTEM

A. Hardware Description and Installation

The data acquisition system is a set of both hardware and software which runs on an IBM-PC or compatible computer. The following equipment is required:

1. IBM-PC or compatible with:
   A. at least 256K RAM,
   B. 8087 numeric coprocessor,
   C. Color/graphics adaptor.
2. Data Translation DT2801-A board.
3. Data Translation DT707 screw terminal panel.
4. Custom External Clock board.

The PC should be equipped with at least 256K of RAM and should also have an Intel 8087 numeric coprocessor installed if the simple data analysis routines are to be run on the PC. The data acquisition program itself, DTDATA, does not require the 8087. For storage of the data, a Winchester hard disk is suggested, though for many applications a single 360K floppy disk drive is adequate. With the Winchester disk, data acquisition at rates as high as 7000 samples per second are possible. Using floppy disks, the maximum throughput is reduced to about 1000 samples per second. The PC must also be equipped with a graphics board and display which are compatible with the IBM color-graphics board and color display. Support for other forms of display boards is not included.

The PC component of the VAS consists of a DT2801-A A/D board which must be installed in an expansion slot of the PC, a custom-made external clock board which also should be installed in an expansion slot of the PC, and the software consisting of the programs DTDATA, PLTPSD, PLTFAS, PLTDAT, and PLTDAT4.

Installation of the DT2801-A in the PC should be performed as specified in the User Manual for DT2801 Series. The DT2801-A can be installed in any unused full-size expansion slot. The board has multiple configuration options but is configured properly at the factory, so no jumpers should need to be changed. The board is configured by default to run with 8 bipolar differential data channels. The board is also factory configured to use DMA channel 1 and a base I/O address of 2EC.

The external clock board should be installed at the same time as the DT2801-A, using another expansion slot. As the bottom of the board has multiple wirewrap posts protruding, exercise care in choosing an appropriate slot; pick an expansion slot which will leave adequate room for the clock board to fit. If it is at all possible that the clock board might make contact with another board, insert a piece of cardboard or some other insulating material between the boards to keep them from contacting.

Once the boards are installed, the PC can be put back together. Both boards have connectors for attaching cables which should be accessible from the back of the PC. When being transported, the cables can be removed from the boards and reinstalled when the system is relocated. The DT2801-A is connected to the DT707 screw terminal panel by a wide ribbon cable. As the DT707 terminates in simple screw posts, the board can be mounted in a small box and the connections for channels 0 to 7 attached to BNC connectors. The posts are marked CH 0 and 0 RET, CH 1 and 1 RET, etc. In addition, the posts marked EXT CLK and D GND should be connected to a 1/8" female phone plug (the ground connection is on the outside of the phone plug); this is the connection for the external clock board connector.

B. External Interface

The DT2801-A is connected to the outside world through the DT707 screw terminal panel, which in turn is attached to BNC connectors to provide up to 8 channels of input data. These channels may then be connected to input signals. In general, this will be through the Kinemetrics SC-1 signal conditioner which provides 4 channels of data.

The input data range of the DT2801-A is from +10V to -10V. This provides full scale readings. The DT2801-A provides overvoltage protection of its inputs to +/-20V and should be protected from higher input voltages. The SC-1 can create signals to +/-16V which will exceed the DT2801-A full scale reading but should not damage the board.

The DT2801-A does not provide lowpass filtering of the input data. In almost all cases, it is extremely important that input analog data be lowpass filtered to prevent a phenomenon known as aliasing from adversely affecting results. The SC-1 provides this filtering.

C. Summary of Programs

Five MS-DOS programs are provided with the VAS. These are DTDATA, PLTPSD, PLTFAS, PLTDAT, and PLTDAT4. DTDATA is used to drive the DT2801-A to collect data and is generally the first program run in an experiment. PLTDAT provides a means of graphically displaying portions of a single channel of data which has been acquired and written to disk. PLTDAT4 is similar to PLTDAT, but will display four channels of data. PLTFAS and PLTPSD calculate Fourier amplitude spectrum estimates and power spectral density estimates from previously collected data and graph the results. DTDATA is used to acquire input data while the other programs are used to provide simple analysis of the data to verify that experimental parameters have been properly set.

D. Using DTDATA to Acquire Data

DTDATA is used to acquire data. It is menu driven and is invoked by entering the command, "DTDATA". It allows the user to set various parameters such as the fileid of the disk file to be created, the number of data channels to be sampled, the sample rate, the duration of the sampling, and the number of channels to be displayed in real time. It also allows the entry of a few notebook entries for each data channel, used to assist the operator in recording pertinent information about the experiment such as the location of the sensors and the settings on the SC-1.

When the DTDATA command is started, it presents the following menu:

```
A.  Recall parameters
B.  Disk filename:  <no disk log>
C.  Number of channels to record: 1
D.  Sample rate per channel: 125.00 Hz
E.  Duration of sampling: <continuous>
F.  Number of display channels: 1
G.  Clocking:  EXTERNAL
H.  Data log
I.  Save parameters

Enter letter of item to change, RETURN to begin, or
ESC to exit:
```

The current settings of the parameters are shown and may be altered in any order simply by typing the letter of the menu item. The parameters can also be saved or recalled from disk.

The current parameters may be stored to disk by selecting menu item I while selecting menu item A causes a previously stored set of parameters to be recalled. The disk file "DTDATA.DEF" is used to store the parameters. Each of the specified parameters except the disk filename are saved, including the data log parameters set by menu selection H (recalling the disk filename would make the accidental erasure of previously recorded data much more likely). The use of parameter save and recall makes collecting data easier and less error prone.

The fileid of the disk file to be used to store the collected data is set by selecting menu item B. When "B" is typed, the following prompt is received:

Enter filename for disk log file or RETURN for none:

The desired MS-DOS fileid may then be entered. In general, a fileid of the form "filename.RAW" should be used, as "RAW" is a special filetype used by the mainframe analysis system, but any valid MS-DOS fileid may be used. If the real-time display of the sampled data is desired, but no disk record of the data is needed, the filename may be left blank.

The number of data channels to be sampled is selected by entering "C" in response to the menu prompt. The following prompt will be received:

Enter number of channels to sample (1-8):

The sample rate is selected by entering "D" in response to the menu prompt. The following will be displayed:

Enter sample rate per channel (in samples/second):

At this point, the number of samples per second per channel can be selected. The value entered will be changed if necessary to conform to the limitations of the hardware, as only a distinct set of sample rates is possible. The actual value used will be very close to the chosen value in most cases; this becomes a problem only at very high sample rates where the difference between

adjacent sample rate possibilities is large. The total sample rate (that is, the number of channels being sampled times the sample rate per channel) is limited to a maximum of about 7500 sps. The minimum value is also limited. When using internal clocking, the overall sample rate is limited to 12.3 sps. When using external clocking, the sample rate per channel is limited to 7.65 sps.

By default, data collection continues until the ESC key is pressed. This can be changed by selecting "E" in response to the menu prompt. The following will be displayed:

```
Enter duration in seconds or 0 for continuous:
```

Data collection will continue for at least the specified length of time. The actual sampling duration may be slightly greater than the specified value.

When the sample rate is 250 sps per channel or below, one channel of data can be displayed on the PC's screen in real time. If two channels are to be displayed, the sample rate per channel can be no greater than 125 sps. To specify the number of channels to be displayed in real time, enter "F" at the menu prompt. The following will be displayed:

```
Enter number of channels to display in real time
(0-2):
```

If one channel is being displayed, the entire upper section of the PC display is used to graph the data. If two channels are being displayed, the upper section of the screen is split to display the two channels side by side. By default, data channel 1 will be displayed on the left side and channel 2 will be displayed on the right side.

The channel being displayed can be changed during data collection by pressing a function key. Pressing F1 causes the channel number being displayed on the left-half of the screen to increase by one while pressing F2 increments the display channel for the right-half display. A sample display showing the beginning of the collection of two channels of data is shown in figure 1.

The display has a resolution of 128 points vertically and 640 points horizontally. Since the input data can take on 4096 distinct values, small amplitude inputs may

Figure 1. Sample display from DTDATA.

show very little detail in the vertical direction if the program is set to display the full input scale. The display gain is therefore made adjustable. Pressing ALT-F1 (hold down the ALT key and depress the F1 key) sets the display to scale the full range of inputs values (from -2048 to 2047) to fit within the 128 point screen. Pressing ALT-F2 spreads out the display to show the input values from -1024 to 1023. ALT-F3 will scale the display to show values from -512 to 511, ALT-F4 shows -256 to 255, ALT-F5 shows -128 to 127 and ALT-F6 shows input values from -64 to 63.

The DT2801-A contains an internal clock which can be used to set the sample rate. For applications where the phase difference between channels is unimportant, the internal clock provides an acceptable clock signal. Applications which require evaluating the phase of cross-spectrum harmonics, or which require careful cross-correlation comparisons may require the use of the external clock. The external clock board creates a clock which allows the near simultaneous sampling of each of the data channels. The internal clock provided by the DT2801-A samples at a constant rate. For example, if the sample rate per channel is set to 10 Hz and 4 channels are being sampled, the internal clock will sample channel 1 at time 0, channel 2 is sampled .025 seconds later, channel 3 is sampled .05 seconds after channel 1, and channel 4 is sampled .075 seconds after channel 1; channel 1 is then sampled again, .1 seconds after its first sample was taken. The external clock board creates a pulse train which causes the channels to be sampled at about the maximum rate the A/D board can convert, then it pauses. Using the external clock, channel 1 would be sampled at time 0, channel 2 is sampled .00004 seconds later, channel 3 is sampled .00008 seconds after channel 1 and channel 4 is sampled .00012 seconds after channel 1; the sampling then pauses for about .09988 seconds and channel 1 is sampled again. This nearly simultaneous sampling is required only when using cross-channel spectrum phase plots, or when doing very exacting cross-correlation analyses.

Entering "G" in response to the DTDATA menu prompt causes the clocking parameter to toggle between INTERNAL and EXTERNAL. When run with the parameter set to EXTERNAL, the external clock board must be plugged into an expansion slot of the PC, and the cable connector from it must be plugged into the external clock input for the DT2801-A board.

Entering "H" in response to the menu prompt causes the data log to be selected. The data log is rather like an experimental notebook which allows a few experiment parameters to be recorded with the data which is collected. For each channel being collected (the number of channels to be collected must be set before the data log is selected), the user is prompted for a 15 character location code, the SC-1 amplifier attenuation setting and lowpass filter setting. The user is also asked whether a 12 dB external attenuator is present on the channel. The location code can contain 15 non-blank characters and would typically be used to record information about the location and orientation of the sensors. The amplifier attenuation is used to scale the input data to allow the actual acceleration to be computed from the input voltages; it is assumed that accelerometers are being used which produce 2.5V/g and that the SC-1 signal conditioner is being used (which amplifies the input signal by 100,000 with 0 dB attenuation). In addition, 12 dB attenuators can be placed on the input lines to reduce the voltage further, and the user must respond "Y" or "N" to the prompt asking whether the attenuator is present. Finally, the lowpass filter setting may be entered as 5 characters. This value is stored as a character string rather than as a floating point number, so any 5 non-blank characters can be entered.

The data log prompts are shown below:

```
Data log -- Present value shown in parens
Channel 1 Data.  Press RETURN to leave unchanged
Enter location code <= 15 chars ():
Enter amplifier attenuation in db (0):
Is external 12 dB attenuator present (N):
Enter low pass filter setting <= 5 chars ():
Channel 2 Data.  Press RETURN to leave unchanged
Enter location code <= 15 chars ():
Enter amplifier attenuation in db (0):
Is external 12 dB attenuator present (N):
Enter low pass filter setting <= 5 chars ():
```

In addition to the data log information, the header of the data file created will contain the sample rate used, and the date and time the data collection was started. The date and time is obtained from MS-DOS and so should be set correctly by the user when MS-DOS is initialized.

E. Using PLTDAT and PLTDAT4 to Display Data

After a file of data has been collected by DTDATA, the PLTDAT and PLTDAT4 programs can be used to verify that the data was indeed collected correctly and was written to disk.

PLTDAT is used to display a single channel of data on the screen of the PC. It requires parameters to define the fileid of the data and the number of the data channel to be displayed. The format of the PLTDAT command is:

PLTDAT	[fileid [channum]]

where:

fileid — is an optional parameter which specifies the name of the file on disk which contains the data to be graphed. If not specified on the command line, the user will be prompted for the name of the file.

channum — is an optional parameter which specifies the channel number of the data to be graphed. If only one channel of data is contained in the specified file, this parameter is not required. If not specified on the command line, the user will be prompted for the channel number if more than one channel exists.

The output of the PLTDAT command is a screen which plots up to 1024 data values. The screen will be annotated with the name of the file being displayed, the channel number, the sample rate, the number of points displayed, and the minimum and maximum value of the data displayed. PLTDAT will leave the plot displayed until a key is pressed.

PLTDAT4 is used to display up to 4 channels of data on the screen of the PC. It requires parameters to define the fileid of the data and the number of the first data channel to be displayed. The format of the PLTDAT4 command is:

PLTDAT4	[fileid [channum]]

where:

fileid is an optional parameter which specifies the name of the file on disk which contains the data to be graphed. If not specified on the command line, the user will be prompted for the name of the file.

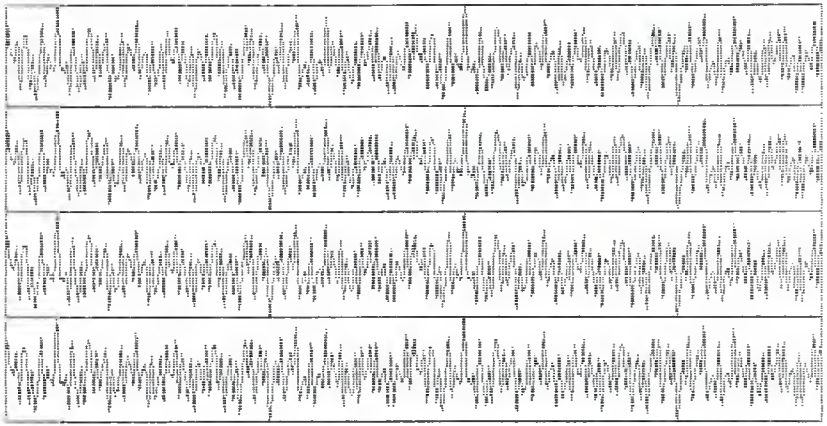
channum is an optional parameter which specifies the channel number of the data to be graphed. If four or fewer channels of data are contained in the specified file, this parameter is not required. If not specified on the command line, the user will be prompted for the channel number if more than four channels exist. PLTDAT4 will show the four sequential channels beginning with the channel specified by the "channum" parameter.

The output of the PLTDAT4 command is a screen which plots up to 1024 data values. The first data channel will be plotted at the top, followed by up to three more channels of data. The screen will be annotated with the name of the file being displayed, the channel number of the first channel displayed, the sample rate, the number of points displayed, and the minimum and maximum value of the data displayed. PLTDAT4 will leave the plot displayed until a key is pressed.

Figure 2 shows an example of the output of the PLTDAT4 command.

F. Using PLTPSD and PLTFAS to Calculate Spectra

Rough estimates of the spectrum of a data channel can be calculated and graphed with the PLTPSD and PLTFAS commands. A power spectral density estimate is calculated with the PLTPSD command while a Fourier amplitude spectrum estimate is calculated with the PLTFAS command. The FAS of signal is simply the square-root of the PSD of the signal.



File 'blah4.raw', 1st Channel = 1, sps = 100.00
maximum = 0.022cm/sec/sec minimum = -0.018cm/sec/sec #pts = 1024

Figure 2. Sample display from PLTDAT4.

The PLTPSD and PLTFAS compute a spectrum estimate of up to 1024 data values. No time averaging or data windowing is provided, so these are extremely rough estimates and will vary considerably from sample to sample. The estimates are intended to provide the user of DTDATA with an idea of the magnitude and location of the major spectral peaks in the data being collected.

PLTPSD PLTFAS	[fileid [channum [logflag]]]

where:

fileid — is an optional parameter which specifies the name of the file on disk which contains the data to be graphed. If not specified on the command line, the user will be prompted for the name of the file.

channum — is an optional parameter which specifies the channel number of the data to be graphed. If only one channel of data is contained in the specified file, this parameter is not required. If not specified on the command line, the user will be prompted for the channel number if more than one channel exists.

logflag — is an optional parameter which is used to specify whether the y-axis is to be linear or logarithmic. If the parameter is absent, the plot will be linear, otherwise it will be logarithmic. Any non-blank characters may be substituted for "logflag."

G. Effectively Using the Acquisition System

The purpose of the PLTDAT, PLTDAT4, PLTFAS, and PLTPSD programs is to assist the user in acquiring good data. This section will attempt to provide some insight into appropriate methods of selecting sample rates and filter settings.

A typical experiment will consist of several distinct parts: equipment test, sensor placement, signal analysis, and data acquisition.

If the sensors are going to be placed in somewhat inaccessible locations, the equipment should be tested before the sensors are put in place. The cables between the SC-1 and the sensors should be attached and DTDATA should be run, choosing a sample rate of about 200 sps, sampling the appropriate number of channels and not logging data to disk. Simply view the real-time display and verify that each of the channels appears to be working. While this won't guarantee that the equipment will work when it is relocated, it will occasionally save some time and effort.

When it appears that the SC-1, the accelerometers, the cables, the PC, the DT2801-A, the DT707, the external clock board, and the software are all functioning properly, the cables can be strung to their appropriate locations. If the system is not working, refer to the section on trouble-shooting.

When the system is working and the sensors are in place, the signal analysis phase can be performed. Here, the experimenter is attempting to understand characteristics of the signals being collected so as to determine what sampling rate should be used for the long duration data collection. Initially, DTDATA should simply be set to display one data channel, sampling at about 200 sps, and the SC-1 attenuator settings for the channels should be adjusted to provide nearly full scale coverage, but being careful not to allow the input signal to actually exceed the input range of the DT2801; it is much better to not utilize the full 12-bit input range of the DT2801-A than to have the signal exceed the input range.

With the attenuation set for each channel, the SC-1 should be set to provide lowpass filtering of the data at about 100 Hz. DTDATA should be used to collect about 1000 data values from each channel (writing them to disk this time) with the sample rate set to 400 sps. PLTDAT or PLTDAT4 can then be run to ensure that the data was actually collected. PLTFAS is then run to examine the location of the spectral peaks. This is done to determine what sample rate should be used for the next collections. The horizontal axis plotted by PLTFAS represents frequency where the left edge is 0 Hz and the right edge is 200 Hz

(one-half of the sample rate). The magnitude and location of the peaks in the plot should be examined to determine the frequency of the largest peak which is to be kept for analysis. The lowpass filter for future collections should then be set to a value slightly above this, and the sample rate should be set to about 4 times the lowpass filter setting for most work. In some cases, it may be necessary to increase the sample rate to greater than four times the lowpass filter setting.

For a given sample rate, only frequencies from 0 to one-half the sample rate can be correctly collected. But the higher frequencies do not disappear, they are folded back into the represented frequency range. If a signal is collected at 400 sps, the represented frequency range is from 0 to 200 Hz. If the signal contains a sinusoid of 210 Hz, it will show up as a sinusoid of 190 Hz. Similarly, a sinusoid of 390 Hz will appear as a 10 Hz signal as will a 410 Hz signal. Thus, analog lowpass filtering is required to remove these high frequency signals before the signal is digitized.

With the lowpass filter set at 100 Hz, the frequencies beyond 100 Hz are attenuated. The SC-1 uses a filter which has an attenuation of 18 dB per octave; for every factor of two increase in frequency (i.e., for each octave) the filter attenuation increases by 18 dB (a factor of 8). Signals with frequencies near 200 Hz are are attenuated by a factor of 8, signals at 300 Hz are attenuated by about 22, and signals at 400 Hz are attenuated by about 64.

Using a lowpass frequency of 100 Hz and a sample rate of 400 sps, the FAS spectrum will range from 0 to 200 Hz. Input signals in the range of 100 Hz to 200 Hz will be appear unaliased, but will be attenuated by the filter. Signals from 200 Hz to 300 Hz will be aliased and will appear as signals from 200 Hz to 100 Hz and will be attenuated by a factor between 8 and 22. Signals from 300 Hz to 400 Hz appear as signals from 100 Hz to 0 Hz and are attenuated by a factor between 22 and 64. Thus, if there are strong spectral peaks in the range from 300 Hz to 400 Hz, an attenuation between 22 and 64 may not be great enough to prevent the peaks from appearing in the spectrum plot from 0 to 100 Hz. In such a case, the sample rate will need to be increased above 4 times the lowpass filter setting. In addition it should be realized that the lowpass filter used by the SC-1 actually begins

attenuating the input signal somewhat before the cutoff frequency; at the specified frequency the signal is attenuated by 3 dB. For greatest accuracy, the filter frequency should be well above the highest frequency spectral component to be kept.

Once an appropriate lowpass filter setting and sample rate are selected, a large number of samples should be collected. In general, at least 8192 samples per channel should be collected, and if possible 32768 or more samples should be collected. Larger sample sizes allow improved spectrum estimates.

The frequency resolution of a spectral estimate is linear; if a signal is acquired at 1024 sps, and 512 frequency components are estimated, each component represents 1 Hz. Relatively, low frequency signals have poorer resolution than high frequency signals. If increased resolution of low frequency components is desired, the sample rate and lowpass filter cutoff should be lowered and another set of data should be collected. In a typical example, an initial set of data might be collected at 200 sps and a second set could be collected at 40 sps. The first run will require less than 3 minutes to collect 32000 samples while the second run will require about 14 minutes.

When possible, the real-time display should be used to monitor the data acquisition to verify that appropriate signals are being collected. Intermittent problems with cable connectors may appear in the middle of a run or large vibrations may cause the signal to exceed the input range of the A/D board. Indeed, it is probably worthwhile to collect several sets of data rather than trust that a single set of data was collected without incident.

H. Trouble-shooting

As with any complex set of technology, the data acquisition system will eventually fail to work. This section attempts to anticipate a few common problems and proposes ways of finding and solving them. Only problems specific to the DTDATA acquisition system and software are explored here; the PC and SC-1 are subject to their own failure modes which must be diagnosed separately.

Symptom: DTDATA presents error message "Timeout waiting for status flag" as data acquisition is started.

Cause: The DT2801-A board is not being addressed correctly by the PC. Verify that it is installed in a slot and that the jumpers are set according to the Data Translation manual. Be sure that no other expansion boards are plugged in which might be conflicting with address used by the DT2801-A. Try placing the DT2801-A in a different expansion slot. Try a different DT2801-A. Try a different PC.

Symptom: DTDATA presents error message "Timeout waiting for A/D data" after screen is cleared. No data is acquired.

Cause: The DT2801-A is not sending data to the PC. If using external clocking, attempt to use internal clocking. If internal clocking works but external clocking fails, verify that the external clock board is plugged into a PC expansion slot, that the board itself has no loose components, and that the connector from the BNC box to the clock board is connected. Check the integrity of the cables and connectors and verify with a VOM meter that the cable and connectors are not shorted and that there is continuity between the output of the external clock board and the DT707 external clock input. Using a logic probe or oscilloscope, unplug the cable connecting the clock board to the BNC box and verify that a pulsed signal is being generated by the external clock board when DTDATA is attempting to collect samples. If the clock board is not generating a signal, recheck that the board is plugged in and all components are installed, and verify that DTDATA is set to use external clocking and that the acquisition has been started by hitting RETURN (the screen of the PC should clear). If the board is generating the signal with the cable connecting it to the BNC box unplugged, verify that the signal continues with the connection in place. Verify that the signal appears on the DT707 screw terminal panel connection. Verify that the signal appears on the DT2801-A board (which may require removing the board from the PC). Verify that no other PC hardware is using I/O addresses 2B0, 2B1, 2B2, and 2B3. Try a different external clock board.

Symptom: DTDATA appears to be collecting data but the real-time display shows a flat line.

Cause: The DT2801-A is receiving very low level signals, incorrect signals, or no signals. Run DTDATA with the input channel in question left unconnected and with the display set for maximum gain (ALT-F6). If the display shows no vertical deflection, the DT2801-A may not be working; try the same test with other channels and check the DT2801-A connection to the PC and to the DT707 screw terminal panel. If a flat line continues on all channels with DTDATA set for full display gain, use a signal generator to apply a known signal to the input. If a single channel appears to have failed, use other channels or have the DT2801-A fixed. If a floating connection shows tiny vertical changes, the DT2801-A is probably working and the problem is with the signal conditioner. (this can be verified by using a signal generator or some other independent source of an input such as a 1.5V battery--apply the input signal to the screw terminal panel directly, bypassing the BNC connectors).

Many other problems may be encountered. A common sense approach of isolating the problem to a particular component will allow the user to solve many problems. Most problems will generally involve connectors. The data acquisition system has connectors from the BNC connector box to the DT707 which is in turn connected to the DT2801-A which is plugged into the PC. In addition, a cable connects the external clock board to the BNC connector box and the board itself plugs into the PC. When the system is not working, the first step is to check all the connectors, using the symptoms of the failure as a clue. Also be sure the software is being used properly. DTDATA attempts to present error messages to the user rather than simply leaving the machine in an undefined state or quietly exiting without having worked; if DTDATA is presenting no error messages and is displaying data which is not quite as expected, the odds are good it is the input signal rather than the acquisition system. Begin with the DT2801-A to SC-1 interface and work toward the transducers when searching for the problem.

III. THE DATA ANALYSIS SYSTEM

A. Overview

The VAS data analysis system runs under IBM's VM/SP CMS operating system and provides functions to compute and display spectral estimates, correlation estimates, coherence estimates and transfer function estimates.

After one or more sequences of data have been collected by DTDATA, they must be uploaded to the mainframe to be processed by the data analysis system. The easiest way to upload the data is with the KERMIT terminal emulator and file transfer program.

Uploading raw data with KERMIT is relatively easy. The steps outlined in the KSU Academic Computing Activities KERMIT handout can be followed if two additional commands are added to the CMS command set. The "SET FILE BINARY" and "SET LRECL 64" commands should be issued after the CMS "KERMIT" command is invoked and before the "RECEIVE" command is issued. This will allow the binary data which is recorded by DTDATA to be uploaded without ASCII to EBCDIC translation. Further information about KERMIT can be obtained from the KSU Computing Activities User Information Center.

The raw data files which are uploaded to CMS must have a filetype of "RAW." That is, the file which was created by DTDATA should be named "filename.RAW" where "filename" can be any alphanumeric field up to eight characters long.

The entire data analysis system in CMS consists of a single CMS command called "VAS." This one command accepts many parameters and options to provide all the required data analysis and display functions. Functions are provided to convert the raw input data into usable form, to compute auto and cross-spectrum estimates, to compute auto and cross-correlations, to compute coherence and transfer functions, and to display or print the results.

In CMS, files on a disk are identified by a filename and a filetype which can each contain up to eight alphanumeric characters. The filename is used by the CMS VAS command to uniquely identify a single experiment and appears on the printed and plotted output as the test name. The filetype is used internally by VAS to identify

the kind of data contained in the file. For instance, "RAW" is used to identify the raw binary data which is in the form created by the DTDATA command. The following filetypes are used by the VAS command:

FILETYPE	Function
RAW	Unprocessed data in the form created by the DTDATA command
Dii	Data for channel "i"
IOO	Various information about the data in other files
Sij	Cross PSD estimates for channels "i" and "j" or auto PSD estimate if i=j
Cij	Auto or cross-correlation estimates for channels "i" and "j"
Kij	Coherence function estimates for channels "i" and "j"
Tij	Transfer function estimates for channels "i" and "j"

The user of the system seldom has to concern himself with filetypes. The filetype of "RAW" must be used when uploading the data, and the use of filetypes will be required when using standard CMS commands to list and manipulate CMS disks.

In general, the analysis of a set of data will follow a fairly well defined sequence. The user will log on to CMS, change the CMS environment as specified in section "J", upload the raw data with KERMIT, then use the "VAS CONVERT" command to prepare the data for analysis. "VAS SPECTRUM" commands are then used to compute desired auto and cross-spectrum estimates. If desired, "VAS COHERENCE" and "VAS TRANSFER" commands can then be used to compute coherence and transfer function estimates. "VAS CORRELATE" may be used to compute auto and cross-correlations. The data itself or the results of the computations can be plotted either interactively or on the CALCOMP drum plotter with the "VAS PLOT" command, while the "VAS PRINT" command produces non-graphical output.

B. The VAS CONVERT Command

After a single file containing the output from a particular experiment is uploaded, it must first be converted to a format more amenable to processing on the

mainframe. The raw data file begins with a header record which contains information about the experiment followed by the data in reversed-byte integer format with the data for multiple channels interleaved. The "VAS CONVERT" command is used to separate the raw format data into one CMS file for each data channel plus an informational file which contains information from the header. The data is kept in files with a filetype of "Dnn" where "nn" is "11" for the first channel, "22" for the second channel, through "88" for the eighth channel. The informational file is given a filetype of "I00" and contains the sample rate used to collect the data, the date and time the data was collected, and for each channel, the lowpass filter frequency and location code.

The format of the "VAS CONVERT" command is:

VAS CONvert	filename

where:

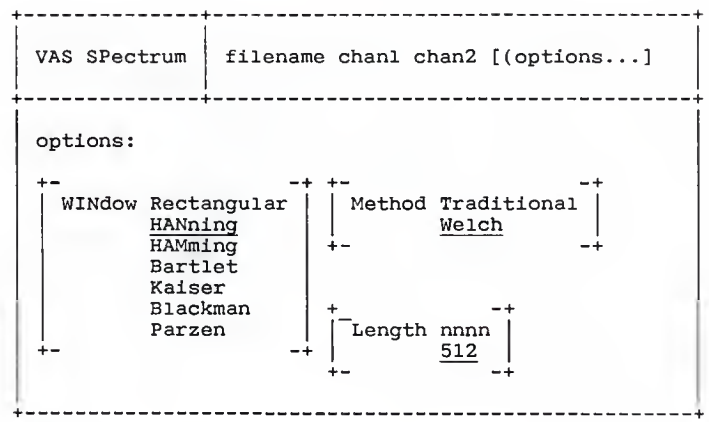
filename specifies the name of the file on disk which contains the raw data to be converted to internal format. The file must have a filetype of RAW.

If the file "SAMP1 RAW" contains the data for an experiment run with four channels, the command "VAS CON SAMP1" would create the five files "SAMP1 D11", "SAMP1 D22", "SAMP1 D33", "SAMP1 D44", and "SAMP1 I00".

C. The VAS SPECTRUM Command

Spectral estimates are computed using the "VAS SPECTRUM" command. Options are provided to select from two spectral estimation techniques, to select one of seven possible smoothing windows, and to select the number of frequency components the estimate is to calculate.

The format of the "VAS SPECTRUM" command is:



where:

filename specifies the name of the files on disk which contain the data.

chan1 specifies the channel number of the first data file.

chan2 specifies the channel number of the second data file.

options:

WINDOW winname
selects the type of smoothing window which is to be used. The default HANNING window usually produces a good estimate and should normally be used. In special cases, an alternate window might produce a better estimate. The window which is specified is recorded in the informational file associated with the experiment (in the "filename I00" file).

LENGTH nnnn
selects the number of frequency components which are to be be computed. The length must be a power of 2 and must be no larger than 2048.

Valid values include 2048, 1024, 512, 256, and 128. The default of 512 is reasonable in most cases. The length which is specified is recorded in the informational file associated with the experiment.

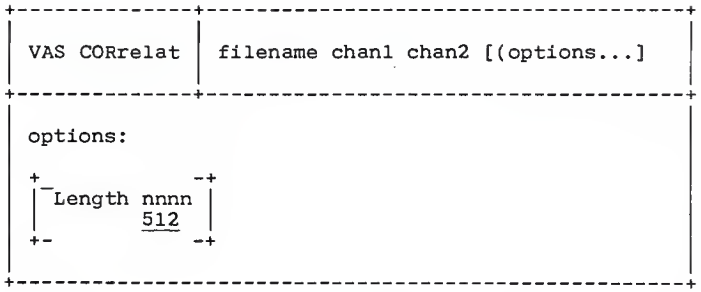
METHOD methodname
selects the spectral estimation technique which is to be used. The default method should generally be used except when conserving CPU time is crucial or in the rare case when the alternate method of spectral estimation might provide a better estimate. The default method will use a little less than twice the CPU time of the traditional method, but usually provides better estimates. The method specified is recorded in the informational file associated with the experiment.

"VAS SPECTRUM" uses files named "filename Dii" and "filename Djj" to produce the spectral estimate which is put in the file "filename Sij". If chan1 and chan2 are identical, an auto spectral estimate is computed while a cross-channel estimate is made if they differ. For example, the command "VAS S SAMP1 1 1" computes a 512 point auto-spectrum estimate of channel 1 of SAMP1. "VAS SPEC SAMP1 1 2 (LEN 2048" produces a cross-spectrum estimate with 2048 values.

The informational file associated with the experiment is updated by the "VAS SPECTRUM" command to record the method used, the number of components computed, and the data window used. This information is used by the "VAS PLOT" and "VAS PRINT" commands to annotate graphs and tables. Note that the informational file only holds this information for the LAST spectrum estimate. If estimates for different channels are made using different methods, windows, or lengths, the informational file will reflect only the values set from the last estimate.

D. The VAS CORRELATION Command

Auto-correlations and cross-correlations are computed by the "VAS CORRELATION" command. The format of the command is:



where:

filename specifies the name of the files on disk which contain the data.

chan1 specifies the channel number of the first data file.

chan2 specifies the channel number of the second data file.

options:

LENGTH nnnn
specifies the number of correlation lags to be computed. The length must be a power of 2 and must be no larger than 2048. The default is 512. Specifying a smaller value does not significantly speed up the computation.

"VAS CORRELATION" uses files named "filename Dii" and "filename Djj" to produce the cross-correlation estimate which is put in the file "filename Cij". If chan1 and chan2 are identical, an auto-correlation is computed. For example, the command "VAS CORR SAMP1 1 1" computes a 512 point auto-correlation of channel 1 of SAMP1. "VAS CORR SAMP1 1 2 (LEN 128" produces a cross-correlation estimate with 128 values.

E. The VAS COHERENCE Command

Coherence functions (also known as coherence spectrums) are computed using the "VAS COHERENCE" command. The format of the command is:

VAS COHerenc	filename chan1 chan2

where:

filename specifies the name of the files on disk which contain the spectral estimates from which the coherence function is to be computed.

chan1 specifies the channel number of the first spectrum file.

chan2 specifies the channel number of the second spectrum file.

"VAS COHERENCE" uses files named "filename Sii", "filename Sjj", and "filename Sij" to produce the coherence estimate which is put in the file "filename Kij". That is, it uses the auto-spectrum estimate of the two channels and their cross-spectrum estimate; these must previously have been calculated by the "VAS SPECTRUM" command. For example, the command "VAS COH SAMP1 1 4" computes a coherence function estimate for channels 1 and 4 of SAMP1.

F. The VAS TRANSFER Command

Frequency domain transfer functions are computed using the "VAS TRANSFER" command. The format of the command is:

VAS Transfer	filename chan1 chan2

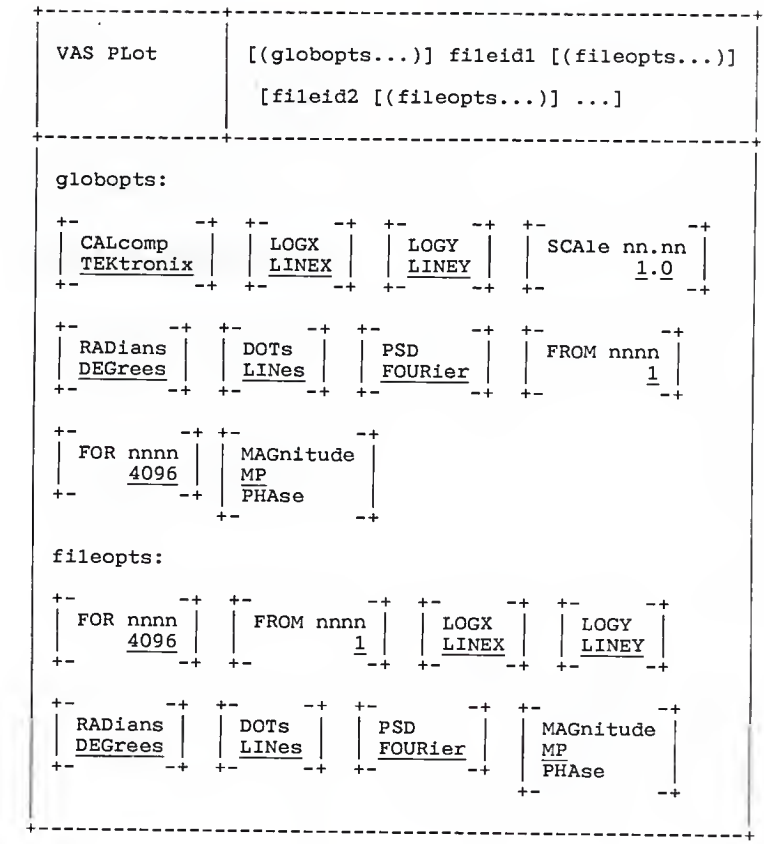
where:

filename	specifies the name of the files on disk which contain the spectral estimates from which the transfer function is to be computed.
chan1	specifies the channel number of the first spectrum file.
chan2	specifies the channel number of the second spectrum file.

"VAS TRANSFER" uses files named "filename Sii" and "filename Sjj" to produce the transfer function estimate which is put in the file "filename Tij". That is, it uses the auto-spectrum estimate of each of the specified channels; these must have previously been calculated by the "VAS SPECTRUM" command. For example, the command "VAS TRAN SAMP1 1 3" computes a transfer function for channels 1 and 3 of SAMP1.

G. The VAS PLOT Command

The results of computing spectrum, correlation, transfer function, and coherence function estimates as well as data can be plotted by using the "VAS PLOT" command. Options on the command allow the plots to be made either interactively on a Tektronix 4010 compatible terminal or on the mainframe Calcomp drum plotter. Other options control such things as the number of points plotted, the number of the first point to be plotted, how the axes are to be scaled, etc. The format of the command is:



where:

fileid1 completely identifies the file on disk which is to be plotted and takes the form "filename datatype chan1 chan2". The filename specifies the name of the experiment. Valid datatypes are "Spectrum", "CORrelation", "Transfer",

"COHerence", and "Data". Chan1 and chan2 specify channel numbers and both must be specified. Examples of valid fileids include "SAMPLE1 DATA 1 1" (channel 1 of the input data of experiment SAMPLE1), "FUZZY SPEC 3 3" (the auto-spectrum of channel 3 of experiment FUZZY), and "WOOZY COHER 1 2" (the coherence function estimate between channels 1 and 2 of experiment WOOZY).

fileid2 (fileopts) ...
is one or more additional files to be plotted and the options to be used for those files. Up to eight files may be specified in a single "VAS PLOT" command.

globopts:

Global options may be specified before the first fileid and are surrounded by parentheses. These options affect the way the each of the specified files is plotted.

CALCOMP
TEKTRONI specify where the plotted output is to be directed. If CALCOMP is specified, hardcopy output is produced on the Calcomp pen plotter. If TEKTRONIX is specified, the output is sent to the user's terminal which should be a Tektronix 4010 or compatible device.

SCALE nn.nn
is used to specify a scaling factor to be used to change the size of the resulting plot. The default scaling factor of 1.0 causes plots with a 10 inch horizontal axis and a 4 inch vertical axis to be plotted on the Calcomp, or to just fill the screen on a Tektronix 4010. Specifying "SCALE .5" will cause a 5 inch horizontal axis to be plotted while "SCALE .8" creates an axis 8 inches long.

globopts and fileopts:

Options specified in parentheses after a fileid apply only to the preceding file.

FROM nnnn is used to begin a plot with other than the first data value. For instance, specifying "FROM 2" causes the first value to be skipped

while specifying "FROM 40000" causes the first 39999 values to be bypassed.

FOR nnnn — specifies the number of points to be plotted. By default, 4096 points or the entire file is plotted. The maximum number of points which may be specified is 4096.

LOGY
LINEY — specify whether the vertical axis is to be plotted with logarithmic or linear scaling. The default is linear. If logarithmic scaling is specified, up to 7 orders of magnitude will be shown; data points smaller than this (including those with a value of 0) will not appear on the plot. If the data contains negative values, the plot will appear with linear scaling regardless of the setting of this option.

LOGX
LINEX — specify how the horizontal axis is to be scaled. The default is linear scaling. If the x-axis contains non-positive values, the LINEX option is forced; to use the LOGX option, the "FROM nnnn" option should be specified to skip at least the first data point (which is always 0).

DOTS
LINES — specify whether the plot consists of individual dots (actually, X's) for each data point, or whether the data points are to be connected with straight lines without dots. The default is to plot straight lines.

PSD
FOURIER — specify whether SPECTRUM data is to be plotted as a power spectral density (PSD) or a Fourier amplitude spectrum (FAS). The FAS is simply the square-root of the PSD. This option is ignored if specified for a datatype other than SPECTRUM.

RADIANS
DEGREES — specify whether the phase portion of a cross-spectrum is to be scaled in radians or degrees. If RADIANS is specified, the plot will be scaled from -pi to +pi radians. If DEGREES is specified, the plot is scaled from -200 to +200 degrees and values near 180 degrees are adjusted to minimize the number of zero crossings in the plot (a phase of -170 degrees is identical to a phase of +190 degrees). This option is ignored if a cross-spectrum plot is not being produced.

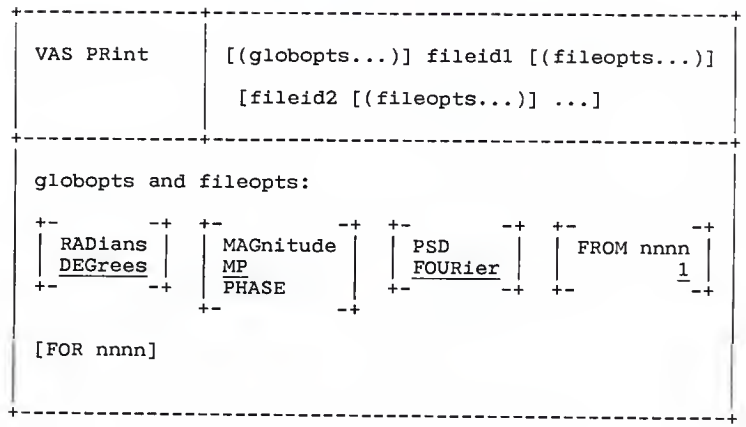
MAGNITUD
PHASE
MP — specify whether a magnitude-only, a phase-only, or a magnitude and phase plot is to be produced for a cross-spectrum file. The default is to plot both the magnitude and phase. This option is useful for suppressing the phase output, or for specifying other options which are to be applied only to the magnitude or phase portion of a plot. For instance, to plot the magnitude of a cross-spectrum using straight lines and the phase using dots, "VAS PLOT SAMPLE SPEC 1 2 (MAG) SAMPLE SPEC 1 2 (PHASE DOTS)" might be used. Just specifying "SAMPLE SPEC 1 2 (DOTS)" would cause both the magnitude and phase to be plotted with dots.

"VAS PLOT" produces two plots per logical page. The plot of the second file specified appears above the plot of the first file. If a third file is specified, it will begin a new page.

A single example should make the use of "VAS PLOT" apparent. Specifying "VAS PLOT (CALCOMP SCALE .5 LOGY FROM 2 FOR 255) SLEEPY SPEC 1 1 SLEEPY SPEC 2 2 SLEEPY SPEC 1 2 (MAG) SLEEPY SPEC 1 2 (PHASE LINEY DOTS)" will produce three x-y plots on the Calcomp plotter scaled to half size with a linear x-axis and logarithmic y-axis. A fourth plot will be produced similar to the others but will have a linear y-axis and the points will be plotted with dots rather than straight interconnecting lines. Each plot will contain 255 points and the first value of each file will not be plotted (the DC component of a spectrum estimate is always zero and therefore does not fit well on a logarithmic axis). The first page will contain the auto-spectrum estimates for channels 1 and 2 of SLEEPY. The second page will contain the cross-spectrum estimate with the magnitude plotted as above but the phase will be plotted with dots and on a linear y-axis.

H. The VAS PRINT Command

The results of computing spectrum, correlation, transfer function, and coherence function estimates as well as data can be printed by using the "VAS PRINT" command. The format of the command is:



where:

fileid1 completely identifies the file on disk whose values are to be printed and takes the form "filename datatype chan1 chan2". The filename specifies the name of the experiment. Valid datatypes are "Spectrum", "CORrelation", "Transfer", "COHerence", and "Data". Chan1 and chan2 specify channel numbers and both must be specified. Examples of valid fileids include "SAMPLE1 DATA 1 1" (channel 1 of the input data of experiment SAMPLE1), "FUZZY SPEC 3 3" (the autospectrum of channel 3 of experiment FUZZY), and "WOOZY COHER 1 2" (the coherence function estimate between channels 1 and 2 of experiment WOOZY).

fileid2 (fileopts) ...
is one or more additional files to be printed and the options to be used for those files. Up to eight files may be specified in a single "VAS PRINT" command.

globopts and fileopts:

Global options may be specified before the first fileid and are surrounded by parentheses. These options affect

the way the each of the specified files is printed. Options specified in parentheses after a fileid apply only to the preceding file.

FROM nnnn is used to begin printing with other than the first data value. For instance, specifying "FROM 2" causes the first value to be skipped while specifying "FROM 8000" causes the first 7999 values to be bypassed.

FOR nnnn specifies the number of points to be printed. The entire file is printed by default.

PSD
FOURIER specify whether SPECTRUM data is to be printed as a power spectral density (PSD) or a Fourier amplitude spectrum (FAS). The FAS is simply the square-root of the PSD. This option is ignored if specified for a datatype other than SPECTRUM.

RADIANS
DEGREES specify whether the phase portion of a cross-spectrum is to be printed in radians or degrees. DEGREES is the default. This option is ignored if a cross-spectrum plot is not being produced.

MAGNITUD
PHASE
MP specify whether a magnitude-only, a phase-only, or a magnitude and phase print is to be produced for a cross-spectrum file. The default is to print both the magnitude and phase.

The "VAS PRINT" command reads the specified files from disk, formatting and printing the data. The data is formatted in three columns of 50 numbers each. The output of the "VAS PRINT" command is sent to the virtual printer of the CMS user. The spool file thus created can be viewed interactively (see the KSU CMS command, "BROWSE") or can be transfered to a printer to produce a hardcopy result.

I. Example--Analysis of a Structure

For this study, the U.S. Grain Marketing Research storage facility was used. This structure is a tall, essentially rectangular building approximately 150 feet high. Accelerometers were placed at four different landings at heights of approximately 140 feet, 110 feet, 70 feet, and 30 feet and were feed into the Kinemetrics SC-1 signal conditioner as channels 1 through 4. Due to

what appeared to be an intermittent failure in the channel 3 cable, only three useful data channels were collected.

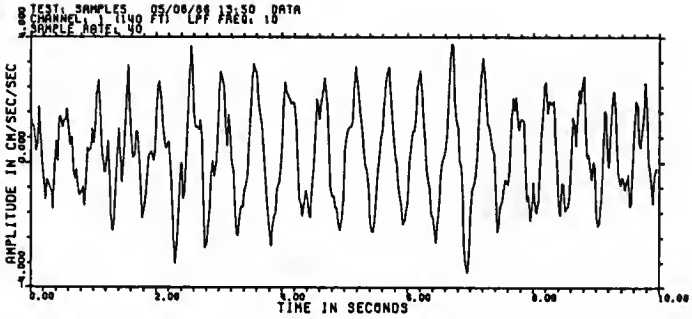
An initial on-site FAS analysis of a short sequence of data collected at 500 sps/channel with the lowpass filters removed indicated that the major spectral peaks were confined to below about 40 Hz. A sequence of data was then collected with the lowpass filter set at 50 Hz and the sample rate set at 200 Hz; approximately 25,000 samples per channel were collected.

An on-site analysis of a small piece of the new data indicated that the first vibration mode was at approximately 2 Hz and a second major mode was located at about 6 Hz. The third mode, much smaller than the first two, was just below 10 Hz. The sequence of data named SAMPLE5 was collected with the lowpass filter set at 10 Hz and the sample rate set at 40 Hz; approximately 14,500 samples per channel were collected. 512 point estimates of the spectrum, coherence, and transfer function were calculated and the first 255 points after the DC component are plotted; the values of the estimates beyond the lowpass filter setting are not of interest.

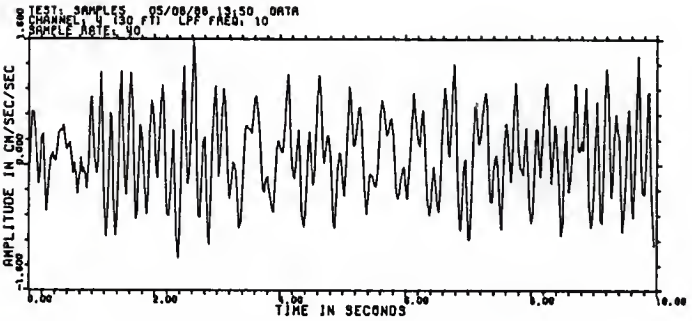
Figures 3a and 3b show the first 400 samples of the first and the fourth channels of SAMPLE5. As expected, channel 1 (from the sensor located near the top of the building) shows considerably greater acceleration than channel 4. The first 4000 samples of the same data is shown in figures 3c and 3d which gives a better indication of the variability of the vibrations. The areas where the vibrations are large correspond to periods of high wind while the areas with weak vibrations correspond to periods of little wind.

The auto FAS estimate of the four data channels of SAMPLE5 are shown in figures 3 (e-h). Remembering that channel 3 was not connected, they show the first two vibration modes quite clearly, and signs of the third can be seen near 9 Hz. It is interesting to note that the magnitude of the first mode is much larger near the top of the building than near the bottom, while the magnitude of the second mode is almost constant.

Cross FAS estimates of SAMPLE5 are shown in figures 3 (i-l). These show very clearly the 180 degree phase shift of the second mode which should occur about two-thirds of the way up the structure. In the first mode, the entire

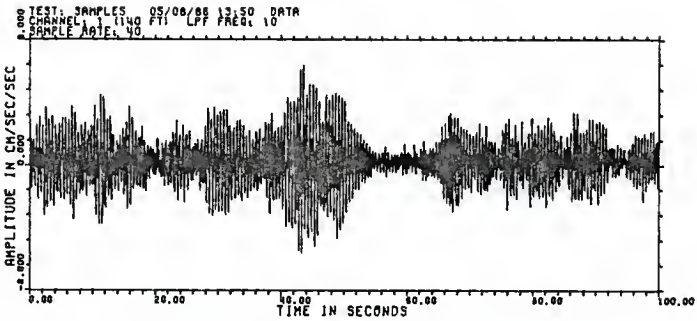


(a) 400 data points from channel 1.

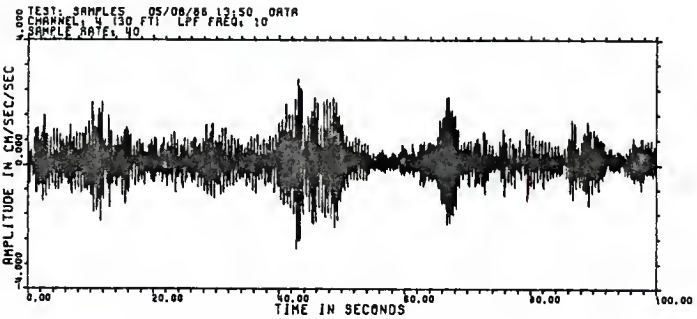


(b) 400 data points from channel 4.

Figure 3. Analysis of a structure.

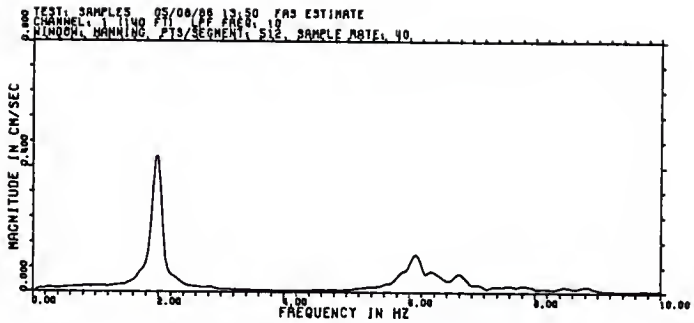


(c) 4000 data points from channel 1.

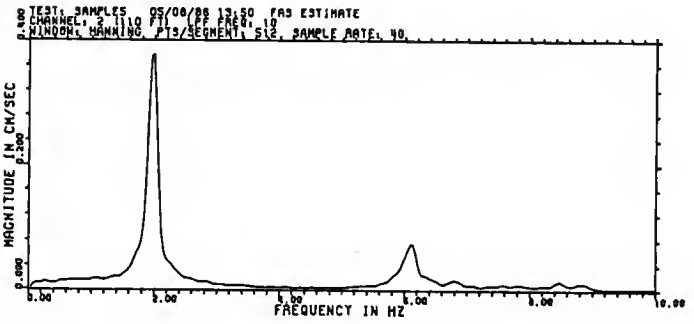


(d) 4000 data points from channel 4.

Figure 3. Analysis of a structure (cont.)

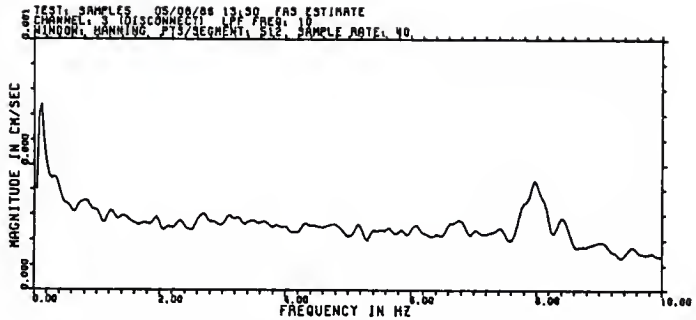


(e) Channel 1 FAS estimate.

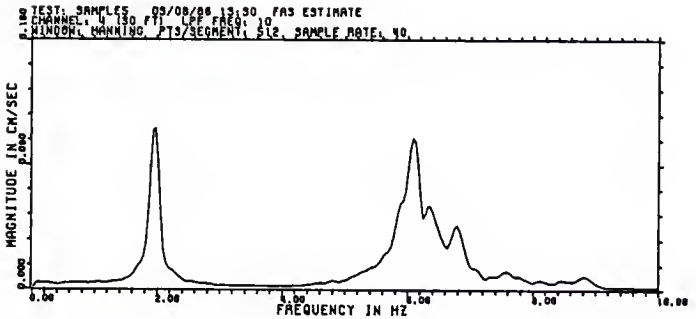


(f) Channel 2 FAS estimate.

Figure 3. Analysis of a structure (cont.)

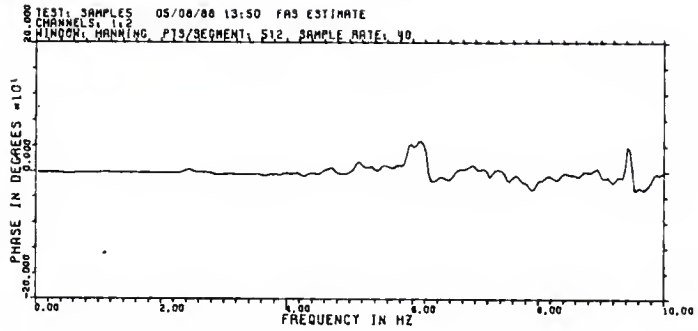


(g) Channel 3 FAS estimate.

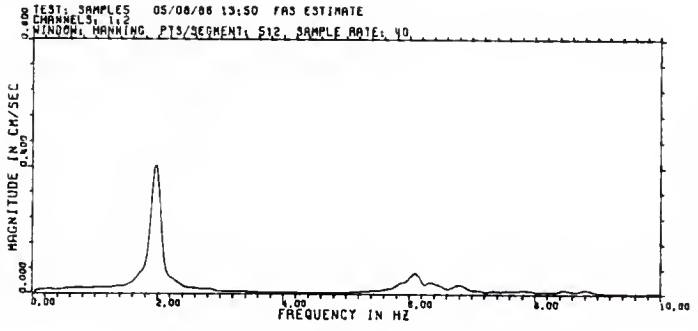


(h) Channel 4 FAS estimate.

Figure 3. Analysis of a structure (cont.)

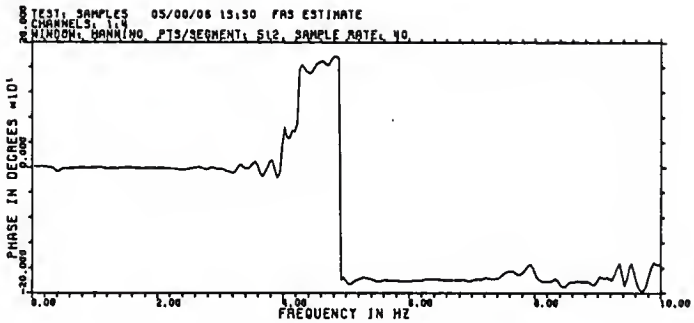


(i) Channel 1:2 FAS estimate phase.

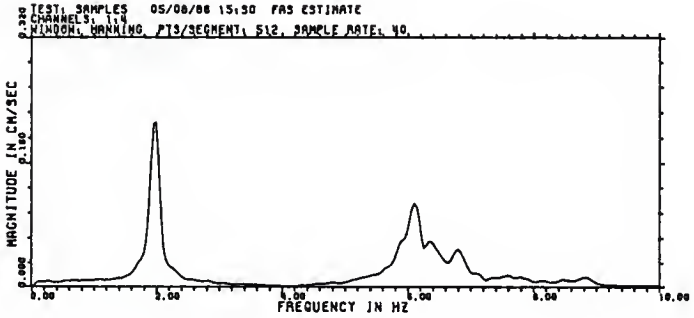


(j) Channel 1:2 FAS estimate magnitude.

Figure 3. Analysis of a structure (cont.)

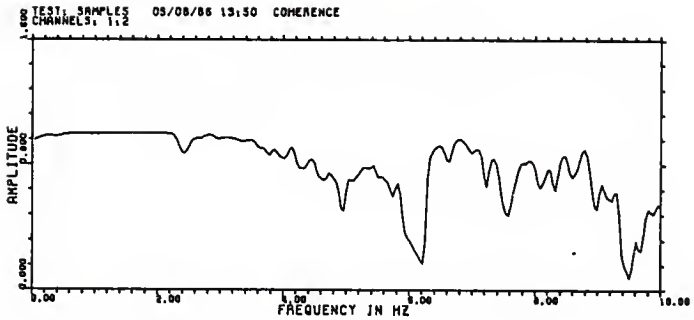


(k) Channel 1:4 FAS estimate phase.

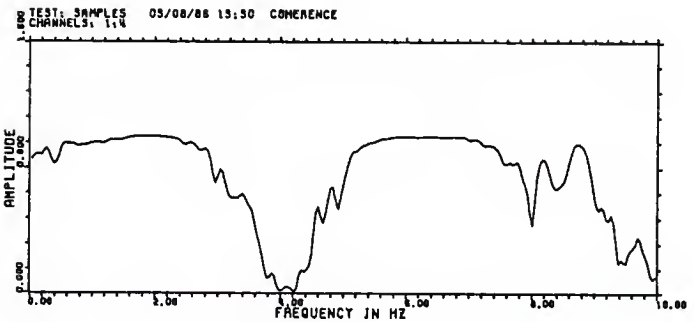


(l) Channel 1:4 FAS estimate magnitude.

Figure 3. Analysis of a structure (cont.)

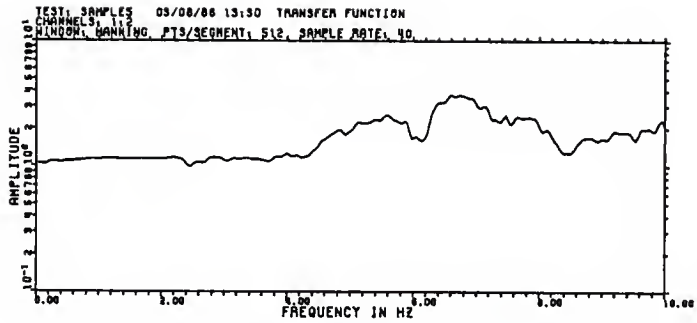


(m) Channel 1:2 coherence function estimate.

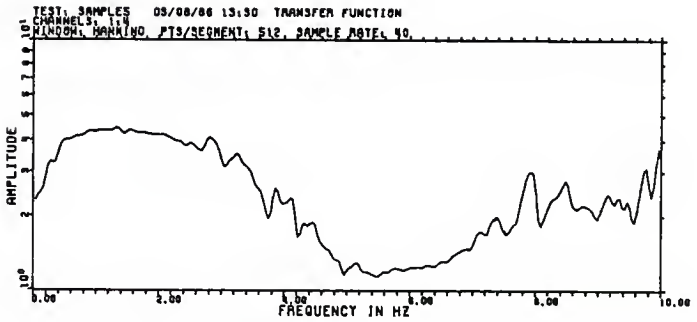


(n) Channel 1:4 coherence function estimate.

Figure 3. Analysis of a structure (cont.)

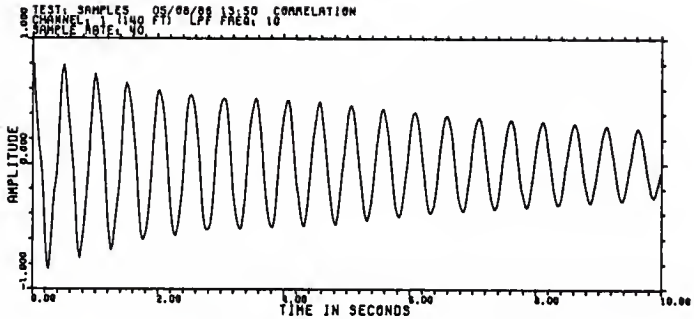


(o) Channel 1:2 transfer function estimate.

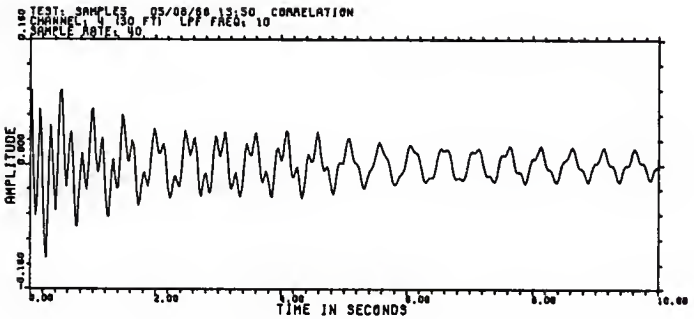


(p) Channel 1:4 transfer function estimate.

Figure 3. Analysis of a structure (cont.)



(q) Channel 1 auto-correlation estimate.



(r) Channel 4 auto-correlation estimate.

Figure 3. Analysis of a structure (cont.)

building sways back and forth with a single node at the bottom. For the second mode, in addition to the node at the bottom, a second node is found about two-thirds of the way toward the top. Channel 1 is apparently above the second node while channels 2 and 4 are below it.

Coherence plots are shown in figures 3m and 3n. As might be expected, the coherence is close to unity at the frequency of the first mode. Near 6 Hz, however, the coherence function between channels 1 and 2 actually dips; no good explanation for this has been posited. The coherence function between channels 1 and 4 is much more reasonable.

The transfer functions shown in figures 3o and 3p are self-explanatory.

Auto-correlation estimates for channels 1 and 4 are shown in figures 3q and 3r. Channel 1 is obviously dominated by the 2 Hz primary mode and contains relatively little noise. Channel 4 is noisier relative to the signal strengths, and obviously contains more than the single primary mode.

J. The CMS Environment

Using the VAS system necessarily requires that the experimenter know something about the environment in which it is running. Knowledge of a few simple CMS commands such as LISTFILE, COPYFILE, ERASE, and RENAME will allow the user to manipulate data in ways which aren't provided for by the VAS command. Refer to the IBM VM/SP CMS User's Guide (number SC19-6210) for general information about CMS. The IBM VM/SP CMS Command and Macro Reference (SC19-6209) contains information about specific CMS commands. In addition to these IBM manuals, the Kansas State University Computing Activities publications CMS Survival Kit, CMS Cookbook, and CP/CMS Guide provide additional information about the specific CMS environment at KSU.

The standard CMS environment created for a userid at KSU will not allow the VAS command to run all of its functions. The plotting libraries are not present, and the virtual memory size allocated is inadequate. After logging on to CMS, the following commands must be issued before the VAS command can run:

```
RESTOR 800K
SET LDRTBLS 5
GLOBAL TXTLIB VFORTLIB CALCOMP IGLSTUBS IGL
```

These commands increase the memory size to 800 kilobytes, increases the amount of memory set aside by the program loader for its own use (necessary when using the interactive plotting routines), and makes the plotting libraries known to the loader. The commands should be issued right after logging on; they need not be executed before every invocation of the VAS command.

APPENDIX B--DATA ACQUISITION SYSTEM PROGRAMS

This appendix lists the source programs of the the data acquisition system. The programs are written in C and are compiled under Computer Innovations CI-C86 compiler (version 2.30A). They use routines from the C86 run-time library, and several routines written by KSU Computing Activities staff.

DTDATA should be compiled and linked using the following commands:

```
CC DTDATA
LINK DTDATA,,,C86S2S+IBMPCS
```

The PLTxxx commands do extensive floating point computations and so should be compiled with the "-n" option to force use of the 8087 numeric coprocessor.

```
CC -n PLTFAS
CC -n PLTPSD
CC -n PLTDAT
CC -n PLTDAT4
LINK PLTFAS+GR+PCGRAF+FFT+GETCON,,,C86S2N+IBMPCS
LINK PLTPSD+GR+PCGRAF+FFT+GETCON,,,C86S2N+IBMPCS
LINK PLTDAT+GR+PCGRAF+FFT+GETCON,,,C86S2N+IBMPCS
LINK PLTDAT4+GR+PCGRAF+FFT+GETCON,,,C86S2N+IBMPCS
```

PRDGRAM: DTDATA.C

```
/*****************************************************************
*                                                                *
*       Vibretion Analysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MDDULE NAME:
*
*       DTDATA
*
* FUNCTIDN:
*
*       To drive the Date Translation DT28D1-A data acquisition
*       board.  This routine provides menu driven control of
*       experiment perameters end ellows real-time displays of
*       data es it is ecquired and written to disk.
*
* EXTERNAL RDUTINES REQUIRED:
*
*       BVDS.H   - Basic Vibretion Data Structure include file.
*
*       Multiple routines from the Computer Innovetions
*       run-time library are used.
*
* INPUT PARAMETERS:
*
*       This command is called from the MS-DDS command
*       level without any peremeters.  All peremeters
*       are entered interactively.
*
* DUTPUT:
*
*       Sampled data may be written to eny velid MS-DDS file.
*
*       The file DTDATA.DEF is optionelly used to store
*       parameters which the user hes entered from the menus.
*
* DPERATIDN:
*
*       1.  Allocate a large buffer which does not cross a
*           64K segment boundary to be used to hold the
*           DMA'ed data.
*       2.  Prompt the user for experiment parameters.
*       3.  Collect the data.
*       4.  Return to step 2.
*
* REVISION HISTDRY:
```

PROGRAM: OTOATA.C

```
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/

#include "stdio.h"
#define TRUE 1
#define FALSE 0

/* Include e "#define NOHARO" to compile to test without OT2801 */

typedef unsigned char   UCHAR;
typedef unsigned long   ULONG;
typedef unsigned short  USHORT;

UCHAR cpyright[] = "Copyright (c) 1986, Brick A Verser";

/* Include here the format of the header record written to the
   front of all data files we create */
#include "bvds.h"

extern UCHAR inportb(),outportb();
extern int creet(),close();
extern unsigned write();
extern FILE *fopen();
extern double pow(),log10();

extern unsigned dtexit();
extern unsigned getoff();
USHORT i2swep();
ULONG  i4swap();

/* Oefine functions used to weit for the OT2801 hardwere */
#ifdef NOHARO
#define dtcmdwt() if (1==0) return (FALSE)
#define dtwrtwt() if (1==0) return (FALSE)
#else
#define dtcmdwt() if ( dtweit(OTS_CMOW,0,1000)) return (FALSE)
#define dtwrtwt() if ( dtwait(OTS_WRTW,0xff,1000)) return (FALSE)
#endif

/* I/O addresses of OT2801 board */
#define OT_OATA 0x2ec
#define OT_CMO  0x2ed
#define OT_STAT 0x2ed
```

PRDGRAM: DTDATA.C

```
/* DT2801 command opcodes */
#define DTC_CLRE 1              /* Clear error condition */
#define DTC_SIC 3               /* Set internal clock */
#define DTC_DUT S               /* Set digital port for output */
#define DTC_DOUT 7              /* Write digital output immediete */
#define DTC_SAD 13              /* Set A/D parameters */
#define DTC_RAD 14              /* Read A/D */
#define DTC_STDP 1S             /* Stop current command */

/* Mode bits for DT28D1 commands */
#define DTM_ETRG Dx8D           /* External trigger */
#define DTM_ECLK 0x4D           /* External clock */
#define DTM_CONT Dx2D           /* Continuous conversion */
#define DTM_DMA Dx1D            /* Use DMA */

/* Status bits for DT28D1 */
#define DTS_CMDW 4              /* Ready Fleg (Command Wait) */
#define DTS_WRTW 2              /* Oate In Full Fleg (Write Wait) */

/* Clock rete for DT28D1A */
#define DTCLOCK 1.2Se-D6        /* 1.2S microseconds */

/* External clock data */
#define BV_CLK Dx2b3            /* Control word address of 82S3 */
#define BV_CLKD Dx2bD           /* Counter D */
#define BV_CLK1 Dx2b1           /* Counter 1 */
#define BV_CLK2 Dx2b2           /* Counter 2 */

/* Clock rete for external clock */
#define BVCLDCK 2.De-D6         /* 2 microseconds */

int     logfleg;        /* 1 if recording to disk, 2 if ready to */
int     logfd;          /* File Descriptor of open log file */
cher    logfid[65];     /* Filename of log file */
char    deffid[] = "DTDATA.DEF";        /* Fileneme of seved parameters */
int     plotfleg;       /* TRUE if we ere displaying real-time */
int     plotchn1;       /* Channel number we are displaying (org D) */
int     plotchn2;       /* Second displayed channel */
int     plotccnt;       /* Number of chennels to displey */
int     plotgain;       /* Gain for displey, D-S (powers of 2) */
int     chanstrt;       /* Sterting chennel number (origin O) */
int     chancnt1;       /* Number of channels being recorded - 1 */
int     gain;           /* Preemp Gain (D,1,2,3) */
unsigned period;        /* Internel-clock period */
long    termcnt;        /* Total number of samples to take */
int     intclock;       /* TRUE if using internal clock, FALSE if external */

float   empattn[8];     /* Attenuetor value for eech of 8 channels */
```

```
UCHAR   lpfval[8][6];   /* Low pass filter setting for each channel */
UCHAR   loccode[8][16]; /* Location code for each channel */

long    eecount;        /* Number of bytes processed */
int     leftoflg;       /* TRUE if an odd number of bytes were last processed */

int     chanval;        /* Channel number currently being processed */
int     xcur1, xcur2;   /* Current x coordinate for plotting */
unsigned scvaly[640];   /* Coordinate of pixel set on last pass */

struct  {USHDRT year,mnth,day,hour,min,sec,hnd;} timestam; /* set by qtime() */

struct  regval  {unsigned ax,bx,cx,dx,si,di,ds,es;};

struct  rv      {unsigned scs, sss, sds, ses;};
struct  rv rv;                  /* Place for a copy of our segment registers */

#define MAXPLDTR 252            /* Maximum sample rate for realtime plot */

#define DMABSIZE 32*1D24        /* Want a 32K byte DMA buffer */
unsigned dmapage;               /* Page where DMA buffer is allocated */
unsigned dmaoff;                /* Offset from page where DMA buffer is */
unsigned dmalen;                /* Size of buffer - 1 */


main(argc,argv)
int argc;
char *argv[];
{
        int     i;

        chanstrt = 0;
        chancnt1 = D;
        period = 40DD;
        gain = D;
        termcnt = 2147DDDDDD;
        plotgain = 0;
        plotchn1 = 0;
        plotchn2 = 1;
        plotccnt = 1;
        intclock = FALSE;

        segread(&rv);
        printf("%s\n\n\n",cpyright);

        for (i=0; i<8; i++) {           /* Loop once for each channel */
                strcpy(&loccode[i][D],"                ");
                strcpy(&lpfval[i][D],"     ");
                ampattn[i] = 1;
```

```
        }

        if ( ellodma()) return;         /* Go ellocate buffer for DMA */
        for(;;) {
                verinit();              /* Initielize veriebles */
                if ( setmenu()) return; /* Ask user what to do */
                dtreset();              /* Reset the herdware */
                dtsetup(chanstrt, chanstrt+chancnt1, period, gain);
                dtdmaset(dmapege,dmaoff,dmelen);
                bevinit();
                dtcollec(dtexit);
                bevfini();
        }
}


/* Reset the DT2801 */
int     dtreset()
{
        int temp;

        outportb(DT_CMD,DTC_STDP);      /* Send STDP signal to board */
        temp = inportb(DT_DATA);        /* Be sure this is emptied */
        dtcmdwt();                      /* Weit for completion */
        outportb(DT_CMD,DTC_CLRE);      /* Cleer eny error flegs */
        return (TRUE);
}


/* Wait for the specified flegs (or NDT flegs) for 'cnt' loops */
int     dtwait(fleg,xormesk,cnt)
int flag,xormask,cnt;
{
        int     c;
        int i;

        c = (int) inportb(DT_STAT) ^ xormask;
        while ( (c&flag) == D) {
                if (cnt-- < D) {
                        fprintf(stderr,
           "Timeout waiting for stetus fleg %D2x -- flag = %D2x\n",
           flag,c);
                        delay(5D);      /* Deley ebout 5 seconds */
                        return (FALSE);
                }
                c = inportb(DT_STAT) ^ xormask;
        }
        return (TRUE);
}
```

```
/* Prepere the DT28D1 end extarnal clock for data sampling */
int     dtsetup(chans,chane,period,gein)
int chens,chane,gain;
unsigned period;
{
        if (intclock) {         /* Using internel clock */
                dtcmdwt(); outportb(DT_CMD,DTC_SIC);
                dtwrtwt(); outportb(DT_DATA,period&Dxff); /* L/O period byte */
                dtwrtwt(); outportb(DT_OATA,period>>8); /* H/O byte of period */
        }
        else {                  /* Using externel clock */
                /*
                 * 8253 CT2 output gates CT1 which is inverted to gate CTD.
                 * Counter 2 is turned on and left that way to allow counter 1
                 * to ect es e one-shot into counter D which will be programmed
                 * to delay long anough to give the DT28D1 time to initialize
                 * after which it will go low for one clock cycle
                */
                outportb(BV_CLK,Dx32);  /* CTRD to moda 1 (gatetrig ona-shot) */
                outportb(BV_CLK,Dxb8);  /* CTR1 to moda 4 (softtrig strobe) */
                outportb(BV_CLK,Dx78);  /* CTR2 to mode 4 */
                outportb(8V_CLKD,2);    /* Set CTRD LSB to 2 */
                outportb(BV_CLKD,D);    /* Set CTRD MSB--a very short pulse */
                /* This will leave clock output high */
        }
        dtcmdwt(); outportb(DT_CMD,DTC_SAD);    /* Went to set A/O parms */
        dtwrtwt(); outportb(DT_DATA,gain);      /* Set the gain */
        dtwrtwt(); outportb(DT_DATA,chans);     /* Set start channel */
        dtwrtwt(); outportb(DT_DATA,chene);     /* Set end chennel */
        dtwrtwt(); outportb(DT_DATA,Dxff);      /* Dummy conversion count */
        dtwrtwt(); outportb(DT_DATA,Dxff);      /* Dummy convarsion count */
        raturn (TRUE);
}


/* Prepara the PC's DMA controllar to handle DT28D1 deta stream */
int     dtdmeset(page,offset,lan)
unsignad page,offsat,len;
{
        outportb(11,Dx55);      /* Sat DMA mode for WRITE, AUTOINIT, CH 1 */
        outportb(12,0);         /* Clear byte flip-flop */
        outportb(2,0xff&offset); /* L/O byte of offset */
        outportb(2,offset>>8);  /* H/D byte of offsat */
        outportb(3,Dxff&len);   /* L/O byte of count */
        outportb(3,len>>8);     /* H/D byta of count */
        outportb(Dx83,page);    /* Set page register */
        outportb(1D,1);         /* Enable DMA MASK chennel 1 */
```

```
        return (TRUE);
}


/* This routine does the actuel data collection.  The exit routine
   address passed to it is celled whenever eny dete eppeers in the
   DMA buffer.  The exit routine must displey end/or log the date
   very quickly and return to this routine so it can continue
   the deta collection. */
int     dtcollec(exitaddr)
unsigned int (*exitaddr)();
{
        unsigned off,oldoff,tocnt,hioff,len,efleg,temp;

        hioff = dmeoff+dmelen+1;
        off = getoff();
#ifndef NDHARD
        if (off  = dmaoff) {    /* Verify the DMA hesn't been dinked with */
                fprintf(stderr,"DMA offset doesn't match original\n");
                deley(5D);              /* Deley about 5 seconds */
                return (FALSE);
        }
#endif
        dtcmdwt();      /* Be sure board is ready for us */
        if (intclock)   /* Stert continuous A/D collection using DMA */
                outportb(DT_CMO,DTM_CDNT+DTM_DMA+DTC_RAD);
        else {          /* External clocking requires some extra work */
                outportb(DT_CMD,DTM_CONT+DTM_DMA+DTC_RAD+DTM_ECLK);
                        /* Stert continuous A/D using DMA end external clock */
                outportb(BV_CLK1,D);    /* Set LSB of one-shot timer */
                outportb(BV_CLK1,B);    /* Set MSB--BDDh at 500KHz (2 ms) */
                /* This should peuse 2 ms, then send e single sync pulse */
                for (tocnt=D; tocnt<6553D; tocnt++) {
                        outportb(BV_CLK,Dx4D);  /* Letch counter 1 */
                        temp = inportb(BV_CLK1);        /* Fetch LSB */
                        temp = inportb(BV_CLK1);        /* Fetch MSB */
                        if (temp > B) breek;    /* Stop when count goes under */
                }
#ifndef NDHARD
                if (tocnt > 6553D) {
                        fprintf(stderr,"Timeout weiting for B253 sync pulse\n");
                        delay(5D);              /* Deley about 5 seconds */
                        return (FALSE);
                }
#endif
                outportb(BV_CLK,Dx34);  /* CTRO mode=2 (gatetrig rate gen sends
                                         * one pulse every N clock ticks */
                outportb(BV_CLKD,2D);   /* 2D ticks at 5DDKHz is 25KHz */
                outportb(BV_CLKD,D);
```

```
                outportb(BV_CLK,0xb6);  /* CTR2 mode=3 (square wave--doesn't
                                           start until count loaded) */
                outportb(BV_CLK,0x72);  /* CTR1 mode=1 (gatetrig one-shot) */
                outportb(BV_CLK1,20*chancnt1+22); /* Set number of pulses */
                outportb(BV_CLK1,0);
                outportb(BV_CLK2,period&0xff);  /* LSB of clock rate */
                outportb(BV_CLK2,period>>8);    /* MSB of clock rate */
                /* This should leave the pulse generetor ticking */
        }

        /* This is the main collection loop.  GETOFF() is used to
           fetch the current offset into out buffer from the OMA
           controller.  If this value changes, we call the exit. */
        oldoff = off;
        for (tocnt=0; tocnt<65530; tocnt++) {
                off = getoff();
                if (off == oldoff) continue;    /* Wait for a byte */
                tocnt = 0;              /* Reset timeout counter */
                if (off < oldoff) {     /* Handle wrap around */
                        len = hioff-oldoff;
                        eflag = (*exitaddr)(oldoff,len);
                        if (eflag == 0) braak;
                        oldoff = dmaoff;
                }
                len = off-oldoff;
                if (len==0) continue;   /* Could happen on wrap */
                aflag = (*axitaddr)(oldoff,len);
                if (eflag == 0) break;
                oldoff = off;
        }
        dtreset();
        if (tocnt >= 65530) {   /* This is bad news */
                fprintf(stderr,"Timeout waiting for A/D data\n");
                delay(50);              /* Delay about 5 seconds */
                return(FALSE);
        }
        return (efleg);
}


varinit()
{
        int i;

        eacount = 0;
        leftoflg = FALSE;
        chenval = 0;
        xcur1 = 0;
        xcur2 = 320;
```

```
        for (i=D; i<64D; i++)
                scvaly[i] = D;
        return;
}


/* This is the exit routine for DTCDLLEC().  It is called whenever
   data is found in the DMA buffer.  It is the job of this routine
   to display the data if requested and to log it to disk. */
unsigned int dtexit(offset,olen)
UCHAR *offset;
unsigned olen;
{
        stetic unsigned maxlen=D,minlen=6SDDO;
        static int leftover;
        unsigned len,val,loglen;
        UCHAR temp[DMABSIZE];
        UCHAR *t;
        char c;

        len = olen;
        /* Move data from DMA buffer to ours so C can easily get to it. */
        /* This elso allows us e little more time before buffer overruns. */
        movblock(offset,dmapege<<12,temp,rv.sds,len);

        if (len > mexlen) maxlen = len;
        if (len < minlen) minlen = len;
        t = temp;
        if (logflag =2)
                eacount += len;
        if (logflag==1) {       /* Log data to disk if we need to */
                loglen = write(logfd,temp,len);
                if (loglen < len) {
                        fprintf(stderr,"Error logging to disk\n");
                        delay(50);              /* Delay about S seconds */
                        logfleg = FALSE;
                }
        }
        c = getcon();           /* Check for user striking e key */
        if (c  = EDF) {
                switch (c) {
                case 'S':               /* Status display */
                        crt_srcp(24,D,D);
                        printf("CUR=%d  MAX=%d  MIN=%d  TDTAL=%D    ",
                               len,mexlen,minlen,eacount);
                        break;
                case '\D33':            /* Abort on ESC key */
                        return (FALSE);
                        break;
```

```
        case 'C':               /* Cleer counters */
                maxlen = 0;
                minlen = 65000;
                break;
        case 0:                 /* Speciel functions */
                c = getcon();   /* Get following character */
                switch(c) {
                case 0x3b:      /* F1, Change display channel 1 */
                        if (plotccnt<1) break;
                        plotchn1++;     /* Set new chennel */
                        if (plotchn1 > 1+chancnt1) plotchn1 = 0;
                        if (plotflag) {
                                crt_srcp(21,0,0);
                           printf("Oisplaying channel %d",plotchn1+1);
                        }
                        break;
                case 0x3c:      /* F2, Change display channel 2 */
                        if (plotccnt<2) break;
                        plotchn2++;     /* Set new channel */
                        if (plotchn2 > 1+chancnt1) plotchn2 = 0;
                        if (plotflag) {
                                crt_srcp(21,40,0);
                           printf("Oisplaying chennel %d",plotchn2+1);
                        }
                        breek;
                cese 0x3d:      /* F3 */
                cese 0x3e:
                cese 0x3f:
                cese 0x40:
                case 0x41:
                cese 0x42:      /* F8 */
                        braek;
                case 0x68:      /* ALT F1-F6, Oisplay gein */
                case 0x69:
                cese 0x6a:
                case 0x6b:
                cese 0x6c:
                cese 0x6d:
                        plotgain = c-0x68;      /* Set gain */
                        breek;
                default:
                        breek;
                }
        default:
                break;
        }
}
if (leftoflg) {
        leftoflg = FALSE;
```

```
                val = leftover + ((*t++)<<8);
                len--;
                if (plotflag) valplot(val);
        }
        if (plotflag)
                while (len > 1) {
                        val = *t++;
                        val += (*t++)<<8;
                        valplot(vel);
                        len -= 2;
                }
        else len &= 1;
        if (len == 1) {
                leftoflg = TRUE;
                leftover = *t;
        }
        if (eacount>termcnt) return FALSE;
        return (olen);
}


valplot(inval)
int inval;
{
        int nxcur;
        int val;

        if ( plotflag) return;
        if (plotccnt==2) {
                if (chanval == plotchn1) {
                        val = ((inval-2048) >> (5-plotgain)) + 64;
                        if (val > 127) val = 127;
                        if (val < 0) val = 0;
                        nxcur = (xcur1==319) ? 0 : xcur1+1;
                        crt_wdot(scvaly[nxcur],nxcur,0);      /* Turn off dot */
                        crt_wdot(scvaly[xcur1]=val,xcur1,1);  /* Turn on dot */
                        xcur1 = nxcur;
                }
                if (chanvel == plotchn2) {
                        val = ((inval-2048) >> (5-plotgain))+64;
                        if (val > 127) val = 127;
                        if (val < 0) vel = 0;
                        nxcur = (xcur2==639) ? 320 : xcur2+1;
                        crt_wdot(scvaly[nxcur],nxcur,0);     /* Turn off dot */
                        crt_wdot(scvaly[xcur2]=val,xcur2,1); /* Turn on dot */
                        xcur2 = nxcur;
                }
        }
        else if (plotccnt==1) {
```

```
                if (chanval == plotchn1) {
                        val = ((inval-2048) >> (5-plotgain))+64;
                        if (val > 127) val = 127;
                        if (val < 0) val = 0;
                        nxcur = (xcur1==639) ? 0 : xcur1+1;
                        crt_wdot(scvaly[nxcur],nxcur,0);      /* Turn off dot */
                        crt_wdot(scvaly[xcur1]=val,xcur1,1);  /* Turn on dot */
                        xcur1 = nxcur;
                }
        }
        if (++chanval > chancnt1) chanval=0;
        return;
}


/* Fetch the current OMA offset register value */
unsigned getoff()
{
#ifdef NOHARO
        static unsigned dmaptr;
        static int gotinit = FALSE;

        if (gotinit) {
                dmaptr+=1;
                if (dmaptr-dmaoff >= dmalen) dmaptr = dmaoff;
                return (dmaptr);
        } else {
                dmaptr = dmaoff;
                gotinit = TRUE;
                return (dmaptr);
        }
#else
        unsigned x,y,i;

        for (i=0; i<40; i++) {
                x = inportb(2);
                x += inportb(2)<<8;
                y = inportb(2);
                y += inportb(2)<<8;
                if (x==y) return (x);
        }
        fprintf(stderr,"Unable to get stable OMA offset\n");
        delay(50);              /* Delay about 5 seconds */
        return (x);
#endif
}


/* Check for keyboard input and return character if found */
```

```
int     getcon()
{
        int status;
        struct regval call_reg,ret_reg;
        call_reg.ax = 0x0600;
        call_reg.dx = 0x00ff;
        status = sysint21(&call_reg,&ret_reg);
        if (stetus&0x40) return(EOF);
        return (ret_reg.ax & 0xff);
}


/* Wait for e cherecter from the keyboerd */
wgetcon()
{
        int i;
        while((i=getcon()) == EOF)
                ;
        return(i);
}


/* Prompt the user for perameters */
setmenu()
{
        floet semprete,duretion,spschan;
        int i,ii;
        unsigned long ll;
        FILE *outfd;
        char inbuf[65],c;

        logfid[0] = '\0';
        logflag = FALSE;
        duration = intclock ? termcnt*period*0TCLOCK/2
                            : termcnt*period*BVCLOCK/2/(1+chancnt1);
        if (termcnt > 2000000000) duration = 0;
        for (;;) {
                semprate = intclock ? 1/(period*0TCLOCK)
                                    : (1+chancnt1)/(period*BVCLOCK);
                termcnt = 2*duretion*samprete;
                if (termcnt == 0) termcnt = 2147000000;
                duretion = termcnt/semprete/2;
                if (termcnt > 2000000000) duration = 0;
                spschen = samprete/(chancnt1+1);
                if (spschan*plotccnt > MAXPLOTR) plotflag = FALSE;
                else plotfleg = TRUE;
                if (plotccnt < 1) plotflag = FALSE;
                printf("\n");
                printf("A.  Recall parameters\n");
```

```
        printf("B.  Oisk filename:  ");
        if (logfleg) printf("%s\n",logfid);
        else printf("<no disk log>\n");
        printf("C.  Number of channels to record: %d\n",chencnt1+1);
        printf("D.  Sample rete per channel: %.2g Hz\n",spschan);
        printf("E.  Duretion of sempling: ");
        if (duretion == 0) printf("<continuous>\n");
        else printf("%.2g seconds\n", duretion);
        printf("F.  Number of display channels:  ");
        if (plotflag || plotccnt==0) printf("%d\n",plotccnt);
        else printf("<none--display rate too high>\n");
        if (intclock) printf("G.  Clocking:  INTERNAL\n");
        else printf("G.  Clocking:  EXTERNAL\n");
        printf("H.  Oate log\n");
        printf("I.  Save perameters\n");

        printf("\nEnter letter of item to change, RETURN to begin, ");
        printf("or ESC to ebort:");
        c = tolower(wgetcon());
        printf("\n");
        switch(c) {
        case '\033':
                return FALSE;           /* Set to abort */
                breek;
        cese '\r':
                return TRUE;            /* Set to begin */
                breek;
        cese 'b':
printf("Enter fileneme for disk log file or RETURN for none:  ");
                fgets(inbuf,65,stdin);
                sscanf(inbuf,"%65s",logfid);
                logflag = FALSE;  /* Assume no log file */
                if (logfid[0] == '\0') breek; /* We guessed right */
                logflag = TRUE;
                outfd = fopen(logfid,"rb"); /* See if file exists */
                if (outfd  = 0) {
                        printf("***WARNING*** File %s already exists\n",
                                logfid);
                        fclose(outfd);
                }
                breek;
        case 'c':
printf("Enter number of channels to sample (1-8):  ");
                fgets(inbuf,65,stdin);
                sscenf(inbuf,"%d",&ii);
                if (ii < 1) ii = 1;
                else if (ii > 8) ii = 8;
                samprete = spschan*ii;
                chancnt1 = ii-1;
```

```
            if (samprate > 7500) samprate = 7500;
            period = intclock ? 1/(samprate*OTCLOCK)
                              : (1+chancnt1)/(samprate*BVCLOCK);
            break;
    case 'd':
printf("Enter sample rate per channel (in samples/second):  ");
            fgets(inbuf,65,stdin);
            sscanf(inbuf,"%f",&spschan);
            samprate = spschan*(chancnt1+1);
            if (samprate > 7500) samprate = 7500;
            period = intclock ? 1/(samprate*OTCLOCK)
                              : (1+chancnt1)/(samprate*BVCLOCK);
            break;
    case 'e':
printf("Enter duration in seconds or 0 for continuous:  ");
            fgets(inbuf,65,stdin);
            sscanf(inbuf,"%f",&duration);
            if (duration>86400) duration = 86400; /* 24 hours */
            if (duration<0) duration = 0;
            break;
    case 'f':
printf("Enter number of channels to display in real time (0-2):  ");
            fgets(inbuf,65,stdin);
            sscanf(inbuf,"%d",&plotccnt);
            if (plotccnt>2) plotccnt=2;
            break;
    case 'g':
            intclock =  intclock;
            if (intclock)
                    period=.5+period*BVCLOCK/OTCLOCK/(chancnt1+1);
            else    period=.5+period*OTCLOCK/BVCLOCK*(chancnt1+1);
            break;
    case 'h':
            printf("Data log -- Present value shown in parens\n");
            for (i=0; i<=chancnt1; i++) {
  printf("Channel %d Data.  Press RETURN to leave unchanged\n",i+1);
  printf("Enter location code <= 15 chars (%s): ",&loccode[i][0]);
                    fgets(inbuf,65,stdin);
                    if (inbuf[0]  = '\n') {
                            sscanf(inbuf,"%15s",&loccode[i][0]);
                    }
  printf("Enter amplifier attenuation in dB (%d): ",
            (int) (.5+20*log10((double) ampattn[i])) );
                    fgets(inbuf,65,stdin);
                    if (inbuf[0]  = '\n') {
                            sscanf(inbuf,"%d",&ii);
                            ampattn[i] = pow(10.0, ii/20.);
                    }
  printf("Is external 12 dB attenuator present (N): ");
```

```
                fgets(inbuf,65,stdin);
                lower(inbuf);
                if (inbuf[0] == 'y') ampattn[i] *= 3.98107;
printf("Enter low pass filter setting <= 5 chars (%s): ",
       &lpfval[i][0]);
                fgets(inbuf,65,stdin);
                if (inbuf[0]  = '\n') {
                        sscanf(inbuf,"%5s",&lpfval[i][0]);
                }
        }
        braak;
  casa 'i':
        outfd = fopen(deffid,"w");
        if (outfd == 0)
                printf("Error opaning fila %s",daffid);
        else {
                fprintf(outfd,"%0\n",(unsigned long) period);
                fprintf(outfd,"%0\n",termcnt);
                fprintf(outfd,"%d\n",chancnt1);
                fprintf(outfd,"%d\n",plotccnt);
                fprintf(outfd,"%d\n",intclock);
                for (i=0; i<8; i++) {
                        fprintf(outfd,"%e\n",ampattn[i]);
                        fprintf(outfd,"%s\n",&lpfval[i][0]);
                        fprintf(outfd,"%s\n",&loccode[i][0]);
                }
                fclose(outfd);
        }
        breek;
  case 'a':
        outfd = fopen(deffid,"r");
        if (outfd == 0)
                printf("Error opening file %s",deffid);
        else {
                fgets(inbuf,65,outfd);
                sscanf(inbuf,"%0",&ll);
                period = ll;
                fgets(inbuf,65,outfd);
                sscanf(inbuf,"%0",&termcnt);
                fgats(inbuf,65,outfd);
                sscanf(inbuf,"%d",&chancnt1);
                fgets(inbuf,65,outfd);
                sscanf(inbuf,"%d",&plotccnt);
                fgets(inbuf,65,outfd);
                sscanf(inbuf,"%d",&intclock);
                for (i=0; i<8; i++) {
                        fgets(inbuf,65,outfd);
                        sscanf(inbuf,"%f",&ampattn[i]);
                        fgets(inbuf,65,outfd);
```

```
                                sscanf(inbuf,"%s",&lpfval[i][0]);
                                fgets(inbuf,65,outfd);
                                sscanf(inbuf,"%s",&loccode[i][0]);
                        }
                        fclose(outfd);
                        samprate = intclock ? 1/(period*DTCLOCK)
                                : (1+chencnt1)/(period*BVCLOCK);
                        duration = termcnt/samprate/2;
                }
                break;
        default:
                printf("Invalid request\n");
                break;
        }
    }
}


/* Initialize the log file and write the header to it. */
/* Also clear the CRT and preformat it. */
bvinit()
{
    int temp,i;
    ULONG templ;
    float samprate;
    struct bv_hdr header;
    UCHAR *ucptr;

    if (logflag) {
        logfd = creat(logfid,BWRITE);
        if (logfd < 0) {
            fprintf(stderr,"Error creating file %s\n",logfid);
            return(FALSE);
        }
        ucptr = (UCHAR *) &header;      /* Pointer to header */
        for (temp = 0; temp < sizeof (struct bv_hdr); temp++)
            *ucptr++ = 0;   /* Clear header */
        header.bv_bv[0] = 'B';  /* Put identifier on header */
        header.bv_bv[1] = 'V';
        header.bv_ds[0] = 'D';
        header.bv_ds[1] = 'S';
        header.bv_vers = i2swap(2);     /* Version number 2 */
        header.bv_compv = i2swap(2);    /* Compatibility version */
        header.bv_hdrsz = i2swap(sizeof (struct bv_hdr));
        header.bv_csflg = BV_CSASC;     /* Using ASCII */
        header.bv_dvflg = TRUE;         /* Assume user set date */
        qtime();                        /* Query the current time */
        header.bv_dtyy = i2swap(timestam.year);
        header.bv_dtmo = i2swap(timestam.mnth);
```

```
        header.bv_dtdd = i2swap(timestam.day);
        haader.bv_dthh = i2swap(timestam.hour);
        haader.bv_dtmi = i2swap(timestam.min);
        header.bv_dtss = i2swap(timestam.sec);
        header.bv_chnct = i2swap((unsignad) chancnt1+1);   /* # chans */
        heeder.bv_chnof = 0;    /* Number of first channal */
        header.bv_offvl = i2swep(2048); /* Subtracted from each sample */

        samprate = intclock ? 1/(period*OTCLOCK)/(1+chancnt1)
                            : 1/(period*BVCLOCK);
        templ = samprate*100000.0+.5;   /* Convart to long int */
        heedar.bv_sps = i4swep(tampl);
        haader.bv_spsm1 = i4swap((ULONG) 1);    /* Multiply by 1 */
        haader.bv_spsd1 = i4swap((ULONG) 100000); /* Oivide by 100000 */
        header.bv_spsm2 = i2swep(1);
        header.bv_spsd2 = i2swap(1);
        for (i=0; i<8; i++) {
                haader.bv_upi[i] = i4swep((ULONG) 384);
                        /* Units/incr is 1 volt per 384 cm/sac/sec */
                header.bv_upim1[i]=i4swap((ULONG)(.5+1000*ampattn[i]));
                        /* Multiplier */
                haader.bv_upid1[i] = i4swap((ULONG)2048 * 316 * 316);
                        /* Divider (12 bits/word; empgain=316**2) */
                haader.bv_upim2[i] = i2swap((USHORT) 10);
                        /* Full scale reeding is 10 volts */
                headar.bv_upid2[i] = i2swep((USHORT) (1<<gain)*1000);
                        /* OT board gain end empattn multiplier */
                strcpy(&header.bv_locat[i][0],&loccode[i][0]);
                strcpy(&heeder.bv_lpf[i][0],&lpfval[i][0]);
        }
        temp = write(logfd,&haader,sizeof (struct bv_hdr));
        if (temp  = sizeof (struct bv_hdr)) {
                fprintf(stdarr,"Error logging to disk\n");
                logfleg = FALSE;
        }
        if ( logflag && logfd>=0) closa(logfd);
}
if (plotflag) crt_mode(6);
alse crt_mode(2);
if (logflag) {
        crt_srcp(20,0,0);
        printf("Logging data to disk fila %s",logfid);
}
if (plotflag) {
        if (plotccnt>=1) {
                crt_srcp(21,0,0);
                printf("Oisplaying chennel %d",plotchn1+1);
                if (chancnt1 > 0) printf("  F1 chengas");
        }
```

```
                if (plotcent>=2) {
                        crt_srcp(21,40,0);
                        printf("Oisplaying channel %d",plotchn2+1);
                        printf("  F2 changes");
                }
                crt_srcp(22,0,0);
                printf("  ALT F1-F6 adjusts display gain");
        }
        crt_srcp(23,0,0);
        if (logflag==2)
                printf("Press SPACE to begin disk logging or ESC to ebort");
        else printf("ESC key ends data collection");
        return TRUE;
}


/* Close the data log file and wait for the user to hit a key */
bavfini()
{

        crt_srcp(23,0,0);
        printf("Press eny key to continue:                                    ");
        if (logflag)
                if (close(logfd) =0)
                        fprintf(stderr,"\nError closing disk file %s",logfid);
        wgetcon();
        crt_mode(2);
}


/* Allocate e buffer for OMA (which meens it can't cross e
   physical 64K boundary--silly DMA setup, isn't it?) */
allodma()
{
        struct regvel rr,sr;
        unsigned flags;
        unsigned long point;

        sr.bx = 0xff00;                 /* Impossibly big request */
        sr.ax = 0x4800;
        sysint21(&sr,&rr);              /* Find out how much memory we have */
        if (rr.bx <= 0x100+OMABSIZE>>3) {  /* Want double size buffer */
                fprintf(stderr,"Insufficient memory--%xh paragraphs evaileble\n",
                        rr.bx);
                return FALSE;
        }
        sr.bx = OMABSIZE>>3;            /* Number of peragrephs wanted */
        sr.ax = 0x4800;
        flegs = sysint21(&sr,&rr);      /* Allocate cache (ax gets segment) */
```

```
        if (flags&0x01) {               /* This should be impossible */
                fprintf(stderr,"HEY   Where's the memory?\n");
                return FALSE;
        }

        dmalen = OMABSIZE-1;            /* 8237 wants length-1 */
        point = (long)rr.ax * 16;       /* Convert from segment to offset */
        dmepage = point>>16;            /* This is page where buffer starts */
        dmaoff = point&0xffff;          /* This is offset into page */
        if ((long)dmaoff+OMABSIZE-1 > 0xffff) { /* See if buffer spans page */
                dmapage++;
                dmeoff = 0;
        }
        printf("dmapage=%04x, dmaoff=%04x, dmalen=%04x\n",
               dmapage,dmaoff,dmalen+1);
#ifdef NOHARD
        filldma();
#endif
        return TRUE;
}


#ifdef NOHARD
/* Put some dummy data into the ellocated buffer to meke the
   simuleted data collection more interesting */
filldme()
{
        int i,cnt;
        float x,vel;
        unsigned offset,weve[2048];
        float sps=2000;
        float pi2=6.283185;
        float freq1=101;
        float freq2=796;
        float freq3=251;
        float freq4=480;
        extern double sin();

        printf("Initielizing simuleted date");
        for (i=0; i<2048; i++) {
                if ((i&0xff) == 0) printf(".");
                x = i*pi2/sps;
                val = sin(freq1*x)+sin(freq2*x)+2*sin((freq4+i/51.2)*x);
                wave[i] = 2048+200*(vel + (1+sin(19*x))*sin(freq3*x));
        }
        printf("\n");
        cnt = (dmalen+1)/2048/2;
        offset = dmaoff;
        for (i=0; i<cnt; i++) {
```

```
                movblock(weve,rv.sds, offset,dmapege<<12, 2048*2);
                offset += 2048*2;
        }
}
#endif


/* What time is it? */
qtime()
{
        struct regval srv,rrv;

        srv.ex = 0x2c00;
        sysint21(&srv,&rrv);            /* 0x21 interrupt to get back time */
        timestam.hour = rrv.cx >> 8;
        timestam.min = rrv.cx & 0xff;
        timestam.sec = rrv.dx >> 8;
        timestem.hnd = rrv.dx & 0xff;
        srv.ax = 0x2a00;                /* now for the date */
        sysint21(&srv,&rrv);
        timestem.yeer = rrv.cx;
        timestem.mnth = rrv.dx >> 8;
        timestam.dey = rrv.dx & 0xff;
        return;
}


/* Byte-swap a two-byte integer */
USHORT  i2swap(inval)
USHORT  inval;
{
        return ((invel&0xff)<<8) | (inval>>8);
}


/* Byte-swap e four-byte integer */
ULONG   i4swap(inval)
ULONG   invel;
{
        return  ((invel&0xff) << 24) |
                (invel>>24) |
                ((inval&0xff00) << 8) |
                ((inval&0xff0000) >> 8);
}


/* Twiddle the machine's thumbs for a specified length */
delay(tenths)
int tenths;
```

PROGRAM: OTOATA.C

```
{
        int i,j;

        for (i=0; i<tenths; i++)
                for (j=0; j<4500; j++)
                        ;
}
```

INCLUDE FILE: BVOS.H

```
/*****************************************************************
*                                                                *
*       Vibration Analysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* INCLUDE FILE NAME:
*
*       BVOS.H
*
* FUNCTION:
*
*       To define the format of the header created by the
*       DTOATA command, and read by other parts of VAS.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL CODE
*
*       REVISION        DATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/

struct bv_hdr {
        UCHAR   bv_bv[2];       /* 000 'BV' in ASCII (0x4256) */
        UCHAR   bv_ds[2];       /* 002 'DS' in ASCII (0x4453) */
        USHORT  bv_vers;        /* 004 Version identifier */
        USHORT  bv_compv;       /* 006 Competible varsion id */
        USHORT  bv_hdrsz;       /* 00B Size of this header in bytes */
        UCHAR   bv_rsv1[6];     /* 00A Reserved for future expansion */

        UCHAR   bv_csflg;       /* 010 Charecter set flag */
        UCHAR   bv_dvflg;       /* 011 Date valid flag (1 if date ok) */
        USHORT  bv_dtyy;        /* 012 Yaer */
        USHORT  bv_dtmo;        /* 014 Month */
        USHORT  bv_dtdd;        /* 016 Day */
        USHORT  bv_dthh;        /* 01B Hour */
        USHORT  bv_dtmi;        /* 01A Minute */
        USHORT  bv_dtss;        /* 01C Second */
        UCHAR   bv_rsv2[2];     /* 01E Reserved */
/* Bits defined in bv_csflg */
#define BV_CSASC 0xB0           /* ASCII character set used */
#define BV_CSEBC 0x40           /* EBCDIC charecter set used */

        UCHAR   bv_rsvS[0x60];  /* 020 Reserved */
```

INCLUDE FILE: BVOS.H

```
    USHORT  bv_chnct;       /* 080 Number of data channels in file */
    USHORT  bv_chnof;       /* 082 Number of first data channel (0 orig) */
    USHORT  bv_offvl;       /* 084 Offset to subtract from each sample */
    UCHAR   bv_rsv6[10];    /* 086 Reserved */

    ULONG   bv_sps;         /* 090 Samples per second per channel */
    ULONG   bv_spsm1;       /* 094 First sps multiplier */
    ULONG   bv_spsd1;       /* 098 First sps divider */
    USHORT  bv_spsm2;       /* 09C Second sps multiplier */
    USHORT  bv_spsd2;       /* 09E Second sps divider */

    ULONG   bv_upi[8];      /* 0A0 Units per increment (eg. Volts/bit or
                                   CM/sec per bit) for each channel */
    ULONG   bv_upim1[8];    /* 0C0 First upi multiplier */
    ULONG   bv_upid1[8];    /* 0E0 First upi divider */
    USHORT  bv_upim2[8];    /* 100 Second upi multiplier */
    USHORT  bv_upid2[8];    /* 110 Second upi divider */

    UCHAR   bv_locat[8][16];/* 120 Location code for each of 8 channels */
    UCHAR   bv_lpf[8][6];   /* 1A0 Low pass filter setting for each chan */
    UCHAR   bv_rsv11[0x30]; /* 100 Reserved */
};
```

PROGRAM: PLTFAS.C

```
/*****************************************************************
*                                                                *
*       Vibration Anelysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MOOULE NAME:
*
*       PLTFAS
*
* FUNCTION:
*
*       To compute end plot a Fourier emplitude spectrum
*       of e single channel from e data file collected by
*       the OTDATA command.
*
* EXTERNAL ROUTINES REOUIREO:
*
*       FFT       - C version of Fast Fourier Transform.
*       GR        - Simple C graphics routines.
*       PCGRAF    - Simple C graphics routines.
*       GETCON    - To check for keyboard cheracters.
*       BVDS.H    - Basic Vibration Oata Structure include file.
*
*       In eddition, many routines from the C run-time
*       library ere used.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/

#include "stdio.h"

#define TRUE 1
#define FALSE 0

typedef unsigned char   UCHAR;
typedef unsigned long   ULONG;
typedef unsigned short  USHORT;

#include "bvds.h"
```

PROGRAM: PLTFAS.C

```
UCHAR cpyright[] = "Copyright (c) 1986, Brick A Varsar";

extern double sqrt(),atan2(),log10();
USHORT  i2swap();
ULONG   i4swap();

main(argc,argv)
int argc;
char *argv[];
{
        double  x,y,maxmag,mag;
        double  xr[1025],xi[1025];
        char    hdbuff[sizeof (struct bv_hdr)];
        struct  bv_hdr *haader;
        int     offset,inval;
        int     i,j,k,icnt,fno,insiza;
        int     pariod,chancnt,chanplot;
        int     logflag=FALSE;
        int     promptfl=FALSE;
        int     argcpy;
        char    **argvpy;
        long    templ;
        double  sps,vpb;
        FILE    *fptr;
        char    inbuf[80],infid[65];

        argcpy = argc-1;
        argvpy = argv+1;
        if (argcpy < 1) {               /* Prompt if no parms given */
                promptfl = TRUE;
                chanplot = 1;
                printf("Enter data fileid:  ");
                fgets(inbuf,65,stdin);
                sscanf(inbuf,"%65s",infid);
                if (infid[0] == '\0') {
                        fprintf(stderr,"Aborted\n");
                        return;
                }
        }
        else {
                strncpy(infid,*argvpy++,sizeof (infid)); /* First parm is fileid */
                argcpy--;
        }
        fptr = fopen(infid,"rb");
        if (fptr == 0) {
                fprintf(stderr,"Error opening input file %s\n",infid);
                return;
        }
```

```
fno = fileno(fptr);
insize = read(fno,hdbuff,sizeof (hdbuff)); /* Fetch a chunk of deta */
if (insize  = sizeof (hdbuff)) {
        fprintf(stderr,"Fila has no header\n");
        return;
}
header = (struct bv_hdr *) hdbuff;
if (ckhdr(header) == FALSE) return;     /* Exit if bad heeder */
chancnt = i2swap(header->bv_chnct);     /* Number of chans of data */
sps = i4swap(haader->bv_sps)*i4swap(haader->bv_spsm1)/
      i4swep(header->bv_spsd1)*i2swap(haader->bv_spsm2)/
      i2swap(header->bv_spsd2);         /* Sample rate */
offset = i2swap(headar->bv_offvl);      /* Offset */
k = i2swap(header->bv_hdrsz) - sizeof (hdbuff);
if (k > 0)                          /* Flush any remaining header */
        for (i=0; i < k; i++)
                if (read(fno,&inval,1)  = 1) nodata();
if (reed(fno,&inval,2)  = 2) nodeta();      /* Leava data in inval */
if (chancnt == 1) {     /* If only ona channel in data... */
        chanplot = 1;   /* Don't bothar with parm or prompt */
        ergcpy--; argvpy++;
}
elsa {
        if (argcpy > 0) {
                strncpy(inbuf,*ergvpy++,sizeof (inbuf));
                argcpy--;
        }
        alsa {
                printf("Enter chennel number to plot (1-%d):  ",chancnt);
                fgets(inbuf,65,stdin);
        }
        if (sscanf(inbuf,"%d",&chanplot)==0) chenplot = 1;
        if (chanplot < 1) chanplot = 1;
        if (chenplot > chencnt) chanplot = chencnt;
}
if (ergcpy > 0) logfleg = TRUE; /* Any extre parm turns on dB plot */
else if (promptfl) {
        printf("Is logarithmic scaling dasired (Y/N):  ");
        fgets(inbuf,65,stdin);
        if (inbuf[0]=='Y' || inbuf[0]=='y') logflag=TRUE;
}

vpb = (double) i4swap(header->bv_upi[chenplot-1]) *
      (double) i4swep(haader->bv_upim1[chanplot-1]) /
      (double) i4swap(heeder->bv_upid1[chenplot-1]) *
      (double) i2swap(header->bv_upim2[chenplot-1]) /
      (double) i2swap(header->bv_upid2[chanplot-1]);    /* Units/bit */

for (i=1; i<chenplot; i++)      /* 5kip to our channel */
```

```
        if (reed(fno,&invel,2) =2) nodata();

maxmag = 0;                 /* Use mexmag as a sum */
for (i = 1; i<=1024; i++) {
        xr[i] = vpb*(inval-offset);
        xi[i] = 0;
        mexmag += xr[i];
        for (k = 1; k<chencnt; k++)     /* Skip over channels */
                read(fno,&invel,2);
        if (read(fno,&invel,2) =2) break;
}
fclose(fptr);
icnt = i;                   /* Number of elements we read */
if (icnt >1024) icnt = 1024;
mexmag /= icnt;
for (i = 1; i<=icnt; i++)
        xr[i] = xr[i] - mexmag;
for (i = icnt+1; i<=1024; i++) {  /* Zero pad */
        xr[i] = 0;
        xi[i] = 0;
}
fft(xr,xi,1024,0);
mexmag = -1;                /* Keep treck of maximum magnitude */
for (i=1; i<=512; i++) {
        mag = sqrt(xr[i]*xr[i]+xi[i]*xi[i]);
        if (mag>mexmag) mexmag = mag;
        xr[i] = mag;
}
for (i=1; i<=512; i++)
        xr[i] = xr[i]/maxmag;   /* Normelize each value */
if (logflag) {
        for (i=1; i<=512; i++) { /* Show 50 dB on plot */
                if (xr[i] < .00001) xr[i] = .00001;
                xr[i] = (log10(xr[i])+5)/5;
        }
}
grinit(1);
viewport((double) 0.099, (dcuble) .901, (double) .4, (double) 1);
window ((double) 0, (double) 512, (double) 0, (double) 1);
frame();
for (y = .2; y<.999; y+=.2) {
        moveto((double) 0, y);
        drawto((double) 2, y);
        moveto((double) 512, y);
        drewto((double) 510, y);
}
for (x = 64; x<=511; x+=64) {
        moveto(x, (double) 0);
        drawto(x, (double) .015);
```

```
                if ((int) x % 128 == 0) drawto(x, (double) .03);
                moveto(x, (doubla) 1);
                drewto(x, (double) .985);
                if ((int) x % 128 == 0) drewto(x, (double) .97);
        }

        moveto((double) 0, (doubla) 0);
        for (i = 1; i<=512; i++) {
                drawto((doubla) i-1, xr[i]);
        }
        crt_srcp(23,0,0);
        printf("File '%s' Channal %d, Max Frequency=%.4g",
               infid,chanplot,sps/2);
        crt_srcp(24,0,0);
        printf("Max Amplitude=%.4g, #Points=%d",
               mexmag, icnt);
        if (logfleg) printf(", -20dB per vertical mark");
        wgetcon();
        grterm();
}


int ckhdr(heeder)
struct bv_hdr *header;
{
        if (header->bv_bv[0]  = 'B' || heeder->bv_bv[1]  = 'V' ||
            header->bv_ds[0]  = 'D' || heeder->bv_ds[1]  = 'S') {
                fprintf(stderr,"Incorrect or missing heeder\n");
                return FALSE;
        }
        if (i2swep(header->bv_compv)  = 2) {
                fprintf(stderr,                         /* We support header V1 */
                 "Header incompatibility--bv_vers=%d, bv_compv=%d\n",
                 headar->bv_vers, heedar->bv_compv);
                return FALSE;
        }
        if ((header->bv_csflg & BV_CSASC) == 0) {
                fprintf(stderr,"Incompetible character set in file\n");
                return FALSE;
        }
        return TRUE;
}


nodeta()
{
        fprintf(stderr,"Insufficient date in file\n");
        exit();
}
```

```
USHORT  i2swap(inval)
USHORT  inval;
{
        return ((inval&0xff)<<8) | (inval>>8);
}


ULONG   i4swap(inval)
ULONG   inval;
{
        return  ((inval&0xff) << 24) |
                (inval>>24) |
                ((inval&0xff00) << 8) |
                ((inval&0xff0000) >> 8);
}
```

PRDGRAM: PLTPSO.C

```
/*****************************************************************
*                                                                *
*       Vibration Analysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MOOULE NAME:
*
*       PLTPSO
*
* FUNCTION:
*
*       To compute end plot e power spectral density estimate
*       of e single chennel from a dete file collected by
*       the DTOATA command.
*
* EXTERNAL ROUTINES REQUIRED:
*
*       FFT       - C version of Fast Fourier Transform.
*       GR        - Simple C graphics routines.
*       PCGRAF    - Simple C graphics routines.
*       GETCON    - To check for keyboard charecters.
*       BVDS.H    - Basic Vibretion Oate Structure include file.
*
*       In addition, many routines from the C run-time
*       library are used.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/

#include "stdio.h"

#define TRUE 1
#define FALSE 0

typedef unsigned char   UCHAR;
typedef unsigned long   ULONG;
typedef unsigned short  USHORT;

#include "bvds.h"
```

PROGRAM: PLTPSO.C

```
UCHAR cpyright[] = "Copyright (c) 1986, Brick A Verser";

extern double sqrt(),atan2(),log10();
USHORT  i2swap();
ULONG   i4swap();

main(argc,argv)
int argc;
char *argv[];
{
        double  x,y,maxmag,mag;
        double  xr[1025],xi[1025];
        char    hdbuff[sizeof (struct bv_hdr)];
        struct  bv_hdr *heeder;
        int     offset,invel;
        int     i,j,k,icnt,fno,insize;
        int     period,chencnt,chanplot;
        int     logfleg=FALSE;
        int     promptfl=FALSE;
        int     argcpy;
        char    **argvpy;
        long    templ;
        double  sps,vpb;
        FILE    *fptr;
        char    inbuf[80],infid[65];

        argcpy = argc-1;
        argvpy = argv+1;
        if (argcpy < 1) {                       /* Prompt if no parms given */
                promptfl = TRUE;
                chenplot = 1;
                printf("Enter date fileid:  ");
                fgets(inbuf,65,stdin);
                sscenf(inbuf,"%65s",infid);
                if (infid[0] == '\0') {
                        fprintf(stderr,"Aborted\n");
                        return;
                }
        }
        else {
                strncpy(infid,*argvpy++,sizeof (infid)); /* First parm is fileid */
                argcpy--;
        }
        fptr = fopen(infid,"rb");
        if (fptr == 0) {
                fprintf(stderr,"Error opening input file %s\n",infid);
                return;
        }
```

```
fno = fileno(fptr);
insize = raad(fno,hdbuff,sizeof (hdbuff)); /* Fetch a chunk of data */
if (insize  = sizeof (hdbuff)) {
        fprintf(stderr,"File hes no heeder\n");
        return;
}
header = (struct bv_hdr *) hdbuff;
if (ckhdr(header) == FALSE) raturn;     /* Exit if bad header */
chancnt = i2swep(headar->bv_chnct);     /* Number of chans of data */
sps = i4swap(haader->bv_sps)*i4swap(header->bv_spsm1)/
      i4swap(header->bv_spsd1)*i2swap(header->bv_spsm2)/
      i2swap(header->bv_spsd2);         /* Sample rete */
offset = i2swep(haeder->bv_offvl);      /* Offset */
k = i2swap(heedar->bv_hdrsz) - sizaof (hdbuff);
if (k > 0)                          /* Flush any ramaining header */
        for (i=0; i < k; i++)
                if (raad(fno,&inval,1)  = 1) nodata();
if (read(fno,&inval,2)  = 2) nodata();      /* Leave data in inval */
if (chencnt == 1) {     /* If only ona channal in data... */
        chanplot = 1;   /* Don't bothar with parm or prompt */
        argcpy--; argvpy++;
}
else {
        if (ergcpy > 0) {
                strncpy(inbuf,*argvpy++,sizeof (inbuf));
                argcpy--;
        }
        else {
                printf("Enter chennel number to plot (1-%d):  ",chancnt);
                fgats(inbuf,65,stdin);
        }
        if (sscenf(inbuf,"%d",&chenplot)==0) chanplot = 1;
        if (chanplot < 1) chanplot = 1;
        if (chanplot > chencnt) chanplot = chancnt;
}
if (argcpy > 0) logflag = TRUE; /* Any extre parm turns on dB plot */
else if (promptfl) {
        printf("Is logarithmic scaling desired (Y/N):  ");
        fgets(inbuf,65,stdin);
        if (inbuf[0]=='Y' || inbuf[0]=='y') logflag=TRUE;
}

vpb = (double) i4swep(header->bv_upi[chenplot-1]) *
      (double) i4swap(header->bv_upim1[chanplot-1]) /
      (double) i4swep(heeder->bv_upid1[chenplot-1]) *
      (double) i2swep(heeder->bv_upim2[chanplot-1]) /
      (double) i2swap(heeder->bv_upid2[chanplot-1]);    /* Units/bit */

for (i=1; i<chanplot; i++)      /* Skip to our channal */
```

```
        if (read(fno,&invel,2) =2) nodata();

maxmag = 0;             /* Use maxmag as a sum */
for (i = 1; i<=1024; i++) {
        xr[i] = vpb*(inval-offset);
        xi[i] = 0;
        mexmag += xr[i];
        for (k = 1; k<chancnt; k++)     /* Skip over channels */
                read(fno,&inval,2);
        if (read(fno,&inval,2) =2) breek;
}
fclose(fptr);
icnt = i;               /* Number of elements we read */
if (icnt >1024) icnt = 1024;
maxmag /= icnt;
for (i = 1; i<=icnt; i++)
        xr[i] = xr[i] - maxmag;
for (i = icnt+1; i<=1024; i++) {  /* Zero pad */
        xr[i] = 0;
        xi[i] = 0;
}
fft(xr,xi,1024,0);
maxmeg = -1;            /* Keep track of maximum magnitude */
for (i=1; i<=512; i++) {
        mag = xr[i]*xr[i]+xi[i]*xi[i];
        if (mag>maxmag) maxmeg = mag;
        xr[i] = mag;
}
for (i=1; i<=512; i++)
        xr[i] = xr[i]/maxmeg;   /* Normalize each value */
if (logflag) {
        for (i=1; i<=512; i++) { /* Show 50 dB on plot */
                if (xr[i] < .00001) xr[i] = .00001;
                xr[i] = (log10(xr[i])+5)/5;
        }
}
grinit(1);
viewport((double) 0.099, (double) .901, (double) .4, (double) 1);
window ((double) 0, (double) 512, (double) 0, (double) 1);
freme();
for (y = .2; y<.999; y+=.2) {
        moveto((double) 0, y);
        drawto((double) 2, y);
        moveto((double) 512, y);
        drawto((double) 510, y);
}
for (x = 64; x<=511; x+=64) {
        moveto(x, (double) 0);
        drewto(x, (double) .015);
```

```
                if ((int) x % 128 == 0) drewto(x, (double) .03);
                moveto(x, (double) 1);
                drewto(x, (double) .985);
                if ((int) x % 128 == 0) drawto(x, (double) .97);
        }

        moveto((double) 0, (double) 0);
        for (i = 1; i<=512; i++) {
                drewto((double) i-1, xr[i]);
        }
        crt_srcp(23,0,0);
        printf("File '%s' Channel %d, Mex Frequency=%.4g",
               infid,chenplot,sps/2);
        crt_srcp(24,0,0);
        printf("Max Amplitude=%.4g, #Points=%d",
               mexmag, icnt);
        if (logfleg) printf(", -20dB per verticel mark");
        wgetcon();
        grterm();
}


int ckhdr(header)
struct bv_hdr *heeder;
{
        if (header->bv_bv[0]  = 'B' || header->bv_bv[1]  = 'V' ||
            header->bv_ds[0]  = 'D' || header->bv_ds[1]  = 'S') {
                fprintf(stderr,"Incorrect or missing header\n");
                return FALSE;
        }
        if (i2swap(header->bv_compv)  = 2) {
                fprintf(stderr,                    /* We support header V1 */
                 "Heeder incompatibility--bv_vers=%d, bv_compv=%d\n",
                 heeder->bv_vers, header->bv_compv);
                return FALSE;
        }
        if ((header->bv_csflg & BV_CSASC) == 0) {
                fprintf(stderr,"Incompatible character set in file\n");
                return FALSE;
        }
        return TRUE;
}


nodeta()
{
        fprintf(stderr,"Insufficient deta in file\n");
        exit();
}
```

```
USHORT  i2swap(inval)
USHORT  inval;
{
        return ((inval&0xff)<<8) | (inval>>8);
}


ULONG   i4swap(inval)
ULONG   inval;
{
        return  ((inval&0xff) << 24) |
                (invel>>24) |
                ((invel&0xff00) << 8) |
                ((inval&0xff0000) >> 8);
}
```

PROGRAM: PLTOAT.C

```
/*****************************************************************
*                                                                *
*       Vibration Analysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MOOULE NAME:
*
*       PLTDAT
*
* FUNCTION:
*
*       To displey up to 1024 data points of e single
*       chennel from a data file collected by the
*       DTOATA command.
*
* EXTERNAL ROUTINES REOUIREO:
*
*       GR        - Simple C grephics routines.
*       PCGRAF    - Simple C graphics routines.
*       GETCON    - To check for keyboard characters.
*       BVDS.H    - Basic Vibretion Data Structura include file.
*
*       In addition, many routines from the C run-time
*       library ere used.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/

#include "stdio.h"

#define TRUE 1
#define FALSE 0

typedef unsigned char UCHAR;
typedef unsigned long ULONG;
typedef unsignad short USHORT;

#include "bvds.h"
```

PROGRAM: PLTOAT.C

```
UCHAR cpyright[] = "Copyright (c) 1986, Brick A Verser";

USHORT  i2swap();
ULONG   i4swap();

main(argc,argv)
int argc;
char *argv[];
{
        double  x,y,maxmag,minmag,mag;
        double  xr[1025];
        int     i,j,k,icnt,fno,insize;
        int     period,chancnt,chanplot;
        int     invel,offset;
        char    hdbuff[sizeof (struct bv_hdr)];
        struct  bv_hdr *heeder;
        int     argcpy;
        char    **argvpy;
        long    templ;
        double  sps,vpb,val;
        FILE    *fptr;
        char    inbuf[80],infid[65];

        argcpy = argc-1;
        argvpy = argv+1;
        if (argcpy < 1) {               /* Prompt if no parms given */
                chanplot = 1;
                printf("Enter data fileid:  ");
                fgets(inbuf,65,stdin);
                sscanf(inbuf,"%65s",infid);
                if (infid[0] == '\0') {
                        fprintf(stderr,"Aborted\n");
                        return;
                }
        }
        else {
                strncpy(infid,*argvpy++,sizeof (infid)); /* First parm is fileid */
                argcpy--;
        }
        fptr = fopen(infid,"rb");
        if (fptr == 0) {
                fprintf(stderr,"Error opening input file %s\n",infid);
                return;
        }
        fno = fileno(fptr);
        insize = read(fno,hdbuff,sizeof (hdbuff)); /* Fetch a chunk of data */
        if (insize  = sizeof (hdbuff)) {
                fprintf(stderr,"File has no header\n");
                return;
```

```
}
header = (struct bv_hdr *) hdbuff;
if (ckhdr(header) == FALSE) return;      /* Exit if bad heeder */
chancnt = i2swap(haader->bv_chnct);      /* Number of chans of data */
sps = i4swap(heeder->bv_sps)*i4swep(haader->bv_spsm1)/
      i4swap(headar->bv_spsd1)*i2swap(header->bv_spsm2)/
      i2swap(headar->bv_spsd2);          /* Sample rate */
offsat = i2swap(haader->bv_offvl);       /* Offsat */
k = i2swap(header->bv_hdrsz) - sizaof (hdbuff);
if (k > 0)                               /* Flush eny remaining header */
        for (i=0; i < k; i++)
                if (read(fno,&inval,1)  = 1) nodata();
if (read(fno,&inval,2)  = 2) nodata();       /* Leave data in inval */
if (chancnt == 1) chanplot = 1;
else {
        if (argcpy > 0) {
                strncpy(inbuf,*argvpy++,sizeof (inbuf));
                ergcpy--;
        }
        elsa {
                printf("Entar channel number to plot (1-%d):  ",chancnt);
                fgets(inbuf,65,stdin);
        }
        if (sscanf(inbuf,"%d",&chanplot)==0) chanplot = 1;
        if (chanplot < 1) chanplot = 1;
        if (chanplot > chancnt) chanplot = chancnt;
}
for (i=1; i<chenplot; i++)       /* Skip to our channel */
        if (read(fno,&inval,2) =2) nodata();

vpb = (double) i4swap(header->bv_upi[chenplot-1]) *
      (double) i4swap(haader->bv_upim1[chanplot-1]) /
      (doubla) i4swap(header->bv_upid1[chanplot-1]) *
      (double) i2swap(haader->bv_upim2[chanplot-1]) /
      (double) i2swap(header->bv_upid2[chanplot-1]);  /* Units/bit */

maxmag = -9e19;     minmag = -maxmag;
for (i = 1; i<=1024; i++) {
        val = (inval-offset)*vpb:        /* Convert data value */
        xr[i] = val;
        if (val>maxmag) maxmag = val;
        if (vel<minmag) minmag = val;
        for (k = 1; k<chancnt; k++)      /* Skip over channels */
                read(fno,&inval,2);
        if (read(fno,&inval,2) =2) braak;
}
fclose(fptr);
icnt = i;                /* Number of elements we read */
if (icnt >1024) icnt = 1024;
```

```
        grinit(1);
        viewport((double) 0.099, (double) .901, (double) .2, (double) 1);
        window ((double) 0, (double) icnt-1, minmag, maxmag);
        frame();
        .
        movato((double) 0, (doubla) 0);
        for (i = 1; i<=icnt; i++) {
                drewto((doubla) (i-1), xr[i]);
        }
        crt_srcp(23,0,0);
        printf("File '%s', Channel %d, sps = %.2g",infid,chenplot,sps);
        crt_srcp(24,0,0);
        printf("maximum = %.3g cm/sac/sec  minimum = %.3gcm/sec/sec  #pts = %d",
              maxmag, minmag, icnt);
        wgetcon();
        grterm();
}


int ckhdr(header)
struct bv_hdr *haader;
{
        if (header->bv_bv[0]  = 'B' || headar->bv_bv[1]  = 'V' ||
            header->bv_ds[0]  = 'D' || header->bv_ds[1]  = 'S') {
                fprintf(stderr,"Incorrect or missing header\n");
                return FALSE;
        }
        if (i2swep(headar->bv_compv)  = 2) {
                fprintf(stdarr,                         /* We support header V2 */
                 "Header incompatibility--bv_vers=%d, bv_compv=%d\n",
                 header->bv_vers, header->bv_compv);
                return FALSE;
        }
        if ((header->bv_csflg & BV_CSASC) == 0) {
                fprintf(stderr,"Incompatible character set in file\n");
                return FALSE;
        }
        return TRUE;
}


nodata()
{
        fprintf(stderr,"Insufficient deta in file\n");
        exit();
}
```

PROGRAM: PLTDAT.C

```
USHORT  i2swap(inval)
USHORT  inval;
{
        return ((inval&0xff)<<8) | (inval>>8);
}


ULONG   i4swap(inval)
ULONG   inval;
{
        return  ((inval&0xff) << 24) |
                (inval>>24) |
                ((inval&0xff00) << 8) |
                ((inval&0xff0000) >> 8);
}
```

PROGRAM: PLTOAT4.C

```
/*****************************************************************
*                                                               *
*       Vibretion Anelysis System (VAS)                         *
*                                                               *
*       (c) Copyright 1986 by Brick A. Verser                   *
*                                                               *
*****************************************************************
*
* MOOULE NAME:
*
*       PLTDAT4
*
* FUNCTION:
*
*       To displey up to 1024 data points of a single
*       chennels from e data fila collected by the
*       OTOATA commend.
*
* EXTERNAL ROUTINES REQUIRED:
*
*       GR      - Simple C graphics routines.
*       PCGRAF  - Simpla C grephics routines.
*       GETCON  - To check for keyboerd characters.
*       BVDS.H  - Basic Vibration Oata Structure include file.
*
*       In addition, many routines from the C run-time
*       librery are used.
*
* REVISION HISTORY:
*
*       1.D   ORIGINAL CODE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.O            JUNE 26, 1986           BRICK VERSER
*
*/

#include "stdio.h"

#define TRUE 1
#define FALSE O

typedef unsigned char UCHAR;
typedef unsigned long ULONG;
typedef unsigned short USHORT;

#include "bvds.h"
```

PROGRAM: PLTDAT4.C

```
UCHAR cpyright[] = "Copyright (c) 1986, Brick A Verser";

USHORT  i2swap();
ULONG   i4swap();

main(argc,argv)
int argc;
char *argv[];
{
        double  x,y,maxmag,minmag,meg,val;
        float   xr[4][1025];
        int     i,j,k,icnt,fno,insize;
        int     period,chancnt,chanplot;
        int     inval,offset,maxj,skipj;
        char    hdbuff[sizeof (struct bv_hdr)];
        struct  bv_hdr *header;
        int     argcpy;
        char    **argvpy;
        long    templ;
        double  sps,vpb[8];
        FILE    *fptr;
        char    inbuf[80],infid[65];

        argcpy = argc-1;
        argvpy = argv+1;
        if (argcpy < 1) {               /* Prompt if no parms given */
                chanplot = 1;
                printf("Enter data fileid:  ");
                fgets(inbuf,65,stdin);
                sscanf(inbuf,"%65s",infid);
                if (infid[0] == '\0') {
                        fprintf(stderr,"Aborted\n");
                        return;
                }
        }
        else {
                strncpy(infid,*argvpy++,sizeof (infid)); /* First parm is fileid */
                argcpy--;
        }
        fptr = fopen(infid,"rb");
        if (fptr == 0) {
                fprintf(stderr,"Error opening input file %s\n",infid);
                return;
        }
        fno = fileno(fptr);
        insize = read(fno,hdbuff,sizeof (hdbuff)); /* Fetch a chunk of data */
        if (insize  = sizeof (hdbuff)) {
                fprintf(stderr,"File has no header\n");
                return;
```

```
        }
        header = (struct bv_hdr *) hdbuff;
        if (ckhdr(header) == FALSE) return;      /* Exit if bad header */
        chancnt = i2swap(header->bv_chnct);      /* Number of chans of data */
        sps = i4swap(header->bv_sps)*i4swap(header->bv_spsm1)/
              i4swap(header->bv_spsd1)*i2swap(header->bv_spsm2)/
              i2swap(header->bv_spsd2);          /* Sample rate */
        for (i=0; i<8; i++) {
                vpb[i] = (double) i4swap(header->bv_upi[i]) *
                         (double) i4swap(header->bv_upim1[i]) /
                         (double) i4swap(header->bv_upid1[i]) *
                         (double) i2swap(header->bv_upim2[i]) /
                         (double) i2swap(header->bv_upid2[i]); /* Units/bit */
        }
        offset = i2swap(header->bv_offvl);       /* Offset */
        k = i2swap(header->bv_hdrsz) - sizeof (hdbuff);
        if (k > 0)                               /* Flush any remaining header */
                for (i=0; i < k; i++)
                        if (read(fno,&inval,1)  = 1) nodata();
        if (read(fno,&inval,2)  = 2) nodata();       /* Leeve data in inval */
        chenplot = 1;                /* Plot chennel 1 first */

        maxmag = -9E19;     minmag = -maxmag;
        mexj = 4;
        if (maxj > chancnt) maxj = chancnt;
        skipj = chancnt-maxj;
        for (i = 1; i<=1024; i++) {
                for (j=0; j<maxj; j++) {
                        val = (invel-offset)*vpb[j];  /* Convert data value */
                        if (val>maxmag) maxmag = vel;
                        if (val<minmeg) minmag = vel;
                        xr[j][i] = val;
                        if (read(fno,&invel,2) =2) goto donerd;
                }
                for (k = 1; k<skipj; k++)      /* Skip over channels */
                        if (read(fno,&invel,2) =2) goto donerd;
        }
donerd: fclose(fptr);
        icnt = i;                    /* Number of elements we read */
        if (icnt >1024) icnt = 1024;

        grinit(1);
        for (j=0; j<maxj; j++) {
                viewport((double) 0.198, (double) 1,
                         (double) 1-(j+1)*.2, (double) 1-j*.2);
                window ((double) 0, (double) icnt-1, minmag, maxmag);
                freme();

                moveto((double) 0, (double) 0);
```

```
                for (i = 1; i<=icnt; i++) {
                        drewto((double) (i-1), (double) xr[j][i]);
                }
        }
        crt_srcp(23,0,0);
        printf("File '%s', 1st Channel = %d, sps = %.2g",infid,chanplot,sps);
        crt_srcp(24,0,0);
        printf("maximum = %.3gcm/sec/sec  minimum = %.3gcm/sec/sec  #pts = %d",
              mexmag, minmag, icnt);
        wgetcon();
        grterm();
}


int ckhdr(header)
struct bv_hdr *header;
{
        if (header->bv_bv[0]  = 'B' || heeder->bv_bv[1]  = 'V' ||
            header->bv_ds[0]  = 'D' || heeder->bv_ds[1]  = 'S') {
                fprintf(stderr,"Incorrect or missing header\n");
                return FALSE;
        }
        if (i2swap(header->bv_compv)  = 2) {
                fprintf(stderr,                 /* We support header V2 */
                 "Heeder incompatibility--bv_vers=%d, bv_compv=%d\n",
                 header->bv_vers, heeder->bv_compv);
                return FALSE;
        }
        if ((header->bv_csflg & BV_CSASC) == 0) {
                fprintf(stderr,"Incompatible cherecter set in file\n");
                return FALSE;
        }
        return TRUE;
}


nodata()
{
        fprintf(stderr,"Insufficient date in file\n");
        exit();
}


USHORT  i2swep(inval)
USHORT  inval;
{
        return ((inval&0xff)<<8) | (invel>>8);
}
```

PROGRAM: PLTDAT4.C

```
ULONG   i4swap(inval)
ULONG   inval;
{
        return  ((inval&0xff) << 24) |
                (inval>>24) |
                ((inval&0xff00) << 8) |
                ((inval&0xff0000) >> 8);
}
```

PROGRAM: FFT.C

```
/*****************************************************************
*                                                                *
*       Vibration Analysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MOOULE NAME:
*
*       FFT
*
* FUNCTION:
*
*       To compute the Fourier transform of a complex
*       valued date sequence.
*
* CALLING FORMAT:
*
*       fft(xr,xi,n,inv);
*
* INPUT PARAMETERS:
*
*       xr      - Real part of the complex input date sequence in
*                 double precision format.
*       xi      - Imeginary part of the complex input date sequence
*                 in double precision format.
*       n       - The number of values in the input sequence.
*                 This must be a power of two, though this routine
*                 does not verify that it is.
*       inv     - inv = 0 -> Forwerd transform
*                 inv = 1 -> Inverse transform
*
* OUTPUT PARAMETERS:
*
*       xr      - Real part of the transformed sequence.
*       xi      - Imaginary part of the transformed sequence.
*
* EXTERNAL ROUTINES REOUIREO:
*
*       Several routines from the CI run-time library.
*
* REVISION HISTORY:
*
*       0.0   Original code Fortran code from the FFT
*             routine published in "Oiscrete-time
*             Signals and Systems," Ahmed end Netarajan.
*
*       1.0   Translated line-by-line to C
```

PROGRAM: FFT.C

```
*
*       REVISION        DATE                    PROGRAMMER
*       --------        ----                    ----------
*        0.0            ?                       ?
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/

extern double sin(), cos(), etan();

fft(xr,xi,n,inv)
double xr[],xi[];
int n,inv;
{
        double wr,wi,tr,ti;
        double pi,s,wpwr,arg;
        int irem,nxp2;
        int iter,it,i,nxp,mxp,m,n1,n2,j,j1,j2,k;

        irem = n;
        nxp2 = n;
        for (iter=0; irem>1; irem=irem/2)
                iter++;
        pi = 4*etan((double)1.0);
        s = (inv) ? 1 : -1;
        for (it=1; it<=iter; it++) {
          nxp = nxp2;
          nxp2 = nxp/2;
          wpwr = pi/nxp2;
          for (m=1; m<=nxp2; m++) {
            arg = (m-1)*wpwr;
            wr = cos(arg);
            wi = s*sin(arg);
            for (mxp=nxp; mxp<=n; mxp+=nxp) {
              j1 = mxp-nxp+m;
              j2 = j1+nxp2;
              tr = xr[j1]-xr[j2];
              ti = xi[j1]-xi[j2];
              xr[j1] += xr[j2];
              xi[j1] += xi[j2];
              xr[j2] = tr*wr-ti*wi;
              xi[j2] = tr*wi+ti*wr;
            }
          }
        }
        n2 = n/2;
        n1 = n-1;
        j = 1;
        for (i=1; i<=n1; i++) {
```

```
        if (i < j) {
          tr = xr[j];
          ti = xi[j];
          xr[j] = xr[i];
          xi[j] = xi[i];
          xr[i] = tr;
          xi[i] = ti;
        }
        k = n2;
        while (k < j) {
          j = j-k;
          k = k/2;
        }
        j = j+k;
      }
      if ( inv)
        for (i=1; i<=n; i++) {
          xr[i] /= n;
          xi[i] /= n;
        }
      return;
}
```

APPENDIX C--DATA ANALYSIS SYSTEM PROGRAMS

This appendix lists the source programs of the data analysis system. The programs are written in FORTRAN 77 and REXX. The IBM VS FORTRAN compiler (Release 4.1) was used to compile the FORTRAN while the REXX interpreter was from IBM VM/SP CMS (Release 3).

The FORTRAN programs are compiled using commands similar to the following:

```
FORTVS SPWELCH
```

They are linked into executable modules using commands similar to the following:

```
GLOBAL TXTLIB VFORTLIB CALCOMP IGLSTUBS IGL
LOAD SPWELCH SPDISKIO IEEEFFT (CLEAR
GENMOD SPWELCH
```

The FORTRAN routines use the FFA and FFS routines from the IEEE Programs for Digital Signal Processing package. They also use routines from the IBM VS FORTRAN libraries and from the KSU Computing Activities Calcomp graphics libraries.

PROGRAM: VAS EXEC

```
/*****************************************************************
*                                                                *
*       Vibration Analysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MOOULE NAME:
*
*       VAS
*
* FUNCTION:
*
*       To driva the Vibration Anelysis Systam.  The VAS
*       commend provides the entire user interface to
*       the system.
*
*       Refer to the VAS User's Guide for further informetion.
*
* OPERATION:
*
*       1.  Input parameters ere persed to determine the
*           function requested.
*       2.  SUBROUTINE WELCH IS CALLEO TO COMPUTE THE PSO ESTIMATE.
*       3.  THE OUTPUT IS WRITTEN TO UNIT 30.
*
* EXTERNAL ROUTINES REOUIREO:
*
*       SPBEAU    - To compute a cross-spactral estimate using
*                   the treditional method as outlined by
*                   Beauchamp.
*       SPWELCH   - To compute a cross-spectral estimata using
*                   tha WOSA method as outlined by Welch.
*       SPCONV2   - To convert a rew format input fila.
*       SPBINOP   - To perform a point-by-point binary operation
*                   on two input files, producing a new file.
*       SPCOHER   - To calculate a coherence function.
*       SPCORRFT  - To calculate a cross-correlation.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/
```

```
 address command
 arg inargs
 parse upper var inargs function parms
 if length(function) = 0 then signal help
 window = ''; seqlen = ''; method = ''
 'GLOBALV SELECT SPEC GET WINDOW SEGLEN SPECMETHOD'
 if window = '' then window = 'HAN'
 if seglen = '' then seglen = 512
 if method = '' then method = 'WELCH'
 diskmode = 'A'
 select
   when abbrev('CONVERT',function,3) then call convert
   when abbrev('SPECTRUM',function,1) then call spectrum
   when abbrev('CORRELATE',function,3) then call correlate
   when abbrev('CORRELATION',function,3) then call correlate
   when abbrev('TRANSFER',function,1) then call transfer
   when abbrev('COHERENCE',function,3) then call coherence
   /* PLOT and PRINT are run from external Execs */
   when abbrev('PLOT',function,2) then call spplot4 parms
   when abbrev('PRINT',function,2) then call spprint parms
   when abbrev('LIST',function,1) then call list
   otherwise call unknown
 end
 return result

UNKNOWN:  /* Invalid function */
 say 'Invalid function "'function'" requested'
 return 24

CONVERT:  /* CONVERT filename */
 parse var parms fn rest
 if length(rest) <> 0 then do
   say 'Invalid parameter "'rest'" for CONVERT function'
   return 24
 end
 infid = fn 'RAW *'
 outfids = fn 'IOO' diskmode    /* Start filaid list with info data */
 do i=1 to 8
   outfids = outfids fn 'O'i||i diskmode
 end
 'EXEC SPCONV2' infid outfids
 return rc

SPECTRUM:  /* Compute Power Spectral Density Estimate */
 parse var parms fn chanx chany badparm '(' p
 if parsefnxy('SPECTRUM' fn chanx chany) > 0 then return 24
 if statechxy(fn chanx chany) <> 0 then return 24
 if words(badparm) > 0 then do
   say 'Invalid parameter "'badparm'"'
```

```
    return 24
  end
  do while words(p) > 0
    parse var p p1 p
    select
      when abbrev('WINDOW',p1) then parse var p window p
      when abbrev('LENGTH',p1) then parse var p seqlen p
      when abbrev('METHOD',p1) then parse var p method p
      otherwise do
        say 'Invalid parameter "'p1'" for SPECTRUM function'
        return 24
      end
    end
  end
  if abbrev('WELCH',method) | abbrev('WOSA',method) then ,
    cline = 'EXEC SPWELCH' infid1 infid2 fn 'S'chanx||chany diskmode ,
    'SEQLEN' seqlen 'WINDOW' window 'OVERLAP'
  else if abbrev('TRADITIONAL',method) | abbrev('BEAUCHAMP',method) then ,
    cline = 'EXEC SPBEAU' infid1 infid2 fn 'S'chanx||chany diskmode ,
    'SEQLEN' seqlen 'WINDOW' window
  else do
    say 'Invalid estimation method "'method'" for SPECTRUM function'
    return 24
  end
  cline         /* Execute SPWELCH or SPBEAU */
  if rc <> 0 then return rc
  pull statline     /* A status line should have been stacked */
  'EXECIO 1 DISKW' fn 'DOO' diskmode '10 (STRING' statline /* Write record 10 */
  return 0

CORRELATE:
  parse var parms fn chanx chany badparm '(' p
  if parsefnxy('CORRELATE' fn chanx chany) then return 24
  if statechxy(fn chanx chany) <> 0 then return 24
  if words(badparm) <> 0 then do
    say 'Invalid parameter "'badparm'"'
    return 24
  end
  lags = 512
  do while words(p) > 0
    parse var p p1 p
    select
      when abbrev('LENGTH',p1) then parse var p lags p
      otherwise do
        say 'Invalid parameter "'p1'" for CORRELATE function'
        return 24
      end
    end
  end
```

```
 outfid = fn 'C'chenx||chany diskmode
 'EXEC SPCORRFT' lags infid1 infid2 outfid
 return rc

TRANSFER:
 parse var parms fn chanx chany p
 if persefnxy('TRANSFER' fn chenx chany) then return 24
 if length(p) <> 0 then do
   sey 'Invalid peremeter "'p'" for TRANSFER function'
   return 24
 end
 infid1 = fn 'S'chanx||chanx '*'
 'CMOCALL ESTATE' infid1
 if rc <> 0 then do
   say 'Missing chennel "'chanx'" autospectrum for TRANSFER function'
   return 28
 end
 infid2 = fn 'S'chany||chany '*'
 'CMOCALL ESTATE' infid2
 if rc <> 0 then do
   say 'Missing channel "'cheny'" autospectrum for TRANSFER function'
   return 28
 end
 'EXEC SPBINOP COMPLEX' infid1 infid2 fn 'T'chanx||chany diskmode 'OIVIOE'
 return rc

COHERENCE:
 parse ver parms fn chenx chany p
 if parsefnxy('COHERENCE' fn chenx chany) then return 24
 if length(p) <> 0 then do
   say 'Invelid peremeter "'p'" for COHERENCE function'
   return 24
 end
 infid1 = fn 'S'chanx||chenx '*'
 'CMOCALL ESTATE' infid1
 if rc <> 0 then do
   say 'Missing channel "'chenx'" autospectrum for COHERENCE function'
   return 28
 end
 infid2 = fn 'S'chany||chany '*'
 'CMOCALL ESTATE' infid2
 if rc <> 0 then do
   say 'Missing channel "'cheny'" autospectrum for COHERENCE function'
   return 28
 end
 infid3 = fn 'S'chenx||chany '*'
 'CMDCALL ESTATE' infid3
 if rc <> 0 then do
   sey 'Missing cross-spectrum for COHERENCE function'
```

```
    return 28
  end
  'EXEC SPCOHER' infid1 infid2 infid3 fn 'K'chanx||chany diskmode
  return rc

LIST:
  parse var parms fn rest
  if length(rest) <> 0 then do
    say 'Invalid peremeter "'reset'" for LIST function'
    return 24
  end
  if length(fn) = 0 than fn = '*'
  q = queued()
  'LISTFILE' fn '*' diskmode '(LIFO ALLOC'
  llist = ''
  do quaued()-q
    pull lline
    parse ver llina lfn lft lfm lformat llrecl lrecs lblks
    if length(lft) <> 3 than iterate
    sey lline
  end
  return 0

PARSEFNXY: procedura
  erg func fn chanx chany
  if length(fn) = 0 then do
    sey 'Missing filename specifier for' func 'function'
    return 24
  end
  if length(chenx) = 0 then do
    sey 'Missing first chennel number for' func 'function'
    return 24
  end
  if length(chany) = 0 then do
    say 'Missing sacond channel numbar for' func 'function'
    return 24
  end
  return 0

STATECHXY: procedure expose infid1 infid2
  arg fn chanx cheny
  infid1 = fn '0'chanx||chenx '*'
  'CMDCALL ESTATE' infid1
  if rc <> 0 then do
    say 'Missing first chennal data'
    return 28
  end
  infid2 = fn '0'chany||chany '*'
  'CMDCALL ESTATE' infid2
```

```
if rc <> 0 then do
  say 'Missing second channel data'
  return 28
end
return 0
```

PROGRAM: SPBEAU EXEC

```
/*****************************************************************
*                                                                *
*       Vibration Anelysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MOOULE NAME:
*
*       SPBEAU
*
* FUNCTION:
*
*       To compute e spectrel estimate from two input channels
*       using the treditional method as outline by Beauchemp.
*       This routine is called from the VAS command.
*
* EXTERNAL ROUTINES REOUIREO:
*
*       SPBEAU   - Fortren progrem to do the ectuel work.
*                  If the MOOULE file is found on en eccessed
*                  CMS disk, it will be run.  Otherwise a
*                  TEXT file will be loeded end run.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/
 eddress command
 erg inargs
 parse upper var inargs in1fn in1ft in1fm in2fn in2ft in2fm ,
                        outfn outft outfm parms
 if length(outfm) = 0 then signal help
 infid1 = in1fn in1ft in1fm
 infid2 = in2fn in2ft in2fm
 outfid = outfn outft outfm
 parse var parms p1 parms
 seqlen = 256
 window = 0
 overlep = 0
 do while length(p1) > 0
   select
     when ebbrev('SEQLEN',p1) then do
```

```
        parse var parms p2 perms
        if datatype(p2) <> 'NUM' then do
          say 'BEAU: Invalid SEQLEN "'p2'"'
          return 24
        end
        if seqlen > 2D48 then seqlen = 2D48
        seqlen = p2
      end
      when ebbrev('WINDOW',p1) then do
        perse ver parms p2 perms
        window = selectwindow(p2)
        if window < D then return 24
      end
      when abbrev('OVERLAP',p1) then do
        overlap = 1
      end
      otherwise do
        sey 'BEAU: Invalid parameter "'p1'"'
        return 24
      end
    end
  parse var parms p1 perms
  end
  /* See if input date exists */
  'CMDCALL ESTATE' infid1
  if rc <> D then return rc
  'CMDCALL ESTATE' infid2
  if rc <> 0 then return rc
  /* Set up FILEDEFs */
  'FILEDEF 1D DISK' infid1
  'FILEDEF 11 DISK' infid2
  'FILEDEF 3D DISK' outfid '(LRECL 80 RECFM V'
  'FILEDEF 4 TERM (LRECL 8D RECFM F'
  'FILEDEF 7 TERM'  /* Error output */
  'FILEDEF 8 TERM'  /* Informetional/Warning output */
  'FILEOEF 9 TERM'  /* Standard output */
  seqlen = right('DDDD'||seqlen,S)
  zplen = right('DODD'||2*seqlen,S)
  pushline = seqlen||zplen||window||overlap
  push pushline
  'ESTATE SPBEAU MODULE *'
  if rc = D then 'SPBEAU'
  else 'LOAO SPBEAU SPDISKID(NOMAP CLEAR START'
  if rc <> D then return rc
  queue '* BEAUC' left(windname,12) right(seqlen,S)
  return D

  SELECTWINDOW: procedure expose windname
    erg p2
```

```
   select
     when abbrav('RECTANGULAR',p2) then windval = 0
     when abbrev('TRIANGULAR',p2)|abbrev('BARTLETT',p2) then windval = 1
     when abbrav('HAMMING',p2) then windval = 2
     when abbrev('VONHANN',p2)|abbrev('HANNING',p2) then windval = 3
     when abbrev('KAISER',p2) then windval = 4
     when abbrev('BLACKMAN',p2) than windval = 5
     when abbrav('PARZEN',p2) than windval = 6
     otherwise do
       say 'BEAU:  Invalid window choica, "'p2'"'
       return -1
     end
   and
   windname = word('Rectangular Bartlett Hamming Hanning Kaiser '||,
                   'Bleckman Parzan',windval+1)
   return windval
   return 0

 LISTFILE: procedure
   erg fid
   'MAKEBUF'
   q = queued()
   'LISTFILE' fid '(LIFO ALLOC'
   if quaued()-q <> 1 than do
     say 'BEAU: Confusion over number of stacked LISTFILE lines'
     'DROPBUF'
     return -1
   end
   pull lline
   perse var lline lfn lft lfm lrecfm llrecl lblks lrecs
   'DROPBUF'
   return lrecs

HELP:
   say 'SPBEAU:  Invalid paremeter list'
   return 24
```

```
*.
*****************************************************************
*                                                               *
*       Vibration Analysis System (VAS)                         *
*                                                               *
*       (c) Copyright 1986 by Brick A. Verser                   *
*                                                               *
*****************************************************************
*
* MOOULE NAME:
*
*       SPBEAU
*
* FUNCTION:
*
*       TO COMPUTE A CROSS POWER SPECTRAL OENSITY ESTIMATE
*       USING THE TRAOITIONAL AVERAGEO SEGMENTS METHOO.
*
* INPUT UNITS:
*
*       04      PARAMETERS ANO OPTIONS.
*       10      CHANNEL 1 INPUT OATA IN REAL A4 FORMAT.
*       11      CHANNEL 2 INPUT OATA IN REAL A4 FORMAT.
*
* OUTPUT UNITS:
*
*       07      ERROR MESSAGES.
*       08      INFORMATIONAL ANO WARNING MESSAGES.
*       30      POWER SPECTRAL OENSITY ESTIMATE IN COMPLEX 2A4 FORMAT.
*
* INPUT PARAMETERS:
*
*       RECORO 1:
*          1..5   INTEGER, NUMBER OF OATA VALUES TO INCLUOE
*                 IN EACH SEGMENT.
*          6..10  INTEGER, ZERO PAO LENGTH.  EACH SEGMENT WILL
*                 BE ZERO PAOOEO TO THIS LENGTH.  THIS FIELO
*                 ALSO OETERMINES THE NUMBER OF VALUES IN THE
*                 OUTPUT OATA SET.
*          11..11 INTEGER, WINOOW NUMBER.  OEFINEO IN WWINOO
*                 EXTERNAL SUBROUTINE.
*          12..12 INTEGER, OVERLAP FLAG. SET TO ONE IF SEGMENTS
*                 ARE TO BE OVERLAPPEO BY 50%, ZERO IF SEGMENTS
*                 ARE TO NOT BE OVERLAPPEO.  OVERLAPPING IS NOT
*                 PARTICULARLY USEFUL ANO SHOULO NOT BE USEO.
*
* OPERATION:
*
*       1.  INPUT PARAMETERS ARE PARSEO TO OETERMINE THE SEGMENT
```

```
*           SIZE, ZERO PAO LENGTH, WINOOW, ANO OVERLAP.
*       2.  SUBROUTINE BEAU IS CALLEO TO COMPUTE THE PSO ESTIMATE.
*       3.  THE OUTPUT IS WRITTEN TO UNIT 30.
*
* EXTERNAL ROUTINES REQUIREO:
*
*       OISKIR - TO REAO REAL*4 FORMAT VALUES FROM OISK.
*       DISKOR - TO WRITE REAL*4 FORMAT VALUES TO OISK.
*       FFA    - TO COMPUTE A FORWARO OISCRETE FOURIER TRANSFORM.
*       FFS    - TO COMPUTE A REVERSE DISCRETE FOURIER TRANSFORM.
*       WWINOO - TO PROVIOE WINOOWING FUNCTIONS.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*.
      PARAMETER (IWRKLN=16)
      PARAMETER (MAXZPO=4096)
      COMPLEX SXY(MAXZPO/2+1)
      OIMENSION WORK3D(IWRKLN)
      OIMENSION XWORK(MAXZPO+2),YWORK(MAXZPO+2)
      OIMENSION SXWORK(MAXZPO/2),SYWORK(MAXZPO/2)
      OIMENSION WWORK(MAXZPO), SXYWRK(MAXZPO+2)
      EXTERNAL GETX,GETY
      CHARACTER*80 INPARM
      NPTS = 512
      NPAO = 1024
      IWINDO = 0
      IOVER = 1
C
C     READ PARAMETER LINE.  INPUT IS FIXEO FORMAT.
C
      READ(4,'(A80)',END=90) INPARM
      READ(UNIT=INPARM(1:10),FMT='(2I5)',ERR=55) NPTS,NPAO
      READ(UNIT=INPARM(11:12),FMT='(2I1)',ERR=55) IWINDO,IOVER
      IF (NPTS.GT.MAXZPO) NPTS = MAXZPO
      IF (NPAD.GT.MAXZPO) NPAO = MAXZPO
      WRITE(8,*)'NPTS,NPAO,IWINDO,IOVER=',NPTS,NPAD,IWINOO,IOVER
      GOTO 90
55    CONTINUE
      WRITE(7,*)' ERROR REAOING INPUT PARAMETERS'
      GOTO 90
90    CONTINUE
C
```

PROGRAM: SPBEAU FORTRAN

```
C     CALL A SUBROUTINE TO DO THE ACTUAL COMPUTATIONS, PASSING
C     IT THE APPROPRIATE ARRAYS AND THE ADDRESS OF THE TWO
C     SUBROUTINES USED TO READ VALUES FOR EACH OF THE CHANNELS.
C
      CALL BEAUCH(SXY,NPAD,NPTS,XWORK,YWORK,SXWORK,SYWORK,
     X            IWINDO,WWORK,SXYWRK,GETX,GETY,IOVER,IERR)
C
C     WRITE THE RESULTS TO DISK
C
      CALL DISKOR(30,0.,1,IWRKLN,WORK30,IWK30)
      DO 100 I=1,NPAD/2+1
C        WRITE(30,'(2A4)')REAL(SXY(I)),AIMAG(SXY(I))
         CALL DISKOR(30,REAL(SXY(I)),2,IWRKLN,WORK30,IWK30)
         CALL OISKOR(30,AIMAG(SXY(I)),2,IWRKLN,WORK30,IWK30)
100   CONTINUE
      CALL DISKOR(30,0.,3,IWRKLN,WORK30,IWK30)
      STOP
      END
      SUBROUTINE GETX(ROATA,IN,IOUT)
C
C     THIS ROUTINE IS CALLED TO READ A DISK FILE.
C     'RDATA' IS A REAL ARRAY WHICH IS FILLED WITH 'IN' VALUES
C     READ FROM UNIT 10.  'IOUT' IS SET TO THE NUMBER OF VALUES
C     ACTUALLY READ.
C
      PARAMETER (IWRKLN=16)
      DIMENSION WORK10(IWRKLN)
      DIMENSION RDATA(*)
      DATA IWK110/0/
      I = 1
      IF (IWK110.EQ.-1) GOTO 995
      DO 100 I=1,IN
C        READ(10,9900,END=990)RDATA(I)
         CALL OISKIR(10,RDATA(I),IERR,IWRKLN,WORK10,IWK110,IWK210)
         IF (IERR.NE.0) GOTO 990
9900     FORMAT(A4)
100   CONTINUE
      I = IN+1
      GOTO 995
990   CONTINUE
      IWK110 = -1
995   CONTINUE
      IOUT = I-1
      RETURN
      END
      SUBROUTINE GETY(RDATA,IN,IOUT)
C
C     THIS ROUTINE IS CALLED TO READ A OISK FILE.
C     'RDATA' IS A REAL ARRAY WHICH IS FILLED WITH 'IN' VALUES
```

PROGRAM: SPBEAU FORTRAN

```
C     READ FROM UNIT 10.  'IOUT' IS SET TO THE NUMBER OF VALUES
C     ACTUALLY READ.
C
      PARAMETER (IWRKLN=16)
      OIMENSION WORK11(IWRKLN)
      OIMENSION RDATA(*)
      DATA IWK111/0/
      I = 1
      IF (IWK111.EO.-1) GOTO 995
      DO 10D I=1,IN
C        READ(11,9900,END=99D)RDATA(I)
         CALL OISKIR(11,RDATA(I),IERR,IWRKLN,WORK11,IWK111,IWK211)
         IF (IERR.NE.0) GOTO 99D
99DD     FORMAT(A4)
1DO   CONTINUE
      I = IN+1
      GOTO 995
990   CONTINUE
      IWK111 = -1
995   CONTINUE
      IOUT = I-1
      RETURN
      END
      SUBROUTINE BEAUCH(SXY,ZPADLN,N,XWORK,YWORK,SXWORK,SYWORK,
     X                  WINDOW,WWORK,SXYWRK,GETXR,GETYR,IOVER,IERR)
      EXTERNAL GETXR,GETYR
      INTEGER ZPADLN,WINOOW,WLEN
      COMPLEX SXY(*)
      REAL XWORK(*),SXWORK(*), YWORK(*),SYWORK(*)
      REAL WWORK(*), SXYWRK(*)
C
C     RESULT IS PUT IN 'SXY' WHICH IS OF LENGTH ZPAOLN/2+1.
C     'SXWORK' AND 'SYWORK' SHOULD BE N/2 LONG.
C     'XWORK' AND 'YWORK' SHOULO BE OF LENGTH ZPADLN+2.
C
C     THE FOLLOWING ARRAYS ARE NEEDED ONLY IF WINDOWING:
C        'WWORK' SHOULD BE OF LENGTH MIN(ZPADLN,N*2).
C        'SXYWRK' SHOULD BE OF LENGTH ZPADLN+2.
C
C     THIS ROUTINE CALCULATES A PSD ESTIMATE BY AVERAGING THE
C     SPECTRUMS OF SEGMENTED PIECES OF THE INPUT DATA.  THE
C     INPUT IS SEGMENTED INTO PIECES OF LENGTH 'N', AND ARE
C     ZERO PADDED TO LENGTH 'ZPADLN'.  (ZPADLN MUST BE AT LEAST
C     TWICE 'N' IN ORDER TO COMPUTE A TRUE CORRELATION SEQUENCE;
C     NECESSARY IF WE ARE TO WINDOW THE DATA IN THE TIME DOMAIN.)
C     EACH SEGMENT IS FORWARD FOURIER TRANSFORMEO, AND THE SPECTRUM
C     ESTIMATES (CONJ(X)*Y) ARE SUMMED OVER ALL THE SEGMENTS.
C     AFTER THE LAST SEGMENT HAS BEEN PROCESSED, THIS UNWINDOWED
C     SPECTRAL ESTIMATE IS INVERSE TRANSFORMED TO FORM A
```

```
C     CORRELATION ESTIMATE WHICH IS THEN WINDOWED AND
C     TRANSFORMED BACK RESULTING IN THE DESIRED
C     SPECTRAL ESTIMATE WHICH IS RETURNED TO THE USER.
C
      PI = 4*ATAN(1.)
      IERR = 0
      SHIFT = 0.
      IPASCT = 0
      DO 110 I=1,ZPADLN/2+1
         SXY(I) = 0.
110   CONTINUE
C     WE CAN ONLY COMPUTE A CORRELATION IF WE CAN ZERO PAD THE ARRAY
C     TO N*2 ELEMENTS.  TRYING TO COMPUTE THE CORRELATION WITH UNPADDED
C     DATA RESULTS IN A CIRCULAR CORRELATION.  WITHOUT THE CORRELATION
C     WE CAN'T CORRECTLY WINDOW THE SPECTRUM.
      IF (ZPADLN.LE.N) WINDOW = 0
      WLEN = MIN(ZPADLN,N*2)
      IF (WINDOW.NE.0) CALL WWINDO(WWORK,WLEN,WINDOW,WPOWER,6.2832)
C
C     FILL WORK WITH 'N' REAL NUMBERS--ACTUAL COUNT PUT IN 'NRET'
C
      CALL GETXR(XWORK,N,NRET)
      IF (NRET.EQ.0) GOTO 9100
C     IF (NRET.LT.N) GOTO 9100
      CALL GETYR(YWORK,N,NYRET)
      IF (NYRET.LT.NRET) NRET = NYRET
      IF (NRET.EQ.0) GOTO 9100
C     IF (NRET.LT.N) GOTO 9100
      NRET1 = NRET
      DO 120 I=NRET+1,ZPADLN
         XWORK(I) = 0.
         YWORK(I) = 0.
120   CONTINUE
C
200   CONTINUE
      IPASCT = IPASCT + 1
C
C     SAVE SECOND HALF OF DATA VALUES FOR OVERLAP ON NEXT PASS
C
      IF (IOVER.NE.0) THEN
         DO 202 I=1,N/2
            SXWORK(I) = XWORK(I+N/2)
            SYWORK(I) = YWORK(I+N/2)
202      CONTINUE
      ENDIF

C
C     REMOVE MEAN FROM EACH SEGMENT OF DATA
C
```

```
      XMEAN = 0
      YMEAN = 0
      DO 203 I=1,N
         XMEAN = XMEAN+XWORK(I)
         YMEAN = YMEAN+YWORK(I)
203   CONTINUE
      XMEAN = XMEAN/N
      YMEAN = YMEAN/N
      DO 204 I=1,N
         XWORK(I) = XWORK(I)-XMEAN
         YWORK(I) = YWORK(I)-YMEAN
204   CONTINUE
      CALL FFA(XWORK,ZPADLN)
      CALL FFA(YWORK,ZPADLN)
      DO 210 I=0,ZPADLN/2
         SXY(I+1) = SXY(I+1) +
     X             CMPLX(XWORK(I*2+1),-XWORK(I*2+2)) *
     X             CMPLX(YWORK(I*2+1), YWORK(I*2+2))
210   CONTINUE
      IF (IOVER.NE.0) THEN
C        COPY LAST HALF OF CURRENT DATA TO FIRST HALF OF ARRAY
         DO 220 I=1,N/2
            XWORK(I) = SXWORK(I)
            YWORK(I) = SYWORK(I)
220      CONTINUE
C        FETCH A NEW SEGMENT OF DATA
         CALL GETXR(XWORK(N/2+1),N/2,NRET)
C        WE'RE DONE IF WE GET AN INCOMPLETE LAST SEGMENT
         IF (NRET.LT.N/2) GOTO 1000
         CALL GETYR(YWORK(N/2+1),N/2,NYRET)
         IF (NYRET.LT.NRET) NRET = NYRET
         IF (NRET.LT.N/2) GOTO 1000
         NRET = N/2+NRET
      ELSE
C        GET A NEW SEGMENT OF DATA, NOT OVERLAPPING SEGMENTS
         CALL GETXR(XWORK(1),N,NRET)
         IF (NRET.LT.N) GOTO 1000
         CALL GETYR(YWORK(1),N,NYRET)
         IF (NYRET.LT.NRET) NRET = NYRET
         IF (NRET.LT.N) GOTO 1000
      ENDIF
C     ZERO PAD DATA TO SPECIFIED LENGTH
      DO 230 I=NRET+1,ZPADLN
         XWORK(I) = 0.
         YWORK(I) = 0.
230   CONTINUE
C     RETURN TO PROCESS THE NEW SEGMENT
      GOTO 200
C
```

```
1000  CONTINUE
C
C     IF NOT WINOOWING WE ARE OONE.  IF WE ARE WINOOWING, WE MUST
C     FIRST CONVERT THE PRELIMINARY (NON-WINOOWEO) SXY ESTIMATE
C     TO THE TIME OOMAIN IN THE FORM OF A CORRELATION ESTIMATE
C     TO APPLY THE WINOOW--THEN WE CAN TRANSFORM IT BACK TO
C     THE FREOUENCY OOMAIN.
C
      IF (WINOOW.NE.O) THEN
         OO 310 I=O,ZPAOLN/2
            SXYWRK(2*I+1) = REAL(SXY(I+1))
            SXYWRK(2*I+2) = AIMAG(SXY(I+1))
310      CONTINUE
C        INVERSE FFT CREATES CORRELATION ESTIMATE
         CALL FFS(SXYWRK,ZPAOLN)
C        MULTIPLY POINT BY POINT WITH WINOOW FUNCTION.  NOTE THAT
C        THE CORRELATION IN SXYWRK IS FOLOEO SUCH THAT ELEMENT 1
C        CORRESPONOS TO WINOOW ELEMENT 6S, 2->66, 3->67, ... 64->128
C        ANO THEN SXYWRK(128)->WINOO(64), 127->63, ..., 6S->1 (ALL
C        OBVIOUSLY ASSUMING AN ARRAY LENGTH OF 128).  ALSO, THE
C        CORRELATION IS VALIO ONLY IF THE ARRAY HAS BEEN ZERO PAOOEO
C        TO A LENGTH OF 2*N OR MORE.
         OO 320 I=1,N
            SXYWRK(I) = SXYWRK(I)*WWORK(N+I)
320      CONTINUE
         OO 322 I=1,N
            SXYWRK(ZPAOLN-N+I) = SXYWRK(ZPAOLN-N+I)*WWORK(I)
322      CONTINUE
C        TRANSFORM BACK TO FREOUENCY OOMAIN
         CALL FFA(SXYWRK,ZPAOLN)
         OO 330 I=O,ZPAOLN/2
            SXY(I+1) = CMPLX(SXYWRK(I*2+1),SXYWRK(I*2+2))
330      CONTINUE
      ENOIF
C
C     CONVERT SUMMEO OATA TO AVERAGEO OATA ANO ACCOUNT FOR
C     1/N FACTOR IN FORWARO FOURIER TRANSFORM ROUTINE.
C
      AN = N
      IF (IPASCT.EO.1) AN = NRET1
      OO 1110 I=1,ZPAOLN/2+1
         SXY(I) = SXY(I)/(IPASCT*AN*AN)
1110  CONTINUE
      GOTO 9999
9100  CONTINUE
      WRITE(7,*)' ERROR--GETXR RETURNEO INSUFFICIENT OATA'
      IERR = 1
      GOTO 9999
9999  CONTINUE
```

```
RETURN
END
```

PROGRAM: SPWELCH EXEC

```
/*****************************************************************
*                                                                *
*       Vibration Analysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MOOULE NAME:
*
*       SPWELCH
*
* FUNCTION:
*
*       To compute a spectrel astimate from two input channels
*       using the traditional mathod as outline by Welch.
*       This routine is called from the VAS command.
*
* EXTERNAL ROUTINES REQUIRED:
*
*       SPWELCH  - Fortran program to do the actual work.
*                  If the MOOULE file is found on an accessed
*                  CMS disk, it will be run.  Otharwise a
*                  TEXT file will be loaded and run.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/
address command
arg inargs
parse upper var inargs in1fn in1ft in1fm in2fn in2ft in2fm ,
                       outfn outft outfm parms
if length(outfm) = 0 then signel help
infid1 = in1fn in1ft in1fm
infid2 = in2fn in2ft in2fm
outfid = outfn outft outfm
parse ver perms p1 parms
seqlen = 256
window = 0
ovarlap = 0
do while length(p1) > 0
  select
    when abbrev('SEQLEN',p1) then do
```

```
      parse var perms p2 perms
      if detatype(p2) <> 'NUM' then do
        say 'WELCH: Invelid SEOLEN "'p2'"'
        return 24
      end
      if seqlen > 2048 then seqlen = 2048
      seqlen = p2
    end
    when abbrev('WINOOW',p1) then do
      perse ver parms p2 parms
      window = selectwindow(p2)
      if window < 0 then return 24
    end
    when abbrev('OVERLAP',p1) then do
      overlep = 1
    end
    otherwise do
      say 'WELCH: Invelid paremeter "'p1'"'
      return 24
    end
  end
parse ver parms p1 perms
end
/* See if input dete exists */
'CMDCALL ESTATE' infid1
if rc <> 0 then return rc
'CMOCALL ESTATE' infid2
if rc <> 0 then return rc
/* Set up FILEOEFs */
'FILEOEF 10 OISK' infid1
'FILEOEF 11 OISK' infid2
'FILEOEF 30 OISK' outfid '(LRECL 80 RECFM V'
'FILEOEF 4 TERM (LRECL 80 RECFM F'
'FILEOEF 7 TERM'  /* Error output */
'FILEOEF 8 TERM'  /* Informetionel/Warning output */
'FILEOEF 9 TERM'  /* Stenderd output */
seqlen = right('0000'||seqlen,5)
zplen = right('0000'||2*seqlen,5)
pushline = seqlen||zplen||window||overlap
push pushline
'ESTATE SPWELCH MOOULE *'
if rc = 0 then 'SPWELCH'
else 'LOAO SPWELCH SPOISKIO(NOMAP CLEAR START'
if rc <> 0 then return rc
queue '* WELCH' left(windneme,12) right(seqlen,5)
return 0

SELECTWINOOW: procedure expose windname
  erg p2
```

```
  select
    when abbrev('RECTANGULAR',p2) then windval = 0
    when abbrev('TRIANGULAR',p2)|abbrev('BARTLETT',p2) then windval = 1
    when abbrev('HAMMING',p2) then windval = 2
    when abbrev('VONHANN',p2)|abbrev('HANNING',p2) then windval = 3
    when abbrev('KAISER',p2) then windval = 4
    when abbrev('BLACKMAN',p2) then windval = 5
    when abbrev('PARZEN',p2) then windval = 6
    otherwise do
      say 'WELCH:  Invalid window choice, "'p2'"'
      return -1
    end
  end
  windname = word('Rectangular Bertlett Hamming Hanning Kaiser '||,
                  'Bleckman Parzen',windval+1)
  return windval

LISTFILE: procedure
  arg fid
  'MAKEBUF'
  q = queued()
  'LISTFILE' fid '(LIFO ALLOC'
  if queued()-q <> 1 then do
    say 'WELCH: Confusion over number of stacked LISTFILE lines'
    'DROPBUF'
    return -1
  end
  pull lline
  perse var lline lfn lft lfm lrecfm llrecl lblks lrecs
  'DROPBUF'
  return lrecs

HELP:
  say 'SPWELCH:  Invalid peremeter list'
  return 24
```

PROGRAM: SPWELCH FORTRAN

```
*.
*****************************************************************
*                                                               *
*       Vibration Analysis System (VAS)                         *
*                                                               *
*       (c) Copyright 1986 by Brick A. Verser                   *
*                                                               *
*****************************************************************
*
* MOOULE NAME:
*
*       SPWELCH
*
* FUNCTION:
*
*       TO COMPUTE A CROSS POWER SPECTRAL OENSITY ESTIMATE
*       USING THE METHOO OF WEIGHTEO OVERLAPPING-SEGMENT
*       AVERAGING (WOSA) AS PROPOSEO BY PETER O. WELCH.
*
* INPUT UNITS:
*
*       04      PARAMETERS ANO OPTIONS.
*       10      CHANNEL 1 INPUT OATA IN REAL A4 FORMAT.
*       11      CHANNEL 2 INPUT OATA IN REAL A4 FORMAT.
*
* OUTPUT UNITS:
*
*       07      ERROR MESSAGES.
*       08      INFORMATIONAL ANO WARNING MESSAGES.
*       3D      POWER SPECTRAL OENSITY ESTIMATE IN COMPLEX 2A4 FORMAT.
*
* INPUT PARAMETERS:
*
*       RECORO 1:
*          1..5   INTEGER, NUMBER OF OATA VALUES TO INCLUOE
*                 IN EACH SEGMENT.
*          6..10  INTEGER, ZERO PAO LENGTH.  EACH SEGMENT WILL
*                 BE ZERO PAOOEO TO THIS LENGTH.  THIS FIELO
*                 ALSO OETERMINES THE NUMBER OF VALUES IN THE
*                 OUTPUT OATA SET.
*          11..11 INTEGER, WINOOW NUMBER.  OEFINEO IN WWINOO
*                 EXTERNAL SUBROUTINE.
*          12..12 INTEGER, OVERLAP FLAG. SET TO ONE IF SEGMENTS
*                 ARE TO BE OVERLAPPEO BY 50%, ZERO IF SEGMENTS
*                 ARE TO NOT BE OVERLAPPEO.
*
* OPERATION:
*
*       1.  INPUT PARAMETERS ARE PARSED TO OETERMINE THE SEGMENT
```

```
*           SIZE, ZERO PAO LENGTH, WINOOW, ANO OVERLAP.
*       2.  SUBROUTINE WELCH IS CALLEO TO COMPUTE THE PSO ESTIMATE.
*       3.  THE OUTPUT IS WRITTEN TO UNIT 30.
*
* EXTERNAL RDUTINES REQUIREO:
*
*       OISKIR - TO REAO REAL*4 FORMAT VALUES FROM OISK.
*       OISKOR - TO WRITE REAL*4 FORMAT VALUES TO OISK.
*       FFA    - TO COMPUTE A FORWARD DISCRETE FOURIER TRANSFORM.
*       WWINOO - TO PROVIOE WINOOWING FUNCTIONS.
*
* REVISION HISTORY:
*
*       1.D   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*.
      PARAMETER (IWRKLN=16)
      PARAMETER (MAXZPO=4096)
      COMPLEX SXY(MAXZPO/2+1)
      OIMENSION WORK30(IWRKLN)
      OIMENSION XWORK(MAXZPO+2),YWORK(MAXZPO+2)
      OIMENSION SXWORK(MAXZPO/2),SYWORK(MAXZPO/2)
      OIMENSION WWORK(MAXZPO), SXYWRK(MAXZPO+2)
      EXTERNAL GETX,GETY
      CHARACTER*80 INPARM
      NPTS = S12
      NPAO = 1D24
      IWINDO = D
      IOVER = 1
C
C     REAO PARAMETER LINE.  INPUT IS FIXEO FORMAT.
C
      REAO(4,'(A80)',ENO=90) INPARM
      REAO(UNIT=INPARM(1:10),FMT='(2I5)',ERR=SS) NPTS,NPAO
      REAO(UNIT=INPARM(11:12),FMT='(2I1)',ERR=SS) IWINOO,IOVER
      IF (NPTS.GT.MAXZPO) NPTS = MAXZPO
      IF (NPAO.GT.MAXZPO) NPAD = MAXZPO
C     WRITE(8,*)'NPTS,NPAD,IWINDO,IOVER=',NPTS,NPAO,IWINOO,IOVER
      GOTO 90
SS    CONTINUE
      WRITE(7,*)' ERROR REAOING INPUT PARAMETERS'
      GOTO 90
90    CONTINUE
C
C     CALL A SUBROUTINE TO DO THE ACTUAL COMPUTATIONS, PASSING
```

```
C     IT THE APPRDPRIATE ARRAYS AND THE ADDRESS DF THE TWO
C     SUBRDUTINES USED TD READ VALUES FOR EACH OF THE CHANNELS.
C
      CALL WELCH(SXY,NPAO,NPTS,XWDRK,YWDRK,SXWDRK,SYWORK,
     X           IWINDD,WWDRK,SXYWRK,GETX,GETY,IOVER,IERR)
C
C     WRITE THE RESULT TD OISK.
C
      CALL OISKOR(30,0.,1,IWRKLN,WDRK30,IWK30)
      OD 100 I=1,NPAD/2+1
         CALL DISKOR(30,REAL(SXY(I)),2,IWRKLN,WDRK30,IWK30)
         CALL DISKOR(30,AIMAG(SXY(I)),2,IWRKLN,WDRK30,IWK30)
100   CONTINUE
      CALL DISKDR(30,0.,3,IWRKLN,WORK30,IWK30)
      STDP
      END
      SUBROUTINE GETX(RDATA,IN,IDUT)
C
C     THIS ROUTINE IS CALLED TD READ A OISK FILE.
C     'RDATA' IS A REAL ARRAY WHICH IS FILLEO WITH 'IN' VALUES
C     READ FRDM UNIT 10.  'IDUT' IS SET TO THE NUMBER OF VALUES
C     ACTUALLY READ.
C
      PARAMETER (IWRKLN=16)
      OIMENSIDN WDRK10(IWRKLN)
      DIMENSIDN RDATA(*)
      DATA IWK110/0/
      I = 1
      IF (IWK110.EQ.-1) GDTO 995
      OD 10D I=1,IN
         CALL OISKIR(10,RDATA(I),IERR,IWRKLN,WDRK10,IWK110,IWK210)
         IF (IERR.NE.0) GDTO 990
9900     FORMAT(A4)
100   CDNTINUE
      I = IN+1
      GDTO 995
990   CDNTINUE
      IWK110 = -1
995   CDNTINUE
      IDUT = I-1
      RETURN
      ENO
      SUBROUTINE GETY(ROATA,IN,IDUT)
C
C     THIS ROUTINE IS CALLED TD READ A DISK FILE,
C     'RDATA' IS A REAL ARRAY WHICH IS FILLED WITH 'IN' VALUES
C     READ FRDM UNIT 10.  'IDUT' IS SET TD THE NUMBER DF VALUES
C     ACTUALLY REAO.
C
```

PROGRAM: SPWELCH FORTRAN

```
      PARAMETER (IWRKLN=16)
      DIMENSION WORK11(IWRKLN)
      DIMENSION RDATA(*)
      DATA IWK111/0/
      I = 1
      IF (IWK111.EQ.-1) GOTO 995
      DO 100 I=1,IN
         CALL DISKIR(11,RDATA(I),IERR,IWRKLN,WORK11,IWK111,IWK211)
         IF (IERR.NE.0) GOTO 990
100   CONTINUE
      I = IN+1
      GOTO 995
990   CONTINUE
      IWK111 = -1
995   CONTINUE
      IOUT = I-1
      RETURN
      END
      SUBROUTINE WELCH(SXY,ZPAOLN,N,XWORK,YWORK,SXWORK,SYWORK,
     X                 WINDOW,WWORK,SXYWRK,GETXR,GETYR,IOVER,IERR)
      EXTERNAL GETXR,GETYR
      INTEGER ZPAOLN,WINDOW,WLEN
      COMPLEX SXY(*)
      REAL XWORK(*),SXWORK(*), YWORK(*),SYWORK(*)
      REAL WWORK(*), SXYWRK(*)
C
C     RESULT IS PUT IN 'SXY' WHICH IS OF LENGTH ZPADLN/2+1.
C     'SXWORK' AND 'SYWORK' SHOULD BE N/2 LONG.
C     'XWORK' AND 'YWORK' SHOULD BE OF LENGTH ZPAOLN+2.
C
C     THE FOLLOWING ARRAYS ARE NEEDED ONLY IF WINDOWING:
C        'WWORK' SHOULD BE OF LENGTH N.
C        'SXYWRK' SHOULD BE OF LENGTH ZPAOLN+2.
C
      PI = 4*ATAN(1.)
      IERR = 0
      SHIFT = 0.
      IPASCT = 0
      DO 110 I=1,ZPADLN/2+1
         SXY(I) = 0.
110   CONTINUE
C
C     IF WINDOWING IS REQUESTED, CALL 'WWINDO' TO FILL ARRAY
C     'WWORK' WITH THE REAL-VALUED WINDOW FUNCTION.  'WPOWER'
C     IS SET TO THE POWER CONTAINED IN THE WINDOW FUNCTION
C     AND IS USED TO SCALE THE RESULTING PSD ESTIMATE.
C
      WPOWER = 1.
      IF (WINDOW.NE.0) THEN
```

```
         CALL WWINDO(WWORK,N,WINDOW,WPOWER,6.2832)
         WPOWER = N/WPOWER
      ENDIF
C
C     FILL WORK WITH 'N' REAL NUMBERS--ACTUAL COUNT PUT IN 'NRET'
C
      CALL CETXR(XWORK,N,NRET)
      IF (NRET.EQ.0) GOTO 910
C     ALLOW FIRST SET OF VALUES TO BE INCOMPLETE
C     IF (NRET.LT.N) GOTO 9100
      NRET1 = NRET
      CALL CETYR(YWORK,N,NYRET)
      IF (NYRET.LT.NRET) NRET = NYRET
      IF (NRET.EQ.0) GOTO 9100
C     IF (NRET.LT.N) GOTO 9100
      DO 120 I=NRET+1,ZPADLN
         XWORK(I) = 0.
         YWORK(I) = 0.
120   CONTINUE
C
200   CONTINUE
      IPASCT = IPASCT + 1
C
C     SAVE SECOND HALF OF DATA VALUES FOR OVERLAP ON NEXT PASS
C
      IF (IOVER.NE.0) THEN
         DO 205 I=1,N/2
            SXWORK(I) = XWORK(I+N/2)
            SYWORK(I) = YWORK(I+N/2)
205      CONTINUE
      ENDIF
C
C     REMOVE MEAN FROM EACH SEGMENT OF DATA
C
      XMEAN = 0
      YMEAN = 0
      DO 210 I=1,N
         XMEAN = XMEAN+XWORK(I)
         YMEAN = YMEAN+YWORK(I)
210   CONTINUE
      XMEAN = XMEAN/N
      YMEAN = YMEAN/N
      DO 215 I=1,N
         XWORK(I) = XWORK(I)-XMEAN
         YWORK(I) = YWORK(I)-YMEAN
215   CONTINUE
C
C     WINDOW THE DATA SEGMENT
C
```

```
      IF (WINDOW.NE.0) THEN
         DO 225 I=1,N
            XWORK(I) = XWORK(I)*WWORK(I)
            YWORK(I) = YWORK(I)*WWORK(I)
225      CONTINUE
      ENDIF
C
C     REMOVE MEAN FROM EACH WINDOWED SEGMENT OF DATA
C
      XMEAN = 0
      YMEAN = 0
      DO 230 I=1,N
         XMEAN = XMEAN+XWORK(I)
         YMEAN = YMEAN+YWORK(I)
230   CONTINUE
      XMEAN = XMEAN/N
      YMEAN = YMEAN/N
      DO 235 I=1,N
         XWORK(I) = XWORK(I)-XMEAN
         YWORK(I) = YWORK(I)-YMEAN
235   CONTINUE
C
C     TRANSFORM THE TWO VECTORS
C
      CALL FFA(XWORK,ZPADLN)
      CALL FFA(YWORK,ZPADLN)
C
C     COMPUTE THE POWER SPECTRAL DENSITY ESTIMATE AS
C     SXY = CONJ(X)*Y
C
      DO 240 I=0,ZPADLN/2
         SXY(I+1) = SXY(I+1) +
     X            CMPLX(XWORK(I*2+1),-XWORK(I*2+2)) *
     X            CMPLX(YWORK(I*2+1), YWORK(I*2+2))
240   CONTINUE
C
C     WE'RE DONE IF WE DIDN'T GET FULL LOAD OF DATA
C
      IF (IPASCT.EQ.1 .AND. NRET.LT.N) THEN
         GOTO 1000
      ELSE IF (IOVER.NE.0 .AND. NRET.LT.N/2) THEN
         GOTO 1000
      ELSE IF (IOVER.EQ.0 .AND. NRET.LT.N) THEN
         GOTO 1000
      ENDIF
C
C     HERE WE GO FOR THE NEXT CHUNK OF DATA.  THERE ARE
C     TWO PATHS BASED ON WHETHER OR NOT WE ARE OVERLAPPING
C     ADJACENT SEGMENTS.
```

```
C
      IF (IOVER.NE.0) THEN
C        COPY LAST HALF OF CURRENT DATA TO FIRST HALF OF ARRAY.
         DO 250 I=1,N/2
            XWORK(I) = SXWORK(I)
            YWORK(I) = SYWORK(I)
250      CONTINUE
C        FETCH A NEW CHUNK OF DATA
         CALL GETXR(XWORK(N/2+1),N/2,NRET)
C        DON'T PROCESS LAST SET OF DATA IF INCOMPLETE--THIS
C        PREVENTS A SHORT SEGMENT OF DATA FROM GETTING UNDUE
C        EMPHASIS IN THE AVERAGING.
         IF (NRET.LT.N/2) GOTO 1000
         CALL GETYR(YWORK(N/2+1),N/2,NYRET)
         IF (NYRET.LT.NRET) NRET = NYRET
         IF (NRET.LT.N/2) GOTO 1000
         NRET = N/2+NRET
      ELSE
C        GET A FULL-LENGTH PIECE IF NOT OVERLAPPING SEGMENTS
         CALL GETXR(XWORK(1),N,NRET)
         IF (NRET.LT.N) GOTO 1000
         CALL GETYR(YWORK(1),N,NYRET)
         IF (NYRET.LT.NRET) NRET = NYRET
         IF (NRET.LT.N) GOTO 1000
      ENDIF
C
C     ZERO PAD DATA TO SPECIFIED LENGTH
C
      DO 260 I=NRET+1,ZPADLN
         XWORK(I) = 0.
         YWORK(I) = 0.
260   CONTINUE
      GOTO 200
C
C
1000  CONTINUE
C
C     IF WE WINDOWED THE DATA, WE NEED TO ADJUST THE SPECTRAL
C     ESTIMATE SO AS TO CONSERVE POWER.  WE ALSO HAVE TO SCALE
C     IT BY THE NUMBER OF SEGMENTS ADDED TOGETHER AND ADJUST
C     FOR THE DFT 1/N FACTOR.
C
      AN = N
      IF (IPASCT.EQ.1) AN = NRET1
      DO 1110 I=1,ZPADLN/2+1
         SXY(I) = SXY(I)*WPOWER/(IPASCT*AN*AN)
1110  CONTINUE
      GOTO 9999
9100  CONTINUE
```

```
      WRITE(7,*)' ERROR--NO OATA IN FILE'
      IERR = 1
      GOTO 9999
9999  CONTINUE
      RETURN
      ENO
```

PROGRAM: SPCORRFT EXEC

```
/*****************************************************************
*                                                                *
*       Vibration Anelysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MOOULE NAME:
*
*       SPCORRFT
*
* FUNCTION:
*
*       To compute e correletion estimate from two input channels.
*       This routine is called from the VAS commend.
*
* EXTERNAL ROUTINES REQUIREO:
*
*       SPCORRFT - Fortran program to do the actual work.
*                  If tha MOOULE file is found on en accessad
*                  CMS disk, it will ba run.  Otherwise a
*                  TEXT file will be loaded end run.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/
 address commend
 erg inergs
 parse upper var inergs legs infn1 inft1 infm1 infn2 inft2 infm2 ,
                         outfn outft outfm rest
 if length(rest) <> 0 then signel help
 if length(outfm) = 0 then signel help
 infid1 = infn1 inft1 infm1
 infid2 = infn2 inft2 infm2
 outfid = outfn outft outfm
 optline = right(legs,5)
 'FILEOEF 10 OISK' infid1
 'FILEOEF 11 DISK' infid2
 'FILEOEF 30 OISK' outfid '(LRECL 80 RECFM V'
 'FILEOEF 4 TERM (LRECL 80 RECFM F'
 'FILEOEF 7 TERM'
 'FILEOEF 8 TERM'
```

PROGRAM: SPCORRFT EXEC

```
 'FILEOEF 9 TERM'
 push optline
 'ESTATE SPCORRFT MOOULE *'
 if rc = 0 then 'SPCORRFT'
 else 'LOAO SPCORRFT SPOISKIO(NOMAP CLEAR START'
 return rc

HELP:
 say 'SPCORRFT:  Invalid parameter list'
 return 24
```

```
*.
*****************************************************************
*                                                               *
*       Vibration Analysis System (VAS)                         *
*                                                               *
*       (c) Copyright 1986 by Brick A. Verser                   *
*                                                               *
*****************************************************************
*
* MOOULE NAME:
*
*       SPCORRFT
*
* FUNCTION:
*
*       TO COMPUTE A CROSS CORRELATION ESTIMATE.
*
* INPUT UNITS:
*
*       04      PARAMETERS ANO OPTIONS.
*       10      CHANNEL 1 INPUT OATA IN REAL A4 FORMAT.
*       11      CHANNEL 2 INPUT OATA IN REAL A4 FORMAT.
*
* OUTPUT UNITS:
*
*       07      ERROR MESSAGES.
*       08      INFORMATIONAL AND WARNING MESSAGES.
*       30      CORRELATION ESTIMATE IN REAL A4 FORMAT.
*
* INPUT PARAMETERS:
*
*       RECORO 1:
*          1..5   INTEGER, NUMBER OF CORRELATION LAGS TO COMPUTE.
*
* OPERATION:
*
*       1.  THE NUMBER OF LAGS TO COMPUTE IS REAO.
*       2.  AN INITIAL PASS OVER THE OATA IS MAOE TO COMPUTE
*           THE MEAN OF THE INPUT OATA.
*       3.  SUBROUTINE CORREL IS CALLEO TO REAO THE OATA
*           A SECONO TIME, COMPUTING THE CORRELATION USING
*           A FAST CORRELATION ALGORITHM (REF: RABINER & GOLO,
*           THEORY ANO APPLICATION OF OIGITAL SIGNAL PROCESSING,
*           PRENTICE-HALL, INC., ENGLEWOOO CLIFFS, NJ, 1975, PG 402).
*       4.  THE OUTPUT IS WRITTEN TO UNIT 30.
*
* EXTERNAL ROUTINES REQUIREO:
*
*       OISKIR - TO REAO REAL*4 FORMAT VALUES FROM OISK.
```

```
*       OISKOR - TO WRITE REAL*4 FORMAT VALUES TO DISK.
*       FFA    - TO COMPUTE A FORWARO OISCRETE FOURIER TRANSFORM.
*       FFS    - TO COMPUTE A INVERSE OISCRETE FOURIER TRANSFORM.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*.
      PARAMETER (IWRKLN=16)
      PARAMETER (MAXLAG=40S6)
      OIMENSION WORK1O(IWRKLN),WORK11(IWRKLN),WORK3O(IWRKLN)
      OIMENSION CXY(MAXLAG+2),XWORK(MAXLAG+2),YWORK(MAXLAG+2)
      OIMENSION XWORK2(MAXLAG/2)
      EXTERNAL GETX,GETY
      CHARACTER*8O INPARM
C
C     REAO PARAMETER LINE.  INPUT IS FIXEO FORMAT.
C
      REAO(4,'(A8O)',ENO=9O) INPARM
      REAO(UNIT=INPARM(1:5),FMT='(I5)',ERR=55) LAGCNT
      LAGCNT = LAGCNT*2
      IF (LAGCNT.GT.MAXLAG) LAGCNT = MAXLAG
      GOTO 9O
55    CONTINUE
      WRITE(7,*)' ERROR REAOING INPUT PARAMETERS'
9O    CONTINUE
      IWK11O =  0
      ICNT = O
      AM1O = O.
      AM11 = O.
C     FIRST PASS COMPUTES MEAN ANO NUMBER OF ELEMENTS
1OO   CONTINUE
         CALL OISKIR(1O,X,IERR,IWRKLN,WORK1O,IWK11O,IWK21O)
         IF (IERR.NE.O) GOTO 19O
         CALL DISKIR(11,Y,IERR,IWRKLN,WORK11,IWK111,IWK211)
         IF (IERR.NE.O) GOTO 19O
         ICNT = ICNT+1
         AM1O = AM1O + X
         AM11 = AM11 + Y
      GOTO 1OO
19O   CONTINUE
      IF (ICNT.EO.O) THEN
         WRITE(7,*)' NO INPUT OATA'
         GOTO 99O
```

```
      ENDIF
      REWIND 10
      REWIND 11
      AM10 = AM1D/ICNT
      AM11 = AM11/ICNT
      CALL CORREL(CXY,LAGCNT,XWORK,YWORK,XWORK2,
     X            GETX,AM10,GETY,AM11,IERR)
      CALL DISKDR(30,0.,1,IWRKLN,WORK30,IWK3D)
      DO 210 I=1,LAGCNT/2
         CXY(I) = CXY(I)/ICNT
         CALL DISKOR(30,CXY(I),2,IWRKLN,WORK3D,IWK30)
210   CONTINUE
      CALL DISKOR(30,0.,3,IWRKLN,WORK3D,IWK30)
990   CONTINUE
      STOP
      END
      SUBROUTINE GETX(RDATA,IN,IOUT)
      PARAMETER (IWRKLN=16)
      DIMENSION WORK10(IWRKLN)
      DIMENSION RDATA(*)
      DATA IWK110/0/
      I = 1
      IF (IWK110.EQ.-1) GOTO 995
      DO 100 I=1,IN
C        READ(10,99D0,END=99D)RDATA(I)
         CALL DISKIR(10,RDATA(I),IERR,IWRKLN,WORK10,IWK110,IWK210)
         IF (IERR.NE.0) GOTO 990
99D0     FORMAT(A4)
100   CONTINUE
      I = IN+1
      GOTO 995
990   CONTINUE
      IWK110 = -1
995   CONTINUE
      IOUT = I-1
      RETURN
      END
      SUBROUTINE GETY(RDATA,IN,IOUT)
      PARAMETER (IWRKLN=16)
      DIMENSION WORK11(IWRKLN)
      DIMENSION RDATA(*)
      DATA IWK111/D/
      I = 1
      IF (IWK111.EQ.-1) GOTO 995
      DO 100 I=1,IN
C        READ(11,9900,END=990)RDATA(I)
         CALL DISKIR(11,RDATA(I),IERR,IWRKLN,WORK11,IWK111,IWK211)
         IF (IERR.NE.0) GOTO 990
9900     FORMAT(A4)
```

```
100   CONTINUE
      I = IN+1
      GOTO 995
990   CONTINUE
      IWK111 = -1
995   CONTINUE
      IOUT = I-1
      RETURN
      END
      SUBROUTINE CORREL(CXY,LAGCNT,XWORK,YWORK,XWORK2,
     X                  GETXR,XMEAN,GETYR,YMEAN,IERR)
      EXTERNAL GETXR,GETYR
      REAL CXY(*),XWORK(*), YWORK(*), XWORK2(*)
      COMPLEX C
C
C     LAGCNT POINTS ARE PUT IN 'CXY' WHICH IS OF LENGTH LAGCNT+2.
C     'XWORK' AND 'YWORK' SHOULD BE OF LENGTH LAGCNT+2.
C     'XWORK2' SHOULD BE OF LENGTH LAGCNT/2.
C
      PI = 4*ATAN(1.)
      IERR = 0
      SHIFT = 0.
      IPASCT = 0
      DO 110 I=1,LAGCNT+2
         CXY(I) = 0.
         XWORK(I) = 0.
110   CONTINUE
C
C     FILL WORK WITH 'N' REAL NUMBERS--ACTUAL COUNT PUT IN 'NRET'
C
      CALL GETXR(XWORK,LAGCNT,NRET)
      IF (NRET.EQ.0) GOTO 9100
      CALL GETYR(YWORK,LAGCNT/2,NYRET)
      IF (NYRET*2.LT.NRET) NRET = NYRET*2
      IF (NRET.EQ.0) GOTO 9100
C
200   CONTINUE
      IPASCT = IPASCT + 1
C     SAVE LAST HALF OF XWORK FOR NEXT PASS
      DO 205 I=1,LAGCNT/2
         XWORK2(I) = XWORK(I+LAGCNT/2)
205   CONTINUE
C     REMOVE MEAN FROM EACH SEQUENCE
      DO 210 I=1,NRET
         XWORK(I) = XWORK(I)-XMEAN
210   CONTINUE
      DO 211 I=1,NYRET
         YWORK(I) = YWORK(I)-YMEAN
211   CONTINUE
```

PROGRAM: SPCORRFT FORTRAN

```
C     IF WE GOT LESS THAN A FULL XWORK, ZERO PAD IT
      DO 202 I=LAGCNT/2+NRET+1,LAGCNT
         XWORK(I) = 0.
2D2   CONTINUE
C     ZERO PAD YWORK TO 'LAGCNT' VALUES
      DO 2D3 I=NYRET+1,LAGCNT
         YWORK(I) = D.
2D3   CONTINUE
      CALL FFA(XWORK,LAGCNT)
      CALL FFA(YWORK,LAGCNT)
      DO 24D I=1,LAGCNT+2-1,2
         C = CMPLX(XWORK(I), XWORK(I+1)) *
     X       CMPLX(YWORK(I),-YWORK(I+1))
         CXY(I) = CXY(I) + REAL(C)
         CXY(I+1) = CXY(I+1) + AIMAG(C)
24D   CONTINUE
      CALL GETXR(XWORK(1+LAGCNT/2),LAGCNT/2,NRET)
      CALL GETYR(YWORK(1),LAGCNT/2,NYRET)
      IF (NYRET.LT.NRET) NRET = NYRET
      IF (NYRET.LE.D) GOTO 1DDO
C     MOVE SAVED LAST HALF OF X DATA TO FRONT OF XWORK
      DO 26D I=1,LAGCNT/2
         XWORK(I) = XWORK2(I)
26D   CONTINUE
      GOTO 2DO
C
1DDD  CONTINUE
C     CONVERT SUMMED DATA TO AVERAGED DATA
C     DO 111D I=1,LAGCNT+2
C        CXY(I) = CXY(I)/IPASCT
C11D  CONTINUE
C     INVERSE FFT CREATES CORRELATION ESTIMATE
      CALL FFS(CXY,LAGCNT)
      GOTO 9999
91DD  CONTINUE
      WRITE(7,*)' ERROR--GETXR RETURNED INSUFFICIENT DATA'
      IERR = 1
      GOTO 9999
9999  CONTINUE
      RETURN
      END
```

PROGRAM: SPCOHER EXEC

```
/*****************************************************************
*                                                                *
*       Vibration Analysis Systam (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Varsar                    *
*                                                                *
*****************************************************************
*
* MOOULE NAME:
*
*       SPCOHER
*
* FUNCTION:
*
*       To computa a coharanca function astimata from two
*       auto-spactrum astimatas and their cross-spectrum.
*       This routina is called from the VAS command.
*
* EXTERNAL ROUTINES REQUIREO:
*
*       SPCOHER  - Fortran program to do tha actual work.
*
* REVISIDN HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/
 addrass command
 arg inargs
 parsa uppar var inargs infn1 inft1 infm1 infn2 inft2 infm2 ,
                        infn12 inft12 infm12 outfn outft outfm rast
 if length(outfm) = 0 than signal halp
 if length(rast) <> 0 than signal halp
 infid1 = infn1 inft1 infm1
 infid2 = infn2 inft2 infm2
 infid12 = infn12 inft12 infm12
 outfid = outfn outft outfm
 'FILEOEF 10 OISK' infid1
 'FILEOEF 11 DISK' infid2
 'FILEOEF 12 OISK' infid12
 'FILEOEF 3D OISK' outfid '(LRECL 80 RECFM V'
 'FILEDEF 4 TERM (LRECL 8D RECFM F'
 'FILEDEF 7 TERM'
 'FILEOEF 8 TERM'
 'FILEOEF 9 TERM'
```

PROGRAM: SPCOHER EXEC

```
 'SPCOHER'
 return rc

HELP:
 say 'SPCOHER:  Invalid parameter list'
 return 24
```

PROGRAM: SPCOHER FORTRAN

```
*.
*****************************************************************
*                                                               *
*       Vibration Analysis System (VAS)                         *
*                                                               *
*       (c) Copyright 1986 by Brick A. Verser                   *
*                                                               *
*****************************************************************
*
* MOOULE NAME:
*
*       SPWELCH
*
* FUNCTION:
*
*       COMPUTE A COHERENCE FUNCTION FROM TWO AUTO-SPECTRUM
*       ESTIMATES ANO ONE CROSS-SPECTRUM ESTIMATE.
*       THE COHERENCE FUNCTION COMPUTEO IS:
*
*                       ABS(S12(F))
*            K12(F) = -----------
*                       S1(F)*S2(F)
*
* INPUT UNITS:
*
*       10      CHANNEL 1 AUTO-SPECTRUM ESTIMATE IN CMPLX 2A4 FORMAT.
*       11      CHANNEL 2 AUTO-SPECTRUM ESTIMATE IN CMPLX 2A4 FORMAT.
*       12      CHAN 1:2 CROSS-SPECTRUM ESTIMATE IN CMPLX 2A4 FORMAT.
*
* OUTPUT UNITS:
*
*       07      ERROR MESSAGES.
*       08      INFORMATIONAL ANO WARNING MESSAGES.
*       30      COHERENCE FUNCTION IN REAL A4 FORMAT.
*
* OPERATION:
*
*       1.  THE OATA IS REAO FROM EACH OF THE THREE INPUTS
*           ANO THE COHERENCE FUNCTION IS COMPUTEO
*           POINT-BY-POINT, WRITING THE RESULT AS THE
*           COMPUTATION PRCCEEOS.
*
* EXTERNAL ROUTINES REOUIREO:
*
*       OISKIR - TO REAO REAL*4 FORMAT VALUES FROM OISK.
*       OISKOR - TO WRITE REAL*4 FORMAT VALUES TO OISK.
*
* REVISION HISTORY:
*
```

PROGRAM: SPCOHER FORTRAN

```
*       1.0   ORIGINAL CODE
*
*       REVISION        DATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*.
      PARAMETER (IWRKLN=16)
      DIMENSION WORK10(IWRKLN),WORK11(IWRKLN),WORK12(IWRKLN)
      DIMENSION WORK30(IWRKLN)
      COMPLEX C12
      CALL DISKOR(30,0.,1,IWRKLN,WORK30,IWK30)
      IWK110 = 0
      IWK111 = 0
      IWK112 = 0
210   CONTINUE
            CALL DISKIR(10,C1,IERR,IWRKLN,WORK10,IWK110,IWK210)
            IF (IERR.NE.0) GOTO 250
            CALL DISKIR(10,XI,IERR,IWRKLN,WORK10,IWK110,IWK210)
            IF (IERR.NE.0) GOTO 250
            CALL DISKIR(11,C2,IERR,IWRKLN,WORK11,IWK111,IWK211)
            IF (IERR.NE.0) GOTO 250
            CALL DISKIR(11,XI,IERR,IWRKLN,WORK11,IWK111,IWK211)
            IF (IERR.NE.0) GOTO 250
            CALL DISKIR(12,XR,IERR,IWRKLN,WORK12,IWK112,IWK212)
            IF (IERR.NE.0) GOTO 250
            CALL DISKIR(12,XI,IERR,IWRKLN,WORK12,IWK112,IWK212)
            IF (IERR.NE.0) GOTO 250
            C12 = CMPLX(XR,XI)
            COH = ABS(C12)**2 / (C1*C2)
            CALL DISKOR(30,COH,2,IWRKLN,WORK30,IWK30)
      GOTO 210
250   CONTINUE
      CALL DISKOR(30,0.,3,IWRKLN,WORK30,IWK30)
      STOP
      END
```

PROGRAM: SPPPLOT4 EXEC

```
/*****************************************************************
*                                                                *
*       Vibretion Analysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MOOULE NAME:
*
*       SPPPLOT4
*
* FUNCTION:
*
*       To drive the SPPLOT4 Fortran program to produce
*       plotted output.  This routine is called by
*       the VAS command.
*
* EXTERNAL ROUTINES REOUIREO:
*
*       SPMAGPHA - To seperete a complex formet spectrum
*                  into magnitude and phase componants.
*       SPPLOT4  - Fortren program to plot date.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/
 eddress commend
 arg oparms
 persa upper var oparms parms
 diskmode = 'A'
 /* Set defaults for global options */
 gscalefact = 1.0       /* Don't anlerge or raduce plot */
 gtekcal = 'TEKTRONIX'  /* Assume interactive plotting */
 glinelogx = 'LINE'     /* Oefault to linear X-axis */
 glinelogy = 'LINE'     /*            lineer Y-exis */
 gdegrad = 'OEGREES'    /* Plot phese in degrees */
 glinesdots = 'LINES'   /* Connect valuas with lines */
 gmagphase = 'MP'       /* Plot both mag and phase of cross spectrum */
 gfrompt = 1            /* Plot beginning with first point */
 gforpts = 999999       /* Plot all points */
 gpowamp = 'FOUR'       /* Assuma we want Fouriar amplituda spectrum */
```

```
parse ver parms fdesc '(' opts ')' parms2
if words(fdesc) = D then do /* We ere beginning with global options */
  do while words(opts) > 0
    perse ver opts opt opts
    if abbrev('LOGX',opt,4) then glinelogx = 'LOG'
    else if ebbrev('LINEX',opt,5) then glinelogx = 'LINE'
    else if ebbrev('LOGY',opt,4) then glinelogy = 'LOG'
    else if abbrev('LINEY',opt,5) then glinelogy = 'LINE'
    else if abbrev('RADIANS',opt,3) then gdegred = 'RAOIANS'
    else if abbrev('OEGREES',opt,3) then gdegred = 'DEGREES'
    else if abbrev('OOTS',opt,3) then glinesdots = 'DOTS'
    else if abbrev('LINES',opt,3) then glinesdots = 'LINES'
    else if ebbrev('MAGNITUOE',opt,3) then gmagphase = 'MAGNITUOE'
    else if ebbrev('PHASE',opt,3) then gmagphase = 'PHASE'
    else if ebbrev('MP',opt,2) then gmagphase = 'MP'
    else if ebbrev('PM',opt,2) then gmagphase = 'MP'
    else if ebbrev('CALCOMP',opt,3) then gtekcal = 'CALCOMP'
    else if abbrev('TEKTRONIX',opt,3) then gtekcel = 'TEKTRONIX'
    else if abbrev('POWER',opt,3) then gpowamp = 'POWER'
    else if abbrev('PSO',opt,3) then gpowamp = 'POWER'
    else if abbrev('FOURIER',opt,4) then gpowamp = 'FOUR'
    else if abbrev('FROM',opt,2) then do
      parse ver opts gfrompt opts  /* FROM option followed by number */
      if datatype(gfrompt) <> 'NUM' then do
        say 'Invelid FROM option "'gfrompt'"'
        return 24
      end
    end
    else if ebbrev('FOR',opt,2) then do
      perse ver opts gforpts opts  /* FOR option followed by number */
      if detatype(gforpts) <> 'NUM' then do
        sey 'Invalid FOR option "'gforpts'"'
        return 24
      end
    end
    else if ebbrev('SCALE',opt,3) then do
      perse ver opts gscalefact opts  /* SCALE option followed by number */
      if datatype(gscalefact) <> 'NUM' then do
        say 'Invelid scale fector "'gscalefact'"'
        return 24
      end
    end
    else do   /* That was our lest hope for e valid option */
      say 'Invelid option "'opt'"'
      return 24
    end
  end
  parse ver parms2 fdesc '(' opts ')' perms2
end
```

```
fidcnt = 0
do while words(fdesc) > 0
  fidcnt = fidcnt+1
  perse ver fdesc fn type chenx cheny fdesc2
  if words(chany) = 0 then do  /* Gotte heve complete file spec */
    say 'Incomplete fileid "'fn fttype chanx'" specified'
    return 24
  end
  if ebbrev('OATA',type) then dt = 'O'
  else if ebbrev('SPECTRUM',type) then dt = 'S'
  else if ebbrev('CORRELATION',type,3) then dt = 'C'
  else if ebbrev('TRANSFER',type) then dt = 'T'
  else if ebbrev('COHERENCE',type,3) then dt = 'K'
  else if type = '.' then dt = '.'
  else do
    say 'Invalid datetype paremeter "'type'"'
    return 24
  end
  if dt = '.' then do
    fid.fidcnt = '. . .'
  end
  else do
    ft = dt||chenx||chany
    fid.fidcnt = fn ft '*'
    'CMOCALL ESTATE' fid.fidcnt
    if rc <> 0 then return rc
  end
  linelogx.fidcnt = glinelogx
  linelogy.fidcnt = glinelogy
  degred.fidcnt = gdegred
  linesdots.fidcnt = glinesdots
  magphese.fidcnt = gmagphese
  frompt.fidcnt = gfrompt
  forpts.fidcnt = gforpts
  powemp.fidcnt = gpowamp
  if words(fdesc2)=0 & words(opts)>0 then do while words(opts)>0
    perse ver opts opt opts
    if ebbrev('LOGX',opt,4) then linelogx.fidcnt = 'LOG'
    else if ebbrev('LINEX',opt,5) then linelogx.fidcnt = 'LINE'
    else if abbrev('LDGY',opt,4) then linelogy.fidcnt = 'LOG'
    else if ebbrev('LINEY',opt,S) then linelogy.fidcnt = 'LINE'
    else if ebbrev('RADIANS',opt,3) then degrad.fidcnt = 'RAOIANS'
    else if ebbrev('OEGREES',opt,3) then degrad.fidcnt = 'OEGREES'
    else if ebbrev('OOTS',opt,3) then linesdots.fidcnt = 'DOTS'
    else if ebbrev('LINES',opt,3) then linesdots.fidcnt = 'LINES'
    else if ebbrev('MAGNITUDE',opt,3) then magphese.fidcnt = 'MAGNITUOE'
    else if abbrev('PHASE',opt,3) then megphase.fidcnt = 'PHASE'
    else if ebbrev('MP',opt,2) then megphese.fidcnt = 'MP'
```

```
    else if abbrev('PM',opt,2) then megphesa.fidcnt = 'MP'
    else if abbrev('POWER',opt,3) then powamp.fidcnt = 'POWER'
    else if ebbrev('PSO',opt,3) then powamp.fidcnt = 'POWER'
    else if abbrav('FOURIER',opt,4) then powamp.fidcnt = 'FOUR'
    else if ebbrev('FROM',opt,2) then do
      parse var opts frompt.fidcnt opts  /* FROM option followed by number */
      if datetype(frompt.fidcnt) <> 'NUM' then do
        say 'Invalid FROM option "'frompt.fidcnt'"'
        return 24
      end
    end
    else if ebbrev('FOR',opt,2) then do
      parse var opts forpts.fidcnt opts  /* FOR option followed by number */
      if datetype(forpts.fidcnt) <> 'NUM' then do
        say 'Invalid FOR option "'forpts.fidcnt'"'
        return 24
      end
    end
    else do
      say 'Invalid option "'opt'"'
      return 24
    end
  end
  if words(fdesc2) = 0 then parse var parms2 fdesc '(' opts ')' parms2
  else fdesc = fdesc2
end

/* Plist has been parsed, so now we can plot */
maxfidcnt = fidcnt
say 'Plotting' maxfidcnt 'files'
if maxfidcnt = 0 than do
  say 'No files specified'
  return 24
end
workfids = '';  utcnt = 0;  inunit = 12
'MAKEBUF'
if gtakcel = 'TEKTRONIX' then qline = 'T'
else qlina = 'C'
qlina = qline||right(gscalefact,15)
queue qline   /* Set output davice and sceling */
do fidcnt = 1 to maxfidcnt
  fid = fid.fidcnt
  fn = word(fid,1); ft = word(fid,2); fm = word(fid,3)
  if fid = '. . .' than do   /* No plot in this position */
    'FILEOEF' inunit+40 'TERM (LRECL 80 RECFM F'
    queue '*SKIPIT'
    inunit = inunit+1
    iterata
  end
```

PROGRAM: SPPPLOT4 EXEC

```
'MAKEBUF'       /* Create a stack just for the EXECIO */
q = queued()
'EXECIO 10 DISKR' fn 'IOO' fm '(FINIS'
q = queued()-q;
sps = 1; infoline = '* UNKNOWN UNKNOWN UNKNOWN'
if q >= 1 then do
  pull sps ddate dtime .
  sps = sps/1
  q = q-1
end
if q >= 8 then do i=1 to 8
  pull lpf.i loccode.i
  loccode.i = strip(loccode.i)
  if datetype(lpf.i,'N') then lpf.i = lpf.i/1
  q = q-1
end
if q >= 1 then pull infoline
'DROPBUF'         /* Drop EXECIO stack */
method = word(infoline,2)
windname = word(infoline,3)
seqlen = word(infoline,4)
if seqlen <> 'UNKNOWN' then seqlen = seqlen/1
fttype=substr(ft,1,1); chanx=substr(ft,2,1); cheny=substr(ft,3,1)
if linesdots.fidcnt = 'LINES' then linesdots = 'L'
else linesdots = 'P'
if linelogx.fidcnt = 'LINE' then linelogx = 'N'
else linelogx = 'L'
if linelogy.fidcnt = 'LINE' then linelogy = 'N'
else linelogy = 'L'
if degred.fidcnt = 'DEGREES' then degred = 'D'
else degred = 'R'
frompt = frompt.fidcnt
forpts = forpts.fidcnt
select
  when fttype='T' | fttype='S' then do   /* Complex data */
    utcnt = utcnt+1; workft1 = 'CMSUT'utcnt
    workfid1 = 'SPEC' workft1 diskmode
    utcnt = utcnt+1; workft2 = 'CMSUT'utcnt
    workfid2 = 'SPEC' workft2 diskmode
    workfids = workfids workfid1 workfid2
    /* Turn into magnitude and phase, converting a PSD into a */
    /* simple FAS if required */
    if powamp.fidcnt = 'FOUR' then spmpopt = 'SORT'; else spmpopt = ''
    'EXEC SPMAGPHA' fn ft fm workfid1 workfid2 spmpopt
    /* Handle magnitude plot */
    if magphase.fidcnt <> 'PHASE' then do /* Option may prevent plot */
      'FILEDEF' inunit 'DISK' workfid1
      'FILEDEF' inunit+40 'TERM (LRECL 80 RECFM F'
      inunit = inunit+1
```

```
  queue 'FREQUENCY IN HZ'
  if fttype <> 'S' then qline = 'AMPLITUDE'
  else if spmpopt = 'SDRT' then qline = 'MAGNITUDE IN CM/SEC'
                           else qline = 'MAGNITUDE IN (CM/SEC)**2'
  queue qline
  qline = 'TEST:' left(fn,8)||'  '||ddate dtime
  if fttype <> 'S' then qline = qline ' TRANSFER FUNCTION'
  else if spmpopt = 'SDRT' then qline = qline ' FAS ESTIMATE'
                           else qline = qline ' PSD ESTIMATE'
  if method <> 'WELCH' then qline = qline '(T)'
  queue qline
  if chenx = chany then do
    qline = 'CHANNEL:' chanx '('loccode.chanx')'
    qline = qline ' LPF FRED:' lpf.chanx
  end
  else qline = 'CHANNELS:' chanx':'chany
  queue qline
  qline = 'WINDDW:' windname ' PTS/SEGMENT:' seqlen
  qline = qline ' SAMPLE RATE:' sps
  queue qline
  xvel = D; xinc = 1
  if seqlen <> 'UNKNOWN' then xinc = sps/2/seqlen
  pline = right(xvel,15)||right(xinc,16)||right(frompt,6)
  pline = pline||right(forpts,6)||linesdots||' '
  pline = pline||linelogx||linelogy
  queue pline
end
/* Handle phese plot (if needed) */
if fttype='S' & chanx<>cheny & magphase.fidcnt <> 'MAGNITUDE' ,
 then do /* Spectrum plot mey need phase */
  'FILEDEF' inunit 'DISK' workfid2
  'FILEDEF' inunit+4D 'TERM (LRECL 8D RECFM F'
  inunit = inunit+1
  if degred = 'R' then temp = 'RADIANS'; else temp = 'DEGREES'
  queue 'FREDUENCY IN HZ'; queue 'PHASE IN' temp
  qline = 'TEST:' left(fn,8)||'  '||ddate dtime
  if spmpopt = 'SDRT' then qline = qline ' FAS ESTIMATE'
                      else qline = qline ' PSD ESTIMATE'
  if method <> 'WELCH' then qline = qline ' (T)'
  queue qline
  if chenx = chany then do
    qline = 'CHANNEL:' chanx '('loccode.chanx')'
    qline = qline ' LPF FRED:' lpf.chanx
  end
  else qline = 'CHANNELS:' chanx':'chany
  queue qline
  qline = 'WINDOW:' windneme ' PTS/SEGMENT:' seqlen
  qline = qline ' SAMPLE RATE:' sps
  queue qline
```

```
        xval = 0; xinc = 1
        if seqlen <> 'UNKNOWN' then xinc = sps/2/seqlen
        pline = right(xval,15)||right(xinc,16)||right(frompt,6)
        pline = pline||right(forpts,6)||linesdots||degrad
        pline = pline||linelogx||linelogy
        queue pline  /* Same X scaling, but Y-axis is from -PI to PI */
      end
    end
    otherwise do
      'FILEOEF' inunit 'OISK' fn ft fm
      'FILEDEF' inunit+40 'TERM (LRECL 80 RECFM F'
      inunit = inunit+1
      if fttype='C' | fttype='O' then queue 'TIME IN SECONOS'
                                 else queue 'FREQUENCY IN HZ'
      if fttype = 'O' then queue 'AMPLITUDE IN CM/SEC/SEC'
                      else queue 'AMPLITUOE'
      qline = 'TEST:' left(fn,8)||'  '||ddate dtime
      if fttype='K' then qline = qline ' COHERENCE'
      else if fttype='C' then qline = qline ' CORRELATION'
      else if fttype='O' then qline = qline ' OATA'
      if method <> 'WELCH' & fttype='K' then qline = qline '(T)'
      queue qline
      if fttype='K' then do  /* Coherence */
        qline = 'CHANNELS:' chenx':'chany
        queue qline
        queue ' '
        xvel = 0; xinc = 1
        if seqlen <> 'UNKNOWN' then xinc = sps/2/seqlen
      end
      else do          /* Correlation or Oata */
        if chenx = chany then do
          qline = 'CHANNEL:' chenx '('loccode.chanx')'
          qline = qline ' LPF FREO:' lpf.chanx
        end
        else qline = 'CHANNELS:' chenx':'chany
        queue qline
        queue 'SAMPLE RATE:' sps
        xval = 0; xinc = 1
        if sps <> 'UNKNOWN' then xinc = 1/sps
      end
      pline = right(xval,15)||right(xinc,15)||right(frompt,6)
      pline = pline||right(forpts,6)||linesdots||' '
      pline = pline||linelogx||linelogy
      queue pline
    end
  end  /* SELECT */
end  /* Once for each file */
'FILEOEF' inunit+40 'OUMMY'
'FILEOEF 4 TERM (LRECL 80 RECFM F'
```

PROGRAM: SPPPLOT4 EXEC

```
'ESTATE SPPLOT4 MOOULE *'
if rc <> 0 then 'LOAD SPPLOT4 SPOISKIO(NOMAP CLEAR START'
else 'SPPLOT4'
'OROPBUF'
do while words(workfids)>0
  parse var workfids fn ft fm workfids
  'ERASE' fn ft fm
end
return 0
```

PROGRAM: SPPLOT4 FORTRAN

```
*.
*****************************************************************
*                                                               *
*       Vibration Analysis System (VAS)                         *
*                                                               *
*       (c) Copyright 1986 by Brick A. Verser                   *
*                                                               *
*****************************************************************
*
* MOOULE NAME:
*
*       SPPLOT
*
* FUNCTION:
*
*       TO PLOT REAL*4 OATA ON TEKTRONIX OR CALCOMP OEVICES.
*
* INPUT UNITS:
*
*       12..29 INPUT OATA TO BE PRINTEO IN REAL A4 FORMAT.
*       62..69 PARAMETERS ANO OPTIONS FOR EACH OATA FILE PRINTEO.
*
* OUTPUT UNITS:
*
*       95      THE PRINTEO PAGES.
*
* INPUT PARAMETERS (UNIT 4):
*
*       RECORO 1:
*          1..1   'T' TO PLOT ON TEKTRONIX, 'C' FOR CALCOMP.
*          2..16  F15.0 FORMAT PLOT SCALE FACTOR.
*
* INPUT PARAMETERS (UNIT5 52..69):
*
*       RECORO 1:
*          1..13  LABEL FOR INOEPENOENT OATA AXIS.
*       RECORO 2:
*          1..13  LABEL FOR OEPENOENT OATA AXIS.
*       RECOROS 3, 4 ANO 5:
*          1..72  GRAPH TITLE LINES.
*       RECORO 6:
*          1..15  F15.0 FORMAT INITIAL INOEPENOENT VARIABLE VALUE.
*          16..30 F15.0 FMT INOEPENOENT VARIABLE INCREMENT.
*          31..36 I6 FMT NUMBER OF THE FIRST POINT TO PRINT.
*          37..42 I6 FMT NUMBER OF POINTS TO PRINT.
*          43..43 ' ' TO PLOT LINES BETWEEN VALUES,
*                 'B' TO PLOT LINES ANO POINTS BETWEEN VALUES,
*                 'P' TO PLOT POINTS ONLY AT VALUES.
*          44..44 'O' TO PLOT PHASE IN OEGREES (FROM -200 TO +200).
```

```
*                'R' TD SCALE PLDT FDR PHASE FRDM -PI TD PI.
*          4S..4S 'L' TD PLOT LDGARITHMIC X-AXIS, 'N' FOR LINEAR.
*          46..46 'L' TD PLDT LDGARITHMIC Y-AXIS, 'N' FDR LINEAR.
*
* DPERATIDN:
*
*       1.  READ AND PARSE GLDBAL PARAMETER LINE.
*       2.  INITIALIZE THE PLDTTING SYSTEM.
*       3.  FDR EACH INPUT FILE, THE PARAMETERS ARE READ.
*           IF END-DF-FILE ENCDUNTERED, EXIT.
*       4.  THE DATA IS READ, SCALED, ANO PLDTTED.
*       S.  GDTD STEP 3.
*
* EXTERNAL RDUTINES REQUIRED:
*
*       OISKIR - TD READ REAL*4 FDRMAT VALUES FRDM DISK.
*
*       THE KSU CALCDMP GRAPHICS LIBRARY IS ALSD USED.
*
* REVISIDN HISTDRY:
*
*       1.D   DRIGINAL CDDE
*
*       REVISIDN        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.D            JUNE 26, 1986           BRICK VERSER
*
*.
      PARAMETER (MAXDIM=4D98)
      PARAMETER (IWRKLN=16)
      DIMENSIDN X(MAXDIM),Y(MAXDIM)
      DIMENSIDN DSKWRK(IWRKLN)
      CHARACTER*1 INCHAR,LPFLG,YPIFLG,XLNLDG,YLNLDG
      CHARACTER*8D INLINE,TITL1,TITL2,TITL3
      CHARACTER*64 XAXLB,YAXLB
      LOGICAL TOPFLG,CALFLG,CMPLFL
C
C     INITIALIZE CDMMDN BLOCK FDR SAXIS, ANO DAXIS
C
      COMMDN/AXICDM/MOEL,STIC,ALTIC,HNUM,HIBCD,DIS1,DIS2,CYMIN1,CYMIN2,
     *HNUM1,HNUM2
      MDEL = 2
      STIC = D.DS
      ALTIC = D.D7
C     NEGATIVE HNUM CAUSES AXIS LABELS TO SHIFT RIGHT, NOT CENTER.
      HNUM = -D.1D
      HIBCD = D.14
      DIS1 = D.03
      DIS2 = D.0S
```

```
      CYMIN1 = 2.0
      CYMIN2 = 2.S
      HNUM1 = 0.12
      HNUM2 = 0.10
C
      PI = 4*ATAN(1.)
      TOPFLG = .FALSE.
C
C     PARM LINE IS 'T'(EKTRONIX) OR 'C'(ALCOMP) ANO SCALE FACTOR
C
      READ(4,'(A80)',ENO=30) INLINE
      INCHAR = INLINE(1:1)
      READ(UNIT=INLINE(2:16),FMT='(F1S.0)') SCAFAC
30    CONTINUE
C
C     INITIALIZE KSU PLOTTING PACKAGE.
C     'SYMSEL E' SELECTS THE SOFTWARE SYMBOL ROUTINE.
C     'OEVICE ...' SELECTS THE APPROPRIATE HAROWARE ORIVER PACKAGE.
C
      IF (INCHAR.EO.'C') THEN
         CALFLG = .TRUE.
         CALL PLTOPT('SYMSEL E OEVICE CALCOMP#')
      ELSE
         SCAFAC = SCAFAC*.9S
         CALFLG = .FALSE.
         CALL PLTOPT('SYMSEL E OEVICE TEK TEKOEV 4010#')
      ENOIF
C
C     INITIALIZE THE PLOTTER ANO ESTABLISH AN ORIGIN LEAVING ROOM
C     FOR THE X-AXIS LABELS.
C
      CALL PLOTS
      CALL FACTOR(SCAFAC)
      XORIG = 2.0
      IF (CALFLG .ANO. SCAFAC.EQ. .S) XORIG = 4.0
      YORIG = .S
      IF (CALFLG .ANO. SCAFAC.EQ. .S) YORIG = 4.5
      CALL PLOT(XORIG,YORIG,23)
C
C     SET LENGTHS OF PLOTS, ANO VARIOUS AXIS PARAMETERS.
C
      IF (CALFLG) THEN
         AXLENX = 10
         TINCX = .2
         LTICX = -10
         XLENI = AXLENX + 2
         IF (SCAFAC.EO. .S) XLENI = XLENI+6
         AXLENY = 4
         TINCY = .4
```

```
            LTICY = 5
            YLENI = AXLENY+1.5
            IF (SCAFAC.EQ. .5) YLENI = YLENI+2.5
         ELSE
            AXLENX = 5
            TINCX = .1
            LTICX = -10
            XLENI = 0
            AXLENY = 2
            TINCY = .2
            LTICY = 5
            YLENI = AXLENY+.95
         ENOIF
         NUNIT = 11
C
C        PROCESS OATA FOR THE NEXT PLOT
C
50       CONTINUE
         NUNIT = NUNIT+1
         NUNIT2 = NUNIT+40
C        FIRST PARM LINE IS X-AXIS LABEL
         READ(UNIT=NUNIT2,FMT='(A80)',ENO=999) INLINE
         XAXLB = INLINE
         IF (XAXLB(1:7) .EQ. '*SKIPIT') THEN
            CLOSE(UNIT=NUNIT2)
            GOTO 180
         ENOIF
C        2NO PARM IS Y-AXIS LABEL
         READ(UNIT=NUNIT2,FMT='(A80)',ENO=999) INLINE
         YAXLB = INLINE
C        3RO, 4TH, AND 5TH PARMS ARE GRAPH TITLE LINES
         READ(UNIT=NUNIT2,FMT='(A80)',ENO=999) INLINE
         TITL1 = INLINE
         READ(UNIT=NUNIT2,FMT='(A80)',ENO=999) INLINE
         TITL2 = INLINE
         READ(UNIT=NUNIT2,FMT='(A80)',ENO=999) INLINE
         TITL3 = INLINE
C        6TH PARM IS A LINE CONTAINING XVAL1, XINC, FROMPT,
C        FORPTS, ANO FOUR FLAGS OETERMINING WHETHER LINES
C        CONNECT POINTS, WHETHER Y-AXIS SHOULO BE SCALEO
C        FOR A PHASE PLOT, WHETHER THE X-AXIS SHOULO BE LINEAR
C        OR LOGARITHMIC, WHETHER THE Y-AXIS SHOULD BE LINEAR/LOG
C        ANO WHETHER A PHASE PLOT SHOULO BE IN DEGREES OR RAOIANS.
         READ(UNIT=NUNIT2,FMT='(A80)',ENO=999) INLINE
         READ(UNIT=INLINE(1:30),FMT='(2F15.0)')XVAL1,XINC
         READ(UNIT=INLINE(31:42),FMT='(BN,2I6)')IFRMPT, IFRPTS
         LPFLG = INLINE(43:43)
         YPIFLG = INLINE(44:44)
         XLNLOG = INLINE(45:45)
```

```
      YLNLOG = INLINE(46:46)
      CLOSE(UNIT=NUNIT2)
      AMINY = 9.9E40
      AMAXY = -AMINY
      IYCNT = 0
      IWK110 = 0
      XMIN = XVAL1+(IFRMPT-1)*XINC
C
C     READ THE WHOLE FILE, OR THE NUMBER OF POINTS IN IFRPTS,
C     BEGINNING WITH POINT NUMBER 'IFRMPT'.
C
      DO 100 I = 1,999999
         IF (IYCNT.GE.IFRPTS) GOTO 110
         IF (IYCNT.GE.MAXDIM-2) GOTO 110
         CALL DISKIR(NUNIT,A,IERR,IWRKLN,DSKWRK,IWK110,IWK210)
         IF (IERR.NE.0) GOTO 110
         IF (I.LT.IFRMPT) GOTO 100
         IYCNT = IYCNT + 1
         X(IYCNT) = XVAL1+(I-1)*XINC
C        IF THIS IS A PHASE PLOT THAT IS TO BE IN DEGREES, PLOT
C        FROM -200 TO +200 DEGREES AND ADJUST THE PHASE SO AS
C        TO TAKE ADVANTAGE OF THE EXTRA 40 DEGREES TO MINIMIZE
C        LARGE CROSSINGS.
         IF (YPIFLG.EQ.'D') THEN
            Y(IYCNT) = A*180/PI
            IF (Y(IYCNT).GT.160.01) THEN
               IF (Y(IYCNT-1).LT.-90) Y(IYCNT) = Y(IYCNT)-360
            ELSE IF (Y(IYCNT).LT.-160.01) THEN
               IF (Y(IYCNT-1).GT.90) Y(IYCNT) = Y(IYCNT)+360
            ENDIF
         ELSE
            Y(IYCNT) = A
         ENDIF
         IF (A.LT.AMINY) AMINY = A
         IF (A.GT.AMAXY) AMAXY = A
100   CONTINUE
110   CONTINUE
      CLOSE(UNIT=NUNIT)
      IF (IYCNT.EQ.0) GOTO 999
      IF (IYCNT.EQ.1) GOTO 50
C
C     SET LAST TWO VALUES OF ARRAYS SO LINE KNOWS HOW TO SCALE PLOT.
C
C     HANDLE PHASE PLOTS SPECIALLY
      IF (YPIFLG.EQ.'R') THEN
         IF (AMAXY .LE. PI) AMAXY = PI
         IF (AMINY .GE. -PI) AMINY = -PI
      ELSE IF (YPIFLG.EQ.'D') THEN
         IF (AMAXY .LE. 200) AMAXY = 200
```

```
            IF (AMINY .GE. -200) AMINY = -200
         ENDIF
         IF (AMINY.NE.AMAXY) THEN
            Y(IYCNT+1) = AMINY
            Y(IYCNT+2) = (AMAXY-AMINY)/AXLENY
         ELSE
C           HANDLE CASE WHERE INDEPENDENT VARIABLE IS A CONSTANT
            IF (AMINY.EQ.0) THEN
               Y(IYCNT+1) = -.01
               Y(IYCNT+2) = .02/AXLENY
            ELSE
               Y(IYCNT+1) = AMINY-AMINY/10
               Y(IYCNT+2) = (AMAXY+AMAXY/10-Y(IYCNT+1))/AXLENY
            ENDIF
         ENDIF
         X(IYCNT+1) = XMIN
         X(IYCNT+2) = (IYCNT-1)*XINC/AXLENX
C        ALL OF THE ABOVE MUST BE REDONE IF WE ARE MAKING LOG PLOTS.
C        FIRST CHECK THAT THE X-AXIS HAS NO NON-POSITIVE VALUES.
C        IF WE PASS, LET THE CALCOMP ROUTINE SCALE THE X-AXIS.
         IF (XLNLOG.EQ.'L') THEN
            IF (XMIN.LE.0) THEN
               XLNLOG = 'N'
            ELSE
               CALL SCALOG(X,AXLENX,IYCNT,1)
            ENDIF
         ENDIF
C        THE Y-AXIS IS HANDLED SIMILARLY, BUT WE FUDGE THE DATA SLIGHTLY
C        TO MAKE IT NICER BY TRUNCATING VERY SMALL VALUES TO 7 ORDERS OF
C        MAGNITUDE LESS THAN THE LARGEST VALUE.
         IF (YLNLOG.EQ.'L') THEN
            IF (AMINY.LT.0) THEN
               YLNLOG = 'N'
            ELSE
               AMAXLY = INT(ALOG10(AMAXY)+1)
               AMINLY = INT(ALOG10(AMINY)-1)
               IF (AMAXLY-AMINLY .GT. 7) THEN
                  AMINY = 10**(AMAXLY-6.99999)
                  DO 130 I=1,IYCNT
                     IF (Y(I).LT.AMINY) Y(I) = AMINY
130               CONTINUE
               ENDIF
               CALL SCALOG(Y,AXLENY,IYCNT,1)
            ENDIF
         ENDIF
C
C        LET THE CALCOMP ROUTINE DO THE LINEAR SCALING.  THIS UNDOES
C        THE MINIMUM TO MAXIMUM SCALING WE DID EARLIER.
C
```

```
      IF (XLNLOG.EQ.'N') CALL SCALE(X,AXLENX,IYCNT,1)
      IF (YLNLOG.EQ.'N' .AND. YPIFLG.NE.'D')
     1   CALL SCALE(Y,AXLENY,IYCNT,1)
C
C     PLOT THE AXES.
C
      IF (XLNLOG.EQ.'N') THEN
C        HANDLE A LINEAR X-AXIS.
         CALL DAXIS(D.,D.,AXLENX,Q.,TINCX,LTICX)
         CALL SAXIS(X(IYCNT+1),X(IYCNT+2)*TINCX*IABS(LTICX),2,1,LTICX,
     %    XAXLB,-LEN(XAXLB))
         CALL DAXIS(D.,AXLENY,AXLENX,0.,TINCX,-LTICX)
      ELSE
C        HANDLE A LOG X-AXIS.
         CALL DLOGAX(D.,D.,AXLENX,X(IYCNT+2),D.,1 )
         CALL SLOGAX(X(IYCNT+1),1,-1,XAXLB,-LEN(XAXLB))
         CALL DLOGAX(D.,AXLENY,AXLENX,X(IYCNT+2),0.,-1 )
      ENDIF
      IF (YLNLOG.EQ.'N') THEN
C        HANDLE A LINEAR Y-AXIS.
         CALL DAXIS(D.,0.,AXLENY,9D.,TINCY,LTICY)
         CALL SAXIS(Y(IYCNT+1),Y(IYCNT+2)*TINCY*IABS(LTICY),3,2,LTICY,
     %    YAXLB,-LEN(YAXLB))
         CALL DAXIS(AXLENX,D.,AXLENY,9D.,TINCY,-LTICY)
      ELSE
C        HANDLE A LOG Y-AXIS.
         CALL DLOGAX(D.,D.,AXLENY,Y(IYCNT+2),9D.,1 )
         CALL SLOGAX(Y(IYCNT+1),2,1,YAXLB,-LEN(YAXLB))
         CALL DLOGAX(AXLENX,D.,AXLENY,Y(IYCNT+2),9D.,-1 )
      ENDIF
C
C     PLOT THE DATA.
C
      LINTYP = D
      IF (LPFLG.EQ.'P') LINTYP = -1
      IF (LPFLG.EQ.'B') LINTYP = 1
      IF (XLNLOG.EQ.'N' .AND. YLNLOG.EQ.'N') THEN
         CALL LINE(X,Y,IYCNT,1,LINTYP,4)
      ELSE IF (XLNLOG.EQ.'L' .AND. YLNLOG.EQ.'L') THEN
         CALL LGLINE(X,Y,IYCNT,1,LINTYP,4,D)
      ELSE IF (XLNLOG.EQ.'N' .AND. YLNLOG.EQ.'L') THEN
         CALL LGLINE(X,Y,IYCNT,1,LINTYP,4,1)
      ELSE IF (XLNLOG.EQ.'L' .AND. YLNLOG.EQ.'N') THEN
         CALL LGLINE(X,Y,IYCNT,1,LINTYP,4,-1)
      ENDIF
C
C     LABEL THE GRAPH.
C
      CALL SYMBOL(D.,AXLENY+.07,.12,TITL3,D.,8D)
```

```
      CALL SYMBOL(0.,AXLENY+.23,.12,TITL2,0.,80)
      CALL SYMBOL(0.,AXLENY+.39,.12,TITL1,0.,80)
180   CONTINUE
C
C     REORIGIN THE PLOTTER FOR THE NEXT GRAPH.  IF THIS IS
C     THE FIRST PLOT ON THE PAGE, SIMPLY MOVE UP THE PAGE.
C     OTHERWISE, IF ON HAROCOPY MOVE TO A NEW PAPER POSITION.
C     IF INTERACTIVE, PAUSE ANO LET THE USER SEE WHAT WE'VE OONE.
C
      IF (TOPFLG) THEN
         IF (CALFLG) THEN
            CALL PLOT(XLENI,-YLENI,23)
         ELSE
            CALL PLOT(-XORIG,-YLENI-YORIG,-3)
            CALL PLOT(XORIG,YORIG,23)
         ENOIF
      ELSE
         CALL PLOT(0.,YLENI,23)
      ENOIF
      TOPFLG = .NOT. TOPFLG
      GOTO 50
C
C     MAKE THE FINAL CALL TO PLOT TO CLOSE THE OUTPUT FILE.
C
999   CONTINUE
      CALL PLOT(0.,0.,999)
1000  FORMAT(A4)
      STOP
      ENO
```

PROGRAM: SPPRINT EXEC

```
/*****************************************************************
*                                                                *
*       Vibration Analysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MOOULE NAME:
*
*       SPPRINT
*
* FUNCTION:
*
*       To drive the SPPRINT Fortran program to produce
*       printed output.  This routine is called by
*       the VAS command.
*
* EXTERNAL ROUTINES REOUIREO:
*
*       SPMAGPHA - To separata a complex format spectrum
*                  into magnituda and phase components.
*       SPPRINT  - Fortran program to format and print data.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/
 address command
 signal on novalue
 arg oparms
 parsa upper var oparms parms
 diskmode = 'A'
 /* Set defaults for global options */
 gdegrad = 'OEGREES'     /* Print phasa in dagrees */
 gmagphase = 'MP'        /* Print both mag and phase of cross spectrum */
 gfrompt = 1             /* Print beginning with first point */
 gforpts = gggggg        /* Print all points */
 gpowamp = 'FOUR'        /* Assume we want Fourier amplitude spectrum */

 parse var parms fdesc '(' opts ')' parms2
 if words(fdesc) = 0 then do /* We ara beginning with global options */
   do while words(opts) > 0
     parse var opts opt opts
```

```
    if abbrev('LOGX',opt,4) then glinelogx = 'LOG'
    else if abbrev('LINEX',opt,5) then glinelogx = 'LINE'
    else if abbrev('LOGY',opt,4) then glinelogy = 'LOG'
    else if abbrev('LINEY',opt,5) then glinelogy = 'LINE'
    else if abbrev('RADIANS',opt,3) then gdegrad = 'RADIANS'
    else if abbrev('DEGREES',opt,3) then gdegrad = 'DEGREES'
    else if abbrev('MAGNITUDE',opt,3) then gmagphase = 'MAGNITUDE'
    else if abbrev('PHASE',opt,3) then gmagphase = 'PHASE'
    else if abbrev('MP',opt,2) then gmagphase = 'MP'
    else if abbrev('PM',opt,2) then gmagphase = 'MP'
    else if abbrev('POWER',opt,3) then gpowamp = 'POWER'
    else if abbrev('PSD',opt,3) then gpowamp = 'POWER'
    else if abbrev('FOURIER',opt,4) then gpowamp = 'FOUR'
    else if abbrev('FROM',opt,2) then do
      parse var opts gfrompt opts  /* FROM option followed by number */
      if datatype(gfrompt) <> 'NUM' then do
        say 'Invalid FROM option "'gfrompt'"'
        return 24
      end
    end
    else if abbrev('FOR',opt,2) then do
      parse var opts gforpts opts  /* FOR option followed by number */
      if datatype(gforpts) <> 'NUM' then do
        say 'Invalid FOR option "'gforpts'"'
        return 24
      end
    end
    else do   /* That was our last hope for a valid option */
      say 'Invalid option "'opt'"'
      return 24
    end
  end
  parse var parms2 fdesc '(' opts ')' parms2
end

fidcnt = 0
do while words(fdesc) > 0
  fidcnt = fidcnt+1
  parse var fdesc fn type chanx chany fdesc2
  if words(chany) = 0 then do  /* Gotta have complete file spec */
    say 'Incomplete fileid "'fn fttype chanx'" specified'
    return 24
  end
  if abbrev('DATA',type) then dt = 'D'
  else if abbrev('SPECTRUM',type) then dt = 'S'
  else if abbrev('CORRELATION',type,3) then dt = 'C'
  else if abbrev('TRANSFER',type) then dt = 'T'
  else if abbrev('COHERENCE',type,3) then dt = 'K'
  else if type = '.' then dt = '.'
```

```
    else do
      say 'Invalid datatype peremeter "'type'"'
      return 24
    end
    ft = dt||chanx||cheny
    fid.fidcnt = fn ft '*'
    'CMDCALL ESTATE' fid.fidcnt
    if rc <> 0 then return rc
    degred.fidcnt = gdegrad
    magphase.fidcnt = gmegphese
    frompt.fidcnt = gfrompt
    forpts.fidcnt = gforpts
    powemp.fidcnt = gpowamp
    if words(fdesc2)=0 & words(opts)>0 then do while words(opts)>0
      perse ver opts opt opts
      if abbrev('RADIANS',opt,3) then degred.fidcnt = 'RAOIANS'
      else if ebbrev('OEGREES',opt,3) then degred.fidcnt = 'OEGREES'
      else if ebbrev('MAGNITUDE',opt,3) then magphasa.fidcnt = 'MAGNITUDE'
      else if abbrev('PHASE',opt,3) then megphese.fidcnt = 'PHASE'
      else if abbrev('MP',opt,2) then magphase.fidcnt = 'MP'
      else if ebbrev('PM',opt,2) then magphase.fidcnt = 'MP'
      else if ebbrev('POWER',opt,3) then powamp.fidcnt = 'POWER'
      else if abbrev('PSO',opt,3) then powamp.fidcnt = 'POWER'
      else if ebbrev('FOURIER',opt,4) then powamp.fidcnt = 'FOUR'
      else if ebbrev('FRDM',opt,2) then do
        parse ver opts frompt.fidcnt opts  /* FROM option followed by number */
        if datatype(frompt.fidcnt) <> 'NUM' then do
          say 'Invalid FROM option "'frompt.fidcnt'"'
          return 24
        end
      end
      else if abbrev('FOR',opt,2) then do
        parse var opts forpts.fidcnt opts  /* FOR option followed by number */
        if datatype(forpts.fidcnt) <> 'NUM' then do
          say 'Invelid FDR option "'forpts.fidcnt'"'
          return 24
        end
      end
      else do
        say 'Invelid option "'opt'"'
        return 24
      end
    end
    if words(fdesc2) = 0 then perse ver perms2 fdesc '(' opts ')' parms2
    else fdesc = fdesc2
  end

  /* Plist has been parsed, so now we can print */
  maxfidcnt = fidcnt
```

```
if maxfidcnt = 0 then do
  say 'No files specified'
  return 24
end
workfids = '';  utcnt = 0;  inunit = 12
'MAKEBUF'
queue ' '     /* Queue a blank line as first parm */
do fidcnt = 1 to maxfidcnt
  fid = fid.fidcnt
  fn = word(fid,1); ft = word(fid,2); fm = word(fid,3)
  'MAKEBUF'       /* Create a stack just for the EXECIO */
  q = queued()
  'EXECIO 10 DISKR' fn 'IOO' fm '(FINIS'
  q = queued()-q;
  sps = 1; infoline = '* UNKNOWN UNKNOWN UNKNOWN'
  if q >= 1 then do
    pull sps ddate dtime .
    sps = sps/1
    q = q-1
  end
  if q >= 8 then do i=1 to 8
    pull lpf.i loccode.i
    loccode.i = strip(loccode.i)
    if datatype(lpf.i,'N') then lpf.i = lpf.i/1
    q = q-1
  end
  if q >= 1 then pull infoline
  'OROPBUF'         /* Orop EXECIO stack */
  method = word(infoline,2)
  windname = word(infoline,3)
  seqlen = word(infoline,4)
  if seqlen <> 'UNKNOWN' then seqlen = seqlen+0
  fttype=substr(ft,1,1); chanx=substr(ft,2,1); cheny=substr(ft,3,1)
  if degred.fidcnt = 'DEGREES' then degred = 'D'
                               else degred = 'R'
  frompt = frompt.fidcnt
  forpts = forpts.fidcnt
  select
    when fttype='T' | fttype='S' then do   /* Complex date */
      utcnt = utcnt+1; workft1 = 'CMSUT'utcnt
      workfid1 = 'SPEC' workft1 diskmode
      utcnt = utcnt+1; workft2 = 'CMSUT'utcnt
      workfid2 = 'SPEC' workft2 diskmode
      workfids = workfids workfid1 workfid2
      /* Turn into magnitude and phase, converting a PSD into a */
      /* simple FAS if required */
      if powamp.fidcnt = 'FOUR' then spmpopt = 'SQRT'; else spmpopt = ''
      'EXEC SPMAGPHA' fn ft fm workfid1 workfid2 spmpopt
      /* Handle magnitude print */
```

```
if magphase.fidcnt <> 'PHASE' then do /* Dption may prevent print */
  'FILEDEF' inunit 'DISK' workfid1
  'FILEDEF' inunit+4D 'TERM (LRECL 8D RECFM F'
  inunit = inunit+1
  queue 'FREDUENCY'; queue 'MAGNITUDE'
  qline = 'TEST:' left(fn,8)||'  '||ddete dtime
  if fttype <> 'S' then qline = qline ' TRANSFER FUNCTIDN'
  else if spmpopt = 'SDRT' then qline = qline ' FAS ESTIMATE'
                           else qline = qline ' PSD ESTIMATE'
  if method <> 'WELCH' then qline = qline '(T)'
  queue qline
  if chanx = chany then do
    qline = 'CHANNEL:' chanx '('loccode.chanx')'
    qline = qline ' LPF FREQ:' lpf.chanx
  end
  else qline = 'CHANNELS:' chanx':'chany
  queue qline
  qline = 'WINOOW:' windname ' PTS/SEGMENT:' seqlen
  qline = qline ' SAMPLE RATE:' sps
  queue qline
  xvel = D; xinc = 1
  if seqlen <> 'UNKNDWN' then xinc = sps/2/seqlen
  pline = right(xval,15)||right(xinc,15)||right(frompt,6)
  pline = pline||right(forpts,6)||' '||' '
  pline = pline||' '||' '
  queue pline
end
/* Hendle phase print (if needed) */
if fttype='S' & chenx<>chany & magphase.fidcnt <> 'MAGNITUDE' ,
 then do /* Spectrum print mey need phase */
  'FILEOEF' inunit 'DISK' workfid2
  'FILEDEF' inunit+4O 'TERM (LRECL 8D RECFM F'
  inunit = inunit+1
  if degrad = 'R' then temp='RAD'; else temp='DEG'
  queue 'FREQUENCY'; queue 'PHASE('temp')'
  qline = 'TEST:' left(fn,8)||'  '||ddate dtime
  if spmpopt = 'SDRT' then qline = qline ' FAS ESTIMATE'
                      else qline = qline ' PSD ESTIMATE'
  if method <> 'WELCH' then qline = qline '(T)'
  queue qline
  if chenx = chany then do
    qline = 'CHANNEL:' chanx '('loccode.chanx')'
    qline = qline ' LPF FRED:' lpf.chenx
  end
  else qline = 'CHANNELS:' chanx':'cheny
  queue qline
  qline = 'WINDDW:' windname ' PTS/SEGMENT:' seqlen
  qline = qline ' SAMPLE RATE:' sps
  queue qline
```

```
        xvel = 0; xinc = 1
        if seqlen <> 'UNKNOWN' then xinc = sps/2/seqlen
        pline = right(xvel,15)||right(xinc,15)||right(frompt,6)
        pline = pline||right(forpts,6)||' '||degrad
        pline = pline||' '||' '
        queue pline
      end
    end
    otherwise do
      'FILEDEF' inunit 'OISK' fn ft fm
      'FILEOEF' inunit+4D 'TERM (LRECL 8D RECFM F'
      inunit = inunit+1
      if fttype='C' | fttype='D' then queue 'TIME(SECS)'
                                 else queue 'FREQUENCY'
      queue 'AMPLITUDE'
      qline = 'TEST:' left(fn,8)||'  '||ddate dtime
      if fttype='K' then qlina = qline ' COHERENCE'
      else if fttype='C' then qline = qline ' CDRRELATION'
      else if fttype='D' then qline = qline ' DATA'
      if method <> 'WELCH' & fttype='K' then qline = qline '(T)'
      queue qline
      if fttype='K' then do  /* Coherence */
        qline = 'CHANNELS:' chenx':'cheny
        queue qline
        queue ' '
        xvel = 0; xinc = 1
        if seqlen <> 'UNKNDWN' then xinc = sps/2/seqlen
      end
      else do          /* Correletion or Data */
        if chanx = chany then do
          qline = 'CHANNEL:' chanx '('loccode.chanx')'
          qline = qline ' LPF FREQ:' lpf.chenx
        end
        else qline = 'CHANNELS:' chanx':'chany
        queue qline
        queue 'SAMPLE RATE:' sps
        xval = 0; xinc = 1
        if sps <> 'UNKNDWN' then xinc = 1/sps
      end
      pline = right(xval,15)||right(xinc,15)||right(frompt,6)
      pline = pline||right(forpts,6)||' '||' '
      pline = pline||' '||' '
      queue pline
    end
  end  /* SELECT */
end  /* Dnce for eech file */
'FILEDEF' inunit+4D 'DUMMY'
'FILEDEF 4 TERM (LRECL 8D RECFM F'
'FILEDEF 95 PRINT (LRECL 133 RECFM FA'
```

PROGRAM: SPPRINT EXEC

```
'ESTATE SPPRINT MODULE *'
if rc <> 0 then 'LOAD SPPRINT SPOISKIO(NOMAP CLEAR START'
else 'SPPRINT'
'DROPBUF'
do while words(workfids)>0
  parse var workfids fn ft fm workfids
  'ERASE' fn ft fm
end
return 0
```

PROGRAM: SPPRINT FORTRAN

```
*.
*****************************************************************
*                                                               *
*       Vibration Analysis System (VAS)                         *
*                                                               *
*       (c) Copyright 1986 by Brick A. Verser                   *
*                                                               *
*****************************************************************
*
* MODULE NAME:
*
*       SPPRINT
*
* FUNCTION:
*
*       TO FORMAT AND PRINT REAL*4 DATA.
*
* INPUT UNITS:
*
*       12..29 INPUT DATA TO BE PRINTED IN REAL A4 FORMAT.
*       S2..69 PARAMETERS AND OPTIONS FOR EACH DATA FILE PRINTED.
*
* OUTPUT UNITS:
*
*       9S     THE PRINTED PAGES.
*
* INPUT PARAMETERS (UNIT 4):
*
*       IGNORED, BUT MUST BE PRESENT
*
* INPUT PARAMETERS (UNITS S2..69):
*
*       RECORD 1:
*          1..13  LABEL FOR INDEPENDENT DATA COLUMNS.
*       RECORD 2:
*          1..13  LABEL FOR DEPENDENT DATA COLUMNS.
*       RECORDS 3, 4 AND 5:
*          1..72  PAGE TITLE LINES.
*       RECORD 6:
*          1..1S  F15.0 FORMAT INITIAL INDEPENDENT VARIABLE VALUE.
*          16..3O F1S.0 FMT INDEPENDENT VARIABLE INCREMENT.
*          31..36 I6 FMT NUMBER OF THE FIRST POINT TO PRINT.
*          37..42 I6 FMT NUMBER OF POINTS TO PRINT.
*          44..44 '0' TO PLOT PHASE IN DEGREES.
*
* OPERATION:
*
*       1.  FOR EACH INPUT FILE, THE PARAMETERS ARE READ.
*           IF END-OF-FILE ENCOUNTERED, EXIT.
```

```
*       2.  THE DATA IS READ AND ACCUMULATED UNTIL AN ENTIRE
*           PAGE WDRTH HAS BEEN READ DR UNTIL END-OF-FILE.
*       3.  THE ACCUMULATED DATA PAGE IS PRINTED.
*           IF END-OF-FILE WAS NOT ENCDUNTERED, GDTD STEP 2.
*       4.  THE INPUT DATA FILE IS CLDSED.  GDTD STEP 1.
*
* EXTERNAL RDUTINES REQUIRED:
*
*       DISKIR - TD READ REAL*4 FDRMAT VALUES FRDM DISK.
*
* REVISIDN HISTDRY:
*
*       1.D   ORIGINAL CDDE
*
*       REVISIDN        DATE                    PRDGRAMMER
*       --------        ----                    ----------
*        1.D            JUNE 26, 1986           BRICK VERSER
*
*.
      PARAMETER (IWRKLN=16)
      PARAMETER (LENPAG=5D)
      DIMENSIDN X(LENPAG*3),Y(LENPAG*3)
      DIMENSIDN DSKWRK(IWRKLN)
      CHARACTER*1 INCHAR,YPIFLG
      CHARACTER*72 TITL1,TITL2,TITL3
      CHARACTER*8D INLINE
      CHARACTER*13 XAXLB,YAXLB
      LDGICAL CMPLFL,EDFFLG
      PI = 4*ATAN(1.)
C
C     FIRST PARM LINE IS THRDWN AWAY
C
      READ(4,'(A8D)') INLINE
      NUNIT = 11
C
C     PRDCESS DATA FDR THE NEXT FILE
C
5D    CDNTINUE
      NUNIT = NUNIT+1
      NUNIT2 = NUNIT+4D
C     FIRST PARM LINE IS X-AXIS LABEL (THE INDEPENDENT VARIABLE AXIS)
      REAO(UNIT=NUNIT2,FMT='(A8D)',END=999) INLINE
      XAXLB = INLINE
C     2ND PARM IS Y-AXIS LABEL (THE DEPENDENT VARIABLE AXIS)
      READ(UNIT=NUNIT2,FMT='(A8D)',ENO=999) INLINE
      YAXLB = INLINE
C     3RD, 4TH AND 5TH PARMS ARE TITLE LINES
      READ(UNIT=NUNIT2,FMT='(A8D)',END=999) INLINE
      TITL1 = INLINE
```

```
      READ(UNIT=NUNIT2,FMT='(A80)',END=999) INLINE
      TITL2 = INLINE
      READ(UNIT=NUNIT2,FMT='(A80)',END=999) INLINE
      TITL3 = INLINE
C     6TH PARM IS A LINE CONTAINING XVAL1, XINC, FROMPT, FORPTS,
C     AND FLAGS, MOST UNUSED BUT ONE DETERMINING WHETHER
C     A PHASE PRINT SHOULD BE SCALED IN DEGREES OR RADIANS.
      READ(UNIT=NUNIT2,FMT='(A80)',END=999) INLINE
      READ(UNIT=INLINE(1:30),FMT='(2F15.0)')XVAL1,XINC
      READ(UNIT=INLINE(31:42),FMT='(BN,2I6)')IFRMPT, IFRPTS
      YPIFLG = INLINE(44:44)
      CLOSE(UNIT=NUNIT2)
      EOFFLG = .FALSE.
      IPAGE = 0
      IWK110 = 0
C
C     SKIP TO FIRST POINT WE'RE SUPPOSED TO PRINT.
C
      ICNTTT = 0
      ICNTPR = 0
      DO 80 I = 1,IFRMPT-1
         CALL DISKIR(NUNIT,A,IERR,IWRKLN,DSKWRK,IWK110,IWK210)
         IF (IERR.NE.0) GOTO 999
         ICNTTT = ICNTTT+1
80    CONTINUE
C
C     READ A PAGE WORTH OF DATA INTO ARRAY 'X'.
C
101   CONTINUE
      IYCNT = 0
      IPAGE = IPAGE+1
      DO 105 I = 1,LENPAG*3
         IF (ICNTPR.GE.IFRPTS) GOTO 110
         CALL DISKIR(NUNIT,A,IERR,IWRKLN,DSKWRK,IWK110,IWK210)
         IF (IERR.NE.0) GOTO 110
         ICNTTT = ICNTTT+1
         IYCNT = IYCNT + 1
         ICNTPR = ICNTPR + 1
         X(I) = XVAL1+(ICNTTT-1)*XINC
C        CONVERT PHASE TO DEGREES IF REQUESTED
         IF (YPIFLG.EQ.'D') THEN
            Y(I) = A*180/PI
         ELSE
            Y(I) = A
         ENDIF
105   CONTINUE
      GOTO 111
C     HERE ON END-OF-FILE.  SET FLAG TO REMEMBER WE FINISHED A FILE.
110   CONTINUE
```

```
      CLOSE(UNIT=NUNIT)
      EOFFLG = .TRUE.
111   CONTINUE
      IF (IYCNT.EQ.0) GOTO 999
C
C     PRINT A PAGE OF DATA WITH 3 COLUMNS AND 50 ROWS OF DATA.
C
      WRITE(95,1010)TITL1,IPAGE
1010  FORMAT('1',A72,' PAGE ',I2)
      WRITE(95,1011)TITL2
1011  FORMAT(1X,A72)
      WRITE(95,1011)TITL3
      WRITE(95,1050)
1050  FORMAT(1X)
      WRITE(95,1015)XAXLB,YAXLB,XAXLB,YAXLB,XAXLB,YAXLB
1015  FORMAT('0',6A13)
      WRITE(95,1050)
      DO 140 I=1,LENPAG
         IF (IYCNT.GE.2*LENPAG+I) THEN
            WRITE(95,1020)X(I),Y(I),X(I+LENPAG),Y(I+LENPAG),
     X                    X(I+2*LENPAG),Y(I+2*LENPAG)
         ELSE IF (IYCNT.GE.LENPAG+I) THEN
            WRITE(95,1020)X(I),Y(I),X(I+LENPAG),Y(I+LENPAG)
         ELSE IF (IYCNT.GE.I) THEN
            WRITE(95,1020)X(I),Y(I)
C020        FORMAT(1X,6(E11.4,2X))
1020        FORMAT(1X,3(F11.5,2X,E11.4,2X))
         ENDIF
140   CONTINUE
      IF (EOFFLG) GOTO 180
      GOTO 101
C
C     HERE AFTER LAST PAGE OF DATA FILE HAS BEEN PRINTED
C
180   CONTINUE
      GOTO 50
C
999   CONTINUE
      STOP
      END
```

PROGRAM: SPBINOP EXEC

```
/*****************************************************************
*                                                                *
*       Vibration Analysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MODULE NAME:
*
*       SPBINOP
*
* FUNCTION:
*
*       To perform e binary oparation on two input files
*       to produce e third file.
*       This routine is callad from the VAS command.
*
* EXTERNAL ROUTINES REQUIRED:
*
*       SPBINOP  - Fortren program to do the actual work.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/
 address command
 arg inargs
 parse upper ver inargs fmt infn1 inft1 infm1 infn2 inft2 infm2 ,
                        outfn outft outfm func
 if length(func) = 0 than signal help
 if abbrev(fmt,'REAL') then fmt = 0
 else if abbrev(fmt,'COMPLEX') then fmt = 1
 else do
   say 'SPBINOP: Invalid data format "'fmt'"'
   return 24
 end
 infid1 = infn1 inft1 infm1
 infid2 = infn2 inft2 infm2
 outfid = outfn outft outfm
 select
   when abbrev('AOD',func) then funcnum = 0
   when abbrev('SUBTRACT',func) then funcnum = 1
   when abbrev('MULTIPLY',func) then funcnum = 2
```

```
   when abbrev('DIVIDE',func) then funcnum = 3
   otherwise do
     say 'SPBINDP: Invalid function "'func'"'
     return 24
   end
 end
 optline = fmt||right(funcnum,2)
 say optline
 'FILEDEF 1D DISK' infid1
 'FILEDEF 11 DISK' infid2
 'FILEDEF 3D DISK' outfid '(LRECL 8D RECFM V'
 'FILEDEF 4 TERM (LRECL 8D RECFM F'
 'FILEDEF 7 TERM'
 'FILEDEF 8 TERM'
 'FILEDEF 9 TERM'
 push optline
 'SPBINDP'
 return rc

HELP:
 say 'SPBINDP:  Invalid parameter list'
 return 24
```

```
*.
*****************************************************************
*                                                               *
*       Vibration Analysis System (VAS)                         *
*                                                               *
*       (c) Copyright 1986 by Brick A. Verser                   *
*                                                               *
*****************************************************************
*
* MOOULE NAME:
*
*       SPWELCH
*
* FUNCTION:
*
*     PERFORM A BINARY OPERATION ON TWO OATA FILES.
*
*
* INPUT UNITS:
*
*       04      PARAMETERS ANO OPTIONS.
*       10      CHANNEL 1 INPUT OATA IN REAL A4 FORMAT.
*       11      CHANNEL 2 INPUT OATA IN REAL A4 FORMAT.
*
* OUTPUT UNITS:
*
*       07      ERROR MESSAGES.
*       08      INFORMATIONAL ANO WARNING MESSAGES.
*       30      RESULT IN REAL A4 FORMAT.
*
* INPUT PARAMETERS:
*
*       RECORO 1:
*          1..1   INTEGER, SPECIFYING THE FORMAT OF THE INPUT.
*                 0:  INPUT CHANNELS ARE IN REAL A4 FORMAT.
*                 1:  INPUT CHANNELS ARE IN CMPLX 2A4 FORMAT.
*          2..3   INTEGER, SPECIFYING THE OPERATION TO BE
*                 PERFORMEO ON THE OATA.
*                 0:  AOOITION
*                 1:  SUBTRACTION
*                 2:  MULTIPLICATION
*                 3:  OIVISION
*
* EXTERNAL ROUTINES REQUIREO:
*
*       OISKIR - TO REAO REAL*4 FORMAT VALUES FROM OISK.
*       OISKOR - TO WRITE REAL*4 FORMAT VALUES TO OISK.
*
* REVISION HISTORY:
```

PROGRAM: SPBINOP FORTRAN

```
*
*       1.0   ORIGINAL CODE
*
*       REVISION        DATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*.
      PARAMETER (IWRKLN=16)
      DIMENSION WORK10(IWRKLN),WORK11(IWRKLN),WORK30(IWRKLN)
      COMPLEX CX,CY,CZ,COP
      IOP = 0
      IFORM = 0
      READ(4,'(I1,I2)',END=190)IFORM,IOP
190   CONTINUE
      CALL DISKOR(30,0.,1,IWRKLN,WORK30,IWK30)
      IWK110 = 0
      IWK111 = 0
      IF (IFORM .EQ. 0) THEN
200      CONTINUE
            CALL DISKIR(10,X,IERR,IWRKLN,WORK10,IWK110,IWK210)
            IF (IERR.NE.0) GOTO 250
            CALL DISKIR(11,Y,IERR,IWRKLN,WORK11,IWK111,IWK211)
            IF (IERR.NE.0) GOTO 250
            Z = OP(X,Y,IOP)
            CALL DISKOR(30,Z,2,IWRKLN,WORK30,IWK30)
         GOTO 200
      ELSE
220      CONTINUE
            CALL DISKIR(10,XR,IERR,IWRKLN,WORK10,IWK110,IWK210)
            IF (IERR.NE.0) GOTO 250
            CALL DISKIR(10,XI,IERR,IWRKLN,WORK10,IWK110,IWK210)
            IF (IERR.NE.0) GOTO 250
            CX = CMPLX(XR,XI)
            CALL DISKIR(11,XR,IERR,IWRKLN,WORK11,IWK111,IWK211)
            IF (IERR.NE.0) GOTO 250
            CALL DISKIR(11,XI,IERR,IWRKLN,WORK11,IWK111,IWK211)
            IF (IERR.NE.0) GOTO 250
            CY = CMPLX(XR,XI)
            CZ = COP(CX,CY,IOP)
            CALL DISKOR(30,REAL(CZ),2,IWRKLN,WORK30,IWK30)
            CALL DISKOR(30,AIMAG(CZ),2,IWRKLN,WORK30,IWK30)
         GOTO 220
      ENDIF
250   CONTINUE
      CALL DISKOR(30,0.,3,IWRKLN,WORK30,IWK30)
      STOP
      END
      FUNCTION OP(X,Y,IOP)
```

PROGRAM: SPBINOP FORTRAN

```
IF (IOP.EO.0) THEN
   OP = X+Y
ELSE IF (IOP.EO.1) THEN
   OP = X-Y
ELSE IF (IOP.EO.2) THEN
   OP = X*Y
ELSE IF (IOP.EQ.3) THEN
   OP = X/Y
ENOIF
RETURN
END
FUNCTION COP(X,Y,IOP)
COMPLEX COP,X,Y
IF (IOP.EO.0) THEN
   COP = X+Y
ELSE IF (IOP.EO.1) THEN
   COP = X-Y
ELSE IF (IOP.EQ.2) THEN
   COP = X*Y
ELSE IF (IOP.EO.3) THEN
   COP = X/Y
ENOIF
RETURN
END
```

PROGRAM: SPCONV2 EXEC

```
/*****************************************************************
*                                                                *
*       Vibretion Anelysis System (VAS)                          *
*                                                                *
*       (c) Copyright 1986 by Brick A. Verser                    *
*                                                                *
*****************************************************************
*
* MOOULE NAME:
*
*       SPMAGPHA
*
* FUNCTION:
*
*       Convert from rew input date (reversed-byte integers
*       with interleeved channels) to ready-to-process real
*       format dete with one channel per file end with
*       an associated information file.  This routine is
*       called by the VAS commend.
*
* EXTERNAL ROUTINES REQUIREO:
*
*       SPCONV2  - Fortren program to do the actual work.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*/
 eddress command
 erg inargs
 parse upper var inargs infn inft infm outfids
 if length(outfids)=0 then signal help
 infid = infn inft infm
 do i=1 to 9 while length(outfids)>0
   parse var outfids outfn outft outfm outfids
   if length(outfm) = 0 then signal help
   outfid = outfn outft outfm
   unit = 29+i
   if unit>30 then 'FILEDEF' unit 'DISK' outfid '(LRECL 80 RECFM V'
   else 'FILEDEF' unit 'OISK' outfid '(LRECL 80 RECFM F'
 end
 'FILEOEF 10 OISK' infid
 'FILEOEF 7 TERM'
 'FILEOEF 8 TERM'
```

PROGRAM: SPCONV2 EXEC

```
 'FILEOEF 9 TERM'
 'SPCONV2'
 return rc

HELP:
 say 'SPCONV2:  Invalid parameter list'
 return 24
```

```
*.
*****************************************************************
*                                                               *
*       Vibration Analysis System (VAS)                         *
*                                                               *
*       (c) Copyright 1986 by Brick A. Verser                   *
*                                                               *
*****************************************************************
*
* MOOULE NAME:
*
*       SPCONV
*
* FUNCTION:
*
*       TO CONVERT FROM RAW REVERESEO-BYTE INTEGER*2 OATA WITH
*       HEAOER ANO INTERLEAVED CHANNELS TO SIMPLE A4 FORMAT REAL
*       FILES WITH ONE CHANNEL PER FILE.
*
* INPUT UNITS:
*
*       10      RAW INPUT OATA INCLUDING VERSION 2 HEAOER.
*
* OUTPUT UNITS:
*
*       07      ERROR MESSAGES.
*       08      INFORMATIONAL ANO WARNING MESSAGES.
*       30      INFORMATIONAL OUTPUT.  THE SAMPLING RATE ANO
*               THE OATE&TIME THE OATA WAS COLLECTEO ARE WRITTEN
*               TO THE FIRST LINE.  THE NEXT EIGHT LINES CONTAIN
*               THE LOW-PASS FILTER SETTING ANO LOCATION COOE FOR
*               EACH OF THE POSSIBLE INPUT CHANNELS.
*       31..38 REAL FORMAT OUTPUT OATA, ONE CHANNEL PER UNIT.
*
* INPUT PARAMETERS:
*
*       NONE
*
* OPERATION:
*
*       1.  THE INPUT HEAOER IS READ ANO PARSEO TO OETERMINE
*           VALUES SUCH AS THE SAMPLING RATE, NUMBER OF OATA
*           CHANNELS, ETC.
*       2.  THE INFORMATIONAL FILE IS WRITTEN.
*       3.  THE OATA IS REAO ANO CONVERTEO FROM REVERSEO BYTE
*           INTEGER*2 FORMAT TO REAL*4 FORMAT ANO EACH CHANNEL
*           OF OATA IS WRITTEN TO A OIFFERENT UNIT.
*
* EXTERNAL ROUTINES REQUIRED:
```

PROGRAM: SPCONV2 FORTRAN

```
*
*       OISKIH - TO REAO INTEGER*2 FORMAT VALUES FROM OISK.
*       OISKOR - TO WRITE REAL*4 FORMAT VALUES TO DISK.
*
* REVISIDN HISTORY:
*
*       1.O   ORIGINAL CODE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.O            JUNE 26, 1986           8RICK VERSER
*
*.
C
C     THE FOLLOWING PARAMETER STATEMENTS OEFINE THE OFFSETS
C     TO THE FIELOS OF INTEREST IN THE RAW OATA HEAOER.
C     REFER THE TO MICROCDMPUTER DATA ACQUISITIDN PROGRAM
C     FOR A COMPLETE OESCRIPTION OF EACH OF THESE FIELDS.
C
      INTEGER O8V,OOS,OVERS,OCOMPV,OHORSZ,OCSFLG
      INTEGER OOVFLG,OOTYY,OOTMO,OOTOO,OOTHH,OOTMI,OOTSS
      INTEGER CSASC,CSE8C,OCHNCT,OOFFVL
      INTEGER OSPS,OSPSM1,OSPSO1,OSPSM2,OSPSO2
      INTEGER OUPI,OUPIM1,OUPIO1,OUPIM2,OUPIO2
      INTEGER OLOCAT,OLPF
      PARAMETER (O8V=O,OOS=2,OVERS=4,OCOMPV=6,OHORSZ=8)
      PARAMETER (OCSFLG=16,OOVFLG=17,OOTYY=18,OOTMO=2O)
      PARAMETER (OOTOO=22,OOTHH=24,OOTMI=26,OOTSS=28)
      PARAMETER (CSASC=128,CSE8C=64)
      PARAMETER (OCHNCT=128,OOFFVL=132)
      PARAMETER (OSPS=144,OSPSM1=148,OSPSO1=1S2,OSPSM2=156,OSPSO2=1S8)
      PARAMETER (OUPI=16O,OUPIM1=192,OUPIO1=224,OUPIM2=2S6,OUPIO2=272)
      PARAMETER (OLOCAT=288,OLPF=416)
      INTEGER*2 8VCONS,OTCONS
      INTEGER*2 HEAOER(O:2SS),ITEMP
      INTEGER HEAO4(O:127)
      EOUIVALENCE (HEAOER(O),HEAO4(O))
      OIMENSION UPI(8),LPF(8),ASCLPF(8),LOCCOO(8),ASCLOC(8)
      CHARACTER*16 LOCCOO,ASCLOC
      CHARACTER*6 LPF,ASCLPF
      EOUIVALENCE (HEAOER(OLOCAT/2),ASCLOC(1))
      EOUIVALENCE (HEAOER(OLPF/2),ASCLPF(1))
      PARAMETER (I28SIZ=4O)
      INTEGER*2 I2BUFF(I28SIZ)
      PARAMETER (IWRKLN=16)
      OIMENSION OSKWRK(IWRKLN,8),IOSKWK(8)
      CHARACTER*3 STRMON,STROAY,STRYR,STRHR,STRMIN
      OATA 8VCONS/16982/,OSCONS/17491/
      DATA IUNINI/31/
```

PROGRAM: SPCONV2 FORTRAN

```
C
C     READ HEADER
C
      IWK110 = 0
      DO 110 I=0,255
         CALL DISKIH(10,HEADER(1),IERR,I2BSIZ,I2BUFF,IWK110,IWK210)
         IF (IERR.NE.0) THEN
            WRITE(7,*)' INCOMPLETE OR MISSING HEADER ON RAW INPUT'
            GOTO 999
         ENDIF
110   CONTINUE
      IF (HEADER(OBV/2).NE.BVCONS .OR. HEADER(ODS/2).NE.DSCONS) THEN
         WRITE(7,*)' NO HEADER ON RAW INPUT DATA FILE'
         GOTO 999
      ENDIF
C
C     DECODE THE FIELDS OF INTEREST IN THE HEADER
C
      ICHANS = HEADER(OCHNCT/2)
      IOFF = HEADER(OOFFVL/2)
      SPS = FLOAT(HEAD4(OSPS/4))*HEAD4(OSPSM1/4)/HEAD4(OSPSD1/4)*
     1      HEADER(DSPSM2/2)/HEADER(OSPSD2/2)
      DO 130 I=1,ICHANS
         UPI(I) = FLOAT(HEAD4(OUPI/4+I-1))*HEAD4(OUPIM1/4+I-1)/
     1            HEAD4(OUPID1/4+I-1)*HEADER(DUPIM2/2+I-1)/
     2            HEADER(OUPID2/2+I-1)
         CALL CVTSTR(ASCLPF(I),LPF(I))
         CALL CVTSTR(ASCLOC(I),LOCCOD(I))
130   CONTINUE
C
C     WRITE THE INFORMATIONAL FILE.
C
C     FIRST LINE OF INFO FILE IS SAMPLE RATE, DATE, AND TIME.
C     BE SURE MM/DD/YY HH:MM ALL HAVE LEADING ZEROS.
      WRITE(UNIT=STRMON,FMT='(I3)') HEADER(ODTMO/2)+100
      WRITE(UNIT=STRDAY,FMT='(I3)') HEADER(ODTDD/2)+100
      WRITE(UNIT=STRYR,FMT='(I3)') HEADER(ODTYY/2)-1900+100
      WRITE(UNIT=STRHR,FMT='(I3)') HEADER(ODTHH/2)+100
      WRITE(UNIT=STRMIN,FMT='(I3)') HEADER(ODTMI/2)+100
      WRITE(30,1000) SPS, STRMON(2:3), STRDAY(2:3),
     1               STRYR(2:3), STRHR(2:3), STRMIN(2:3)
1000  FORMAT(E15.7,' ',A2,'/',A2,'/',A2,' ',A2,':',A2)
C     LOW PASS FILTER SETTING AND LOCATION CODE FOR EACH CHANNEL
      DO 150 I=1,ICHANS
         WRITE(30,'(A6,10X,A16)')LPF(I),LOCCOD(I)
150   CONTINUE
C     BE SURE EXACTLY 8 RECORDS ARE WRITTEN
      DO 151 I=ICHANS+1,8
         WRITE(30,'(1X)')
```

```
151   CONTINUE
      ITOT = 0
C
C     READ DATA, CONVERT TO REAL FORMAT, AND WRITE TO FILES
C
C     OPEN OUTPUT UNITS
      DO 210 I=1,ICHANS
         CALL DISKOR(I+IUNINI-1,0.,1,IWRKLN,DSKWRK(1,I),IOSKWK(I))
210   CONTINUE
C
C     READ A DATA VALUE, SWAP THE BYTES, AND WRITE IT
C
220   CONTINUE
         CALL DISKIH(10,ITEMP,IERR,I2BSIZ,I2BUFF,IWK110,IWK210)
         IF (IERR.NE.0) GOTO 270
         IUNIT = MOD(ITOT,ICHANS)+IUNINI
         X = (ISWAP2(ITEMP) - IOFF) * UPI(IUNIT-IUNINI+1)
         CALL DISKOR(IUNIT,X,2,IWRKLN,DSKWRK(1,IUNIT-(IUNINI-1)),
     X               IOSKWK(IUNIT-(IUNINI-1)))
         ITOT = ITOT+1
      GOTO 220
270   CONTINUE
C
C     CLOSE OUTPUT FILES
C
      DO 310 I=1,ICHANS
         CALL DISKOR(I+IUNINI-1,0.,3,IWRKLN,DSKWRK(1,I),IOSKWK(I))
310   CONTINUE
999   CONTINUE
      STOP
      END
      FUNCTION ISWAP2(I)
      INTEGER*2 I
C
C     THIS FUNCTION SWAPS THE BYTES OF AN INTEGER*2 VALUE
C
C     ISWAP2 = ISHFT(I,-8)+ISHFT(IAND(JJ,255),8)
      J = I
      IF (J.LT.0) J = J+65536
      ISWAP2 = J/256 + (J-J/256*256)*256
      RETURN
      END
      FUNCTION ISWAP4(LSW,MSW)
      INTEGER*2 LSW,MSW
C
C     THIS FUNCTION SWAPS THE BYTES OF AN INTEGER*4 VALUE
C
      ISWAP4 = ISWAP2(LSW)+ISWAP2(MSW)*65536
      RETURN
```

PROGRAM: SPCONV2 FORTRAN

```
      ENO
      SUBROUTINE CVTSTR(INS,OUTS)
      CHARACTER*(*) INS,OUTS
      CHARACTER*1 ASC2EB
C
C     CONVERT STRING 'INS' FROM ASCII TO EBCDIC ANO PLACE IN 'OUTS'
C
      DO 110 I=1,MIN(LEN(INS),LEN(OUTS))
         OUTS(I:I) = ASC2EB(INS(I:I))
110   CONTINUE
      RETURN
      END
      CHARACTER*1 FUNCTION ASC2EB(INC)
      CHARACTER*1 INC
C
C     CONVERT THE SINGLE CHARACTER 'INC' FROM ASCII TO EBCOIC
C
      OIMENSION TABLE(0:127)
      CHARACTER*1 TABLE
      OATA TABLE/32*' ', ' ', ' ', '"', '#', '$', '%', '&', ''', '(',
     1           ')', '*', '+', ',', '-', '.', '/', '0', '1', '2',
     2           '3', '4', '5', '6', '7', '8', '9', ':', ';', '<',
     3           '=', '>', '?', '@', 'A', 'B', 'C', 'O', 'E', 'F',
     4           'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
     5           'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z',
     6           '[', '\', ']', '\', '_', '`', 'a', 'b', 'c', 'd',
     7           'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
     8           'o', 'p', 'q', r', 's', 't', 'u', 'v', 'w', 'x',
     9           'y', 'z', '', '|', '', '~', ' '/
      ASC2EB = TABLE(ICHAR(INC))
      RETURN
      END
```

PROGRAM: WWINDO FORTRAN

```
*.
*****************************************************************
*                                                               *
*       Vibration Analysis System (VAS)                         *
*                                                               *
*       (c) Copyright 1986 by Brick A. Verser                   *
*                                                               *
*****************************************************************
*
* MODULE NAME:
*
*       WWINDO
*
* FUNCTION:
*
*     TO CREATE ONE OF SIX POSSIBLE WINDOW FUNCTIONS AND
*     PLACE THEM IN AN ARRAY.
*
*     THE FOLLOWING CODE IS BORROWED FROM "RALPH", THE
*     VAX/VMS SIGNAL PROCESSING PROGRAM OF THE KANSAS STATE
*     UNIVERSITY OEPARTMENT OF ELECTRICAL ENGINEERING.
*
* CALLING PARAMETERS:
*
*       CALL WWINDO(ARRAY,LENGTH,WINTYP,POWER,THETA)
*
* INPUT PARAMETERS:
*
*       LENGTH - AN INTEGER SPECIFYING THE NUMBER OF WINOOW
*                COMPONENTS TO BE GENERATED AND PLACED IN 'ARRAY'.
*       WINTYP - AN INTEGER SPECIFYING WHICH WINDOW TO GENERATE.
*                1:  TRIANGLE (ALSO CALLEO BARTLETT)
*                2:  HAMMING (ALSO CALLED RAISED-COSINE)
*                3:  HANNING (ALSO CALLED COSINE)
*                4:  KAISER (WITH INPUT PARAMETER='THETA')
*                S:  BLACKMAN-HARRIS
*                6:  PARZEN
*       THETA  - PARAMETER SPECIFYING THE SHAPE OF THE KAISER WINDOW.
*
* OUTPUT PARAMETERS:
*
*       ARRAY  - THE GENERATED WINDOW FUNCTION OF LENGTH 'LENGTH'.
*       POWER  - THE VARIANCE OF THE GENERATED WINDOW.
*
* REVISION HISTORY:
*
*       0.D   ORIGINAL RALPH CODE.
*       1.D   UPDATES TO CREATE WINDOW FUNCTION IN A VECTOR
*             RATHER THAN TO ACTUALLY WINDOW A VECTOR.
```

PROGRAM: WWINOO FORTRAN

```
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        0.0            ?                       ?
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*.
      SUBROUTINE WWINOO(ARRAY,LENGTH,WINTYP,POWER,THETA)
      INTEGER COLUMN,LENGTH,WINTYP
      REAL POWER,ARRAY(*)
      POWER = 0.
      OO 10 I = 1,LENGTH
        IF (WINTYP.EO.1) CALL TRIGEN(I,LENGTH,VAL)
        IF (WINTYP.EO.2) CALL HAMGEN(I,LENGTH,VAL)
        IF (WINTYP.EO.3) CALL VANGEN(I,LENGTH,VAL)
        IF (WINTYP.EQ.4) CALL KAIGEN(I,LENGTH,VAL,THETA)
        IF (WINTYP.EO.5) CALL BLKGEN(I,LENGTH,VAL)
        IF (WINTYP.EO.6) CALL PARGEN(I,LENGTH,VAL)
C       ARRAY(I) = ARRAY(I)*VAL
        ARRAY(I) = VAL
        POWER = POWER + VAL**2
10    CONTINUE
      RETURN
      ENO
       SUBROUTINE WINGEN
       PARAMETER (PI = 3.141593)
C
       ENTRY TRIGEN(I,LENGTH,VAL)
              IF ((I-1) .LE. (LENGTH/2)) THEN
                VAL = (I-1) /(LENGTH/2.)
              ELSE
                VAL = (LENGTH-(I-1))/(LENGTH/2.)
              ENOIF
       RETURN
C
       ENTRY HAMGEN(I,LENGTH,VAL)
              VAL = 0.54 - 0.46*COS(2*PI*(I-1)/LENGTH)
       RETURN
C
       ENTRY VANGEN(I,LENGTH,VAL)
              VAL = 0.5 - 0.5*COS(2*PI*(I-1)/LENGTH)
       RETURN
C
       ENTRY KAIGEN(I,LENGTH,VAL,THETA)
       AUGUM1 = (I - (LENGTH/2. + 1)) / (LENGTH/2.)
       AUGUM2 = 1.0 - AUGUM1**2
       VAL = BESSIO(THETA*SQRT(AUGUM2))/BESSIO(THETA)
       RETURN
C
```

```
      ENTRY BLKGEN(I,LENGTH,VAL)
            TERM1 = 0.35875
            TERM2 = 0.48829 * COS(2*PI*1*(I-1)/LENGTH)
            TERM3 = 0.14128 * COS(2*PI*2*(I-1)/LENGTH)
            TERM4 = 0.01168 * COS(2*PI*3*(I-1)/LENGTH)
C
            VAL = TERM1 - TERM2 + TERM3 - TERM4
      RETURN
C
      ENTRY PARGEN(I,LENGTH,VAL)
            ALPHA  = ABS((I-1)-(LENGTH/2.)) / (LENGTH/2.)
            IF ((I-1).GE.(LENGTH/4) .AND. (I-1).LE.(3*LENGTH/4)) THEN
              VAL = 1.0 - 6*ALPHA**2*(1.0-ALPHA)
            ELSE
              VAL = 2 * (1.0 - ALPHA)**3
            ENDIF
      RETURN
      END
      FUNCTION BESSID(X)
      INTEGER FACTRL
      BESSIO = 1.
      DO 10 I = 1,1D
        BESSIO = BESSID+(1./FACTRL(I)*(X/2)**I)**2
10    CONTINUE
      RETURN
      END
      FUNCTION FACTRL(K)
      INTEGER FACTRL
      FACTRL = 1
      DO 10 I = 1,K
        FACTRL = FACTRL*I
10    CONTINUE
      RETURN
      END
```

PROGRAM: SPOISKIO FORTRAN

```
*.
*****************************************************************
*                                                               *
*       Vibration Analysis System (VAS)                         *
*                                                               *
*       (c) Copyright 1986 by Brick A. Verser                   *
*                                                               *
*****************************************************************
*
* MOOULE NAME:
*
*       SPDISKIO
*
* FUNCTION:
*
*       TO EFFICIENTLY HANDLE OISK INPUT/DUTPUT FUNCTIONS
*       FOR THE VIBRATION ANALYSIS SYSTEM.
*
*       ROUTINES ARE PROVIOEO TO REAO HALF-WORO INTEGERS,
*       FULL-WORO INTEGERS, ANO FOUR-BYTE REAL FORMAT BINARY
*       OATA.  A ROUTINE TO WRITE FOUR-BYTE REAL FORMAT
*       BINARY OATA IS ALSO PROVIDEO.
*
*       THE SUBRDUTINES CONTAINEO IN THIS FILE ARE:
*
*            OISKIR - READ REAL*4 FORMAT OATA.
*            OISKII - REAO INTEGER*4 FORMAT OATA.
*            OISKIH - REAO INTEGER*2 FDRMAT OATA.
*            OISKOR - WRITE REAL*4 FORMAT OATA.
*
* OPERATION:
*
*       A NON-STANOARO OPTION OF THE READ ANO WRITE
*       STATEMENTS IS USEO TO EFFICIENTLY ALLOW
*       PACKING MULTIPLE VALUES IN A SINGLE LINE.
*       USING THE NUM= OPERANO OF REAO ANO WRITE
*       A VARIABLE LENGTH CMS FILE IS READ OR WRITTEN
*       WHERE THE LAST LINE OF THE FILE NEEO NOT BE
*       FULL AS IS REOUIREO BY STANOARO FORTRAN 77.
*
* REVISION HISTORY:
*
*       1.0   ORIGINAL COOE
*
*       REVISION        OATE                    PROGRAMMER
*       --------        ----                    ----------
*        1.0            JUNE 26, 1986           BRICK VERSER
*
*.
```

PROGRAM: SPOISKIO FORTRAN

```
*.
*****************************************************************
*
* SUBROUTINE NAME:
*
*       DISKIR
*
* FUNCTION:
*
*       TO REAO BINARY REAL*4 FORMAT OATA FROM OISK.
*
* CALLING FORMAT:
*
*       CALL OISKIR(IUNIT,VALUE,IERR,IXWLEN,XWORK,IWORK,IWORK2)
*
* INPUT PARAMETERS:
*
*       IUNIT  - AN INTEGER UNIT NUMBER FROM WHICH THE OATA
*                IS TO BE REAO.
*       IXWLEN - AN INTEGER SPECIFYING THE LENGTH OF THE
*                WORK VECTOR SPECIFIEO BY XWORK.
*       XWORK  - A REAL*4 VECTOR OF LENGTH IXWLEN USEO TO
*                STORE INPUT OATA BETWEEN CALLS TO OISKIR.
*                THE OATA IN THIS VECTOR MUST NOT BE ALTEREO
*                BETWEEN SUCCESSIVE CALLS TO THE ROUTINE.
*       IWORK  - AN INTEGER PARAMETER ANO WORK VARIABLE WHICH
*                SHOULO BE SET TO ZERO ON THE INITIAL CALL.
*                THIS VALUE MUST NOT BE ALTEREO BETWEEN
*                CALLS TO THIS ROUTINE.
*       IWORK2 - AN INTEGER WORK VARIABLE WHICH SHOULO NOT
*                BE ALTEREO BETWEEN CALLS TO THIS ROUTINE.
*
* OUTPUT PARAMETERS:
*
*       VALUE  - THE NEXT REAL*4 VALUE REAO FROM THE INPUT UNIT.
*       IERR   - AN INTEGER INOICATOR SET TO NEGATIVE ONE ON
*                ENO-OF-FILE.
*
*.
      SUBROUTINE OISKIR(IUNIT,VALUE,IERR,IXWLEN,XWORK,IWORK,IWORK2)
      OIMENSION XWORK(IXWLEN)
      IF (IWORK.EO.O .OR. IWORK.GT.IWORK2) THEN
         REAO(UNIT=IUNIT,ENO=200,NUM=IWORK2) XWORK
         IWORK2 = IWORK2/4
         IWORK = 1
      ENOIF
      IF (IWORK.GT.IWORK2) GOTO 200
      VALUE = XWORK(IWORK)
      IERR = 0
```

```
      IWORK = IWORK+1
      GOTO 999
2DD   CONTINUE
      VALUE = D
      IERR = -1
999   CONTINUE
      RETURN
      END
*.
*****************************************************************
*
* SUBROUTINE NAME:
*
*       DISKII
*
* FUNCTION:
*
*       TO READ BINARY INTEGER*4 FORMAT DATA FROM DISK.
*
* CALLING FORMAT:
*
*       CALL DISKII(IUNIT,IVALUE,IERR,IXWLEN,IXWORK,IWORK,IWORK2)
*
* INPUT PARAMETERS:
*
*       IUNIT  - AN INTEGER UNIT NUMBER FROM WHICH THE DATA
*                IS TO BE READ.
*       IXWLEN - AN INTEGER SPECIFYING THE LENGTH OF THE
*                WORK VECTOR SPECIFIED BY XWORK.
*       IXWORK - AN INTEGER*4 VECTOR OF LENGTH IXWLEN USED TO
*                STORE INPUT DATA BETWEEN CALLS TO DISKIR.
*                THE OATA IN THIS VECTOR MUST NOT BE ALTERED
*                BETWEEN SUCCESSIVE CALLS TO THE ROUTINE.
*       IWORK  - AN INTEGER PARAMETER ANO WORK VARIABLE WHICH
*                SHOULD BE SET TO ZERO ON THE INITIAL CALL.
*                THIS VALUE MUST NOT BE ALTERED BETWEEN
*                CALLS TO THIS ROUTINE.
*       IWORK2 - AN INTEGER WORK VARIABLE WHICH SHOULD NOT
*                BE ALTERED BETWEEN CALLS TO THIS ROUTINE.
*
* OUTPUT PARAMETERS:
*
*       IVALUE - THE NEXT INTEGER*4 VALUE READ FROM THE INPUT UNIT.
*       IERR   - AN INTEGER INDICATOR SET TO NEGATIVE ONE ON
*                END-OF-FILE.
*
*.
      SUBROUTINE DISKII(IUNIT,IVALUE,IERR,IXWLEN,IXWORK,IWORK,IWORK2)
      DIMENSION IXWORK(IXWLEN)
```

```
      IF (IWORK.EO.O .OR. IWORK.GT.IWORK2) THEN
         REAO(UNIT=IUNIT,ENO=200,NUM=IWORK2) IXWORK
         IWORK2=IWDRK2/4
         IWORK = 1
      ENOIF
      IF (IWORK.GT.IWORK2) GOTO 200
      IVALUE = IXWDRK(IWORK)
      IERR = O
      IWORK = IWORK+1
      GOTO 999
200   CONTINUE
      IVALUE = O
      IERR = -1
999   CONTINUE
      RETURN
      ENO
*.
*****************************************************************
*
* SUBRDUTINE NAME:
*
*       DISKIR
*
* FUNCTION:
*
*       TO REAO BINARY REAL*4 FORMAT DATA FROM DISK.
*
* CALLING FORMAT:
*
*       CALL OISKOR(IUNIT,VALUE,IPRM,IXWLEN,XWORK,IWORK)
*
* INPUT PARAMETERS:
*
*       IUNIT  - AN INTEGER UNIT NUMBER FROM WHICH THE OATA
*                IS TD BE READ.
*       VALUE  - THE REAL*4 VARIABLE TO BE WRITTEN.
*       IPRM   - AN INTEGER SPECIFYING WHICH DF THREE
*                POSSIBLE CALLS THIS IS.
*                INIT:  IPRM = O.  THIS CALL IS USEO TO
*                       INITIALIZE THE I/O SYSTEM.
*                       NO OATA IS WRITTEN BY THIS CALL.
*                WRITE: IPRM = 2.  THIS CALL IS USEO TO
*                       WRITE 'VALUE'.
*                FLUSH: IPRM = 3.  THIS CALL IS USED TO
*                       FLUSH THE REMAINING OUTPUT BUFFER
*                       TO OISK.  A FORTRAN CLOSE STATEMENT
*                       IS NOT ACTUALLY ISSUEO.
*       IXWLEN - AN INTEGER SPECIFYING THE LENGTH OF THE
*                WORK VECTOR SPECIFIED BY XWORK.
```

```
*       XWORK  - A REAL*4 VECTOR OF LENGTH IXWLEN USED TO
*                STORE INPUT DATA BETWEEN CALLS TO OISKIR.
*                THE DATA IN THIS VECTOR MUST NOT BE ALTERED
*                BETWEEN SUCCESSIVE CALLS TO THE ROUTINE.
*       IWORK  - AN INTEGER WORK VARIABLE WHICH SHOULD NOT
*                BE ALTERED BETWEEN CALLS TO THIS ROUTINE.
*
*.
      SUBROUTINE DISKOR(IUNIT,VALUE,IPRM,IXWLEN,XWORK,IWORK)
      DIMENSION XWORK(IXWLEN)
      IF (IPRM.EQ.3) THEN
         IF (IWORK.GT.1) WRITE(UNIT=IUNIT) (XWORK(I),I=1,IWORK-1)
         GOTO 999
      ENDIF
      IF (IPRM.EQ.1) THEN
         IWORK = 1
         GOTO 999
      ENDIF
      XWORK(IWORK) = VALUE
      IF (IWORK.GE.IXWLEN) THEN
         WRITE(UNIT=IUNIT) XWORK
         IWORK = 0
      ENOIF
      IWORK = IWORK + 1
999   CONTINUE
      RETURN
      END
*.
*****************************************************************
*
* SUBROUTINE NAME:
*
*       DISKIH
*
* FUNCTION:
*
*       TO READ BINARY INTEGER*2 FORMAT DATA FROM DISK.
*
* CALLING FORMAT:
*
*       CALL OISKIH(IUNIT,IVALUE,IERR,IXWLEN,IXWORK,IWORK,IWORK2)
*
* INPUT PARAMETERS:
*
*       IUNIT  - AN INTEGER UNIT NUMBER FROM WHICH THE DATA
*                IS TO BE READ.
*       IXWLEN - AN INTEGER SPECIFYING THE LENGTH OF THE
*                WORK VECTOR SPECIFIEO BY XWORK.
*       IXWORK - AN INTEGER*2 VECTOR OF LENGTH IXWLEN USED TO
```

PROGRAM: SPOISKIO FORTRAN

```
*                STORE INPUT OATA BETWEEN CALLS TO OISKIR.
*                THE OATA IN THIS VECTOR MUST NOT BE ALTEREO
*                BETWEEN SUCCESSIVE CALLS TO THE ROUTINE.
*       IWORK  - AN INTEGER PARAMETER ANO WORK VARIABLE WHICH
*                SHOULO BE SET TO ZERO ON THE INITIAL CALL.
*                THIS VALUE MUST NOT BE ALTEREO BETWEEN
*                CALLS TO THIS ROUTINE.
*       IWORK2 - AN INTEGER WORK VARIABLE WHICH SHOULO NOT
*                BE ALTERED BETWEEN CALLS TO THIS ROUTINE.
*
* OUTPUT PARAMETERS:
*
*       IVALUE - THE NEXT INTEGER*2 VALUE REAO FROM THE INPUT UNIT.
*       IERR   - AN INTEGER INOICATOR SET TO NEGATIVE ONE ON
*                ENO-OF-FILE.
*
*.
      SUBROUTINE OISKIH(IUNIT,IVAL,IERR,ISLEN,ISAVE,IWORK,IWORK2)
      INTEGER*2 ISAVE(ISLEN),IVAL
      IF (IWORK.EQ.O .OR. IWORK.GT.IWORK2) THEN
         REAO(UNIT=IUNIT,ENO=200,NUM=IWORK2) ISAVE
         IWORK2 = IWORK2/2
         IWORK = 1
      ENOIF
      IF (IWORK.GT.IWORK2) GOTO 200
      IVAL = ISAVE(IWORK)
      IERR = O
      IWORK = IWORK+1
      GOTO 999
200   CONTINUE
      VALUE = O
      IERR = -1
999   CONTINUE
      RETURN
      ENO
```

A Vibration Analysis System Using Spectral Estimation Techniques

by

Brick Andrew Verser

B.S., Kansas State University, 1982

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

Kansas State University
Manhattan, Kansas

1986

The use of spectral analysis can provide useful insights into the behaviour of a structure under dynamic loading. Unfortunately, the commercial equipment often used in these studies is somewhat expensive and cumbersome. This paper describes a microcomputer-based data acquisition system and a mainframe-based data analysis system, together referred to as the Vibration Analysis System (VAS), which were created to provide many of the signal collection and analysis facilities needed in the analysis of structures. The VAS data acquisition system can sample up to eight channels of data at up to 7500 samples per second, and can continuously collect millions of samples. The VAS data analysis system calculates auto and cross-spectrum estimates, auto and cross-correlation estimates, coherence function estimates, and transfer functions estimates, allowing the results to be plotted or printed.